10703 Deep Reinforcement Learning and Control

Russ Salakhutdinov
Machine Learning Department
rsalakhu@cs.cmu.edu

Slides developed and borrowed from
Katerina Fragkiadaki

Optimal Control, Trajectory Optimization, Learning Dynamics II

So far..

- Most Reinforcement Learning literature: Learning policies without knowing how the world works (model-free)
- Model based RL: build a model from data and use it to sample experience as opposed to just use experience by interacting with the world
- Imitation learning: learn from a teacher -> most recent and successful formulations suggest to train a classifier/regressor with data augmentation/scheduling to handle compounding errors
- Planners (e.g. Monte Carlo Tree Search)-> Search for actions while knowing how the world works (model-based)->Search in *Discrete* space.
- Imitation learning: learn from a planner: imitate MCTS to learn to play Atari games better than what you would get from model-free RL

So far..

- Most Reinforcement Learning literature: Learning policies without knowing how the world works (model-free)
- Model based RL: build a model from data and use it to sample experience as opposed to just use experience by interacting with the world
- Imitation learning: learn from a teacher -> most recent and successful formulations suggest to train a classifier/regressor with data augmentation/scheduling to handle compounding errors
- Planners (e.g. Monte Carlo Tree Search)-> Search for actions while knowing how the world works (model-based)->Search in *Discrete* space.
- Imitation learning: learn from a planner: imitate MCTS to learn to play Atari games better than what you would get from model-free RL
- Optimal Control: Search for actions while knowing how the world works (model-based) -> Search in Continuous space. (We will use derivates).

So far..

- Optimal Control, trajectory optimization formulation
- Special but important case: Linear dynamics, quadratic costs (LQR)
- iterative-LQR / Differential Dynamic programming for Non-linear dynamical systems
- Examples of when it works and when it does not
- Learning Dynamics: Global Models

Optimal Control (Open Loop)

- The optimal control problem:

$$\min_{x,u} \sum_{t=0}^{T} c_t(x_t, u_t)$$

$$\text{s.t. } x_0 = \bar{x}_0$$

$$x_{t+1} = f(x_t, u_t) \quad t = 0, ..., T-1$$

Optimal Control (Open Loop)

- The optimal control problem:

$$\min_{x,u} \sum_{t=0}^{T} c_t(x_t, u_t)$$

$$\text{s.t. } x_0 = \bar{x}_0$$

$$x_{t+1} = f(x_t, u_t) \quad t = 0, ..., T-1$$

- Solution:
  - Sequence of controls u and resulting state sequence x
- In general non-convex optimization problem, can be solved with sequential convex programming (SCP): https://stanford.edu/class/ee364b/lectures/seq_slides.pdf

Optimal Control (Closed Loop)

Given: $\bar{x}_0$

For $t = 0, 1, 2, ..., T$

- Solve

$$\min_{x,u} \sum_{k=t}^{T} c_k(x_k, u_k)$$
$$\text{s.t. } x_{k+1} = f(x_k, u_k), \quad \forall k \in \{t, t+1, ..., T-1\}$$
$$x_t = \bar{x}_t$$

- Execute u_t
- Observe resulting state, $\bar{x}_{t+1}$
- Initialize with solution from $t-1$ to solve fast at time t

Linear case: LQR

- Very special case: Optimal Control for Linear Dynamic Systems and Quadratic Cost (a.k.a. LQ Regulator setting)
- Can solve continuous state-space optimal control problem exactly
- Running time: $O(Tn^3)$

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} \quad c(\mathbf{x}_1,\mathbf{u}_1) + c(f(\mathbf{x}_1,\mathbf{u}_1),\mathbf{u}_2) + \cdots + c(f(f(\ldots)\ldots),\mathbf{u}_T)$$

$$f(\mathbf{x}_t,\mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

linear

$$c(\mathbf{x}_t,\mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

quadratic

Non-linear case:Use iterative approximations!

First order Taylor expansion for the dynamics around a trajectory $\hat{x}_t, \hat{u}_t, t = 1 \cdots T$

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

Second order Taylor expansion for the cost around a trajectory $\hat{x}_t, \hat{u}_t, t = 1 \cdots T$:

$$c(\mathbf{x}_t, \mathbf{u}_t) \approx c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) + \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}^T \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \begin{bmatrix} \mathbf{x}_t - \hat{\mathbf{x}}_t \\ \mathbf{u}_t - \hat{\mathbf{u}}_t \end{bmatrix}$$

$$\bar{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \underbrace{\mathbf{F}_t}_{\nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}$$

$$\bar{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}^T \underbrace{\mathbf{C}_t}_{\nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)} \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \delta\mathbf{x}_t \\ \delta\mathbf{u}_t \end{bmatrix}^T \underbrace{\mathbf{c}_t}_{\nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)}$$

$$\delta\mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$$
$$\delta\mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t$$

Now we can run LQR with dynamics $\bar{f}$, cost $\bar{c}$, state $\delta\mathbf{x}_t$, and action $\delta\mathbf{u}_t$

Iterative LQR (i-LQR)

Initialization: Given $\hat{x}_0$, pick a random control sequence $\hat{u}_0 \ldots \hat{u}_T$ and obtain corresponding state sequence $\hat{x}_0 \ldots \hat{x}_T$

until convergence:

$$\mathbf{F}_t = \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ \forall t$$

$$\mathbf{c}_t = \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ \forall t$$

$$\mathbf{C}_t = \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ \forall t$$

Run LQR backward pass on state $\delta \mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and action $\delta \mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t \ \forall t$

Run forward pass with real nonlinear dynamics and $u_t = \hat{u}_t + K_t(x_t - \hat{x}_T) + k_t \ \forall t$

Update $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ based on states and actions in forward pass $\forall t$

Iterative LQR (i-LQR)

Initialization: Given $\hat{x}_0$, pick a random control sequence $\hat{u}_0 \ldots \hat{u}_T$ and obtain corresponding state sequence $\hat{x}_0 \ldots \hat{x}_T$

until convergence:

$$\mathbf{F}_t = \nabla_{\mathbf{x}_t, \mathbf{u}_t} f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ \forall t$$

$$\mathbf{c}_t = \nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ \forall t$$

$$\mathbf{C}_t = \nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ \forall t$$

Linear approximation around $\hat{x}, \hat{u}$

Find $\Delta u_t, t = 1 \ldots T$ so that $\hat{u}_t + \Delta u_t$ minimizes the linear approximation

Run LQR backward pass on state $\delta \mathbf{x}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$ and action $\delta \mathbf{u}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t \ \forall t$

Run forward pass with real nonlinear dynamics and $u_t = \hat{u}_t + K_t(x_t - \hat{x}_T) + k_t \ \forall t$

Update $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ based on states and actions in forward pass $\forall t$

Go to the $\hat{x}' = \hat{x} + \Delta x_t$ and $\hat{u}' = \hat{u} + \Delta u_t$

Model Predictive Control

- Model predictive control.
- At time t when asked to generate control input u_t, we could re-solve the control problem using iLQR over the time steps t through T

every time step:

observe the state $\mathbf{x}_t$

use iLQR to plan $\mathbf{u}_t, \ldots, \mathbf{u}_T$ to minimize $\sum_{t'=t}^{t+T} c(\mathbf{x}_{t'}, \mathbf{u}_{t'})$

execute action $\mathbf{u}_t$, discard $\mathbf{u}_{t+1}, \ldots, \mathbf{u}_{t+T}$

- Re-planning entire trajectory is often impractical -> in practice: replay over horizon H (receding horizon control)

i-LQR: When it works

Synthesis and stabilization of complex behaviors with online trajectory optimization

Yuval Tassa, Tom Erez and Emo Todorov

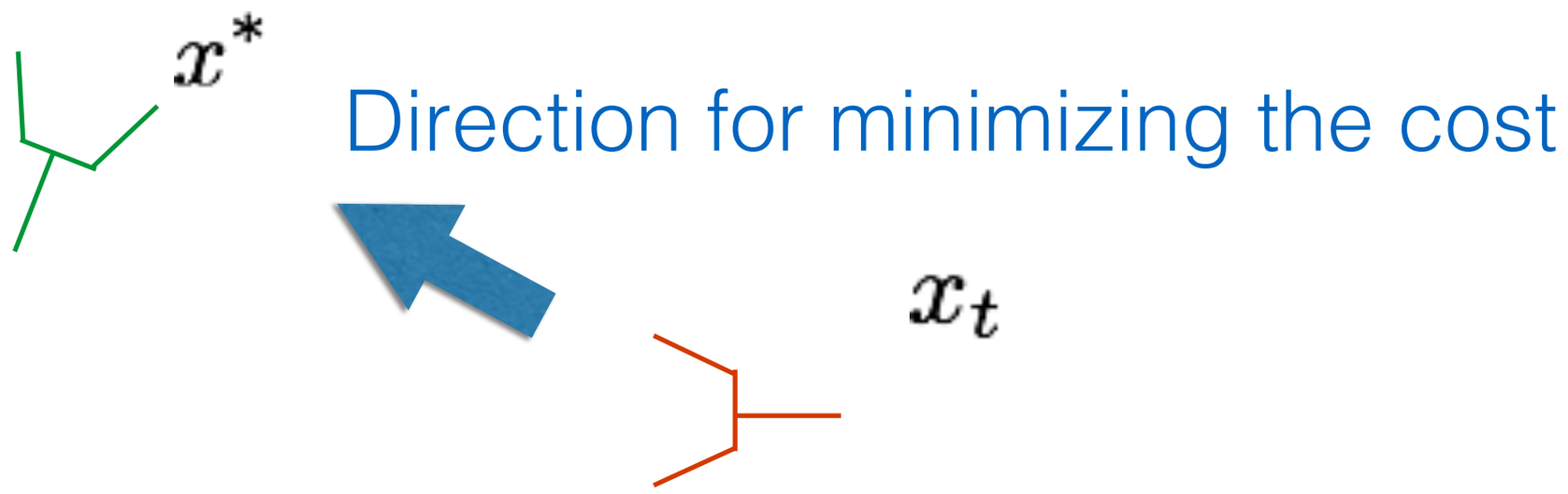
Movement Control Laboratory
University of Washington

IROS 2012

Synthesis and stabilization of complex behaviors with online trajectory optimization, Tassa, Erez, Todorov, IROS 2012

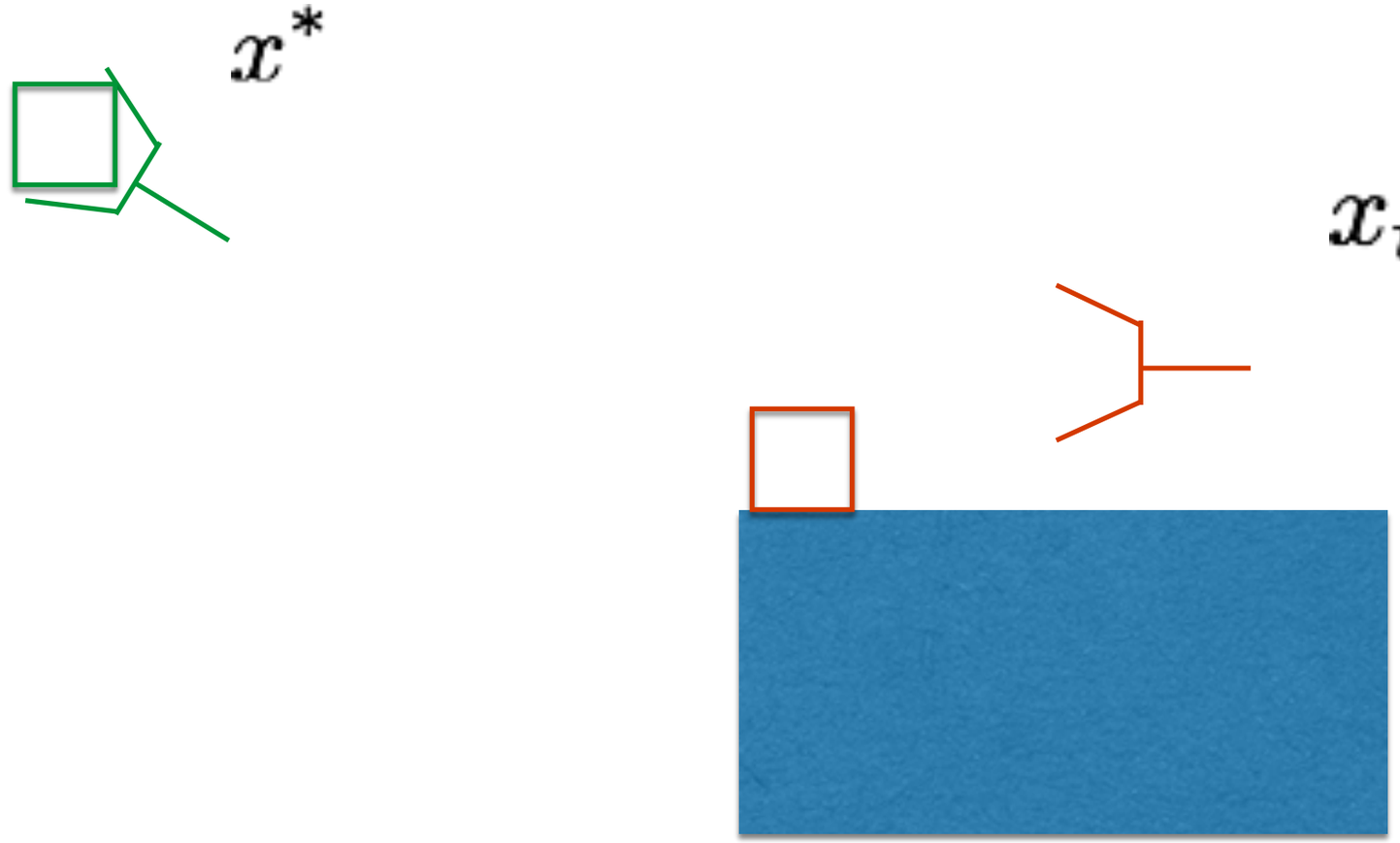
i-LQR: When it works

Cost: $\|x_t - x^*\|$



i-LQR: When it doesn't work

Cost: $\|x_t - x^*\|$



Due to discontinuities of contact, the local search fails! Solution?

Initialize using a human demonstration instead of random!

Learning Dexterous manipulation Policies from Experience and Imitation, Kumar, Gupta, Todorov, Levine 2016

Learning Dynamics

- So far we assumed we knew how the world works: $f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_{t+1}$
- We used discrete or continuous searches (e.g., MCTS(samples) and i-LQR(derivatives)) to look-ahead. We got good results, better than model-free RL. Dynamics help.
- However knowing the transition model (other than for rigid objects) is unrealistic! In fact, we do not have good physics simulators easily accessible for even basic things like turning a wheel. A huge part (deformable objects, object interactions, etc) we do not know how to model well.
- Even for rigid objects, where we know the equations of motion, we do not know the coefficients, e.g., friction, mass etc.
- Thus: instead of assuming them known, we need to learn dynamics.

