A. Additional Experiment Details

In this section, we provide additional details of the experimental set-up and hyperparameters.

A.1. Classification

For N-way, K-shot classification, each gradient is computed using a batch size of NK examples. For Omniglot, the 5-way convolutional and non-convolutional MAML models were each trained with 1 gradient step with step size $\alpha = 0.4$ and a meta batch-size of 32 tasks. The network was evaluated using 3 gradient steps with the same step size $\alpha = 0.4$. The 20-way convolutional MAML model was trained and evaluated with 5 gradient steps with step size $\alpha = 0.1$. During training, the meta batch-size was set to 16 tasks. For MiniImagenet, both models were trained using 5 gradient steps of size $\alpha = 0.01$, and evaluated using 10 gradient steps at test time. Following Ravi & Larochelle (2017), 15 examples per class were used for evaluating the post-update meta-gradient. We used a meta batch-size of 4 and 2 tasks for 1-shot and 5-shot training respectively. All models were trained for 60000 iterations on a single NVIDIA Pascal Titan X GPU.

A.2. Reinforcement Learning

In all reinforcement learning experiments, the MAML policy was trained using a single gradient step with $\alpha = 0.1$. During evaluation, we found that halving the learning rate after the first gradient step produced superior performance. Thus, the step size during adaptation was set to $\alpha = 0.1$ for the first step, and $\alpha = 0.05$ for all future steps. The step sizes for the baseline methods were manually tuned for each domain. In the 2D navigation, we used a meta batch size of 20; in the locomotion problems, we used a meta batch size of 40 tasks. The MAML models were trained for up to 500 meta-iterations, and the model with the best average return during training was used for evaluation. For the ant goal velocity task, we added a positive reward bonus at each timestep to prevent the ant from ending the episode.

B. Additional Sinusoid Results

In Figure 6, we show the full quantitative results of the MAML model trained on 10-shot learning and evaluated on 5-shot, 10-shot, and 20-shot. In Figure 7, we show the qualitative performance of MAML and the pretrained baseline on randomly sampled sinusoids.

C. Additional Comparisons

In this section, we include more thorough evaluations of our approach, including additional multi-task baselines and a comparison representative of the approach of Rei (2015).

C.1. Multi-task baselines

The pretraining baseline in the main text trained a single network on all tasks, which we referred to as "pretraining on all tasks". To evaluate the model, as with MAML, we fine-tuned this model on each test task using K examples. In the domains that we study, different tasks involve different output values for the same input. As a result, by pre-training on all tasks, the model would learn to output the average output for a particular input value. In some instances, this model may learn very little about the actual domain, and instead learn about the range of the output space.

We experimented with a multi-task method to provide a point of comparison, where instead of averaging in the output space, we averaged in the parameter space. To achieve averaging in parameter space, we sequentially trained 500 separate models on 500 tasks drawn from $p(\mathcal{T})$. Each model was initialized randomly and trained on a large amount of data from its assigned task. We then took the average parameter vector across models and fine-tuned on 5 datapoints with a tuned step size. All of our experiments for this method were on the sinusoid task because of computational requirements. The error of the individual regressors was low: less than 0.02 on their respective sine waves.

We tried three variants of this set-up. During training of the individual regressors, we tried using one of the following: no regularization, standard ℓ_2 weight decay, and ℓ_2 weight regularization to the mean parameter vector thus far of the trained regressors. The latter two variants encourage the individual models to find parsimonious solutions. When using regularization, we set the magnitude of the regularization to be as high as possible without significantly deterring performance. In our results, we refer to this approach as "multi-task". As seen in the results in Table 2, we find averaging in the parameter space (multi-task) performed worse than averaging in the output space (pre-training on all tasks). This suggests that it is difficult to find parsimonious solutions to multiple tasks when training on tasks separately, and that MAML is learning a solution that is more sophisticated than the mean optimal parameter vector.

C.2. Context vector adaptation

Rei (2015) developed a method which learns a context vector that can be adapted online, with an application to recurrent language models. The parameters in this context vector are learned and adapted in the same way as the parameters in the MAML model. To provide a comparison to using such a context vector for meta-learning problems, we concatenated a set of free parameters $\mathbf{z}$ to the input $\mathbf{x}$, and only allowed the gradient steps to modify $\mathbf{z}$, rather than modifying the model parameters θ, as in MAML. For im-

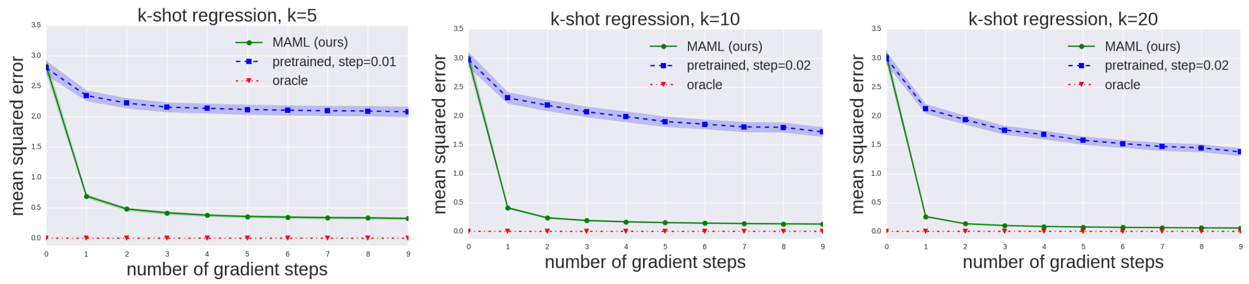

Figure 6. Quantitative sinusoid regression results showing test-time learning curves with varying numbers of K test-time samples. Each gradient step is computed using the same K examples. Note that MAML continues to improve with additional gradient steps without overfitting to the extremely small dataset during meta-testing, and achieves a loss that is substantially lower than the baseline fine-tuning approach.

