DreamerV2 and Backpropagation-based Policy Gradients with World Models

Presentation by Sergey Sedov

Literature

Do Transformer World Models Give Better Policy Gradients?

DreamerV2: Mastering Atari with Discrete World Models

Lil'Log: Policy Gradient Algorithms

Myriad: a real-world testbed to bridge trajectory optimization and deep learning

Main questions that we are going to answer

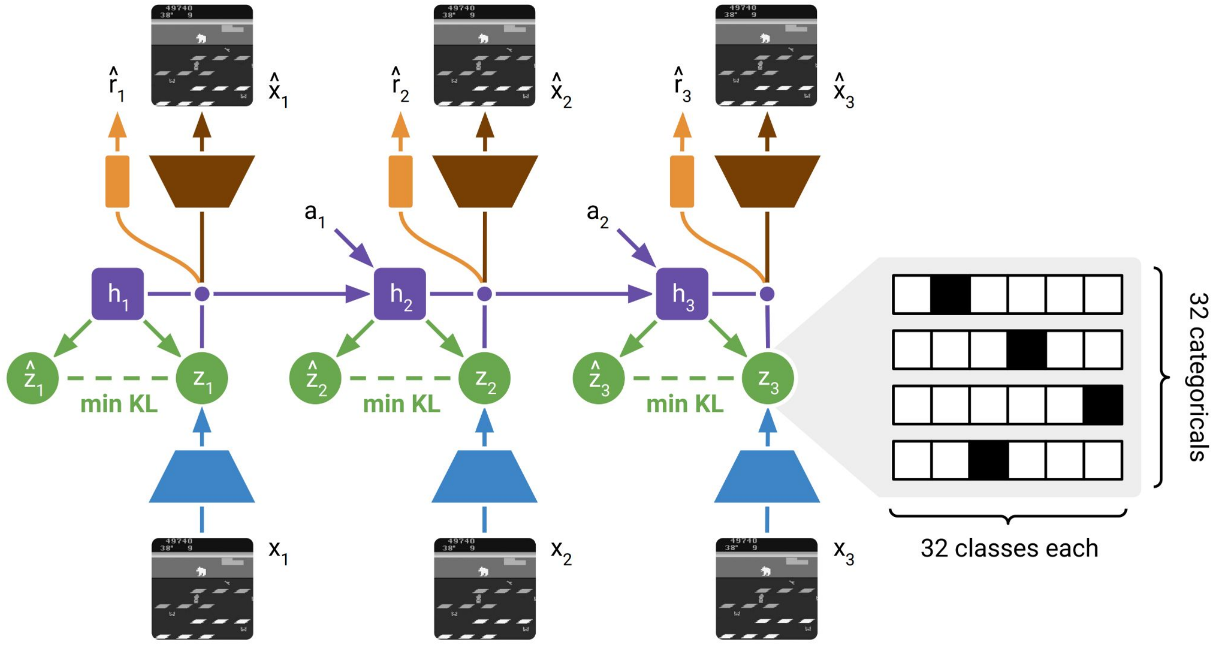
1. How to train an Actor in World Models? DreamerV2 example.
   What are the trade-offs of policy learning in model-based RL methods?

   REINFORCE vs Backpropagation-based Policy Optimization (BPO)

2. Why don’t Transformers succeed in current World Models applications?
3. When are backpropagation-based policy gradients unstable?
4. Can we improve something in terms of policy gradient bounds?

World Models recap: World Model Learning

DreamerV2



RSSM
- Recurrent model: $h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}$
- Representation model: $z_t \sim q_\phi(z_t \mid h_t, x_t)$
- Transition predictor: $\hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t)$

Image predictor: $\hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t)$

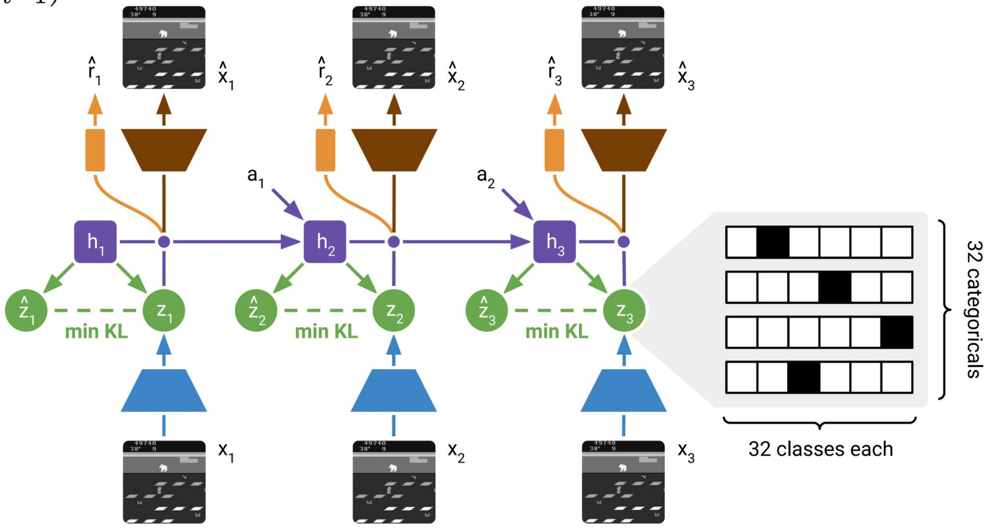
Reward predictor: $\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$

Discount predictor: $\hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid h_t, z_t)$.

World Models recap: World Model Learning

RSSM:

$$
\begin{aligned}
&\text{Recurrent model:} && h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\
&\text{Representation model:} && z_t \sim q_\phi(z_t \mid h_t, x_t) \\
&\text{Transition predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\
&\text{Image predictor:} && \hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t) \\
&\text{Reward predictor:} && \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\
&\text{Discount predictor:} && \hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid h_t, z_t).
\end{aligned}
$$



DreamerV2

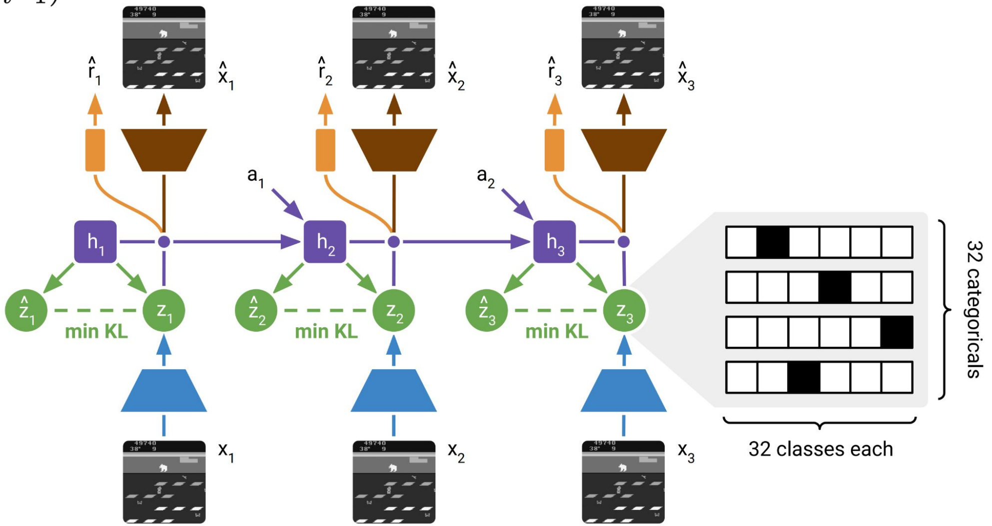
Train jointly on: image reconstruction

$$\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi(z_{1:T} \mid a_{1:T}, x_{1:T})}\Big[\sum_{t=1}^{T} \underbrace{-\ln p_\phi(x_t \mid h_t, z_t)}_{\text{image log loss}}$$

World Models recap: World Model Learning

RSSM $\left\{ \begin{array}{ll} \text{Recurrent model:} & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \text{Representation model:} & z_t \sim q_\phi(z_t \mid h_t, x_t) \\ \text{Transition predictor:} & \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \end{array} \right.$

$$\begin{array}{ll} \text{Image predictor:} & \hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t) \\ \text{Reward predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\ \text{Discount predictor:} & \hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid h_t, z_t). \end{array}$$



Train jointly on: image reconstruction, reward prediction

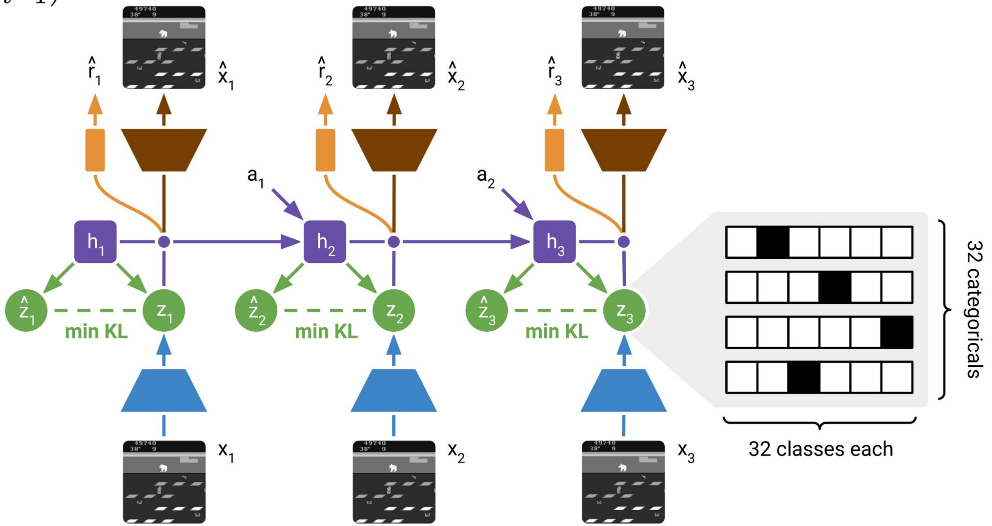
DreamerV2

$$\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi(z_{1:T} \mid a_{1:T}, x_{1:T})}\Big[\textstyle\sum_{t=1}^{T} \underbrace{-\ln p_\phi(x_t \mid h_t, z_t)}_{\text{image log loss}} \underbrace{-\ln p_\phi(r_t \mid h_t, z_t)}_{\text{reward log loss}}$$

World Models recap: World Model Learning

$$\text{RSSM}\begin{cases}\text{Recurrent model:} & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \text{Representation model:} & z_t \sim q_\phi(z_t \mid h_t, x_t) \\ \text{Transition predictor:} & \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t)\end{cases}$$

$$\begin{array}{ll}\text{Image predictor:} & \hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t) \\ \text{Reward predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\ \text{Discount predictor:} & \hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid h_t, z_t).\end{array}$$



Train jointly on: image reconstruction, reward prediction, continue prediction

DreamerV2

$$\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi(z_{1:T} \mid a_{1:T}, x_{1:T})}\Big[\textstyle\sum_{t=1}^{T} \underbrace{-\ln p_\phi(x_t \mid h_t, z_t)}_{\text{image log loss}} \underbrace{-\ln p_\phi(r_t \mid h_t, z_t)}_{\text{reward log loss}} \underbrace{-\ln p_\phi(\gamma_t \mid h_t, z_t)}_{\text{discount log loss}}$$

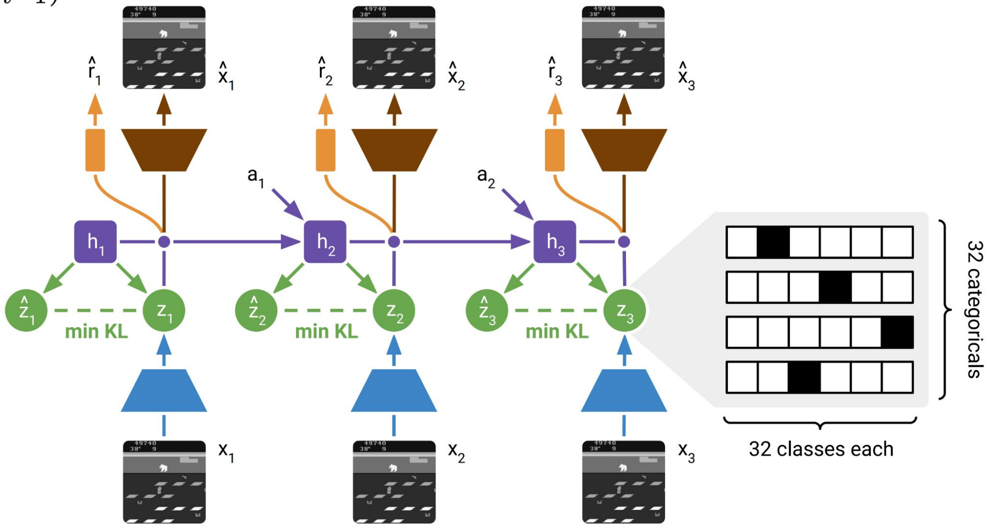
World Models recap: World Model Learning

RSSM { Recurrent model: $h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1})$
Representation model: $z_t \sim q_\phi(z_t \mid h_t, x_t)$
Transition predictor: $\hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t)$ }

Image predictor: $\hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t)$

Reward predictor: $\hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)$

Discount predictor: $\hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid h_t, z_t).$

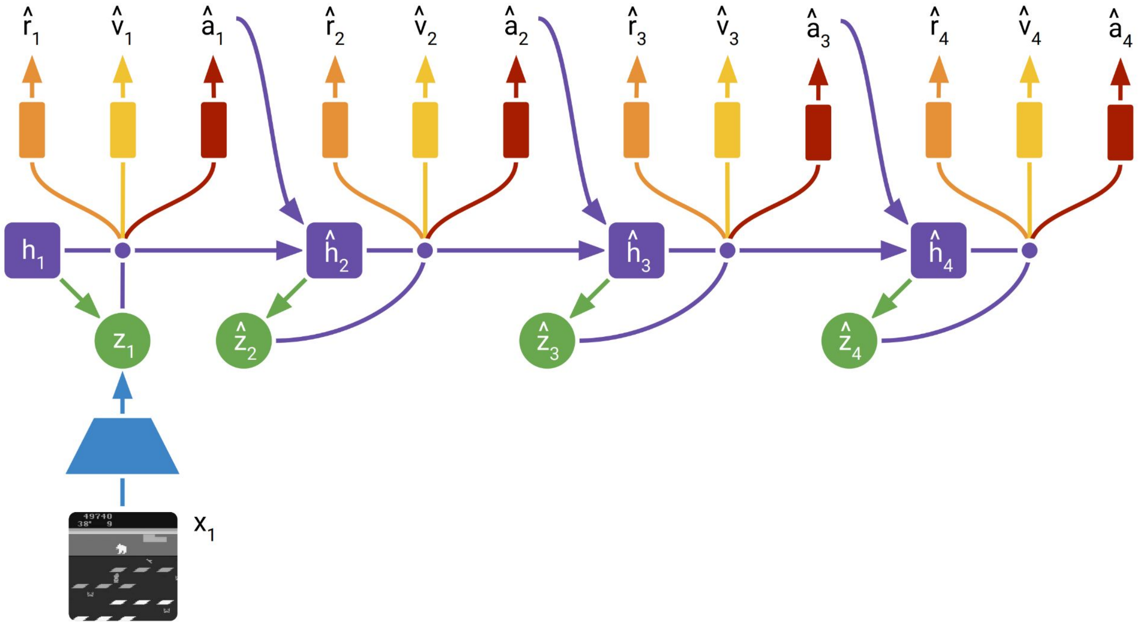


DreamerV2

Train jointly on: image reconstruction, reward prediction, continue prediction, dynamics.

$$\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi(z_{1:T} \mid a_{1:T}, x_{1:T})}\Big[\textstyle\sum_{t=1}^{T} \underbrace{-\ln p_\phi(x_t \mid h_t, z_t)}_{\text{image log loss}} \underbrace{-\ln p_\phi(r_t \mid h_t, z_t)}_{\text{reward log loss}} \underbrace{-\ln p_\phi(\gamma_t \mid h_t, z_t)}_{\text{discount log loss}}$$
$$\underbrace{+\beta\,\mathrm{KL}\big[q_\phi(z_t \mid h_t, x_t) \,\big\|\, p_\phi(z_t \mid h_t)\big]}_{\text{KL loss}}\Big].$$

World Models recap: Actor & Critic Learning



DreamerV2

Actor: $\hat{a}_t \sim p_\psi(\hat{a}_t \mid \hat{z}_t)$ Critic: $v_\xi(\hat{z}_t) \approx \mathrm{E}_{p_\phi, p_\psi}\left[\sum_{\tau \geq t} \hat{\gamma}^{\tau - t} \hat{r}_\tau\right]$

World Models recap: Critic Learning

Critic is trained with MSE loss on λ-target of value function:

$$V_t^\lambda \doteq \hat{r}_t + \hat{\gamma}_t \begin{cases} (1-\lambda) v_\xi(\hat{z}_{t+1}) + \lambda V_{t+1}^\lambda & \text{if} \quad t < H, \\ v_\xi(\hat{z}_H) & \text{if} \quad t = H. \end{cases}$$

Intuitively, it is a weighted average of n-step returns for different horizons, where longer horizons are weighted exponentially less.

World Models recap: Actor Learning

Policy learning objective is much more interesting: it combines Reinforce policy gradient estimate with dynamic backpropagation of value gradients:

$$\mathcal{L}(\psi) \doteq \mathrm{E}_{p_\phi, p_\psi}\Big[\textstyle\sum_{t=1}^{H-1} \big(\underbrace{-\rho \ln p_\psi(\hat{a}_t \mid \hat{z}_t)\, \mathrm{sg}(V_t^\lambda - v_\xi(\hat{z}_t))}_{\text{reinforce}} \underbrace{-(1-\rho) V_t^\lambda}_{\substack{\text{dynamics}\\ \text{backprop}}} \underbrace{-\eta\, \mathrm{H}[a_t \mid \hat{z}_t]}_{\text{entropy regularizer}} \big) \Big]$$

What does it mean? Let’s break it down.

Monte-Carlo Policy Optimization

The goal of the policy optimization is to find θ, such that:

$$\text{maximize} \quad J^f(\boldsymbol{\theta}; H) := \sum_{t=1}^{H} r(s_t)$$

We seek to estimate $\nabla_{\boldsymbol{\theta}} J^f(\boldsymbol{\theta}; H)$ in order to maximize the reward.

In **model-free RL** we don't know the transition function, and REINFORCE algorithm saves us due to Policy Gradient Theorem:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_\pi[Q^\pi(s,a)\nabla_\theta \ln \pi_\theta(a|s)] \\ &= \mathbb{E}_\pi[G_t \nabla_\theta \ln \pi_\theta(A_t|S_t)] \qquad ;\text{Because } Q^\pi(S_t,A_t) = \mathbb{E}_\pi[G_t|S_t,A_t]\end{aligned}$$

From Lil'Log

where we sample G_t from real sample trajectories.

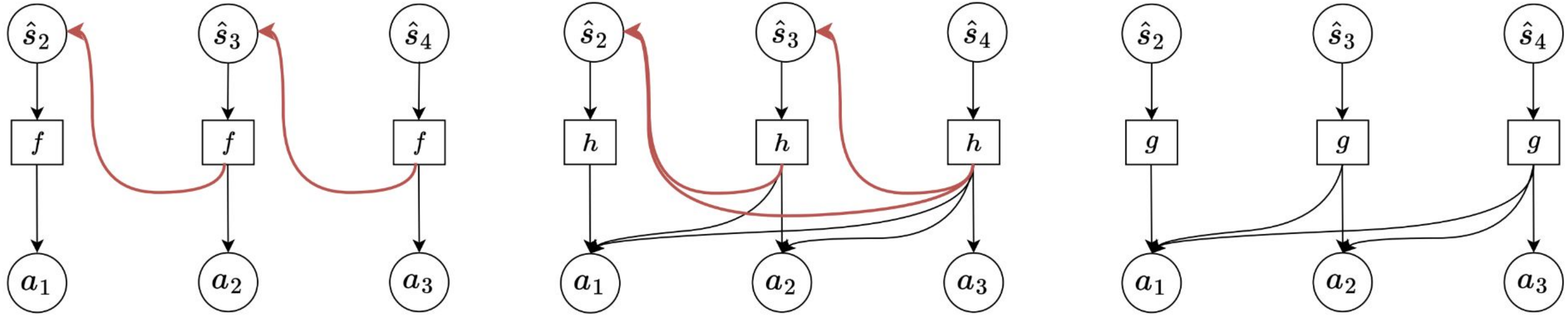
Backpropagation-based Policy Optimization

However, we deal with **model-based** RL methods, as we model the transition function using the World Model itself:

$$\text{maximize} \quad J^f(\boldsymbol{\theta}; H) := \sum_{t=1}^{H} r(s_t), \quad \text{subject to} \quad s_{t+1} = f(s_t, a_t),\, a_k = \pi_{\boldsymbol{\theta}}(s_k)$$

It means that besides using REINFORCE, we can recursively propagate the value gradient to the policy model through transition function f!

Backpropagation-based Policy Optimization



(a) Markovian world model. (b) History World Model. (c) Actions World Model.

Figure 1: **Diagram illustrating gradient flows through different world model types from states to actions.** Circuitous (longer than necessary) gradient paths go through connections highlighted in red. An Actions World Model has no circuitous gradient paths, allowing gradients to directly flow from states to actions through a single application of a world model.

There's no gradient flow from actions to states - authors consider stop-gradient case:

$$\text{maximize} \quad J^f(\boldsymbol{\theta}; H) := \sum_{t=1}^{H} r(s_t), \quad \text{subject to} \quad s_{t+1} = f(s_t, a_t),\ a_k = \pi_{\boldsymbol{\theta}}(\text{sg}[s_k])$$

Backpropagation-based Policy Optimization

Dynamics model loss:

$$\ell^{\hat{f}}(\tau;\boldsymbol{\psi}) = \sum_{t=1}^{H-1} \| s_{t+1}^{\tau} - \hat{f}_{\boldsymbol{\psi}}(s_t^{\tau}, a_t^{\tau}) \|^2,$$

where: $\tau = (s_1^{\tau}, a_1^{\tau}, \ldots, s_H^{\tau})$ is a trajectory

Policy model (Actor):

$$\text{maximize} \quad J^f(\boldsymbol{\theta}; H) := \sum_{t=1}^{H} r(s_t),$$

$$\text{subject to} \quad s_{t+1} = f(s_t, a_t),\ a_k = \pi_{\boldsymbol{\theta}}(\underline{\mathrm{sg}[s_k]})$$

Algorithm 1: Backpropagation-based Policy Optimization (BPO)

Input: Initial buffer $\mathcal{B}$, initial policy parameters $\boldsymbol{\theta}$, initial model parameters $\boldsymbol{\psi}$, learning rates $\{\alpha_{\boldsymbol{\theta}}, \alpha_{\boldsymbol{\psi}}\}$, world model class $\mathcal{W}$.

while not exceeding training steps **do**
  Collect an episode with $\pi_{\boldsymbol{\theta}}$ and add it to $\mathcal{B}$
  for each world model learning step **do**
    $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \alpha_{\boldsymbol{\psi}} \nabla_{\boldsymbol{\psi}} \ell^{\mathcal{W}}(\tau; \boldsymbol{\psi}), \quad \tau \sim \mathcal{B}$
  end for
  for each policy learning step **do**
    Compute $J^{\mathcal{W}}(\boldsymbol{\theta}; \boldsymbol{\psi})$ by unrolling the world model
    Compute $\nabla_{\theta} J^{\mathcal{W}}(\boldsymbol{\theta}; \boldsymbol{\psi})$ by backpropagation
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} J^{\mathcal{W}}(\boldsymbol{\theta}; \boldsymbol{\psi})$
  end for
end while

REINFORCE or Dynamic Backpropagation (BPO)?

$$\mathcal{L}(\psi) \doteq \mathrm{E}_{p_\phi, p_\psi}\Big[\sum_{t=1}^{H-1}\big(\underbrace{-\rho \ln p_\psi(\hat{a}_t \mid \hat{z}_t)\,\mathrm{sg}(V_t^\lambda - v_\xi(\hat{z}_t))}_{\text{reinforce}} \underbrace{-(1-\rho)V_t^\lambda}_{\substack{\text{dynamics}\\ \text{backprop}}} \underbrace{-\eta\,\mathrm{H}[a_t|\hat{z}_t]}_{\text{entropy regularizer}}\big)\Big]$$

DreamerV2 Actor loss combines both REINFORCE and BPO due to the trade-off:

- REINFORCE: requires Monte Carlo sampling of full trajectories, reward signal has **no bias**, but **high variance**, **low sampe efficiency**.
- Backpropagation-based Policy Optimization: **low reward variance**, but has **bias** due to straight-through gradients in World Model.

REINFORCE or Dynamic Backpropagation (BPO)?

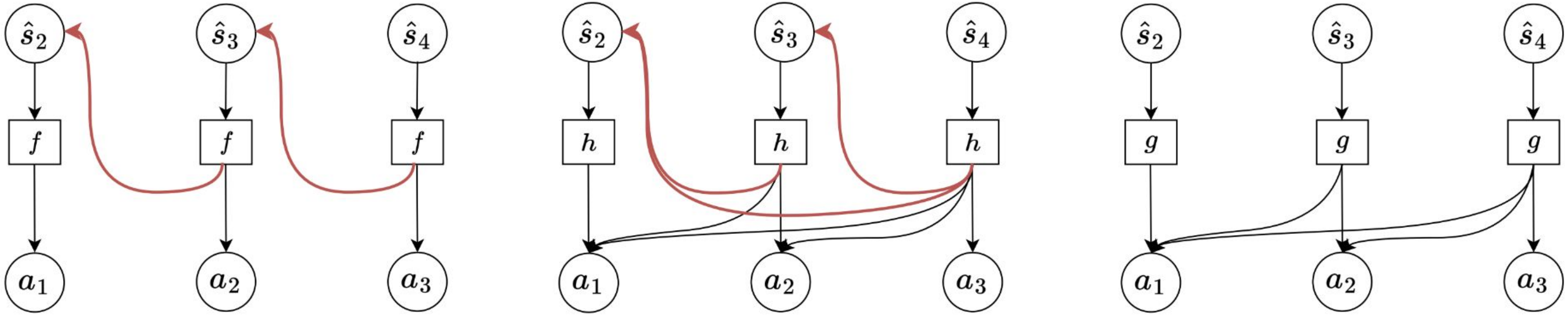
$$\mathcal{L}(\psi) \doteq \mathrm{E}_{p_\phi, p_\psi}\Big[\textstyle\sum_{t=1}^{H-1}\big(\underbrace{-\rho \ln p_\psi(\hat{a}_t \mid \hat{z}_t)\,\mathrm{sg}(V_t^\lambda - v_\xi(\hat{z}_t))}_{\text{reinforce}} \underbrace{-(1-\rho)V_t^\lambda}_{\substack{\text{dynamics}\\ \text{backprop}}} \underbrace{-\eta\,\mathrm{H}[a_t|\hat{z}_t]}_{\text{entropy regularizer}}\big)\Big]$$

DreamerV2 Actor loss combines both REINFORCE and BPO due to the trade-off:

- REINFORCE: requires Monte Carlo sampling of full trajectories, reward signal has **no bias**, but **high variance**, **low sampe efficiency**.
- Backpropagation-based Policy Optimization: **low reward variance**, but has **bias** due to straight-through gradients in World Model.

Is bias the main drawback of BPO? The paper analyzes the following question: when are BPO recurrent gradients unstable?

Transformers in World Models: History World Models



(a) Markovian world model.

(b) History World Model.

(c) Actions World Model.

Figure 1: **Diagram illustrating gradient flows through different world model types from states to actions.** Circuitous (longer than necessary) gradient paths go through connections highlighted in red. An Actions World Model has no circuitous gradient paths, allowing gradients to directly flow from states to actions through a single application of a world model.