Learning Dynamics

Newtonian Physics equations

VS

general parametric form (no priors from Physics knowledge)

System identification: when we assume the dynamics equations given and only have *few* unknown parameters

Much easier to learn but suffers from under-modeling, bad models

Neural networks: *tons* of unknown parameters

Very flexible, very hard to get it to generalize

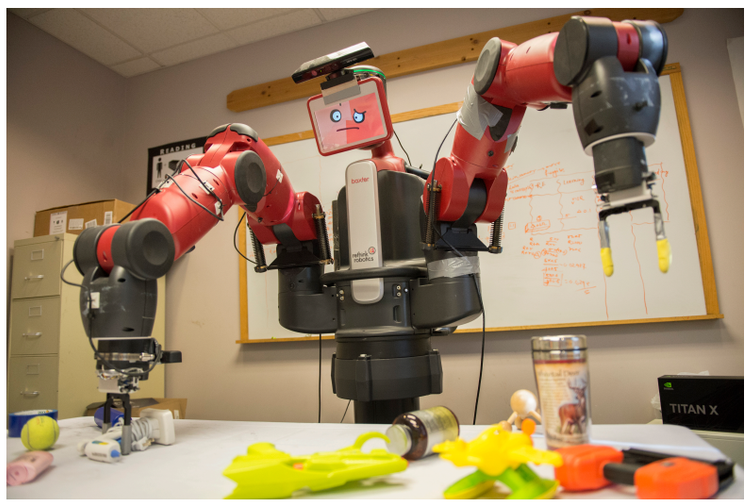
Learning Dynamics: Regression

Search with learnt dynamics:

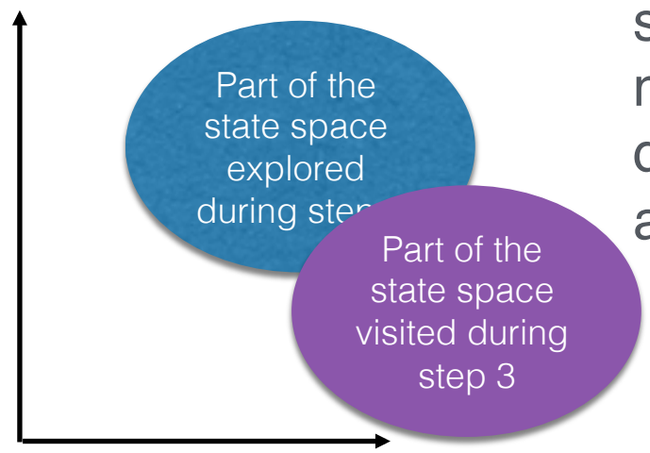
1. Run base policy $\pi_0(u_t|x_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(x, u, x')_i\}$
2. Learn dynamics model $f(x, u)$ to minimize $\sum_i \|f(x_i, u_i) - x'_i\|^2$
3. Plan under the learnt dynamics to choose actions (e.g., MCTS or i-LQR)

Let's apply it:

1. A robot randomly interacts (pokes objects) and collects data (x,u,x')
2. Fit dynamics model
3. Plan actions to accomplish a particular goal, e.g., push an object as far away as possible



What goes wrong:



state distribution mismatch between dynamics learning and policy execution

$$p_{\pi_f}(\mathbf{x}_t) \neq p_{\pi_0}(\mathbf{x}_t)$$

Learning Dynamics: DAGGER

1. Run base policy $\pi_0(u_t|x_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(x, u, x')_i\}$
2. Learn dynamics model $f(x, u)$ to minimize $\sum_i \|f(x_i, u_i) - x'_i\|^2$
3. Plan under the learnt dynamics to choose actions (e.g., MCTS or i-LQR)
4. Execute those actions and add the resulting data $\{(x, u, x')_j\}$ to $\mathcal{D}$

Learning Dynamics: DAGGER

1. Run base policy $\pi_0(u_t|x_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(x, u, x')_i\}$
2. Learn dynamics model $f(x, u)$ to minimize $\sum_i \|f(x_i, u_i) - x'_i\|^2$
3. Plan under the learnt dynamics to choose actions (e.g., MCTS or i-LQR)
4. Execute those actions and add the resulting data $\{(x, u, x')_j\}$ to $\mathcal{D}$

If the action sequence is long the model errors accumulate in time.

It is impossible our dynamic model to be perfect and tolerate long chaining.

It is also not biologically plausible, e.g., humans are very bad at predicting ball collisions accurately.

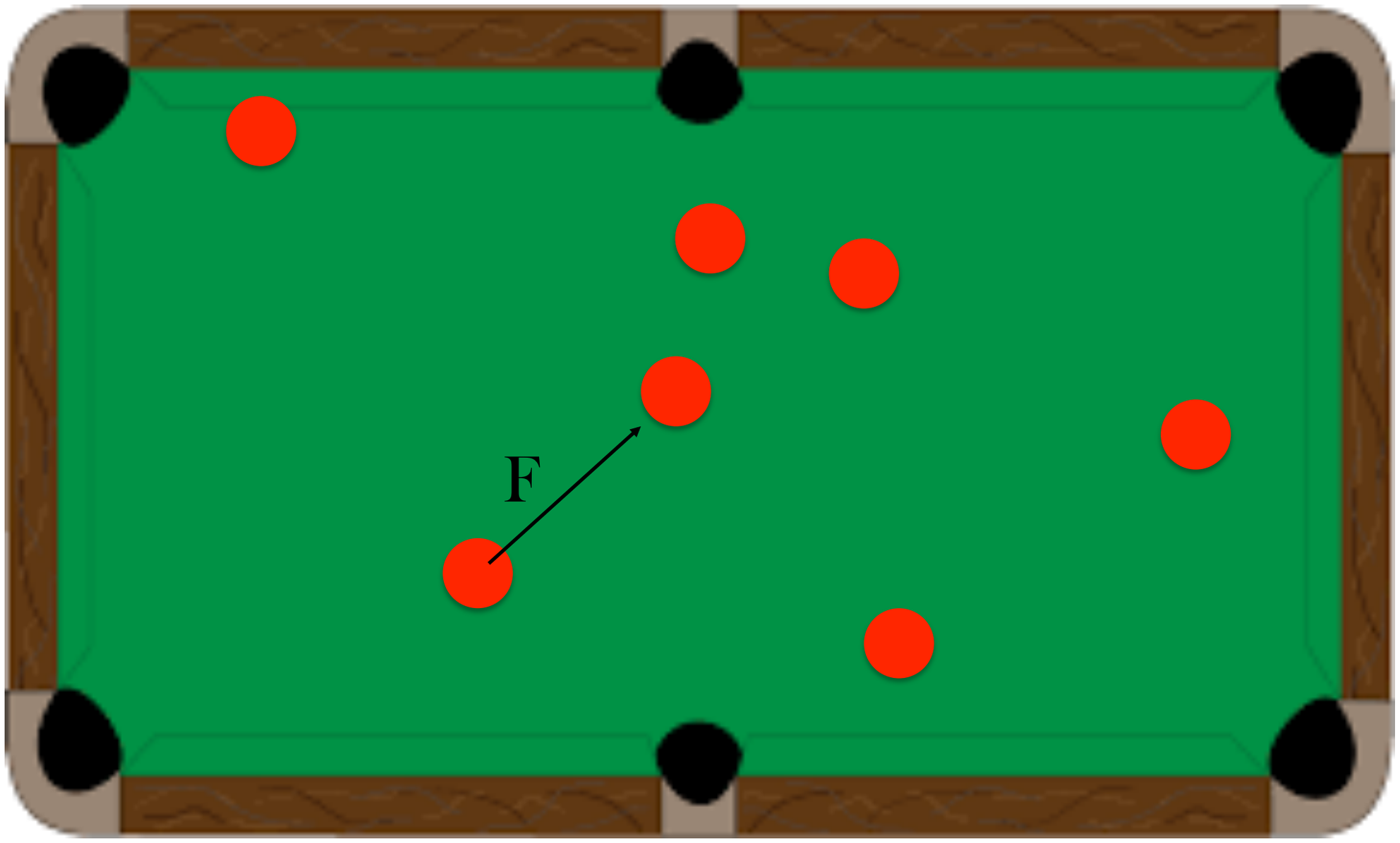
Learning Dynamics: MPC

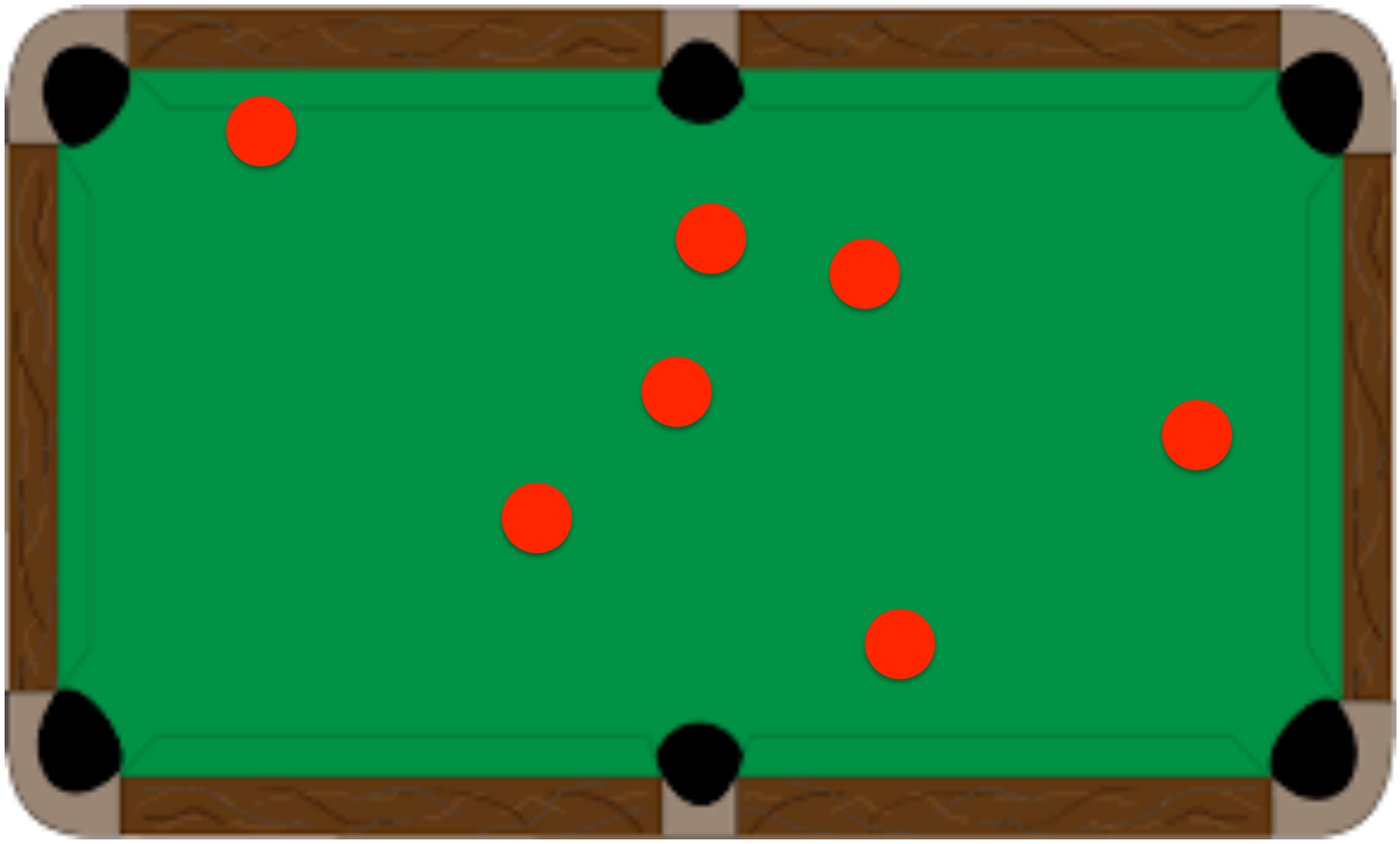
1. Run base policy $\pi_0(u_t|x_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(x, u, x')_i\}$
2. Learn dynamics model $f(x, u)$ to minimize $\sum_i \|f(x_i, u_i) - x'_i\|^2$
3. Plan under the learnt dynamics to choose actions (e.g., MCTS or i-LQR)
4. Execute the first planned action, observe resulting state x' (MPC)
5. Append (x, u, x') to dataset $\mathcal{D}$

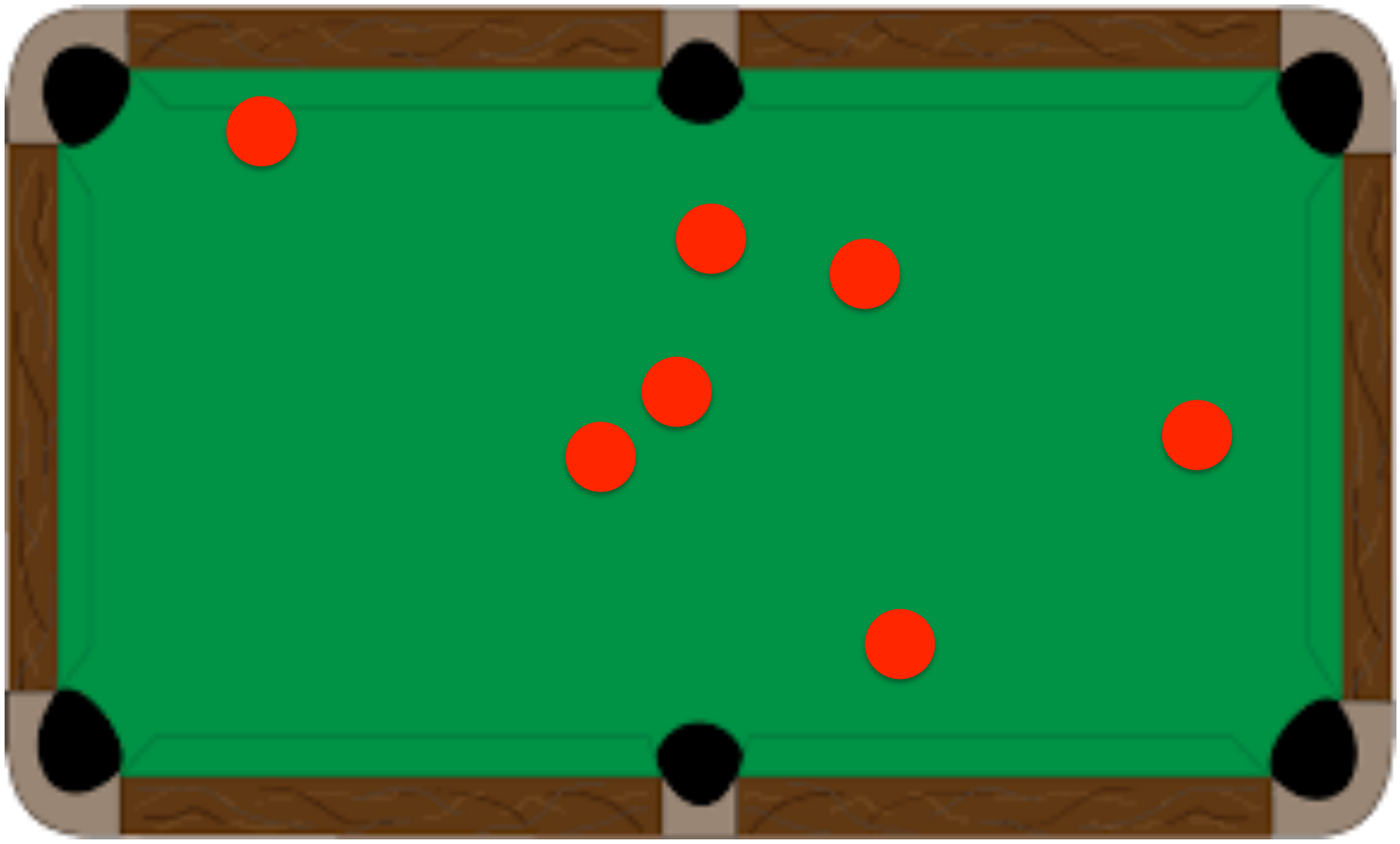
Learning Dynamics (Summary)

