Thesis

Methods for Calibrated Uncertainty Quantification and Understanding its Utility

Youngseog Chung

CMU-**-**-**-**

July 14, 2025

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

This work was funded in part by DOE grant number DE-******.

Keywords: uncertainty quantification, calibration, probabilistic forecasting, decision making, machine learning

Abstract

As machine learning models have become more capable of dealing with complex data, they have been entrusted with an increasing array of predictive tasks. With such growing reliance on model predictions, being able to assess whether a given model prediction is reliable has become equally important. Uncertainty quantification (UQ) plays a critical role in this context by providing a measure of confidence in a model's predictions. In this thesis, I address the problem of UQ in machine learning in three different stages. The first section presents an overview of evaluation in UQ and an open-source software which provides various utilities in evaluating, visualizing, and recalibrating predictive uncertainty. The second section discusses algorithms designed to produce well-calibrated predictive uncertainties in regression models, which output a distribution over continuous-valued outputs. The first work in this section presents a suite of algorithms for training univariate probabilistic regression models, and the second work discusses an extension to the multivariate setting. The third section presents the utilization of predictive uncertainties in the decision-making setting. The application setting dictates how the uncertainties will be used, and I present a collection of works which utilizes uncertainties in the single-step decision making setting, sequential decision-making setting, and in model-based reinforcement learning.

Acknowledgments

The work comprising this thesis was supported in part by the Kwanjeong Educational Foundation, DOE grant number DE-SC0021414 and DE-AC02-76SF00515, NSF (#1651565), ONR (N000141912145), AFOSR (FA95501910024), ARO (W911NF-21-1-0125), DOE (DE-AC02-76SF00515) and Sloan Fellowship, NSF grant DGE1745016, the AI Research Institutes Program funded by the National Science Foundation under AI Institute for Societal Decision Making (AI-SDM), Award No. 2229881, US Department of Energy grants under contract numbers DE-SC0021414 and DE-AC02-09CH1146 and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1745016 and DGE2140739, the McCune Foundation grant FP00004784, and the Bloomberg Data Science Ph.D. Fellowship .

Contents

List of Figures

List of Tables

1 | Introduction

1.1 Why uncertainty?

As machine learning models have become more capable of dealing with complex data, they have been entrusted with an increasing array of predictive tasks. These tasks range anywhere from predicting the trajectory of vehicles for autonomous driving [Galvão and Huda, 2024], predicting the structure of protein [Jumper et al., 2021], and even modeling the dynamics of plasma during nuclear fusion reactions [Char et al., 2024]. With growing reliance on model predictions for increasingly complex tasks, one important question a user of these models may ask is whether these model can be trusted. Uncertainty quantification (UQ) plays a critical role in this context by providing a measure of confidence in a model's predictions. Given that most machine learning models are designed and trained in a probabilistic manner, we can leverage the predicted probabilities as a means to express uncertainty: the model can output highly diffuse probabilities when it is uncertain, and concentrated probabilities when it is confident. As such, uncertainty quantification and probabilistic forecasting can be considered synonymous [Gneiting et al., 2007]. There are also cases where uncertainties are expressed in ways that do not necessarily imply a strict notion of probabilities: e.g. a subjective scale might state one of `[likely, unlikely, unknown]`, or a model may simply output a scalar score whose absolute value has little meaning and only relative values are meaningful. This thesis assumes that uncertainties are always expressed via probabilities. When expressing uncertainties as probability distributions, the prediction target will usually determine the type of these distributions. For continuous targets (e.g. position delta of a car given actuator input), the uncertainties can be expressed with continuous distributions (e.g. Gaussian distributions). For discrete targets (e.g. classification of an image of a tumor as benign or malignant), categorical distributions are used to express uncertainty over the possible classes.

The discussion around UQ is usually centered around

1. what the source of the uncertainties are
2. what the quantified uncertainties mean
3. how to use the quantified uncertainties

Predictive uncertainties provide a signal which convey a meaning of confidence in the prediction. The lack of confidence is usually attributed to two main sources: aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty stems from inherent randomness or noise in the true data generating process. Given the exact same set of observable variables in a system, the outcome may follow a distribution - e.g. given the exact same weather conditions on two separate days, it may rain heavily on one day, and not rain at all in the other day. In face of such stochasticity in the true labels, a predictor should always be uncertain and try to describe the underlying distribution "as best as possible", where the measure of goodness here will be discussed with the evaluation metrics in the following paragraph. However, even when modeling a fully deterministic system, a predictive model can still be uncertain, especially when the model has simply not seen enough data. Such source of uncertainty is referred to as "epistemic uncertainty", which implies a lack of knowledge. In principle, epistemic knowledge should disappear in the limit of data. The combination of both source are said to comprise "total uncertainty" of a predictive model. While this decomposition of uncertainty is interesting to consider, modeling each source separately is usually not straightforward and the purported utility of decomposing the uncertainties can also be misleading. In this thesis, we do not necessarily distinguish the various sources of uncertainty and assume that we always model the total uncertainty. With this assumption, we proceed to discussions on how to derive meaning from uncertainty estimates and their evaluation metrics.

Assuming a probabilistic output from a classifier, suppose a weather forecasting model predicts rain with probability 90%. How should a user interpret this message of 90%? Under which weather outcomes should the user be satisfied or dissatisfied with this probabilistic prediction? The perception of forecasts is especially a pertinent problem in probabilistic forecasting and uncertainty quantification [Silver, 2019]. Given discrete outcomes (in this case, binary outcomes as rain or no rain), judging a prediction that is probabilistic (rather than deterministic) is not entirely straightforward, and it is susceptible to the criticism that they always are simply a "hedge" against being wrong.

Deriving meaning from probabilistic predictions naturally leads to the discussion around *evaluation*, since the evaluation metric can dictate what kind of meaning one should expect from the predictions. As one might suspect based on the preceding discussion, evaluation in uncertainty quantification is multi-faceted. There is no single metric which serves as an unequivocal signal of goodness of the probabilistic prediction. Among the various evaluation metrics in uncertainty, a set of metrics that are highly interpretable is calibration.

Calibration is measures the alignment between predicted probabilities and empirical frequencies, or the lack of alignment thereof. When an uncertainty-aware machine learning model is calibrated, the user of this model can trust that the predicted event will occur with a frequency consistent with the predicted probability. Thus, only calibrated models allow the interpretation of its outputs as true probabilities, and the user of the model can rely on.

However, calibration itself does not ensure useful probabilistic forecasts. A weather forecasting model which outputs a constant 60% chance of rain everyday, regardless of season and weather events, may still be calibrated if on average, an area receives rain 60% of the days of a year. This forecast is hardly useful since it tells you nothing about a given day. Gneiting et al. [2007] calls this the "climatological forecaster". Additional aspects such as *sharpness* should be considered in evaluations, and proper scoring rules are a family of metrics which consider both calibration and sharpness jointly.

Ultimately, the signal of confidence or uncertainty in the prediction should serve as a guide in the receiver or user of the prediction. There are many downstream tasks where the interpretation, application, and utilization of uncertainty is critical. Specific algorithms within the active learning framework [Settles, 2009] rely on uncertainty estimates to efficiently guide queries from the oracle, and Bayesian optimization assumes a similar problem setup.

In this thesis, I argue that the downstream task should dictate what kind of signals one should desire from uncertainty estimates. Consider the case of weather forecasting. A power trader which primarily deals with renewable energy (the supply and demand of which is heavily dependent on weather) would use these probabilistic forecasts to make trades they believe are justified by the forecast. Meanwhile, an airline will also use these forecasts to make decisions about going through with their scheduled flights or delaying or canceling. At the same time, average Joe that likes playing tennis may decide to risk getting rained on and go through with a scheduled tennis session or decide to cancel based on their decision-making framework which uses the weather forecast. In lieu of the true data generating distribution of the weather event (it is unclear if there even exists a true latent distribution), each of these decision-making scenarios likely require different aspects from the probabilisitic forecasts for optimal decision-making. Zhao et al. [2021b] lays out the connections between notions of calibration and the decision-making settings that each notion of calibration provides guarantees over. Obviously, other decision-making tasks would require other notions of goodness in the uncertainty estimates, which may not even be certain notions of calibration. This thesis provides a glimpse into how uncertainties can inform decision-making processes via a non-exhaustive suite of problem settings. Specifically, we discuss utilizing uncertainties in the single-step decision-making setting where the task is defined by a utility (or cost) matrix, in sequential decision-making where a user repeatedly queries a probabilistic model to minimize regret in hindsight, and in model-based reinforcement learning, where a model of the dynamics is leveraged for policy learning.

1.2 Overview of Thesis

This thesis is comprised of three main parts.

Part 1

The first part of the thesis is focused on evaluation in uncertainty quantification. In this section, we discuss evaluation metrics in uncertainty quantification, and as a means of providing a practical tool to complement the discussion, we present Uncertainty Toolbox. This toolbox is an open-source Python library that my collaborators and I created specifically with the intent of providing a collection of standardized implementations of evaluation metrics for practical and pedagogical purposes. The key message from this discussion is that uncertainty quantification requires a holistic review of various metrics, which may also be prescribed by the downstream task the uncertainty will be used for. This part is based on Chung et al. [2021a].

Part 2

Given that the first section on evaluation will establish calibration as an interpretable and key metric in uncertainty quantification, the second part of this thesis highlights methods to produce calibrated uncertainties from probabilistic machine learning models in the regression setting. Calibration in probabilistic regression is defined in terms of the quantile function, and the conventional method to learn quantiles from data is the check score, or pinball loss. The first work in this section challenges this understanding, specifically in the deep learning setting, and proposes a suite of alternative methods to learn quantiles that are calibrated. The second work in this section extends the discussion to the multivariate regression setting, where even the definition of calibration is unclear due quantiles not being well-defined for multivariate distributions. We propose a method of defining calibration leveraging the notion of highest density regions and further propose a calibration algorithm to optimize for this notion of calibration. This part is based on Chung et al. [2021b] and Chung et al. [2024].

Part 3

In the third and final section, we discuss the utility of calibration in downstream application settings. The field of uncertainty quantification has developed largely detached from the considerations for utilizing the uncertainty estimates for downstream applications. Consider a probabilistic model that has been optimized for calibration. Is it obvious that this model should perform optimally for Bayesian optimization compared to another model that is not calibrated? Should a calibrated model be ideal for model-based reinforcement learning? These connections are not straight-forward. In this thesis, we contend that the downstream application should dictate the desirable characteristics of the uncertainty estimates. To this end, this section presents a suite of works in problem settings which directly utilize uncertainties to guide decision-making. The settings considered include the single-step decision-making, sequential decision-making, and reinforcement learning. This part is based on Chung et al. [2020a], Chung et al. [2023a], and Char et al. [2023c].

Part I

Evaluation in Uncertainty Quantification

2 | Uncertainty Toolbox: an Open-Source Library for Assessing, Visualizing, and Improving Uncertainty Quantification

2.1 Introduction

As machine learning (ML) systems are increasingly deployed on an array of high-stakes tasks, there is a growing need to robustly quantify their predictive uncertainties. Uncertainty quantification (UQ) in machine learning generally refers to the task of quantifying the confidence of a given prediction, and this measure of confidence can be especially crucial in a variety of downstream applications, including Bayesian optimization [Jones et al., 1998, Shahriari et al., 2015], model-based reinforcement learning [Malik et al., 2019, Yu et al., 2020], and in high-stakes prediction settings where errors incur large costs [Wexler, 2017, Rudin, 2019].

UQ is often performed via *distributional predictions* (in contrast with *point predictions*). Hence, given inputs $x \in \mathcal{X}$ and targets $y \in \mathcal{Y}$, one common goal in UQ is to approximate the true conditional distribution of y given x. In the supervised setting where we only have access to a limited data sample, we are then faced with the question, "how can one verify whether a distributional prediction is close to the true distribution using only a finite dataset?" Many works in UQ tend to be disjoint in the evaluation metric utilized, which sends divided signals about which metrics *should* or *should not* be used. For example, some works report likelihood on a test set [Lakshminarayanan et al., 2017, Detlefsen et al., 2019, Zhao et al., 2020], some works use other proper scoring rules [Maciejowska et al., 2016, Askanazi et al., 2018, Bowman et al., 2020, Bracher et al., 2021], while others focus on calibration metrics [Kuleshov et al., 2018, Cui et al., 2020]. Further, with disparate implementations for each metric, it is often the case that reported numerical results are not directly comparable across different works, even if a similar metric is used.

To address this, we present *Uncertainty Toolbox*: an open-source python library that helps to assess, visualize, and improve UQ. There are other libraries such as Uncertainty Baselines [Nado et al., 2021] and Robustness Metrics [Djolonga et al., 2020] that focus on aspects of UQ in the *classification* setting. Uncertainty Toolbox focuses on the *regression* setting and additionally aims to provide user-friendly utilities such as visualizations, a glossary of terms, and an organized collection of key paper references.

We begin our discussion by first introducing the contents of Uncertainty Toolbox. We then provide an overview of evaluation metrics in UQ. Afterwards, we demonstrate the functionalities of the toolbox with a case study where we train probabilistic neural networks (PNNs) [Nix and

Weigend, 1994, Lakshminarayanan et al., 2017] with a set of different loss functions, and evaluate the resulting trained models using metrics and visualizations in the toolbox. This case study shows that certain evaluation metrics shed light on different aspects of UQ performance, and makes the case for using a suite of metrics for a comprehensive evaluation.

2.2 Uncertainty Toolbox

Uncertainty Toolbox comprises four main functionalities, which we detail below.

Evaluation Metrics The toolbox provides implementations for a suite of evaluation metrics. The main categories of metrics are: calibration, group calibration, sharpness, and proper scoring rules. We discuss each of these metric types in the following section (Section 3.2.2).

Recalibration We further implement recalibration methods that leverage isotonic regression [Kuleshov et al., 2018]. Concretely, recalibration aims to improve the average calibration (defined in Eq. (3.3)) of distributional predictions.

Visualizations The toolbox offers a range of easy-to-use visualization utilies to help in inspecting and evaluating UQ quality. These plotting utilities focus on visualizing the predicted distribution, calibration, and prediction accuracy.

Pedagogy For those unfamiliar with the area of predictive UQ, we provide a glossary that communicates the core concepts in this area, and maintain a paper list which organizes some of the key papers in the field.

We hope the toolbox serves as an intuitive guide for those unfamiliar but interested in utilizing UQ, and as a practical tool and point of reference for those active in UQ research.

Uncertainty Toolbox is available at the following page:
`https://github.com/uncertainty-toolbox/uncertainty-toolbox`.

2.3 Evaluation Metrics in Predictive UQ

To summarize the notation and setting: **X, Y** denote random variables; x, y denote realized values; and $\mathcal{X}, \mathcal{Y}$ denote sets of possible values. Further, for any random variable, we denote the true CDF as $\mathbb{F}$, its inverse (i.e. the quantile function) as $\mathbb{Q}$, the corresponding density function as f, and the space of distributions as $\mathcal{F}$. Estimates of these true functions will be denoted with a hat, e.g. $\hat{\mathbb{F}}$ and $\hat{f}$. Lastly, we consider the regression setting where $\mathcal{Y} \subset \mathbb{R}$ and $\mathcal{X} \subset \mathbb{R}^n$.

Many recent works have focused on evaluation metrics involving notions of *calibration* and *sharpness* [Gneiting et al., 2007, Guo et al., 2017, Kuleshov et al., 2018, Song et al., 2019, Tran et al., 2020, Zhao et al., 2020, Fasiolo et al., 2020, Cui et al., 2020]. Calibration in the regression setting is defined in terms of quantiles, and broadly speaking, it requires that the probability of observing the target random variable below a predicted p^{th} quantile is equal to the *expected probability* p, for all $p \in (0, 1)$. We refer to the former quantity as the *observed probability* (also referred to as empirical probability) and denote it $p^{\text{obs}}(p)$, for an expected probability p. Calibration requires $p^{\text{obs}}(p) = p$, $\forall p \in (0, 1)$. From this generic statement, we can describe different notions of calibration based on how p^{obs} is defined.

The most common form of calibration is **average calibration**, where $\hat{Q}_p(x)$ is the estimated p^{th} quantile of $\mathbf{Y}|x$,

$$p_{avg}^{\text{obs}}(p) := \mathbb{E}_{x \sim \mathcal{F}_{\mathbf{X}}}[\mathbb{F}_{\mathbf{Y}|x}(\hat{Q}_p(x))], \;\; \forall p \in (0,1), \tag{2.1}$$

i.e. the probability of observing the target below the quantile prediction, *averaged over* $\mathcal{F}_{\mathbf{X}}$, is equal to p. Average calibration is often referred to simply as "calibration" [Kuleshov et al., 2018, Cui et al., 2020], and it is amenable to estimation in finite datasets, as follows. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^N$, we can estimate $p_{avg}^{\text{obs}}(p)$ with $\hat{p}_{avg}^{\text{obs}}(D,p) = \frac{1}{N}\sum_{i=1}^N \mathbb{I}\{y_i \leq \hat{Q}_p(x_i)\}$. The degree of error in average calibration is commonly measured by *expected calibration error* Guo et al. [2017], Tran et al. [2020], Cui et al. [2020], $\text{ECE}(D, \hat{Q}) = \frac{1}{m}\sum_{j=1}^m \left|\hat{p}_{avg}^{\text{obs}}(D, p_j) - p_j\right|$, where p_j is a range of expected probabilities of interest. Note that if our quantile estimate achieves average calibration then $\hat{p}_{avg}^{\text{obs}} \to p$ (and thus ECE $\to 0$) as $N \to \infty$, $\forall p \in (0,1)$.

