Transferring Knowledge across Learning Processes

Sebastian Flennerhag[*]
Alan Turing Institute
sebastianflennerhag@gmail.com

Andreas Damianou
Amazon, Cambridge, UK
damianou@amazon.com

Pablo Garcia Moreno
Amazon, Cambridge, UK
morepabl@amazon.com

Neil Lawrence
Amazon, Cambridge, UK
lawrennd@amazon.com

1 Introduction

In complex transfer learning scenarios new tasks might not be tightly linked to previous tasks. Approaches that transfer information contained only in the final parameters of a source model will therefore struggle. Instead, transfer learning at a higher level of abstraction is needed. We propose Leap, a framework that achieves this by transferring knowledge across learning processes. We associate each task with a manifold on which the training process travels from initialization to final parameters and construct a meta learning objective that minimizes the expected length of this path. Our framework leverages only information obtained during training and can be computed on the fly at negligible cost. We demonstrate that our framework outperforms competing methods, both in meta learning and transfer learning, on a set of computer vision tasks. Finally, we demonstrate that Leap can transfer knowledge across learning processes in demanding Reinforcement learning environments (Atari) that involve millions of gradient steps.

2 Transferring Knowledge over Learning Processes

Our framework is based on the idea that transfer learning can be achieved by leveraging information contained across similar learning processes. Exploiting that this information is encoded in the geometry of the loss surface, we leverage geometrical quantities to facilitate the learning process with respect to new tasks. Consider a learning objective f that maps a parameterization $\theta \in \mathbb{R}^n$ to a scalar loss value given input $x \in \mathbb{R}^m$ and target $y \in \mathbb{R}^c$. We then have the gradient descent update as:

$$\theta^{i+1} = \theta^i - \alpha^i S^i \nabla f(\theta^i). \tag{1}$$

We assume this process converges to some stationary point $\theta^\star$ after K gradient steps. Because the learning process in eq. 1 follows the gradient trajectory, it constantly provides information about the geometry of the loss surface. Gradients that largely point in the same direction indicate a convex loss surface, whereas gradients with frequently opposing directions indicate an ill-conditioned loss landscape, something we would like to avoid. Leveraging this insight, we propose a framework for transfer learning that exploits the accumulation of geometric information by constructing a meta objective that minimizes the expected length of the gradient descent path across tasks. In doing so, the meta objective intrinsically balances local geometries across tasks and encourages an initialization as close to task-specific minima as possible, taking the gradient trajectory into account.

The gradient descent update can be seen as the discrete approximation of the gradient flow $\dot{\theta}(t) = -\alpha(t)S(t)\nabla f(\theta(t))$, where we use $\theta(t)$ to differentiate discrete and continuous domains. The

[*]Work done while this author was in Amazon.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

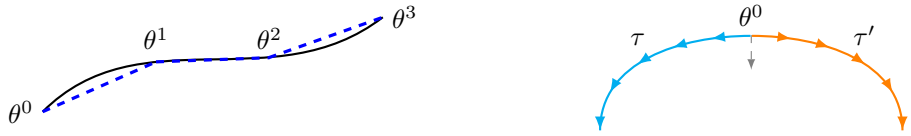

Figure 1: *Left:* Illustration of the discrete approximation of the gradient flow. Curvature and step-size determine the fidelity of the approximation. *Right:* Illustration of the meta objective (eq. 3). Gradient paths from different tasks exert influence on θ^0 by pulling it forward along their respective path. The learned initialization finds an equilibrium balance between these competing forces.

gradient flow provides information about movements in parameter space, but does not directly account for performance. To avoid information loss, we embed the gradient flow in a Riemann manifold M defined by the coordinate system $(\theta, f(\theta))$, i.e. the loss surface. Following the gradient flow on M from an initialization θ^0 yields a *curve* $\gamma : [0, 1] \mapsto M$. Minimizing the length of γ can be approximated by the minimization of the chordal distance (fig. 1; Ahlberg et al., 1967) (see Appendix A for details). Specifically, we define the length of the observed gradient path as:

$$d(\theta^0) = \sum_{i=0}^{K-1} \|\gamma^{i+1} - \gamma^i\|^2 = \sum_{i=0}^{K-1} \|\theta^{i+1} - \theta^i\|^2 + \|f(\theta^{i+1}) - f(\theta^i)\|^2. \tag{2}$$

Note that d involves only terms encountered during the training phase. We exploit this later to construct the gradient using only terms encountered during the training phase, enabling us to perform gradient descent on the meta objective at negligible cost (eq. 8 in Appendix).