Table 2. Additional multi-task baselines on the sinusoid regression domain, showing 5-shot mean squared error. The results suggest that MAML is learning a solution more sophisticated than the mean optimal parameter vector.

num. grad steps	1	5	10
multi-task, no reg	4.19	3.85	3.69
multi-task, l2 reg	7.18	5.69	5.60
multi-task, reg to mean θ	2.91	2.72	2.71
pretrain on all tasks	2.41	2.23	2.19
MAML (ours)	**0.67**	**0.38**	**0.35**

Table 3. 5-way Omniglot Classification

	1-shot	5-shot
context vector	$94.9 \pm 0.9\%$	$97.7 \pm 0.3\%$
MAML	$\mathbf{98.7} \pm \mathbf{0.4}\%$	$\mathbf{99.9} \pm \mathbf{0.1}\%$

age inputs, $\mathbf{z}$ was concatenated channel-wise with the input image. We ran this method on Omniglot and two RL domains following the same experimental protocol. We report the results in Tables 3, 4, and 5. Learning an adaptable context vector performed well on the toy pointmass problem, but sub-par on more difficult problems, likely due to a less flexible meta-optimization.

Table 4. 2D Pointmass, average return

num. grad steps	0	1	2	3
context vector	-42.42	-13.90	-5.17	$\mathbf{-3.18}$
MAML (ours)	-40.41	$\mathbf{-11.68}$	$\mathbf{-3.33}$	-3.23

Table 5. Half-cheetah forward/backward, average return

num. grad steps	0	1	2	3
context vector	-40.49	-44.08	-38.27	-42.50
MAML (ours)	-50.69	**293.19**	**313.48**	**315.65**

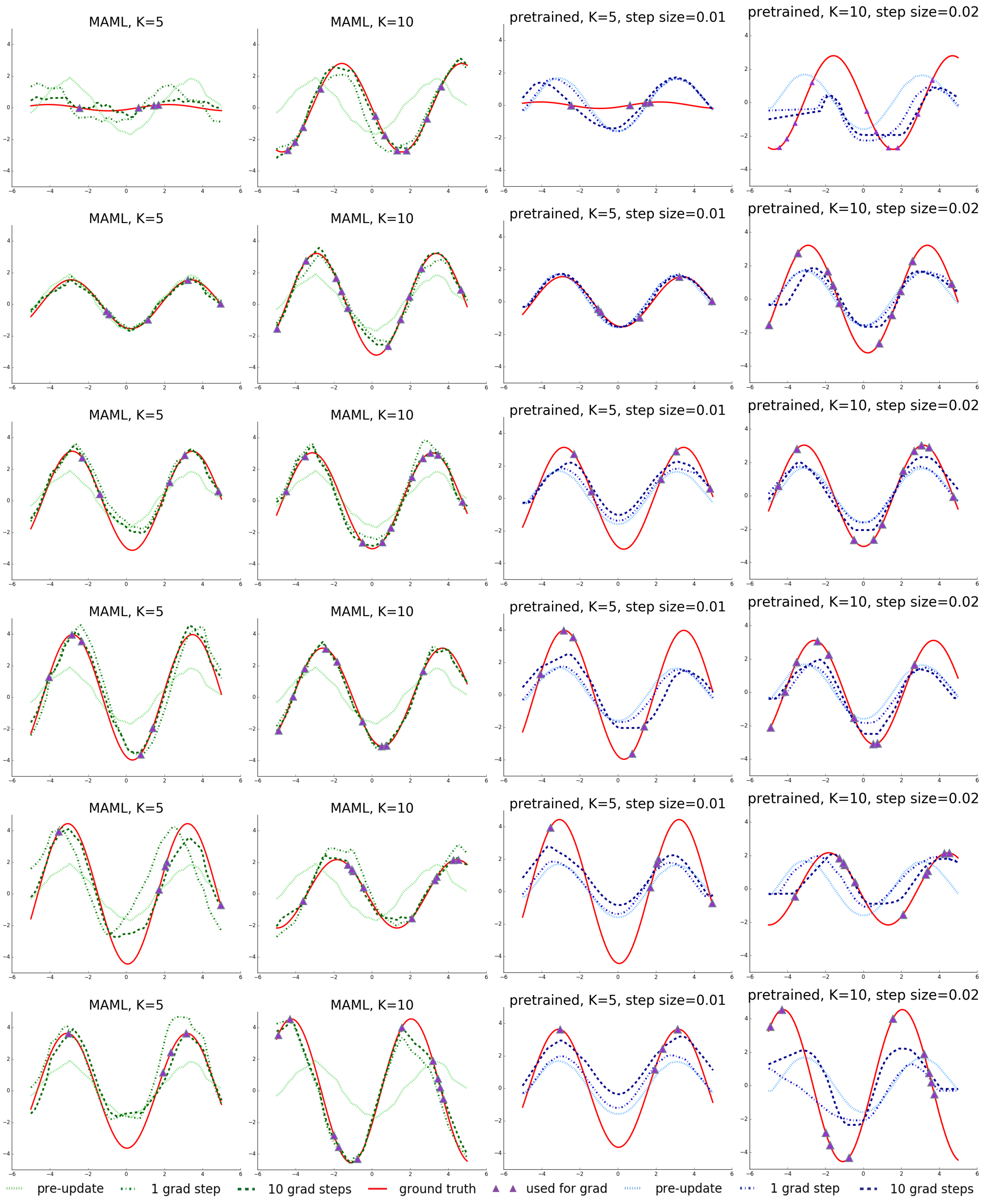

Figure 7. A random sample of qualitative results from the sinusoid regression task.