ViNT: A Foundation Model for Visual Navigation

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, Sergey Levine

CoRL 2023 · UC Berkeley

MOAI Seminar

Presenter 유소은

Introduction

Foundation Models: Pre-train Once, Adapt Everywhere

"Train once on Internet-scale data — then reuse the same model for countless new tasks."

① Zero-shot transfer

Use the pre-trained model **as-is** on a new task — no extra training.

② Prompt-tuning

Keep the model; learn a small **"soft prompt"** input that steers it to the task.

③ Fine-tuning

Update the model with a **small amount** of target-task data.

e.g., adapting a vision model to a new domain

- Language and vision models already work this way — robot navigation does not yet.
- **what is required of a foundation model for mobile robotics?**

Why Is Robot Navigation Different?

“Every robot sees, moves, and operates differently — so robot data is fragmented.”

① Sensors

Different cameras: focal length, field of view, mounting position.

The same scene looks different to each robot.

② Dynamics

Different sizes and speeds: a TurtleBot at **0.5 m/s** vs. an ATV at **10 m/s**.

The same command means different motion.

③ Environments & tasks

Offices, sidewalks, off-road trails; goals given as images, GPS points, or “turn left”.

Each application looks like a new problem.

Result: data fragmentation → specialist models

Each dataset is small and tied to one robot, so prior work trains a **separate model** for each robot, environment, or task.

What Should a Robot Foundation Model Do?

(i) Zero-shot deployment

Works out of the box in novel settings: different **sensors**, robot **embodiments**, and **environments**.

(ii) Downstream adaptation

Can be adapted to a task of choice: different **objectives**, **goal specification types**, and **behaviors**.

- **visual navigation** — the robot moves using only its own (egocentric) camera images
- a base level of generalization out of the box, improved further by fine-tuning on **a small amount of data**

Overall Architecture

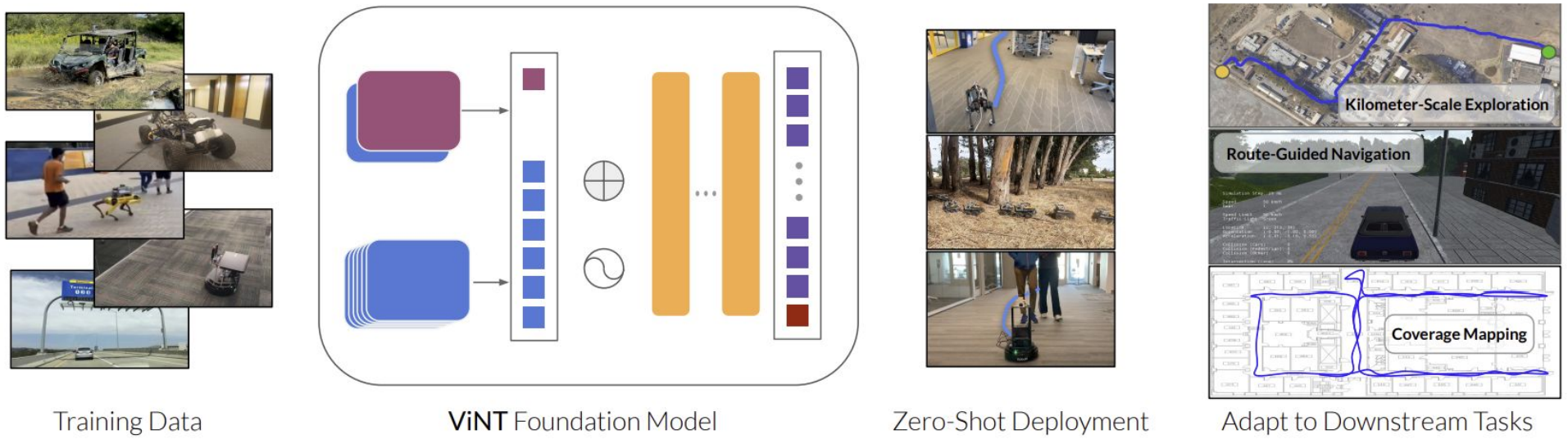


Fig. 1 — Overview of the ViNT foundation model

Training Data
100+ h from 8 robots, all existing datasets

ViNT Model
31M-param Transformer, image-goal objective

Zero-Shot Deployment
New robots and places, no fine-tuning

Downstream Tasks
Exploration, routing, mapping with little data

Strengths of ViNT

① Zero-shot generalization

Controls robots and places never seen in training with no extra training.

Successful exploration and navigation out of the box

② Efficient adaptation

Masters a new environment or a new goal type with **≈ 1 hour** of data.

Suggests high-quality navigational priors

③ Emergent good habits

Without explicit supervision: follows roads, keeps to hallway centers, **avoids moving pedestrians**.

Learned implicitly from messy, diverse data

One pre-trained model → many robots, many places, many tasks

Key Contributions

- **Generality** → embodiment-agnostic action space + image-goal objective: **one model drives many robots**

  Any dataset with video + actions can be used for training

- **Adaptability** → diffusion subgoals to **explore unseen environments**; prompt-tuning for **new goal types** (GPS, routing commands)

  Only the goal encoder is swapped; the rest of ViNT is reused

ViNT in Six Keywords

① Image-Goal Training

Learn from any video + action data

② Goal Fusion

Encode the goal relative to the current view

③ Transformer Backbone

7 tokens → actions + distance, 31M params

④ Embodiment-Agnostic Actions

Normalized waypoints work on any robot

⑤ Diffusion Subgoals + Graph

Explore and plan at kilometer scale

⑥ Prompt-Tuning Adaptation

Swap the goal encoder for GPS or commands

Method: ViNT

Training Task: Image-Goal Navigation

Input

Current + past images $o_{t-P:t}$ and a **goal image** o_s — what the robot would see at the goal



Output

① temporal distance d — steps needed to reach the goal

② H future actions â leading toward the goal

→ Any navigation dataset from any robot can be used: **maximal data diversity**, the key ingredient of a foundation model