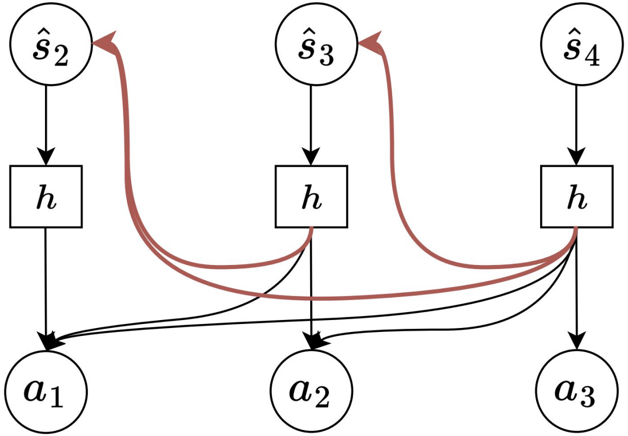
History World Models

History World Models condition on the full history of states and actions:

$$\ell^h(\tau; \boldsymbol{\psi}) = \sum_{t=1}^{H-1} \|s_{t+1}^\tau - h_{\boldsymbol{\psi}}(s_{1:t}^\tau, a_{1:t}^\tau)\|^2$$

$$\text{maximize} \quad J^h(\boldsymbol{\theta}; H) := \sum_{t=1}^{H} r(\hat{s}_t),$$

$$\text{subject to} \quad \hat{s}_{t+1} = h(\hat{s}_{1:t}, a_{1:k}),\ a_k = \pi_{\boldsymbol{\theta}}(\mathrm{sg}[\hat{s}_k])$$



Policy Gradient bound for HWM

Even if the gradient of transformer HWM is bounded, the policy gradient may grow exponentially w.r.t. H, due to circuitous gradient paths:

Theorem 1. *Let the gradient norm of h with respect to its inputs be bounded by L_a and L_s: $\|\frac{\partial h(\hat{s}_{1:t}, a_{1:t})}{\partial a_k}\| \leq L_a$ and $\|\frac{\partial h(\hat{s}_{1:t}, a_{1:t})}{\partial \hat{s}_i}\| \leq L_s$ for all $s_{1:t}, a_{1:t}, k, i$. Let r be the L_r-Lipschitz reward function from a Markov Decision Process $\mathcal{M}$, $\Pi_{\boldsymbol{\theta}}$ a parametric space of differentiable deterministic L_π-policies. Given $\pi_{\boldsymbol{\theta}} \in \Pi_{\boldsymbol{\theta}}$, the norm of the policy gradient $\nabla_{\boldsymbol{\theta}} J^h(\boldsymbol{\theta}; H)$ of $\pi_{\boldsymbol{\theta}}$ under a History World Model h grows asymptotically as a function of the horizon H as:*

$$\|\nabla_\theta J^h(\boldsymbol{\theta}; H)\| = O(HL_r + H^2 L_\pi + H^2 L_a + H^2 L_s^H) = O(L_s^H) \ .$$

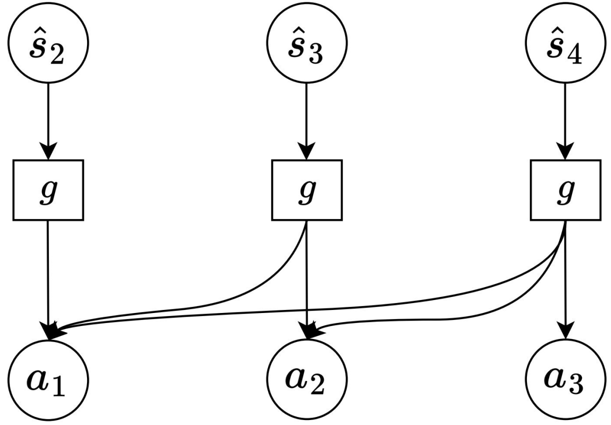
All bounds are tight

Action World Models

Authors aim to find the proper application of Transformers in World Models.
They suggest conditioning on action sequences only:

$$\ell^g(\tau; \boldsymbol{\psi}) = \sum_{t=1}^{H-1} \| s_{t+1}^{\tau} - g_{\boldsymbol{\psi}}(s_1, a_{1:t}^{\tau}) \|^2$$

$$\begin{aligned} &\text{maximize} \quad J^g(\boldsymbol{\theta}; H) := \sum_{t=1}^{H} r(\hat{s}_t), \\ &\text{subject to} \quad \hat{s}_{t+1} = g(s_1, a_{1:t}),\ a_k = \pi_{\boldsymbol{\theta}}(\mathrm{sg}[\hat{s}_k]) \end{aligned}$$



Action World Models

Policy Gradient of AWM with RNN transition model = Policy Gradient of MWM

Proposition 1. *Let f-RNN be a recurrent network with its recurrent cell being the dynamics f of the MDP $\mathcal{M}$, and $g_{f\text{-RNN}}$ denote an AWM instantiated with f-RNN. Then,*

$$\nabla_{\boldsymbol{\theta}} J^{g_{f\text{-RNN}}}(\boldsymbol{\theta}; H) = \nabla_{\boldsymbol{\theta}} J^{f}(\boldsymbol{\theta}; H).$$

The above proposition tells us that the policy gradient computed through a Markovian model is, in fact, equivalent to the one computed through an AWM when instantiating g as a recurrent neural network with a specific recurrent cell. Crucially, this not only provides grounding for gradient estimation with AWMs but also solidifies a fundamental fact that will be analyzed in-depth in this section: policy gradient computation by differentiating through unrolled Markovian models can be understood to be fundamentally ill-behaved due to its correspondence to an RNN structure.

Action World Models

Policy Gradient of RNN-based AWM can explode exponentially => MWM's too!

$$(g_{\mathrm{RNN}}) \qquad x_{t+1} = \sigma(W_x x_t) + W_a a_t + b; \qquad \hat{s}_{t+1} = W_o x_{t+1}, \tag{4}$$

where σ is an activation function with gradient norm bounded by $\|diag(\sigma'(x))\| \leq \frac{1}{\beta}$ for some constant β. Then, the following result holds.

Corollary 2.1. *Let g_{RNN} be an Actions World Model instantiated with a recurrent neural network as in Equation 4 and $\eta = \|W_x^T\|\frac{1}{\beta}$. The asymptotic behavior of the norm of the policy gradient $\nabla_{\boldsymbol{\theta}} J^{g_{\mathrm{RNN}}}(\boldsymbol{\theta}; H)$ as a function of the horizon H can be described as:*

$$\|\nabla_{\boldsymbol{\theta}} J^{g_{\mathrm{RNN}}}(\boldsymbol{\theta}; H)\| = O\left(\eta^H\right).$$

Action World Models

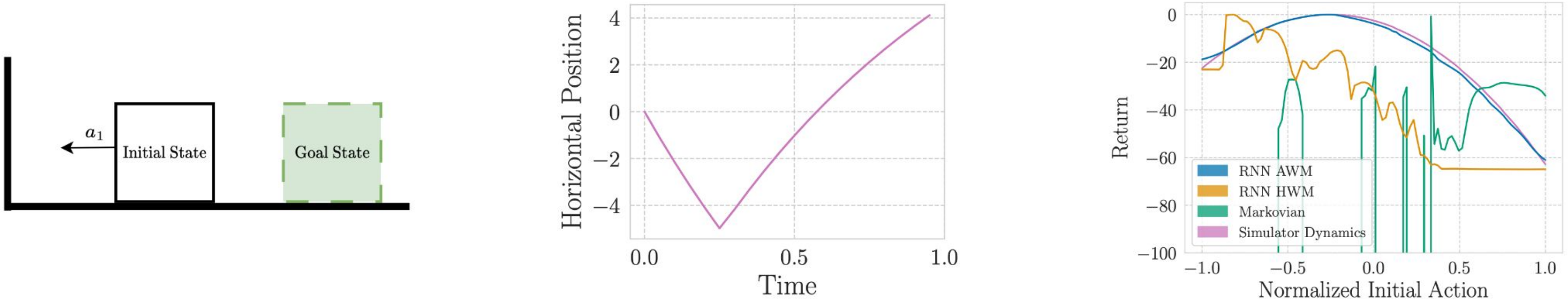
Policy Gradient of Transformer-based AWM is bounded:

Corollary 2.2. *Let g_{ATT} be an attention-based Actions World Model instantiated with self-attention as in equation 5. The asymptotic behavior of the norm of the policy gradient $\nabla_{\boldsymbol{\theta}} J^{g_{ATT}}(\boldsymbol{\theta}; H)$ as a function of the horizon H can be described as:*

$$\|\nabla_{\boldsymbol{\theta}} J^{g_{ATT}}(\boldsymbol{\theta}; H)\| = O\left(H^3\right).$$

Experiments: non-differentiable points

State-conditioning in World Models blows gradients in non-differentiable point:



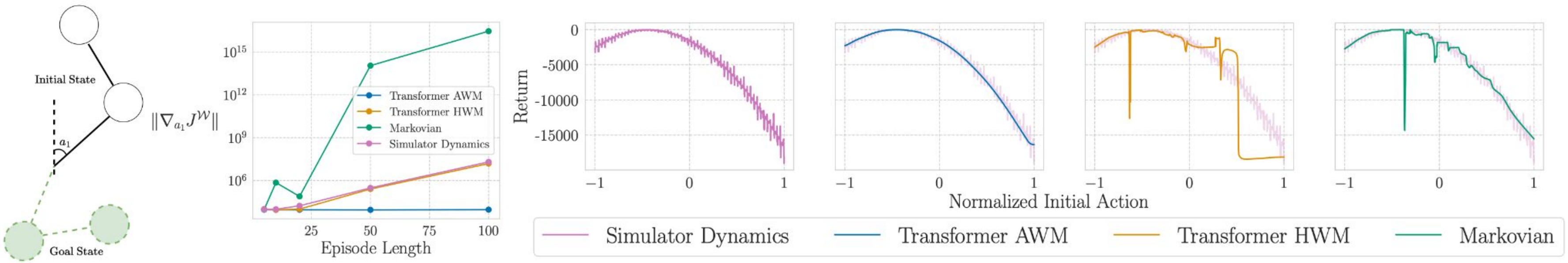
(a) One-bounce environment overview.

(b) Example trajectory with non-differentiable point at $t \approx 0.25$.

(c) Final return with respect to the initial action for different models.

Figure 3: **AWMs ignore non-differentiable points in the state space.** (a) After the block is pushed with some initial action, it bounces off the wall, instantaneously reversing its velocity. (b) Visualization of the point of non-differentiability in the state space. (c) Learning a Markovian model or a HWM causes catastrophic compounding errors, but an AWM can still accurately model the final reward when varying the initial action. Learned dynamics are trained offline on a dataset collected using random actions.

Experiments: Double-pendulum chaotic dynamics



(a) Double-pendulum.

(b) Gradient norms.

(c) Final returns with respect to the initial action for different models when $H = 100$.

Figure 4: **Transformer AWMs smooths out chaotic dynamics.** (a) A double-pendulum environment where an initial position must be chosen in order to achieve some pre-determined goal state after H steps. Different transition models are learned on a data set of random trajectories. (b) The mean gradient norm of the final state with respect to the initial action for each model is computed over 50 different random actions for different horizons. (c) Final return according to different models with respect to different initial actions for $H = 100$.

Experiments: Double-pendulum chaotic dynamics

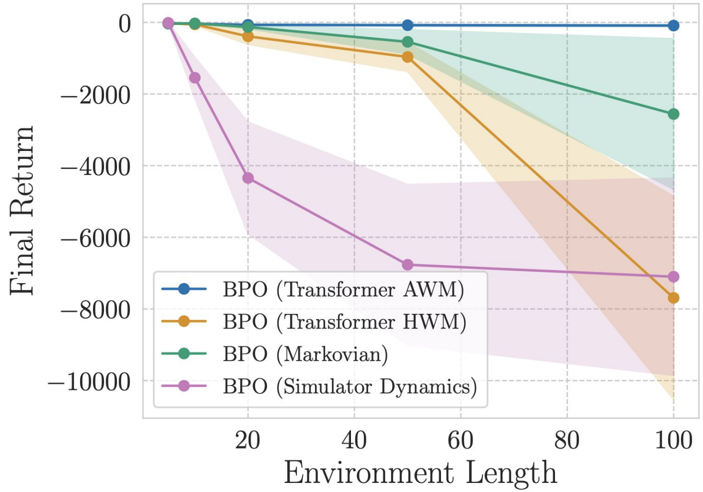
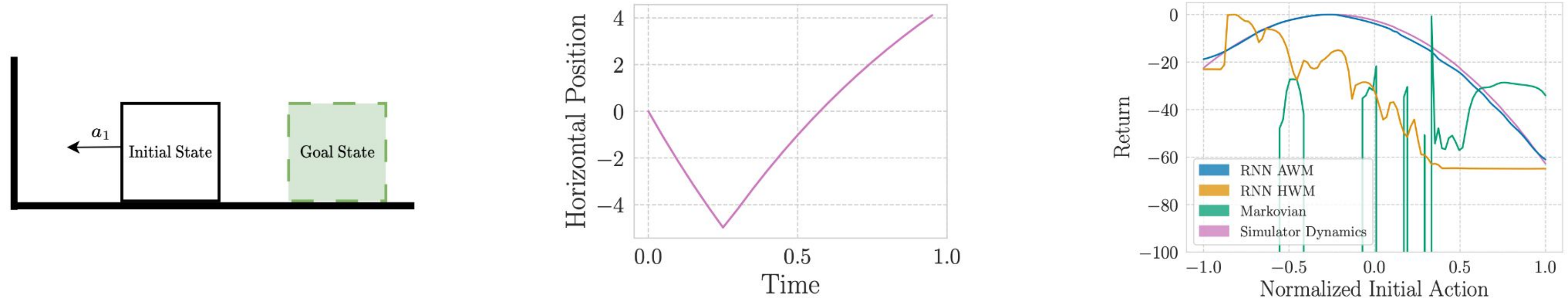


Figure 2: **Transformer AWMs outperforms all BPO baselines in chaotic environments.** Final performances of BPO with different world models on the double-pendulum environment (10 seeds $\pm$ std).

Experiments: non-differentiable points

State-conditioning in World Models blows gradients in non-differentiable point:



(a) One-bounce environment overview.

(b) Example trajectory with non-differentiable point at $t \approx 0.25$.

(c) Final return with respect to the initial action for different models.

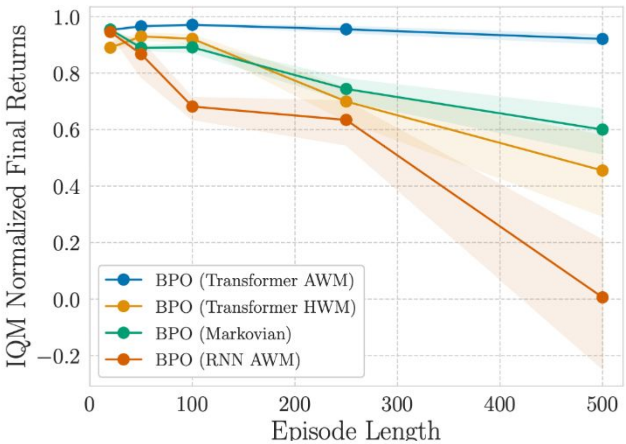
Figure 3: **AWMs ignore non-differentiable points in the state space.** (a) After the block is pushed with some initial action, it bounces off the wall, instantaneously reversing its velocity. (b) Visualization of the point of non-differentiability in the state space. (c) Learning a Markovian model or a HWM causes catastrophic compounding errors, but an AWM can still accurately model the final reward when varying the initial action. Learned dynamics are trained offline on a dataset collected using random actions.

Experiments: Myriad testbed

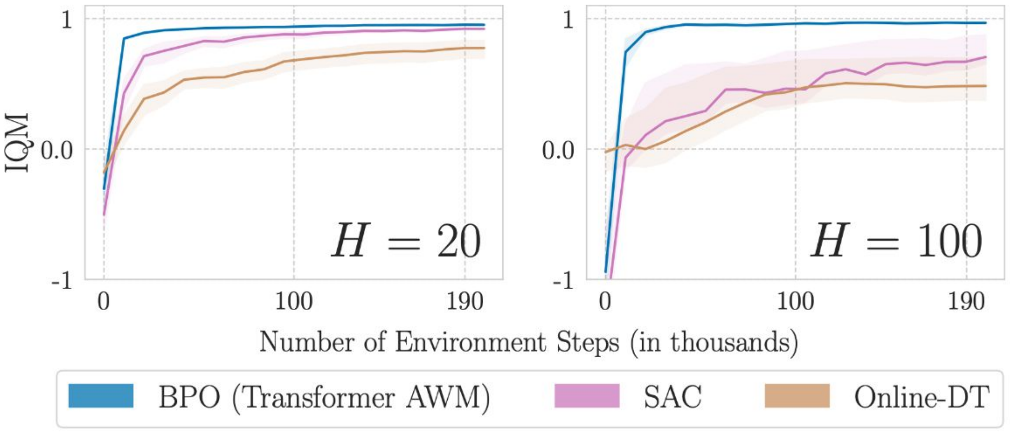
Optimal Control tasks to benchmark RL vs Optimization methods

Name	Brief Description	Fixed $\boldsymbol{x}_T$	Terminal Cost
Bacteria*	Manage bacteria population levels	No	Yes
Bear Populations*	Manage metapopulation of bears	No	No
Bioreactor*	Grow bacteria population	No	No
Cancer Treatment*	Decrease tumour size via chemotherapy	No	No
Cart-Pole Swing-Up	Swing up pendulum by translating pivot	Yes	No
Epidemic*	Control epidemic via vaccination	No	No
Glucose*	Manage blood glucose via insulin injections	No	No
Harvest*	Maximize harvest yield	No	No
HIV Treatment*	Manage HIV via chemotherapy	No	No
Mould Fungicide*	Control mould population via fungicide	No	No
Mountain Car	Drive up valley with limited force	Yes	No
Pendulum	Swing up pendulum by rotating pivot	Yes	No
Predator Prey*	Minimize pest population	Yes	Yes
Rocket Landing	Land a rocket	Yes	No
Simple Case	Use for initial algorithm testing	No	No
Timber Harvest*	Optimize tree harvesting	No	No
Tumour*	Block tumour blood supply	No	Yes
Van Der Pol	Forced Van der Pol oscillator	Yes	No

Experiments: Myriad testbed



(a) Final performances of BPO for different horizons.



(b) Learning curves compared to other baselines.

Figure 5: **Policy optimization with transformer AWMs gives better policies for long horizons.** (a) Final performance of BPO with different world models on Myriad (10 seeds $\pm$ 95% C.I.). (b) Learning curves of BPO through a transformer AWM, a SAC agent, and an Online-DT agent on 20 and 100 length horizons (10 seeds $\pm$ 95% C.I.).

That’s it! Thank you for your attention!

DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning

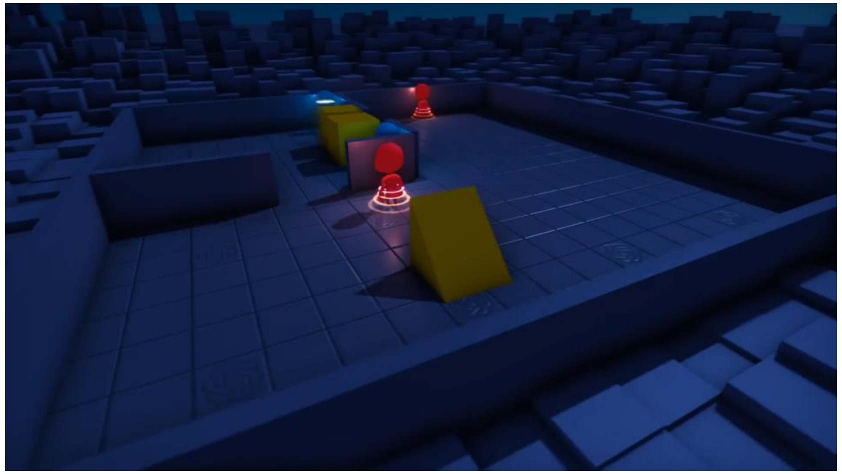
Gaoyue Zhou **Hengkai Pan** **Yann LeCun** **Lerrel Pinto**

Paper presented by **Pratyaksh Prabhav Rao**

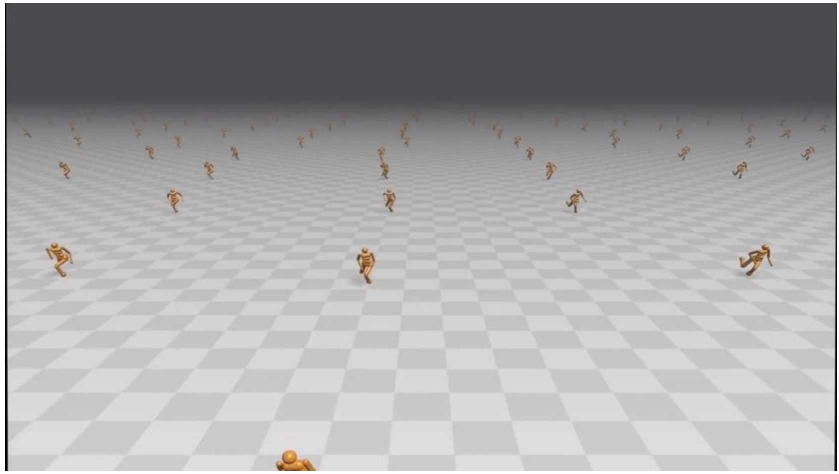
NEW YORK UNIVERSITY

Introduction

Boston Dynamics



DeepMind



NVIDIA

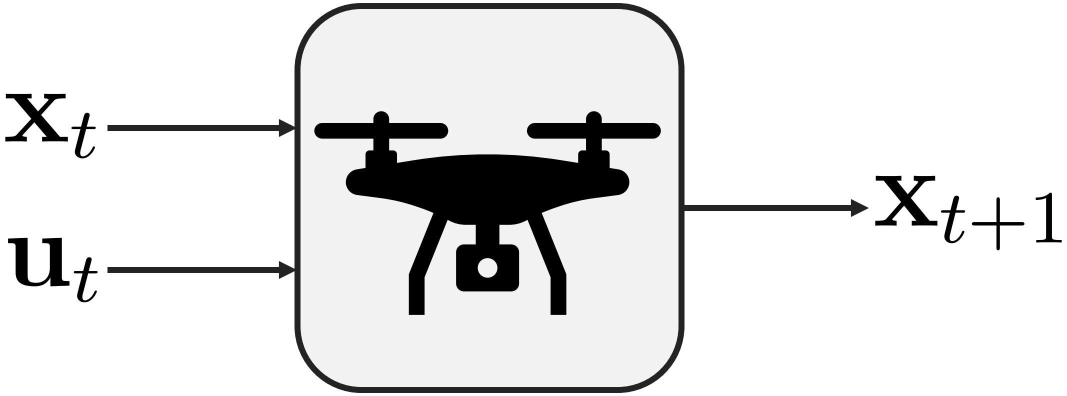
X Unable to generalize

Can we design our decision-making algorithm to effectively generalize to many tasks?

World Models

What is a world model?

$$\mathbf{x}_{t+1} = h(\mathbf{x}_t, \mathbf{u}_t, \theta)$$



Agile Quadrotor Flight [1]

$\mathbf{x}$ is the system state

$\mathbf{u}$ is the control action

h is the dynamics model

θ is the model parameters

[1] Hanover, Drew, Philipp Foehn, Sihao Sun, Elia Kaufmann, and Davide Scaramuzza. "Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors." IEEE Robotics and Automation Letters 7, no. 2 (2021): 690-697.

Related Works

Online Model-based Learning

- ✓ Data efficiency
- ✓ Improve downstream control task
- x World model conditioned on policy
- x Cannot generalize

Offline World Models

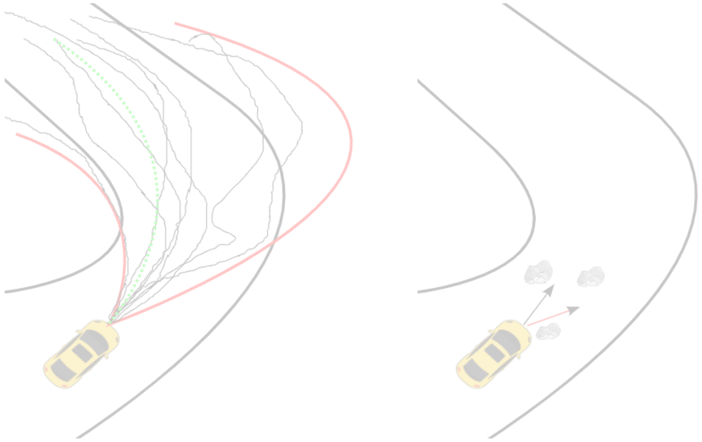
- ✓ High fidelity
- ✓ General purpose
- x Conditioned on text
- x Computationally expensive

Challenges

Trainable on offline, pre-collected trajectories?

Support test-time behavior optimization?

PineconeAI



Model Predictive Control [1]

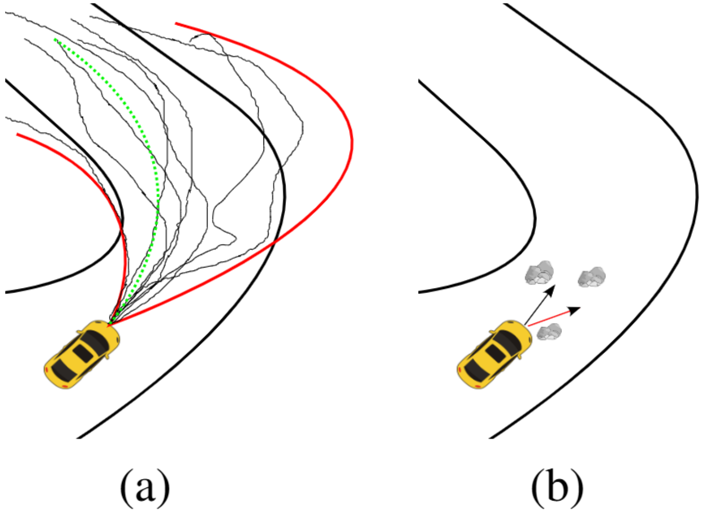
[1] Williams, Grady, Brian Goldfain, Paul Drews, Kamil Saigol, James M. Rehg, and Evangelos A. Theodorou. "Robust Sampling Based Model Predictive Control with Sparse Objective Information." In Robotics: Science and Systems, vol. 14, p. 2018. 2018.

Challenges

Trainable on offline, pre-collected trajectories?

Support test-time behavior optimization?

