A Background

A.1 Meta-Learning

We present a unified perspective on meta-learning in the task-segmented setting, which allows straightforward presentation of the algorithms used in this work as well as generalization to the task-unsegmented case. The core idea of meta-learning is to directly optimize the few-shot learning performance of a machine learning model over a *distribution* of learning tasks, rather than just a single task, with the goal of this learning performance generalizing to other tasks from this distribution.

A meta-learning method consists of two phases: meta-training and online adaptation. Let $\boldsymbol{\theta}$ be the parameters of this model learned via meta-training. During online adaptation, the model uses context data $\mathcal{D}_t = (\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t})$ from within one task to compute statistics

$$\boldsymbol{\eta}_t = f_{\boldsymbol{\theta}}(\mathcal{D}_t) \tag{8}$$

where f is a function parameterized by $\boldsymbol{\theta}$. For example, in MAML (Finn et al., 2017), the statistics are the neural network weights after gradient updates computed using $\mathcal{D}_t$. In neural processes (Garnelo et al., 2018), the statistics are the aggregated context parameters computed via encoding and aggregating the context data. For recurrent network-based meta-learning algorithms, these statistics correspond to the hidden state of the network. For a simple nearest-neighbors model, $\boldsymbol{\eta}$ may simply be the context data. The model then performs predictions by using these statistics to define a conditional distribution on $\boldsymbol{y}$ given new inputs $\boldsymbol{x}$,

$$\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\eta}_t).$$

Adopting a Bayesian perspective, we refer to $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\eta}_t)$ as the posterior predictive distribution.

The performance of this model on this task can be evaluated by considering how well this posterior predictive distribution matches the true task data distribution,

$$\mathcal{L}(\mathcal{D}_t, \boldsymbol{\theta}) = D(p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{T}_i) \| p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}, f_{\boldsymbol{\theta}}(\mathcal{D}_t)))$$

where D is a measure of the dissimilarity of the two distributions, e.g. the KL divergence, for which this objective becomes standard negative log likelihood minimization.

Meta-learning optimizes the parameters $\boldsymbol{\theta}$ such that the model performs well across a distribution of tasks,

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathbb{E}_{\mathcal{D}_t \sim \mathcal{T}_i} \left[\mathcal{L}(\mathcal{D}_t, \boldsymbol{\theta}) \right] \right].$$

Across most meta-learning algorithms, including all of those referenced above, both the update rule and the prediction function are chosen to be differentiable operations, such that the parameters can be optimized via stochastic gradient descent. Given a dataset pre-segmented into groups of data from individual tasks, standard meta-learning algorithms operate via first sampling a group for which $\mathcal{T}$ is fixed, treating part of that group as the context data $\mathcal{D}_t$ and sampling from the remainder to obtain test points $(\boldsymbol{x}, \boldsymbol{y})$ from the same task. While this strategy can be very effective and produce expressive models that are capable of few-shot learning in complex domains, it relies on task segmentation which in many settings, especially continual learning, is not easily available.

A.2 Bayesian Online Changepoint Detection

To enable meta-learning without task segmentation, we extend prior work in changepoint detection. Specifically, we build on Bayesian online changepoint detection (Adams & MacKay, 2007), an approach for detecting changepoints (i.e. task switches) originally presented in a streaming unconditional density estimation context, which we review here.

BOCPD operates by maintaining a belief distribution over *run lengths*, i.e. how many of the past data points $\boldsymbol{y}_t$ correspond to the current task. At time t, run length of $r_t = \tau$ indicates that the task has switched τ timesteps ago, i.e. $\mathcal{D}_{-\tau} = \boldsymbol{y}_{t-\tau:t}$ are all drawn from a shared task $\mathcal{T}$. A belief that $r_t = 0$ implies that there has been a task switch, and that the current datapoint $\boldsymbol{y}_t$ was drawn from a new task $\mathcal{T}' \sim p(\mathcal{T})$. We denote this belief distribution at time t as $b_t(r_t) = p(r_t \mid \boldsymbol{y}_{1:t-1})$.

Given r_t, we know the past r_t data points all correspond to the same task, and thus the density $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}, r_t)$ corresponds to the posterior predictive density after conditioning on the past r_t data points. We can reason about the overall posterior predictive by marginalizing over the run length r_t according to $b_t(r_t)$,

$$p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}) = \sum_{r_t=0}^{t-1} p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}, r_t) b_t(r_t),$$

