Appendix

A Geometries on learning manifolds

To formalize the notion of the distance of a learning process, we define a task manifold M as a submanifold of $\mathbb{R}^{n+1}$ given by the graph of f. Every point $p = (\theta, f(\theta)) \in M$ is locally homeomorphic to a Euclidean subspace, described by the tangent space T_pM. Taking $\mathbb{R}^{n+1}$ to be Euclidean, it is a Riemann manifold. By virtue of being a submanifold of $\mathbb{R}^{n+1}$, M is also a Riemann manifold. As such, M comes equipped with an smoothly varying inner product $g_p : T_pM \times T_pM \mapsto \mathbb{R}$ on tangent spaces, allowing us to measure the length of a path on M. In particular, the length (or energy) of any *curve* $\gamma : [0, 1] \mapsto M$ is defined by accumulating infinitesimal changes along the trajectory,

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

where $\dot\gamma(t) = \frac{d}{dt}\gamma(t) \in T_{\gamma(t)}M$ is a tangent vector of $\gamma(t) = (\theta(t), f(\theta(t))) \in M$. We use parenthesis (i.e. $\gamma(t)$) to differentiate discrete and continuous domains. With M being a submanifold of $\mathbb{R}^{n+1}$, it inherits its metric via pullback, with induced metric $g_{\gamma(t)}(\dot\gamma(t), \dot\gamma(t)) = \langle\dot\gamma(t), \dot\gamma(t)\rangle$. Different Riemann metrics yield different manifold structures. In particular, if the model underlying f admits a predictive probability distribution $P(y \mid x)$, the task manifold can be given an information geometric interpretation by choosing the Fisher matrix as Riemann metric, in which case the task manifold is defined over the space of probability distributions [4]. If eq. 1 is defined as natural gradient descent, the learning process corresponds to gradient descent on this manifold [3, 15, 18, 14].

Having a complete description of a task manifold, we can measure the length of a learning process by noting 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. Going forward, we define γ to be this unique curve and refer to it as the *gradient path* from θ^0 on M. The gradient path length or energy can be computed exactly, but in practice we observe a discrete learning process. Analogously to how the gradient update rule approximates the gradient flow, the gradient path length or energy can be approximated by the cumulative chordal distance [2],

$$d_p(\theta^0, M) = \sum_{i=0}^{K-1} \|\gamma^{i+1} - \gamma^i\|_2^p, \qquad p \in \{1, 2\}. \tag{7}$$

B Proof of theorem 1

Proof. We first establish that, for all s,

$$\mathbb{E}_\tau\, d(\psi^0_{s+1}, M_\tau) = F(\psi^0_{s+1}) = \bar F(\psi^0_{s+1}; \Psi_{s+1}) \le \bar F(\psi^0_s; \Psi_s) = F(\psi^0_s) = \mathbb{E}_\tau\, d(\psi^0_s, M_\tau),$$

with strict inequality for at least some s. Because $\{\beta_s\}_{s=1}^\infty$ satisfies the gradient descent criteria, it follows that the sequence $\{\psi^0_s\}_{s=1}^\infty$ is convergent. To complete the proof we must show that this limit point is in Θ. To this end, we show that for β_s sufficiently small, for all s, $\lim_{i\to\infty}\psi^i_{s+1} = \lim_{i\to\infty}\psi^i_s$. That is, each updated initialization incrementally reduces the expected gradient path length while converging to the same limit point as ψ^0_0. Since $\psi^0_0 \in \Theta$ by assumption, we obtain $\psi^0_s \in \Theta$ for all s as an immediate consequence.

To establish $\mathbb{E}_\tau\, d(\psi^0_{s+1}, M_\tau) \le \mathbb{E}_\tau\, d(\psi^0_s, M_\tau)$, with strict inequality for some s, let

$$z^i_\tau = (\psi^{s,i}_\tau, f_\tau(\psi^{s,i}_\tau)) \qquad x^i_\tau = (\psi^{s+1,i}_\tau, f_\tau(\psi^{s+1,i}_\tau))$$
$$h^i_\tau = (\psi^{s,i+1}_\tau, f_\tau(\psi^{s,i+1}_\tau)) \qquad y^i_\tau = (\psi^{s+1,i+1}_\tau, f_\tau(\psi^{s+1,i+1}_\tau))$$

and denote by $\mathbb{E}_{\tau,i}$ the expectation over gradient paths, $\mathbb{E}_{\tau\sim p(\tau)}\sum_{i=1}^{K_\tau}$. Note that

