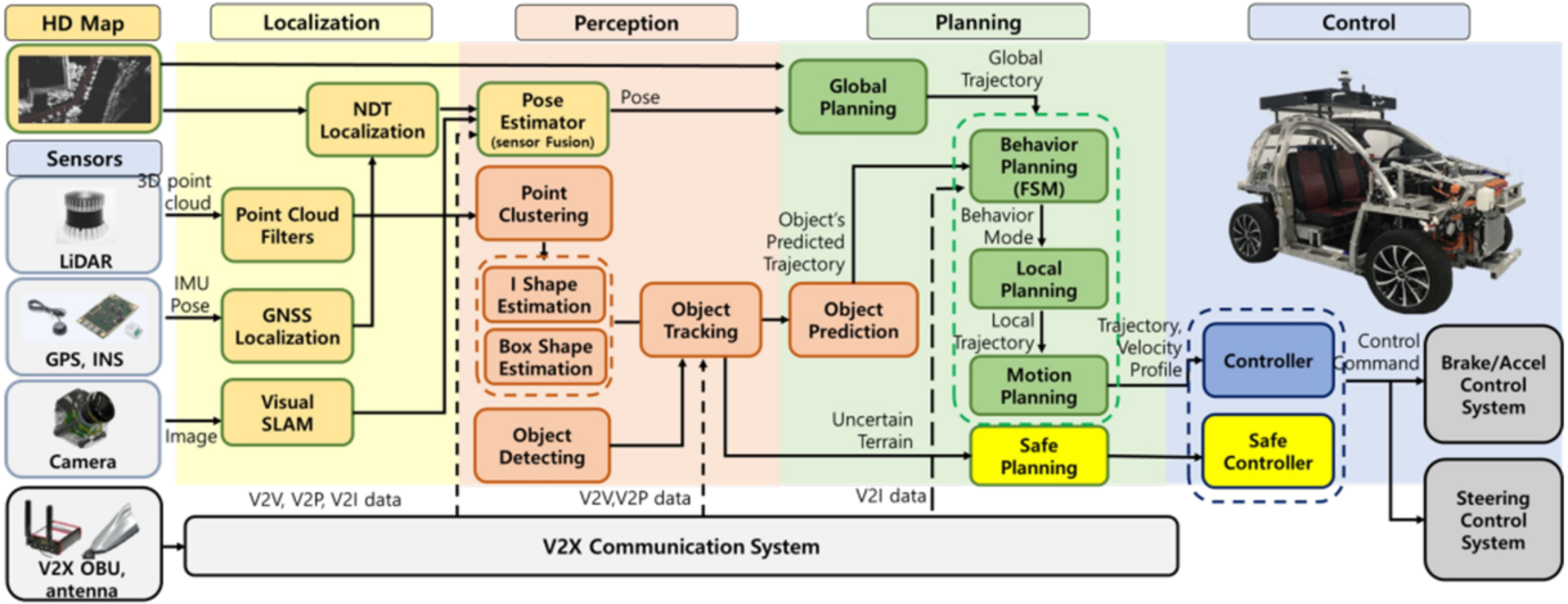
(Some examples of) Machine Learning for robotics and autonomous vehicles

SAUTOS Thematic School

David Filliat - U2IS - ENSTA Paris

Perception / Decision / Action

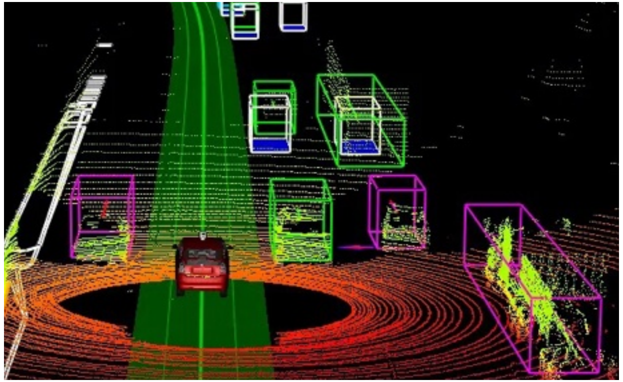
AI / Software view of autonomous systems architecture

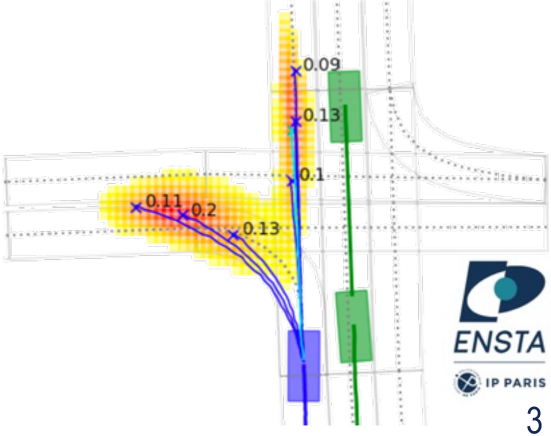


Machine Learning for autonomous systems

Machine Learning is important for robotics and intelligent vehicles

- Necessary for Perception
  - Vision, Lidar, Multi-sensor systems
  - Object/obstacle detection, road/path detection
  - Less for sensor fusion, mapping, localization…
- Useful for Decision
  - Trajectory prediction, manoeuvre prediction
  - Risk estimation (but guarantees ?)
- Promising (?) for Action
  - Learning model for MPC / tuning 'classical' controllers
  - End to end driving (e.g., Imitation learning)
  - Reinforcement learning





Challenges (~ School program)

For autonomous system

- Robust environment perception (rare situations, weather…)
- Safe / guaranteed algorithms for (perception,) localization, planning, control …
- Testing, validation, certification (ISO XXX)
- Interaction with humans (inside/outside the vehicle)

In machine learning more specifically

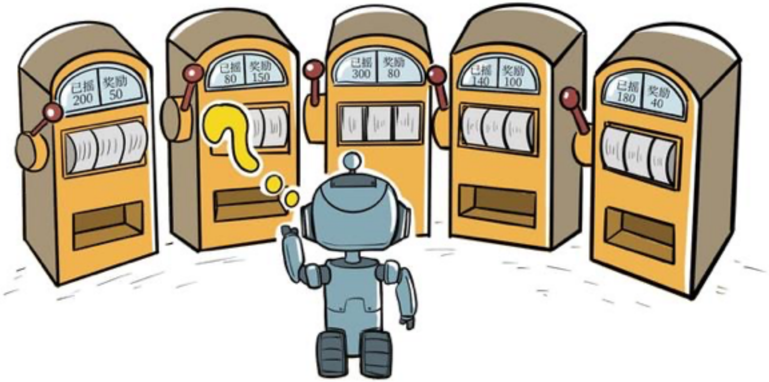
- Data annotation / self-supervision
- Robustness / Generalization / Domain adaptation
- Validation / Guarantees / Explicability of learned models
- Exploration / safety for Reinforcement learning

Reinforcement Learning (RL)

Reinforcement learning

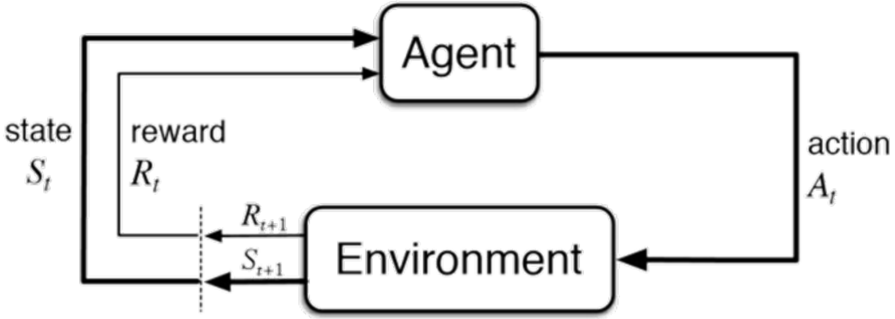
Definition

- Sequential decision problem
- No ‘output’ variable as in supervised learning, but a measure of answer/behaviour quality (**reward**)
- Simplest form : ‘bandit’ : choose between alternative with different rewards -> Find the best strategy to minimise **regret** (i.e., explore/exploit)

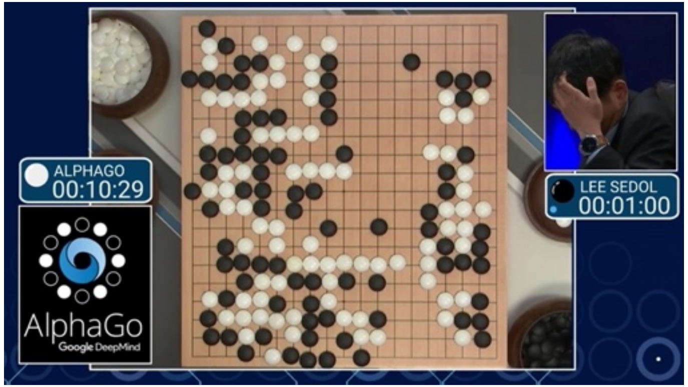


Reinforcement learning

– Problems with evolving state (Markov Decision Processes)
– Rewards can be delayed (e.g., win a game)



– Find a 'policy' mapping 'state' to 'action' maximizing sum of rewards



Value-Based Reinforcement Learning

Markov Decision Process, Policy & RL objective:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$$
$$T = (s, a, r, s+)$$

a = π(s)

$$\boxed{\pi^* \in \arg\max_{\pi} \mathop{\mathbb{E}}_{T \sim \pi, P^\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]}$$

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s; a_t = a \right]$$

Bellman evaluation operator:

$$\mathcal{T}^\pi Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s+ \sim P^\pi(s+|s)} Q(s+, \pi)$$

Bellman error and residuals (TD-Learning)

$$(\mathcal{B}^\pi Q)(s,a) = Q^\pi(s,a) - \mathcal{T}^\pi Q^\pi(s,a)$$
$$\mathcal{L}_{\mathrm{BR}} = \mathbb{E}_T \left[||(\mathcal{B}^\pi Q)(s,a)||^2 \right]$$

RL in Robotics vs Games

Deep Reinforcement Learning

- Big (cheap) data
- Slow Learning (millions of interactions with environment ➔ simulation)
- Learn one task defined by researcher
- Quite unstable (hyperparameters, …)

Reinforcement Learning for robotics

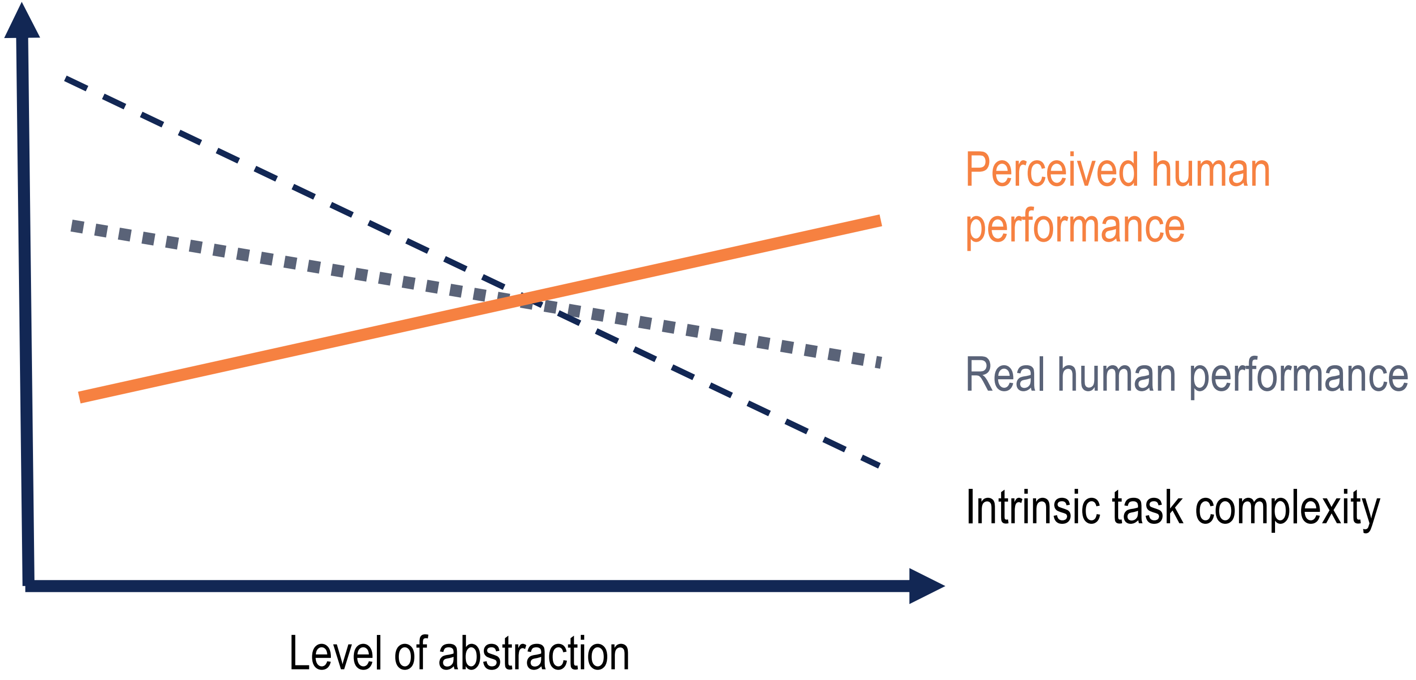
- Little (expensive) data
- Would need fast incremental learning during interaction with real world
- Learn multiple tasks
- No researcher → autonomous learning
- Needs to be stable, robust

Moravec Paradox

Encoded in the large, highly evolved sensory and motor portions of the human brain is a billion years of experience about the nature of the world and how to survive in it. The deliberate process we call reasoning is, I believe, the thinnest veneer of human thought, effective only because it is supported by this much older and much more powerful, though usually unconscious, sensorimotor knowledge. We are all prodigious olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it.