We now turn to the transfer learning setting, where we face a set of tasks, each with a distinct loss surface. We can transfer knowledge across these learning process, and in particular information about the local geometry, by aggregating information obtained along the gradient path. This allows us to choose an initialization such that loss surfaces are well behaved and gradient descent converges rapidly in expectation. Formally, we define a task $\tau = (f_\tau, p_\tau, u_\tau)$ as the process of learning to approximate the relationship $x \to y$ through samples from the data distribution $p_\tau(x, y)$. This process is defined by the gradient update rule u_τ (as defined in eq. 1), applied iteratively to minimize the task objective f_τ. For a given task τ, the relevant initializations to choose among are those that converge to stationary points $\theta^\star$ with approximately optimal performance, i.e. an upper bound of the form $f(\theta^\star) \leq \min_{\theta'} f(\theta') + \epsilon$ for some small $\epsilon > 0$. Since this must hold for all tasks, the set of feasible initializations are those that satisfy this performance constraint across tasks defined by $\Theta = \cap_\tau \{\theta^0 \mid f_\tau(\theta^\star_\tau) \leq \min_{\theta'} f_\tau(\theta') + \epsilon\}$. Given a distribution $p(\tau)$ over tasks, minimizing $\mathbb{E}_{\tau \sim p(\tau)} \left[d(\theta^0) \right]$ subject to $\theta^0 \in \Theta$ defines our meta objective,

$$\begin{aligned} \min_{\theta^0 \in \Theta} \quad & \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{K_\tau - 1} \|\theta_\tau^{i+1} - \theta_\tau^i\|^2 + \|f_\tau(\theta_\tau^{i+1}) - f_\tau(\theta_\tau^i)\|^2 \right], \\ \text{s.t.} \quad & \theta_\tau^{i+1} = u_\tau(\theta_\tau^i), \qquad \theta_\tau^0 = \theta^0. \end{aligned} \tag{3}$$

Crucially, this meta objective is robust to variations in the geometry of loss surfaces, as it naturaly balances complementary and competing learning processes (fig. 1). For instance, there may be an initialization that can solve a small subset of tasks in a handful of gradient steps, but would be catastrophic for a larger set of tasks. When transferring knowledge via the initialization, this means we must trade off commonalities and differences between gradient paths.

2.1 The Pull-Forward Algorithm

In practice, backpropagating through eq. 3 is prohibitively costly. Moreover, Θ is unknown and extremely hard to model. However, because the relevant upper bound is the performance we can obtain without using prior information, i.e. with a random initialization, we can construct a set of baseline gradient paths and improve the initialization with respect to these. Repeating the process, we obtain an algorithm that is guaranteed to solve eq. 3. We propose the *Pull-Forward* objective (algorithm 1), which is guaranteed to find a solution to eq. 3. We take a random initialization, shared across all

Algorithm 1 Leap: Transferring Knowledge over Learning Processes

Require: $p(\tau)$, $\tau = (f_\tau, u_\tau, p_\tau)$: distribution over tasks
Require: β: step size
1: randomly initialize ψ^0
2: **while** not done **do**
3: $\quad \nabla F \leftarrow 0$: initialize meta gradient
4: $\quad$ sample task batch $\mathcal{B}$ from $p(\tau)$
5: $\quad$ **for all** $\tau \in \mathcal{B}$ **do**
6: $\quad\quad \psi_\tau^0 \leftarrow \psi^0$: initialize baselines
7: $\quad\quad$ **for all** $i \in \{0, \dots, K_\tau - 1\}$ **do**
8: $\quad\quad\quad$ compute $f_\tau(\psi_\tau^i)$ and ψ_τ^{i+1}
9: $\quad\quad\quad$ increment ∇F as in eq. 8 of appendix C
10: $\quad\quad$ **end for**
11: $\quad$ **end for**
12: $\quad \psi^0 \leftarrow \psi^0 - \frac{\beta}{|\mathcal{B}|} \nabla F$: update initialization
13: **end while**

