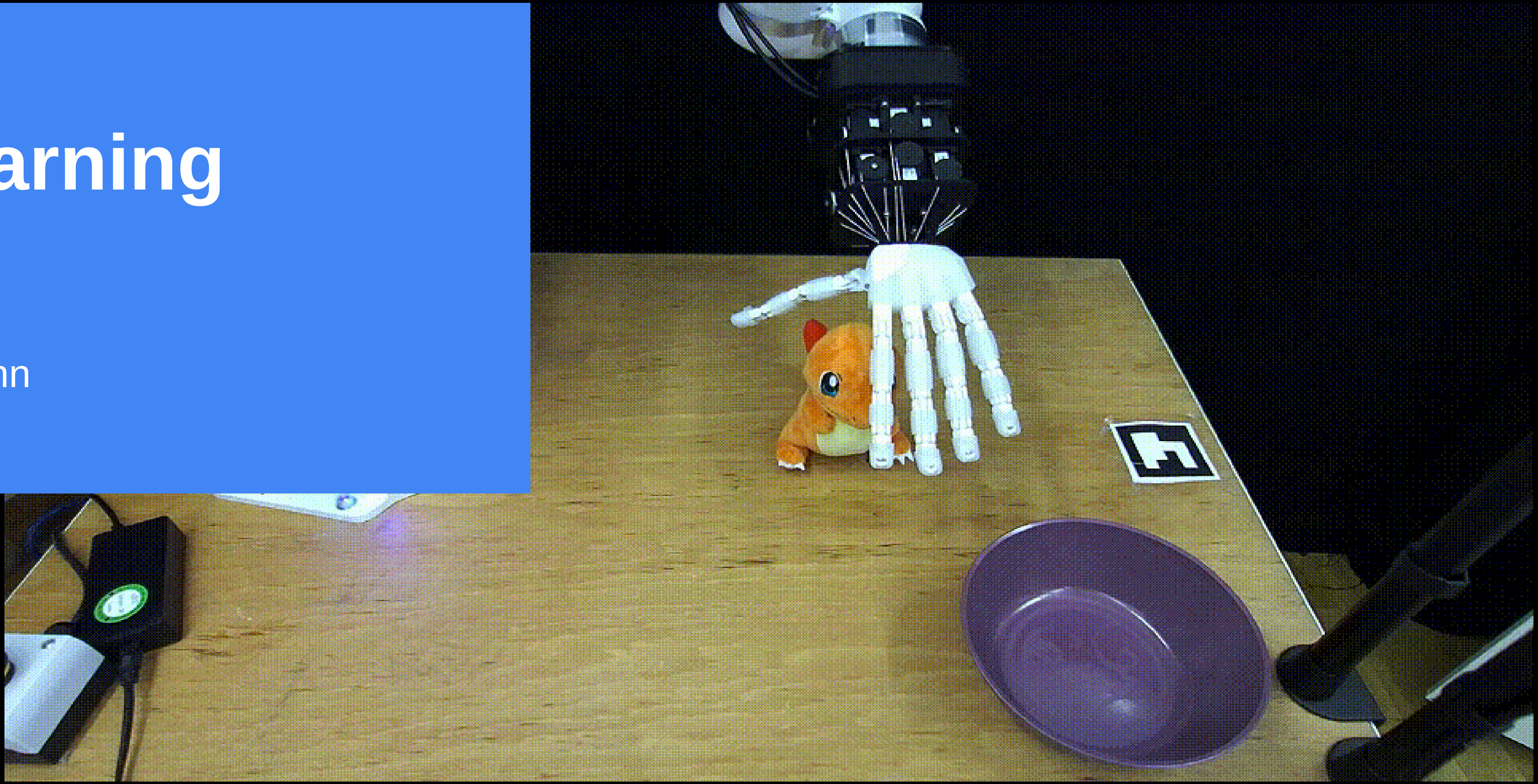
ETH zürich

SoftRobotics Laboratory

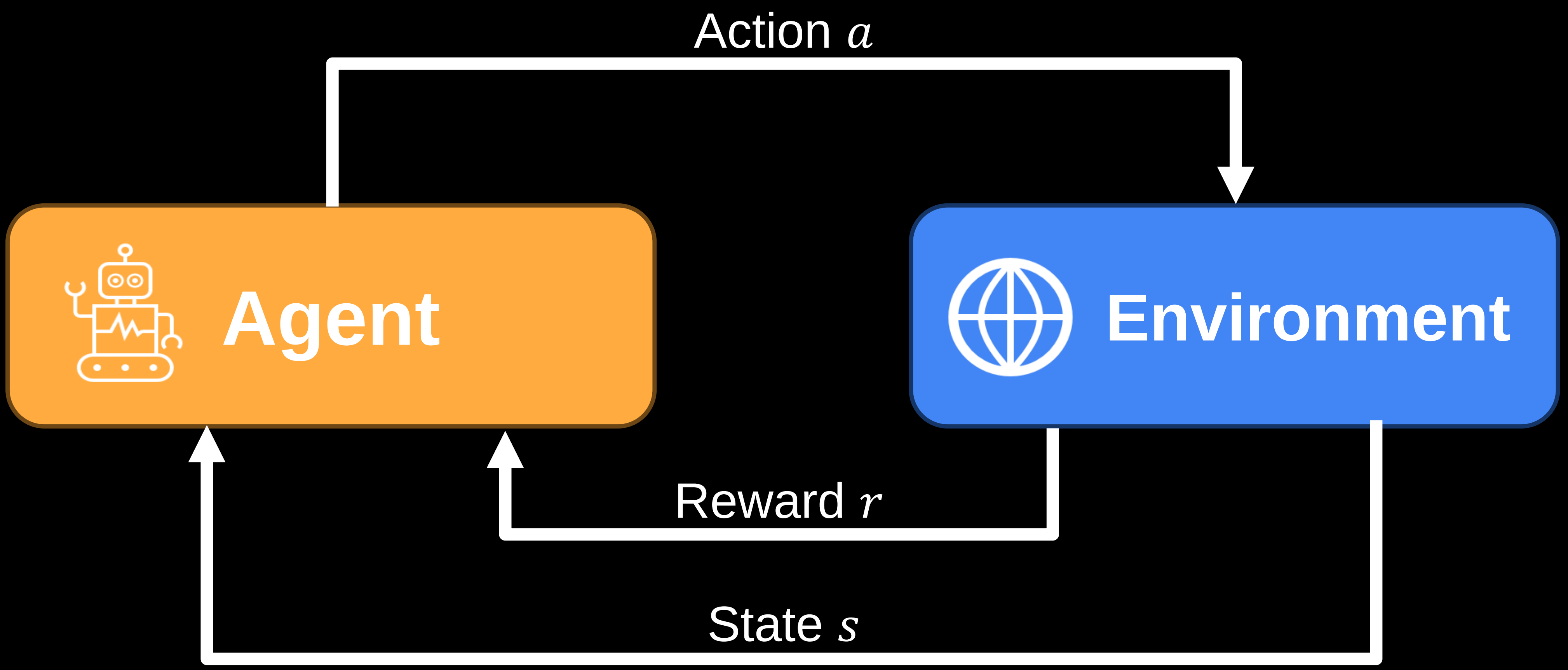
	17 Nov	24 Nov	1 Dec
14:15-15:00	**Imitation Learning** *Davide Liconti*	**Guest Lecture** *Oier Mees @ Microsoft Spatial AI Lab*	**Product Development** *Benedek Forrai @ Mimic Robotics*
15:15-16:00	**Training Policies with SRL_IL (workshop)** *Chenyu Yang*	**VLAs and Foundation Models** *Davide Liconti*	**Class Wrap-up and Q&A** *Robert Katzschmann*

Imitation learning

Davide Liconti
Prof. Robert Katzschmann
17 November 2025



Reinforcement Learning (RL) Recap



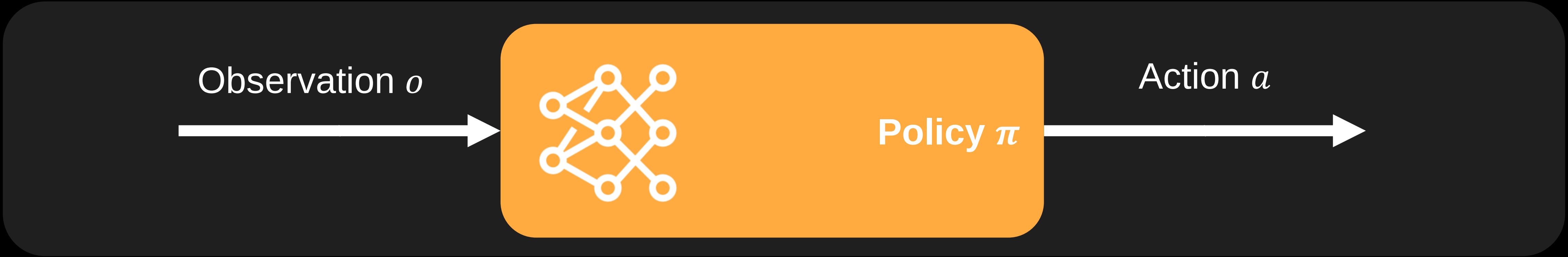
How to define the reward?

What’s the reward function for this task?

- ***Sparse reward***: +1 when tied (RL won’t explore enough).
- ***Dense reward***: but what is the distance to “tiedness”?
- Multi-step, long-horizon manipulation → reward shaping quickly becomes *arbitrary* or *impossible*.