It may be possible to have an uninformative, yet average calibrated model. For example, quantile predictions that match the true *marginal* quantiles of $\mathcal{F}_{\mathbf{Y}}$ will be average calibrated, but will hardly be useful since they do not depend on the input x. Therefore, the notion of **sharpness** is also considered, which quantifies the concentration of distributional predictions [Gneiting et al., 2007]. For example, in predictions that parameterize a Gaussian, the variance of the predicted distribution is often taken as a measure of sharpness. There generally exists a tradeoff between average calibration and sharpness [Murphy, 1973, Gneiting et al., 2007].

Recent works have suggested a notion of calibration stronger than average calibration, called adversarial group calibration [Zhao et al., 2020]. This stems from the notion of **group calibration** [Kleinberg et al., 2016, Hébert-Johnson et al., 2017], which prescribes measurable subsets $\mathcal{S}_i \subset \mathcal{X}$ s.t. $P_{x \sim \mathcal{F}_{\mathbf{X}}}(x \in \mathcal{S}_i) > 0$, $i = 1, \ldots, k$, and requires the predictions to be average calibrated within each subset. Adversarial group calibration then requires average calibration for *any subset of* $\mathcal{X}$ *with non-zero measure*. Denote $\mathbf{X}_{\mathcal{S}}$ as a random variable that is conditioned on being in the set $\mathcal{S}$. For **adversarial group calibration**, the observed probability is

$$\begin{aligned} p_{adv}^{\text{obs}}(p) := \;\; & \mathbb{E}_{x \sim \mathcal{F}_{\mathbf{X}_{\mathcal{S}}}}[\mathbb{F}_{\mathbf{Y}|x}(\hat{Q}_p(x))], \\ \forall p \in (0,1), \;\; & \forall \mathcal{S} \subset \mathcal{X} \text{ s.t. } P_{x \sim \mathcal{F}_{\mathbf{X}}}(x \in \mathcal{S}) > 0. \end{aligned} \tag{2.2}$$

With a finite dataset, we can measure a proxy of adversarial group calibration by measuring average calibration within all subsets of the data with sufficiently many points.

An alternative but widely used family of evaluation metrics is **proper scoring rules** [Gneiting and Raftery, 2007]. Proper scoring rules are summary statistics of overall performance of a distributional prediction, and are defined such that the true underlying distribution optimizes the expectation of the scoring rule. Given a scoring rule $S(\hat{F}, (x, y))$, where $x \sim \mathbb{F}_{\mathbf{X}}$, $y \sim \mathbb{F}_{\mathbf{Y}|x}$, the expectation of the scoring rule is $S(\hat{F}, \mathbb{F}) = \mathbb{E}_{\mathbf{X},\mathbf{Y}}[S(\hat{F}, (x, y))]$, and S is said to be a proper scoring rule if $S(\mathbb{F}, \mathbb{F}) \geq S(\hat{F}, \mathbb{F}), \forall \hat{F} \in \mathcal{F}$.

There are a variety of proper scoring rules, based on the representation of the distributional prediction. Since these rules consider both calibration and sharpness together in a single value [Gneiting et al., 2007], they also serve as optimization objectives for UQ. For example, the logarithmic score is a popular proper scoring rule for density predictions [Lakshminarayanan et al., 2017, Pearce et al., 2018a, Detlefsen et al., 2019], and it is used as a loss function via *negative log-likelihood* (**NLL**). The **check score** is widely used for quantile predictions and also known as the *pinball* *loss*. The **interval score** is commonly used for prediction intervals (a pair of quantiles with a prescribed expected coverage), and the continuous ranked probability score (**CRPS**) is popular for CDF predictions [1]. We refer the reader to Gneiting and Raftery [2007] for the definition of each scoring rule.

Given the wide range of metrics available, one might naturally ask, "is there one metric to rule them all?" Previous work has investigated some aspects of this question. For example, Chung et al. [2020b] noted the mismatch between the check score and average calibration, and Gneiting and Raftery [2007] and Bracher et al. [2021] point out cases in which disagreements can occur between some scoring rules. Still, whether there exists a golden metric in UQ is an open research problem. We instead suggest that there is virtue in inspecting various metrics simultaneously, which is made easy by Uncertainty Toolbox, as we show below.

2.4 Case Study on Training, Evaluating PNNs

To demonstrate the capabilities of Uncertainty Toolbox, we provide a case study on training PNNs with various loss objectives, and use the toolbox to examine the results.

A PNN is a neural network that assumes a conditional Gaussian for the predictive distribution, thus for any input point x, outputs an estimate of the mean and the covariance, $\hat{\mu}(x)$, $\hat{\Sigma}(x)$. This NN structure has been proposed as early as Nix and Weigend [1994], but it has been popularized as a UQ method in deep learning by Lakshminarayanan et al. [2017], and it remains one of the most popular UQ methods to date.

The standard method of training PNNs is to optimize the logarithmic score, i.e. NLL loss. However, based on the training principle, "optimize a proper score to improve UQ quality" [Lakshminarayanan et al., 2017] (also referred to as "optimum score estimation" by Gneiting and Raftery [2007]), we can in fact optimize many more proper scoring rules. In this study, we train PNNs using several different methods by optimizing with respect to either NLL, CRPS, check score, or interval score. Afterwards, we assess the predictive UQ quality with Uncertainty Toolbox. We summarize main details of the experiment below (full details in Appendix 2.A).

Dataset The data was generated with a mean function $y = \sin(x/2) + x\cos(0.8x)$ and heteroscedastic Gaussian noise was added to generate the observations, y, for each input $x \sim \text{unif}[-10, 10]$. The train, validation and test splits consisted of $200, 100, 100$ points, respectively.

Training A separate model was trained for each loss function with full batch gradient decent and learning rate $1e^{-3}$, for 2000 epochs while tracking the validation loss. To optimize the check and interval scores, a batch of 30 expected probabilities $p_i \sim \text{unif}(0, 1)$ was selected and the scores for each p_i were summed to compute the loss [Tagasovska and Lopez-Paz, 2019, Chung et al., 2020b]. All reported results are based on the model with best validation loss.

Analysis We first visually observe UQ performance on the test set. Figure 2.2 (row 1) shows all of the methods approximately recovering the true level of heteroscedastic noise. Notably, NLL converges to a solution s.t. for $x < -5$, there is high error in mean estimation, which is compensated for with high (and wrong) variance estimates. The widths of the prediction intervals (PIs) in

[1]Proper scoring rules are usually *positively oriented* (i.e. greater value is more desirable), and their negative is taken as a loss function to minimize. In our work, we always report proper scoring rules in their *negative orientation* (i.e. as a loss).

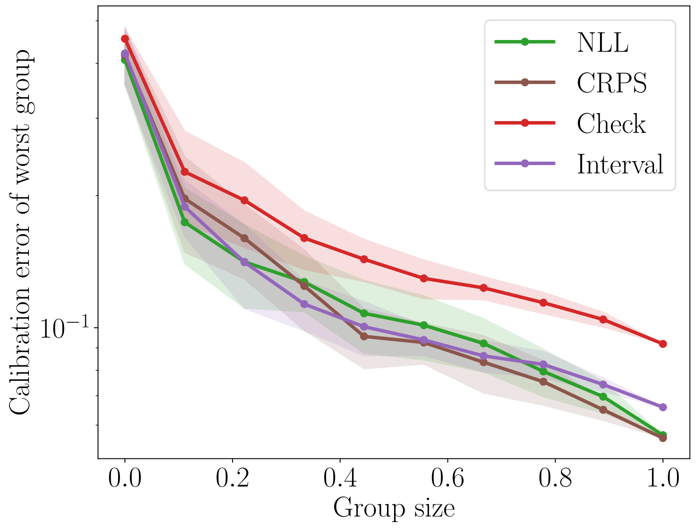


Figure 2.1: **Adversarial Group Calibration.** Group size refers to proportion of test dataset size, and the shades represent ± 1 standard error for the calibration error of the worst group.

		Metrics							
		RMSE	MAE	ECE	Sharpness	NLL	CRPS	Check	Interval
Methods	NLL	1.689	0.852	0.057	1.451	2.214	0.604	0.305	2.990
	CRPS	**0.864**	0.568	**0.056**	0.729	1.266	**0.427**	**0.215**	2.323
	Check	0.880	**0.566**	0.092	**0.720**	4.264	0.434	0.219	2.434
	Interval	0.916	0.600	0.066	0.722	**0.780**	0.447	0.226	**2.309**
	Ground Truth	*0.824*	*0.530*	*0.013*	*0.831*	*−0.083*	*0.370*	*0.187*	*1.758*

Table 2.1: **Scalar Evaluation Metrics.** Each row shows evaluations metrics for a single method (i.e. loss function). RMSE (root mean squared error) and MAE (mean absolute error) are accuracy metrics. The best method for each metric is in **bold**. While these values are based on one seed, we show results across 5 random seeds with standard error in Appendix 2.B.

Figure 2.2 (row 2) also show how NLL is erroneously too wide. Meanwhile, comparisons with the ground truth PIs (far right plot) show that CRPS, Check, and Interval all tend to have too narrow (sharp) predictions. This is further confirmed via the *Sharpness* metric column in Table 2.1, and we can also observe the ramifications in *average calibration* in Figure 2.2 (row 3): NLL's observed proportions in an interval tend to be greater than the expected proportion, signaling under-confidence (i.e. PIs that are too wide). The opposite case occurs for the other methods, and over-confidence (due to PIs that are too sharp) is especially pronounced in CRPS and Check. While NLL may seem average calibrated (with second lowest ECE), adversarial group calibration in Figure 2.1 shows that CRPS and Interval are better calibrated for smaller subsets of the domain, and achieve better adversarial group calibration.

While the proper score metrics in Table 2.1 add another facet to the analysis, they also underscore the complex nature of assessing UQ. Each proper score has its own, separate ranking of the four methods, and they are also split on which one is best; simply given this set of proper scoring rules,

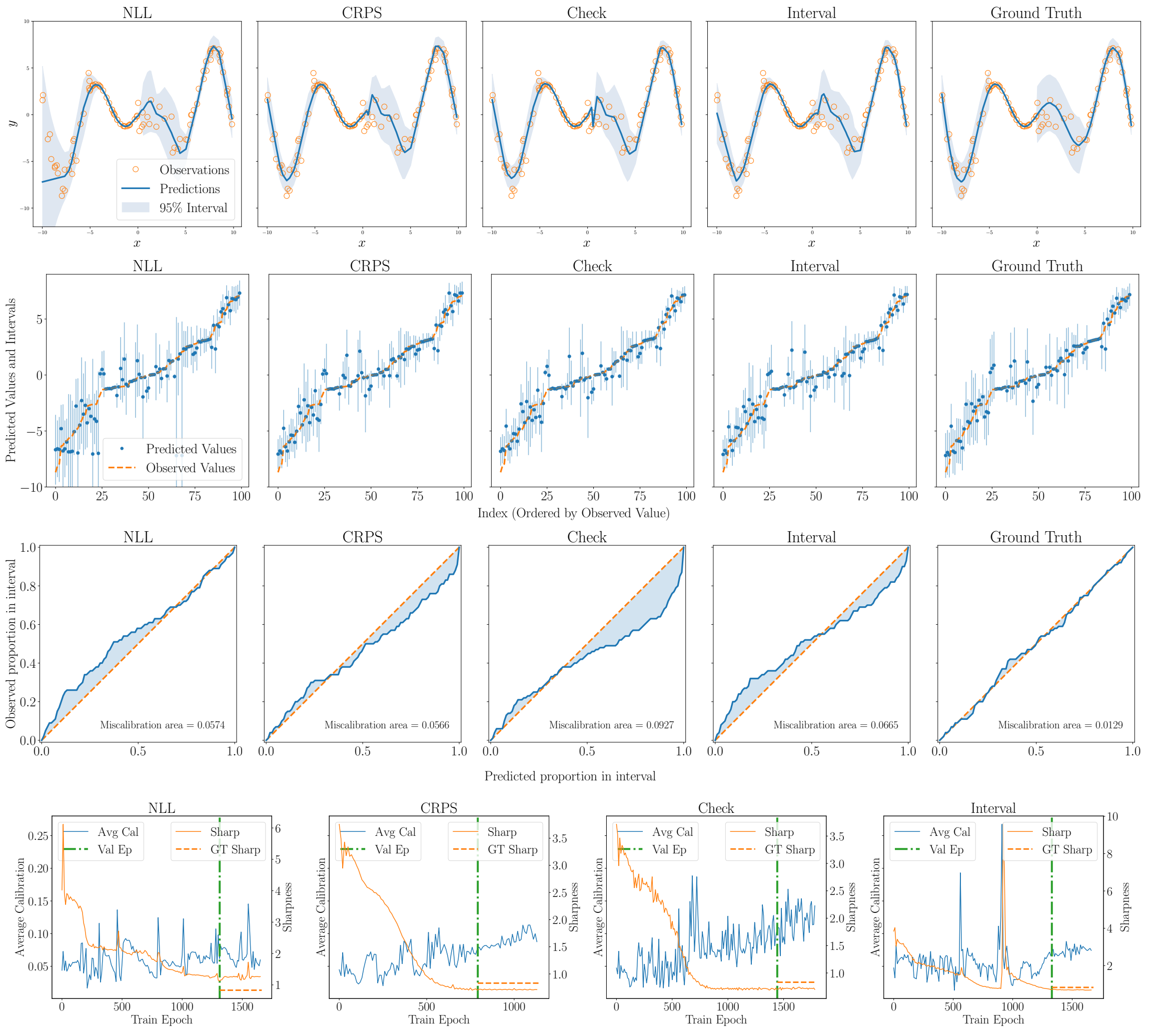


Figure 2.2: Rows from top to bottom: **(1)** Test observations, with predicted mean and confidence bands. **(2)** Test observations, the predicted mean, and prediction interval, in order of test observations. **(3)** Average calibration plot, with predicted proportions (*expected probability*) on x axis, observed proportions (*observed probability*) on y axis. **(4)** Training curves: average calibration (left y axis), sharpness (right y axis). GT Sharp denotes the true sharpness (noise level) of the data, and Val Ep denotes the epoch with lowest validation loss.

we believe it would be difficult to choose a single best method. Lastly, we note how a lower proper score may not necessarily indicate better calibration (Figure 2.2 (row 4)). Even while the proper scores improve on the test set (until around the validated epoch), calibration tends to get worse, while the predictions get sharper. Notably, CRPS and Check converge to a solution which is sharper than the true sharpness. This is problematic for calibration since a UQ sharper than the true sharpness will never be calibrated.

Conclusion This case study demonstrates that, even with numerous evaluation metrics at our disposal, the analysis of UQ for regression problems may not be straightforward. It also highlights limitations of the evaluation metrics, as relying on a single one (or small subset), may imply a conclusion counter to what other metrics signal. In the face of such limitations, we believe it is important to examine a suite of metrics simultaneously and perform a holistic evaluation of UQ quality. Not only does Uncertainty Toolbox provide this functionality, but it also offers recalibration for pre-trained UQ models, and resources that give key terms, explanations, and seminal works for those unfamiliar with the field. We hope that this toolbox is useful for accelerating and uniting efforts for uncertainty in machine learning.

Appendices for Chapter 2

2.A Details of the Case Study Experiment

2.A.1 Details on Dataset

The synthetic dataset in Section 2.4 was created with a mean function $y = \sin(x/2) + x\cos(0.8x)$ with $x \sim$ unif$[-10, 10]$. The support $[-10, 10]$ was partitioned into 4 quadrants, and different levels of 0 mean, Gaussian noise was added to the mean function to create the y observations.

$$\begin{aligned} -10 \leq x < -5 &: \text{ noise} \sim \mathcal{N}(0, 1^2) \\ -5 \leq x < 0 &: \text{ noise} \sim \mathcal{N}(0, 0.01^2) \\ 0 \leq x < 5 &: \text{ noise} \sim \mathcal{N}(0, 1.5^2) \\ 5 \leq x \leq 10 &: \text{ noise} \sim \mathcal{N}(0, 0.5^2) \end{aligned}$$

2.A.2 Model Details

We used the same neural network architecture across all methods (i.e. loss functions): 3 layers of 64 hidden units with ReLU non-linearities, and 2 output units: one for the conditional mean $\hat{\mu}(x)$ and one for the conditional log-variance $\log \hat{\sigma^2}(x)$. We used the same learning rate $1e^{-3}$ and full batch size (200) for all methods. During training, we track the corresponding loss function on the validation set, and at the end of 2000 epochs, the final model was backtracked to the model with lowest validation loss. All reported test results are based on this backtracked model.

2.A.3 Calculation of Evaluation Metrics

This section describes how each of the reported metrics is computed within Uncertainty Toolbox, given a finite dataset $D = \{(x_i, y_i)_{i=1}^N\}$.

Accuracy Metrics

The root mean squared error (RMSE) and mean absolute error (MAE) are computed with the mean prediction $\hat{\mu}(x)$, following the standard definitions.

$$\text{RMSE}(D, \hat{\mu}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{\mu}(x_i))^2}$$

$$\text{MAE}(D, \hat{\mu}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{\mu}(x_i)|$$

Calibration Metrics

To measure the calibration metrics (average calibration, adversarial group calibration), expected probabilities are discretized from 0.01 to 0.99 in 0.01 increments (i.e. 0.01, 0.02, . . . , 0.97, 0.98, 0.99). and the observed probabilities are calculated for each of these 99 expected probabilities.

ECE (measure of average calibration) is computed following the definition given in Section 3.2.2.

