Мультимодальные и поведенческие модели

Imitation Learning and Action Chunking

MIPT VLA 2026 | 28 сентября 2026

Алексей Староверов



МФТИ

Center for Cognitive Modeling

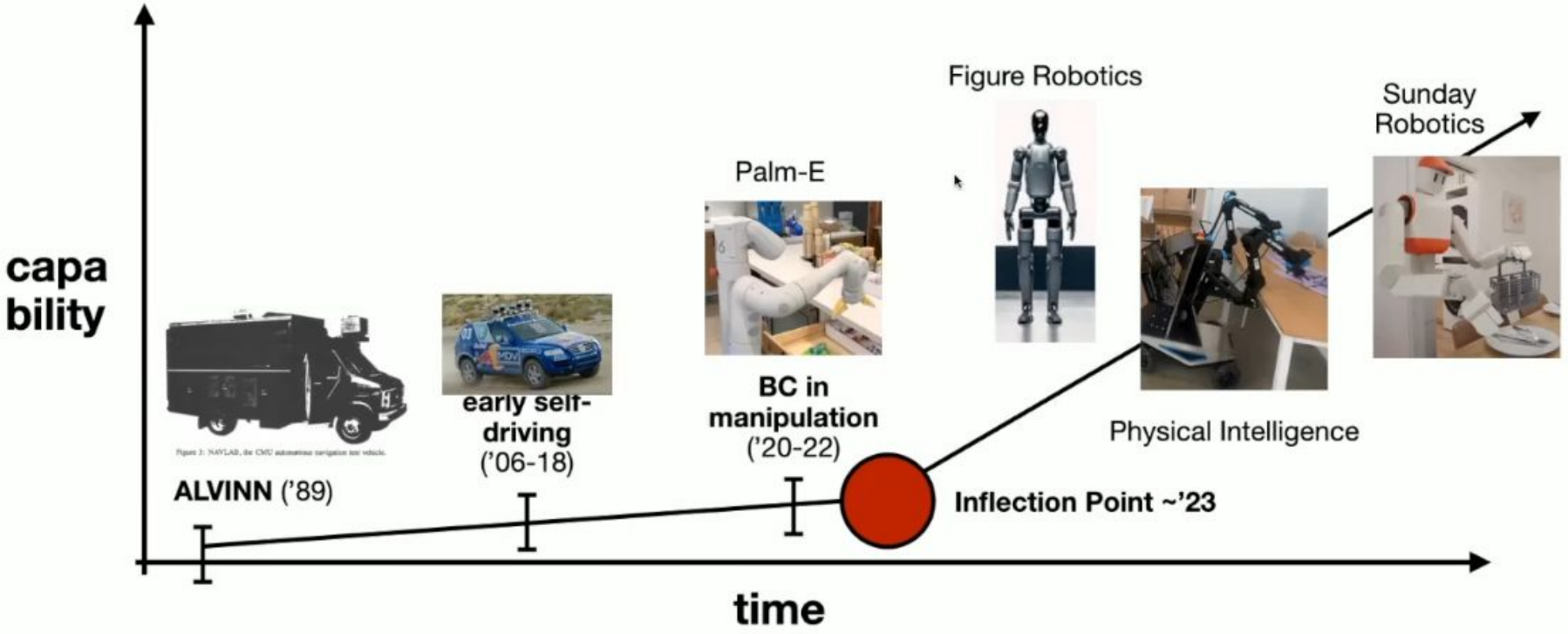
План лекции

Action Chunking and Exploratory Data Collection Yield Exponential Improvements in Behavior Cloning for Continuous Control

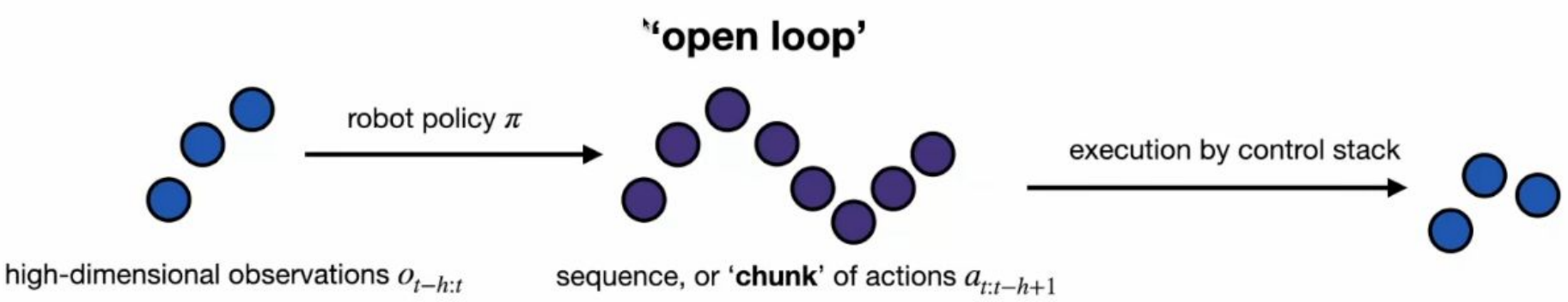
Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

Action Chunking with Transformers (ACT)

Inflection point of Robot learning



Action-Chunking



Action Chunking is a Foundational Innovation

Why is Robotic BC challenging?

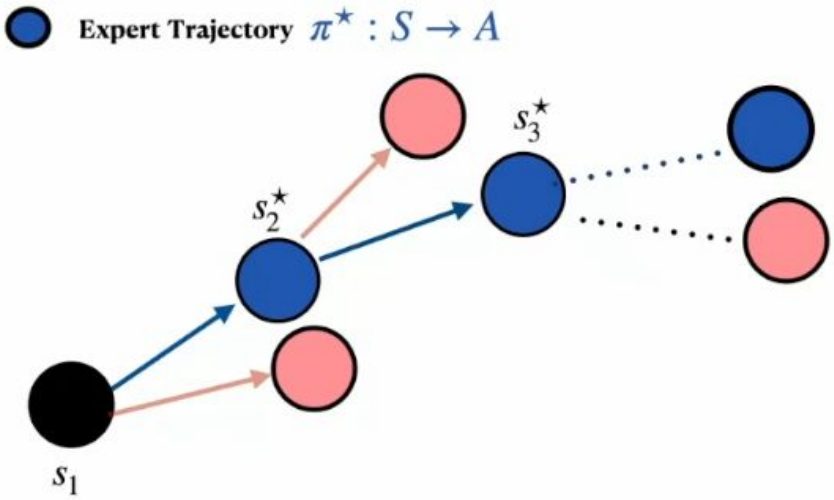
Behavior Cloning: Learner selects $\hat{\pi}$ to minimize **supervised learning error**

$$\hat{\pi} \approx \arg\min_{\pi \in \Pi} \sum_{i=1}^{n} \sum_{t=1}^{H} \text{loss}(s_t^{(i)}, a_t^{(i)}, \pi)$$