$$\bar{F}(\psi_s^0,\Psi_s)=\mathbb{E}_{\tau,i}\left\|h_\tau^i-z_\tau^i\right\|_2^p \qquad \bar{F}(\psi_s^0,\Psi_{s+1})=\mathbb{E}_{\tau,i}\left\|y_\tau^i-z_\tau^i\right\|_2^p$$
$$\bar{F}(\psi_{s+1}^0\Psi_s)=\mathbb{E}_{\tau,i}\left\|h_\tau^i-x_\tau^i\right\|_2^p \qquad \bar{F}(\psi_{s+1}^0,\Psi_{s+1})=\mathbb{E}_{\tau,i}\left\|y_\tau^i-x_\tau^i\right\|_2^p$$

with $p=2$ defining the meta objective in terms of the gradient path energy and $p=1$ in terms of the gradient path length. As we are exclusively concerned with the Euclidean norm, we omit the subscript. By assumption, every β_s is sufficiently small to satisfy the gradient descent criteria $\bar{F}(\psi_s^0;\Psi_s)\geq\bar{F}(\psi_{s+1}^0;\Psi_s)$. Adding and subtracting $\bar{F}(\psi_{s+1}^0,\Psi_{s+1})$ to the RHS, we have

$$\begin{aligned}\mathbb{E}_{\tau,i}\left\|h_\tau^i-z_\tau^i\right\|^p &\geq \mathbb{E}_{\tau,i}\left\|h_\tau^i-x_\tau^i\right\|^p\\ &=\mathbb{E}_{\tau,i}\left\|y_\tau^i-x_\tau^i\right\|^p+\left\|h_\tau^i-x_\tau^i\right\|^p-\left\|y_\tau^i-x_\tau^i\right\|^p.\end{aligned}$$

It follows that $\mathbb{E}_\tau\, d(\psi_s^0,M_\tau)\geq\mathbb{E}_\tau\, d(\psi_{s+1}^0,M_\tau)$ if $\mathbb{E}_{\tau,i}\left\|h_\tau^i-x_\tau^i\right\|^p\geq\mathbb{E}_{\tau,i}\left\|y_\tau^i-x_\tau^i\right\|^p$. As our main concern is existence, we will show something stronger, namely that there exists α_τ^i such that

$$\left\|h_\tau^i-x_\tau^i\right\|^p\geq\left\|y_\tau^i-x_\tau^i\right\|^p \qquad \forall i,\tau,s,p$$

with at least one such inequality strict for some i,τ,s, in which case $d_p(\psi_{s+1}^0,M_\tau)<d_p(\psi_s^0,M_\tau)$ for any $p\in\{1,2\}$. We proceed by establishing the inequality for $p=2$ and obtain $p=1$ as an immediate consequence of monotonicity of the square root. Expanding $\left\|h_\tau^i-x_\tau^i\right\|^2$ we have

$$\begin{aligned}\left\|h_\tau^i-x_\tau^i\right\|^2-\left\|y_\tau^i-x_\tau^i\right\|^2&=\left\|(h_\tau^i-z_\tau^i)+(z_\tau^i-x_\tau^i)\right\|^2-\left\|y_\tau^i-x_\tau^i\right\|^2\\ &=\left\|h_\tau^i-z_\tau^i\right\|^2+2\langle h_\tau^i-z_\tau^i,z_\tau^i-x_\tau^i\rangle+\left\|z_\tau^i-x_\tau^i\right\|^2-\left\|y_\tau^i-x_\tau^i\right\|^2.\end{aligned}$$

Every term except $\left\|z_\tau^i-x_\tau^i\right\|^2$ can be minimized by choosing α_τ^i small, whereas $\left\|z_\tau^i-x_\tau^i\right\|^2$ is controlled by β_s. Thus, our strategy is to make all terms except $\left\|z_\tau^i-x_\tau^i\right\|^2$ small, for a given β_s, by placing an upper bound on α_τ^i. We first show that $\left\|h_\tau^i-z_\tau^i\right\|^2-\left\|y_\tau^i-x_\tau^i\right\|^2=O\left({\alpha_\tau^i}^2\right)$. Some care is needed as the $(n+1)$th dimension is the loss value associated with the other n dimensions. Define $\hat{z}_\tau^i=\psi_\tau^{s,i}$, so that $z_\tau^i=(\hat{z}_\tau^i,f_\tau(\hat{z}_\tau^i))$. Similarly define $\hat{x}_\tau^i,\hat{h}_\tau^i$, and $\hat{y}_\tau^i$ to obtain

