Fast Context Adaptation via Meta-Learning

Luisa Zintgraf [1] **Kyriacos Shiarlis** [1 2] **Vitaly Kurin** [1 2] **Katja Hofmann** [3] **Shimon Whiteson** [1 2]

Abstract

We propose CAVIA for meta-learning, a simple extension to MAML that is less prone to meta-overfitting, easier to parallelise, and more interpretable. CAVIA partitions the model parameters into two parts: *context parameters* that serve as additional input to the model and are adapted on individual tasks, and *shared parameters* that are meta-trained and shared across tasks. At test time, only the context parameters are updated, leading to a low-dimensional task representation. We show empirically that CAVIA outperforms MAML for regression, classification, and reinforcement learning. Our experiments also highlight weaknesses in current benchmarks, in that the amount of adaptation needed in some cases is small.

1. Introduction

The challenge of fast adaptation in machine learning is to learn on previously unseen tasks *fast* and with *little data*. In principle, this can be achieved by leveraging knowledge obtained in other, related tasks. However, the best way to do so remains an open question. We are interested in the meta-learning approach to fast adaptation, i.e., *learning how to learn* on unseen problems/datasets within few shots.

One approach for fast adaptation is to use gradient-based methods: at test time, only one or a few gradient update steps are performed to solve the new task, using a task-specific loss function. *Model agnostic meta learning* (MAML) (Finn et al., 2017a) is a general and powerful gradient-based meta-learning algorithm, which learns a model initialisation that allows fast adaptation at test time. Given that MAML is model-agnostic, it can be used with any gradient-based learning algorithm, and a variety of methods build on it (a.o., Lee & Choi (2018); Li et al. (2017); Al-Shedivat et al. (2018); Grant et al. (2018)).

MAML is trained with an interleaved training procedure, comprised of inner loop and outer loop updates that operate on a batch of related tasks at each iteration. In the inner loop, MAML learns task-specific network parameters by performing one gradient step on a task-specific loss. Then, in the outer loop, the model parameters from *before* the inner loop update are updated to reduce the loss *after* the inner loop update on the individual tasks. Hence, MAML learns a model initialisation that can generalise to a new task after only a few gradient updates at test time. One drawback of MAML is meta-overfitting: since the entire network is updated on just a few data points at test time, it can easily overfit (Mishra et al., 2018).

In this paper, we propose an alternative to MAML which is more interpretable and less prone to overfitting, without compromising performance. Our method, *fast **c**ontext **a**daptation **via** meta-learning* (CAVIA) learns a single model that adapts to a new task via gradient descent by updating only a set of input parameters at test time, instead of the entire network. These inputs, which we call *context parameters* ϕ (see Figure 1), can be interpreted as a task embedding that modulates the behaviour of the model. We confirm empirically that the learned context parameters indeed match the latent task structure. Like MAML, our method is model-agnostic, i.e., it can be applied to any model that is trained via gradient descent.

CAVIA is trained with an interleaved training procedure similar to MAML: in the inner loop only the context parameters ϕ are updated, and in the outer loop the rest of the model parameters, θ, are updated (which requires backpropagating through the inner-loop update). This allows CAVIA to explicitly optimise the task-independent parameters θ for good performance across tasks, while ensuring that the task-specific parameters ϕ can quickly adapt to new tasks at test time. alternative updates matter

The separation of parameters into task-specific and task-independent parts has several advantages. First, the size of both components can be chosen appropriately for the task. The network parameters θ can be made expressive enough without overfitting to a single task in the inner loop, which MAML is prone to. Furthermore, for many practical problems we have prior knowledge of which aspects vary across tasks and hence how much capacity ϕ should have.

[1]University of Oxford [2]Latent Logic [3]Microsoft Research. Correspondence to: Luisa Zintgraf <luisa.zintgraf@cs.ox.ac.uk>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Second, CAVIA is significantly easier to parallelise compared to MAML: learning task-specific context parameters for a batch of tasks can be parallelised in the inner loop. Other benefits are that parameter copies are not necessary which saves memory writes; we do not need to manually access and perform operations on the network weights and biases to set up the computation graphs; and CAVIA can help distributed machine learning systems, where the same model is deployed to different machines and we wish to learn different contexts concurrently.

CAVIA is conceptually related to embedding-based approaches for fast adaptation such as *conditional neural processes* (CNPs) (Garnelo et al., 2018) and *meta-learning with latent embedding optimisation* (LEO) (Rusu et al., 2019). These share the benefit of learning a low-dimensional representation of the task, which has the potential to lead to greater interpretability compared to MAML. In contrast to existing methods, CAVIA uses the same network to learn the embedding (during a backward pass) and make predictions (during a forward pass). Therefore CAVIA has fewer parameters to train, but must compute higher-order gradients during training.

Our experiments show that CAVIA outperforms MAML and CNPs on regression problems, can outperform MAML on a challenging classification benchmark by scaling up the network without overfitting, and outperforms MAML in a reinforcement learning setting while adapting significantly fewer parameters. We also show that CAVIA is robust to hyperparameters and demonstrate that the context parameters represent meaningful embeddings of tasks. Our experiments also highlight a weakness in current benchmarks in meta-learning, in that the amount of adaptation needed is small in some cases, confirming that task inference and multitask learning are enough to do well.

2. Background

Our goal is to learn models that can quickly adapt to new tasks with little data. Hence, learning on the new task is preceded by meta-learning on a set of related tasks.

2.1. Problem Setting

In few-shot learning problems, we are given distributions over training tasks $p_{\text{train}}(\mathcal{T})$ and test tasks $p_{\text{test}}(\mathcal{T})$. Training tasks can be used to learn how to adapt fast to any of the tasks with little per-task data, and evaluation is then done on (previously unseen) test tasks. Unless stated otherwise, we assume that $p_{\text{train}} = p_{\text{test}}$ and refer to both as p. Tasks in p typically share some structure, so that transferring knowledge between tasks can speed up learning. During each meta-training iteration, a batch of N tasks $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^N$ is sampled from p.

Supervised Learning. Supervised learning learns a model $f : x \mapsto \hat{y}$ that maps data points $x \in \mathcal{X}$ that have a true label $y \in \mathcal{Y}$ to predictions $\hat{y} \in \mathcal{Y}$. A task $\mathcal{T}_i = (\mathcal{X}, \mathcal{Y}, \mathcal{L}, q)$ is a tuple where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the output space, $\mathcal{L}(y, \hat{y})$ is a task-specific loss function, and $q(x, y)$ is a distribution over labelled data points. We assume that all data points are drawn i.i.d. from q. Different tasks can be created by changing any element of $\mathcal{T}_i$.

Training in supervised meta-learning proceeds over meta-training iterations, where for each $\mathcal{T}_i \in \mathbf{T}$, we sample two datasets $\mathcal{D}_i^{\text{train}}$ and $\mathcal{D}_i^{\text{test}}$ from $q_{\mathcal{T}_i}$:

$$\mathcal{D}_i^{\text{train}} = \{(x,y)^{i,m}\}_{m=1}^{M_i^{\text{train}}}, \quad \mathcal{D}_i^{\text{test}} = \{(x,y)^{i,m}\}_{m=1}^{M_i^{\text{test}}}, \quad (1)$$

where $(x, y) \sim q_{\mathcal{T}_i}$ and M_i^{train} and M_i^{test} are the number of training and test datapoints. The training data is used to update f, and the test data is then used to evaluate how good this update was, and adjust f or the update rule accordingly.