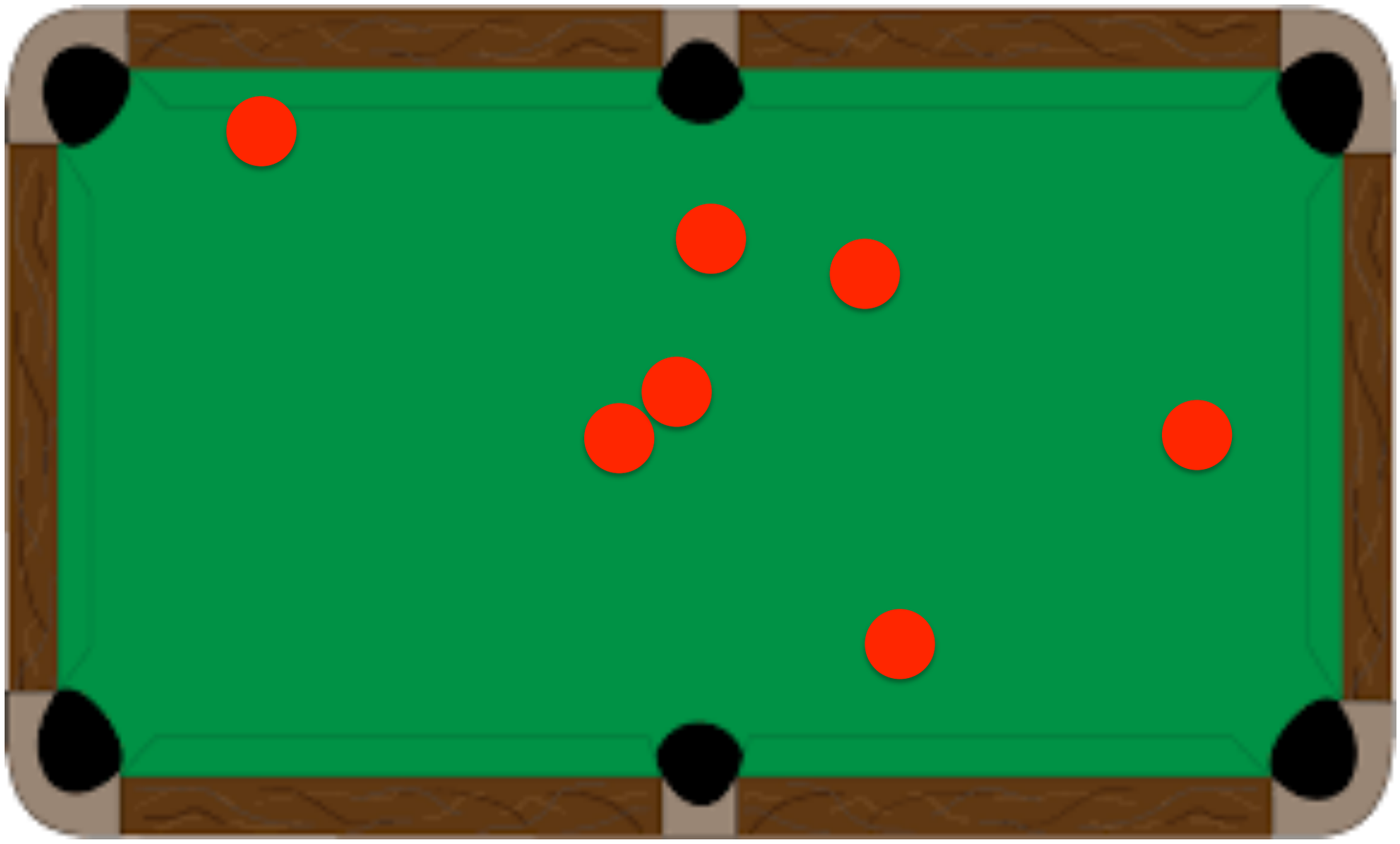
- Regression problem: collect random samples, train dynamics, plan
  - Pro: simple, no iterative procedure
  - Con: distribution mismatch problem
- DAGGER: iteratively collect data, replan, collect data
  - Pro: simple, solves distribution mismatch
  - Con: open loop plan might perform poorly, esp. in stochastic domains
- MPC: iteratively collect data using MPC (replan at each step)
  - Pro: robust to small model errors
  - Con: computationally expensive, but have a planning algorithm available

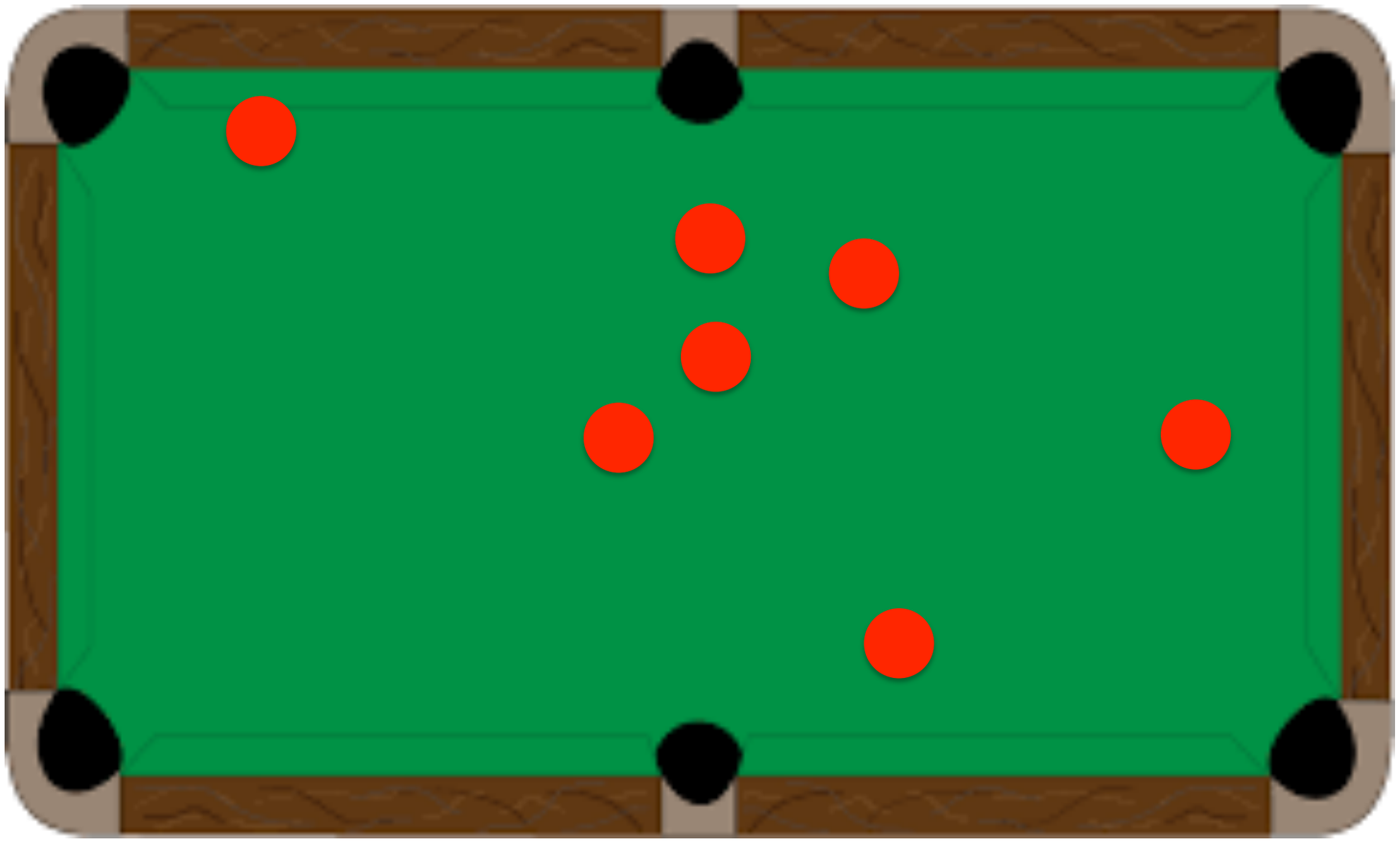
Examples of Learning Dynamics

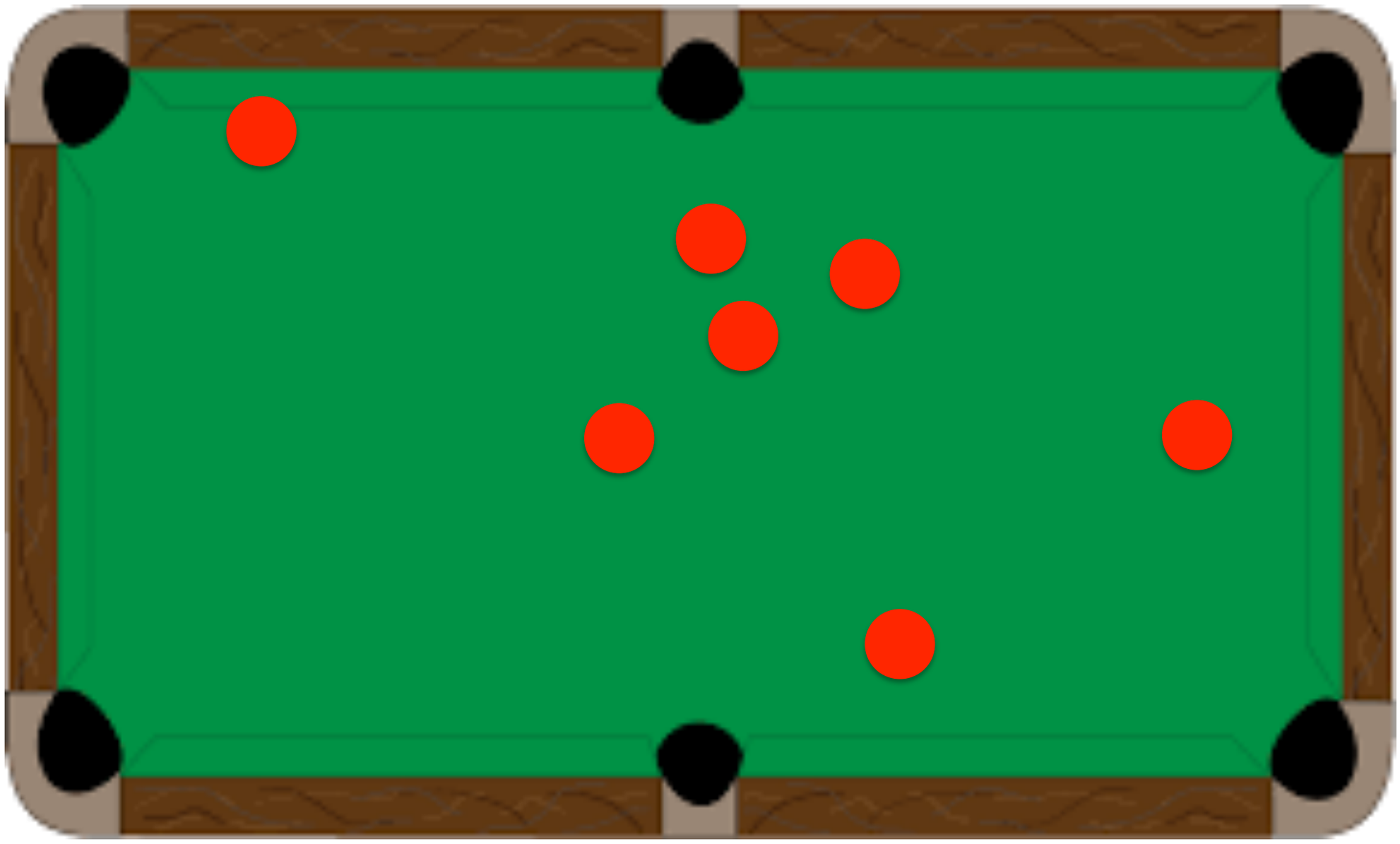

F











Learning Action-Conditioned Dynamics

➢ Newtonian Physics $\mathbf{F}, \mathbf{I}, \alpha, \mathbf{m}, k$

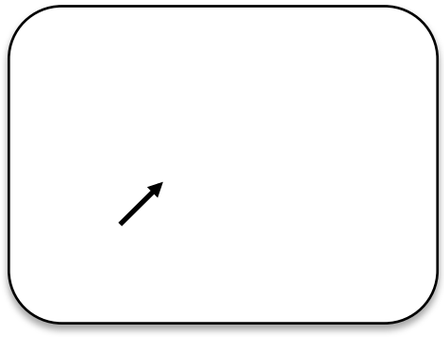
Galileo: Perceiving Physical Object Properties by Integrating a Physics Engine with Deep Learning, Wu et al.

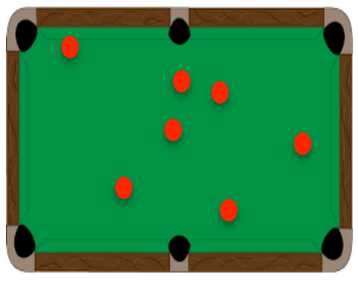
Newtonian Image Understanding: Unfolding the Dynamics of Objects in Static Images, Mottaghi et al.

