Distribution Calibration for Reinforcement Learning

Ethan Yu **Marwan Gedeon Achi**

Abstract

Model-based reinforcement learning (MBRL) offers a powerful framework for efficient planning and policy evaluation by leveraging transition models to approximate environmental dynamics. Central to its success is the reliability of these models, which depends on their ability to produce calibrated predictions—where predicted probabilities align with observed outcomes. Calibration plays a critical role in enabling robust decision-making in uncertain and dynamic environments, facilitating the development of better policies for maximizing cumulative rewards. By quantifying uncertainty through calibrated distributions over future states, MBRL can achieve greater accuracy in both predictions and policy optimization. This work investigates the interplay between transition model calibration and policy performance, highlighting its implications for advancing MBRL in real-world applications.

1 Introduction

Model-based reinforcement learning (MBRL) leverages transition models to approximate environmental dynamics, enabling efficient planning and policy evaluation. However, the reliability of these models hinges on their calibration — ensuring that predicted probabilities align with observed outcomes. Calibration is particularly critical for improving decision-making in uncertain and dynamic environments. In particular, RL frameworks that seek to find the optimal policy for maximizing cumulative reward rely on a transition model to estimate expected reward under the evolution of their environment; thus finding 'good' transition models leads to better policies. Moreover, in the context where transition models output distributions over future states, calibrating these distributions allow us to more accurately quantify uncertainty in our predictions and policy choices.

This paper focuses on methods to calibrate transition models in MBRL, reviewing approaches proposed by Kuleshov et al. (2018) [4] and Malik et al. (2019) [5]. We begin by reviewing the formulation of model-based RL as a Markov Decision Process. Then, we define calibration and its extension [4] to the distributional setting and examine the impact of distributional calibration on policy performance. Finally, we review the construction of calibration datasets [5] and the problem of fitting recalibration functions on our base model, employing techniques such as isotonic regression and GP-Beta recalibration [9].

We validate our approach on inventory management. Our results demonstrate that calibration significantly improves both cumulative rewards and resource efficiency in model-based planning methods, even in the presence of transition model weaknesses. For instance, isotonic regression calibration achieves a 15% reduction in waste and a 20% increase in rewards compared to uncalibrated methods.

Interestingly, despite an average mean squared error (MSE) of 4.7 in the Bayesian neural network's predictions, calibration methods were able to substantially enhance planning outcomes. In particular, isotonic regression consistently outperformed baseline (and GP-Beta calibration) due to its flexibility in handling heavily skewed transition distributions. For context, GP-Beta assumes that the base distribution is Guassian.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Our findings underscore the critical role of calibration in aligning predicted probabilities with observed outcomes, enabling robust decision-making under uncertainty. Furthermore, the results suggest that transition model calibration holds significant potential for improving model-based reinforcement learning algorithms with minimal computational and implementation overhead.

2 Background

2.1 Motivations for Model-based RL

Reinforcement learning (RL) provides a general framework for solving sequential decision-making problems where an agent interacts with an environment to maximize cumulative rewards. By learning optimal policies through exploration, RL has achieved remarkable success in various domains, including robotics, games, and scientific discovery. The theoretical appeal of RL lies in its generality: given a sufficiently rich representation of the environment, RL can solve any problem reducible to a Markov Decision Process (MDP).

In an ideal scenario, online RL—where agents interact directly with the environment to collect data, improve policies, and learn dynamics—would enable agents to converge to optimal policies. However, practical considerations often render online RL infeasible for many real-world applications. These challenges include high sample complexity, the prohibitive cost of running experiments in physical environments, and the risk of unsafe exploration during learning. For example, in domains such as autonomous driving or healthcare, conducting exploratory actions directly in the environment may lead to catastrophic outcomes.

This tension between the theoretical elegance of online RL and the practical constraints of real-world environments motivates the development of more efficient and safer alternatives. Model-based reinforcement learning (MBRL) addresses these concerns by introducing a transition model $\hat{T}$ that approximates the environment's dynamics. By leveraging this learned model, MBRL enables agents to plan and evaluate policies without requiring extensive interaction with a real environment. In this way, MBRL offers a promising path forward, balancing the need for sample efficiency with the safety and practicality required for deployment in real-world systems.

2.2 Transition Models for Model-based Reinforcement Learning

Model-based RL frameworks aim to optimize the actions of an agent that interacts sequentially in an environment, where algorithms optimize over one or both of the following two objects:

1. A *transition model* $\hat{T}$ which models the expected evolution of the environment by producing a one-step ahead probability forecast over states, given the current state and action chosen by the agents.
2. A *(probabilistic) policy* π that samples an action at each time step based on the current state.

Generally, the evolution of the environment is assumed to be governed by some transition function T, but model-based RL systems usually try to learn a $\hat{T}$ that approximates T. Improving this learned $\hat{T}$ through calibration is the topic of this paper.

