Transferring Knowledge across Learning Processes

Sebastian Flennerhag[*]
The Alan Turing Institute
sflennerhag@turing.ac.uk

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

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

Andreas Damianou
Amazon, Cambridge, UK
damianou@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 [9, 1]. 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 frame transfer learning as a meta learning problem with respect to the learning process [22, 7, 21, 19, 5, 26, 8], and associate each task with a manifold. We characterize how the training process travels from initialization to final parameters on this manifold, and construct a meta learning objective that learns an initialization that minimizes the expected length of this path.

As the training process grows longer in terms of the distance traversed on the loss surface (fig. 1), the geometry of this surface grows increasingly important. When adapting to a new task through a single or a handful of gradient steps, the geometry can largely be ignored. In contrast, for hundreds or thousands of gradient updates, it is the dominant feature of the training process. Our proposed method, Leap, scales meta learning beyond few-shot learning [11, 19, 8]. It is a light-weight framework that leverages only information obtained during training and can be computed on the fly at negligible cost. Leap outperforms competing methods, both in meta learning and transfer learning, on a set of computer vision tasks. Finally, we show that Leap transfers knowledge across learning processes in demanding reinforcement learning environments (Atari) involving millions of gradient steps.

2 Transferring Knowledge over Learning Processes

2.1 Gradient Paths on Task Manifolds

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. Given a learning objective f that consumes input $x \in \mathbb{R}^m$ and target $y \in \mathbb{R}^c$ and maps a parameterization $\theta \in \mathbb{R}^n$ to a scalar loss value, we 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. To distinguish different learning processes originating from the same initialization, we need a notion of their

[*]Work done while at Amazon.

2nd Workshop on Meta-Learning at NeurIPS 2018, Montréal, Canada.

Figure 1: *Left:* illustration of Leap (algorithm 1) for two tasks, τ and τ'. From an initialization θ^0, the learning process of each task generates gradient paths, Ψ_τ and $\Psi_{\tau'}$, which Leap uses to minimize the expected path length. Iterating the process, Leap converges to a locally Pareto optimal initialization. *Right:* the pull-forward objective (eq. 5) used to minimize the expected gradient path length. Any gradient path $\Psi_\tau = \{\psi^i_\tau\}_{i=1}^{K_\tau}$ acts on θ^0 by pulling each θ^i_τ towards ψ^{i+1}_τ.

length. The longer the process, the worse the initialization is (conditional on reaching equivalent performance, discussed further below). Measuring the Euclidean distance between initialization and final parameters is misleading as it ignores the actual path taken. This becomes crucial when we compare paths from different tasks, as gradient paths from different tasks can originate from the same initialization and converge to similar final parameters, but take very different paths. Therefore, to capture the length of a learning process we must associate it with the loss surface it traversed.

The process of learning a task can be seen as a curve on a specific task manifold M. We exploit that, by definition, any learning process traverses the loss surface of f and propose a framework for transfer learning that aims to minimize the expected length of gradient descent paths *across task manifolds*. In doing so, the meta objective intrinsically balances local geometries across tasks and encourages an initialization that makes the learning process as short as possible. The length (or energy) of any *curve* $\gamma : [0, 1] \mapsto M$ is defined by accumulating infinitesimal changes along the trajectory, detailed in appendix A. To measure the length, we need a Riemann metric on the manifold; for the loss surface the induced Riemann metric is the standard inner product. Other Riemann metrics give rise to different manifold structures, possibly with information geometric interpretations [4, 15, 18, 14].

