# Calibrated Model-Based Deep Reinforcement Learning

**Ali Malik** [*1] **Volodymyr Kuleshov** [*1 2] **Jiaming Song** [1] **Danny Nemer** [2] **Harlan Seymour** [2] **Stefano Ermon** [1]

## Abstract

Estimates of predictive uncertainty are important for accurate model-based planning and reinforcement learning. However, predictive uncertainties — especially ones derived from modern deep learning systems — can be inaccurate and impose a bottleneck on performance. This paper explores which uncertainties are needed for model-based reinforcement learning and argues that ideal uncertainties should be calibrated, i.e. their probabilities should match empirical frequencies of predicted events. We describe a simple way to augment any model-based reinforcement learning agent with a calibrated model and show that doing so consistently improves planning, sample complexity, and exploration. On the HALFCHEETAH MuJoCo task, our system achieves state-of-the-art performance using 50% fewer samples than the current leading approach. Our findings suggest that calibration can improve the performance of model-based reinforcement learning with minimal computational and implementation overhead.

## 1. Introduction

Methods for accurately assessing predictive uncertainty are important components of modern decision-making systems. Probabilistic methods have been used to improve the safety, interpretability, and performance of decision-making agents in various domains, including medicine (Jiang et al., 2011), robotics (Chua et al., 2018; Buckman et al., 2018), and operations research (Van Roy et al., 1997).

In model-based reinforcement learning — a setting in which an agent learns a model of the world from past experience and uses it to plan future decisions — capturing uncertainty in the agent's model is particularly important (Deisenroth & Rasmussen, 2011). Planning with a probabilistic model improves performance and sample complexity, especially when representing the model using a deep neural network (Rajeswaran et al., 2016; Chua et al., 2018).

Despite their importance in decision-making, predictive uncertainties can be unreliable, especially when derived from deep neural networks (Guo et al., 2017a). Although several modern approaches such as deep ensembles (Lakshminarayanan et al., 2017b) and approximations of Bayesian inference (Gal & Ghahramani, 2016a;b; Gal et al., 2017) provide uncertainties from deep neural networks, these methods suffer from shortcomings that reduce their effectiveness for planning (Kuleshov et al., 2018).

In this paper, we study the kind of uncertainties that are needed for model-based reinforcement learning and argue that ideal predictive uncertainties should be calibrated, i.e. their probabilities should match empirical frequencies of predicted events. We propose a simple way to augment any model-based reinforcement learning algorithm with a calibrated model by adapting recent advances in uncertainty estimation for deep neural networks (Kuleshov et al., 2018). We complement our approach with diagnostic tools, best-practices, and intuition on how to apply calibration in reinforcement learning.

We validate our approach on benchmarks for contextual bandits and continuous control (Li et al., 2010; Todorov et al., 2012), as well as on a planning problem in inventory management (Van Roy et al., 1997). Our results show that calibration consistently improves the cumulative reward and the sample complexity of model-based agents, and also enhances their ability to balance exploration and exploitation in contextual bandit settings. Most interestingly, on the HALFCHEETAH task, our system achieves state-of-the-art performance, using 50% fewer samples than the previous leading approach (Chua et al., 2018). Our results suggest that calibrated uncertainties have the potential to improve model-based reinforcement learning algorithms with minimal computational and implementation overhead.

**Contributions.** In summary, this paper[1] adapts recent advances in uncertainty estimation for deep neural networks

[*]Equal contribution [1]Department of Computer Science, Stanford University, USA [2]Afresh Technologies, San Francisco, USA. Correspondence to: Ali Malik <malikali@stanford.edu>, Volodymyr Kuleshov <kuleshov@cs.stanford.edu>.

*Proceedings of the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

[1]Our code is available at https://github.com/ermongroup/CalibratedModelBasedRL

to reinforcement learning and proposes a simple way to improve any model-based algorithm with calibrated uncertainties. We explain how this technique improves the accuracy of planning and the ability of agents to balance exploration and exploitation. Our method consistently improves performance on several reinforcement learning tasks, including contextual bandits, inventory management and continuous control.

## 2. Background

### 2.1. Model-Based Reinforcement Learning

Let $\mathcal{S}$ and $\mathcal{A}$ denote (possibly continuous) state and action spaces in a Markov Decision Process $(\mathcal{S}, \mathcal{A}, T, r)$, and let $\Pi$ denote the set of all stationary stochastic policies $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that choose actions in $\mathcal{A}$ given states in $\mathcal{S}$. The successor state $s'$ for a given action $a$ from current state $s$ are drawn from the dynamics function $T(s'|s, a)$. We work in the $\gamma$-discounted infinite horizon setting and we will use an expectation with respect to a policy $\pi \in \Pi$ to denote an expectation with respect to the trajectory it generates: $\mathbb{E}_\pi[r(s, a)] \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, where $s_0 \sim p_0$, $a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim T(\cdot|s_t, a_t)$ for $t \geq 0$. $p_0$ is the initial distribution over states and $r(s_t, a_t)$ is the reward at time $t$.

Typically, $\mathcal{S}, \mathcal{A}, \gamma$ are known, while the dynamics model $T(s'|s, a)$ and the reward function $r(s, a)$ are not known explicitly. This work focuses on model-based reinforcement learning, in which the agent learns an approximate model $\hat{T}(s'|s, a)$ of the world from samples obtained by interacting with the environment and uses this model to plan its future decisions.