Hans Moravec

Moravec Paradox



Reinforcement Learning and Robotics

Robotics constraints

- Data are expensive (vs games), robots are slow, break easily
- Search (behaviour) space are huge, enough (iid) data difficult to gather
- Incremental learning, multi-tasks learning …

How to improve efficiency of reinforcement learning on real robots ?

- Learn compact representation to accelerate learning in real life
- Use auxiliary tasks to accelerate learning in real life
- Learn in simulation and transfer to real life

All of the above ?

State Representation Learning (SRL)

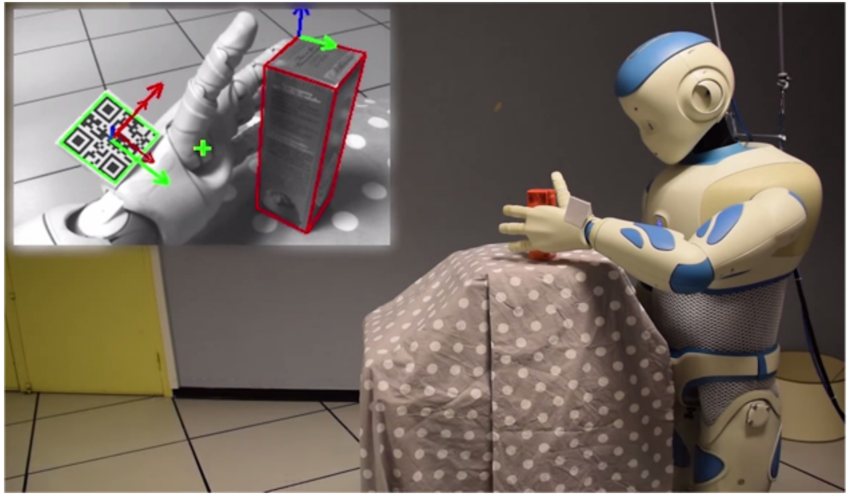
States ? $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$

Often, robot controllers require simple, ‘high-level’, low dimension inputs (the ‘state’ of the robot/world)

– E.g., grasping: object position, gripper position
  driving: road direction, obstacle positions, …

Vision based control requires filtering to get this information

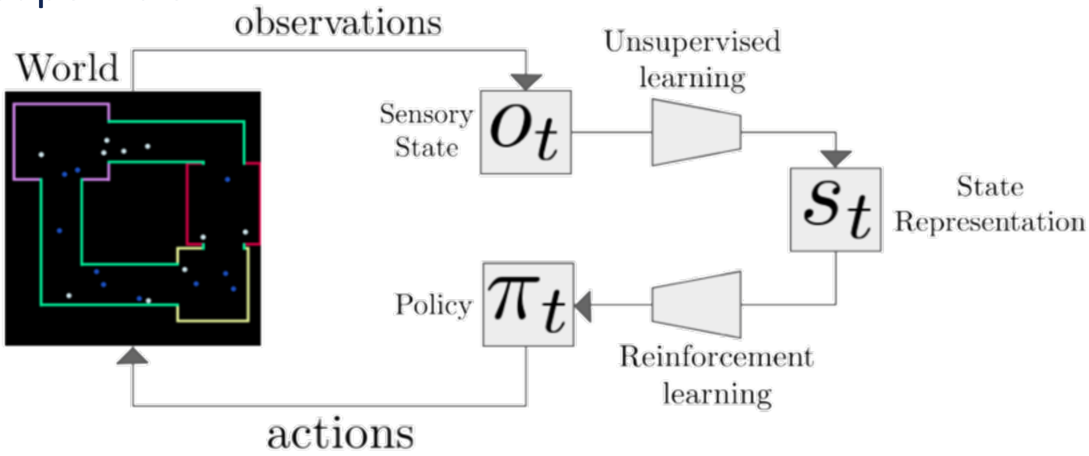
– Many solutions,
– often hand-crafted,
– task specific



State representation learning

Finding the right representation without supervision

- Deep learning makes it possible to learn useful representation
- Representations can be specialized for robotics/control
- Exploit observations / actions / rewards sequences, avoid human supervision



Why learning states ?

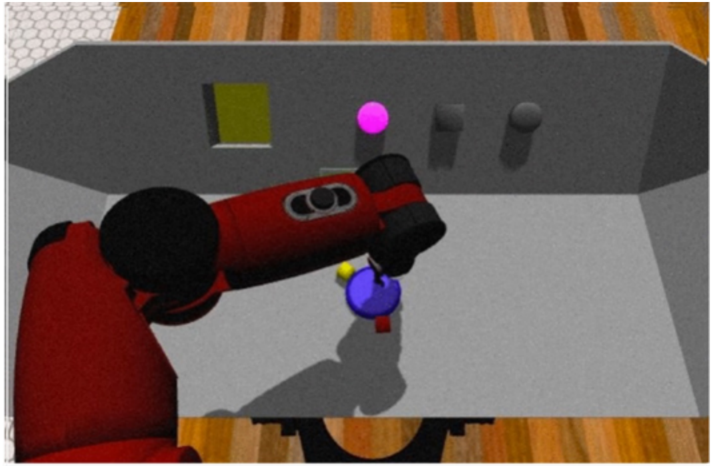
DREAM approach
[Doncieux et al., FiN18]

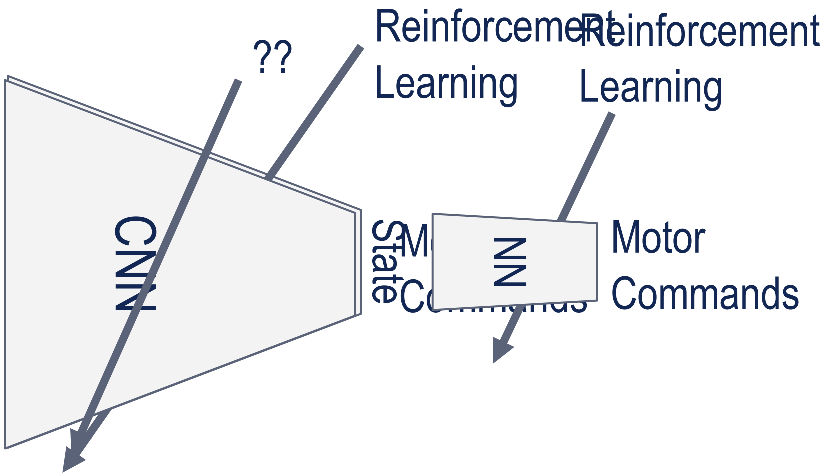
Facilitate adaptation to new task

– Discover the relevant state from exploration/demonstrations

Controllers are easier to train in such lower dimension

– Possibly faster than end-to-end; Could help transfer across tasks





(State) Representation Learning ?

SRL is a particular case of Representation Learning

- Entails a control/robotics context where actions are possible
- Often looks for interpretable info./ info. with a physical meaning
  - e.g., position, speed, angle...
- Disentanglement can also be interesting

SRL can be an objective by itself, but is often present in more general approaches

- e.g., as an auxiliary task in RL

What is a good state ?

A good state representation is : [Böhmer'15]

- Markovian, i.e. it summarizes all the necessary information to be able to choose an action within the policy, by looking only at the current state.
- Able to:
  - Represent the true value of the current state well enough for policy improvement.
  - Generalize the learned value-function to unseen states with similar futures.
- Low dimensional for efficient estimation.

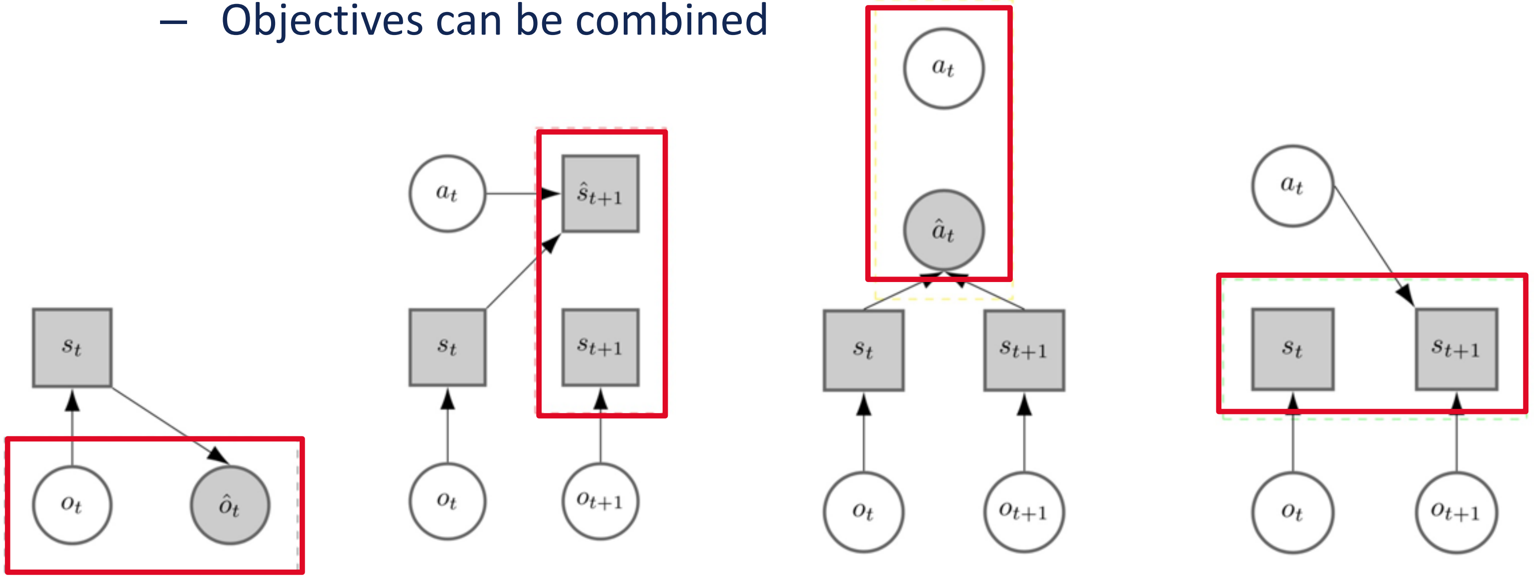
A good state representation should be : [Achille and Soatto, 2017]

- Sufficient (i.e. ~markovian)
- Minimal (Occams razor).
- Easy to work with i.e., **disentangled**

SRL approaches

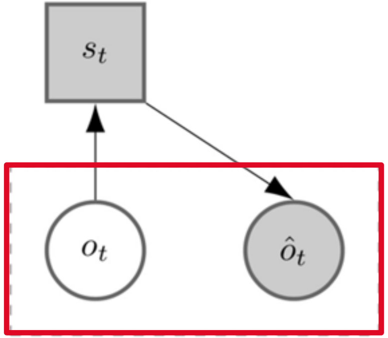
Learning state representation using self-supervision [Lesort & al., NN18]

- Several objectives can be exploited without human labelling
- Objectives can be combined



Autoencoders Forward Models Inverse models Priors

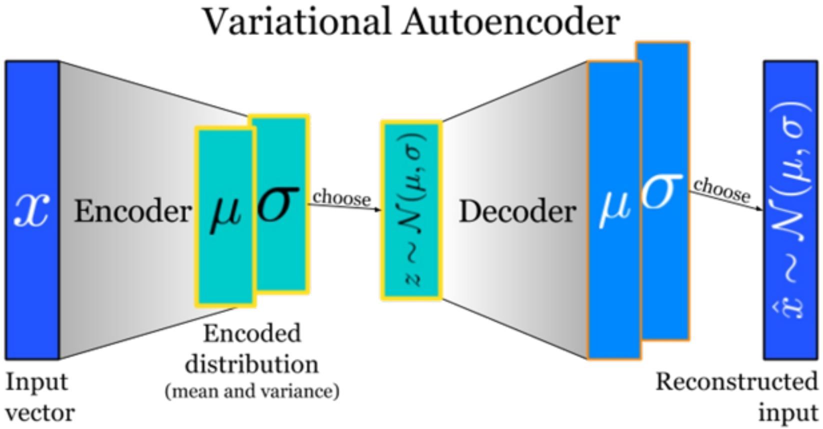
Reconstructing the observation



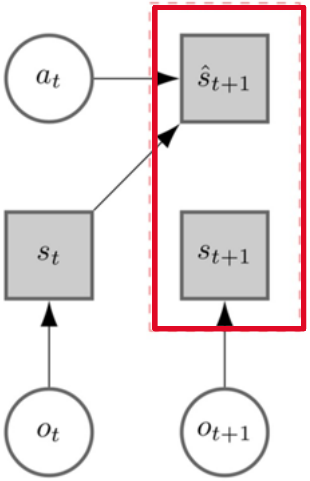
Train a state that is sufficient for reconstructing the input observation

- AE, DAE, VAE, …
- (Bi)GANs, …

Downside: sensitive to irrelevant variations (wrt actions)



