Appendix

A. Additional details of RNN architecture

A.1. Shortcut connection

Since we expect $\mathbf{m}_{ts}^n$ to be the primary driver of update step direction, and in order to further reduce the information which must be stored in the Parameter RNN hidden state, we included a meta-trainable linear projection from the average rescaled gradients $\mathbf{m}_{ts}^n$ and the update directions $\Delta\theta_t^n$ and $\Delta\phi_t^n$.

B. Additional details of meta-training process

B.1. Heavy-tailed distribution over training steps

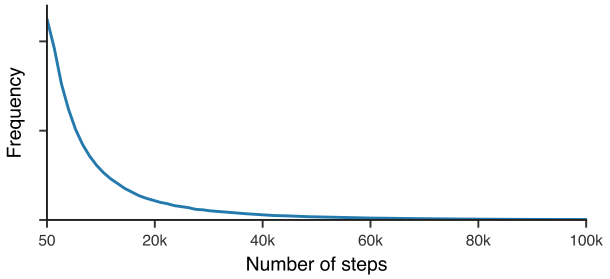

Figure App.1. A histogram of the total number of training iterations run on target problems during meta-training. The total number of unrolls is drawn from an exponential distribution with scale 50 plus a constant offset of 1. The number of training iterations within each unroll is drawn from an exponential distribution with scale 200 and a constant offset of 50.

C. Architecture updates

The Inception V3 experiment in Figure 4b used a slightly newer version of the learned optimizer codebase. The changes were:

C.1. Parameter noise during training

Due to the use of small meta-training problems in Section 4.1, during meta-training the learned optimizer is often able to optimize the problem almost exactly early in the unrolled optimization, after which the meta-loss s becomes relatively uninformative. In order to better simulate tasks which take many steps to optimize, small Gaussian noise is added to the parameters during each optimization step. This effectively moves the loss landscape underneath the optimizer, providing a more informative learning signal after many unrolls, and forcing the learned optimizer to be robust to a new type of noise. Specifically, the parameter update becomes

$$\theta_t^{n+1} = \theta_t^n + \Delta\theta_t^n + \alpha\tilde{\mathbf{n}}^t \tag{15}$$

$$\tilde{\mathbf{n}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{16}$$

where the noise scale α is drawn from a log uniform distribution between 10^{-10} and 10^{-2} for each problem.

C.2. Momentum from *previous* timescale

In Equation 3 we scale the average gradients $\bar{\mathbf{g}}_{ts}^n$ by a running estimate $\sqrt{\lambda_{ts}^n}$ of the root-mean-square magnitude of $\bar{\mathbf{g}}_{ts}^n$. This is a mismatch with Adam, where the average gradient is scaled by a running estimate of the root-mean-square magnitude of the *non-averaged* gradients. In order to be consistent with this, and in order to encourage better use of the dynamic range of $\mathbf{m}_{ts}^n$ (as defined in the text body, it spends much of its time with values near 1 or -1), we modify Equation 3 to normalize the average gradient $\bar{\mathbf{g}}_{ts}^n$ by $\sqrt{\lambda_{ts}^n}$ from the immediately faster timescale,

$$\mathbf{m}_{ts}^n = \frac{\bar{\mathbf{g}}_{ts}^n}{\sqrt{\lambda_{t(s-1)}^n}}, \tag{17}$$

and where we define the average gradient at the fastest time scale to be the raw gradient, $\bar{\mathbf{g}}_{t(-1)}^n = \mathbf{g}_t^n$

C.3. No normalization of step length

In order to simplify interactions between parameters, we no longer force a normalization of the parameter and attention update directions $\mathbf{d}_{\theta t}^n$ and $\mathbf{d}_{\phi t}^n$. We *do* still decompose the update into the product of a learning rate and a step. Since the attended update direction is now able to take on a different magnitude, the separate attention log learning rate η_ϕ^n is no longer required, and is eliminated. Equations 5 and 6 thus become

$$\Delta\theta_t^n = \exp\left(\eta_{\theta t}^n\right)\mathbf{d}_{\theta t}^n, \tag{18}$$

$$\Delta\phi_t^n = \exp\left(\eta_{\theta t}^n\right)\mathbf{d}_{\phi t}^n. \tag{19}$$

C.4. More stable meta-training hyper-parameters

The distribution over meta-loss gradients is observed to be assymmetrical and heavy tailed. This combination is known to cause biased parameter updates in RMSProp and Adam, since both optimizers underweight the contribution from extremely rare extremely large gradients. In order to reduce this tendency, we updated the mean-quare-gradient momentum term γ to be 0.999, rather than 0.9 in the meta-optimizer RMSProp (Section 4.4).