Algorithm 1 Meta-Learning via Online Changepoint Analysis: Training

Require: Training data $\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}$, number of training iterations N, initial model parameters $\boldsymbol{\theta}$

for $i = 1$ to N **do**
  Sample training batch $\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T}$ from the full timeseries.
  Initialize belief over run length $b_1(r_1 = 0) = 1$
  Initialize posterior statistics $\boldsymbol{\eta}_0[r = 0]$ according to $\boldsymbol{\theta}$
  for $t = 1$ to T **do**
    Observe $\boldsymbol{x}_t$
    Compute $b_t(r_t \mid \boldsymbol{x}_t)$ according to (2)
    Predict $p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}_t \mid \boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t-1})$ according to (6)
    Observe $\boldsymbol{y}_t$
    Incur NLL loss $\ell_t = -\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_t \mid \boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t-1})$
    Compute updated posteriors $\boldsymbol{\eta}_t[r_t]$ for all r_t according to (7)
    Compute $b_t(r_t \mid \boldsymbol{x}_t, \boldsymbol{y}_t)$ according to (3)
    Compute updated belief over run length b_{t+1} according to (5) and (4)
  end for
  Compute $\nabla_{\boldsymbol{\theta}} \sum_{t=k}^{k+T} \ell_t$ and perform gradient descent update to $\boldsymbol{\theta}$
end for

where $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}, r_t)$ is referred to as the *underlying predictive model* (UPM). BOCPD recursively computes posterior predictive densities for each value of $r_t \in \{0, \dots, t-1\}$, and then evaluates new datapoints $\boldsymbol{y}_{t+1}$ under these posterior predictive densities to update the belief distribution $b(r_t)$. In this work, we extend this approach of Adams & MacKay (2007) beyond Bayesian unconditional density estimation to apply to general meta-learning models operating in the conditional density estimation setting, and derive these update rules in more detail for our context.

B MOCA Details

B.1 Batch Training MOCA

In practice, we sample batches of length T from the full training time series, and train on these components. While this artificially increases the observed hazard rate (as a result of the initial belief over run length being 0 with probability 1), it substantially reduces the computational burden of training. Because MOCA maintains a posterior for each possible run length, computational requirements grow linearly with T. Iterating over the whole training time series without any hypothesis pruning can be prohibitively expensive. While a variety of different pruning methods within BOCPD have been proposed (Wilson et al., 2010; Saatci et al., 2010), we require a pruning method which does not break model differentiability. Note that at test-time, we no longer require differentiability and so previously developed pruning methods may be applied.

Empirically, we observe diminishing marginal returns when training on longer sequences. Fig. 3 shows the performance of MOCA for varying training sequence lengths (T). In all experiments presented in the body of the paper, we use $T = 100$. As discussed, small T values artificially inflate the observed hazard rate, so we expect to see performance improve with larger T values. Fig. 3 shows that this effect results in diminishing marginal returns, with little performance improvement beyond $T = 100$. Longer training sequences lead to increased computation per iteration (as MOCA is linear in the runlength), as well as an increased memory burden (especially during training, when the computation graph must be retained by automatic differentiation frameworks). Thus, we believe it is best to train on the shortest possible sequences, and propose $T = 1/\lambda$ (where λ is the hazard rate) as a rough rule of thumb.

C Making your MOCA: Model Instantiations

Thus far, we have presented MOCA at an abstract level, highlighting the fact that it can be used with any meta-learning model that admits the probabilistic interpretation as an underlying predictive model. However, there are several practical considerations in the choice of meta-learning algorithm which can influence the computational efficiency and overall performance of MOCA. For the experiments in this paper, we leverage two meta-learning algorithms which offer a clean Bayesian learning interpretation, relatively low-dimensional posterior statistics, recursive updates for these statistics, and computationally efficient likelihood evaluation under the posterior predictive. For regression experiments, we use ALPaCA (Harrison et al., 2018); for classification experiments, we use a novel algorithm based on similar Bayesian updates which we refer to as PCOC, for probabilistic clustering

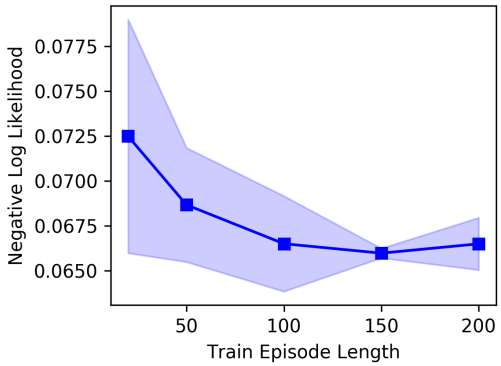


Figure 3: Performance versus the training horizon (T) for the sinusoid with hazard 0.01. The lowest hazard was used to increase the effects of the short training horizon. A minor decrease in performance is visible for very small training horizons (around 20), but flattens off around 100 and above. It is expected that these diminishing marginal returns will occur for all systems and hazard rates.

for online classification. For completeness, we offer a high level overview of these algorithms and show how they fit into the MOCA framework in the following subsections.

C.1 ALPaCA: Bayesian Meta-Learning for Regression

ALPaCA (Harrison et al., 2018) is a meta-learning approach for which the base learning model is Bayesian linear regression in a learned feature space $\boldsymbol{y} \mid \boldsymbol{x} \sim \mathcal{N}(K^T \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{w}), \Sigma_\epsilon)$ where $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{w})$ is a feed-forward neural network with weights $\boldsymbol{w}$ mapping inputs $\boldsymbol{x}$ to a n_ϕ-dimensional feature space. ALPaCA maintains a matrix-normal distribution over K, and thus, assuming Gaussian likelihood, results in a matrix-normal posterior distribution over K. This posterior inference may be performed exactly, and computed recursively. The matrix-normal distribution on the last layer results in a Gaussian posterior predictive density.