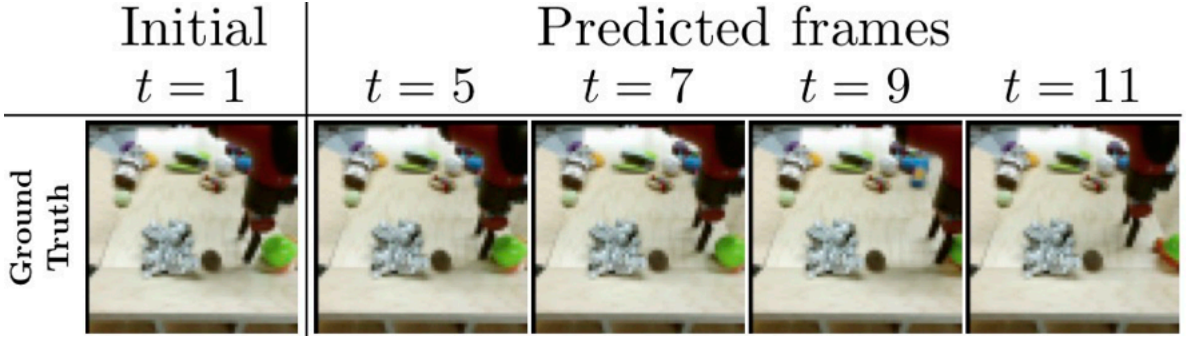
Forward models



Find state from which it is easy to predict next state
- Additional constraints to avoid fixed representations (AE, triplet loss…)
- Impose constraints on forward model (e.g., linear model)

Naturally discard irrelevant features

Can be used for model-based RL, MPC,…



Inverse models

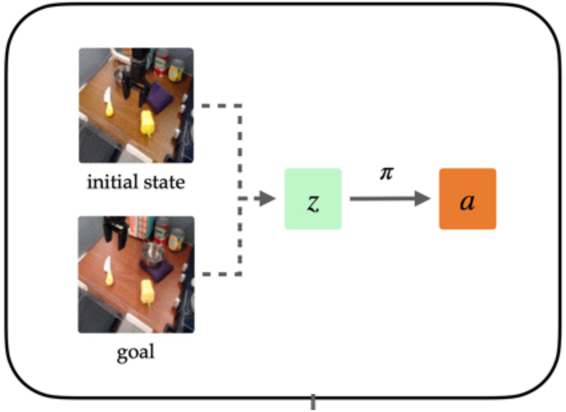
Find a state sufficient to recover action from 2 observations

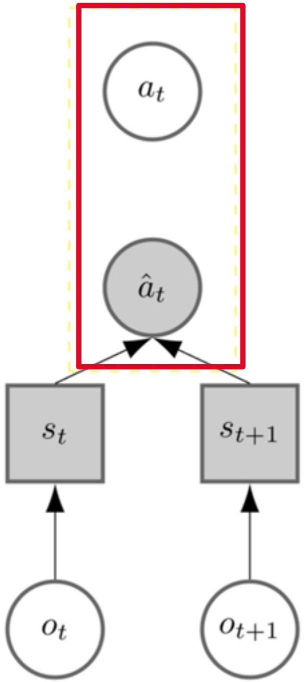
- Impose constraints on model (e.g., linear model)

Focus on states that can be controlled

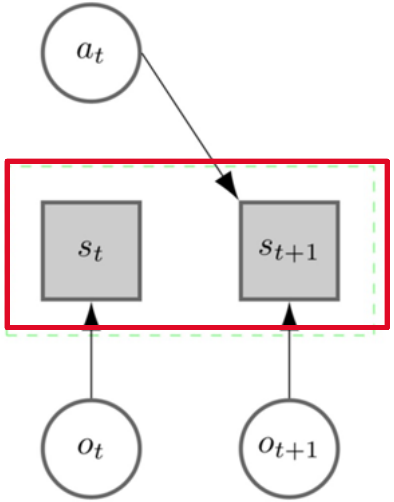
Useful for a direct control model

- E.g., goal conditioned policies
- $a = \pi(s,g)$





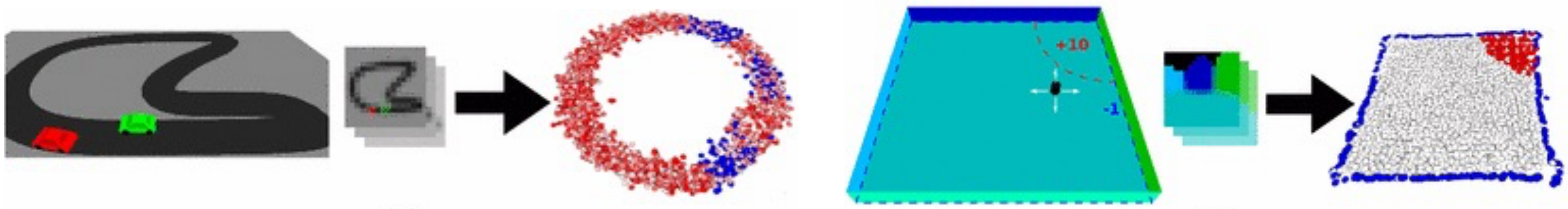
Prior models



Encode high-level constraints on the states

- Temporal continuity
- Controllability
- Inertia
- etc….

May exploit rewards



(a) (b)

Robotic Priors [Jonschkowski et. al. 2015]

Use *a priori* knowledge to learn representations relevant to the task

- **Temporal coherence Prior:** *Two states close to each other in time are also close to each other in the state representation space.*

$$L_{Temp}(D, \hat{\phi}) = \mathrm{E}[\| \Delta\hat{s}_t \|^2] \; , \tag{1}$$

- **Proportionality Prior:** *Two identical actions should result in two proportional magnitude state variations.*

$$L_{Prop}(D, \hat{\phi}) = \mathrm{E}[(\| \Delta\hat{s}_{t_2} \| - \| \Delta\hat{s}_{t_1} \|)^2 | a_{t_1} = a_{t_2}] \; , \tag{2}$$

- **Repeatability Prior:** *Two identical actions applied at similar states should provide similar state variations, not only in magnitude but also in direction.*

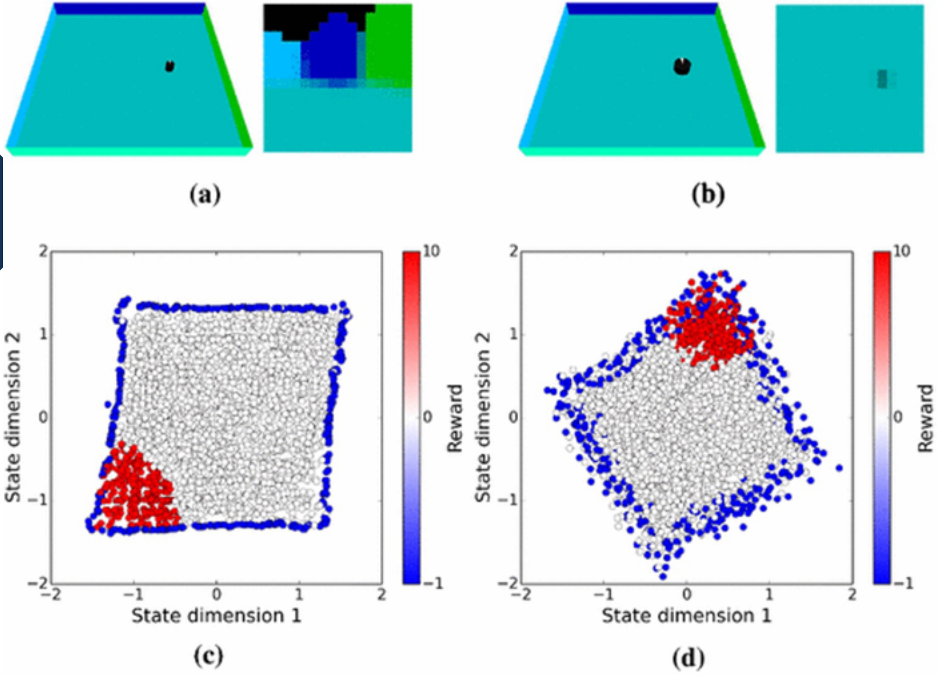
$$L_{Rep}(D, \hat{\phi}) = \mathrm{E}[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \| \Delta\hat{s}_{t_2} - \Delta\hat{s}_{t_1} \|^2 | \, a_{t_1} = a_{t_2}] \; , \tag{3}$$

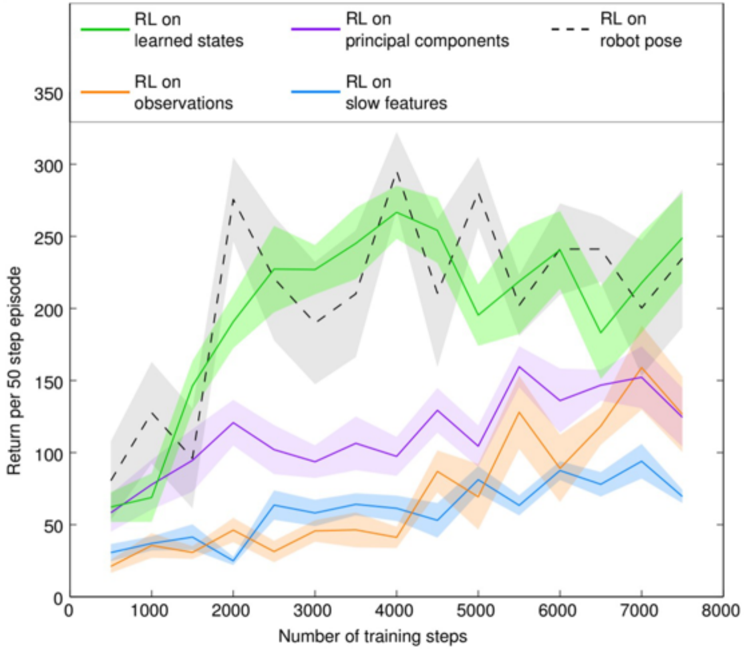
- **Causality Prior:** *If two states on which the same action is applied give two different rewards, they should not be close to each other in the state representation space.*

$$L_{Caus}(D, \hat{\phi}) = \mathrm{E}[e^{-\|\hat{s}_{t_2} - \hat{s}_{t_1}\|^2} \mid a_{t_1} = a_{t_2}, r_{t_1+1} \neq r_{t_2+1}] \; , \tag{4}$$

Robotic Priors [Jonschkowski et. al. 2015]

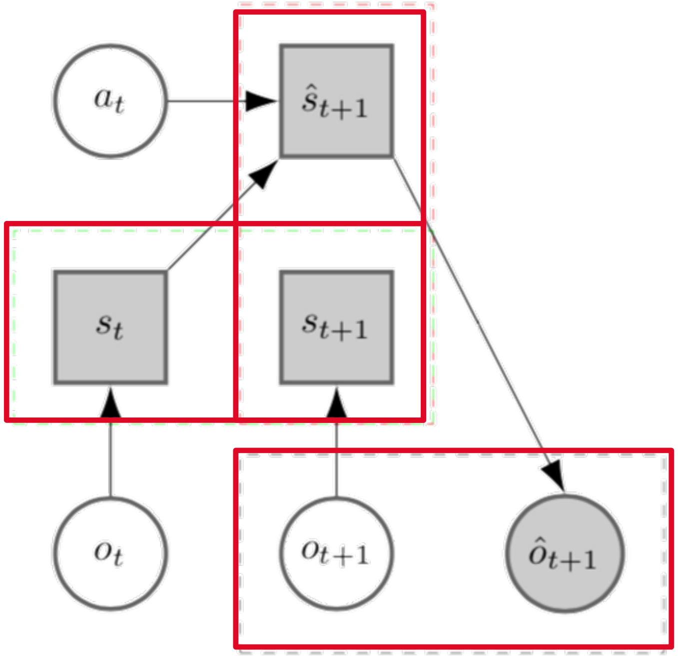
Use *a priori* knowledge to learn representations relevant to the task





Mixing objectives

Integrating several approaches



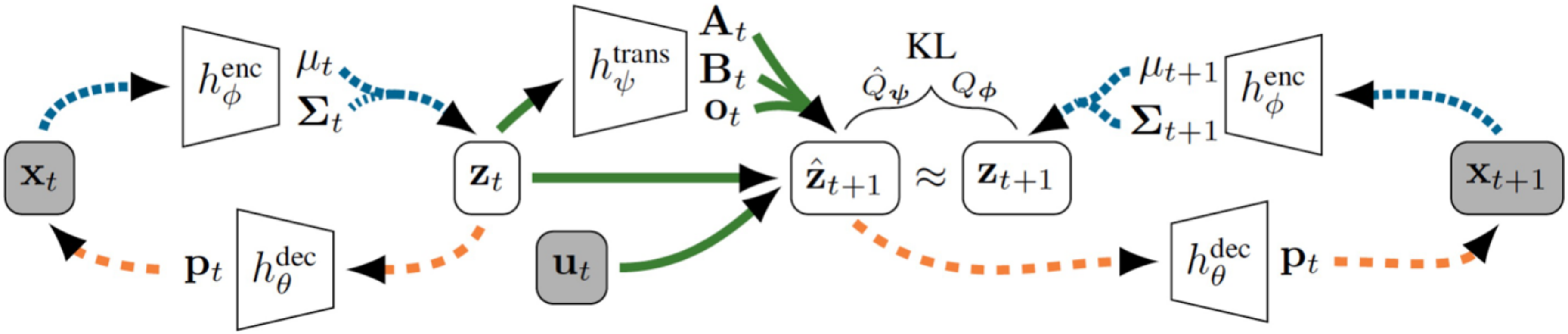
Embed to Control (E2C)

[Watter'18]