### 2.2. Probabilistic Models

This paper focuses on probabilistic dynamics models $\widehat{T}(s'|s, a)$ that take a current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, and output a probability distribution over future states $s'$. We represent the output distribution over the next states, $\widehat{T}(\cdot|s, a)$, as a cumulative distribution function $F_{s,a} : \mathcal{S} \to [0, 1]$, which is defined for both discrete and continuous $\mathcal{S}$.

**Calibration.** A key desirable property of probabilistic forecasts is calibration. Intuitively, a transition model $\widehat{T}(s'|s, a)$ is calibrated if whenever it assigns a probability of 0.8 to an event — such as a state transition $(s, a, s')$ — that transition should occur about 80% of the time.

Since many important reinforcement learning problems and benchmarks, including all the ones that are studied in this paper, involve continuous state spaces $\mathcal{S}$, we formally define here calibration for continuous variables, and refer the reader to (Foster & Vohra, 1998; Gammerman & Vovk, 2007; Kuleshov & Ermon, 2017) for the discrete-variable definition.

Recall that the forecaster $\widehat{T}(s'|s, a)$ outputs a CDF $F_{s,a}$ over future states $s'$. We will use $F^{-1}_{s,a} : [0, 1] \to \mathbb{R}$ to denote the quantile function $F^{-1}_{s,a}(p) = \inf\{y : p \leq F_{s,a}(y)\}$ (see Kuleshov et al. (2018) for the multivariate extension). Formally, we say that the forecaster $\widehat{T}(s'|s, a)$ is calibrated over observations $\{(s_t, a_t), s'_t\}\}_{t=1}^N$ if

$$\frac{\sum_{t=1}^N \mathbb{I}\{s'_t \leq F^{-1}_{s_t,a_t}(p)\}}{N} \to p \text{ for all } p \in [0, 1] \quad (1)$$

as $N \to \infty$. In other words, the empirical and predicted CDFs should match as the dataset size goes to infinity. Intuitively, the above equation indicates that $s'$ should fall in a 90% predicted probability interval approximately 90% of the time.

When the $s_t, a_t, s'_t$ are i.i.d. realizations of random variables $S, A, S' \sim \mathbb{P}$, a sufficient condition for calibration is: $P(S' \leq F^{-1}_{S,A}(p)) = p$ for all $p \in [0, 1]$.

**Recalibration.** Most predictive models are not calibrated out-of-the-box due to modeling bias and computational approximations. However, given an arbitrary pre-trained forecaster $H : \mathcal{X} \to (\mathcal{Y} \to [0, 1])$ that outputs CDFs $F$, we may train an auxiliary model $R : [0, 1] \to [0, 1]$ such that the forecasts $R \circ F$ are calibrated in the limit of enough data. This procedure, called recalibration, is simple to implement, computationally inexpensive, and can be applied to any probabilistic regression model in a black-box manner. Furthermore, it does not increase the loss function of the original model if it belongs to a large family of objectives called proper losses (Kull & Flach, 2015; Kuleshov & Ermon, 2017).

**Algorithm 1** CALIBRATE: Recalibration of Transition Dynamics

**Input:** Uncalibrated transition model $\widehat{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ that outputs CDFs $F_{s,a} : \mathcal{S} \to [0, 1]$, and calibration set $\mathcal{D}_{\text{cal}} = \{(s_t, a_t), s_{t+1}\}\}_{t=1}^N$
**Output:** Auxiliary recalibration model $R : [0, 1] \to [0, 1]$.

1. Construct a recalibration dataset

$$\mathcal{D} = \left\{\left(F_{s_t,a_t}(s_{t+1}), \hat{P}(F_{s_t,a_t}(s_{t+1}))\right)\right\}_{t=1}^N$$

where

$$\hat{P}(p) = \frac{1}{N}\sum_{t=1}^N \mathbb{I}[F_{s_t,a_t}(s_{t+1}) \leq p].$$

2. Train a model $R : [0, 1] \to [0, 1]$ (e.g. sigmoid or isotonic regression) on $\mathcal{D}$.

When $\mathcal{S}$ is discrete, a popular choice of $R$ is Platt scaling (Platt et al., 1999); Kuleshov et al. (2018) extends Platt

scaling to continuous variables. Either of these methods can be used within our framework. Since this paper focuses on continuous state spaces, we use the method of Kuleshov et al. (2018) described in Algorithm 1, unless otherwise indicated.

## 3. Calibrated Model-Based Reinforcement Learning

In this section, we examine calibration in the context of reinforcement learning and describe a general approach for augmenting any model-based reinforcement learning agent with a calibrated state transition model that improves its performance.

**Intuition.** To better understand why calibration is important in model-based planning, consider a simple MDP with two states $s_{\text{good}}$ and $s_{\text{bad}}$. The former has a high reward $r(s_{\text{good}}) = 1$ and the latter has a low reward $r(s_{\text{bad}}) = -1$.

Consider the reward $\hat{r}$ that we expect to obtain from taking action $a$ in $s_{\text{good}}$ if we believe our model, and that is given by $\hat{r} = -1 \cdot \hat{T}(s_{\text{bad}}|s_{\text{good}}, a) + 1 \cdot \hat{T}(s_{\text{good}}|s_{\text{good}}, a)$. If the true transition probability is $T(s_{\text{good}}|s_{\text{good}}, a) = 80\%$, but our model $\hat{T}$ is miscalibrated and predicts 60%, then the average observed reward from $a$ in $s_{\text{good}}$ will in the long run not equal to the reward $\hat{r}$ predicted by our model, and hence will be incorrect. An incorrect estimate of the expected reward may in turn lead us to choosing a sub-optimal action.