$$\begin{aligned}\left\|h_\tau^i-z_\tau^i\right\|^2&=\left\|\hat{h}_\tau^i-\hat{z}_\tau^i\right\|^2+\left(f_\tau(\hat{h}_\tau^i)-f_\tau(\hat{z}_\tau^i)\right)^2\\ \left\|y_\tau^i-x_\tau^i\right\|^2&=\left\|\hat{y}_\tau^i-\hat{x}_\tau^i\right\|^2+\left(f_\tau(\hat{y}_\tau^i)-f_\tau(\hat{x}_\tau^i)\right)^2\\ 2\langle h_\tau^i-z_\tau^i,z_\tau^i-x_\tau^i\rangle&=2\langle\hat{h}_\tau^i-\hat{z}_\tau^i,\hat{z}_\tau^i-\hat{x}_\tau^i\rangle+(f_\tau(\hat{h}_\tau^i)-f_\tau(\hat{z}_\tau^i))(f_\tau(\hat{z}_\tau^i)-f_\tau(\hat{x}_\tau^i)).\end{aligned}$$

Consider $\left\|\hat{h}_\tau^i-\hat{z}_\tau^i\right\|^2-\left\|\hat{y}_\tau^i-\hat{x}_\tau^i\right\|^2$. Note that $\hat{h}_\tau^i=\hat{z}_\tau^i-\alpha_\tau^i g(\hat{z}_\tau^i)$, where $g(\hat{z}_\tau^i)=S_\tau^{s,i}\nabla f(\hat{z}_\tau^i)$, and similarly $\hat{y}_\tau^i=\hat{x}_\tau^i-\alpha_\tau^i g(\hat{x}_\tau^i)$ with $g(\hat{x}_\tau^i)=S_\tau^{s+1,i}\nabla f(\hat{x}_\tau^i)$. Thus, $\left\|\hat{h}_\tau^i-\hat{z}_\tau^i\right\|^2={\alpha_\tau^i}^2\|g(\hat{z}_\tau^i)\|^2$ and similarly for $\left\|\hat{y}_\tau^i-\hat{x}_\tau^i\right\|^2$, so

$$\left\|\hat{h}_\tau^i-\hat{z}_\tau^i\right\|^2-\left\|\hat{y}_\tau^i-\hat{x}_\tau^i\right\|^2={\alpha_\tau^i}^2\left(\|g(\hat{z}_\tau^i)\|^2-\|g(\hat{x}_\tau^i)\|^2\right)=O\left({\alpha_\tau^i}^2\right).$$

Now consider $\left(f_\tau(\hat{h}_\tau^i)-f_\tau(\hat{z}_\tau^i)\right)^2-\left(f_\tau(\hat{y}_\tau^i)-f_\tau(\hat{x}_\tau^i)\right)^2$. Using the above identities and first-order Taylor series expansion, we have

$$\begin{aligned}\left(f_\tau(\hat{h}_\tau^i)-f_\tau(\hat{z}_\tau^i)\right)^2&=\left(\nabla f_\tau(\hat{z}_\tau^i)^T(\hat{h}_\tau^i-\hat{z}_\tau^i)+O\left(\alpha_\tau^i\right)\right)^2\\ &=\left(-\alpha_\tau^i\nabla f_\tau(\hat{z}_\tau^i)^T g(\hat{z}_\tau^i)+O\left(\alpha_\tau^i\right)\right)^2=O\left({\alpha_\tau^i}^2\right),\end{aligned}$$

and similarly for $(f_\tau(\hat{y}_\tau^i) - f_\tau(\hat{x}_\tau^i))^2$. As such, $\|h_\tau^i - z_\tau^i\|^2 - \|y_\tau^i - x_\tau^i\|^2 = O\left({\alpha_\tau^i}^2\right)$.