The procedure in which we measure adversarial group calibration is the following. For a given test set, we scale group size between 1% and 100% of the full test set size, in 10 equi-spaced intervals. With each group size, we draw 20 random groups from the test set and record the worst calibration incurred across these 20 random groups. The adversarial group calibration figure (Figure 2.1) plots the mean worst calibration incurred with ± 1 standard error in shades, for each group size. This is also the method used by Zhao et al. [2020] to measure adversarial group calibration.

Sharpness

Sharpness is measured as the mean of the standard deviation predictions on the test set. Note that sharpness is a property of the prediction *only* and does not take into consideration the true distribution.

Proper Scoring Rules

The proper scoring rules (NLL, CRPS, check score, interval score) are measured as the mean of the score on the test set.

2.B Numerical Results Across Multiple Trials

The results presented in Section 2.4 are based on one random seed. Below, we present the numerical results across 5 random seeds: $[0, 1, 2, 3, 4]$.

		Metrics			
		RMSE	MAE	ECE	Sharpness
Methods	NLL	2.048 ± 0.125	1.073 ± 0.080	$\mathbf{0.029} \pm 0.007$	1.746 ± 0.155
	CRPS	$\mathbf{1.023} \pm 0.090$	$\mathbf{0.661} \pm 0.054$	0.044 ± 0.005	0.897 ± 0.114
	Check	1.045 ± 0.105	0.672 ± 0.065	0.050 ± 0.011	$\mathbf{0.874} \pm 0.117$
	Interval	1.169 ± 0.187	0.745 ± 0.101	0.039 ± 0.009	0.915 ± 0.130
	Ground Truth	*0.962 ± 0.064*	*0.618 ± 0.042*	*0.019 ± 0.002*	*0.925 ± 0.052*

		Metrics			
		NLL	CRPS	Check	Interval
Methods	NLL	1.677 ± 0.343	0.766 ± 0.060	0.386 ± 0.030	3.885 ± 0.330
	CRPS	1.112 ± 0.111	$\mathbf{0.492} \pm 0.040$	$\mathbf{0.248} \pm 0.020$	$\mathbf{2.687} \pm 0.186$
	Check	1.635 ± 0.661	0.501 ± 0.048	0.253 ± 0.024	2.741 ± 0.224
	Interval	$\mathbf{0.961} \pm 0.062$	0.546 ± 0.073	0.276 ± 0.037	2.875 ± 0.352
	Ground Truth	*0.187 ± 0.115*	*0.435 ± 0.033*	*0.219 ± 0.017*	*2.122 ± 0.177*

Table 2.2: **Scalar Evaluation Metrics.** Each row shows evaluation metrics for a single method (i.e. loss function), and the mean with ± 1 standard error is shown. The best mean for each metric has been **bolded**.

Part II

Methods for Calibrated Uncertainty Quantification

3 | Beyond Pinball Loss: Quantile Methods for Calibrated Uncertainty Quantification

Among the many ways of quantifying uncertainty in a regression setting, specifying the full quantile function is attractive, as quantiles are amenable to interpretation and evaluation. A model that predicts the true conditional quantiles for each input, at all quantile levels, presents a correct and efficient representation of the underlying uncertainty. To achieve this, many current quantile-based methods focus on optimizing the pinball loss. However, this loss restricts the scope of applicable regression models, limits the ability to target many desirable properties (e.g. calibration, sharpness, centered intervals), and may produce poor conditional quantiles. In this work, we develop new quantile methods that address these shortcomings. In particular, we propose methods that can apply to any class of regression model, select an explicit balance between calibration and sharpness, optimize for calibration of centered intervals, and produce more accurate conditional quantiles. We provide a thorough experimental evaluation of our methods, which includes a high dimensional uncertainty quantification task in nuclear fusion. Code is available at `https://github.com/YoungseogChung/calibrated-quantile-uq`.

3.1 Introduction

Uncertainty quantification (UQ) in machine learning typically refers to the task of quantifying the confidence of a given prediction. This measure of certainty can be crucial in a variety of downstream applications, including Bayesian optimization [Kushner, 1964, Mockus et al., 1978, Shahriari et al., 2015], model-based reinforcement learning [Yu et al., 2020, Malik et al., 2019, Chua et al., 2018, Garcia and Fernández, 2012], and in high-stakes predictions where mistakes incur large costs [Rudin, 2019, Wexler, 2017].

While the common goal of UQ is to describe predictive distributions over outputs for given inputs, the representation of the distributional prediction varies across methods. For example, some methods assume a parametric distribution and return parameter estimates [Zhao et al., 2020, Detlefsen et al., 2019, Lakshminarayanan et al., 2017], while others return density function estimates, as is common in Bayesian methods [Maddox et al., 2019, Liu et al., 2019, Hernández-Lobato and Adams, 2015, Blundell et al., 2015, Koller and Friedman, 2009, Rasmussen, 2004]. Alternatively, many methods represent predictive uncertainty with quantile estimates [Fasiolo et al., 2020, Salem et al., 2020, Tagasovska and Lopez-Paz, 2019, Pearce et al., 2018b, Jeon et al., 2016].

Quantiles provide an attractive representation for uncertainty because they can be used to model complex distributions without parametric assumptions, are interpretable with units in the target output space, allow for easy construction of prediction intervals, and can be used to efficiently sample from the predictive distribution via inverse transform sampling [Hao et al., 2007, Koenker and Hallock, 2001]. Learning the quantile for a single quantile level is a well studied problem in quantile regression (QR) [Koenker, 2005, Koenker and Hallock, 2001], which typically involves optimizing the so-called *pinball loss*, a tilted transformation of the absolute value function. Given a target y, a prediction $\hat{y}$, and quantile level $\tau \in (0, 1)$, the pinball loss ρ_τ is defined as

$$\rho_\tau(y, \hat{y}) = (\hat{y} - y)(\mathbb{I}\{y \leq \hat{y}\} - \tau). \tag{3.1}$$

By training for all quantiles simultaneously, recent works have made concrete steps in incorporating QR methods to form competitive UQ methods which output the full predictive distribution [Rodrigues and Pereira, 2020, Tagasovska and Lopez-Paz, 2019].

In this work, we highlight some limitations of the pinball loss and propose several methods to address these shortcomings. Specifically, we explore the following:

- **Model agnostic QR.** Optimizing the pinball loss often restricts the choice of model family for which we can provide UQ. We propose an algorithm to learn all quantiles simultaneously by utilizing methods from conditional density estimation. This algorithm is agnostic to model class and can be applied to *any* regression model.
- **Explicitly balancing calibration and sharpness.** While the pinball loss, as a proper scoring rule, targets both calibration and sharpness, the balance between these two quantities is made implicitly, which may result in a poor optimization objective. We propose a tunable loss function that targets calibration and sharpness separately, and allows the end-user to set an *explicit* balance.
- **Centered intervals.** In practice, we often desire uncertainty predictions made with centered intervals, which are not targeted via the pinball loss. We propose an alternative loss function that is better suited for this goal.
- **Encouraging individual calibration.** Perfect quantile forecasts will satisfy *individual* calibration (Eq. 3.2), which is a much stricter condition than the more-commonly used notion of *average* calibration (Eq. 3.3). We introduce a training procedure that aims to improve quantile predictions beyond average calibration, and demonstrate its efficacy via *adversarial group* calibration (Eq. 3.4).

We proceed by first describing methods of assessing the quality of predictive UQ and the pitfalls of optimizing the pinball loss in Section 4.2. Drawing motivation from this, we then present our proposed methods in Section 3.3. In Section 6.3, we demonstrate our methods experimentally, where we model predictive uncertainty on benchmark datasets, and on a high-dimensional, real-world uncertainty estimation task in the area of nuclear fusion.

3.2 Preliminaries and Background

We first lay out the notation, terminology, and class of models considered in this paper. Then we provide an overview of evaluation metrics in UQ and demonstrate how the pinball loss may be inadequate both as an evaluation metric and as an optimization objective.

3.2.1 Notation

Bold upper case letters **X, Y** denote random variables, lower case letters x, y, denote their values, and calligraphic upper case letters $\mathcal{X}, \mathcal{Y}$ denote sets of possible values. We use $x \in \mathcal{X}$ to denote the input feature vector and $y \in \mathcal{Y}$ to denote the corresponding target. Additionally, we consider the regression setting where $\mathcal{Y} \subset \mathbb{R}$ and $\mathcal{X} \subset \mathbb{R}^n$. We use $\mathbb{F}_{\mathbf{X}}, \mathbb{F}_{\mathbf{Y}|x}, \mathbb{F}_{\mathbf{Y}}$ to denote the true cumulative distribution of the subscript random variable. For any $x \in \mathcal{X}$, we assume there exists a true conditional distribution $\mathbb{F}_{\mathbf{Y}|x}$ over $\mathcal{Y}$, and we assume $\mathbb{Q}_p(x)$ denotes the true p^{th} quantile of this distribution, i.e. $\mathbb{F}_{\mathbf{Y}|x}(\mathbb{Q}_p(x)) = p$. Any estimates of the true functions $\mathbb{F}, \mathbb{Q}_p$ will be denoted with a hat, $\hat{\mathbb{F}}, \hat{Q}_p$. We will specifically refer to any family of estimates for $\mathbb{Q}_p$, with $p \in (0, 1)$, as a "quantile model", denoted $\hat{Q} : \mathcal{X} \times (0, 1) \to \mathcal{Y}$. Unless otherwise noted, we will always consider the *conditional* problem of estimating quantities in the target space $\mathcal{Y}$, conditioned on a value $x \in \mathcal{X}$.

3.2.2 Assessing the Quality of Predictive UQ

While various metrics have been proposed to assess the quality of UQ, there has been a great deal of recent focus on the notions of *calibration* and *sharpness* [Fasiolo et al., 2020, Cui et al., 2020, Zhao et al., 2020, Tran et al., 2020, Song et al., 2019, Kuleshov et al., 2018, Guo et al., 2017, Gneiting et al., 2007]. We introduce calibration here, but for a more thorough treatment, see Zhao et al. [2020]. Broadly speaking, calibration in the regression setting requires that the probability of observing the target random variable below a predicted p^{th} quantile is equal to the *expected probability* p, for all $p \in (0, 1)$. We refer to the former quantity as the *observed probability* and denote it $p^{\text{obs}}(p)$, for an expected probability p, which we will write as p^{obs} when it is clear from context. Calibration requires $p^{\text{obs}}(p) = p, \forall p \in (0, 1)$. From this generic statement, we can describe different notions of calibration based on how p^{obs} is defined.

A model is **individually calibrated** if it outputs the true conditional quantiles, i.e. $\hat{Q}_p(x) = \mathbb{Q}_p(x)$. In this case, we define the observed probability to be

$$p^{\text{obs}}_{indv}(p, x) := \mathbb{F}_{\mathbf{Y}|x}(\hat{Q}_p(x)), \ \ \forall x \in \mathcal{X}, \ \ \forall p \in (0, 1). \tag{3.2}$$

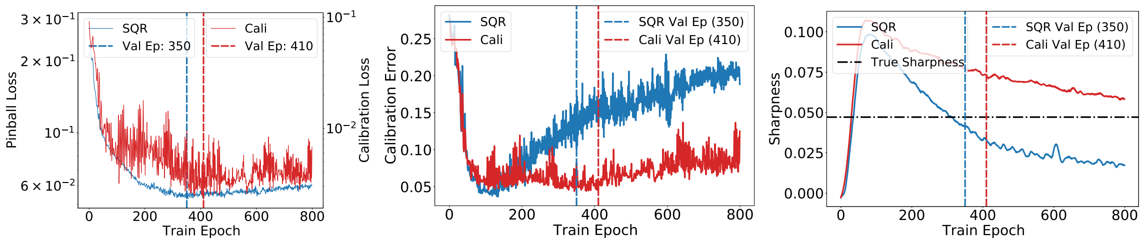
In words, this requires that the probability of observing y below the quantile prediction is equal to p, *at each point $x \in \mathcal{X}$, individually*. If we can verify this property for all $x \in \mathcal{X}$, then by definition, we will know the quantile output is correct and precisely the true conditional quantile. However, individual calibration is typically unverifiable with finite datasets in the assumption-less case [Zhao et al., 2020].

A relaxed condition is **average calibration**. In this case, we define the observed probability to be

$$p^{\text{obs}}_{avg}(p) := \mathbb{E}_{x \sim \mathcal{F}_{\mathbf{X}}}[\mathbb{F}_{\mathbf{Y}|x}(\hat{Q}_p(x))], \ \ \forall p \in (0, 1), \tag{3.3}$$

i.e. the probability of observing the target below the quantile prediction, *averaged over $\mathcal{F}_{\mathbf{X}}$*, is equal to p. Average calibration is often referred to simply as "calibration" [Cui et al., 2020, Kuleshov et al., 2018]. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^N$, we can estimate $p^{\text{obs}}_{avg}(p)$ with $\hat{p}^{\text{obs}}_{avg}(D, p) = \frac{1}{N}\sum_{i=1}^N \mathbb{I}\{y_i \le \hat{Q}_p(x_i)\}$. Note that if our quantile estimate achieves average calibration then $\hat{p}^{\text{obs}}_{avg} \to p$ as $N \to \infty, \forall p \in (0, 1)$. The degree of error in average calibration is commonly measured by *expected calibration error* Tran et al. [2020], Cui et al. [2020], Guo et al. [2017], $\text{ECE}(D, \hat{Q}) = \frac{1}{m}\sum_{j=1}^m \left|\hat{p}^{\text{obs}}_{avg}(D, p_j) - p_j\right|$, where $p_j \sim \text{Unif}(0, 1)$.

It may be possible to have an uninformative, yet average calibrated model. For example, quantile predictions that match the true *marginal* quantiles of $\mathcal{F}_{\mathbf{Y}}$ will be average calibrated, but will hardly



(a) Test Loss Curves (b) Test Calibration (c) Test Sharpness

Figure 3.1: **(a)** Test loss continues to decrease until the validated epoch. **(b-c)** At the validated epoch, *SQR* (optimizes *pinball loss*) is highly miscalibrated while sharper than the true sharpness level. *Cali* (optimizes proposed *calibration loss*) is better calibrated while less sharp than the true sharpness.

be useful since they do not depend on the input x. Therefore, the notion of **sharpness** is also considered, which quantifies the concentration of distributional predictions [Gneiting et al., 2007]. For example, for non-parametric predictions, the width of a centered 95% prediction interval is often used as a measure of sharpness. There generally exists a tradeoff between average calibration and sharpness [Gneiting et al., 2007, Murphy, 1973].

Recent works have suggested a notion of calibration stronger than average calibration, called adversarial group calibration [Zhao et al., 2020]. This stems from the notion of **group calibration** [Hébert-Johnson et al., 2017, Kleinberg et al., 2016], which prescribes measurable subsets $\mathcal{S}_i \subset \mathcal{X}$ s.t. $P_{x \sim \mathcal{F}_\mathbf{X}}(x \in \mathcal{S}_i) > 0$, $i = 1, \ldots, k$, and requires the predictions to be average calibrated within each subset. Adversarial group calibration then requires average calibration for *any subset of $\mathcal{X}$ with non-zero measure*. Denote $\mathbf{X}_\mathcal{S}$ as a random variable that is conditioned on being in the set $\mathcal{S}$. For **adversarial group calibration**, the observed probability is

$$p^{\text{obs}}_{adv}(p) := \mathbb{E}_{x \sim \mathcal{F}_{\mathbf{X}_\mathcal{S}}}[\mathbb{F}_{\mathbf{Y}|x}(\hat{Q}_p(x))], \ \ \forall p \in (0,1), \ \ \forall \mathcal{S} \subset \mathcal{X} \text{ s.t. } P_{x \sim \mathcal{F}_\mathbf{X}}(x \in \mathcal{S}) > 0. \tag{3.4}$$

With a finite dataset, we can measure a proxy of adversarial group calibration by measuring the average calibration within all subsets of the dataset with sufficiently many points.

Intuitively, individual calibration inspects the discrepancy between p^{obs} and p for individual inputs $x \in \mathcal{X}$, adversarial group calibration relaxes this by inspecting any subset of $\mathcal{X}$ with non-zero measure, and average calibration relaxes this further by considering the full distribution of $\mathbf{X}$.

One alternative family of evaluation metrics is **proper scoring rules** [Gneiting and Raftery, 2007]. Proper scoring rules are summary statistics of overall performance of a distributional prediction and consider both calibration and sharpness jointly [Gneiting et al., 2007]. For example, negative log-likelihood (NLL) is a proper scoring rule that is commonly used with density predictions [Detlefsen et al., 2019, Pearce et al., 2018a, Lakshminarayanan et al., 2017]. For quantile predictions, one proper score is the **check score**, *which is identical to the pinball loss*. Since proper scoring rules consider both calibration and sharpness together in a single value, they can serve as optimization objectives for UQ. For example, optimizing the pinball loss is the traditional method in quantile regression [Koenker and Bassett Jr, 1978], and many recent quantile-based UQ methods focus on optimizing this objective [Rodrigues and Pereira, 2020, Tagasovska and Lopez-Paz, 2019, Cannon, 2018, Xu et al., 2017].

In this work, however, we note that the balance between calibration and sharpness implied by the pinball loss is arbitrary and depends on the expressivity of the model class—and with highly expressive models, this balance can be heavily skewed towards sharpness. In their seminal work on probabilistic forecasts, Gneiting and Raftery [2007] contend that the goal of probabilistic forecasting is to "maximize the sharpness of the predictive distribution subject to calibration", i.e. calibration should be first achieved and then sharpness optimized. We show that common machine learning methods that use the pinball loss objective may in fact lead to an arbitrary and miscalibrated UQ.