Model Architecture Overview

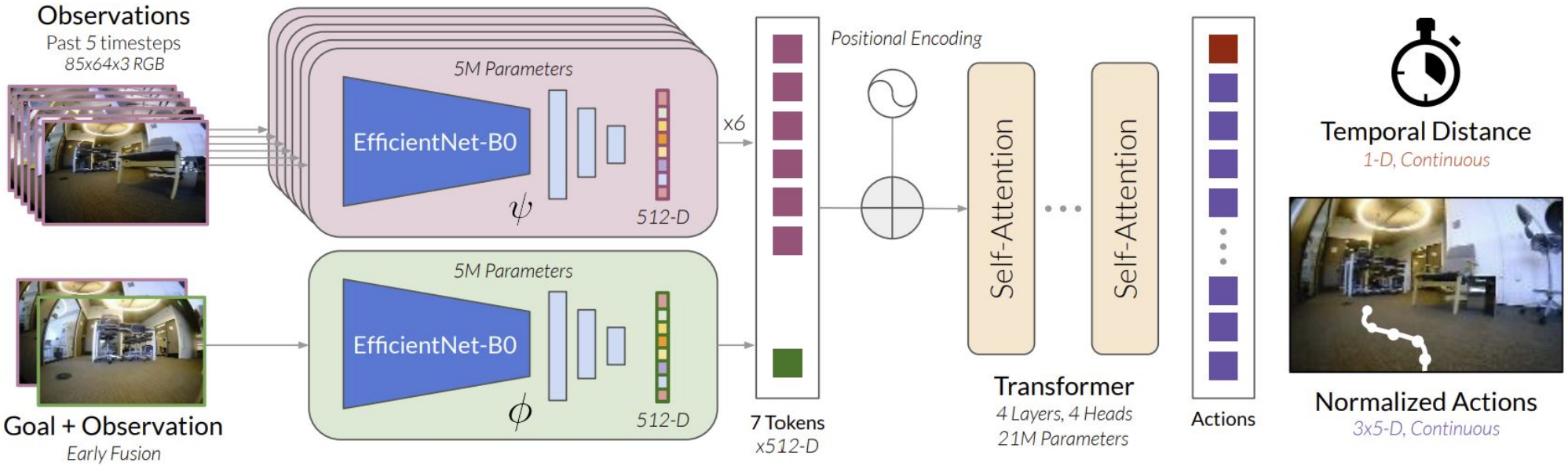


Fig. 2 — ViNT model architecture

Input: current + 5 past RGB images (85 × 64) & goal image → **Output**: temporal distance d & H = 5 future actions

6 + 1
observation + goal tokens

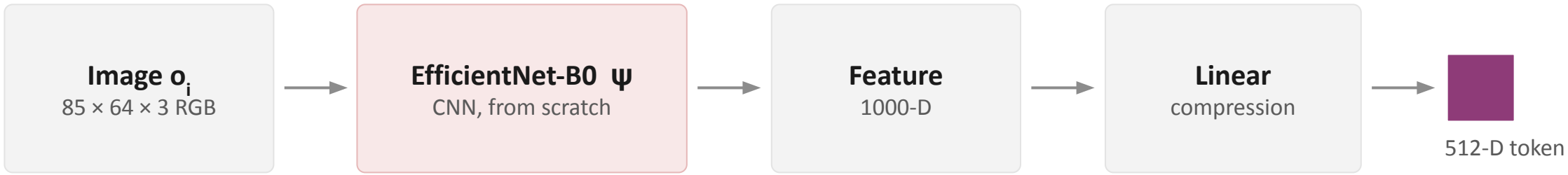
512-D
token size d_{model}

4 × 4
Transformer layers × heads

31M
parameters in total

① Observation Encoder ψ: Tokenizing the Past

× 6 frames (current + 5 past), each encoded independently



- **EfficientNet-B0**: a lightweight CNN — all 18 layers, **trained from scratch**; the final conv-layer feature is flattened
- Each frame becomes its own token ⟶ 6 observation tokens carry the **recent history** (motion, what was just passed)
- Small images + small CNN (≈ 5M params) ⟶ fast inference on **resource-constrained robots**

② Goal Fusion: Goal Relative to Current Position

Problem

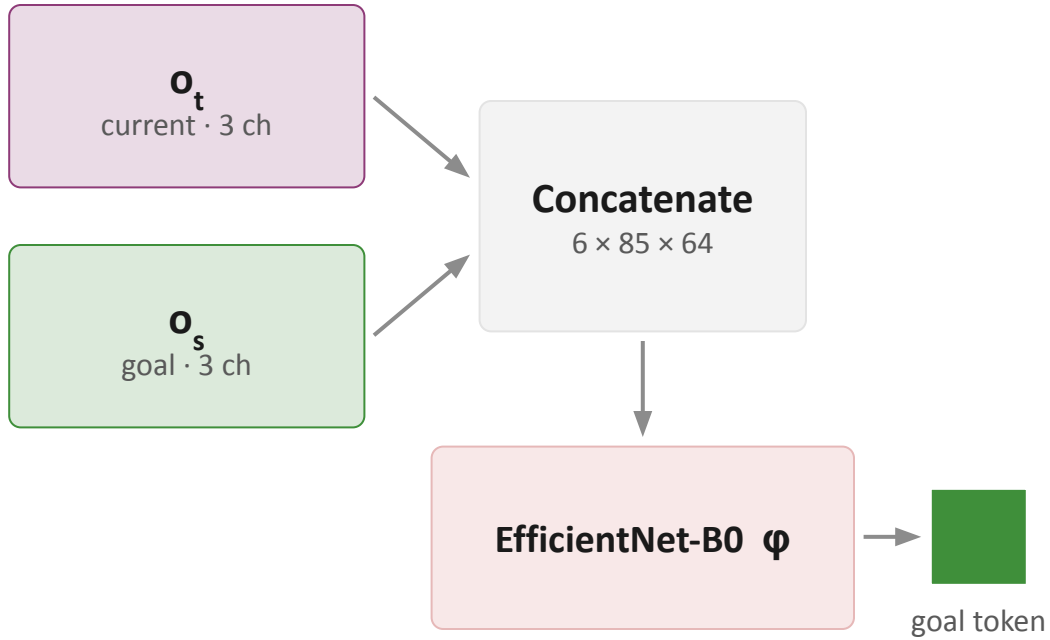
Encoding the goal image alone, $\varphi(o_s)$, made the model often ignore the goal entirely

Hypothesis

Useful features are **relative** — the difference between the current view and the goal, not the goal itself

Solution

Stack o_t and o_s along channels ($3 + 3 = 6 \times 85 \times 64$) ⟶ a second EfficientNet-B0 $\varphi(o_t, o_s)$ ⟶ **goal token**



Not “what does the goal look like?” but “**how does the goal differ from what I see now?**”