Reinforcement Learning. Reinforcement learning (RL) learns a policy π that maps states $s \in \mathcal{S}$ to actions $a \in \mathcal{A}$. Each task corresponds to a *Markov decision process* (MDP): a tuple $\mathcal{T}_i = (\mathcal{S}, \mathcal{A}, r, q, q_0)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $r(s_t, a_t, s_{t+1})$ is a reward function, $q(s_{t+1}|s_t, a_t)$ is a transition function, and $q_0(s_0)$ is an initial state distribution. The goal is to maximise the expected cumulative reward $\mathcal{J}$ under π,

$$\mathcal{J}(\pi) = \mathbb{E}_{q_0,q,\pi}\left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t, s_{t+1})\right], \quad (2)$$

where $H \in \mathbb{N}$ is the horizon and $\gamma \in [0, 1]$ is the discount factor. During each meta-training iteration, for each $\mathcal{T}_i \in \mathbf{T}$, we first collect a trajectory

$$\tau_i^{\text{train}} = \{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{M_i^{\text{train}}-1}, a_{M_i^{\text{train}}-1}, r_{M_i^{\text{train}}-1}, s_{M_i^{\text{train}}}\},$$

where the initial state s_0 is sampled from q_0, the actions are chosen by the current policy π, the state transitions according to q, and M_i^{train} is the number of environment interactions. We unify several episodes in this formulation: if the horizon H is reached within the trajectory, the environment is reset using q_0. Once the trajectory is collected, this data is used to update the policy. Another trajectory τ_i^{test} is then collected by rolling out the updated policy for M_i^{test} time steps. This test trajectory is used to evaluate the quality of the update on that task, and to adjust π or the update rule accordingly.

Evaluation for both supervised and reinforcement learning problems is done on a new (unseen) set of tasks drawn from p. For each such task, the model is updated using $\mathcal{L}$ or $\mathcal{J}$ and only a few data points ($\mathcal{D}^{\text{train}}$ or τ^{train}). Performance of the updated model is reported on $\mathcal{D}^{\text{test}}$ or τ^{test}.

2.2. Model-Agnostic Meta-Learning

One method for few-shot learning is *model-agnostic meta-learning* (Finn et al., 2017a, MAML). MAML learns an initialisation for the parameters θ of a model f_θ such that, given a new task, a good model for that task can be learned with only a small number of gradient steps and data points. In the inner loop, MAML computes new task-specific parameters θ_i (starting from θ) via one gradient update[1],

$$\theta_i = \theta - \alpha \nabla_\theta \frac{1}{M^i_{\text{train}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_\theta(x), y) \; . \tag{3}$$

For the meta-update in the outer loop, the *original* model parameters θ are then updated with respect to the performance after the inner-loop update, i.e.,

$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i\in\mathbf{T}} \frac{1}{M^i_{\text{test}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}(x), y) \; . \tag{4}$$

The result of training is a model initialisation θ that can be adapted with just a few gradient steps to any new task drawn from p. Since the gradient is taken with respect to the parameters θ before the inner-loop update (3), the outer-loop update (4) involves higher order derivatives of θ.

3. CAVIA

We propose *fast **c**ontext **a**daptation **via** meta-learning* (CAVIA), which partitions the model parameters into two parts: context parameters ϕ are adapted in the inner loop for each task, and parameters θ are meta-learned in the outer loop and shared across tasks.

3.1. Supervised Learning

At every meta-training iteration and for the current batch $\mathbf{T}$ of tasks, we use the training data $\mathcal{D}_i^{\text{train}}$ of each task $\mathcal{T}_i \in \mathbf{T}$ as follows. Starting from a fixed value ϕ_0 (we typically choose $\phi_0 = \mathbf{0}$; see Section 3.4), we learn task-specific parameters ϕ_i via one gradient update:

$$\phi_i = \phi_0 - \alpha \nabla_\phi \frac{1}{M_i^{\text{train}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_0,\theta}(x), y). \tag{5}$$

While we only take the gradient with respect to ϕ, the updated parameter ϕ_i is also a function of θ, since during backpropagation, gradients flow through the model. Given updated parameters ϕ_i for all sampled tasks, we proceed to the meta-learning step, in which θ is updated:

$$\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i\in\mathbf{T}} \frac{1}{M_i^{\text{test}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i,\theta}(x), y) \; . \tag{6}$$

[1]We outline is MAML for one gradient update step and the supervised learning setting, but it can be used with several gradient update steps and for reinforcement learning problems as well.

This update includes higher order gradients in θ due to the dependency on (5). At test time, *only* the context parameters are updated using Equation (5), and θ is held fixed.

3.2. Reinforcement Learning

During each iteration, for a current batch of MDPs $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^N$, we proceed as follows. Given ϕ_0, we collect a rollout τ_i^{train} by executing the policy $\pi_{\phi_0,\theta}$. We then compute task-specific parameters ϕ_i via one gradient update:

$$\phi_i = \phi_0 + \alpha \nabla_\phi \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{train}}, \pi_{\phi_0,\theta}), \tag{7}$$

where $\tilde{\mathcal{J}}(\tau, \pi)$ is the objective function of any gradient-based reinforcement learning method that uses trajectories τ produced by a parameterised policy π to update that policy's parameters. After updating the policy, we collect another trajectory τ_i^{test} to evaluate the updated policy, where actions are chosen according to the updated policy $\pi_{\phi_i,\theta}$.

After doing this for all tasks in $\mathbf{T}$, the meta-update step updates θ to maximise the average performance across tasks (after individually updating ϕ for them),

$$\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{N} \sum_{\text{MDP}_i\in\mathbf{T}} \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{test}}, \pi_{\phi_i,\theta}). \tag{8}$$

This update includes higher order gradients in θ due to the dependency on (7).

3.3. Conditioning on Context Parameters

Since ϕ is independent of the network input, we need to decide where and how to condition the network on them. For an output node $h_i^{(l)}$ at a fully connected layer l, we can for example simply concatenate ϕ to the inputs of that layer:

$$h_i^{(l)} = g\left(\sum_{j=1}^{J} \theta_{j,i}^{(l,h)}\, h_j^{(l-1)} + \sum_{k=1}^{K} \theta_{k,i}^{(l,\phi)}\, \phi_{0,k} + b\right), \tag{9}$$

where g is a nonlinear activation function, b is a bias parameter, $\theta_{j,i}^{(l,h)}$ are the weights associated with layer input $h_j^{(l-1)}$, and $\theta_{k,i}^{(l,\phi)}$ are the weights associated with the context parameter $\phi_{0,k}$. This is illustrated in Figure 1. In our experiments, for fully connected networks, we add the context parameter at the first layer, i.e., concatenate them to the input. Other conditioning methods can be used with CAVIA as well: e.g., for convolutional networks, we use *feature-wise linear modulation* FiLM (Perez et al., 2017), which performs an affine transformation on the feature maps. Given context parameters ϕ and a convolutional layer that outputs M feature maps $\{h_i\}_{i=1}^M$, FiLM linearly transforms each feature map $FiLM(h_i) = \gamma_i h_i + \beta$, where $\gamma, \beta \in \mathbb{R}^M$ are a function of the context parameters. We use a fully connected layer $[\gamma, \beta] = \sum_{k=1}^{K} \theta_{k,i}^{(l,\phi)} \phi_{0,k} + b$ with the identity function at the output.

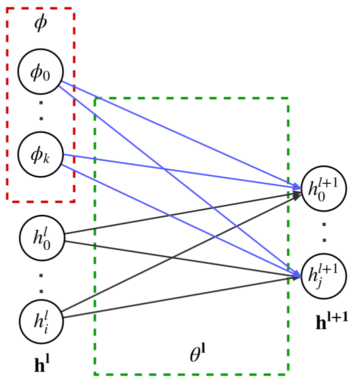


Figure 1. **Context adaptation.** A network layer h^l is augmented with additional context parameters ϕ (red) initialised to 0 before each adaptation step and updated by gradient descent during each inner loop and at test time. Network parameters θ (green) are only updated in the outer loop and shared across tasks. Hence, they stay fixed at test time. By initialising ϕ to 0, the network parameters associated with the context parameters (blue) do not affect the output of the layer before adaptation. After the first adaptation step they modulate the rest of the network to solve the new task.

