Online Calibrated and Conformal Prediction Improves Bayesian Optimization

Shachi Deshpande
Cornell Tech and Cornell University

Charles Marx
Stanford University

Volodymyr Kuleshov
Cornell Tech and Cornell University

Abstract

Accurate uncertainty estimates are important in sequential model-based decision-making tasks such as Bayesian optimization. However, these estimates can be imperfect if the data violates assumptions made by the model (e.g., Gaussianity). This paper studies which uncertainties are needed in model-based decision-making and in Bayesian optimization, and argues that uncertainties can benefit from calibration—i.e., an 80% predictive interval should contain the true outcome 80% of the time. Maintaining calibration, however, can be challenging when the data is non-stationary and depends on our actions. We propose using simple algorithms based on online learning to provably maintain calibration on non-i.i.d. data, and we show how to integrate these algorithms in Bayesian optimization with minimal overhead. Empirically, we find that calibrated Bayesian optimization converges to better optima in fewer steps, and we demonstrate improved performance on standard benchmark functions and hyperparameter optimization tasks.

1 INTRODUCTION

Bayesian optimization has emerged as a powerful tool for optimizing objective functions that are initially unknown and that are learned via evaluation queries (Thornton et al., 2013; Shahriari et al., 2016; Bergstra et al., 2011). In practice, querying such objectives can be expensive: for example, in hyperparameter optimization (Snoek et al., 2012), queries may involve training a machine learning model from scratch. Bayesian optimization aims to minimize the number of objective function queries by relying on a probabilistic model to guide search (Frazier, 2018). However, probabilistic models are not always accurate and may be overconfident, which slows down optimization and may yield suboptimal local optima (Guo et al., 2017).

This paper aims to improve the performance of algorithms for online sequential decision-making under uncertainty, particularly Bayesian optimization. We start by examining which uncertainties are needed in model based decision-making and argue that ideal uncertainties should be calibrated (Gneiting and Raftery, 2007; Kuleshov et al., 2018). Intuitively, calibration means that an 80% confidence interval should contain the true outcome 80% of the time. Calibration helps balance exploration and exploitation, estimate the expected value of the objective, and reach the optimum in fewer steps (Malik et al., 2019).

Enforcing calibration in sequential tasks is challenging because the data is non-stationary (it is determined by our actions). We introduce simple algorithms based on online learning (Shalev-Shwartz et al., 2012) that provably yield calibrated uncertainties, even on data chosen by an adversary. Moreover, our methods can be added to any Bayesian optimization algorithm with minimal modifications and with minimal computational and implementation overhead. Empirically, we show that our techniques improve Bayesian optimization and yield faster convergence to higher quality optima across a range of standard benchmark functions and hyperparameter optimization tasks.

Contributions In summary, this work makes the following contributions: (1) we study which uncertainties are needed in online sequential decision-making and and provide a formal analysis of the benefits of calibrated uncertainties; (2) we introduce simple algorithms than enforce calibration on non-stationary data and that can be added within any existing Bayesian optimization algorithm with minimal computational and implementation overhead; (3) we demonstrate that our methods accelerate optimization on tasks such as hyperparameter search.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

2 BACKGROUND

2.1 Calibrated & Conformal Prediction

Traditionally, predictive uncertainty in statistics is evaluated using proper scoring rules (Gneiting and Raftery, 2007). Proper scoring rules measure precisely two characteristics of a forecast: calibration and sharpness (Murphy and Winkler, 1987). Intuitively, calibration means that a 90% confidence interval contains the outcome about 90% of the time. Sharpness means that confidence intervals should be tight. Maximally tight and calibrated confidence intervals are Bayes optimal.

Calibration. Formally, suppose we have a model $M : \mathcal{X} \to \mathcal{P}(\mathbb{R})$ that outputs a probabilistic forecast $Q_x \in \mathcal{P}(\mathbb{R})$ of a target variable $y \in \mathbb{R}$ given an input $x \in \mathcal{X}$. In this paper, we will assume that Q_x is represented by a quantile function. When the target $y \in \mathbb{R}$ is continuous, calibration is often defined as $P(Y \leq Q_X(p)) = p$, $\forall p \in [0, 1]$ (Kuleshov et al., 2018). In an online setting, this definition becomes

$$\frac{\sum_{t=1}^{T} \mathbb{I}\{y_t \leq Q_t(p)\}}{T} \to p \text{ for all } p \in [0, 1] \qquad (1)$$

as $T \to \infty$, where $\mathbb{I}$ is the indicator function and $Q_t = M(x_t)$ is the forecast at time t.

Calibrated Prediction Out of the box, most models M are not calibrated. Post-hoc calibration and conformal prediction yield calibrated forecasts by adjusting predicted probabilities Q from M on a held out dataset (Shafer and Vovk, 2007; Kuleshov et al., 2018; Vovk et al., 2020).

For training a recalibrator over our probabilistic model, we compute the CDF F_t at each data-point y_t using the formulation $F_t = [\mathcal{M}(x_t)](y_t)$. This can be used to estimate the the empirical fraction of data-points below each quantile. Algorithm 1 based on based on Kuleshov et al. (2018) outlines this procedure.

Algorithm 1: Calibration of Probabilistic Model

Input: Dataset of probabilistic forecasts and outcomes $\{[\mathcal{M}(x_t)](y_t), y_t\}_{t=1}^{N}$

1. Form recalibration set $\mathcal{D} = \{[F_t, \hat{P}(F_t)\}_{t=1}^{N}$ where $F_t = [\mathcal{M}(x_t)](y_t)$ and $\hat{P}(p) = |\{y_t | [F_t \leq p, t = 1, .., N\}|/N$.
2. Train recalibrator model $\mathcal{R}$ on dataset $\mathcal{D}$.

2.2 Bayesian Optimization

Uncertainty estimation plays an important role in sequential decision-making, where we observe a sequence of inputs $x_t \in \mathcal{X}$ for $t = 1, 2, ..., T$ and choose actions $a_t \in \mathcal{A}$, after which nature reveals outcomes y_t. In such settings, an accurate probabilistic model of y_t given x_t is useful for choosing a_t.

Bayesian optimization is a sequential decision-making process that seeks to find a global minimum $x^\star \in \arg\min_{x \in \mathcal{X}} f(x)$ of an unknown black-box objective function $f : \mathcal{X} \to \mathbb{R}$ over an input space $\mathcal{X} \subseteq \mathbb{R}^D$. Please note that f can be deterministic or non-deterministic. Computing $f(x)$ is usually computationally expensive; furthermore we may not have access to the values of f or its gradient. A classical application area of Bayesian optimization is hyperparameter search, where $x \in \mathcal{X}$ are choices of hyperparameters, and $f(x)$ is the resulting performance of a machine learning model.

At each step t, Bayesian optimization forms a probabilistic model $M_t : \mathcal{X} \to \mathcal{P}(\mathbb{R})$ of f; the output of $M_t(x)$ is a probability distribution over the value of $f(x)$. We use uncertainty estimates from this probabilistic model to pick x_t and we update M_t.

Algorithm 2: Bayesian Optimization

Initialize model M_0 with data $\mathcal{D}_0 = \{x_t, y_t\}_{t=1}^{N}$;
for $t = 1, 2, ..., T$ **do**
  $x_t = \arg\max_{x \in \mathcal{X}} \text{Acquisition}(x, M_{t-1})$;
  $y_t = f(x_t)$;
  $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$;
  Construct model M_t on data $\mathcal{D}$;
end

Above, Acquisition(x, M) is an acquisition function; common examples include expected improvement, probability of improvement, and upper counfidence bounds (UCB) (Frazier, 2018).

3 UNCERTAINTY IN BAYESIAN OPTIMIZATION

3.1 Which Uncertainties Are Needed in Online Decision-Making?

Online sequential decision-making tasks such as Bayesian optimization benefit from an accurate probabilistic model to determine which actions to choose. However, because data is limited and because of the need to make modeling assumptions, most predictive models are not optimal. This raises the question: which aspects of a predictive model are important for a downstream decision-making task?

This paper argues that the calibration-sharpness tradeoff has important implications on downstream performance. In particular, we argue that among models that attain a given proper loss, it is better to achieve good levels of calibration.

Why is Calibration Useful? A key challenge faced by decision-making algorithms is balancing exploration—e.g., learning the shape of the unknown function f in Bayesian optimization—against exploitation—e.g., selecting points x at which f takes small values. Exploration-exploitation decisions are often made using a probabilistic model. In regions that are unexplored, the confidence interval around the value of $f(x)$ should be large to promote exploration. Calibration helps mitigate over-confidence and promotes accurate confidence intervals that encourage exploration.

Another benefit of calibrated models is the accurate computation of expected values of future outcomes. Since an expectation is a sum weighted by probabilities of future events, aligning predicted and empirical probabilities is crucial. Accurate estimates of expected utility yield improved planning performance in model-based algorithms (Malik et al., 2019).

3.2 Formal Analysis

Notation. Consider a setting where we sequentially minimize a loss function $\ell : \mathcal{Y} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}_+$ over a set of outcomes $\mathcal{Y}$, actions $\mathcal{A}$, and features $\mathcal{X}$. The loss $\ell(y, a, x)$ quantifies the error of an action $a \in \mathcal{A}$ in a state $x \in \mathcal{X}$ given outcome $y \in \mathcal{Y}$.

Bayesian decision-making theory provides a principled approach for selecting actions in the above scenario. We rely on a predictive model M of y and select decisions that minimize the expected loss:

$$a(x) = \arg\min_a \mathbb{E}_{y\sim M(x)}[\ell(y, a, x)] \tag{2}$$

$$\ell(x) = \min_a \mathbb{E}_{y\sim M(x)}[\ell(y, a, x)]. \tag{3}$$

Here, $a(x)$ is the action that minimizes the expected loss under M. If M were a perfect predictive model, the above decision-making strategy would be optimal. In practice, inaccurate models can yield imperfect decisions. We argue below that in some cases, calibration is a weaker condition that helps accurately estimate the value of a loss function.

Specifically, we provide a concentration inequality on estimates of the loss that generalizes results by Zhao et al. (2020) in the i.i.d. setting. Our result holds for losses $\ell(y, a, x)$ that are monotonically non-increasing or non-decreasing in y. Note that common acquisition functions used in Bayesian optimization yield ℓ that satisfy this condition.