tasks, as our starting point and assume that this initialization is in Θ. From this initial point (line 1), sample a set of tasks from $p(\tau)$ and obtain baseline gradient paths for each, $\Psi = \{\Psi_\tau\}_\tau$, where $\Psi_\tau = \{\psi_\tau^i\}_{i=0}^{K_\tau}$ is the baseline gradient path for task τ. Note that all tasks share the same initial point, $\psi_\tau^0 = \psi^0 \in \Theta$. These baselines correspond to task-specific learning processes (line 8) that we use to freeze the forward point θ_τ^{i+1} in all norm terms in eq. 3. For instance, $\|\theta_\tau^{i+1} - \theta_\tau^i\|^2$ becomes $\|\psi_\tau^{i+1} - \theta_\tau^i\|^2$. As such, optimizing θ^0 with respect to Ψ pulls the initialization forward along each task-specific gradient path, reducing the gradient path while being anchored to baseline gradient paths that converge to good minima. To summarize, we define the Pull-Forward objective as

$$\begin{aligned} \min_{\theta^0} \quad & F(\theta^0; \Psi) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{K_\tau - 1} \|\psi_\tau^{i+1} - \theta_\tau^i\|^2 + \|f_\tau(\psi_\tau^{i+1}) - f_\tau(\theta_\tau^i)\|^2 \right], \\ \text{s.t.} \quad & \theta_\tau^{i+1} = u_\tau(\theta_\tau^i), \qquad \theta_\tau^0 = \theta^0. \end{aligned} \tag{4}$$

Intuitively, because $F(\psi^0, \Psi) = \mathbb{E}_{\tau \sim p(\tau)} \big[d(\theta^0) \big]$, the pull-forward objective can be understood as a greedy search procedure that incrementally improves upon the observed gradient path. This improved initialization can then be used to obtain a new and more demanding baseline to further improve the initialization. Iterating this process yields a sequence of candidate solutions, all in Θ, with incrementally shorter gradient paths. In theorem 1, we show that the limit point of this sequence is a local minimum of 3.

Theorem 1 (Pull-forward)**.** *Define a sequence of initializations* $\{\psi_s^0\}_{s \in \mathbb{N}}$ *by* $\psi_{s+1}^0 = \psi_s^0 - \beta_s \nabla F(\psi_s^0; \Psi_s), \quad \psi_0^0 \in \Theta$. *For* $\beta_s > 0$ *sufficiently small, there exist learning rates schedules* $\{\alpha_\tau^i\}_{i=1}^{K_\tau}$ *for all tasks such that* $\psi_{k \to \infty}^0$ *is a local minimum of eq. 3.*

Proof; see appendix B. Crucially, when F is evaluated at ψ^0, the meta gradient ∇F can be computed analytically using *only* terms already computed during the task training. As such, the Pull-Forward algorithm can be computed on the fly during training at negligible cost.

3 Empirical Results

We consider three experiments designed to provide increasingly complex transfer learning environments. We measure transfer learning in terms of final test error and *area under the error curve*, defined as the error score on the training set during training.

Omniglot: We transfer knowledge between alphabets in Omniglot (Lake et al., 2015). We compare the rate of convergence as we vary the number of pretraining tasks from 1 to 30, holding out 10 tasks for evaluation. We compare against no pretraining, fine-tuning in single-headed mode, fine-tuning in multi-headed mode, first-order approximation MAML and Reptile. See appendix D for details. For

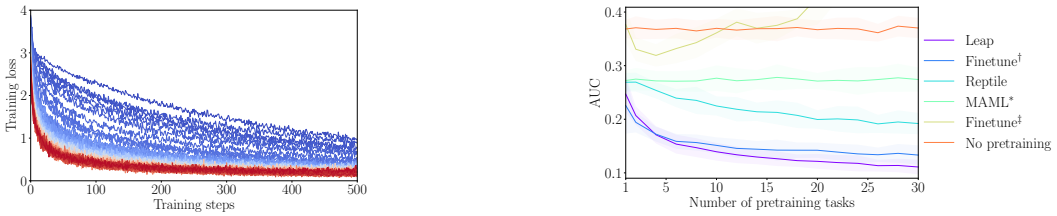

Figure 2: Transfer learning on Omniglot. *Left:* Evolution of training curves during meta training of Leap. Blue signifies beginning of training initial and red end of training. *Right:* AUC under error curve across number of pretraining tasks. *Author's suggested first-order approximation; †multi-headed finetuning; ‡single-headed finetuning. Shading: standard deviation across 10 seeds.

more than 4 tasks, Leap consistently achieves superior performance (fig. 2). The gap between Leap and other meta learning frameworks is large and Leap is in fact the only meta learner to outperform finetuning. Multi-task finetuning is a tough benchmark to beat as tasks are very similar, but with a sufficiently rich task distribution Leap is superior, and importantly, the performance margin grows with the number of pretraining tasks.