3.4. Context Parameter Initialisation

When learning a new task, ϕ have to be initialised to some value, ϕ_0. We argue that, instead of meta-learning this initialisation as well, a fixed ϕ_0 is sufficient: in (9), if both $\theta_{j,i}^{(l,\phi)}$ and ϕ_0 are meta-learned, the learned initialisation of ϕ can be subsumed into the bias parameter b, and ϕ_0 can be set to a fixed value. Hence, the initialisation of the context parameters does not have to be meta-learned and parameter copies are not required during training. In our implementation we set the initial context parameter to a vector filled with zeros, $\phi_0 = \mathbf{0} = [0, \ldots, 0]^\top$.

Furthermore, not updating the context parameters ϕ in the outer loop allows for a more flexible and expressive gradient in the inner loop. Consequently, CAVIA is more robust to the inner loop learning rate, α in (5). Before an inner loop update, the part of the model associated with ϕ does not affect the output (since they are inputs and initialised at 0). During the inner update, only ϕ changes and can affect the output of the network at test time. Even if this update is large, the parameters $\theta_{k,i}^{(l,\phi)}$ that connect ϕ to the rest of the model (shown in blue in Figure 1), are automatically scaled during the outer loop. In other words, $\theta_{k,i}^{(l,\phi)}$ compensates in the outer loop for any excessively large inner loop update of ϕ. However, doing large gradient updates in every outer loop update step as well would lead to divergence and numerical overflow. In Section 5.1, we show empirically that the decoupling of learning ϕ and θ can indeed make CAVIA more robust to the initial learning rate compared to also learning the initialisation of the context parameters.

4. Related Work

One general approach to meta-learning is to learn the algorithm or update function itself (Schmidhuber, 1987; Bengio et al., 1992; Andrychowicz et al., 2016; Ravi & Larochelle, 2017). Another approach is gradient-based meta-learning, which learns a model initialisation such that at test time, a new task can be learned within a few gradient steps. Examples are MAML (Finn et al., 2017a) and its probabilistic variants (Grant et al., 2018; Yoon et al., 2018; Finn et al., 2018); REPTILE (Nichol & Schulman, 2018), which does not require second order gradient computation; and Meta-SGD (Li et al., 2017), which learns the per-parameter inner loop learning rate. The main difference to our work is that CAVIA adapts only a few parameters at test time, and these parameters determine only input context.

Closely related are MT-Nets (Lee & Choi, 2018), which learn *which* parameters to update in MAML. MT-Nets learn: an M-Net which is a mask indicating which parameters to update in the inner loop, sampled (from a learned probability distribution) for each new task; and a T-net which learns a task-specific update direction and step size. CAVIA is a simpler, more interpretable alternative where the task-specific and shared parameters are disjoint sets.

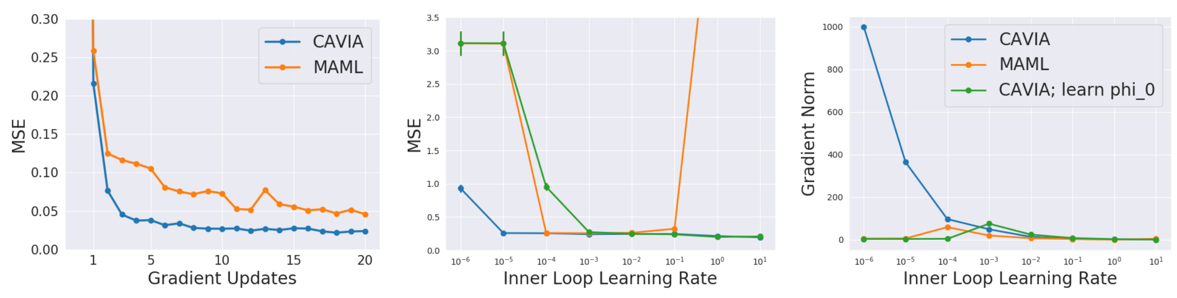
Additional input biases to MAML were considered by Finn et al. (2017b), who show that this improves performance on a robotic manipulation setting. By contrast, we update *only* the context parameters in the inner loop, and initialise them to 0 before adaptation to a new task. Rei (2015) propose a similar approach in the context of neural language models, where a context vector represents the sentence that is currently being processed (see also the Appendix of Finn et al. (2017a)). Unlike CAVIA, this approach updates context parameters in the outer loop, i.e., it learns the initialisation of ϕ. This coupling of the gradient updates leads to a less flexible meta-update and is not as robust to the inner loop learning rate like CAVIA, as we show empirically in 5.1.

Silver et al. (2008) proposed context features as a component of inductive transfer, using a predefined one-hot encoded task-specifying context as input to the network. They show that this works better than learning a shared feature extractor and having separate heads for all tasks. In this paper, we instead *learn* this contextual input from data of a new task. Such context features can also be learned by a separate embedding network as in, e.g., Oreshkin et al. (2018) and Garnelo et al. (2018), who use the task's training set to condition the prediction network. CAVIA instead learns the context parameters via backpropagation through the same network used to solve the task.

Several methods learn to produce network weights from task-specific embeddings or labelled datapoints (Gordon et al., 2018; Rusu et al., 2019), which then operate on the

	Number of Additional Input Parameters						
Method	0	1	2	3	4	5	50
CAVIA	-	$0.84(\pm 0.06)$	$0.21(\pm 0.02)$	$0.20(\pm 0.02)$	$\mathbf{0.19}(\pm 0.02)$	$\mathbf{0.19}(\pm 0.02)$	$\mathbf{0.19}(\pm 0.02)$
MAML	$0.33(\pm 0.02)$	$0.29(\pm 0.02)$	$0.24(\pm 0.02)$	$0.24(\pm 0.02)$	$\mathbf{0.23}(\pm 0.02)$	$\mathbf{0.23}(\pm 0.02)$	$\mathbf{0.23}(\pm 0.02)$

Table 1. Results for the sine curve regression task. Shown is the mean-squared error of CAVIA and MAML for varying number of input parameters, with 95% confidence intervals in brackets.



(a) Test Performance (b) Learning Rates (c) Gradient Norms

Figure 2. Analysis of the sine curve experiments. (a) Test performance after several gradient steps (on the same batch) averaged over 1000 unseen tasks. Both CAVIA and MAML continue to learn, but MAML performs worse and is less stable. (b) Test Performance after training with different inner loop learning rates. (c) CAVIA scales the model weights so that the inner learning rate is compensated by the context parameters gradients magnitude.

task-specific inputs. By contrast, we learn an embedding that modulates a fixed network, and is independent of the task-specific inputs during the forward pass. Specific to few-shot image classification, metric-based approaches learn to relate (embeddings of) labelled images and new instances of the same classes (Snell et al., 2017; Sung et al., 2018). By contrast, CAVIA can be used for regression and reinforcement problems as well. Other meta-learning methods are also motivated by the practical difficulties of learning in high-dimensional parameter spaces, and the relative ease of fast adaptation in lower dimensional space (e.g., Sæmundsson et al., 2018; Zhou et al., 2018).

In the context of reinforcement learning, Gupta et al. (2018) condition the policy on a latent random variable trained similarly to CAVIA, together with the reparametrisation trick (although they do not explicitly interpret these parameters as task embeddings). This latent variable is sampled once per episode, and thus allows for structured exploration. Unlike CAVIA, they adapt the entire network at test time, which can be prone to overfitting.

5. Experiments

In this section, we empirically evaluate CAVIA on regression, classification, and RL tasks. We show that: 1) adapting a small number of input parameters (instead of the entire network) is sufficient to yield performance equivalent to or better than MAML, 2) CAVIA is robust to the task-specific learning rate and scales well without overfitting, and 3) an embedding of the task emerges in the context parameters solely via backpropagation. Code is available at `https://github.com/lmzintgraf/cavia`.