Theorem 1. *Let M be a quantile calibrated model as in (1) and let $\ell(y, a, x)$ be a monotonic loss. Then for any sequence $(x_t, y_t)_{t=1}^T$ and $r > 1$, we have:*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\left[\ell(y_t, a(x_t), x_t) \geq r\ell(x_t))\right] \leq 1/r \tag{4}$$

Note that this statement represents an extension of Markov's inequality. See Appendix F for a proof.

4 ALGORITHMS FOR ONLINE CALIBRATION

Enforcing calibration in an online decision-making setting is challenging because the data distribution is non-stationary (it is influenced by the agent's decisions). Note that the objective itself is stationary and fixed; however, the distribution of the data at each timestep depends on the output of the algorithm at previous timesteps, hence is non-IID (see Appendix D.5). We address the issue of non-stationary data distribution by introducing new algorithms based on online learning (Shalev-Shwartz et al., 2012). Online learning provides methods that provably produce accurate predictors on any stream of datapoints, including data chosen by an adversary.

4.1 Online Recalibration

Setup At each time step $t = 1, 2, ..., T$ we observe features $x_t \in \mathcal{X}$. A predictive model produces a forecast $Q_t : [0, 1] \to \mathbb{R}$ based on x_t that takes the form of a quantile function. We assume that Q_t is invertible and use the convention that $Q_t(p) = \infty$ for $p > 1$ and $Q_t(p) = -\infty$ for $p < 0$. As an example, a Gaussian process model (Rasmussen and Williams, 2005) used for Bayesian optimization outputs forecasts Q_t given by the quantile function of a Gaussian distribution.

Initially, the forecasts Q_t may be miscalibrated; we seek to compose Q_t with a recalibrator $R_t : [0, 1] \to [0, 1]$ such that $Q_t \circ R_t$ is calibrated as in (1). After we choose $Q_t \circ R_t$, nature reveals a label $y_t \in \mathbb{R}$; we use $o_t(y_t, p) = \mathbb{I}\{y_t \leq Q_t(p)\}$ to denote the indicator of the outcome that y_t falls below the p-th quantile. Our goal can be equivalently defined as choosing R_t such that,

$$\frac{1}{T} \sum_{t=1}^{T} o_t(y_t, R_t(p)) \to p \text{ as } T \to \infty \ \forall p \in [0, 1]. \tag{5}$$

Algorithms Our strategy will be to construct R_t by *optimizing for calibration on historical data.*

Specifically, we choose R_t such that

$$R_t(p) \in \arg\min_q \left[\psi(q) + \sum_{s=1}^{t-1} \ell_{sp}(y_s, q) \right], \quad (6)$$

where $\ell_{sp} : \mathbb{R} \times [0,1] \to \mathbb{R}_+$ is a loss function (possibly varying in time s) that quantifies miscalibration and $\psi(y) : \mathbb{R} \to \mathbb{R}_+$ is a regularizer. Note that the loss function ℓ_{sp} is internal to the recalibrator. We refer to the dataset used to define R_t (i.e., the dataset over which we compute the arg min) as the **calibration set**; it consists of forecasts Q_s and outcomes y_s, and is denoted as

$$\mathcal{C}_t = \{(Q_s, y_s) \mid s = 1, 2, ..., t-1\}. \quad (7)$$

Normally $\mathcal{C}_t$ consists of data from all timesteps $s < t$; we will discuss additional ways of constructing $\mathcal{C}_t$ below.

Equation 6 establishes a close connection to online optimization: it implements a classical online learning algorithm called *follow the regularized leader* (FTLR) (Shalev-Shwartz et al., 2012). Below, we will adopt ℓ_{sp} that are derived from the pinball loss—a generalization of the L1 loss motivated by conditional quantile estimation.

4.2 Recalibration via Online Optimization

Consider first the simpler problem of finding a $q_t \in [0,1]$ such that $Q_t(q_t)$ is an estimate of the p-th conditional quantile, i.e., $\frac{1}{T}\sum_{t=1}^{T} o_t(y_t, q_t) - p \to 0$ as $T \to \infty$.

Quantile Pinball Loss We propose to optimize a modification of the pinball loss which we call the quantile pinball loss (QPL), defined as

$$\ell_{tp}(y_t, q) = (q - Q_t^{-1}(y_t))(o_t(y_t, q) - p). \quad (8)$$

Note that ℓ_{tp} is convex: its graph is V-shaped with the slopes of the two lines defining the V being p and $1-p$; when $p = 0.5$, this essentially yields the L1 loss. The offline minimizer of the pinball loss yields a consistent estimator for the p-th quantile of the data in a batch setting (Koenker and Bassett Jr, 1978); here, we derive the same property in the online setting.

Lemma 1. *The quantile pinball loss serves as a quantile estimator, in that* $\arg\min_q \sum_{s=1}^{t} \ell_{sp}(y_s, q)$ *over a dataset* $(y_s)_{s=1}^{T}$ *yields a p-th quantile of the dataset.*

Please refer to Appendix G for a complete proof and discussion of QPL.

Optimization Our proposed algorithm optimizes the QPL using a classical online learning algorithm called online gradient descent (OGD). This algorithm may be defined as optimizing (6) for a specific choice of loss function (thus it is an instance of the follow-the-regularized leader framework). Specifically, we define q_t at each step as

$$q_t \in \arg\min_q \left[\frac{1}{2\eta} q^2 + \sum_{s=1}^{t-1} \overline{\ell}_{sp}(q) \right], \quad (9)$$

where $\overline{\ell}_{tp}(q)$ is the linearization of the QPL at $q = q_t$, defined as

$$\overline{\ell}_{tp}(q) = (q - q_t)(o_t(y_t, q_t) - p).$$

Here, we have dropped an additive constant from the linearization since this term does not involve q and therefore has no effect on the minimization problem in Equation (9). Computing q_t can be as simple as a convex optimization problem over a scalar in $[0,1]$ (solvable using e.g., binary search).

The linearization $\overline{\ell}_{tp}(q)$ approximates $\ell_{tp}(y_t, q)$ everywhere by the supporting hyperplane given by a subgradient at $q = q_t$. Since $\psi(q) = \frac{1}{2\eta} q^2$, we can also set the derivative of the objective in (9) to zero and show that $q_t = \sum_{s=1}^{t-1} \eta g_s$ for $g_s \in \partial_q \ell_{sp}(q_s)$, where $\partial_q \ell_{sp}(q_s)$ is the subgradient at the points q_s for $s < t$. This derivation shows that our proposed algorithm is equivalent to online subgradient descent (OSD) on q (Hazan et al., 2016), and reveals the ties between our approach and online learning.

Quantile Calibration Online learning algorithms such as OSD and FTRL guarantee vanishing regret on any sequence of (x_t, y_t). Here, we also show that these algorithms yield quantile calibration. Interestingly, our proof does not directly rely on regret minimization.

Theorem 2. *For any sequence* $(Q_t, y_t, q_t)_{t=1}^{T}$*, where* q_t *satisfies* (9) *with* $\eta > 0$ *and* $p \in [0,1]$*, we have*

$$\left| \frac{1}{T} \sum_{t=1}^{T} o_t(y_t, q_t) - p \right| \leq \frac{1+\eta}{\eta T}. \quad (10)$$

Proof (Sketch). We can show that for any T, $-\eta \leq q_T \leq 1 + \eta$. Observe also from (8) that the subgradient $g_t \in \partial \ell_t(q_t)$ can be written as $(o_t(y_t, q_t)) - p)$. Since $q_{T+1} = \sum_{t=1}^{T-1} \eta g_t$, and $q_{T+1} \in [-\eta, 1+\eta]$, we get the desired result by substituting in the previous expression for g_t and dividing both sides by ηT. □

See Appendix G for the full proof. Our result is reminiscent of adaptive conformal inference (Gibbs and Candes, 2021). However, key differences include: (1) our approach draws novel connections to online learning; (2) our results naturally generalize to full quantile recalibration; (3) our results hold as $\eta \to \infty$.

Quantile Function Recalibration We may now define a full recalibrator R_t pointwise as

$$R_t(p) = \inf \arg\min_q \left[\frac{1}{2\eta} q^2 + \sum_{s=1}^{t-1} \bar{\ell}_{sp}(y_t, q) \right], \quad (11)$$

where $\bar{\ell}_{sp}$ is defined as in (9); the inf is used to break ties if the arg min is a set. It is easy to show that $R_t(p)$ is a monotone function in p, hence does not suffer from the crossing quantiles problem. It is easy to compute $R_t(p)$ at any t, p by solving (11), and we can also approximate R_t via a linear interpolation at a number quantiles. The correctness result below follows from Theorem 2 and the properties of the quantile pinball loss; see Appendix G for the proof.

Theorem 3. *For any sequence $(Q_t, y_t)_{t=1}^T$ and $\eta > 0$, each R_t is a monotone function, and for all $p \in [0, 1]$,*

$$\left| \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{y_t \leq Q_t(R_t(p))\} - p \right| \leq \frac{1+\eta}{\eta T}.$$

5 CALIBRATED BAYESIAN OPTIMIZATION

Next, we apply our online calibration algorithms to a sequential decision-making task: Bayesian optimization. Algorithm 3 outlines our proposed procedure. At each step t, we compose the Bayesian optimization model M_t (which we can assume without loss of generality as being represented by a quantile function) with a recalibrator R_t. We define R_t as in (11); we denote the resulting subroutine as $\textsc{Calibrate}(M_t, \mathcal{D}_t)$.

Algorithm 3: Calibrated Bayesian Optimization

Initialize model M_0 with data $\mathcal{D}_0 = \{x_t, y_t\}_{t=1}^N$;
$R_0 = \textsc{Calibrate}(M_0, \mathcal{D}_0)$;
for $t = 1, 2, ..., T$ **do**
  $x_t = \arg\max_{x \in \mathcal{X}} \text{Acquisition}(x, M_{t-1} \circ R_{t-1})$;
  $y_t = f(x_t)$;
  $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$;
  Construct model M_t on data $\mathcal{D}$;
  $R_t = \textsc{Calibrate}(M_t, \mathcal{D}_t)$;
end

Practical Considerations. We also propose an additional heuristic for constructing the calibration set $\mathcal{C}_t$ (Algorithm 4). Our heuristic is motivated by the fact that Bayesian optimization is typically run for a short number of steps T and offers further performance gains (see Section 6.1).