③ Transformer Backbone



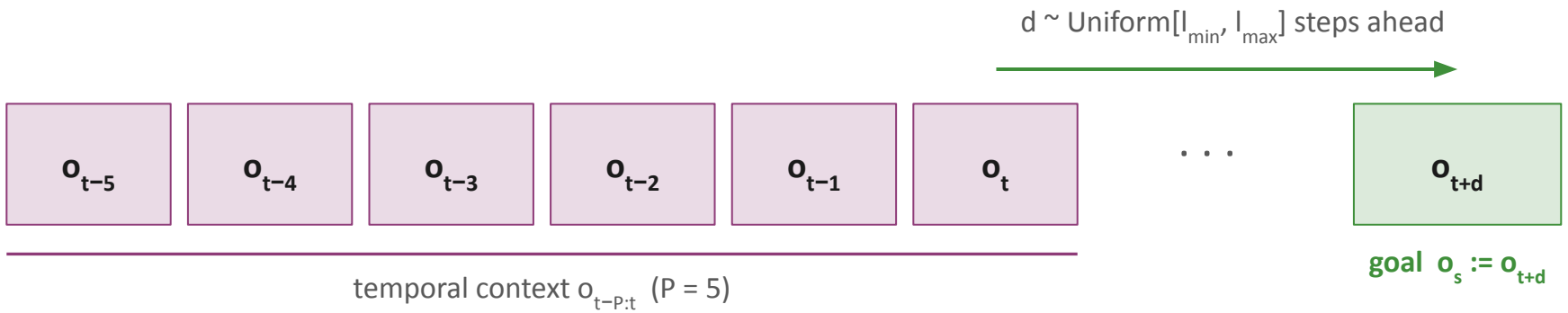
5M × 2
EfficientNet encoders ψ and φ

21M
Transformer

31M
total — light enough for onboard use

- Self-attention lets the goal token and the history tokens **exchange information** before predicting
- Two heads share one representation: **how far** the goal is (d) and **how to get there** (â)

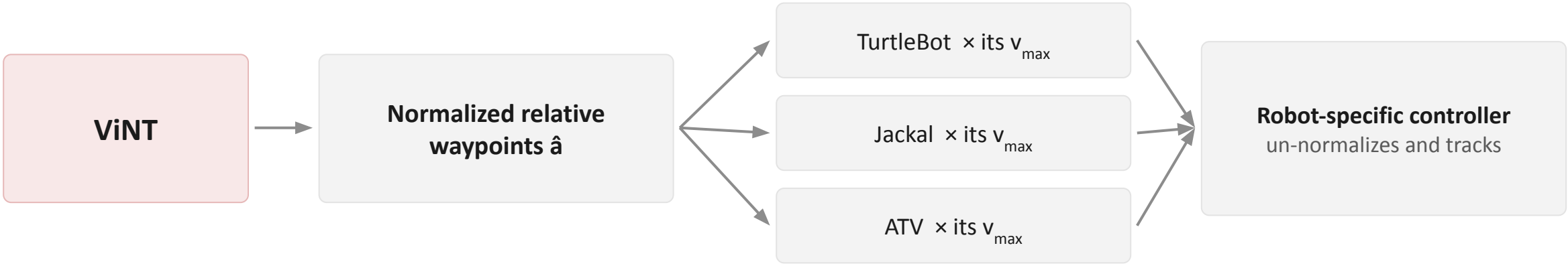
Training Objective: Self-Supervised from Raw Trajectories



$$\mathcal{L}_{\text{ViNT}}(\phi, \psi, f) = \mathbb{E}_\tau \mathbb{E}_t \mathbb{E}_d \ [\log p(\hat{a} | f(\psi(o)_{t:t-P}, \phi(o_t, o_s)) + \lambda \log p(d | f(\psi(o)_{t:t-P}, \phi(o_t, o_s))]$$

- The future frame becomes the goal → labels come from the trajectory itself: **no human annotation**
- One model learns both **how to get there** (â) and **how far it is** (d) — the distance later builds the planner's graph

Embodiment-Agnostic Action Space & Deployment



- Output = **relative waypoints**, not motor commands ⟶ low-level control is abstracted away
- Waypoints are **normalized by each robot's top speed** ⟶ diverse robots share one action space
- Deployment: any robot with an onboard camera + a velocity-tracking controller; ViNT runs at **4 Hz**, a PD controller tracks the waypoints (receding horizon)

Training Data

100+ h
real-world trajectories

8
robot platforms

0.2–10 m/s
range of robot speeds

0 h
collected for ViNT — all existing datasets

Environment	Datasets (platform)
Indoor — offices, hallways	GoStanford (TurtleBot2) · SACSoN (TurtleBot2) · CoryHall (RC car)
Sidewalks & suburban	SCAND (Spot, Jackal) · Berkeley (Jackal)
Off-road trails	RECON (Jackal) · Seattle (Warthog) · TartanDrive (ATV) · NeBula (ATV)
On-road	BDD (passenger cars)

Long-Horizon Navigation

- ViNT alone reaches a **nearby** image goal — real tasks need **far-away** goals (km), goals that are **not images**, and **unseen** places
- Fix: wrap ViNT in a planner with an episodic memory (**topological graph**) and exploratory **subgoal proposals**