PineconeAI

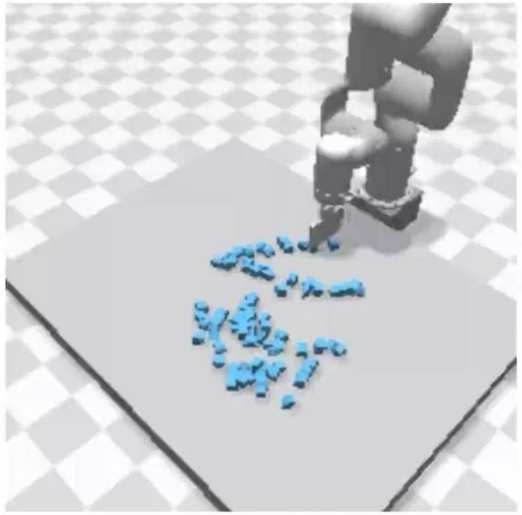


Model Predictive Control [1]

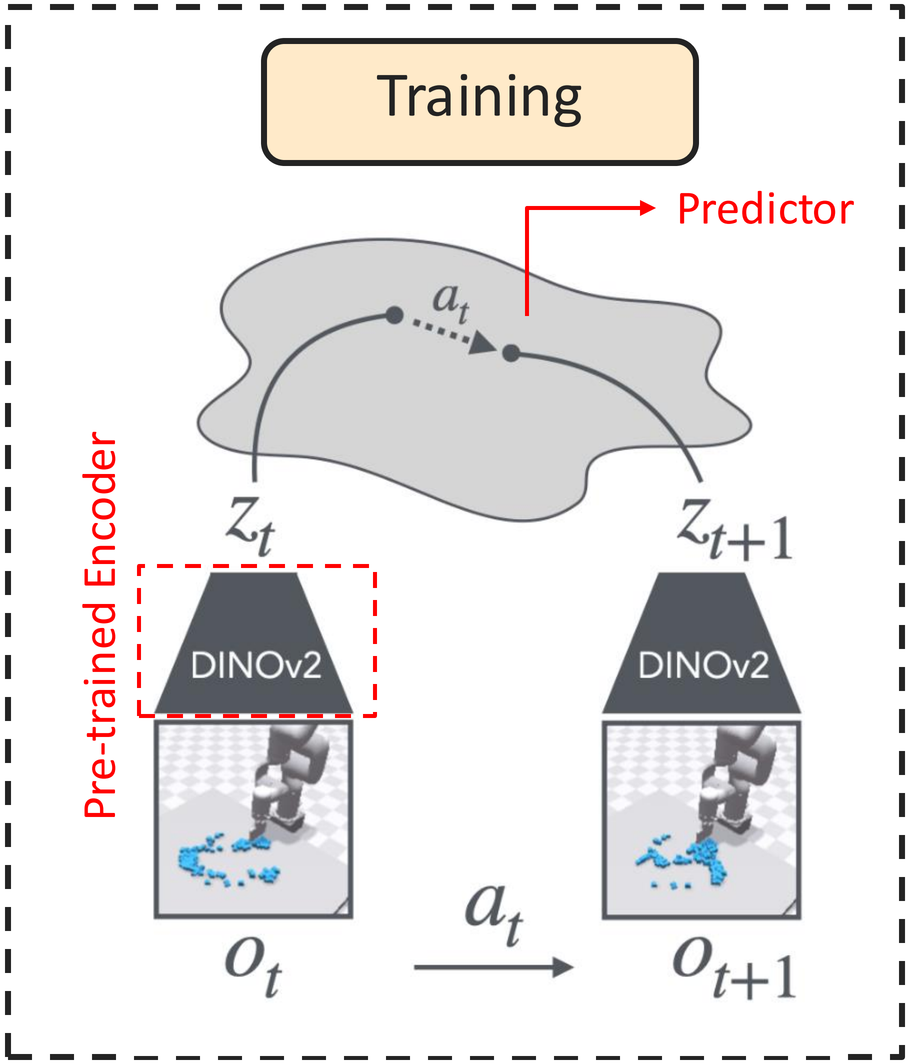
[1] Williams, Grady, Brian Goldfain, Paul Drews, Kamil Saigol, James M. Rehg, and Evangelos A. Theodorou. "Robust Sampling Based Model Predictive Control with Sparse Objective Information." In Robotics: Science and Systems, vol. 14, p. 2018. 2018.

Problem Formulation

Initial State



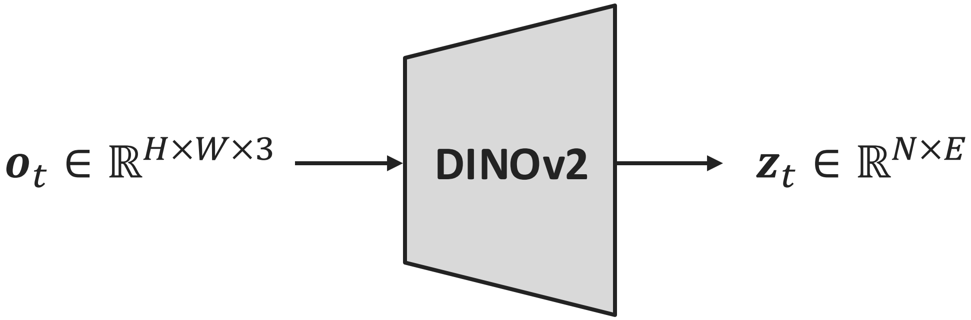
Methodology



$$\mathbf{z}_t = \mathrm{Enc}_\theta(\mathbf{o}_t)$$
$$\mathbf{z}_{t+1} = \mathrm{p}_\theta(\mathbf{z}_{t-H:t}, \mathbf{a}_{t-H:t})$$

H denotes context length

- Observation model remain **frozen** during training
- Extracts Patch level features
- Decoder-only transformer
- Causal attention with auto-regression



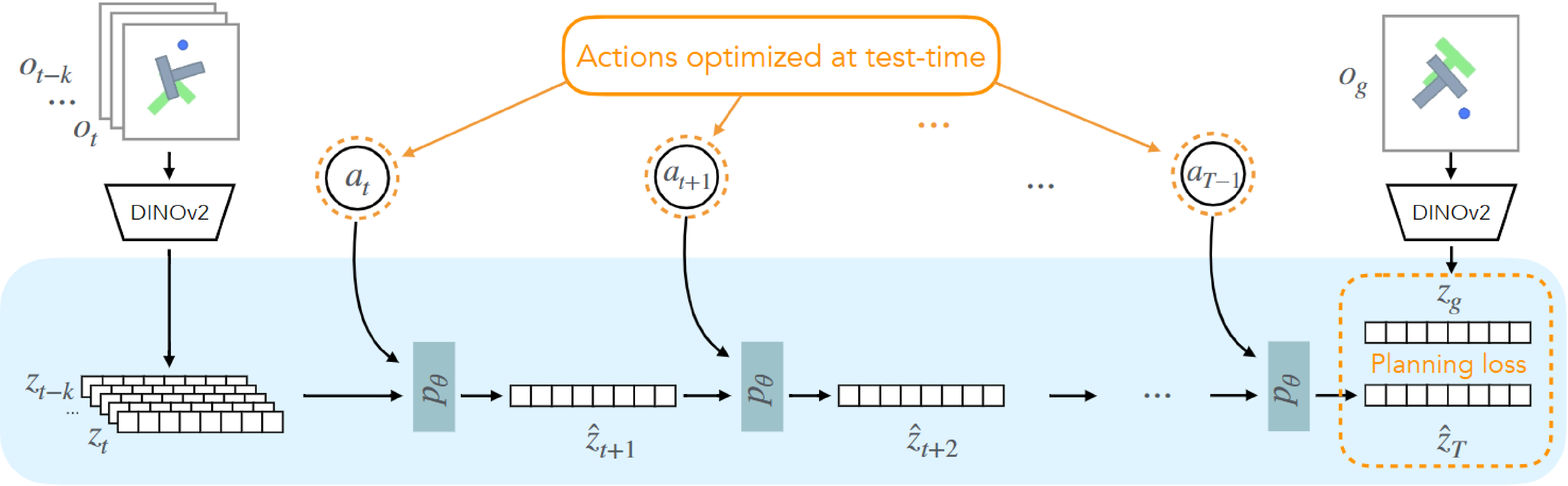
Trained using teacher forcing to prevent collapse

N denotes no. of patches

E denotes embedding dimension

NYU

Methodology



- **Cross-Entropy method (CEM)** optimization

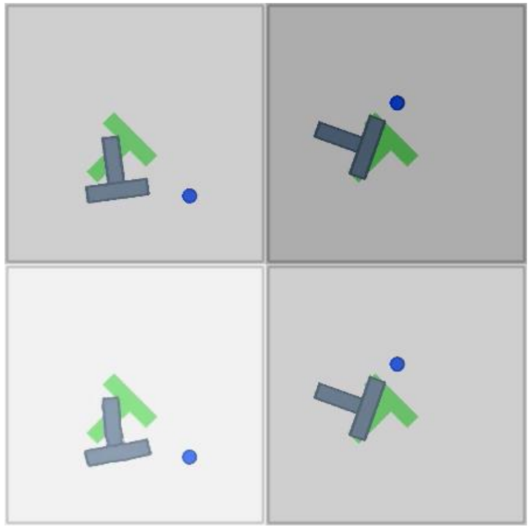
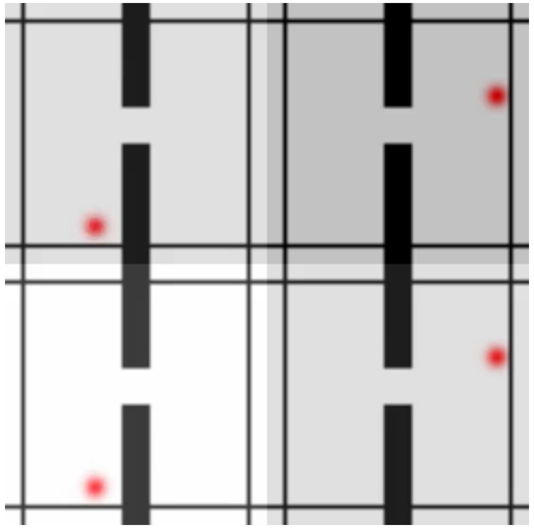
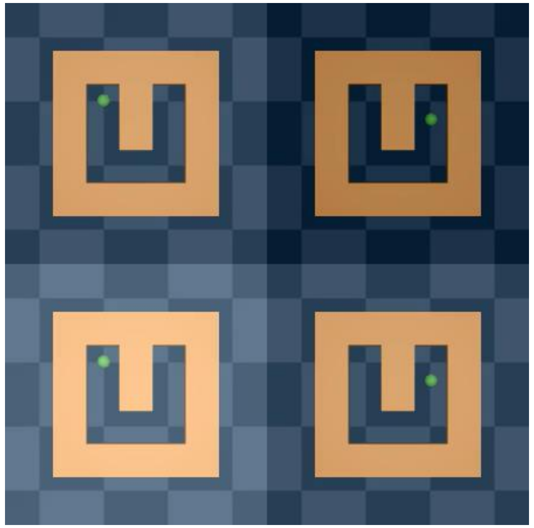
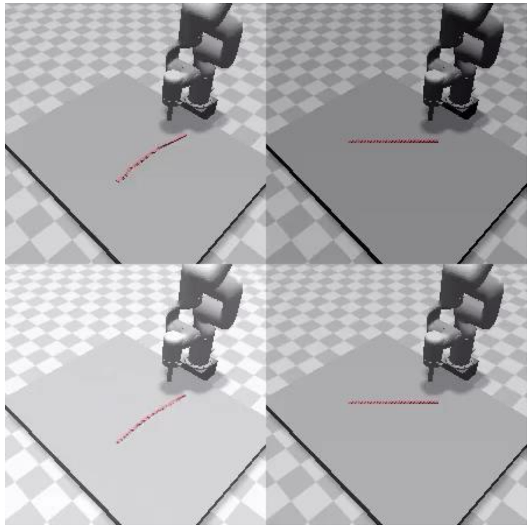
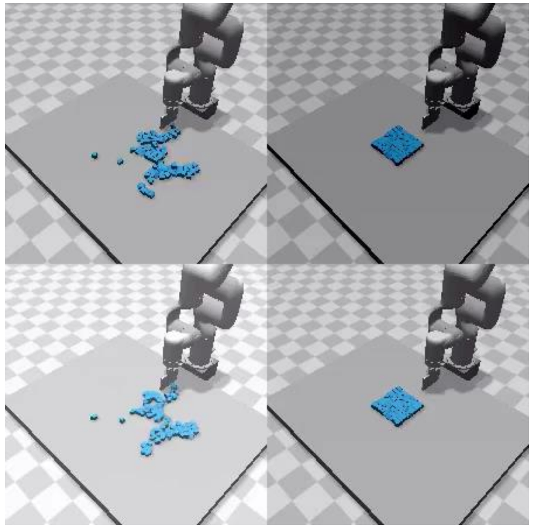
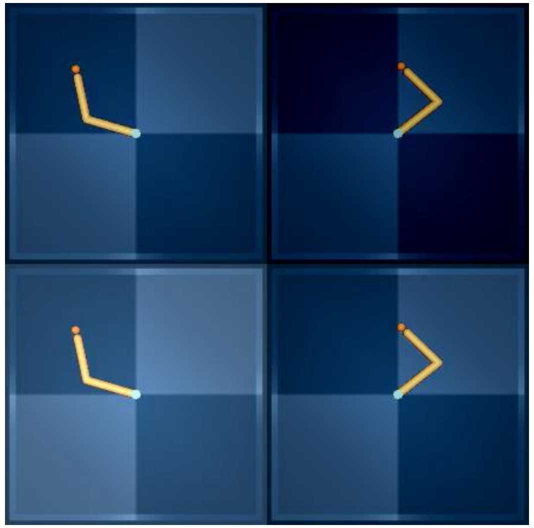
$$C = ||\hat{\boldsymbol{z}}_T - \boldsymbol{z}_g||^2$$

T is planning horizon

$\boldsymbol{z}_g$ denotes goal embedding

Results

Can it be used for Visual Planning ?

Can it Generalize to Unseen Scenarios?

- Dino-WM is compared against wide variety of baselines –
  - IRIS [1]
  - DreamerV3 [2]
  - TD-MPC2 [3]
  - AVDC [4]

Model-based online learning with no reward prediction

Diffusion Model

Table 3. Planning results for offline world models on three suites with unseen environment configurations.

Model	WallRandom SR ↑	PushObj SR ↑	GranularRandom CD ↓
IRIS	0.06	0.14	0.86
DreamerV3	0.76	0.18	1.53
R3M	0.40	0.16	1.12
ResNet	0.40	0.14	0.98
DINO CLS	0.64	0.18	1.36
Ours	**0.82**	**0.34**	**0.63**

[1] Micheli, Vincent, Eloi Alonso, and François Fleuret. "Transformers are sample-efficient world models." arXiv preprint arXiv:2209.00588 (2022).
[2] Hafner, Danijar, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. "Mastering diverse domains through world models." arXiv preprint arXiv:2301.04104 (2023).
[3] Hansen, Nicklas, Hao Su, and Xiaolong Wang. "Td-mpc2: Scalable, robust world models for continuous control." arXiv preprint arXiv:2310.16828 (2023).
[4] Ko, Po-Chen, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B. Tenenbaum. "Learning to act from actionless videos through dense correspondences." arXiv preprint arXiv:2310.08576 (2023).

Conclusion

- DINO-WM models **visual dynamics** in **latent space** and generalizes to unseen simulation setups
- Enables **zero-shot planning**
- **Limitations** –
  - DINO-WM assumes having access to offline datasets with sufficient state-action coverage
  - No real-world experiments

Thanks!

Questions?

NYU

Diffusion for World Modeling: Visual Details Matter in Atari

Rooholla Khorrambakht

rk4342@nyu.edu

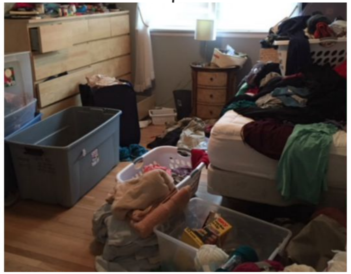
Relationship to Embodied AI

It is very hard to have robots in the wild because:

- Hand-crafting mathematical states often intractable
- Objective/reward definition is ambiguous for many tasks

$$\dot{x} = f(x, u) \ where \ x = ?$$

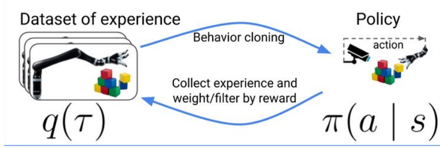
$$l(x_i) = ?$$



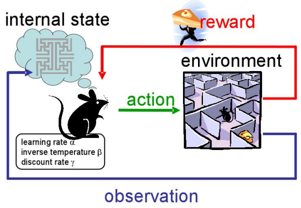
System-1: Habits and Memorization

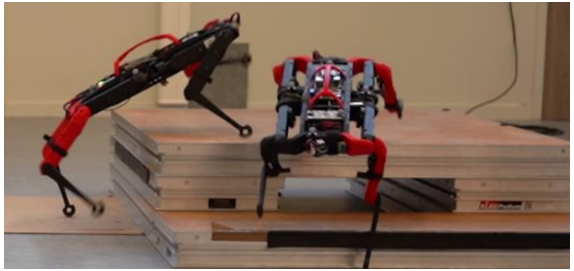
Memorizing the mapping from vision/observation to action:

Behavior Cloning



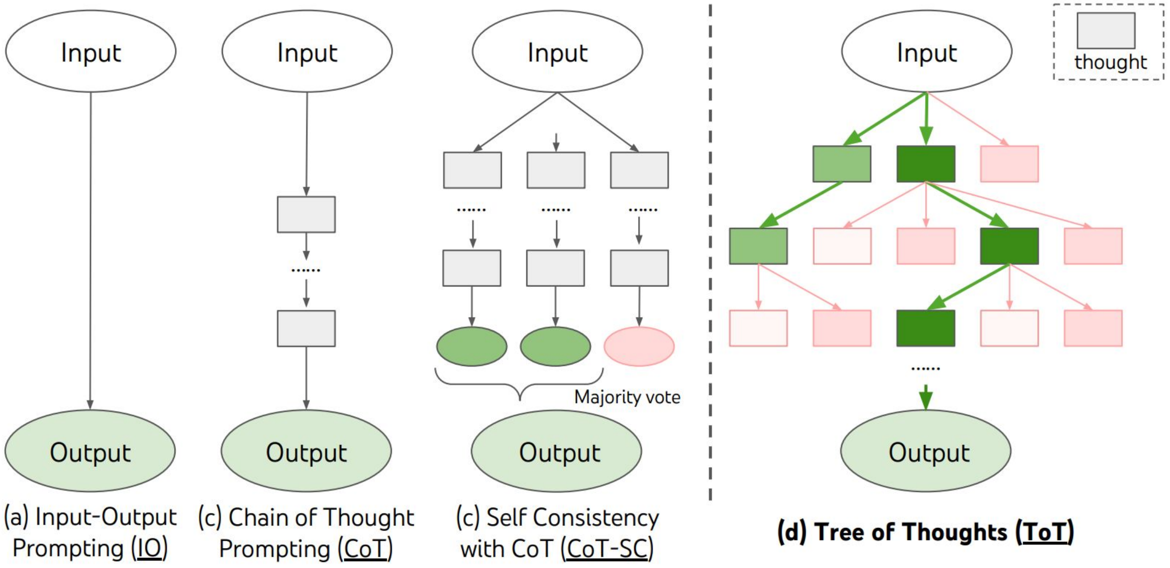
Reinforcement Learning





System-2: Planning and Search

We want to spend more compute for complicated/unseen tasks:

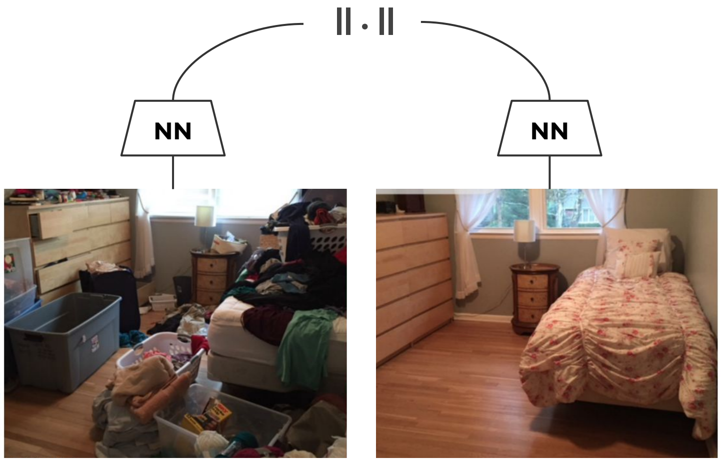


Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models, 2023." *URL https://arxiv. org/pdf/2305.10601. pdf* (2023).

System-2: Ingredients

Two main components needed to formulate the search/optimization:

Dynamics Model
(World Model)

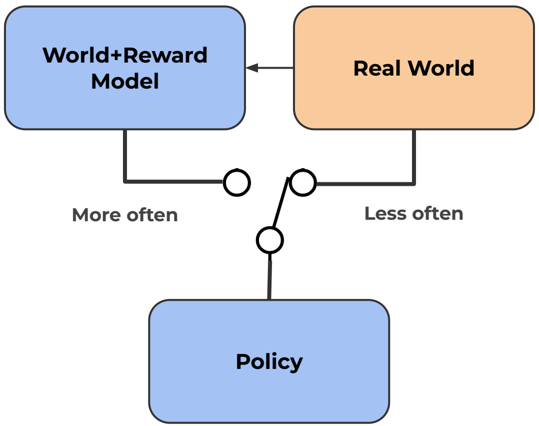


Reward/Value Model

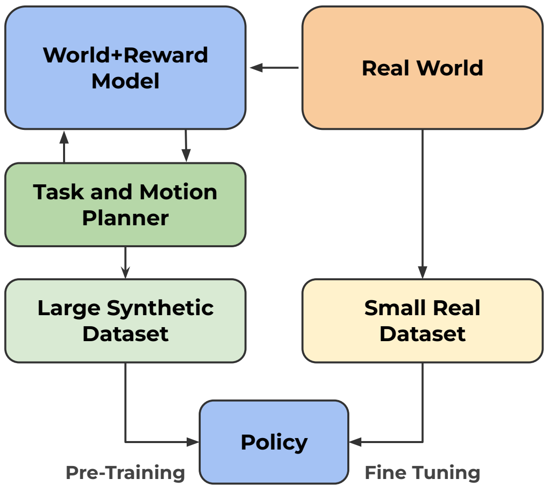
System-2 to System-1 Distillation

Deliberate thinking for everything we do it too expensive. Generalizable skills should be memorized like words in a language.

Learning In Imagination in RL



Bootstrapping in Behavior Cloning



NYU

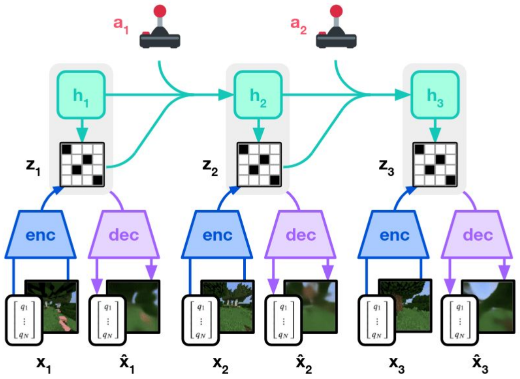
Diffusion for World Modeling: Visual Details Matter in Atari

(NeurIPS 2024)

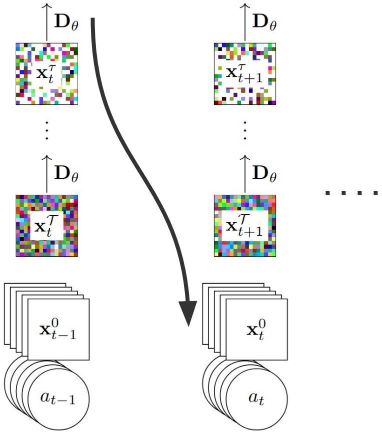
Motivation

The paper is motivated by:

- Lossy compression in latent-dynamics models
- Diffusion models are very successful in high-fidelity image synthesis



Before



DIAMOND

NYU

Contributions

This is an important paper because:

- Computationally efficient world models leveraging NVIDIA EDM diffusion models.
- Joint training of state-of-the-art RL agents in the world-model’s imagination
- Demonstrating the world-model’s long rollouts on challenging 3D games
- Well-documented and reproducible implementation

Diffusion Recap

Diffusion models learn to reverse a progressive noising process (diffusion) to recover clean data points.

Diffusion Process

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, \tau)d\tau + g(\tau)d\mathbf{w}$$

Reverse Process (Also Diffusion)

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, \tau) - g(\tau)^2 \nabla_{\mathbf{x}} \log p^{\tau}(\mathbf{x})]d\tau + g(\tau)d\bar{\mathbf{w}}$$

$$\Big\downarrow \text{ Affine Process}$$

Denoising Score Matching

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\mathbf{S}_\theta(\mathbf{x}^\tau, \tau) - \nabla_{\mathbf{x}^\tau} \log p^{0\tau}(\mathbf{x}^\tau \mid \mathbf{x}^0)\|^2\right]$$

$$\Big\downarrow \text{ P is known Gaussian Distribution}$$

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\mathbf{D}_\theta(\mathbf{x}^\tau, \tau) - \mathbf{x}^0\|^2\right], \quad \mathbf{D}_\theta(\mathbf{x}^\tau, \tau) = \mathbf{S}_\theta(\mathbf{x}^\tau, \tau)\sigma^2(\tau) + \mathbf{x}^\tau$$

Practical Choice of Diffusion Paradigm

EDM expands the design exploration space and propose efficient and modular preconditioning, noise scheduling, and integration methods.

$$p^{0\tau}(\mathbf{x}_{t+1}^{\tau} \mid \mathbf{x}_{t+1}^{0}) = \mathcal{N}(\mathbf{x}_{t+1}^{\tau}; \mathbf{x}_{t+1}^{0}, \sigma^2(\tau)\mathbf{I})$$

Diffusion Process

Empirical schedule to keep the objective medium variance.

$$\mathbf{D}_\theta(\mathbf{x}_{t+1}^{\tau}, y_t^{\tau}) = c_{\text{skip}}^{\tau}\, \mathbf{x}_{t+1}^{\tau} + c_{\text{out}}^{\tau}\, \mathbf{F}_\theta\left(c_{\text{in}}^{\tau}\, \mathbf{x}_{t+1}^{\tau}, y_t^{\tau}\right)$$

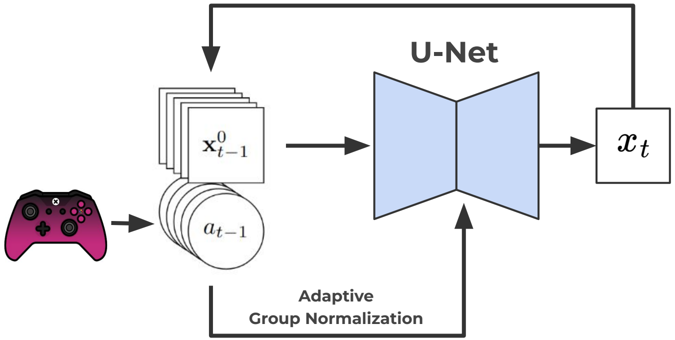
Preconditioning

$$\mathcal{L}(\theta) = \mathbb{E}\left[\| \underbrace{\mathbf{F}_\theta\left(c_{\text{in}}^{\tau}\mathbf{x}_{t+1}^{\tau}, y_t^{\tau}\right)}_{\text{Network prediction}} - \underbrace{\frac{1}{c_{\text{out}}^{\tau}}\left(\mathbf{x}_{t+1}^{0} - c_{\text{skip}}^{\tau}\mathbf{x}_{t+1}^{\tau}\right)}_{\text{Network training target}} \|^2\right]$$

Karras, Tero, et al. "Elucidating the design space of diffusion-based generative models." *Advances in neural information processing systems* 35 (2022): 26565-26577.