The model-based RL framework can be more formally defined as a Markov Decision Process [6], with the tuple:

$$(S, A, T, r, \gamma)$$

Here we have:

- S is the state space, which may be discrete or continuous. For this project, we typically have $S \subseteq \mathbb{R}$
- A is the action space (also discrete or continuous)
- $T : S \times A \to \mathcal{P}(S)$ is the (ground truth) transition model (for this paper we assume that the output is a probability distribution in the form of a CDF over S)
- $r : S \to \mathbb{R}$ is a reward function for our agent
- $\gamma \in [0, 1]$, a discounting parameter

In this setting, we usually assume that T and r are not known (although in this paper, we fix a particular r for each problem). The Markov Decision Process iterates for rounds $t = 0, 1, 2, ...K$, and the goal of the algorithm is to find the optimal policy $\pi : S \to \mathcal{P}(\mathcal{A})$ which maximizes our expected reward. In particular, for a time horizon K, define the expected reward of π from state s at time t as:

$$E[\pi^{(s)}] = E\left[\sum_{k=0}^{K} \gamma^k r(s_{t+k}, a_{t+k}) | s_t = s\right]$$

Where in the above, for all j in $\{0, ..., K\}$, $a_{t+j} \sim \pi(\cdot|s_{t+j})$ and $a_{t+j+1} \sim T(\cdot|s_{t+j}, a_{t+j})$. For instance, given an initial state $s_0 = s$ we may want to find a policy:

$$\pi^* = \text{argmax}_{\pi:S\to\mathcal{P}(A)} E[\pi^{(s_0)}] = \text{argmax}_{\pi:S\to\mathcal{P}(A)} E\left[\sum_{t=0}^{K} \gamma^k r(s_t, a_t) | s_0 = s\right]$$

When the time horizon K is set to ∞, we say that we are in the infinite time horizon setting.

Note that optimizing for this policy π^* directly requires knowledge of the transition function T and reward function r. In this paper, we fix a particular reward function r for each problem. The transition function T, however, remains unknown.

Instead, using data from previous interactions, we train a transition model $\hat{T}$ and are interested in making this model as 'good' (close to the true transition function T) as possible.

With $\hat{T}$ close to T, we can hope to optimize for π by estimating expected rewards with respect to evolution of the environment under our transition model $\hat{T}$.Training good transition models (ideally, with accurate uncertainties in their predictions) is a key problem in model-based RL [4]; as we will see, distributional calibration is a computationally cheap way to attempt to improve any uncalibrated transition model $\hat{T}$.

2.3 Calibration

2.3.1 Calibrating a Single Model

Calibration generally measures the accuracy of a model *conditional on its own prediction.* More formally, consider an arbitrary distribution $D_{X,Y}$ over (x, y) pairs with $x \in X$ and $y \in Y \subseteq \mathbb{R}$ (a natural case is $Y \subseteq [0, 1]$. A model (with a range of finite cardinality) $f : X \to \mathbb{R}$ is (mean) calibrated if for all values v in the range of f, we have:

$$E_{D_{X,Y}}[y|f(x) = v] = v$$

That is, when f predicts $f(x) = v$, the expected value of y is indeed v. Note that calibration is not necessarily powerful on its own as it does not account for the variance of f's conditional predictions, in particular, for any distribution $D \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$, the constant model:

$$f_c(x) = E_{y\sim D}[y]$$

is by definition calibrated, without making any use of the features in X. So, calibration can be viewed as an augmentation or additional desirable property in a model that has learned useful relationships between features X and labels Y. More formally, we might ask for 'sharpness' [4]: that the variance $\text{Var}_{(x,y)\sim D}[y|f(x) = v]$ is small for all values v.

For the definition of calibration above, given enough data, simple algorithms can take any uncalibrated base model f and return a calibrated model $\hat{f}$ with strictly lower squared error and a range that is not any larger [8]. For instance, given f, we can estimate for all v the function $c(v) = E_{(x,y\sim D}[y|f(x) = v]$ and return the model:

$$\hat{f}(x) = c(f(x))$$

which will be calibrated by definition [8].

