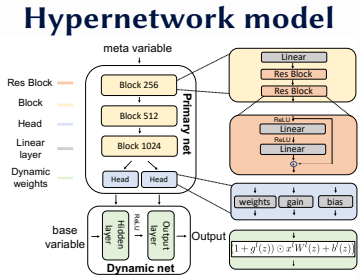
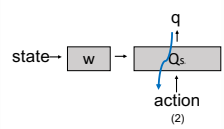
Recomposing the Reinforcement Learning Building Blocks with Hypernetworks

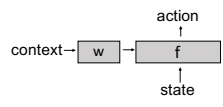
Elad Sarafian*, Shai Keynan* and Sarit Kraus

Bar-Ilan University

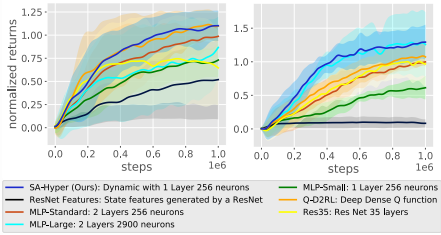


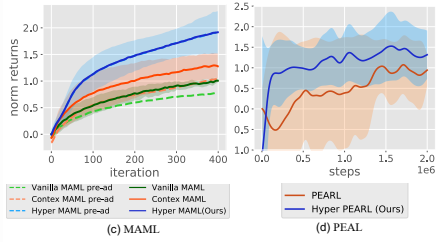


(a) SA-Hyper (Q-net)



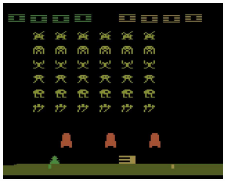
(b) Meta-Policy





Motivation: The role of neural networks in the RL Renaissance

The Reinforcement Learning Renaissance is attributed to the integration with Deep Neural Networks.



(a) DQN (2013)

(b) AlphaZero (2018)

(c) AlphaStar (2019)

So why State-Of-The-Art Deep RL papers still develop their algorithms on old-fashioned neural-network architectures?

Neural-network architecture choices for reinforcement learning remain relatively under-explored (Sinha et al. 2020).

A reminder: the Q-function and the RL objective

Find a policy that "maximizes" the Q-function over its state distribution

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s\sim d^\pi}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s,a)\right]\right] \tag{1}$$

Where

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\middle| a_t\sim\pi(\cdot|s_t), s_0=s, a_0=a\right]. \tag{2}$$

SOTA off-policy RL algorithms estimates the Q-function with a neural model.

- Estimating the Q-function is not the goal of the algorithm, rather maximizing it.
- Its input is elements from the Cartesian product of the state and action domains.

A reminder: optimizing the policy with the Q-function estimation

A stochastic policy with neural-network based parameterization is

$$\pi_\phi(a|s) = \mu_\phi(\varepsilon|s) \text{s.t.} \varepsilon \sim p_\varepsilon, \tag{3}$$

To optimize π_ϕ we apply gradient ascent steps in the action gradient direction of the Q-function

$$\phi \leftarrow \phi + \eta \mathbb{E}_{\left\{ \substack{s \sim \mathcal{D} \\ \varepsilon \sim p_\varepsilon} \right\}} \left[\nabla_\phi \mu_\phi(\varepsilon|s) \nabla_a Q_\theta^\pi(s, \mu_\phi(\varepsilon|s)) \right] \tag{4}$$

To optimize π we need to find the Q-function derivative with respect to the action input.

- We do not directly learn $\nabla_a Q^\pi$.
- Is $\nabla_a Q_\theta^\pi$ a sufficient approximation to $\nabla_a Q^\pi$ when Q^π is approximated with a neural model Q_θ^π ?

Proposition 1 rephrased

Under sufficient conditions, **better gradient approximation allows larger policy optimization step**.