Multiple objectives

- Reconstruct observation using VAE
- Learn a locally linear forward model
- Exploit this forward model in optimal control setting

$$\hat{s}_{t+1} \sim \mathcal{N}(\mu = W * \hat{s}_t + U * a_t + V, \sigma)$$



SRL : state of the art

[Lesort et al, NN18]

State Representation Learning for Control: An Overview

Timothée Lesort[1, 2], Natalia Díaz-Rodríguez[1], Jean-François Goudou[2], and David Filliat[1]

[1]U2IS, ENSTA ParisTech, Inria FLOWERS team, Université Paris Saclay, Palaiseau, France.,
{timothee.lesort, natalia.diaz, david.filliat}@ensta-paristech.fr
[2]Thales, Theresis Laboratory, Palaiseau, France.,
{jean-francois.goudou@thalesgroup.com

THALES

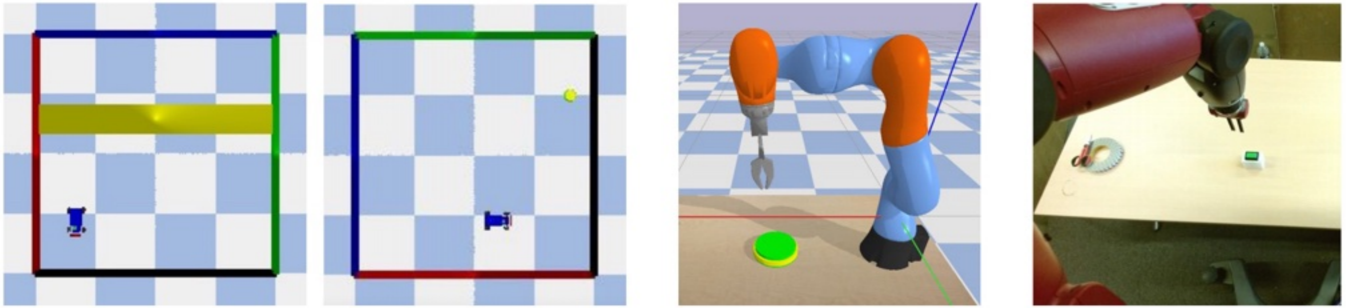
Contents

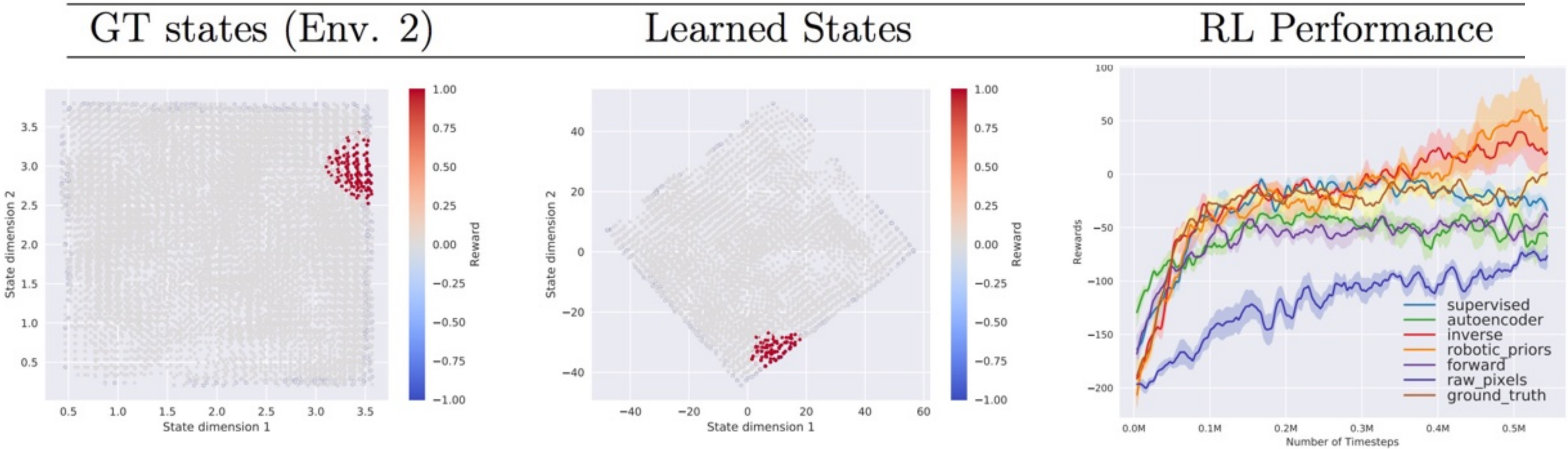
ENSTA IP PARIS

State representation learning Toolbox

[Lesort et al. 18]
[Raffin et al. 18]]

https://github.com/araffin/robotics-rl-srl





SRL Toolbox

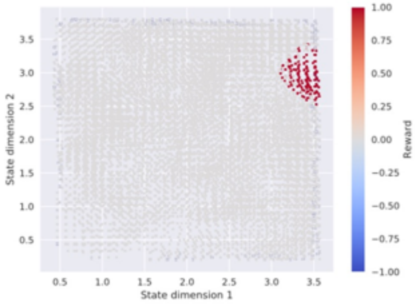
A set of baselines

- Auto Encoders
- Variational Auto Encoders
- Robotic priors
- Forward Models
- Inverse models

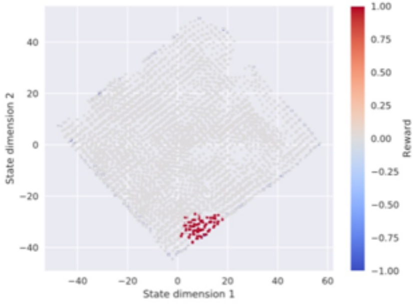
A set of evaluation tools

- RL (Stable Baselines)
- PPO, CMA-ES, ARS, …
- KNN-MSE
- Ground truth correlation

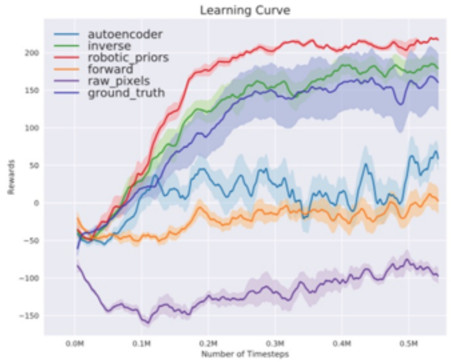
Ground Truth States



Learned States

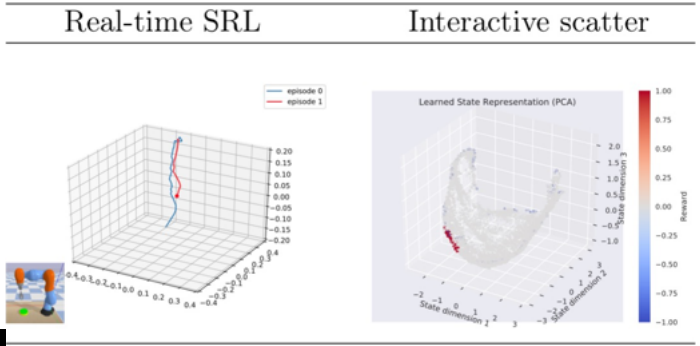


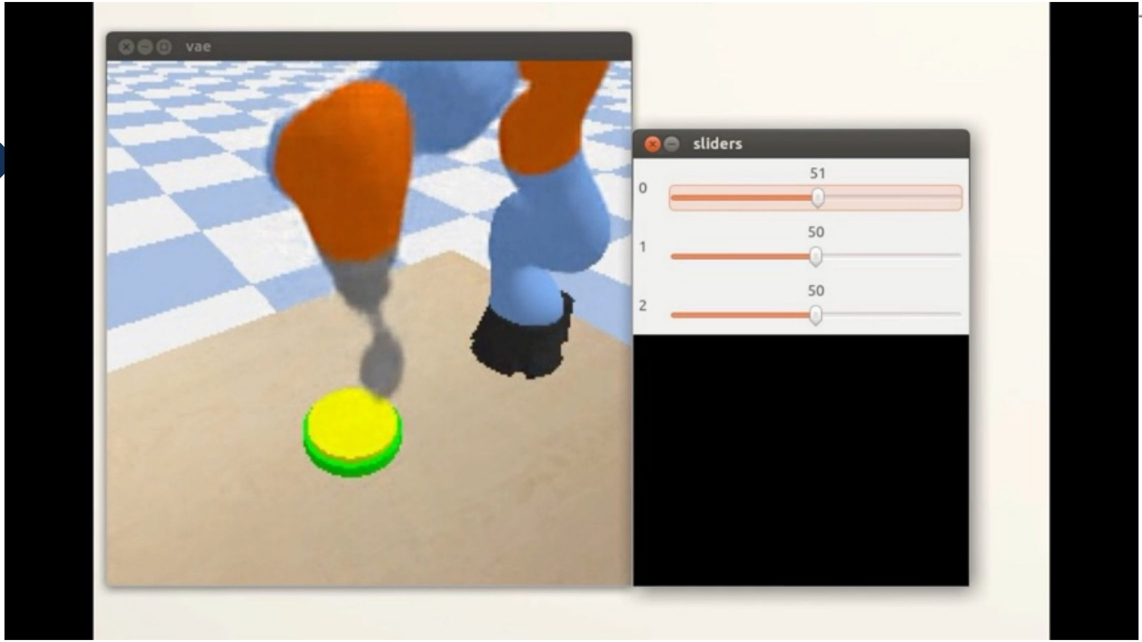
RL Performance



SRL Toolbox

A set of visualization tools

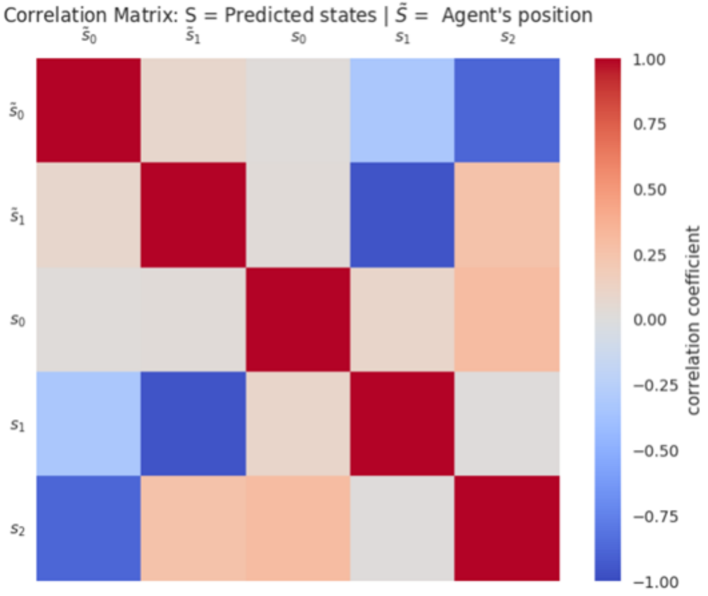




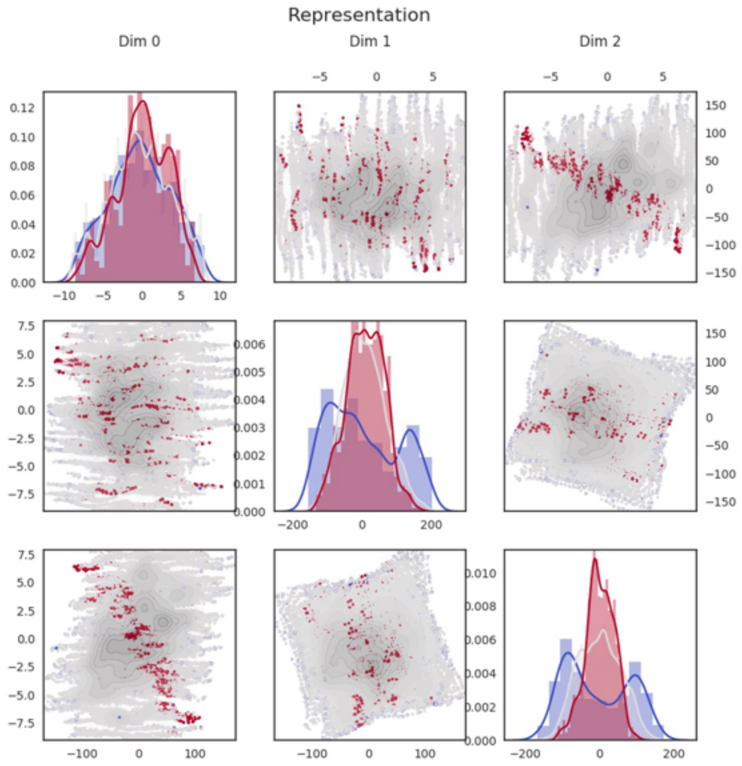
Latent visualization

SRL Toolbox

A set of visualization tools



State / GT correlation



State vs State plot

SRL Toolbox

Some lessons learned

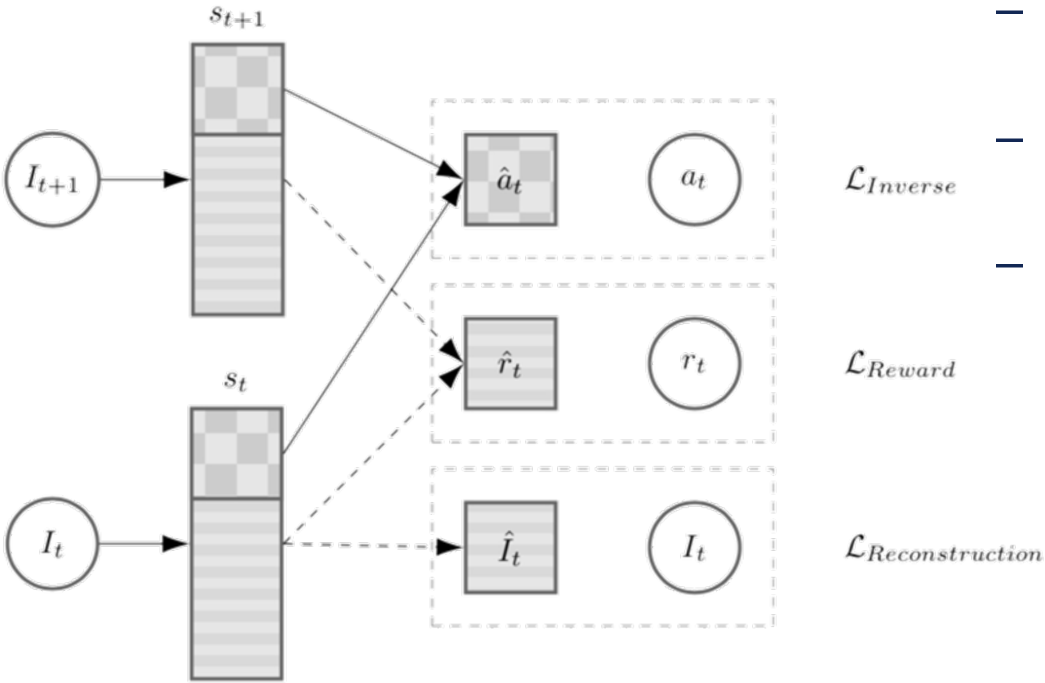
- Many methods' performance is quite task specific
  - E.g. robotic priors fail on robotic arms
- Autoencoders/VAE work quite well if extreme (small or large) noise
- Predicting a forward and inverse model often efficient
- Random states often reasonably efficient
- SRL + RL usually more efficient than end-to-end RL