Finally, consider the inner product $\langle h_\tau^i - z_\tau^i, z_\tau^i - x_\tau^i \rangle$. From above we have that $(f_\tau(\hat{h}_\tau^i) - f_\tau(\hat{z}_\tau^i))^2 = -\alpha_\tau^i(\nabla f_\tau(\hat{z}_\tau^i)^T g(\hat{z}_\tau^i) - R_\tau^i) = -\alpha_\tau^i \xi_\tau^i$, where R_τ^i denotes an upper bound on the residual. We extend g to operate on z_τ^i by defining $\tilde{g}(z_\tau^i) = (g(\hat{z}_\tau^i), \xi_\tau^i)$. Returning to $\|h_\tau^i - x_\tau^i\|^2 - \|y_\tau^i - x_\tau^i\|^2$, we have

$$\begin{aligned}\|h_\tau^i - x_\tau^i\|^2 - \|y_\tau^i - x_\tau^i\|^2 &= \|z_\tau^i - x_\tau^i\|^2 + 2\langle h_\tau^i - z_\tau^i, z_\tau^i - x_\tau^i \rangle + O\left({\alpha_\tau^i}^2\right) \\ &= \|z_\tau^i - x_\tau^i\|^2 - 2\alpha_\tau^i \langle \tilde{g}(z_\tau^i), z_\tau^i - x_\tau^i \rangle + O\left({\alpha_\tau^i}^2\right).\end{aligned}$$

The first term is non-negative, and importantly, always non-zero whenever $\beta_s \neq 0$. Furthermore, α_τ^i can always be made sufficiently small for $\|z_\tau^i - x_\tau^i\|^2$ to dominate the residual, so we can focus on the inner product $\langle \tilde{g}(z_\tau^i), z_\tau^i - x_\tau^i \rangle$. If it is negative, all terms are positive and we have $\|h_\tau^i - x_\tau^i\|^2 \geq \|y_\tau^i - x_\tau^i\|^2$ as desired. If not, $\|z_\tau^i - x_\tau^i\|^2$ dominates if

$$\alpha_\tau^i \leq \frac{\|z_\tau^i - x_\tau^i\|^2}{2\langle \tilde{g}(z_\tau^i), z_\tau^i - x_\tau^i \rangle} \in (0, \infty).$$

Thus, for α_τ^i sufficiently small, we have $\|h_\tau^i - x_\tau^i\|^2 \geq \|y_\tau^i - x_\tau^i\|^2 \; \forall i, \tau, s$, with strict inequality whenever $\langle \tilde{g}(z_\tau^i), z_\tau^i - x_\tau^i \rangle < 0$ or the bound on α_τ^i holds strictly. This establishes $d_2(\psi_{s+1}^0, M_\tau) \leq d_2(\psi_s^0, M_\tau)$ for all τ, s, with strict inequality for at least some τ, s. To also establish it for the gradient path length ($p = 1$), taking square roots on both sides of $\|h_\tau^i - x_\tau^i\|^2 \geq \|y_\tau^i - x_\tau^i\|^2$ yields the desired results, and so $\|h_\tau^i - x_\tau^i\|^p \geq \|y_\tau^i - x_\tau^i\|^p$ for $p \in \{1, 2\}$, establishing

$$d(\psi_{s+1}^0, M_\tau) = \bar{F}(\psi_{s+1}^0; \Psi_{s+1}) \leq \bar{F}(\psi_s^0; \Psi_s) = d(\psi_s^0, M_\tau) \quad \forall \tau, s$$

with strict inequality for at least some τ, s, in particular whenever $\beta_s \neq 0$ and α_τ^i sufficiently small.

Then, to see that the limit point of Ψ_{s+1} is the same as that of Ψ_s for β_s sufficiently small, note that $x_\tau^i = y_\tau^{i-1}$. As before, by the gradient descent criteria, β_s is such that

$$\mathbb{E}_{\tau,i} \|h_\tau^i - x_\tau^i\|^p = \mathbb{E}_{\tau,i} \|h_\tau^i - y_\tau^{i-1}\|^p \leq \mathbb{E}_{\tau,i} \|h_\tau^i - z_\tau^i\|^p = \mathbb{E}_{\tau,i} {\alpha_\tau^i}^p \|\tilde{g}(z_\tau^i)\|^p.$$

Define ϵ_τ^i as the noise residual from the expectation; each y_τ^{i-1} is bounded by $\|h_\tau^i - y_\tau^{i-1}\|^p \leq {\alpha_\tau^i}^p \|\tilde{g}(z_\tau^i)\|^p + \epsilon_\tau^i$. For β_s small this noise component vanishes, and since $\{\alpha_\tau^i\}_i$ is a converging sequence, the bound on y_τ^{i-1} grows increasingly tight. It follows then that $\{\psi_{s+1}^i\}_{i=1}^\infty$ converges to the same limit point as $\{\psi_s^i\}_{i=1}^\infty$, yielding $\psi_{s+1}^0 \in \Theta$ for all s, as desired. ■