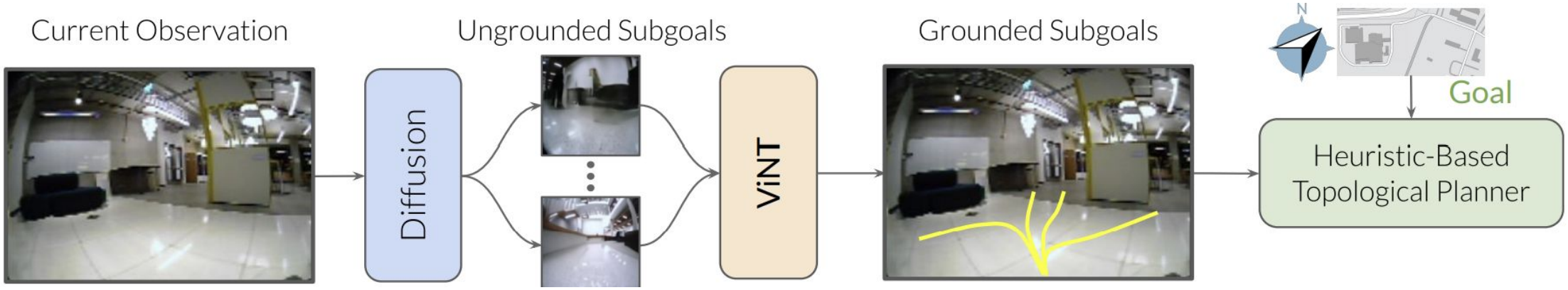


Fig. 3 — Long-horizon navigation in unseen environments with ViNT

① Propose
Diffusion samples subgoal images

② Ground
ViNT predicts distance + actions

③ Score & pick
Heuristic + A*- like planner

④ Drive & remember
ViNT drives; node added to graph

Subgoal Proposals with Image Diffusion

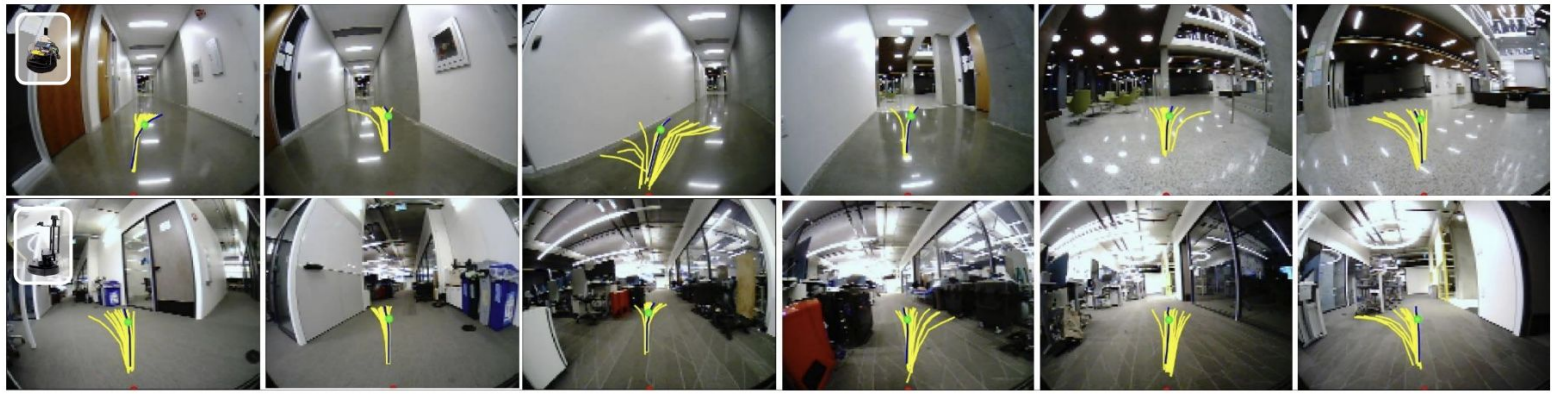


Figure 6: Visualizing ViNT exploration rollouts in challenging indoor environments using the Vizbot (top) and LoCoBot (bottom) robotic platforms. Future action samples $\hat{a}$ obtained by *spatially grounding* the subgoal candidates are shown in yellow, with the best actions corresponding to the best candidate marked in blue.

- Train $g(s_i \mid o_t)$: image-to-image diffusion on ViNT data; targets are frames **5–20 steps** ahead
- Sample K candidates → **diverse** possible futures (left, right, through the door …)
- Samples are **not spatially grounded** → ViNT predicts distance $d(s_i, o_t)$ and actions $a(s_i, o_t)$ for each → grounded subgoals

Planning with a Topological Graph

- Graph M = the robot's **episodic memory**: nodes = subgoal observations; edges = paths driven, or pairs ViNT predicts reachable
- Built **on the fly** while exploring; reused later to find shortcuts to new goals
- Keep an open set Ω of unvisited candidates; visit the one with the **lowest cost f(s)**, then sample new candidates there

$$f(s) = d_{\mathcal{M}}(o_t, s^-) + d_{\text{pred}}(s^-, s) + h(s, G, C)$$

$d_M(o_t, s^-)$
distance along the graph from the robot to the parent node s^-

$d_{pred}(s^-, s)$
ViNT-predicted distance from the parent to candidate s

h(s, G, C)
heuristic: how promising s is for reaching goal G, given context C

Three Heuristics, Three Behaviors

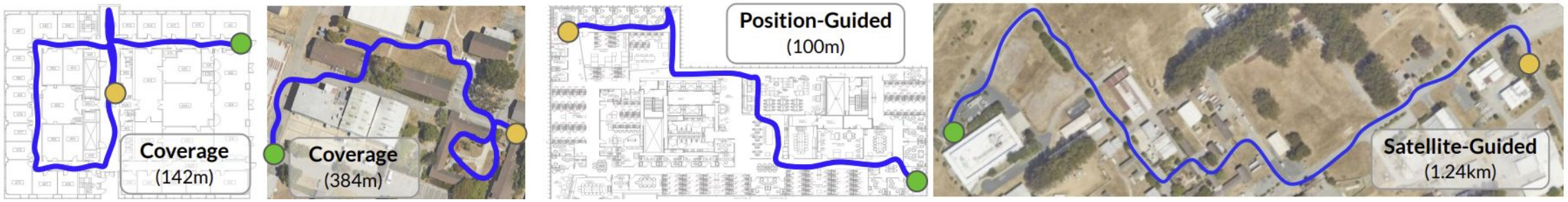


Fig. 5 — Example trajectories from start (orange) to goal (green)

Coverage exploration

h(s) = 0

No guidance → explore to cover the area; goal location unknown

Position-guided

h(s) = ‖s − G‖

Euclidean distance to a 2D goal (odometry indoors, GPS outdoors)

