Transfer Learning for Reinforcement Learning.

Yohannis K Telila

y.telila@studenti.unipi.it

Sept. 16

Why Transfer Learning for DRL?

- Performance issue.
- Large amount of time and interaction with env’t it take to learn suitable policy.
- Most Deep RL algorithm performance are measured primarily in a simulation.
- Solving the problem from scratch is not a natural approach
- Cost of failure

Proposed papers

- Sim-to-Real Robot Learning from Pixels with Progressive Nets.
  Andrei et al., 2018

- Domain Adaptation For Reinforcement Learning On The Atari.
  Thomas et al., 2018

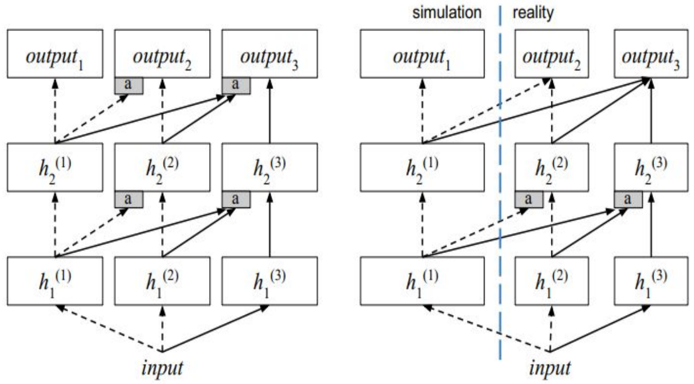
- Transfer Learning for Reinforcement Learning on a Physical Robot.
  Samuel et al., 2010.

Sim-to-Real Robot Learning from Pixels with Progressive Nets.

Andrei et al., 2018

Introduction

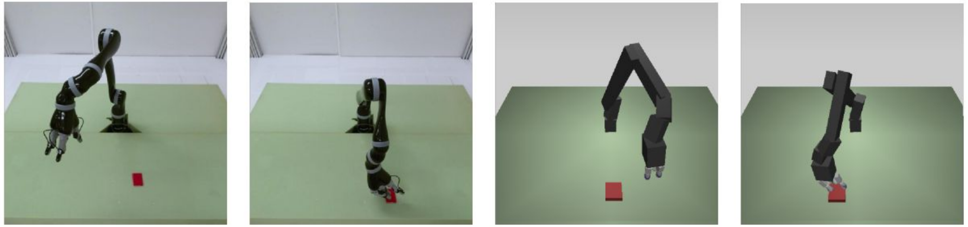
- This approach relies on the ***progressive nets*** architecture
- A progressive network starts with a single column.
- Columns in progressive networks are free to reuse, modify or ignore previously learned features via the ***lateral connections.***
- Each column is trained to solve a particular Markov Decision Process (MDP)
- Why?
  - ➔ Feature transfer without destruction from fine tune.
  - ➔ Columns may be heterogeneous. Important for solving different task.

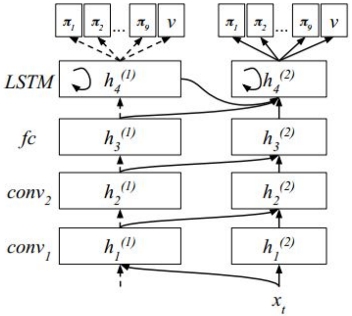


$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right),$$

Experiment

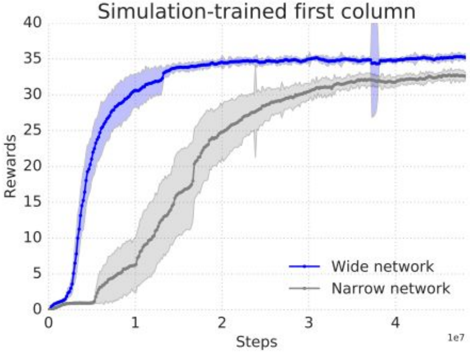
- The task of reaching to a visual target with Jaco arm (MuJoCo physics simulator).
- Reward of +1 if its palm is within 10cm of the target. And 50 step of episodes.
- Input : 3x64x64 pixels; output: 28 (9 discrete joint policies plus one value function).