To measure the length of a learning process, we note that gradient descent can be seen as a discrete approximation to the scaled gradient flow $\dot{\theta}(t) = -S(t)\nabla f(\theta(t))$. This flow describes a curve that originates in $\gamma(0) = (\theta^0, f(\theta^0))$ and follows the scaled gradient at each point. We refer to this curve as the *gradient path* from θ^0 on M with approximate length or energy given by

$$d_p(\theta^0, M) = \sum_{i=0}^{K-1} \|\gamma^{i+1} - \gamma^i\|_2^p, \qquad p \in \{1, 2\}. \tag{2}$$

We write d when the distinction between the length or energy metric is immaterial. Importantly, d involves only terms seen during task training. We exploit this later when we construct the meta gradient, enabling us to perform gradient descent on the meta objective at negligible cost (eq. 8). We now turn to the transfer learning setting, where we face a set of tasks, each with a distinct task manifold. Our framework is built on the idea that we can transfer knowledge across learning processes via the local geometry by aggregating information obtained along observed gradient paths. As such, Leap finds an initialization from which learning converges as rapidly as possible in expectation.

2.2 Meta Learning across Task Manifolds

Formally, we define a task $\tau = (f_\tau, p_\tau, u_\tau)$ as the process of learning to approximate the relationship $x \mapsto y$ 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 K_τ times to minimize the task objective f_τ. Thus, a learning process starts at $\theta^0_\tau = \theta^0$ and progresses via $\theta^{i+1}_\tau = u_\tau(\theta^i_\tau)$ until $\theta^{K_\tau}_\tau$ is obtained. The sequence $\{\theta^i_\tau\}_{i=0}^{K_\tau}$ defines an approximate gradient path on the task manifold M_τ with distance $d(\theta^0; M_\tau)$.

Because the gradient path length does not distinguish between the generalizing performance at limit points, it is necessary to introduce a feasibility constraint to ensure only initializations with some minimum level of performance are considered. We leverage that transfer learning never happens in a vacuum; we always have a second-best option. Starting from this "second-best" initialization, ψ^0,

Algorithm 1 Leap: Transferring Knowledge over Learning Processes

Require: $p(\tau),\ \tau = (f_\tau, p_\tau, u_\tau)$: distribution over tasks
Require: β: step size
randomly initialize θ^0
while not done **do**
  $\nabla F \leftarrow 0$: initialize meta gradient
  sample task batch $\mathcal{B}$ from $p(\tau)$
  for all $\tau \in \mathcal{B}$ **do**
    $\psi_\tau^0 \leftarrow \theta^0$ initialize task baseline
    for all $i \in \{0, \dots, K_\tau - 1\}$ **do**
      $\psi_\tau^{i+1} \leftarrow u_\tau(\psi_\tau^i)$: update baseline
      $\theta_\tau^i \leftarrow \psi_\tau^i$: follow baseline (since $\psi_\tau^0 = \theta^0$)
      increment ∇F using the pull-forward gradient (appendix C)
    end for
  end for
  $\theta^0 \leftarrow \theta^0 - \frac{\beta}{|\mathcal{B}|}\nabla F$: update initialization
end while

gives us the level of performance we would otherwise obtain on each task. This provides us with an upper bound: a candidate solution θ^0 has to satisfy $f_\tau(\theta_\tau^{K_\tau}) \leq f_\tau(\psi_\tau^{K_\tau})$ for every task in $p(\tau)$. Taking this requirement as our feasibility constraint, we obtain the canonical meta-objective

$$\begin{aligned} \min_{\theta^0} \quad & F(\theta^0) = \mathbb{E}_{\tau \sim p(\tau)}\left[d(\theta^0; M_\tau)\right] \\ \text{s.t.} \quad & \theta_\tau^{i+1} = u_\tau(\theta_\tau^i), \quad \theta_\tau^0 = \theta^0, \\ & \theta^0 \in \Theta = \cap_\tau \left\{\theta^0 \;\middle|\; f_\tau(\theta_\tau^{K_\tau}) \leq f_\tau(\psi_\tau^{K_\tau})\right\}. \end{aligned} \tag{3}$$