5.1. Regression

5.1.1. Sine Curves

We start with the regression problem of fitting sine curves from Finn et al. (2017a). A task is defined by the amplitude and phase of the sine curve and generated by uniformly sampling the amplitude from $[0.1, 0.5]$ and the phase from $[0, \pi]$. For training, ten labelled datapoints (uniformly sampled from $x \in [-5, 5]$) are given for each task for the inner loop update, and we optimise a mean-squared error (MSE) loss. We use a neural network with two hidden layers and 40 nodes each. The number of context parameters varies between 2 and 50. Per meta-update we use a batch of 25 tasks. During testing we present the model with ten datapoints from 1000 newly sampled tasks and measure MSE over 100 test points.

To allow a fair comparison, we add additional input biases to MAML (the same number as context parameters that CAVIA uses), an extension that was also done by Finn et al. (2017b). These additional parameters are meta-learned together with the rest of the network.

Table 1 shows that CAVIA outperforms MAML even when MAML gets the same number of additional parameters, despite the fact that CAVIA adapts only 2-5 parameters,

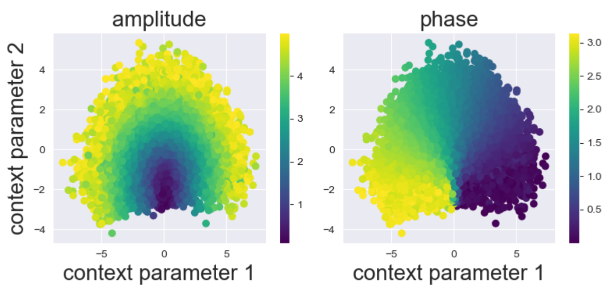


Figure 3. Visualisation of what the two context parameters learn on a new task. Shown is the value they take after 5 gradient update steps on a new task. Each dot is one random task; its colour indicates the amplitude (left) or phase (right) of that task.

instead of around 1600. CAVIA's performance on the regression task correlates with how many variables are needed to encode the tasks. In these experiments, two parameters vary between tasks, which is exactly the context parameter dimensionality at which CAVIA starts to perform well (the optimal encoding is three dimensional, as phase is periodic). This suggests CAVIA indeed learns task descriptions in the context parameters via backpropagation at test time. Figure 3 illustrates this by plotting the value of the learned inputs against the amplitude/phase of the task in the case of two context parameters. The model learns a smooth embedding in which interpolation between tasks is possible.

We also test how well CAVIA can continue learning at test time, when more gradient steps are performed than during training. Figure 2a shows that CAVIA outperforms MAML even after taking several gradient update steps and is more stable, as indicated by the monotonic learning curve.

As described in Section 3.4, CAVIA can scale the gradients of the context parameters since they are inputs to the model and trained separately. Figure 2b shows the performance of CAVIA, MAML, and CAVIA when also learning the initialisation of ϕ (i.e., updating the context parameters in the outer loop), for a varying learning rate from 10^{-6} to 10. CAVIA is robust to changes in learning rate while MAML performs well only in a small range. Figure 2c gives insight into how CAVIA does this: we plot the inner learning rate against the norm of the gradient of the context parameters at test time. The weights are adjusted so that lower learning rates bring about larger context parameter gradients and vice-versa. MT-Nets (Lee & Choi, 2018), which learn which subset of parameters to adapt on a new task, are also robust to the inner-loop learning rate, but in a smaller range than CAVIA.[2] Similarly, Li et al. (2017) show that MAML can be improved by learning a parameter-specific learning rate, which, however, introduces a lot of additional parameters.

[2]We do not show the numbers they report since we outperform them significantly, likely due to a different experimental protocol.

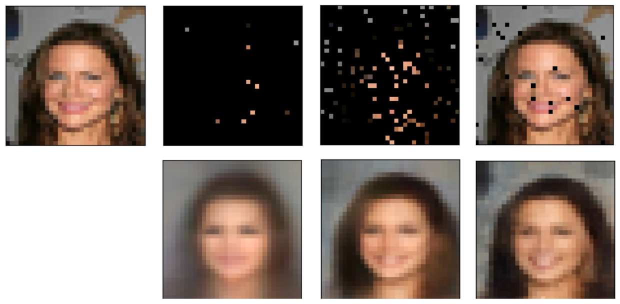


Figure 4. Image completion results on CelebA. Top row: true image on the left, and the training pixels for 10, 100, and 1000 training points. Bottom row: prediction of CAVIA when 128 context parameters were updated for 5 gradient steps.

	Random Pixels			Ordered Pixels		
	10	100	1000	10	100	1000
CNP*	0.039	0.016	0.009	0.057	**0.047**	0.021
MAML	0.040	0.017	**0.006**	0.055	**0.047**	0.007
CAVIA	**0.037**	**0.014**	**0.006**	**0.053**	**0.047**	**0.006**

Table 2. Pixel-wise MSE (for the entire image) for the image completion task on the CelebA data set. We test different number of available training points per image (10, 100, 1000). The trainig pixels are chosen either at random or ordered from the top-left corner to the bottom-right. (*Results from Garnelo et al. (2018))

5.1.2. Image Completion

To evaluate CAVIA on a more challenging regression task, we consider image completion (Garnelo et al., 2018). The task is to predict pixel values from coordinates, i.e., learn a function $f : [0,1]^2 \to [0,1]^3$ (for RGB values) which maps 2D pixel coordinates $x \in [0,1]^2$ to pixel intensities $y \in [0,1]^3$. An individual picture is considered a single task, and we are given a few pixels as a training set $\mathcal{D}^{\text{train}}$ and use the entire image as the test set $\mathcal{D}^{\text{test}}$ (including the training set). We train CAVIA on the CelebA (Liu et al., 2015) training set, perform model selection on the validation set, and evaluate on the test set.

Garnelo et al. (2018) use an MLP encoder with three hidden layers and 128 nodes each, a 128-dimensional embedding size, and a decoder with five hidden layers with 128 nodes each. To allow a fair comparison, we therefore choose a context vector of size 128, and use an MLP with five hidden layers (128 nodes each) for the main network. We chose an inner-learning rate of 1.0 without tuning. To train MAML, we use the same five-layer MLP network including 128 additional input biases, and an inner-loop learning rate of 0.1 (other tested learning rates: 1.0, 0.01). Both CAVIA and MAML were trained with five inner-loop gradient updates.

Table 2 shows the results in terms of pixel-wise MSE for different numbers of training pixels ($k = 10, 100, 1000$ shot),

	5-way accuracy	
Method	1-shot	5-shot
Matching Nets (Vinyals et al., 2016)	46.6%	60.0%
Meta LSTM (Ravi & Larochelle, 2017)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
Prototypical Networks (Snell et al., 2017)	$46.61 \pm 0.78\%$	$65.77 \pm 0.70\%$
Meta-SGD (Li et al., 2017)	$50.47 \pm 1.87\%$	$64.03 \pm 0.94\%$
REPTILE (Nichol & Schulman, 2018)	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$
MT-NET (Lee & Choi, 2018)	$\mathbf{51.70} \pm 1.84\%$	-
VERSA (Gordon et al., 2018)	$\mathbf{53.40} \pm 1.82\%$	$\mathbf{67.37} \pm 0.86$
MAML (32) (Finn et al., 2017a)	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
MAML (64)	$44.70 \pm 1.69\%$	$61.87 \pm 0.93\%$
CAVIA (32)	$47.24 \pm 0.65\%$	$59.05 \pm 0.54\%$
CAVIA (128)	$49.84 \pm 0.68\%$	$64.63 \pm 0.54\%$
CAVIA (512)	$\mathbf{51.82} \pm 0.65\%$	$65.85 \pm 0.55\%$
CAVIA (512, first order)	$49.92 \pm 0.68\%$	$63.59 \pm 0.57\%$

Table 3. Few-shot classification results on the Mini-Imagenet test set (average accuracy with 95% confidence intervals on a random set of 1000 tasks). For MAML, we show the results reported by Finn et al. (2017a), and when using a larger network (results obtained with the author's open sourced code and unchanged hyperparameters except the number of filters). These results show that CAVIA is able to scale to larger networks without overfitting, and outperforms MAML by doing so. We also include other CNN-based methods with similar experimental protocol. Note however that we did not tune CAVIA to compete with these methods, but focus on the comparison to MAML in this experiment.

and for the case of randomly selected pixels and ordered pixels (i.e., selecting pixels starting from the top left of the image). CAVIA outperforms CNPs and MAML in most settings. Figure 4 shows an example image reconstruction produced by CAVIA (see Appendix C.3 for more results).