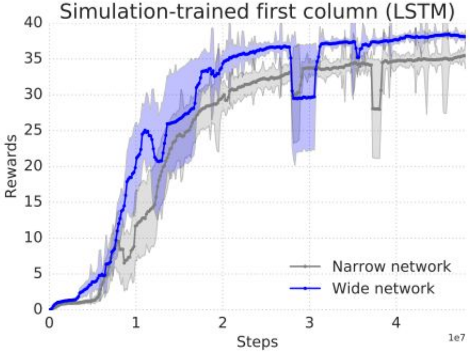


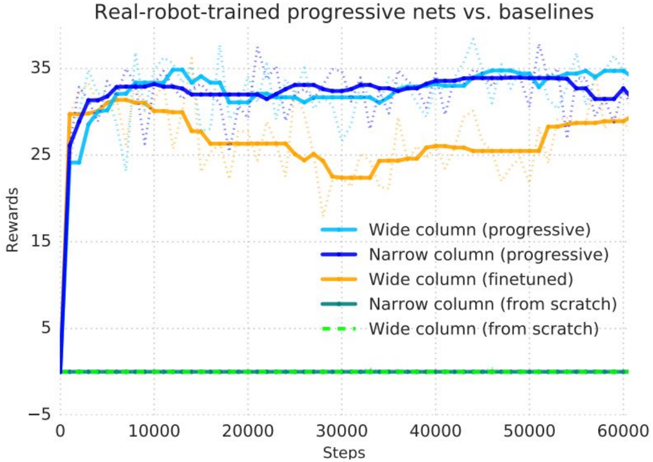


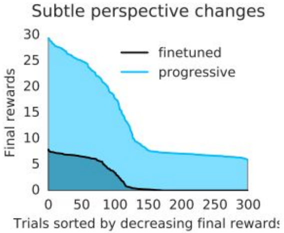
	feedforward		recurrent	
	wide	narrow	wide	narrow
fc (output)	28	28	28	28
LSTM	-	-	128	16
fc	512	32	128	16
conv 2	32	8	32	8
conv 1	16	8	16	8
params	621K	39K	299K	37K

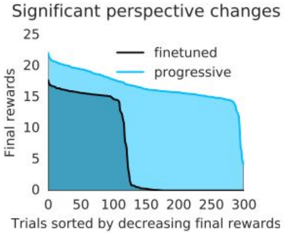
Result

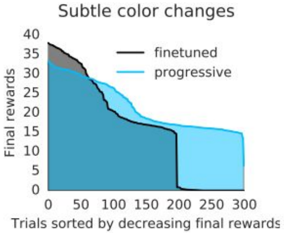

Simulation-trained first column
Rewards
Steps
1e7
Wide network
Narrow network

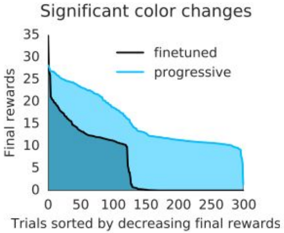

Simulation-trained first column (LSTM)
Rewards
Steps
1e7
Narrow network
Wide network


Real-robot-trained progressive nets vs. baselines
Rewards
Steps
Wide column (progressive)
Narrow column (progressive)
Wide column (finetuned)
Narrow column (from scratch)
Wide column (from scratch)


Subtle perspective changes
Final rewards
finetuned
progressive
Trials sorted by decreasing final reward


Significant perspective changes
Final rewards
finetuned
progressive
Trials sorted by decreasing final rewards


Subtle color changes
Final rewards
finetuned
progressive
Trials sorted by decreasing final rewards


Significant color changes
Final rewards
finetuned
progressive
Trials sorted by decreasing final rewards

Domain Adaptation For Reinforcement Learning

Thomas et al., 2018

Introduction.

