# Advanced Topics in Deep Reinforcement Learning

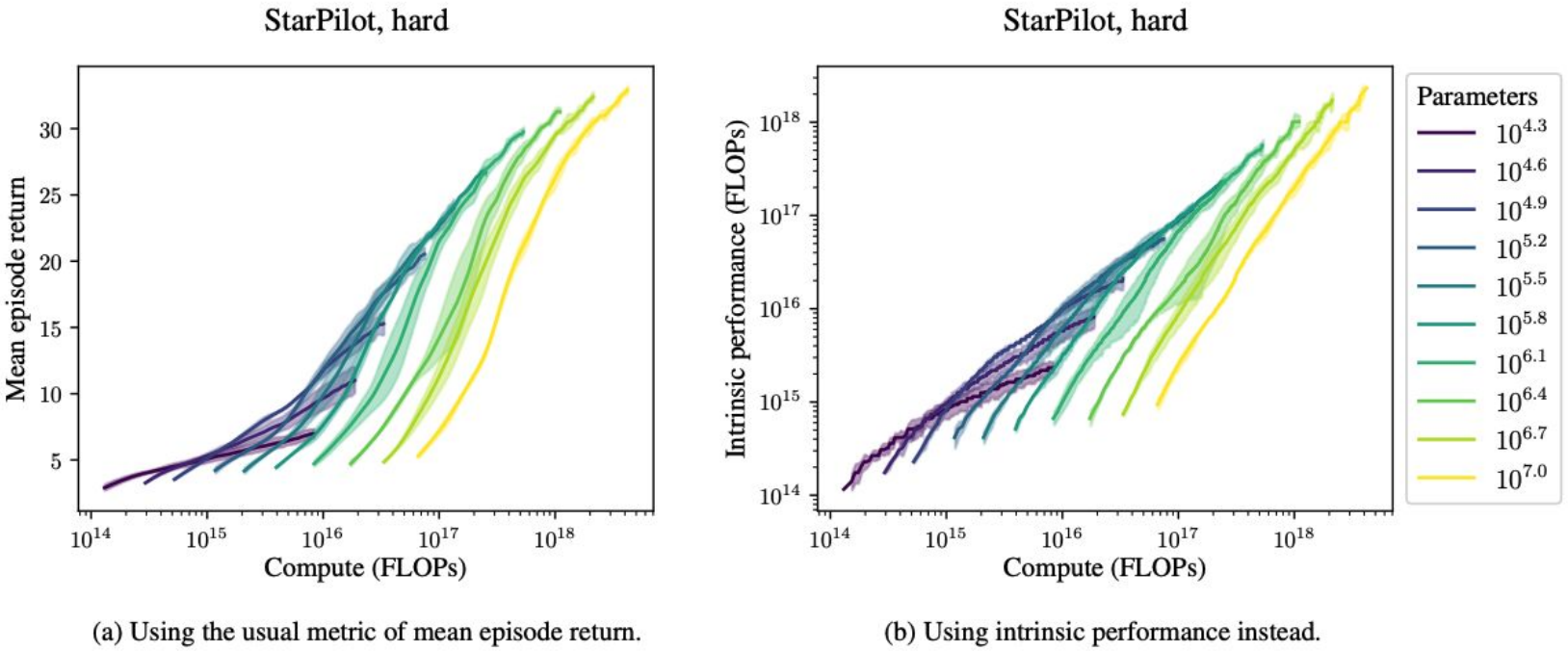
*Architectures*

# Which DL Architectures Are Common in RL?

- Default: (small) MLPs
- For image-based environments: CNNs
  - Size varies drastically from env to env
  - ProcGen and Atari can use decently sized ResNets, e.g. ResNet15
- RNNs have been used especially in generalization tasks
- Example overview for one environment: Crafter
- Recently: sequence modelling is becoming popular

| Algorithm | Score (%) | Reward | Open Source |
|---|---|---|---|
| Curious Replay | 19.4±1.6 | - | AutonomousAgentsLab/cr-dv3 |
| PPO (ResNet) | 15.6±1.6 | 10.3±0.5 | snu-mllab/Achievement-Distillation |
| DreamerV3 | 14.5±1.6 | 11.7±1.9 | danijar/dreamerv3 |
| LSTM-SPCNN | 12.1±0.8 | — | astanic/crafter-ood |
| EDE | 11.7±1.0 | — | yidingjiang/ede |
| OC-SA | 11.1±0.7 | — | astanic/crafter-ood |
| DreamerV2 | 10.0±1.2 | 9.0±1.7 | danijar/dreamerv2 |
| PPO | 4.6±0.3 | 4.2±1.2 | DLR-RM/stable-baselines3 |
| Rainbow | 4.3±0.2 | 6.0±1.3 | Kaixhin/Rainbow |

# Scaling Laws in RL? [Hilton et al. 2023]




(a) Using the usual metric of mean episode return.