If there exists a gradient estimation $g(s, a)$ and $0 < \alpha < 1$ s.t.

$$\|\overline{\nabla}_\phi \cdot g - \overline{\nabla}_\phi \cdot \nabla_a Q^\pi\| \leq \alpha \|\overline{\nabla}_\phi \cdot \nabla_a Q^\pi\| \tag{5}$$

then the ascent step $\phi' \leftarrow \phi + \eta \overline{\nabla}_\phi \cdot g$ with $\eta \leq \frac{1}{\tilde{k}} \frac{1-\alpha}{(1+\alpha)^2}$ guarantees a positive empirical advantage policy.

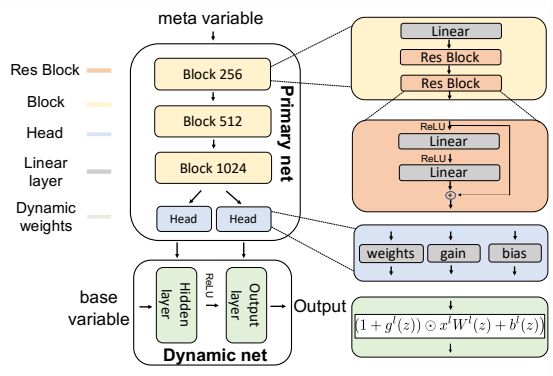
To summarize, we wish to find a neural model for the Q-function s.t.

- It is suited for processing inputs in the form (s, a), where s serves as a context.
- It has good gradient model with respect to the input a.

Recomposing the Q-function with Hypernetworks

Hypernetwork is an architecture designed to process a tuple (x, z) and output a value y.

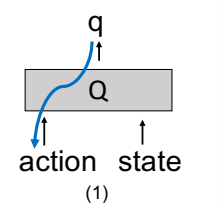
- A *primary* network $w_\theta : Z \to \mathbb{R}^{n_w}$ produces weights.
- A *dynamic* network $f_{w_\theta(z)} : X \to Y$ outputs the prediction.
- z is termed a context or *meta* variable.
- x is termed the *base* variable.

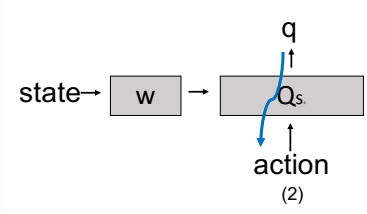


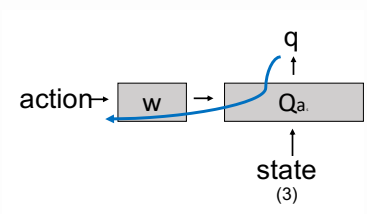
Recomposing the Q-function with Hypernetworks

Consider three alternative models:

1. **MLP network**
   - States and actions are concatenated.
2. **State-Action Hypernetwork (SA-Hyper)**
   - State is a meta-variable.
3. **Action-State Hypernetwork (AS-Hyper)**
   - Action is as meta-variable.







- **MLP**

$$\nabla_a Q_\theta^\pi(s,a) = W^a \Lambda^1(s,a) \left(\prod_{l=2}^{L-1} W^l \Lambda^l(s,a) \right) W^L$$

Function of the state only via the diagonal elements - active neurons map.

- **AS-Hyper**

$$\nabla_a Q_\theta^\pi(s,a) = \nabla_a w(a) \nabla_w f_w(s)$$

$n_w \gg n_a$ leads to a large null-space.

