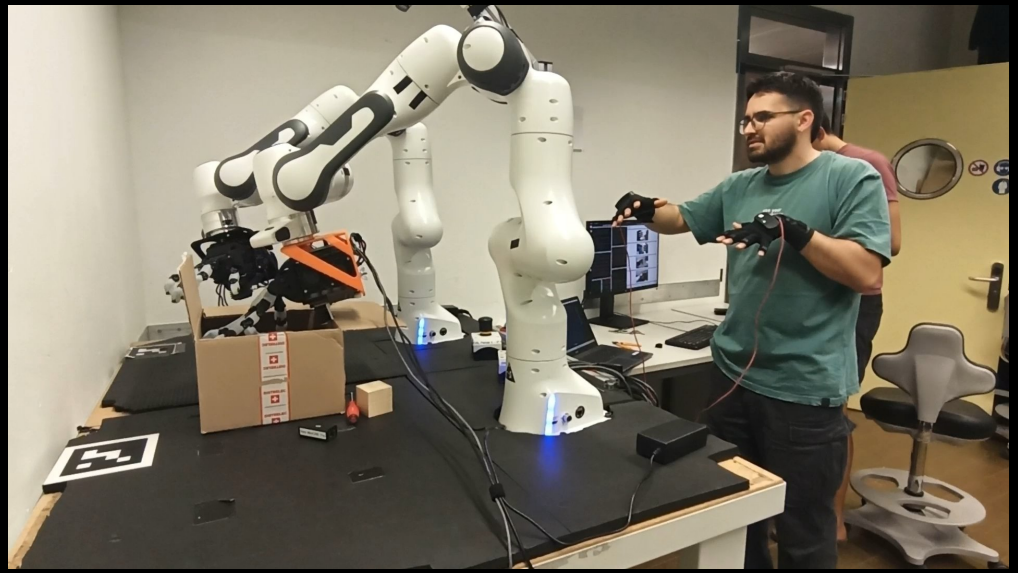
Teleoperation and Behavior Cloning

Real-World Robotics
Focus talk 2026

Human demonstrations
Robot actions
Closed-loop learning



The robot learning loop



Collect demonstrations

Synchronized observations + actions

Train a policy

Predictions change future observations

Execute on the robot

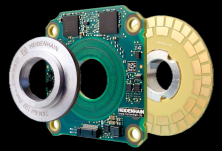
Diagnose failures

New coverage and recovery demonstrations

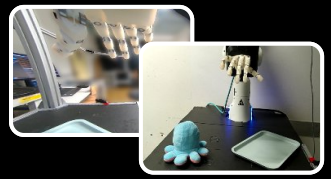
Robotics data is highly multimodal

Observations

Proprioception



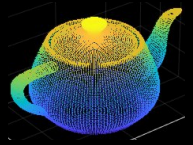
Wrist / Workspace RGBs



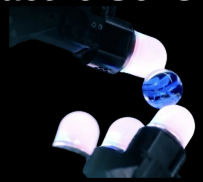
Language Command

“put the plush on the tray”

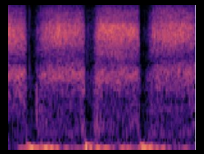
3D

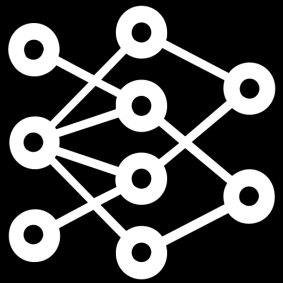


Tactile Sensing



Audio





Actions

Different Meanings



end-effector space



Joint space

Different Dimensions

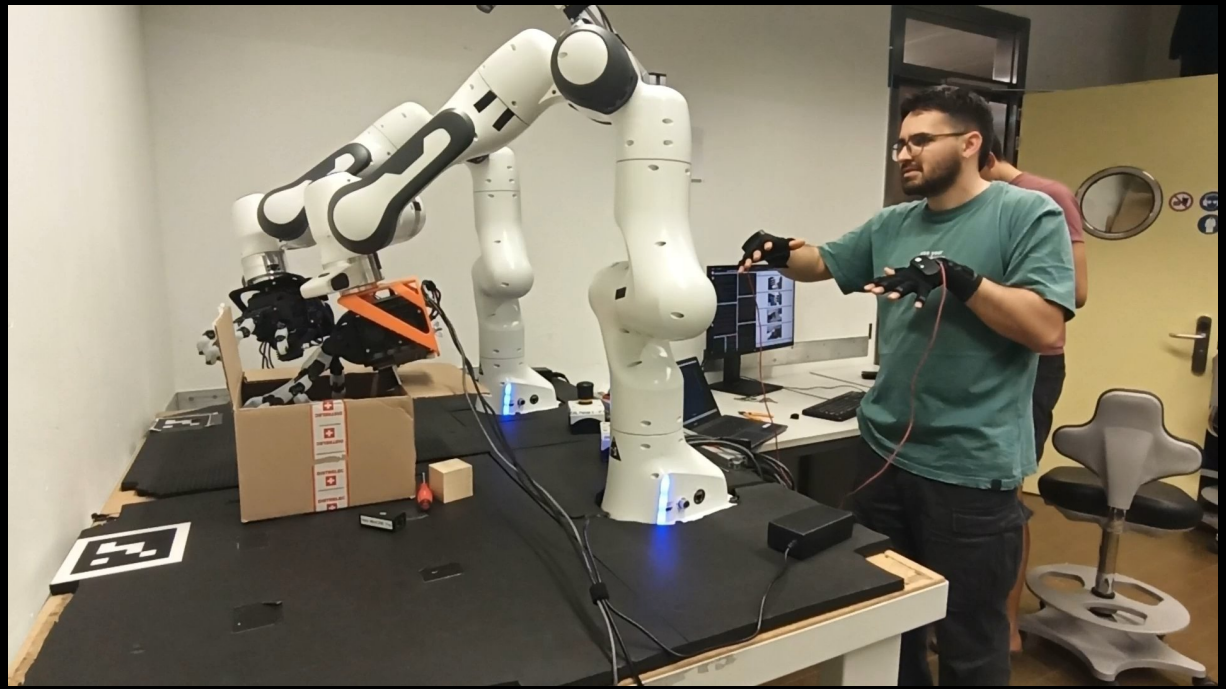






Teleoperation interfaces

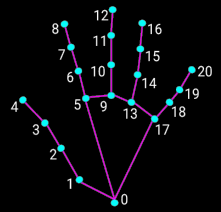
Leader–follower, retargeting, or supervisory control
Example: glove motion retargeted to robot joints