2.3.2 Motivations for Calibrating Transition Models

Transition models in model-based reinforcement learning (MBRL) predict the probabilistic evolution of the environment given a state and action. Ensuring these models are calibrated, meaning their predictions align with observed outcomes, is crucial for several reasons:

1. **Reliable Decision-Making:** A calibrated transition model provides accurate probabilities for future states, enabling agents to make informed decisions that maximize rewards while minimizing risks.
2. **Uncertainty Management:** Calibration ensures that the model's uncertainty estimates are meaningful, providing a realistic sense of confidence in its predictions. This is particularly important when planning under uncertainty or optimizing safety-critical tasks.
3. **Improved Planning Accuracy:** Planning algorithms depend on transition probabilities to simulate and evaluate future trajectories. Calibrated models reduce errors in these simulations, leading to more effective policies.
4. **Alignment with Observed Dynamics:** By aligning predicted probabilities with the true dynamics of the environment, calibrated models are less prone to overconfidence or underconfidence, which can distort policy learning and evaluation.
5. **General Applicability:** In dynamic, real-world environments, the ability of a model to generalize to unseen states and actions is enhanced by calibration, as it ensures predictions remain consistent with observed patterns.
6. **Robustness to Model Imperfections:** Calibration mitigates the impact of inaccuracies in the learned dynamics, allowing the agent to adapt and perform reliably even with an imperfect model.

Overall, calibration enhances the trustworthiness and effectiveness of transition models, making them better suited for complex decision-making tasks in uncertain and dynamic environments.

2.3.3 Calibration for Model-based RL

There are two issues with applying the definition above directly to transition models for RL, which we address in the next section using an approach developed by Kuleshov et al. in 2019 [4].

1. Calibration as we've defined it so far applies to the predictions a single model S. But we are interested in calibrating a transition model $\hat{T} : S \times A \to \mathcal{P}(S)$, which itself outputs a probability distribution in the form of a CDF F_t at time t. So calibrating our transition model $\hat{T}$ means ensuring that $F_t = \hat{T}(s_t, a_t)$ adheres to some notion of calibration for all inputs $(s, a) \in S \times A$ to $\hat{T}$
2. Perhaps more importantly, the calibration above is well defined only when our conditioning events $f(x) = v$ make sense. For a (continuous) CDF F_t, it's not immediately clear how to define conditioning events or even in what format to collect data for recalibration.

Despite these issues, calibration remains a desirable goal for transition models. In particular, Malik et. al [5] give a strong result relating calibrated transition models to optimal policies. Given a Markov Decision Process , a policy $\pi \in S \to \mathcal{P}(A)$, a state $s \in S$, recursively define the value iteration V from s as the expected reward under π starting at s; that is:

$$V(s) = E_{a \sim \pi(\cdot|s)} \left[\sum_{s' \in S} T(s'|s, a) \cdot r(s') + V(s') \right]$$

With this definition we have the following theorem:

Theorem (1): *Policy Value of Calibrated Transitions*
Let (S, A, T, r, γ) be a Markov Decision Process with ground truth transition model T. Let $V(\pi) = E_{s \sim \sigma_\pi}[V(s)]$, where σ_π is the stationary decision induced by π, be the expected value iteration under T. Then if we calculate $V'(\pi)$ by replacing T with a ***calibrated*** transition model $\hat{T}$, we have:

$$V(\pi) = V'(\pi)$$

This is a strong result; it implies that a calibrated transition model is sufficient to identify the optimal policy defined on its induced stationary distribution.

With these calibration benefits in mind, we move to the definition of distributional calibration over time proposed by Kuleshov et al. [4].

3 Distributional Calibration

3.1 Defining Distributional Calibration

In this section we describe the formulation of Kuleshov et. al [4] for calibrated regression in the transition model setting, where $\hat{T}$ outputs a CDF F_t over the action space S given input $(s_t, a_t) \in S \times A$. As we will see, solving distributional calibration in this setting relies on estimating the inverse CDF of these output distributions, thus we denote the inverse CDF as:

$$F^{-1}(p) = \inf\{s \in S : p \leq F(s)\}$$

Definition (1): *Calibrated Transition Model*
With this, we can define the notion of calibration for a transition model $\hat{T}$ which outputs CDFs $F_t = \hat{T}(s_t, a_t)$ over the state space S. F_t is distributionally calibrated, if as $N \to \infty$:

$$\frac{\sum_{t=0}^{N-1} \mathbb{I}[s_{t+1} \leq F_t^{-1}(p)]}{N} \to p \text{ for all } p \in [0, 1] \tag{1}$$

Essentially, this requires that the empirical and predicted CDFs match as we keep forecasting using the transition model.

Under this definition, Kuleshov et al. [4] also propose the variance $\text{Var}[F_t]$ of a random variable drawn from CDF F_t as a measure of sharpness analogous to sharpness in the discrete state space setting.

3.2 Recalibrating Forecasters

Note that the definition above for a calibrated transition model is adapted from Kuleshov et al. 2018 [4]. In their paper, the CDF F_t is given by a forecaster H over $(x, y) \in \mathcal{X} \times \mathcal{Y}$ pairs, and the output $F_t = H(x_t)$ is a CDF over the target variable $y_t \in \mathcal{Y}$.