(b) Using intrinsic performance instead.

Figure 1: Learning curves as a function of total training compute for StarPilot, an environment from Procgen Benchmark, using CNNs of different widths. Mean $\pm 1$ sample standard deviation over three seeds shown.

# Representations

Representations are an important factor for model architecture.

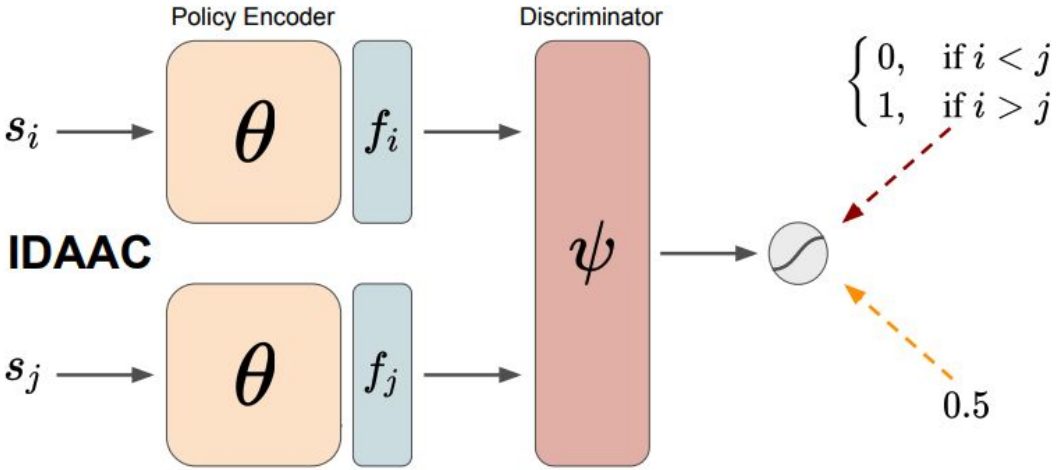
- Complex observation shapes require processing
- Information-dense observations need more complex representations
- Distractors need to be filtered out
- The agent needs to learn feature importances for different points in time

Broadly two options:

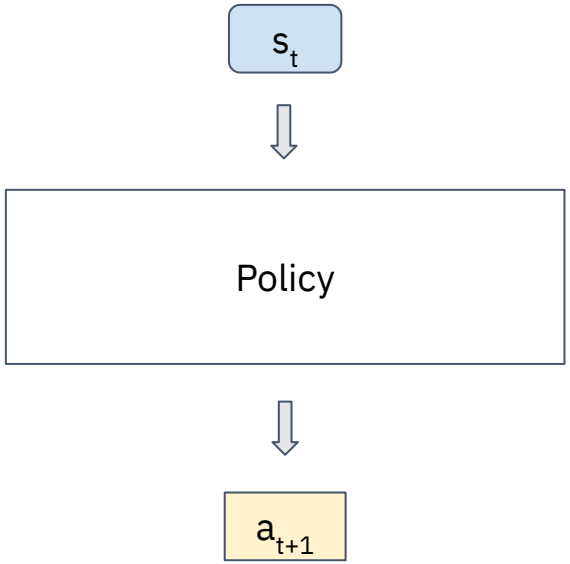
1. Add additional loss term for representations
2. Build representation learning into architectures

# Representations: IDAAC [Raileanu & Fergus 2021]

Intuition: value prediction needs more complex representations than policy.