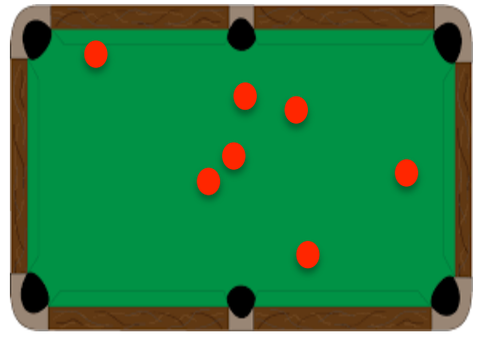
➢ Predictive Physics $(\mathbf{F}, \mathbf{x_{1 \cdots t}}) \rightarrow \mathbf{y_{t+1 \cdots t+T}}$

Learning Action-Conditioned Dynamics

$$(\mathbf{F}, \mathbf{x_{1 \cdots t}}) \rightarrow \mathbf{x_{t+1 \cdots t+T}}$$

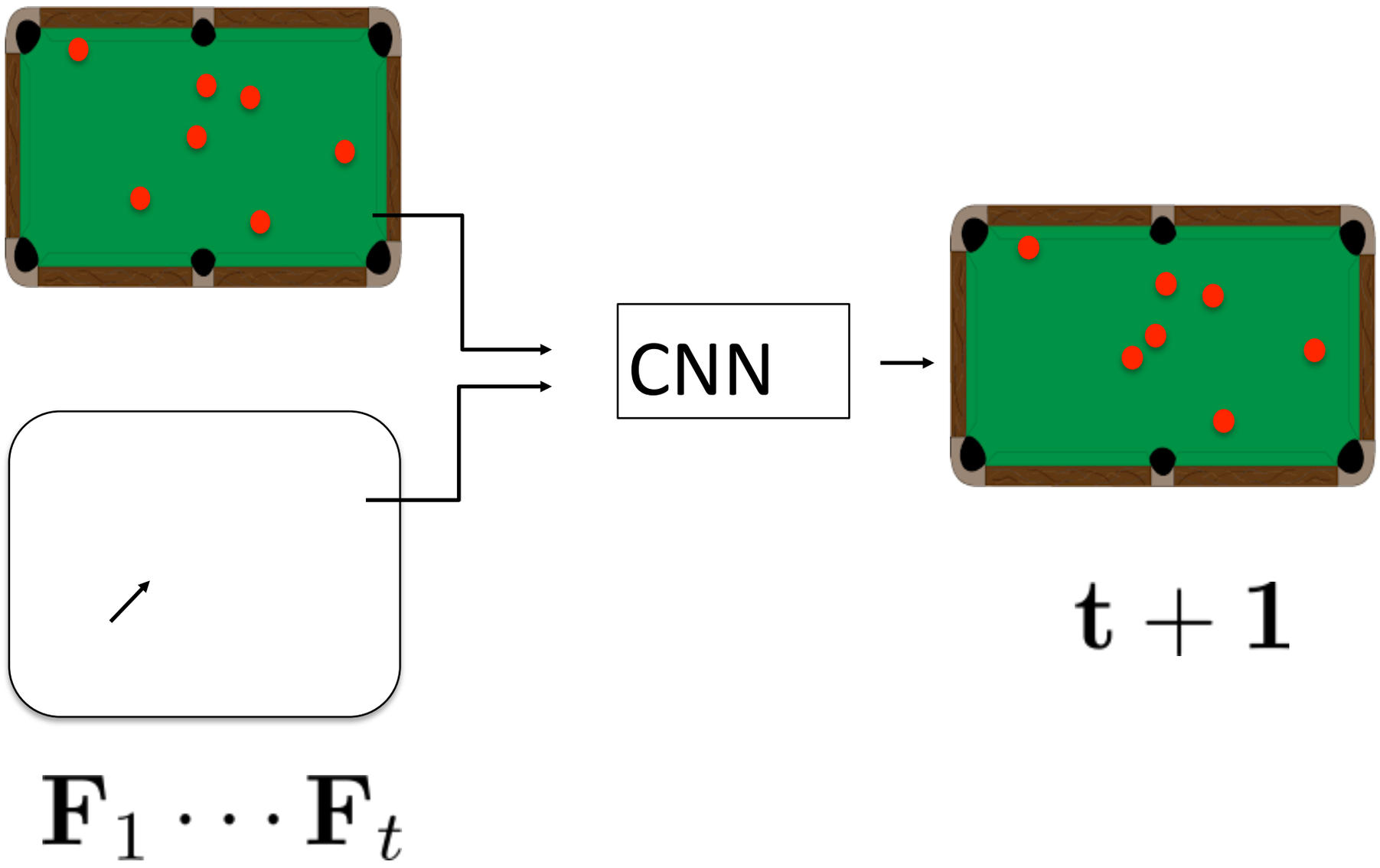






Predictive Visual Models of Physics for Playing Billiards,
K.F.*, Pulkit Agrawal*, Sergey Levine, Jitendra Malik, ICLR 2016

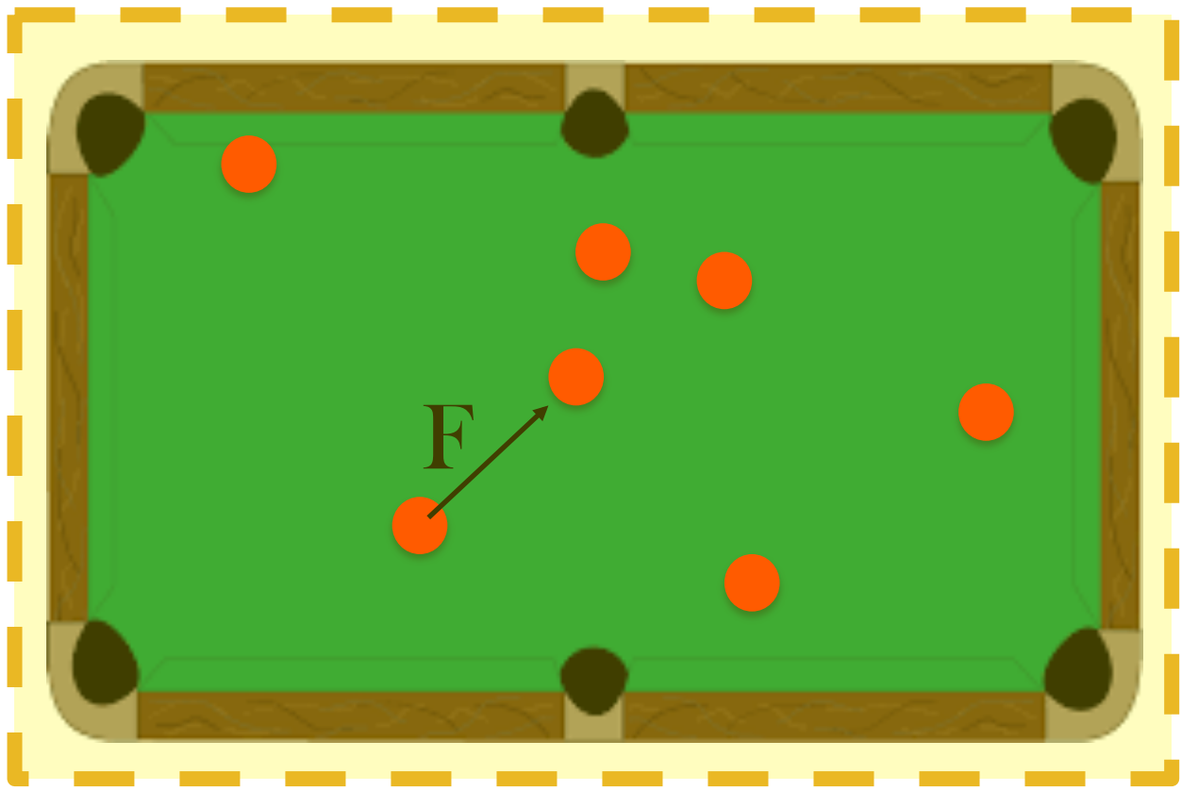
Learning Action-Conditioned Dynamics



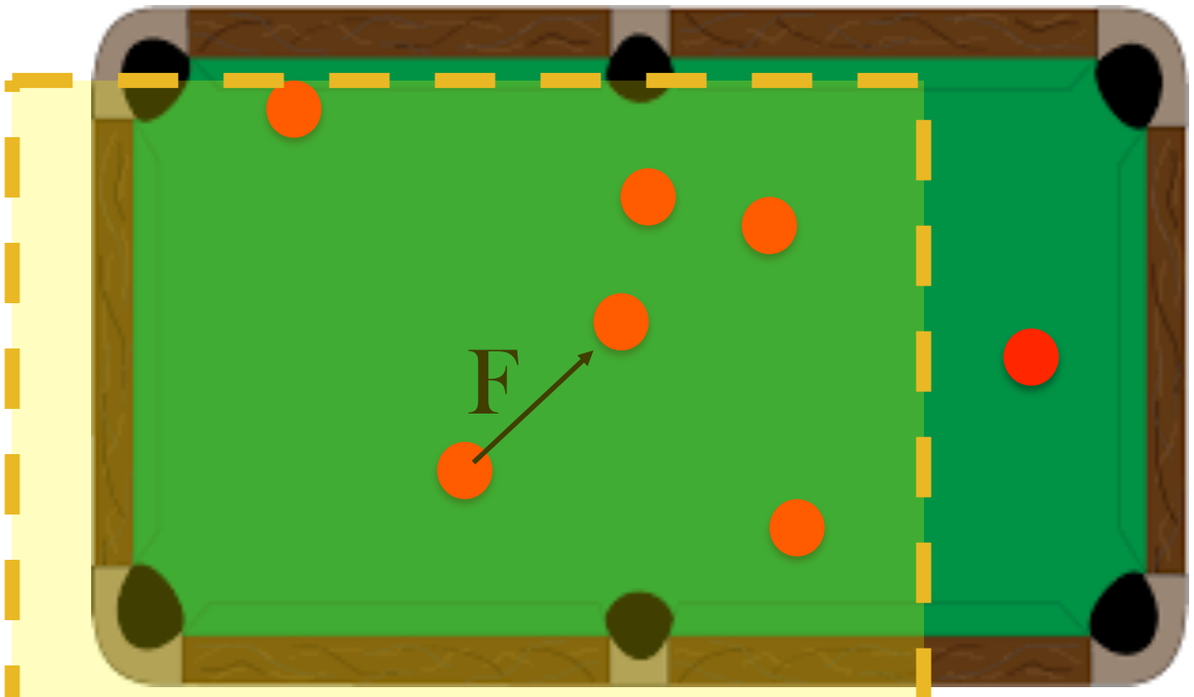
Problems:

- Forces are translation invariant!