Imitation Learning (IL)

aka Behavior Cloning (BC), aka Learning from Demonstrations (LfD)



Given a dataset of "Expert transitions"

$$\mathfrak{D} = \{(o_t, a_t)\}_{t=1}^{N}$$

Simple supervised regression : $\pi = \arg\min \ \mathbb{E}_{(o,a)\sim\mathfrak{D}}\left[\|a - \pi(o)\|^2\right]$

How to collect data? → previous lecture

Imitation Learning (IL)

why does it dominate today

Today **IL** is by far the dominant approach for manipulation

- **No** complex or arbitrary **reward shaping**
- **No sim2real gap** (for real world teleop data)
- Simpler, easier to debug and interpret
- Can theoretically **scale** with data and compute

LLMs (e.g., GPT) are trained in a similar form as IL.
They are trained to predict the next token given a ”context” of recent tokens.

Why is learning actions different than learning next token?

Imitation Learning (IL)

end-to-end learning

Traditional Perception + Control

Observations o → **State Estimation** → Robot States s_r + Environment States s_{env} → **Control Policy π** → Action a

End-to-end Policies

Observation o → **Policy π** → Action a

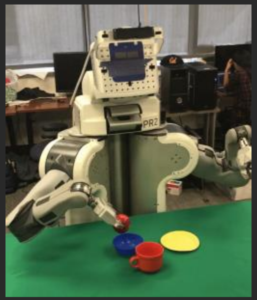
Visuomotor Policies

End-to-end Policies

Observation o → **Policy π** → Action a

Types of Observations

RGB Images



Others

- *Depth*
- *Tactile*
- ***Proprioception***
- *etc.*

Issues with Observations

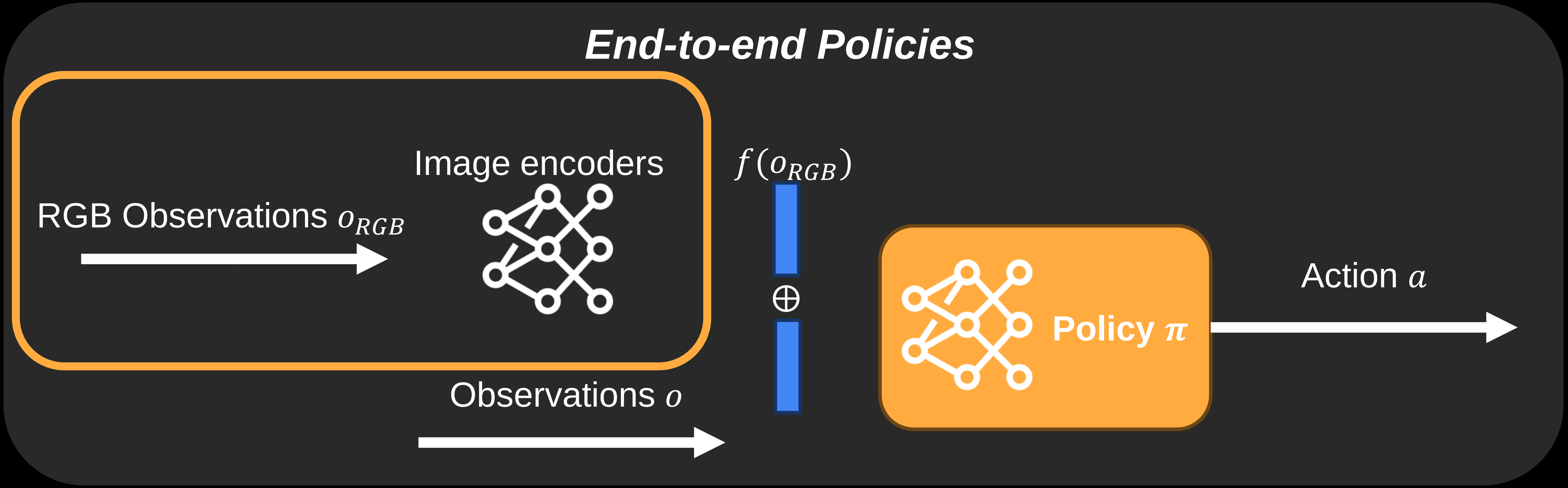
Pixel-space is a terrible space to do control

- High dimensional
- Discrete and non-differentiable

Need latent, meaningful features

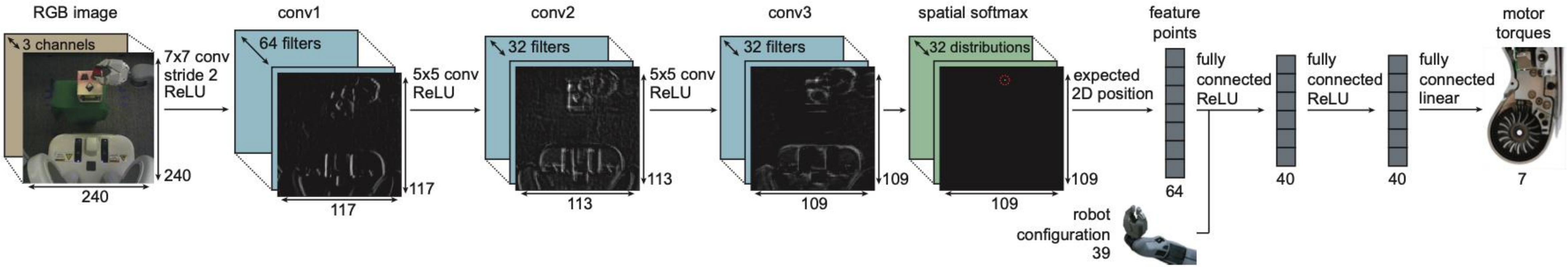
ETH zürich SoftRobotics Laboratory

Visuomotor Policies: Encoding Observations



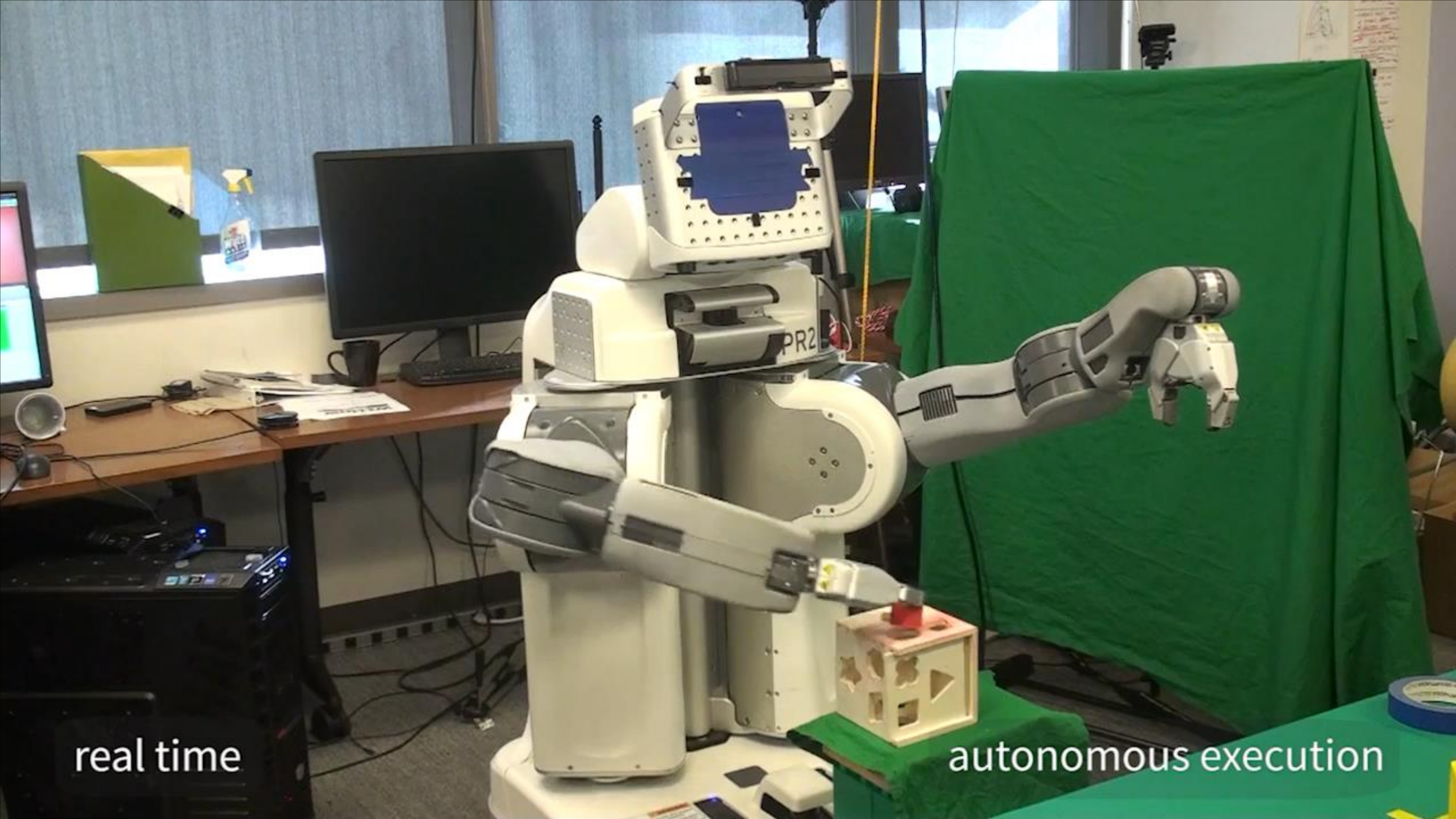
Deep Visuomotor Policies

Sergey Levine*, Chelsea Finn*, Trevor Darrell, Pieter Abbeel, "End-to-end training of deep visuomotor policies", *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334--1373, 2016.