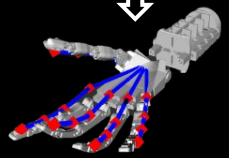
Bimanual Teleoperation with Orca Hands

Retargeting and reference frames

Indirect Teleoperation



⇩



⇩

Hand joints

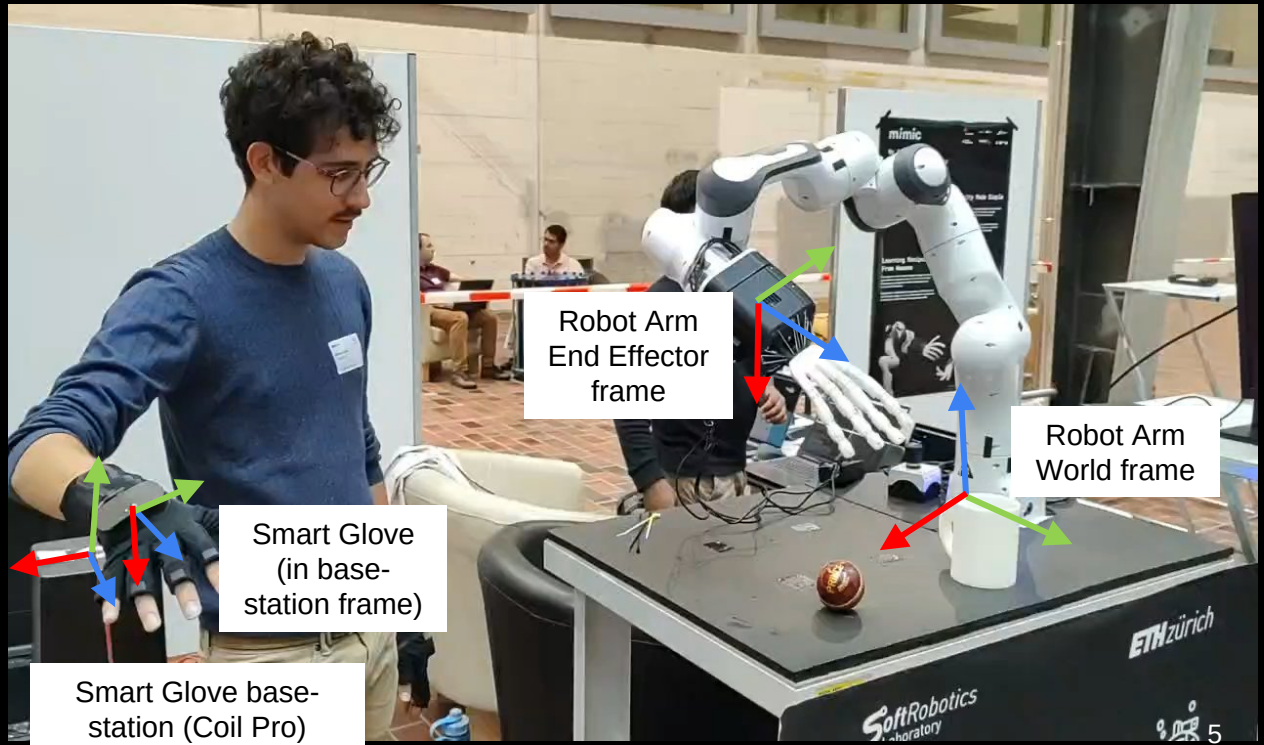
Wrist relative pose

⇩

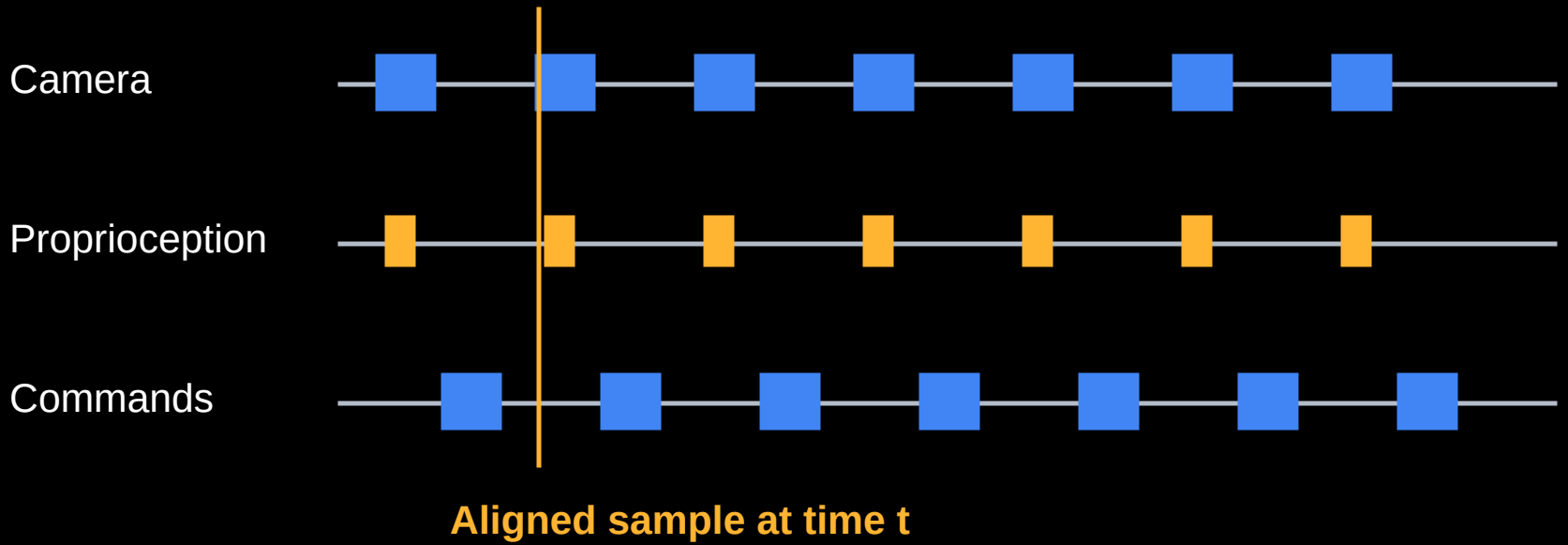
IK

⇩

Robot joints



Feedback and timing shape the training labels



Record images, measured state, issued actions, and timestamps.
Check units, reference frames, sample rates, and execution delay.

Portable data collection with UMI

UMI employs hand-held grippers coupled with interface design for portable and information-rich data collection for challenging bimanual and dynamic manipulation demonstrations.



