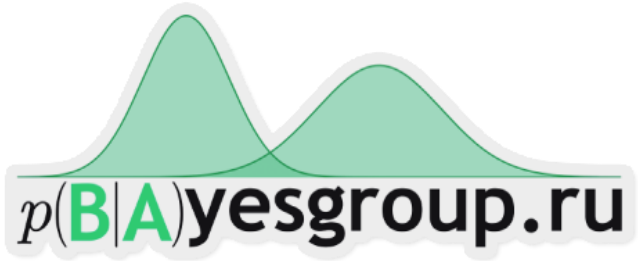
Meta-learning in neural networks

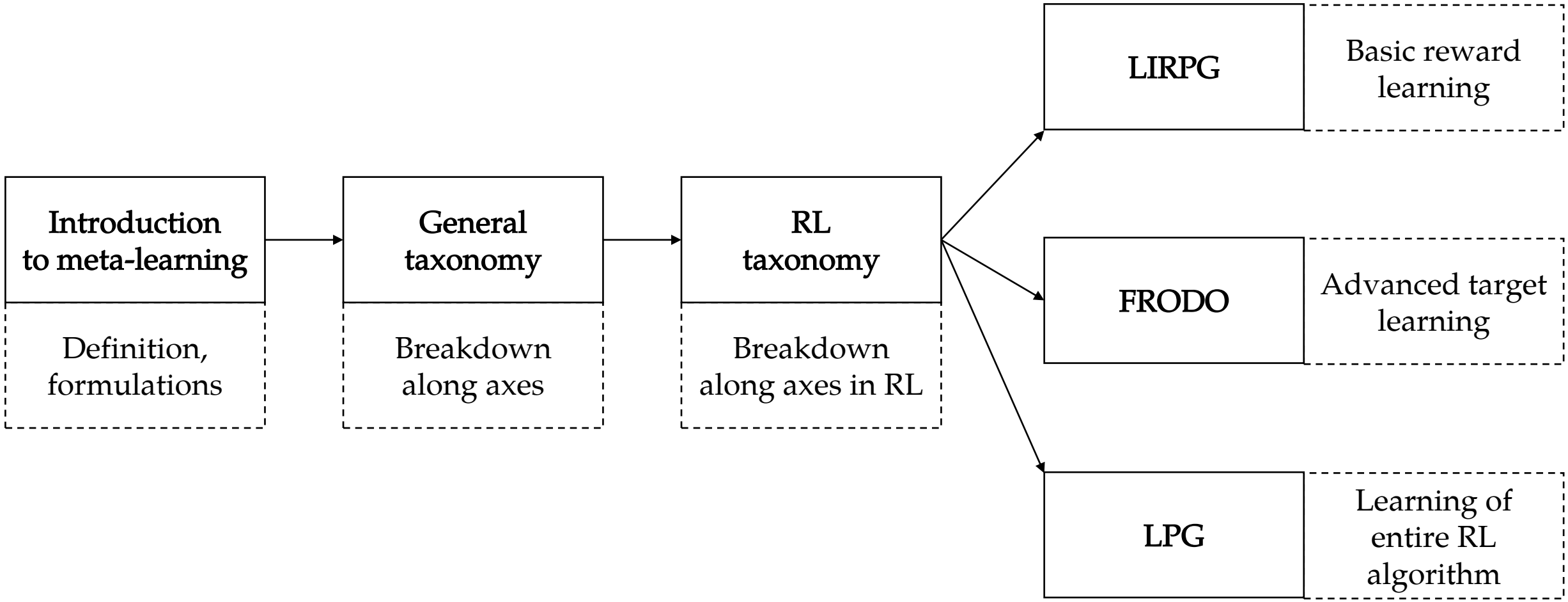
Alexander Grishin

SAMSUNG Research Russia



HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

Plan


Introduction to meta-learning
Definition, formulations
General taxonomy
Breakdown along axes
RL taxonomy
Breakdown along axes in RL
LIRPG
Basic reward learning
FRODO
Advanced target learning
LPG
Learning of entire RL algorithm

Informal definition

Conventional ML

Improving model predictions over data instances

$$\text{Training data } \mathcal{D} \xrightarrow{\mathcal{A}(\cdot;\omega)} \text{Better model } \theta^* \leftrightarrow f(\cdot;\theta^*)$$

Meta ML

Improving learning algorithm over learning episodes

$$\text{Learning process } \mathcal{D}, \mathcal{A} \xrightarrow{\mathcal{A}^{\text{meta}}(\cdot,\cdot)} \text{Better algorithm } \omega^* \leftrightarrow \mathcal{A}(\cdot;\omega^*)$$

Task distribution view

Conventional ML

Train data, meta-parameter

$$\omega, \mathcal{D}^{\text{train}}$$

Train parameters

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\mathcal{D}^{\text{train}}; \theta, \omega)$$

Apply to test data

$$f(\mathcal{D}^{\text{test}}, \theta^*)$$

Overfitting: parameters work well on train part, but fail on test one

Meta ML

Empirical task distribution

$$\mathcal{D}_{\text{source}} = \{(\mathcal{D}^{\text{train}}_{\text{source}}, \mathcal{D}^{\text{test}}_{\text{source}})^{(i)}\}_{i=1}^{M}$$

Train meta-parameters

$$\omega^* = \arg\max_{w} \log p(w \mid \mathcal{D}_{\text{source}})$$

Apply to target task(s)

$$\theta^* = \arg\max_{\theta} \log p(\theta \mid w^*, \mathcal{D}^{\text{train}}_{\text{target}})$$

$$f(\mathcal{D}^{\text{test}}_{\text{target}}, \theta^*)$$

Meta overfitting: meta-parameters work well on source tasks, but fail on target task(s)

Bilevel optimization view

$$\text{Outer problem:} \quad w^* = \arg\min_w \sum_{i=1}^{M} \mathcal{L}^{\text{meta}}(\theta^{*(i)}(w), w, \mathcal{D}_{\text{source}}^{\text{val}\ (i)})$$

$$\text{Inner problem:} \quad \text{s.t.} \quad \theta^{*(i)}(\omega) = \arg\min_\theta \mathcal{L}^{\text{task}}(\theta, \omega, \mathcal{D}_{\text{source}}^{\text{train}}(i))$$

- There could be only one task
- We could care not only about final performance but also about intermediate one
- Meta-parameters could be instrumental
  (i.e., they could affect only resulting model but not the loss directly)
- There are some theoretical guarantees*

*Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R. and Pontil, M., 2018, July. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning* (pp. 1568-1577). PMLR.

Taxonomy

Why?

Goal of meta-learning

- Fast adaptation vs asymptotic performance
- Multi vs single-task
- Online vs offline

How?

Optimizer of the outer problem

- Gradient
- Reinforcement learning
- Evolution

What?

Choice of meta-knowledge ω to meta-learn

- Initial Condition
- Optimizer
- Hyperparameter
- Feed-Forward model
- Metric
- Loss/Reward
- Architecture
- Exploration Policy
- Dataset/Environment
- Instance Weights
- Feature/Metric
- Data Augmentation/Noise
- Modules
- Annotation Policy

Taxonomy in RL

Why?

Goal of meta-learning

- Fast adaptation vs asymptotic performance ✖
- Multi vs single-task ✔
- Online vs offline ✔

How?

Optimizer of the outer problem

- Gradient ✖
- Reinforcement learning ✔
- Evolution ✔

What?