The name "visuomotor" is chosen to emphasize that the policy is trained to predict actions directly from RGB camera images

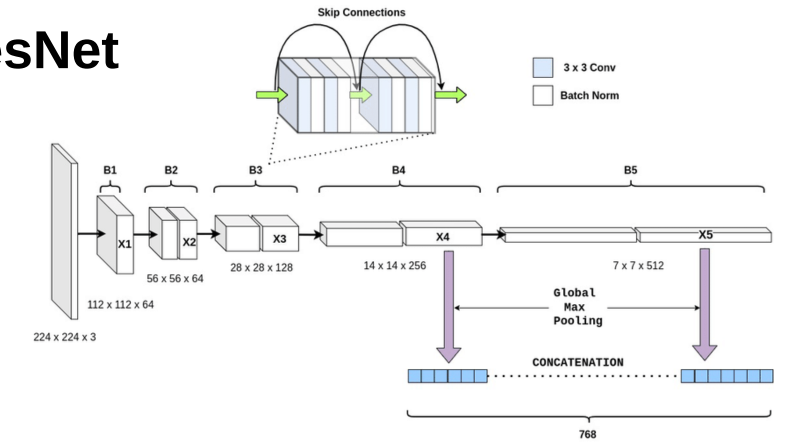
The action space was directly motor torques. Today, higher level actions are used, and let IK + low-level control transfer to torques


PR2
real time
autonomous execution

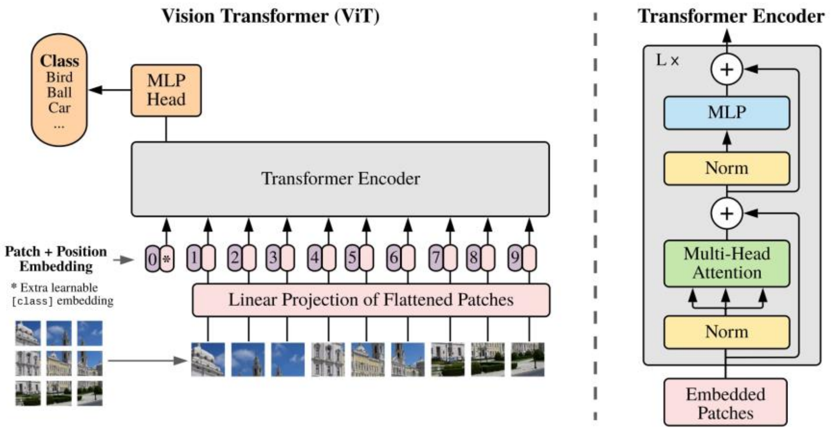
Visual Encoders Trained on Large Data

Today, a lot of visual encoders trained on large image datasets

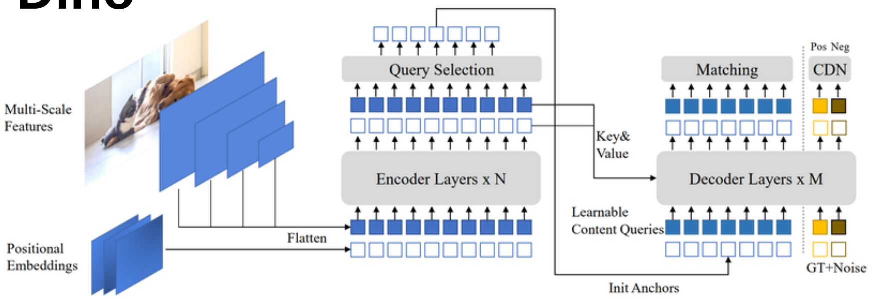
ResNet



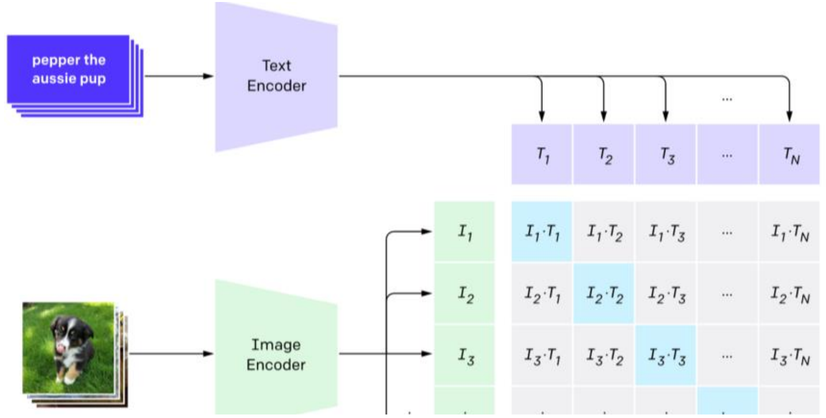
ViT



Dino



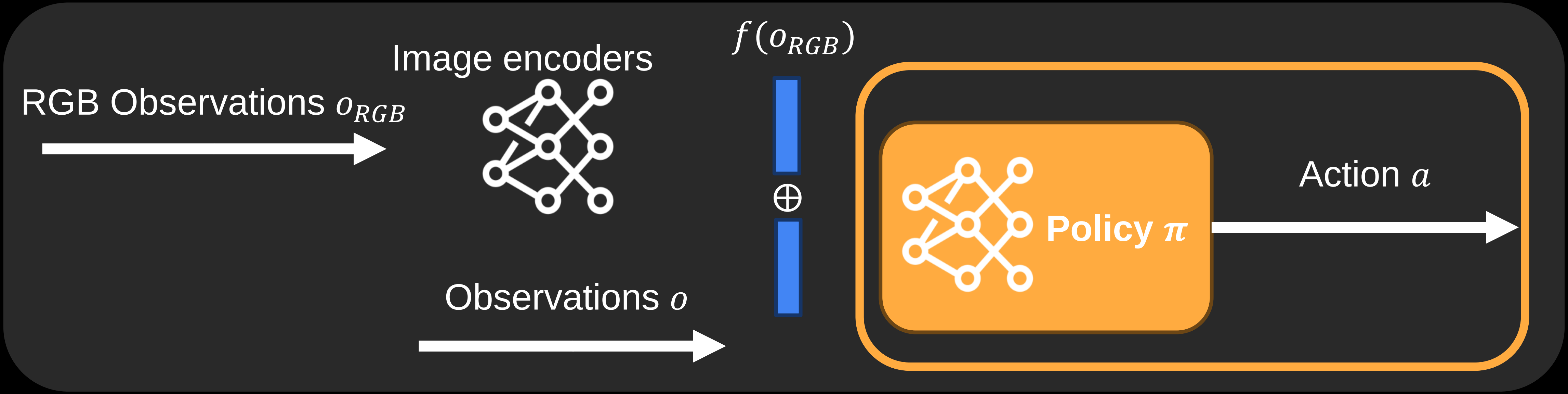
CLIP



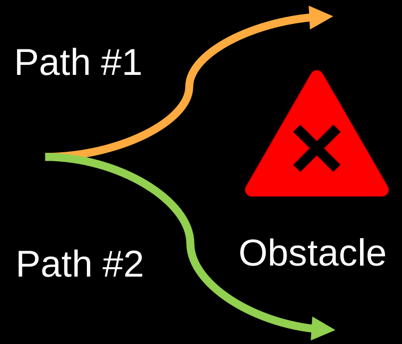
We assume $f(o_{RGB,t}) = \text{image\_encoder}\, o_{RGB,t}$.

The policy maps π: $f(o_{RGB}) + o \rightarrow a$

Visuomotor Policies : Predicting Actions

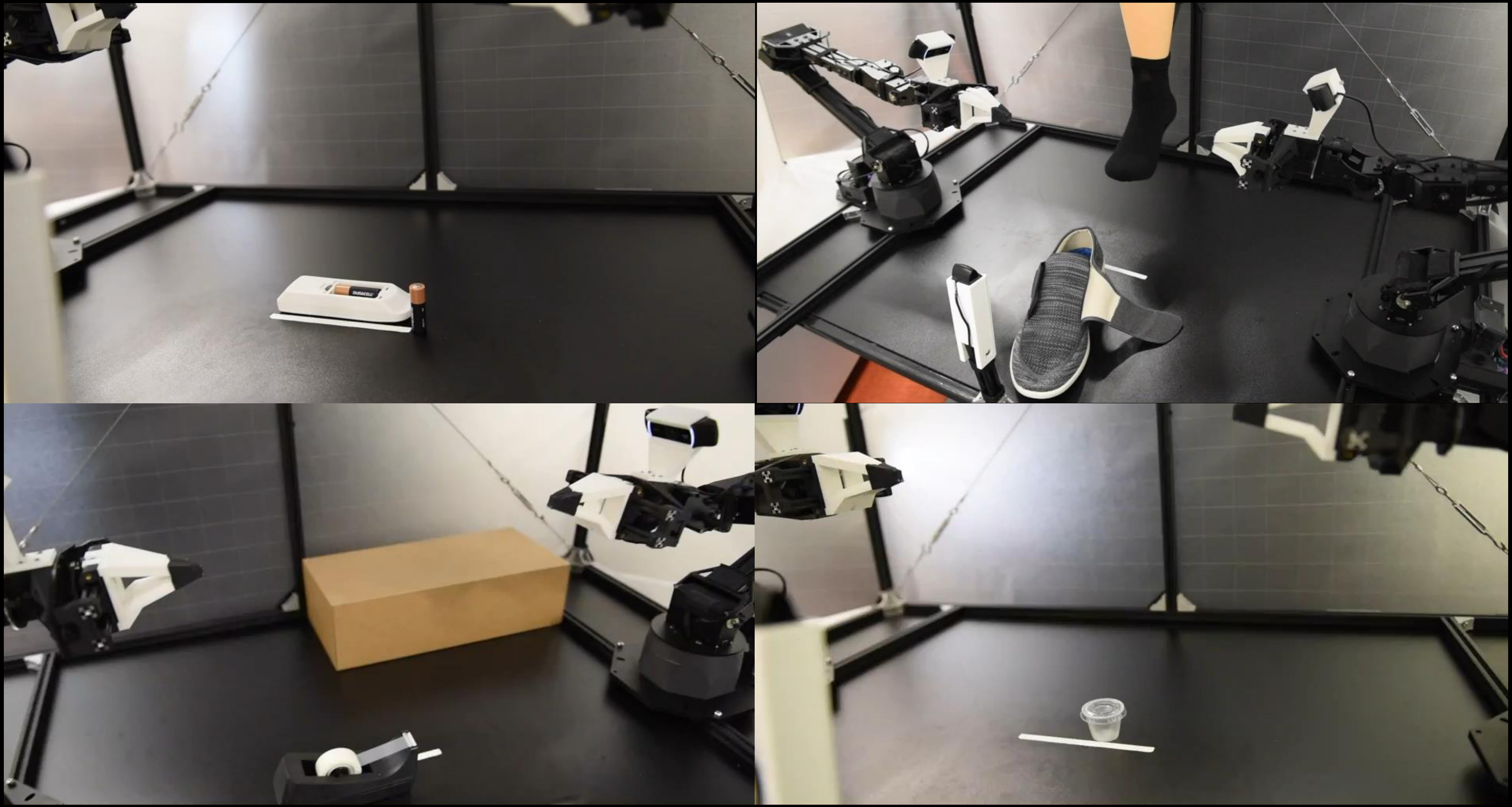


What is the best way to predict actions?