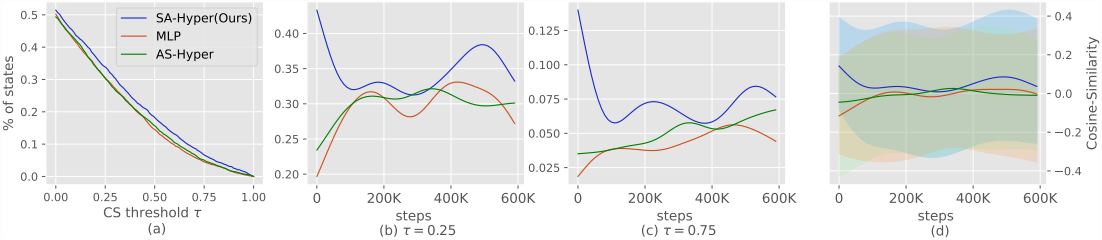
- **SA-Hyper**

$$\nabla_a Q_\theta^\pi(s,a) = W^1(s) \Lambda^1(s,a) \left(\prod_{l=2}^{L-1} W^l(s) \Lambda^l(s,a) \right) W^L(s)$$

$W^l(s)$ enables higher expressiveness of the gradient.

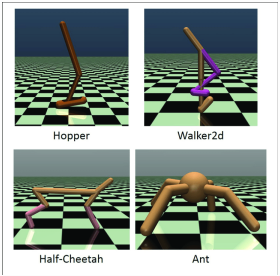
Empirical Gradient Analysis

- Practically, due to the Temporal-Difference (TD) learning bias we cannot hope to reconstruct the true Q-function scale.
- Instead, we use the Cosine Similarity (CS) as a surrogate for measuring the gradient accuracy.
- Empirically low Cosine-Similarity is expected (Ilyas et al., 2019).

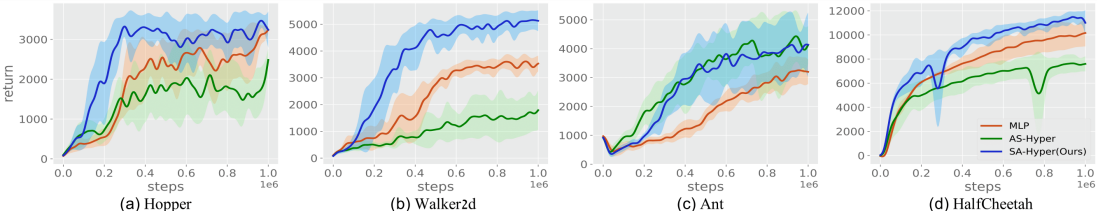


Empirical Results

- **Algorithms**: TD3 and SAC
- **Environments**: Mujoco
- **Implementation**: Authors' official implementations
- **Hyper parameters**: Adjust only the learning rate
- **Evaluation**: 5 different seeds



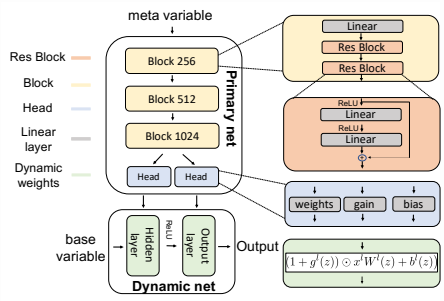
Comparing different Hypernetwork configuration with TD3



(a) Hopper (b) Walker2d (c) Ant (d) HalfCheetah

Empirical Results

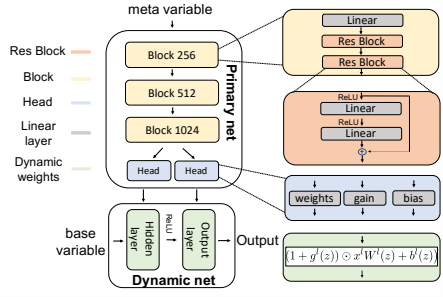
We compare the Hypernetwork model to several baselines:

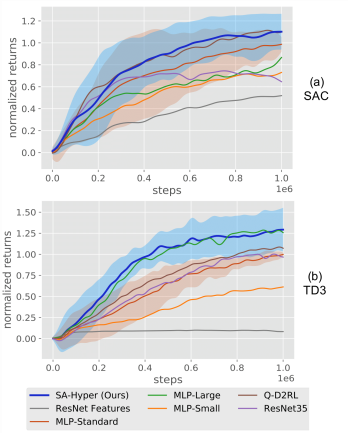


- **primary**: ResNet with 9M parameters.
- **dynamic**: 1 hidden layer, 256 neurons.