We fix the prior $K \sim \mathcal{MN}(\bar{K}_0, \Sigma_\epsilon, \Lambda_0^{-1})$. In this matrix-normal prior, $\bar{K}_0 \in \mathbb{R}^{n_\phi \times n_y}$ is the prior mean and Λ_0 is a $n_\phi \times n_\phi$ precision matrix (inverse of the covariance). Given this prior and data model, the posterior may be recursively computed as follows. First, we define $Q_t = \Lambda_t^{-1} \bar{K}_t$. Then, the one step posterior update is

$$\Lambda_{t+1}^{-1} = \Lambda_t^{-1} - \frac{(\Lambda_t^{-1}\boldsymbol{\phi}(\boldsymbol{x}_{t+1}))(\Lambda_t^{-1}\boldsymbol{\phi}(\boldsymbol{x}_{t+1}))^T}{1 + \boldsymbol{\phi}^T(\boldsymbol{x}_{t+1})\Lambda_t^{-1}\boldsymbol{\phi}(\boldsymbol{x}_{t+1})} \qquad Q_{t+1} = \boldsymbol{y}_{t+1}\boldsymbol{\phi}^T(\boldsymbol{x}_{t+1}) + Q_t \tag{9}$$

and the posterior predictive distribution is

$$p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}_{t+1} \mid \boldsymbol{x}_{1:t+1}, \boldsymbol{y}_{1:t}) = \mathcal{N}((\Lambda_t^{-1}Q_t)^T\boldsymbol{\phi}(\boldsymbol{x}_{t+1}), (1 + \boldsymbol{\phi}^T(\boldsymbol{x}_{t+1})\Lambda_t^{-1}\boldsymbol{\phi}(\boldsymbol{x}_{t+1}))\Sigma_\epsilon). \tag{10}$$

In summary, ALPaCA is a meta learning model for which the posterior statistics are $\boldsymbol{\eta}_t = \{Q_t, \Lambda_t^{-1}\}$, and the recursive update rule $h(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\eta})$ is given by (9). The parameters that are meta-learned are the prior statistics, the feature network weights, and the noise covariance: $\boldsymbol{\theta} = \{\bar{K}_0, \Lambda_0, \boldsymbol{w}, \Sigma_\epsilon\}$. Note that, as is typical in regression, ALPaCA only models the conditional density $p(\boldsymbol{y} \mid \boldsymbol{x})$, implicitly assuming that $p(\boldsymbol{x})$ is independent of the underlying task.

C.2 PCOC: Bayesian Meta-Learning for Classification

In the classification setting, one can obtain a similar Bayesian meta-learning algorithm by performing Gaussian discriminant analysis in a learned feature space. This is a novel approach to meta-learning for classification which we term probabilistic clustering for online classification (PCOC, pronounced "peacock"). We present a concise description of this algorithm here but defer to the appendix for a more detailed discussion.

In PCOC we process labeled input/class pairs $(\boldsymbol{x}_t, y_t)$ by encoding the input through an embedding network $\boldsymbol{z}_t = \boldsymbol{\phi}(\boldsymbol{x}_t; \boldsymbol{w})$, and performing Bayesian density estimation for every class. Specifically, we assume a Categorical-Gaussian generative model in this embedding space, and impose the conjugate Dirichlet prior over the class probabilities and a Gaussian prior over the mean for each class,

$$\begin{aligned} y_t &\sim \mathrm{Cat}(p_1, \ldots, p_{n_y}), & p_1, \ldots, p_{n_y} &\sim \mathrm{Dir}(\boldsymbol{\alpha}_0), \\ \boldsymbol{z}_t \mid y_t &\sim \mathcal{N}(\bar{z}_{y_t}, \Sigma_{\epsilon, y_t}), & \bar{z}_{y_t} &\sim \mathcal{N}(\mu_{y_t,0}, \Lambda_{y_t,0}^{-1}). \end{aligned}$$

Given labeled context data $(\boldsymbol{x}_t, y_t)$, the algorithm updates its belief over the Gaussian mean for the corresponding class, as well as its belief over the probability of each class. As with ALPaCA,

these posterior computations can be performed through closed form recursive updates. Defining $\boldsymbol{q}_{i,t} = \Lambda_{i,t}\boldsymbol{\mu}_{i,t}$, we have