Universal Manipulation Interface
https://umi-gripper.github.io/

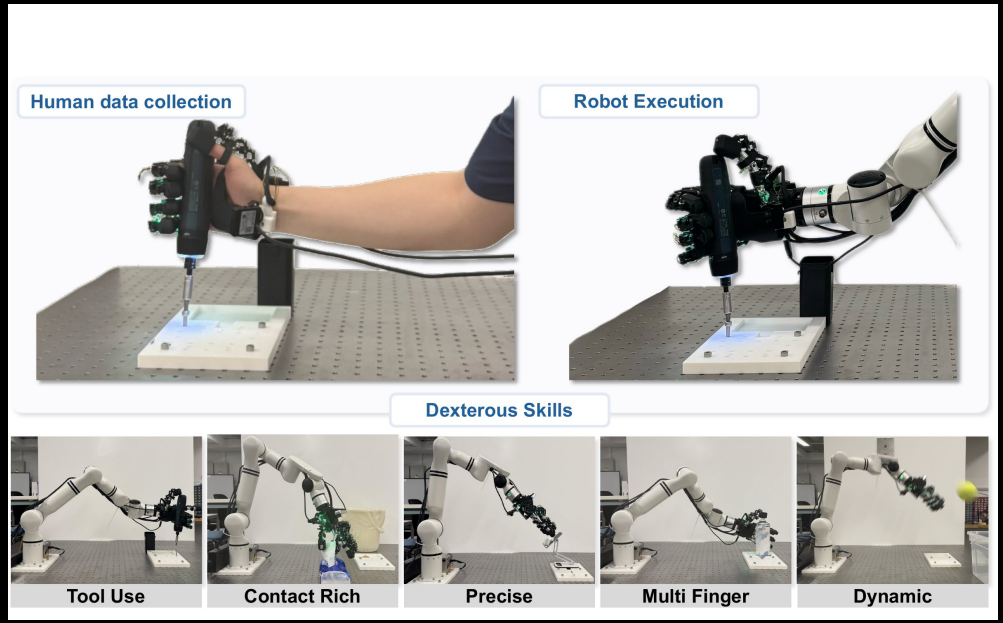
Dexterous interfaces reduce the embodiment gap

DexUMI (2025)

Wearable exoskeleton
Robot-hand image adaptation

SEED-UMI (2026)

Shared human–robot interface
Paired motion supervision



Interface design changes the learning problem.

Xu et al., DexUMI (2025)
Yu et al., SEED-UMI (2026)

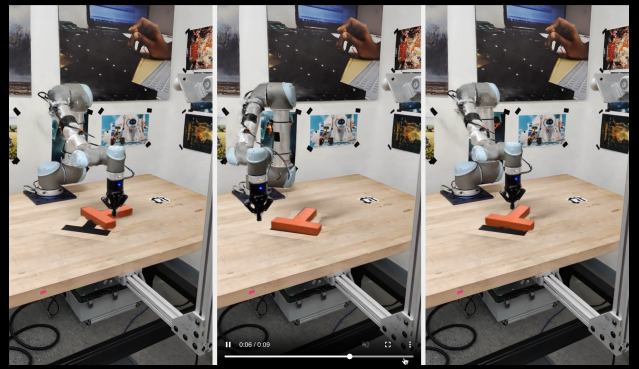
Human video and simulation complement robot data

Human video



Visual and behavioral priors
Missing robot action labels
Human–robot embodiment gap

Simulation

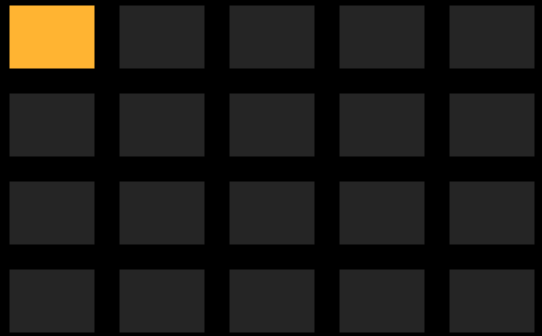


Actions, states, and variation
Parallel data generation
Physics and appearance gap

Ego4D and SplatSim (https://splatsim.github.io)

Demonstration diversity improves coverage

Repeated condition



Objects

Environments

Broader conditions



Objects

Illustrative coverage grid

Vary objects, scenes, initial states, and successful recoveries.

Lin et al., Data Scaling Laws in Imitation Learning for Robotic Manipulation, ICLR 2025

Behavior cloning learns from observation–action pairs



Training targets that come from the demonstrator



Demonstrated actions
A_t

Example regression loss: $L = \text{mean} \, \| \hat{A}_t - A_t \|_1$

Small errors change the observations the policy sees



Expert demonstrations



Covered states

Collect recovery examples
where the policy actually fails.

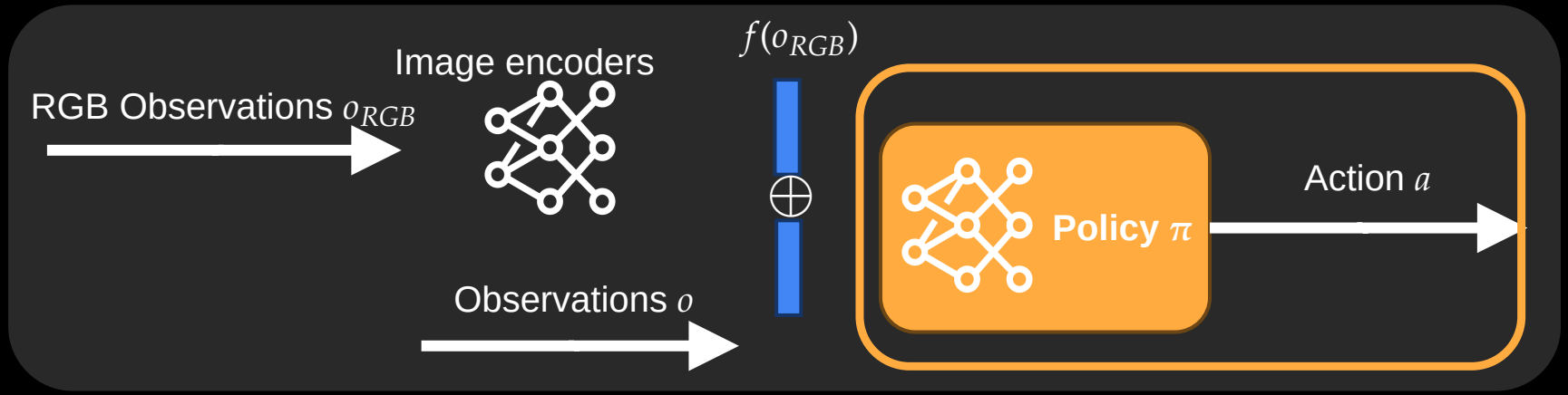
Policy rollout

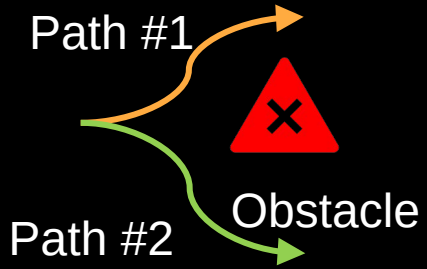


A small deviation can lead
to an unfamiliar state

Ross, Gordon & Bagnell, DAgger, AISTATS 2011

Multiple valid actions





What is the best way to predict actions?