Proposition 1. *Consider a finite dataset D, the pinball loss ρ_τ (Eq. 3.1) and a quantile model $f : \mathcal{X} \times (0, 1) \to \mathcal{Y}$ that is average calibrated on D, i.e. $ECE(D, f) = 0$. Then there always exists another quantile model $g : \mathcal{X} \times (0, 1) \to \mathcal{Y}$, such that, for any quantile level $\tau \in (0, 1)$, g has lower pinball loss than f on D, i.e. $\sum_{i=1}^{N} \rho_\tau(y_i, g_\tau(x_i)) < \sum_{i=1}^{N} \rho_\tau(y_i, f_\tau(x_i))$, but worse average calibration than f, i.e. $ECE(D, g) > ECE(D, f)$.*

Proof: The proof is given in Appendix 3.A.1.

This proposition essentially states how the pinball loss can become detached from calibration, and we show its practical ramifications via a synthetic example in Figure 3.1 (experiment details in Appendix 3.E.1). We first note in Figure 3.1 (a) and (b) that even while the pinball loss decreases on the test set, test calibration worsens (while sharpness improves). Further, at the best validation epoch, optimizing the pinball loss converges to a solution that is sharper than the true noise level. Note that a UQ that is sharper than the true noise level will *never* be calibrated (meanwhile, a less sharp prediction *can still be calibrated*, e.g. the marginal distribution $\mathcal{F}_{\mathbf{Y}}$). While this may seem like an issue that can simply be addressed with regularization, we demonstrate in Appendix 3.E.2 how that is not the case. These pitfalls motivate our methods in Section 3.3.

3.3 Methods

We propose four methods that aim to produce an improved quantile model. The first is a model-agnostic procedure that relies on conditional density estimation (Section 3.3.1). To address settings where density estimation may be difficult, we then propose two loss functions to optimize with differentiable models: the combined calibration loss (Section 3.3.2), which directly optimizes calibration and sharpness, and the interval score (Section 3.3.3), which is a proper scoring rule for centered intervals. Finally, we propose a group batching method (Section 3.3.4) that can be applied to the batch optimization procedure for any loss function (e.g. combined calibration loss, interval score, and even pinball loss) to induce better convergence towards adversarial group calibration.

3.3.1 Utilizing Conditional Density Estimation for Model Agnostic QR

One drawback of many existing quantile-based UQ methods is that their training procedure requires differentiable models. In fact, most UQ methods require a specific class of models because of their modeling structure or their loss objective (e.g. Gaussian processes [Rasmussen, 2004], dropout [Gal and Ghahramani, 2016], latent variable models [Koller and Friedman, 2009], simultaneous pinball loss [Tagasovska and Lopez-Paz, 2019], and NLL-based losses [Lakshminarayanan et al., 2017]). This model restriction can be especially unfavorable in practical settings. A domain expert with an established point prediction model and compute infrastructure may want to add UQ without much additional overhead.

To address these issues, we can consider the following model-agnostic procedure. Instead of optimizing a designated loss function, we can consider splitting the given problem into two parts: estimate conditional quantiles directly from data, then regress onto these estimates. The benefit of this method is that, granted we can estimate the conditional quantiles accurately, we can use any regression model to regress onto these quantile estimates. Further, this regression task directly targets the goal of producing the true conditional quantiles (i.e. individual calibration). This

Algorithm 1 MAQR

```
1: Input: Train data $\{x_i, y_i\}_{i=1}^N$, trained regression model $\hat{f}(x)$
2: Calculate residuals $\epsilon_i = y_i - \hat{f}(x_i),\ i \in [N]$, and denote the residual dataset $R = \{x_i, \epsilon_i\}_{i=1}^N$
3: Initialize $D \leftarrow \varnothing$
4: for $i = 1$ to $N$ do
5:     $D_i \leftarrow$ CondQuantilesEstimators($R$, $i$) (Algorithm 2)
6:     $D \leftarrow D \cup D_i$
7: end for
8: Use $D$ to fit a regression model $\hat{g}$
       $\hat{g} : (x, p) \mapsto \epsilon$
9: Output: $\hat{f} + \hat{g}$
```

Algorithm 2 CondQuantilesEstimators

```
1: Input: Dataset $\{x_i, \epsilon_i\}_{i=1}^N$, point index $k \in [N]$
2: $E_{k,d_N} \leftarrow \{\epsilon_i : \text{dist}(x_k, x_i) \leq d_N, i \in [N]\}$
3: Construct an empirical CDF with $E_{k,d_N}$ to produce $\hat{F}_{\mathbf{E}|x_k} : \epsilon \mapsto p \in [0, 1]$
4: Initialize $D \leftarrow \varnothing$
5: for each $\epsilon_j$ in $E_{k,d_N}$ do
6:     $\hat{p}_{k,j} \leftarrow \hat{F}_{\mathbf{E}|x_k}(\epsilon_j)$
7:     $D \leftarrow D \cup \{x_k, \hat{p}_{k,j}, \epsilon_j\}$
8: end for
9: Output: $D$
```

procedure, which we refer to as *Model Agnostic QR* (MAQR), is outlined in Algorithm 1.

MAQR is based on the key assumption that nearby points in $\mathcal{X}$ will have similar conditional distributions, i.e. if $x_j \approx x_k$ then $\mathcal{F}_{\mathbf{Y}|x_j} \approx \mathcal{F}_{\mathbf{Y}|x_k}$. Given this smoothness assumption, we can group neighboring points to estimate the conditional density at each locality over $\mathcal{X}$, with locality determined by the hyperparameter d_N (Algorithm 2, line 2). We then construct an empirical CDF with the group of neighboring points, and conditional quantile estimates are produced with this empirical CDF. These estimates are collected into D (Algorithm 1, line 6), which is ultimately used as the training set for the quantile model $\hat{g}$.

In practice, we perform these steps with *residuals*, by first estimating a mean function $\hat{f}$ (Algorithm 1, line 1). This practical choice stems from existing works in conditional density estimation, which suggests that having 0 conditional mean in the data provides benefits in terms of lower asymptotic mean squared error in the conditional density predictions [Hyndman et al., 1996]. Further, this demonstrates how MAQR can be readily applied in the application setting where an accurate point prediction model often already exist.

Algorithm 1 is a specific implementation of a more general model-agnostic algorithm, in which we directly estimate conditional quantiles from the data with tools from conditional density estimation. We note that using KDEs for conditional density estimation is a well studied problem with theoretical guarantees [Holmes et al., 2007, Hyndman et al., 1996, Stute et al., 1986]. In the case the distance in $\mathcal{X}$ is measured using a uniform kernel with mild assumptions on the bandwidth, Algorithm 1 falls under the guarantees stated by Stute et al. [1986].

Theorem 1 [Stute et al., 1986]. *Assume $\mathcal{Y} \subset \mathbb{R}$, $\mathcal{X} \subset \mathbb{R}^n$, dist$(x_i, x_j) := |x_i - x_j|_\infty$, and that $\hat{F}_{\mathbf{E}|x}$ is constructed using the procedure given in line 5 of Algorithm 1 (i.e. $x_i = x$). Further assume that, as $N \to \infty$, $d_N \to 0$ and that $\sum_{N \geq 1} \exp(-\rho N d_N^n) < \infty$, $\forall \rho > 0$. Then, as $N \to \infty$, for almost all $x \in \mathcal{X}$, $\sup_\epsilon [\hat{F}_{\mathbf{E}|x}(\epsilon) - \mathcal{F}_{\mathbf{E}|x}(\epsilon)] \to 0$ with probability 1.*

This theorem states that in the limit of data, for almost all $x \in \mathcal{X}$, the CDF estimate $\hat{F}_{\mathbf{E}|x}$ will converge uniformly to the true CDF $\mathcal{F}_{\mathbf{E}|x}$ with probability 1. The dataset, D, will therefore be populated with good estimates of the conditional quantile and quantile level pair for x. In

Appendix 3.B, we state the general form of Algorithm 1 and also demonstrate how the algorithm is model agnostic. Through our experiments in Section 6.3, we will show empirically that utilizing these density estimates sidesteps the issues inherent to the pinball loss and produces much higher quality quantile predictions.

3.3.2 Explicitly balancing calibration and sharpness with the combined calibration loss

While MAQR can produce strong results, its performance can suffer in high-dimensional settings, where nonparametric conditional density estimation methods falter. Neural networks (NNs) have shown good performance in high dimensional settings, given their high capacity to approximate complex functions and recent advances in fast gradient-based optimization. We therefore propose a loss-based approach to estimating conditional quantiles for NNs and other differentiable models.

Drawing motivation from the *arbitrary* balance between calibration and sharpness that pinball loss *implicitly* provides, we propose objectives separately for calibration and sharpness, Then, we combine the two objectives into a single loss function that provides an explicit balance between calibration and sharpness that can be chosen by the end user.

We first consider calibration of a quantile prediction, $\hat{Q}_p \in \mathcal{Y}$ for quantile level $p \in (0, 1)$. Here, we omit conditioning on x for clarity. For this prediction to be average calibrated, exactly a p proportion of the true density should lie below $\hat{Q}_p$, i.e. $p_{avg}^{\text{obs}} = P(Y \leq \hat{Q}_p) = p$. While calibration (e.g. $|p_{avg}^{\text{obs}} - p|$) is a non-differentiable objective, by inducing a truncated distribution based on the current level of calibration, we can construct the following calibration objective, which is minimized if and only if the prediction is average calibrated:

$$\begin{aligned}\mathcal{C}(\hat{Q}_p, p) = \mathbb{I}\{\hat{p}_p < p\} * \mathbb{E}[Y - \hat{Q}_p | Y > \hat{Q}_p] * P(Y > \hat{Q}_p) \\ + \mathbb{I}\{\hat{p}_p > p\} * \mathbb{E}[\hat{Q}_p - Y | \hat{Q}_p > Y] * P(\hat{Q}_p > Y), \text{where } \hat{p}_p = P(Y \leq \hat{Q}_p).\end{aligned} \tag{3.5}$$

The empirical calibration objective, $\mathcal{C}(D, \hat{Q}_p, p)$, is then defined as follows:

$$\begin{aligned}\mathcal{C}(D, \hat{Q}, p) = \mathbb{I}\{\hat{p}_{avg}^{\text{obs}} < p\} * \frac{1}{N}\sum_{i=1}^{N}\left[(y_i - \hat{Q}_p(x_i))\mathbb{I}\{y_i > \hat{Q}_p(x_i)\}\right] \\ + \mathbb{I}\{\hat{p}_{avg}^{\text{obs}} > p\} * \frac{1}{N}\sum_{i=1}^{N}\left[(\hat{Q}_p(x_i) - y_i)\mathbb{I}\{\hat{Q}_p(x_i) > y_i\}\right].\end{aligned} \tag{3.6}$$

Note 1: *Intuition of the calibration objective.* For any given p, consider the case when the quantile estimate $\hat{Q}_p$ is below the true p^{th} quantile $\mathbb{Q}_p$. Since $\hat{Q}_p < \mathbb{Q}_p \implies \hat{p}_p < p$, this implies that too much data density lies above $\hat{Q}_p$. In this case, $\mathcal{C}(\hat{Q}_p, p)$ reduces to $\mathbb{E}[Y - \hat{Q}_p | Y > \hat{Q}_p] * P(Y > \hat{Q}_p)$. $\hat{Q}_p$ is pulled higher with the expectation of the truncated distribution that places $\hat{Q}_p$ at the lower bound of the support. In the opposite case, when $\hat{Q}_p > \mathbb{Q}_p$, $\hat{Q}_p$ is pulled lower by the same logic.

Note 2: *Is the proposed calibration objective a proper scoring rule?* Strictly speaking, the calibration objective is a non-decomposable function, hence deviates from the standard convention of proper scoring rules Gneiting and Raftery [2007], which can be "decomposed" into scores for individual examples (x_i, y_i). This simply arises from the fact that measuring average calibration (i.e. $\hat{p}_{avg}^{\text{obs}}$) is non-decomposable. Proper scoring rules are defined such that an optimum of the *expected score* (or *risk*, if we consider the score as a *loss function*) occurs at the true distribution quantity. While an example level *loss* or *score* does not exist due to non-decomposability, we can still show the (expectation-level) score (i.e. $\mathcal{C}(\hat{Q}_p, p)$) is minimized by the true distribution and hence enjoys the optimum property of proper scoring rules.

Proposition 2. *For any quantile level $p \in (0,1)$, the true quantile function $\mathbb{Q}_p$ minimizes the calibration objective, $\mathcal{C}(\hat{\mathbb{Q}}_p, p)$. Further, on a finite dataset D, the empirical calibration objective, $\mathcal{C}(D, \hat{\mathbb{Q}}_p, p)$, is minimized by an average calibrated solution on D, i.e. when $\hat{p}^{obs}_{avg}(D, p) = p$.*

Proof: The proof is given in Appendix 3.A.2.

Note 3: *Non-zero gradients for miscalibrated predictions $\hat{Q}_p$.* We can further show that for a miscalibrated quantile prediction, the gradients of $\mathcal{C}$ are always non-zero. When $\hat{p}_p < p$, $\partial\mathcal{C}(\hat{Q}_p, p)/\partial\hat{Q}_p = -P(Y > \hat{Q}_p) < 0$. Thus increasing $\hat{Q}_p$ decreases the objective $\mathcal{C}$. Similarly, when $\hat{p}_p > p$, $\partial\mathcal{C}(\hat{Q}_p, p)/\partial\hat{Q}_p = P(Y < \hat{Q}_p) > 0$, and an analogous argument follows (proof in Appendix 3.A.3).

As discussed in Section 3.2.2, average calibration by itself is not a sufficient condition for meaningful UQ, hence we also desire *sharp* quantile models, with more-concentrated (less dispersed) distributions. We can induce this property in quantile predictions by predicting the $(1-p)^{\text{th}}$ quantile $\hat{Q}_{1-p}(x_i)$ alongside each prediction $\hat{Q}_p(x_i)$ and penalizing the width between the quantile predictions:

$$\mathcal{P}(\hat{Q}_p, p) = \mathbb{E}\left[\left|\hat{Q}_p - \hat{Q}_{1-p}\right|\right]. \tag{3.7}$$

The empirical sharpness objective, $\mathcal{P}(D, \hat{Q}_p, p)$, is then defined as follows:

$$\mathcal{P}(D, \hat{Q}, p) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}\hat{Q}_{1-p}(x_i) - \hat{Q}_p(x_i) & (p \leq 0.5)\\ \hat{Q}_p(x_i) - \hat{Q}_{1-p}(x_i) & (p > 0.5).\end{cases} \tag{3.8}$$

It is important to note that the true underlying distribution will not have 0 sharpness if there is significant noise, and sharpness should be optimized subject to calibration. Therefore, we should only penalize sharpness when the data suggests our quantiles are too dispersed, i.e. when $\left|p^{\text{obs}}_{avg}(p) - p^{\text{obs}}_{avg}(1-p)\right|$, the *observed coverage* between the pair of quantiles $\hat{Q}_p(x_i)$ and $\hat{Q}_{1-p}(x_i)$, is greater than $|2p-1|$, the *expected coverage*.

Combining the calibration and sharpness terms, we have the **combined calibration loss**

$$\mathcal{L}(D, \hat{Q}_p, p) = (1-\lambda)\mathcal{C}(D, \hat{Q}_p, p) + \lambda\mathcal{P}(D, \hat{Q}_p, p). \tag{3.9}$$

The hyperparameter $\lambda \in [0,1]$ sets the explicit balance between calibration and sharpness. Note that setting $\lambda = 0$ may not always be desirable, since optimizing $\mathcal{C}(D, \hat{Q}_p, p)$ alone may converge to quantiles of the marginal distribution, $F_{\mathbf{Y}}$. Further, in certain downstream applications that utilize UQ, a sharper prediction, even at the cost of worse calibration, may result in higher utility, and λ can be tuned according to the utility function of the application. In our experiments, we tune λ by cross-validating with adversarial group calibration as it is the strictest notion of calibration that can be estimated with a finite dataset. Since we learn a quantile model that outputs the conditional quantile estimates for all probabilities, our training objective is $\mathbb{E}_{p\sim\text{Unif}(0,1)}\mathcal{L}(D, \hat{\mathbb{Q}}_p, p)$.

3.3.3 Encouraging calibration of centered intervals with the interval score

The combined calibration loss (Eq. 3.9) optimizes average calibration, which targets observed probabilities below a quantile. In many applications, however, we may desire a centered prediction interval (PI) which requires a pair of quantile predictions. A centered 95% PI, for example, is a pair of quantile predictions at quantile levels 0.025 and 0.975. Hence, for the average calibration of the

p^{th} *centered interval*, we want $\left[\hat{p}^{\text{obs}}_{avg}(0.5 + \frac{p}{2}) - \hat{p}^{\text{obs}}_{avg}(0.5 - \frac{p}{2})\right]$ (the PI's observed probability, a.k.a. prediction interval coverage probability (PICP) [Tagasovska and Lopez-Paz, 2019, Kabir et al., 2018, Pearce et al., 2018b]) to be equal to the expected probability p. While we can modify the objective in Eq. 3.9 to adhere to this altered goal, here we propose simultaneously optimizing the **interval score** (or Winkler score) [Gneiting and Raftery, 2007, Winkler, 1972] for all expected probabilities $p \in (0, 1)$, and bring to light a proper scoring rule that has largely been neglected for the purpose of *learning quantiles*. While some previous works utilize the interval score to *evaluate* interval predictions [Bracher et al., 2021, Bowman et al., 2020, Askanazi et al., 2018, Maciejowska et al., 2016], to the best of our knowledge, no previous work has focused on simultaneously optimizing it and shown a thorough experimental evaluation as we provide in Section 6.3.