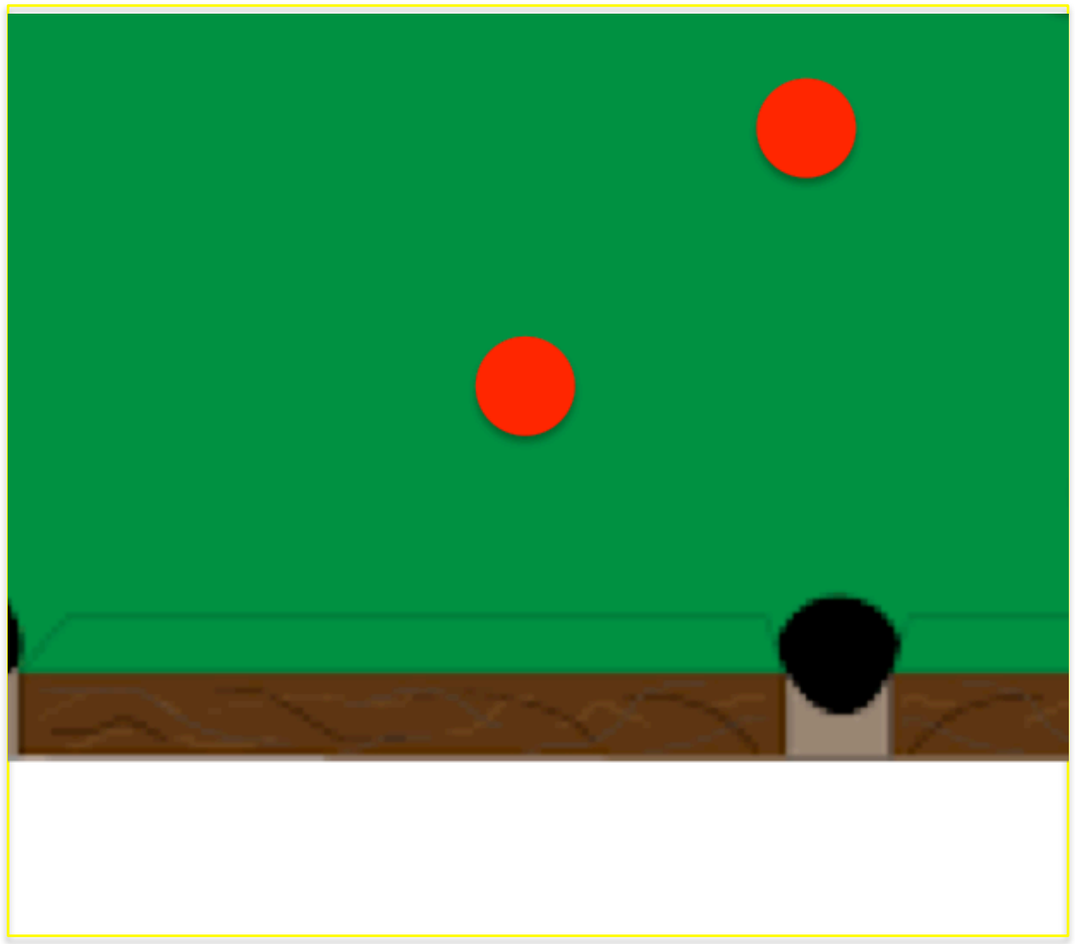
Object-centric prediction

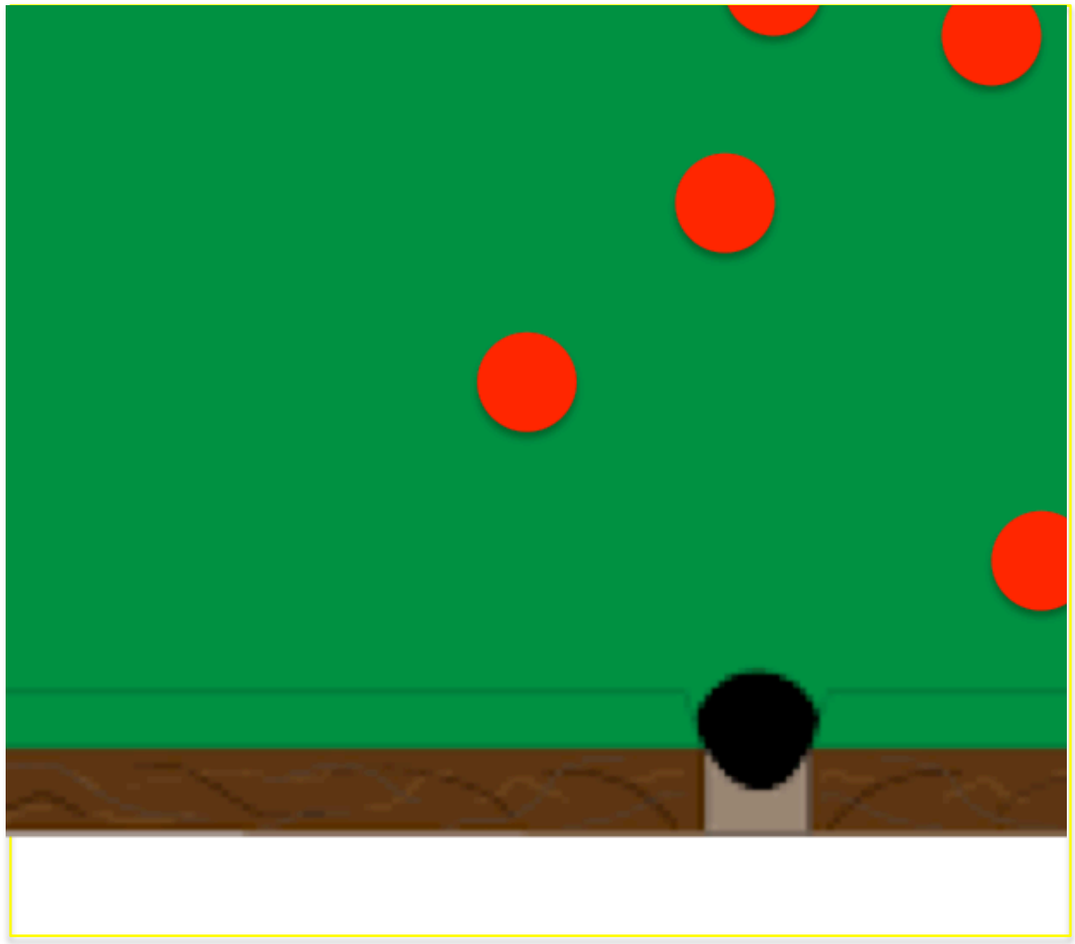


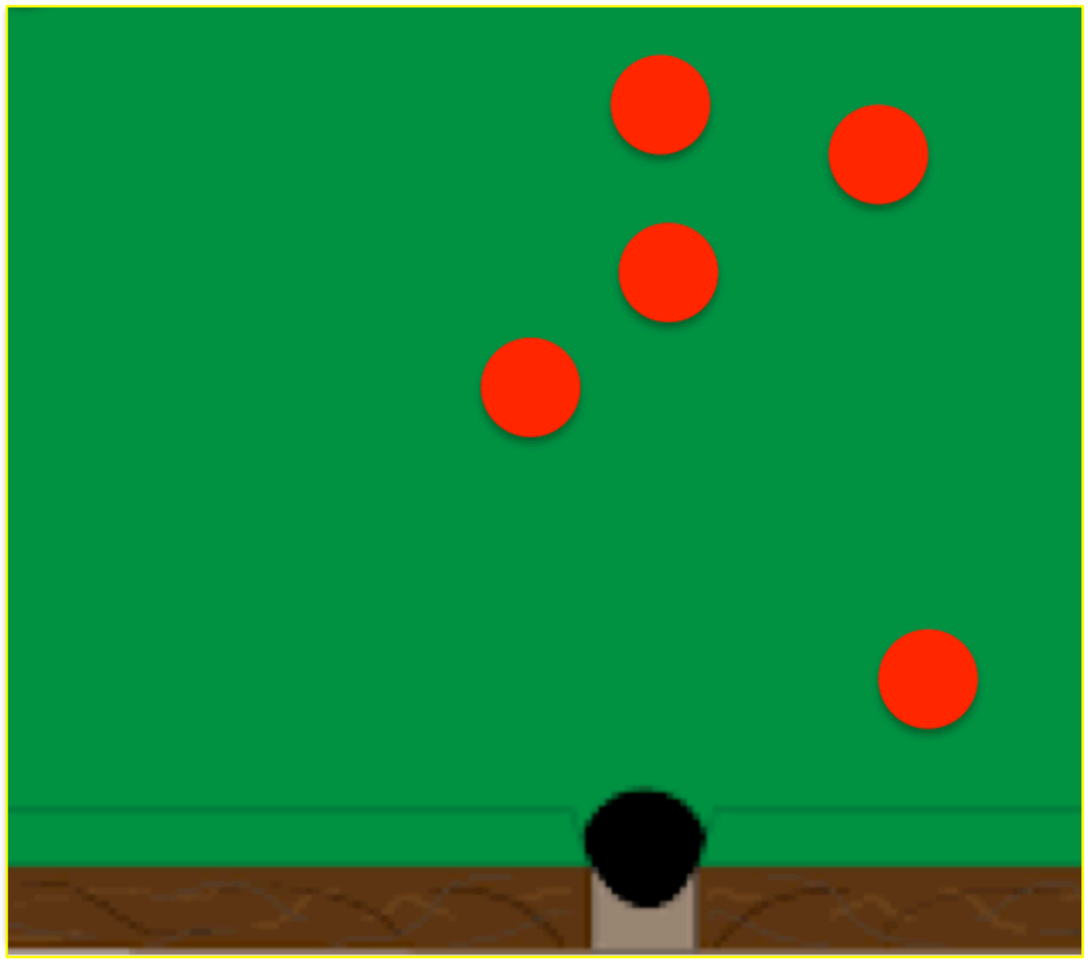
World-Centric Prediction

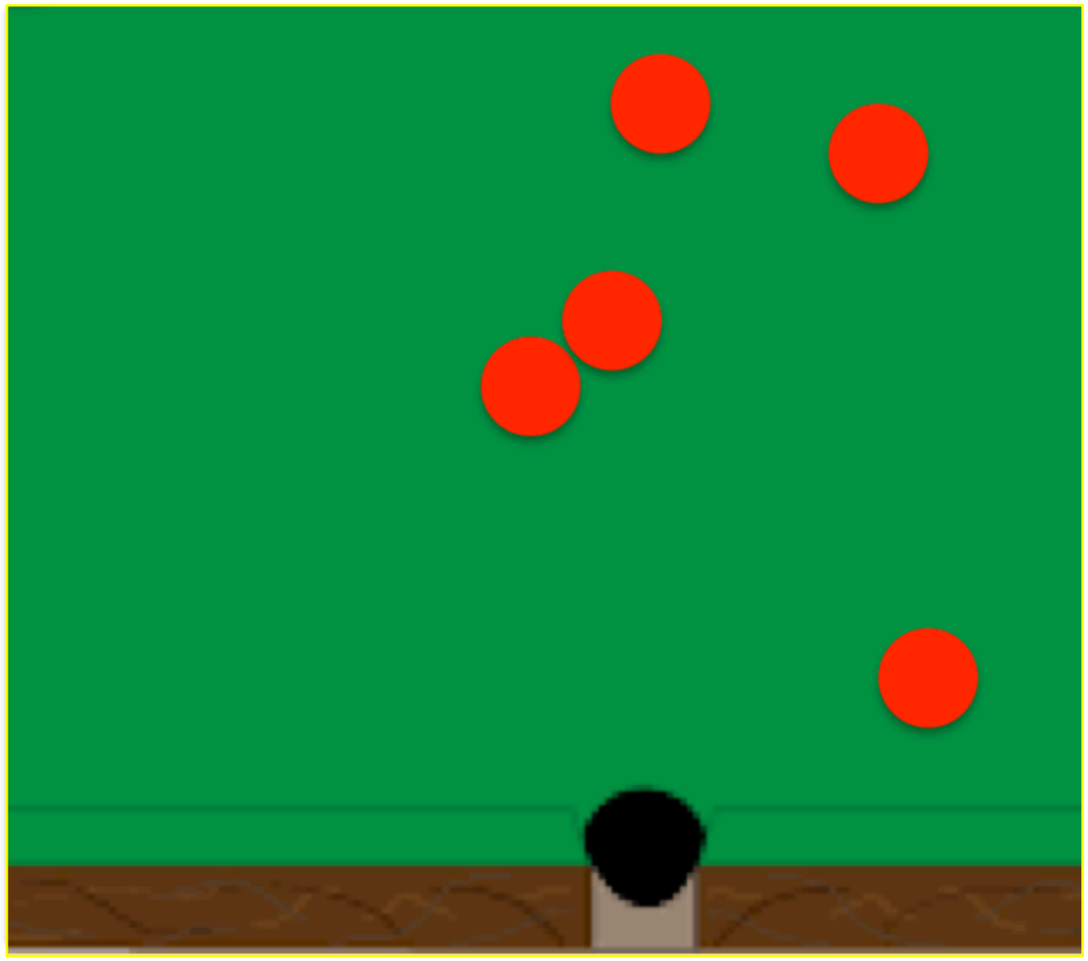


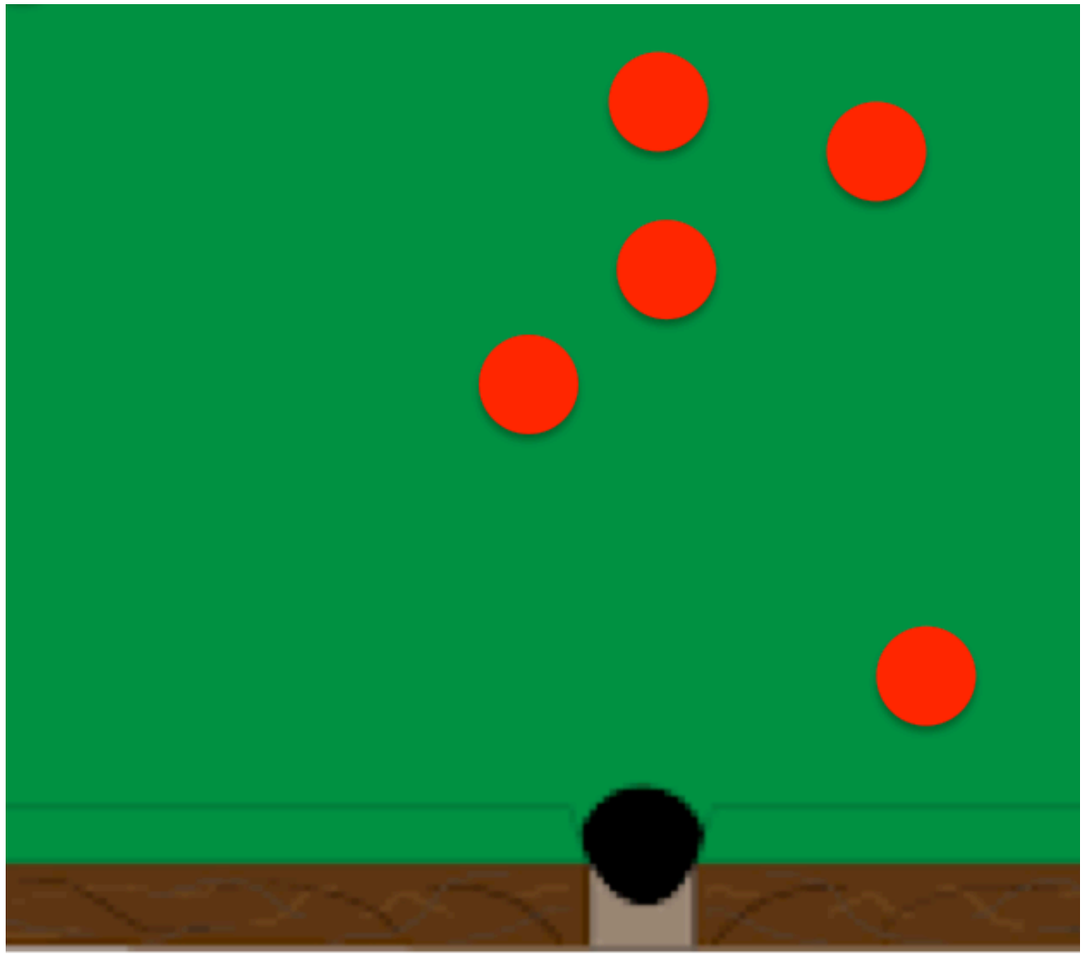
Object-Centric Prediction

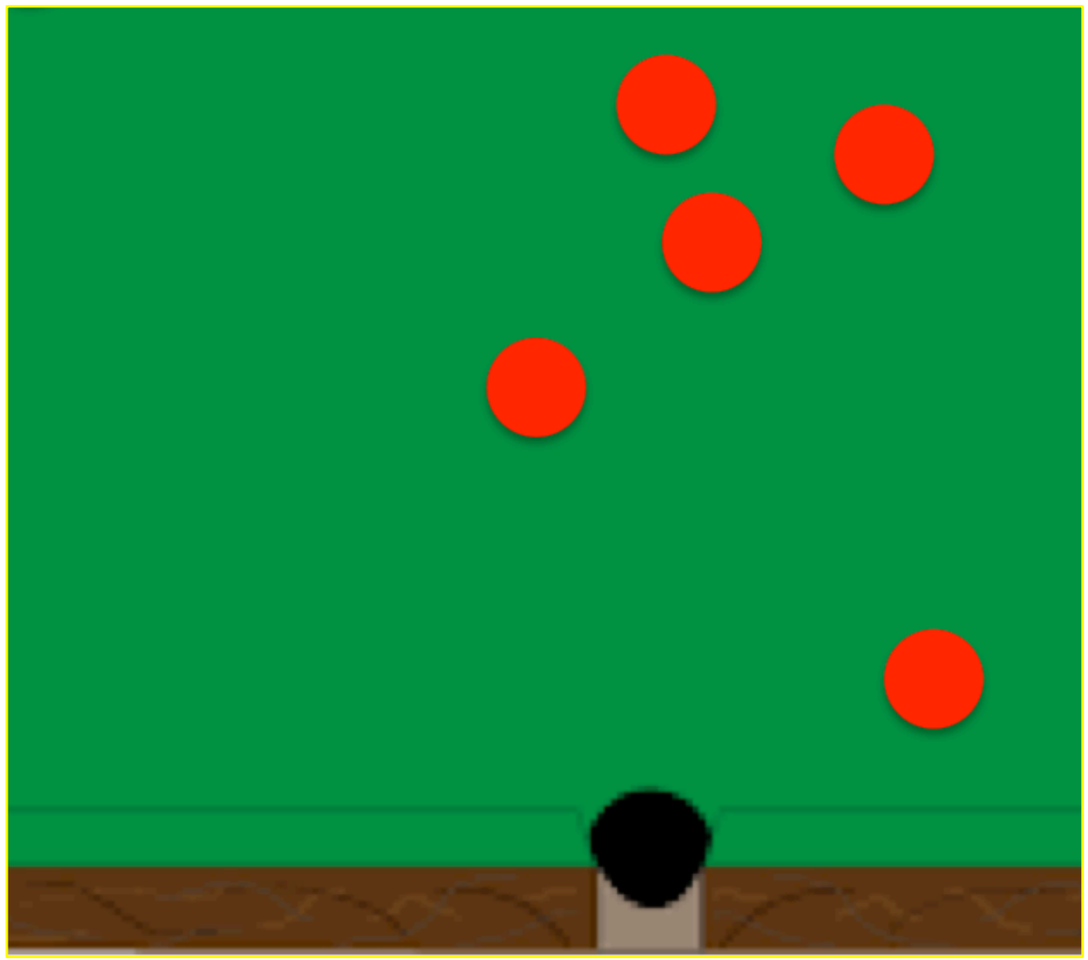


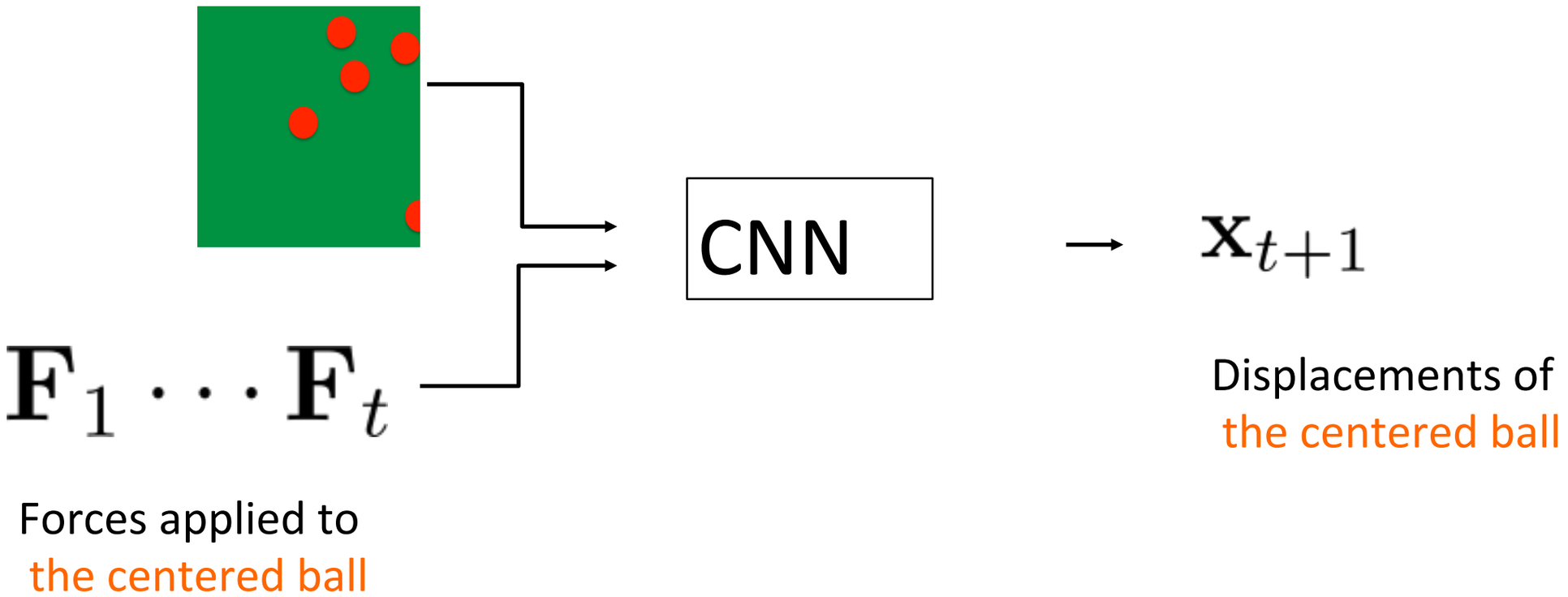






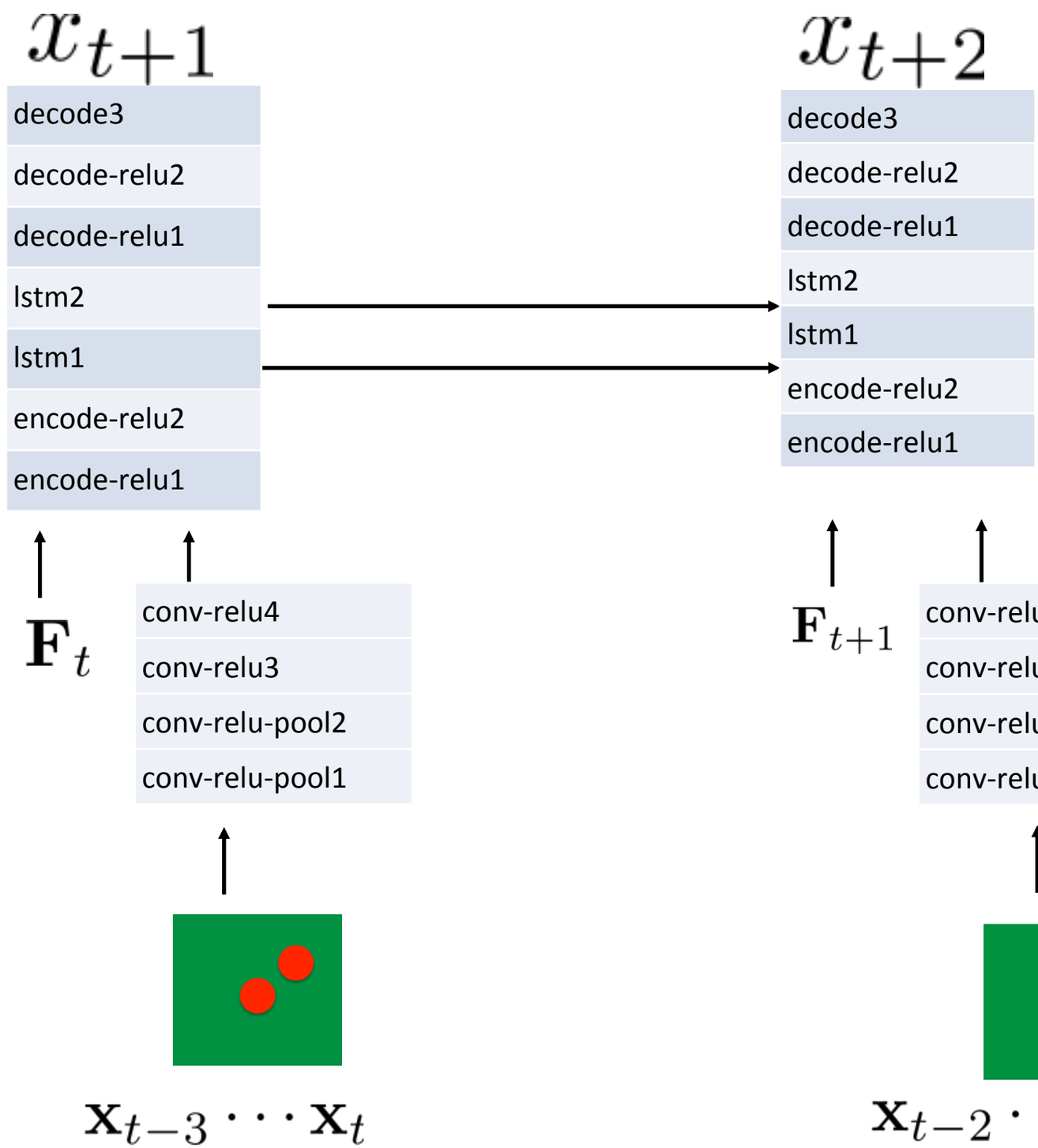




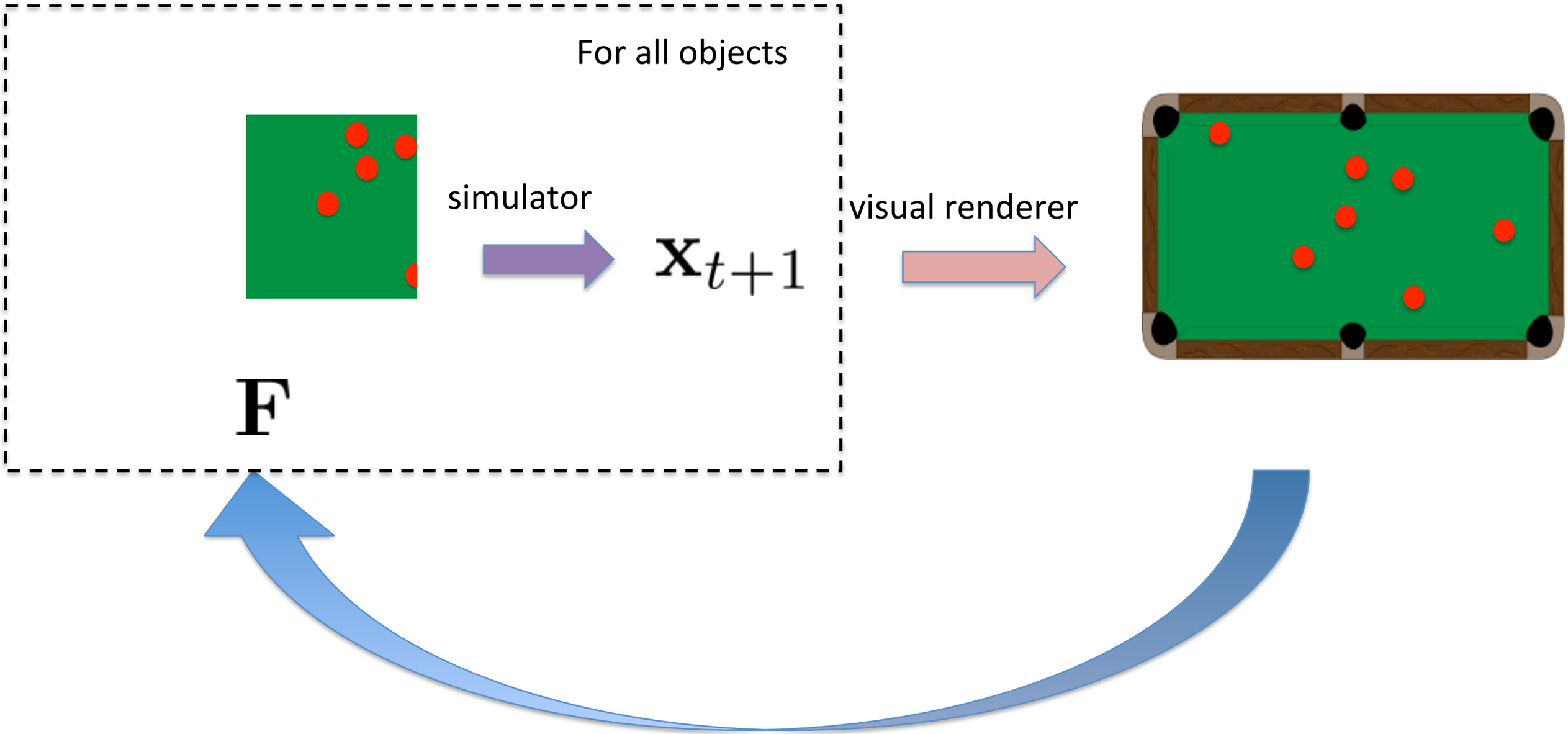


Learning the Laws of Physics for Playing Billiards,
Katerina Fragkiadaki*, Pulkit Agrawal*, Sergey Levine, Jitendra Malik

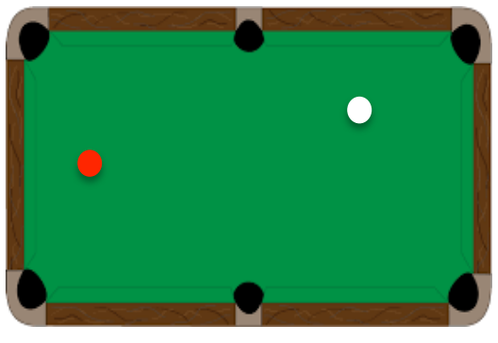
Network Architecture



Test Time: Visual Imaginations

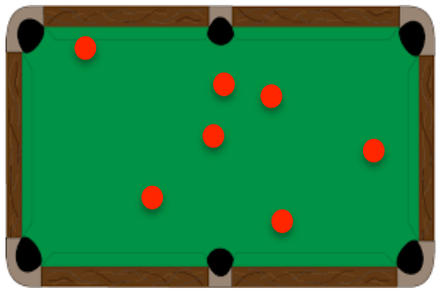