C Computing the meta-gradient

Differentiating $\bar{F}$, we have

$$\nabla \bar{F}(\theta^0, \Psi) = -p \, \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{K_\tau - 1} J_\tau^i(\theta_\tau^0)^T \left(\Delta f_\tau^i \, \nabla f_\tau(\theta_\tau^i) + \Delta \theta_\tau^i \right) \left(\|\bar{\gamma}_\tau^i - \gamma_\tau^i\|_2^p \right)^{p-2} \right] \tag{8}$$

where J_τ^i denotes the Jacobian of θ_τ^i with respect to the initialization, $\Delta f_\tau^i = f_\tau(\psi_\tau^{i+1}) - f_\tau(\theta_\tau^i)$ and $\Delta \theta_\tau^i = \psi_\tau^{i+1} - \theta_\tau^i$. To render the meta gradient tractable, we need to approximate the Jacobians, as these are extremely costly to compute. Empirical evidence suggest that they are largely redundant [8, 17]. Nichol et al. ([17]) further shows that the resulting meta-gradient remains faithful to the original meta objective. We provide some further support for this approximation and find that the identity approximation holds relatively well (see appendix D).

In practice we use stochastic gradient descent during task training. This injects noise in f as well as in its gradient, resulting in a noisy gradient path. Noise in the gradient path does not prevent

Leap from converging, as long as the gradient path converges. However, noise reduces the rate of convergence, in particular when a noisy gradient step results in $f_\tau(\psi_\tau^{s+1}) - f_\tau(\theta_\tau^i) > 0$. This causes the term $\Delta f_\tau^i \nabla f_\tau(\theta_\tau^i)$ in eq. 8 to point in an ascent direction. We found that adding a stabilizer to ensure we always follow the descent direction significantly speeds up convergence and allows the use of larger learning rates. In this paper, we augment $\bar{F}$ with a stabilizer of the form

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

Adding $\nabla\mu$ (re-scaled if necessary) to the meta-gradient is equivalent to replacing Δf_τ^i with $-|\Delta f_\tau^i|$ in eq. 8. This ensures that we never follow $\nabla f_\tau(\theta_\tau^i)$ in the ascent direction. As $\Delta f_\tau^i > 0$ grows large, the stabilizer increasingly de-prioritize following the line segment, instead encouraging Leap to move away from problematic neighborhoods of the loss surface. This stabilizer is a heuristic, there are many others that could prove helpful. In appendix E we perform an ablation study and find that the stabilizer is not necessary for Leap to converge, but it does speed up convergence significantly.

D Ablation Study: Approximating Jacobians

To understand the role of the Jacobians, note that (we drop task subscripts for simplicity)

$$\begin{aligned} J^{i+1}(\theta^0) &= \left(I_n - \alpha^i S^i H_f(\theta^i)\right) J^i(\theta^0) = \prod_{j=0}^{i} \left(I_n - \alpha^j S^j H_f(\theta^j)\right) \\ &= I_n - \sum_{j=0}^{i} \alpha^i S^i H_f(\theta^i) + O\left(\alpha^{i^2}\right), \end{aligned}$$

where $H_f(\theta^j)$ denotes the Hessian of f at θ^j. Thus, changes to θ^{i+1} are translated into θ^0 via all intermediary Hessians. This makes the Jacobians memoryless up to second-order curvature. Importantly, the effect of curvature can directly be controlled via α^i, and by choosing α^i small we can ensure $J^i(\theta^0) \approx I_n$ to be a reasonable assumption. In practice, this approximation works well [c.f. 8, 17]. Moreover, as a practical matter, if the alternative is some other approximation to the Hessians, the amount of noise injected grows exponentially with every iteration, making it an extremely challenging problem beyond the scope of this paper.

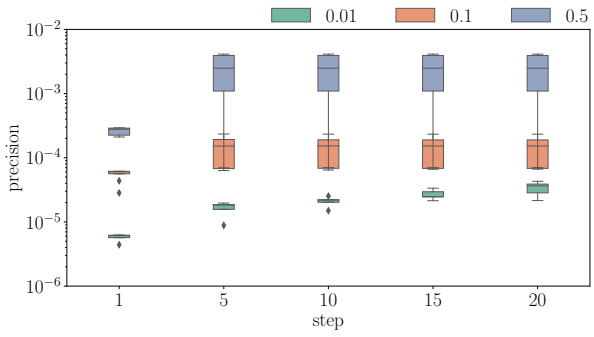

Figure 4: Relative precision of Jacobian approximation. Precision is calculated for the Jacobian of the first layer, across different learning rates (colors) and gradient steps.