These results show that it is possible to learn an embedding only via backpropagation and with far fewer parameters than when using a separate embedding network.

5.2. Classification

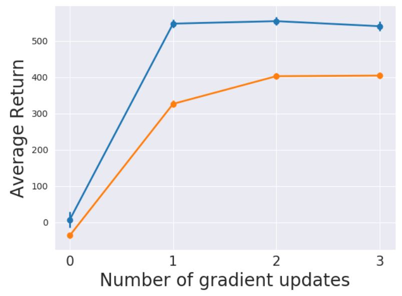
To evaluate how well CAVIA can scale to problems that require larger networks, we test it on the few-shot image classification benchmark Mini-Imagenet (Ravi & Larochelle, 2017). In N-way K-shot classification, a task is a random selection of N classes, for each of which the model gets to see K examples. From these it must learn to classify unseen images from the N classes. The Mini-Imagenet dataset consists of 64 training classes, 12 validation classes, and 24 test classes. During training, we generate a task by selecting N classes at random from the 64 classes and training the model on K examples of each, i.e., a batch of $N \times K$ images. The meta-update is done on a set of unseen images of the same classes.

On this benchmark, MAML uses a network with four convolutional layers with 32 filters each and one fully connected layer at the output (Finn et al., 2017a). We use the same network architecture, but between 32 and 512 filters per layer. We use 100 context parameters and add a FiLM layer that conditions on these after the third convolutional layer and whose parameters are meta-learned with the rest of the network, i.e., they are part of θ. All our models were trained with two gradient steps in the inner loop and evaluated with two gradient steps. Following (Finn et al., 2017a), we ran each experiment for $60,000$ meta-iterations and selected the model with the highest validation accuracy for evaluation on the test set.

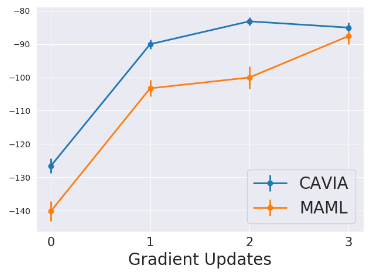
Table 3 shows our results on Mini-Imagenet held-out test data for 5-way 1-shot and 5-shot classification. Our smallest model (32 filters) underperforms MAML (within the confidence intervals), and our largest model (512 filters) clearly outperforms MAML. We also include results for the first order approximation of our largest models, where the gradient with respect to θ is not backpropagated through the inner loop update of the context parameters ϕ. As expected, this results in a lower accuracy (a drop of 2%) , but we still outperform MAML.

CAVIA benefits from increasing model expressiveness: since we only adapt the context parameters in the inner loop per task, we can substantially increase the network size without overfitting during the inner loop update. We tested scaling up MAML to a larger network size as well (see Table 3), but found that this hurt accuracy, which was also observed by Mishra et al. (2018).

Note that we focus on the comparison to MAML in these experiments, since our goal is to show that CAVIA can scale to larger networks without overfitting compared to MAML, and can be used out-of-the-box for classification problems as well. We did not tune CAVIA in terms of network architecture or other hyperparameters, but only varied the number of filters at each convolutional layer. The best performing



(a) Direction



(b) Velocity

Figure 5. Performance of CAVIA and MAML on the RL Cheetah experiments. Both agents were trained to perform one gradient update, but are evaluated for several update steps. Results are averaged over 40 randomly selected tasks.

method with similar architecture and experimental protocol is VERSA (Gordon et al., 2018), which learns to produce weights of the classifier, instead of modulating the network. The current state-of-the-art results in Mini-Imagenet is (to the best of our knowledge) the method LEO Rusu et al. (2019) who use pre-trained feature representations from a deep residual network (He et al., 2016) and a different experimental protocol. CAVIA can be used with such embeddings as well, and we expect an increase in performance.

In conclusion, CAVIA can achieve much higher accuracies than MAML by increasing the network size, without overfitting. Our results are obtained by adjusting only 100 parameters at test time (instead of $> 30,0000$ like MAML), which embed the five different classes of the current task.

5.3. Reinforcement Learning

To demonstrate the versatility of CAVIA, we also apply it to two high dimensional reinforcement learning MuJoCo (Todorov et al., 2012) tasks using the setup of Finn et al. (2017a). In the first experiment, a Cheetah robot must run in a particular, randomly chosen direction (forward/backward), and receives as reward its speed in that direction. In the second experiment, the Cheetah robot must run at a particular velocity, chosen uniformly at random between 0.0 and 2.0. The agent's reward is the negative absolute value between its current and the target velocity. Each rollout has a length of 200, and we use 20 rollouts per gradient step during training, and a meta-batchsize of 40 tasks per outer update. As in Finn et al. (2017a), our agents are trained for one gradient update, using policy gradient with generalised advantage estimation (Schulman et al., 2015b) in the inner loop and TRPO (Schulman et al., 2015a) in the outer loop update. Following the protocol of Finn et al. (2017a), both CAVIA and MAML were trained for up to 500 meta-iterations, and the models with the best average return during training were used for evaluation. For these tasks, we use 50 context parameters for CAVIA and an inner-loop learning rate of 10. We found that starting with a higher learning rate helps for RL problems, since the policy update in the outer loop has a stronger signal from the context parameters.

Figure 5 shows the performance of the CAVIA and MAML agents at test time, after up to three gradient steps (averaged over 40 randomly selected test tasks). Both models keep learning for several updates, although they were only trained for one update step. CAVIA outperforms MAML on both domains after one gradient update step, while updating only 50 parameters at test time per task compared to $> 10,000$. For the Cheetah Velocity experiment, MAML catches up after three gradient update steps.

6. Conclusion and Future Work

CAVIA is a meta-learning approach that separates the model into task-specific context parameters and parameters that are shared across tasks. We demonstrated experimentally that CAVIA is robust to the inner loop learning rate and yields task embeddings in the context parameters. CAVIA outperforms MAML on challenging regression, classification, and reinforcement learning problems, while adapting fewer parameters at test time and being less prone to overfitting.

We are interested in extending our experimental evaluation to settings with multi-modal task distributions, as well as settings where more generalisation beyond task identification is necessary at test time. One possible approach here is to combine CAVIA with MAML-style updates in the future: i.e., having two separate inner loops (producing ϕ_i and θ_i), and one outer loop.

We are interested in extending CAVIA to more challenging RL problems and exploring its role in allowing for smart exploration in order to identify the task at hand, for example building on the work of Gupta et al. (2018) who use probabilistic context variables, or Stadie et al. (2018), who propose E-MAML for RL problems, an extension for MAML which accounts for the effect of the initial sampling distribution (policy) before the inner-loop update.