- This approach relies on the ***Domain Adaptation technique.***
- One way to view learning within DRL is to consider the hidden layers as learning a state representation on top of which a policy can be learned.
- **Domain Adaptation** methods in this context are about learning to construct that state space for corresponding inputs in the target domain
- The proposed approach uses **Adversarial Auto-encoder(AAE)** to impose this regularisation in an unsupervised manner. (Adversarial Domain Adaptation)
- If the tasks are similar we expect similar embeddings. [This is in pre training step]

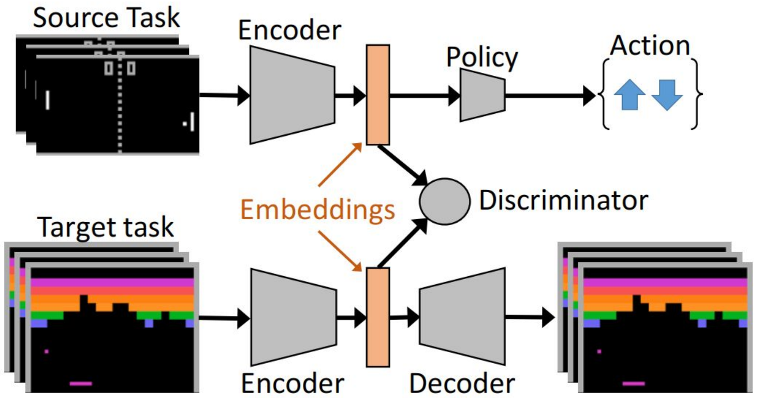


Figure 2: Agent's architecture for domain adaptation

Experiment.

- Experiment carried out on three games; Pong, Breakout and Tennis to verify the effectiveness of the approach.[somehow related]

- Experiment procedure.

    1. Train agent to play Pong in the standard RL fashion;[A2C]

    2. Run the agent through a few games, capturing the final hidden layer output. [100k samples collected]

    3. Using these two data sets train the AAE.

    4. Take the weights of the Generator of our AAE, to initialise the weights of a new Policy network for the target task.

    5. Train the policy on the target task.

Fig. 1: Pong (left) to Breakout (middle) and Tennis (right)

Result.

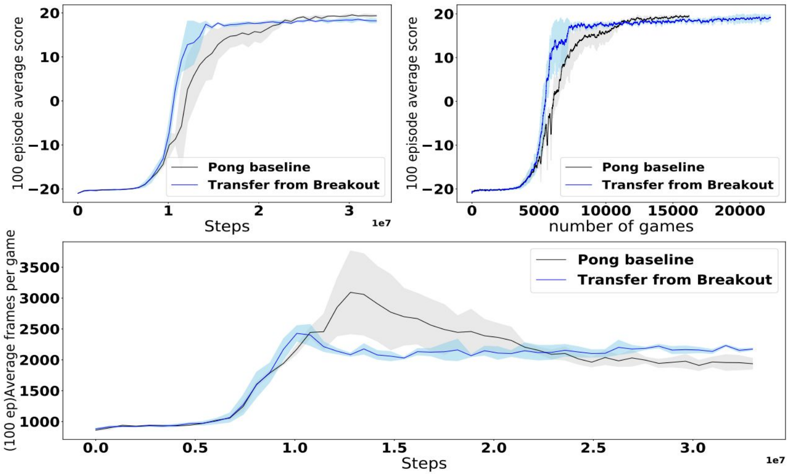


Fig: **TopLeft:** Learning Curve for Pong, **TopRight:** Plots the Average score against number of games completed. **Bottom:** 100 episode average number of frames in a game.