While these definitions are almost equivalent, the more general formulation from Kuleshov et al. allows them to consider the case where $(X, Y) \sim D$ for some distribution D (note that this is not necessarily possible for the RL setting, as the state, action, next action tuples may depend on the transition model). Given $(X, Y) \sim D$, they apply the above definition to conclude that H is distributionally calibrated if:

$$\mathbb{P}_D(Y \leq F_X^{-1}(p)) = p, \forall p \in [0, 1]$$

where $F_X = H(X)$ is the forecast at X.

This conclusion leads to the following observation.

1. Suppose that we want to train a recalibration function $R : [0, 1] \to [0, 1]$ such that $R \circ F_t$ is calibrated for all $t = 1, ..., N$
2. If we know that at all time steps t, $(X, Y) \sim D$ for some distribution D, then setting $R(p) = \mathbb{P}_D(Y \leq F_X^(p)), \forall p \in [0, 1]$ guarantees that $R \circ F_t$ is calibrated.
3. Given a dataset $\{(x_t, y_t)\}_{t=1}^N$ and an uncalibrated forecaster H, we can estimate $R(p) = \mathbb{P}_D(Y \leq F_X^{-1}(p))$ for any value of p.

Together, these observations give rise to the following algorithm which fits a regression on predicted observation probabilities to find a calibrating function R [4].

Algorithm 1 Recalibration of Regression Models

Input: Uncalibrated model $H : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ and calibration set $S = \{(x_t, y_t)\}_{t=1}^{N}$.
Output: Auxiliary recalibration model $R : [0, 1] \to [0, 1]$.

1. Construct a recalibration dataset:

$$\mathcal{D} = \left\{ \left([H(x_t)](y_t), \hat{R}([H(x_t)](y_t)) \right) \right\}_{t=1}^{T},$$

where

$$\hat{R}(p) = |\{y_t \mid [H(x_t)](y_t) \leq p, t = 1, \ldots, N\}|/N$$

2. Train a model R (e.g., isotonic regression) on $\mathcal{D}$.

So the goal is to ensure that the final CDF satisfies $R(F_{x_t}(y)) = \mathcal{P}_D(Y \leq y|X = x_t)$ for all output CDFs F_t. In particular, in the case where $(X, Y) \sim D$ for some distribution D, this algorithm is trying to learn the function $R : [0, 1] \to [0, 1]$ such that $R(p) = \mathbb{P}_D(Y \leq F_X^{-1}(p))$; i.e. $(R \circ F_{x_t})(y)$ returns p such that:

$$p = \mathcal{P}_D(Y \leq y|X = x_t)$$

So the correct R will give us $R \circ F_{x_t}$ which accurately measures target quantiles [4].

Finding this R is done by fitting a regression model (see the next section) to $(p, \hat{R}(p))$ where:

$$\hat{R}(p) = \frac{|\{y_t \mid [H(x_t)](y_t) \leq p, t = 1, \ldots, N\}|}{N}$$

measures the empirical probability that y_t falls below the pth quantile of F_{x_t}. Thus R maps target quantiles p output by F_{x_t} to the 'real' quantile corresponding to the uncalibrated prediction p.

3.3 Recalibration Datasets for RL

As noted in the section above, since the state, action, next state tuples in the RL model are not necessarily drawn from some joint distribution, (and in fact may depend on the transition model $\hat{T}$ itself) we cannot apply the analysis where $(X, Y) \sim D$ for some distribution D. In particular, this implies that we cannot frame recalibration directly as finding $R(p) = \mathbb{P}_D(Y \leq F_X^{-1}(p))$, since without a distribution D over state, action, next state, neither the label (next state) or input (state, action) distributions are well defined.

However, Malik et. al in 2019 [5] showed that recalibrating transition models using a similar approach, where we augment a base model $\hat{T}$ with a recalibration function R, yields more calibrated forecast distributions and leads to better learned policies (measured by cumulative reward). In our experiments, we follow their approach while using isotonic regression to learn R and observe improvements to both calibration and baseline performance (see results section).

Below, we reformulate Algorithm (1) in the context of model-based RL. They key is to modify the targets for our regression to R, so that:

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

measures the empirical number of observed transitions from $(s_t, a_t) \to s_{t+1}$ falls below the pth quantile of F_{s_t,a_t}, averaged over all time steps $t = 1, ..., n$. As in the distributional setting, the goal is to find R that maps from uncalibrated predictions to true quantiles, the latter of which is estimated empirically. So our algorithm is:

4 Calibrated Regression Methods

Given the approach outlined above for defining distributional calibration and collecting datasets to calibrate a base model R, the remaining problem is learning a function $R : [0, 1] \to [0, 1]$. Given

Algorithm 2 Recalibration for Transition Models [5]

Input: Potentially uncalibrated transition model $\hat{T} : S \times A \to \mathcal{P}(S)$ which outputs CDFs $F_{s,a} : S \to [0, 1]$, and dataset of observed transitions $D = \{(s_t, a_t, s_{t+1})\}_{t=1}^{N}$.
Output: Auxiliary recalibration model $R : [0, 1] \to [0, 1]$.