Policy Learning

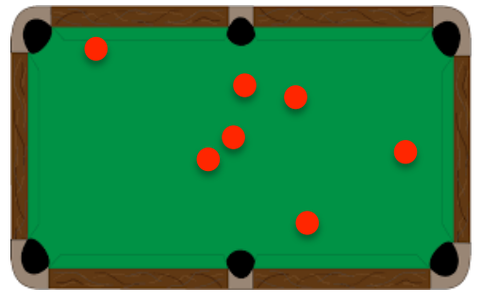


F

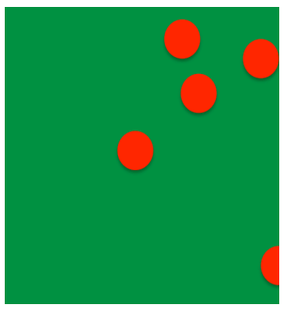
Learning Action Conditioned Dynamics



F



Handling uncertainty in prediction



$\mathbf{F}_1 \cdots \mathbf{F}_t$

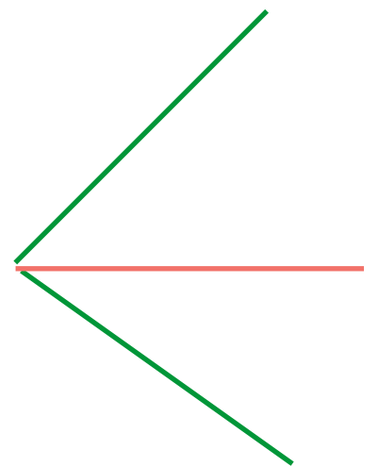
Forces applied to
the centered ball

CNN → $\mathbf{x}_{t+1}$

Displacements of
the centered ball

The model presented so far was deterministic: regression

It cannot handle uncertainty! Regression to the mean:



Solutions:

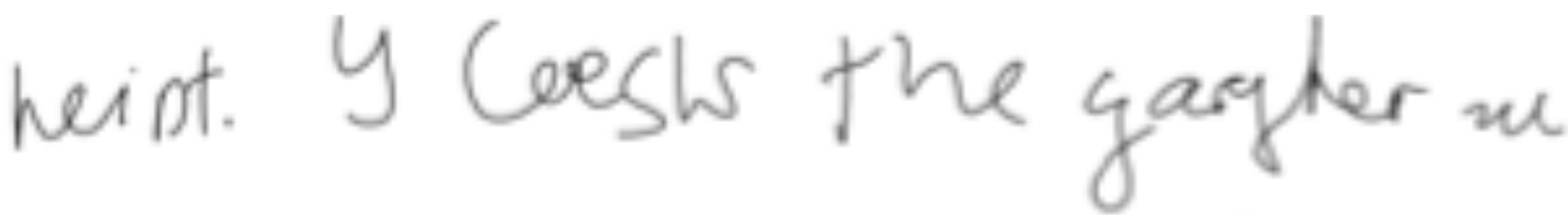
- Gaussian Mixture Model
- Stochastic networks

Text/Handwriting Generation using RNNs

Machine generated sentence:

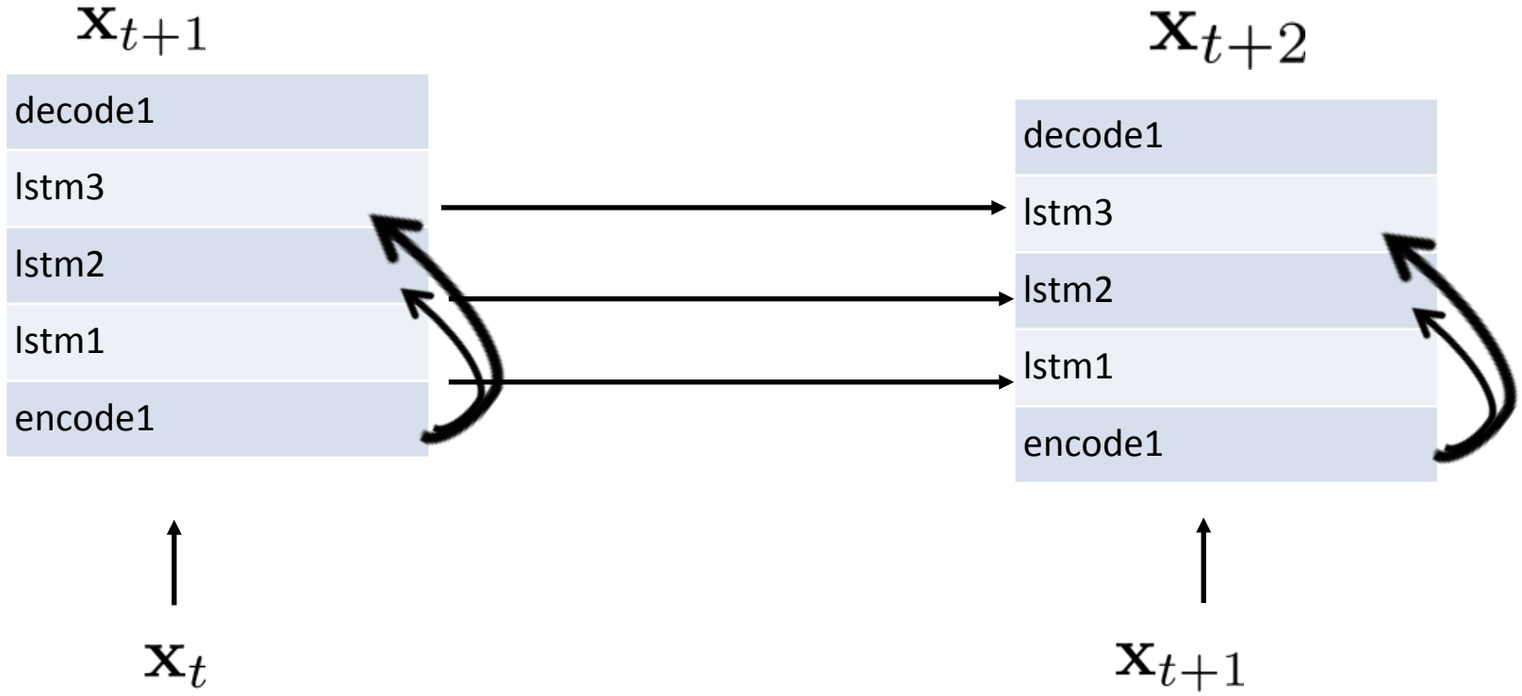
Besides these markets (notably a son of humor).