Multi-CV: Inspired by Serrà et al. (2018), we consider computer vision datasets as tasks. We pretrain on all but one task, which is held out for final evaluation. For details and results see appendix E. This experiment is more challenging both due to greater task diversity and greater complexity among tasks. We report results on held-out tasks in table 2. Leap outperforms both baselines on all but one transfer learning task (Facescrub), where any form of pretraining results in worse performance.

Atari: To demonstrate that Leap can scale to large problems, we apply it in a reinforcement learning domain, specifically Atari 2600 games (Bellemare et al., 2013). We use a variant of the actor-critic architecture for all experiments (Sutton et al., 1998); specifically we learn both the policy π and the value function V from raw pixels, with π and V sharing a convolutional encoder (Schwarz et al., 2018). We apply Leap with respect to this convolutional encoder. During meta training, we sample mini-batches from 27 games in the suite that has an action space dimensionality of at most 10, holding out three of these games for evaluation, along with remaining games. We train on each task in the mini-batch for a million training steps, accumulating the meta gradient as in algorithm 1. See appendix F for details. Already after 100 meta training steps there is a distinct improvement with respect to the initialization. On games in the pretraining set, Leap learns a useful policy significantly faster (see comparison on Pong in fig. 3). On held-out games with equivalent state spaces (Breakout), Leap learns faster than a random initialization. Notably, in the case of SpaceInvader it reaches a level unattainable by our baseline. Finally on games with almost twice as large an action space as any game seen during meta training, Leap is still able to guarantee performance on par with the baseline, and in fact outperforms it in most cases. In the case of one of the evaluated environments (KungFuMaster), Leap is reaches a significantly higher performance level.

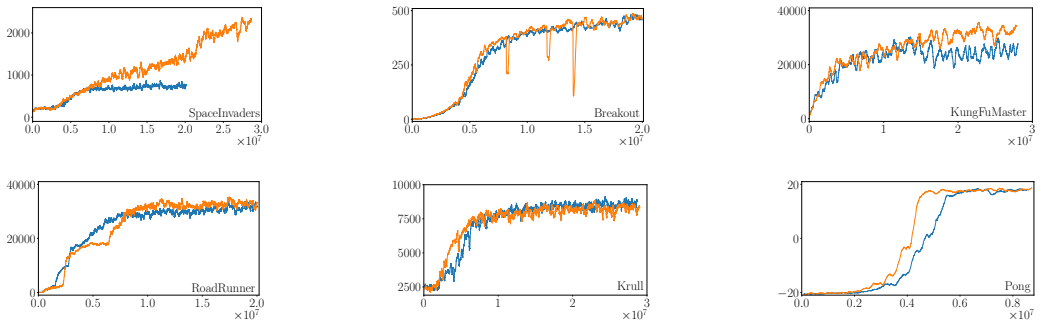

Figure 3: Average episode returns on Atari games over training steps. Scores are reported as moving average over 100 episodes. Except for Pong, all games are held out during meta training. Leap (orange) performs at least on par with a random initialization (blue), and outperforms the baseline in the majority of cases.

References

J Harold Ahlberg, Edwin Norman Nilson, and Joseph Leonard Walsh. *The Theory of Splines and Their Applications*. Academic Press, 1967. p. 51.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv.org*, March 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL http://science.sciencemag.org/content/350/6266/1332.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv.org*, March 2018.

Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370*, 2018.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT Press, Cambridge, 1998.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. *arXiv.org*, June 2016.