- Multimodality: for the same observation there can be multiple valid actions
- A point prediction can average valid actions into an invalid motion.

Action chunks separate prediction from execution



Observation history O_t



Past

t

Predict H future actions

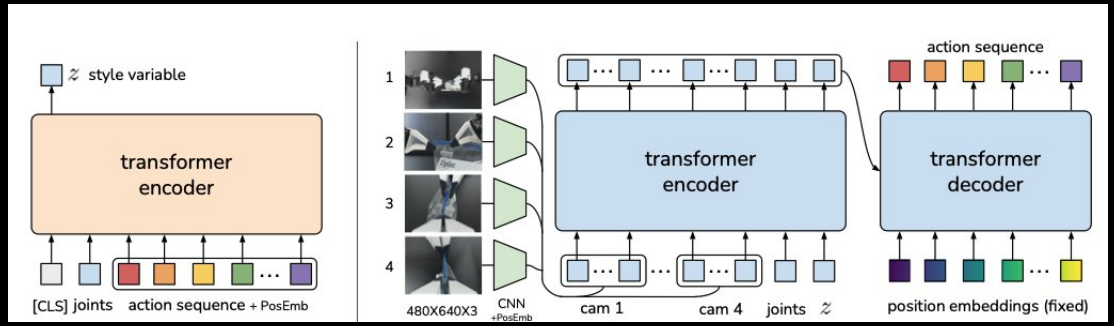


Execute h

Replan with a new observation

Longer execution horizons can delay reactions to new observations.

ACT predicts a chunk with a conditional VAE



Training

Encode expert actions
Decode a future chunk
L1 reconstruction + β KL

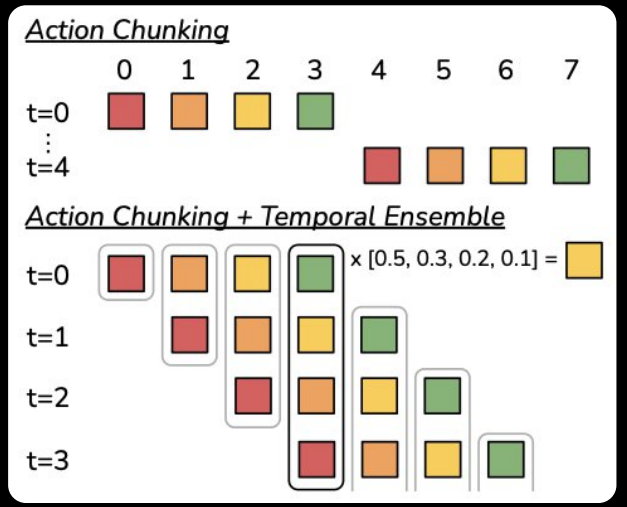
Inference

Set latent z = 0
Use current observations

The transformer models dependencies across future actions.

ACT : Temporal Ensemble

- Naïve implementation of action chunking can be suboptimal
- If new observation is incorporated abruptly every k steps this can result in jerky robot motion



Temporal ensemble

Query the policy at every timestep, aggregate them with a weighted exponential average

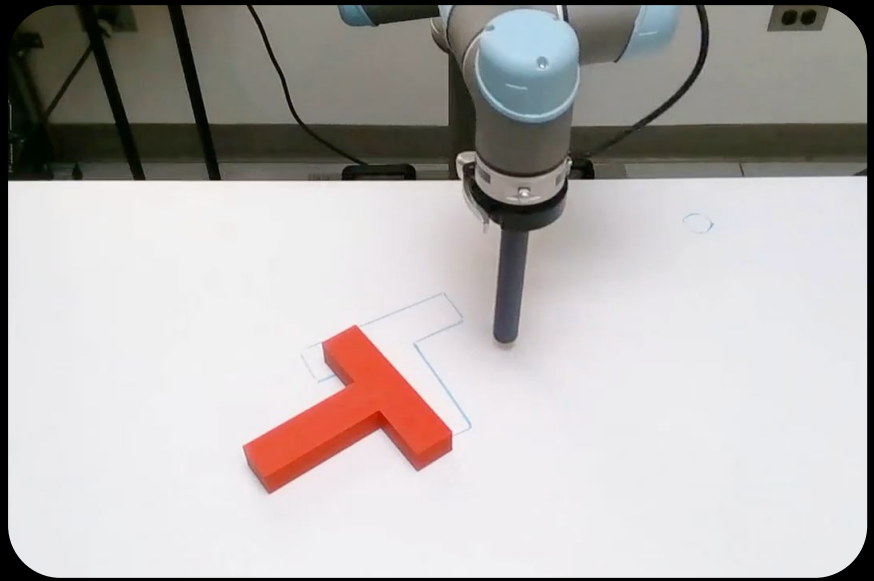
$$w_i = \exp(-m * i)$$

Diffusion Policy

Apply diffusion process to robot action prediction

Denoising actions

- Train: add noise to expert action chunks
- Predict noise conditioned on observations
- Sample: denoise, then execute a prefix