To understand how this approximation limits our choice of learning rates α^i, we conduct an ablation study in the Omniglot experiment setting. We are interested in the relative precision of the identity approximation under different learning rates and across time steps, which we define as

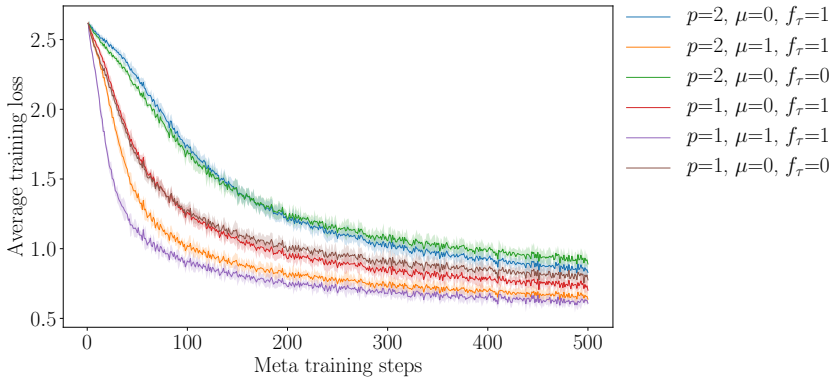

Figure 5: Average task training loss over meta-training steps. p denotes the $\bar{d}_p$ used in the meta objective, $\mu = 1$ the use of the stabilizer, and $f_\tau = 1$ the inclusion of the loss in the task manifold.

$$\rho\left(i, \{\alpha^j\}_{j=0}^{i}\right) = \frac{\|I_n - J^i(\theta^0)\|_1}{\|J^i(\theta^0)\|_1},$$

where the norm is the Schatten 1-norm. We use the same four-layer convolutional neural network as in the Omniglot experiment (appendix F). For each choice of learning rate, we train a model from a random initialization for 20 steps and compute ρ every 5 steps. Due to exponential growth of memory consumption, we were unable to compute ρ for more than 20 gradient steps. We report the relative precision of the first convolutional layer. We do not report the Jacobian with respect to the other layers, all being considerably larger, was too costly. We computed ρ for all layers on the first five gradient steps and found no significant variation in precision across layers. Consequently, we prioritize reporting how precision varies with the number of gradient steps. As in the main experiments, we use stochastic gradient descent. We evaluate $\alpha^i = \alpha \in \{0.01, 0.1, 0.5\}$ across 5 different tasks. Figure 4 summarizes our results.

Reassuringly, we find the identity approximation to be accurate to at least the fourth decimal for learning rates we use in practice, and to the third decimal for the largest learning rate (0.5) we were able to converge with. Importantly, except for the smallest learning rate, the quality of the approximation is constant in the number of gradient steps. The smallest learning rate that exhibits some deterioration on the fifth decimal, however larger learning rates provide an upper bound that is constant on the fourth decimal, indicating that this is of minor concern. Finally, we note that while these results suggest the identity approximation to be a reasonable approach on the class of problems we consider, other settings may put stricter limits on the effective size of learning rates.

E Ablation Study: Leap Hyper-Parameters

As Leap is a general framework, we have several degrees of freedom in specifying a meta learner. In particular, we are free to choose the task manifold structure, the gradient path distance metric, d_p, and whether to incorporate stabilizers. These are non-trivial choices and to ascertain the importance of each, we conduct an ablation study. We vary (a) the task manifold between using the full loss surface and only parameter space, (b) the gradient path distance metric between using the energy or length, and (c) inclusion of the stabilizer μ in the meta objective. We stay as close as possible to the set-up used in the Omniglot experiment (appendix F), fixing the number of pretraining tasks to 30. We use fewer training steps in the inner and outer loop during meta-training; we perform 500 meta gradient updates, which is sufficient to converge, and we use a maximum of 100 gradient steps on each task during meta-training (i.e. $K_\tau \leq 100$; c.f. fig. 2). All other hyper-parameters are the same.

Our ablation study indicates that the richer the task manifold and the more accurate the gradient path length is approximated, the better Leap performs (fig. 5). Further, adding a stabilizer has the intended effect and leads to significantly faster convergence. The simplest configuration, defined in terms of the gradient path energy and with the task manifold identifies as parameter space, yields a meta gradient equivalent to the update rule used in Reptile. We find this configuration to be less efficient

in terms of convergence and we observe a significant deterioration in performance. Extending the task manifold to the loss surface does not improve meta-training convergence speed, but does cut prediction error in half. Adding the stabilizer significantly speeds up convergence. These conclusions also hold under the gradient path length as distance measure, and in general using the gradient path length does better than using the gradient path energy as the distance measure.