Satellite-guided

learned h

A CNN on an overhead satellite image scores whether s lies on a path to G

Adapting ViNT to Downstream Tasks

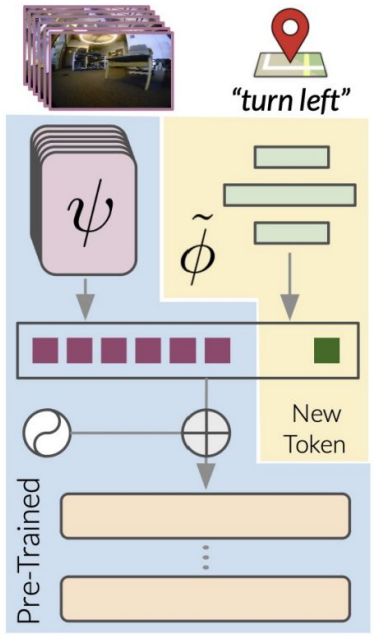


Fig. 4 — A new tunable goal token

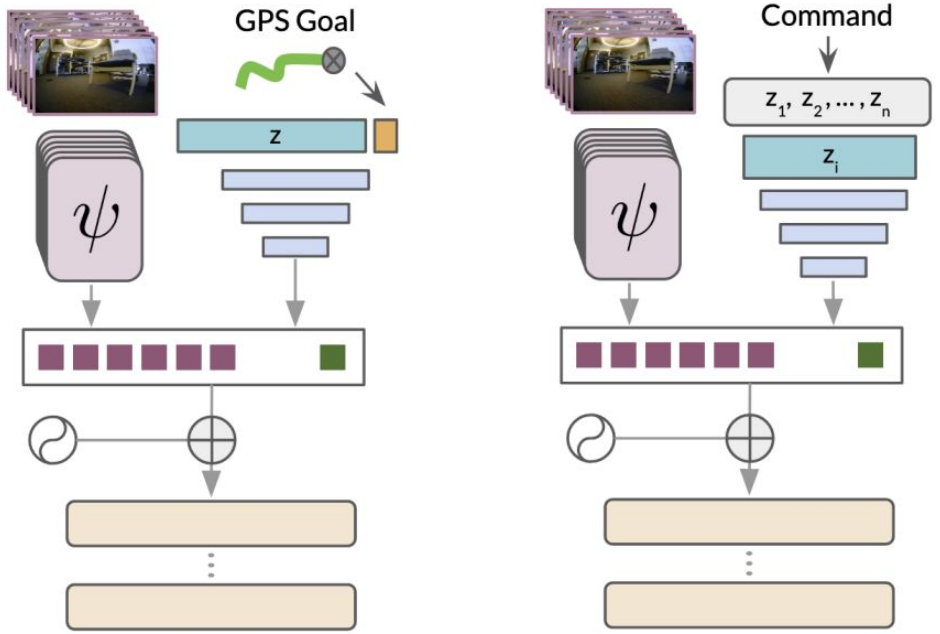


Fig. 14 — GPS (left) and command (right) adaptation

① Full model fine-tuning

Masters a new environment or embodiment with **≈ 1 hour** of data — no training from scratch

② Adapting to a new modalities

Replace φ(o_t, o_s) with a small new encoder $\tilde{\varphi}(\sigma)$ that maps GPS or commands into the same goal-token space

$$\mathcal{L}_{\text{adapt}} = \mathbb{E}_{\tau \in \mathcal{D}_F} \mathbb{E}_t \mathbb{E}_d \left[\log p(\hat{a} | f(\psi(o)_{t:t-P}, \tilde{\phi}(\sigma)) + \lambda \log p(d | f(\psi(o)_{t:t-P}, \tilde{\phi}(\sigma))) \right]$$

Real-World Evaluation

Experimental Setup: Four Questions

Q1 · Exploration

Can ViNT efficiently explore unseen environments and use heuristics?

→ Sec. 6.1

Q2 · Zero-shot

Does ViNT generalize to novel robots, environments, and obstacles?

→ Sec. 6.2

Q3 · Fine-tuning

Can ViNT be fine-tuned to improve in out-of-distribution environments?

→ Sec. 6.3

Q4 · Adaptation

Can ViNT be adapted to new task specifications and modalities?

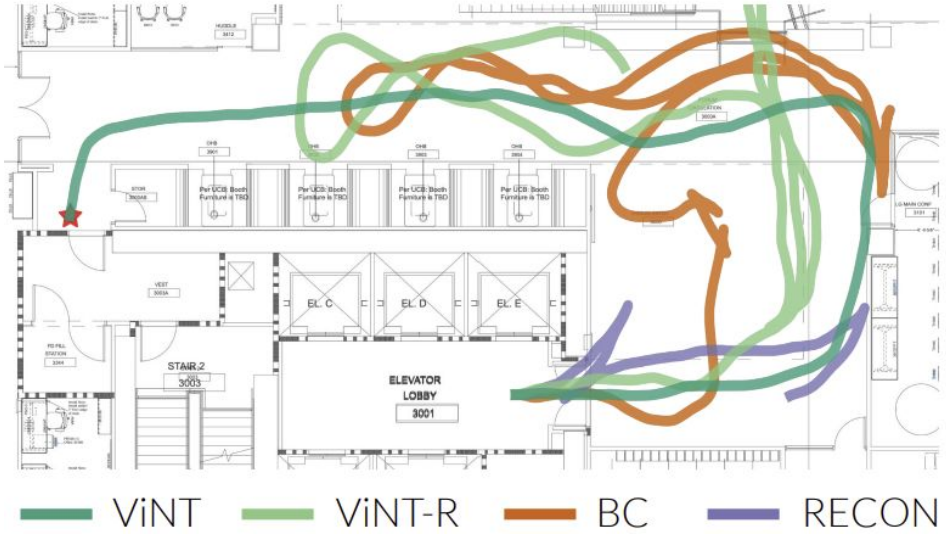
→ Sec. 6.4

- 5 platforms incl. a **drone**, a **quadruped** (Unitree Go 1), Vizbot, LoCoBot, and Jackal — Go 1 and LoCoBot never appear in training
- No careful system identification or sensor homogenization — every robot keeps its own controller and onboard stack

Q1. Coverage Exploration in Unseen Environments

Method	Indoor	Outdoor
End-to-End BC	0.72	0.44
End-to-End GCG	—	0.61
RECON	0.19	0.23
ViNT-R (random subgoals)	0.81	—
ViNT	0.94	1.00

Table 1 (left, p.7) — success rate



ViNT · ViNT-R · BC · RECON trajectories (indoor)

- Task: find a goal image whose location is **unknown**, with no other cues, in unseen indoor & outdoor environments
- End-to-end policies avoid collisions but **get stuck in small areas**; graph-based search reasons explicitly about coverage
- ViNT-R still does well with random subgoals → the policy itself is robust; diffusion subgoals add efficiency

Q1. Guided Exploration at Kilometer Scale

Setting (context)	Success	SPL	Distance
Indoor · 2D position	0.60 → **0.90**	—	56 m → **91 m**
Outdoor · GPS	0.64 → **0.95**	0.42 → **0.84**	720 m → **1270 m**
Outdoor · Satellite image	0.77 → **1.00**	0.68 → **0.94**	780 m → **1040 m**