For a point (x, y), if we denote a $(1-\alpha)$ centered PI as $\hat{l}, \hat{u}$, i.e. $\hat{l} = \hat{Q}_{\frac{\alpha}{2}}(x)$ and $\hat{u} = \hat{Q}_{1-\frac{\alpha}{2}}(x)$, the interval score is defined as $S_\alpha(\hat{l}, \hat{u}; y) = (\hat{u} - \hat{l}) + \frac{2}{\alpha}(\hat{l} - y)\mathbb{I}\{y < \hat{l}\} + \frac{2}{\alpha}(y - \hat{u})\mathbb{I}\{y > \hat{u}\}$. We show in Appendix 3.A.4 that the minimum of the expectation of the interval score is attained at the true conditional quantiles, $\hat{l} = \mathbb{Q}_{\frac{\alpha}{2}}(\cdot)$, $\hat{u} = \mathbb{Q}_{1-\frac{\alpha}{2}}(\cdot)$. We train our quantile model for all centered intervals (and hence all quantile levels) simultaneously by setting our loss as $\mathbb{E}_{\alpha\sim\text{Unif}(0,1)}S_\alpha$.

3.3.4 Inducing adversarial group calibration with group batching

The calibration loss (Section 3.3.2) and the interval score (Section 3.3.3) optimize for the *average* calibration of quantiles and centered intervals, respectively. To get closer to individual calibration, one condition we can additionally require is *adversarial group calibration*. Since adversarial group calibration requires average calibration over any subset of non-zero measure over the domain, this is not fully observable with finite datasets D for all subset sizes. However, for any subset in D with enough datapoints, we can still estimate average calibration over the subset. Hence, we can apply our optimization objectives onto appropriately large subsets to induce adversarial group calibration.

In practice, this involves constructing subsets within the domain and taking gradient steps based on the loss over each subset. In naive implementations of stochastic gradient descent, a random batch is drawn *uniformly* from the training dataset D, and a gradient step is taken according to the loss over this batch. This is also the case in *SQR* [Tagasovska and Lopez-Paz, 2019]. The uniform draw of the batch will tend to preserve $\mathcal{F}_{\mathbf{X}}$ (the marginal distribution of $\mathbf{X}$), hence optimizing average calibration over this batch will only induce average calibration of the model.

Instead, deliberately grouping the datapoints based on input features, and then batching and taking gradient steps based on these batches, induces better adversarial group calibration. We find in our experiments that adversarial group calibration improves significantly with simple implementations of group batching, and in Section 3.4.2, we show through an ablation study that group batching can improve average calibration and adversarial group calibration of *SQR* as well.

To summarize, the main idea we introduce here with group batching is that, only taking uniform batches from the training set (thus only drawing batches which preserve $\mathcal{F}_{\mathbf{X}}$) can be detrimental when optimizing for calibration. Thus, additionally drawing batches based on deliberate groupings within the training set (thus, batches which do not preserve $\mathcal{F}_{\mathbf{X}}$) can help to induce a stronger notion of calibration (i.e. adversarial group calibration) in the model than average calibration. This concept is quite general and allows for variations in implementations when constructing the groups. In Appendix 3.C.3 we provide details on how we implemented group batching for our experiments and ablation study.

		SQR	*mPAIC*	***Interval***	***Cali***	***MAQR***
UCI	Concrete	$9.3 \pm 1.5(\underline{7.0} \pm 1.0)$	$6.2 \pm 0.5(14.2 \pm 0.8)$	$\mathbf{3.7} \pm 0.6(18.1 \pm 0.6)$	$5.6 \pm 0.8(17.3 \pm 1.5)$	$5.3 \pm 0.4(16.0 \pm 0.4)$
	Power	$2.6 \pm 0.4(13.4 \pm 0.2)$	$5.2 \pm 0.4(13.5 \pm 0.3)$	$2.2 \pm 0.4(21.0 \pm 1.0)$	$2.0 \pm 0.1(\underline{13.1} \pm 0.1)$	$\mathbf{1.6} \pm 0.3(19.9 \pm 0.2)$
	Wine	$4.2 \pm 0.2(29.5 \pm 0.4)$	$10.3 \pm 0.3(37.7 \pm 0.5)$	$5.0 \pm 0.8(41.4 \pm 2.5)$	$4.2 \pm 0.4(\underline{26.0} \pm 0.8)$	$\mathbf{2.7} \pm 0.2(39.3 \pm 0.5)$
	Yacht	$9.4 \pm 0.9(\underline{1.0} \pm 0.1)$	$10.8 \pm 2.3(2.6 \pm 0.4)$	$7.5 \pm 0.9(4.5 \pm 1.0)$	$8.3 \pm 0.6(2.0 \pm 0.4)$	$\mathbf{6.8} \pm 2.1(2.4 \pm 0.3)$
	Naval	$9.7 \pm 1.6(3.5 \pm 0.4)$	$3.1 \pm 0.5(63.0 \pm 1.8)$	$4.7 \pm 1.4(28.4 \pm 3.6)$	$5.9 \pm 0.7(3.0 \pm 0.2)$	$\mathbf{2.3} \pm 0.2(\underline{1.7} \pm 0.1)$
	Energy	$9.8 \pm 0.8(\underline{2.0} \pm 0.1)$	$10.4 \pm 0.5(4.3 \pm 0.2)$	$4.3 \pm 0.6(5.1 \pm 0.9)$	$5.8 \pm 0.4(3.6 \pm 0.3)$	$\mathbf{3.5} \pm 1.0(3.2 \pm 0.1)$
	Boston	$9.0 \pm 0.8(\underline{9.3} \pm 0.7)$	$8.7 \pm 1.3(12.3 \pm 0.7)$	$6.9 \pm 1.1(20.3 \pm 0.5)$	$8.5 \pm 1.5(10.9 \pm 0.6)$	$\mathbf{6.2} \pm 1.8(10.9 \pm 0.8)$
	Kin8nm	$4.4 \pm 0.1(\underline{11.4} \pm 0.2)$	$6.6 \pm 0.4(17.0 \pm 0.5)$	$2.9 \pm 0.4(16.9 \pm 0.5)$	$3.5 \pm 0.3(13.7 \pm 0.7)$	$\mathbf{1.8} \pm 0.4(17.1 \pm 0.1)$

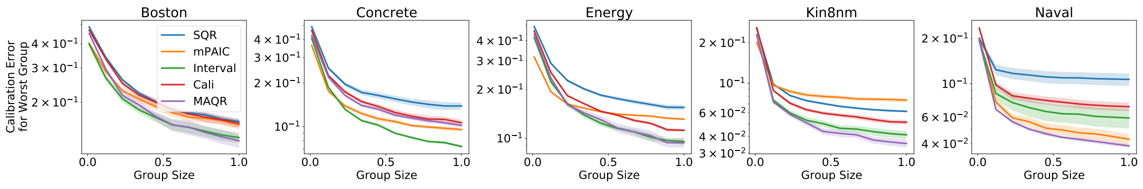


Figure 3.2: **UCI Experiments. (Top Table):** Average calibration (measured by ECE) and sharpness in parentheses. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined (all values multiplied by 100 for readability). **(Bottom Figure):** Adversarial group calibration for the first 5 UCI datasets (full set of results in Appendix 3.D.1). Group size refers to proportion of test dataset size.

		SQR	*mPAIC*	***Interval***	***Cali***
Fusion	aminor	$5.8 \pm 0.9(\underline{1.6} \pm 0.0)$	$13.5 \pm 0.0(5.3 \pm 0.0)$	$3.6 \pm 0.7(3.7 \pm 0.1)$	$\mathbf{2.9} \pm 0.2(2.2 \pm 0.0)$
	betan	$\mathbf{3.1} \pm 0.4(\underline{2.8} \pm 0.1)$	$9.2 \pm 0.4(5.5 \pm 0.1)$	$5.1 \pm 0.4(5.7 \pm 0.3)$	$3.4 \pm 0.5(3.3 \pm 0.2)$
	dssdenest	$4.4 \pm 0.5(\underline{7.2} \pm 0.2)$	$8.4 \pm 0.1(13.7 \pm 0.1)$	$\mathbf{2.9} \pm 0.3(13.3 \pm 0.4)$	$4.1 \pm 0.5(8.9 \pm 0.3)$
	ip	$3.2 \pm 0.3(\underline{2.4} \pm 0.1)$	$13.5 \pm 0.2(9.0 \pm 0.2)$	$4.4 \pm 0.3(5.4 \pm 0.1)$	$\mathbf{2.3} \pm 0.2(3.8 \pm 0.1)$
	kappa	$4.4 \pm 0.6(\underline{2.5} \pm 0.1)$	$14.8 \pm 0.5(9.0 \pm 0.4)$	$4.2 \pm 0.4(6.1 \pm 0.2)$	$\mathbf{3.6} \pm 0.2(3.4 \pm 0.1)$
	li	$4.4 \pm 0.6(\underline{1.0} \pm 0.1)$	$12.6 \pm 0.6(2.7 \pm 0.1)$	$3.7 \pm 0.6(2.2 \pm 0.1)$	$\mathbf{2.9} \pm 0.4(1.3 \pm 0.1)$
	R0	$5.3 \pm 0.8(\underline{3.3} \pm 0.1)$	$8.7 \pm 0.4(7.2 \pm 0.3)$	$\mathbf{3.4} \pm 0.2(6.2 \pm 0.1)$	$3.6 \pm 0.2(4.0 \pm 0.3)$
	tribot	$4.6 \pm 0.4(\underline{2.4} \pm 0.1)$	$15.1 \pm 0.7(8.8 \pm 0.7)$	$\mathbf{3.9} \pm 0.5(6.0 \pm 0.2)$	$4.6 \pm 0.5(3.0 \pm 0.2)$
	tritop	$5.6 \pm 0.7(\underline{2.7} \pm 0.1)$	$12.9 \pm 0.7(7.2 \pm 0.7)$	$3.7 \pm 0.8(6.5 \pm 0.4)$	$\mathbf{2.5} \pm 0.3(4.4 \pm 0.1)$
	volume	$5.8 \pm 1.2(\underline{0.9} \pm 0.0)$	$16.9 \pm 1.1(3.9 \pm 0.4)$	$3.6 \pm 0.3(2.0 \pm 0.1)$	$\mathbf{2.8} \pm 0.1(1.2 \pm 0.0)$

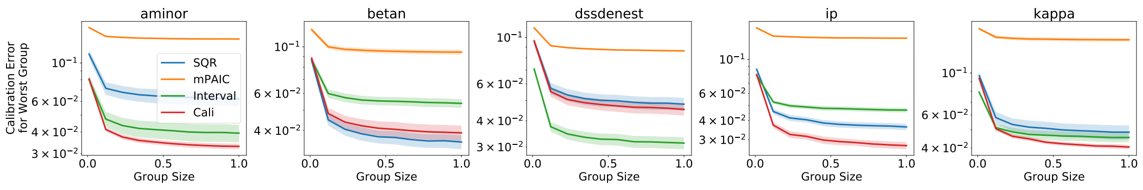


Figure 3.3: **Fusion Experiments. (Top Table):** Average calibration (measured by ECE) and sharpness in parentheses. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined (all values multiplied by 100 for readability). **(Bottom Figure):** Adversarial group calibration for the first 5 fusion datasets (full set of results in Appendix 3.D.2). Group size refers to proportion of test dataset size.

3.4 Experiments

We demonstrate the performances of our proposed methods on the standard 8 UCI datasets Asuncion and Newman [2007], and on a real-world problem in nuclear fusion. To assess the predictions, we use all metrics from Section 3.2.2 that can be estimated with a finite test dataset: 1) average calibration vs sharpness, 2) adversarial group calibration, 3) centered interval calibration, 4) check score, and 5) interval score. The results for the first two of these metrics are displayed in the main body of the paper, and the results for the other three metrics are shown in Appendix 3.D due to space restrictions. We describe how each of these evaluation metrics are calculated in Appendix 3.C.4.

	Cali		*SQR*	
	Random Batch	***Group Batch***	*Random Batch*	***Group Batch***
Boston	$9.7 \pm 1.3(\underline{10.2} \pm 0.7)$	$\mathbf{8.5} \pm 1.5(10.9 \pm 0.6)$	$10.9 \pm 1.0(\underline{8.8} \pm 1.1)$	$\mathbf{9.8} \pm 1.2(9.5 \pm 0.9)$
Concrete	$6.6 \pm 0.9(17.6 \pm 2.3)$	$\mathbf{5.6} \pm 0.8(\underline{17.3} \pm 1.5)$	$9.8 \pm 1.3(\underline{7.0} \pm 0.6)$	$\mathbf{7.1} \pm 0.9(8.5 \pm 0.6)$
Energy	$9.2 \pm 0.3(\underline{2.8} \pm 0.1)$	$\mathbf{5.8} \pm 0.4(3.6 \pm 0.3)$	$10.2 \pm 0.8(\underline{1.8} \pm 0.1)$	$\mathbf{6.9} \pm 1.1(2.4 \pm 0.2)$
Kin8nm	$\mathbf{3.4} \pm 0.3(\underline{13.7} \pm 0.4)$	$3.5 \pm 0.3(\underline{13.7} \pm 0.7)$	$4.7 \pm 0.3(\underline{11.1} \pm 0.1)$	$\mathbf{3.9} \pm 0.4(11.3 \pm 0.2)$
Wine	$4.4 \pm 0.5(\underline{25.6} \pm 0.8)$	$\mathbf{4.2} \pm 0.4(26.0 \pm 0.8)$	$4.5 \pm 0.4(29.7 \pm 0.5)$	$\mathbf{4.0} \pm 0.4(\underline{28.5} \pm 0.8)$
Yacht	$11.1 \pm 1.8(\underline{1.8} \pm 0.1)$	$\mathbf{8.3} \pm 0.6(2.0 \pm 0.4)$	$9.0 \pm 0.9(\underline{0.9} \pm 0.1)$	$\mathbf{8.9} \pm 0.9(2.3 \pm 0.2)$
Power	$\mathbf{1.7} \pm 0.2(14.2 \pm 0.3)$	$2.0 \pm 0.1(\underline{13.1} \pm 0.1)$	$\mathbf{2.5} \pm 0.3(14.0 \pm 0.5)$	$2.9 \pm 0.5(\underline{13.6} \pm 0.8)$
Naval	$2.8 \pm 0.2(\underline{12.1} \pm 3.1)$	$\mathbf{2.4} \pm 0.3(50.6 \pm 8.6)$	$8.6 \pm 1.6(\underline{3.6} \pm 0.1)$	$\mathbf{5.3} \pm 0.8(6.0 \pm 0.5)$

Figure 3.4: **Group Batching Ablation: Average Calibration and Sharpness.** The table shows mean ECE and sharpness (in parentheses) and their standard error with and without group batching. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined for *Cali* and *SQR* separately. All values have been multiplied by 100 for readability.

We provide comparisons against current state-of-the-art UQ methods for which computing the above metrics is tractable. *SQR* Tagasovska and Lopez-Paz [2019] is an NN model that optimizes the pinball loss for a batch of random quantile levels $p \sim \text{Unif}(0, 1)$. *mPAIC* Zhao et al. [2020] is an extension of probabilistic neural networks Lakshminarayanan et al. [2017], Nix and Weigend [1994] that optimizes a combination of the standard Gaussian NLL and a loss that induces individual calibration. While additional quantile and PI based UQ methods exist, they are mostly designed to output a single quantile level or interval coverage level, which makes measuring calibration extremely expensive (training up to 100 models separately). For these methods, we provide a comparison on a simplified task of predicting the 95% PI in Appendix 3.E.3. Lastly, we could also consider the recalibration algorithm of Kuleshov et al. [2018], which is not a standalone UQ method, but a post-hoc refinement step that can be applied on top of other methods. We discuss recalibration results in Appendix 3.E.4. All results report the mean and error across 5 trials. Error bars and shaded bands in plots indicate ± 1 standard error.

3.4.1 UCI and Fusion Experiments

UCI datasets: We evaluate 5 methods on the 8 UCI benchmark datasets: three proposed algorithms—*MAQR*, *Cali* (combined calibration loss), and *Interval* (interval score)—and 2 alternative algorithms—*SQR* and *mPAIC*. Appendix 3.C.1 includes more details on the experiment setup and hyperparameters.

Fusion datasets: We further evaluate the methods on a high-dimensional task from nuclear fusion: quantifying uncertainty in plasma dynamics. Recently, there has been increasing interest in applying machine learning to prediction and control tasks in the area of nuclear fusion Chung et al. [2020a], Char et al. [2019], Mehta et al. [2021a]. The plasma data in this paper was recorded from the DIII-D tokamak, a nuclear fusion device operated by General Atomics Luxon [2002]. Plasma dynamics during fusion reactions are highly stochastic and running live fusion experiments is costly. Hence, practitioners use this dataset to learn a dynamics model of the system for various purposes, such as controller learning to optimize reaction efficiency and stability Fu et al. [2020], Boyer et al. [2019a,b]. There are 10 scalar target signals to model in this dataset (full list in Appendix 3.C.2), each describing a particular aspect of the current state of plasma. For each signal, the input features are 468 dimensional. We do not apply *MAQR* on this dataset because of the high computational costs and statistical challenge associated with nonparametric density estimation in high dimensions. Appendix 3.C.2 provides more details on the dataset, experiment setup, and hyperparameters.