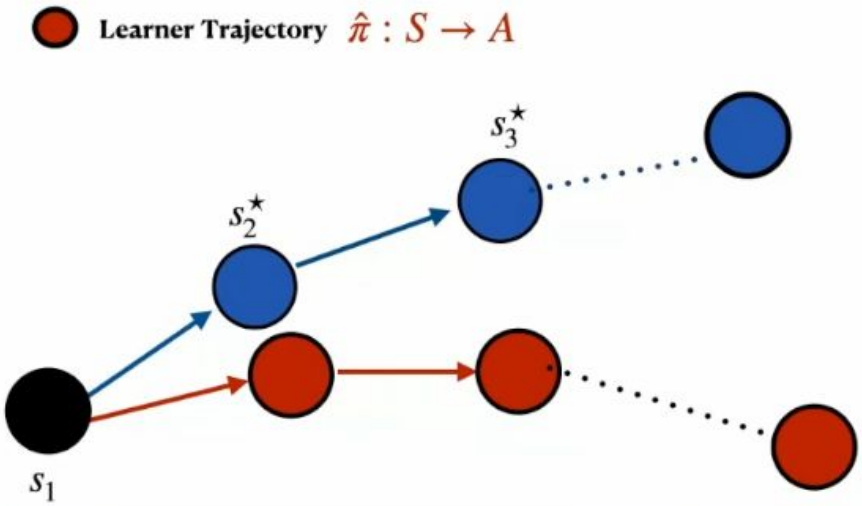
Central Challenge: Mismatch between BC error and cumulative reward:

Does $L_{\text{bc}}(\hat{\pi}) - L_{\text{bc}}(\pi^\star) \leq \text{small}$ **imply** $J(\pi^\star) - J(\hat{\pi}) \leq \text{small}$ **?**

Test-Time Error?



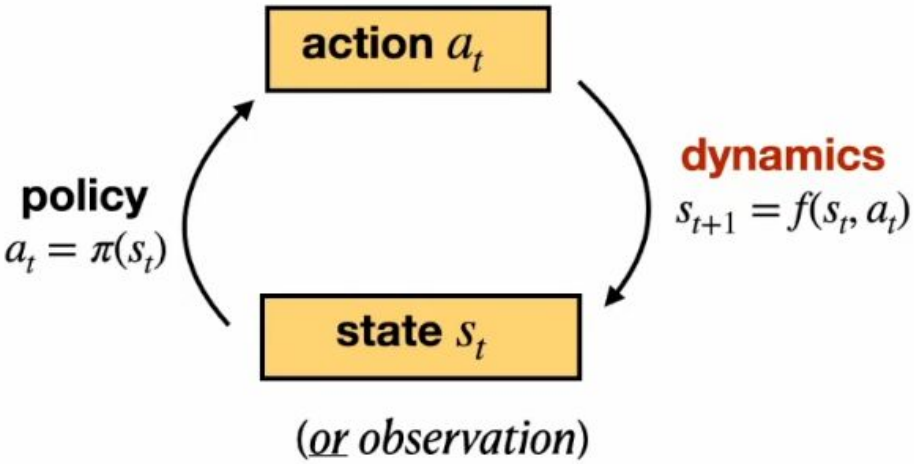
Learner selects $\hat{\pi}$ to minimize supervised learning error L_{bc}



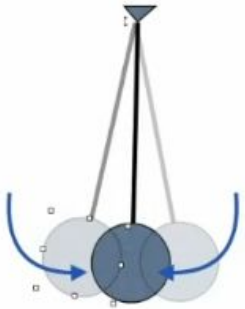
But errors can accumulate!

aka 'curse of horizon'

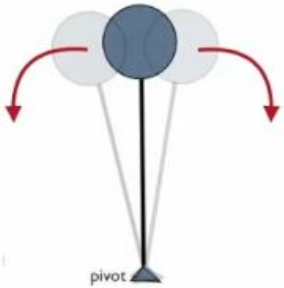
Robot Learning as a Control Problem



Definition: The **stability** of a control system refers to the **sensitivity** of the dynamics to perturbations in inputs.

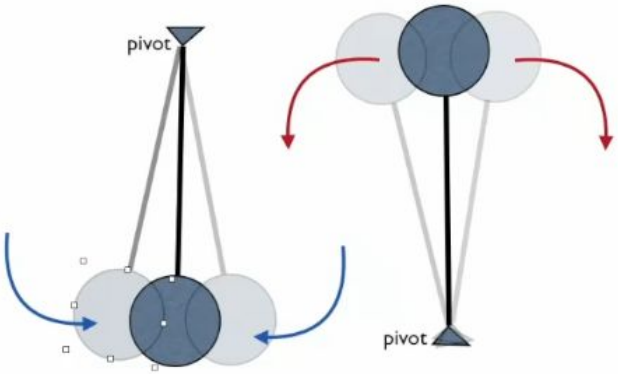


stable



unstable

Control Theoretic Stability



Definition (Informal)**: A dynamical system** is said to be **stable** if it has limited sensitivity to perturbations of input.

Incremental Stability, Agrachev '06

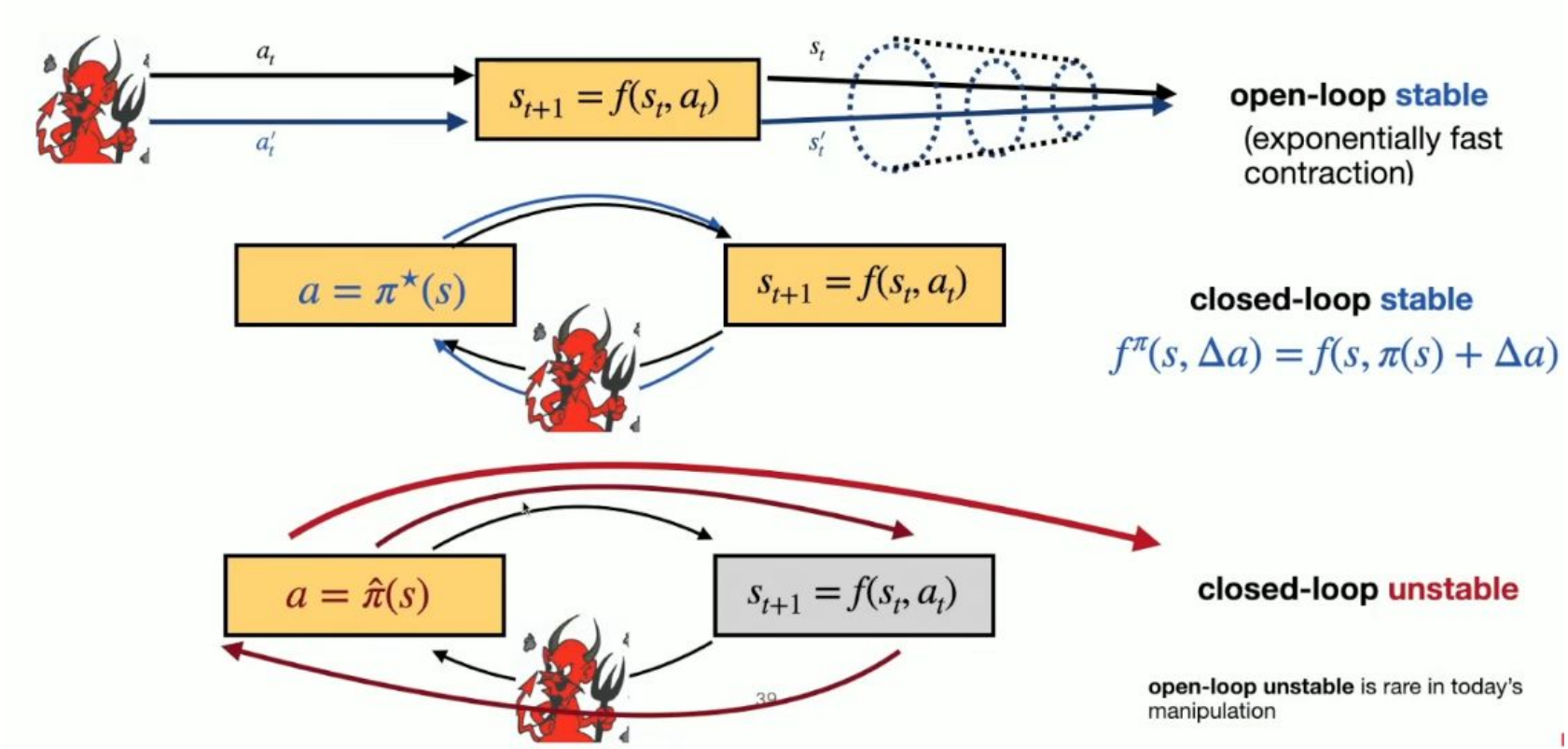
Definition: Dynamics $s_{t+1} = f(s_t, a_t)$ are (C, ρ) -**EISS** if, for same initial condition $s_1 = s'_1$