Specifically, Algorithm 4 builds a recalibration dataset $\mathcal{D}_{\text{recal}}$ via cross-validation on $\mathcal{D}$. At each step of cross-validation, we train M on the training folds and compute forecasts on the test folds, thus generating a calibration dataset (7) using $\mathcal{D}_{\text{test}}$. The union of all the forecasts on the test folds produces the final calibration dataset on which the recalibrator is trained. In our experiments, we used leave-one-out cross-validation within CreateSplits. Formally, given a dataset $\mathcal{D} = \{x_t, y_t\}_{t=1}^N$, $\textsc{CreateSplits}(\mathcal{D})$ produces N splits of the form $\{(\mathcal{D} \setminus \{x_i, y_i\}, \{x_i, y_i\}) \mid i = 1, 2, ..., N\}$.

Algorithm 4: Calibrate

Input: Model M, Dataset $\mathcal{D} = \{x_t, y_t\}_{t=0}^N$;
Initialize recalibration dataset $\mathcal{C} = \emptyset$;
$S = \textsc{CreateSplits}(\text{D})$;
For each $(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$ in S:
1. Train base model M on dataset $\mathcal{D}_{\text{train}}$;
2. Create calibration set $\mathcal{C}'$ from $\mathcal{D}_{\text{test}}$;
3. $\mathcal{C} = \mathcal{C} \cup \mathcal{C}'$;

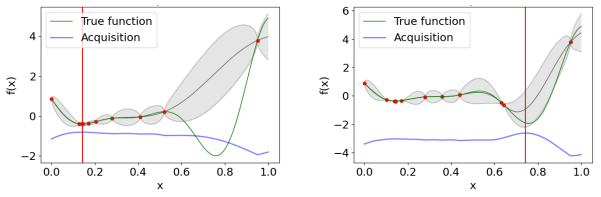
Return recalibrator R obtained via (11) on $\mathcal{C}$;

Understanding Acquisition Functions Next, we discuss how calibration is useful in combination with various acquisition functions. The **probability of improvement** can be written as $1 - F(f(x_{\text{new}}) + \epsilon)$, where F is the CDF predicted by the model. In a calibrated model, this predicted probability in the long run matches the empirical probability of observing an improvement. The **expected improvement** can be defined as $\mathbb{E}[\max(f(x_{\text{old}}) - f(x_{\text{new}}), 0)]$. Theorem 1 suggests that this expectation is easier to evaluate under a calibrated model. **Upper confidence bounds** are $\mu(x) + \alpha \cdot \sigma(x)$ for Gaussians and $Q_x(\alpha)$ for general Q, i.e., the α-th quantile. Recalibration ensures that $Q_x(\alpha)$ is truly the α-th quantile (it is above the true y a fraction α of the time). Appendix C discusses acquisition functions further.

6 EXPERIMENTS

We perform experiments on several benchmark objective functions that are standard in the Bayesian optimization literature, as well as on a number of hyperparameter optimization tasks.

Setup. We use the Gaussian Process (GP) as our base model in the following experiments. However, our method can be applied to any probabilistic model underlying Bayesian optimization in general (Snoek et al., 2015), (Springenberg et al., 2016). See Appendix A for additional implementation details.
Analysis of Calibration. We assess the calibration of the original probabilistic model underlying Bayesian optimization using calibration scores as defined by



(a) Uncalibrated (b) Calibrated

Figure 1: Comparison of Uncalibrated and Calibrated Bayesian Optimization on the Forrester Function (Green) Using the UCB Acquisition Function (Blue).

Kuleshov et al. (2018). Thus, $\text{cal}(F_1, y_1, .., F_n, y_n) = \sum_{j=1}^{m}(p_j - \hat{p_j})^2$, where $0 \leq p_1 < p_2 < .. < p_m \leq 1$ are m confidence levels we use to compute the calibration score. $\hat{p_j}$ is estimated as $\hat{p_j} = |\{y_t | [F_t \leq p_j, t = 1, .., N\}|/N$. The calibration scores are computed on a test dataset $\{F_t, y_t\}_{t=0}^{T}$. This test dataset is constructed by setting $F_t = F_{x_{\text{next}}}(y_{\text{next}})$ and $y_t = y_{\text{next}}$ at every step t before updating model $\mathcal{M}$ in Algorithm 3. In our experiments, we average the calibration score at every step of Bayesian optimization over 5 repetitions of the experiment for each benchmark function.

6.1 Benchmark Optimization Tasks

We visualize runs of calibrated and plain Bayesian optimization on a simple 1D task — the Forrester function in Figure 1. We use the Upper Confidence Bound (UCB) as our acquisition function. Both functions start at the same three points, which miss the global optimum in $[0.6, 0.9]$. The base GP is overconfident, gets stuck at a local minimum near 0.2, and never explores the optimal region. However, the calibrated method learns that its confidence intervals are too narrow and expands them. This leads it to quickly identify the global optimum in $[0.6, 0.9]$. Please refer to Appendix E for additional plots.

In Figure 2(a), we compare calibrated and uncalibrated Bayesian optimization on the Forrester function. In Figure 2(b) and Figure 2(c), we compare the performance of calibrated method against uncalibrated method under EI acquisition function on the 2D Ackley function and 10D Alpine function (Surjanovic and Bingham). In Appendix A.4, we also run the 10D Alpine function for a greater number of steps (100) and continue to see that the calibrated method produces a lower minima. In Figure 3, we compare the performance of our method on the Sixhump Camel function while varying the acquisition function. In all these examples, the calibrated method finds the global minimum before the uncalibrated method on average.

Comparison Against Additional Baselines. In Figure 4, we see that our calibrated method compares favorably against the modern conformal Bayesian optimization (Stanton et al., 2023) and Adaptive UCB for Bayesian optimization with unknown kernel hyperparameters (Berkenkamp et al., 2019). Please refer to Appendix A.2 for additional results.

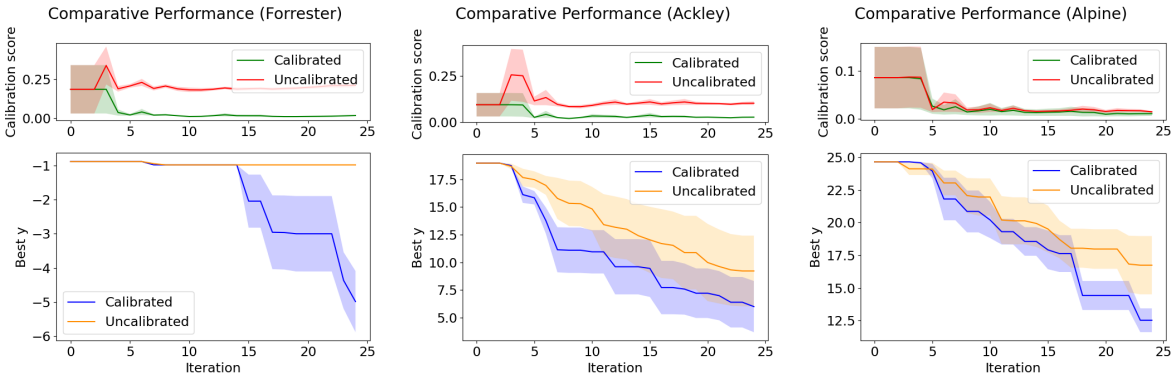
In Table 1, we compare our method against baselines including input and output warping (Snoek et al., 2014; Snelson et al., 2004), Box-Cox and log-transforms on the output (Rios and Tobar, 2018) and an ensemble of Gaussian processes (Lakshminarayanan et al., 2016). Metrics used to perform this evaluation include the minimum value m of the objective function achieved by Bayesian optimization, fraction f of experimental repetitions where the baseline performs worse than our method and normalized area under the Bayesian optimization curve a. We show the error bars in braces for the minimum value m. Although f is not amenable to the same error bar computation, we can estimate the variance using analytical formula for Bernoulli(p) as $p(1-p)$. Also, the error bars on reported AUC values are < 0.02. Please refer to Appendix A.1 for detailed definition of these metrics.

In Table 1, we see that warping methods occasionally improve over the uncalibrated method, often worsen it, and recalibration is almost always more effective than warping. Both uncertainty estimation methods (ensembling and calibration) improve performance relative to the uncalibrated method, with calibration yielding the bigger improvement.

Additional Optimization Tasks. In Table 2, we evaluate our method on seven benchmark optimization tasks (Surjanovic and Bingham) from the Bayesian optimization literature. We fixed the choice of hyperparameters in the calibration algorithm (3 randomly chosen initialization points, Matern kernel, PI acquisition, time-series splits and 25 BO steps). Similar to Table 1, the error bars on reported AUC values are < 0.02 and f is not amenable to error bar calculation.

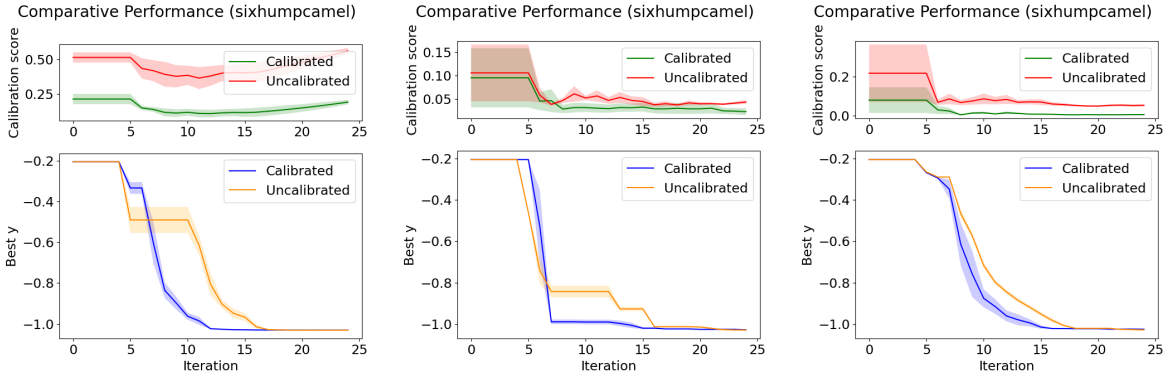
On some functions (e.g., Dropwave, Beal), the calibrated and uncalibrated methods perform similarly. The Cross-in-Tray function has multiple sharp edges, possibly making it harder for the GP to model and worsening the performance of our method. On the McCormick function, the uncalibrated method makes more progress in the initial stages, but the calibrated method finds the minimum earlier.