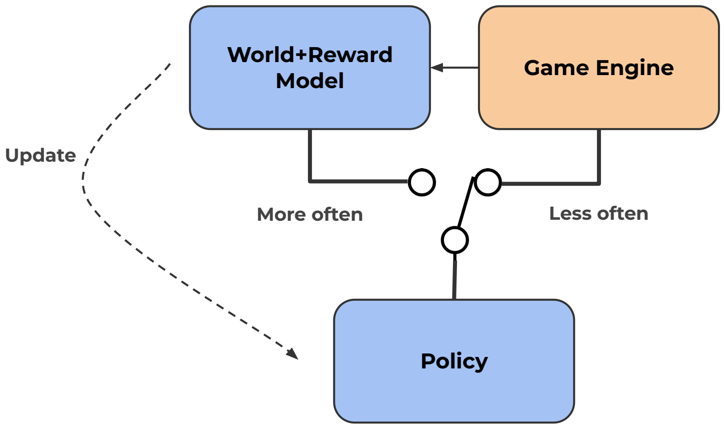
Architecture

A CNN-based conditional autoregressive architecture is adopted:



RL Training in Imagination

Reinforce algorithm is used to train an agent entirely in the imagination of the world model:

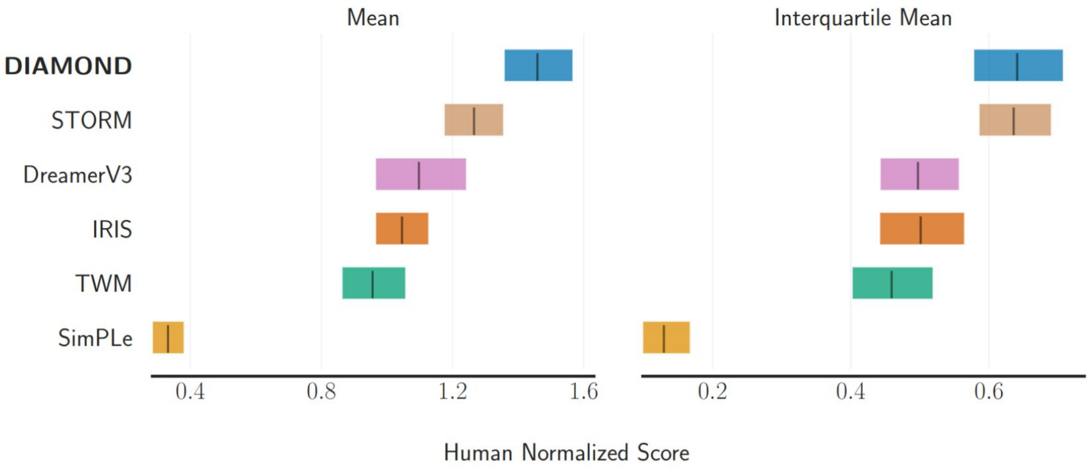


Qualitative Results

Authors train the world-model for Atari (jointly with an RL player) and Counter-Strike 3D game (off policy).

Atari 100K Benchmark

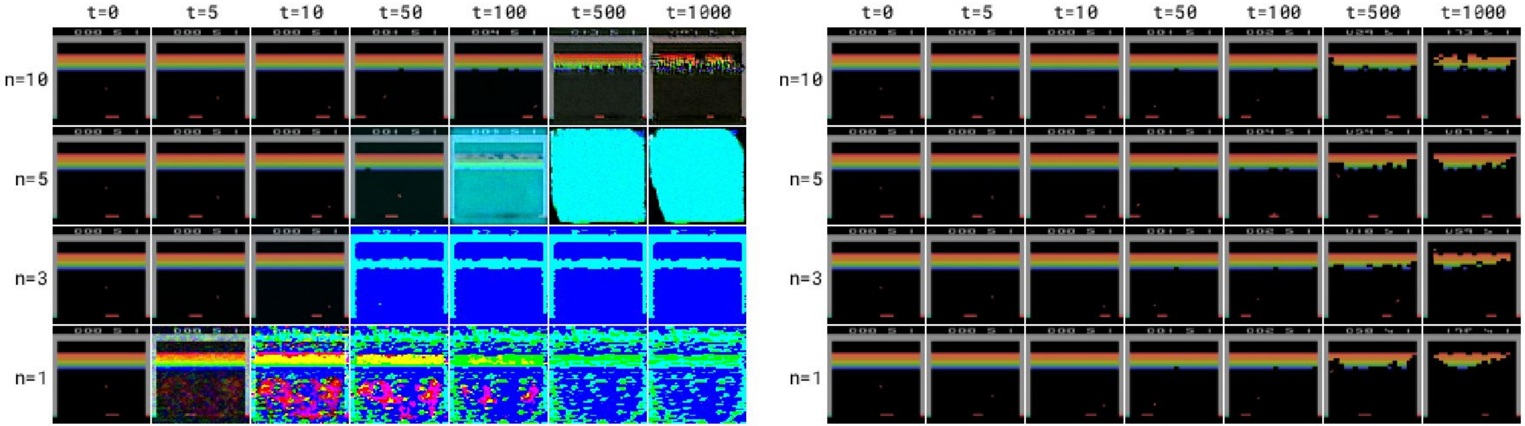
Thanks to the world model (learning in imagination) the agent learns to play each game after 100K interactions as opposed to 50M steps.



NYU

Choice of Diffusion Framework

Thanks to the world model (learning in imagination) the agent learns to play each game after 100K interactions as opposed to 50M steps.



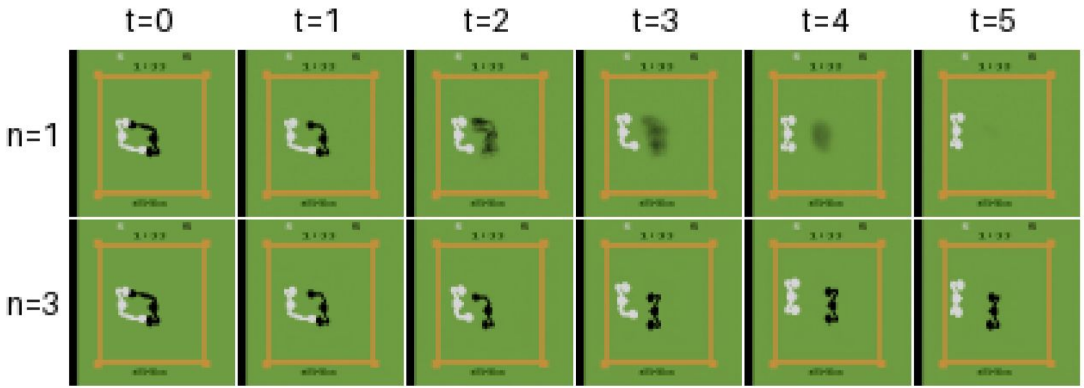
(a) DDPM-based world model trajectories.

(b) EDM-based world model trajectories.

NYU

Number of Denoising Steps

Multiple number of steps is essential for capturing the multi-modality. The black boxer disappears due to mode averaging:



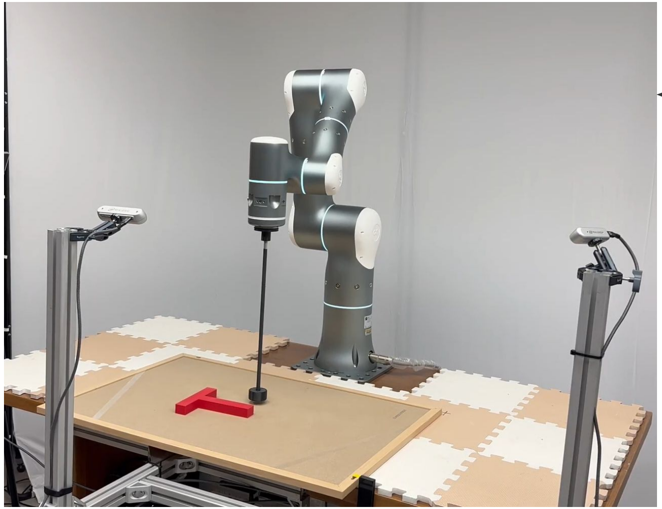
Limitations

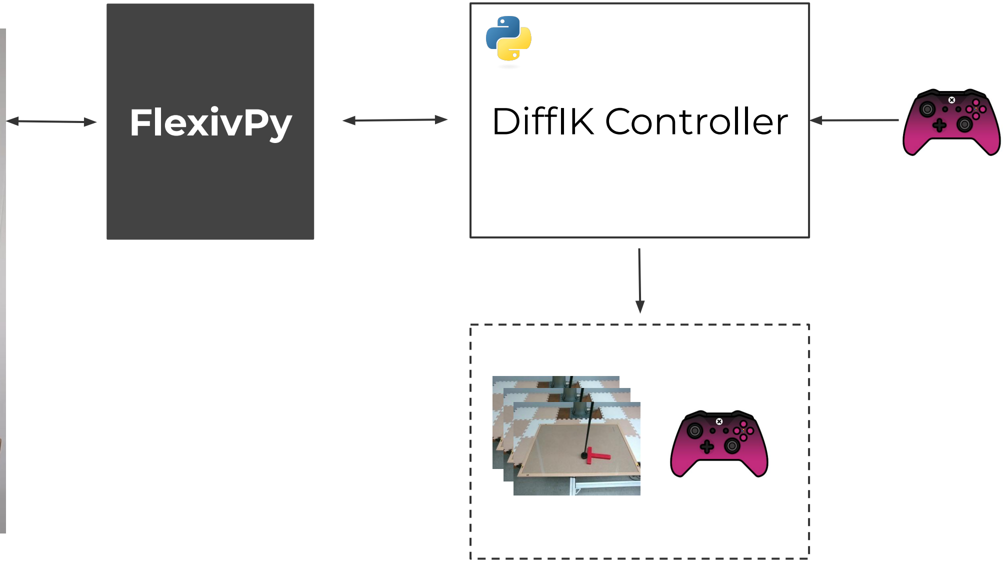
DIAMOND is subject to the following issues:

- Computationally expensive
- No direct way of leveraging pre-trained encoders (e.g. DINO)
- Evaluation of RL training is only done for discrete action environments
- The architecture has a shallow memory (as deep as the input frames stack)

Demo: Training on Custom Data

Next we trained this world model on a real-world robotic setup:



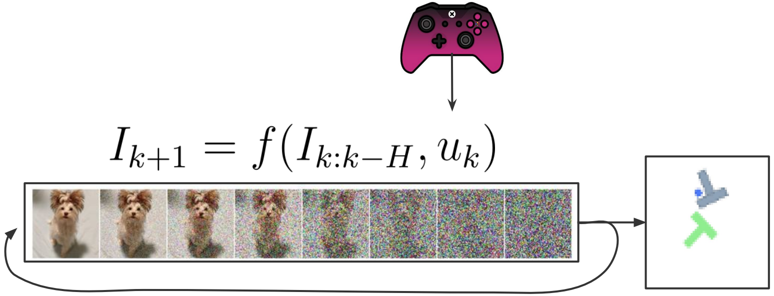


4 Hours of Data While Interacting with the T

Interacting with the Learned World Model

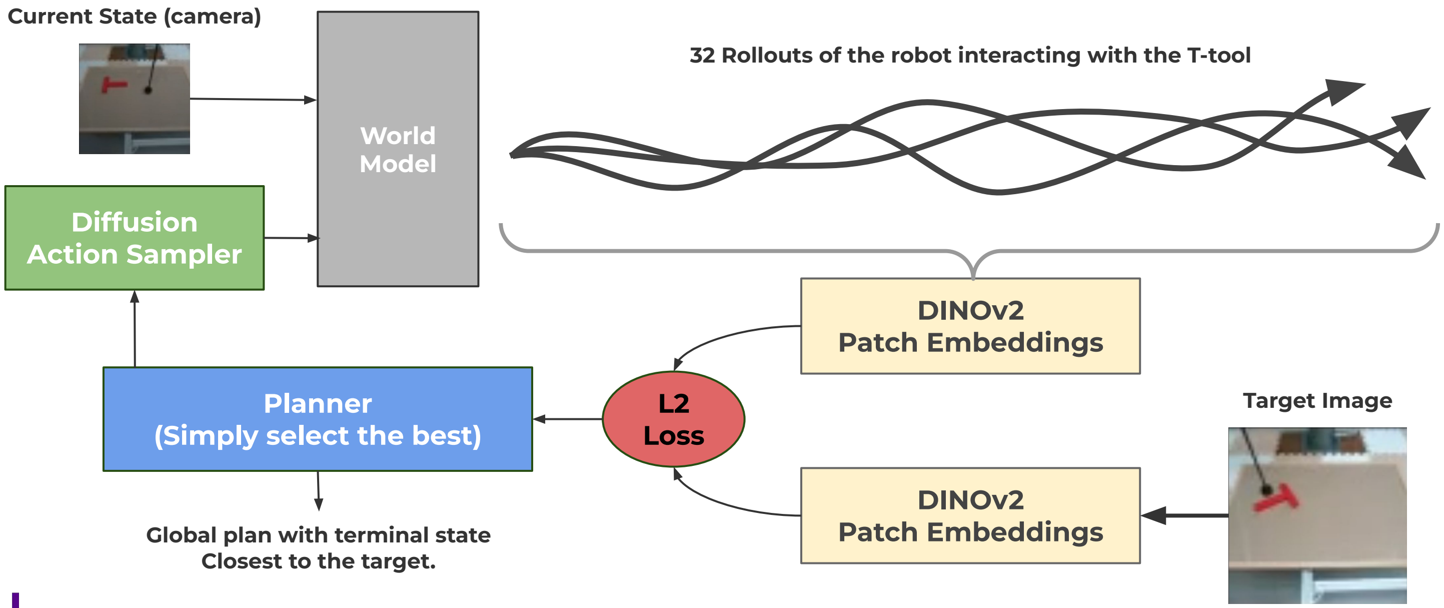
The world model captures the contact-rich interaction dynamics:

$$I_{k+1} = f(I_{k:k-H}, u_k)$$



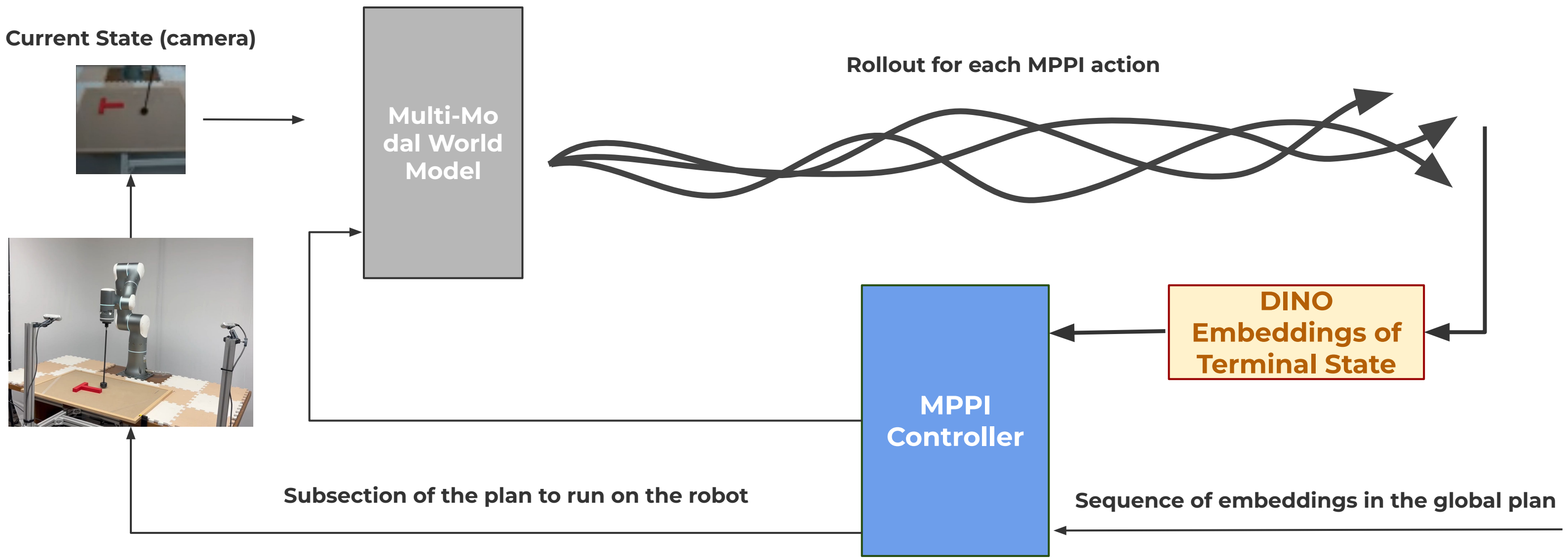
Global Planner

In addition to the world model, we also trained a diffusion based sampler to generate random interaction rollouts. We used this sampler to make a simple global planner:



Local Planner

With the global plan as input, the MPPI planner closes the feedback between vision and the actions to make sure the robot stays close to the global plan:



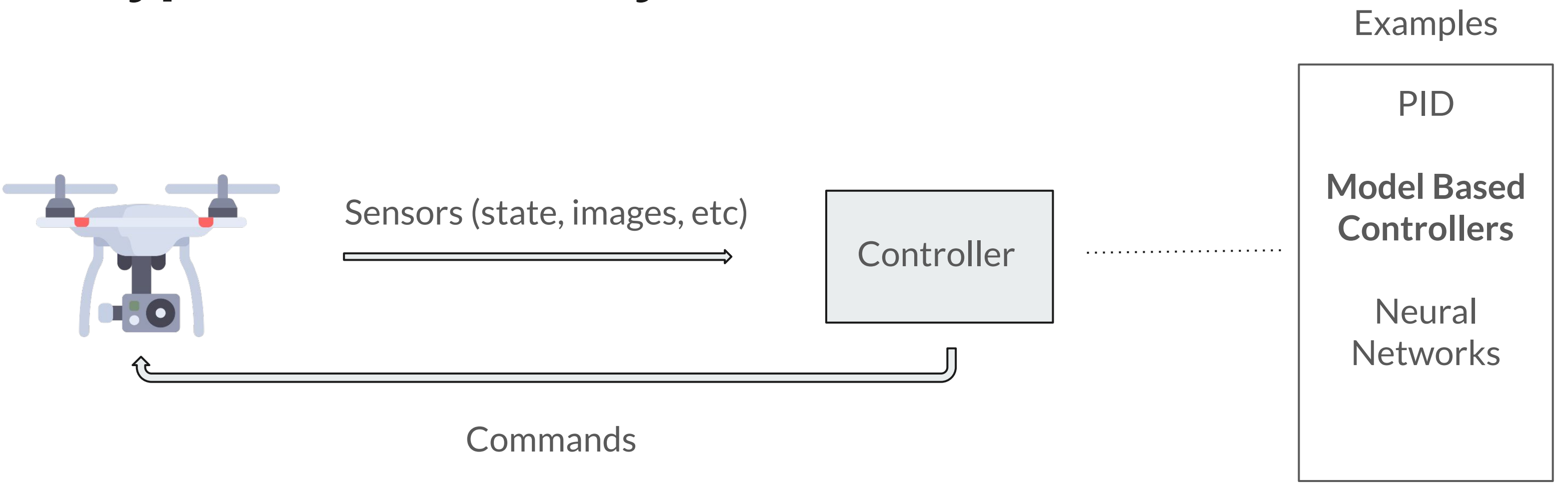
Running On The Robot

Further optimization is required but our preliminary deployments was promising:

Differential MPC

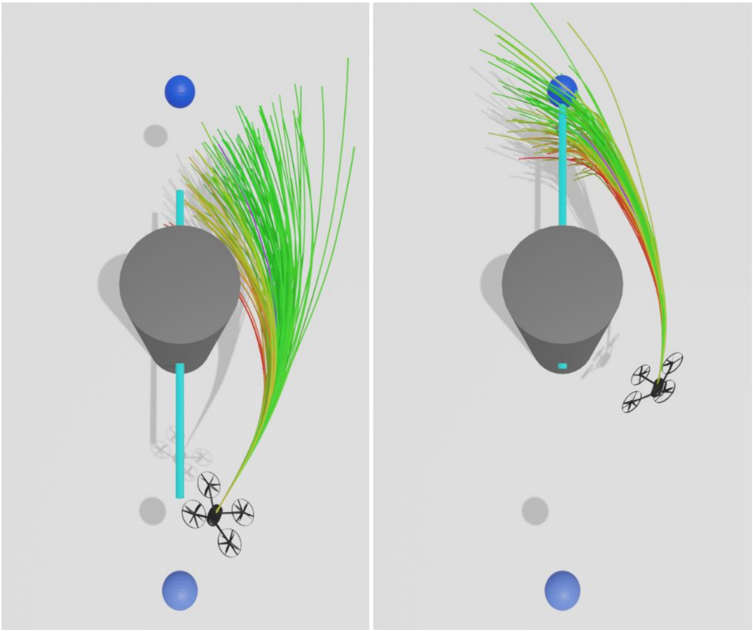
Mrunal Sarvaiya

Typical Robotics Systems



Model Predictive Control

- A type of **model** based controller
- **Optimization** or **sampling** based method
- Exploits the **dynamics** model
- Why select MPC over a learned controller?
  - Use known model
  - Interpretable weights
  - Reduce out of distribution issues



Optimization Based MPC

horizon length

$$\min_{\mathbf{x}(k),\mathbf{u}(k)} \sum_{k=0}^{N-1} L(\mathbf{x}(k), \mathbf{u}(k), m(k))$$

Cost Function

subject to $\mathbf{x}(k+1) = F(\mathbf{x}(k), \mathbf{u}(k), m(k))$

Dynamics

$$\mathbf{x}(0) = \mathbf{x}_0,$$

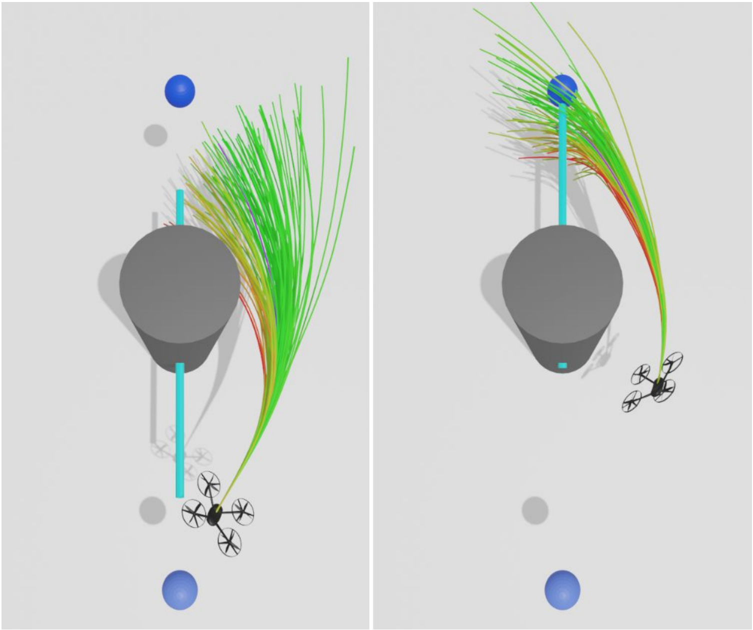
Initial Condition

$$H_x(\mathbf{x}(k)) \leq 0, H_u(\mathbf{u}(k)) \leq 0$$

State and Input inequality constraints

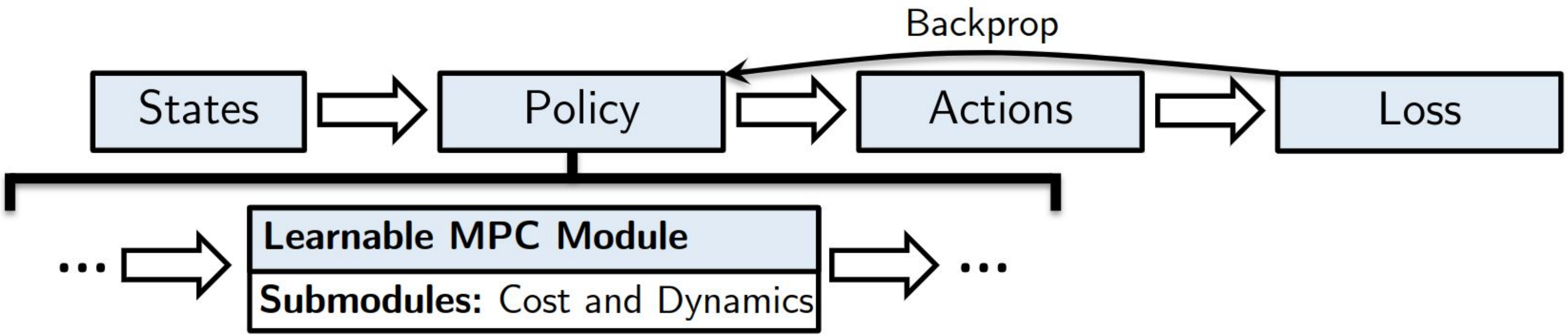
Tuning is Difficult

- Task specific cost tuning
- Tuning soft constraint weights
- Needs to be done manually, since MPC is a **black box** to learning algorithms



Differential MPC

- Backprop through MPC
- Think of it as a NN policy with a strong inductive bias that stems from control theory
- Learnable weights
- Eg. Loss = Tracking error



Differential MPC

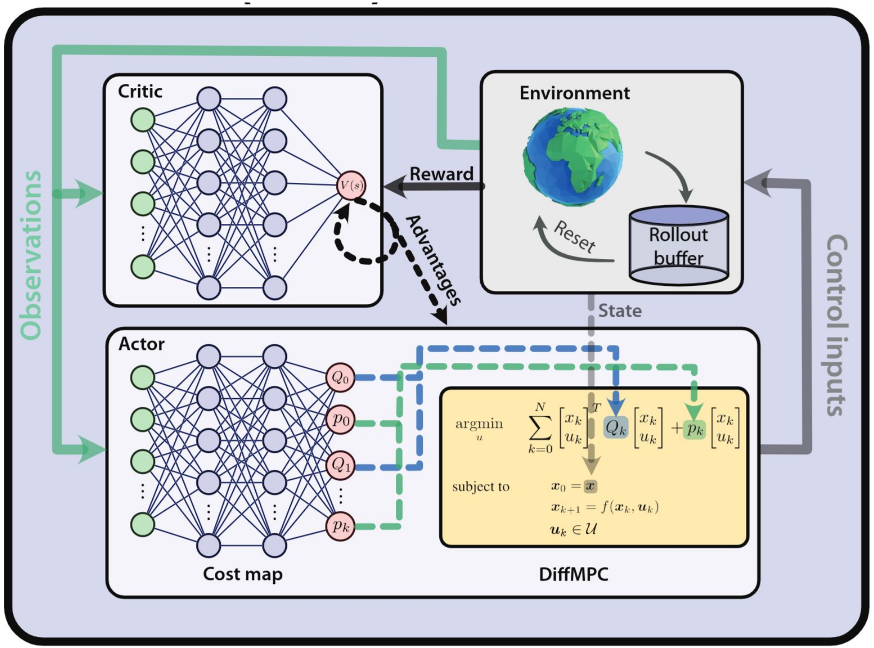
- Learn an interpretable controller
- Weights can be task dependent
- **Select what is learnable**
    - Convenient way to add inductive biases

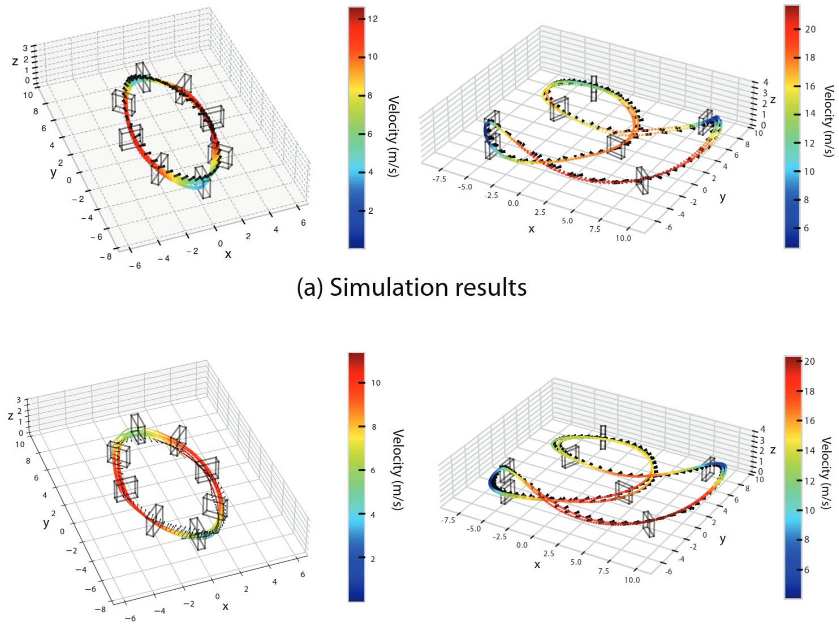
$$L = x_N^T \boxed{Q_N} x_N^T + \sum_{k=0}^{N-1} x_k^T Q_k x_k + u_k^T R_k u_k$$

$$L = x_N^T Q_N x_N^T + \sum_{k=0}^{N-1} x_k^T \boxed{Q_k} x_k + u_k^T \boxed{R_k} u_k$$

$$L = x_N^T \boxed{Q_N} x_N^T + \sum_{k=0}^{N-1} x_k^T \boxed{Q_k} x_k + u_k^T \boxed{R_k} u_k$$

Integrate with RL

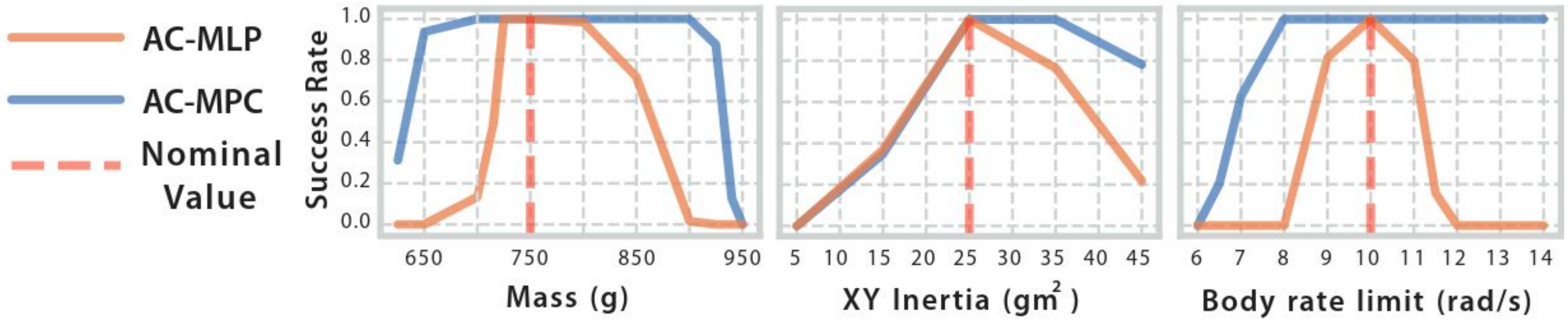




(a) Simulation results

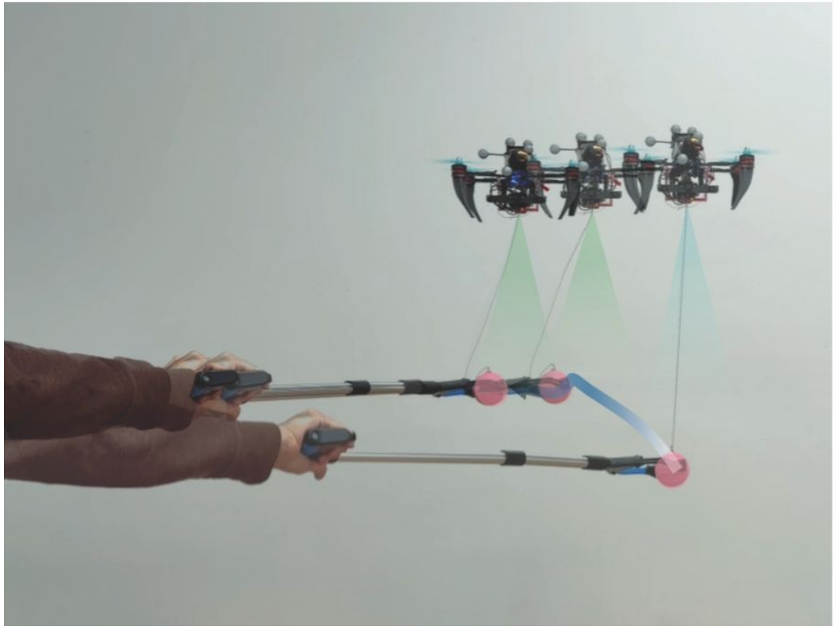
(b) Real-world results

Robustness to Dynamics Mismatch



Differential QP

- Backprop through a quadratic program
- Add differentiable constraints to a policy
- Eg. Field of view constraints for aerial transportation



Differential MPC

Pros

- Differentiable version of a widely used controller
- Not a black box anymore
- MPC can utilize arbitrary sensor inputs

Cons

- Does not natively support input constraints
- Backprop is slow
- Optimization based control or sampling based control?