Similarly, suppose 'that the model is over-confident and $\hat{T}(s_{\text{good}}|s_{\text{good}}, a) = 0$; intuitively, we may decide that it is not useful to try $a$ in $s_{\text{good}}$, as it leads to $s_{\text{bad}}$ with 100% probability. However, this is a suboptimal decision, and we may discover that $a$ is a good action after trying it out a few times. This is an instance of the classical exploration-exploitation problem; many approaches to this problem (such as the UCB family of algorithms) rely on accurate confidence bounds and are likely to benefit from calibrated uncertaintites that more accurately reflect the true probability of transitioning to a particular state.

### 3.1. Method

In Algorithm 2, we present a simple procedure that augments a model-based reinforcement learning algorithm with an extra step that ensures the calibration of its transition model. Algorithm 2 effectively corresponds to standard model-based reinforcement learning with the addition of Step 4, in which we train a recalibrator $R$ such that $R \circ T$ is calibrated. The subroutine CALIBRATE can be an instance of Platt scaling, for discrete $S$, or the method of Kuleshov et al. (2018), when $S$ is continuous (see Algorithm 1).

In the rest of this section, we describe best practices for applying this method.

**Algorithm 2** Calibrated Model-Based Reinforcement Learning

**Input:** Initial transition model $\widehat{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ and initial dataset of state transitions $\mathcal{D} = \{(s_t, a_t), s_{t+1}\}_{t=1}^{N}$
Repeat until sufficient level of performance is reached:

1. Run the agent and collect a dataset of state transitions $\mathcal{D}_{\text{new}} \leftarrow \text{EXECUTEPLANNING}(\widehat{T})$. Gather all experience data $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\text{new}}$.
2. Let $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{cal}} \leftarrow \text{PARTITIONDATA}(\mathcal{D})$ be the training and calibration sets, respectively.
3. Train a transition model $\widehat{T} \leftarrow \text{TRAINMODEL}(\mathcal{D}_{\text{train}})$.
4. Train the recalibrator $R \leftarrow \text{CALIBRATE}(\widehat{T}, \mathcal{D}_{\text{cal}})$.
5. Let $\widehat{T} \leftarrow R \circ \widehat{T}$ be the new, recalibrated transition model.

**Choice of Calibration Set.** In order to obtain calibration estimates that don't overfit to the training data, it is recommended that the model and the recalibrator $R$ should be trained on separate data (Zadrozny & Elkan, 2001; Kuleshov et al., 2018). An alternative, less wasteful approach is to split the training data into $K$ folds, using one of the folds for calibration and the rest for model training. This can be repeated for each of the folds and the outputs of the models for can be averaged.

**Diagnostic Tools.** An essential tool for visualising calibration of predicted CDFs $F_1, \dots F_N$ is the reliability curve (Gneiting et al., 2007). This plot displays the empirical frequency of points in a given interval relative to the predicted fraction of points in that interval. Formally, we choose $m$ thresholds $0 \le p_1 \le \cdots \le p_m \le 1$ and, for each threshold $p_j$, compute the empirical frequency $\hat{p}_j = |y_t : F_t(y_t) \le p_j, t = 1, \dots, N|/N$

Plotting $\{(p_j, \hat{p}_j)\}$ gives us a sense of the calibration of the model (see Figure 1), with a straight line corresponding to perfect calibration. An equivalent, alternative visualisation is to plot a histogram of of the probability integral transform $\{F_t(y_t)\}_{t=1}^{N}$ and see if it looks like a uniform distribution (Gneiting et al., 2007).

These visualisations can be quantified by defining the calibration loss[2] of a model:

$$\text{cal}(F_1, y_1, \dots F_t, y_Y) = \sum_{j=1}^{m} (\hat{p}_j - p_j)^2, \qquad (2)$$

as the sum of the squares of the residuals $(\hat{p}_j - p_j)^2$. These diagnostic tools should be evaluated on unseen data distinct

[2]This is the calibration term in the two-component decomposition of the Brier score.

from the training and calibration sets as it may reveal signs of overfitting.

### 3.2. Applications to Deep Reinforcement Learning

Although deep neural networks can significantly improve model-based planning algorithms (Higuera et al., 2018; Chua et al., 2018), their estimates of predictive uncertainty are often inaccurate (Guo et al., 2017a; Kuleshov et al., 2018).

**Variational Dropout.** One popular approach to deriving uncertainty estimates from deep neural networks involves using dropout. Taking the mean and the variance of dropout samples leads to a principled Gaussian approximation of the posterior predictive distribution from a Bayesian neural network (in regression) (Gal & Ghahramani, 2016a). To use Algorithm 2 we may instantiate CALIBRATE with the method of Kuleshov et al. (2018) and pass it the predictive Gaussian derived from the dropout samples.

More generally, our method can be naturally applied on top of any probabilistic model without any need to modify or retrain this model.

## 4. The Benefits of Calibration in Model-Based Reinforcement Learning

Next, we examine specific ways in which Algorithm 2 can improve model-based reinforcement learning agents.