Acknowledgements

We thank Frans Oliehoek, Wendelin Boehmer, Mark Finean and Joost van Amersfoort for useful discussions and feedback. We would also like to thank Chelsea Finn, Jackie Loong and Tristan Deleu for their open-sourced MAML implementations. The NVIDIA DGX-1 used for this research was donated by the NVIDIA corporation. L. Zintgraf is supported by the Microsoft Research PhD Scholarship Program. Vitaly Kurin is supported by the Samsung R&D Institute UK studentship. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). We thank the anonymous reviewers for their valuable feedback which helped us improve this paper.

References

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *Sixth International Conference on Learning Representations (ICLR 2018)*, 2018.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pp. 6–8. Univ. of Texas, 1992.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning, 2017*, 2017b.

Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018).*, 2018.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018).*, 2018.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. Meta-learning probabilistic inference for prediction. In *Seventh International Conference on Learning Representations (ICLR 2019)*, 2018.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. In *Sixth International Conference on Learning Representations (ICLR 2018)*, 2018.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Lee, Y. and Choi, S. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pp. 2933–2942, 2018.

Li, Z., Zhou, F., Chen, F., and Li, H. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *Sixth International Conference on Learning Representations (ICLR 2018)*, 2018.

Nichol, A. and Schulman, J. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

Oreshkin, B. N., Lacoste, A., and Rodriguez, P. Tadam: Task dependent adaptive metric for improved few-shot learning. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2017.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR), 2017*, 2017.

Rei, M. Online representation learning in recurrent neural language models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *Seventh International Conference on Learning Representations (ICLR 2019)*, 2019.

Sæmundsson, S., Hofmann, K., and Deisenroth, M. P. Meta reinforcement learning with latent variable gaussian processes. In *Conference on Uncertainty in Artificial Intelligence 2018*, 2018.

Schmidhuber, J. *Evolutionary Principles in Self-referential Learning: On Learning how to Learn: the Meta-meta-meta...-hook*. 1987.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *Fourth International Conference on Learning Representations (ICLR 2016)*, 2015b.

Silver, D. L., Poirier, R., and Currie, D. Inductive transfer with context-sensitive neural networks. *Machine Learning*, 73(3):313, 2008.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Stadie, B. C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, pp. 7343–7353, 2018.

Zhou, F., Wu, B., and Li, Z. Deep meta-learning: Learning to learn in the concept space. *arXiv preprint arXiv:1802.03596*, 2018.

Fast Context Adaptation via Meta-Learning

Supplementary Material

A. Pseudocode

Algorithm 1 CAVIA for Supervised Learning

```
Require: Distribution over tasks $p(\mathcal{T})$
Require: Step sizes $\alpha$ and $\beta$
Require: Initial model $f_{\phi_0,\theta}$ with $\theta$ initialised randomly
    and $\phi_0 = 0$
 1: while not done do
 2:     Sample batch of tasks $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^N$ where $\mathcal{T}_i \sim p$
 3:     for all $\mathcal{T}_i \in \mathbf{T}$ do
 4:         $\mathcal{D}_i^{\text{train}}, \mathcal{D}_i^{\text{test}} \sim q_{\mathcal{T}_i}$
 5:         $\phi_0 = 0$
 6:         $\phi_i = \phi_0 - \alpha \nabla_\phi \frac{1}{M_i^{\text{train}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_0,\theta}(x), y)$
 7:     end for
 8:     $\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i\in\mathbf{T}} \frac{1}{M_i^{\text{test}}} \sum_{(x,y)\in\mathcal{D}_i^{\text{test}}} \mathcal{L}_{\mathcal{T}_i}(f_{\phi_i,\theta}(x, y))$
 9: end while
```

Algorithm 2 CAVIA for RL

```
Require: Distribution over tasks $p(\mathcal{T})$
Require: Step sizes $\alpha$ and $\beta$
Require: Initial policy $\pi_{\phi_0,\theta}$ with $\theta$ initialised randomly
    and $\phi_0 = 0$
 1: while not done do
 2:     Sample batch of tasks $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^N$ where $\mathcal{T}_i \sim p$
 3:     for all $\mathcal{T}_i \in \mathbf{T}$ do
 4:         Collect rollout $\tau_i^{\text{train}}$ using $\pi_{\phi_0,\theta}$
 5:         $\phi_i = \phi_0 + \alpha \nabla_\phi \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{train}}, \pi_{\phi_0,\theta})$
 6:         Collect rollout $\tau_i^{\text{test}}$ using $\pi_{\phi_i,\theta}$
 7:     end for
 8:     $\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i\in\mathbf{T}} \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{test}}, \pi_{\phi_i,\theta})$
 9: end while
```

B. Practical Tips

B.1. Implementation

The context parameters ϕ can be added to any network, and do not require direct access to the rest of the network weights like MAML. In PyTorch this can be done as follows. To add CAVIA parameters to a network, it is necessary to first initialise them to zero when the model is initialised:

```
self.context_params =
torch.zeros(size=[self.num_context_params],
requires_grad=True)
```

Add a way to reset the context parameters to zero (e.g., a method that just does the above). During the forward pass, add the context parameters to the input by concatenating it (when using a fully connected network):

```
x = torch.cat((x,
self.context_params.expand(x.shape[0],
-1)), dim=1)
```

(This is for fully connected networks. We refer the reader to our implementation for how to use FiLM to condition CNNs.) To correctly set the computation graph for the outer loop, it is necessary to assign the context parameters manually with their gradient. In the inner loop, compute the gradient:

```
grad = torch.autograd.grad(task_loss,
model.context_params,
create_graph=True)[0]
```

The option *create_graph* will make sure that you can take the gradient of *grad* again. Then, update the context parameters using one gradient descent step

```
model.context_params = model.context_params
- lr_inner * grad
```

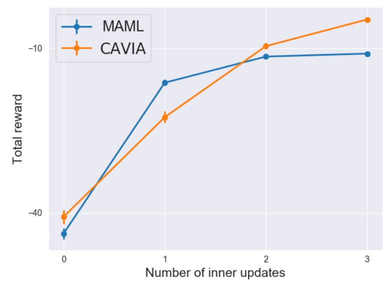
If you now do another forward pass and compute the gradient of the model parameters θ (for the outer loop), these will include higher order gradients because *grad* above includes gradients of θ, and because we kept the computation graph via the option *grad*. To see how to train CAVIA and aggregate the meta-gradient over several tasks, see our implementation at [blinded; see supplementary material].

B.2. Hyperparameter Selection

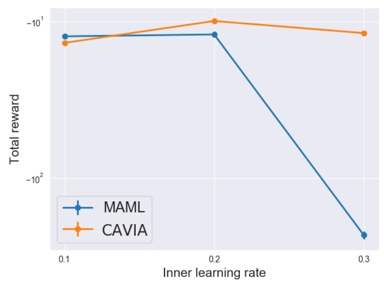
The choice of network architecture/size and context parameters can be guided by domain knowledge. E.g., for the few-shot image classification problem, an appropriate model is a deep convolutional model. For the context parameters, it is important to make sure they are not underparameterised. CAVIA can deal with larger than necessary context parameters (see Table 1), although it might start overfitting in the inner loop at some point (we have not experiences this in practise). Regarding learning rates, we always started with an inner loop learning rate of 1 and the Adam optimiser with the standard learning rate of 0.001 for the outer loop.

For CNNs, we found that adding the context parameters not at the input layer, but after several (in our case after the third out of four) convolutions works best. We believe this is because the lower-level features that the first convolutions extract are useful for any image classification task, and we only want our task embedding to influence the activations at the deeper layers. In our experiments we used a FiLM network with no hidden layers. We tried deeper versions, but this resulted in inferior performance.

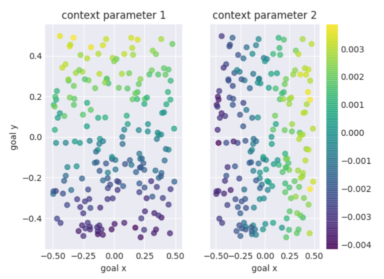
We also tested to add context parameters at several layers instead of only one. However, in our experience this resulted in similar (regression and RL) or worse (in the case of CNNs) performance.