Analysis of Results: Figure 3.2 displays average calibration-sharpness for all 8 UCI datasets and adversarial group calibration for the first 5 UCI datasets (alphabetical order). *MAQR* produces the best average calibrated models on 7 of the 8 UCI datasets, and adversarial group calibration also indicates that *MAQR* tends to achieve the lowest calibration error across *any* random subgroup of *any* size with more than one point (6 out of 8 datasets). Excluding *MAQR*, *Interval* and *Cali* achieve competitive average calibration and adversarial group calibration on 7 out of 8 UCI datasets. Notably, *SQR* tends to produce the sharpest predictions across all datasets (often at the cost of worse calibration), which is not surprising following the discussion in Section 3.2.2. Lastly, we also note that *Interval* and *Cali* tend to be less brittle compared to *SQR* and *mPAIC*, which incur major failures in some cases (e.g. *mPAIC* in Kin8nm, *SQR* in Naval).

The fusion experiment results display a similar pattern (Figure 3.3). *Interval* and *Cali* achieve competitive average calibration and adversarial group calibration on 9 out of 10 fusion datasets, and *SQR* produces the sharpest UQ at the cost worse average and adversarial group calibration (except on betan). We also observe that *mPAIC* performs the poorest on the fusion datasets, with least calibrated and least sharp predictions. It is generally known that plasma dynamics display complex stochasticity Fu et al. [2020], Kowal et al. [2020], and hence we suspect *mPAIC*'s performance degrades significantly because it assumes a Gaussian output and is trained according to the Gaussian likelihood.

Only a subset of the plots and metrics are shown in the main paper due to space restrictions. Readers are encouraged to see the full set of results in Appendix 3.D, but we summarize the main findings here. The check score, interval score, and centered interval calibration in the tables in Appendix 3.D.1 also rank *MAQR*'s prediction as best on the UCI datasets. This is surprising since *SQR* and *Interval* train NNs of the same capacity to *explicitly minimize* the check and interval scores, respectively. This indicates the distribution predicted by *MAQR* is fundamentally different as it utilizes direct estimates of the conditional distribution, while the other methods all optimize a specific loss function. The centered interval calibration metric on the fusion datasets (Appendix 3.D.2) indicates that *Cali* and *Interval* both produce competitive centered PI's (9 out of 10 fusion datasets), suggesting that both methods have generally estimated the conditional quantiles better than the alternative algorithms.

3.4.2 Ablation Study: Effect of Group Batching

We provide an ablation study on the effect of group batching (experiment details in Appendix 3.F.1). Figure 3.4 displays how group batching affects average calibration-sharpness of *Cali* and *SQR* on all 8 UCI datasets. Average calibration improved on 6 out of 8 datasets for *Cali*, and on 7 out of 8 datasets on *SQR*. While sharpness tends to worsen with group batching, it's important to note that less sharp (i.e. more dispersed) predictions *are* desirable if there is high noise in the true data distribution. In Appendix 3.F.2, we show that group batching also improves adversarial group calibration, suggesting that the underlying data distribution truly does have high noise. This study indicates that deliberately taking batches during training that do not follow $\mathcal{F}_{\mathbf{X}}$ can improve UQ performance.

3.5 Discussion

In this paper, we have proposed four methods to improve quantile estimates for calibrated uncertainty quantification in regression. We assert that the pinball loss may not be an adequate objective to optimize in order to achieve calibration, and that our proposed methods provide better means of learning calibrated conditional quantiles. We have also extended the scope of regression models on which quantile-based UQ can be applied by developing a model-agnostic method. This can be of practical interest to users that have specific training infrastructure or preexisting regression procedures, since these procedures can be leveraged to quantify uncertainty without much additional overhead.

By providing an extensive evaluation with a suite of metrics, we also aim to show that there is not one single metric that captures full information about the performance of a UQ procedure. Rather, a holistic review of multiple types of metrics sheds light on different aspects of performance. This point has motivated us to additionally develop and make publicly available *Uncertainty Toolbox* Chung et al. [2021a], a Python library for evaluation, visualization and recalibration of predictive uncertainty, which includes the full suite of metrics discussed in this work.

In certain applications, users may also be concerned with modeling epistemic uncertainty. This is somewhat orthogonal to the goals of this paper; nevertheless, we provide a discussion on how bootstrapped ensembles of our methods can incorporate epistemic uncertainty in Appendix 3.G.

Appendices for Chapter 3

3.A Theoretical Results

3.A.1 Proof of Proposition 1

Proposition 1. *Consider a finite dataset D, the pinball loss ρ_τ (Eq. 3.1) and a quantile model $f : \mathcal{X} \times (0,1) \to \mathcal{Y}$ that is average calibrated on D, i.e. ECE$(D, f) = 0$. Then there always exists another quantile model $g : \mathcal{X} \times (0,1) \to \mathcal{Y}$, such that, for any quantile level $\tau \in (0,1)$, g has lower pinball loss than f on D, i.e. $\sum_{i=1}^{N} \rho_\tau(y_i, g_\tau(x_i)) < \sum_{i=1}^{N} \rho_\tau(y_i, f_\tau(x_i))$, but worse average calibration than f, i.e. ECE$(D, g) >$ ECE(D, f).*

Proof:

Denote $\{p_j\}_{j=1}^m$ as the set of quantiles used to compute $ECE(D, f)$. Since f is average calibrated on D, for any $\tau \in \{p_j\}_{j=1}^m$, $\hat{p}^{\text{obs}}_{avg}(D, \tau)$ of f is equal to τ, i.e. $\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{y_i \leq f_\tau(x_i)\} = \tau$.

Hence,

- $|\{(x_i, y_i) \mid y_i \leq f_\tau(x_i)\}| = \tau * N$,
- $|\{(x_i, y_i) \mid y_i > f_\tau(x_i)\}| = (1-\tau) * N$.

Denote $\{(x_i, y_i) \mid y_i \leq f_\tau(x_i)\}$ as S_f^{under} and $\{(x_i, y_i) \mid y_i > f_\tau(x_i)\}$ as S_f^{over}.

Let $\rho_\tau(D, f_\tau)$ be the pinball loss of f_τ on D, i.e. $\rho_\tau(D, f_\tau) = \sum_{i=1}^{N} \rho_\tau(y_i, f_\tau(x_i))$.

Then,

$$\rho_\tau(D, f_\tau) = \sum_{i=1}^{N} \rho_\tau(y_i, f_\tau(x_i)) = \sum_{(x_i,y_i)\in S_f^{\text{under}}} (f_\tau(x_i) - y_i)(1-\tau) + \sum_{(x_i,y_i)\in S_f^{\text{over}}} (f_\tau(x_i) - y_i)(-\tau).$$

We can construct another quantile model g, s.t. its prediction for τ, g_τ, is as follows: take any point $(x_k, y_k) \in S_f^{\text{over}}$ and set $g_\tau(x_k) = y_k - \frac{\tau}{2(1-\tau)}(f_\tau(x_k) - y_k)$. For all other points, $(x_i, y_i)_{i\neq k} \in D$, set $g_\tau(x_i) = f_\tau(x_i)$.

Since $y_k - g_\tau(x_k) = \frac{\tau}{2(1-\tau)}(f_\tau(x_k) - y_k) < 0$, we have that $y_k < g_\tau(x_k)$.

Therefore, $|\{(x_i, y_i) \mid y_i \leq g_\tau(x_i)\}| = (\tau * N) + 1$ and $|\{(x_i, y_i) \mid y_i > g_\tau(x_i)\}| = ((1-\tau) * N) - 1$.

Therefore, $|\hat{p}^{\text{obs}}_{avg}(D, \tau)$ of $g - \tau| > |\hat{p}^{\text{obs}}_{avg}(D, \tau)$ of $f - \tau|$, which implies that according to $\{p_j\}_{j=1}^m$, $ECE(D, g) > ECE(D, f)$.

We further consider the pinball loss of g_τ on D, $\rho_\tau(D, g_\tau)$.

$\rho_\tau(D, g_\tau) = \sum_{i=1}^{N} \rho_\tau(y_i, g_\tau(x_i)) = \rho_\tau(y_k, g_\tau(x_k)) + \sum_{i\in[N], i\neq k} \rho_\tau(y_i, f_\tau(x_i))$.

Note that the term, $\rho_\tau(y_k, g_\tau(x_k))$ satisfies:

$$\begin{aligned}
\rho_\tau(y_k, g_\tau(x_k)) &= (g_\tau(x_k) - y_k)(1-\tau) \\
&= \frac{\tau}{2(1-\tau)}(y_k - f_\tau(x_k))(1-\tau) \\
&= \frac{\tau}{2}(y_k - f_\tau(x_k)) \\
&< \tau(y_k - f_\tau(x_k)) \\
&= (-\tau)(f_\tau(x_k) - y_k) \\
&= \rho_\tau(y_k, f_\tau(x_k))
\end{aligned}$$

Therefore, $\rho_\tau(D, g_\tau) < \rho_\tau(D, f_\tau)$, i.e. the pinball loss of g_τ on D is less than the pinball loss of f_τ on D.

Note that for any $\tau \notin \{p_j\}_{j=1}^m$, we can follow the same steps to construct g_τ s.t. $|\hat{p}_{avg}^{\text{obs}}(D,\tau)$ of $g - \tau| > |\hat{p}_{avg}^{\text{obs}}(D,\tau)$ of $f - \tau|$, i.e. g_τ is more miscalibrated than f_τ. Therefore, for any quantile level $\tau \in (0,1)$, $\sum_{i=1}^N \rho_\tau(y_i, g_\tau(x_i)) < \sum_{i=1}^N \rho_\tau(y_i, f_\tau(x_i))$, but $\text{ECE}(D,g) > \text{ECE}(D,f)$.

$\square$

3.A.2 Proof of Proposition 2

Proposition 2. *For any quantile level* $p \in (0,1)$*, the true quantile function* $\mathbb{Q}_p$ *minimizes the calibration objective,* $\mathcal{C}(\hat{\mathbb{Q}}_p, p)$*. Further, on a finite dataset D, the empirical calibration objective,* $\mathcal{C}(D, \hat{\mathbb{Q}}_p, p)$*, is minimized by an average calibrated solution on D, i.e. when* $\hat{p}_{avg}^{obs}(D,p) = p$*.*

Proof:

Recall the calibration objective for a quantile level $p \in (0,1)$,

$$\begin{aligned}
\mathcal{C}(\hat{Q}_p, p) = \mathbb{I}\{\hat{p}_p < p\} * \mathbb{E}[Y - \hat{Q}_p | Y > \hat{Q}_p] * P(Y > \hat{Q}_p) \\
+ \mathbb{I}\{\hat{p}_p > p\} * \mathbb{E}[\hat{Q}_p - Y | \hat{Q}_p > Y] * P(\hat{Q}_p > Y), \text{where } \hat{p}_p = P(Y \leq \hat{Q}_p).
\end{aligned}$$

For the true quantile function $\mathbb{Q}_p$, $P(Y \leq \mathbb{Q}_p) = p$, thus achieves the minimum value of 0 for $\mathcal{C}(\hat{Q}_p, p)$, as the two non-negative terms of $\mathcal{C}(\hat{Q}_p, p)$ are 0.

Further, recall the empirical calibration objective,

$$\begin{aligned}
\mathcal{C}(D, \hat{Q}, p) = \mathbb{I}\{\hat{p}_{avg}^{\text{obs}} < p\} * \frac{1}{N}\sum_{i=1}^N \Big[(y_i - \hat{Q}_p(x_i))\mathbb{I}\{y_i > \hat{Q}_p(x_i)\}\Big] \\
+ \mathbb{I}\{\hat{p}_{avg}^{\text{obs}} > p\} * \frac{1}{N}\sum_{i=1}^N \Big[(\hat{Q}_p(x_i) - y_i)\mathbb{I}\{\hat{Q}_p(x_i) > y_i\}\Big]
\end{aligned}$$

An average calibrated solution on the dataset D satisfies $\hat{p}_{avg}^{\text{obs}} = p$, thus achieves the minimum value of 0 for $\mathcal{C}(D, \hat{Q}, p)$, as the two non-negative terms of $\mathcal{C}(D, \hat{Q}, p)$ are 0.

$\square$

3.A.3 Derivation of Gradients of Calibration Objective

Denote the CDF and PDF of the random variable Y as $\mathbb{F}_Y$ and f_Y.

- When $\hat{p}_p = P(Y \leq \hat{Q}_p) < p$:

$$\begin{aligned}
\mathcal{C}(\hat{Q}_p, p) &= \mathbb{E}[Y - \hat{Q}_p | Y > \hat{Q}_p] * P(Y > \hat{Q}_p) \\
&= \left(\mathbb{E}[Y|Y > \hat{Q}_p] - \hat{Q}_p\right) * P(Y > \hat{Q}_p) \\
&= \left(\frac{\int_{\hat{Q}_p}^{\infty} y f_Y(y) dy}{P(Y > \hat{Q}_p)} - \hat{Q}_p\right) * P(Y > \hat{Q}_p) \\
&= \left(\int_{\hat{Q}_p}^{\infty} y f_Y(y) dy\right) - \left(\hat{Q}_p * P(Y > \hat{Q}_p)\right) \\
&= \left(\int_{\hat{Q}_p}^{\infty} y f_Y(y) dy\right) - \left(\hat{Q}_p * (1 - F_Y(\hat{Q}_p))\right)
\end{aligned}$$

Note that,

$$\begin{aligned}
\frac{\partial\left(\int_{\hat{Q}_p}^{\infty} y f_Y(y) dy\right)}{\partial \hat{Q}_p} &= -\hat{Q}_p * f_Y(\hat{Q}_p) \\
\frac{\partial\left(\hat{Q}_p * (1 - F_Y(\hat{Q}_p)\right)}{\partial \hat{Q}_p} &= (1 - F_Y(\hat{Q}_p) + \hat{Q}_P * (-f_y(\hat{Q}_p))
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial \mathcal{C}(\hat{Q}_p, p)}{\partial \hat{Q}_p} &= \frac{\partial\left(\int_{\hat{Q}_p}^{\infty} y f_Y(y) dy\right)}{\partial \hat{Q}_p} - \frac{\partial\left(\hat{Q}_p * (1 - F_Y(\hat{Q}_p)\right)}{\partial \hat{Q}_p} \\
&= -\hat{Q}_p * f_Y(\hat{Q}_p) - (1 - F_Y(\hat{Q}_p) + \hat{Q}_P * f_y(\hat{Q}_p) \\
&= -(1 - F_Y(\hat{Q}_p) \\
&= -P(Y > \hat{Q}_p)
\end{aligned}$$

- When $\hat{p}_p = P(Y \leq \hat{Q}_p) > p$:

$$
\begin{aligned}
\mathcal{C}(\hat{Q}_p, p) &= \mathbb{E}[\hat{Q}_p - Y|\hat{Q}_p > Y] * P(\hat{Q}_p > Y) \\
&= \Big(\hat{Q}_p - \mathbb{E}[Y|\hat{Q}_p > Y]\Big) * P(\hat{Q}_p > Y) \\
&= \left(\hat{Q}_p - \frac{\int_{-\infty}^{\hat{Q}_p} y f_Y(y)dy}{P(\hat{Q}_p > Y)}\right) * P(\hat{Q}_p > Y) \\
&= \Big(\hat{Q}_p * P(\hat{Q}_p > Y)\Big) - \left(\int_{-\infty}^{\hat{Q}_p} y f_Y(y)dy\right) \\
&= \Big(\hat{Q}_p * F_Y(\hat{Q}_p)\Big) - \left(\int_{-\infty}^{\hat{Q}_p} y f_Y(y)dy\right)
\end{aligned}
$$

Note that,

$$
\begin{aligned}
\frac{\partial\left(\hat{Q}_p * F_Y(\hat{Q}_p)\right)}{\partial \hat{Q}_p} &= F_Y(\hat{Q}_p) + \hat{Q}_P * f_Y(\hat{Q}_p) \\
\frac{\partial\left(\int_{-\infty}^{\hat{Q}_p} y f_Y(y)dy\right)}{\partial \hat{Q}_p} &= \hat{Q}_p * f_Y(\hat{Q}_p)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\frac{\partial \mathcal{C}(\hat{Q}_p, p)}{\partial \hat{Q}_p} &= \frac{\partial\left(\hat{Q}_p * F_Y(\hat{Q}_p)\right)}{\partial \hat{Q}_p} - \frac{\partial\left(\int_{-\infty}^{\hat{Q}_p} y f_Y(y)dy\right)}{\partial \hat{Q}_p} \\
&= F_Y(\hat{Q}_p) + \hat{Q}_P * f_Y(\hat{Q}_p) - \hat{Q}_p * f_Y(\hat{Q}_p) \\
&= F_Y(\hat{Q}_p) \\
&= P(Y < \hat{Q}_p).
\end{aligned}
$$

□

3.A.4 Optimum of Interval Score

Following notation from Section 3.3.3, we denote $\hat{l} = \hat{Q}(x, \frac{\alpha}{2})$ and $\hat{u} = \hat{Q}(x, 1 - \frac{\alpha}{2})$, and we omit conditioning on x for clarity.

