Automated Reinforcement Learning (AutoRL): A Survey and Open Problems

Jack Parker-Holder[1,*], Raghu Rajan[2,*], Xingyou Song[3,*], André Biedenkapp[2], Yingjie Miao[3], Theresa Eimer[4], Baohe Zhang[2], Vu Nguyen[5], Roberto Calandra[6], Aleksandra Faust[3,†], Frank Hutter[2,7,†], Marius Lindauer[4,†]

[*] co-first authors

Paper link:

[†] co-last authors

universität freiburg

[1] University of Oxford
[2] University of Freiburg
[3] Google Research, Brain Team
[4] Leibniz University Hannover
[5] Amazon Australia
[6] Meta AI
[7] Bosch Center for Artificial Intelligence

Overview

- Reinforcement Learning (RL) often highly **sensitive** to design choices
- **AutoML** has automated design choices in other parts of Machine Learning
  - Initial **promising** results in RL
- Additional **challenges unique to RL**
- **AutoRL** has been **gathering momentum** as an important area of research
  - **Existing approaches** like metaRL, curriculum learning, meta-gradients
- This work **aims** to:
  - Unify the field of AutoRL with a **common taxonomy**
  - **Survey** each of these areas in detail
  - Pose **open problems**

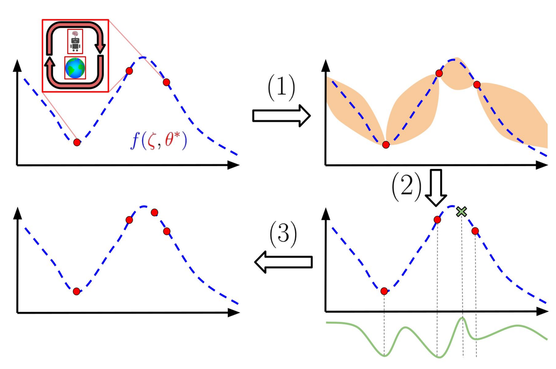
Taxonomy and General Properties

Class	Algorithm properties					What is automated?
Random/Grid Search (4.1)	†††	■	⇉	✓	≐	hyperparameters, architecture, algorithm
Bayesian Optimization (4.2)	†††	■	⇉	✓	≐	hyperparameters, architecture, algorithm
Evolutionary Approaches (4.3)	†††	■	⇉	✓	≈	hyperparameters, architecture, algorithm
Meta-Gradients (4.4)	†	∇	→	●	≈	hyperparameters
Blackbox Online Tuning (4.5)	†	■	→	✓	≈	hyperparameters
Learning Algorithms (4.6)	†††	■	⇉	●	≐	algorithm
Environment Design (4.7)	†††	■	⇉	●	≈	environment

† only uses a single trial, ††† requires multiple trials
∇ requires differentiable variables, ■ works with non-differentiable hyperparameters
⇉ parallelizable → not parallelizable
✓ works for any RL algorithm, ● works for only some classes of RL algorithms
≐ static optimization, ≈ dynamic optimization

Bayesian Optimization (BO) Based

- Builds a **model** of the response surface
  - Queries 'better' points to evaluate
- Trades off **exploration-exploitation**
- AlphaGo improved from 50% to 65% win rate in self-play [Chen et al. 2018]
- Multi-fidelity
  - BOHB [Falkner et al. 2017] used for tuning **architecture and HPs** for Learning to design RNA [Runge et al. 2019]
  - BO for Iterative Learning (BOIL) [Nguyen et al. 2020] used **knowledge of learning curves** to efficiently tune HPs
- Not many approaches yet that perform dynamic tuning



Prototypical BO procedure

Meta-Gradients

- Optimise meta-parameters in an outer loop using **gradients** of an objective **w.r.t. meta-parameters**, optimise parameters in an inner loop
- Tune **online** in a single run
- Efficient
- Require **differentiable** outer objective
- Meta-gradient RL [Xu et al. 2018] considered gradients of the objective w.r.t. the **bootstrapping hyperparameter, λ**, and the **discount factor, γ**
- RL-DARTS [Miao et al., 2021] performs **differentiable architecture search** in an RL setting