Choice of meta-knowledge ω to meta-learn

- Initial Condition
- Optimizer
- Hyperparameter
- Feed-Forward model
- Metric
- Loss/Reward
- Architecture
- Exploration Policy
- Dataset/Environment
- Instance Weights
- Feature/Metric
- Data Augmentation/Noise
- Modules
- Annotation Policy

Taxonomy in RL

Why? How? What?

Algorithm	Algorithm properties			What is meta-learned?
IDBD, SMD [30, 27]	†	□	→	learning rate
SGD^2 [1]	†††	■	←	optimiser
RL^2, Meta-RL [9, 39]	†††	■	X	recurrent network
MAML, REPTILE [11, 23]	†††	□	←	initial params
Meta-Gradient [43, 46]	†	□	→	γ, λ, reward
Meta-Gradient [38, 44, 40]	†	□	←	auxiliary tasks, hyperparams, reward weights
ML^3, MetaGenRL [2, 19]	†††	■	←	loss function
Evolved PG [16]	†††	■	X	loss function
Oh et al. 2020 [24]	†††	■	←	target vector
This paper	†	■	←	target

□ white box, ■ black box, † single lifetime, ††† multi-lifetime

← backward mode, → forward mode, X no meta-gradient

*Xu, Z., van Hasselt, H., Hessel, M., Oh, J., Singh, S. and Silver, D., 2020. Meta-gradient reinforcement learning with an objective discovered online. arXiv preprint arXiv:2007.08433.

LIRPG

Algorithm	Algorithm properties			What is meta-learned?
IDBD, SMD [30, 27]	†	□	→	learning rate
SGD^2 [1]	†††	■	←	optimiser
RL^2, Meta-RL [9, 39]	†††	■	X	recurrent network
MAML, REPTILE [11, 23]	†††	□	←	initial params
Meta-Gradient [43, 46]	†	□	→	γ, λ, reward
Meta-Gradient [38, 44, 40]	†	□	←	auxiliary tasks, hyperparams, reward weights
ML^3, MetaGenRL [2, 19]	†††	■	←	loss function
Evolved PG [16]	†††	■	X	loss function
Oh et al. 2020 [24]	†††	■	←	target vector
This paper	†	■	←	target

□ white box, ■ black box, † single lifetime, ††† multi-lifetime

← backward mode, → forward mode, X no meta-gradient

*Zheng, Z., Oh, J. and Singh, S., 2018. On learning intrinsic rewards for policy gradient methods. arXiv preprint arXiv:1804.06459.

LIRPG

Motivation: reward is a good metric but (probably) bad learning signal

Idea: transform reward and try to make it a good learning signal

$$r(s,a) = r^{\text{ex}}(s,a) + r_\eta^{\text{in}}(s,a)$$

- θ: policy parameters
- η: intrinsic reward parameters
- r^{ex}: extrinsic reward from the environment
- $r_\eta^{in} = r_\eta^{in}(s,a)$: intrinsic reward estimated by η
- $G^{ex}(s_t,a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r_i^{ex}$
- $G^{in}(s_t,a_t) = \sum_{i=t}^{\infty} \gamma^{t-i} r_\eta^{in}(s_i,a_i)$
- $G^{ex+in}(s_t,a_t) = \sum_{i=t}^{\infty} \gamma^{i-t}(r_i^{ex} + \lambda r_\eta^{in}(s_i,a_i))$
- $J^{ex} = E_\theta[\sum_{t=0}^{\infty} \gamma^t r_t^{ex}]$
- $J^{in} = E_\theta[\sum_{t=0}^{\infty} \gamma^t r_\eta^{in}(s_t,a_t)]$
- $J^{ex+in} = E_\theta[\sum_{t=0}^{\infty} \gamma^t (r_t^{ex} + \lambda r_\eta^{in}(s_t,a_t)]$
- λ: relative weight of intrinsic reward.

Before:

$$\theta' = \theta + \alpha \hat{\nabla}_\theta J^{\text{ex}}(\theta)$$

$$\hat{\nabla}_\theta J^{\text{ex}}(\theta) = \frac{1}{T} \sum_{t=1}^{T} G^{\text{ex}}(s_t,a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

After:

$$\theta' = \theta + \alpha \hat{\nabla}_\theta J_\eta^{\text{ex+in}}(\theta)$$

$$\hat{\nabla}_\theta J_\eta^{\text{ex+in}}(\theta) = \frac{1}{T} \sum_{t=1}^{T} G_\eta^{\text{ex+in}}(s_t,a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\boxed{\eta' = \eta + \alpha \hat{\nabla}_{\theta'} J^{\text{ex}}(\theta') \nabla_\eta \theta'}$$

$$\nabla_\eta \theta' = \frac{\alpha\lambda}{T} \sum_{t=1}^{T} \nabla_\eta G_\eta^{\text{ex}}(s_t,a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

*Zheng, Z., Oh, J. and Singh, S., 2018. On learning intrinsic rewards for policy gradient methods. arXiv preprint arXiv:1804.06459.

LIRPG

Nuances:

- Weight of intrinsic reward is 0.01
- Importance sampling instead of additional sampling
- A2C instead of REINFORCE

- θ: policy parameters
- η: intrinsic reward parameters
- r^{ex}: extrinsic reward from the environment
- $r^{in}_{\eta} = r^{in}_{\eta}(s, a)$: intrinsic reward estimated by η
- $G^{ex}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} r^{ex}_i$
- $G^{in}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{t-i} r^{in}_{\eta}(s_i, a_i)$
- $G^{ex+in}(s_t, a_t) = \sum_{i=t}^{\infty} \gamma^{i-t} (r^{ex}_i + \lambda r^{in}_{\eta}(s_i, a_i))$
- $J^{ex} = E_{\theta}[\sum_{t=0}^{\infty} \gamma^t r^{ex}_t]$
- $J^{in} = E_{\theta}[\sum_{t=0}^{\infty} \gamma^t r^{in}_{\eta}(s_t, a_t)]$
- $J^{ex+in} = E_{\theta}[\sum_{t=0}^{\infty} \gamma^t (r^{ex}_t + \lambda r^{in}_{\eta}(s_t, a_t)]$
- λ: relative weight of intrinsic reward.

Algorithm 1 LIRPG: Learning Intrinsic Reward for Policy Gradient

Input: step-size parameters α and β
Init: initialize θ and η with random values
repeat
  Sample a trajectory $\mathcal{D} = \{s_0, a_0, s_1, a_1, \cdots\}$ by interacting with the environment using π_θ
  Approximate $\nabla_\theta J^{ex+in}(\theta; \mathcal{D})$ by Equation 4
  Update $\theta' \leftarrow \theta + \alpha \nabla_\theta J^{ex+in}(\theta; \mathcal{D})$
  Approximate $\nabla_{\theta'} J^{ex}(\theta'; \mathcal{D})$ on $\mathcal{D}$ by Equation 11
  Approximate $\nabla_\eta \theta'$ by Equation 10
  Compute $\nabla_\eta J^{ex} = \nabla_{\theta'} J^{ex}(\theta'; \mathcal{D}) \nabla_\eta \theta'$
  Update $\eta' \leftarrow \eta + \beta \nabla_\eta J^{ex}$
until done

*Zheng, Z., Oh, J. and Singh, S., 2018. On learning intrinsic rewards for policy gradient methods. arXiv preprint arXiv:1804.06459.