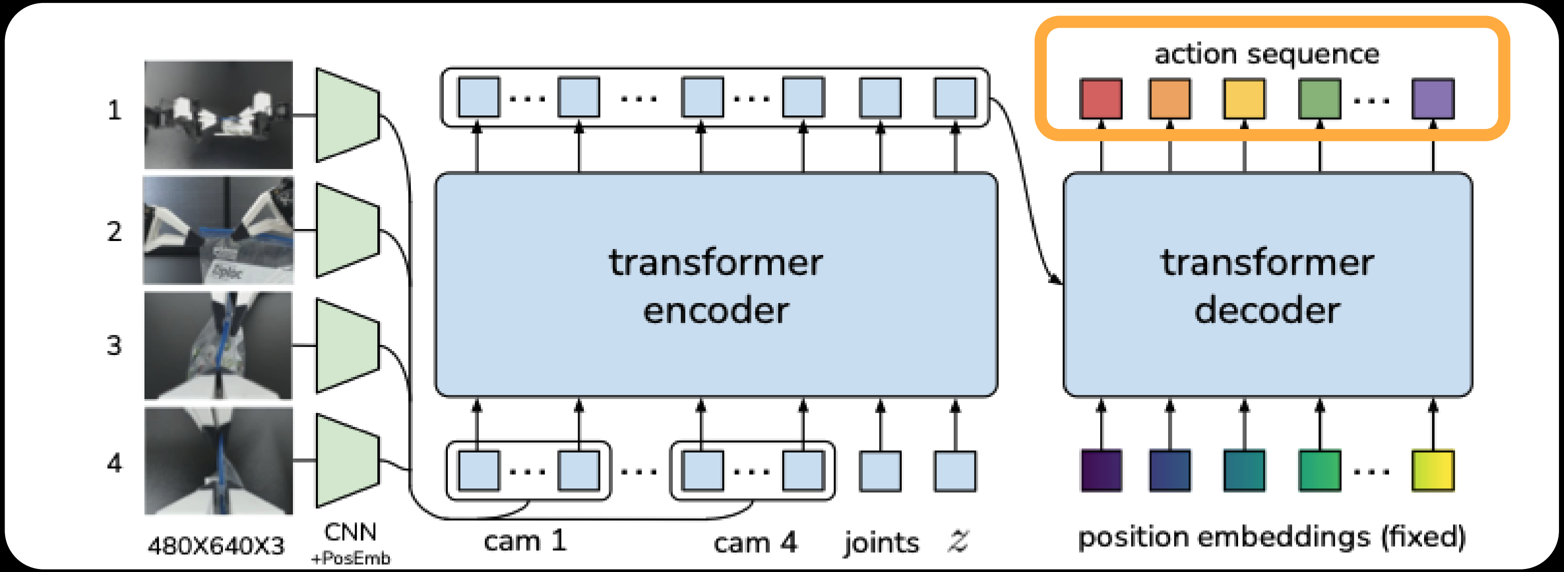
- Multimodality: for the same observation there can be multiple valid actions
- In 2023, ***ACT*** (Action Chunking Transformers) and ***Diffusion Policy*** showed convincing capabilities to learn complex dexterous actions

ACT (Action Chunking Transformers)



Zhao, T. Z., Kumar, V., Levine, S., Finn, C. (2023). *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*

ACT (Action Chunking Transformers)



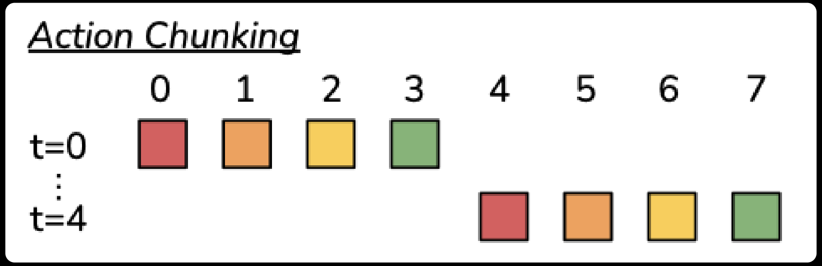
Predict a ***chunk*** of ***k*** actions, instead of a single step

$$\pi_\theta(a_t|o_t) \rightarrow \pi_\theta(a_{t:t+k}|o_t)$$

- Avoid compounding error (wrong predictions quickly leads to out of distribution)
- Shorter horizon control
- Help model non-Markovian behavior (e.g., pauses in the middle of a demonstration)

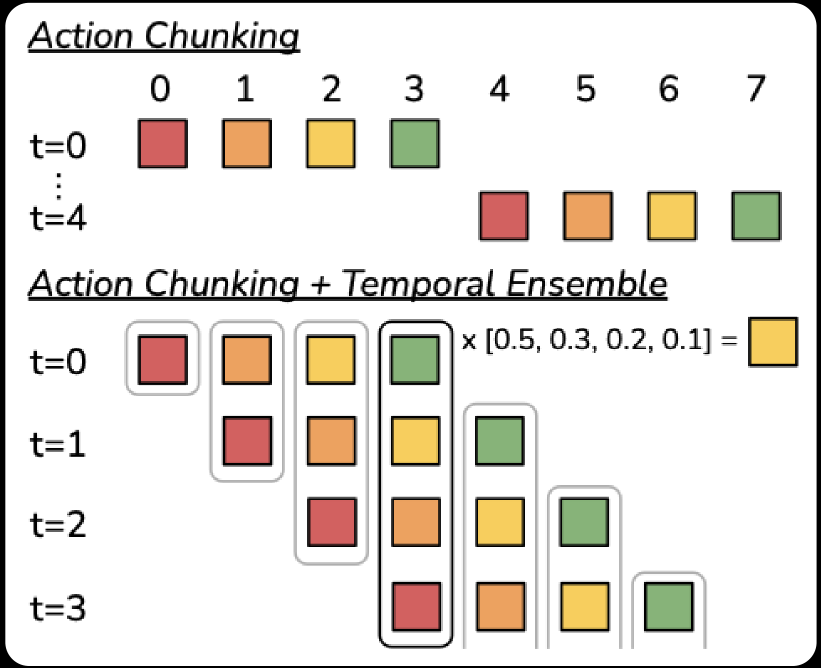
ACT : Temporal Ensemble

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



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

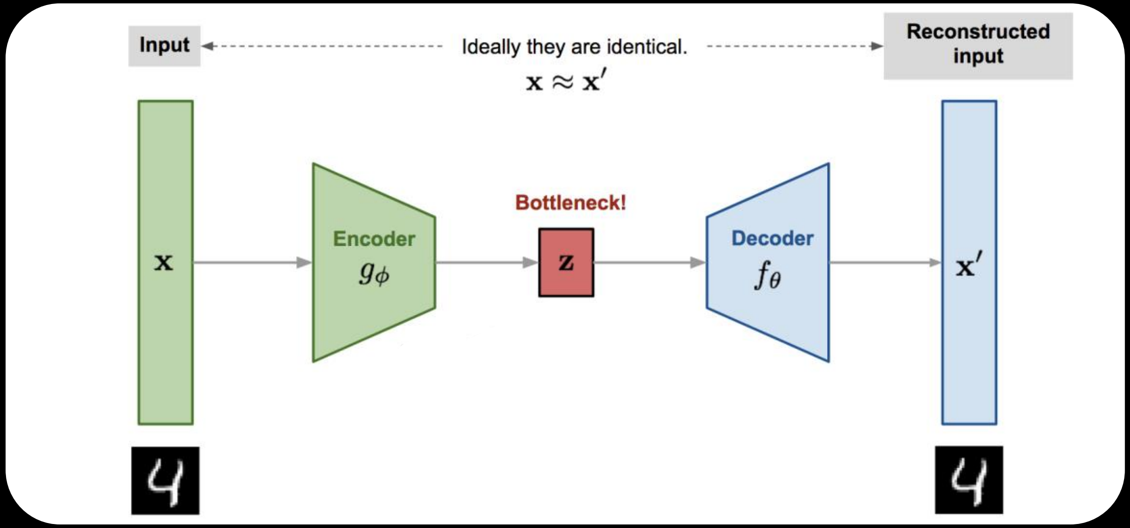
Query the policy at every timestep, aggregate them with a weighted exponential average