Appendix

A Geometries on learning manifolds

The gradient descent update can be seen as the discrete approximation of the gradient flow $\dot{\theta}(t) = -\alpha(t)S(t)\nabla f(\theta(t))$, where we use $\theta(t)$ to differentiate discrete and continuous domains. The gradient flow provides information about movements in parameter space, but does not directly account for performance. To avoid information loss, we embed the gradient flow in a Riemann manifold M defined by the coordinate system $(\theta, f(\theta))$, i.e. the loss surface (**??**). On this manifold, following the gradient flow from an initialization θ^0 yields a *curve* $\gamma : [0, 1] \mapsto M$ defined by:

$$\dot{\gamma}(t) = \big(\dot{\theta}(t), \dot{f}(t)\big), \qquad \gamma(0) = \big(\theta^0, f(\theta^0)\big), \qquad \gamma(1) = \big(\theta^\star, f(\theta^\star)\big), \tag{5}$$

where $\dot{f}(t) = \dot{\theta}(t)^T \nabla f(\theta(t))$. We refer to γ as the *gradient path* from θ^0 to $\theta^\star$ on M. The length or energy of the gradient path is defined by accumulating infinitesimal changes along its trajectory,

$$\text{Length}(\gamma) = \int_0^1 \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle}\, dt, \qquad \text{Energy}(\gamma) = \int_0^1 \langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle\, dt. \tag{6}$$

Analogously to how the gradient update rule approximates the gradient flow, these measures can be approximated by the cumulative chordal distance (fig. 1; Ahlberg et al., 1967). Focusing on the energy functional,[2] we define the length of the observed gradient path as:

$$d(\theta^0) = \sum_{i=0}^{K-1} \|\gamma^{i+1} - \gamma^i\|^2 = \sum_{i=0}^{K-1} \|\theta^{i+1} - \theta^i\|^2 + \|f(\theta^{i+1}) - f(\theta^i)\|^2. \tag{7}$$

B Proof of theorem 1

Proof. We prove the theorem by showing that $F(\psi^0_{s+1}; \Psi_{s+1}) < F(\psi^0_s; \Psi_s)$, with $\psi^0_{s+1} \in \Theta$ for all s. To simplify notation, we define

$$\begin{aligned} z^i_\tau &= (\psi^{s,i}_\tau, f_\tau(\psi^{s,i}_\tau)), & x^i_\tau &= (\psi^{s+1,i}_\tau, f_\tau(\psi^{s+1,i}_\tau)), \\ h^i_\tau &= (\psi^{s,i+1}_\tau, f_\tau(\psi^{s,i+1}_\tau)), & y^i_\tau &= (\psi^{s+1,i+1}_\tau, f_\tau(\psi^{s+1,i+1}_\tau)). \end{aligned}$$

By assumption, β_s is sufficiently small to satisfy $F(\psi_{s+1}; \Psi_s) \leq F(\psi_s; \Psi_s)$, from which we have

$$\begin{aligned} &\mathbb{E}_{\tau,i} \|h^i_\tau - x^i_\tau\|^2 \leq \mathbb{E}_{\tau,i} \|h^i_\tau - z^i_\tau\|^2 \\ \implies &\mathbb{E}_{\tau,i} \|z^i_\tau\|^2 \geq \mathbb{E}_{\tau,i} \|x^i_\tau\|^2 + 2\langle h^i_\tau, z^i_\tau \rangle - 2\langle h^i_\tau, x^i_\tau \rangle. \end{aligned}$$

where we write $\mathbb{E}_{\tau,i} = \mathbb{E}_{\tau \sim p(\tau)} \sum_{i=1}^{K_\tau}$ as shorthand. Applying this to $\|y^i_\tau - z^i_\tau\|^2$ yields

$$\begin{aligned} \mathbb{E}_{\tau,i} \|y^i_\tau - z^i_\tau\|^2 &= \mathbb{E}_{\tau,i} \|y^i_\tau\|^2 - 2\langle y^i_\tau, z^i_\tau \rangle + \|z^i_\tau\|^2 \\ &\geq \mathbb{E}_{\tau,i} \|y^i_\tau\|^2 - 2\langle y^i_\tau, z^i_\tau \rangle + \|x^i_\tau\|^2 + 2\langle h^i_\tau, z^i_\tau \rangle - 2\langle h^i_\tau, x^i_\tau \rangle \\ &= \mathbb{E}_{\tau,i} \|y^i_\tau - x^i_\tau\|^2 + 2\langle y^i_\tau, x^i_\tau \rangle - 2\langle y^i_\tau, z^i_\tau \rangle + 2\langle h^i_\tau, z^i_\tau \rangle - 2\langle h^i_\tau, x^i_\tau \rangle \\ &= \mathbb{E}_{\tau,i} \|y^i_\tau - x^i_\tau\|^2 + 2\langle y^i_\tau - h^i_\tau, x^i_\tau - z^i_\tau \rangle. \end{aligned}$$