LIRPG

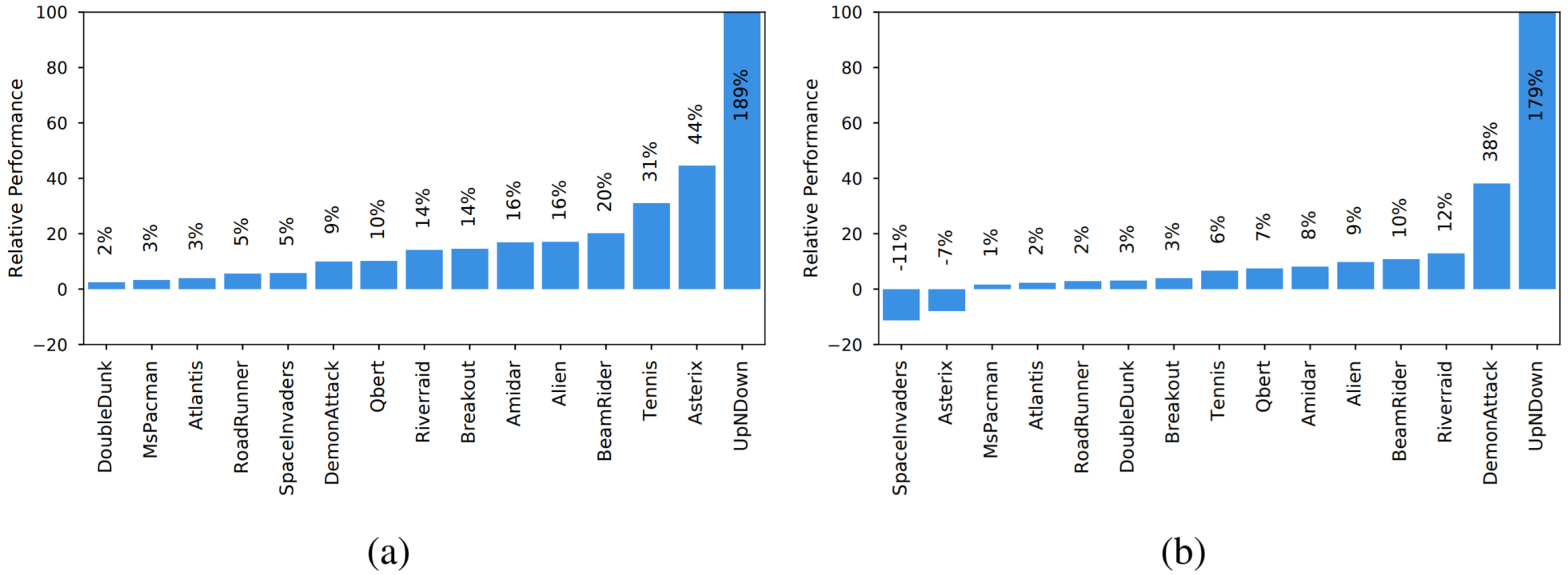
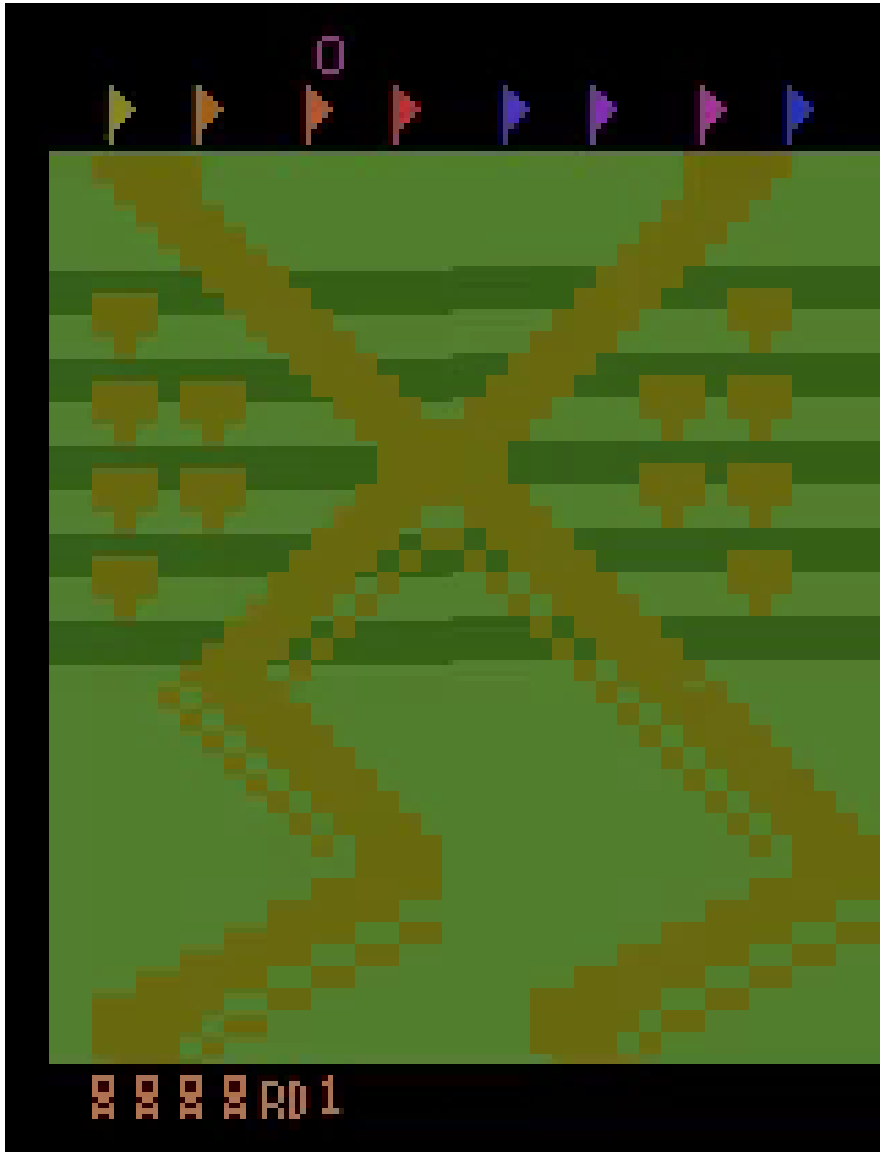
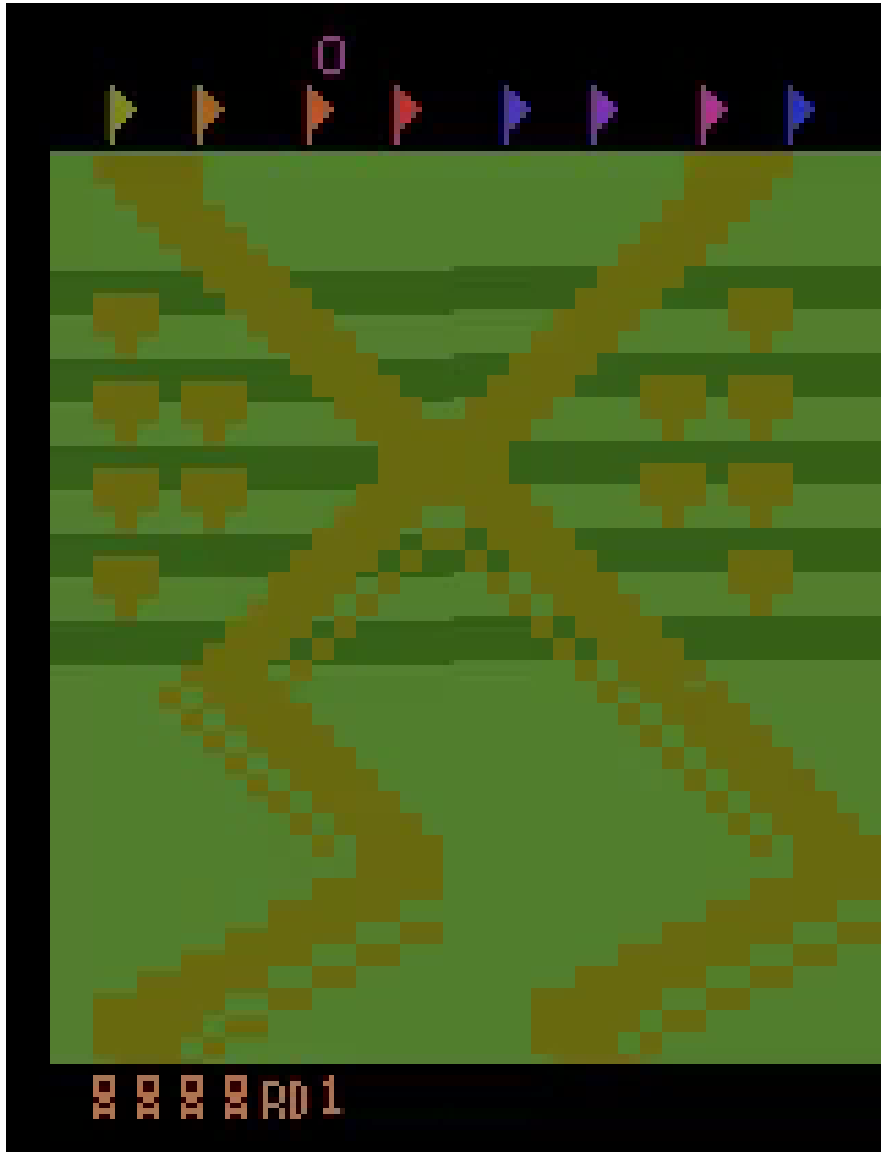