$$\|s'_{t+1} - s'_{t+1}\| \leq C \sum_{s \leq t} \rho^{t-s} \|a_s - a'_s\|$$

$\rho \in (0,1)$ = **exponentially quick forgetting of mistakes**

EISS - Exponential Input-to-State Stability

Scenarios of (Incremental) Stability



A Benign Problem Regime

Theorem (SPJ '25)**:** Consider the following setting

(1) The learner is given a set of pairs $(\pi, f) \in \mathcal{I}$ of possible expert/dynamics

(2) Each $(\pi, f) \in \mathcal{I}$ is **smooth, deterministic, Markovian & stable***

(3) The learner is **promised** that $(\pi^\star, f^\star) \in \mathcal{I}$ (well-specification)

(4) The learner must return **smooth, Markovian, deterministic*** $\hat{\pi}$

(generalizations permitted)

Markovian Matches the Expert

Continuous-action imitation is hard

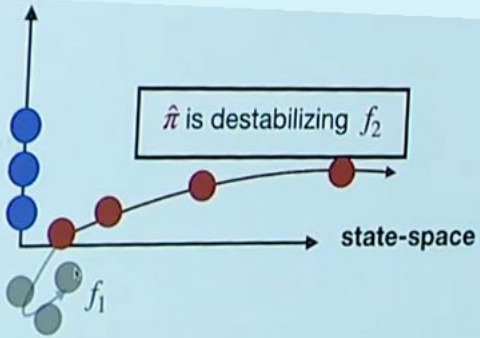
Theorem (SPJ '25)**:** Fix any scaling law $\epsilon_n = n^{-\alpha}$. There exists a set of $(\pi, f) \in \mathcal{F}$ satisfying the **niceness** assumptions of the past slide such that

Supervised learning on the square loss has $L_{\mathrm{bc}}(\hat{\pi}) \lesssim n^{-\alpha}$

Any algorithm satisfying (4) suffers $J(\pi^\star) - J(\hat{\pi}) \gtrsim \min\{2^H \cdot n^{-\alpha}, 1\}$

Construction: $\exists$ instances (π_1, f_1) and (π_2, f_2) which are **indistinguishable** on subspace

However, any $\hat{\pi}$ which stabilizes f_1 and agrees on subspace must **destabilize** f_2



Motivating Action Chunking

Theorem (SPJ '25)**:** Consider the following setting

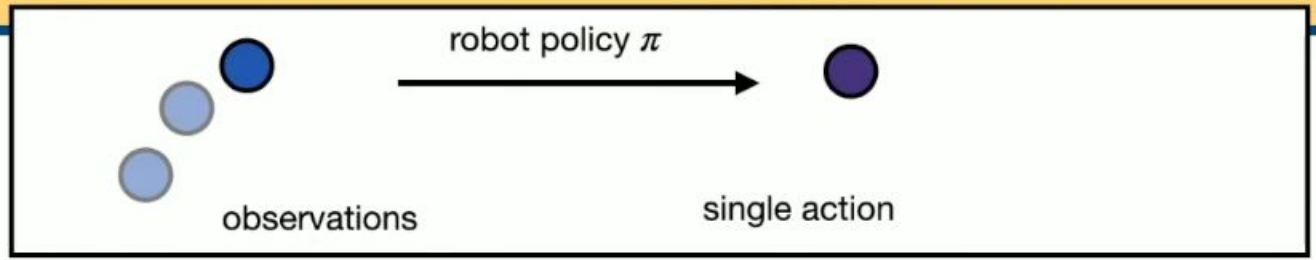
(1) There learner is given a set of pairs $(\pi, f) \in \mathcal{F}$ of possible expert/dynamics

(2) There learner is **promised** that each $(\pi, f) \in \mathcal{F}$ is **smooth & stable***

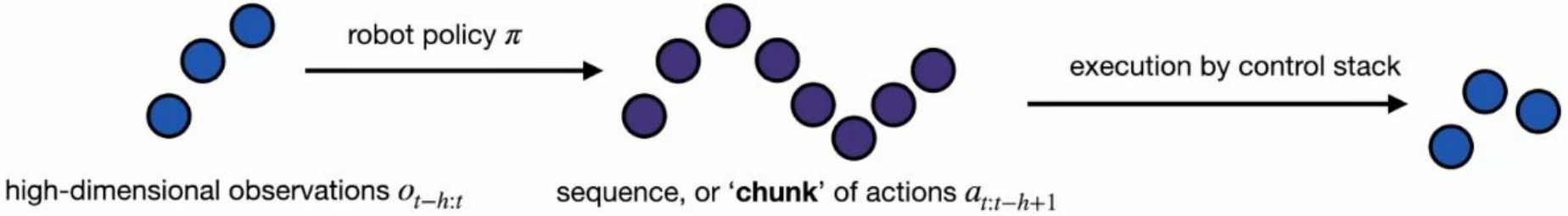
(3) There learner is **promised** that $(\pi^\star, f^\star) \in \mathcal{F}$ (realizability)

(4) The learner must return **smooth, Markovian, deterministic*** $\hat{\pi}$

(generalizations permitted)



Motivating Action Chunking



actions correlated across time

(Non-Markov!)