Sensitivity Analysis. We perform a sensitivity analysis of our method over six of its hyper-parameters (e.g., kernel types, acquisition functions) for the Forrester function. See Table 4 in Appendix A.3 for the full experimental results. Our results indicate good



(a) Forrester (b) Ackley (2D) (c) Alpine (10D)

Figure 2: Comparison of Bayesian Optimization in Benchmark Functions. In the top plots, we see that calibrated method reduces calibration error. The bottom plots show that on an average, the calibrated method identifies the minimum using less iterations.



(a) UCB (b) EI (c) PI

Figure 3: Comparison of Calibrated and Uncalibrated Bayesian Optimization on Six-hump-camel Function (2D) for Various Acquisition Functions. The top plots show that the calibrated method reduces calibration error. In the bottom plots, we see that the calibrated method identifies the minimum using fewer iterations.

robustness across hyper-parameters; we also show slight improvements from our cross-validation heuristic.

6.2 Hyperparameter Optimization Tasks

Online LDA In the Online LDA algorithm (Hoffman et al., 2010), we have three hyperparameters: τ_0 and κ which control the learning rate and minibatch size s (Appendix A.5). The objective function runs the Online LDA algorithm with these hyperparameters to convergence on the training set and outputs test set perplexity. We run this experiment on the 20 Newsgroups dataset. In Figure 5(a), we see that the calibrated method achieves a configuration of hyperparameters giving lower average perplexity. The error bars around the averaged runs are intersecting significantly due to variation across experiment repetitions. Hence, we add a separate plot showing the average improvement over time, defined as the difference between the best minimum found by the uncalibrated method and the best minimum found by the calibrated method. We observe that it is positive most of the time.

Image Classification We define a Convolutional Neural Network (CNN) for image classification with 6 tunable hyperparameters: batch size, learning rate, filter size, etc. (see Appendix A.6). Our objective function trains the CNN and returns the classification error on test dataset. We run our experiments on CIFAR10 (Krizhevsky, 2009) and SVHN datasets (Netzer et al., 2011). On the CIFAR10 dataset (Figure 5(b)), we see that the calibrated method achieves a lower classification error on an average at the end of

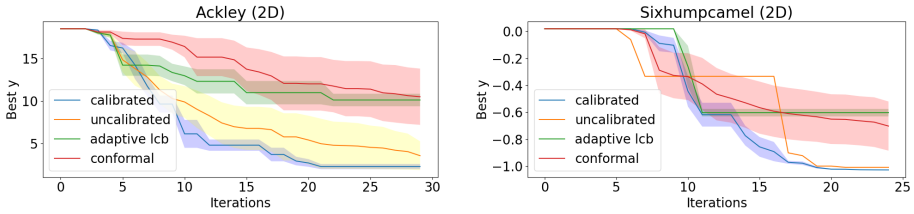
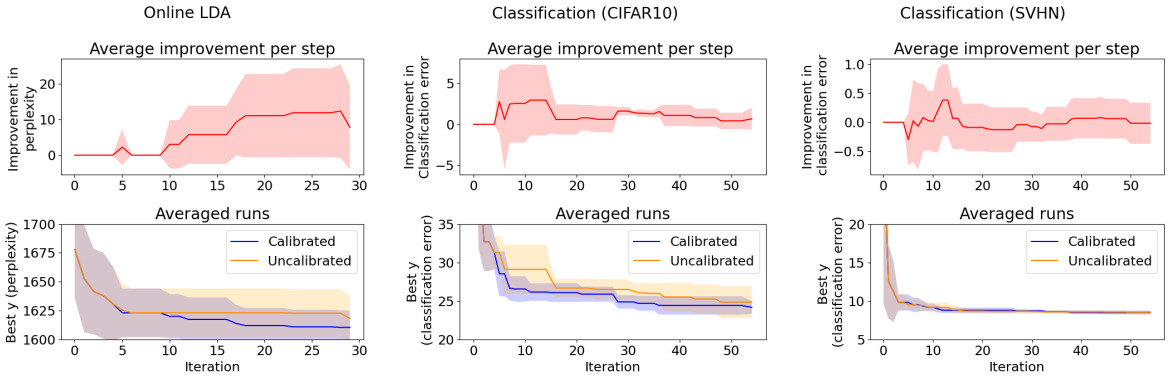


Figure 4: Comparison of Calibrated Bayesian Optimization with Additional Baselines. The calibrated method outperforms the modern Bayesian optimization baseline (Stanton et al., 2023) and Adaptive UCB for Bayesian optimization with unknown kernel hyperparameters (Berkenkamp et al., 2019). The acquisition function used here is UCB.

Table 1: Comparison of Calibrated Bayesian Optimization Against Baselines.

Objective Function	Method	Minimum Found (↓)	Fraction of Runs Where Calibrated Beats Baseline (↑)	Area Under the Curve (↓)
Forrester (1D)	Uncalibrated method	-0.986 (0.001)	0.8	0.9866
	Input warping	-0.889 (0.000)	**1**	1.0000
	Output warping	**-6.021 (0.000)**	0.0	**0.2409**
	Boxcox transformation of output	-4.806 (0.780)	0.4	0.4911
	Log transformation of output	-0.986 (0.000)	**1**	0.9865
	Bagged ensemble of 5 GPs	-3.891(1.098)	0.8	0.8382
	Calibrated Method (Ours)	-4.983 (0.894)	**1**	0.8187
Ackley (2D)	Uncalibrated method	12.359 (2.439)	0.8	0.7815
	Input warping	14.061 (1.312)	0.8	0.8125
	Output warping	10.459 (3.365)	0.6	0.7257
	Boxcox transformation of output	14.421 (1.423)	0.8	0.9015
	Log transformation of output	8.330 (0.849)	0.8	0.6524
	Bagged ensemble of 5 GPs	6.011 (2.544)	0.6	0.6013
	Calibrated Method (Ours)	**5.998 (2.314)**	**1**	**0.5516**
Alpine (10D)	Uncalibrated method	15.506 (1.275)	0.6	0.6527
	Input warping	15.697 (1.740)	0.8	0.6492
	Output warping	13.531 (2.127)	0.6	**0.5857**
	Boxcox transformation of output	15.715 (0.603)	0.8	0.6253
	Log transformation of output	20.996 (1.661)	0.8	0.7931
	Bagged ensemble of 5 GPs	16.677 (1.699)	0.8	0.7334
	Calibrated Method (Ours)	**12.537 (0.909)**	**1**	0.6423



(a) Online LDA (b) CIFAR10 (c) SVHN

Figure 5: Hyperparameter Optimization Experiments. **Top:** The average improvement made by the calibrated method over the uncalibrated method. **Bottom:** The best minimum found by each method per iteration.

50 iterations of Bayesian optimization. The calibrated method also achieves this minimum using about 50%

Table 2: Evaluating Calibrated Bayesian Optimization on Additional Tasks. Calibration strictly improves performance on four benchmarks and performance is similar on two tasks. Lower performance on Cross-in-Tray could be attributed to the presence of multiple sharp edges and corners that are hard to model via a GP.

Optimization benchmark	% Runs Where Calibrated is Best (↑)	Area Under the Curve, Calibrated (↓)	Area Under the Curve, Uncalibrated (↓)
Cosines	0.8	0.2973	0.3395
Beale	0.6	0.0929	0.0930
Mccormick	0.8	0.1335	0.1297
Powers	0.8	0.2083	0.2325
Cross-in-Tray	0.2	0.2494	0.2217
Ackley	0.8	0.3617	0.4314
Dropwave	0.6	0.0455	0.0452

less number of steps.

7 DISCUSSION & RELATED WORK

Bayesian optimization is commonly used for optimizing black-box objective functions in applications like robotics (Calandra et al., 2016), reinforcement learning (Brochu et al., 2010), hyperparameter optimization (Bergstra et al., 2011), recommender systems (Vanchinathan et al., 2014), automatic machine learning, (Thornton et al., 2013) and materials design (Frazier and Wang, 2015). The choices like acquisition function (Snoek et al., 2012), kernel function (Duvenaud et al., 2013) and transformation of input spaces (Snoek et al., 2014) are important in making sure that the optimization works efficiently. Calibration in online sequential decision making and iterative design tasks has also been explored for prediction sets from a conformal prediction perspective (Fannjiang et al., 2022; Stanton et al., 2023). While existing works approach a similar problem, they address distribution mismatch via techniques based on importance sampling, while our paper develops techniques based on online learning.

Calibrated Uncertainties Platt scaling (Platt, 1999) and isotonic regression (Niculescu-Mizil and Caruana, 2005) are popular ways for calibrating uncertainties. This concept can be extended to regression calibration (Kuleshov et al., 2018), distribution calibration (Song et al., 2019), online learning (Kuleshov and Ermon, 2017b), and structured prediction (Kuleshov and Liang, 2015). Accurate uncertainty representation has been studied for applications like model-based reinforcement learning (Malik et al., 2019), semi-supervised learning (Kuleshov and Ermon, 2017a), neural networks (Guo et al., 2017) and natural language processing models (Nguyen and O'Connor, 2015).

Limitations There are still some tasks such as SVHN image classification where calibration does not improve performance. Studying the properties of objective functions where calibrated Bayesian optimization produces lower benefits can be useful in identifying directions for further improvements in the algorithm.

Conclusion The accuracy of uncertainty envelopes is important for balancing exploration and exploitation in Bayesian optimization. We show that we can calibrate the probabilistic model without additional function evaluations. Our approach improves the performance of Bayesian optimization in standard benchmark functions and hyperparameter optimization tasks.