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

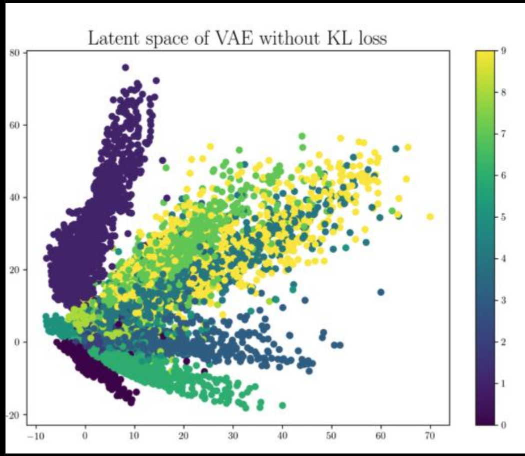
ETH zürich SoftRobotics Laboratory

ACT : VAEs

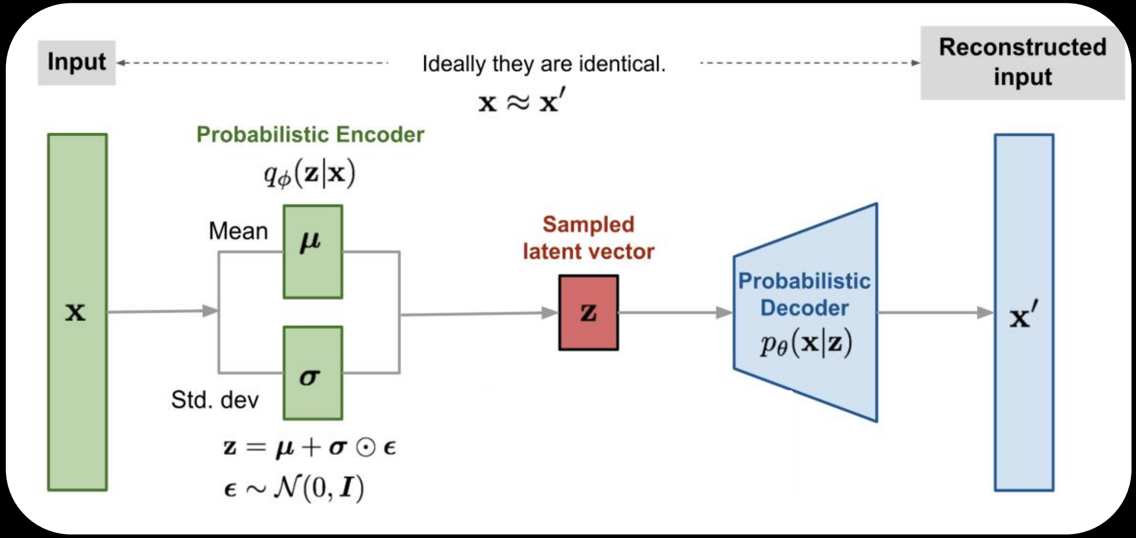
AutoEncoder



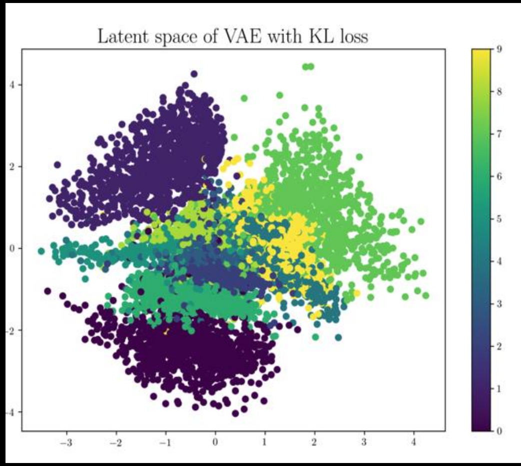
$$L = \|f(g(x)) - x\| \\ = \|x' - x\|$$



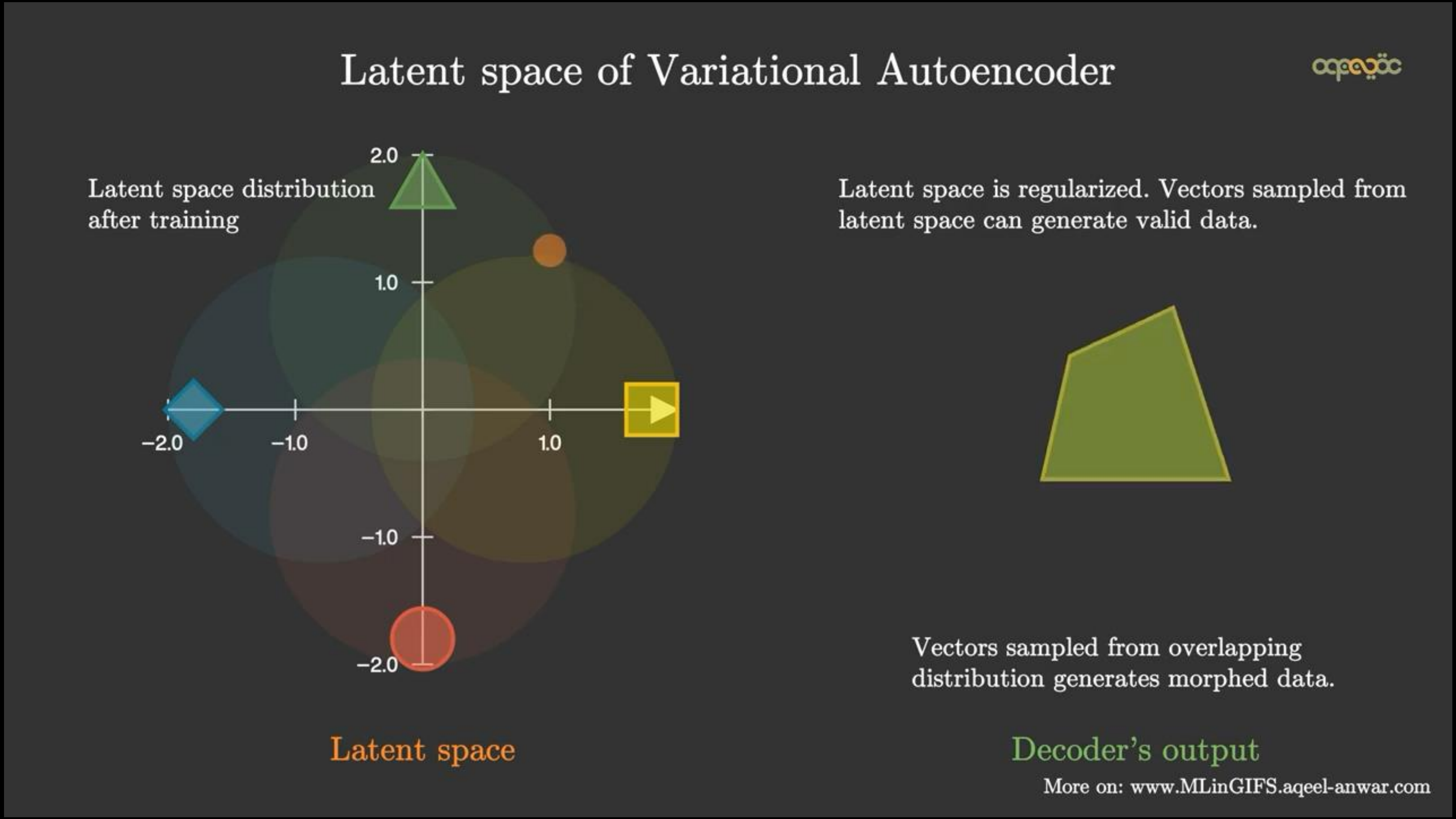
Variational Autoencoder



$$\mathbf{L} = \overbrace{\|x' - x\|}^{\text{reconstruction}} + \underbrace{D_{KL}(N(\mu_x, \sigma_x) \mid N(0,1))}_{\text{regularization}}$$



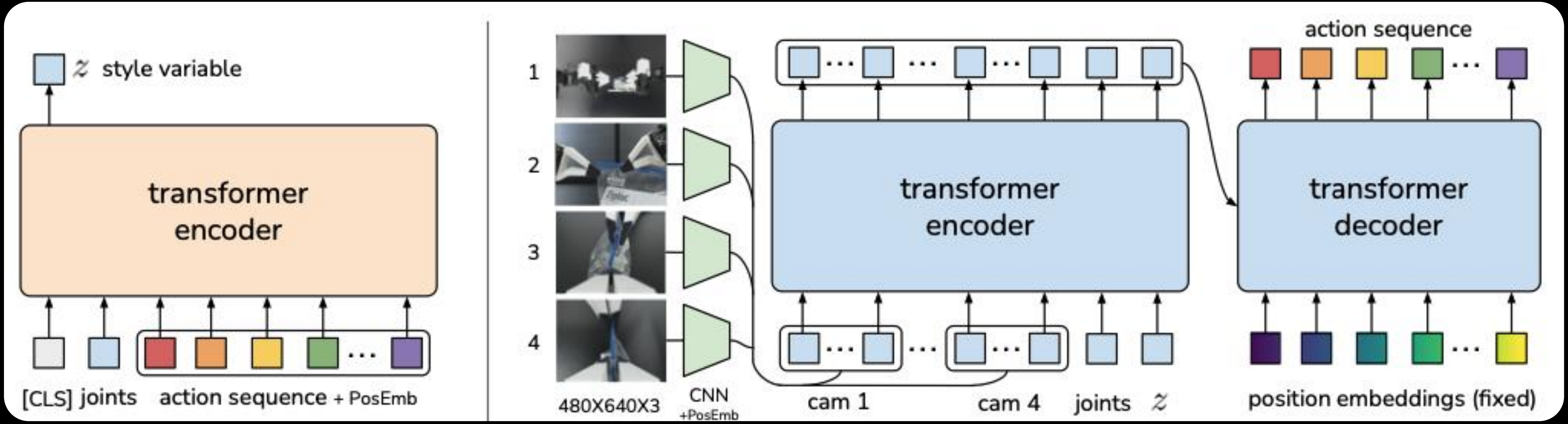
ACT : VAEs



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 (CVAE)