Machine generated handwriting:



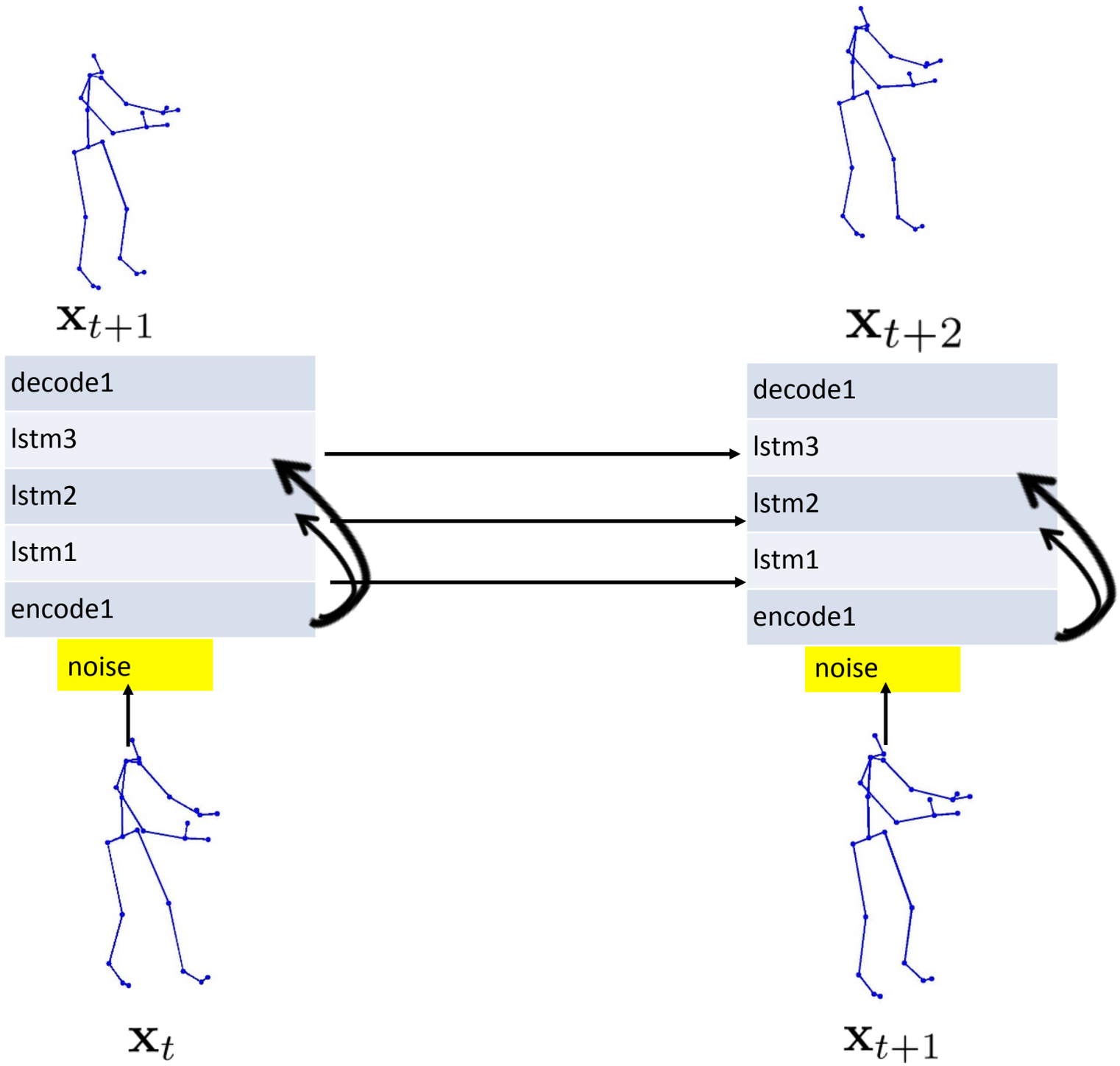
Graves et al., Sutskever et al.

Text/Handwriting Generation using RNNs

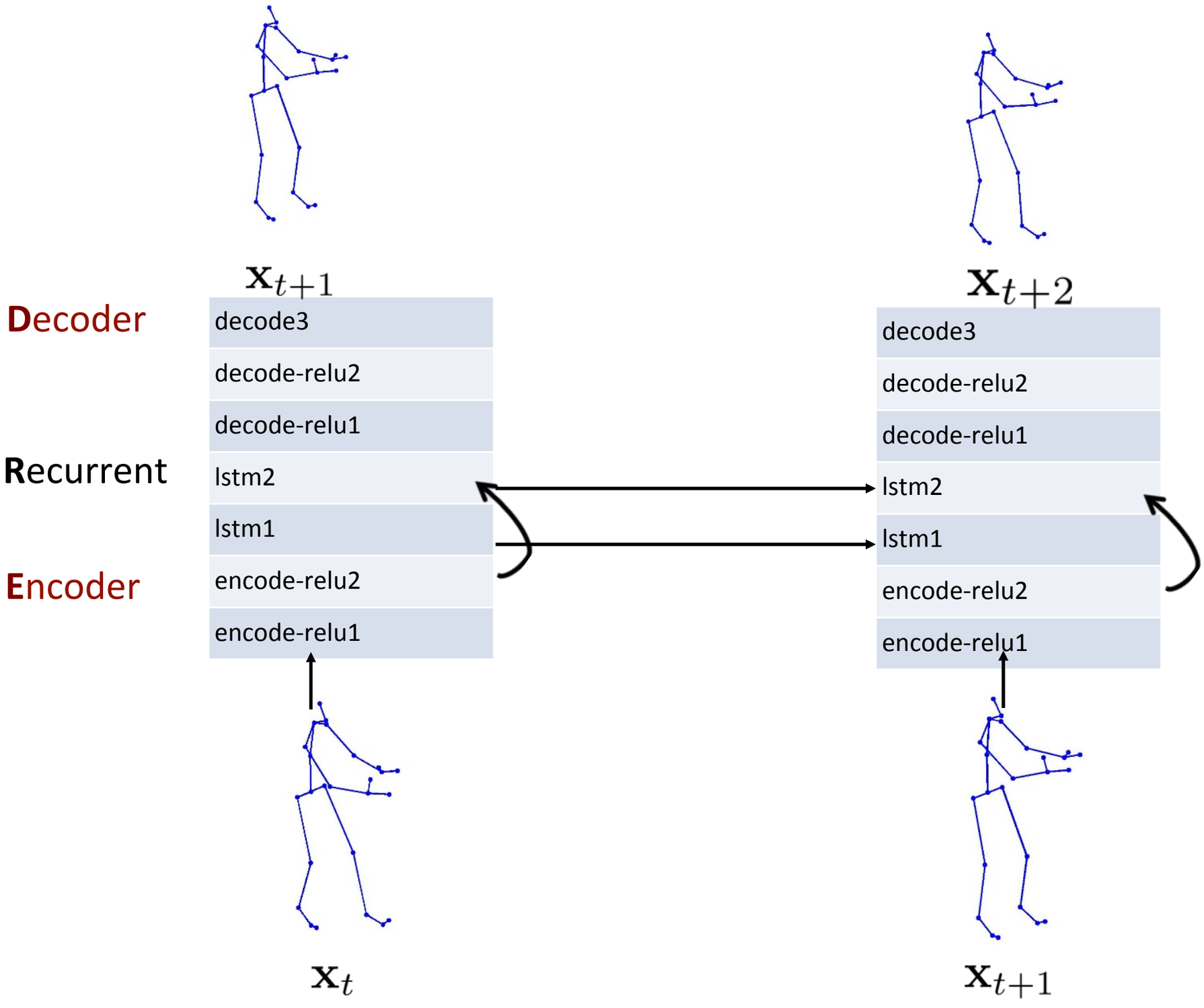


Graves et al.

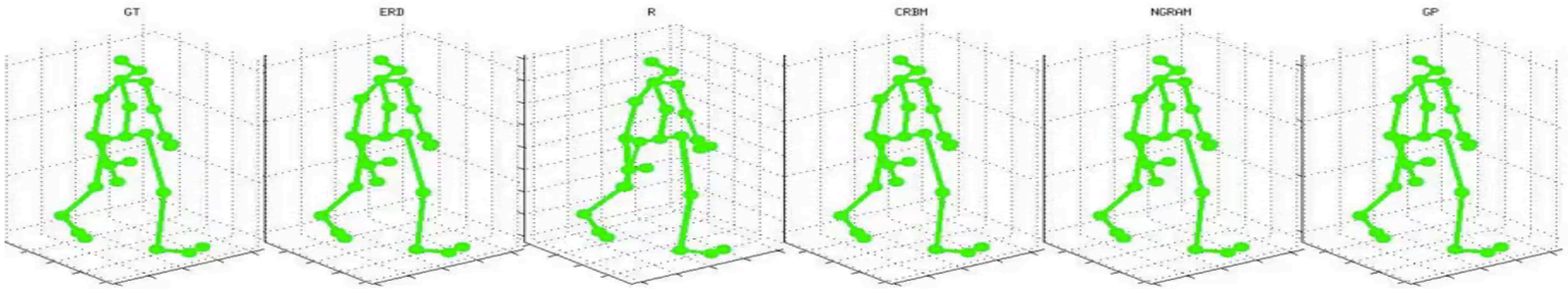
Learning Dynamics from Motion Capture



Graves et al.[47]


$\mathbf{x}_{t+1}$
$\mathbf{x}_{t+2}$
Decoder
decode3
decode-relu2
decode-relu1
Recurrent
lstm2
lstm1
Encoder
encode-relu2
encode-relu1
$\mathbf{x}_t$
$\mathbf{x}_{t+1}$