Figure 2: (a) Improvements of LIRPG augmented agents over A2C baseline agents. (b) Improvements of LIRPG augmented agents over live-bonus augmented A2C baseline agents. In both figures, the columns correspond to different games labeled on the x-axes and the y-axes show human score normalized improvements.

*Zheng, Z., Oh, J. and Singh, S., 2018. On learning intrinsic rewards for policy gradient methods. arXiv preprint arXiv:1804.06459.


0
RD 1


0
RD 1

FRODO

Algorithm	Algorithm properties			What is meta-learned?
IDBD, SMD [30, 27]	†	□	→	learning rate
SGD^2 [1]	†††	■	←	optimiser
RL^2, Meta-RL [9, 39]	†††	■	X	recurrent network
MAML, REPTILE [11, 23]	†††	□	←	initial params
Meta-Gradient [43, 46]	†	□	←	γ, λ, reward
Meta-Gradient [38, 44, 40]	†	□	←	auxiliary tasks, hyperparams, reward weights
ML^3, MetaGenRL [2, 19]	†††	■	←	loss function
Evolved PG [16]	†††	■	X	loss function
Oh et al. 2020 [24]	†††	■	←	target vector
This paper	†	■	←	target

□ white box, ■ black box, † single lifetime, ††† multi-lifetime

← backward mode, → forward mode, X no meta-gradient

*Xu, Z., van Hasselt, H., Hessel, M., Oh, J., Singh, S. and Silver, D., 2020. Meta-gradient reinforcement learning with an objective discovered online. arXiv preprint arXiv:2007.08433.

FRODO

- Learn target, not reward

Before (A2C): $\Delta\theta \propto \frac{1}{T}\sum_{t=1}^{T}\Big((G(s_t,a_t)-V_\theta(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)+c_1\nabla_\theta(G(s_t,a_t)-V_\theta(s_t))^2+c_2\nabla_\theta\mathbb{H}(\pi(\cdot|s_t))\Big)$

After: $\Delta\theta \propto \frac{1}{T}\sum_{t=1}^{T}\Big((g_\eta(\tau_{t+})-V_\theta(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)+c_1\nabla_\theta(g_\eta(\tau_{t+})-V_\theta(s_t))^2+c_2\nabla_\theta\mathbb{H}(\pi(\cdot|s_t))\Big)$

- Multi-step inner loop

η fixed, $\quad \theta_i \xrightarrow{\eta} \theta_{i+1} \xrightarrow{\eta} \ldots \xrightarrow{\eta} \theta_{i+M}$

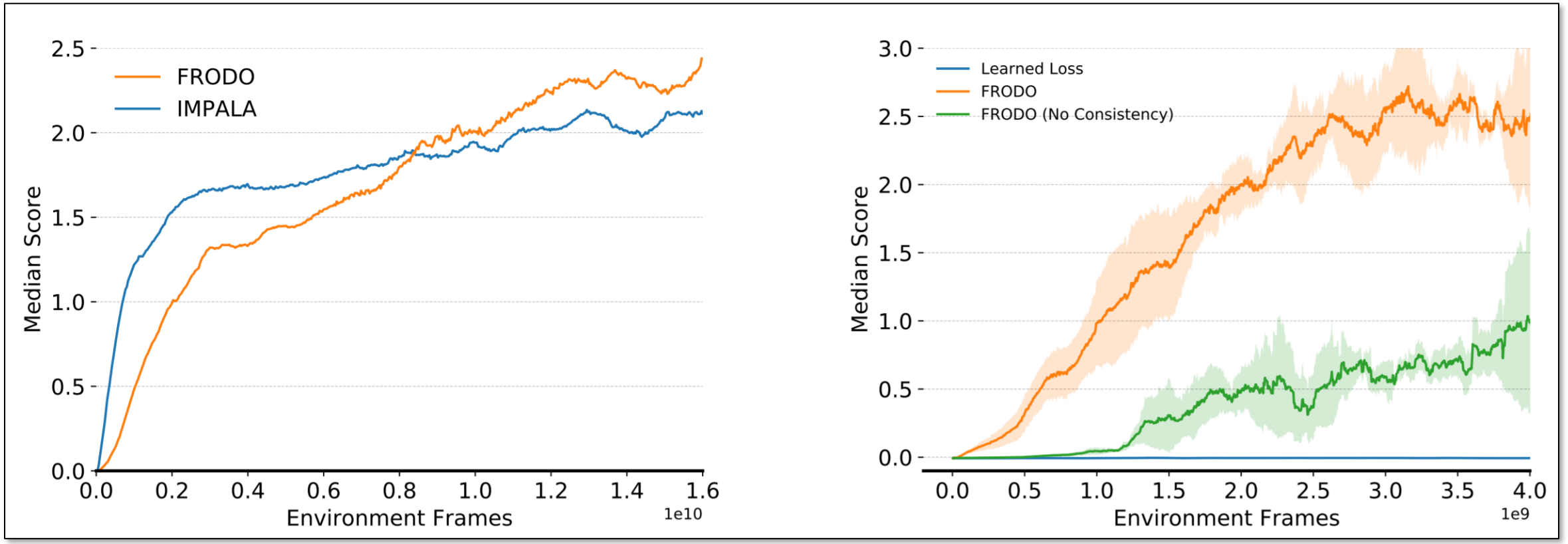
$\Delta\eta \propto \nabla_{\theta_{i+M}} L^{\text{outer}}(\tau_{i+M+1}, \theta_{i+M}(\theta_i,\eta))\nabla_\eta\theta_{i+M}(\theta_i,\eta)$

- Consistency loss

$$||G_\eta(\tau_{t+}) - (G_\eta(\tau_{(t+1)+}) + r_t)||^2 \to \min$$

*Xu, Z., van Hasselt, H., Hessel, M., Oh, J., Singh, S. and Silver, D., 2020. Meta-gradient reinforcement learning with an objective discovered online. arXiv preprint arXiv:2007.08433.