Define $L_{\text{bc,chunk}}(\pi) = \mathbb{E}^{\pi}\left[\sum_{t=1}^{H}\left(\sum_{i=0}^{k-1}\|\pi(s_t, i) - a_{t+i}\|^2\right)\right]$

Motivating Action Chunking

Theorem (ZFPM**S** '25)**:** Suppose that all $(\pi, f) \in \mathcal{F}$ are **open/closed-loop stable**. Then, for all chunk lengths $k \geq k_\star$

$$J(\pi^\star) - J(\hat{\pi}) \leq C_\star \, L_{\text{bc,chunk}}(\hat{\pi})$$

1. **Action Chunking Avoids Exponential Compounding Error**

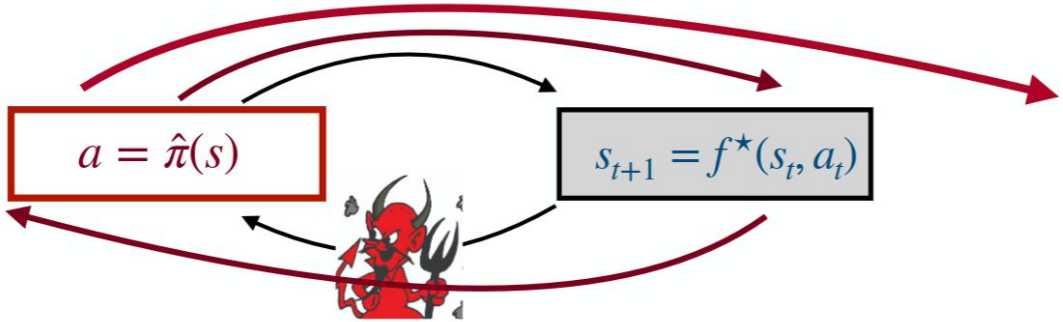
2. $k_\star, C_\star$ **polynomial in relevant factors / horizon-independent**

3. **Effect saturates at critical** $k_\star$ **and is horizon independent!**

Motivating Action Chunking

Just because $(\pi^\star, f^\star) \in \mathcal{I}$ is open & closed-loop stable….

Does not mean that $(\hat{\pi}, f^\star)$ is closed loop stable.

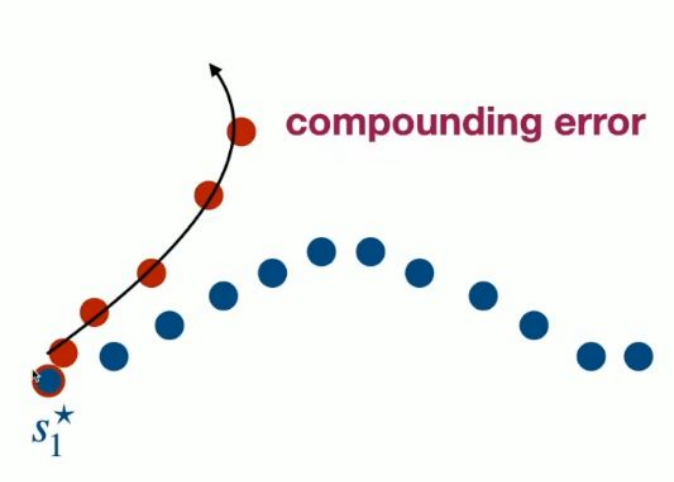


closed-loop unstable

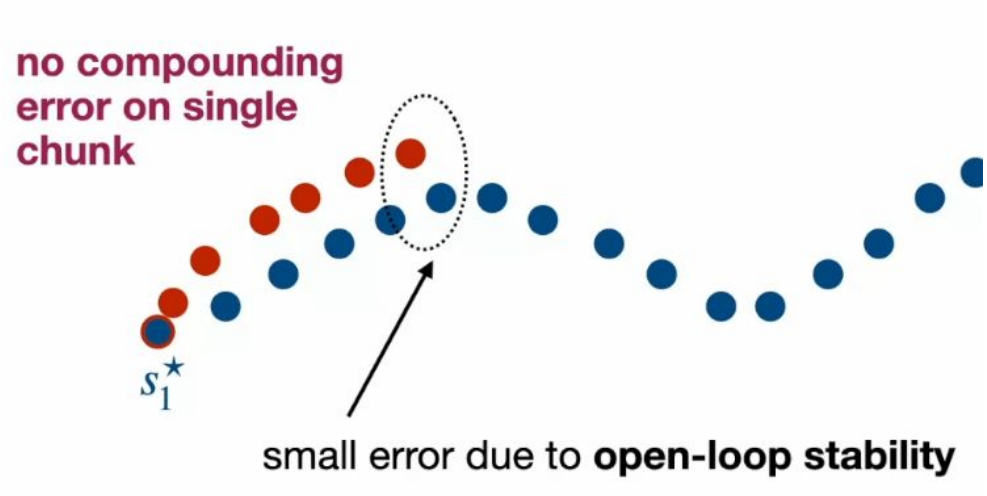
Because $f^\star$ is unknown, it may be impossible to ensure $(\hat{\pi}, f^\star)$ is closed-loop stable