Ground-truth ERD LSTM-3LR CRBM NGRAM GP

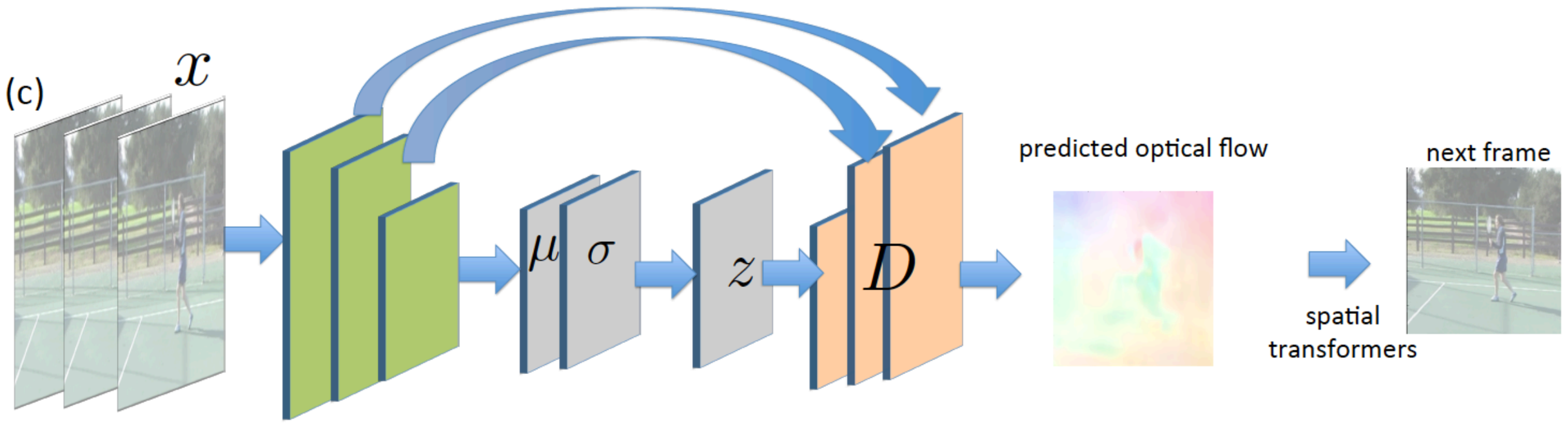


green: conditioning

blue: generation

Video prediction under multimodal future distributions

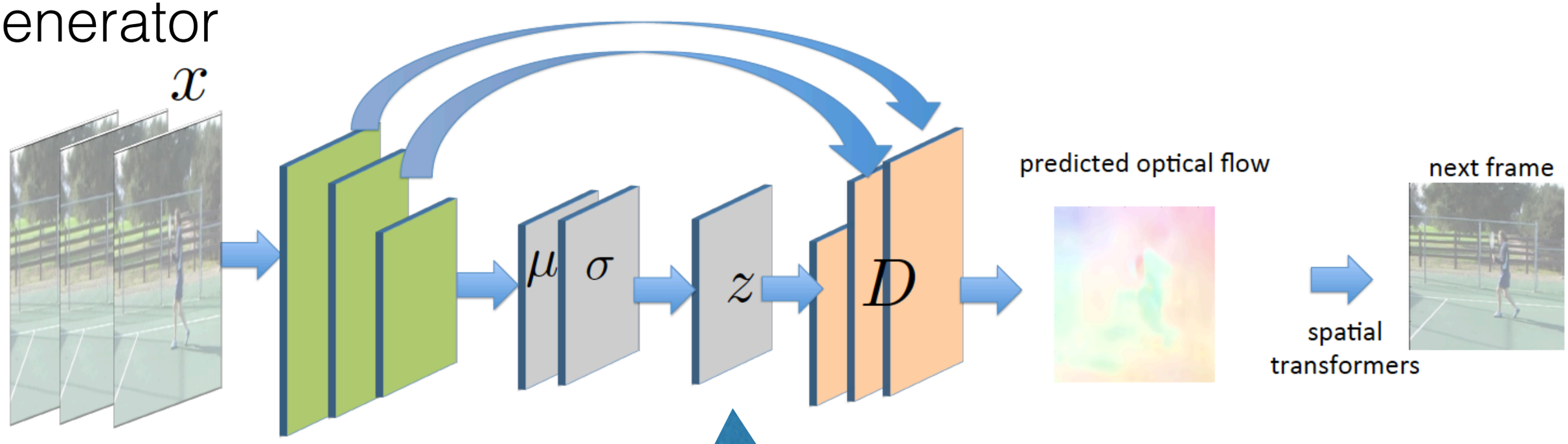
- Stochastic networks



Video prediction under multimodal future distributions

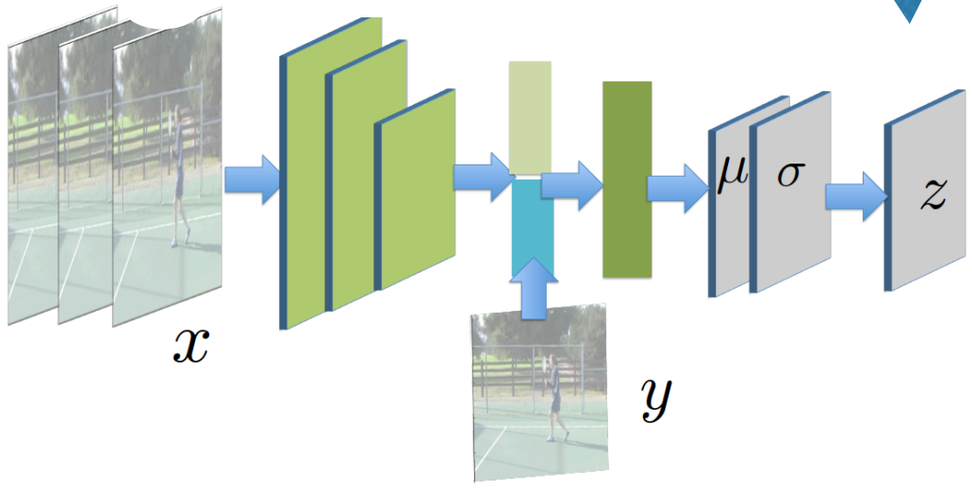
Train it with (conditional) variational autoencoder

Generator



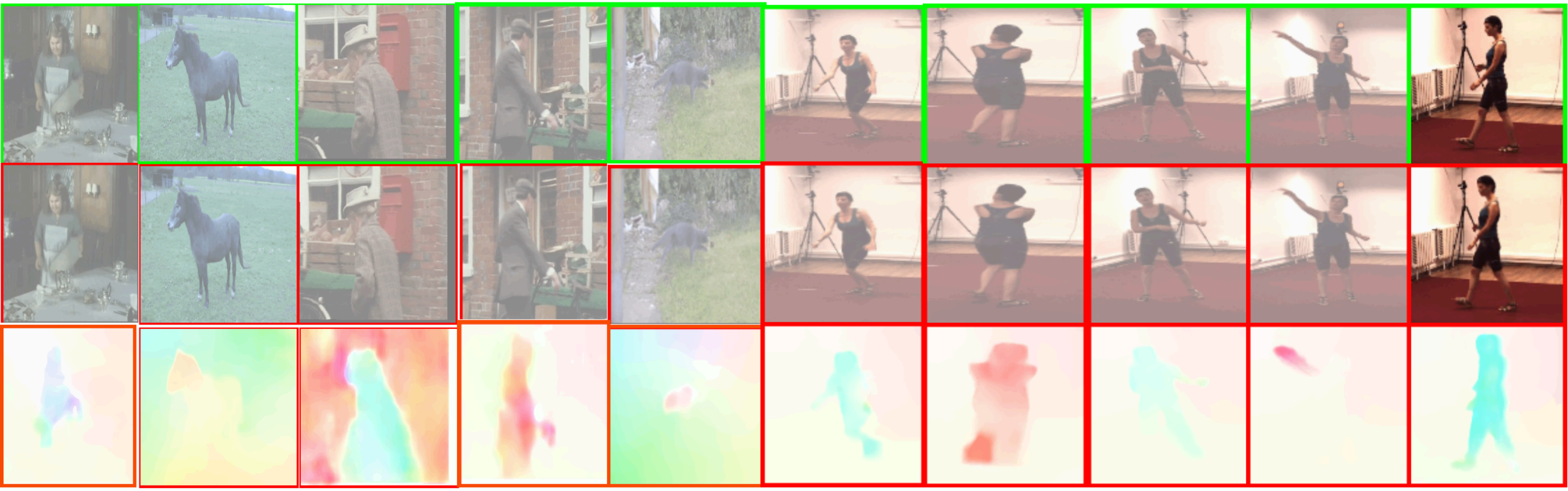
KL

Recognition network



During training the loss is a combination of KL divergence and reconstruction loss of the recognition network

Video prediction under multimodal future distributions



Video prediction under multimodal future distributions

green: input, red: sampled future motion field and corresponding frame completion

Video prediction under multimodal future distributions

green: input
red: sampled future completion

more future completions

green: input
red: sampled future completion

more future completions

Global models

- Despite the abundance of research papers on the forecasting problem, such approaches so far have not been successful in aiding model predictive control
- Naturally, not all part of the state space will have accurate dynamics, and such inaccuracies will be exploited buy the planner
- For MPC, local models currently dominate

Summary

- Optimal Control, trajectory optimization formulation
- Special but important case: Linear dynamics, quadratic costs (LQR)
- iterative-LQR / Differential Dynamic programming for Non-linear dynamical systems
- Examples of when it works and when it does not
- Learning Dynamics: Global Models

Flash Back

- Why we need to have this separate optimization over cost, and then separately planning/RL over this cost to find the policy? Because the dynamics where unknown..

- Let's assume they are known.

- Can we imitate the expert and backdrop through all the way till the rewards simply by imitating expert behavior (e.g., through supervised learning), in an end-to-end fashion?

Value Iteration Sub-Network

Each iteration of VI may be seen as passing the previous value function V_n and reward function R through a convolution layer and max-pooling layer.

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$$
$$V_{n+1}(s) = \max_a Q_n(s,a)$$

Value Iteration Networks, Tamar et al. 2016

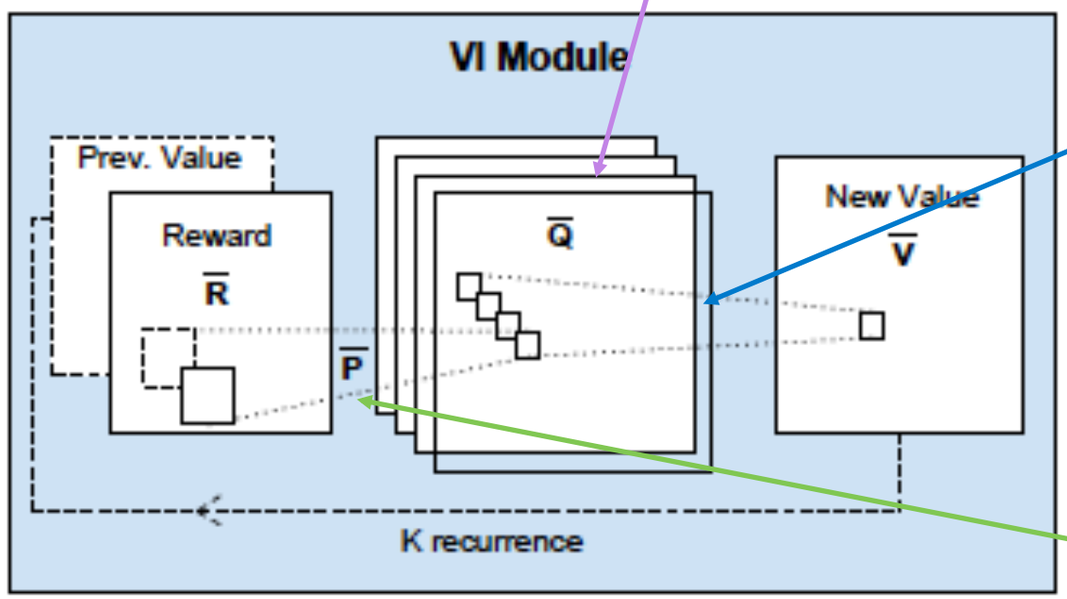
Value Iteration Sub-Network

Each iteration of VI may be seen as passing the previous value function V_n and reward function R through a convolution layer and max-pooling layer.

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$$

$$V_{n+1}(s) = \max_a Q_n(s,a)$$

Each channel in the convolution layer corresponds to the Q -function for a specific action



Max-pooling

convolution kernel weights correspond to the discounted transition probabilities

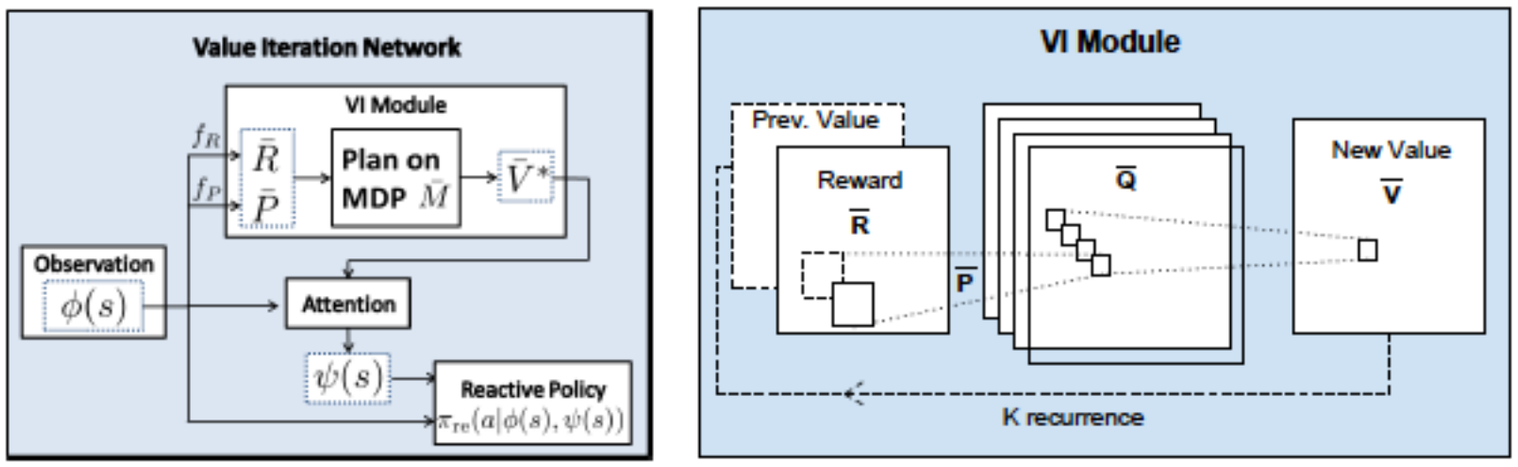
By recurrently applying a convolution layer K times, K iterations of VI are effectively performed.

Value Iteration Networks, Tamar et al. 2016

Value Iteration Network

- Notice that the optimal action at each state depends only on the value function of its immediate neighbors->locality->attention

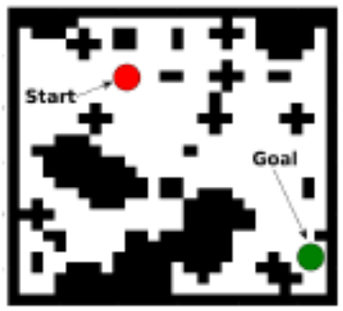
$$\bar{\pi}^*(\bar{s}) = \arg\max_{\bar{a}} \bar{R}(\bar{s},\bar{a}) + \gamma \sum_{\bar{s}'} \bar{P}(\bar{s}'|\bar{s},\bar{a})\bar{V}^*(\bar{s}').$$

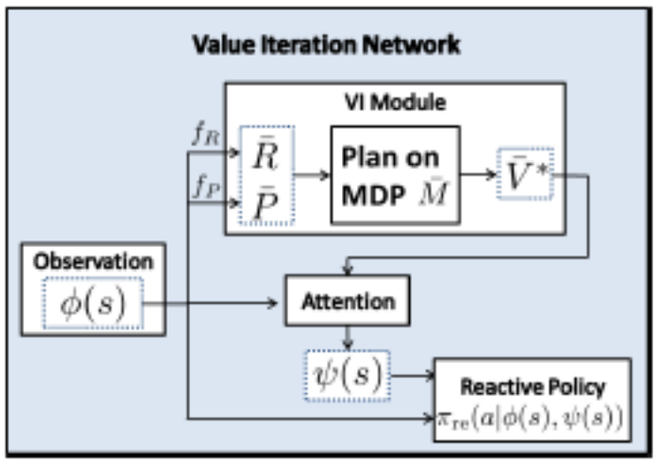


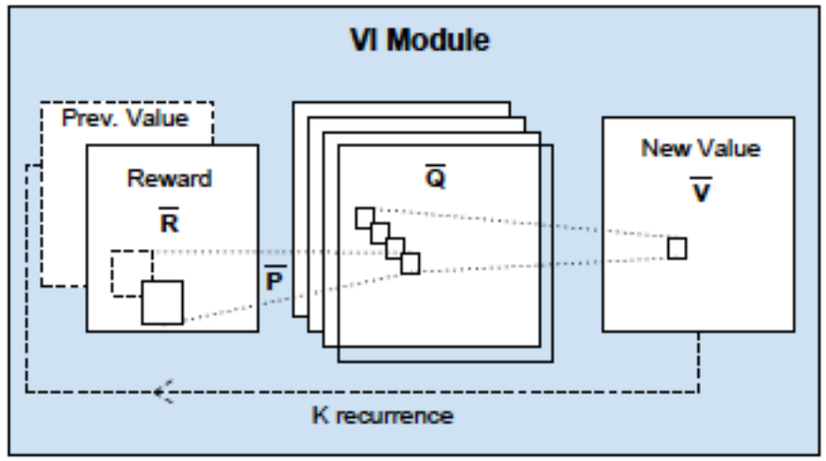
- To estimate the optimal action in each state, I only need to use the value functions in the vicinity of the state, then train and CNN with a standard architecture

Value Iteration Networks, Tamar et al. 2016

Value Iteration Network







Domain	VIN			CNN			FCN		
	Prediction loss	Success rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.
8×8	0.004	**99.6%**	0.001	0.02	97.9%	0.006	0.01	97.3%	0.004
16×16	0.05	**99.3%**	0.089	0.10	87.6%	0.06	0.07	88.3%	0.05
28×28	0.11	**97%**	0.086	0.13	74.2%	0.078	0.09	76.6%	0.08

Value Iteration Networks, Tamar et al. 2016