Encoder

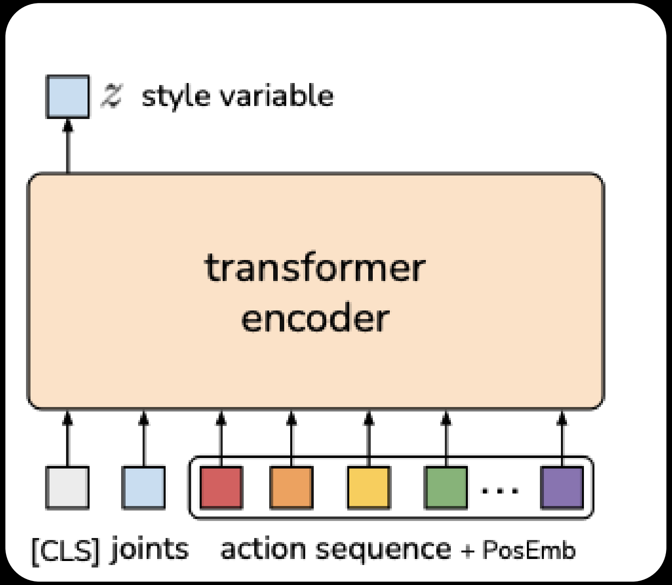
$$q_\phi(z \mid a_{t:t+k}, \bar{o}_t)$$

Conditional Decoder

$$\pi_\theta(a_{t:t+k-1} \mid o_t, z)$$

ACT : CVAE

Train the policy as a conditional variational autoencoder



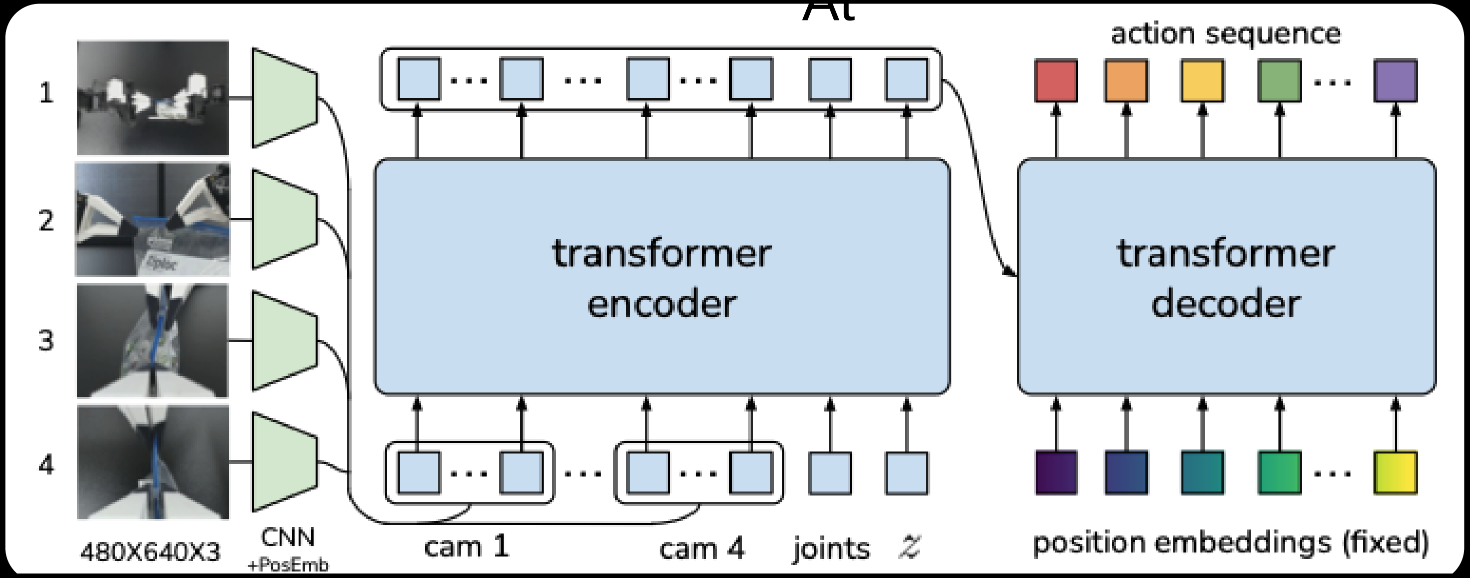
At training time, we encode the action sequence and the proprioception using a transformer

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

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

ACT: CVAE

Train the policy as a conditional variational autoencoder



Then, we decode the encoded vector z conditioned on image features and propioception.

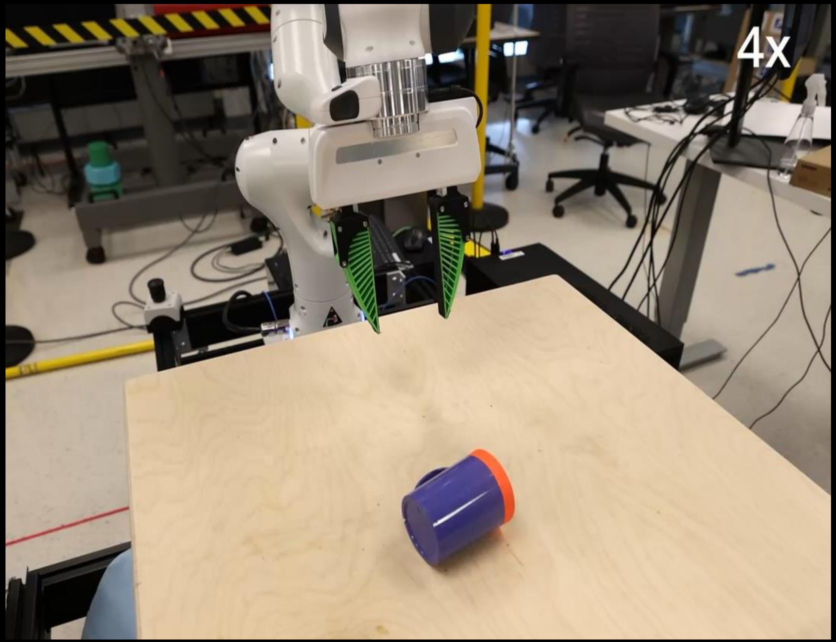
$$\boldsymbol{a_{t_t+k} = \pi_\theta(a_{t:t+k-1} | o_t, z)}$$

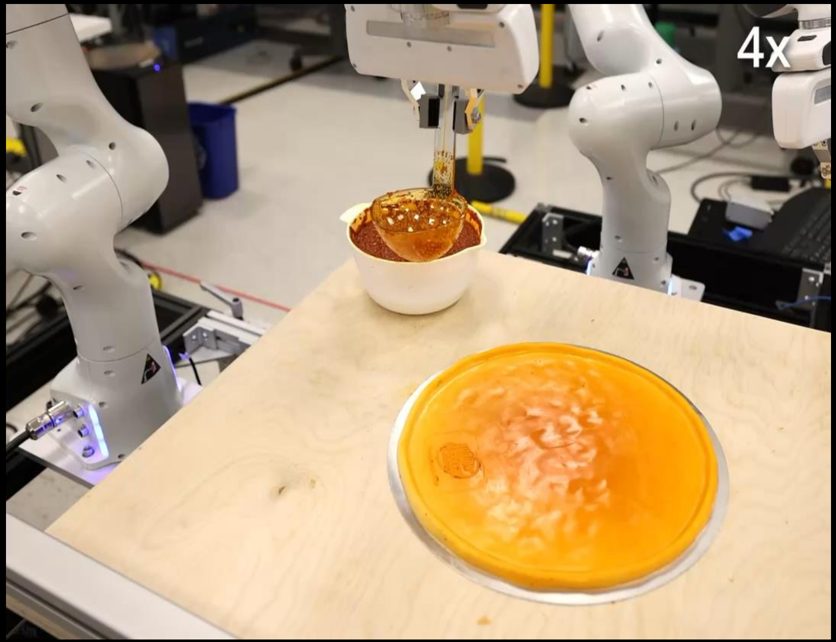
Training objective is the VAE loss

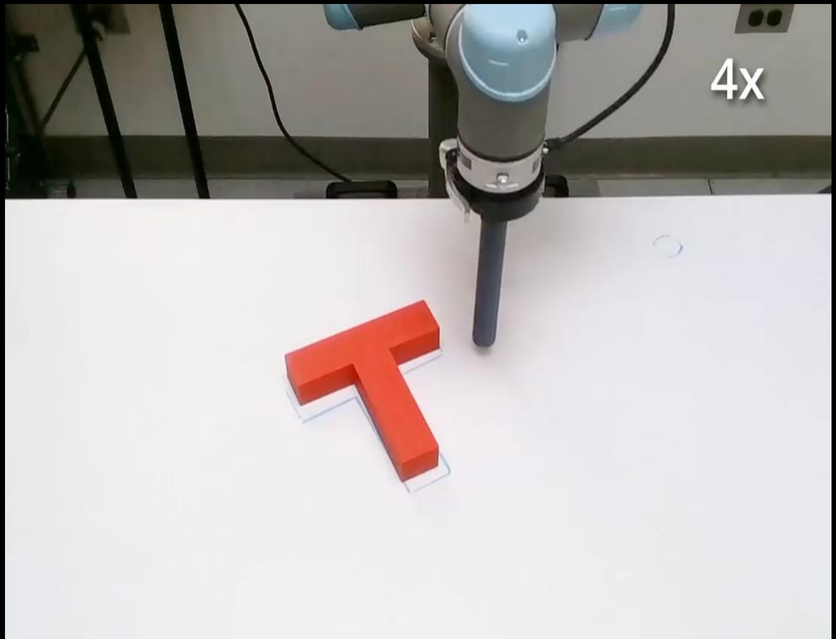
$$\|a_{t:t+k} - \hat{a}_{t:t+k}\|^2 + \beta * D_{KL}(\mathcal{N}(\mu_z, \sigma_z), \mathcal{N}(0,1))$$

At inference time **z is set to 0** (mean of the distribution), since we don't have access to future actions