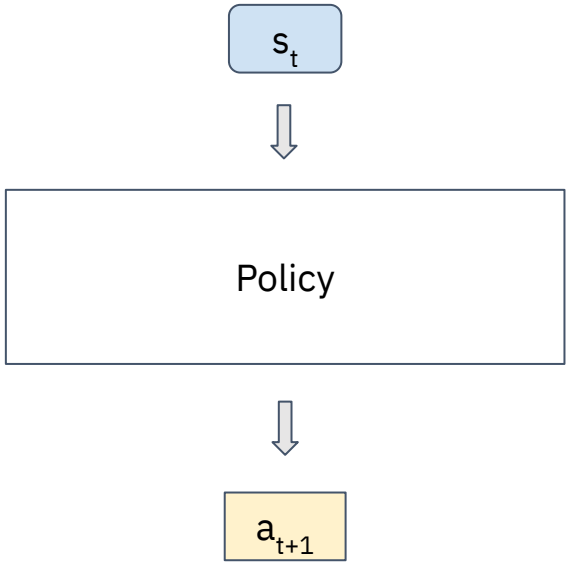
# From Point To Sequence Prediction



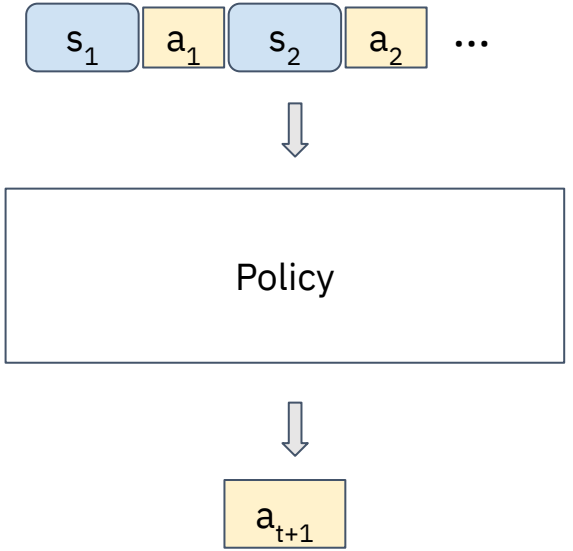

State to Action Mapping

Doesn’t work well without Markov assumption

# From Point To Sequence Prediction




State to Action Mapping

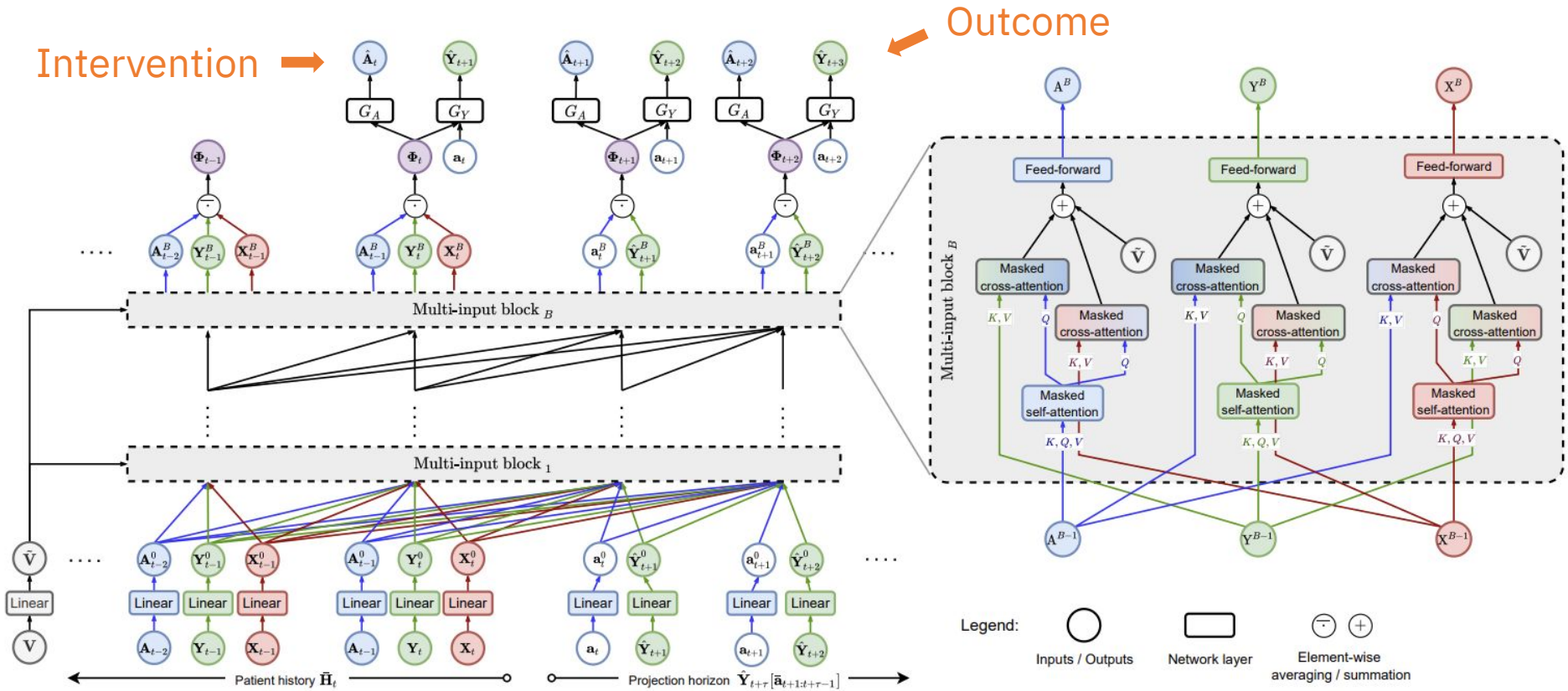



Trajectory to Action Mapping