Thank you!

Questions?

NYU

MP3 - A Unified Model to Map, Perceive, Predict and Plan

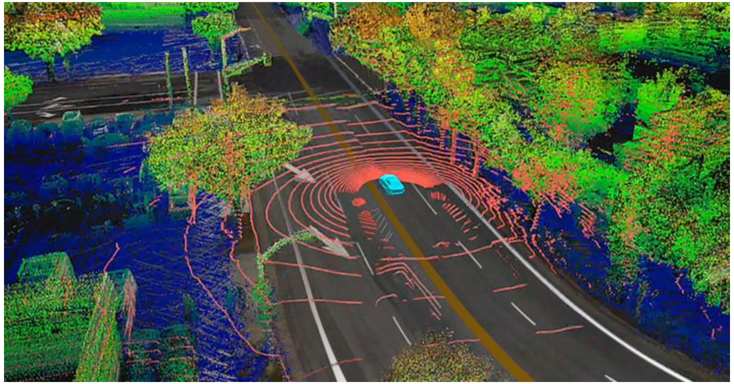
PRESENTED BY SUSHMA MAREDDY

Why HD Maps ?

01

NYU

What are HD maps for autonomous driving?



- Traditional GPS lacks the precision and dynamic data needed for driverless cars.
- HD maps enable high-precision localization by mapping a vehicle's exact position in relation to its environment.
- They combine real-time data from sensors, LiDAR, cameras, satellite imagery, and GPS.

NYU

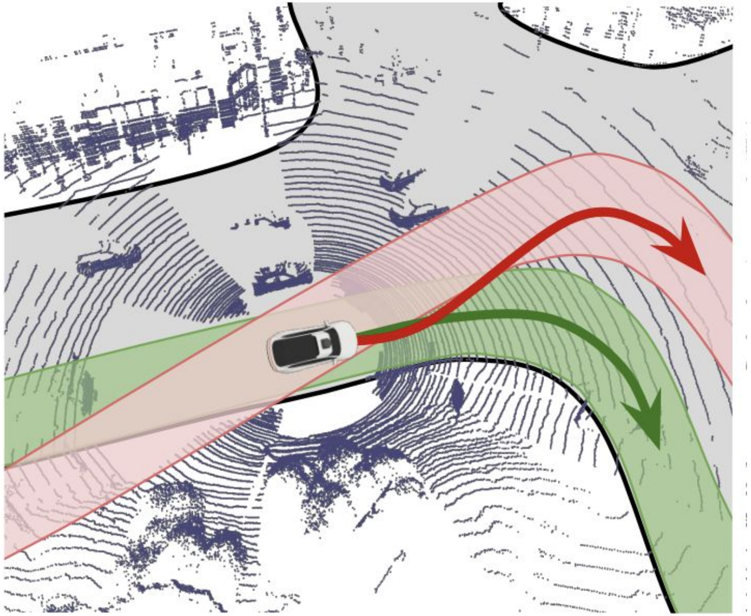
Why Autonomous Vehicles Need HD Maps ?

- HD maps are essential for lane-level navigation and safe autonomy.
- Unlike humans, self-driving cars can't adapt to inaccurate maps—even with sensors and cameras.
- HD maps offer inch-perfect detail: lanes, crosswalks, traffic signals, and barriers.
- Onboard sensors alone aren't enough for complex driving tasks.
- Example: Early Tesla FSD struggles show HD maps are a necessity, not a luxury.

Challenges with High-Definition Maps

- **High Cost and Maintenance:** Creating and updating HD maps is expensive and time-consuming, making it difficult to scale self-driving solutions globally.
- **Localization Errors:** Even small localization errors can lead to unsafe situations, such as driving off the road or into oncoming traffic.

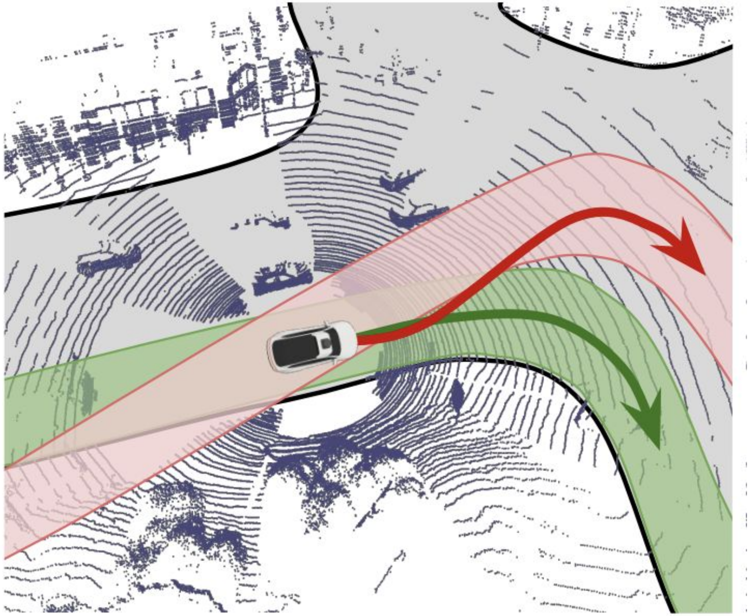
Driving with an HD map



Challenges with High-Definition Maps

- **Lack of Real-Time Information:** HD maps do not reflect real-time changes like road closures or construction, which can cause confusion and hazards.
- **Dependence on Vendors:** Relying on HD maps creates a dependency on map vendors, limiting flexibility and adaptability

Driving with an HD map



Motivation for Mapless Driving

02

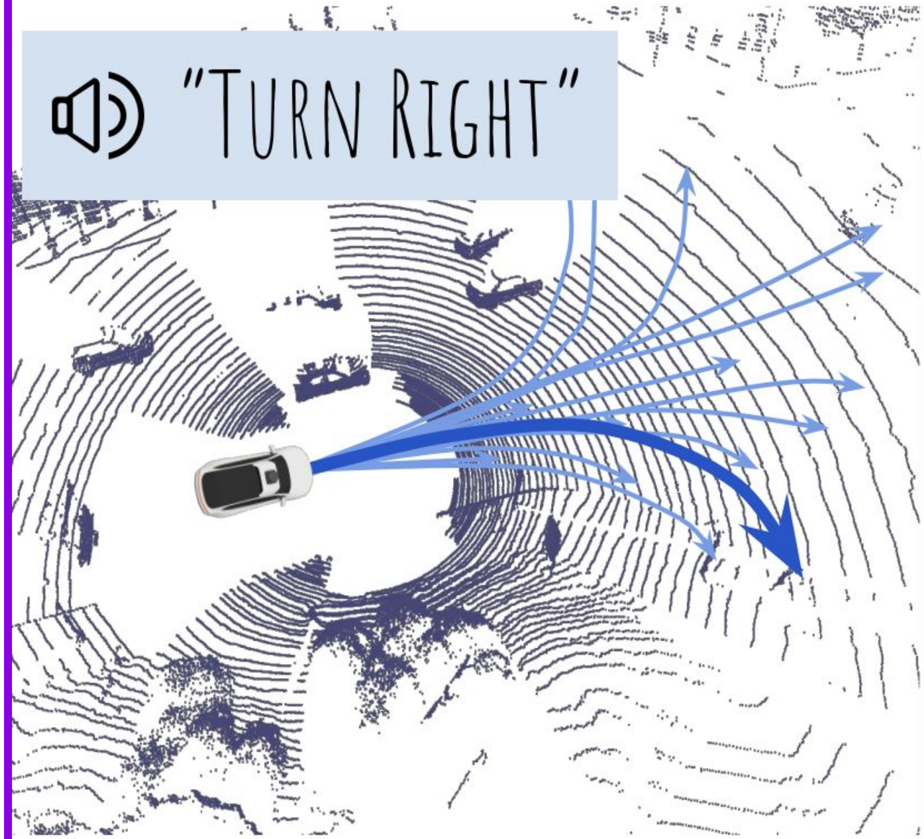
NYU

Why Mapless Driving?

- Limitations of HD maps: Costly, hard to scale, maintenance issues.
- Safety risks with HD map failures (e.g., localization errors).
- Advantages of mapless driving:
  - Lower cost.
  - Greater robustness to localization errors.
  - Scalable to diverse environments.

NYU

Mapless driving



Mapless driving can interpret the scene from sensors and achieve a safe plan that follows a high-level command

Key Objectives of MP3

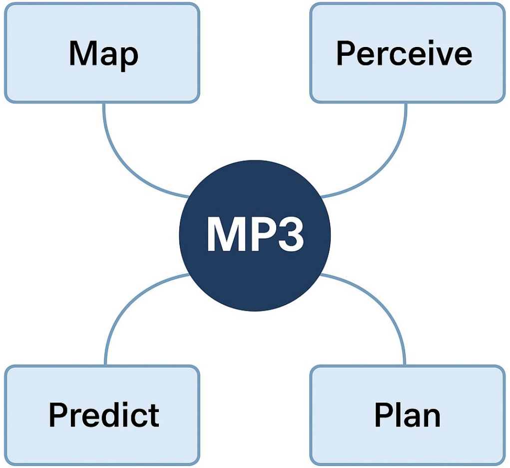
03

NYU

MP3: Key Objectives

Create an end-to-end mapless driving system that:

1. **Maps** - Generates an online map from raw sensor data.
2. **Perceives** - Understands dynamic and static environments.
3. **Predicts** - Anticipates the motion of surrounding objects.
4. **Plans** - Generates safe and comfortable driving trajectories.



Main Goal: Achieve **safe and robust autonomous driving** without HD maps.

NYU

Related Work

04

NYU

Online Mapping

What others did:

- Many systems use offline HD maps, built using satellite images or special mapping vehicles. These take time, cost money, and aren't usable for on-the-fly driving.
- Some newer methods try to predict road layouts in real time from images or LiDAR.
- But they're often focused only on highways (which are easier than city streets).
- And they make discrete decisions (like saying "this is a lane" or not) — which can be risky if something is missed.
- They also lose uncertainty (e.g., how confident are we that this is a lane?), which matters for safety.

NYU

What MP3 does differently

Online Mapping

MP3 uses a **dense, continuous map representation** — it keeps **all the information** and **understands uncertainty**, making it much safer for planning.

Perception and Prediction

What others did:

- Traditional systems detect individual objects (cars, pedestrians) and try to predict their future movements.
  - Some predict a few possible future paths, or
  - Use occupancy maps (grids that show what space might be occupied).

- But many rely on confidence thresholds — if the model isn't confident, it might miss detecting a real object entirely.

- Some methods predict what's in a scene, but not how things will move (no motion prediction).

- Others try to predict motion fields, but can’t handle multi-modal behavior — for example, a pedestrian might go left or right, and we need to account for both.

What MP3 does differently

Online Mapping

MP3 uses a **dense, continuous map representation** — it keeps **all the information** and **understands uncertainty**, making it much safer for planning.

Perception and Prediction

MP3 predicts a new kind of **"occupancy flow"**, which:

- Shows where things are,
- Predicts **how they will move** over time,
- And can **handle multiple possibilities** — all in a **scene-level** (not just per-object) way.

Motion Planning

What others did:

- Some early models just took sensor data and directly output driving commands (steer, brake). These can be unstable and hard to debug.
- Newer ones use cost maps to decide which paths are safest or best.
  - But many of them still rely on HD maps or hand-designed rules.
  - Others try to learn cost maps from data, but don’t predict what’s in the scene (no perception), so they can be unsafe.

What MP3 does differently

Online Mapping

MP3 uses a **dense, continuous map representation** — it keeps **all the information** and **understands uncertainty**, making it much safer for planning.

Perception and Prediction

MP3 predicts a new kind of **"occupancy flow"**, which:

- Shows where things are,
- Predicts **how they will move** over time,
- And can **handle multiple possibilities** — all in a **scene-level** (not just per-object) way.

Motion Planning

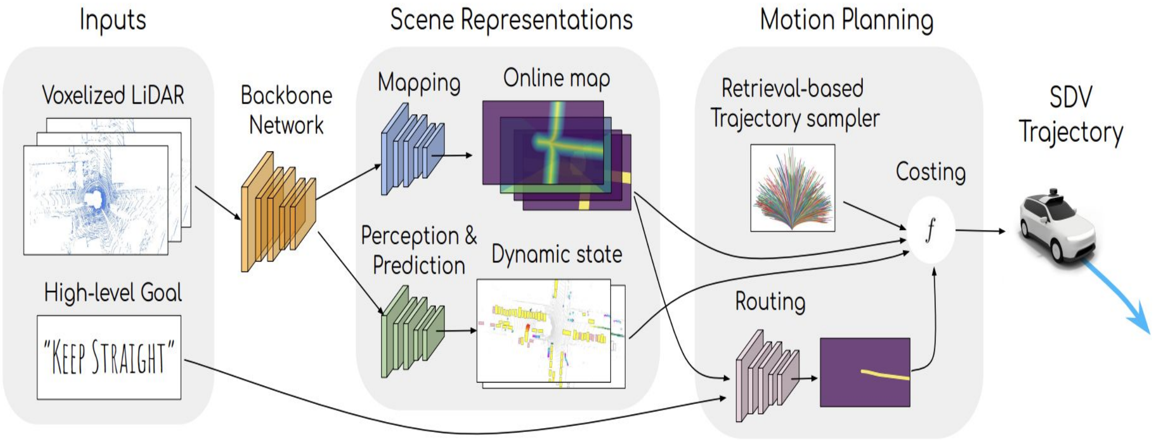
- **Retrieves examples** from expert human drivers to help it plan.
- Builds its own **route map** using sensor data and a **high-level command** like "keep straight".
- Uses its **own predicted occupancy flow** to make safe and explainable plans — all **without HD maps**.

MP3 System Architecture

05

NYU

MP3 System Architecture



- MP3 pipeline consists of:
  - Mapping and Perception
  - Prediction of Dynamic Occupancy
  - Motion Planning
- Uses LiDAR point clouds, camera data, and high-level driving commands.

Extracting Geometric and Semantic Features

06

NYU

Extracting Geometric and Semantic Features

Input Data:

- 10 LiDAR point clouds sweeps (1 second of history)
- Voxelized into Bird's Eye View (BEV)

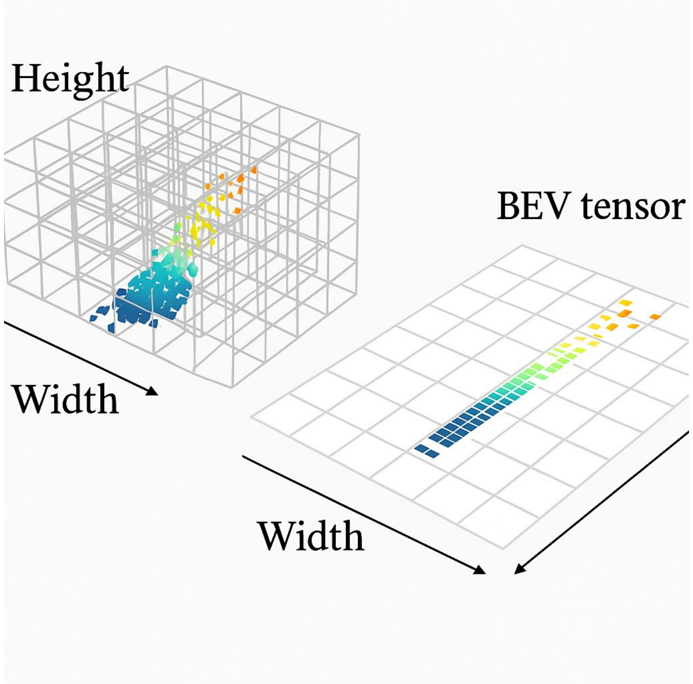
Spatial Specs:

- Resolution: 0.2 m/voxel
- Region of interest:
    - **Length**: 140m (70 front + 70 back)
    - **Width**: 80m (40 left + 40 right)
    - **Height**: 5m

Tensor Shape: 3D Tensor $\left(\frac{H}{a}, \frac{W}{a}, \frac{Z}{a} \cdot T_p\right)$ = (400,700,250)

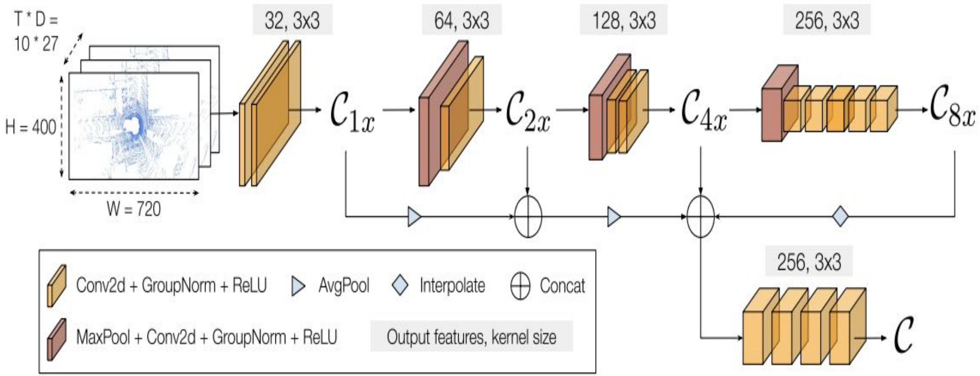
Preprocessing:

- Motion compensation using odometry
- Height + time concatenated in channels (no 3D conv needed)



NYU

Backbone Network for Scene Understanding



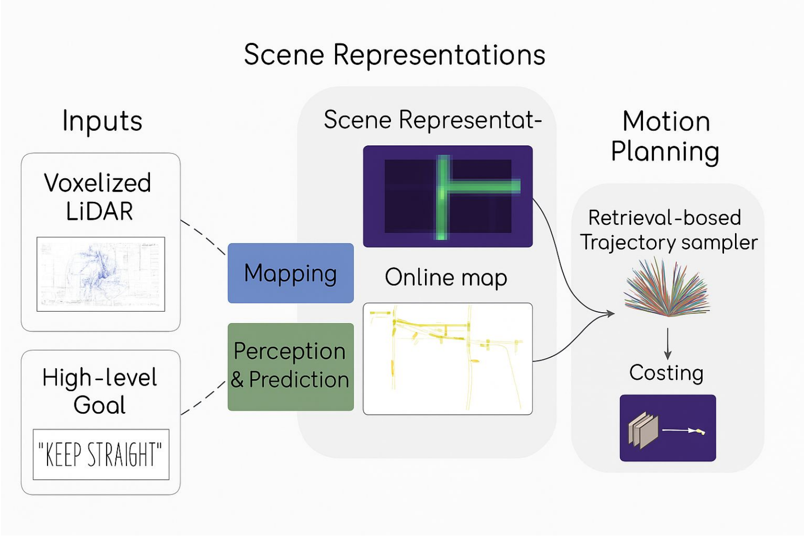
- The backbone network is crucial for extracting geometric and semantic features from sensor data.
- **Input:** Voxelized LiDAR point clouds in Bird's Eye View (BEV).
- **Processing:**
  - Uses multi-resolution convolutional blocks to extract features.
  - Downsampling layers to aggregate information from spatial and temporal domains.
- **Output:** Rich scene context features for mapping, perception, and prediction.

Interpretable Scene Representations

07

NYU

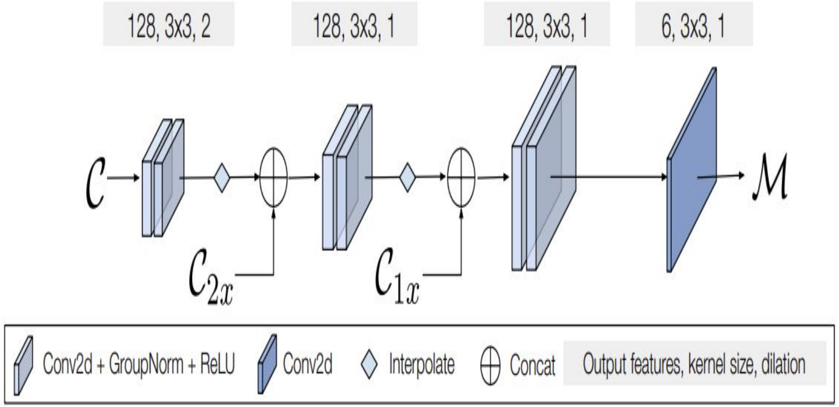
Interpretable Scene Representations



- MP3 replaces HD maps with **interpretable, probabilistic scene representations**.
- These representations:
  - Provide **semantic understanding** of the static world (road layout, intersections).
  - Predict **dynamic actor behavior** (position, velocity, intent).
  - Are interpretable and uncertainty-aware → suitable for **safe motion planning**.

NYU

Mapping Module



- **Input**: Multi-scale features (C1x, C2x, C) from backbone
- **Architecture**: Multi-resolution CNN combining fine details & global context
- **Techniques**: Max-pooling & interpolation to align feature scales
- **Output**: 6-channel probabilistic map:
  - Drivable area (Bernoulli)
  - Intersections (Bernoulli)
  - Lane distance (Laplacian: mean, std)
  - Lane direction (Von Mises: Von Mises: angle, concentration)

NYU

Online Map Representation

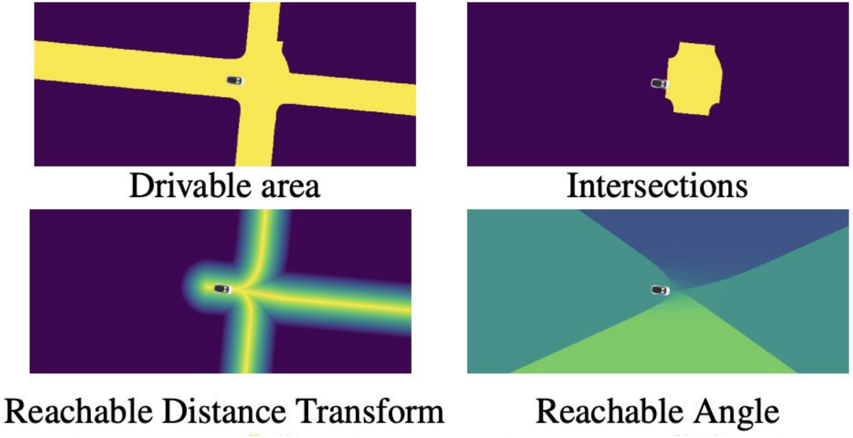
Drivable Area: Only where the SDV can legally go.

Reachable Lanes:

- Stay near centerline.
- Follow proper lane orientation.

Intersections:

- Must handle stop/yield/red-light conditions.



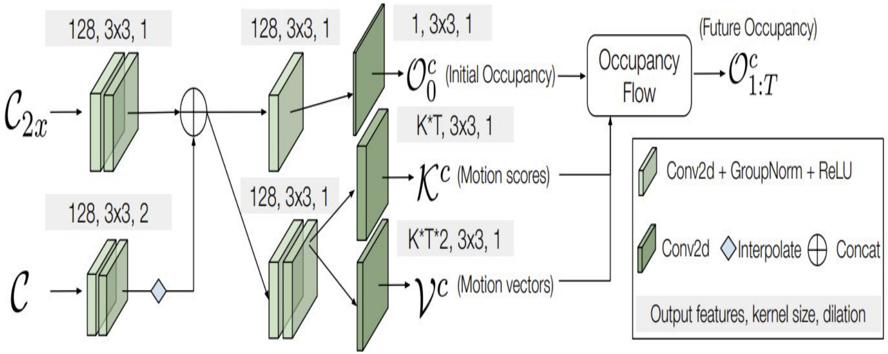
All predictions are **probabilistic**, so uncertainty can be reasoned over.