1. Construct a recalibration dataset:

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

where

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

2. Train a model R (e.g., isotonic regression) on $\mathcal{D}$.

that this is a simple uni-variate regression task, there are plenty of model classes to choose from. However, a key property to preserve the structure of a CDF F under the composition $R \circ F$ is that R be isotonic (that is, monotonically non-decreasing) over the input space. This is because F measures $\mathcal{P}(\mathcal{Y} \leq \dagger)$ for some target variable y, and by definition this cannot decrease as the input y increases.

In the case of calibrating classification distributions, a standard approach is Platt scaling [7]. This is often applied to turn SVM margins into class probabilities. Platt scaling typically fits a sigmoid regression with two parameters; the approach of Kuleshov et. al [4] extends the Platt scaling approach to continuous variables.

While Malik et. al [5] achieve good results by applying the extended Platt scaling approach to RL, we use two different models for learning R, with the intent of (1) testing alternative approaches (2) testing an approach that doesn't expect sigmoidal residuals (as Platt scaling typically does).

4.1 Isotonic Regression

The first method we use is isotonic regression [1], which fits R to the class of isotonic functions mapping to $[0, 1]$. This is a powerful, non-parametric approach, and thus may require lots of data but can fit any distribution with enough $i.i.d$ samples. With isotonic regression we are able to significantly improve calibration and base error for RL tasks (see experiments and results).

Formally, given a set of n observations (p_t, r_t) where $p_t \in [0, 1]$ and $r_t \in [0, 1]$, the isotonic regression solves the following optimization problem:

$$\min_{R} \sum_{t=1}^{N} (r_t - R(p_t))^2 \quad \text{subject to} \quad R(p_{t'}) \leq R(p_t) \; \forall \; p_{t'} < p_t,$$
$$R(p_t) \in [0, 1] \quad \forall i.$$

The isotonic regression problem can be efficiently solved using the *pool adjacent violators algorithm* [1]. PAVA operates iteratively to adjust any violations of the monotonicity constraint.

PAVA Algorithm:

1. Initialize $R(p_t) = r_t$ for all t.
2. Traverse the data points sequentially. For any adjacent pair (p_t, p_{t+1}) where $R(p_t) > R(p_{t+1})$, merge these points into a single "block" and replace their values with the weighted average:

$$R(p_t) = R(p_{t+1}) = \frac{w_t r_t + w_{t+1} r_{t+1}}{w_t + w_{t+1}},$$

where w_t and w_{t+1} are the weights associated with r_t and r_{t+1}, respectively (default $w_t = 1$ if no weights are specified).

3. Repeat the merging process iteratively, ensuring that monotonicity constraints are satisfied for all adjacent pairs. When a block is formed, it is treated as a single entity for subsequent merges.
4. Stop when all monotonicity constraints are satisfied.

The computational complexity of PAVA is $O(n)$, making isotonic regression practical even for large datasets. In fact, compared to the problem of training baseline transition models, this regression problem is computationally cheap.

This means that given a base model $\hat{T}$, we can 'easily' recalibrate by fitting R on the regression dataset $\{(p_t, \hat{R}(p_t)\}_{t=1}^{N}$ defined in section 3.3, using Isotonic Regression. As we see in our results, this also leads to transition models with better calibration and better overall performance.

4.2 GP-Beta Method

So far, we have looked at recalibration methods for finding $R : [0, 1] \to [0, 1]$. But many forecasting algorithms output distributions that are normal and parameterized by a mean and variance. Consider the setting where $\hat{T}(s_t, a_t) = N(\mu_t, \sigma_t)$ at all time steps t. While these distributions may all be completely unique, we know that they are all normal and can thus define 'distance' between the distributions at two different time steps; that is, we can use any kernel distance function over (μ_t, σ_t) to identify similar CDFs. In this setting, Song et al. [9] make the following observations:

1. Since the CDF $N(\mu_t, \sigma_t)$ at time t is a distribution, it can be calibrated with a beta calibration map $[0, 1] \to [0, 1]$. These maps take three scalar parameters (a, b, c), so we can improve $N(\mu_t, \sigma_t)$'s calibration by finding the corrects params (a_t, b_t, c_t)
2. The CDF at each time step t is unique, so we don't directly have data for learning (a_t, b_t, c_t). But we can 'borrow' information from other CDFs 'close' to (μ_t, σ_t).
3. In particular we can assume that the parameters are drawn from three latent functions (w_a, w_b, w_c), and learn to encode CDFs into the latent space using a multi output Gaussian Process.