References

The GPyOpt authors. Gpyopt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt, 2016.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. No-regret bayesian optimization with unknown hyperparameters, 2019.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010.

R. Calandra, A. Seyfarth, and J Peters. Bayesian optimization for learning gaits under uncertainty. *Ann Math Artif Intell*, page 5–23, 2016. URL https://doi.org/10.1007/s10472-015-9463-9.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search, 2013.

Clara Fannjiang, Stephen Bates, Anastasios N. Angelopoulos, Jennifer Listgarten, and Michael I. Jordan. Conformal prediction for the design problem. *CoRR*, abs/2202.03613, 2022. URL https://arxiv.org/abs/2202.03613.

Peter I. Frazier. A tutorial on bayesian optimization, 2018.

Peter I. Frazier and Jialei Wang. Bayesian optimization for materials design. *Springer Series in Materials Science*, page 45–75, Dec 2015. ISSN 2196-2812. doi: 10.1007/978-3-319-23871-5_3. URL http://dx.doi.org/10.1007/978-3-319-23871-5_3.

Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Matthew Hoffman, Francis Bach, and David Blei. Online learning for latent dirichlet allocation. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf.

Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Volodymyr Kuleshov and Stefano Ermon. Deep hybrid models: bridging discriminative and generative approaches. In *Uncertainty in Artificial Intelligence*, 2017a.

Volodymyr Kuleshov and Stefano Ermon. Estimating uncertainty online against an adversary. In *AAAI*, pages 2110–2116, 2017b.

Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/52d2752b150f9c35ccb6869cbf074e48-Paper.pdf.

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

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2016. URL https://arxiv.org/abs/1612.01474.

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

Allan H Murphy and Robert L Winkler. A general framework for forecast verification. *Monthly weather review*, 115(7):1330–1338, 1987.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on*

Deep Learning and Unsupervised Feature Learning 2011, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Khanh Nguyen and Brendan O'Connor. Posterior calibration and exploratory analysis for natural language processing models, 2015.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102430. URL https://doi.org/10.1145/1102351.1102430.

Christopher Paciorek and Mark Schervish. Nonstationary covariance functions for gaussian process regression. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003. URL https://proceedings.neurips.cc/paper_files/paper/2003/file/326a8c055c0d04f5b06544665d8bb3ea-Paper.pdf.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Gonzalo Rios and Felipe Tobar. Learning non-gaussian time series using the box-cox gaussian process. pages 1–8, 07 2018. doi: 10.1109/IJCNN.2018.8489648.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. 2007. doi: 10.48550/ARXIV.0706.3188. URL https://arxiv.org/abs/0706.3188.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Edward Snelson, Zoubin Ghahramani, and Carl Rasmussen. Warped gaussian processes. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004. URL https://proceedings.neurips.cc/paper/2003/file/6b5754d737784b51ec5075c0dc437bf0-Paper.pdf.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

Jasper Snoek, Kevin Swersky, Richard S. Zemel, and Ryan P. Adams. Input warping for bayesian optimization of non-stationary functions, 2014.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/snoek15.html.

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

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/a96d3afec184766bfeca7a9f989fc7e7-Paper.pdf.

Samuel Stanton, Wesley Maddox, and Andrew Gordon Wilson. Bayesian optimization with conformal prediction sets. In *International Conference on Artificial Intelligence and Statistics*, pages 959–986. PMLR, 2023.

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved October 8, 2022, from http://www.sfu.ca/~ssurjano.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, 2013.

Hastagiri P. Vanchinathan, I. Nikolic, F. D. Bona, and Andreas Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *RecSys '14*, 2014.

Vladimir Vovk, Ivan Petej, Paolo Toccaceli, Alexander Gammerman, Ernst Ahlberg, and Lars Carlsson. Conformal calibrators. In Alexander Gammerman, Vladimir Vovk, Zhiyuan Luo, Evgueni N. Smirnov, Giovanni Cherubin, and Marco

Christini, editors, *Conformal and Probabilistic Prediction and Applications, COPA 2020, 9-11 September 2020, Virtual Event, Verona, Italy*, volume 128 of *Proceedings of Machine Learning Research*, pages 84–99. PMLR, 2020. URL http://proceedings.mlr.press/v128/vovk20a.html.

Shengjia Zhao, Tengyu Ma, and Stefano Ermon. Individual calibration with randomized forecasting, 2020. URL https://arxiv.org/abs/2006.10288.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Online Calibrated Uncertainty Estimation for Bayesian Optimization (Supplementary Material)

A ADDITIONAL DETAILS ON EXPERIMENTS

We implement our method on top of the GPyOpt library (authors, 2016) (BSD 3 Clause License) in Python. The output values of objective function are normalized before training the base GP model. The GP uses Radial Basis Function (RBF) kernel. For the experiments on hyperparameter optimization tasks, we used Expected Improvement as acquisition function. We use 5 randomly chosen data-points to initialize the base GP. The experiments are repeated 5 times and the results are averaged over these 5 runs.

Table 3: Comparison of Calibrated Bayesian Optimization Against Baselines. We see that the calibrated method compares favorably against Input warping ((Snoek et al., 2014)), Output warping (tanh as in (Snelson et al., 2004)), Boxcox and Log transformation of output ((Rios and Tobar, 2018)), and Bagged ensemble of 5 GPs ((Lakshminarayanan et al., 2016)).

Objective Function	Method	Minimum Found (↓)	Fraction of Runs Where Calibrated Beats Baseline (↑)	Area Under the Curve (↓)
Forrester (1D)	Calibrated Method	-4.983 (0.894)	1	0.8187
	Uncalibrated method	-0.986 (0.001)	0.8	0.9866
	Input warping	-0.889 (0.000)	1	1.0000
	Output warping	-6.021 (0.000)	0.0	0.2409
	Boxcox transformation of output	-4.806 (0.780)	0.4	0.4911
	Log transformation of output	-0.986 (0.000)	1	0.9865
	Bagged ensemble of 5 GPs	-3.891(1.098)	0.8	0.8382
Ackley (2D)	Calibrated Method	5.998 (2.314)	1	0.5516
	Uncalibrated method	12.359 (2.439)	0.8	0.7815
	Input warping	14.061 (1.312)	0.8	0.8125
	Output warping	10.459 (3.365)	0.6	0.7257
	Boxcox transformation of output	14.421 (1.423)	0.8	0.9015
	Log transformation of output	8.330 (0.849)	0.8	0.6524
	Bagged ensemble of 5 GPs	6.011 (2.544)	0.6	0.6013
Cosines (2D)	Calibrated Method	-1.5983 (0.0006)	1	0.2969
	Uncalibrated method	-1.5974 (0.00102)	0.8	0.3441
	Input warping	-1.3054 (0.0347)	1.0	0.9534
	Output warping	-1.5994 (0.0001)	0.8	0.4730
	Boxcox transformation of output	-1.5306 (0.0590)	0.8	0.5969
	Log transformation of output	-1.5992 (0.0003)	0.8	0.3642
	Bagged ensemble of 5 GPs	-1.5989 (0.0003)	0.4	0.3179
Alpine (10D)	Calibrated Method	12.537 (0.909)	1	0.6423
	Uncalibrated method	15.506 (1.275)	0.6	0.6527
	Input warping	15.697 (1.740)	0.8	0.6492
	Output warping	13.531 (2.127)	0.6	0.5857
	Boxcox transformation of output	15.715 (0.603)	0.8	0.6253
	Log transformation of output	20.996 (1.661)	0.8	0.7931
	Bagged ensemble of 5 GPs	16.677 (1.699)	0.8	0.7334

A.1 Evaluation Metrics

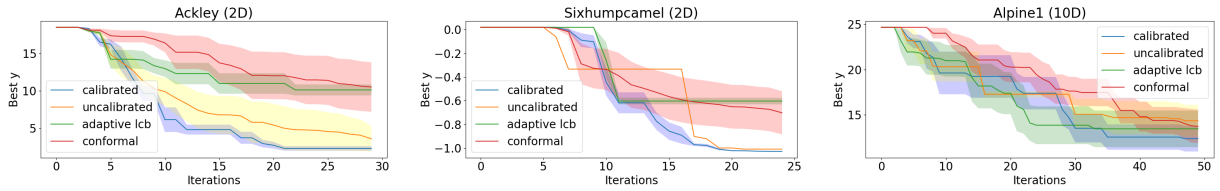
We define the following metrics to compare calibrated Bayesian optimization with uncalibrated method and other baselines.

1. Minimum value m of the objective function achieved by Bayesian optimization. Lower values of m are better.
2. Fraction f of experimental repetitions where the baseline performs worse than our method. A baseline performs worse than our method if it does not find a lower minimum, if it finds the same minimum in a larger number of steps, or if its optimization curve is entirely above that of the calibrated method. Higher values of f are better.
3. Normalized area under the optimization curve a. For each method, we compute the area under its optimization curve (i.e., its optimum as a function of the number of optimization steps; see Figure 2) and normalize it relative to the area of the rectangle formed by upper and lower bounds along the x, y axes (hence max a is one). Smaller values of a indicate that the method reaches the minimum faster, hence are better.

A.2 Comparing Against Additional Baselines

Our method outperformed the modern conformal baseline (Stanton et al., 2023) that applies conformal prediction for calibrated uncertainties in Bayesian optimization. We also outperform the Adaptive UCB (Berkenkamp et al., 2019) method (that adjusts the UCB interval adaptively) on two tasks and are within the margin of error on the third.

We produce additional results with several other baselines in Table 3.

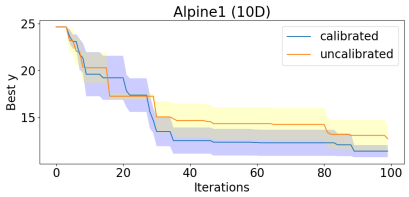


A.3 Sensitivity Analysis

We produce additional results on sensitivity analysis in Table 4.

A.4 Increasing the number of optimization steps

On the most challenging benchmark function (Alpine 10D), **we ran our method for 100 steps and we observed results consistent with 25 steps** . We see an improvement in the minima found at the 100-th step from 12.75 to 11.43. Note that on non-synthetic hyperparameter optimization tasks, we ran for >50 steps and observed that the methods plateaued.