- **MLP-Standard**: 2 hidden layers, 256 neurons.
- **MLP-Large**: 2 hidden layers, 2900 neurons (same parameter size $9M$).
- **MLP-Small**: 1 hidden layer, 256 neurons (same dynamic architecture).
- **ResNet-Features**: state features generated by the Primary (same primary and dynamic architectures).
- **Deep-ResNet**: 35 Residual Blocks, half parameter size 2x Slower.
- **Q-D2RL**: Deep Dense model for the Q-function (Sinha et al., 2020).

We compare the Hypernetwork model to several baselines:





- **primary**: ResNet with 9M parameters.
- **dynamic**: 1 hidden layer, 256 neurons.

Background: Meta-RL

- Distribution of different tasks $p(\mathcal{T})$
- Objective: Learning a meta-policy that should generalize over tasks.
- **Methods**:
  - **PEARL** (Rakelly et al., 2019) Learned context $c = q_\nu(z|c^{\mathcal{T}_i})$ based on history $\pi(a|s,c)$

$$\mathcal{L}^{critic}_{pearl}(\theta,\nu) = \mathbb{E}_{\mathcal{T}}\left[\mathbb{E}_{q_\nu(z|c^{\mathcal{T}_i})}\left[\mathcal{L}^{critic}_{sac}(\theta,\nu) + D_{KL}\left(q_\nu(z|c^{\mathcal{T}_i})|p(z)\right)\right]\right] \tag{6}$$

  - **MAML** (Finn et al., 2017) Policy adjusts its weights, no explicit context

$$\nabla_\phi J_{maml}(\phi) = \mathbb{E}_{\left\{\substack{\mathcal{T}_i\sim p(\mathcal{T})\\ \pi_{\phi_i}}\right\}}\left[\sum_{t=0}^{\infty}\hat{A}_{i,t}\nabla_\phi \log \pi_{\phi_i}(a_t|s_t)\right] \tag{7}$$

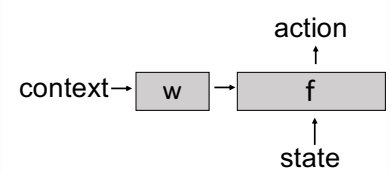
    ϕ_i is the adjusted weights for task i.
  - In practice, often we have access to an **Oracle-Context** that specify the task.

Meta-Policy

- Context as a meta-variable

$$\pi_\phi(a|s,c) = \pi_{w(c;\phi)}(a|s) \qquad (8)$$

- The primary generates a different policy for each task.

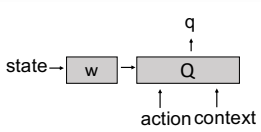


Meta-Q-function

- A context is added also to the Q-function's model

$$Q^\pi(s,a,c) \simeq Q^\pi_{w(s;\theta)}(a,c) \qquad (9)$$

- The **context gradient** backpropagates through the dynamic network.



Reducing the Gradient Variance

- MAML objective with MLP

$$\nabla_\phi J(\phi) = \sum_{\mathcal{T}_i} \sum_{s \in \mathcal{T}_i} \hat{A}_{i,s} \frac{\nabla_\phi \pi_\phi(s, c_i)}{\pi_\phi(s, c_i)}$$