FRODO



*Xu, Z., van Hasselt, H., Hessel, M., Oh, J., Singh, S. and Silver, D., 2020. Meta-gradient reinforcement learning with an objective discovered online. arXiv preprint arXiv:2007.08433.

LPG

Algorithm	Algorithm properties			What is meta-learned?
IDBD, SMD [30, 27]	†	□	→	learning rate
SGD^2 [1]	†††	■	←	optimiser
RL^2, Meta-RL [9, 39]	†††	■	X	recurrent network
MAML, REPTILE [11, 23]	†††	□	←	initial params
Meta-Gradient [43, 46]	†	□	←	γ, λ, reward
Meta-Gradient [38, 44, 40]	†	□	←	auxiliary tasks, hyperparams, reward weights
ML^3, MetaGenRL [2, 19]	†††	■	←	loss function
Evolved PG [16]	†††	■	X	loss function
Oh et al. 2020 [24]	†††	■	←	target vector
This paper	†	■	←	target

□ white box, ■ black box, † single lifetime, ††† multi-lifetime

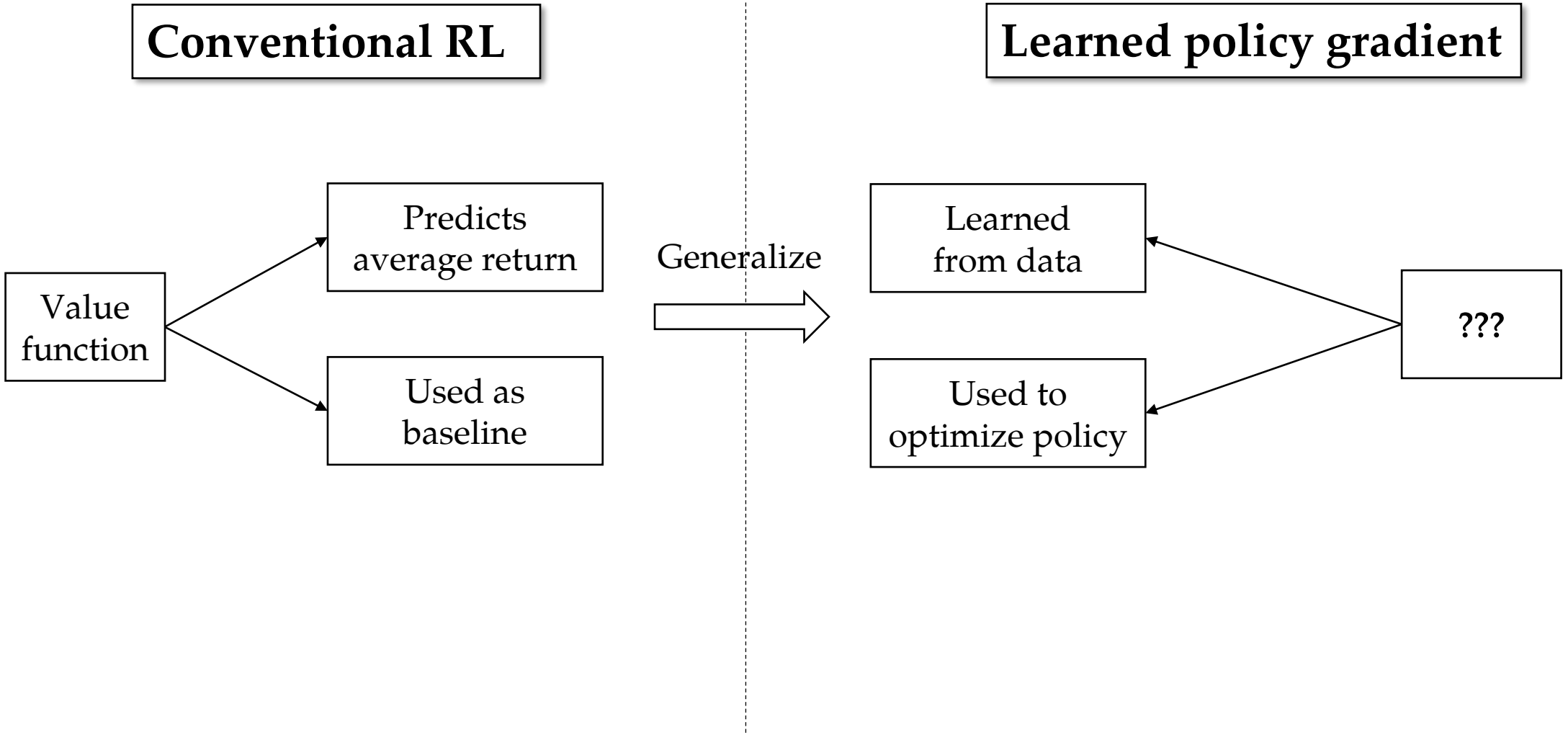
← backward mode, → forward mode, X no meta-gradient

*Xu, Z., van Hasselt, H., Hessel, M., Oh, J., Singh, S. and Silver, D., 2020. Meta-gradient reinforcement learning with an objective discovered online. arXiv preprint arXiv:2007.08433.

LPG

*Oh, J., Hessel, M., Czarnecki, W.M., Xu, Z., van Hasselt, H., Singh, S. and Silver, D., 2020. Discovering reinforcement learning algorithms. arXiv preprint arXiv:2007.08794.

LPG



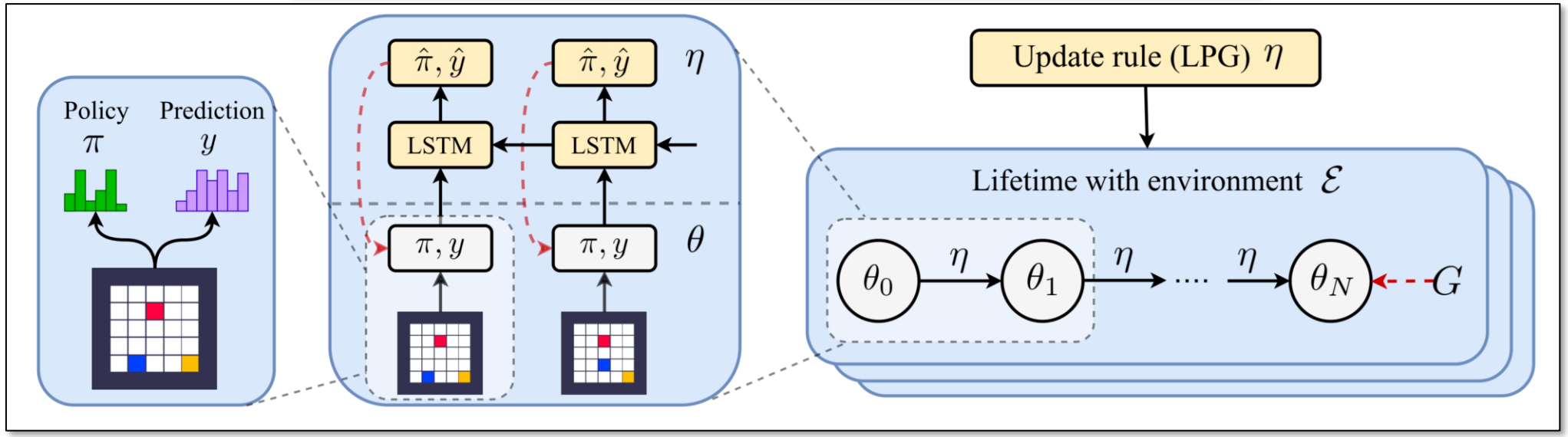
*Oh, J., Hessel, M., Czarnecki, W.M., Xu, Z., van Hasselt, H., Singh, S. and Silver, D., 2020. Discovering reinforcement learning algorithms. arXiv preprint arXiv:2007.08794.