A.5 Online LDA

We use the grid of parameters mentioned in Table 5 as the input domain while running Bayesian optimization. We run this algorithm on the 20 Newsgroups dataset which contains 20,000 news documents partitioned evenly across 20 different newsgroups. We train the algorithm on 11,000 randomly chosen documents. A test-dataset of 2200 articles is used to assess the perplexity.

A.6 Image Classification Using Neural Networks

We provide the range of hyperparameters considered while performing Bayesian optimization to determine their optimal configuration with reference to the image classification experiments in Table 6.

Table 4: Sensitivity Analysis of Calibrated Bayesian Optimization (Algorithm 3 and 4) for the Forrester Function.

Hyper-Parameter	Modification	% Runs Where Calibrated is Best (↑)	Area Under the Curve, Calibrated (↓)	Area Under the Curve, Uncalibrated (↓)
Kernel of Base Model $\mathcal{M}$	Matern	0.8	0.2811	0.3454
	Linear	1.0	0.5221	1.0000
	RBF	0.4	0.2366	0.2922
	Periodic	0.6	0.2586	0.3305
Number of Time-series splits in CreateSplits	N-1	0.8	0.2811	0.3454
	N-2	0.8	0.2768	0.3454
	N-3	0.8	0.2653	0.3454
	N-4	0.6	0.2643	0.3454
Recalibrator Model $\mathcal{R}$	GP	0.8	0.2810	0.3454
	MLP	0.8	0.2785	0.3454
Number of Data Points for Initializing Base Model	3	0.8	0.2810	0.3454
	4	0.8	0.0557	0.0614
	7	0.4	0.0719	0.0736
	10	0.4	0.0825	0.0817
Initialization Design (Base model initialized with 4 data points)	Random	0.8	0.0557	0.0614
	Sobol	0.6	0.0415	0.0414
	Latin	0.2	0.2358	0.2181
Acquisition function (Linear kernel)	LCB	1.0	1.0000	1.0000
	EI	1.0	0.5221	1.0000
	PI	0.8	0.6920	1.0000

Table 5: Hyperparameters for Online LDA

Name of HP	Bounds	Type of domain
Minibatch size	[1, 128]	Discrete (log-scale)
κ	[0.5, 1]	Continuous (step-size=0.1)
τ_0	[1, 32]	Discrete (log-scale)

Table 6: Hyperparameters for CNN (CIFAR10 and SVHN classification)

Name of HP	Bounds	Type of domain
Batch size	[32, 512]	Discrete (step size 32)
Learning rate	[0.0000001, 0.1]	Continuous (log-scale)
Learning rate decay	[0.0000001, 0.001]	Continuous (log-scale)
L2 regularization	[0.0000001, 0.001]	Continuous (log-scale)
Outchannels in fc layer	[256, 512]	Discrete (step size=16)
Outchannels in conv layer	[128, 256]	Discrete (step size=16)

B CALIBRATION OF PROBABILISTIC MODEL

For training a recalibrator over our probabilistic model, we compute the CDF F_t at each data-point y_t using the formulation $F_t = [\mathcal{M}(x_t)](y_t)$. This can be used to estimate the the empirical fraction of data-points below each quantile. Algorithm 5 based on based on Kuleshov et al. (2018) outlines this procedure.

C EXAMINING AQUISITION FUNCTIONS

We analyze the role of calibration in common acquisition functions used in Bayesian optimization.

Algorithm 5: Calibration of Probabilistic Model

Input: Dataset of probabilistic forecasts and outcomes $\{[\mathcal{M}(x_t)](y_t), y_t\}_{t=1}^N$

1. Form recalibration set $\mathcal{D} = \{[F_t, \hat{P}(F_t)\}_{t=1}^N$ where $F_t = [\mathcal{M}(x_t)](y_t)$ and $\hat{P}(p) = |\{y_t | [F_t \leq p, t = 1, .., N\}|/N$.
2. Train recalibrator model $\mathcal{R}$ on dataset $\mathcal{D}$.

Probability of Improvement. The probability of improvement is given by $P(f(x) \geq (f(x^+) + \epsilon)$, where $\epsilon > 0$ and x^+ is the previous best point. Note that this corresponds to $1 - F_x(f(x^+) + \epsilon)$, where F_x is the CDF at x that is predicted by the model. In a quantile-calibrated model, these probabilities on average correspond to the empirical probability of observing an improvement event. This leads to acquisition function values that more accurately reflect the value of exploring specific regions. Furthermore, if the model is calibrated, we keep working with calibrated values throughout the optimization process, as x^+ changes.

Expected Improvement. The expected improvement can be defined as $\mathbb{E}[\max(f(x) - f(x^+), 0)]$. This corresponds to computing the expected value of the random variable $R = \max(Y - c, 0)$, where Y is the random variable that we are trying to model by $\mathcal{M}$, and $c \in \mathbb{R}$ is a constant. If we have a calibrated distribution over Y, it is easy to derive from it a calibrated distribution over R. By Proposition **??**, we can estimate $\mathbb{E}[R]$ under the calibrated model, just as we can estimate the probability of improvement in expectation.

Upper Confidence Bounds. The UCB acquisition function for a Gaussian process is defined as $\mu(x) + \gamma \cdot \sigma(x)$ at point x. For non-Gaussian models, this naturally generalizes to a quantile $F_x^{-1}(\alpha)$ of the predicted distribution F. In this context, recalibration adjusts confidence intervals such that $\alpha \in [0, 1]$ corresponds to an interval that is above the true y a fraction α of the time. This makes it easier to select a hyper-parameter α. Moreover, as α or γ are typically annealed, calibration induces a better and smoother annealing schedule.

D ADDITIONAL DISCUSSION

A key conceptual contribution of our work is a new angle for reasoning about uncertainty in the context of sequential decision-making. There exist many known decompositions of uncertainty, e.g. epistemic vs. aleatoric. Our work argues for using a different decomposition of uncertainty that is rarely used: calibration + sharpness.

This decomposition is interesting because the calibration property can be easily enforced in practice; at the same time this property greatly improves sequential decision-making for reasons we explain in the paper, and enforcing it results in significant practical benefits. This fact is currently underappreciated; our work contributes to a body of literature (see e.g., Malik et al. (2019)) that helps popularize the idea of reasoning about uncertainty through the lens of calibration and sharpness, and can lead to significant practical improvements in uncertainty-aware algorithms that adopt our angle.

From a methodological perspective, our work resolves challenges in applying calibration in the context of Bayesian optimization. We introduce recalibration mechanisms based on leave-one-out cross-validation with a temporal ordering and we design specific classes of Gaussian recalibrators that are compatible with GP outputs and acquisition function inputs. We discovered that more naive applications of calibration fail, and our methods are non-trivial. Finally, we show empirically that our ideas have significant practical benefits.

D.1 On the Computational Cost of Calibrated Bayesian Optimization

Calibration increases the computational cost of Bayesian optimization (since we fit multiple GP models, and not just one). However, in most applications, we expect that the cost of fitting GPs will be negligible compared to the cost of evaluating the objective function at a datapoint. For example, in hyper-parameter optimization, the cost of training a new neural network with a new set of hyper-parameters vastly exceeds the cost of fitting a GP. Hence, calibrated and uncalibrated methods are in practice comparable in terms of their computational costs, and training multiple GPs does not limit the applicability of our method.

In terms of time complexity, the increase in computational costs to run Algorithm 4 after each standard Bayesian

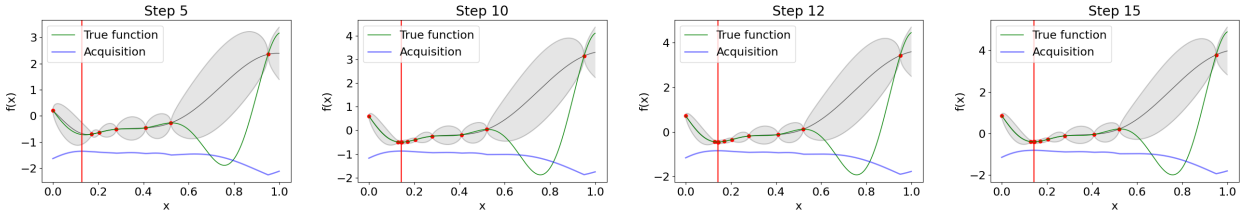
optimization step in Algorithm 3 depends linearly on the number of cross-validation splits ($|S|$) and the time complexity to train the model $\mathcal{M}$. The time complexity also depends on an additive term consisting of dataset size N multiplied with the inference time complexity of model $\mathcal{M}$. This additive term comes from running step 2 in Algorithm 4 cumulatively on all the $\mathcal{D}_{\text{test}}$ sets. An additive term also corresponds to time-complexity to train recalibrator R in the end. The increase in overall space complexity depends linearly on the number of cross-validation splits $|S|$ and the size of dataset N together with an additive term to store the trained recalibrator $\mathcal{R}$. In our experiments, the model $\mathcal{M}$ is itself a Gaussian Process, but other models can be also be used to perform calibrated Bayesian optimization.

The experiments with real world hyperparameter optimization tasks were run on a GPU cluster since training the neural network and Online LDA models with a chosen set of hyperparameters incurs a high computational cost. However, all other experiments that compute blackbox objective function with analytic formulas could be performed on a laptop with 2.8GHz quad-core Intel i7 processor.

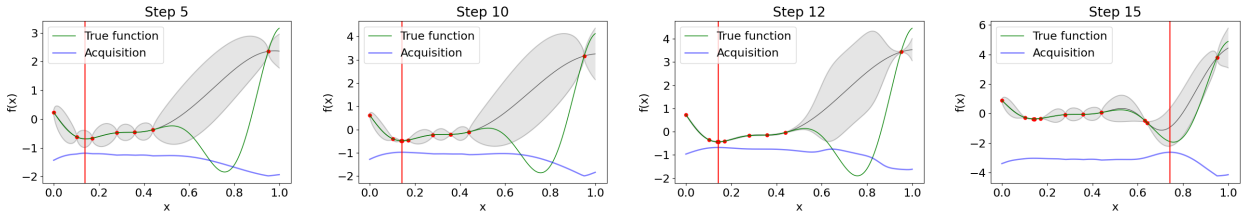
D.2 On Epistemic vs. Aleatoric Uncertainties

Our method calibrates both epistemic and aleatoric uncertainties equally well. The concept of calibration is complementary and orthogonal to the concept of epistemic vs. aleatoric uncertainty. Our method takes any probabilistic prediction P(y) over y (regardless of whether uncertainties P(y) are epistemic or aleatoric) and recalibrates it, resulting in improved performance.