It remains to show that the expectation over $\langle y^i_\tau - h^i_\tau, x^i_\tau - z^i_\tau \rangle$ is positive for certain choices of learning rates α^i_τ. Define $\hat{z}^i_\tau = \psi^{s,i}$ and similarly for $\hat{x}^i_\tau, \hat{h}^i_\tau$ and $\hat{y}^i_\tau$, so that $\langle y^i_\tau - h^i_\tau, x^i_\tau - z^i_\tau \rangle =$

[2] The energy functional is a more stable optimization objective, while every minimizer of the energy functional also minimizes the length.

$\langle \hat{y}_\tau^i - \hat{h}_\tau^i, \hat{x}_\tau^i - \hat{z}_\tau^i \rangle + c_\tau^i$, where $c_\tau^i = (f_\tau(\hat{y}_\tau^i) - f_\tau(\hat{h}_\tau^i))(f_\tau(\hat{x}_\tau^i) - f_\tau(\hat{z}_\tau^i))$. Now consider c_τ^i. Note that we can equally write $\hat{y}_\tau^i = \hat{x}_\tau^i - \alpha_\tau^i g(\hat{x}_\tau^i)$ with $g(\hat{x}_\tau^i) = S_\tau^{s+1,i} \nabla f(\hat{x}_\tau^i)$ and similarly with $\hat{h}_\tau^i$. Let $\delta_\tau^i = f_\tau(\hat{x}_\tau^i) - f_\tau(\hat{z}_\tau^i)$. Using first-order Taylor series expansion, we have

$$\begin{aligned} c_\tau^i &= \Big[\delta_\tau^i + \nabla f_\tau(\hat{x}_\tau^i)^T(\hat{y}_\tau^i - \hat{x}_\tau^i) - \nabla f_\tau(\hat{z}_\tau^i)^T(\hat{h}_\tau^i - \hat{z}_\tau^i)\Big]\delta_\tau^i \\ &= \Big[\delta_\tau^i - \alpha_\tau^i \nabla f_\tau(\hat{x}_\tau^i)^T g(\hat{x}_\tau^i) + \alpha_\tau^i \nabla f_\tau(\hat{z}_\tau^i)^T g(\hat{z}_\tau^i)\Big]\delta_\tau^i \\ &= {\delta_\tau^i}^2 - \alpha_\tau^i \xi_\tau^i \delta_\tau^i, \end{aligned}$$

where $\xi_\tau^i = \nabla f_\tau(\hat{x}_\tau^i)^T g(\hat{x}_\tau^i) - f_\tau(\hat{z}_\tau^i)^T g(\hat{z}_\tau^i)$. Upon substitution and rearrangement, we find

$$\mathbb{E}_{\tau,i}\langle \hat{y}_\tau^i - \hat{h}_\tau^i, \hat{x}_\tau^i - \hat{z}_\tau^i \rangle + c_\tau^i = \mathbb{E}_{\tau,i} \|x_\tau^i - z_\tau^i\|^2 - \alpha_\tau^i \Big[\langle \hat{x}_\tau^i - \hat{z}_\tau^i, g(\hat{x}_\tau^i) - g(\hat{z}_\tau^i)\rangle + \xi_\tau^i(\hat{x}_\tau^i - \hat{z}_\tau^i)\Big].$$

If the right-most term is positive in expectation, we are done. If not, note that it vanishes as α_τ^i grows small relative to β_s. In particular, choose α_τ^i to satisfy[3]

$$\alpha_\tau^i \leq \omega^{-1} \|x_\tau^i - z_\tau^i\|^2 \in (0, \infty),$$

where

$$\omega = \sup_{\tau,i} \quad \langle \hat{x}_\tau^i - \hat{z}_\tau^i, g(\hat{x}_\tau^i) - g(\hat{z}_\tau^i)\rangle + \xi_\tau^i(\hat{x}_\tau^i - \hat{z}_\tau^i).$$

Note that we must have $\omega > 0$, since if not there is no need to bound α_τ^i. This establishes $F(\psi_{s+1}^0, \Psi_{s+1}) \leq F(\psi_s^0, \Psi_s)$.