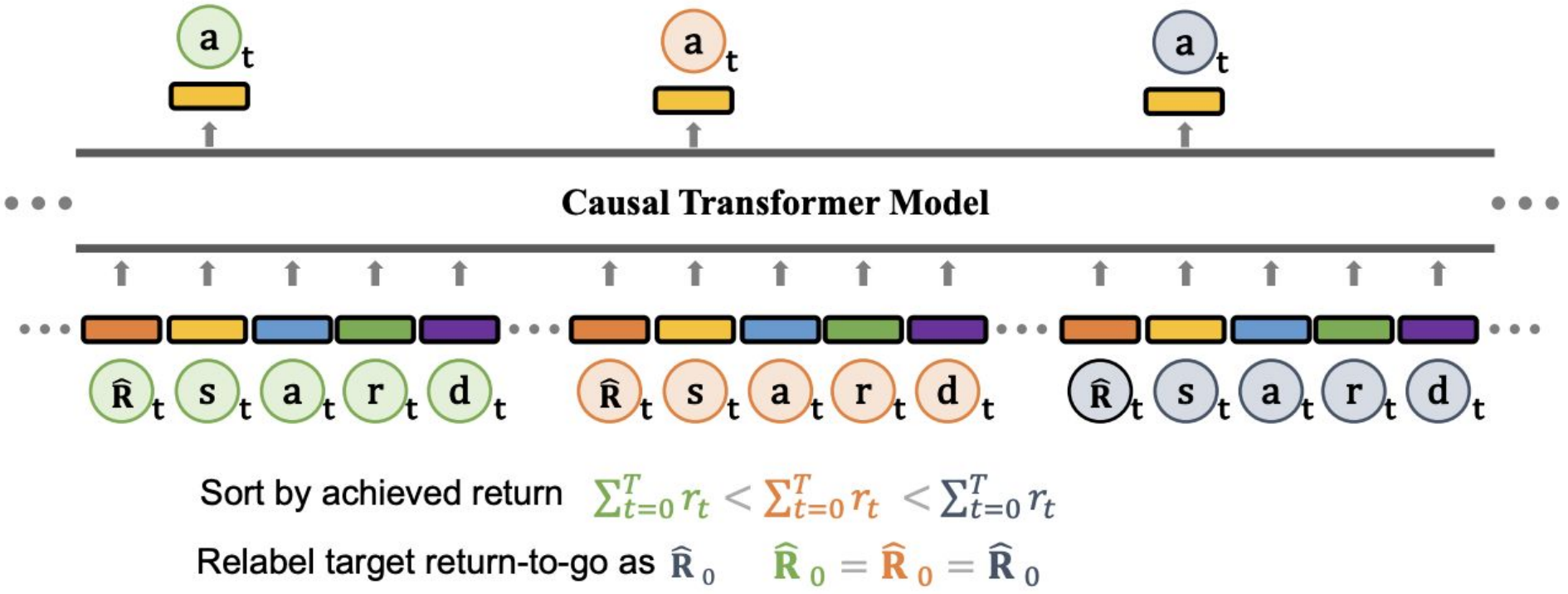
# Sequence Models As Policies

- We can use the whole trajectory to make predictions
- Markov assumption becomes less relevant
- We can detect environment changes during the episode
- In principle, we can even input the full training sequence
- More flexibility than single state predictions