LPG

	In LPG	You can think of
What agent should predict?	$\hat{y}_t = \hat{y}_t(\{r_i, d_i, \gamma, \pi(a_i\|s_i), y_\theta(s_i)\}_{i \geq t})$	$\mathbb{E}_{a_t \sim \pi_\theta(s_t)}(r_t + \gamma V_\theta(s_{t+1}))$
How agent predicts it?	$y_\theta(s_t)$	$V_\theta(s_t)$
How are predictions used for policy improvement?	$\hat{\pi}_t = \hat{\pi}_t(\{r_i, d_i, \gamma, \pi(a_i\|s_i), y_\theta(s_i)\}_{i \geq t})$	$G_t - V_\theta(s_t)$

$$\Delta\theta \propto \mathbb{E}\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - V_\theta(s_t)) - \alpha \nabla_\theta \big(V_\theta(s_t) - \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)}(r_t + \gamma V_\theta(s_{t+1}))\big)^2\Big]$$

$$\Delta\theta \propto \mathbb{E}\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{\pi}_t - \alpha_y \nabla_\theta \mathrm{D}_{\mathrm{KL}}(y_\theta(s_t)||\hat{y}_t)\Big]$$



LPG

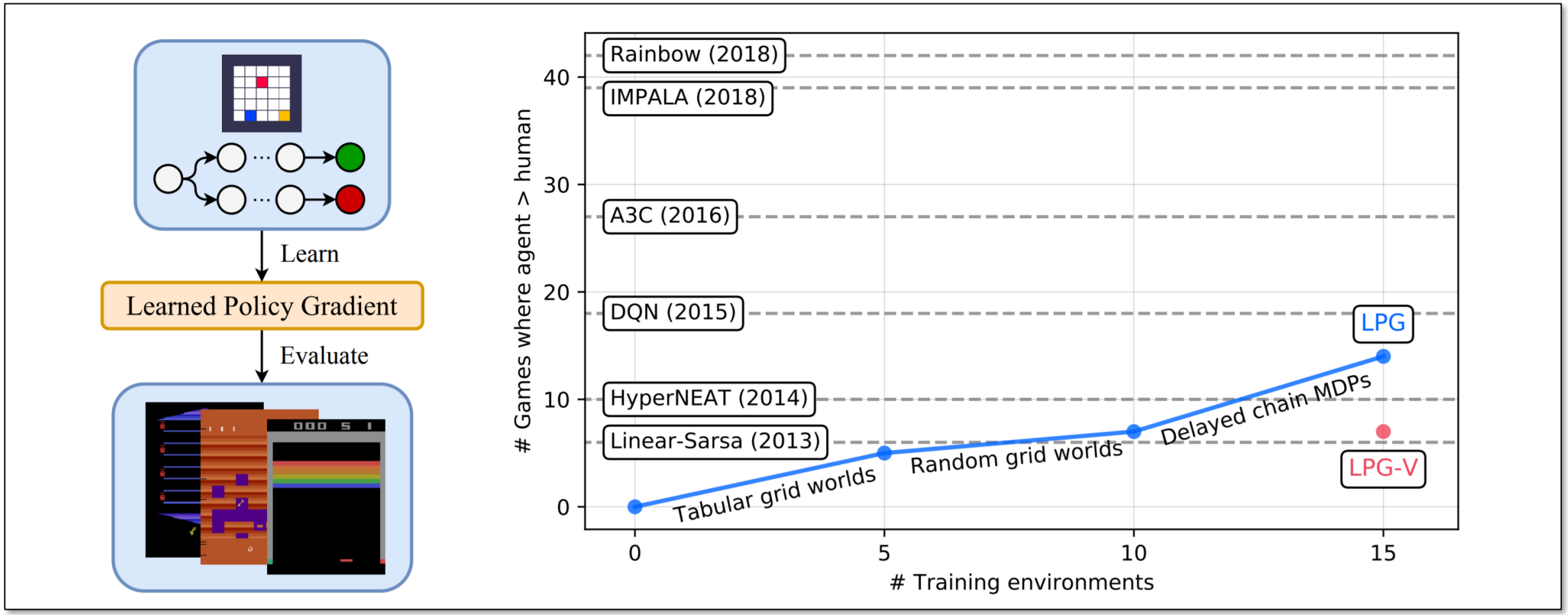
Meta-gradient (for specific timestep):

$$\nabla_\eta \log \pi_{\theta_N}(a|s)G + \beta_0 \nabla_\eta \mathcal{H}(\pi_{\theta_N}) + \beta_1 \nabla_\eta \mathcal{H}(y_{\theta_N}) - \beta_2 \nabla_\eta ||\hat{\pi}||_2^2 - \beta_3 \nabla_\eta ||\hat{y}||_2^2$$

- Performance of policy in environment
- Regularization of agent's output
- Regularization of learned algorithm's

*Oh, J., Hessel, M., Czarnecki, W.M., Xu, Z., van Hasselt, H., Singh, S. and Silver, D., 2020. Discovering reinforcement learning algorithms. arXiv preprint arXiv:2007.08794.

LPG



*Oh, J., Hessel, M., Czarnecki, W.M., Xu, Z., van Hasselt, H., Singh, S. and Silver, D., 2020. Discovering reinforcement learning algorithms. arXiv preprint arXiv:2007.08794.

Closing thoughts

- Meta learning brings (self/un)supervised learning's efficiency to the new level (at least on paper)
- Meta learning able to overcome algorithms' inductive bias
- Meta RL promise to solve RL issues of the day: brittleness and lack of data
- Meta RL is young and slightly awkward for now but has a big potential

The End

Extra slides

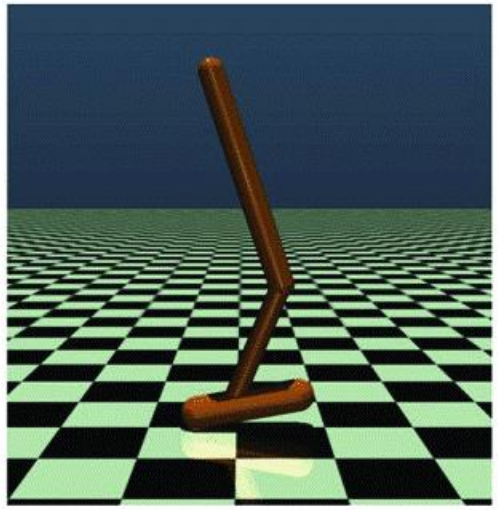
Meta-Representation	Meta-Optimizer		
	Gradient	RL	Evolution
Initial Condition	[16], [79], [88], [102], [166], [166]–[168]	[169]–[171] [16], [63], [64]	[172], [173]
Optimizer	[19], [94] [21], [39], [79], [106], [107], [174]	[81], [93]	
Hyperparam	[17], [69] [71]	[175], [176]	[173] [177]
Feed-Forward model	[38], [45], [86], [110], [178], [179] [180]–[182]	[22], [114], [116]	
Metric	[20], [90], [91]		
Loss/Reward	[42], [95] [127] [124]	[126] [121], [183] [124]	[123] [23] [177]
Architecture	[18] [135]	[26]	[25]
Exploration Policy		[24], [184]–[188]	
Dataset/Environment	[156] [159]	[162]	[163]
Instance Weights	[151], [152], [155]		
Feature/Metric	[20], [90]–[92]		
Data Augmentation/Noise	[145] [119] [189]	[144]	[146]
Modules	[140], [141]		
Annotation Policy	[190], [191]	[192]	