(a) Multiple gradient updates.



(b) Learning rate comparison.



(c) Goal position embedding.

Figure 6. Results for the 2D navigation reinforcement learning problem.

C. Experiments

C.1. Classification: Details

For Mini-Imagenet, our model takes as input images of size $84 \times 84 \times 3$ and has 5 outputs, one for each class. The model has four modules that each consist of: a $2D$ convolution with a 3×3 kernel, padding 1 and 128 filters, a batch normalisation layer, a max-pooling operation with kernel size 2, if applicable a FiLM transformation (only at the third convolution, details below), and a ReLU activation function. The output size of these four blocks is $5 \times 5 \times 128$, which we flatten to a vector and feed into one fully connected layer. The FiLM layer itself is a fully connected layer with inputs ϕ and a 256-dimensional output and the identity function at the output. The output is divided into γ and β, each of dimension 128, which are used to transform the filters that the convolutional operation outputs. The context vector is of size 100 (other sizes tested: 50, 200) and is added after the third convolution (other versions tested: at the first, second or fourth convolution).

The network weights are initialised using He et al. (2015), the bias parameters are initialised to zero (except at the FiLM layer). We use the Adam optimiser for the meta-update step with an initial learning rate of 0.001. This learning rate is annealed every $5,000$ steps by multiplying it by 0.9. The inner learning rate is set to 0.1 (others tested: 1.0, 0.01). We use a meta batchsize of 4 and 2 tasks for 1-shot and 5-shot classification respectively. For the batch norm statistics, we always use the current batch – also during testing. I.e., for 5-way 1-shot classification the batch size at test time is 5, and we use this batch for normalisation.

C.2. Reinforcement Learning: Additional Experiments

We also perform reinforcement learning experiments on the simple 2D Navigation task of Finn et al. (2017a). The agent moves in a 2D world using continuous actions and at each timestep is given a negative reward proportional to its distance from a pre-defined goal position. Each task has a new unknown goal position.

We follow the same procedure as Finn et al. (2017a). Goals are sampled from an interval of $(x, y) = [-0.5, 0.5]$. At each step we sample 20 tasks for both the inner and outer loops and testing is performed on 40 new unseen tasks. We learn for 500 iterations and optimise for one gradient update in the inner loop. The best performing policy during training is then presented with new test tasks and allowed two gradient updates. For each update, the total reward over 20 rollouts per task is measured. We use a two-layer network with 100 units per layer and ReLU nonlinearities to represent the policy and a linear value function approximator. For CAVIA we use five context parameters at the input layer.

Figure 6a shows that the two methods are highly competitive. We think that the similar performance is mostly due to a ceiling effect, since the domain is relatively simple. Notably, CAVIA adapts only five parameters at test time, whereas MAML adapts around $10,000$. Figure 6b, which plots performance for several learning rates (at test time, after two gradient updates), shows that CAVIA is again less sensitive to the inner loop learning rate. Only when using a learning rate of 0.1 is MAML competitive in performance.[3]

As with regression, the optimal task embedding is low dimensional enough to plot. We therefore apply CAVIA with two context parameters and plot how these correlate with the actual position of the goal for 200 test tasks. Figure 6c shows that the context parameters obtained after two policy gradient updates represent a disentangled embedding of the actual task. Specifically, context parameter 1 encodes the y position of the goal, while context parameter 2 encodes the x position. Hence, CAVIA can learn compact interpretable task embeddings via backpropagation through the inner loss.

C.3. Additional CelebA Image Completion Results

The following images show additional results for the CelebA image completion task.

[3]For MAML we halve the learning rate after the first gradient update, following Finn et al. (2017a).

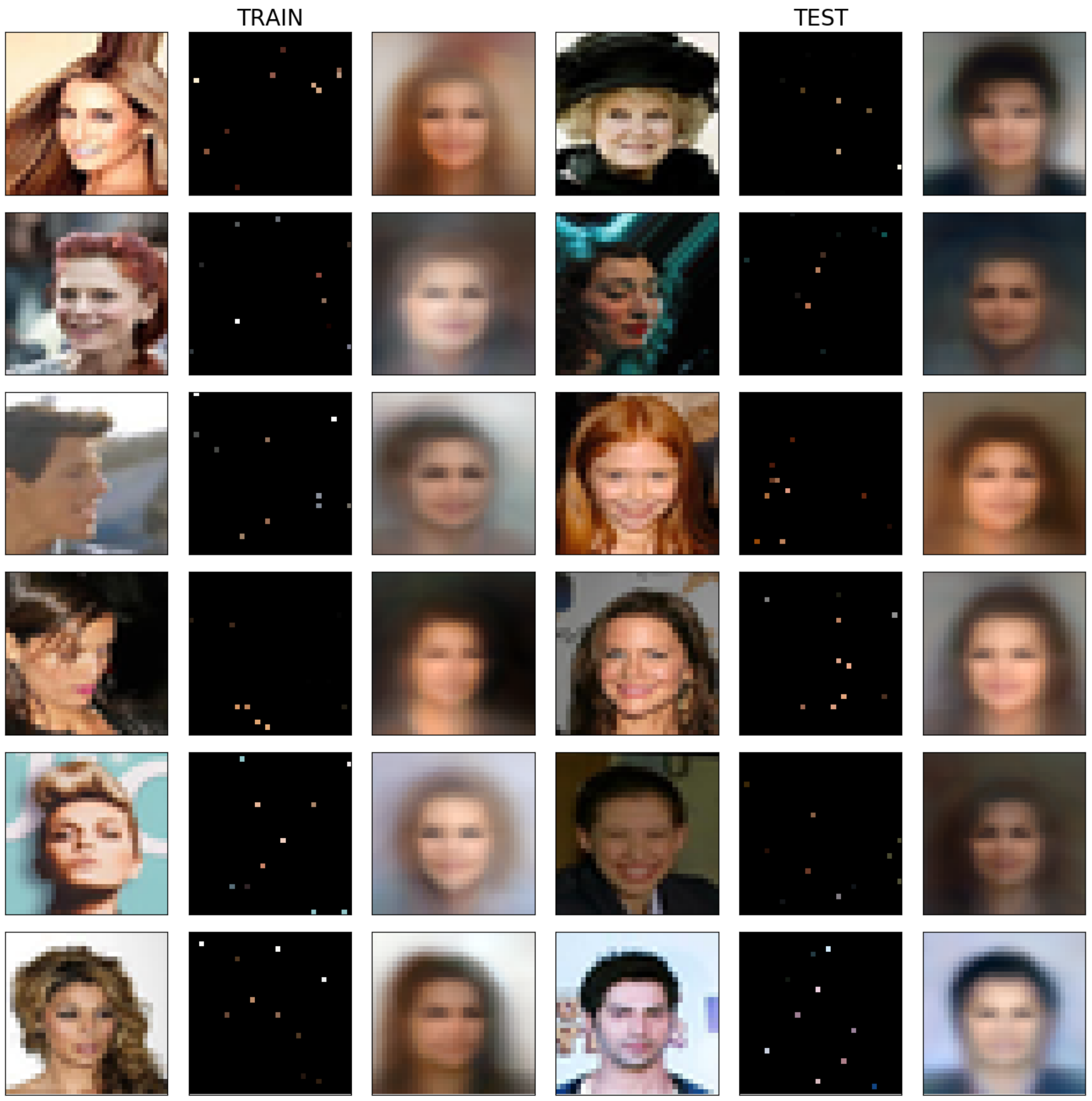


Figure 7. Additional image completion results for the CelebA image completion problem, when $k = 10$ pixels are given.