Assume $\hat{l} \leq \hat{u}$. Then,

$$
\begin{aligned}
&\mathbb{E}\left[S_\alpha(\hat{l}, \hat{u}; y)\right] \\
&= \int_{-\infty}^{\hat{l}} S_\alpha(\hat{l}, \hat{u}; y) d\mathcal{F}(y) + \int_{\hat{l}}^{\hat{u}} S_\alpha(\hat{l}, \hat{u}; y) d\mathcal{F}(y) + \int_{\hat{u}}^{\infty} S_\alpha(\hat{l}, \hat{u}; y) d\mathcal{F}(y) \\
&= (\hat{u} - \hat{l}) + \frac{2}{\alpha} \int_{-\infty}^{\hat{l}} (\hat{l} - y) d\mathcal{F}(y) + \frac{2}{\alpha} \int_{\hat{u}}^{\infty} (y - \hat{u}) d\mathcal{F}(y) \\
&\frac{\partial \mathbb{E}\left[S_\alpha(\hat{l}, \hat{u}; y)\right]}{\partial \hat{l}} = -1 + \frac{2}{\alpha} \int_{-\infty}^{\hat{l}} d\mathcal{F}(y) = -1 + \frac{2}{\alpha} \mathcal{F}(\hat{l}) \\
&\frac{\partial \mathbb{E}\left[S_\alpha(\hat{l}, \hat{u}; y)\right]}{\partial \hat{u}} = 1 - \frac{2}{\alpha} \int_{\hat{u}}^{\infty} d\mathcal{F}(y) = 1 - \frac{2}{\alpha}(1 - \mathcal{F}(\hat{u})).
\end{aligned}
$$

Setting $\frac{\partial \mathbb{E}\left[S_\alpha(\hat{l}, \hat{u}; y)\right]}{\partial \hat{l}}$ and $\frac{\partial \mathbb{E}\left[S_\alpha(\hat{l}, \hat{u}; y)\right]}{\partial \hat{u}}$ to zero reveals the interval loss minima at the respective true quantiles,

$$
\begin{aligned}
&\mathcal{F}(\hat{l}) = \frac{\alpha}{2} \text{ and } \mathcal{F}(\hat{u}) = 1 - \frac{\alpha}{2} \\
&\text{i.e. } \hat{l} = \mathbb{Q}(\cdot, \frac{\alpha}{2}) \text{ and } \hat{u} = \mathbb{Q}(\cdot, 1 - \frac{\alpha}{2})
\end{aligned}
$$

□

3.B Model Agnostic Quantile Regression

3.B.1 General Algorithm for Model Agnostic Quantile Regression

As stated in Section 3.3.1, Algorithm 1 is one implementation of a general model-agnostic quantile regression procedure, in which we take direct estimates of the target density and regress onto these estimates. This general framework is stated in Algorithm 3

Algorithm 3 General Algorithm for Model Agnostic Quantile Regression

```
1: **Input:** Train data $\{x_i, y_i\}_{i=1}^N$
2: Initialize $D \leftarrow \varnothing$
3: **for** $i = 1$ **to** $N$ **do**
4:     Select a set of quantile levels $\{p_k\}_{k=1}^m,\ p_k \in [0, 1]$
5:     $\hat{q}_{i,p_k} \leftarrow$ KDE estimate of $\mathbb{Q}(x_i, p_k)$, $k = 1, \ldots, m$
6:     $D \leftarrow D \cup \{x_i, p_k, \hat{q}_{i,p_k}\}_{k=1}^m$
7: **end for**
8: Use $D$ to fit a regression model $\hat{Q}$
       $\hat{Q} : (x_i, p_k) \mapsto \hat{q}_{i,p_k}, \quad k = 1, \ldots, m$
9: **Output:** $\hat{g}, \quad k = 1, \ldots, m$
```

Algorithm 1 implements the KDE step of Algorithm 3 (Line 5 of Algorithm 3) by using a uniform kernel over $\mathcal{X}$ (Line 2 of Algorithm 2) and $\mathcal{Y}$ (Lines 3,5,6 of Algorithm 2).

It should also be noted that many other conditional KDE methods can be used to construct the dataset D. We refer the reader to Holmes et al. [2007], Hyndman et al. [1996] for a more thorough treatment of methods in conditional KDE.

Lastly, this algorithm is model agnostic because *any* regression model can be used for $\hat{g}$ in Algorithm 3, and for $\hat{f}$ and $\hat{g}$ in Algorithm 1. In our specific implementation of Algorithm 1, we used a neural network for $\hat{g}$ to fit the quantile dataset D, but we can also use other models, such as a random forest or gradient-boosted trees. In particular, we have replaced $\hat{g}$ in Algorithm 1 with a gradient-boosted tree model and observed very similar UQ performance on the UCI datasets as reported in Section 3.4.1 (we omit numerical values because they are very similar and otherwise uninformative).

3.B.2 Algorithm Complexity

Lines 2 and 3 of Algorithm 2 requires calculating the distance between x_k and all other $x_i, 1 \leq i \leq N$, and ordering these distances to construct the empirical CDF. Let the distance calculation between a pair of points take constant C time. Ordering the distance requires sorting the N distances. Hence, Lines 5 and 6 takes $\mathcal{O}(N \log N)$ time.

If K points are in the set E_{k,d_N}, Lines 5, 6, 7 of Algorithm 2 are done for each of the K points. We consider the worst case when each set E_{k,d_N} contains all N points, which costs $\mathcal{O}(N)$.

The above two procedures are done for all N points (Line 4 of Algorithm 1), therefore the for loop from Lines 4 to 7 in Algorithm 1 requires $\mathcal{O}(N^2 \log N)$ time. This loop takes into account creating the dataset D for the quantile model. The rest of the algorithm constitutes fitting a regression model with this dataset D, which we do not analyze here.

We now discuss the space complexity. In Lines 5, 6, 7 of Algorithm 2, we only draw the quantiles at which a discontinuous step occurs in the constructed empirical CDF. For example, if we constructed an empirical CDF with three equally weighted points, we will only draw the quantiles $[1/3, 2/3, 1]$. Following this procedure, in the worst case, for each of the N points, we will construct a CDF with all N points, and hence draw N quantiles to append to the quantile model dataset D. If we consider the space complexity of the mean function and the quantile model to be constant, the algorithm requires $O(N^2)$ space in total.

3.C Details on Datasets and Setup of Main Experiments

3.C.1 UCI Experiment Details

Here, we provide details on the setup of the UCI experiments presented in Section 3.4.1.

For each of the 8 UCI datasets, we split 10% of the data into the test set, and we further split 20% of the remaining 90% of the data for the validation, resulting in a train/validation/test split of proportions $72\%, 18\%, 10\%$. For all tasks and all 8 datasets, the data was preprocessed by centering to zero mean and scaling to unit variance.

We used the same NN architecture across all methods: 2 layers of 64 hidden units with ReLU non-linearities. We used the same learning rate, $1e^{-3}$, and the same batch size, 64, for all methods. For all methods, training was stopped early if the validation loss did not decrease for more than 200 epochs, until a maximum of 10000 epochs. If training was stopped early, the final model was backtracked to the model with lowest validation loss.

IndvCal (individual calibration) has one hyperparameter, α, which balances the NLL loss and the individual calibration loss in the loss function. We 5-fold cross-validated α in [0.0, 1.0] in 20 equi-spaced intervals based on Pareto optimality in test set NLL and adversarial group calibration. If there were multiple α values that were Pareto optimal, we chose the value that had the best test set adversarial group calibration.

Cali (penalized calibration loss) has one hyperparameter, λ, which balances the calibration loss and sharpness penalty in the loss function. We tuned λ according to the same grid as above for α, based on the criterion of adversarial group calibration. Note that adversarial group calibration will not always favor lower values of λ as $\lambda = 0$ will only target average calibration, which, in the degenerate case, may converge to the marginal distribution of $\mathcal{F}_{\mathbf{Y}}$. This state will achieve very poor adversarial group calibration.

Group batching was applied to *Interval* (interval score) and *Cali* according to the implementation detailed in Appendix 3.C.3. During training, we alternated between "group batching epochs" and "regular batching epochs" (where batches are drawn uniformly from the training set), and the frequency of group batching epochs was a hyperparameter we tuned with cross-validation in [1, 2, 3, 5, 10, 30, 100], based on the criterion of adversarial group calibration.

MAQR has a two-step training process: we first learn a mean model, then construct a quantile dataset D, then regress onto this dataset with the quantile model. Both the mean model and the quantile model had the same NN architecture as mentioned above. The mean model was trained with the MSE loss according to the same, aforementioned training procedure. For each UCI dataset, we learned one mean model from the first seed, and re-used this mean model for all other seeds. Using this mean model, we then populated the quantile dataset according to the method outlined in Algorithm 1. Algorithm 1 requires one hyperparameter: the distance threshold in $\mathcal{X}$ space (d_N in Line 2 of Algorithm 2). We tuned this hyperparameter by setting the minimum distance required to include k number of points, on average, in constructing an empirical CDF at each training point. We tuned this parameter with cross-validation using the grid $k \in [10, 20, 30, 40, 50]$. The quantile model was trained according to the same training procedure but with one difference: the batch size was set to 1024 because the quantile dataset D could become very large due to many conditional quantile estimates at each training point.

All methods, for all datasets were repeated with 5 seeds: [0, 1, 2, 3, 4].

3.C.2 Fusion Experiment Details

We first describe the fusion dataset from Section 3.4.1, then the details of the fusion experiment set up.

The fusion dataset was recorded from the DIII-D tokamak in San Diego, CA, USA, and describes the dynamics of plasma during a nuclear fusion reaction within the tokamak. Consent and access to use this dataset was obtained via collaborations with the Princeton Plasma Physics Lab.

While the dataset in its raw format is a time-series of the state variables and action variables, for the purposes of a supervised learning problem to learn the dynamics of plasma, it has been re-structrued into a *(state, action, next state)* format. Therefore, the modeling task at hand is to learn the mapping *(state, action)* to *(next state)*, and the UQ task is then to learn the distribution over the next state given the current state and action.

There are 10 plasma state variables that we use both as the input state variables and the target variables. For the action variables, we use the power level of 8 neutral beams, which are a primary means of controlling plasma in a tokamak. These variables are described in Figure 3.5.

As input to the dynamics model, we model the current state as a 200 millisecond (ms) history window of the 10 state variables and 8 action variables, and we model the current action as a 200ms window into the future of the 8 action variables. The target variables are modeled as the change

	State Variables
aminor	Minor Radius
dssdenest	Line Averaged Electron Density
efsbetan	Normalized Beta
efsli	Internal Inductance
efsvolume	Plasma Volume
ip	Plasma Curent
kappa	Elongation
R0	Major Radius
tribot	Bottom Triangularity
tritop	Top Triangularity

	Action Variables
pinj_15l	Co-current Beam 1 Power
pinj_15r	Co-current Beam 2 Power
pinj_21l	Counter-current Beam 1 Power
pinj_21r	Counter-current Beam 2 Power
pinj_30l	Co-current Beam 3 Power
pinj_30r	Co-current Beam 4 Power
pinj_33l	Co-current Beam 5 Power
pinj_33r	Co-current Beam 6 Power

Figure 3.5: **State and Action Variables for Fusion Dataset.** 10 variables describe the current state of plasma, and the action space is 8 dimensional.

(or delta) of the state variables 200ms in the future. The state and action features are engineered according to the method used by Fu et al. [2020]. Each 200ms window is taken as one “frame”, the 200ms window is further divided into 2 “frames” of equal length (100ms each), as well as thirds. Then we calculate the mean, variance, and slope of each state and action variable for each of these frames, and collect these statistics as the features. Hence, each 200ms window for 1 variable is summarized into 18 features (3 statistics per frame, and 6 frames per 200ms window). Since there are 10 state variables and 8 action variables, and since the state window is 200ms long and the action window is 200ms long, the input is a 468 dimensional array (10 state variables + 8 previous action variables + 8 current action variables for a total of 26 input variables, and 18 features per variable). Once these features are created, we centered and scaled the inputs and targets to zero mean and unit variance.

There were a total of 100K training data points, and 10K validation and 10K test data points.

The training and hyperparameter tuning procedures were exactly the same as the UCI experiments (detailed above in Appendix 3.C.1), except for two differences: 1) we have increased the NN capacity to 3 hidden layers of 100 hidden units and 2) the batch size was set to 500.

The fusion experiment was likewise repeated 5 times with the seeds [0, 1, 2, 3, 4].

3.C.3 Group Batching Implementation Details

Group batching, as introduced in Section 3.3.4, is a general procedure in which deliberate subsets of the training data are constructed and batched from during train time. There is no “correct” method to form these subsets, because the main point is to simply *avoid drawing batches only from $\mathcal{F}_{\boldsymbol{X}}$*. One could consider thresholding the values of each dimension of the domain and discretizing the subsets according to the thresholds, and also taking unions of these discretizations to form new subsets.

This implementation can be computationally demanding, because for each threshold setting, one has to make sure to choose a subset with sufficiently many points, and iteratively increase or decrease the dimension thresholds if no subset with sufficiently many points can be found. This computational cost increases significantly with dimension.

Our implementation of group batching for all our experiments were as follows: we sort the datapoints according to a single dimension, then take consecutive sets of size equal to the batch size, and use these sets as the batches to take gradient steps over during an epoch. We repeat the above process by cycling through each dimension for sorting. While this process is very simple and inexpensive and only considers a single dimension in constructing the subsets, this group batching scheme has shown to be very effective in our experiments.

3.C.4 Calculation of Evaluation Metrics

To measure the calibration metrics (average, adversarial group, centered interval), we discretized the expected probabilities from 0.01 to 0.99 in 0.01 increments (i.e. 0.01, 0.02, . . . , 0.97, 0.98, 0.99) and calculated ECE according to this finite discretization.

To measure centered interval calibration, for each expected probability p, we predict centered $100 \times p\%$ PIs, and calculate $\hat{p}^{\text{obs}}_{avg}$ as the proportion of test points falling within the PI, i.e. $\hat{p}^{\text{obs}}_{avg}(p)$ here would be calculated as

$$\hat{p}^{\text{obs}}_{avg}(p) \text{ for centered intervals } = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\{\hat{Q}_{(0.5-p/2)}(x_i) \leq y_i \leq \hat{Q}_{(0.5+p/2)}(x_i)\}.$$

The procedure in which we measure adversarial group calibration is the following. For a given test set, we scale group size between 1% and 100% of the full test set size, in 10 equi-spaced intervals, and for each group size, we draw 20 random groups from the test set and record the worst calibration incurred across these 20 random groups. This is also the method used by Zhao et al. [2020] to measure adversarial group calibration.

Sharpness was measured as the mean width of the 95% centered PI (i.e. between $p = 0.025$ and 0.975).

The proper scoring rules (check score, interval score) were measured as the average of the score on the test set.

3.C.5 Discussion of Compute and Resources

All experiments were run on a single NVIDIA GeForce GTX 1080Ti GPU, with a Intel(R) Xeon(R) Silver 4110 CPU. The fusion datasets were the largest (100K training) and highest dimensional (468 dimensional) and took the longest to run: training one model to convergence took roughly $\sim$ 1 hour.

3.D Full Experimental Results

In this section, we provide all of the experimental results from Section 6.3, for all metrics: 1) average calibration - sharpness, 2) adversarial group calibration, 3) check score, 4) interval score, and 5) centered interval calibration

For average calibration and sharpness, we present the numeric tables again, along with a visualization which plots calibration and sharpness along the x and y axes.

3.D.1 Full UCI Experiment Results

		SQR	*mPAIC*	***Interval***	***Cali***	***MAQR***
UCI	Concrete	$9.3 \pm 1.5(\underline{7.0} \pm 1.0)$	$6.2 \pm 0.5(14.2 \pm 0.8)$	$\mathbf{3.7} \pm 0.6(18.1 \pm 0.6)$	$5.6 \pm 0.8(17.3 \pm 1.5)$	$5.3 \pm 0.4(16.0 \pm 0.4)$
	Power	$2.6 \pm 0.4(13.4 \pm 0.2)$	$5.2 \pm 0.4(13.5 \pm 0.3)$	$2.2 \pm 0.4(21.0 \pm 1.0)$	$2.0 \pm 0.1(\underline{13.1} \pm 0.1)$	$\mathbf{1.6} \pm 0.3(19.9 \pm 0.2)$
	Wine	$4.2 \pm 0.2(29.5 \pm 0.4)$	$10.3 \pm 0.3(37.7 \pm 0.5)$	$5.0 \pm 0.8(41.4 \pm 2.5)$	$4.2 \pm 0.4(\underline{26.0} \pm 0.8)$	$\mathbf{2.7} \pm 0.2(39.3 \pm 0.5)$
	Yacht	$9.4 \pm 0.9(\underline{1.0} \pm 0.1)$	$10.8 \pm 2.3(2.6 \pm 0.4)$	$7.5 \pm 0.9(4.5 \pm 1.0)$	$8.3 \pm 0.6(2.0 \pm 0.4)$	$\mathbf{6.8} \pm 2.1(2.4 \pm 0.3)$
	Naval	$9.7 \pm 1.6(3.5 \pm 0.4)$	$3.1 \pm 0.5(63.0 \pm 1.8)$	$4.7 \pm 1.4(28.4 \pm 3.6)$	$5.9 \pm 0.7(3.0 \pm 0.2)$	$\mathbf{2.3} \pm 0.2(\underline{1.7} \pm 0.1)$
	Energy	$9.8 \pm 0.8(\underline{2.0} \pm 0.1)$	$10.4 \pm 0.5(4.3 \pm 0.2)$	$4.3 \pm 0.6(5.1 \pm 0.9)$	$5.8 \pm 0.4(3.6 \pm 0.3)$	$\mathbf{3.5} \pm 1.0(3.2 \pm 0.1)$
	Boston	$9.0 \pm 0.8(\underline{9.3} \pm 0.7)$	$8.7 \pm 1.3(12.3 \pm 0.7)$	$6.9 \pm 1.1(20.3 \pm 0.5)$	$8.5 \pm 1.5(10.9 \pm 0.6)$	$\mathbf{6.2} \pm 1.8(10.9 \pm 0.8)$
	Kin8nm	$4.4 \pm 0.1(\underline{11.4} \pm 0.2)$	$6.6 \pm 0.4(17.0 \pm 0.5)$	$2.9 \pm 0.4(16.9 \pm 0.5)$	$3.5 \pm 0.3(13.7 \pm 0.7)$	$\mathbf{1.8} \pm 0.4(17.1 \pm 0.1)$

Figure 3.6: **UCI Average Calibration-Sharpness Table.** The table shows mean average calibration (measured by ECE) and sharpness in parentheses, along with ± 1 standard error. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined. All values have been multiplied by 100 for readability (same table as Figure 3.2 (Top), repeated here for completeness).

