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 CA VIA for meta-learning, a simple extension to MAML that is less prone to meta-overfitting, easier to parallelise, and more interpretable. CA VIA 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 CA VIA 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)).

CA VIA 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 CA VIA 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.

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.

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 context adaptation via meta-learning (CA VIA) 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.
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 amount of adaptation needed is small and can 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 \( p_{\text{train}}(T) \) and test tasks \( p_{\text{test}}(T) \). 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 \( T = \{T_i\}_{i=1}^{N} \) 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 \( T_i = (\mathcal{X}, \mathcal{Y}, \mathcal{L}, q) \) 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 \( T_i \).

Training in supervised meta-learning proceeds over meta-training iterations, where for each \( T_i \) in \( T \), we sample two datasets \( D_{\text{train}}^i \) and \( D_{\text{test}}^i \) from \( q_{T_i} \):

\[
D_{\text{train}}^i = \{(x, y)^{i,m}_{m=1}\}, \quad D_{\text{test}}^i = \{(x, y)^{i,m}_{m=1}\}, \quad (1)
\]

where \((x, y) \sim q_{T_i}\) and \( M_{\text{train}}^i \) and \( M_{\text{test}}^i \) 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 \( \pi \) that maps states \( s \in \mathcal{S} \) to actions \( a \in \mathcal{A} \). Each task corresponds to a Markov decision process (MDP): a tuple \( T_i = (\mathcal{S}, \mathcal{A}, r, q, q_0) \), 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 \( J \) under \( \pi \),

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

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

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

where the initial state \( s_0 \) is sampled from \( q_0 \), the actions are chosen by the current policy \( \pi \), the state transitions according to \( q \), and \( M_i^{\text{train}} \) 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 \( \tau_i^{\text{test}} \) is then collected by rolling out the updated policy for \( M_i^{\text{test}} \) time steps. This test trajectory is used to evaluate the quality of the update on that task, and to adjust \( \pi \) 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 \( J \) and only a few data points (\( D_{\text{train}} \) or \( \tau_{\text{train}} \)). Performance of the updated model is reported on \( D_{\text{test}} \) or \( \tau_{\text{test}} \).
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 \( \theta \) of a model \( f_\theta \) 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 \( \theta_i \) (starting from \( \theta \)) via one gradient update\(^1\),

\[
\theta_i = \theta - \alpha \nabla_\theta \frac{1}{M_i} \sum_{(x,y) \in D_i} \mathcal{L}_{T_i}(f_\theta(x),y) .
\tag{3}
\]

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

\[
\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{T_i \in T} \frac{1}{M_i} \sum_{(x,y) \in D_i} \mathcal{L}_{T_i}(f_\theta(x),y) .
\tag{4}
\]

The result of training is a model initialisation \( \theta \) 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 \( \theta \) before the inner-loop update (3), the outer-loop update (4) involves higher order derivatives of \( \theta \).

3. CAVIA

We propose fast context adaptation via meta-learning (CA VIA), which partitions the model parameters into two parts: context parameters \( \phi \) are adapted in the inner loop for each task, and parameters \( \theta \) 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 \( T \) of tasks, we use the training data \( D_i^{\text{train}} \) of each task \( T_i \) as follows. Starting from a fixed value \( \phi_0 \) (we typically choose \( \phi_0 = 0 \); see Section 3.4), we learn task-specific parameters \( \phi_i \) via one gradient update:

\[
\phi_i = \phi_0 - \alpha \nabla_\phi \frac{1}{M_i} \sum_{(x,y) \in D_i} \mathcal{L}_{T_i}(f_{\phi_0,\theta}(x),y). \tag{5}
\]

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

\[
\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{N} \sum_{T_i \in T} \frac{1}{M_i} \sum_{(x,y) \in D_i} \mathcal{L}_{T_i}(f_{\phi_i,\theta}(x),y) . \tag{6}
\]