- Encoding robot state AND environment state may be difficult
  - E.g. robotic priors work with fixed goal, but not moving goal

SRL - Split model

[Raffin et al. SPIRL19]

Learning structured state representation



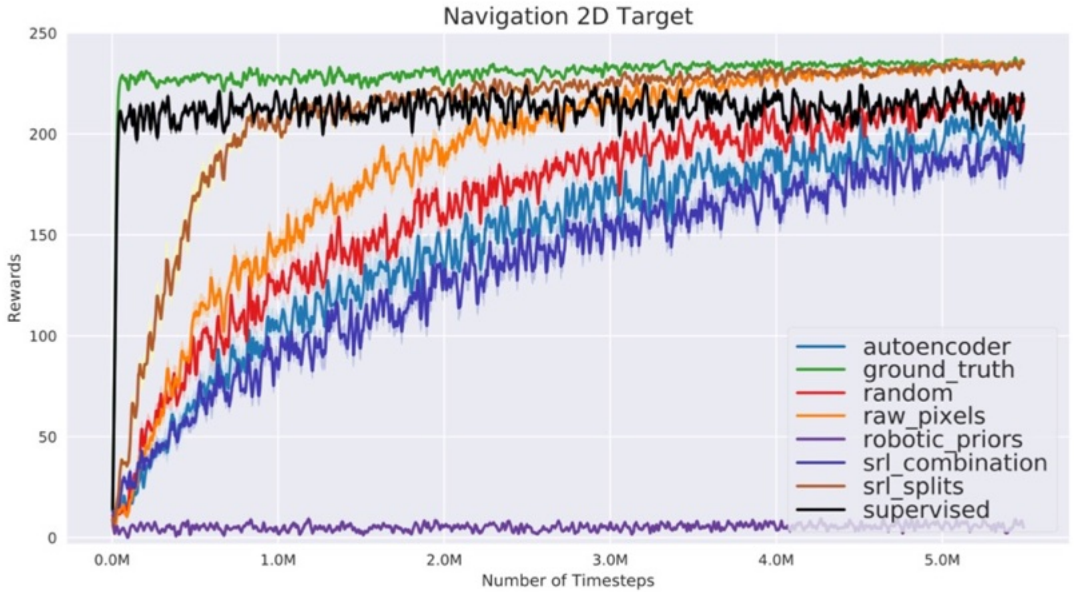
- Structure / disentangle / split state representation
- Forward/inverse models -> robot state
- Autoencoder/reward -> environment state

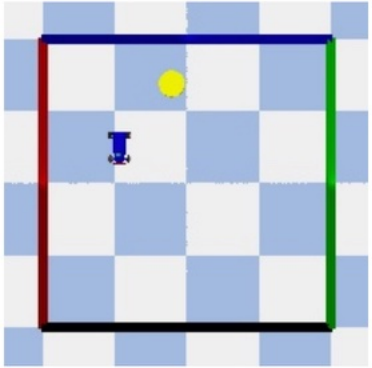
SRL - Split model

[Raffin et al. SPIRL19]

Learning structured state representation

- Can learn representation with moving goal
- Better sample efficiency / robustness



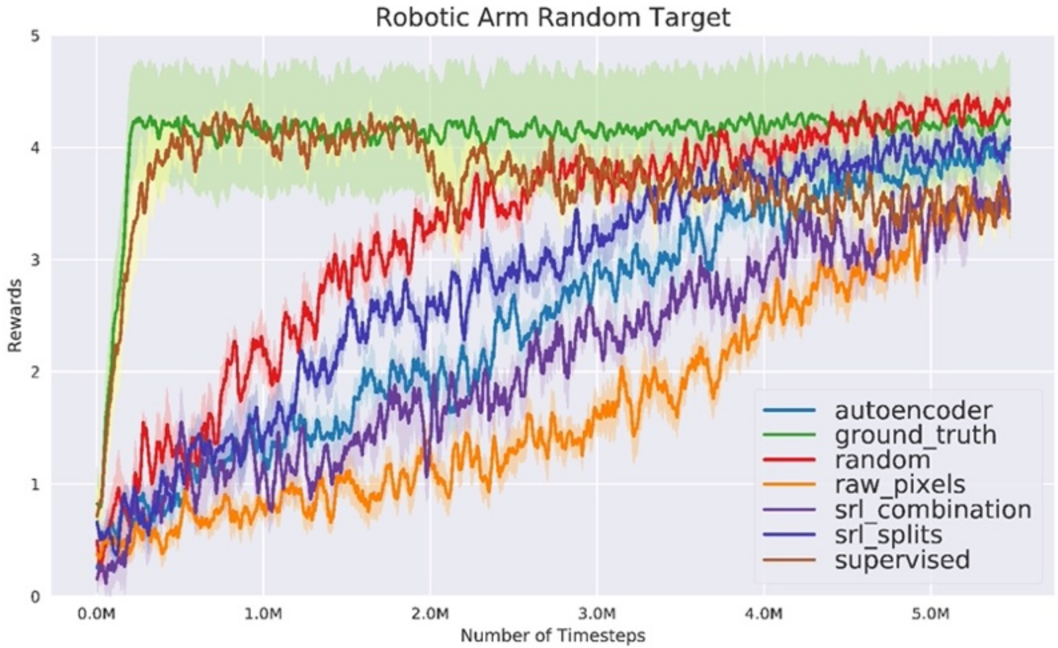


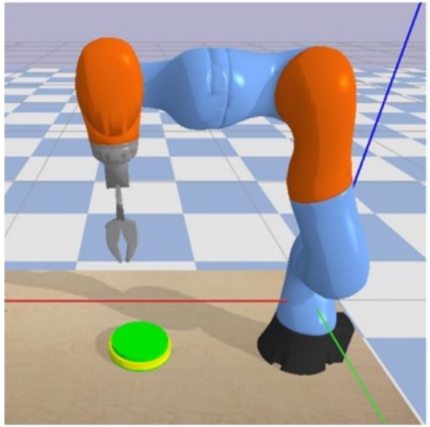
SRL - Split model

[Raffin et al. SPIRL19]

Learning structured state representation

- But not so efficient on more complex tasks





Continual
State Representation Learning

Continual Learning

Review paper
[Lesort et al, IF20]

Learning when tasks evolve

- New samples in existing classes / New classes
- Without forgetting previous tasks
- Gradient descent (fine tuning) will forget....

Several existing approaches

- Rehearsal (memorize old information to replay it)
- Regularization of important weights (e.g. EWC)
- Architectural evolution
- Generative Replay (use generative model to memorize old information)

S-TRIGGER (Self-TRIGgered GEnerative Replay)

[Caselle-Dupré et al., IJCNN21]

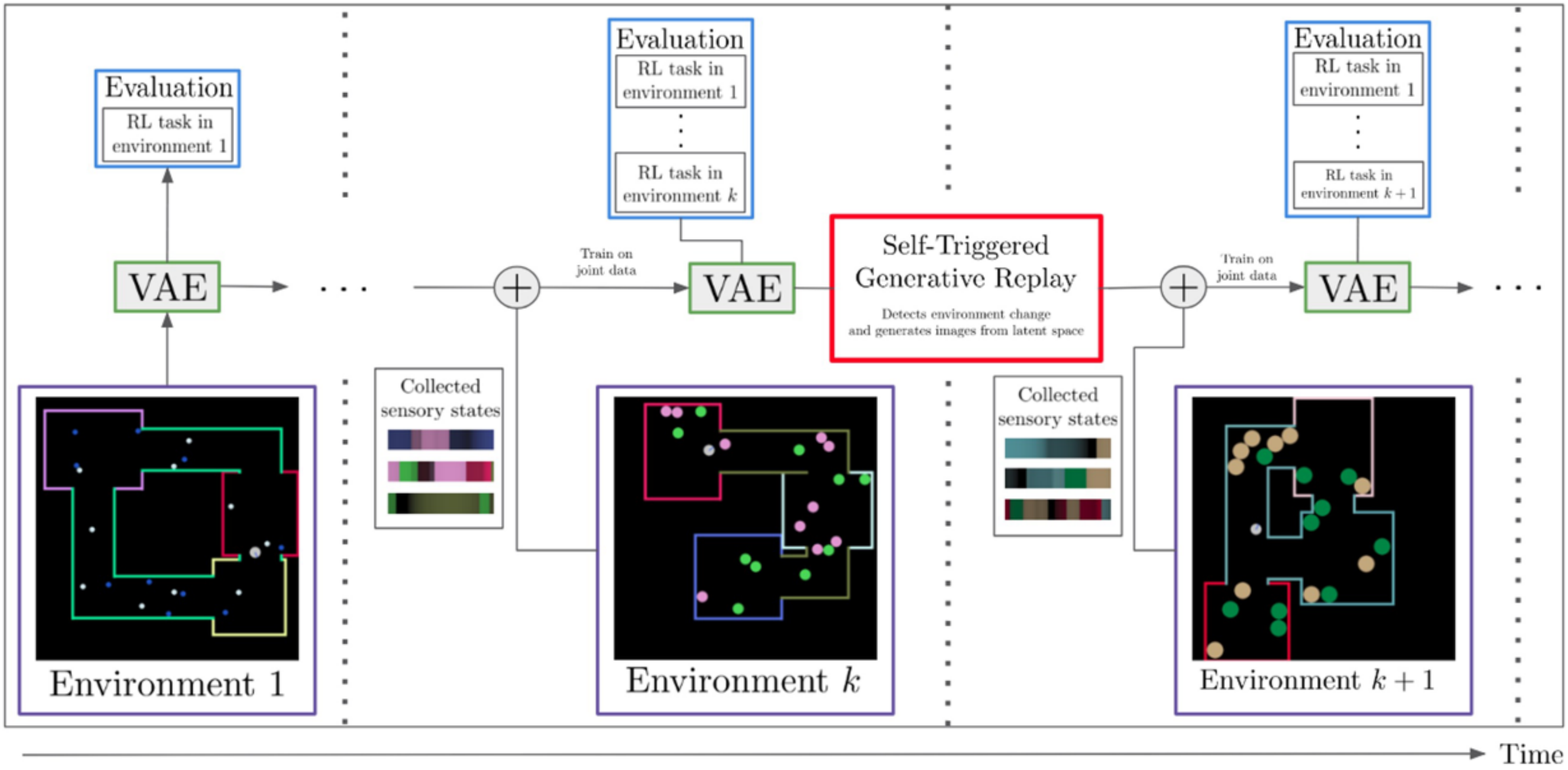
Continual State Representation Learning

- Agent facing tasks in sequence
- Environment appearance will change (and possibly reward)
- Can we detect the modifications automatically ?
- Can we adapt state representation without forgetting ?