Motivating Action Chunking

No Action Chunking

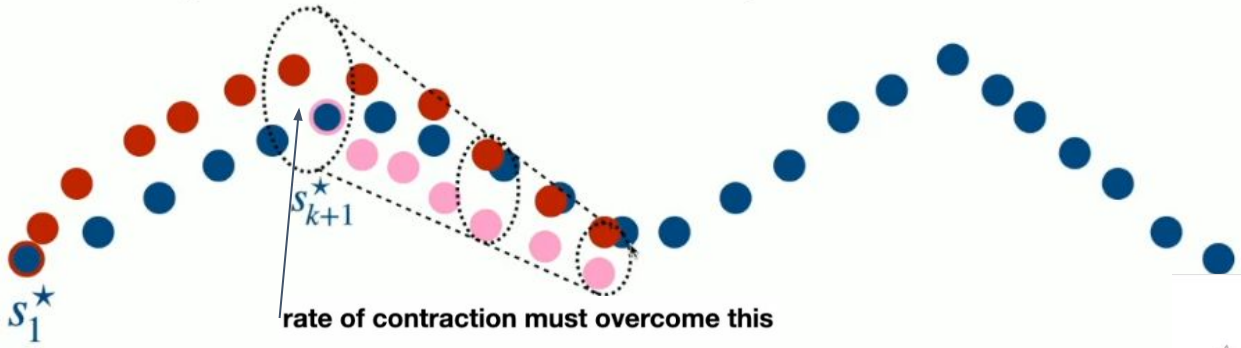


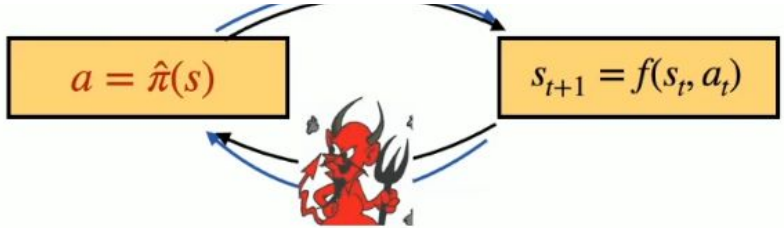
First Action Chunk

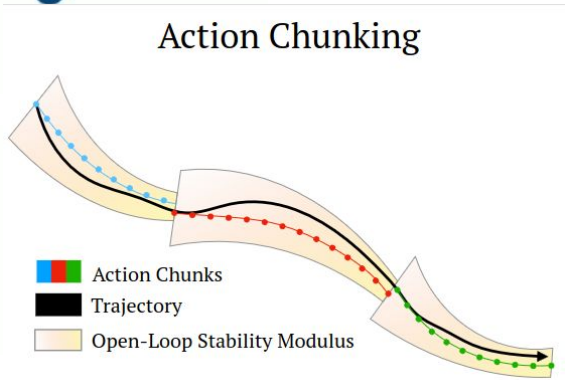


Second Action Chunk example

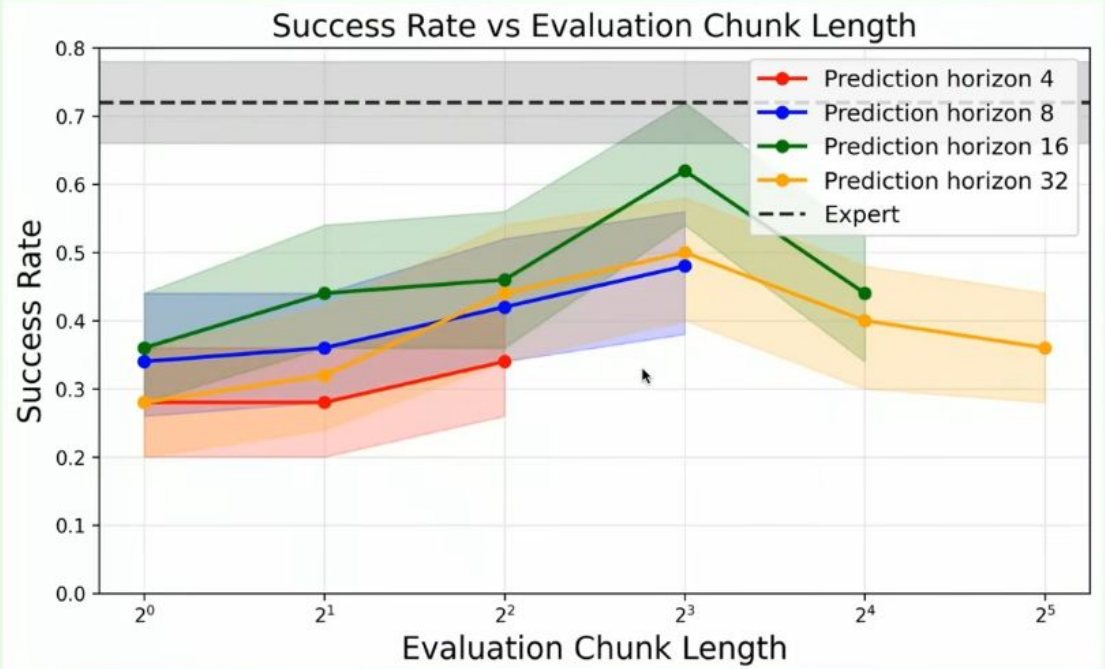
Key Finding: If $\hat{\pi}$ is Lipschitz and (some other assumptions), and chunks are large enough in **open-loop rate-of-contraction** of dynamics, then $\hat{\pi}$ is closed-loop stable .







Testing the Stability Hypothesis



We train a state-based regression policy from deterministic expert

Theory predicts length of deployed chunk as determining compounding error

Other hypotheses

1. representation learning
2. partial observation
3. receding horizon control

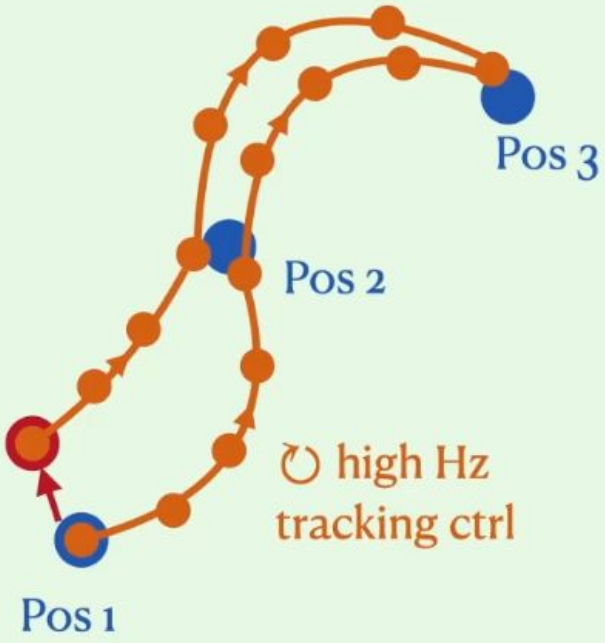
No Free Lunch: Open-Loop Stability

Core Assumption: Dynamics are Open-Loop Stable

Position control puts us in this regime!

1. position commands (e.g. 20 Hz) **tracked** by low-level controller, e.g. PD torque

2. "Open-loop" system mapping positions to state at policy freq (20 Hz) is **stable**



Inflection Point: Dynamical Reparametrization

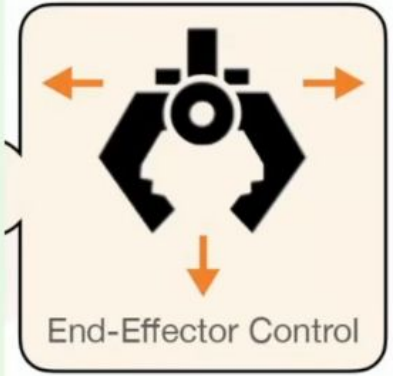
Core Assumption: Dynamics are Open-Loop Stable

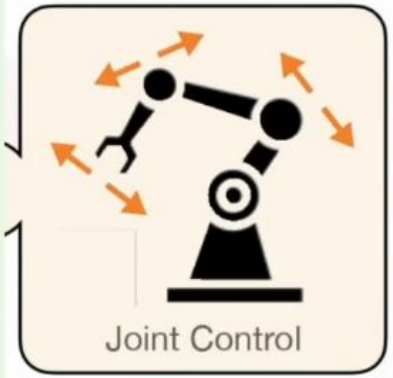
Key Idea: '23 Inflection **reparametrized** system dynamics to ensure **control-theoretic properties** requisite for BC.

action-chunking (**no more compounding error!**)

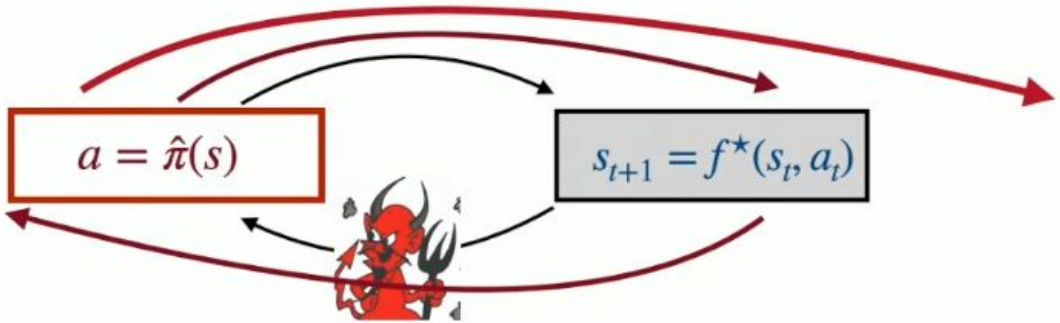
end effector control (*stable, but error prone*)

raw torques at joints (*unstable*)





Downsides of Action Chunking



closed-loop unstable

(1) **action-chunking 'slows' the policy to prevent instability** (System II)

(2) **it addresses instability related to the embodiment** (Cleaner movements, less jitters)