### 4.1. Model-Based Planning

The first benefit of a calibrated model is enabling more accurate planning using standard algorithms such as value iteration or model predictive control (Sutton & Barto, 2018). Each of these methods involves estimates of future reward. For example, value iteration performs the update

$$V'(s) \leftarrow \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\sum_{s'\in S}\hat{T}(s'|s,a)(r(s')+V(s'))\right].$$

Crucially, this algorithm requires accurate estimates of the expected reward $\sum_{s'\in S}\hat{T}(s'|s,a)r(s')$. Similarly, online planning algorithms involve computing the expected reward of a finite sequence of actions, which has a similar form. If the model is miscalibrated, then the predicted distribution $\hat{T}(s'|s,a)$ will not accurately reflect the true distribution of states that the agent will encounter in the real world. As a result, planning performed in the model will be inaccurate.

**Reasons for Miscalibration.** We identify two common reasons. On one hand, real-world dynamics can be complex, and a limited model may not be able to assign probability mass correctly across the entire domain of $s'$. Its predictive distribution can be too restricted or too diffuse. On the other hand, if we use an expressive probabilistic model such as a Bayesian deep neural network, inference in the model may become intractable. Common approximation techniques such as variational inference (Gal & Ghahramani, 2016a) tend to under-estimate the support of the distribution, and are often over-confident.

| | *LinUCB* | *CalLinUCB* | *Optimal* |
|---|---|---|---|
| Linear | $1209.8 \pm 12.1$ | $1210.3 \pm 12.1$ | 1231.8 |
| Beta | $1176.3 \pm 11.9$ | $1174.6 \pm 12.0$ | 1202.3 |
| Mushroom | $1429.4 \pm 154.0$ | $\mathbf{1676.1 \pm 164.1}$ | 3122.0 |
| Covertype | $558.14 \pm 3.5$ | $\mathbf{677.8 \pm 5.0}$ | 1200.0 |
| Adult | $131.3 \pm 1.2$ | $\mathbf{198.9 \pm 4.7}$ | 1200.0 |
| Census | $207.6 \pm 1.7$ | $\mathbf{603.7 \pm 3.8}$ | 1200.0 |

*Table 1.* Performance of calibrated/uncalibrated LinUCB on a variety of datasets, averaged over 10 trials. The calibrated algorithm (CalLinUCB) does better on all non-synthetic datasets (bottom four rows) and has similar performance on the synthetic datasets (top two rows).

**Calibration and Sharpness.** Note however, that calibration by itself is not a sufficient property to ensure the transition model is useful. For example, a model $T(s'|s,a)$ that predicts $\mathbb{E}[S']$ for every action $a$ is calibrated, but is not useful because it doesn't allow us to choose an action $a$. A useful transition model also needs to be sharp, which intuitively means that predictions for different actions should be maximally distinct. This property is analogous to the calibration vs. sharpness in supervised learning. The best way to improve sharpness is typically to gather more data.

### 4.2. Balancing Exploration and Exploitation

Balancing exploration and exploitation successfully is a fundamental challenge for many reinforcement learning (RL) algorithms. A large family of algorithms tackle this problem using notions of uncertainty/confidence to guide their exploration process. For example, upper confidence bound (UCB, Auer et al. (2002)) algorithms pick the action which has the highest upper bound on its reward confidence interval.

In situations where the outputs of the algorithms are uncalibrated, the confidence intervals might provide unreliable upper confidence bounds, resulting in suboptimal performance. For example, in a two-arm bandit problem, if a model is under-estimating the reward of the best arm and has high confidence, it's upper confidence bound will be low, and it will not be selected. More generally, UCB-style methods need uncertainty estimates to be on the same "order of magnitude" so that arms can be compared against each other; calibration helps ensure that.

## 5. Experiments

We evaluate our calibrated model-based reinforcement learning method on several different environments and algorithms, including contextual bandits, inventory management, and continuous control for robotics.

### 5.1. Balancing Exploration and Exploitation

To test the effect of calibration on exploration/exploitation, we look at the contextual multi-armed bandit problem (Li et al., 2010). At each timestep, an agent is shown a context vector $\mathbf{x}$ and must pick an arm $a \in \mathcal{A}$ from a finite set $\mathcal{A}$. After picking an arm, the agent receives a reward $r_{a,\mathbf{x}}$ which depends both on the arm picked and also on the context vector shown to the agent. The agent's goal over time is to learn the relationship between the context vector and reward gained from each arm so that it can pick the arm with the highest expected reward at each timestep.

**Setup.** For our experiments, we focus on the LinUCB algorithm (Li et al., 2010) — a well-known instantiation of the UCB approach to contextual bandits. LinUCB assumes a linear relationship between the context vector and the expected reward of an arm: for each arm $a \in \mathcal{A}$, there is an unknown coefficient vector $\theta_a^*$ such that $\mathbb{E}[r_{a,\mathbf{x}}] = \mathbf{x}^\top \theta_a^*$.

LinUCB learns a predictive distribution over this reward using Bayesian ridge regression, in which $\theta_a^*$ has a Gaussian posterior $\mathcal{N}(\hat{\theta}_a, \hat{\Sigma}_a)$. The posterior predictive distribution is also Gaussian, with mean $\mathbf{x}^\top \hat{\theta}_a$ and with standard deviation $\sqrt{\mathbf{x}^\top \hat{\Sigma}_a^{-1} \mathbf{x}}$. Thus, the algorithm picks the arm with the highest $\alpha$-quantile, given by

$$\arg\max_{a \in \mathcal{A}} \left( \mathbf{x}^\top \hat{\theta}_a + \alpha \cdot \sqrt{\mathbf{x}^\top \hat{\Sigma}_a^{-1} \mathbf{x}} \right). \tag{3}$$

We apply the recalibration scheme in Algorithm 2 of Kuleshov et al. (2018) to these predicted Gaussian distributions.