Using VAE for SRL

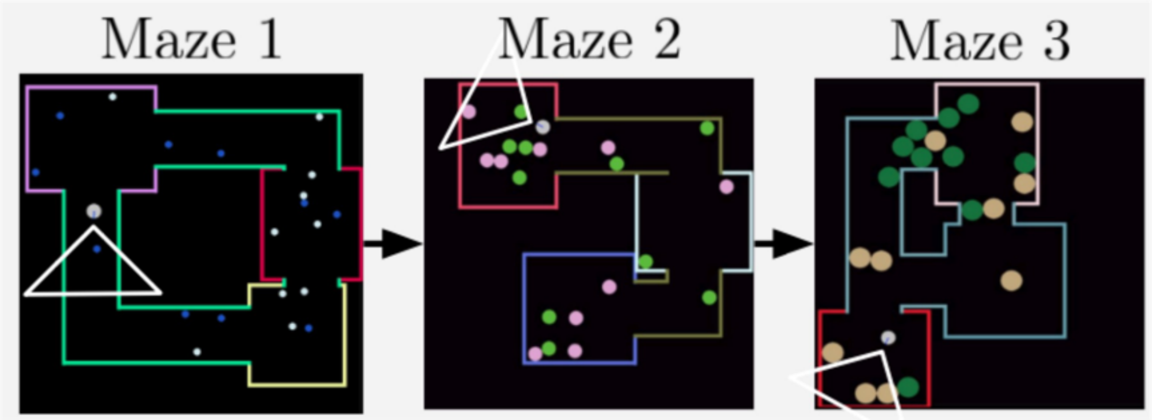
- Learn relevant state (with limitations…)
- Detect modification : statistical test on reconstruction error (Welch's t-test)
- Use VAE for Generative Replay in order to update State Representation

S-TRIGGER (Self-TRIGgered GEnerative Replay)

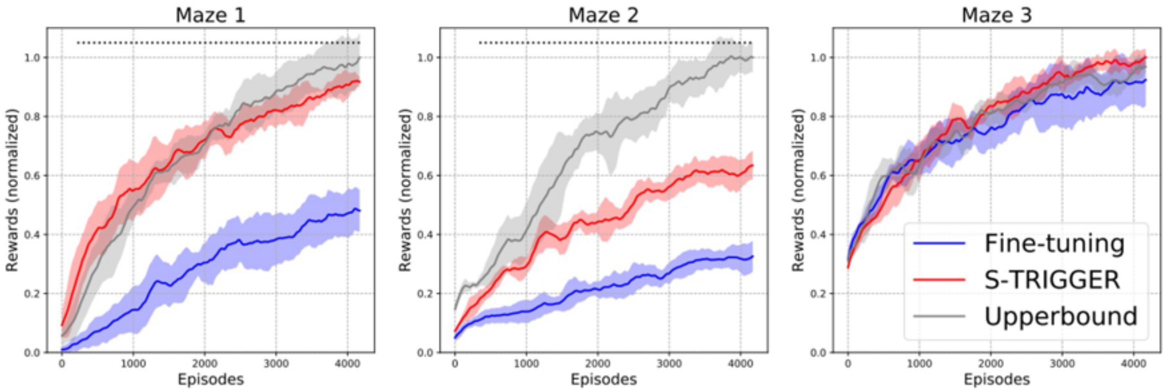


S-TRIGGER (Self-TRIGgered GEnerative Replay)

Envs:



RL with final state:



S-TRIGGER (Self-TRIGgered GEnerative Replay)

Advantages

- Limited forgetting of previously acquired information
- No need to access to past data
- Bounded system size
- Automatic detection of environment changes

Limitations

- Focuses on appearance only, don’t detect change in dynamics or task
- Rely on VAE, limited for more complex tasks

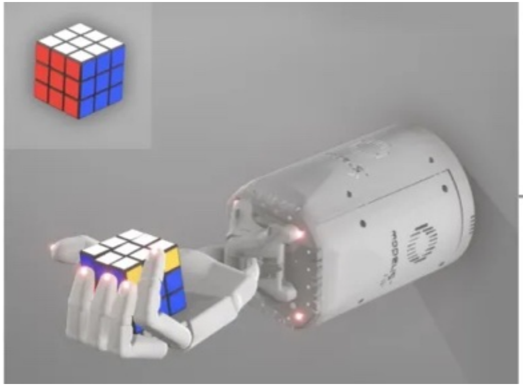
Robust
Reinforcement Learning

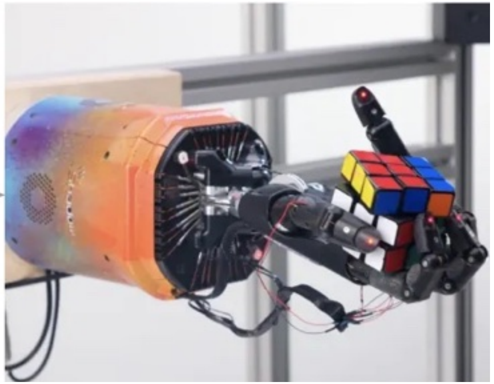
Robustness in RL

Generalization in RL

- Policies are often evaluated in their training environment

  (train == test !!)
- Controllers would need robustness to irrelevant appearance / dynamics changes
- One example is when training in simulation / testing in real life



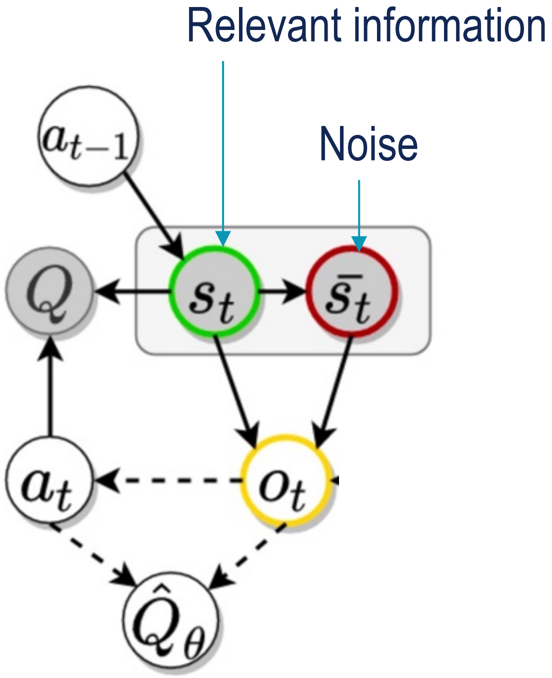


Robustness in RL

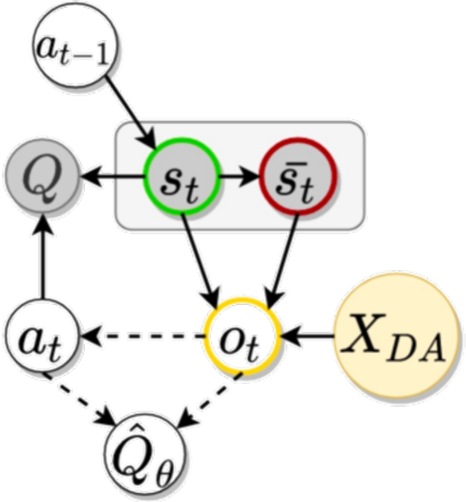
Sim to Real

- Because RL still very sample hungry
- Need to face a large domain gap : Sim2Real challenge
- Should rely on relevant info (e.g., 3D pose), and discard irrelevant features (e.g., appearance)
- Existing solutions : Data augmentation, domain randomization …
- VIBR : better exploitation of domain randomization

[Dupuis et al., 23]

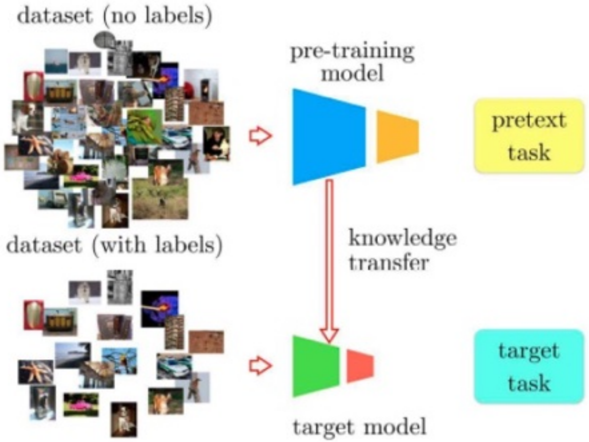


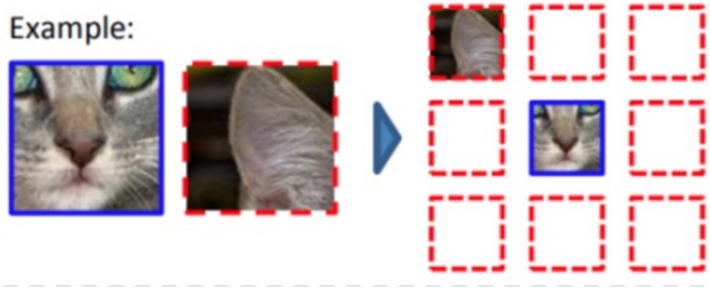
Data augmentation



Train representations insensitive to added noise

- Common in all deep learning models
- Self-supervised representation learning in Deep Learning



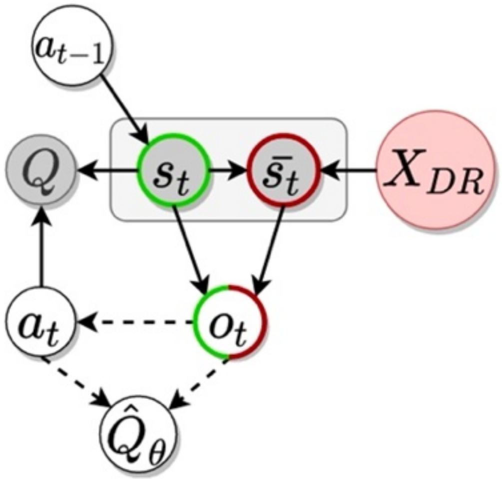


- Many possible pretext tasks : rotation, localization, inpainting, remove data augmentation…

Domain randomization

Train representation insensitive to noise in the simulation parameters

- Well adapted to RL and Sim2Real transfer
- More information about underlying state than Data Augmentation
- Can focus / control robustness to particular features



Distracting Control Suite

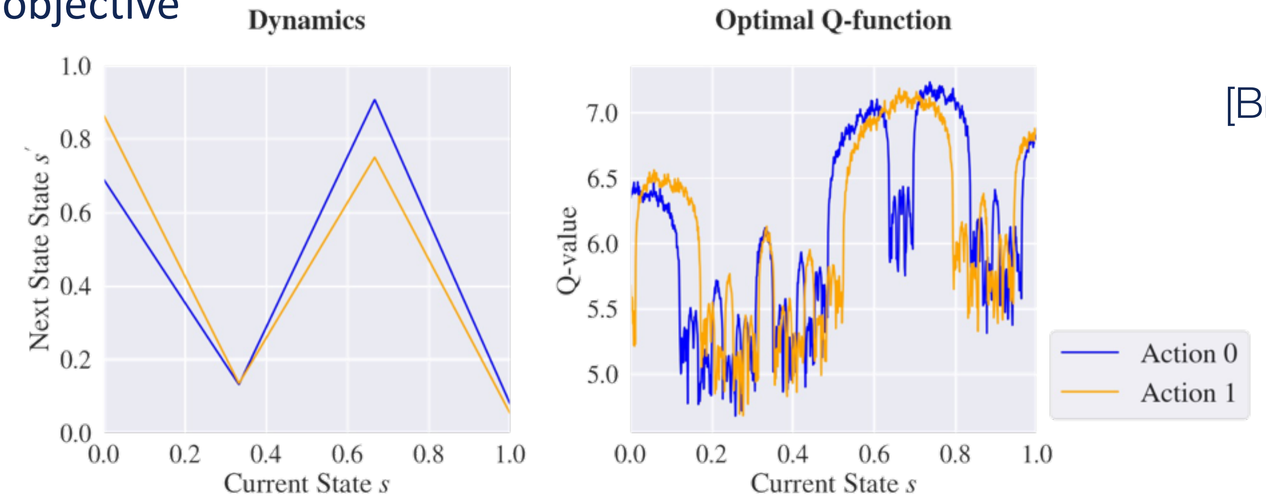
Based upon Deepmind Control Suite

- 6 control problems in a physics simulation
- Different action spaces (joint positions/velocities)
- Different reward modalities (dense and sparse)
- High-dimensional state (image 100x100)
- Multiple variants with a curriculum of visual distractions
  - Camera position (static or dynamic) / Body color /Background videos

Invariant representation or value function ?

State Representation Learning

- Learn a feature vector invariant to noise
- In general quite difficult, much more information than what RL needs
- Representation learning objective is not aligned with true task objective



[Brellman et al. 23]

Invariant representation or value function ?