Perception and Prediction architecture

08

NYU

Perception and Prediction architecture:



- Separate network per dynamic object class:
  - Vehicles
  - Pedestrians
  - Bicyclists
- **Inputs:**
  - Context features from backbone.
- **Architecture:**
  - Separate CNNs per object class
  - Dilation for large receptive field with fewer parameters
  - Combines multi-scale features for richer motion cues
- **Output:**
  - Occupancy map (current positions)
  - Motion mode scores (multi-modal behavior)
  - Motion vectors (future movement)
- **Future prediction:** Uses motion fields to warp occupancy over time

NYU

Dynamic Occupancy Field

09

NYU

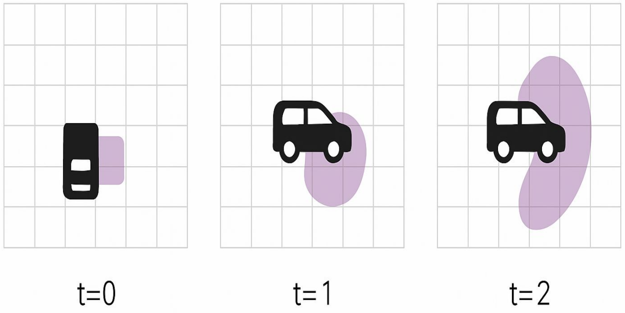
MP3's Approach: Occupancy Flow Field

MP3 predicts:

- Initial occupancy of dynamic objects at time t = 0.
- A temporal motion field (velocity vectors) that warps this occupancy across time.

Key Components:

- Resolution: 0.4 meters/pixel (BEV grid)
- Prediction Horizon: 11 steps → 5 seconds (0.5s each)
- Each occupied pixel gets:
  - A 2D velocity vector (vx, vy)
  - One for each motion mode
- Separate predictions for:
  - Vehicles
  - Pedestrians
  - Cyclists



Probabilistic Modeling

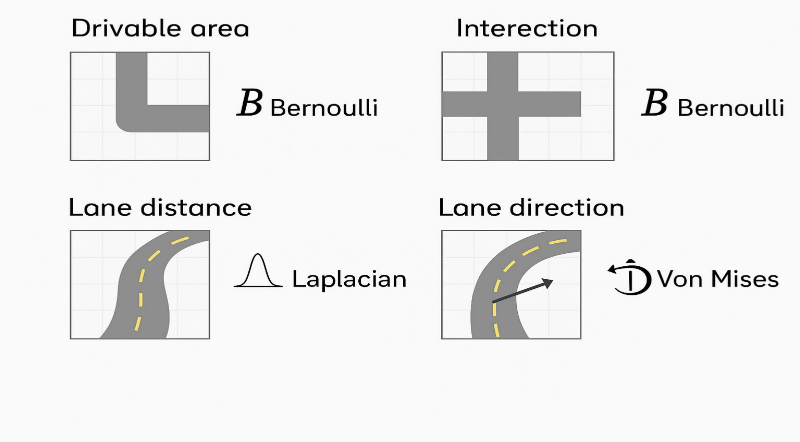
10

NYU

Probabilistic Modeling : Static Map

Goal: Model uncertainty in static scene layers.

- Each BEV cell is an **independent variable**
- Distributions per channel:
    - Drivable area: Bernoulli
    - Intersection: Bernoulli
    - Lane distance: Laplacian
    - Lane direction: Von Mises

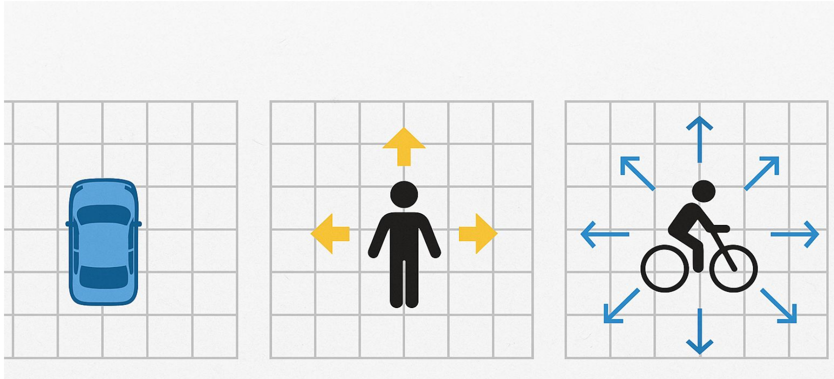


NYU

Probabilistic Modeling – Dynamic Field

Each class (vehicle, pedestrian, cyclist) modeled separately:

- **Occupancy ($Oc□_{,i}$)**: Bernoulli
- **Motion Mode ($Kc□_{,i}$)**: Categorical over K modes
- **Motion Vectors ($Vc□_{,i,}□$)**: 2D velocity for each mode



Why this is useful:

- Multi-modal behavior → car may turn OR go straight
- Allows capturing uncertainty in motion prediction

Occupancy Flow: Modeling Movement Over Time

Flow Event Definition (from cell i_1 to i_2):

$$F_{(t,i_1)\to(t+1,i_2)} = \bigcup_k \{O_{t,i_1} \wedge K_{t,i_1} = k \wedge V_{t,i_1,k} = i_2\}$$

Flow Probability:

$$p\left(F^c_{(t,i_1)\to(t+1,i_2)}\right) = \sum_k p\left(O^c_{t,i_1}\right) \cdot p\left(K^c_{t,i_1} = k\right) \cdot p\left(V^c_{t,i_1,k} = i_2\right)$$

Motion vector projection:

- Uses bilinear interpolation for smooth mass distribution

Future Occupancy Estimation

At each time step, update occupancy with:

$$p\left(O_{t+1,i}^{c}\right) = 1 - \prod_{j}\left(1 - p\left(F_{(t,j)\to(t+1,i)}^{c}\right)\right)$$

- **Interpretation**:
  A cell is **occupied at t+1** if **anything flows into it** from t.

- Efficient & consistent computation over time

- Enables safety-aware planning with dynamic agents

Why Probabilistic Occupancy Flow Is Powerful

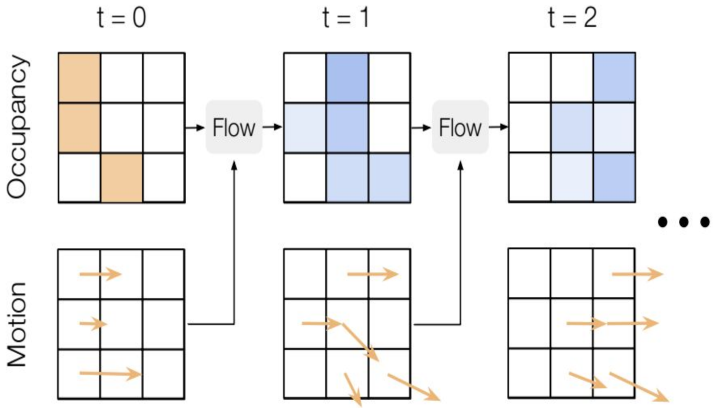


Figure 4: The motion field warps the occupancy over time. Transparency denotes probability. Color differences the predicted layers by the network and the future occupancy. We depict the particular case of unimodal motion $(K = 1)$.

- Replaces fragile object detection + forecasting pipeline
- Captures **multi-modal**, uncertain agent behavior
- Models **interactions** (e.g., car-yielding to pedestrians)
- Fully differentiable and interpretable
- Enables **risk-aware**, **goal-directed planning**

NYU

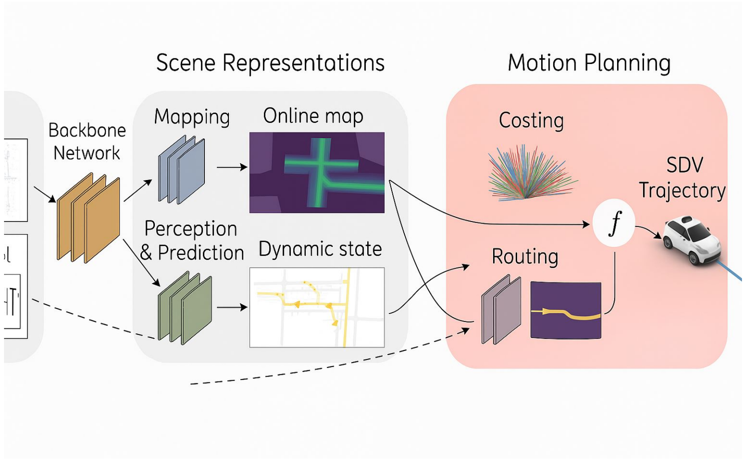
Motion Planning

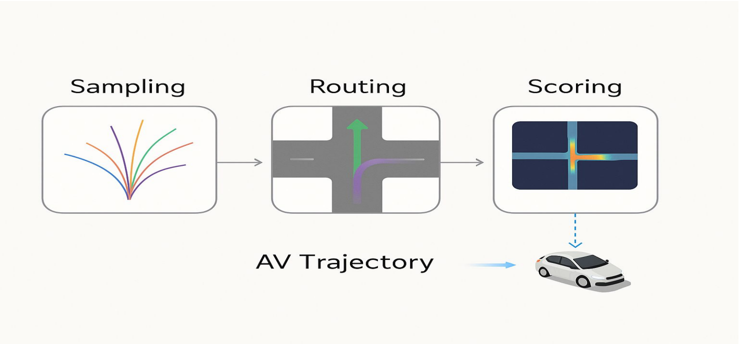
11

Goal :

- Generate trajectories that are:
  a. **Safe** (avoid collisions)
  b. **Comfortable** (low jerk, smooth)
  c. **Goal-directed** (follow command)
- Approach:
  a. Sample kinematically feasible trajectories
  b. Score them using learned cost function
  c. Select best trajectory:

$$\tau^* = \arg\min_{\tau \in T(x_0)} f(\tau, M, O, K, V; w)$$



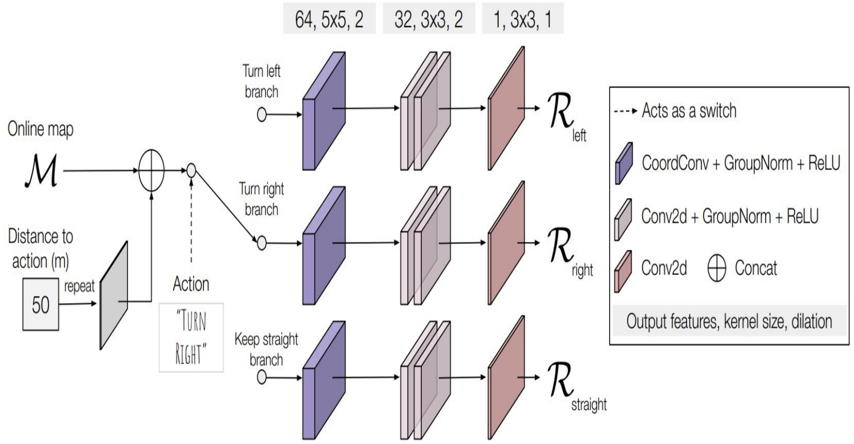


Trajectory Sampling

- **150+ hours of driving logs** → clustered into **3000 trajectory prototypes**
- **Binned** by (velocity, acceleration, curvature)
- Retrieval is **based on SDV initial state**
- Trajectories re-rolled out using **(a, κ)** and **bicycle model** (for smoothness)

NYU

Routing Command & Prediction



- **Driving command:**
  - a ∈ {keep lane, turn left, turn right}
  - d: estimated distance to action
- **Routing Network:**
  - 3 CNN branches (one per action)
  - Input: predicted map + repeated distance (CoordConv)
  - Output: dense route probability map

Trajectory Scoring Functions – Routing

- Encourage SDV to stay on predicted route
- Add "cost-to-go" for goal beyond horizon
- Penalize off-route maneuvers

Trajectory Scoring – Map Constraints

Map Alignment Cost Terms

- Stay near lane center: use M_D (distance map)
- Align to lane direction: use M_θ
- Penalize uncertainty: $f_d(x, M_\theta, M_D) = \sum_i v_x(\sigma_i^D + \frac{1}{k_i^\theta})$
- Stay on road: use M_A → penalize off-road cells

Trajectory Scoring – Safety

Safety & Collision Cost

- Avoid dynamic objects using predicted occupancy O: $f_o(x_t, O) = \sum_c \max_{i \in m(x_t)} P(O^c_{t,i})$
- Maintain **safe headway**: $f_h(x_t, O, K, V) = \sum_i P(O_{t,i}) \cdot \mathbb{E}_{K_{t,i}}[h(x_t, V_{t,i})]$
- Considers relative velocity & stopping distance

NYU

Trajectory Scoring – Comfort

- Penalize **jerk** (sudden changes in acceleration)
- Penalize **lateral acceleration** (unpleasant sideways motion during turns)
- Encourage smooth, human-like driving

Learning Strategy

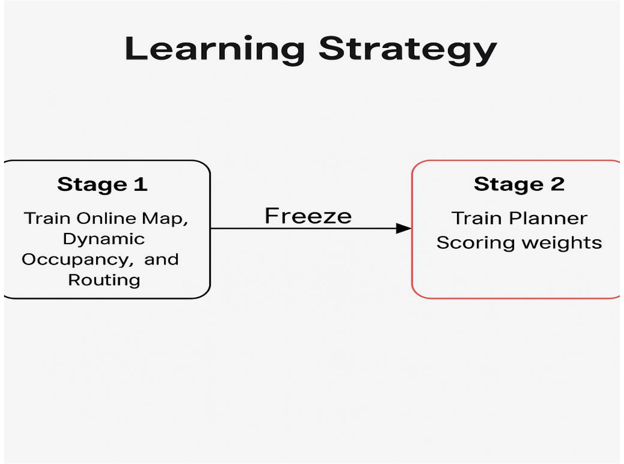
12

NYU

Two-Stage Training Overview

Stage 1 (Multi-task Learning): Train Online Map, Dynamic Occupancy, and Routing.

Stage 2: Freeze above, train Planner Scoring weights.



Stage 1 - Multi-task Learning

Combined loss: To do so, we linearly combine the mapping loss LM, occupancy loss LO, motion loss LK,V , and routing loss LR $$L = L_O + \lambda_{K,V} L_{K,V} + \lambda_M L_M + \lambda_R L_R$$

Hyperparameters:

$$\lambda_{K,V} = 0.1,\ \lambda_M = 0.5,\ \lambda_R = 2.0$$

Each component trained with proper distribution loss:

- Map: NLL (Bernoulli, Laplacian, Von Mises)
- Occupancy: Cross-entropy + Hard Negative Mining
- Motion Modes: Unsupervised cross-entropy
- Motion Vectors: Huber loss

Stage 2 - Max-Margin Trajectory Scoring

Goal: Penalize low-cost unsafe trajectories

Use **Max-Margin Loss**:

$$L_M = \max_{\tau} \left[f_r(\tau_h) - f_r(\tau) + l_{im} + \sum_t (f_o^t(\tau_h) - f_o^t(\tau) + l_o^t) \right]_+$$

Encourages expert-like, safe behavior.

NYU

Experimental Evaluation

13

NYU

Experimental Setup

Dataset: URBANEXPERT

- 5k train / 500 val / 1k test
- 25s per scenario
- 33x more driving data than KITTI
- Geographically non-overlapping splits

Baselines:

- Imitation Learning (IL)
- Conditional Imitation Learning (CIL)
- Trajectory Classification(TC)
- Neural Motion Planner(NMP)
- Conditional Neural Motion Planner(CNMP)

Evaluation Criteria

- Planning performance
- Safety & comfort metrics
- Route-following accuracy
- Collision and off-road rates

Closed-Loop Evaluation

- Realistic LiDAR simulation with dynamic actors
- **164 curated scenarios**, 18s each
- Actors adapt reactively using **Intelligent Driver Model**
- Measures: Success rate, L2 to expert, Progress (m/event), Comfort

Model	Success (%)↑	OffRoute (%)↓	L2 (m)↓	Progress per event (m) ↑ any event	collision	off-road	off-route	oncoming	Comfort jerk$(\frac{m}{s^3})$↓	lat.acc. $(\frac{m}{s^2})$↓
IL	0.00	99.39	39.10	15.69	44.49	36.40	30.28	65.18	98.99	0.91
CIL	0.00	99.39	35.53	15.85	38.50	34.68	35.64	54.58	52.88	0.81
TC	12.80	67.07	30.35	51.17	127.87	288.07	105.26	329.90	3.15	0.25
NMP	22.56	64.02	27.95	69.83	331.81	721.74	104.70	1229.82	3.04	0.14
CNMP	21.34	47.56	27.45	74.85	158.85	646.49	198.28	543.32	2.96	0.26
MP3	**74.39**	**14.63**	**12.95**	**218.40**	**1037.08**	**1136.49**	**409.34**	**1465.27**	**1.64**	**0.10**

NYU

Open-Loop Evaluation

- Plans are made from expert's state → useful for analysis but less realistic
- IL and CIL show good imitation but poor safety
- MP3 still most robust and safest

Model	Collisions (%) 0-3s	Collisions (%) 0-5s	L2 (m) @3s	L2 (m) @5s	Progress(m) 0-5s	OffRoute(%) 0-5s	OffRoad(%) 0-5s	Oncoming(%) 0-5s	lat.acc.($\frac{m}{s^2}$) 0-5s	Jerk ($\frac{m}{s^3}$) 0-5s
IL	2.17	9.54	**1.36**	**3.77**	23.62	5.05	4.46	3.05	**1.00**	2.47
CIL	2.20	10.15	1.38	3.79	23.58	5.16	5.28	3.64	1.10	2.60
TC	1.72	6.95	2.02	4.34	22.26	2.68	0.28	0.62	1.47	7.48
NMP	0.83	5.18	1.75	4.47	23.09	1.59	**0.00**	0.21	1.14	3.98
CNMP	1.03	5.45	1.62	4.02	22.99	**0.14**	0.07	0.14	1.28	3.97
MP3	**0.21**	**2.07**	1.71	4.54	**25.15**	0.15	0.42	**0.09**	1.23	**1.88**

NYU

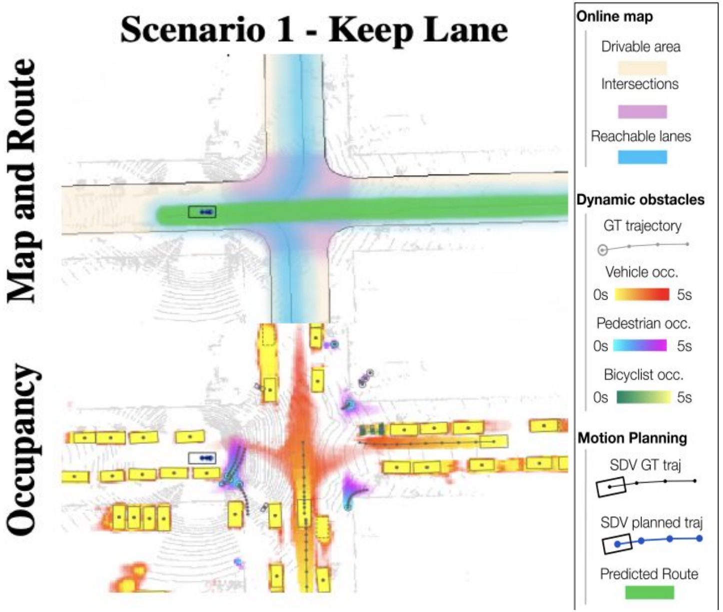
Qualitative Results

14

NYU

Scenario 1: Keep Lane

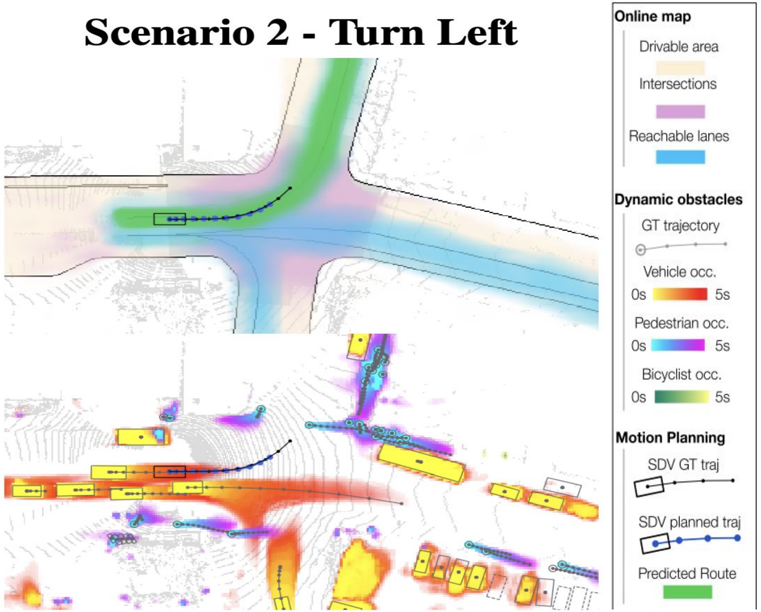
- SDV instructed to go straight at intersection
- Pedestrians emerge from occlusion
- **Model predicts multimodal pedestrian motion & stops safely**



NYU

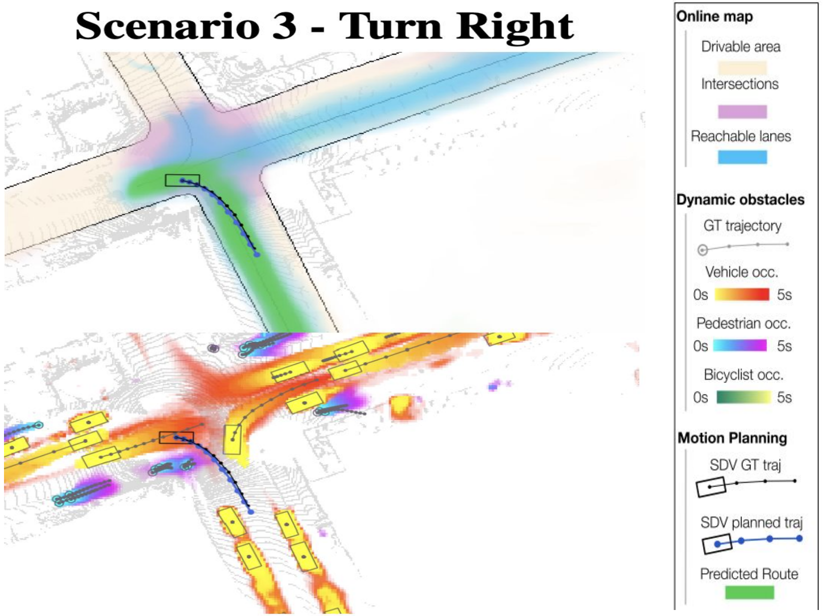
Scenario 2 - Turn Left

- Complex intersection with dense actors
- **Route prediction aligns with command**
- Planner progresses **smoothly through traffic**



Scenario 3 - Turn Right

- Route prediction successfully captures the turn
- SDV avoids surrounding dynamic agents
- **Follows expert-like trajectory with safe margin**



Conclusion

15

NYU

Conclusion

End-to-End Mapless Driving

- Directly processes raw LiDAR data
- No dependency on HD maps

Interpretable Probabilistic Representations

- Online map: drivable areas, lane structure, intersections
- Dynamic occupancy: future motion & uncertainty
- Used as cost functions in planning

Neural Motion Planner

- Sample-based trajectory generation
- Optimized for safety, comfort, and goal progress

Key Results

- +3× success rate over baselines
- Most comfortable & safest trajectories
- Robust generalization in closed-loop simulation without fine-tuning

THANK YOU !

NYU

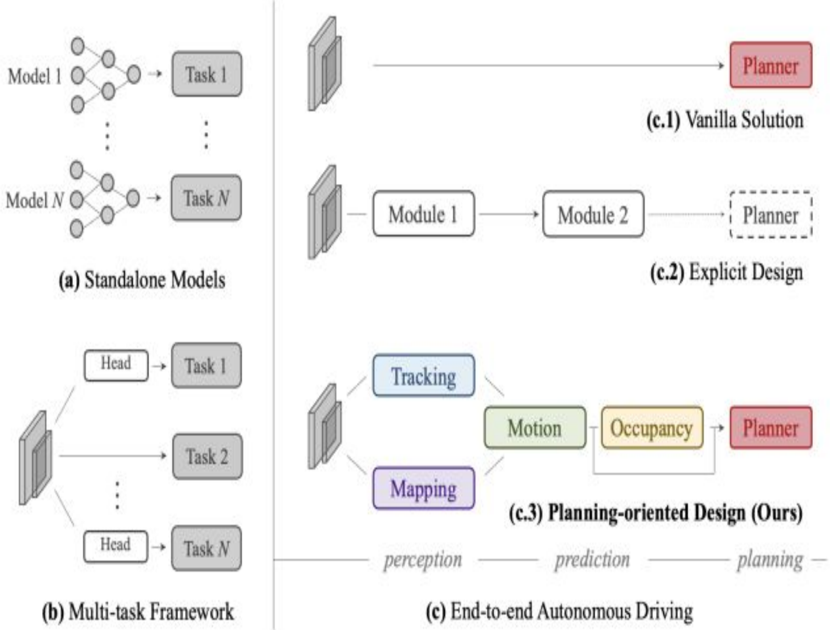
Questions?

Planning-oriented Autonomous Driving

Presented by Raman Jha
04/3/2025

Introduction

- **UniAD** is a unified framework for autonomous driving that **integrates perception, prediction, and planning** tasks into a single end-to-end system.
- Unlike traditional modular approaches, **UniAD adopts a planning-oriented philosophy**, ensuring that all preceding tasks contribute directly to safe and efficient driving decisions.
- The framework uses **query-based interfaces to connect modules**, enabling flexible feature sharing and robust task coordination



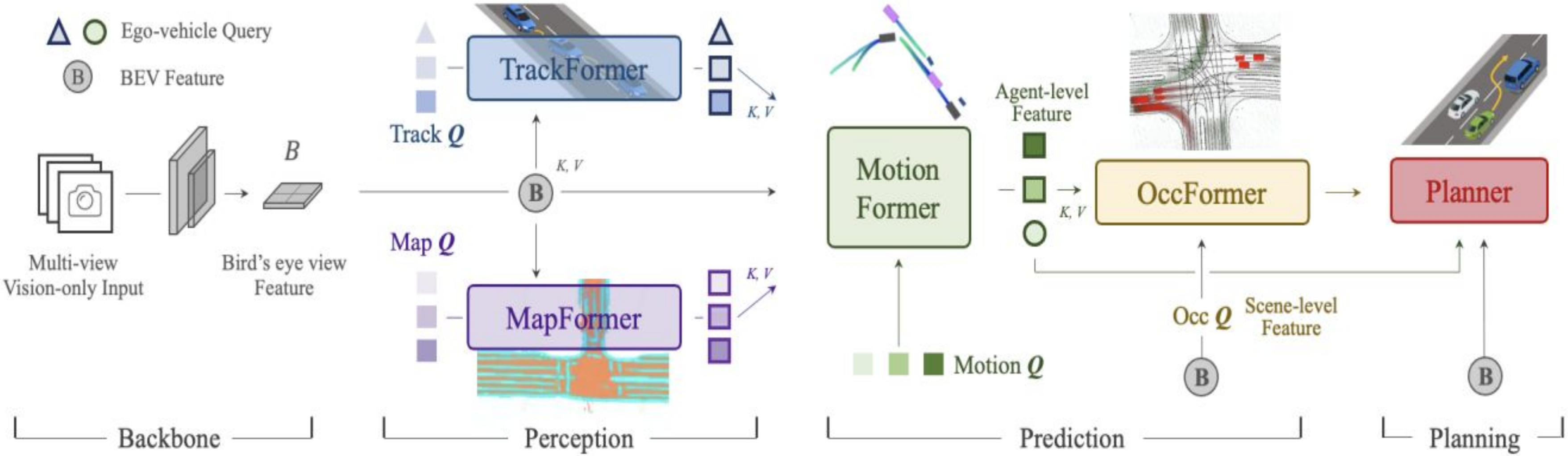
NYU

Background

- **Traditional autonomous driving systems** often rely on standalone models for individual tasks or multi-task learning paradigms with separate heads, which can lead to cascading errors and poor task coordination.
- **End-to-end approaches** have emerged to unify perception, prediction, and planning but often lack interpretability and robustness in dynamic urban environments.
- **UniAD addresses these challenges** by explicitly modeling intermediate representations (e.g., occupancy maps, agent trajectories) and optimizing the system for planning as the ultimate goal.