Table 2 (p.7) — ViKiNG (previous state of the art) → ViNT

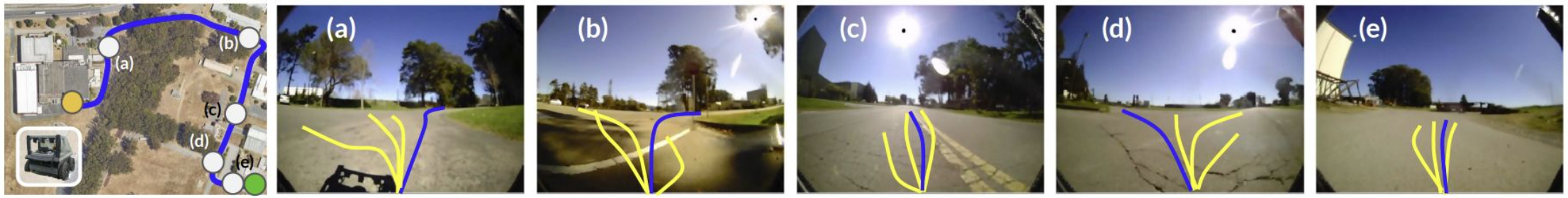


Fig. 7 — 765 m satellite-guided rollout: candidates (yellow), chosen (blue)

SPL = Success weighted by Path Length (1.0 = as efficient as the shortest path) · Distance = traveled before a collision / intervention

Q2. Zero-Shot: One Policy to Drive Any Robot

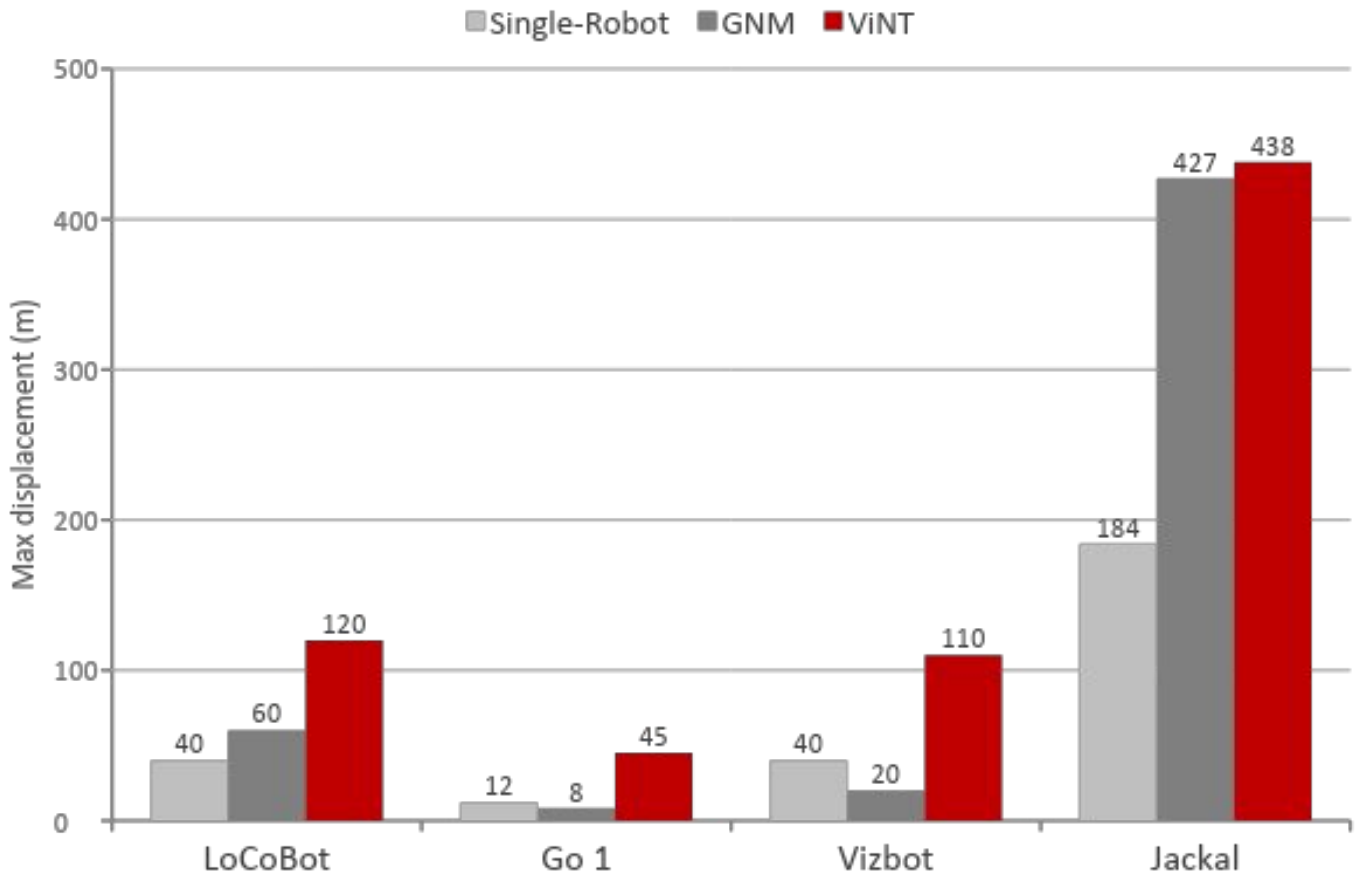


Table 3 (Sec. 6.2, p.8) — max displacement without intervention

- Same pre-trained ViNT, **no fine-tuning**, on 4 robots (undirected exploration)
- Cross-robot training **beats single-robot specialists** on every robot
- Larger capacity beats GNM, especially on the unseen **Go 1** quadruped (45 m vs. 8 m)
- **Positive transfer** on Vizbot: 110 m vs. 40 m for the specialist — a key foundation-model property

Q3. Fine-Tuning in a New Domain: CARLA



Fig. 8 — CARLA test environment & routing view

1. Why CARLA? ① simulated images look **different** from the real training data ② on-road driving needs lane keeping and smooth turns — **absent from training data**

- Fine-tune the whole model end-to-end: lr 1×10^{-4}, 5 epochs; at most 5 h of on-task data
- Data from a rule-based autopilot: Town 01 for training, held-out Town 02 for testing
- Metrics
  - **Success** = progress toward the goal before a collision
  - **In lane** = fraction of trajectories staying in the lane