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \mathbf{1}_{y_t} \qquad \boldsymbol{q}_{y_t,t} = \boldsymbol{q}_{y_t,t-1} + \Sigma_{\epsilon,y_t}\boldsymbol{\phi}(\boldsymbol{x}_t) \qquad \Lambda_{y_t,t} = \Lambda_{y_t,t-1} + \Sigma_{\epsilon,y_t}$$

where $\mathbf{1}_i$ denotes a one-hot vector with a one at index i. Terms not related to class y_t are left unchanged in this recursive update. Given this set of posterior parameters $\boldsymbol{\eta}_t = \{\boldsymbol{\alpha}_t, \boldsymbol{q}_{1:J,t}, \Lambda_{1:J,t}\}$, the posterior predictive density in the embedding space can be computed as

$$p(y) = \alpha_{y,t}/(\sum_{i=1}^{J}\alpha_{i,t}) \qquad p(\boldsymbol{z}, y) = p(y)\mathcal{N}(\boldsymbol{z}; \Lambda_{y,t}^{-1}\boldsymbol{q}_{y,t}, \Lambda_{y,t}^{-1} + \Sigma_{\epsilon,y})$$

where $\mathcal{N}(\boldsymbol{z}; \mu, \Sigma)$ denotes the Gaussian pdf with mean μ and covariance Σ evaluated at $\boldsymbol{z}$. Applying Bayes rule, the posterior predictive on y_{t+1} given $\boldsymbol{x}_{t+1}$ is

$$p(y_{t+1} = j \mid \boldsymbol{x}_{1:t+1}, \boldsymbol{y}_{1:t}) = \frac{p(\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x}_t), y = j)}{\sum_{i=1}^{J} p(\boldsymbol{z} = \boldsymbol{\phi}(\boldsymbol{x}_t), y = i)}.$$

This generative modeling approach also allows computing $p(\boldsymbol{z}_{t+1} \mid \boldsymbol{\eta}_t)$ by simply marginalizing out y from the joint density of $p(\boldsymbol{z}, y)$,

$$p(\boldsymbol{z}_{t+1} \mid \boldsymbol{\eta}_t) = \sum_{y=1}^{J} p(y)\mathcal{N}(\boldsymbol{z}_{t+1}; \mu_t, \Lambda_{y,t}^{-1} + \Sigma_{\epsilon,y})$$

As this only depends on the input $\boldsymbol{x}$, we can use this likelihood within MOCA to update the run length belief upon seeing $\boldsymbol{x}_t$ and before predicting $\hat{y}_t$.

In summary, PCOC performs Bayesian Gaussian discriminant analysis for online classification, and meta-learns the parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}_0, \boldsymbol{q}_{1:J,0}, \Lambda_{1:J,0}, \boldsymbol{w}, \Sigma_{\epsilon,1:J}\}$ for efficient few-shot online classification. In practice, we assume that all the covariances are diagonal to limit memory footprint of the posterior parameters. PCOC can be thought of a Bayesian analogue of prototypical networks (Snell et al., 2017).

D Related Work

Online Learning, Continuous Learning, and Concept Drift Adaptation. A substantial literature exists on online, continual and lifelong learning (Hazan, 2016; Chen & Liu, 2016). While these terms are often used interchangeably and inconsistently, they all roughly correspond to the problem of learning within a streaming series of tasks, wherein it is desirable to re-use information from previous tasks while avoiding negative transfer French (1999); Thrun & Pratt (2012). Typically, continual learning assumes access to task segmentation information, whereas online learning does not (Aljundi et al., 2019). Regularization approaches (Kirkpatrick et al., 2017; Hazan, 2016; Li & Hoiem, 2017) have been shown to be an effective method for avoiding forgetting in continual learning. By augmenting the loss function for a new task with a penalty for deviation from the parameters learned for previous tasks, the regularizing effects of a prior are mimicked; in contrast we explicitly learn a prior over task weights that is meta-trained to be rapidly adaptive. Thus, MOCA is capable of avoiding substantial negative transfer by detecting task change, and rapidly adapting to new tasks. Aljundi et al. (2019) loosen the assumption of task segmentation in continual learning and operate in a similar setting to that addressed herein, but their work still focuses on learning a single set of parameters that perform well on all tasks; in contrast, we operate in the meta-learning setting, aiming to learn parameters that accelerate online adaptation within a task.

Meta-Learning for Continuous and Online Learning. While continual learning techniques have mitigated forgetting in changing problem settings, large learning models have been slow to adapt to new tasks, due in part to the propensity of neural network models to overfit to small amounts of data. In response to this, there has been substantial interest in applying ideas from meta-learning to continual learning to enable rapid adaptation to new tasks. Indeed, some modern meta-learning models such as MAML (Finn et al., 2017) may be interpreted as regularization methods (Grant et al., 2018), wherein the regularization term is explicitly learned for fast adaptation. In the streaming data setting, several works (Nagabandi et al., 2019a; He et al., 2019) use a sliding window approach, wherein a small amount of recent data is used for conditioning. By not explicitly detecting task change and choosing the window length in response, these models risk suffering from negative transfer. Indeed, MOCA may be interpreted as an adaptive sliding window model, that actively infers the optimal window length. Nagabandi et al. (2019b) and Jerfel et al. (2019) aim to detect task changes

via combining mean estimation of the dependent variable with MAML models. However, these models are both less expressive than MOCA (which maintains a full Bayesian posterior) and are not capable of task-unsegmented training. Instead, these models require pre-training with a meta-dataset that is segmented by task, limiting their applicability relative to MOCA.