# The Causal Transformer [Melnychuk et al. 2022]



# Agentic Transformer [Liu & Abeel 2023]



# SSMs for RL [Lu et al. 2023]

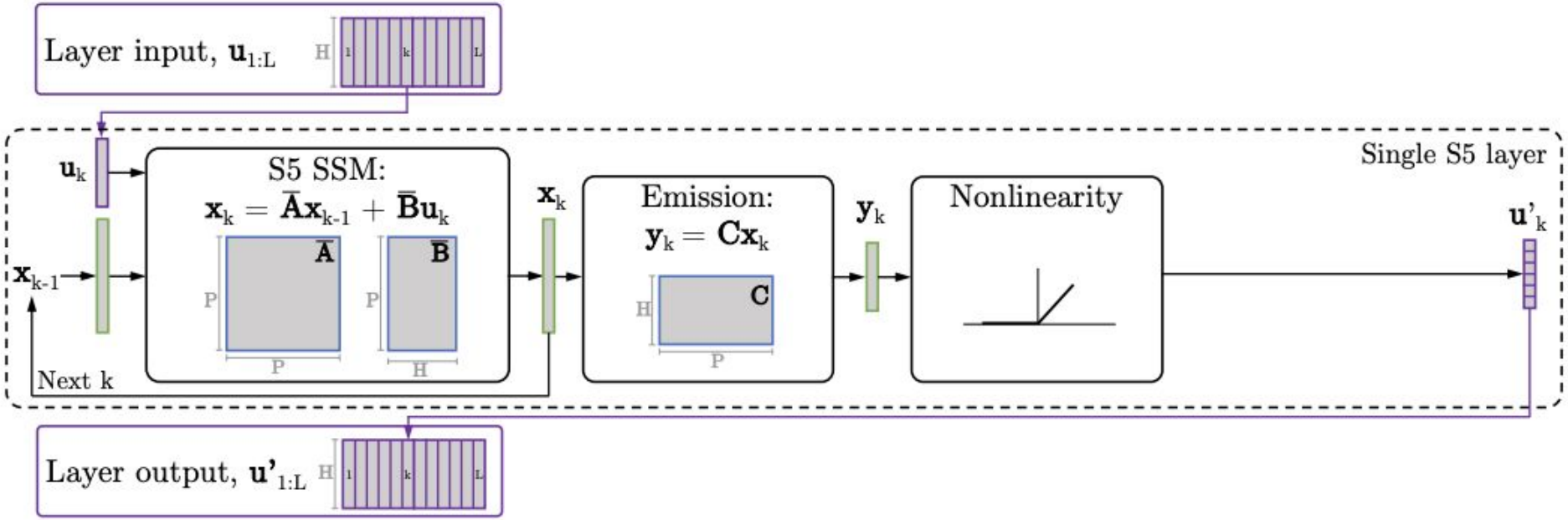
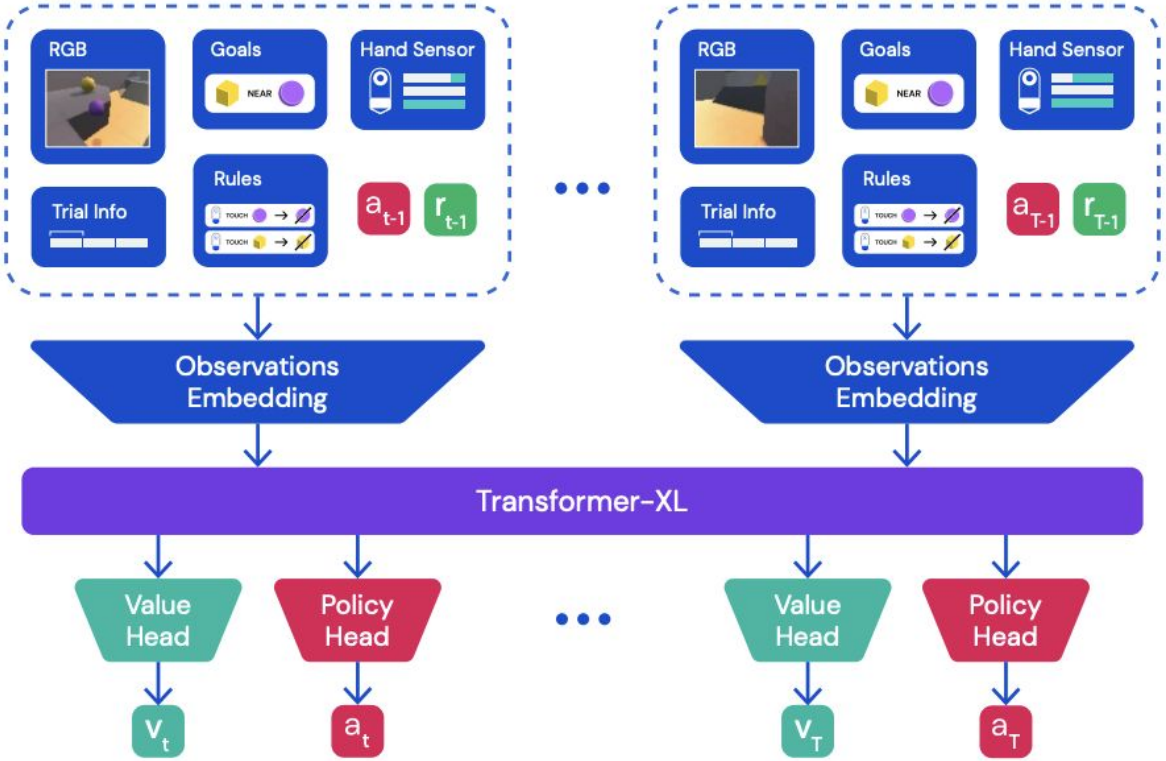