TABLE 1

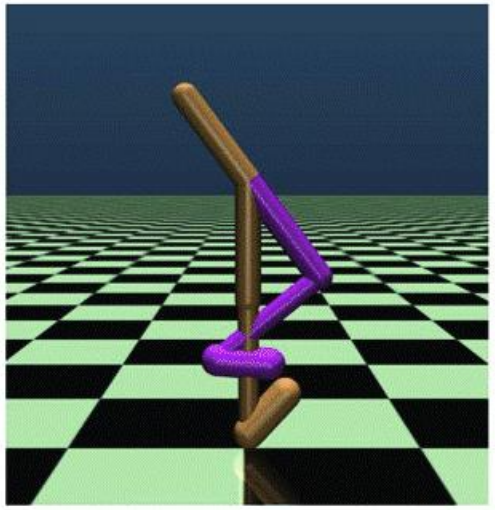
Research papers according to our taxonomy. We use color to indicate salient meta-objective or application goal. We focus on the main goal of each paper for simplicity. The color code is: sample efficiency (red), learning speed (green), asymptotic performance (purple), cross-domain (blue).

*Hospedales, T., Antoniou, A., Micaelli, P. and Storkey, A., 2020. Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439.

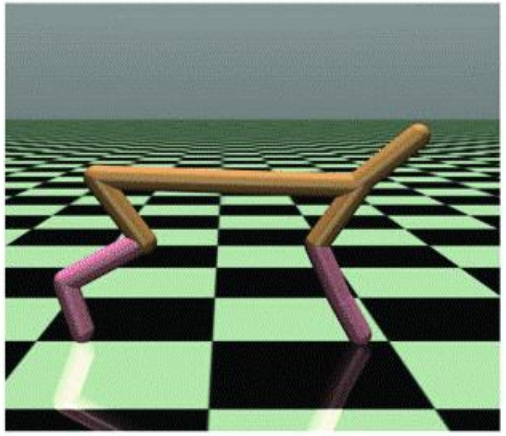
Extra slides



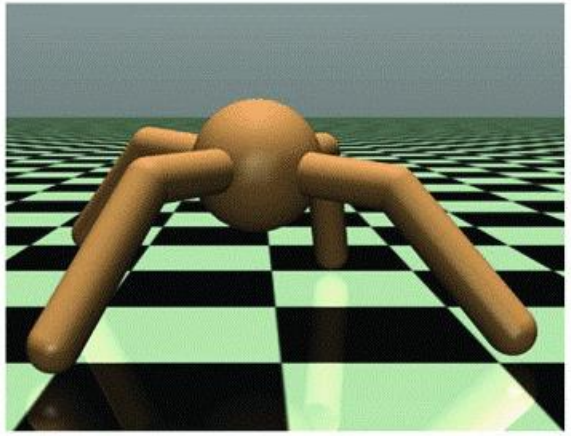
Hopper



Walker2d



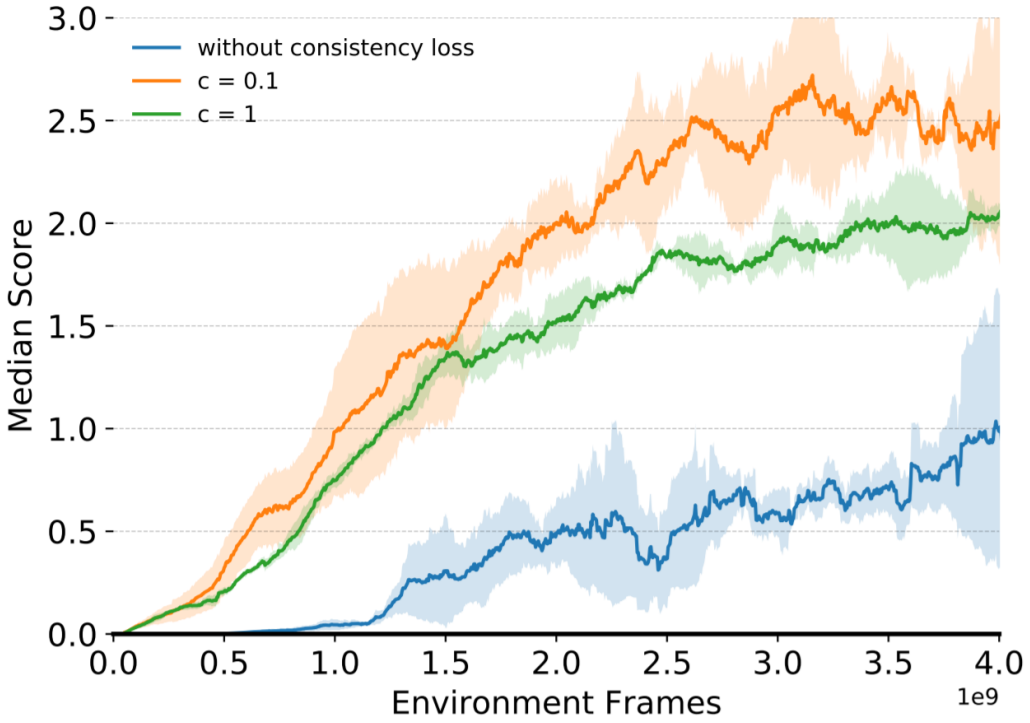
Half-Cheetah



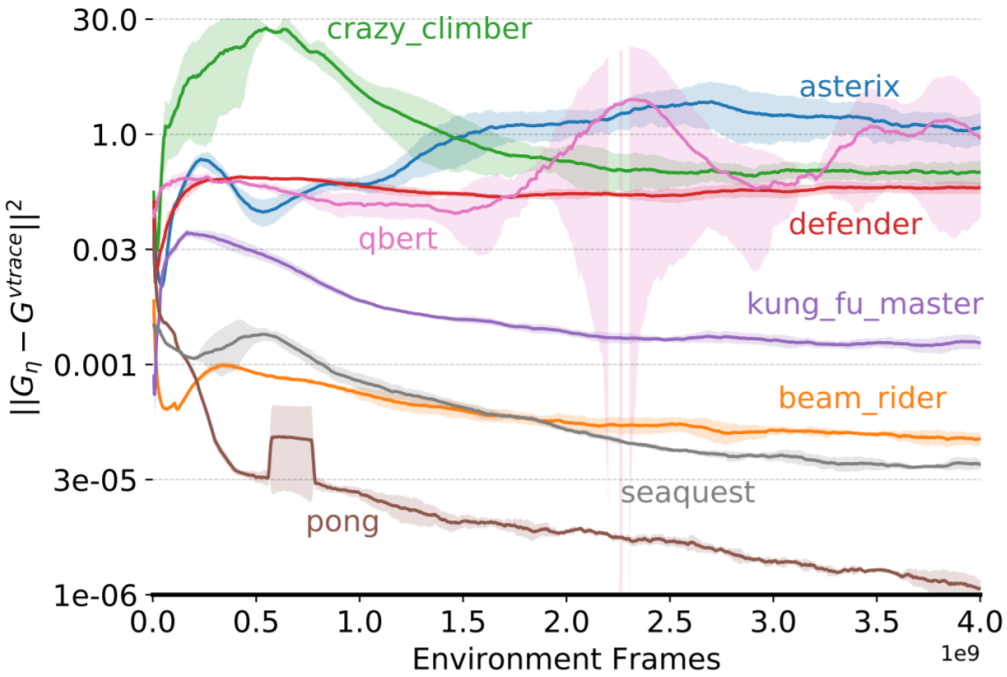
Ant

Humanoid

Extra slides



(c) A parameter study for the value of the consistency weight c. Too high value of this hyperparameter can reduce performance in the long run.