Empirical Bayes for Changepoint Models. The Bayesian online changepoint framework of Adams & MacKay (2007) (which we leverage in this paper) and the similar, simultaneous work of Fearnhead & Liu (2007) have generated a substantial body of follow-on work since their publication. Due to the simplicity of these algorithms—in particular, the ability to compute closed-form posteriors as opposed to being forced to turn to approximate methods such as MCMC—many practical modifications and extensions have been developed. Of particular relevance are two works that investigate empirical Bayes for the underlying predictive model, which is a similar problem to that addressed herein. In particular, Paquet (2007) develop a forward-backward algorithm that allows closed-form max likelihood estimation of the prior for simple distributions via EM. Turner et al. (2009) derive general-purpose gradients for hyperparameter optimization within the BOCPD model. This approach is similar to our work, although we use neural network meta-learning models and rely on automatic differentiation for gradient computation.

E Further Experimental Results

E.1 Test Time versus Train Time Task Segmentation

We investigate in isolation the effects of task-segmentation information when provided at train-time and at test-time. To characterize the impact of test time segmentation, we train an oracle model and at test time, remove task segmentation and replace it with MOCA's run length estimation. We then compare this to the oracle model tested with segmentation, so the only difference is availability of test-time segmentation. Similarly, to characterize the impact of train time segmentation, we provide a model trained using MOCA with task segmentation at test time and compare this to a the same MOCA model when tested without segmentation.

Regression. Fig. 4 shows the performance of MOCA when augmented with task segmentation at test time (violet), compared to unsegmented (blue), as well as the oracle model without test segmentation (teal) compared to with test segmentation (grey). We find that as the hazard rate increases, both the value of segmentation in training and value of segmentation at test time increases steadily. Because our regression version of MOCA is not performing density estimation for the independent variable, it is not able to detect a changepoint before incurring the loss associated with an incorrect prediction. Thus, for high hazard rates, considerable loss is incurred, increasing the value of task segmentation. Interestingly and counter-intuitively, the model trained with MOCA outperforms the model trained with oracle supervision, when both are given oracle supervision at test time. The MOCA training results in a small "curriculum" effect due to the non-zero weight on placed on the prior for every training iteration; in comparison, for the oracle model, a missed prediction with a highly concentrated yet incorrect posterior occasionally results in a very large loss than may destabilize training. When the oracle model is trained with a small belief weight on the prior (even down to e.g. 10^{-16}), the performance matches the MOCA model. This suggests that MOCA may be beneficial in training by acting as a form of curriculum.

Classification. The relative effect of the MOCA train and test is apparent for Rainbow MNIST. For high hazard rates, as expected, MOCA at test time performs slightly worse than the oracle model. The majority of performance degradation is thus due to MOCA training. Performance degradation due to MOCA training is largest for this experiment, compared to the sinusoid and miniImageNet. Because the changing digit color results in a relatively clear indicator of changepoints, and MOCA performs a belief update based on both the image and the label, MOCA performs comparably to the oracle model at test time.

On miniImageNet, in contrast to the Rainbow MNIST experiment, there is a large and constant (with respect to hazard rate) performance decrease moving from oracle to MOCA at test time. Interestingly, one would expect the performance decrease with respect to hazard rate to be attributable primarily to lack of task segmentation at test time—in fact, it appears that the trend is primarily a consequence of MOCA training. This also holds for the Rainbow MNIST experiments. This is likely a consequence of the limited amount of data, as the trend is not apparent for the sinusoid experiment.

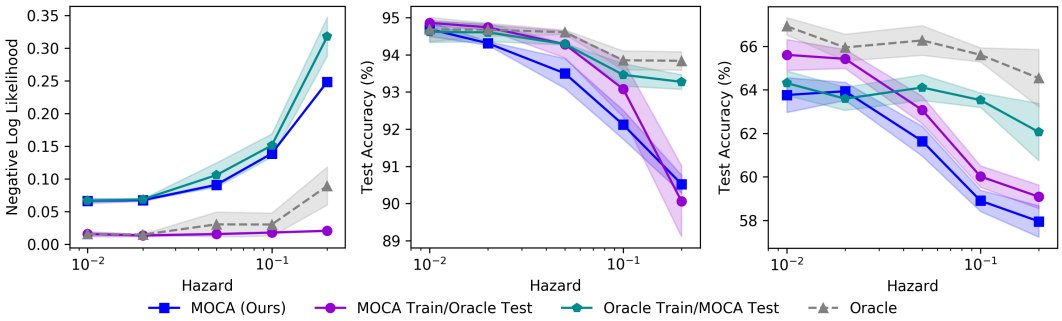


Figure 4: Performance change from augmenting a model trained with MOCA with task supervision at test time (violet) and from using changepoint estimation at test time for a model trained with task-supervision (teal), for sinusoid (**left**), Rainbow MNIST (**center**), and miniImageNet (**right**).