2.3 Leap

Solving eq. 3 naïvely requires training to convergence on each task to evaluate the feasibility constraint, which can be very costly. Fortunately, because we have access to ψ^0, we can solve eq. 3 more efficiently by obtaining gradient paths $\Psi_\tau = \{\psi_\tau^i\}_{i=0}^{K_\tau}$ from ψ^0 for each task τ in a batch $\mathcal{B}$. These provide a set of baselines $\Psi = \{\Psi_\tau\}_{\tau \in \mathcal{B}}$ that we can use to incrementally improve the initialization with respect to. This improved initialization converges to the same limit points, but with shorter expected gradient path distances (theorem 1). As such, it becomes the new second-best option; Leap (algorithm 1) repeats this process, ultimately finding a solution to the canonical meta objective. Formally, we use the baselines to modify the gradient path distance metric in eq. 2 by freezing the forward point γ_τ^{i+1} in all norms,

$$\bar{d}_p(\theta^0; M_\tau, \Psi_\tau) = \sum_{i=0}^{K_\tau - 1} \|\bar{\gamma}_\tau^{i+1} - \gamma_\tau^i\|_2^p, \tag{4}$$

where $\bar{\gamma}_\tau^i = (\psi_\tau^i, f(\psi_\tau^i))$ represents the frozen forward point from the baseline and $\gamma_\tau^i = (\theta_\tau^i, f(\theta_\tau^i))$ the point on the gradient path originating from θ^0. This surrogate distance metric directly encodes the feasibility constraint; optimizing θ^0 with respect to Ψ therefore pulls the initialization forward along each task-specific gradient path in an unconstrained variant of eq. 3 that replaces Θ with Ψ,

$$\begin{aligned} \min_{\theta^0} \quad & \bar{F}(\theta^0; \Psi) = \mathbb{E}_{\tau \sim p(\tau)}\left[\bar{d}(\theta^0; M_\tau, \Psi_\tau)\right], \\ \text{s.t.} \quad & \theta_\tau^{i+1} = u_\tau(\theta_\tau^i), \quad \theta_\tau^0 = \theta^0. \end{aligned} \tag{5}$$

We refer to eq. 5 as the *pull-forward* objective. Leap therefore produces a sequence of candidate solutions to eq. 3, all in Θ, with incrementally shorter gradient paths. In principle, $\bar{F}$ can be evaluated for any θ^0, but a more efficient strategy is to evaluate θ^0 at ψ^0. In this case, $\bar{d} = d$, so that $\bar{F} = F$.

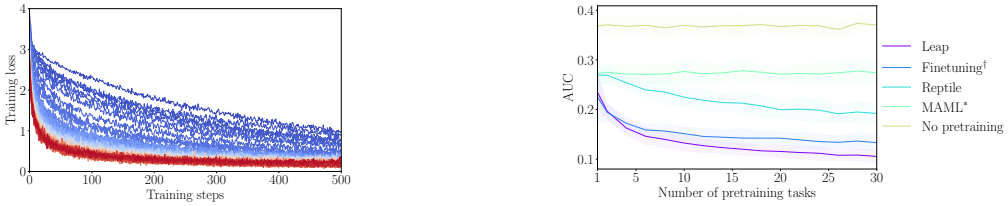

Figure 2: Transfer learning on Omniglot. *Left:* Evolution of training curves during meta training of Leap. *Right:* AUC across number of pretraining tasks. *Author's suggested first-order approximation; †multi-headed finetuning; Shading: standard deviation across 10 seeds.

Theorem 1 (Pull-forward)**.** *Let $\psi_0^0 \in \Theta$ be given and define a sequence of initializations $\{\psi_s^0\}_{s\in\mathbb{N}}$ by $\psi_{s+1}^0 = \psi_s^0 - \beta_s \nabla \bar{F}(\psi_s^0; \Psi_s)$. 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 limit point in Θ.*

Proof; see appendix B. Crucially, when F is evaluated at ψ^0, an approximate meta gradient ∇F can be computed analytically using *only* terms already computed during the task training. As such, Leap can be computed on the fly during training at negligible cost. By explicitly taking the task geometry into account, Leap outperforms competing methods, both in meta learning and transfer learning, and can scale to learning processes that involves millions of gradient steps.