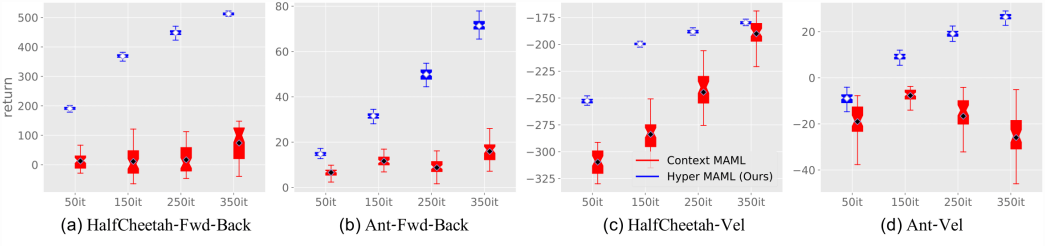
- MAML objective with Hypernetwork

$$\nabla_\phi J(\phi) = \sum_{\mathcal{T}_i} \nabla_\phi w(c_i) \cdot \sum_{s \in \mathcal{T}_i} \hat{A}_{i,s} \frac{\nabla_w \pi_{w(c_i)}(s)}{\pi_{w(c_i)}(s)}$$

- The state-dependent part of the gradients $\nabla_w \pi_{w(c_i)}(s)$ is averaged over the task distribution for each task.
- The task dependent gradients of the primary weights $\nabla_\phi w(c_i)$ are averaged over the task distribution.
- The disentanglement reduces the gradient noise and should translate to more accurate learning steps.

Reducing the Gradient Variance

- We trained meta-policies based on context-MAML:
  1. **MLP**
  2. **Hypernetwork**
- We estimated the gradient noise by constructing 50 different uncorrelated gradients and evaluating the updated policy's performance.

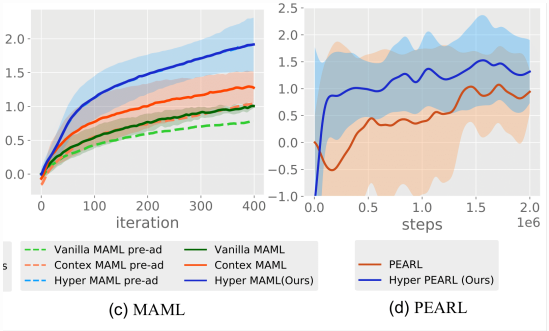


(a) HalfCheetah-Fwd-Back (b) Ant-Fwd-Back (c) HalfCheetah-Vel (d) Ant-Vel

- The variance of the Hypernetwork model is significantly lower than the MLP model across all tasks and environments

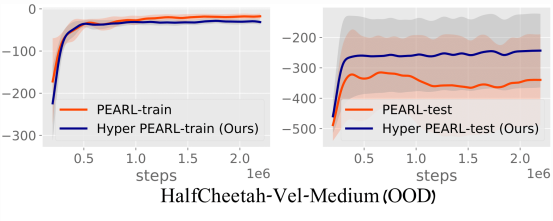
Empirical Results

- **Algorithms**: MAML, PEARL
- **Hyper parameters**:
  Adjust only the learning rate
- **Implementation**:
  Authors' official implementations
- **Evaluation**: 5 different seeds
- **Tasks**:
  - Reach a distant point
  - Target velocity
  - OOD- Out of Distribution

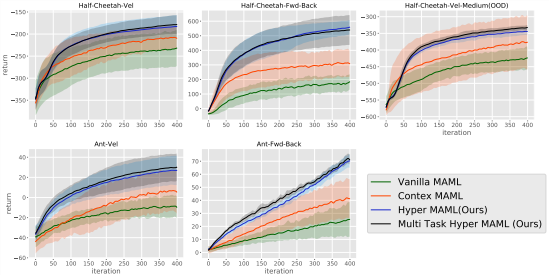


(c) MAML (d) PEARL

Out-Of-Distribution Generalization



Eliminating the adaptation step during training



Conclusions

- The unique nature of the RL setting requires unconventional models.
  - Unlike supervised tasks, we care only about the gradient accuracy of our Q-function model.
  - The network needs to model the inherent state-action interaction and context-state interaction in Meta-RL.
- Advantages over MLP models
  - Better estimation of the Q-function **gradient** which is required to train actor-critic algorithms.
  - Reduces the **gradient variance** in Meta-RL.
  - **Outperform** in final reward and efficiency.

Thank You

Code:
https://github.com/keynans/HypeRL