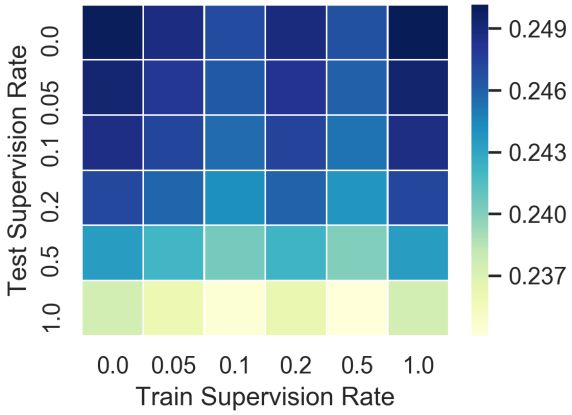


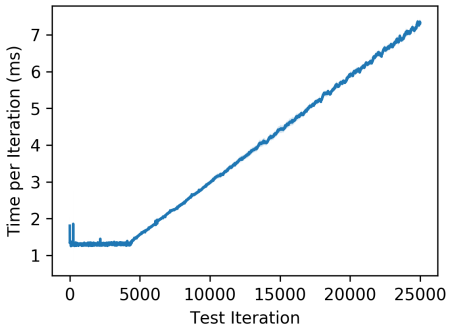
Figure 5: Test negative log likelihood of MOCA on the sinusoid problem with partial task segmentation. The partial segmentation during training results in negligible performance increase, while partial supervision at test time uniformly improves performance. Note that each column corresponds to one trained model, and thus the randomly varying performance across train supervision rates may be explained by simply results of minor differences in individual models.

E.2 MOCA with Partial Task Segmentation

Since MOCA explicitly reasons about a belief over run-lengths, it can operate anywhere in the spectrum of the task-unsegmented case as presented so far, to the fully task-segmented setting of standard meta-learning. At every time step t, the user can override the belief $b_t(r_t)$ to provide a degree of supervision. At known changepoints, for example, the user can override $b_t(r_t)$ to have all its mass on $r_t = 0$. If the task is known *not* to change at the given time, the user can set the hazard probability to 0 when updating the belief for the next timestep. If a user applies both of these overrides, it amounts to effectively sidestepping the Bayesian reasoning over changepoints and revealing this information to the meta-learning algorithm. If the user only applies the former, the user effectively indicates to the algorithm when known changepoints occur, but the algorithm is free to propagate this belief forward in time according to the update rules, and detect further changepoints that were not known to the user. Finally, the Bayesian framework allows a supervisor to provide their belief over a changepoint, which may not have probability mass entirely at $r_t = 0$. Thus, MOCA flexibly incorporates any type of task supervision available to a system designer.

Fig. 5 shows the performance of partial task segmentation at both train and test for the sinusoid problem, for the hazard rate 0.2. This problem was chosen as the results were highly repeatable and thus the trend is more readily observed. Here, we label a changepoint with some probability, which we refer to as the supervision rate. We do not provide supervision for any non-changepoint timesteps, and thus a supervision rate of 1 corresponds to labeling every changepoint but is not equivalent to the oracle. Specifically, the model may still have false positive changepoints, but is incapable of false negatives. This figure shows that the performance monotonically improves with increasing train supervision rate, but is largely invariant under varying train supervision. This performance improvement agrees with Fig. 4, which shows that for the sinusoid problem, performance is improved by full online segmentation. Indeed, these results show that training with MOCA results in models with comparable test performance to those with supervised changepoints, and thus there is little marginal value to task segmentation during training.

Figure 6: Time per iteration versus iteration number at test time. Note that the right hand side of the curve shows the expected linear complexity expected of MOCA. Note that for these experiments, no hypothesis pruning was performed. and thus at test time performance could be constant time as opposed to linear. This figure shows 95% confidence intervals for 10 trials, but the repeatability of the computation time is consistent enough that they are not visible.



E.3 Computational Performance

Fig. 6 shows the computational performance at test time on the sinusoid problem. Note that the right hand side of the curve shows a linear trend that is expected from the growing run length belief vector. However, even for 25000 iterations, the execution time is approximately 7ms for one iteration. These experiments were performed on an Nvidia Titan Xp GPU. Interestingly, on the left hand side of the curve, the time per iteration is effectively constant until the number of iterations approaches approximately 4500. Based on our code profiling, we hypothesize that this is an artifact of overhead in matrix multiplication computations done on the GPU.

F Experimental Details

F.1 Sinusoid

To test the performance of the MOCA framework combined with ALPaCA for the regression setting, we investigate a switching sinusoid regression problem. The standard sinusoid regression problem, in which randomly sampled phase and amplitude constitute a task, is a standard benchmark in meta-learning (Finn et al., 2017). Moreover, a switching sinusoid problem is a popular benchmark in continuous learning (He et al., 2019; Javed & White, 2019). Each task consists of a randomly sampled phase in the range $[0, \pi]$ and amplitude in $[0.1, 5]$. This task was investigated for varying hazard rates. For the experiments in this paper, samples from the sinusoid had additive zero-mean Gaussian noise of variance 0.05.