**Data.** We evaluate the calibrated version (CalLinUCB) and uncalibrated version (LinUCB) of the LinUCB algorithm on both synthetic data that satisfies the linearity assumption of the algorithm, as well as on real UCI datasets from Li et al. (2010). We run the tests on 2000 examples over 10 trials and compute the average cumulative reward.

**Results.** We expect the LinUCB algorithm to already be calibrated on the synthetic linear data since the model is well-specified, implying no difference in performance between CalLinUCB and LinUCB. On the real UCI datasets however, the linear assumption might not hold, resulting in miscalibrated estimates of the expected reward.

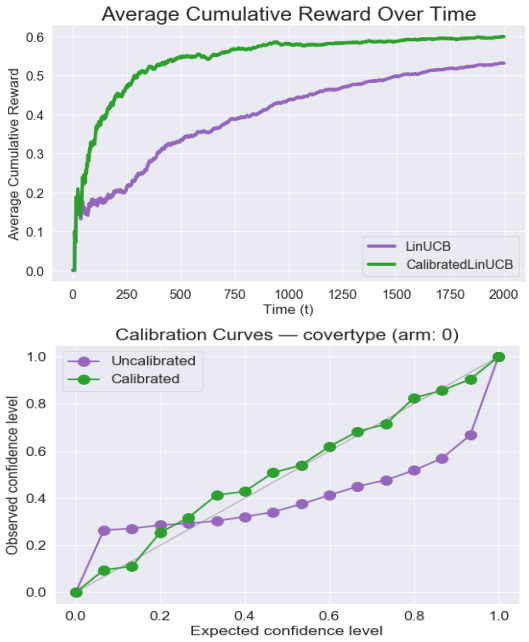


*Figure 1.* Top: Performnce of CalibLinUCB and LinUCB on the UCI covertype dataset. Bottom: Calibration curves of the LinUCB algorithms on the covertype dataset

In Table 1, we can see that indeed there is no significant difference in performance between the CalLinUCB and LinUCB algorithms on the synthetic linear dataset—they both preform optimally. On the UCI datasets however, we see a noticeable improvement with CalLinUCB on almost all tasks, suggesting that recalibration aids exploration/exploitation in a setting where the model is misspecified. Note that both CalLinUCB and LinUCB perform below the optimum on these datasets, implying linear models are not expressive enough in general for these tasks.

**Analysis.** To get a sense of the effect of calibration on the model's confidence estimates, we can plot the predicted reward with 90% confidence intervals that the algorithm expected for a chosen arm $a$ at timestep $t$. We can then compare how good this prediction was with respect to the true observed reward. Specifically, we can look at the timesteps where the CalLinUCB algorithm picked the optimal action but the LinUCB algorithm did not, and look at both the algorithms' belief about the predicted reward from both of these actions. An example of this plot can be seen in the appendix on Figure 3.

A key takeaway from this plot is that the uncalibrated algorithm systematically underestimates the expected reward from the optimal action and overestimates the expected reward of the action it chose instead, resulting in suboptimal actions. The calibrated model does not suffer from this defect, and thus performs better on the task.

### 5.2. Model-Based Planning

#### 5.2.1. Inventory Management

Our first model-based planning task is inventory management (Van Roy et al., 1997). A decision-making agent controls the inventory of a perishable good in a store. Each day, the agent orders items into the store; if the agent under-orders, the store runs out of stock; if the agent over-orders, perishable items are lost due to spoilage. Perishable inventory management systems have the potential to positively impact the environment by minimizing food waste and enabling a more effective use of resources (Vermeulen et al., 2012).

**Model.** We formalize perishable inventory management for one item using a Markov decision process $(\mathcal{S}, \mathcal{A}, P, r)$. States $s \in S$ are tuples $(d_s, q_s)$ consisting of a calendar day $d_s$ and an inventory state $q_s \in \mathbb{Z}^L$, where $L \geq 1$ is the item shelf-life. Each component $(q_s)_l$ indicates the number of units in the store that expire in $l$ days; the total inventory level is $t_s = \sum_{l=1}^{L} (q_s)_l$. Transition probabilities $P$ are defined as follows: each day sees a random demand of $D(s) \in \mathbb{Z}$ units and sales of $\max(t^{(s)} - D(s), 0)$ units, sampled at random from all the units in the inventory; at the end of the state transition, the shelf-life of the remaining items is decreased by one (spoiled items are recorded and thrown away). Actions $a \in \mathcal{A} \subseteq \mathbb{Z}$ correspond to orders: the store receives items with a shelf life of $L$ before entering the next state $s'$. In our experiments we choose the reward $r$ to be the sum of waste and unmet demand due to stock-outs.

**Data.** We use the Corporacion Favorita Kaggle dataset, which consists of historical sales from a supermarket chain in Ecuador. We experiment on the 100 highest-selling items and use data from 2014-01-01 to 2016-05-31 for training and data from 2016-06-01 to 2016-08-31 for testing.