Diffusion Policy

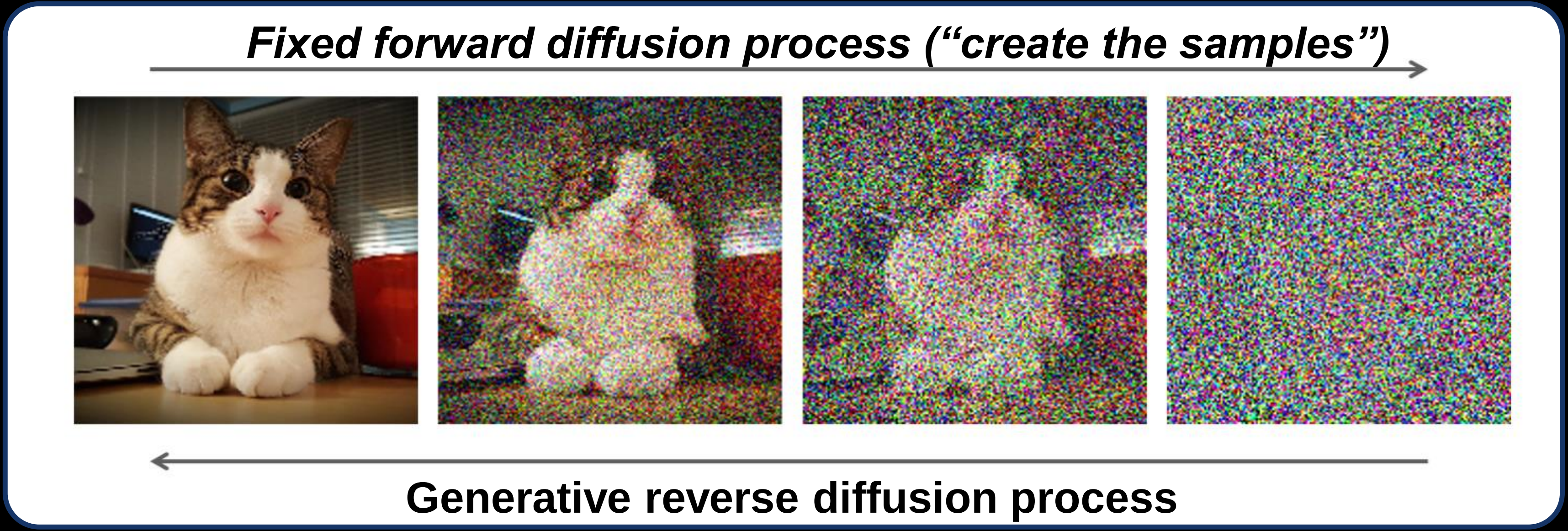






Chi, C. et al. (2023). *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion*

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

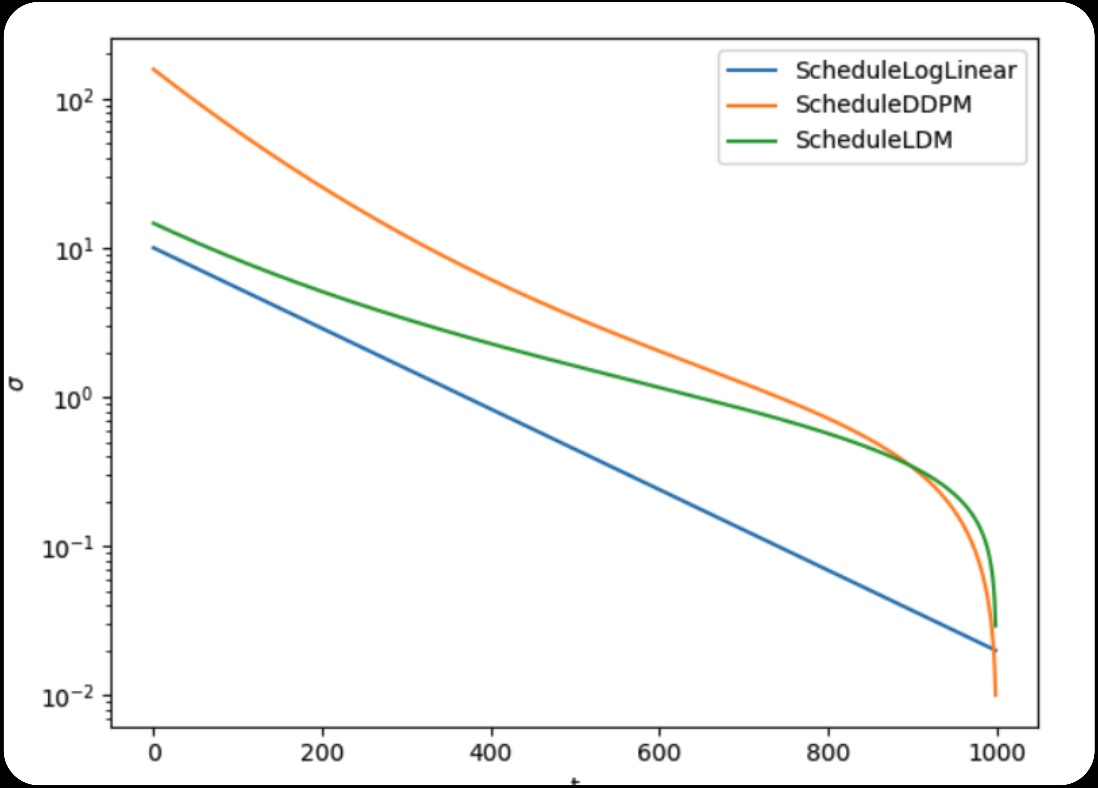
How do we add noise?

- 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)

