Table 1: A comparison between fixed reinforcement learning algorithms (REINFORCE), backpropagation-based meta RL (MAML, MetaGenRL, LPG), black-box (MetaRNN), and our black-box method with symmetries (SymLA). $\pi_\theta^{(s)}$ denotes a *stationary* policy that is updated at fixed intervals by backpropagation.

	REINFORCE	MetaGenRL / LPG	MAML	MetaRNN	SymLA (ours)
Meta variables	/	ϕ	Initial θ_0	θ	θ
Learned variables	θ	θ	θ	RNN state h	RNN states $h_{ab}^{(k)}$
Learning algorithm	fixed loss func L + Backprop	learned loss func L_ϕ + Backprop	fixed loss func L + Backprop	π_θ	π_θ
Policy	$\pi_\theta^{(s)}$	$\pi_\theta^{(s)}$	$\pi_\theta^{(s)}$	π_θ	π_θ
Black box	✗	✗	✗	✓	✓
Symmetries in learning algorithm	✓	✓	(✓)	✗	✓

A Related Work

Learning to reinforcement learn can be implemented with varying degrees of inductive biases.

Black-Box Meta RL Black-box meta RL can be implemented by policies that receive the reward signal as input [Schmidhuber, 1992b] and use memory to learn, such as recurrence in RNNs [Hochreiter et al., 2001, Wang et al., 2016, Duan et al., 2016]. These approaches do not feature the symmetries discussed in this paper which leads to a tendency of overfitting.

Learned Learning Rules & Fast Weights In the supervised and reinforcement learning contexts, learned learning rules [Bengio et al., 1992] or fast weights [Schmidhuber, 1992a, 1993, Miconi et al., 2018, Schlag et al., 2020, Najarro and Risi, 2020] describe (meta-)learned mechanisms (slow weights) that update fast weights to implement learning. This often involves outer-products and can be generalised to black-box meta learning with parameter sharing [Kirsch and Schmidhuber, 2020]. None of these approaches feature all of the symmetries we discuss above and are applicable to RL.

Backpropagation-based Meta RL Alternatives to black-box meta RL include learning a weight initialization and adapting it with a human-engineered RL algorithm [Finn et al., 2017], warping computed gradients [Flennerhag et al., 2019], meta-learning hyper-parameters [Sutton, 1992, Xu et al., 2018] or meta-learning objective functions corresponding to the learning algorithm [Houthooft et al., 2018, Bechtle et al., 2021, Kirsch et al., 2019, Xu et al., 2020, Oh et al., 2020].

Neural Network Symmetries Symmetries in neural networks have mainly been investigated to reflect the structure of the input data. This includes applications of convolutions [Fukushima, 1979], deep sets [Zaheer et al., 2017], graph neural networks [Wu et al., 2020], and geometric deep learning [Bronstein et al., 2017]. While many meta learning algorithms exhibit symmetries [Bengio et al., 1992], in particular backpropagation-based meta learning [Andrychowicz et al., 2016, Finn et al., 2017, Flennerhag et al., 2019, Kirsch et al., 2019], the effects of these symmetries have not been discussed in detail. In this work, we provide such a discussion and experimental investigation in the context of meta RL.

B Bandits from Wang et al. [2016]

In our experiments, we use bandits of varying difficulty from Wang et al. [2016]. Let p_1 be the probability of the first arm for a payout of $r = 1$, $r = 0$ otherwise, and p_2 the payout for the second arm. Then, we define the

- uniform independent bandit with $p_1 \sim U[0, 1]$ and $p_2 \sim U[0, 1]$,
- uniform dependent bandit with $p_1 \sim U[0, 1]$ and $p_2 = 1 - p_1$,
- easy dependent bandit with $p_1 \sim Cat[0.1, 0.9]$ and $p_2 = 1 - p_1$,
- medium dependent bandit with $p_1 \sim Cat[0.25, 0.75]$ and $p_2 = 1 - p_1$,
- hard dependent bandit with $p_1 \sim Cat[0.4, 0.6]$ and $p_2 = 1 - p_1$.

Algorithm 1 SymLA meta training

Require: Distribution over RL environment(s) $p(e)$
$\theta \leftarrow$ initialize LSTM parameters
while meta loss has not converged **do** ▷ Outer loop in parallel over envs $e \sim p(e)$ and samples $\phi \sim \mathbb{N}(\phi|\theta, \Sigma)$
$\{h_{ab}\} \leftarrow$ initialize LSTM states $\forall a, b$
$o_1 \sim p(o_1)$ ▷ Initialize environment e
for $t \in \{1, \dots, L\}$ **do** ▷ Inner loop over lifetime in environment e
$h_{ab} \leftarrow f_{\text{LSTM}}(h_{ab}, o_{t,a}, a_{t-1,b}, r_{t-1}, \overrightarrow{m}_b, \overleftarrow{m}_a) \quad \forall a, b$ ▷ Equation 8
$\overrightarrow{m}_b \leftarrow \sum_a f_{\overrightarrow{m}}(h_{ab}) \quad \forall b$ ▷ Create forward messages
$\overleftarrow{m}_a \leftarrow \sum_b f_{\overleftarrow{m}}(h_{ab}) \quad \forall a$ ▷ Create backward messages
$y \leftarrow \overrightarrow{m}_{\cdot 1}$ ▷ Read out action
$a_t \sim p(a_t; y)$ ▷ Sample action from distribution parameterized by y
Send action a_t to environment e, observe o_{t+1} and r_t
$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_{\phi \sim \mathbb{N}(\phi|\theta,\Sigma)}[\mathbb{E}_{e \sim p(e)}[\sum_{t=1}^{L} r_t^{(e)}(\phi)]]$ ▷ Update θ using evolution strategies (Equation 9)

C Hyper-parameters

C.1 SymLA Architecture

We use a single recurrent layer, $K = 1$, with a message size of $\overleftarrow{M} = 8$ and $\overrightarrow{M} = 8$. To produce the next state h_{ab} according to Equation 8, we use parameter-shared LSTMs with a hidden size of $N = 16$ ($N = 64$ for bandits to match Wang et al. [2016]) and run the recurrent cell for 2 micro ticks.

C.2 Meta Learning / Outer Loop

We estimate gradients ∇_θ using evolutionary strategies [Salimans et al., 2017] with 10 evaluations per population sample to estimate the fitness value (100 evaluations for bandits). Then, we apply those using Adam with a learning rate of $\alpha = 0.01$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ ($\alpha = 0.2$ for bandits). We use a fixed noise standard deviation of $\sigma = 0.035$ ($\sigma = 0.2$ for bandits) and a population size of 512. Our inner loop has a length of $L = 500$ ($L = 100$ for bandits), concatenating multiple episodes. We meta-optimize for $4,000$ outer steps for bandit experiments, and $20,000$ otherwise.

C.3 Generalisation to Unseen Environments

We apply a random linear transformation (Glorot normal) to environment observations, mapping those to a 16-dimensional vector.