**Algorithms.** We learn a probabilistic model $\hat{M} : \mathcal{S} \to (\mathbb{R} \to [0, 1])$ of the demand $D(s')$ in a future state $s'$ based on information available in the present state $s$. Specifically, we train a Bayesian DenseNet (Huang et al., 2017) to predict sales on each of the next five days based on features from the current day (sales serve as a proxy for demand). We use autoregressive features from the past four days, 7-, 14-, and 28-day rolling means of historical sales, binary indicators for the day of the week and the week of the year, and sine and cosine features over the number of days elapsed in the year. The Bayesian DenseNet has five layers of 128 hidden units with a dropout rate of 0.5 and parametric ReLU nonlinearities. We use variational dropout (Gal & Ghahramani, 2016b) to compute probabilistic forecasts from the model.

We use our learned distribution over $D(s')$ to perform online planning on the test set using model predictive control (MPC) learned on the training set. Specifically, we sample 5,000 random trajectories over a 5-step horizon, and choose the first action of the trajectory with the highest expected reward under the model. We estimate the expected reward of each trajectory using 300 Monte Carlo samples from the model.

We also compare the planning approach to a simple heuristic rule that always sets the inventory to $1.5 \cdot \mathbb{E}[D(s')]$, which is the expected demand multiplied by a small safety factor.

| | *Calibrated* | *Uncalibrated* | *Heuristic* |
|---|---|---|---|
| Shipped | 332,150 | 319,692 | 338,011 |
| Wasted | 7,466 | 3,148 | 13,699 |
| Stockouts | 9,327 | 17,358 | 11,817 |
| % Waste | 2.2% | 1.0% | 4.1% |
| % Stockouts | 2.8% | 5.4% | 3.5% |
| *Reward* | **-16,793** | -20,506 | -25,516 |

*Table 2.* Performance of calibrated model planning on an inventory management task. Calibration significantly improves cumulative reward. Numbers are in units, averaged over ten trials.

**Results.** We evaluate the agent within the inventory management MDP; the demand $D(s)$ is instantiated with the historical sales on test day $d(s)$ (which the agent did not observe). We measure total cumulative waste and stockouts over the 100 items in the dataset, and we report them as a fraction of the total number of units shipped to the store.

Table 2 shows that calibration improves the total cumulative reward by 14%. The calibrated model incurs waste and out-of-stocks ratios of 2.2% and 2.8%, respectively, compared to 1.0% and 5.4% for the uncalibrated one. These values are skewed towards a smaller waste, while the objective function penalizes both equally. The heuristic has ratios of 4.1% and 3.5%.

#### 5.2.2. Mujoco Environments

Our second model-based planning task is continuous control from OpenAI Gym (Brockman et al., 2016) and the Mujoco robotics simulation environment (Todorov et al., 2012). Here the agent makes decisions about its torque controls given observation states (e.g. location / velocity of joints) that maximizes the expected return reward. These environments are standard benchmark tasks for deep reinforcement learning.

**Setup.** We consider calibrating the probablistic ensemble dynamics model proposed in (Chua et al., 2018). In this approach, the agent learns an ensemble of probabilistic neural networks (PE) that captures the environment dynamics $s_{t+1} \sim f_\theta(s_t, a_t)$, which is used for model-based planning with model predictive control. The policy and ensemble model are then updated in an iterative fashion. Chua et al.

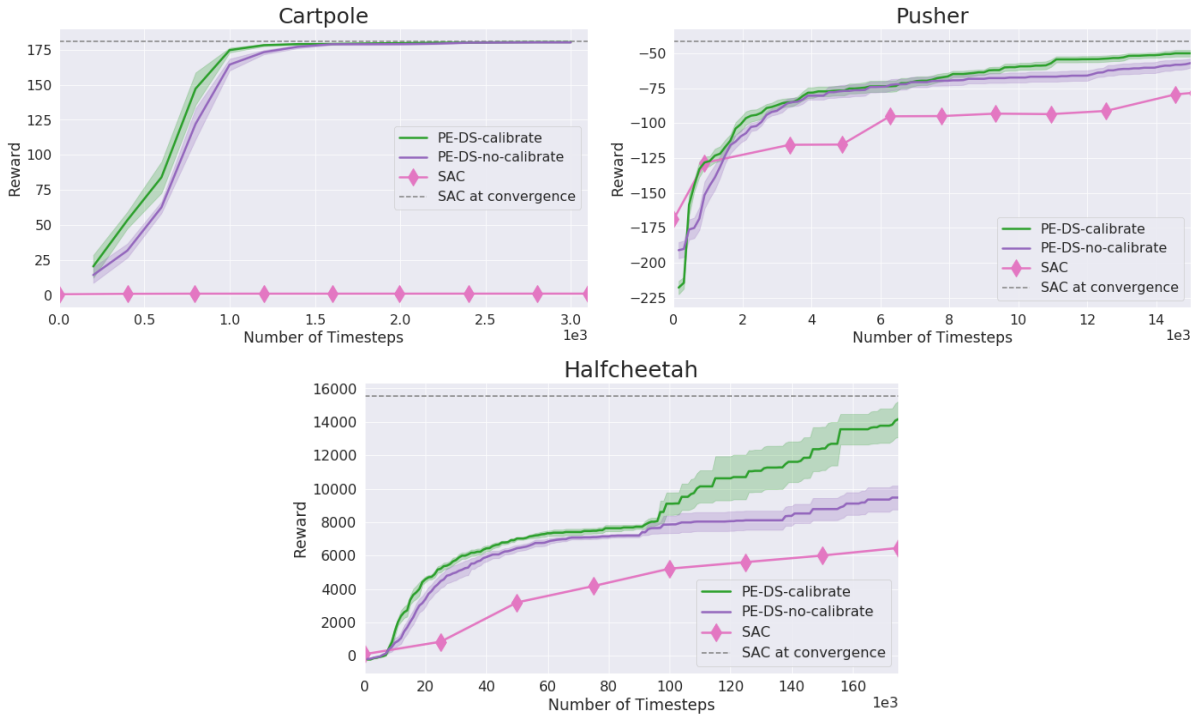


*Figure 2.* Performance on different control tasks. The calibrated algorithm does at least as good, and often much better than the uncalibrated models. Plots show maximum reward obtained so far, averaged over 10 trials. Standard error is displayed as the shaded areas.