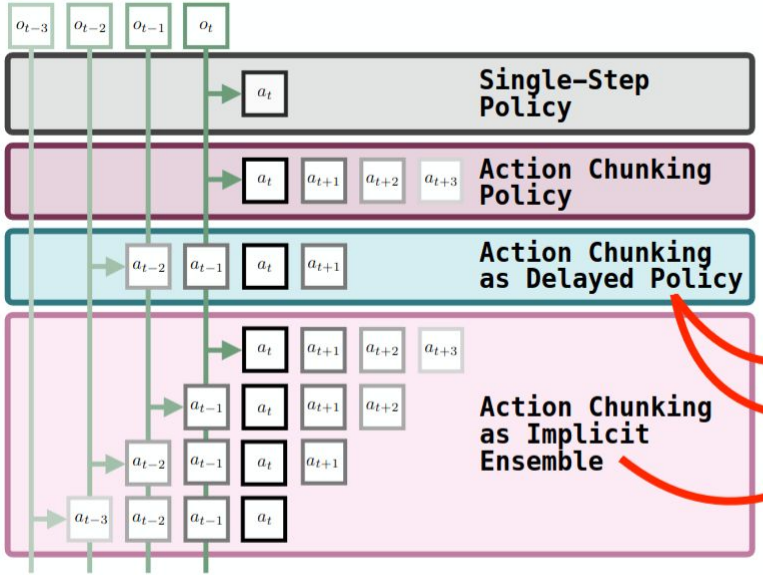
Downsides of Action Chunking

What about when the object + embodiment system is not stable?



Slippage means even if we control the hand, we can't guarantee stable control of the object.

Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

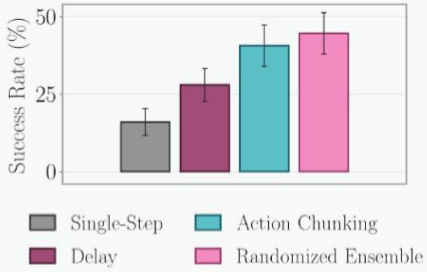


Why does action chunking improve BC performance?

- ✗ Temporal Consistency
- ✗ Horizon Reduction
- ✗ Representation Learning
- ✓ Non-Markovian Expressivity
- ✓ Reduced Compounding Error
- ✓ Ensemble Generalization

Matching action chunking without action chunking



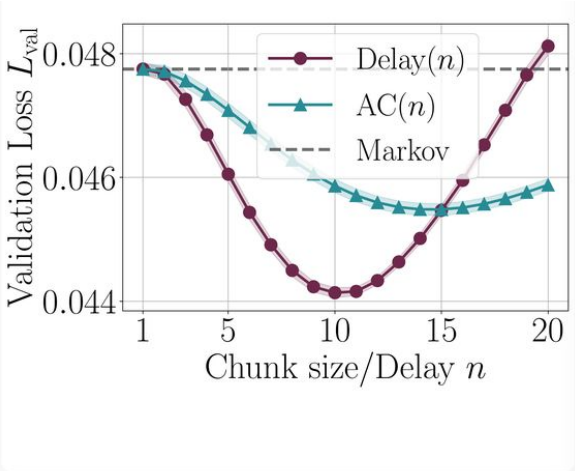


Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

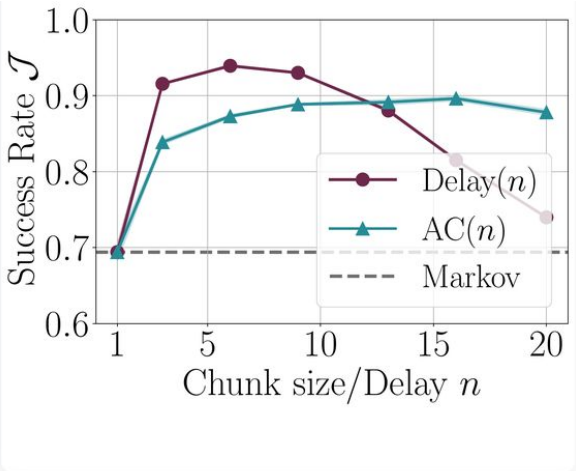
1. **Temporal consistency**: Action-chunked policies can better represent the temporally correlated behaviors exhibited by human demonstrators [1, 2, 6].
2. **Horizon reduction**: Action chunks reduce the effective horizon of the environment, mitigating compounding error [1].
3. **Representation learning**: Action chunking serves as an auxiliary loss in policy training, improving representation learning and generalization [2, 7].

Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

Action chunking improves performance by capturing non-Markovianity in human demonstrators, but "temporal consistency" is not, in general, required, and it suffices to simply predict actions based on past observations



Action prediction loss on held-out LIBERO-90 demonstration trajectories. Demonstrator actions can be more accurately predicted from an observation 10 steps in the past than the current observation.



Success rate on LIBERO-90 of action chunking policies and delayed policies, for different action chunk length and delay n. Delayed policies exceed the performance of action chunking for the correct choice of delay.

Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

Action chunking reduces compounding error by predicting actions based on past observations

THEORETICAL RESULT

Lower bound for Markovian behavioral cloning

There exists a deterministic environment with horizon H, a Markov demonstrator π_{demo}, and a Markov policy $\hat{\pi}$ with supervised learning loss ϵ, such that $\mathcal{J}(\pi_{\text{demo}}) \geq \mathcal{J}(\hat{\pi}) + \Omega(2^H \cdot \epsilon)$.

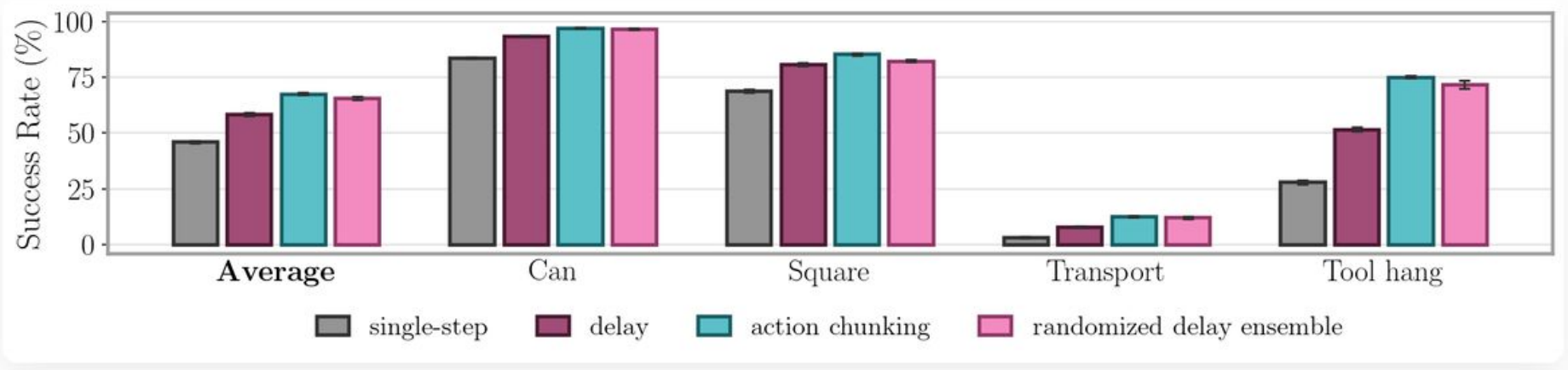
THEORETICAL RESULT

Upper bound for action chunking and delayed behavioral cloning

Suppose that the generalization error of each action chunk component/delayed policy is no larger than that of a Markovian learner, and is bounded by ϵ. Then both the action-chunked and delayed policies satisfy $\mathcal{J}(\pi_{\text{demo}}) - \mathcal{J}(\hat{\pi}_k),\ \mathcal{J}(\pi_{\text{demo}}) - \mathcal{J}(\pi^k_{\text{delay}}[\hat{\pi}_k]) \leq \mathcal{O}((k+1)^{H/k} \cdot \epsilon)$, where k is the chunk size/delay.

Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

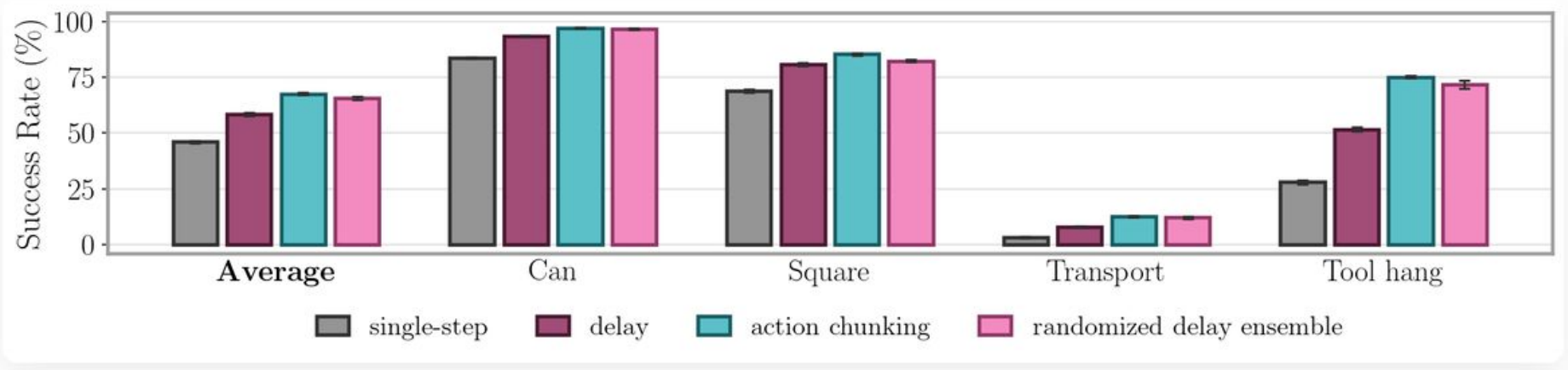
Action chunking implicitly instantiates an ensemble and benefits from ensemble-like generalization



At each step we select a random delay, predict the action chunk based on the observation at this delay, then execute the action from this chunk corresponding to the current step. In this way we both predict actions based on past observations, but also leverage all learned temporal relationships present in an action chunking policy—capturing all three effects without deploying actual action chunks

Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control?

Action chunking implicitly instantiates an ensemble and benefits from ensemble-like generalization



At each step we select a random delay, predict the action chunk based on the observation at this delay, then execute the action from this chunk corresponding to the current step. In this way we both predict actions based on past observations, but also leverage all learned temporal relationships present in an action chunking policy—capturing all three effects without deploying actual action chunks

Action Chunking with Transformers (ACT)

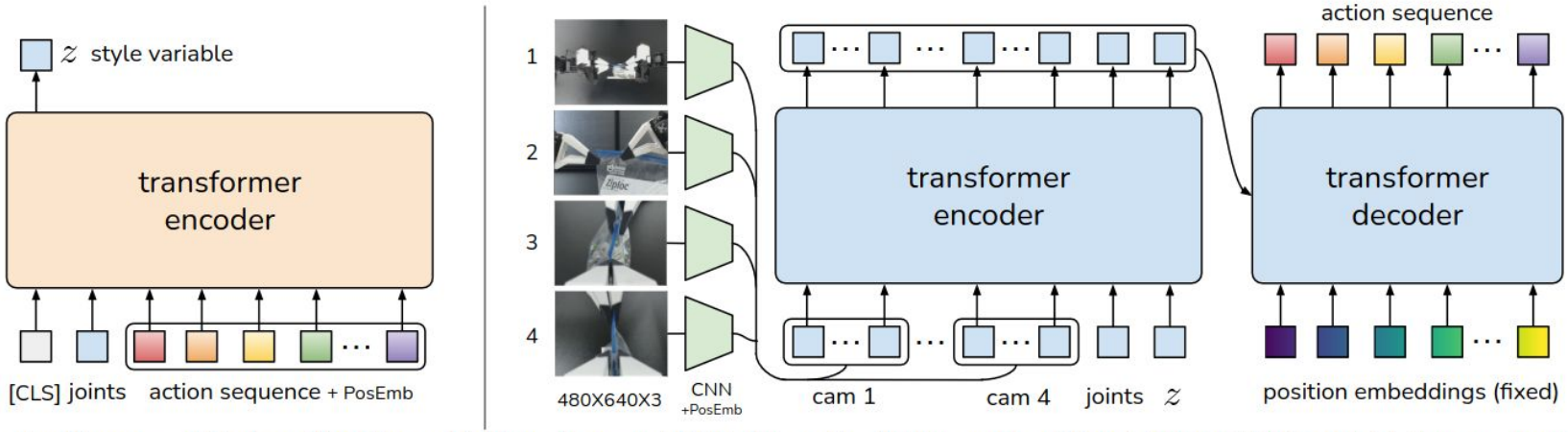


Fig. 4: *Architecture of Action Chunking with Transformers (ACT).* We train ACT as a Conditional VAE (CVAE), which has an encoder and a decoder. *Left:* The encoder of the CVAE compresses action sequence and joint observation into z, the style variable. The encoder is discarded at test time. *Right:* The decoder or policy of ACT synthesizes images from multiple viewpoints, joint positions, and z with a transformer encoder, and predicts a sequence of actions with a transformer decoder. z is simply set to the mean of the prior (i.e. zero) at test time.

Action Chunking with Transformers (ACT)

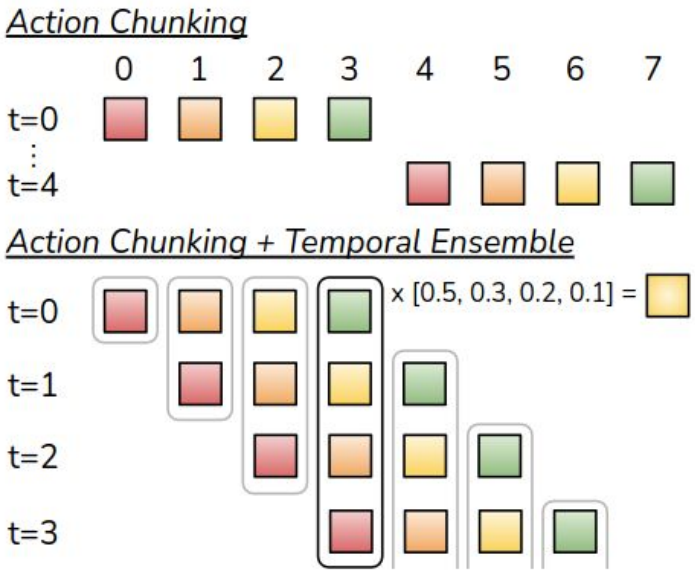


Fig. 5: We employ both Action Chunking and Temporal Ensembling when applying actions, instead of interleaving observing and executing.