3 Empirical Results

We conduct three experiments in increasingly complex transfer learning environments. We measure transfer learning in terms of final test error and area under the error curve on the training set.

Omniglot: We transfer knowledge between alphabets in Omniglot [12], varying the number of pretraining tasks from 1 to 30, holding out 10 tasks for evaluation. We compare against no pretraining, finetuning, first-order approximation of MAML [8], and Reptile [17]. See appendix F for details. The gap between Leap and other meta learning frameworks is large; Leap is in fact the only meta learner to outperform finetuning. When pretraining on more than 4 tasks, Leap achieves superior performance and the performance margin grows with the number of pretraining tasks (fig. 2).

Multi-CV: We consider a set of computer vision datasets as tasks and pretrain on all but one, which is held out for final evaluation. For details and results see appendix G. This is a more challenging setting both due to greater diversity and complexity among tasks. Leap outperforms baselines on all but one transfer learning task, where any form of pretraining results in worse performance.

Atari: To demonstrate that Leap can scale to large problems, we apply it to agents playing Atari 2600 games [6]. We use a variant of the actor-critic architecture for all experiments [25]; see appendix H for details. We train on each task in the meta-batch for five million training steps, accumulating the meta gradient as in algorithm 1. We train Leap for 100 meta gradient steps, which is sufficient to see a difference on a held-out set of games. Leap learns a useful policy significantly faster and more consistently than a random initialization fig. 3.

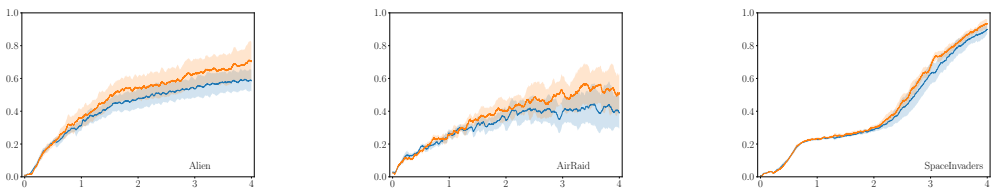

Figure 3: Mean normalized episode scores on Atari games across training steps. Scores are reported as moving average over 500 episodes. Shaded regions depict two standard deviations across ten seeds. Leap (orange) generally outperforms a random initialization (blue). While AirRaid is in the pretraining set, neither Alien nor SpaceInvader is.

References

[1] A. Achille, T. Eccles, L. Matthey, C. P. Burgess, N. Watters, A. Lerchner, and I. Higgins. Life-long disentangled representation learning with cross-domain latent homologies. *arXiv preprint arXiv:1808.06508*, 2018.

[2] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. *The Theory of Splines and Their Applications*. Academic Press, 1967. p. 51.

[3] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[4] S.-i. Amari and H. Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Society, 2007.

[5] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016.

[6] 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, 2013.

[7] Y. Bengio, S. Bengio, and J. Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle, 1991.

[8] C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, 2017.

[9] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017.

[10] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

[11] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2011.

[12] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[13] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. In *International Conference on Learning Representations*, 2017.

[14] K. Luk and R. Grosse. A coordinate-free construction of scalable natural gradient. *arXiv preprint arXiv:1808.10340*, 2018.

[15] J. Martens. Deep learning via hessian-free optimization. In *International Conference on Machine Learning*, 2010.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[17] A. Nichol, J. Achiam, and J. Schulman. On First-Order Meta-Learning Algorithms. *arXiv preprint ArXiv:1803.02999*, 2018.

[18] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. In *International Conference on Learning Representations*, 2014.

[19] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2016.

[20] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[21] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

[22] J. Schmidhuber. *Evolutionary principles in self-referential learning*. PhD thesis, Technische Universität München, 1987.

[23] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018.

[24] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, 2018.

[25] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. MIT Press, Cambridge, 1998.

[26] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems*, 2016.