These observations allow them to develop the **GP-Beta** method for calibrating distributional outputs which follow normal distributions. From Song et. al [9], we see empirically that this Gaussian-process based approach, when applied to normal forecasts, is able to leverage similarities between 'similar' distributions to achieve distributional calibration.

Thus in the context of reinforcement learning, when our transition models produce Gaussian PDFs, the GP-beta method is a promising direction for achieving calibration. Note, however, that this approach is a significant step-up in compute compared to simpler schemes like isotonic regression, making recalibration more 'expensive'.

In our experiments, we applied the GP-beta method on output CDFs produced by variational dropout on a Bayesian Neural Network [3]. To pass these CDFs to the GP-beta algorithm in $N(\mu, \sigma)$ format, we estimate the mean and variance empirically. The results, in our case, didn't show better calibration or base performance for our models (see results section). We suspect that our output CDFs were not truly normally distributed; further work on model-based RL where transition models are explicitly Gaussian may show additional empirical benefits to the GP-beta method in the context of RL.

5 Applications of Model-based Reinforcement Learning

Repository with implementation and model weights: https://github.com/ethayu/CalibratedRL.

5.1 Inventory Management: A Practical Application

Inventory management for perishable goods poses unique challenges due to the tradeoff between over-ordering, which leads to spoilage, and under-ordering, which results in stockouts and unmet demand. Addressing these challenges efficiently has significant implications for both profitability and

sustainability. Through the Corporación Favorita Grocery Sales dataset [2], we evaluate the effects of calibration on policy performance.

5.1.1 Dataset and Preprocessing

We use the Corporación Favorita Grocery Sales dataset, which contains daily sales data for thousands of items across multiple stores. Due to computational constraints, we focus on the top 100 selling items across the top 100 stores, identified using the following steps:

1. Top Stores: Aggregated sales data by store and selected the 100 stores with the highest total sales.
2. Top Items: Further filtered the data for the 100 most sold items within these stores.

To enable learning of temporal patterns, we engineered features from the sales data:

1. Rolling Averages: Computed 7-day, 14-day, and 28-day rolling averages of daily sales for each item.
2. Periodic Features: Added sine and cosine encodings for the day of the year to capture seasonality.
3. Day and Week Features: Included categorical encodings for the day of the week and week of the year.

The processed data was split into training (before June 1, 2017) and testing (June 1, 2017, onward) datasets. Missing values in rolling averages were filled with zeros to handle edge cases.

5.1.2 Problem Formulation as a Markov Decision Process (MDP)

The inventory management task is formulated as a Markov Decision Process (MDP) defined by the tuple (S, A, P, r):

1. State Space (S): Each state $s \in S$ is represented by a tuple (d_s, q_s), where d_s is the day feature (encoded as a calendar day), and q_s is the inventory level feature. Practically, d_s is the "Periodic" and "Day and Week" features from the prior section, and q_s is the "Rolling Average" feature from the prior section.
2. Action Space (A): Actions $a \in A$ correspond to ordering decisions, indicating the number of units ordered for the day. Actions are bounded by realistic constraints on store inventory limits and vendor supply capacities.
3. Transition Dynamics (P): The transition dynamics are stochastic, governed by:
   (a) Demand $D(s)$: A random variable representing daily customer demand for the item.
   (b) Sales: Calculated as $\max(q_s - D(s), 0)$, accounting for stockouts.
4. Reward Function (r): The reward balances profit from sales against costs associated with ordering and spoilage. The goal is to maximize the total reward over a defined time horizon. To be specific, we define reward as $-(w+s)$, where w (waste) is $\max(\texttt{inventory\_level} - \texttt{demand}, 0)$ and s (stockout) is $\max(\texttt{demand} - \texttt{inventory\_level}, 0)$.

5.1.3 Transition Model: Bayesian Neural Networks

For each item, a Bayesian neural network is used as the transition model $\hat{T}$ to predict the distribution of next-day demand. The architecture and implementation details are as follows:

1. Architecture:
   (a) Input Layer: Accepts 7 features, including rolling averages, periodic encodings, and categorical day/week information.
   (b) Hidden Layers: A stack of five fully connected layers with ReLU activations and 128 units each.
   (c) Dropout: Applied with a rate of 0.5 during training to approximate Bayesian uncertainty.

(d) Output Layer: Predicts a scalar value for the expected demand.

2. Probabilistic Predictions:

   (a) Monte Carlo (MC) Dropout: Samples predictions from the network by enabling dropout during inference. As specified in variational dropout [3], this generates a distribution of predictions, allowing uncertainty estimation. For our experiments, we generated this emperical distribution with 300 samples.

Note that since there are 100 items, we therefore trained 100 networks.