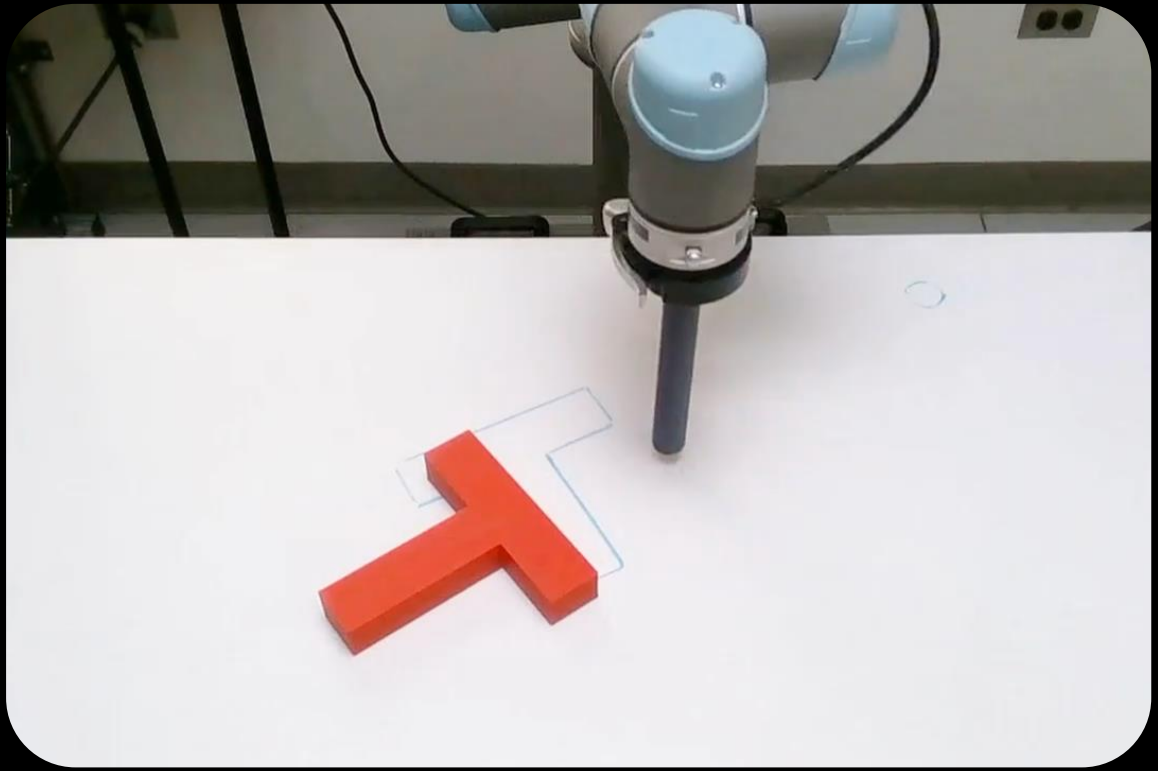


Diffusion Policy

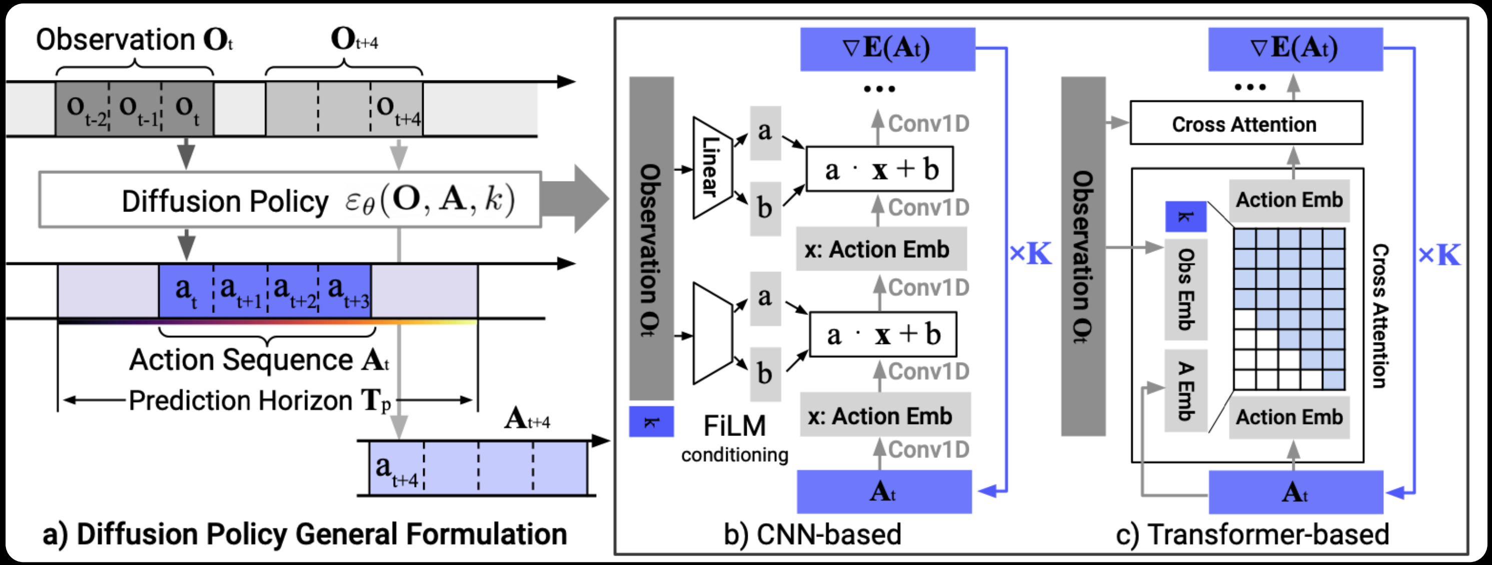
Apply diffusion process to robot action prediction

Advantages

- Handles multimodal action distribution (many valid actions for a given observation)
- Works in high-dimensional space (e.g., image generation)
- Stable training



Diffusion Policy



Training

$$L(\theta) = \left\| \epsilon^k - \epsilon_\theta(O_t, A_t^0 + \epsilon^k, k)) \right\|^2$$

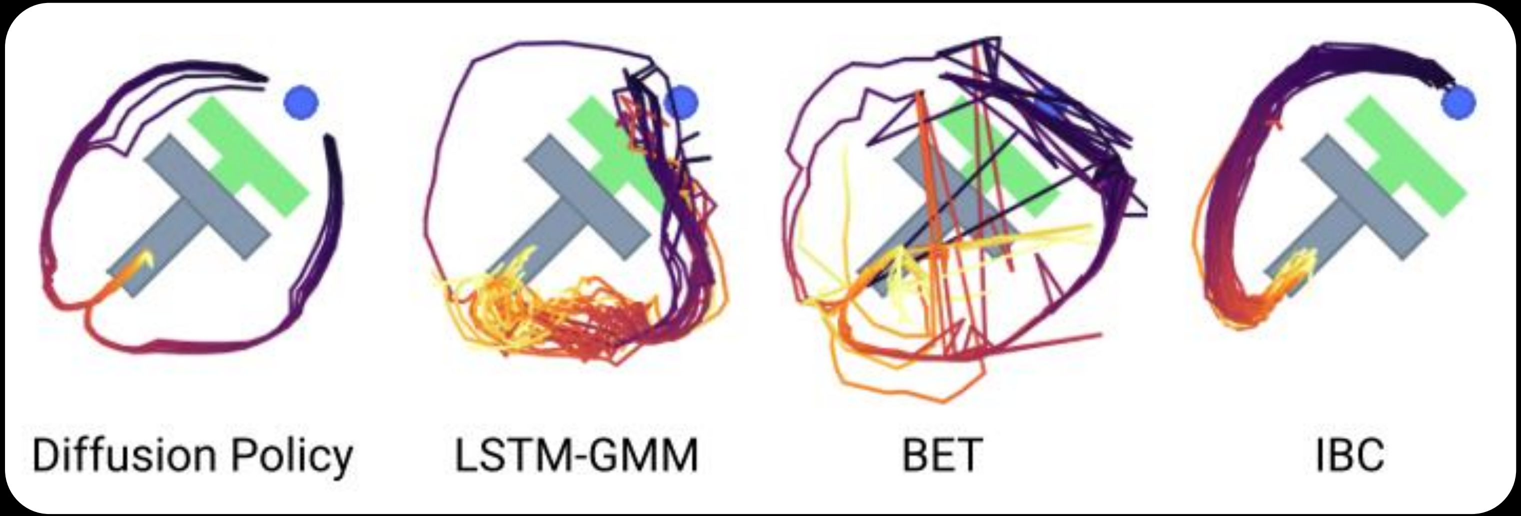
O_t : **history** of observations
A_t^0 : future action sequence
k : noise timestep (scheduler)
t : action timestep

Denoising Process

We use a DDPM to approximate the conditional distribution $p(A_t | O_t)$

$$\mathbf{A}_t^{k-1} = \alpha\left(\mathbf{A}_t^k - \gamma\varepsilon_\theta\left(\mathbf{O}_t, \mathbf{A}_t^k, k\right) + \mathcal{N}\left(0, \sigma^2\mathbf{I}\right)\right)$$

Diffusion Policy - Multimodality

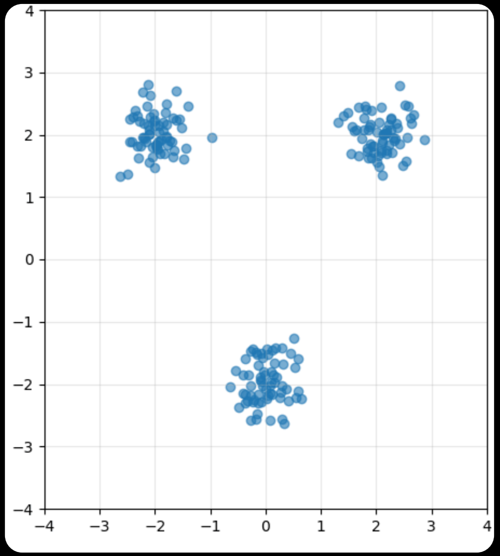


Diffusion Policy naturally models multimodal action distributions:

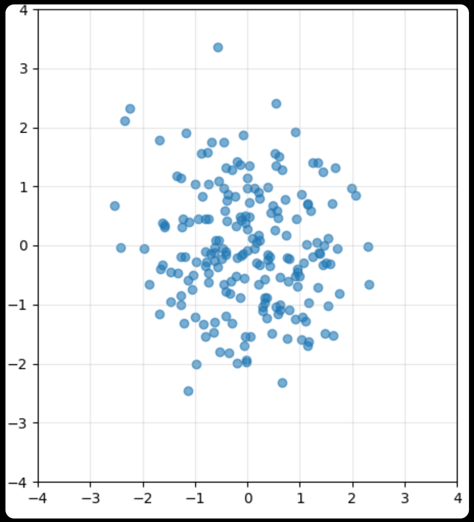
- **Each** rollout **commits to a coherent action mode**, but different rollouts represent **different valid strategies -** capturing true multimodality.

Side quest: Flow Matching

Real data distribution

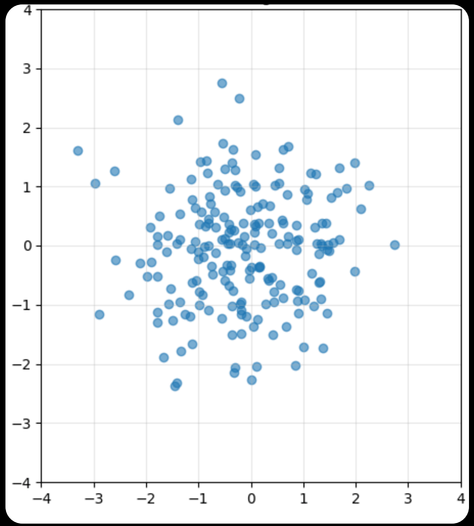


Diffusion



Diffusion Models gradually add noise to data until it becomes pure noise, then learn to reverse this process

Flow Matching



Flow matching creates a continuous path (or flow) between noise and data distribution

Side quest: Flow Matching

```
# DDPM: learn to predict noise
for each training step:

    x0 = sample_data()
    t  = random_timestep()

    # Add noise
    eps = random_noise()
    xt  = add_noise(x0, eps, t)

    # Predict the noise
    eps_pred = model(xt, t)

    # Train to match the true noise
    loss = MSE(eps_pred, eps)

    update(model, loss)
```

Noise scheduler

```
# Flow Matching: learn velocity from noise → data
for each training step:

    x0 = sample_data()
    z  = random_noise()
    t  = random_time_between_0_and_1()

    # Point along straight line from noise to data
    xt = (1 - t) * z + t * x0

    # Target velocity field
    v_target = x0 - z

    # Predict velocity
    v_pred = model(xt, t)

    # Train to match the velocity
    loss = MSE(v_pred, v_target)

    update(model, loss)
```

Key Takeaways

- Imitation Learning turns control into supervised learning
- Pixels become useful through feature encoders
- Demonstrations are noisy & multimodal
  → generative models (CVAE in ACT, diffusion in DP) capture variability and improve precision in challenging manipulation tasks.

Next lecture : From single-task to multi-task? What happens if we scale the model up?
VLA models

Acknowledgements and useful resources

- MIT 6.8210 Underactuated Robotics, chapter 21 https://underactuated.mit.edu/imitation.html
- Diffusion models :
  - (easy): https://chenyang.co/diffusion.html
  - (more complex) https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- Flow matching vs Diffusion https://harshm121.medium.com/flow-matching-vs-diffusion-79578a16c510
- Diffusion and Flow models lecture https://arxiv.org/pdf/2506.02070
- Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). *End-to-End Training of Deep Visuomotor Policies.* JMLR, 17(1), 1334-1373.
- Chi, C. et al. (2023). *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion.* arXiv:2303.04137.
- Zhao, T. Z., Kumar, V., Levine, S., Finn, C. (2023). *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware* (introduces ACT). arXiv:2304.13705.