Specifically, let $\mathcal{M}(x)$ be a probabilistic model that outputs a probabilistic forecast $P(y)$ over the target y. The $\mathcal{M}(x)$ may model purely aleatoric uncertainties (e.g., $\mathcal{M}$ is a neural network with a softmax or Gaussian output layer) or epistemic uncertainties (e.g., $\mathcal{M}$ is a GP). In either case, $P(y)$ is just a distribution for which we can assess calibration. Our method improves calibration equally well regardless of the type of $\mathcal{M}(x)$ that generated $P(y)$. Improved calibration in turn increases optimization performance of both Bayesian and non-Bayesian base models $\mathcal{M}(x)$. For more details, please consider the analysis of Bayesian and non-Bayesian methods by Kuleshov et al. (2018) (our work extends their recalibration technique and inherits its properties).



(a) Uncalibrated Bayesian optimization produces overconfident intervals and the global minimum is not explored



(b) Calibrated Bayesian optimization produces wider confidence intervals at Step 12 and finds the global optimum

Figure 6: Selected steps of uncalibrated and calibrated Bayesian optimization on the Forrester function (green) using the UCB acquisition function (blue). The global minimum lies near 0.8; however, after sampling 3 initial points at random, the model is constant in $[0.6, 0.9]$, while the true function has a large dip. Since confidence intervals in $[0.6, 0.9]$ are fairly narrow in the uncalibrated method, and the optimization algorithm never explores it, the global minimum is missed by a large margin. In the calibrated method however, the recalibrator learns after iteration 12 that the model is overconfident, expanding its confidence intervals. This leads the calibrated model to explore in $[0.6, 0.9]$ and find the global minimum.

D.3 On Sensitivity With Respect to the Gaussian Assumption

The effects of the Gaussian assumption can be seen in Table 4. These additional tasks cover benchmark functions which are closer (e.g. cosines) and farther (e.g. cross-in-tray) from Gaussian assumptions. We do observe lower performance of both calibrated and uncalibrated Bayesian optimization methods on non-Gaussian tasks. This suggests an opportunity to improve Bayesian optimization by leveraging non-Gaussian models as a replacement for the classical GP approach.

D.4 On Sensitivity With Respect to Higher Dimensions

In the paper, we have results for the Alpine function in 10 dimensions and the hyperparameter optimization tasks have 3-6 input dimensions each. Thus, we observe the benefits of calibration in higher dimensions as well. However, we did observe that in higher dimensions the improvement offered by calibration starts to become gradually less pronounced, which may be attributed to the curse of dimensionality (the difficulty of estimating densities in high dimensions).

D.5 Non-stationarity in Bayesian Optimization and Calibration Literature

The notion of non-stationarity of outcome function in Bayesian optimization is different from non-stationarity of data distribution in calibration literature.

A stationary objective function in the context of Bayesian optimization refers to unchanging characteristics smoothness of the function with changing inputs (Paciorek and Schervish, 2003).

For example, when modelled using a Gaussian Process (GP) regression, stationarity of the modelled function refers to the property of translation invariance of covariance between two outputs (Snoek et al., 2014). The properties of a GP regression are determined by the mean function $m : \mathcal{X} \to \mathbb{R}$ and covariance function or kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Given a set of data-points $\mathcal{D} = \{x_t, y_t\}_{t=1}^T$ such that $\mathbf{X} = \{x_t\}_{t=1}^T$ and $\mathbf{y} = \{y_t\}_{t=1}^T$, the mean μ and covariance Σ as modelled by the GP can be expressed as

$$\mu(x, \mathcal{D}) = m(\mathbf{X}) + K(\mathbf{X}, x)^T (K(\mathbf{X}, \mathbf{X})^{-1})(\mathbf{y} - m(\mathbf{X}))$$

$$\Sigma(x, x', \mathcal{D}) = K(x, x') - K(\mathbf{X}, x)^T (K(\mathbf{X}, \mathbf{X})^{-1}) K(\mathbf{X}, x')$$

Here, an example of stationary kernel is the popular $5/2$ Matern kernel

$K_{52}(x, x') = k\left(1 + \sqrt{5r^2} + \frac{5r^2}{3}\right)\exp(-\sqrt{5r^2})$, where $r = \sum_{d=1}^{D}(x_d - x'_d)^2$ for D-dimensional input. We can see that this kernel is invariant to translations in input space. If the smoothness of true output function varies with input (i.e., non-stationary function), then a stationary kernel in GP regression may not be adequate to model the function.

A non-stationary data-distribution in calibration, on the other hand, refers to the determination of next data-point dependent on previous data-points. The data-points chosen sequentially in Bayesian optimization are not independent of each other, thus producing a non-stationary data-distribution.

E ANALYSING CALIBRATION FOR FORRESTER FUNCTION

In Figure 6, we see a visual comparison of optimization performed by calibrated method against the uncalibrated method.

F Monotonic Loss Bound

We have shown that a calibrated model can be used to estimate expectations on average. Here, we complement these results with additional concentration inequalities which show that estimates of the calibrated loss do not exceed the true loss by too much. Note that this statement represents an extension of the Markov inequality.

Theorem 1. *Let M be a quantile calibrated model as in (1) and let $\ell(y, a, x)$ be a monotonic loss. Then for any sequence $(x_t, y_t)_{t=1}^T$ and $r > 1$, we have:*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\left[\ell(y_t, a(x_t), x_t) \geq r\ell(x_t))\right] \leq 1/r \tag{4}$$

Proof. Recall that $M(x)$ is a distribution over $\mathcal{Y}$, with a density p_x, a quantile function Q_x, and a cdf F_x. Note that for any x and $s \in (0, 1)$ and $y' \leq F_x^{-1}(1 - s)$ we have:

$$\begin{aligned}
\ell(x) &= \int \ell(y, a(x), x) p_x(y) dy \\
&\geq \int_{y\geq y'} \ell(y, a(x), x) p_x(y) dy \\
&\geq \ell(y', a(x), x) \int_{y\geq y'} p_x(y) dy \\
&\geq s\ell(y', a(x), x)
\end{aligned}$$

The above logic implies that whenever $\ell(x) \leq s\ell(y, a, x)$, we have $y \geq F_x^{-1}(1 - s)$ or $F_x(y) \geq (1 - s)$. Thus, we have for all t,

$$\mathbb{I}\{\ell(x_t) \leq s\ell(y_t, a_t, x_t)\} \leq \mathbb{I}\{F_{x_t}(y_t) \geq (1 - s)\}.$$

Therefore, we can write

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{\ell(x_t) \leq s\ell(y_t, a_t, x_t)\} \leq \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{F_{x_t}(y_t) \geq (1 - s)\} = s + o(T),$$

where the last equality follows because M is calibrated. Therefore, the claim holds in the limit as $T \to \infty$ for $r = 1/s$. The argument is similar if ℓ is monotonically non-increasing. In that case, we can show that whenever $y' > F_x^{-1}(s)$, we have $\ell(x) \geq s\ell(x, y', a(x))$. Thus, whenever $\ell(x) \leq s\ell(y, a, x)$, we have $y \leq F_x^{-1}(s)$ or $F_x(y) \leq s$. Because, F_x is calibrated, we again have that

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\{\ell(x_t) \leq s\ell(y_t, a_t, x_t)\} \leq \sum_{t=1}^{T} \mathbb{I}\{F_{x_t}(y_t) < s\} = s + o(T),$$

and the claim holds with $r = 1/s$. □

Note that this implies the same result for a distribution calibrated model, since distribution calibration implies quantile calibration.

G Algorithms for Online Calibration

Here, we introduce algorithms that enforce calibration in an online setting. This task is challenging because the data distribution is the result of a sequential decision-making task. This distribution is therefore non-stationary: it is determined by our actions.

Setup At each time step $t = 1, 2, ...$ we observe a stream of datapoints comprised of features $x_t \in \mathbb{R}^d$. After x_t is revealed, a base uncalibrated model (e.g., a Bayesian optimization model) produces a forecast; we represent this forecast via a quantile function $Q_t : [0, 1] \to \mathbb{R}$ that targets a label $y_t \in \mathbb{R}$. We assume that labels y_t are bounded with $|y_t| < B$, where $B > 0$. We also assume that Q_t is strictly increasing and differentiable. Below, we may sometimes use the notation $Q(p)$ for $p \notin [0, 1]$; in such cases, we use the convention that $Q(p) = -\infty$ for $p < 0$ and $Q(p) = \infty$ for $p > 1$.

The model Q_t may produce miscalibrated outputs; we seek to compose Q_t with a recalibrator $R_t : [0, 1] \to [0, 1]$ such that $Q_t \circ R_t$ is calibrated. After we choose $Q_t \circ R_t$, nature reveals a label $y_t \in \mathbb{R}$. Our goal is to select R_t

such that online quantile calibration (1) holds. Specifically, we use $o_t(y_t, p) = \mathbb{I}\{y_t \leq Q_t(p)\}$ as an indicator of the binary outcome that y_t falls below the p-th quantile. Our goal is to choose R_t such that for all $p > 0$

$$\frac{1}{T}\sum_{t=1}^{T} o_t(y_t, R_t(p)) - p \to 0 \text{ as } T \to \infty. \tag{12}$$

Crucially, we want (12) to hold on any sequence of (Q_t, x_t, y_t).