Finally, because $\mathbb{E}_{\tau,i} \|h_\tau^i - x_\tau^i\|^2 \leq \mathbb{E}_{\tau,i} \|h_\tau^i - z_\tau^i\|^2$, it follows that y_τ^{i-1} is tied to h_τ^i by some ball of the form $\|h_\tau^i - y_\tau^{i-1}\|^2 \leq {\alpha_\tau^i}^2 \|g(\hat{z}_\tau^i)\|^2 + \epsilon_\tau^i$, where ϵ_τ^i captures noise from the expectation. For β_s small this noise component vanishes, and since $\{\alpha_\tau^i\}_i$ is a converging sequence, the bounds grows increasingly tight. It follows then that $\{\psi_{s+1}^i\}_i$ converges to the same stationary point as $\{\psi_s^i\}_i$, yielding $\psi_{s+1}^0 \in \Theta$ for all s, as desired. ■

C Computing the meta-gradient

In practice we use stochastic gradient descent, and noise can cause $f_\tau(\psi_\tau^{s+1}) - f_\tau(\theta_\tau^i) > 0$. In such cases, the baseline is pulling θ^0 in the wrong direction and to prevent such distortions, we add regularizing terms that reverse the sign on the loss delta,

$$\mu_\tau^i(\theta_\tau^i; \psi_\tau^{s+1}) = \begin{cases} 0 & \text{if} \quad f(\psi_\tau^{i+1}) \leq f(\psi_\tau^i), \\ -2(f(\psi_\tau^{i+1}) - f(\theta_\tau^i))^2 & \text{else.} \end{cases}$$

μ_τ^i helps ensuring that θ_τ^i is pulled towards a minimum and is constructed to introduce the absolute value $|f_\tau(\psi_\tau^{i+1}) - f_\tau(\theta_\tau^i)|$ in the meta gradient. Evaluating the meta gradient at ψ^0, we have

$$\nabla F(\psi^0, \Psi) = 2\, \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{K_\tau - 1} J^i(\psi^0)^T \left(|\Delta f_\tau^i|\, \nabla f(\psi^i) - \Delta \psi_\tau^i \right) \right], \tag{8}$$

where J^i denotes the Jacobian of θ^i with respect to the initialization, $\Delta f_\tau^i = f_\tau(\psi_\tau^{i+1}) - f_\tau(\psi_\tau^i)$ and $\Delta \psi_\tau^i = \psi_\tau^{i+1} - \psi_\tau^i$. Ignoring the Jacobian, this expression contains only terms encountered during

[3] Generally, this bound is loose since for β_s small $g(\hat{x}_\tau^i)$ point in the same direction as $g(\hat{z}_\tau^i)$.

standard training and can be computed at a negligible cost (line 9 in algorithm 1). The Jacobians are costly to compute. Empirical evidence suggest that they are largely redundant (Finn et al., 2017; Nichol et al., 2018). We can understand this by noting that the elements of J can be written on the form $1 - h$, where h decreases exponentially fast as the learning rate schedule decreases. As such, for sufficiently small learning rates, $J^i = I_n$ is a reasonable approximation.

D Experiment details: Omniglot

Omniglot contains 50 alphabets, each consisting of a set of characters that in turn have 20 unique samples. We pretrain on up to 30 tasks, holding 10 tasks out for validation and 10 tasks for final evaluation. We use the same convolutional neural network architecture as in previous works (Vinyals et al., 2016; Finn et al., 2017; Schwarz et al., 2018) and train using stochastic gradient descent. For each alphabet, we define a task as a 20-class classification problem. For alphabets with more than 20 characters, we pick 20 characters at random, while alphabets with fewer characters are dropped from the validation set. The validation set is used to for early stopping of the meta training process. On each task, we train the model using stochastic gradient descent with a batch size of 20, and a learning rate of 0.1, except for fine-tuning where we find 0.01 to perform better than 0.1 or 0.001. The initialization is trained using stochastic gradient descent with a learning rate of 0.1, except for in the case of Reptile where a lower learning rate of 0.01 performed best. We use a task batch size of 20, sampled with replacement. We train on a given task until we observe no more improvement on a set of held-out images. Following Schwarz et al. (2018), we augment the dataset with 36 random rotations.