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

3.2. Reinforcement Learning

During each iteration, for a current batch of MDPs \( T = \{ T_i \}_{i=1}^N \), we proceed as follows. Given \( \phi_0 \), we collect a rollout \( r_i^{\text{train}} \) by executing the policy \( \pi_{\phi_0,\theta} \). We then compute task-specific parameters \( \phi_i \) via one gradient update:

\[
\phi_i = \phi_0 + \alpha \nabla_\phi \tilde{J}_{T_i}(r_i^{\text{train}}, \pi_{\phi_0,\theta}), \tag{7}
\]

where \( \tilde{J}(\tau, \pi) \) is the objective function of any gradient-based reinforcement learning method that uses trajectories \( \tau \) produced by a parameterised policy \( \pi \) to update that policy’s parameters. After updating the policy, we collect another trajectory \( r_i^{\text{test}} \) 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 \( T \), the meta-update step updates \( \theta \) to maximise the average performance across tasks (after individually updating \( \phi \) for them),

\[
\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{N} \sum_{T_i \in T} \tilde{J}_{T_i}(r_i^{\text{test}}, \pi_{\phi_i,\theta}). \tag{8}
\]

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

3.3. Conditioning on Context Parameters

Since \( \phi \) 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 \( \phi \) to the inputs of that layer:

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

where \( g \) is a nonlinear activation function, \( b \) is a bias parameter, \( \theta_j^{(l,h)} \) are the weights associated with layer input \( h_j^{(l-1)} \), and \( \theta_k^{(l,\phi)} \) 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 \( \phi \) and a convolutional layer that outputs \( M \) feature maps \( \{ h_i \}_{i=1}^M \), FiLM linearly transforms each feature map \( FiLM(h_i) = \gamma_i h_i + \beta_i \), where \( \gamma, \beta \in \mathbb{R}^M \) are a function of the context parameters. We use a fully connected layer

\[
[\gamma, \beta] = \sum_{k=1}^{K} \theta_k^{(l,\phi)} \phi_{0,k} + b
\]

with the identity function at the output.
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 \( \phi \). 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
CA VIA: Fast Context Adaptation via Meta-Learning

<table>
<thead>
<tr>
<th>Method</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAVIA</td>
<td>-</td>
<td>0.84(±0.06)</td>
<td>0.21(±0.02)</td>
<td>0.20(±0.02)</td>
<td><strong>0.19</strong>(±0.02)</td>
<td><strong>0.19</strong>(±0.02)</td>
<td><strong>0.19</strong>(±0.02)</td>
</tr>
<tr>
<td>MAML</td>
<td>0.33(±0.02)</td>
<td>0.29(±0.02)</td>
<td>0.24(±0.02)</td>
<td>0.24(±0.02)</td>
<td><strong>0.23</strong>(±0.02)</td>
<td><strong>0.23</strong>(±0.02)</td>
<td><strong>0.23</strong>(±0.02)</td>
</tr>
</tbody>
</table>

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.

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.

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, π]. 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 \( \phi \) (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.

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 \( f : [0, 1]^2 \rightarrow [0, 1]^3 \) (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 \( D^{train} \) and use the entire image as the test set \( D^{test} \) (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.

<table>
<thead>
<tr>
<th></th>
<th>Random Pixels</th>
<th>Ordered Pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>CNP</td>
<td>0.039</td>
<td>0.016</td>
</tr>
<tr>
<td>MAML</td>
<td>0.040</td>
<td>0.017</td>
</tr>
<tr>
<td>CAVIA</td>
<td><strong>0.037</strong></td>
<td><strong>0.014</strong></td>
</tr>
</tbody>
</table>

\( *\)Results from Garnelo et al. (2018)
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 ?? 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 $\theta$. 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 $\theta$ is not backpropagated through the inner loop update of the context parameters $\phi$. 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
CA VIA: Fast Context Adaptation via Meta-Learning

Figure 5. Performance of CA VIA 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. CA VIA can be used with such embeddings as well, and we expect an increase in performance.

In conclusion, CA VIA 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 > 30,000 like MAML), which embed the five different classes of the current task.

5.3. Reinforcement Learning

To demonstrate the versatility of CA VIA, 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 CA VIA 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 CA VIA 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 CA VIA 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. CA VIA 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

CA VIA 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 CA VIA is robust to the inner loop learning rate and yields task embeddings in the context parameters. CA VIA 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 CA VIA with MAML-style updates in the future: i.e., having two separate inner loops (producing $\phi_i$ and $\theta_i$), and one outer loop.

We are interested in extending CA VIA 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 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