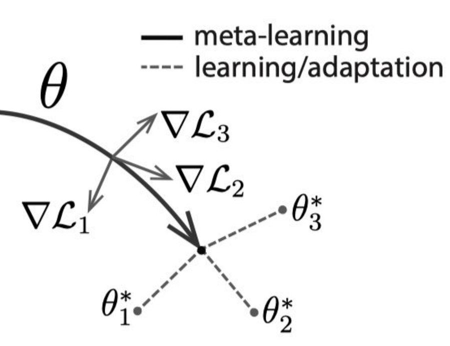
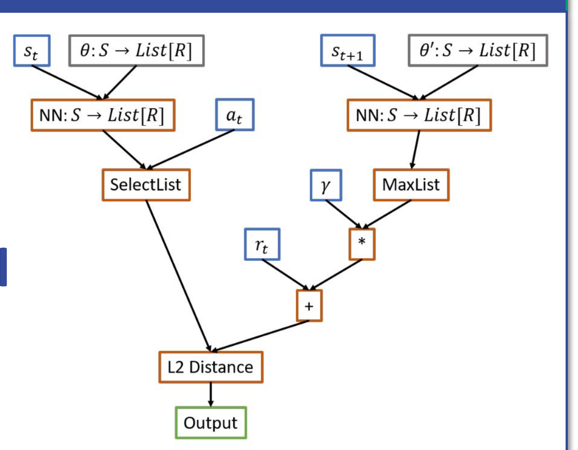


Image taken from MAML [Finn et al. 2017]

Learning RL Algorithms

- **Learning to Learn:** RL2 [Duan et al. 2016] use an RNN with **past history as input** to tackle interrelated tasks
- **Meta-learn loss function:** Loss function is a **neural network** as in Evolved Policy Gradient [Houthooft et al., 2018] which provides a loss function to be optimised in an inner loop. Or the loss function is represented as a **symbolic expression**, e.g., as a Directed Acyclic Graph (DAG) in Evolving reinforcement learning algorithms [Co-Reyes et al. 2021]
- Most **MetaRL** methods come under this category

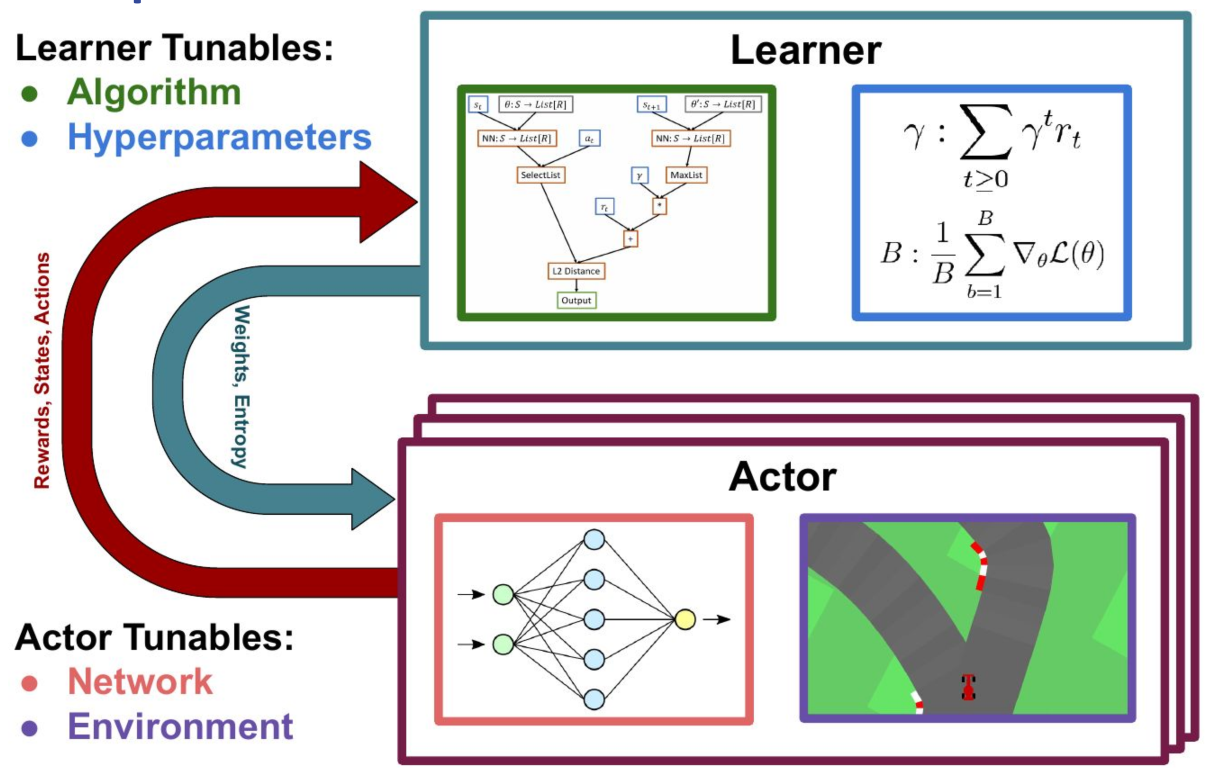


Visualisation of an RL loss function as a DAG

AutoRL

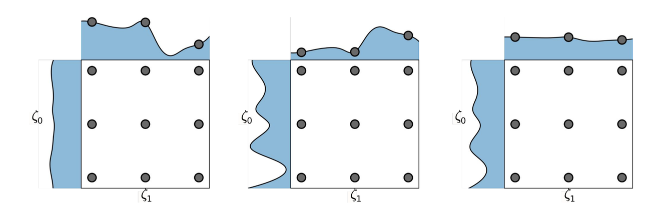
- **Bi-level optimization:** $\max_\zeta f(\zeta,\theta^*)$ s.t. $\theta^* \in \arg\max_\theta J(\theta;\zeta)$