Table 1: Mean test error on held out tasks.*Author's suggested first-order approximation; †multi-headed finetuning; ‡single-headed finetuning.

Method No. Pretraining tasks	Leap	Finetune†	Reptile	MAML*	Finetune‡	No pretraining
1	8.7	7.5	8.4	8.8	15.4	17.0
2	6.6	6.3	8.0	8.8	11.7	16.8
4	4.0	4.7	7.2	8.6	10.9	17.9
6	3.0	4.4	6.3	8.4	11.6	16.1
8	3.1	4.2	6.1	8.7	12.3	15.9
10	2.8	4.0	5.7	9.2	13.7	16.1
12	2.7	3.6	5.4	8.6	15.5	16.2
14	2.7	3.8	5.2	8.6	15.1	16.6
16	2.3	3.9	5.1	9.1	16.1	16.7
18	2.8	3.7	5.1	8.7	17.3	15.9
20	2.6	3.9	4.7	9.4	21.6	16.6
22	2.4	3.7	4.9	8.6	26.0	16.2
24	2.4	3.5	5.0	8.7	30.7	15.9
26	2.2	3.3	4.5	9.2	37.7	16.9
28	2.7	3.8	4.8	9.3	40.3	17.0
30	2.1	3.6	4.6	8.7	43.6	16.1

E Experiment details: Multi-CV

In this experiment, we allow different architectures between tasks by using a distinct final linear layers for each task. We use the same neural network architecture as in the Omniglot experiment, and compare Leap against a baseline with no pretraining and one using multitask finetuning. On a given task, we train the model using stochastic gradient descent with a learning rate of 0.1 and a batch size of 32. Training is stopped once the validation accuracy does not improve or once a preset number of epochs is reached. For pretraining, we cap the number of training epochs to 10, and for the held-out task we cap the number of epochs to 100. We train Leap using Adam (Kingma & Ba, 2014) with a learning rate of 0.01 and a task batch size of 10. Once meta training is complete, we train on a

held-out dataset until the validation error stops improving, or the training process reaches 100 epochs. We no dataset augmentation; MNIST images are zero padded to have 32×32 images. We use the same normalizations as in Serrà et al. (2018).

Table 2: Transfer learning results on Multi-CV benchmark. All methods are trained until convergence on held-out tasks. † Area under training error curve; scaled to 0–100.

Held-out task	Method	Test (%)	Train (%)	AUC†	
Facescrub	Leap	19.9	0.0	11.6	
	Fine-tuning	32.7	0.0	13.2	
	No pre-training	18.2	0.0	10.5	$*$
NotMNIST	Leap	5.3	0.6	2.9	$*$
	Fine-tuning	5.4	2.0	4.4	
	No pre-training	5.4	2.6	5.1	
MNIST	Leap	0.7	0.1	0.6	$*$
	Fine-tuning	0.9	0.1	0.8	
	No pre-training	0.9	0.2	1.0	
Fashion MNIST	Leap	8.0	4.2	6.8	$*$
	Fine-tuning	8.9	3.8	7.0	
	No pre-training	8.4	4.7	7.8	
Cifar10	Leap	21.2	10.8	17.5	$*$
	Fine-tuning	27.4	13.3	20.7	
	No pre-training	26.2	13.1	23.0	
SVHN	Leap	8.4	5.6	7.5	$*$
	Fine-tuning	10.9	6.1	9.3	
	No pre-training	10.3	6.9	11.5	
Cifar100	Leap	52.0	30.5	43.4	$*$
	Fine-tuning	59.2	31.5	44.1	
	No pre-training	54.8	33.1	50.1	
Traffic Signs	Leap	2.9	0.0	1.2	$*$
	Fine-tuning	5.7	0.0	1.7	
	No pre-training	3.6	0.0	2.4	

F Experiment details: Atari

We use the same network as in Mnih et al. (2013), adopting it to actor-critic algorithms by estimating both value and policy through linear layers connected to the final output of a shared convolutional network. As is standard, we use downsampled $84 \times 84 \times 3$ RGB images as input. On each task, we use a batch size of 32, an unroll length of 5 and update the model parameters with RMSProp (using $\epsilon = 10^{-4}, \alpha = 0.99$) with a learning rate of 10^{-4}. We set the entropy cost to 0.01 and clip the absolute value of the rewards to maximum 5.0. The discounting factor was set to 0.99.