State Representation Learning

- Same model capacity for two tasks instead of one: RL and representation learning : trade-off between the two

Invariant value function

- In value-based RL, we only need robustness of value function
- Learning invariant scalar value function is much simpler

Invariant prediction is a better option when sufficient

$\boxed{\text{1st trick}}$

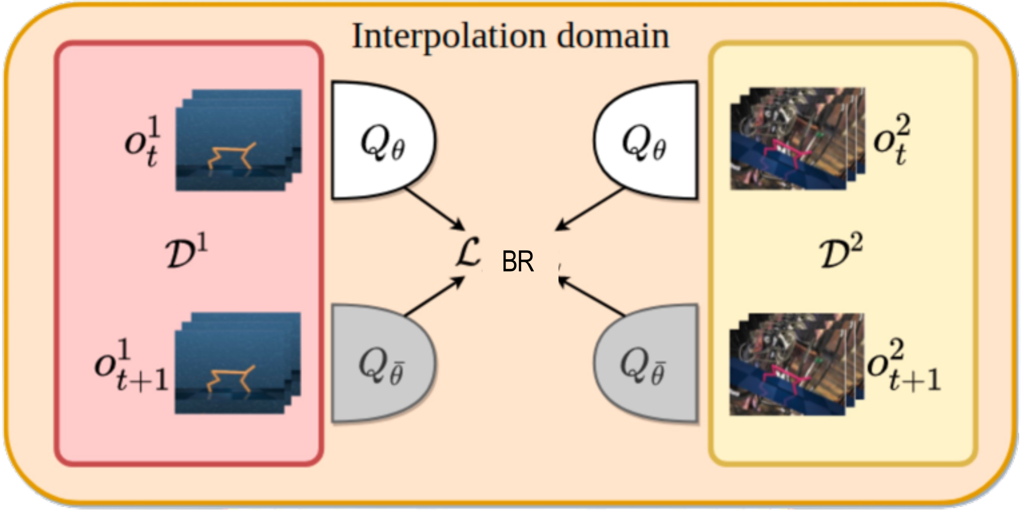
- But will loose generalization to other task

View-Invariant Bellman Residuals (VIBR)

Invariant value function learning using Domain Randomization

- Take advantage of simulation to generate two images with same state
- Use Bellman Residual in each domain
- Use Bellman Residual in cross domains

2nd trick



$$(\mathcal{B}^\pi Q)(s,a) = Q^\pi(s,a) - \mathcal{T}^\pi Q^\pi(s,a)$$

$$\left\{\begin{array}{l} Q^1 - TQ^1 \\ Q^1 - TQ^2 \\ Q^2 - TQ^1 \\ Q^2 - TQ^2 \end{array}\right.$$

$$\mathcal{L}_{\mathrm{BR}}(k,l) := \|\mathcal{B}^\pi Q_\theta(x^k, x^l)\|^2$$

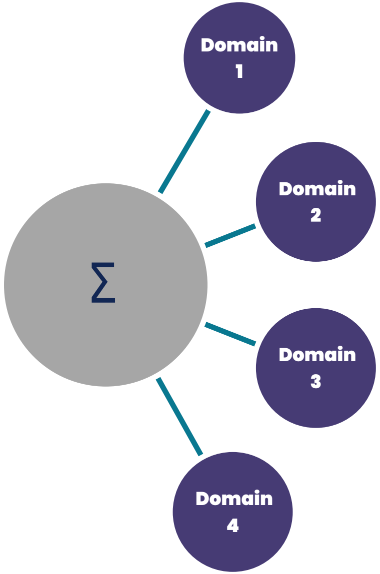
$$\hat{\mathbb{E}}\left[\mathcal{L}_{\mathrm{BR}}(k,l)\right] = \frac{1}{K^2}\sum_{k,l}\mathcal{L}_{\mathrm{BR}}(k,l)$$

(ERM)

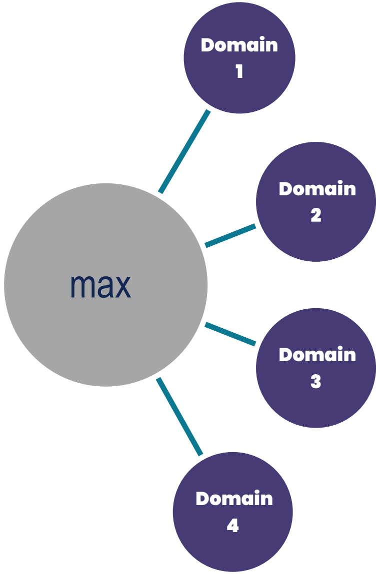
Risk Minimization

Train a model over **K distinct training domains** to minimize the **OOD risk**

Empirical Risk Minimization

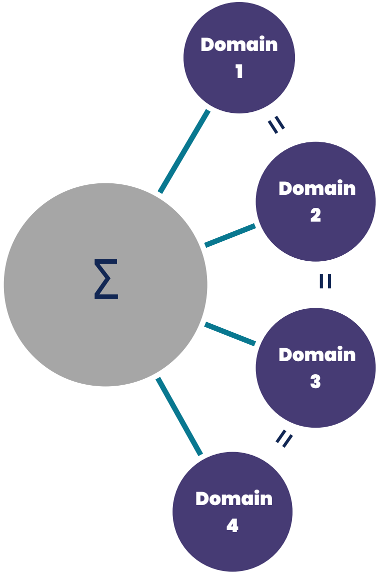


Robust Optimization



Risk Invariance

3rd trick



View-Invariant Bellman Residuals (VIBR)

Invariant value function / cross domain Bellman residuals / risk invariance

Training objective: $\mathcal{L}_{\mathrm{BR}}(k,l) := \|\mathcal{B}^{\pi} Q_{\theta}(x^k, x^l)\|^2$ (Pairwise Bellman residuals)

$$\mathcal{L}_{\mathrm{VIBR}} = \widehat{\mathbb{E}}_{(k,l)}\left[\mathcal{L}_{\mathrm{BR}}(k,l)\right] + \beta\widehat{\mathrm{Var}}(\mathcal{L}_{\mathrm{BR}}(k,l))$$

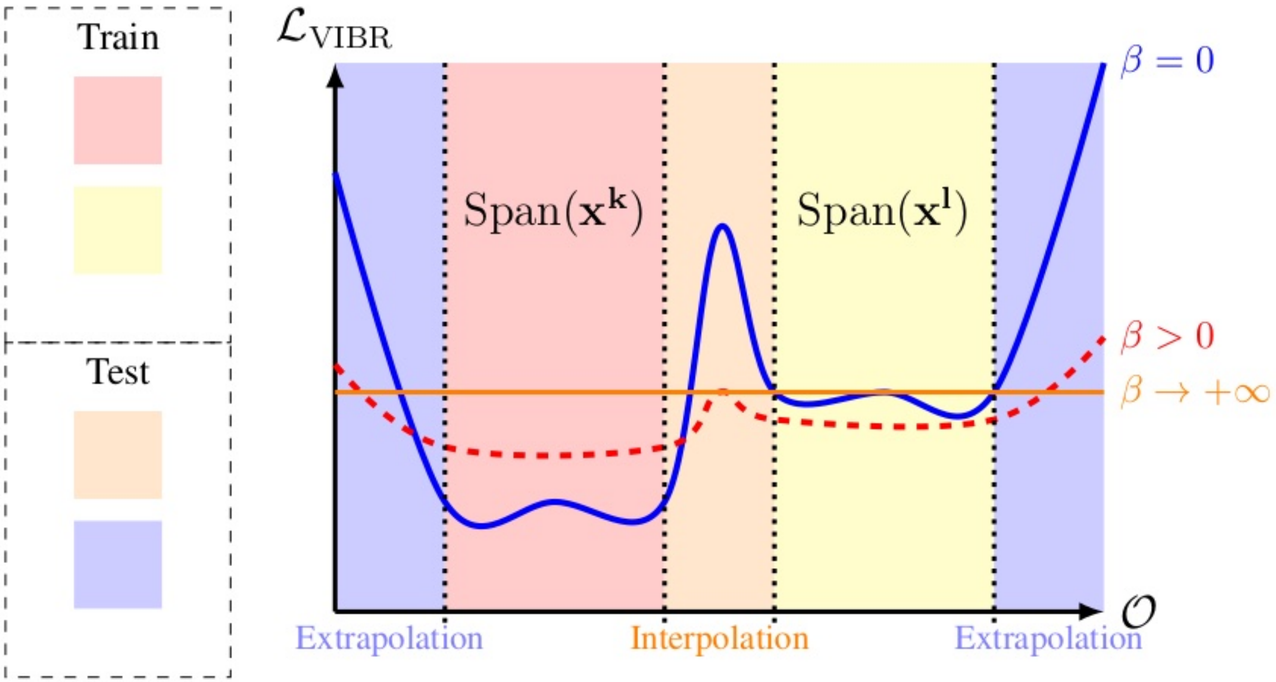
$$\text{where} \quad \widehat{\mathrm{Var}}(\mathcal{L}_{\mathrm{BR}}(k,l)) = \frac{1}{K^2}\sum_{k,l}\Big(\mathcal{L}_{\mathrm{BR}}(k,l) - \widehat{\mathbb{E}}\left[\mathcal{L}_{\mathrm{BR}}(k,l)\right]\Big)^2$$

(Risk Invariance)

[Dupuis et al., 23]

Risk Minimization

Intuition of the effect on generalization



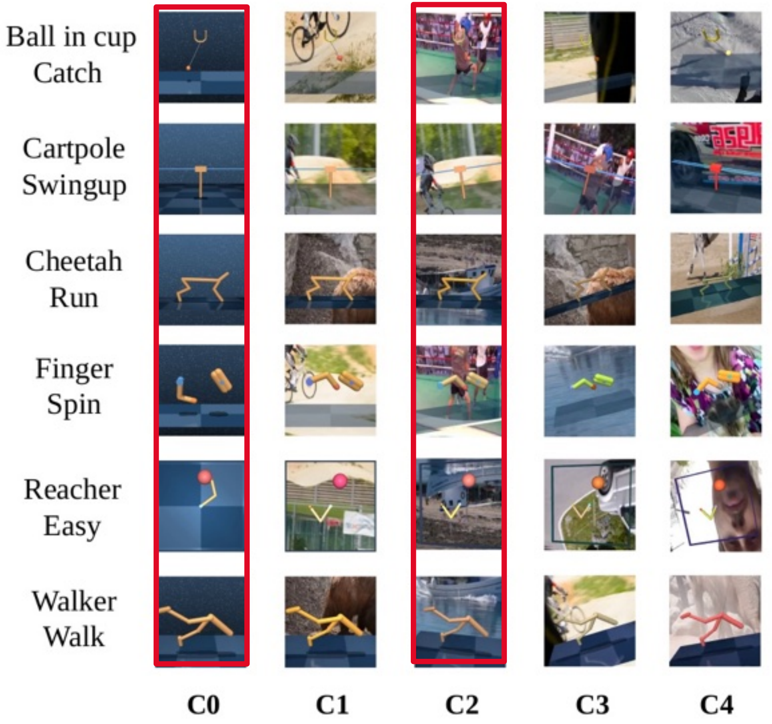
Comparison with baselines

Pure RL

- **DrQ:** Regularization of Q-value network by data augmentation + ensembling

RL + Representation Learning

- **CURL:** Contrastive learning auxiliary task
- **SPR**: Self-supervised next latent state prediction auxiliary task
- **DBC**: Learns task-relevant representations with a metric-based self-supervised objective
- **FM:** Naïve baseline of feature matching (MSE)



VIBR trained on C0 and C2

Robust Evaluation

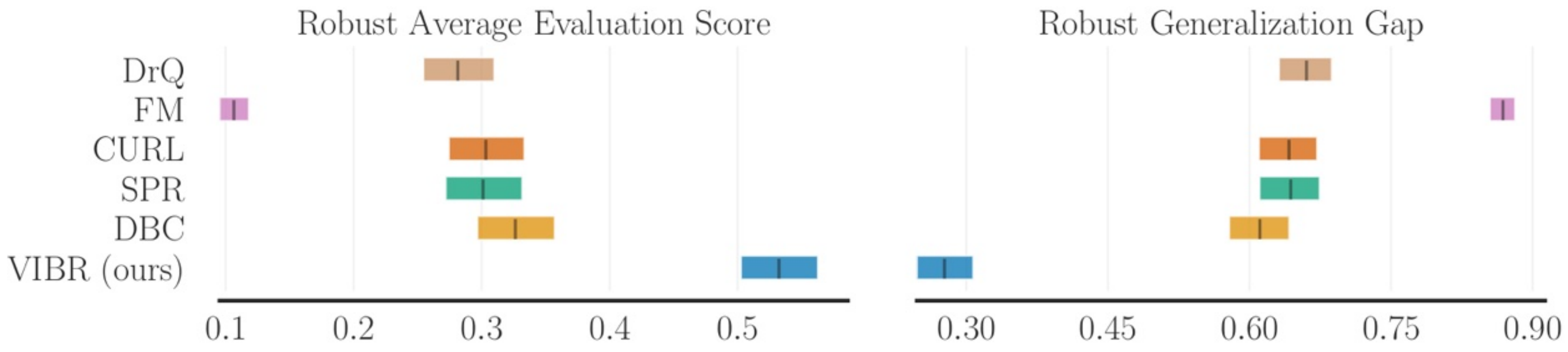
Results over 4 random seeds

- Normalized Return per episode
- Bootstrapped metrics over seed and evaluation episode:

Robust Average (IQM)