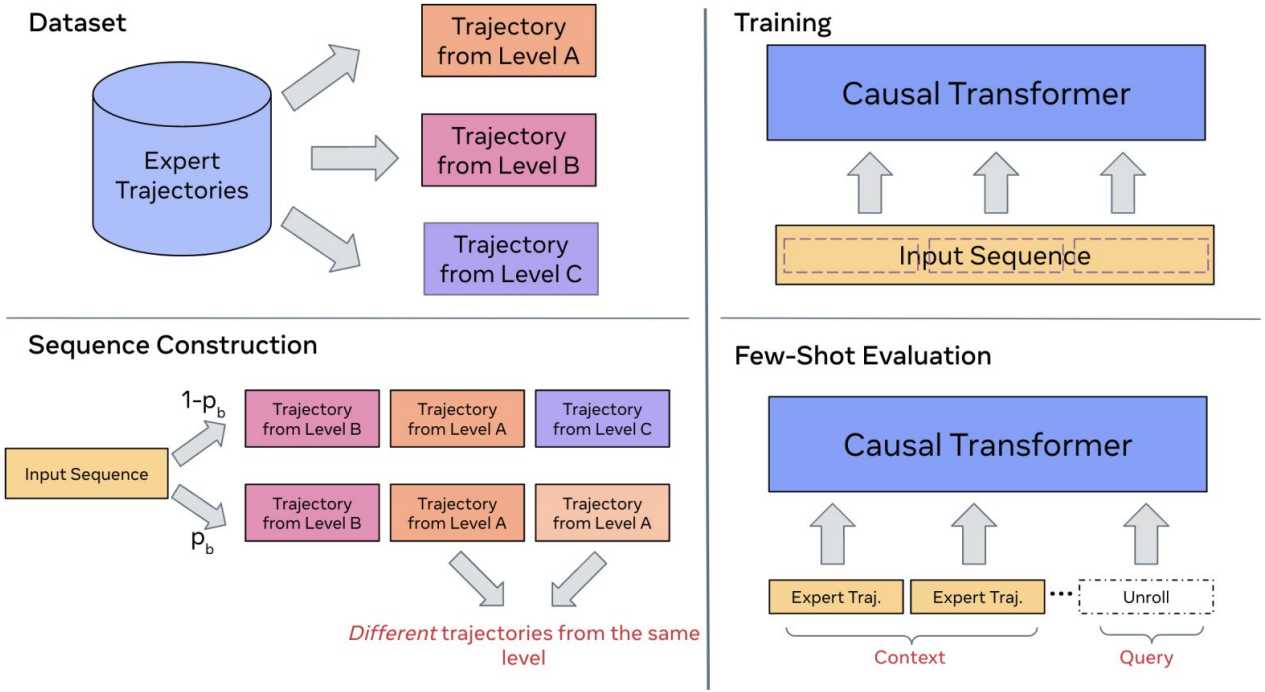

Figure from Smith et al. 2023

# Adaption & Generalization in Sequences




Bauer et al. 2023

# Adaption & Generalization in Sequences




Rapharty et al. 2023