Flow matching learns a velocity field for actions

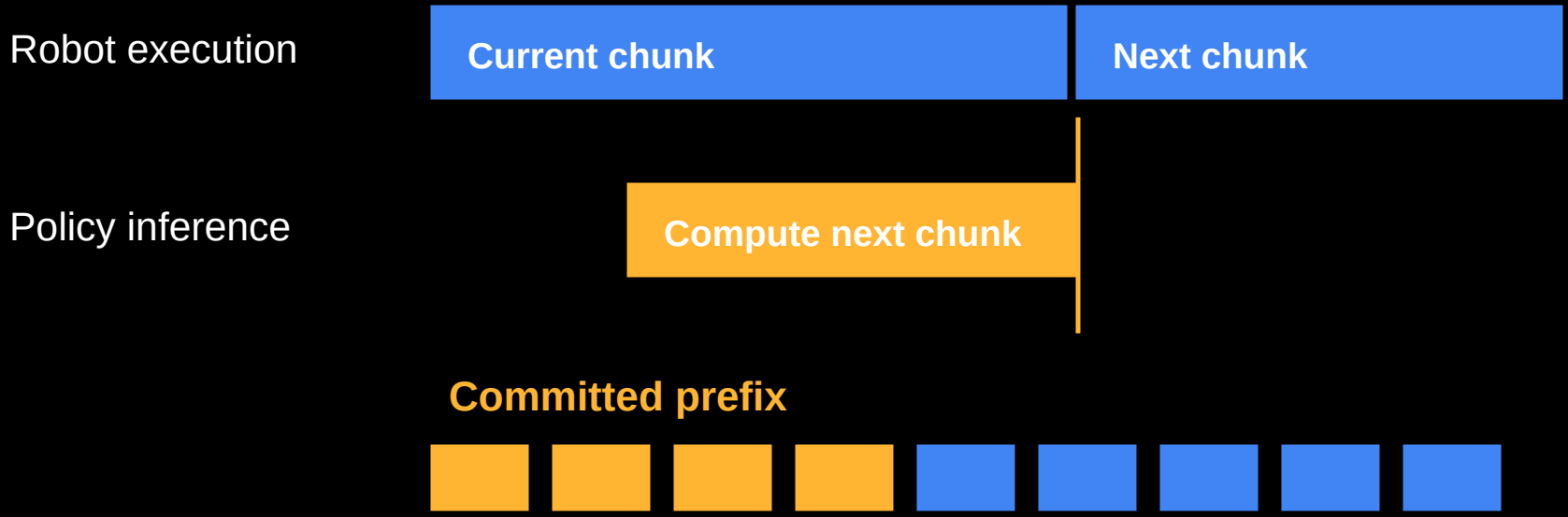


$X_s = (1 - s)\,Z + s\,A$ target velocity $= A - Z$

Training path shown. At inference, integrate the learned velocity field.

Lipman et al., Flow Matching for Generative Modeling, ICLR 2023

Real-time execution must account for inference delay



RTC: preserve committed actions and generate a continuation.
REMAC: learn corrections for perception–execution mismatch.

Black et al., RTC, NeurIPS 2025
Wang et al., REMAC, ICLR 2026

Rollouts reveal what offline loss cannot



Test condition	Observed failure	Next intervention
Changed object	Grasp fails	Add object diversity
Displaced start	No recovery	Collect recovery demos
Fast contact	Overshoot	Check timing + control

Evaluate → **Diagnose** → **Recollect**

Your team’s demonstration-to-policy system



Data → **Policy** → **Evaluation**

What can the robot observe?
Which actions can it execute?

How are alternatives represented?
When does it replan?

Which conditions are held out?
What will you collect next?

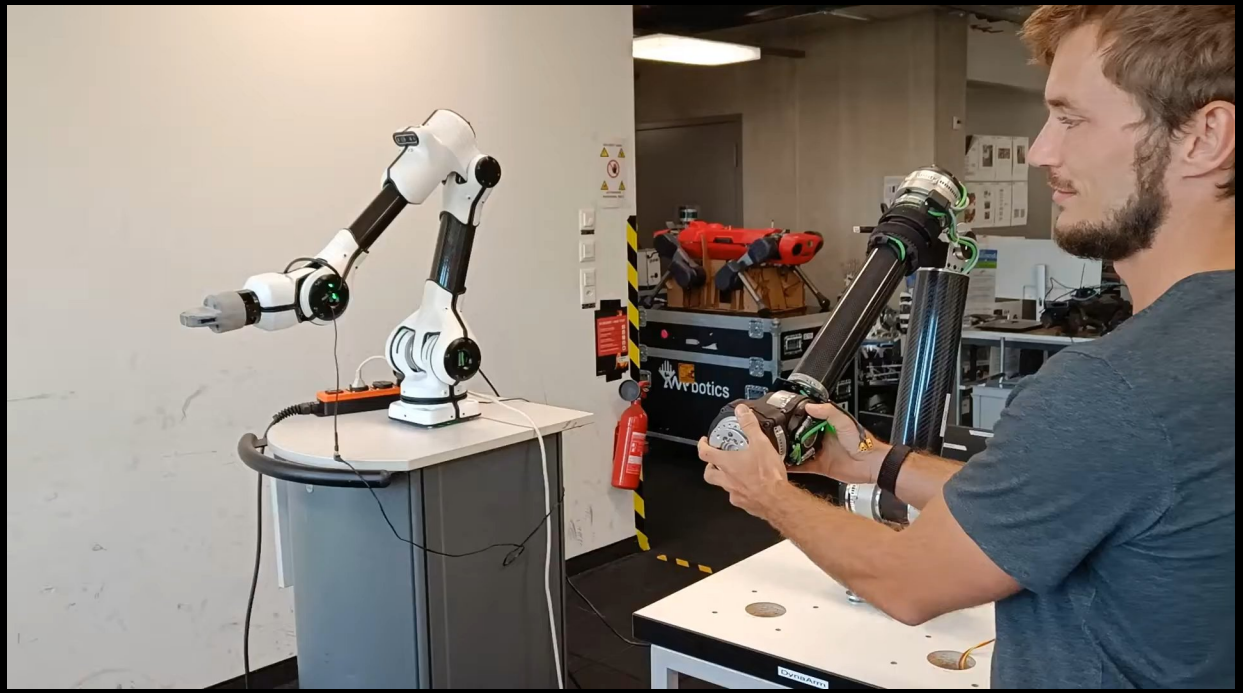
Next: language-conditioned, multi-task robot policies

Questions?

Teleoperation

Direct Teleoperation

Leader-Follower systems, directly send joint values to the other robot



Duatic AG
https://duatic.com/

Teleoperation

High Level Teleoperation

Human gives *goal- or task-level commands* (e.g. keypoints), robot handles low-level control.



Cybathlon 2024 Competition
https://cybathlon.com/en/events/edition/cybathlon-2024

Teleoperation



Haptics

Including tactile and kinesthetic feedback for more accurate and intuitive control



https://haptx.com

Simulation Data

- Don’t need physical access to robots, can collect data anywhere.