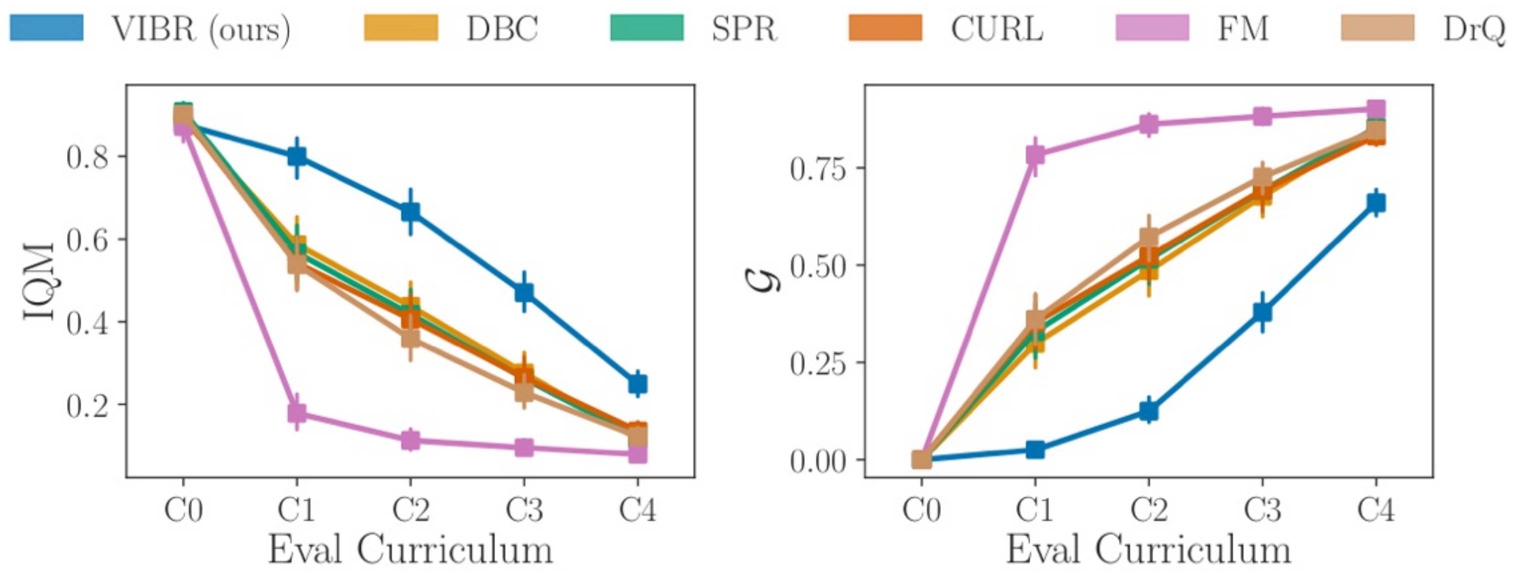
Generalization Gap $\mathcal{G}(C_i) = 1 - \frac{\mathrm{IQM}(C_i)}{\mathrm{IQM}(C0)}$

https://github.com/google-research/rliable



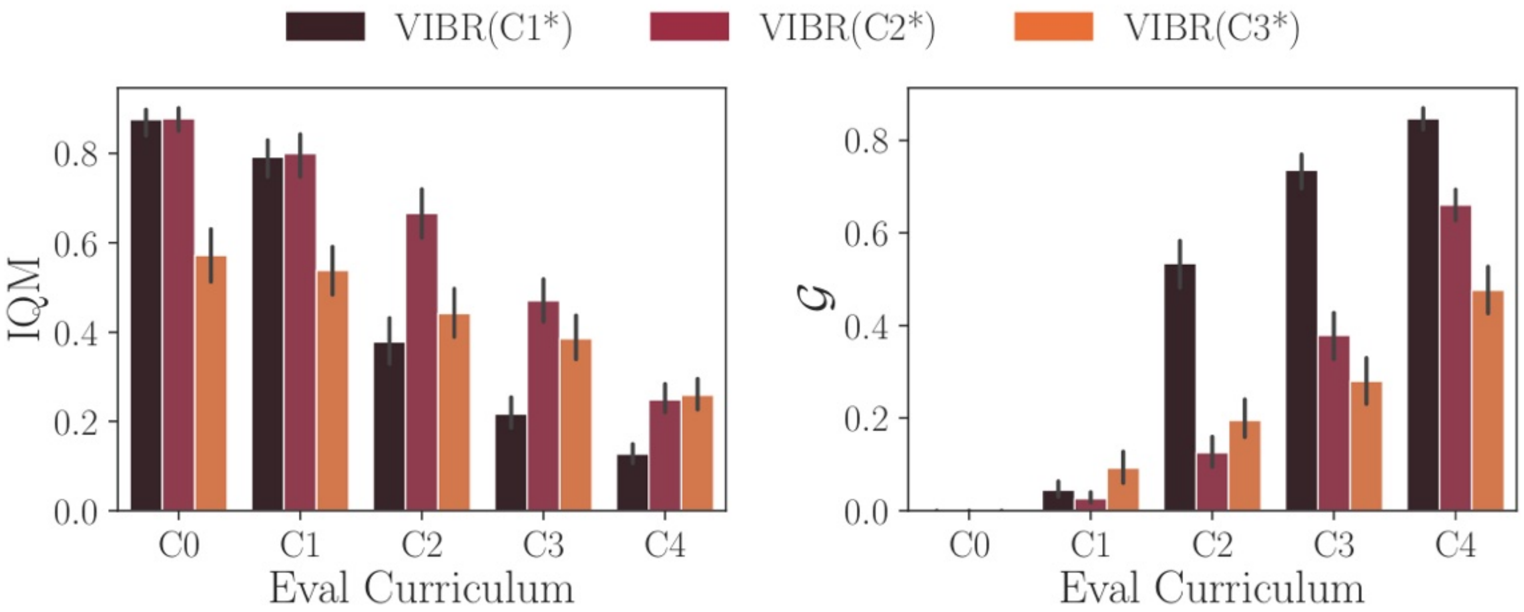
Detailed results per curriculum

VIBR flattens the performance curve across domains

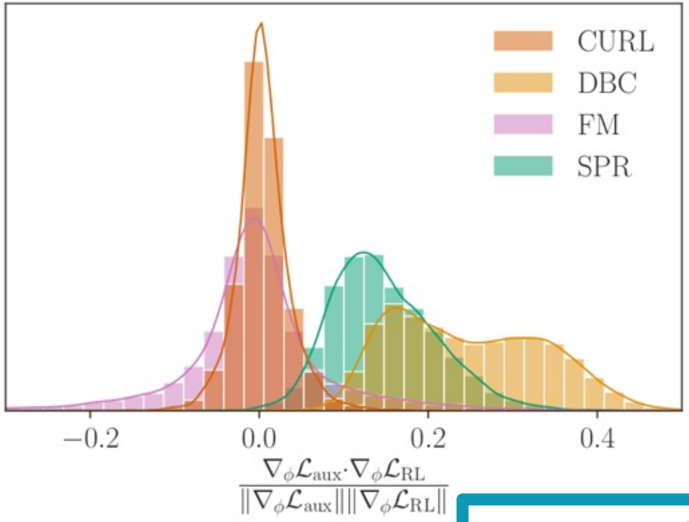


Detailed results per curriculum

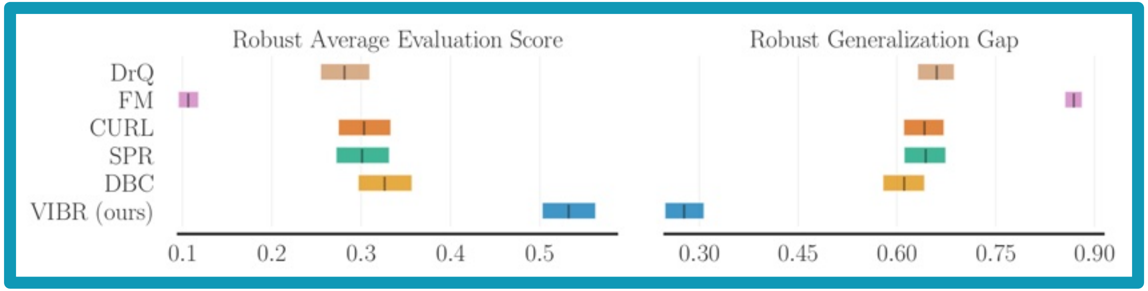
Training distribution matters



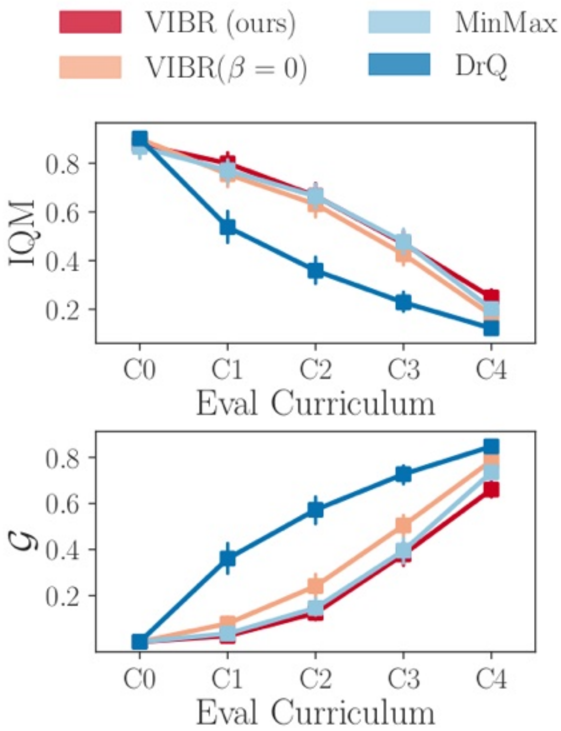
Failure case of invariant representation learning

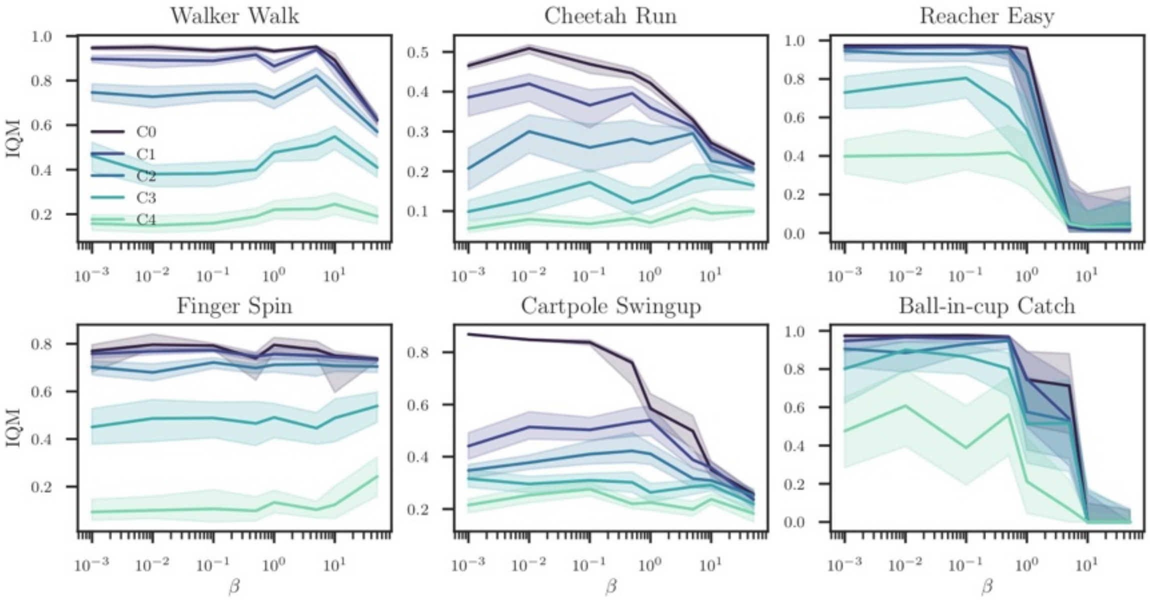


- FM & CURL auxiliary tasks are completely orthogonal to RL
- SPR & DBC are more aligned with RL
- Better alignment seem to help performance a little (expected)
- FM completely fails: auxiliary task ignored by model



Impact of variance regularization vs multi-view training





Optimal choice of beta depends on the task

Conclusion

Reinforcement learning has difficulties linked to the robotics context, but can exploit constraints/knowledge

Take advantage of the domain

- Exploit constraints on relevant info (low dim, controllable, predictable…)
- Exploit unsupervised (self supervised) learning
- Learn in simulation using easy to simulate features (e.g. 3D motion)
- Exploit efficiently domain randomization

Many approaches

- Many existing approaches that can be combined
- Proposed a new way to combine AE & models, perform continual learning, increase invariance to noise…

Perspectives

Very active area

- Many Sim to Real transfer approaches (domain randomization, domain adaptation, …)
- Many new state representation learning approaches associated to unsupervised pretraining of CNNs
- Some fixed representation may be useful (e.g., Fourrier features)

[Brellman et al. 21]

- Define / improve representation disentanglement (explicability)
- Merge everything ?
  - Supervised/self supervised pre-training in simulation with SRL, randomization, view invariance,…
  - Ensure disentanglement/interpretability in simulation
  - Fine tuning on real data with offline RL, or online with SRL as auxiliary tasks

Behind these results

Students

- Hugo Caselles-Dupré
- Timothée Lesort
- Antonin Raffin
- Ashley Hill
- René Traoré
- David Brellmann
- Tom Dupuis

Colleagues

- Natalia Diaz
- Michael Garcia Ortiz
- Jean François Goudou
- Quoc Cong Pham
- Jaonary Rabarisoa
- Goran Frehse

Projects

- H2020 DREAM
- H2020 VeriDREAM

References

[Lesort et al, NN18] **State Representation Learning for Control: An Overview.** Timothée Lesort, Natalia Díaz-Rodríguez, Jean-François Goudou, David Filliat Neural Networks, Elsevier, 2018, 108, pp.379-392.

[Raffin et al, SPIRL19] **Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics** Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, David Filliat SPiRL 2019 : Workshop on Structure and Priors in Reinforcement Learning at ICLR 2019.

[Lesort et al, IF20] **Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges.** Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., & Díaz-Rodríguez, N. (2020). Information fusion, 58, 52-68.

[Caselle-Dupré et al., IJCNN21] **S-trigger: Continual state representation learning via self-triggered generative replay.** Caselles-Dupré, H., Garcia-Ortiz, M., & Filliat, D. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7).

[Brellman et al. 21] **Fourier Features in Reinforcement Learning with Neural Networks** David Brellmann, Goran Frehse, David Fiilliat, submitted TMLR

[Dupuis et al. 23] **Vibr: Learning view-invariant value functions for robust visual control** Tom Dupuis, Jaonary Rabarisoa, Quoc Cuong Pham, David Fiilliat, CoLLAS 2023.

ENSTA
IP PARIS