F Experiment details: Omniglot

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

Method No. Pretraining tasks	Leap	Finetuning†	Reptile	MAML*	No pretraining
1	7.8	7.5	8.4	8.8	17.0
2	6.5	6.3	8.0	8.8	16.8
4	3.8	4.7	7.2	8.6	17.9
6	3.1	4.4	6.3	8.4	16.1
8	2.8	4.2	6.1	8.7	15.9
10	2.7	4.0	5.7	9.2	16.1
12	2.5	3.6	5.4	8.6	16.2
14	2.4	3.8	5.2	8.6	16.6
16	2.5	3.9	5.1	9.1	16.7
18	2.2	3.7	5.1	8.7	15.9
20	2.4	3.9	4.7	9.4	16.6
22	2.1	3.7	4.9	8.6	16.2
24	1.9	3.5	5.0	8.7	15.9
26	2.3	3.3	4.5	9.2	16.9
28	2.0	3.8	4.8	9.3	17.0
30	1.8	3.6	4.6	8.7	16.1

Table 2: Summary of hyper-parameters for Omniglot. "Meta" refers to the outer training loop, 'task" refers to the inner training loop.

Method Hyper-parameter	Leap	Finetuning	Reptile	MAML*	No pretraining
Meta learning rate	0.1	—	0.1	0.1	—
Task learning rate	0.1	0.01	0.1	0.1	0.1
Meta training max steps	5000	5000	5000	5000	—
Task training max steps	1000	1000	1000	1000	—
Meta batch size	20	20	20	20	—
Task batch size	20	20	20	20	—

Omniglot contains 50 alphabets, each with a set of characters that in turn have 20 unique samples. We treat each alphabet as a distinct task and pretrain on up to 30 alphabets, holding up to 10 out for validation and 10 for final evaluation. We use the same convolutional neural network architecture as in previous works [26, 8, 23]. This model stacks a module, comprised of a 3×3 convolution with 64 filters, followed by batch-normalization, ReLU activation and 2×2 max-pooling, four times. All images are downsampled to 28×28, resulting in a $1 \times 1 \times 64$ feature map that is passed on to a final linear layer. We define a task as a 20-class classification problem with classes drawn from a distinct alphabet. For alphabets with more than 20 characters, we pick 20 characters at random, while alphabets with fewer characters (4) are dropped from the validation set (used for early stopping of meta training). On each task, we train a model using stochastic gradient descent. For each model, we evaluated learning rates in the range $[0.001, 0.01, 0.1, 0.5]$; see table 2 for hyper-parameters.

Table 3: 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. ‡Our implementation.

Held-out task	Method	Test (%)	Train (%)	AUC†
Facescrub	Leap	19.9	0.0	11.6
	Finetuning	32.7	0.0	13.2
	Progressive Nets‡	**18.0**	0.0	**8.9**
	HAT‡	25.6	0.1	14.6
	No pretraining	18.2	0.0	10.5
NotMNIST	Leap	**5.3**	**0.6**	**2.9**
	Finetuning	5.4	2.0	4.4
	Progressive Nets‡	5.4	3.1	3.7
	HAT‡	6.0	2.8	5.4
	No pretraining	5.4	2.6	5.1
MNIST	Leap	**0.7**	0.1	**0.6**
	Finetuning	0.9	0.1	0.8
	Progressive Nets‡	0.8	**0.0**	0.7
	HAT‡	0.8	0.3	1.2
	No pretraining	0.9	0.2	1.0
Fashion MNIST	Leap	**8.0**	4.2	**6.8**
	Finetuning	8.9	**3.8**	7.0
	Progressive Nets‡	8.7	5.4	9.2
	HAT‡	9.5	5.5	8.1
	No pretraining	8.4	4.7	7.8
Cifar10	Leap	**21.2**	**10.8**	**17.5**
	Finetuning	27.4	13.3	20.7
	Progressive Nets‡	24.2	15.2	24.0
	HAT‡	27.7	21.2	27.3
	No pretraining	26.2	13.1	23.0
SVHN	Leap	**8.4**	**5.6**	**7.5**
	Finetuning	10.9	6.1	10.5
	Progressive Nets‡	10.1	6.3	13.8
	HAT‡	10.5	5.7	8.5
	No pretraining	10.3	6.9	11.5
Cifar100	Leap	**52.0**	**30.5**	**43.4**
	Finetuning	59.2	31.5	44.1
	Progressive Nets‡	55.7	42.1	54.6
	HAT‡	62.0	49.8	58.4
	No pretraining	54.8	33.1	50.1
Traffic Signs	Leap	**2.9**	0.0	**1.2**
	Finetuning	5.7	0.0	1.7
	Progressive Nets‡	3.6	0.0	4.0
	HAT‡	5.4	0.0	2.3
	No pretraining	3.6	0.0	2.4