Q3. Data Efficiency: Under 1 Hour Is Enough

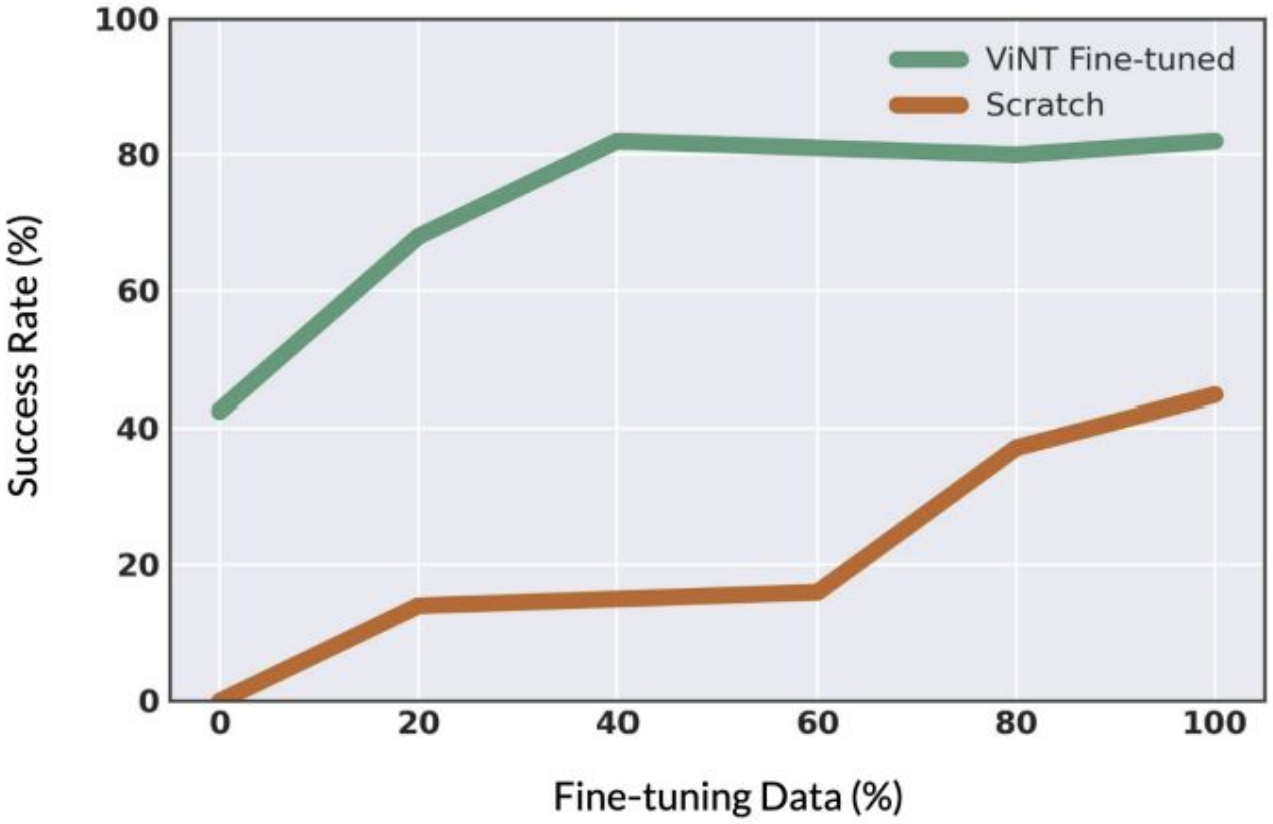


Table 3 (right) — ViNT fine-tuned (green) vs. scratch (orange)

40%

success with **no fine-tuning** — navigational affordances transfer zero-shot

80%

success with **less than 1 hour** of fine-tuning data

5×

ViNT fine-tuning beats a single-domain model trained with 5× more data

Q3 & Q4 Results: Pre-Training Matters

Q4 setup: same CARLA task, but the goal is a **GPS position** or a **routing command** (left / right / straight, like map directions) instead of an image — same on-task data for every backbone

Method	Images		Positions	Routing
	Success	In lane	Success	Success
Scratch	0.45	0.74	0.79	0.43
ImageNet	0.22	0.71	0.59	0.45
SimCLR	0.21	0.63	0.70	0.64
VC-1	0.19	0.65	0.49	0.38
GNM	0.49	0.66	0.45	0.49
ViNT	0.82	0.82	0.89	0.72

Table 3 (left, Sec. 6.3–6.4, p.9)

- General visual features (ImageNet, SimCLR, VC-1) are **not enough** — even below Scratch on image goals
- GNM has a navigation prior but low capacity → **struggles to adapt** to new goal types
- ViNT is best on all three goal types → a **foundation for downstream tasks**

Emergent Behaviors ①: Safe Defaults & Preferences



Fig. 9 — Invalid diffusion samples



Fig. 10 — Paved roads (left) and hallways (right)

Implicit navigation affordances

- Given **random, unreachable** subgoals (ViNT-R), ViNT still explores collision-free and reaches the goal **≈ 80%** of the time
- → a safe "default" behavior; bad diffusion samples are simply ignored

Implicit navigation preferences

- Follows **paved roads** outdoors and drives in the **middle of hallways** indoors
- → learned even though much of the training data is weavy and suboptimal

Emergent Behaviors ②: Navigating Around Pedestrians



Fig. 11 — Robustness to dynamic pedestrians

- Trained only on **offline data** with a simple self-supervised objective — no pedestrian labels, no reinforcement learning
- Collision avoidance generalizes to **dynamic obstacles**: avoids two walking pedestrians, recovers to the original path, and reaches the goal
- Diverse data + a large model → useful behaviors that were **never explicitly trained**

Conclusion

Conclusion

1. A simple, general objective (**reach an image goal**) + **embodiment-agnostic actions** let one model learn from any robot's video — the recipe for a navigation **foundation model**.

2. Goal fusion keeps relative goal features while leaving the goal encoder **swappable** → GPS and routing tasks with about an hour of data.

3. A short-horizon policy + **diffusion subgoals** + a **topological graph** reaches **kilometer-scale** navigation; diverse data even yields good default habits.