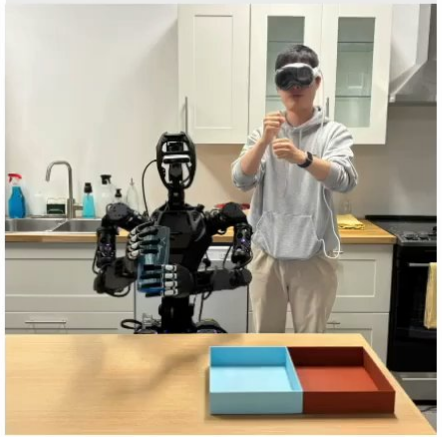


DART: Dexterous Augmented Reality Teleoperation Platform for Large-Scale Robot Data Collection in Simulation
https://openreview.net/pdf/b0393e5ce2ed5eab3b89e05d5e6855b36ef4875f.pdf

Simulation Data

Simulation can be used for data augmentation

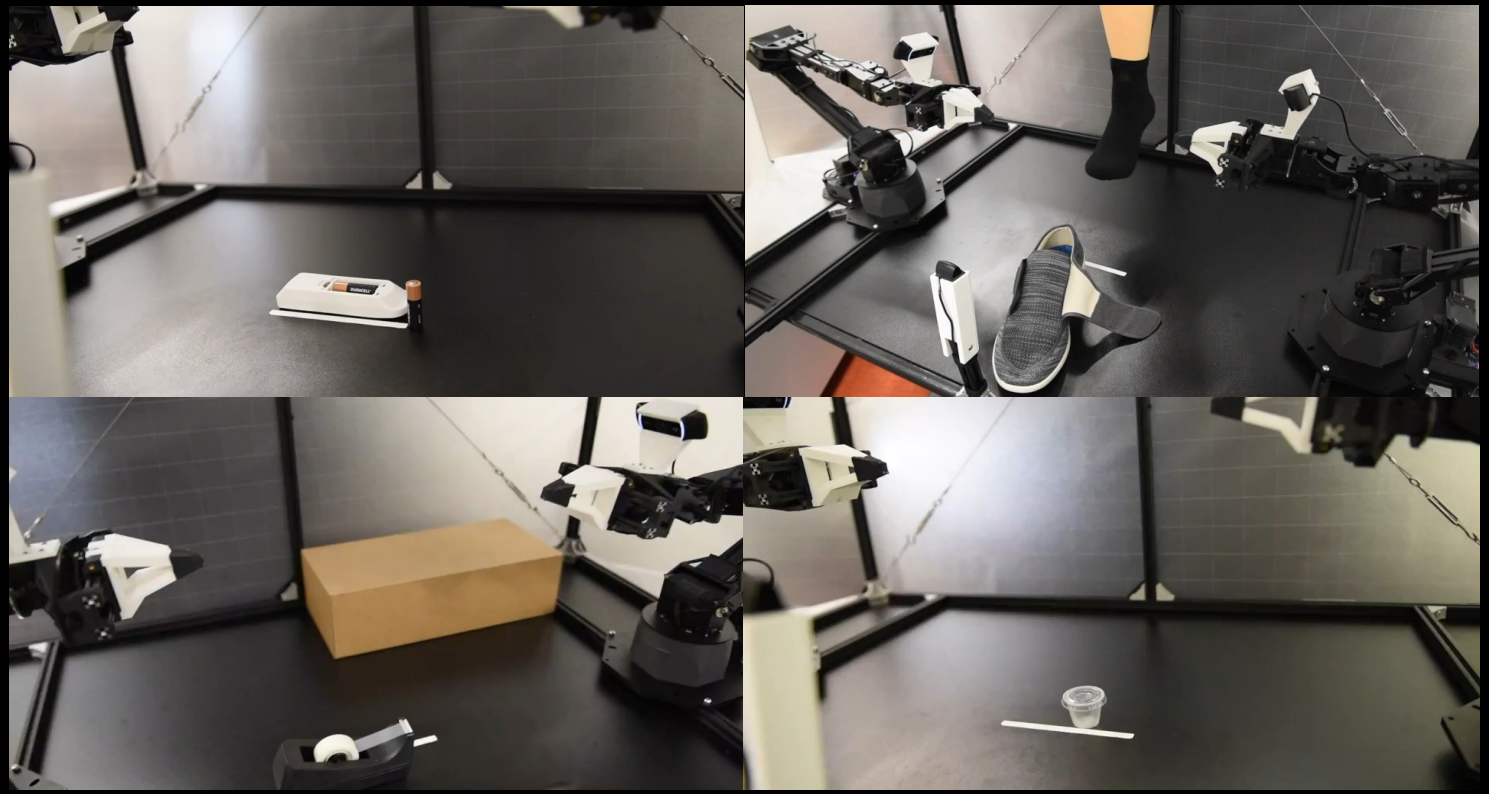
Human teleoperation



DexMimicGen: Automated Data Generation for Bimanual Dexterous Manipulation via Im

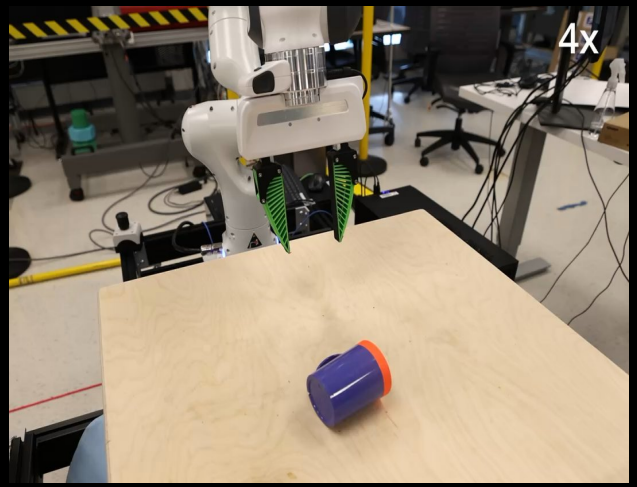
https://arxiv.org/pdf/2410.24185

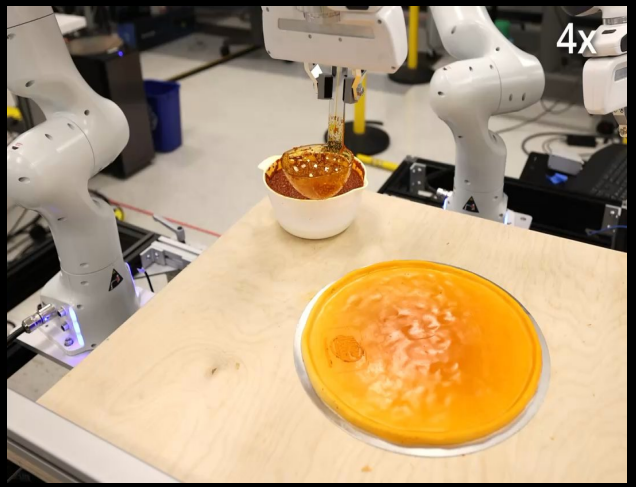
ACT (Action Chunking Transformers)