Optimization Problem Our algorithms construct R_t via an optimization problem. Specifically, we will consider R_t of the form

$$R_t(p) = \arg\min_q \left[\psi(q) - \sum_{s=1}^{t-1} \ell_p(y_s, q)\right],$$

where $\ell_p : \mathbb{R} \times [0,1] \to \mathbb{R}_+$ is a loss function that we will define and $\psi(y) : \mathbb{R} \to \mathbb{R}_+$ is a regularizer (possibly equal to zero everywhere). We choose the loss ℓ_p such that minimizing the average ℓ_p over the previously observed data yields an estimate of the p-th conditional quantile. More specifically, we seek to define ℓ_p such that the probability p is remapped to a probability q for which the event $\{y_t \leq Q_t(q)\}$ is observed a fraction p of the time. Examples of suitable losses ℓ_p include the pinball loss—a generalization of the L1 loss motivated by conditional quantile estimation that we define below—as well as the weighted misclassification loss, which yields an algorithm analogous to that of (Kuleshov et al., 2018).

We establish that R_t obtained via the above construction yields calibrated forecasts through online optimization, a set of techniques that can provably minimize a loss function on distribution-free (possibly adversarial) data. We start by defining our algorithm for one quantile; then we use it to define a complete recalibrator R_t.

G.0.1 Recalibrating One Quantile

First, consider the simpler problem of finding a $q_t \in [0,1]$ such that $Q_t(q_t)$ is an estimate of the p-th conditional quantile, i.e., $\frac{1}{T}\sum_{t=1}^{T} o_t(y_t, q_t) - p \to 0$ as $T \to \infty$.

The Pinball Loss Our strategy for computing q_t relies on online optimization. Specifically, we define a loss $\ell(y, y')$ and an update rule for q_t such that $\sum_{t=1}^{T} \ell(y_t, Q_t(q_t))$ is minimized. Our loss will be inspired by the pinball loss. Given a target quantile p, the pinball loss ℓ_p defined as

$$\begin{aligned}\ell_p(y_t, y) &= (y_t - y)\cdot p \cdot \mathbb{I}\{y_t > y\} + (y - y_t)\cdot(1-p)\cdot\mathbb{I}\{y_t \leq y\} \\ &= (y - y_t)(o_t(y_t, y) - p).\end{aligned}$$

Observe that ℓ_p is convex: its graph is V-shaped with the slopes of the two lines defining the V being p and $1-p$. When $p = 0.5$, the pinball loss coincides with the L1 loss (up to a multiplicative scaling factor). The pinball loss ℓ_p is interesting because the minimizer of ℓ_p over a set of datapoints y_t yields a consistent estimator for the p-th quantile of this set of datapoints.

Our algorithm optimizes a modification of the pinball loss which we call the quantile pinball loss (QPL) and which is defined as

$$\begin{aligned}\ell_{tp}(y_t, q) &= (Q_t^{-1}(y_t) - q)\cdot p \cdot \mathbb{I}\{y_t > Q_t(q)\} + (q - Q_t^{-1}(y_t))\cdot(1-p)\cdot\mathbb{I}\{y_t \leq Q_t(q)\} \\ &= (q - Q_t^{-1}(y_t))(o_t(y_t, q) - p).\end{aligned}$$

Note that the QPL still has the same V-shape as the original quantile loss, with each part of the V having a slope of p and $(1-p)$. We can also show that the QPL features the same attractive property as the pinball loss in that it serves as a quantile estimator.

Lemma 2. *The quantile pinball loss serves as a quantile estimator, in that* $\arg\min_q \sum_{s=1}^{t} \ell_{sp}(y_s, q)$ *over a dataset* $(y_s)_{s=1}^{T}$ *yields a p-th quantile of the dataset.*

Proof. Note that the QPL is convex, as it is the weighted sum of two convex functions, $(q - Q_t^{-1}(y_t))_+$ and $(q)_+$. We minimize the QPL by setting its derivative to zero, giving:

$$\begin{aligned} 0 &= \frac{d}{dq}\sum_{s=1}^{t} \ell_{sp}(y_s, q) \\ &= \frac{d}{dq}\sum_{s=1}^{t} (q - Q_t^{-1}(y_t))(o_t(y_t, q) - p) \\ &= \sum_{s=1}^{t} (o_t(y_t, q) - p) \end{aligned}$$

Thus, the minimum is achieved by a q in the p-th quantile of $(y_s)_{s=1}^T$. □

Regularized Online Gradient Descent We consider the online optimization problem where at each step we choose a prediction q_t for the p-th quantile. Nature then reveals y_t and we incur the quantile pinball loss $\ell_p(y_t, q_t)$. We optimize this problem via regularized online gradient descent (OGD). Recall that OGD is an online optimization method for optimizing a sequence of functions. Note that OGD on the QPL is equivalent to Follow-The-Regularized-Leader (FTLR) on the linearized QPL. Thus, we introduce $\ell_t(q)$ as the linearization of the QPL at $q = q_t$, defined as

$$\ell_t(q) := (q - q_t)\partial_q \ell_{tp}(y_t, q_t) = (q - q_t)(o_t(y_t, q_t) - p)$$

where the constant does not depend on q. The linearization $\ell_t(q)$ approximates $\ell(y_t, q)$ everywhere by the supporting hyperplane given by a subgradient at $q = q_t$. Then, at step t, we choose q_t to minimize

$$q_t = \arg\min_q (\psi(q) + \sum_{s=1}^{t-1} \ell_s(q)), \tag{13}$$

where $\psi(q) : \mathbb{R} \to \mathbb{R}_+$ is a regularizer. Observe that if we choose $\psi(q) = \frac{1}{2\eta}q^2$, this choice yields an exact solution to (13), where $q_t = \sum_{s=1}^{t-1} \eta g_s$ and $g_s \in \partial \ell_s(q_s)$ is a subgradient at the points q_s for $s < t$. This derivation shows that we can compute the set of q_t via gradient descent (although this is not strictly necessary).

Quantile Calibration Online gradient descent normally yields guarantees on the regret of a model. Here, we also show that minimizing the quantile pinball loss induces quantile calibration. This is a condition that is derived from the average gradient of the function, as opposed to the regret. We first establish a technical lemma, then use the lemma to establish quantile calibration. The arguments for Lemma 3 and Theorem 2 are inspired by results shown for adaptive conformal inference (Gibbs and Candes, 2021).

Lemma 3. *For any t, we have that q_t is contained in $[-\eta, 1 + \eta]$.*

Proof. Suppose not, and let t be the first time step for which $q_t < -\eta$ (the case for $q_t > 1 + \eta$ is identical). Note that $|q_t - q_{t-1}| = \eta |o_{t-1}(y_{t-1}, q_{t-1}) - p| \leq \eta$. Thus, we know that $q_t < q_{t-1} < 0$. The first inequality comes from the minimality of t, and the second comes from the fact that $|q_t - q_{t-1}| \leq \eta$. However, $q_{t-1} < 0$ implies that $Q_{t-1}(q_{t-1}) = -\infty$. Thus, $o_t(y_t, q_t) = 1$ and $q_t = q_{t-1} + \eta(o_t(y_t, q_t) - p) > q_{t-1}$. This contradicts that $q_t < q_{t-1}$. □

Theorem 2. *For any sequence $(Q_t, y_t, q_t)_{t=1}^T$, where q_t satisfies (9) with $\eta > 0$ and $p \in [0, 1]$, we have*

$$\left| \frac{1}{T}\sum_{t=1}^{T} o_t(y_t, q_t) - p \right| \leq \frac{1 + \eta}{\eta T}. \tag{10}$$

Proof. Observe that the subgradient $g_t \in \partial \ell_t(q_t)$ at the point q_t can be written as $g_t = o_t(y_t, q_t) - p$. Thus, we can write

$$q_t = \sum_{s=1}^{t-1} \eta g_s = \sum_{s=1}^{t-1} \eta(o_s(y_s, q_s) - p).$$

Dividing both sides by $t\eta$ gives

$$\frac{q_t}{t\eta} = \frac{1}{t}\sum_{s=1}^{t}(o_s(y_s, q_s) - p).$$

Taking the absolute value and applying the lemma that $q_t \in [-\eta, 1+\eta]$ gives the desired result.

□

This proves that the above algorithm yields a valid method for one quantile p.

G.0.2 Quantile Function Recalibration

Consider now a setting where we seek to define a full recalibrator $R(p)$. Our approach is to define the full recalibrator for p.

Lemma 4. *For the linearized quantile pinball losses $\ell_{p_1 t}$ and $\ell_{p_2 t}$ with $p_1 > p_2$, we have* $\arg\min_q \frac{q^2}{2\eta} + \sum_{t=1}^{T} \ell_{p_1 t}(q) > \arg\min_q \frac{q^2}{2\eta} + \sum_{t=1}^{T} \ell_{p_2 t}(q)$.

Proof. Recall that $\ell_{pt}(q) = (q - q_t)(o_t(y_t, q_t) - p)$. Denote the objective for a target quantile p as $J_p(q) = \frac{q^2}{2\eta} + \sum_{t=1}^{T} \ell_{pt}(q)$. Observe that $J_p(q)$ is a quadratic (since ℓ_{pt} is linear), and is uniquely minimized by $q_p^* = -\eta \sum_{t=1}^{T}(o_t(y_t, q_t) - p)$. Therefore, $q_{p_1}^* - q_{p_2}^* = \eta(p_1 - p_2) > 0$.

□

Lemma 4 guarantees that if we fit quantiles individually by optimizing the quantile pinball loss, the resulting quantile estimates will not "cross", in that larger target quantiles admit larger estimates. This is desirable, in that we can produce a valid quantile function representing all the quantile estimates simultaneously. Therefore, we define the recalibrator $R(p)$ for each p via the algorithm in the previous section. Moreover, we can compute an approximate $R(p)$ by computing it at a fixed number of quantiles, and then interpolating. Alternatively, we may compute $R(p)$ at an arbitrary p by solving the optimization problem.

Theorem 3. *For any sequence $(Q_t, y_t)_{t=1}^{T}$ and $\eta > 0$, each R_t is a monotone function, and for all $p \in [0, 1]$,*

$$\left| \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}\{y_t \leq Q_t(R_t(p))\} - p \right| \leq \frac{1+\eta}{\eta T}.$$

Proof. The inequality is a direct application of Theorem 2, where $R_t(p)$ is the value q that minimizes the FTRL objective (6). The fact that R_t is monotonically increasing is a direct result of Lemma 4. □