To avoid unnecessary computation while evaluating each model across different number of pretraining tasks and seeds we use early stopping in both the task training loop and the meta training loop. For task training, we stop when we reach close to perfect validation performance: each task has 20 characters, each of which has 20 samples. For each character, we hold 5 samples out to create a task validation set. We stop training when validation error is below 2% on 30 batches in a row. We ran preliminary experiments to ensure this stopping criteria had no effect on final test performance. For the meta training loop, the validation set consists of held-out tasks. Meta training is stopped when the

Table 4: Summary of hyper-parameters for Multi-CV. "Meta" refers to the outer training loop, 'task" refers to the inner training loop. †During pretraining / on held-out task.

Method Hyper-parameter	Leap	Finetune	Progressive Nets	HAT	No pretraining
Meta learning rate	0.01	—	—	—	—
Task learning rate	0.1	0.1	0.1	0.1	0.1
Meta training max steps	1000	1000	1000	1000	—
Task training max epochs†	10/100	1/100	1/100	1/100	—/100
Meta batch size	10	10	10	10	—
Task batch size	32	32	32	32	—

average validation error on these validation tasks has not improved for 10 consecutive meta gradient steps. To increase task complexity, we augment all datasets with 36 uniformly spaced rotations.

G Experiment details: Multi-CV

We allow different architectures between tasks by using different final linear layers for each task. We use the same convolutional encoder as in the Omniglot experiment (appendix F). Leap learns an initialization for the convolutional encoder; on each task, the final linear layer is always randomly initialized. We compare Leap against (a) a baseline with no pretraining, (b) multitask finetuning, (c) HAT [24], and (d) Progressive Nets [20]. For HAT, we use the original formulation, but allow multiple task revisits (until convergence). For Progressive Nets, we allow lateral connections between all tasks and multiple task revisits (until convergence). Note that this makes Progressive Nets over 8 times larger in terms of learnable parameters than the other models. inproceedings We train using stochastic gradient descent with cosine annealing [13]. During meta training, we sample a batch of 10 tasks at random from the pretraining set and train until the early stopping criterion is triggered or the maximum amount of epochs is reached (see table 4). We used the same interval for selecting learning rates as in the Omniglot experiment (appendix F). Only Leap benefited from using more than 1 epoch as the upper limit on task training steps during pretraining. In the case of Leap, the initialization is updated after all tasks in the meta batch has been trained to convergence; for other models, there is no distinction between initialization and task parameters. On a given task, training is stopped if the maximum number of epochs is reached (table 4) or if the validation error fails to improve over 10 consecutive gradient steps. Similarly, meta training is stopped once the mean validation error fails to improve over 10 consecutive meta training batches. We use Adam [10] for the meta gradient update with a constant learning rate of 0.01. We use no dataset augmentation; MNIST images are zero padded to have 32×32 images; we use the same normalizations as [24].

H Experiment details: Atari

We use the same network as in [16], adopting it to actor-critic algorithms by estimating both value function and policy through linear layers connected to the final output of a shared convolutional network. Following standard practice, we use downsampled $84 \times 84 \times 3$ RGB images as input. Leap is applied with respect to the convolutional encoder (as final linear layers vary in size across environments). We use all environments with an action space of at most 10 as our pretraining pool, holding out Breakout and SpaceInvaders. During meta training, we sample a batch of 16 games at random from a pretraining pool of 27 games. On each game in the batch, a network is initialized using the shared initialization and trained independently for 5 million steps, accumulating the meta gradient across games on the fly. Consequently, the baseline and Leap differs only with respect to the initialization of the convolutional encoder. We trained Leap for 100 steps. The meta learned initialization was evaluated on the held-out games, a random selection of games seen during pretraining, and a random selection of games with action spaces larger than 10. 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. We use a discounting factor of 0.99.