- **Pipeline components:**

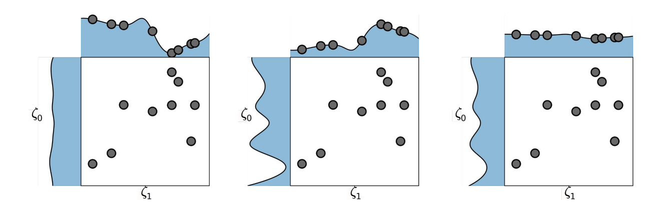


Random/Grid Search Based

- **Easy** to implement
- Good for **visualizing**
- **Do not use information** obtained during optimisation
  - **Multi-fidelity** methods like Hyperband [Li et al. 2017] implicitly do this
- **Do not scale well** to high dimensions and are **not dynamic**



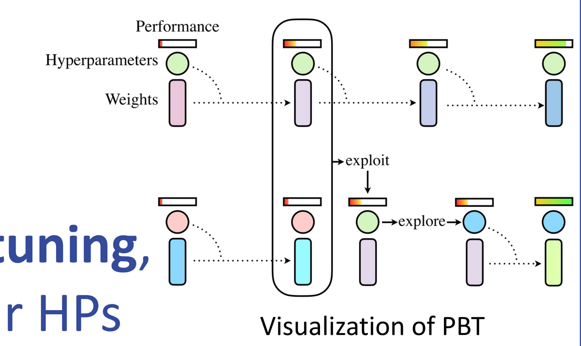
Grid search at various points during the optimisation



Random search at various points during the optimisation

Evolutionary Approaches

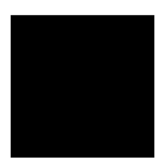
- Maintain **populations** and **mutate** members' hyperparameters and parameters
- Population-Based Training (PBT) [Jaderberg et al. 2017]-like methods capable of **dynamic tuning**, **exploit** top-performing members, **explore** their HPs
  - Zhang et al. 2021 compare random, BO-based, PBT-like approaches
- Methods like NEAT [Stanley & Miikkulainen, 2002] evolve both Neural Network **weights** and **architectures**
- **Hybrid approaches** such as PB2 [Parker-Holder et al. 2020] and DEHB [Awad et al. 2021] employ models to increase **efficiency**



Visualization of PBT

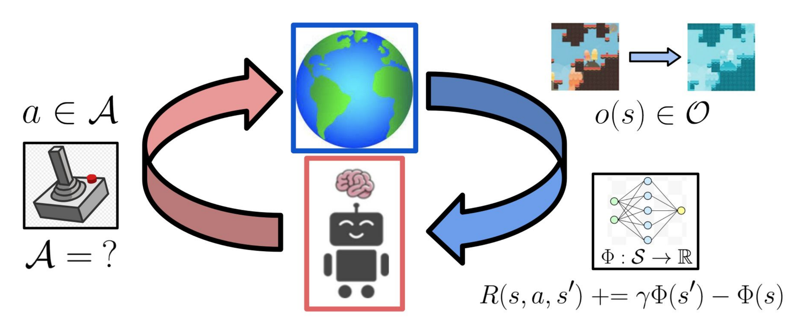
Blackbox Online Tuning

- Adapt HPs **on the fly**
- Agent57 [Badia et al. 2020] uses **multi-armed bandits** to adaptively select from **several exploration policies** and achieves superhuman performance in all 57 Atari games
- More **flexible** as it is blackbox but can be inefficient



Environment Design

- Optimise environment components of a POMDP
- **Reward Shaping:** Faust et al. 2019 use evolutionary search to shape parametric rewards
- **Observation Space:** DrAC [Raileanu et al. 2020] use bandits to select **image transformation** (e.g., crop, rotate, flip) to apply to the observations
- **Multiple Environment Components, Unsupervised:** Curriculum learning approaches such as POET [Wang et al. 2019] and PAIRED [Dennis et al. 2020] modify the **initial state distribution** and **state/observation space** to present easier problems initially to speed up learning
- **Multiple Environment Components, Supervised:** Learning Synthetic Environments [Ferreira et al. 2021] learns **dynamics** and **reward functions** as NNs which are optimised in an outer loop



Examples of Optimizable components of an environment: Action Space, A; Observation Space, O; Reward function, R