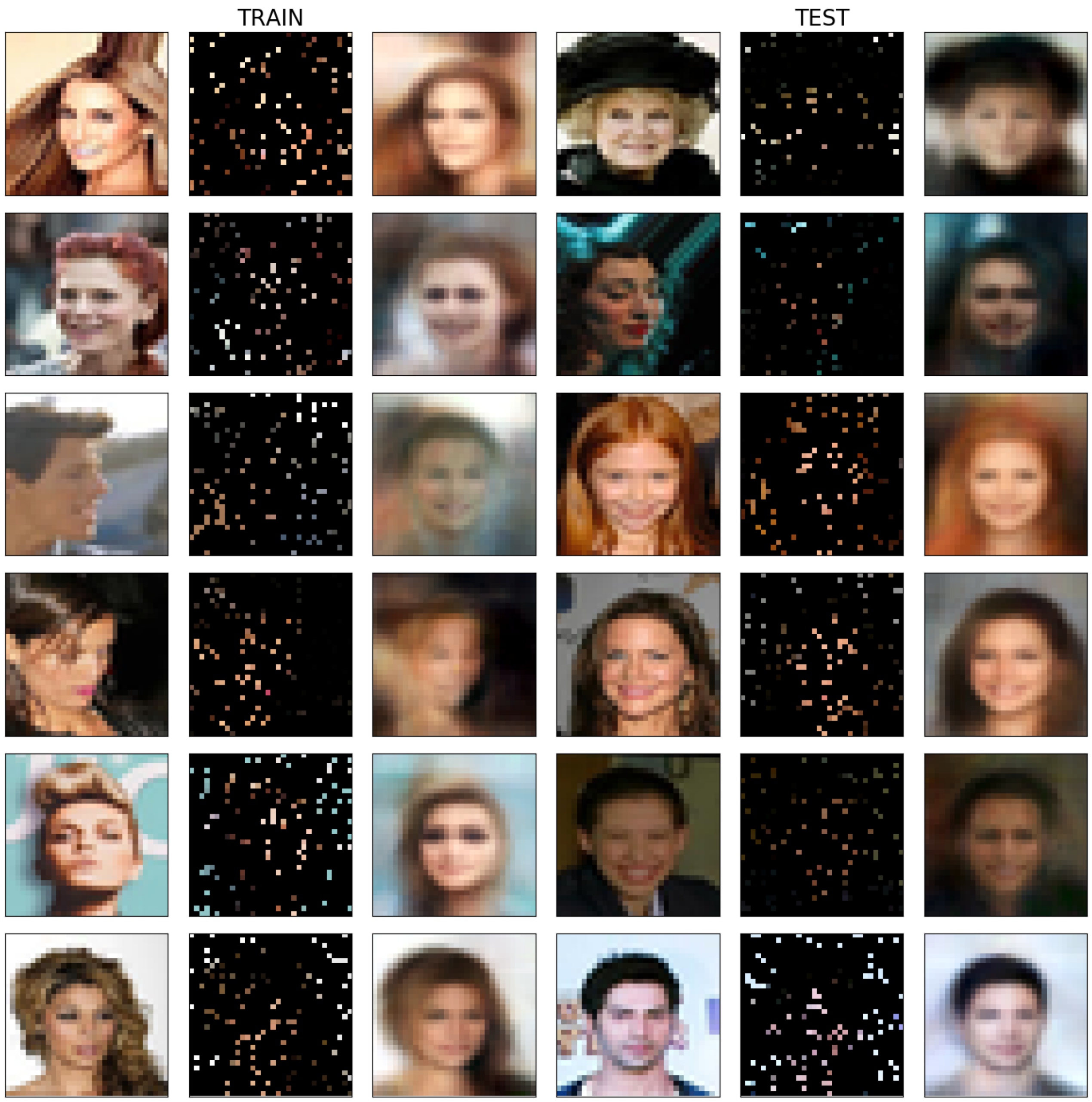


Figure 8. Additional image completion results for the CelebA image completion problem, when $k = 10$ pixels are given.

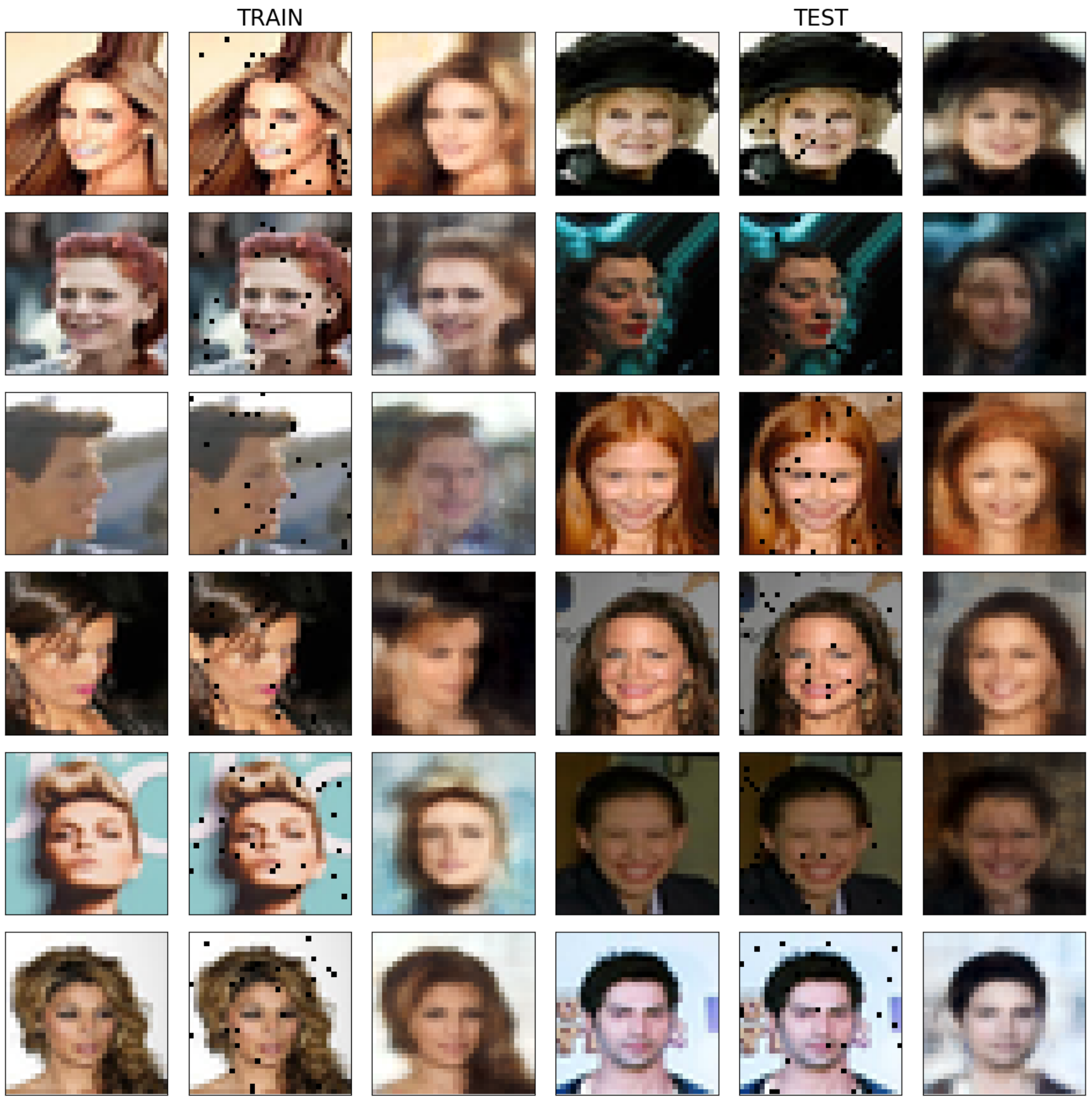


Figure 9. Additional image completion results for the CelebA image completion problem, when $k = 10$ pixels are given.