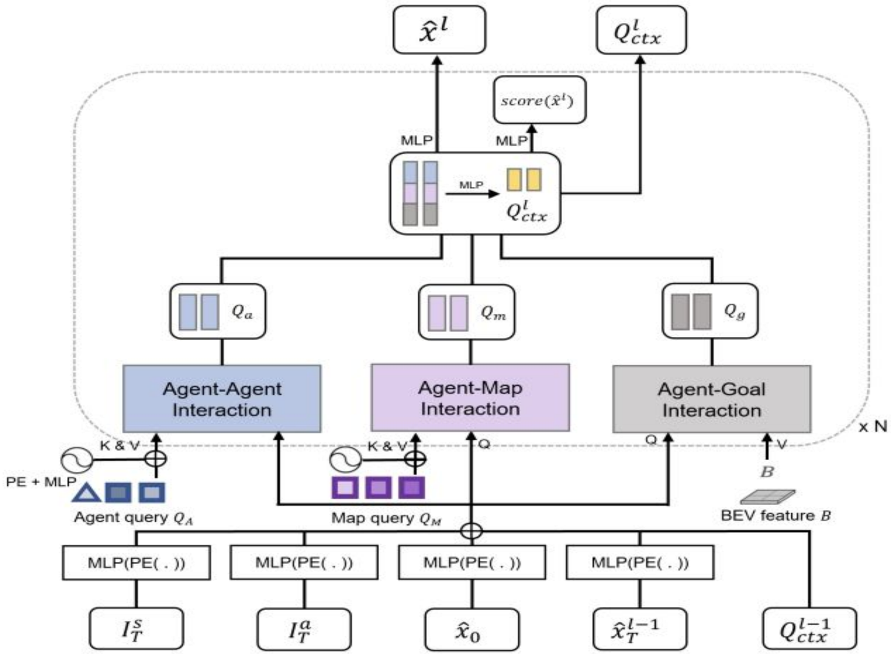
Design	Approach	Perception			Prediction		Plan
		Det.	Track	Map	Motion	Occ.	
(b)	NMP [101]	✓			✓		✓
	NEAT [19]			✓			✓
	BEVerse [105]	✓		✓		✓	
(c.1)	[14, 16, 78, 97]						✓
(c.2)	PnPNet† [57]	✓	✓		✓		
	ViP3D† [30]	✓	✓		✓		
	P3 [82]					✓	✓
	MP3 [11]			✓		✓	✓
	ST-P3 [38]			✓		✓	✓
	LAV [15]	✓		✓	✓		✓
(c.3)	**UniAD** (ours)	✓	✓	✓	✓	✓	✓

Model Architecture



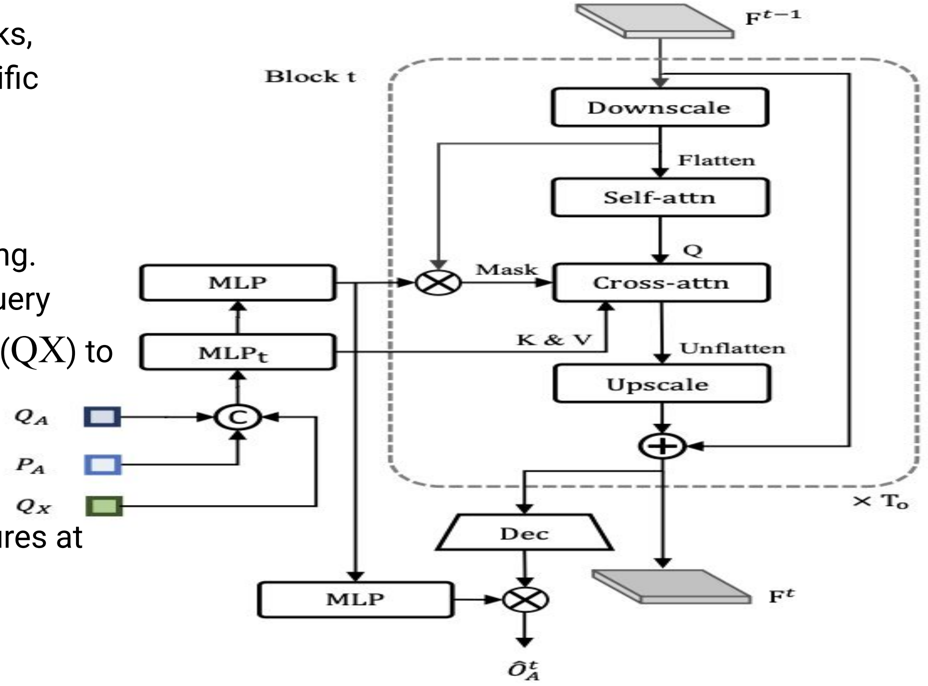
Motionformer

- **Structure**: MotionFormer consists of N stacked transformer layers for agent-agent, agent-map, and agent-goal interactions.
- **Modules**:
    - Agent-agent and agent-map interactions use standard transformer decoder layers.
    - Agent-goal interaction is based on the deformable cross-attention module.
- **Inputs:**
- ITs: Scene-level anchor endpoint.
- ITa: Clustered agent-level anchor endpoint.
- x^0o: Current position of the agent.
- x^Tl−1: Predicted goal point from the previous layer.
- Qctxl−1: Query context from the preceding layer.



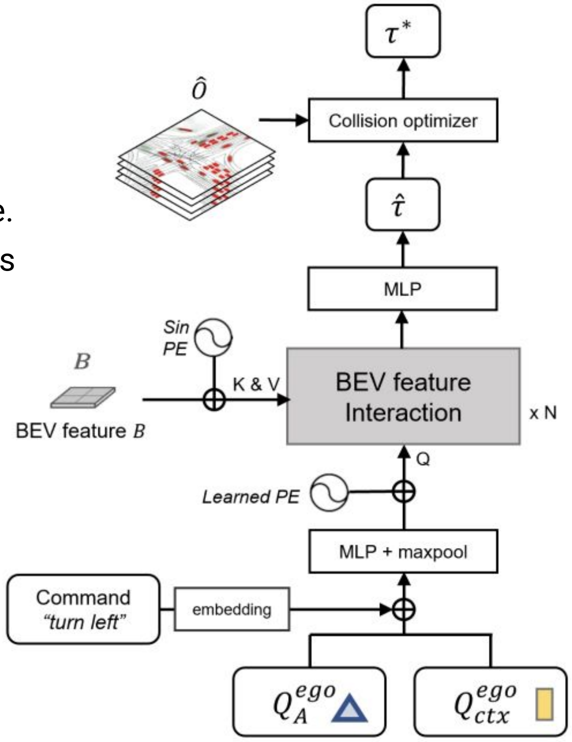
OccFormer

- **Structure**: OccFormer comprises To sequential blocks, where each block predicts the occupancy for a specific frame within the temporal horizon.
- **Features Incorporated:**
  - **Dense Scene Features:** Encoded from BEV representations for global scene understanding.
  - **Sparse Agent Features:** Derived from track query (QA), agent position (PA), and motion query (QX) to inject agent-level knowledge.
- **Instance-Level Occupancy:**
  - Generated via matrix multiplication between agent-level features and decoded dense features at the end of each block (O^At)



Planner

- **Inputs:**
  - QegoA: Ego-vehicle query from the tracking module.
  - Qegoctx: Ego-vehicle query from the motion forecasting module.
  - High-level command embeddings indicating navigation directions (e.g., turn left, go straight).
- **Processing:**
  - Queries are encoded via MLP layers and aggregated using max-pooling to select salient modal features.
  - BEV feature interaction is performed using stacked transformer decoder layers (N layers).
- **Output:**
  - Predicts future waypoints (τ^) for ego-vehicle planning while optimizing trajectories to avoid collisions based on predicted occupancy maps (O^).



Loss Function

$$L_1 = L_{\text{track}} + L_{\text{map}}.$$

$$L_2 = L_{\text{track}} + L_{\text{map}} + L_{\text{motion}} + L_{\text{occ}} + L_{\text{plan}}.$$

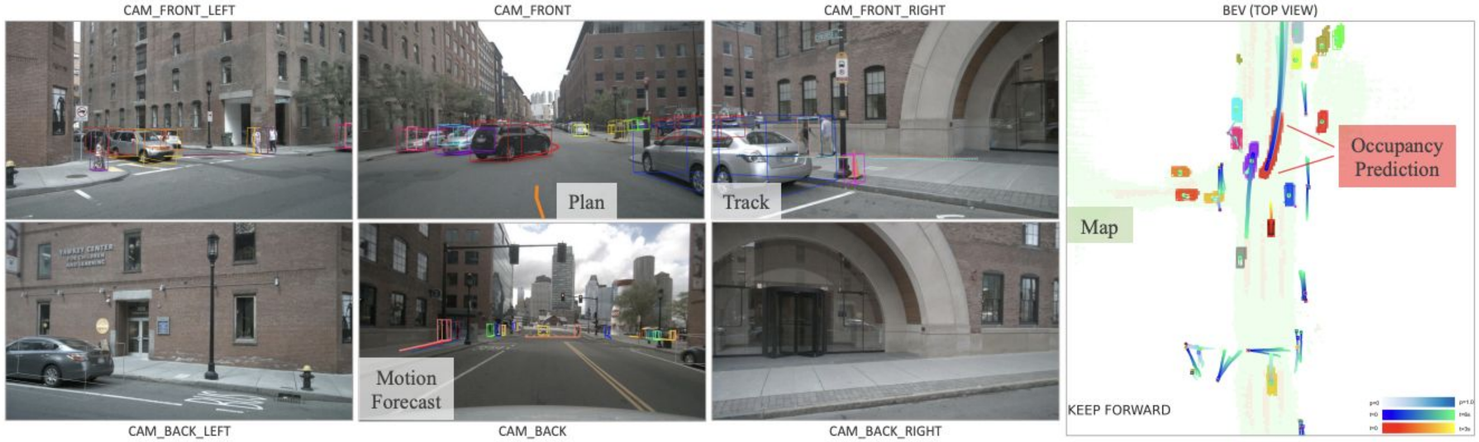
Stage One Loss Function

Combines tracking loss (Hungarian loss with Focal and L1 components) and mapping loss (Focal, L1, GIoU, and Dice losses) to pre-train perception tasks:

Stage Two Loss Function

Integrates all task-specific losses (tracking, mapping, motion forecasting, occupancy prediction, and planning) for end-to-end training

Qualitative Results



- **Task Results:** Predictions from motion and occupancy modules are consistent, visualized in surround-view images and BEV.
- **Ego-Vehicle Behavior:** Ego vehicle yields to a front black car, demonstrating safe decision-making.
- **Agent Representation:** Each agent is illustrated with a unique color for clarity.
- **Trajectory Visualization:**
  - Image View: Displays top-1 trajectory from motion forecasting.
  - BEV View: Shows top-3 trajectories for better spatial understanding.

NYU

Quantitative Results

Method	AMOTA↑	AMOTP↓	Recall↑	IDS↓
Immortal Tracker† [93]	0.378	1.119	0.478	936
ViP3D [30]	0.217	1.625	0.363	-
QD3DT [36]	0.242	1.518	0.399	-
MUTR3D [104]	0.294	1.498	0.427	3822
UniAD	**0.359**	**1.320**	**0.467**	**906**

Multi-object tracking

Method	Lanes↑	Drivable↑	Divider↑	Crossing↑
VPN [72]	18.0	76.0	-	-
LSS [76]	18.3	73.9	-	-
BEVFormer [55]	23.9	**77.5**	-	-
BEVerse† [105]	-	-	**30.6**	**17.2**
UniAD	**31.3**	69.1	25.7	13.8

Online mapping

- **UniAD Performance:** Outperforms previous end-to-end MOT techniques with image inputs on all metrics.
- **Comparison Note:** Tracking-by-detection methods with post-association are implemented using BEVFormer for fair evaluation.

- **Performance:** UniAD achieves competitive results against state-of-the-art perception-oriented methods with comprehensive road semantics.
- **Segmentation Metric:** Reports segmentation IoU (%) for lanes, drivable areas, dividers, and crossings.
- **Comparison Note:** Methods are implemented with BEVFormer for fair evaluation

Quantitative Results

Method	minADE(m)↓	minFDE(m)↓	MR↓	EPA↑
PnPNet† [57]	1.15	1.95	0.226	0.222
ViP3D [30]	2.05	2.84	0.246	0.226
Constant Pos.	5.80	10.27	0.347	-
Constant Vel.	2.13	4.01	0.318	-
UniAD	**0.71**	**1.02**	**0.151**	**0.456**

Motion forecasting.

Method	IoU-n.↑	IoU-f.↑	VPQ-n.↑	VPQ-f.↑
FIERY [35]	59.4	36.7	50.2	29.9
StretchBEV [1]	55.5	37.1	46.0	29.0
ST-P3 [38]	-	38.9	-	32.1
BEVerse† [105]	61.4	**40.9**	54.3	**36.1**
UniAD	**63.4**	40.2	**54.7**	33.5

Occupancy prediction

- **Performance:** UniAD significantly outperforms prior vision-based end-to-end methods across all metrics.
- **Comparative Settings:** Evaluated with two vehicle modeling settings—constant positions and constant velocities.
- **Reimplementation:** Prior methods reimplemented with BEVFormer for fair comparisons.

- **Improvement in Nearby Areas:** UniAD achieves significant gains in near evaluation ranges (30×30m), critical for planning accuracy.
- **Evaluation Ranges:** Results are reported for "n." (near) and "f." (far, 50×50m) evaluation ranges.
- **Training Note:** Models trained with heavy augmentations yield improved occupancy prediction metrics.

Quantitative Results

Method	L2(m)↓ 1s	2s	3s	Avg.	Col. Rate(%)↓ 1s	2s	3s	Avg.
NMP† [101]	-	-	2.31	-	-	-	1.92	-
SA-NMP† [101]	-	-	2.05	-	-	-	1.59	-
FF† [37]	0.55	1.20	2.54	1.43	0.06	0.17	1.07	0.43
EO† [47]	0.67	1.36	2.78	1.60	0.04	0.09	0.88	0.33
ST-P3 [38]	1.33	2.11	2.90	2.11	0.23	0.62	1.27	0.71
UniAD	**0.48**	**0.96**	**1.65**	**1.03**	**0.05**	**0.17**	**0.71**	**0.31**

Planning

- **Performance:** UniAD achieves the lowest L2 error and collision rate across all time intervals.
- **Comparison:** Outperforms LiDAR-based methods in most cases, demonstrating superior safety.
- **Validation:** Results verify the effectiveness of integrating motion and occupancy prediction for safe planning.

NYU

Ablation Study

ID	Scene-l. Anch.	Goal Inter.	Ego Q	NLO.	minADE↓	minFDE↓	MR↓	minFDE-mAP*↑
1					0.844	1.336	0.177	0.246
2	✓				0.768	1.159	0.164	0.267
3	✓	✓			0.755	1.130	0.168	0.264
4	✓	✓	✓		0.747	1.096	0.156	0.266
5	✓	✓	✓	✓	**0.710**	**1.004**	**0.146**	**0.273**

Ablation for designs in the **motion forecasting module**

ID	Cross. Attn.	Attn. Mask	Mask Feat.	IoU-n.↑	IoU-f.↑	VPQ-n.↑	VPQ-f.↑
1				61.2	**39.7**	51.5	31.8
2	✓			61.3	39.4	51.0	31.8
3	✓	✓		62.3	**39.7**	52.4	32.5
4	✓	✓	✓	**62.6**	39.5	**53.2**	**32.8**

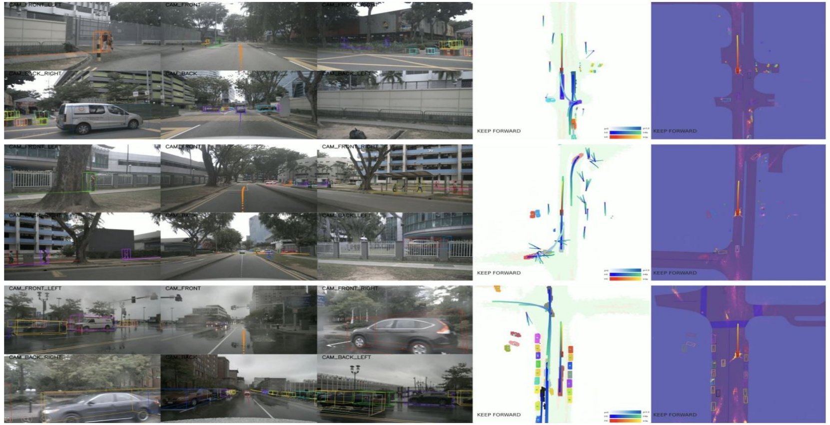
Ablation for designs in the **occupancy prediction module**

ID	BEV Att.	Col. Loss	Occ. Optim.	L2↓ 1s	L2↓ 2s	L2↓ 3s	Col. Rate↓ 1s	Col. Rate↓ 2s	Col. Rate↓ 3s
1				**0.44**	**0.99**	**1.71**	0.56	0.88	1.64
2	✓			**0.44**	1.04	1.81	0.35	0.71	1.58
3	✓	✓		**0.44**	1.02	1.76	0.30	0.51	1.39
4	✓	✓	✓	0.54	1.09	1.81	**0.13**	**0.42**	**1.05**

Ablation for designs in the **planning module**

Strengths

- **UniAD** integrates **perception, prediction, and planning** into a unified end-to-end framework for enhanced coordination.
- **Query-based design enables flexible feature sharing** across tasks, improving accuracy and task interaction.
- Achieved **state-of-the-art performance** in motion forecasting, occupancy prediction, and safe planning metrics.
- **Reduces cascading errors and enhances interpretability** through explicit intermediate representations.



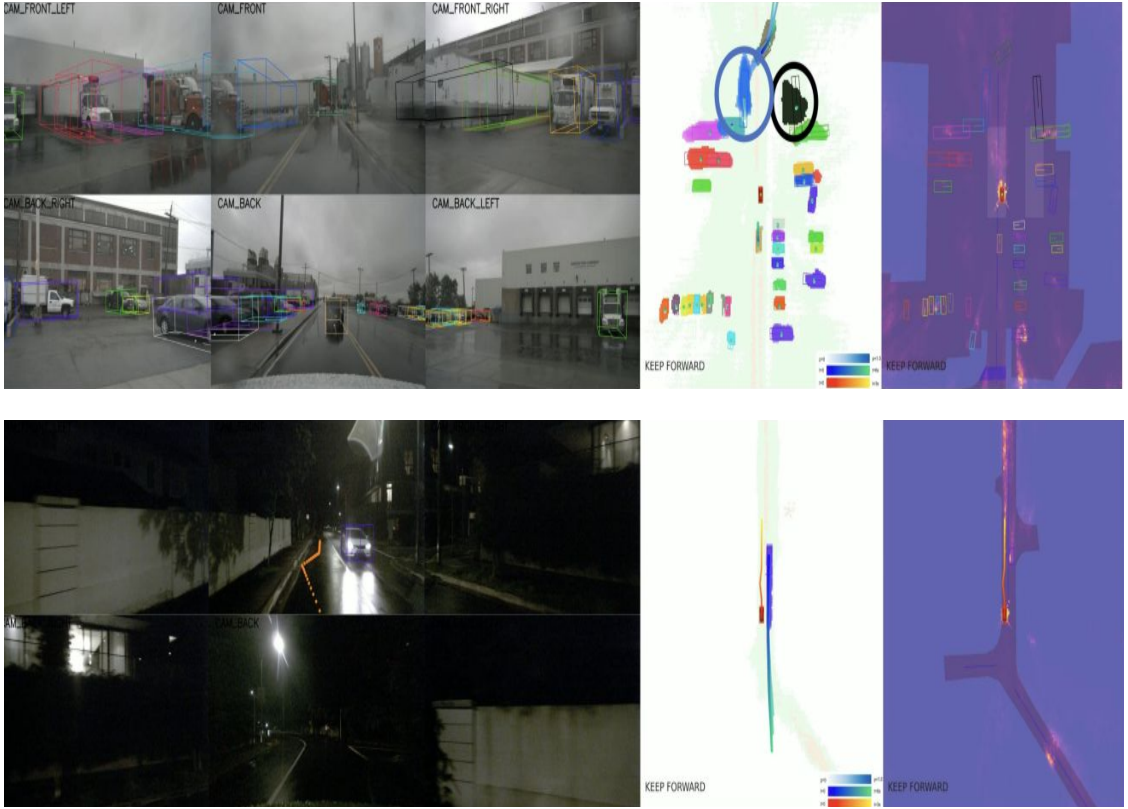
Cruising around urban areas

Obstacles avoidance visualization

NYU

Weakness

- **High computational complexity** limits deployment on resource-constrained platforms.
- Struggles with **long-tail scenarios** like large trailers or poorly lit environments.
- Adding more tasks may increase **system complexity and training difficulty**.



Applications in Embodied environment

1. **Urban Autonomous Driving:** Real-time navigation in dense traffic, handling tasks like obstacle avoidance and pedestrian yielding.
2. **Simulated Driving (CARLA):** Testing UniAD's performance in diverse traffic scenarios such as intersections and roundabouts.
3. **Warehouse Robots:** Guiding autonomous robots for dynamic obstacle avoidance and route planning in warehouses.
4. **Collaborative Driving:** Coordinating vehicle-to-vehicle communication for safe and efficient traffic flow.

NYU

Result

Future Scope, and Extensions:

Conclusion:

- UniAD introduces a novel planning-oriented framework that unifies perception, prediction, and planning tasks, achieving state-of-the-art performance across multiple benchmarks.
- The query-based design ensures effective task coordination and interpretability, paving the way for safer and more robust autonomous driving systems.

Future Scope:

- Optimize the framework for lightweight deployment in real-time applications.
- Extend UniAD to include additional tasks like depth estimation and behavior prediction.
- Explore vehicle-to-vehicle communication for collaborative driving scenarios.


@LateNightSeth
THANK YOU!


Anybody got any questions?

EmbodiedGPT: Vision-Language Pre-Training via Embodied Chain of Thought

BY MU ET AL (2023): HTTPS://ARXIV.ORG/ABS/2305.15021

By Jovita Gandhi

End-to-End Planning

- A process that spans the entire workflow or task, from initial input to the final output, without relying on manual interventions or hand crafted intermediate steps.

- Often used in AI, robotics & Machine Learning to describe systems that directly learn or optimize a complete solution pipeline.

Some definations in the paper

- **Foundation Models:** Large pre-trained model trained on broad, diverse data at scale, designed to be adaptable to a wide range of down stream tasks with minimal task specific tuning.
- **Downstream Tasks:** Applying the learnt general knowldge from pre- training to do something specific. Eg: Nagivation to objects, “Bring me the apple”
- **Egocentric Videos:** 1 st person Point of view. Recorded with wearable cameras
- **Low-Level control tasks:** Requires precision & real-teim feedback. Eg: set torque on joint 2 to 0.1 nm

Problem the paper addresses

- Embodied AI required egocentric data.
- Structured language instruction for precise planning requires high efforts and costs.
- Less high quality embodied multi-modal data available
- Apply LLM’s to field of robotics in a generalised manner
- Leverage the “chain-of-thought” capability for structured planning
- How to use the output language plan for downstream manipulation tasks in a **end-to-end** manner.

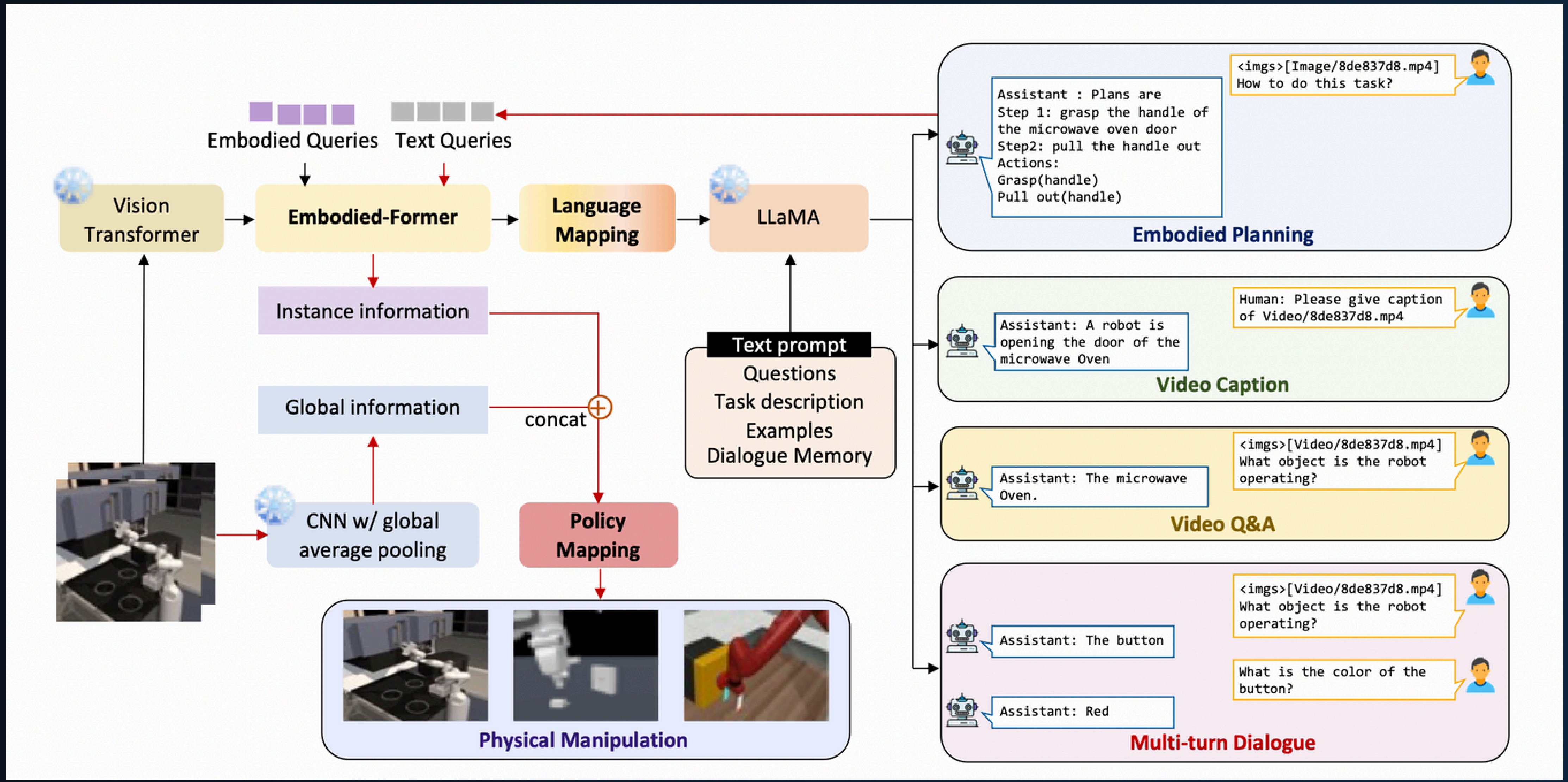
Solution undertaken - overview

- Built a large scale embodied planning dataset: **EgoCOT**
- Created an **EgoVQA** dataset
- Based off of the datasets, **presented** end-to-end multi-modal embodied foundation model called **EmbodiedGPT.**

Current works & where they lack

- Models such as Uniter, Oscar, VinVL, & LiT are large scale foundation models for vision language pre - training that freeze the image encoder.
- Whereas, Frozen & VGPT freeze the language model.
- Due to lack of open-source data for multi modal embodied planning, these works struggle to perform details task decomposition and lack ability to generate precise executable plans.