5.2 Model Rollouts for Planning and Evaluation

In this section, we describe the application of learned transition models to plan and evaluate inventory management policies. Two distinct approaches are implemented for generating rollouts: a heuristic-based planner and a model predictive control (MPC) planner. These planners leverage probabilistic predictions from the Bayesian neural network transition model.

5.2.1 Heuristic Planning

The heuristic planner operates under the assumption that inventory should be adjusted to a safety factor times the expected demand to mitigate stockouts. The implementation details are as follows:

- **Expected Demand Estimation:** Using the probabilistic forward pass of the transition model, the expected demand for the next day is computed.
- **Safety Factor Adjustment:** The expected demand is scaled by a safety factor (1.5 in our experiment). The resulting value is interpreted as the target inventory level.
- **Action Calculation:** The planner determines the order quantity (action) as the difference between the target inventory level and the current inventory. Negative actions are clamped to zero to ensure no invalid orders are issued.

The heuristic planner is computationally efficient, making it suitable for real-time decision-making. However, it may not optimize long-term objectives, as it considers only the immediate future.

5.2.2 Model Predictive Control (MPC) Planning

The MPC planner extends the heuristic approach by optimizing over a finite planning horizon. At each decision point, it evaluates multiple trajectories to identify the best initial action. Key components of the implementation include:

- **Horizon and Trajectory Sampling:** The MPC planner simulates rollouts of length h (horizon) for N sampled trajectories. At each step, the planner uses the transition model to predict the demand distribution for the next state. Due to computational limits, we set $h = 5$ and $N = 20$.
- **Action Selection:** For each trajectory, the planner determines an action based on the sampled demand and current inventory. The action minimizes penalties associated with waste (excess inventory) and stockouts (unmet demand).
- **Reward Accumulation:** Rewards are computed as the negative sum of waste and stockouts. The trajectory with the highest cumulative reward determines the action to be taken in the current state.

This approach explicitly incorporates long-term planning, allowing it to balance immediate actions with future outcomes. The MPC framework's flexibility also enables the incorporation of additional constraints or objectives.

5.2.3 Evaluation Protocol

The rollouts generated by both planners are evaluated on test data to assess their performance. The evaluation process is structured as follows:

- **Metrics:** Key metrics include the total number of units shipped, wasted inventory, stockouts, percentage of waste and stockouts relative to shipments, and cumulative reward.
- **Simulation Procedure:** For each item, the planners process the test dataset sequentially, updating inventory levels based on proposed actions and observed demand (from testset). Waste and stockouts are recorded at each step. Due to time constraints, we only tested 10 rollouts for each of the 100 items. Recall that for each rollout, we must plan 6 times for each of the policy and calibration method combinations. For MPC in particular, since we have a simulation horizon of 5 and 200 sampled trajectories (each of which samples 300 times to generate the distribution), this amounts to $10 \times 100 \times 5 \times 200 \times 300 = 300,000,000$ calls to the transition model.
- **Experimental Setup:** The evaluation is conducted for the top 100 items, using the test dataset generated during preprocessing. Results are aggregated across items via averaging.

5.3 Calibration Model Training and Usage

Calibration refines the probabilistic predictions generated by the Bayesian neural network transition model, ensuring alignment between predicted probabilities and observed outcomes. Two methods, Isotonic Regression and GPBeta Regression, are employed to achieve calibration.

5.3.1 Training Calibration Models

1. We perform inference of the trained transition models on the train set, and record the empirical PDFs.
2. Then, we compute the empirical cumulative density function (CDF), denoted as $F_{\text{emp}}(y)$.
3. Then, we aggregate the sampled distributions into the dataset $\hat{R}$ as described in Algorithm 2. We leverage the `netcal` library to run Isotonic and GPBeta Regression.
4. The calibrated models are serialized as `isotonic_item_{id}.pkl` and `gpbeta_item_{id}.pkl`.
5. Finally, we apply the calibration models R (200 of them, 2 for each of 100 items) in the matter described in Algorithm 2 ($\hat{T} \coloneqq R(\hat{T})$).

6 Results

	MPC	Heuristic	MPC Isotonic	Heuristic Isotonic	MPC GP-Beta	Heuristic GP-Beta
Shipped	179.79	243.28	148.77	206.63	181.98	247.27
Wasted	87.24	156.82	60.88	114.18	88.18	159.87
Stockouts	50.09	24.73	67.04	38.53	54.22	25.73
% Waste	48.52%	64.46%	40.92%	55.26%	48.46%	64.66%
% Stockouts	27.86%	10.16%	45.06%	18.65%	29.79%	10.40%
Reward	-137.33	-181.55	-127.92	-152.71	-142.39	-185.60

Table 1: Summary Results Across Methods

6.1 Discussion

The results presented in Table 1 provide insight into the performance of various planning methods, both with and without calibration. While the outcomes demonstrate the promise of calibration for improving model-based planning, they also highlight the inherent challenges posed by weaknesses in the underlying transition model.

