Meta-Reinforcement Learning for Adaptive Autonomous Driving

Yesmina Jaafra, Jean Luc Laurent, Aline Deruyver, Mohamed S. Naceur

Motivation

« Concepts in the real world are not eternally fixed entities or structures, but can have a different appearance or definition or meaning in different contexts ».

- Several tasks require context-aware adaptation given their **high-dimensional** and **non-stationary** environments:

"Autonomous driving task"

- It is still an « unsolved problem » more than one decade after the promising 2007 DARPA Urban Challenge:
  - **Uncertainty** of human behavior and unexpected players trajectories,
  - **Complexity** of scene perception, changing weather, lighting conditions and traffic rules,
  - A **wide space** of elaborated actions combining vehicle commands and reconciling divergent objectives.

Important variability and conflicting dynamics!

- Reinforcement learning (RL) is an efficient **goal-oriented** optimization **beating human performance** in many domains as **games** and **advanced robotic** manipulations.

What about RL performance in non-stationary real-world tasks?

Literature results:

- RL methods are **context-specific** approaches and **fail to generalize** in complex and non-stationary environments.
- Finding compliant with "NFL theorems" demonstrating that the **universal learner does not exist** and each algorithm performs well only on a set of tasks delimiting its area of expertise.

Research Questions

1. How to build **generalizable** policies in **non-stationary** environments where the agent should draw prediction rules effective in different contexts?
2. How to perform a **continuous control** of generated policies in **high-dimensional** tasks?

Meta-learning with Neural Network controller

- Extend RL application to the challenging task of urban autonomous driving.

→ Build an approach of meta-learning compatible with RL setting.

- Given a distribution of Markov decision processes T_i, the meta-objective consists in maximizing expected rewards across tasks.

$$\max_{\theta} \mathbb{E}_{T_i \sim p(T)} R_{T_i}(\theta_i') \; where \; \theta_i' = \theta + \alpha \nabla_\theta R_{T_i}(\theta)$$

- **Inner Loop : "**Task-specific learning"
  Compute the policy parameters θ_i' for each sampled task T_i using gradient descent.

$$\theta_i' = \theta + \alpha \nabla_\theta \log \pi_\theta(s_t|a_t) R_{T_i}(\theta)$$

Is it enough for driving?

Issue: Autonomous driving yields a large amount of uncertainty. The reward function R_t suffers from high variance slowing down the policy convergence.

Solution: **Evaluate the policy** predictions with a **convolutional neural network controller** approximating the **value function**. Replace R_t with the temporal difference error resulting from the policy evaluation.

Result: Generate added **bonuses** to guide the agent towards more robust strategies.

$$\theta_i' = \theta + \alpha \nabla_\theta \log \pi_\theta(s_t|a_t) \delta_t$$

- Where δ_t is the multi-step TD error that consists in bootstrapping the sampled returns from the value function estimate:

$$\delta_t = \left[\sum_{j=t}^{t+H-1} \gamma^{j-t} r_j \right] + \gamma^H V(s_{t+H}) - V(s_t)$$

- **Outer loop:** "Across-task learning":
  Transfer the task-specific models learned through **policy evaluations** to the meta-learner.
  A meta-gradient update will generate an **optimized initialization** for the RL agent to adapt faster in new test environments.

$$\theta \leftarrow \theta + \beta \nabla_\theta \sum_{T_i \sim p(T)} R_{T_i}(\theta_i')$$

- Iterate until an accepted performance is reached…

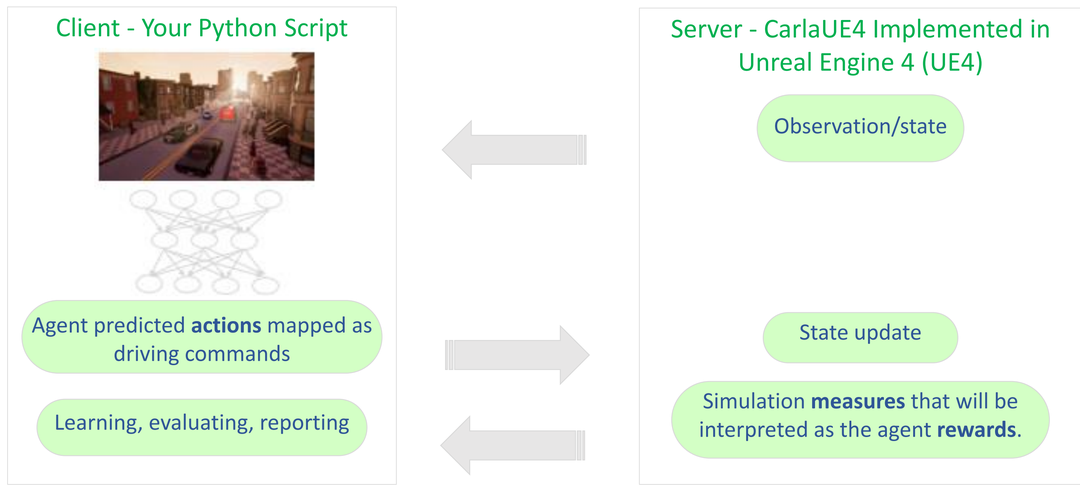
yasmina.jaafra@etu.unistra.fr

Drive in CARLA Simulator

Task specification:

CARLA simulator: A server-client system

- Quite realistic and customizable environments: static objects Vs Dynamic non-player characters



- **Observation space**: Sensors outputs (RGB images) reflecting current state.
- **Action space.** Discrete driving instructions (steering, throttle, brake and combinations).
- **Reward:** A weighted sum of measurements. The distance traveled to target, speed, collisions damage and overlaps with sidewalk and opposite lane.

Environment Variability: Induce non-stationary environments across training episodes by varying several server settings.
(1)Task complexity: different towns, start and end positions (straight or with-turn driving).
(2) Traffic density: control the number of dynamic objects such as pedestrians and vehicles.
(3)Visual effects: select a combination of weather and illumination conditions (controlling sun position, radiation intensity, cloudiness and precipitation.
→ "seen": environments used for meta-training
→"unseen"; environments exclusively used in test-time adaptation

Fig. 1: Sample of experimental environments

Findings:

Does our approach (1) **adapt faster** at training time and (2) **generalize better** to unseen environments?

- Compare the continuous-adapting **MRL** initialization with conventionally **pre-trained** and **randomly** initialized RL algorithms
- **Adaptation performance:**

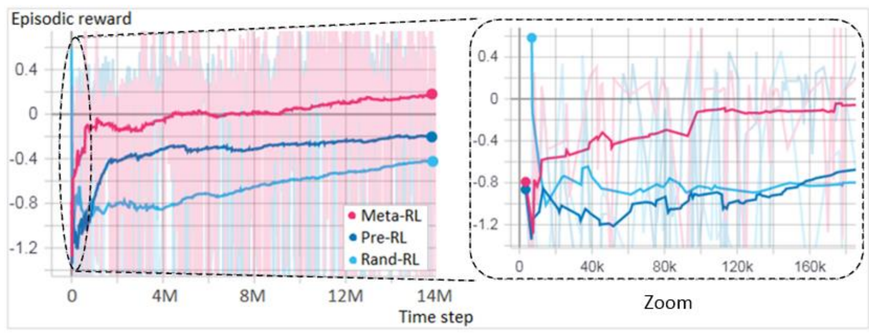
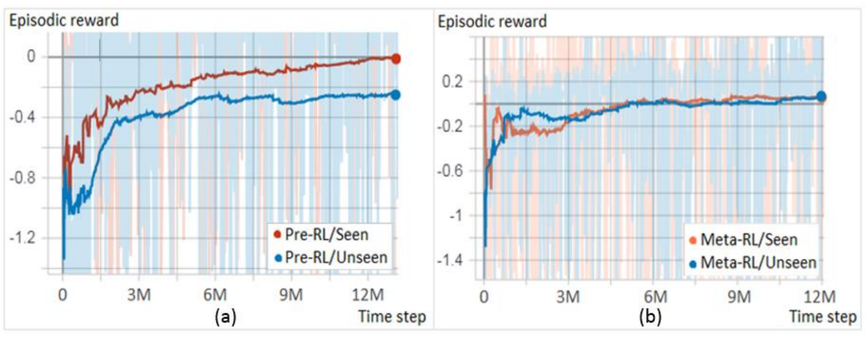


Fig. 2: Test-time adaptation performance of the 3 models.

- Our approach generates models adapting faster in "unseen" environments. It has distinctly surpassed pre-trained and randomly initialized RL algorithms after 10k steps (500 gradient descents).
- **Generalization capabilities:**
  We compare the models behavior on "seen" and "unseen" environments



- Fig 3.a: The performance of the pre-trained standard RL decreased notably in "unseen" environments due to the lack of generalization capabilities.
- Fig 3.b: In the contrary, the results do **not show** a significant "shortfall" of our approach performance between the 2 scenarios reflecting its adaptation in non-stationary conditions.

Future work

Although the results show a promising performance of the continuous-adapting MRL, we think that further evaluation of the approach is still required. We plan in future work to focus on few-shot learning pertinence when evolving from low to high-dimensional RL settings. We will examine if a specific threshold (k shots) is necessary to guarantee efficient gradient descents for complex tasks. We additionally propose to consider the assumption that few-shot regimes may bias meta-learning by "allowing it to perform better from limited experience but also limits its capacity of utilizing more data".