(2018) introduce several strategies for particle-based state propagation, including trajectory sampling with bootstraped models (PE-TS); and distribution sampling (PE-DS), which samples from a multimodal distribution as follows:

$$s_{t+1} \sim \mathcal{N}(\mathbb{E}[s^p_{t+1}], \mathrm{Var}[s^p_{t+1}]), \quad s^p_{t+1} \sim f_\theta(s_t, a_t) \quad (4)$$

PE-TS and PE-DS achieve the highest sample efficiency among the methods proposed in (Chua et al., 2018).

To calibrate the model, we add a final sigmoid recalibration layer to the sampling procedure in PE-DS at each step. This logistic layer is applied separately per output state dimension and serves as the recalibrator $R$. It is trained on the procedure described in Algorithm 1, after every trial, on a separate calibration set using cross entropy loss.

We consider three continuous control environments from Chua et al. (2018) [3]. For model learning and model-based planning, we follow the training procedure and hyperparameters in Chua et al. (2018), as described in https://github.com/kchua/handful-of-trials. We also compare our method against Soft Actor-Critic (Haarnoja et al., 2018) which is one of the state-of-the-art model-free reinforcement learning algorithms. We use the final convergence reward of SAC as a criterion for the highest possible reward achieved in the task (although it may require orders of magnitude more samples from the environment).

[3]We omitted the reacher environment because the reference papers did not have SAC results for it.

**Results.** One of the most important criteria for evaluating reinforcement learning algorithms is sample complexity, i.e, the amount of interactions with the environment in order to reach a certain high expected return. We compare the sample complexities of SAC, PE-DS and calibrated PE-DS in Figure 2. Compared to the model-free SAC method, both the model-based methods use much fewer samples from the environment to reach the convergence performance of SAC. However, our recalibrated PE-DS method compares favorably to PE-DS on all three environments.

Notably, the calibrated PE-DS method outperforms both PE-DS by a significant margin on the HalfCheetah environment, reaching near optimal performance at only around 180k timesteps. To our knowledge, the calibrated PE-DS is the most efficient method on these environments in terms of sample complexity.

**Analysis.** In Figure 4 in the appendix, we visualise the 1-step prediction accuracy for action dimension zero in the Cartpole environment for both PE-DS and calibrated PE-DS. This figure shows that the calibrated PE-DS model is more accurate, has tighter uncertainty bounds, and is better calibrated, especially in earlier trials. Interestingly, we also

observe a superior expected return for calibrated PE-DS for earlier trials in Figure 2, suggesting that being calibrated is correlated with improvements in model-based prediction and planning.

## 6. Discussion

**Limitations.** Algorithm 2 calibrates the forecasts $F_t$ by using the empirical cumulative frequency of the outputs $\{F_j(y_j)\}_{j=1}^{T}$ across the entire dataset. This kind of calibration is effective at correcting uncertainty predictions that are systematic across all the CDF distributions output by the model.

A potential failure mode for this method is when not all forecasts are not from the same family of distributions. For example, consider a contextual bandits setting with 1-dimensional context vectors where the reward for an arm is Gaussian for all contexts $x < 10$, but some asymmetric distribution for all other contexts. If the agent observes more contexts in the first range, it will be biased towards recalibrating to Gaussian distributions, which could lead to calibrated, but diffuse confidence intervals.

Another limitation of the method is its scalability to high-dimensional spaces. In our work, the uncalibrated forecasts were fully factored, and could be recalibrated component-wise. For non-factored distributions, recalibration is computationally intractable and requires approximations such as ones developed for multi-class classification (Zadrozny & Elkan, 2002).

Finally, it is possible that uncalibrated forecasts are still effective if they induce a model that correctly ranks the agent's actions in terms of their expected reward (even when the estimates of the reward themselves are incorrect). Such a model can be useful for planning, but may not yield the other benefits of a calibrated model, such as interpretability and safety guarantees.

**Extensions to Safety.** Calibration also plays an important role in the domain of RL safety. In this setting, an agent must learn about the environment while ensuring it does not enter states that are "unsafe". In situations where the agent is planning its next action, if it determines the 90% confidence interval of the predicted next state to be in a safe area but this confidence is miscalibrated, then the agent has a higher chance of entering a failure state.

Recalibration can therefore improve the safety of model-based reinforcement learning agents, as has been suggested in previous work (Berkenkamp et al., 2017).