Thank you

Q & A

Appendix: Key Hyperparameters

ViNT model	
Parameters	31M
RGB resolution	85 × 64
Encoder	EfficientNet-B0
Token dimension	512
Layers / heads	4 / 4
Attn. hidden dim	2048
Context P	5
Horizon H	5
MLP layers	256-128-64-32

ViNT training	
Epochs	30
Batch size	300
Learning rate	5×10^{-4}
Optimizer	AdamW
Schedule	4 warmup, cosine
Compute	8 × V100, 30 h
Fine-tuning LR	1×10^{-4}
Distance λ	0.01
Max distance	20

Diffusion model	
Parameters	318M
Resolution	128 × 128
Up/down blocks	4
Train steps	250,000
Compute	v4-8 TPU, 30 h
Sampler	DDIM, 200 steps
Guidance weight	1.0
CFG mask prop.	0.2
Batch size	128

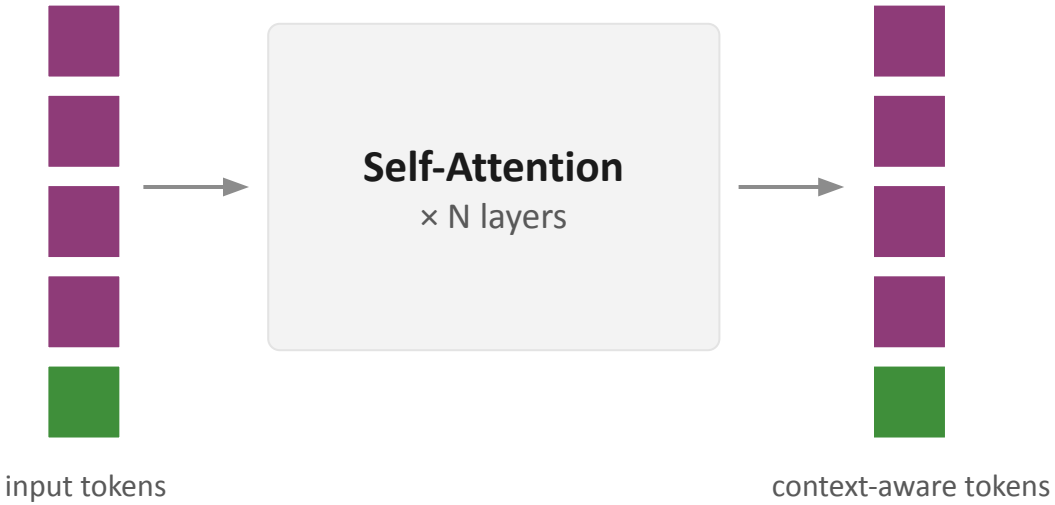
Table 6 (p.19) — batch size and training time vary with hardware

Appendix: Training Dataset

Dataset	Platform	Speed	Total	Used	Environment
GoStanford	TurtleBot2	0.5 m/s	17 h	14 h	office
RECON	Jackal	1 m/s	25 h	25 h	off-road
CoryHall	RC Car	1.2 m/s	2 h	2 h	hallways
Berkeley	Jackal	2 m/s	4 h	4 h	suburban
SCAND-S	Spot	1.5 m/s	8 h	4 h	sidewalks
SCAND-J	Jackal	2 m/s	1 h	1 h	sidewalks
Seattle	Warthog	5 m/s	1 h	1 h	off-road
TartanDrive	ATV	10 m/s	7 h	5 h	off-road
NeBula	ATV	10 m/s	10 h	10 h	off-road
SACSoN	TurtleBot2	0.5 m/s	75 h	10 h	office
BDD	Car(s)	20 m/s	10 h	4 h	on-road
Total			160 h	80 h	

Background: Three Building Blocks

Background ① Transformer & Self-Attention



- **Token** = a fixed-size vector for one piece of input (an image, a word, a GPS point …)
- **Self-attention**: each token scores every other token (query · key), turns the scores into weights with softmax, and takes a weighted sum of their values
- Stacking layers lets information flow across the whole sequence
- **Why it matters for ViNT**: images and other goal types all become 512-D tokens, so one input can be **swapped** for another

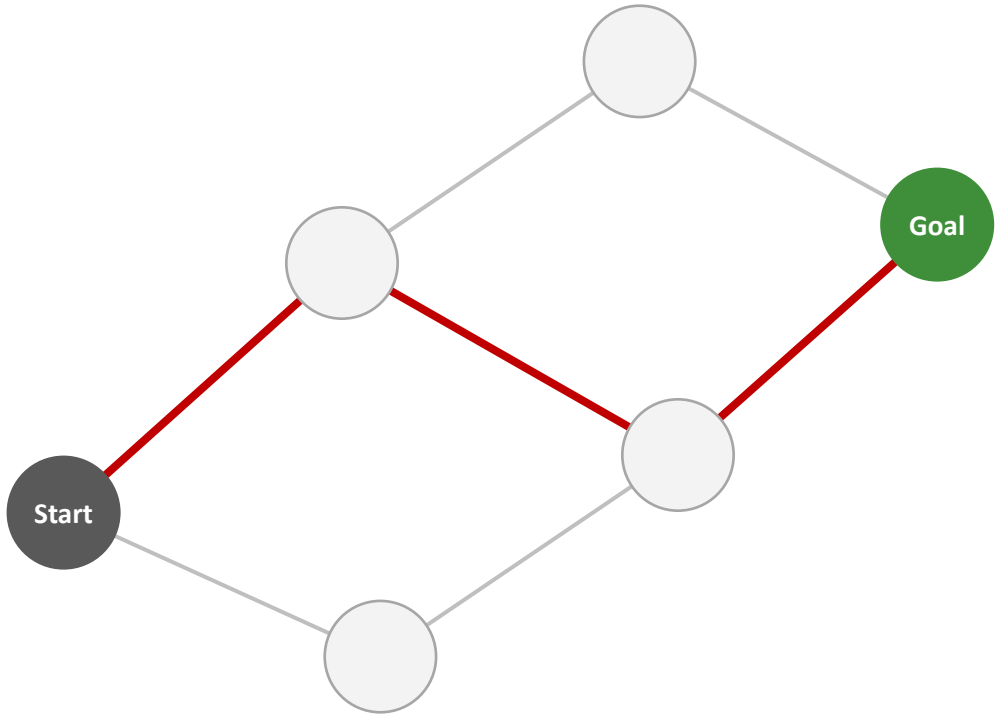
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Background ② Diffusion Models



- **Training**: corrupt real images with noise; learn to predict and remove that noise
- **Generation**: start from pure noise → denoise T times → a **new, realistic** sample
- **Conditional (image-to-image)**: guide generation with an input image — in ViNT, $g(s_i \mid o_t)$ proposes "**places I could reach from here**"
- Strength: **diverse** samples in high-dimensional spaces such as RGB images

Background ③ Topological Graphs & A* Search



node = a place (a stored camera image)

edge = shows the robot can drive between them

- **Topological graph**: nodes are places (camera images) edges link places the robot can drive between — **no metric map**
- **A* search** expands the node

  with the lowest $f(n) = g(n) + h(n)$

  g(n): cost already traveled

  h(n): heuristic estimate of the remaining cost to the goal
- Good heuristic → search heads for the goal; h = 0 → exploration spreads evenly
- **In ViNT**: "physical" A* — the robot actually drives to the chosen node, observes, and expands again