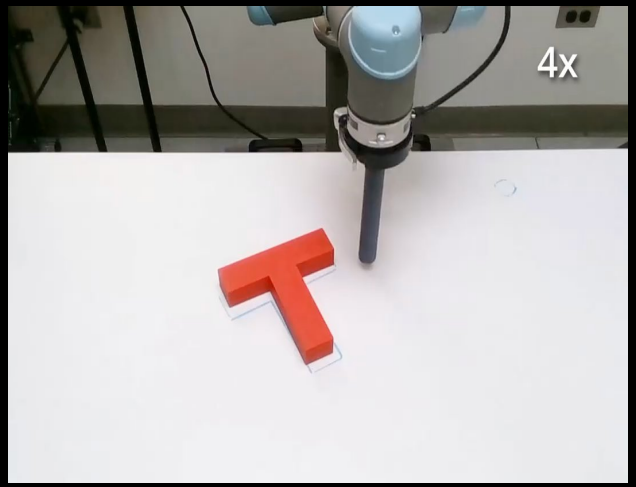


Zhao, T. Z., Kumar, V., Levine, S., Finn, C.

Diffusion Policy



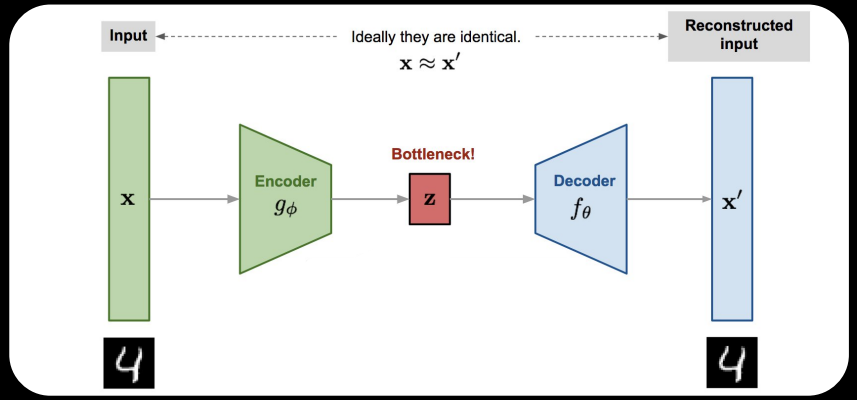




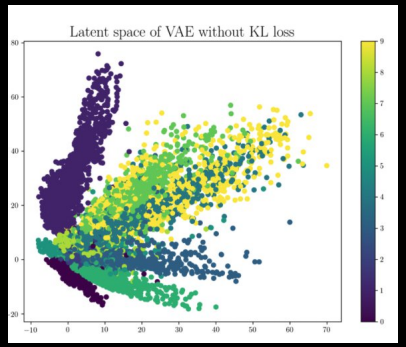
Chi, C. et al. (2023). *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion*