# Pros And Cons Of Sequence Models

## Pros:

- Flexibility
- Few-shot learning
- Dealing with PoMDPs

# Pros And Cons Of Sequence Models

Pros:

- Flexibility
- Few-shot learning
- Dealing with PoMDPs

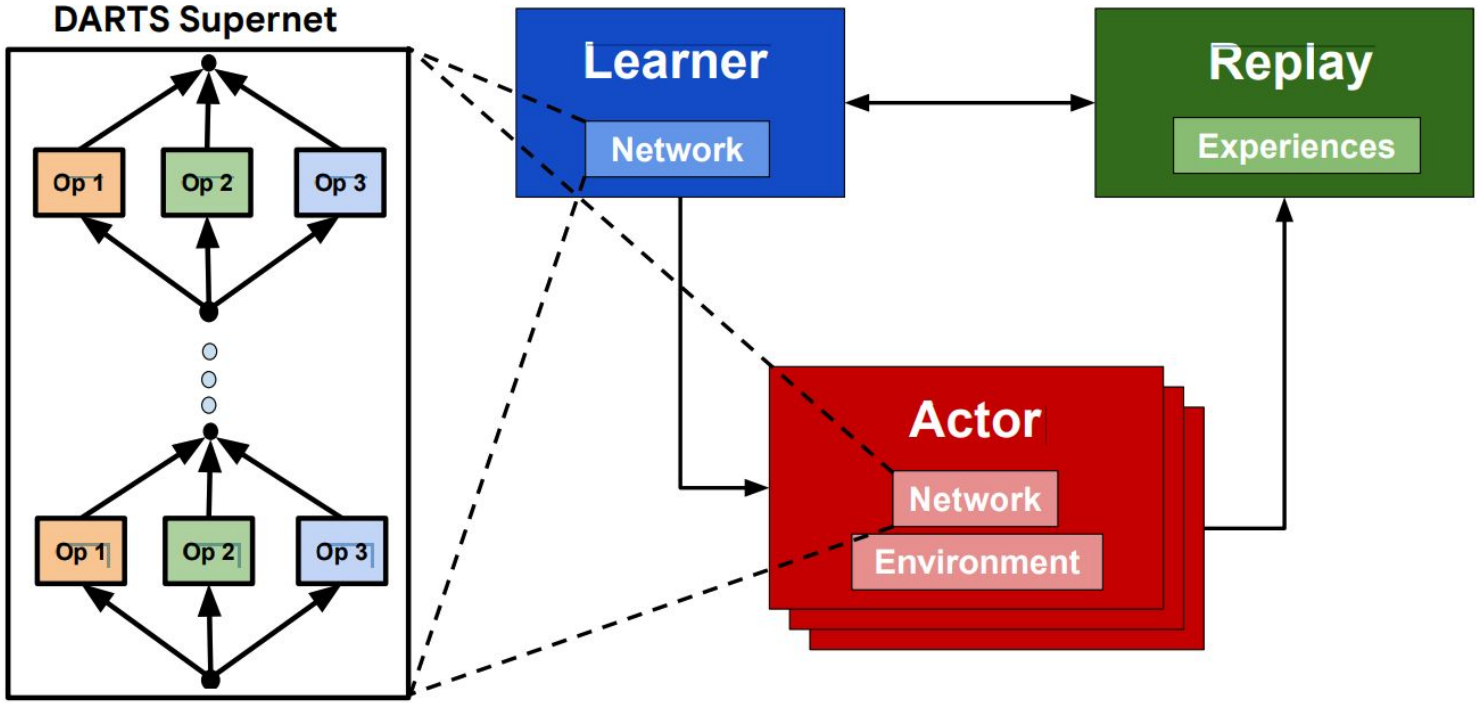
Cons:

- Data Hungry
- Training Time
- Compute Overhead
- Harder to Train

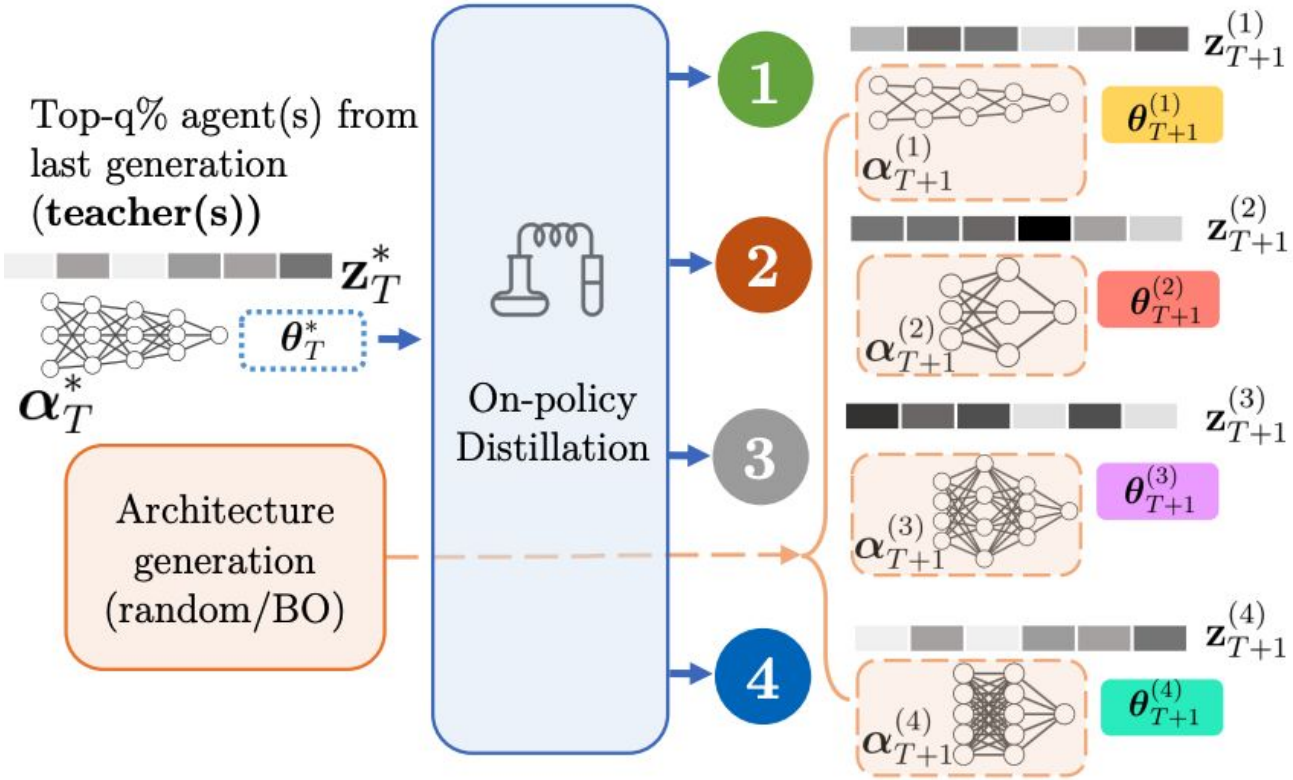
# Architecture Search for RL

- NAS looks for architecture in an automated way
- Usually, this requires many evaluations and is thus expensive
- Hyperparameters may need to be tuned for each architecture separately
- Search spaces can be very diverse, e.g.:
  - n_layers, n_neurons
  - DARTS (searching for a single CNN cell that can be duplicated)
  - In theory: full space of all connections

# The Classic: DARTS [Miao et al. 2023]



# Dynamically: BG-PBT [Wan et al. 2022]



# My Understanding of Architectures in RL

- ❑ I can name common RL architectures
- ❑ I understand the difference between point and sequence predictions
- ❑ I have an intuition of how sequence prediction in RL works
- ❑ I can suggest an rough architecture for a given usecase
- ❑ I know how architectures in RL have been searched for
- ❑ I understand how architecture interacts with representation