(d) Difference between the vtrace target and the learned update target G_η. We observe large variations throughout training and across games.

Extra slides

Table 1: Methods for discovering RL algorithms.

Algorithm	Discovery	Method	Generality	Train	Test
RL^2 [10, 43]	N/A	∇	Domain-specific	3D maze	Similar 3D maze
EPG [21]	$\hat{\pi}$	ES	Domain-specific	MuJoCo	Similar MuJoCo
ML^3 [7]	$\hat{\pi}$	$\nabla\nabla$	Domain-specific	MuJoCo	Similar MuJoCo
MetaGenRL [23]	$\hat{\pi}$	$\nabla\nabla$	General	MuJoCo	Unseen MuJoCo
LPG	$\hat{\pi}, \hat{y}$	$\nabla\nabla$	General	Toy	Unseen Atari

$\hat{\pi}$: policy update rule, $\hat{y}$: prediction update rule (i.e., semantics of agent's prediction).
∇: gradient descent, $\nabla\nabla$: meta-gradient descent, ES: evolutionary strategy.

Extra slides

Algorithm 1 Meta-Training of Learned Policy Gradient

Input: $p(\mathcal{E})$: Environment distribution, $p(\theta_0)$: Initial agent parameter distribution
Initialise meta-parameters η and hyperparameter sampling distribution $p(\alpha|\mathcal{E})$
Sample batch of environment-agent-hyperparameters $\{\mathcal{E} \sim p(\mathcal{E}), \theta \sim p(\theta_0), \alpha \sim p(\alpha|\mathcal{E})\}_i$
repeat
  for all lifetimes $\{\mathcal{E}, \theta, \alpha\}_i$ **do**
    Update parameters θ using η and α for K times using Eq. (2)
    Compute meta-gradient using Eq. (4)
    if lifetime ended **then**
      Update hyperparameter sampling distribution $p(\alpha|\mathcal{E})$
      Reset lifetime $\mathcal{E} \sim p(\mathcal{E}), \theta \sim p(\theta_0), \alpha \sim p(\alpha|\mathcal{E})$
    end if
  end for
  Update meta-parameters η using the meta-gradients averaged over all lifetimes.
until η converges

Extra slides

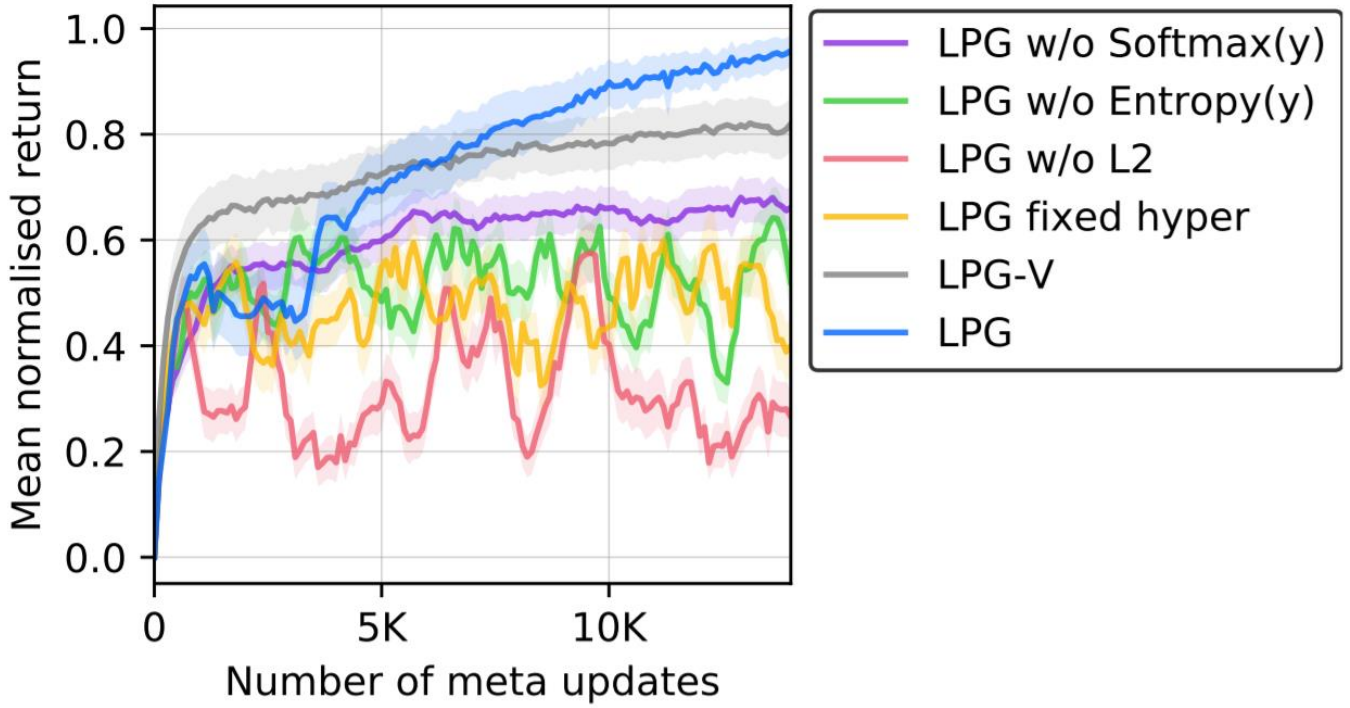
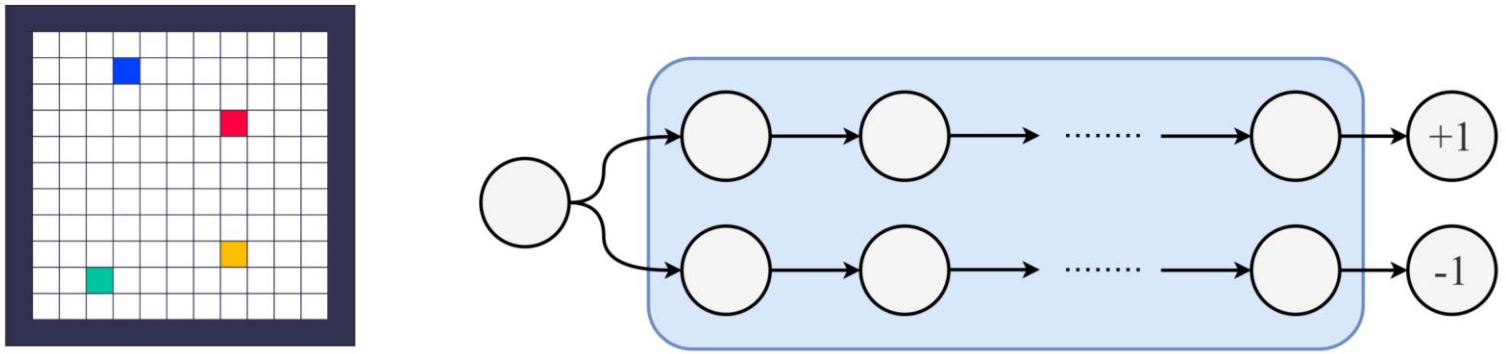


Figure 7: Ablation study. Each curve shows normalised return ($G_{\text{norm}} \in [0, 1]$) averaged over 15 training environments throughout meta-training.

Extra slides



(a) Grid world (b) Delayed chain MDP

Figure 2: Training environments. (a) The agent receives the corresponding rewards by collecting objects. (b) The first action determines the reward at the end of the episode.