F.2 Rainbow MNIST

The Rainbow MNIST dataset (introduced in Finn et al. (2019)) contains 56 different color/scale/rotation transformations of the MNIST dataset, where one transformation constitutes a task. We split this dataset into a train set of 49 transformations and a test set of 7. For hyperparameter optimization, we split the train set into a training set of 42 transformations and a validation of 7. However, because the dataset represents a fairly small amount of tasks (relative to the sinusoid problem, which has infinite), after hyperparameters were set we trained on all 49 tasks. We found this notably improved performance. Note that the same approach was used in Snell et al. (2017).

F.3 miniImageNet

We use the miniImageNet dataset of Vinyals et al. (2016), a standard benchmark in few-shot learning. However, the standard few-shot learning problem does not require data points to be assigned to a certain class label. Instead, given context data, the goal is to associated the test data with the correct context data. We argue that this problem setting is implausible for the continual learning setting: while observing a data stream, you are also inferring the set of possible labels. Moreover, after a task change, there is no context data to associate a new point with. Therefore we instead assume a known set of classes. We group the 100 classes of miniImageNet in to five super-classes, and perform five-way classification given these. These super-classes vary in intra-class diversity of sub-classes: for example, one of the super-class is entirely composed of sub-classes that are breeds of dogs, while another corresponds to buildings, furniture, and household objects. Thus, the strength of the prior information for each super-class varies. Moreover, the intra-class similarities are quite weak, and thus generalization from the train set to the test set is difficult and few-shot learning is still necessary and beneficial. The super-classes are detailed in table **??**.

Class	Description	Train/Val/Test	Synsets
1	Non-dog animals	Train	n01532829, n01558993, n01704323, n01749939, n01770081, n01843383, n01910747, n02074367, n02165456, n02457408, n02606052, n04275548
		Validation	n01855672, n02138441, n02174001
		Test	n01930112, n01981276, n02129165, n02219486, n02443484
2	Dogs, foxes, wolves	Train	n02089867, n02091831, n02101006, n02105505, n02108089, n02108551, n02108915, n02111277, n02113712, n02120079
		Validation	n02091244, n02114548
		Test	n02099601, n02110063, n02110341, n02116738
3	Vehicles, musical instruments, nature/outdoors	Train	n02687172, n02966193, n03017168, n03838899, n03854065, n04251144, n04389033, n04509417, n04515003, n04612504, n09246464, n13054560
		Validation	n02950826, n02981792, n03417042, n03584254, n03773504, n09256479
		Test	n03272010, n04146614
4	Food, kitchen equipment, clothing	Train	n02747177, n02795169, n02823428, n03047690, n03062245, n03207743, n03337140, n03400231, n03476684, n03527444, n03676483, n04596742, n07584110, n07697537, n07747607, n13133613
		Validation	n03770439, n03980874
		Test	n03146219, n03775546, n04522168, n07613480
5	Building, furniture, household items	Train	n03220513, n03347037, n03888605, n03908618, n03924679, n03998194, n04067472, n04243546, n04258138, n04296562, n04435653, n04443257, n04604644, n06794110
		Validation	n02971356, n03075370, n03535780
		Test	n02871525, n03127925, n03544143, n04149813, n04418357

Table 1: Our super-class groupings for miniImageNet experiments.

The super-classes are roughly balanced in terms of number of classes contained. Each task correspond to sampling a class from within each super-class, which was fixed for the duration of that task. Each super-class was sampled with equal probability.

F.4 Baselines

Three baselines were used, described below:

- **Train on Everything**: This baseline consists of ignoring task variation and treating the training timeseries as one dataset. Note that many datasets contain latent temporal information that is ignored, and so this approach is effectively common practice.
- **Oracle**: In this baseline, the same ALPaCA and PCOC models were used as in MOCA, but with exact knowledge of the task switch times. Note that within a regret setting, one typically compares to the best achievable performance. The oracle actually outperforms the best achieveable performance in this problem setting, as it takes at least one data point (and the associated prediction, on which loss is incurred) to become aware of the task variation.
- **Sliding Window**: The sliding window approach is commonly used within problems that exhibit time variation, both within meta-learning (Nagabandi et al., 2019a) and continual learning (He et al., 2019; Gama et al., 2014). In this approach, the last n data points are used for conditioning, under the expectation that the most recent data is the most predictive of the observations in the near future. Typically, some form of validation is used to choose the window length, n. As MOCA is performing a form of adaptive windowing, it should ideally outperform any fixed window length. We compare to three window lengths ($n = 5, 10, 50$), each of which are well-suited to part of the range of hazard rates that we consider.