EmbodiedGPT



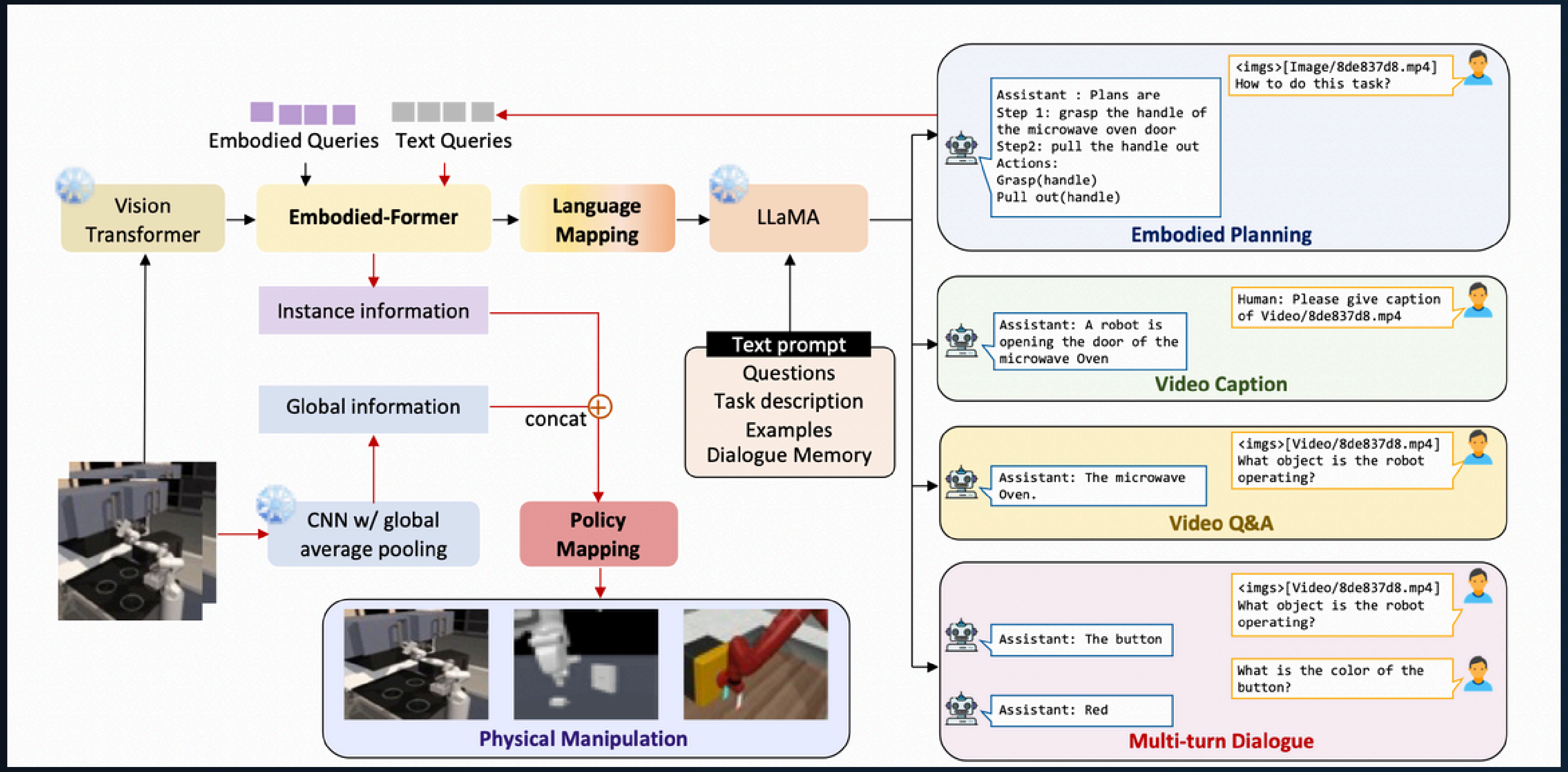
Goals:

- Imitate human-like perception and interaction
- identify relevant objects
- analyse spatial relationships
- formulate detailed task plan

Model features:

- Pre-trained vision transformer as the visual encoder
- Pre trained LLaMA model as the vision model

EmbodiedGPT



Process:

- compact visual features are extracted from output of vision model.
- Mapped to language modality through language mapping layer, and embeddings sent to the frozen LLaMA for visual caption, visualQA and embodied planning.
- This is used to query highly relevant features.
- Which are then utilised to generate low level control commands for task execution through a downstream policy network.

Addiotnally, to enhance performance, for generalisability, a novel video-language pre-training paradigm that leverages a cognitive chain of thought to prdouce planning from egocentric video inputs was introduced.

The math of the Architecture

$\mathcal{E}(\cdot)$ x_{vis} y_{query} N $z' \in \mathbb{R}^{N \times D'}$

$\mathcal{E}_{\text{vis}} : x_{\text{vis}} \to y_{\text{vis}}$ x_{plan} $M : z \to z'$

$\mathcal{E}_{\text{txt}} : x_{\text{txt}} \to y_{\text{txt}}$ z_{global} $z_{\text{instance}} = \mathcal{E}(x_{\text{vis}}, x_{\text{plan}}, y_{\text{query}})$

$z \in \mathbb{R}^{N \times D}$ $a = g(z_{\text{instance}}, z_{\text{global}})$

Whoa! Lots of info, let's break this down

The MATH

Visual Input processing

$\mathcal{E}(\cdot)$ Embodied Former

x_{vis} Image/video frame tokens

$\mathcal{E}_{\text{vis}} : x_{\text{vis}} \rightarrow y_{\text{vis}}$ E. applies visual feature extraction

Text Input processing

$\mathcal{E}_{\text{txt}} : x_{\text{txt}} \rightarrow y_{\text{txt}}$ E. applies text feature extraction

Cross-Modal interaction

N Learnable embodied query embeddings y_{query}

$z \in \mathbb{R}^{N \times D}$ Output of compact cross modal embedding after interacting with the cross attention with visual features, and the self attention with text features

Visual Language Mapping

$M : z \rightarrow z'$ Linear projection

$z' \in \mathbb{R}^{N \times D'}$ Project Z to match LLM input dimension. Ouptut z' : soft visual prompts for the frozen language model

Planning Generation

x_{plan} Frozen LLM receives z' + text prompt and outputs chain of thought embodied plan

Instance Feature Querying for control

$$z_{\text{instance}} = \mathcal{E}(x_{\text{vis}}, x_{\text{plan}}, y_{\text{query}})$$

Use the inputs to output task relevant
instance level features

Global feature extraction

z_{global} Use pre-trained ResNet-50 with global avaerage pooling to output scene context

Low-level Action Generation

$$a = g(z_{\text{instance}}, z_{\text{global}})$$

oolicy network g(.), an MLP takes the inputs and putputs the action command, eg: joint angles, velocities

Training set up for EmbodiedGPT

- ViT-G/14 image encoder from EVA-CLIP and a LLaMA-7B language model are used, both kept frozen during vision-language pre-training.
- The language model is fine-tuned beforehand on instruction-following datasets (ShareGPT and GPT-4 generated data), and model weights are converted to FP16 to improve training efficiency.

Stages

- Stage 1 & 2: Focus on pre training in basic cognitive and responsive skills
- Stage 3: Involves training the embodied AI task with egocentric video-text data on **EgoCOT.**

Stage	Objective	Datasets	Training Focus
1	Image-text conversation alignment pre training	- COCO Caption [44] - CC3M (595K filtered pairs) [45] - Re-captioned LAION-400M (491K) via BLIP-2 [17]	• Train Embodied-former and language projection • Freeze vision/language model parameters
2	Enhance complex sentence understanding and reasoning	- Complex_Reasoning_77k - LLaVA_Instruct_150K	• Update language projection and prefix language adapter
3	Train on embodied AI task with egocentric video-text data	- EgoCOT	• End-to-end training on egocentric vision-language grounded tasks

Stage 3: Embodied “chain-of-thought” training with EgoCOT

- **Vision Transfer with Conv3D:** Adapt the pre-trained image encoder from stage 2 to videos using Conv3D (time offset = 2, 8 frames total).
- **Chain-of-Thought Prompting:** Introduce vision-language pretraining with structured prompts that include task description, planning steps, and verb-noun action summaries (see Listing 1).
- **Fine-tuning for Temporal Reasoning:** To avoid overfitting, fine-tune the patch embedding, language projection, and prefix adapter to better capture temporal information.

```
Watch this video, identify the actions and devise a plan using chain-of-thought. Extract
    detailed actions using this schema:
Task: {"task description"}
Plan: {"plan with chain-of-thought"} Actions: {{"number"}: {'verb'}({'noun'})}.
```

Listing 1: Prompt we used for chain-of-thought pre-training.

EgoCot creation

- Obtained from Ego4D dataset [9,645 untrimmed videos of various durations from 5 seconds to 7 hours]
- 2 stages of data cleaning:
  - filtered videos with missing or short narrations, and with unsure tags
  - excluded videos without human-object interaction
- Ultimately left with 2927 hours of video [3.85 million narrations, from 129 different scenarios]

```
You need to generate plans with chain of thought for each task, and then extract
    detailed actions (collocation of nouns and verbs) from the plan.
The action can be of the following form:
[action_name], eg., turn left;
[action_name] argument1, eg., pick up(apple);
[action_name] argument1 argument2, eg., put(apple, table)
Task: pick up a cup on the table
plans: grasp the handle of the cup with the gripper and lift it up
Actions:
1. grasp(handle of the cup, gripper)
2. lift up(cup)
```

Listing 2: Prompt we used for creating EgoCOT dataset.

More math/notations

- To pair each narrated sentence Ti, with a relevant video segment Vi, first use its timestamp ti from the Ego4D dataset.
- Then calculate the average time gap between narrations in a video as βi*, and normalize it using a global scaling factor αα(set to 4.9 seconds).
- The start and end of each clip are defined as $[t_i^{start}, t_i^{end}] = [t_i - \beta_i/2\alpha,\ t_i + \beta_i/2\alpha]$, ensuring each segment captures the action context around the narration.
- This method automatically aligns video segments with narrations without manual annotation. These segments are then used to generate chain-of-thought plans and action labels via ChatGPT.

$$\beta_i = \sum_{j=0}^{n-1} \left(t_{j+1} - t_j\right)/n$$

Adjustable parameter equal to the average temporal distance between consecutive narrations in a given video

Finally, similarity score

- To pair each narrated sentence Ti, with a relevant video segment Vi, first use its timestamp ti from the Ego4D dataset.
- Then calculate the average time gap between narrations in a video as βi*, and normalize it using a global scaling factor αα(set to 4.9 seconds).
- The start and end of each clip are defined as $[t_i^{start}, t_i^{end}] = [t_i - \beta_i/2\alpha,\ t_i + \beta_i/2\alpha]$, ensuring each segment captures the action context around the narration.
- This method automatically aligns video segments with narrations without manual annotation. These segments are then used to generate chain-of-thought plans and action labels via ChatGPT.

$$\beta_i = \sum_{j=0}^{n-1} (t_{j+1} - t_j)/n$$

Adjustable parameter equal to the average temporal distance between consecutive narrations in a given video

Finally, similarity score : Post Procedure

- To ensure video-caption-plan quality, compute similarity scores between video frames and text using CLIP embeddings. Each frame Ii and corresponding text Ti are encoded into feature vectors YIi and Yt, and their cosine similarity is calculated as $S(y_T, y_I) = \frac{y_T \cdot y_I}{||y_T|| \, ||y_I||}$

- Since each video has multiple keyframes, they compute an ensemble similarity score across all frames: $E(V,T) = \frac{1}{n}\sum_{i=1}^{n} S(y_{T_i}, y_{I_i})$

- This averaged score E(V,T), ensures robust alignment across frames and is used to filter out mismatched video-caption-plan triplets, keeping only high-quality examples for the EgoCOT dataset.

Prompt to create EgoVQA dataset

- For each caption in Ego4D dataset, ChatGPT was used to generate five QA pairs.

```
Please ask some questions accroding to the verbs and nouns in the sentence.
For example, in this sentence "a man is picking up a cup", the verb is picking up and the
      noun is cup, therefor questions can be "what is the object the man is picking up?"
     or "what operation is performed on the cup?".
Then You need to give the answer.

input: a man is picking up a cup
question: What is the object the man is picking up
answer: The cup
```

Listing 3: Prompt used for creating EgoVQA dataset.

Evaluation metrics

Evaluation Metric	Explanation
Object Recognition Accuracy	This metric measures the ability of a system to accurately identify objects from images or videos. A higher accuracy indicates that the system can correctly recognize the objects present in the given visual data.
Spatial Relationship Understanding	Spatial relationship understanding refers to the system's capability to accurately discern the spatial relationships between objects in a scene. It evaluates whether the system can determine the relative positions, orientations, distances, and other spatial attributes of objects with precision.
Level of Redundancy in the Answer	The level of redundancy in the answer assesses the amount of unnecessary or repetitive information present in the system's response. Lower redundancy indicates that the system provides concise and non-repetitive answers, which is generally preferred as it reduces verbosity and improves clarity.
Reasonability of the Planning	The reasonability of the planning metric gauges the logical coherence and appropriateness of the system's planning process. It examines whether the system's generated plans are sensible and align with the given goals or objectives.
Executability of the Planning	This metric evaluates the feasibility and practicality of the system's generated plans. It assesses whether the plans can be executed successfully in the real world.

Table 3: Explanation of Evaluation Metric

Experiment 1 : Eval on Image input tasks

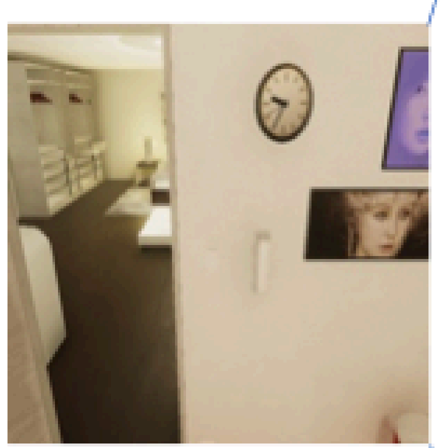
- What was evaluated?
  - Quality of generated captions and planning with given image

Model	Object(↑)	Spatial(↑)	Redundancy(↓)	Plan Reasonable(↑)	Plan Executable(↑)
Minigpt4	5.6	4.8	4.4	4.5	4.8
LLaVA-7B	7.3	7.4	3.9	7.5	6.6
LLaVA-13B	**8.5**	8.6	3.4	8.4	7.6
EmbodiedGPT	8.4	**8.8**	**2.6**	**8.8**	**8.4**

Table 1: Generate Quality Evaluation on image input tasks.

- Where did these numbers come from?
  - User study with **30** participants evaluated image captions and embodied plans generated by different models. Participants rated outputs from **10 MS-COCO images**, **5** simulated embodied scenarios, and **5** real-world planning tasks across five criteria on a 1–10 scale. The average scores are reported in Table 1.

Experiment 1 : Eval on Image input tasks



Human: give a detailed explanation of the image, describe the scene and the observed objects, and spatial relationships in this image. Where should I go if I wish to find a coat hanger in this image?

EmbodiedGPT:
1. The image is of a doorway with a clock on the wall above the door and a picture hanging on the wall.
2. There is a coat hanger on the floor, placed towards the left side of the scene.
3. The coat hanger is positioned near the right side of the doorway, and it is visible to the left of the woman's picture.
4. The coat hanger is near the edge of the image, and it is positioned in a way that it does not obstruct the view of the clock or the woman's picture.

Visual ChatGPT:
The image shows a bathroom with a clock on the wall and a toilet in the comer. The clock is white and has a black face with two hands owalls are tiled and there is a window in the room.
The image shows a bathroom with a clock and a toilet. There is no coat hanger visible in the image.

Figure 3: Comparison between EmbodiedGPT and VisualGPT in the question-answering task.

- Visual ChatGPT fails to find coat hanger due to its relieance solely on caption model for extracting visual information

Experiment 2 : Eval on Video input embodied AI Tasks

- What was evaluated?
  - Recognition ability of videos & planning abilites of EmbodiedGPT from embodied control tasks on standard embodied AI benchmarks
- How was it evaluated?
  - Benchmark models are Franka Kitchen and Meta-World
  - Meta-world requires complex object manipulation skills, eg: ring on a peg, picking and placing etc
  - Franka kitchen focuses on sliding open a door, opening a cabinet/microwave
- Results: EmbodiedGPT accurately interpret the embodied control task and provided a step-by-step planning

Experiment 2 : Eval on Video input embodied AI Tasks

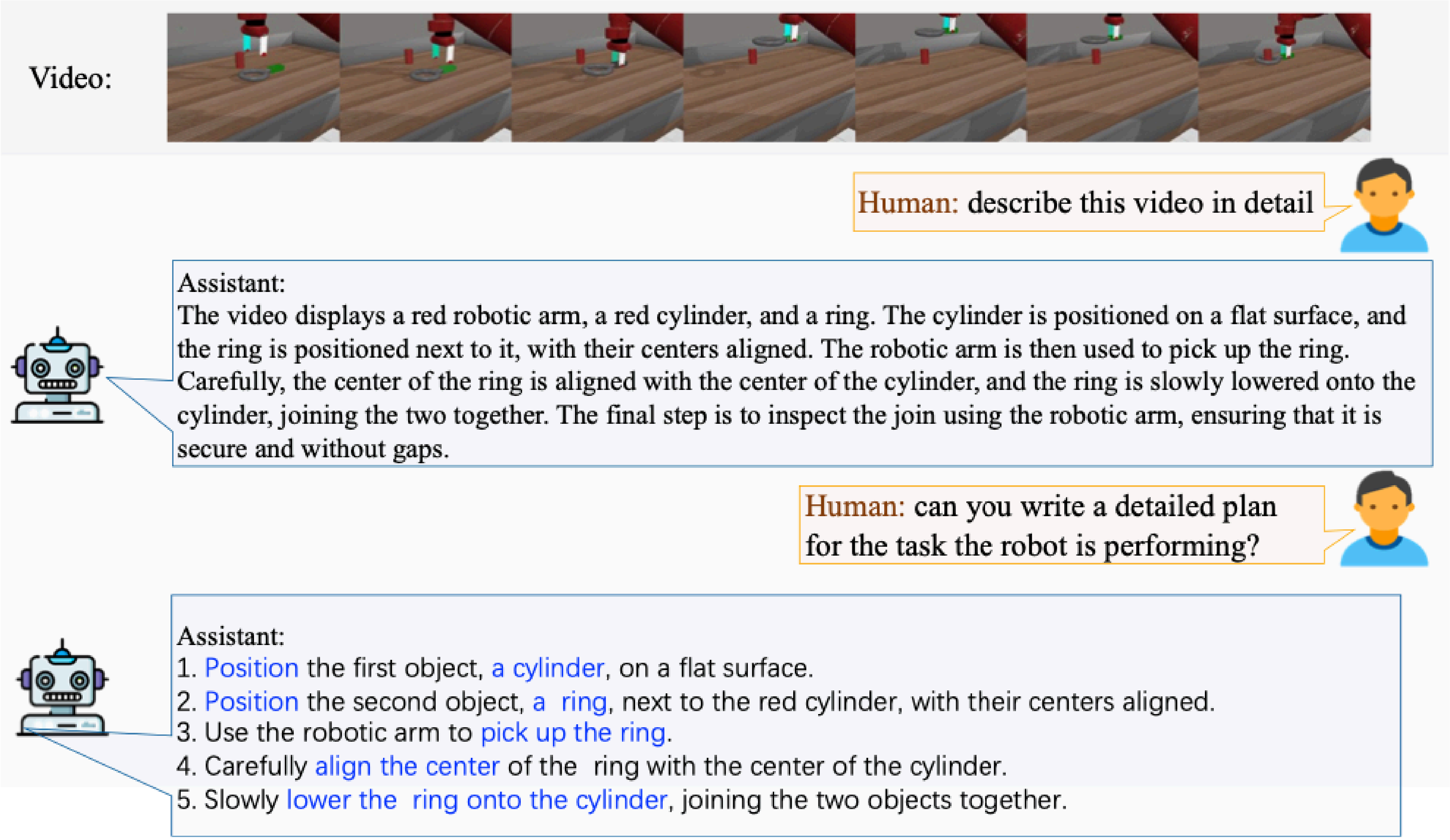


Figure 4: Example of video input embodied AI tasks on Meta-World benchmark. EmbodiedGPT accurately analyzes embodied control tasks in demonstration videos and provides precise planning.

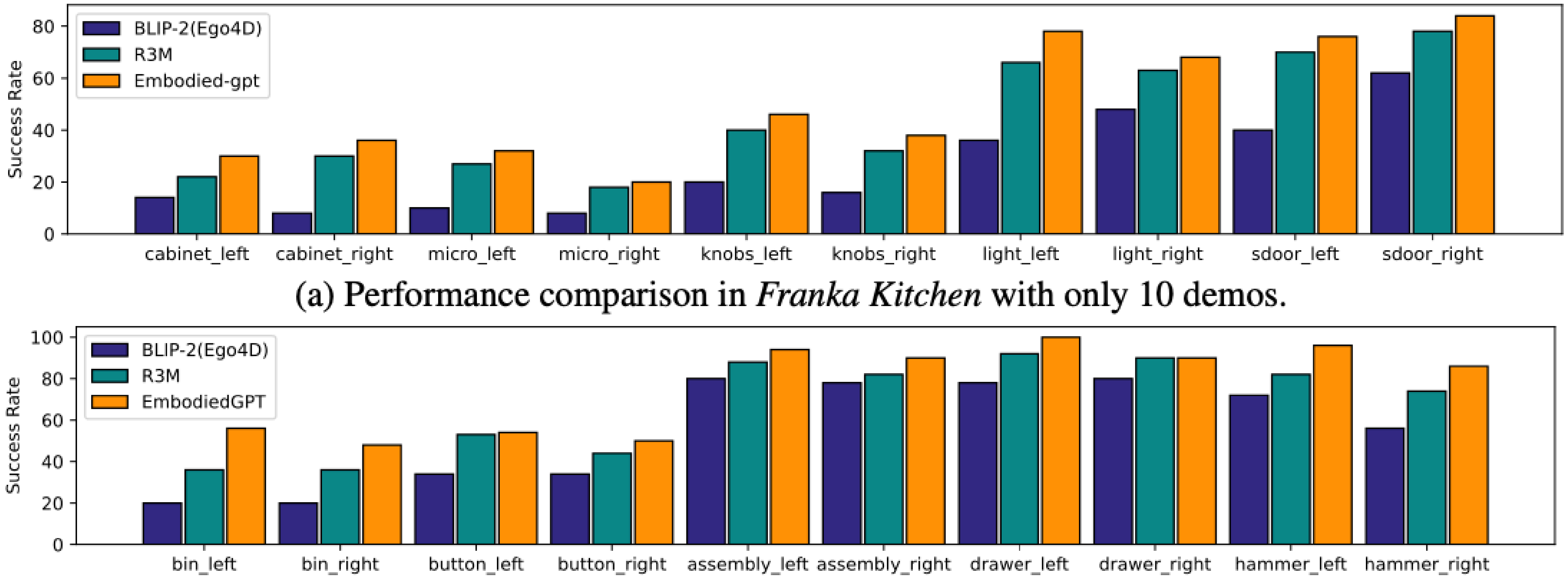
Experiment 2 : Eval on Video input embodied AI Tasks

- What happens next?
    - The output planning is fed into the Embodied-former module of EmbodiedGPT to query highly relevant features for use as inputs in the policy network and the low-level actions are generated by the policy network to interact with the environment

Experiment 3 : Eval on embodied control tasks

- What was evaluated?
  - Compare model with SOTA [R3M] & an ablation version called [BLIP-2[Ego4D], and is only fine tuned on the video captioning task, has the same parameters as EmbodiedGPT
  - The policy network was trained using few-shot learning with either 10 or 25 demonstrations per task
  - Performance is evaluated over 100 trials using visual observations across 5 tasks, 5 seeds, and 2 camera views.
- What was found?
  - EmbodiedGPT consistently outperforms baseline models, highlighting the effectiveness of training with EgoCOT.

Experiment 3 : Eval on embodied control tasks

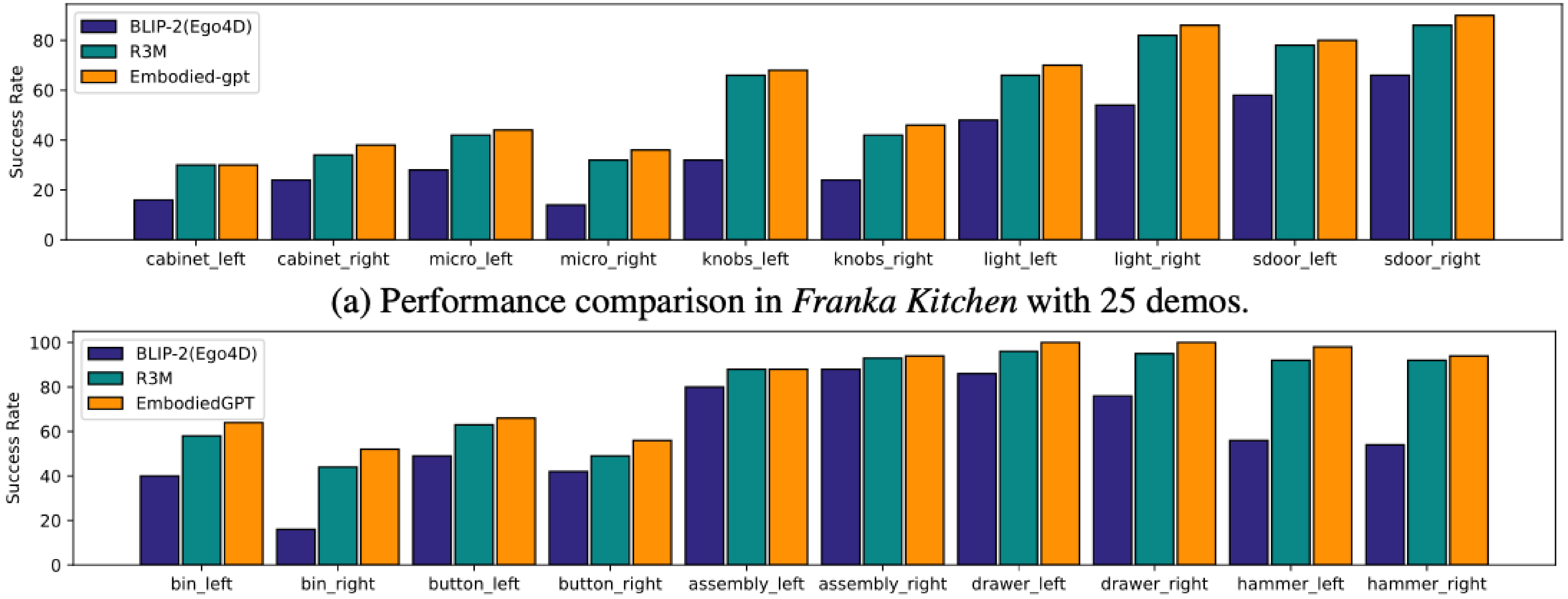


(a) Performance comparison in *Franka Kitchen* with only 10 demos.

(b) Performance comparison in *Meta-World* with only 10 demos.

Figure 5: Performance of EmbodiedGPT in low-level control tasks with 10 demonstration demos.

Experiment 3 : Eval on embodied control tasks



(a) Performance comparison in *Franka Kitchen* with 25 demos.

(b) Performance comparison in *Meta-World* with 25 demos.

Figure 6: Performance of EmbodiedGPT in low-level control tasks with 25 demonstration demos.

Finally, almost there!

- Key Takeaways:
  - Introduce EmbodiedGPT, an end-to-end multi-modal foundational model for embodied AI.
  - Enables agents to perform step-by-step planning and low-level action execution.
  - Built on a large-scale dataset, EgoCOT, with chain-of-thought planning annotations.
  - Uses prefix tuning to efficiently train high-quality planning behavior.
  - Seamlessly integrates high-level planning with low-level control.

Finally, almost there!

- Key Takeaways:
  - Achieves state-of-the-art or comparable performance on multiple embodied tasks.
  - Limitations:
    - Freezes vision and language model weights due to compute limits.
    - Highly reliant on textual inputs and may not fully exploit visual cues
  - Future work: Joint training and adding modalities like speech.

- EgoCAT Data set: https://github.com/EmbodiedGPT/EgoCOT_Dataset?tab=readme-ov-file

Thank you!

BY MU ET AL (2023): HTTPS://ARXIV.ORG/ABS/2305.15021