Algorithm 1 ACT Training

Given: Demo dataset $\mathcal{D}$, chunk size k, weight β.
Let a_t, o_t represent action and observation at timestep t, $\bar{o}_t$ represent o_t without image observations.
Initialize encoder $q_\phi(z|a_{t:t+k}, \bar{o}_t)$
Initialize decoder $\pi_\theta(\hat{a}_{t:t+k}|o_t, z)$
for iteration $n = 1, 2, ...$ **do**
    Sample o_t, $a_{t:t+k}$ from $\mathcal{D}$
    Sample z from $q_\phi(z|a_{t:t+k}, \bar{o}_t)$
    Predict $\hat{a}_{t:t+k}$ from $\pi_\theta(\hat{a}_{t:t+k}|o_t, z)$
    $\mathcal{L}_{reconst} = MSE(\hat{a}_{t:t+k}, a_{t:t+k})$
    $\mathcal{L}_{reg} = D_{KL}(q_\phi(z|a_{t:t+k}, \bar{o}_t) \parallel \mathcal{N}(0, I))$
    Update θ, ϕ with ADAM and $\mathcal{L} = \mathcal{L}_{reconst} + \beta\mathcal{L}_{reg}$

Algorithm 2 ACT Inference

Given: trained π_θ, episode length T, weight m.
Initialize FIFO buffers $\mathcal{B}[0 : T]$, where $\mathcal{B}[t]$ stores actions predicted *for* timestep t.
for timestep $t = 1, 2, ...T$ **do**
    Predict $\hat{a}_{t:t+k}$ with $\pi_\theta(\hat{a}_{t:t+k}|o_t, z)$ where $z = 0$
    Add $\hat{a}_{t:t+k}$ to buffers $\mathcal{B}[t : t + k]$ respectively
    Obtain current step actions $A_t = \mathcal{B}[t]$
    Apply $a_t = \sum_i w_i A_t[i] / \sum_i w_i$, with $w_i = \exp(-m * i)$

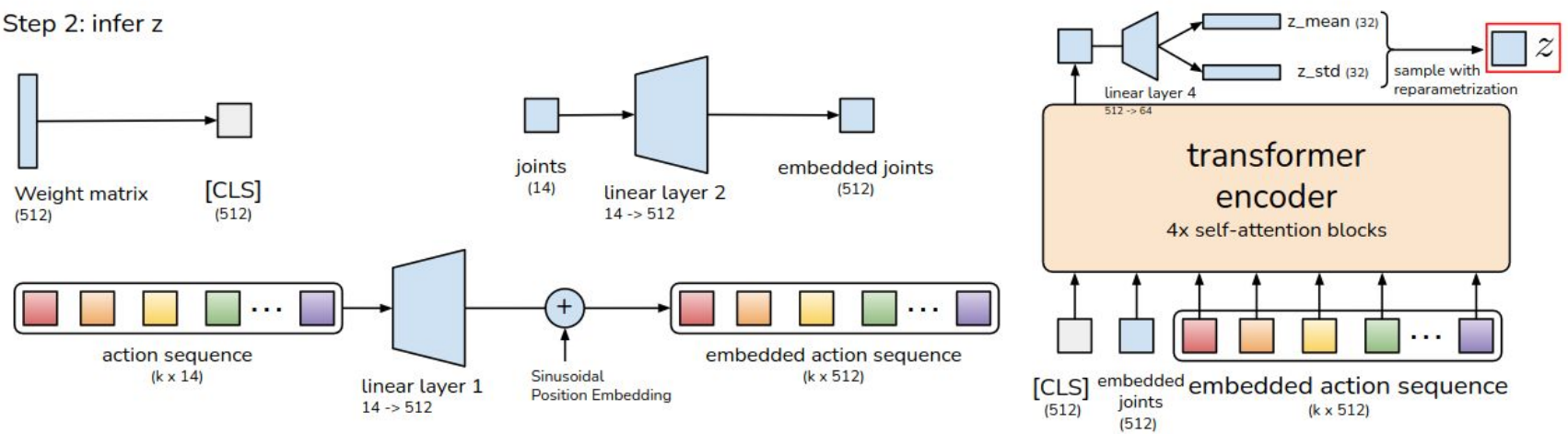
Action Chunking with Transformers (ACT)

Training

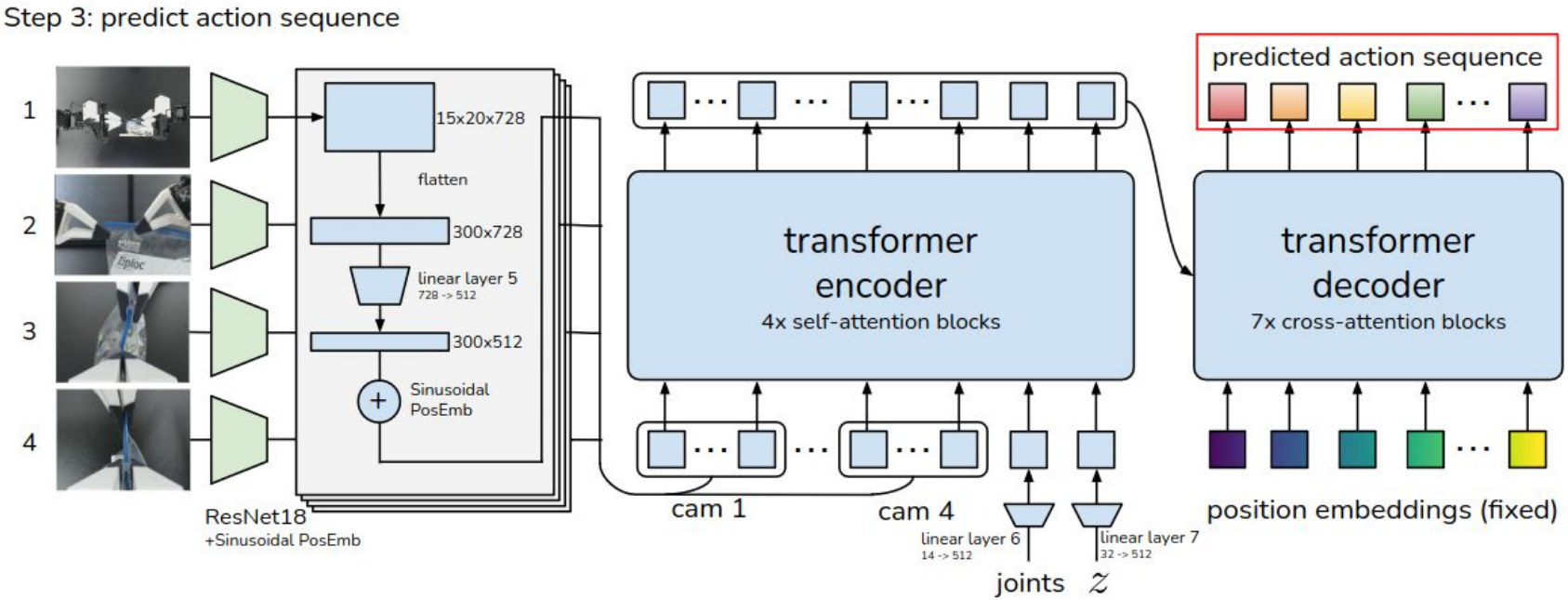
Step 1: sample data



Step 2: infer z



Action Chunking with Transformers (ACT)



Action Chunking with Transformers (ACT)