ACT : VAEs

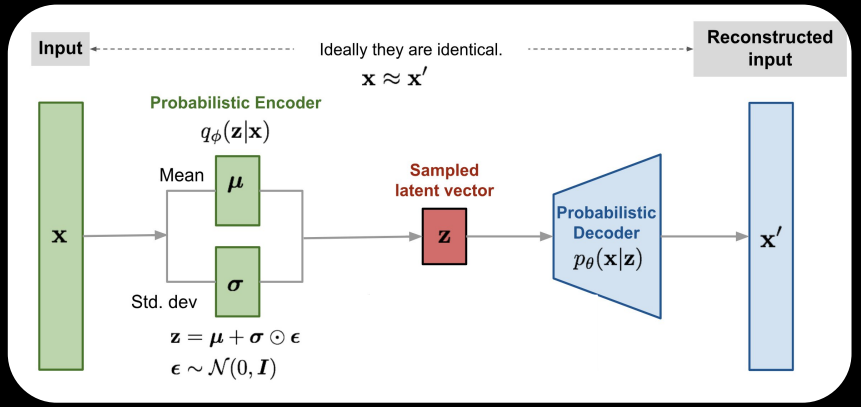
AutoEncoder



$$L = \|f(g(x)) - x\| = \|x\prime - x\|$$



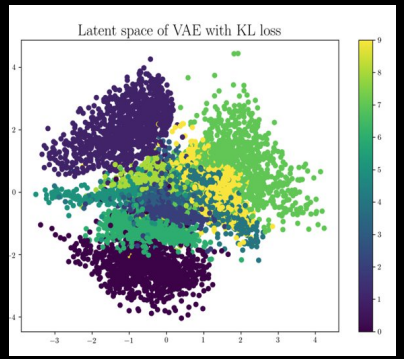
Variational Autoencoder



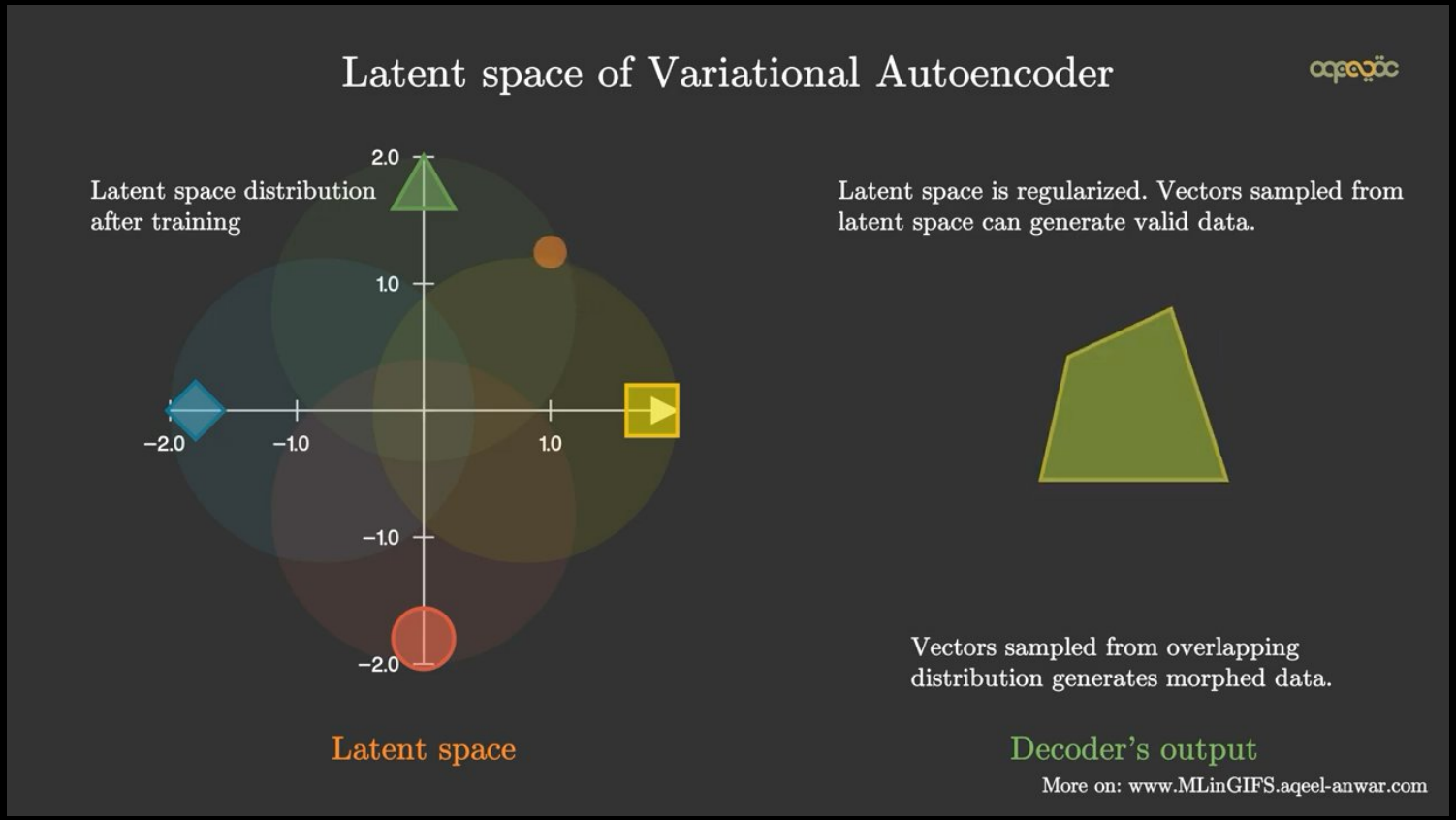
reconstruction

$$\mathbf{L} = \|x\prime - x\| + D_{KL}\left(N\left(\mu_{x\prime}, \sigma_x\right) \mid N(0,1)\right)$$

regularization

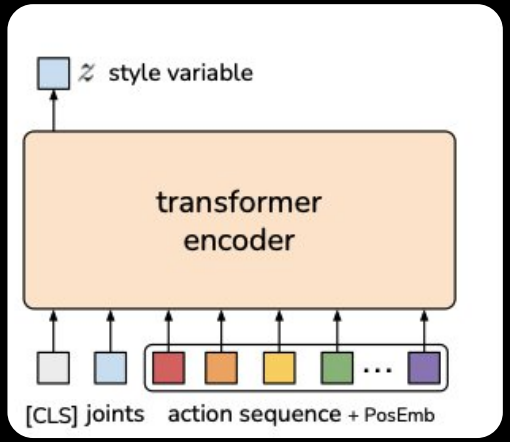


ACT : VAEs


Latent space of Variational Autoencoder
Latent space distribution after training
2.0
1.0
−2.0
−1.0
1.0
−1.0
−2.0
Latent space is regularized. Vectors sampled from latent space can generate valid data.
Vectors sampled from overlapping distribution generates morphed data.
Latent space
Decoder's output
More on: www.MLinGIFS.aqeel-anwar.com

ACT : CVAE

Train the policy as a conditional variational autoencoder

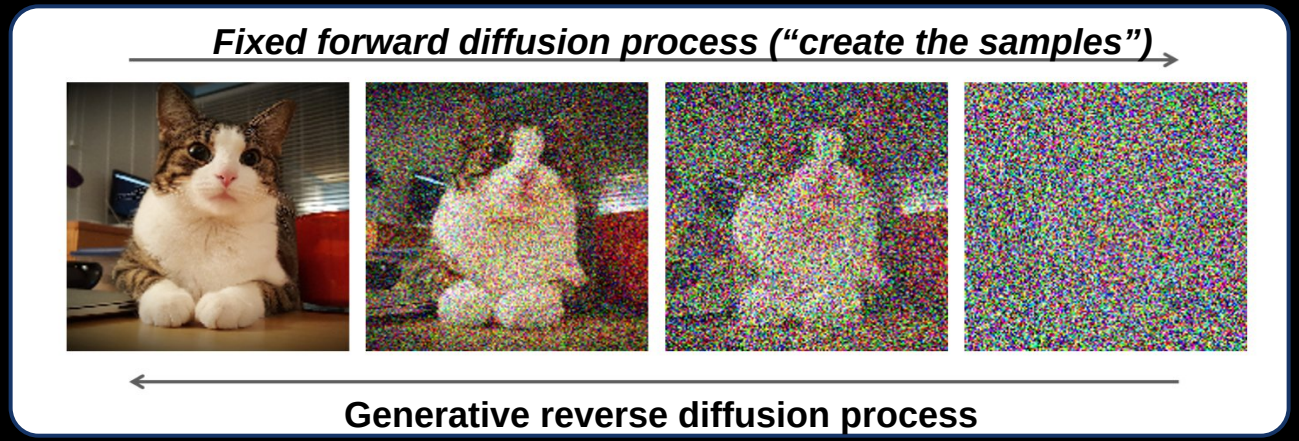


At training time, we encode the action sequence and the proprioception using a transformer

The output is the encoded representation $z = q_\phi(z \mid a_{t:t+k}, \overline{o_t})$

$\overline{o_t} = o_t - o_{RGB}$

Diffusion Policy – Diffusion Process



1. Sampling $x_0 \sim \mathfrak{D}$, noise level $\sigma \sim [\sigma_{min}, \sigma_{max}]$, noise $\epsilon \sim N(0, I)$
2. Generating noisy data $x^\sigma = x^0 + \sigma\,\epsilon$
3. Predicting ϵ (direction of noise) from x^σ by minimizing squared loss

This amounts to training a θ-parametrized neural network $\epsilon_\theta(x, \sigma)$, by minimizing the loss function $L(\theta) = \mathbb{E}\left\|\epsilon_\theta\left(x^0 + \sigma_t\epsilon, \sigma_t\right) - \epsilon\right\|^2$

Diffusion Policy – Diffusion Process



How do we add noise?

- In practice, σ is not sampled uniformly in the interval $[\sigma_{min}, \sigma_{max}]$, but the interval is discretized into K values called ***schedules*** and then sampled uniformly between the schedules.
- If we now follow the noise schedule, we can find the denoising formulation
- A well-known scheduler is the **ScheduleDDPM**

  (Denoising Diffusion Probabilistics Models)