## 7. Related Work

**Model-based Reinforcement Learning.** Compared to model-free RL, model-based RL has the potential to utilize the learned model to reduce sample complexity. This is crucial where decisions made in the real environment could be costly or harmful, such as robotics (Chua et al., 2018), dialogue systems (Singh et al., 2000), education (Rollinson & Brunskill, 2015), scientific discovery (McIntire et al., 2016), or conservation planning (Ermon et al., 2012). However, one particular challenge to model-based RL is the model bias. An variety of solutions have been proposed to reduce model bias, such as model ensembles (Clavera et al., 2018; Kurutach et al., 2018; Depeweg et al., 2016; Chua et al., 2018) or combining with model-free approaches (Buckman et al., 2018). Our approach is orthogonal to these techniques, as we perform recalibration over an existing model.

**Calibration.** Two of the most widely used calibration procedures are Platt scaling (Platt et al., 1999) and isotonic regression (Niculescu-Mizil & Caruana, 2005). They can be extended from binary to multi-class classification (Zadrozny & Elkan, 2002), to structured prediction (Kuleshov & Liang, 2015), and to regression (Kuleshov et al., 2018). Calibration has recently been studied in the context of deep neural networks (Guo et al., 2017b; Gal et al., 2017; Lakshminarayanan et al., 2017a), identifying important shortcomings in their uncertainties.

**Probabilistic forecasting.** Calibration has been studied extensively in statistics (Murphy, 1973; Dawid, 1984) as a criterion for evaluating forecasts (Gneiting & Raftery, 2007), including from a Bayesian perspective Dawid (1984). Recent studies on calibration have focused on applications in weather forecasting (Gneiting & Raftery, 2005), and have led to implementations in forecasting systems (Raftery et al., 2005). Gneiting et al. (2007) introduced a number of definitions of calibration for continuous variables, complementing early work on classification (Murphy, 1973).

## 8. Conclusion

Probabilistic models of the environment can significantly improve the performance of reinforcement learning agents. However, proper uncertainty quantification is crucial for planning and managing exploration/exploitation tradeoffs. We demonstrated a general recalibration technique that can be combined with most model-based reinforcement learning algorithms to improve performance. Our approach is widely applicable, leads to minimal computational overhead, and empirically results in significant performance improvements across a range of tasks.

## Acknowledgments

This research was supported by NSF (#1651565, #1522054, #1733686), ONR (N00014-19-1-2145), AFOSR (FA9550-19-1-0024), Amazon AWS, and Lam Research.

## References

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002. ISSN 0885-6125. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 908–918. Curran Associates, Inc., 2017.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Buckman, J., Hafner, D., Tucker, G., Brevdo, E., and Lee, H. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pp. 8234–8244, 2018.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 4759–4770. Curran Associates, Inc., 2018.

Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., and Abbeel, P. Model-based reinforcement learning via meta-policy optimization. *arXiv preprint arXiv:1809.05214*, 2018.

Dawid, A. P. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, 147:278–292, 1984.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

Ermon, S., Conrad, J., Gomes, C. P., and Selman, B. Playing games against nature: optimal policies for renewable resource allocation. 2012.

Foster, D. P. and Vohra, R. V. Asymptotic calibration. *Biometrika*, 85(2):379–390, 1998.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, 2016a.

Gal, Y. and Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027, 2016b.

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3581–3590, 2017.

Gammerman, A. and Vovk, V. Hedging predictions in machine learning. *The Computer Journal*, 50(2):151–163, 2007.

Gneiting, T. and Raftery, A. E. Weather forecasting with ensemble methods. *Science*, 310(5746):248–249, 2005.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69 (2):243–268, 2007.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017a. URL http://arxiv.org/abs/1706.04599.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017b.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Higuera, J. C. G., Meger, D., and Dudek, G. Synthesizing neural network controllers with probabilistic model based reinforcement learning. *arXiv preprint arXiv:1803.02291*, 2018.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269. IEEE, 2017.

Jiang, X., Osl, M., Kim, J., and Ohno-Machado, L. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19(2):263–274, 2011.

Kuleshov, V. and Ermon, S. Estimating uncertainty online against an adversary. In *AAAI*, pp. 2110–2116, 2017.

Kuleshov, V. and Liang, P. Calibrated structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Kuleshov, V., Fenner, N., and Ermon, S. Accurate uncertainties for deep learning using calibrated regression. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2796–2804, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/kuleshov18a.html.

Kull, M. and Flach, P. Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 68–85. Springer, 2015.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2017a.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017b.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pp. 661–670, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772758. URL http://doi.acm.org/10.1145/1772690.1772758.

McIntire, M., Ratner, D., and Ermon, S. Sparse gaussian processes for bayesian optimization. In *UAI*, 2016.

Murphy, A. H. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4):595–600, 1973.

Niculescu-Mizil, A. and Caruana, R. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632, 2005.

Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Raftery, A. E., Gneiting, T., Balabdaoui, F., and Polakowski, M. Using bayesian model averaging to calibrate forecast ensembles. *Monthly weather review*, 133(5):1155–1174, 2005.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Rollinson, J. and Brunskill, E. From predictive models to instructional policies. *International Educational Data Mining Society*, 2015.

Singh, S. P., Kearns, M. J., Litman, D. J., and Walker, M. A. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, pp. 956–962, 2000.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 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.

Van Roy, B., Bertsekas, D. P., Lee, Y., and Tsitsiklis, J. N. A neuro-dynamic programming approach to retailer inventory management. In *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, volume 4, pp. 4052–4057. IEEE, 1997.

Vermeulen, S. J., Campbell, B. M., and Ingram, J. S. Climate change and food systems. *Annual Review of Environment and Resources*, 37, 2012.

Zadrozny, B. and Elkan, C. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 609–616, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655658.

Zadrozny, B. and Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 694–699, 2002.