F.5 Training Details

Sinusoid. A standard feedforward network consisting of two hidden layers of 128 units was used with ReLU nonlinearities. These layers were followed by a 32 units layer and another tanh nonlinearity. Finally, the output layer (for which we learn a prior) was of size 32×1. The same architecture was used for all baselines. This is the same architecture for sinusoid regression as was

used in Harrison et al. (2018) (with the exception of using ReLU nonlinearities instead of all tanh nonlinearities). The following parameters were used for training:

- Optimizer: Adam (Kingma & Ba, 2015)
- Learning rate: 0.02
- Batch size: 50
- Batch length: 100
- Train iterations: 7500

Batch length here corresponds to the number of timesteps in each training batch. Note that longer batch lengths are necessary to achieve good performance on low hazard rates, as short batch lengths artificially increase the hazard rate as a result of the assumption that each batch begins with a new task. The learning rate was decayed every 1000 training iterations.

We allowed the noise variance to be learned by the model. This, counter-intuitively, resulted in a substantial performance improvement over a fixed (accurate) noise variance. This is due to a curriculum effect, where the model early one increases the noise variance and learns roughly accurate features, followed by slowly decreasing the noise variance to the correct value.

Rainbow MNIST. In our experiments, we used the same architecture as was used as in Snell et al. (2017); Vinyals et al. (2016). It is often unclear in recent work on few-shot learning whether performance improvements are due to improvements in the meta-learning scheme or the network architecture used (although these things are not easily disentangled). As such, the architecture we use in this experiment provides fair comparison to previous few-shot learning work. This architecture consists of four blocks of 64 3×3 convolution filters, followed by a batchnorm, ReLU nonlinearity and 2×2 max pool. On the last conv black, we removed the batchnorm and the nonlinearity. For the 28×28 Rainbow MNIST dataset, this encoder leads to a 64 dimensional embedding space. For the "train on everything" baseline, we used the same architecture followed by a fully connected layer and a softmax. This architecture is standard for image classification and has a comparable number of parameters to our model.

We used a diagonal covariance factorization within PCOC, substantially reducing the number of terms in the covariance matrix for each class and improving the performance of the model (due to the necessary inversion of the posterior predictive covariance). We learned a prior mean and variance for each class, as well as a noise covariance for each class (again, diagonal). We also fixed the Dirichlet priors to be large, effectively imbuing the model with the knowledge that the classes were balanced. The following parameters were used for training:

- Optimizer: Adam
- Learning rate: 0.02
- Batch size: 10
- Batch length: 100
- Train iterations: 5000

The learning rate was decayed every 1500 training iterations.

miniImageNet. Finally, for miniImageNet, we used six convolution blocks, each as previous described. This resulted in a 64 dimensional embedding space. We initially attempted to use the same four-conv backbone as for Rainbow MNIST, but the resulting 1600 dimensional embedding space had unreasonable memory requirements for batches lengths of 100. Again, for the "train on everything" baseline, we used the same architectures with one fully connected layer followed by a softmax. The following parameters were used for training:

- Optimizer: Adam
- Learning rate: 0.002
- Batch size: 10
- Batch length: 100
- Train iterations: 3000

The learning rate was decayed every 1000 training iterations. We used the validation set to monitor performance, and as in Chen et al. (2019), we used the highest validation accuracy iteration for test. We also performed data augmentation as in Chen et al. (2019) by adding random reflections and color jitter to the training data.

F.6 Test Details

For all problems, a test horizon of 400 was used. Again, the longest possible test horizon was used to avoid artificial distortion of the test hazard rate. Both both problems, a batch of 200 evaluations was performed, and all confidence intervals correspond to 95%.

G Future Work

While MOCA addresses a continual learning problem setting, we have not formulated MOCA as an online learning algorithm. Specifically, MOCA meta-trains on an offline time-series, and keeps the parameters $\boldsymbol{\theta}$ fixed online, whereas an online learning algorithm would not have this train/test distinction, and would consider updating $\boldsymbol{\theta}$ continuously (Hazan, 2016). However, in order to do this with MOCA, we would need to keep a running buffer of all data observed so far and to use as training data to update $\boldsymbol{\theta}$, which may be expensive in real-world domains where large volumes of data (e.g. high definition video from a large collection of cameras on an autonomous vehicle). Extending MOCA toward either strictly online training or a scheme to maintain an efficient replay buffer (Mnih et al., 2013; Vitter, 1985), is a promising direction of future work. Indeed, it may be possible to use MOCA's changepoint analysis to inform which data to save.

Beyond the continual learning extension, data efficiency may be improved by re-using information from previous tasks or modeling task evolution dynamics. Previous work (Nagabandi et al., 2019b; Jerfel et al., 2019; Knoblauch & Damoulas, 2018) has addressed the case in which tasks reoccur in both meta-learning and the BOCPD framework, and thus knowledge (in the form of a posterior estimate) may be re-used. In this work, we address the case in which tasks are sampled i.i.d. from a (typically continuous) distribution, and thus knowledge re-use is often impractical or adds marginal value. Broadly, moving beyond the assumption of i.i.d. tasks to task having associated dynamics (Al-Shedivat et al., 2018) represents a promising future direction.