6.1.1 Impact of Transition Model Weaknesses

The high levels of waste and suboptimal rewards across all methods can be attributed, in part, to the limitations of the transition model. With an average mean squared error (MSE) of 4.7, the predictions of the Bayesian neural network often deviated significantly from actual outcomes.

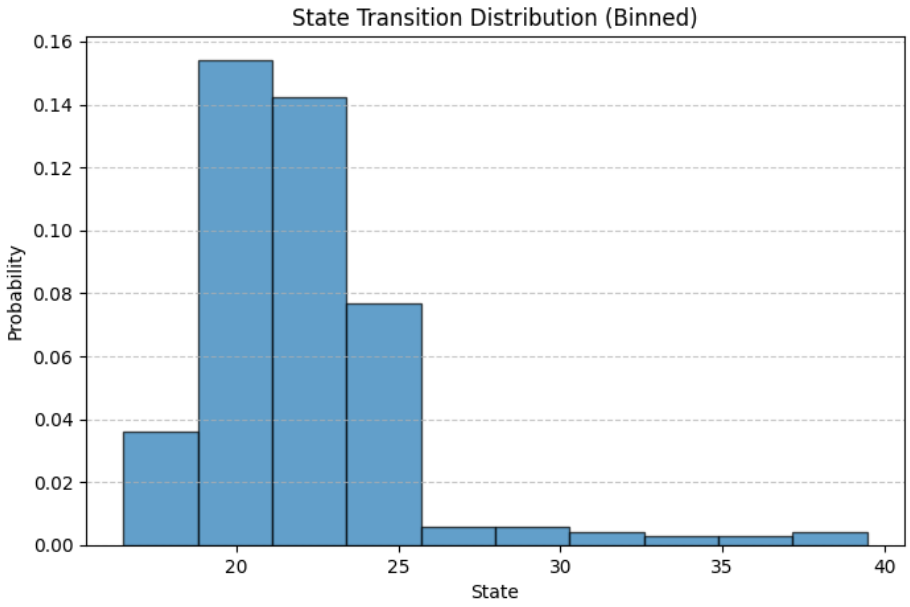

Figure 1: Transition Distribution for Item 1473482 on July 16, 2017

It is also important to note that computational constraints further limited the ability to refine the transition model. This constraint underscores the need for future work exploring more computationally efficient architectures or leveraging larger-scale resources for training.

6.1.2 Effectiveness of Calibration Methods

Despite these limitations, calibrating the transition model significantly improved performance across most metrics. Isotonic regression calibration consistently outperformed the GPBeta approach, achieving lower waste and stockout percentages and yielding higher rewards. This superiority is likely due to the flexibility of isotonic regression, which makes fewer assumptions about the underlying distribution of the transition model's outputs.

6.1.3 Insights and Future Directions

The results highlight the importance of transition model calibration for improving decision-making in model-based reinforcement learning. Calibration not only improves the alignment between predicted and observed probabilities but also enhances the robustness of planning methods, particularly under uncertainty.

Future work should aim to address the following:

- **Transition Model Improvement:** Reducing MSE through advanced architectures, feature engineering, or transfer learning could enhance the quality of the base model, leading to better overall performance. An immediate improvement on the transition model end could be utilizing a state-space model such as an LSTM, instead of forcing the model to have past context only through rolling averages.
- **Calibration Techniques:** Exploring non-parametric calibration methods beyond isotonic regression and GPBeta could provide better performance, particularly for heavily skewed distributions.

In conclusion, while the current approach demonstrates the efficacy of calibration in improving planning outcomes, the results also underscore the critical role of the transition model's quality. By addressing the identified limitations and pursuing the suggested directions, future work can achieve

more reliable and scalable solutions for model-based planning in complex real-world scenarios. Nonetheless, we have demonstrated that model calibration is beneficial for RL policy rollout.

References

[1] Jan de Leeuw, Kurt Hornik, and Patrick Mair. Isotone optimization in r: Pool-adjacent-violators algorithm (pava) and active set methods. *Journal of Statistical Software*, 32(5):1–24, 2009.

[2] Corporación Favorita, inversion, Julia Elliott, and Mark McDonald. Corporación favorita grocery sales forecasting. `https://kaggle.com/competitions/favorita-grocery-sales-forecasting`, 2017. Kaggle.

[3] Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick, 2015.

[4] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression, 2018.

[5] Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated model-based deep reinforcement learning, 2019.

[6] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey, 2022.

[7] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10, 06 2000.

[8] Aaron Roth. Uncertain: Modern topics in uncertainty estimation. Lecture notes, CIS 6200, University of Pennsylvania, 2024.

[9] Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. Distribution calibration for regression, 2019.