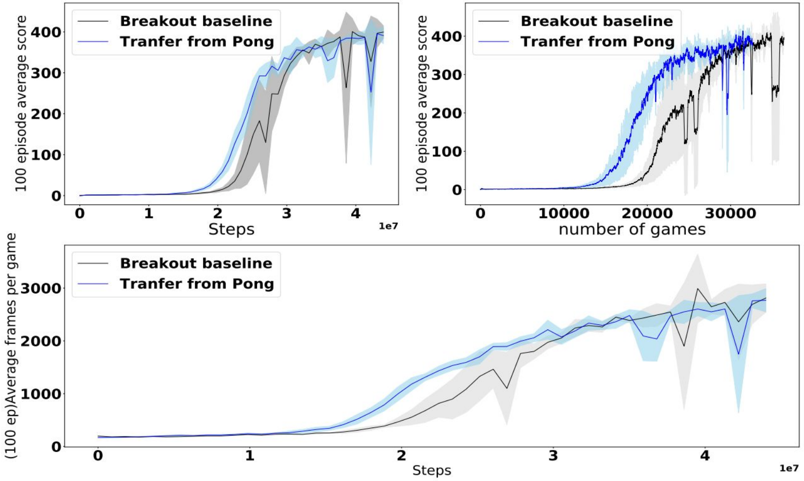


Fig: **TopLeft:** Learning Curve for Breakout, **TopRight:** Plots the Average score against number of games completed. **Bottom:** 100 episode average number of frames in a game

Transfer Learning for Reinforcement Learning on a Physical Robot.

Samuel et al., 2010.

Introduction.

- This approach uses “**Q-value reuse**” with **Sarsa(λ)** to transfer information from source task to a new target task.
- Involves reusing value function $\mathbf{Q_{source}}$ learned from Source task, as a starting point for a new problem value Function $\mathbf{Q_{target}}$
- Problem : The state and action space might not coincide.
- Soln: The agent has been given the mapping

$$\chi_X(s_{\text{target}}) = s_{\text{source}}$$
$$\chi_A(a_{\text{target}}) = a_{\text{source}}.$$

  - The agent’s new value function is given by

$$Q(s,a) = Q_{\text{source}}(\chi_X(s), \chi_A(a)) + Q_{\text{target}}(s,a)$$

- If there is no corresponding action in the source task, they are Initialized by average action values across all states.(it’s a choice)

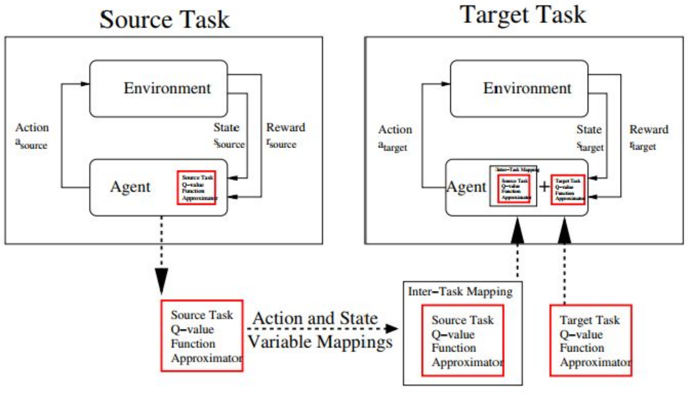
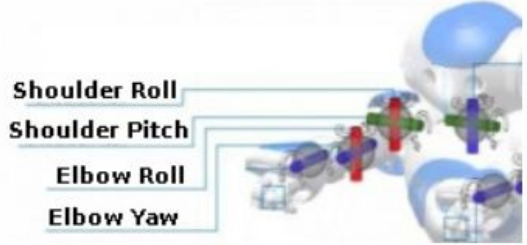
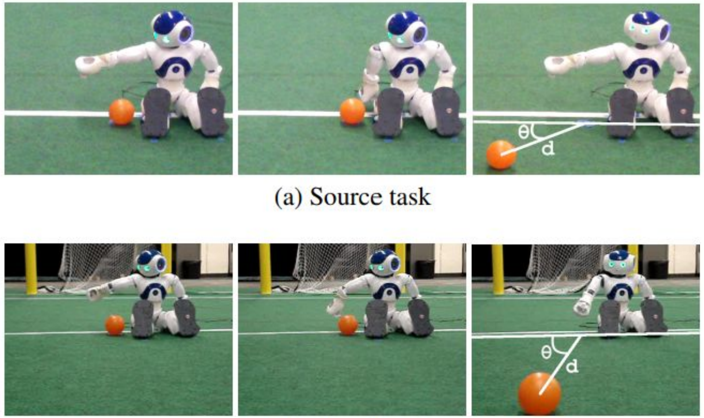


Figure 1: Q-Value Reuse

Experiment.

- The robot’s task was to hit an orange ball as far as possible at a 45◦ angle with its right hand.
- The reward signal is given by $r = d * \cos(\theta)$, d-distance
- Source Task:
  - The robot will only use two shoulder joint and 4 observations[position and velocity of each joint].
- Target Task:
  - The robot can use all 4 joints to help him kick the ball and eight observations [the position and velocity for each joint].
- The robot will learn faster in the source task because of simpler problem.
- Both the source and target tasks were learned on the physical robot{simulation also tested}.

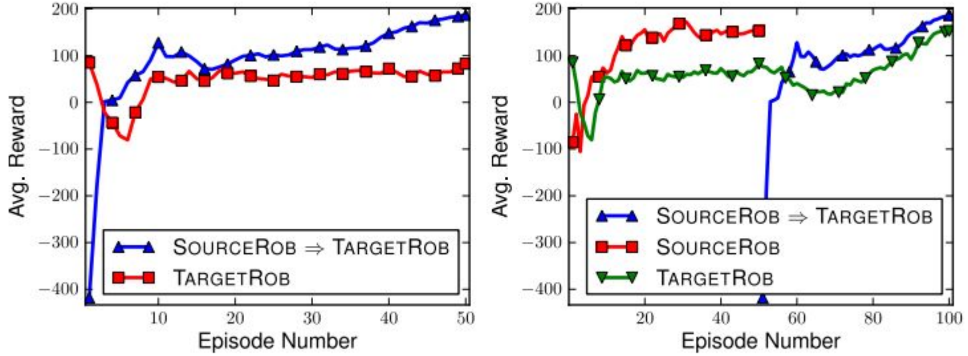




(a) Source task

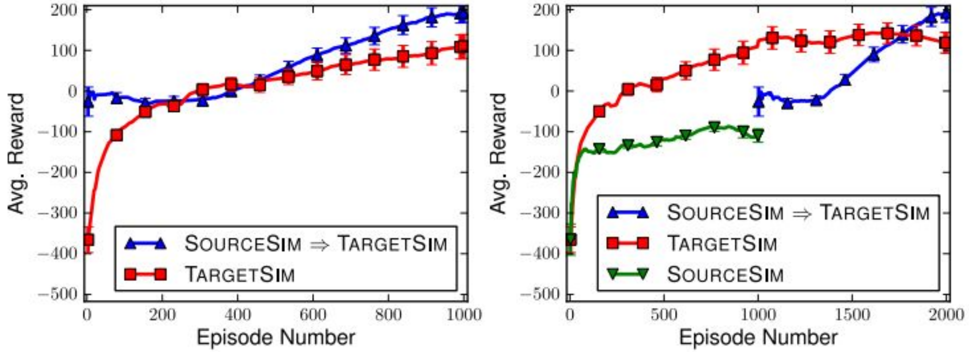
(b) Target task

Result.



(a) Weak Transfer

(b) Strong Transfer



(a) Weak Transfer

(b) Strong Transfer

Summary.

Transfer learning vs Domain adaptation

- Multiple tasks may be learned sequentially, without needing to specify source and target tasks.

MORE RELATED PAPERS:

- Latent Structure Matching for knowledge Transfer in Reinforcement Learning. Yi et al., 2020

- Efficient Deep Reinforcement Learning via Adaptive Policy Transfer. Tianpei et al., 2020

END.

THANK YOU !