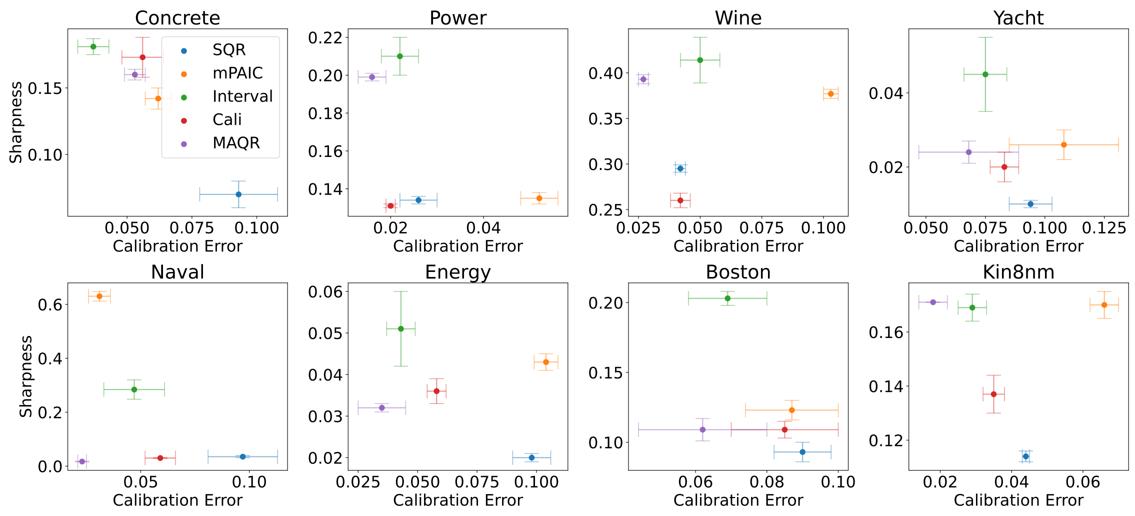


Figure 3.7: **UCI Average Calibration-Sharpness Plot.** Visualization of average calibration-sharpness from UCI experiments in Section 3.4.1

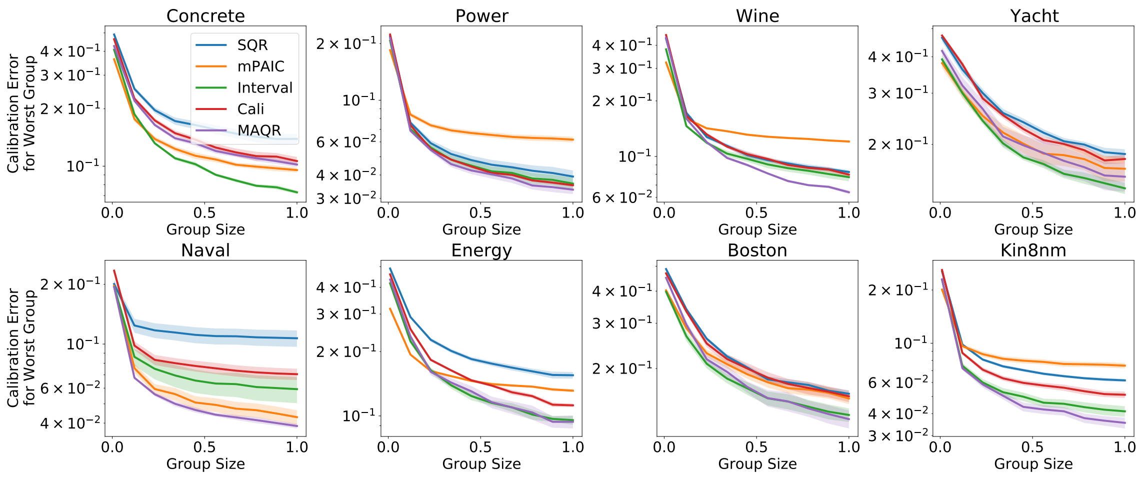


Figure 3.8: **UCI Adversarial Group Caibration.** The plot displays the worst calibration error incurred for any group of any given size. The mean is plotted along with ± 1 standard error in shades. Group size here refers to the proportion of the test dataset size (full results for Figure 3.2 (Bottom)).

	SQR	*mPAIC*	***Interval***	***Cali***	***MAQR***
concrete	$0.085(0.006)$	0.085 ± 0.005	0.086 ± 0.004	0.118 ± 0.006	$\mathbf{0.059 \pm 0.008}$
power	$\mathbf{0.057 \pm 0.001}$	0.070 ± 0.001	0.062 ± 0.001	0.064 ± 0.001	0.058 ± 0.001
wine	0.205 ± 0.008	0.219 ± 0.004	0.214 ± 0.006	0.210 ± 0.008	$\mathbf{0.191 \pm 0.003}$
yacht	0.012 ± 0.002	0.015 ± 0.002	0.018 ± 0.003	0.019 ± 0.004	$\mathbf{0.007 \pm 0.001}$
naval	0.070 ± 0.001	0.276 ± 0.004	0.066 ± 0.013	0.159 ± 0.029	$\mathbf{0.004 \pm 0.000}$
energy	0.014 ± 0.000	0.015 ± 0.001	0.017 ± 0.003	0.017 ± 0.002	$\mathbf{0.010 \pm 0.001}$
boston	0.088 ± 0.008	0.095 ± 0.008	0.094 ± 0.009	0.103 ± 0.013	$\mathbf{0.063 \pm 0.016}$
kin8nm	0.078 ± 0.001	0.104 ± 0.003	0.077 ± 0.001	0.096 ± 0.005	$\mathbf{0.070 \pm 0.001}$

Figure 3.9: **UCI Check Score.** Full check score results of UCI experiments from Section 3.4.1. Mean score across 5 trials is given, along with ± 1 standard error. The best mean has been bolded. *MAQR* tends to achieve the best check score, which is surprising given that *SQR* utilizes the same model class to optimize the check score directly.

	SQR	*mPAIC*	***Interval***	***Cali***	***MAQR***
concrete	2.038 ± 0.225	1.157 ± 0.069	0.943 ± 0.053	1.465 ± 0.086	$\mathbf{0.672 \pm 0.118}$
power	0.834 ± 0.022	0.917 ± 0.021	0.620 ± 0.010	0.699 ± 0.019	$\mathbf{0.592 \pm 0.009}$
wine	3.242 ± 0.166	3.168 ± 0.019	2.197 ± 0.045	2.498 ± 0.135	$\mathbf{2.052 \pm 0.052}$
yacht	0.314 ± 0.061	0.197 ± 0.036	0.190 ± 0.021	0.298 ± 0.063	$\mathbf{0.086 \pm 0.016}$
naval	0.097 ± 0.011	3.112 ± 0.053	0.620 ± 0.114	1.560 ± 0.268	$\mathbf{0.044 \pm 0.001}$
energy	0.290 ± 0.016	0.223 ± 0.017	0.182 ± 0.026	0.204 ± 0.018	$\mathbf{0.101 \pm 0.006}$
boston	1.833 ± 0.299	1.395 ± 0.176	1.010 ± 0.118	1.449 ± 0.259	$\mathbf{0.864 \pm 0.287}$
kin8nm	1.241 ± 0.041	1.347 ± 0.031	0.776 ± 0.017	1.121 ± 0.072	$\mathbf{0.691 \pm 0.015}$

Figure 3.10: **UCI Interval Score** Full interval score results of UCI experiments from Section 3.4.1. Mean score across 5 trials is given, along with ± 1 standard error. The best mean has been bolded. *MAQR* tends to achieve the best interval score, which is surprising given that *Interval* utilizes the same model class to optimize the interval score directly.

	SQR	*mPAIC*	***Interval***	***Cali***	***MAQR***
concrete	0.186 ± 0.031	0.089 ± 0.005	0.061 ± 0.008	0.096 ± 0.013	$\mathbf{0.059 \pm 0.020}$
power	0.045 ± 0.004	0.068 ± 0.008	0.023 ± 0.003	0.037 ± 0.002	$\mathbf{0.010 \pm 0.002}$
wine	0.053 ± 0.006	0.169 ± 0.008	0.079 ± 0.014	0.065 ± 0.007	$\mathbf{0.045 \pm 0.005}$
yacht	0.135 ± 0.009	0.100 ± 0.020	0.121 ± 0.005	0.129 ± 0.016	$\mathbf{0.085 \pm 0.024}$
naval	0.128 ± 0.031	0.039 ± 0.003	0.043 ± 0.014	0.110 ± 0.013	$\mathbf{0.012 \pm 0.002}$
energy	0.174 ± 0.011	0.163 ± 0.009	0.060 ± 0.010	0.090 ± 0.011	$\mathbf{0.052 \pm 0.018}$
boston	0.163 ± 0.020	$\mathbf{0.050 \pm 0.007}$	0.079 ± 0.015	0.138 ± 0.028	0.092 ± 0.041
kin8nm	0.070 ± 0.005	0.080 ± 0.002	0.048 ± 0.006	0.067 ± 0.005	$\mathbf{0.019 \pm 0.008}$

Figure 3.11: **UCI Centered Interval Calibration** Full centered interval calibration results of UCI experiments from Section 3.4.1. Mean score across 5 trials is given, along with ± 1 standard error. The best mean has been bolded. *MAQR* tends to achieve the best centered interval calibration.

3.D.2 Full Fusion Experiment Results

		SQR	*mPAIC*	***Interval***	***Cali***
Fusion	aminor	$5.8 \pm 0.9(\underline{1.6} \pm 0.0)$	$13.5 \pm 0.0(5.3 \pm 0.0)$	$3.6 \pm 0.7(3.7 \pm 0.1)$	$\mathbf{2.9} \pm 0.2(2.2 \pm 0.0)$
	betan	$\mathbf{3.1} \pm 0.4(\underline{2.8} \pm 0.1)$	$9.2 \pm 0.4(5.5 \pm 0.1)$	$5.1 \pm 0.4(5.7 \pm 0.3)$	$3.4 \pm 0.5(3.3 \pm 0.2)$
	dssdenest	$4.4 \pm 0.5(\underline{7.2} \pm 0.2)$	$8.4 \pm 0.1(13.7 \pm 0.1)$	$\mathbf{2.9} \pm 0.3(13.3 \pm 0.4)$	$4.1 \pm 0.5(8.9 \pm 0.3)$
	ip	$3.2 \pm 0.3(\underline{2.4} \pm 0.1)$	$13.5 \pm 0.2(9.0 \pm 0.2)$	$4.4 \pm 0.3(5.4 \pm 0.1)$	$\mathbf{2.3} \pm 0.2(3.8 \pm 0.1)$
	kappa	$4.4 \pm 0.6(\underline{2.5} \pm 0.1)$	$14.8 \pm 0.5(9.0 \pm 0.4)$	$4.2 \pm 0.4(6.1 \pm 0.2)$	$\mathbf{3.6} \pm 0.2(3.4 \pm 0.1)$
	li	$4.4 \pm 0.6(\underline{1.0} \pm 0.1)$	$12.6 \pm 0.6(2.7 \pm 0.1)$	$3.7 \pm 0.6(2.2 \pm 0.1)$	$\mathbf{2.9} \pm 0.4(1.3 \pm 0.1)$
	R0	$5.3 \pm 0.8(\underline{3.3} \pm 0.1)$	$8.7 \pm 0.4(7.2 \pm 0.3)$	$\mathbf{3.4} \pm 0.2(6.2 \pm 0.1)$	$3.6 \pm 0.2(4.0 \pm 0.3)$
	tribot	$4.6 \pm 0.4(\underline{2.4} \pm 0.1)$	$15.1 \pm 0.7(8.8 \pm 0.7)$	$\mathbf{3.9} \pm 0.5(6.0 \pm 0.2)$	$4.6 \pm 0.5(3.0 \pm 0.2)$
	tritop	$5.6 \pm 0.7(\underline{2.7} \pm 0.1)$	$12.9 \pm 0.7(7.2 \pm 0.7)$	$3.7 \pm 0.8(6.5 \pm 0.4)$	$\mathbf{2.5} \pm 0.3(4.4 \pm 0.1)$
	volume	$5.8 \pm 1.2(\underline{0.9} \pm 0.0)$	$16.9 \pm 1.1(3.9 \pm 0.4)$	$3.6 \pm 0.3(2.0 \pm 0.1)$	$\mathbf{2.8} \pm 0.1(1.2 \pm 0.0)$

Figure 3.12: **Fusion Average Calibration-Sharpness Table.** The table shows mean average calibration (measured by ECE) and sharpness in parentheses, along with ± 1 standard error. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined. All values have been multiplied by 100 for readability. *Cali* tends to achieve the best average calibration, while *SQR* achieves the sharpest predictions (same table as Figure 3.3 (Top), repeated here for completeness)

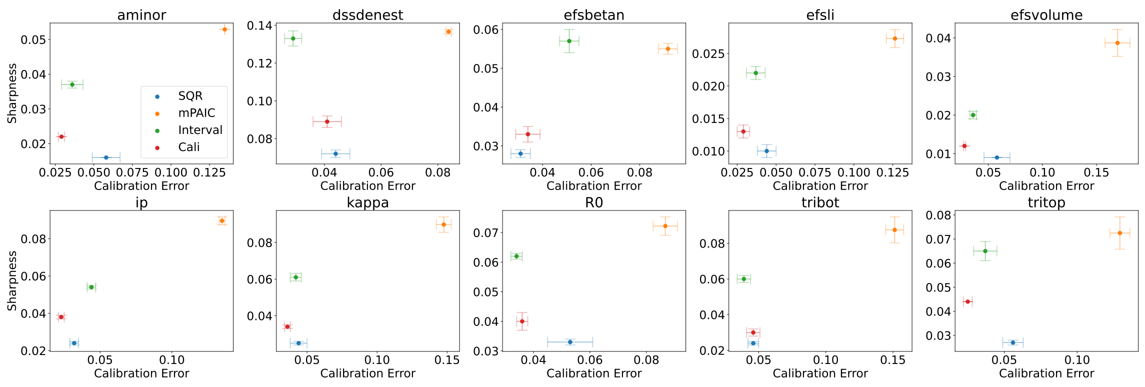


Figure 3.13: **Fusion Average Calibration-Sharpness Plot.** Visualization of average calibration-sharpness from fusion experiments from Section 3.4.1

.

3.E Additional Experiments

3.E.1 Details of Synthetic Experiment (Figure 3.1)

Dataset: The synthetic dataset is based on the Boston UCI dataset. A NN model, μ_θ, was trained on the train split of the Boston dataset to predict the mean by optimizing the MSE loss (same architecture and training details as described in Appendix 3.C.1). Afterwards, all points, $\{(x_i, y_i)\}_{i=1}^N$, in the full Boston dataset were re-labelled with the prediction of the mean model, $(x_i, \mu_\theta(x_i))$. Then, uniform noise, ϵ_i was added to these re-labelled mean values to create the observations, $\tilde{y}_i$, i.e. $\tilde{y}_i = \mu_\theta(x_i) + \epsilon_i$. The uniform noise was 0 mean, with width of the support equal to 5% of the

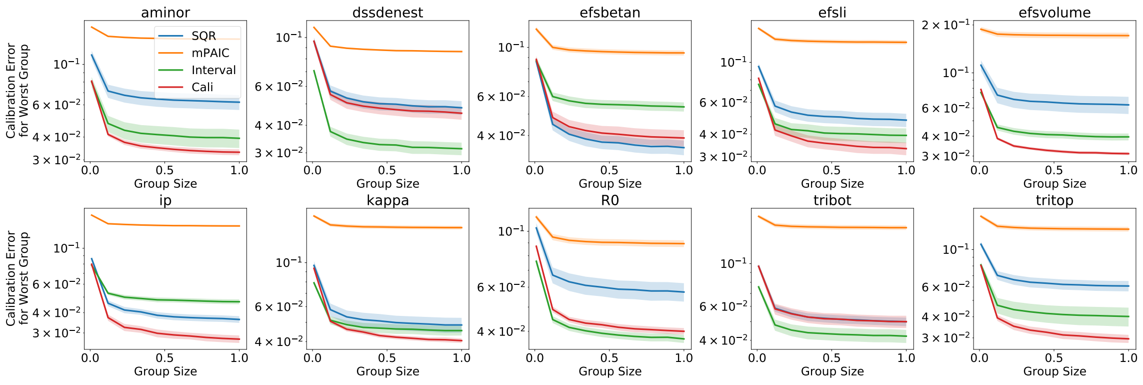


Figure 3.14: **Fusion Adversarial Group Calibration** Full fusion results of Figure 3.3 (Bottom). *Cali* and *Interval* tend to achieve the best calibration for any group of any size in the test set.

	SQR	*mPAIC*	***Interval***	***Cali***
aminor	$\mathbf{0.087 \pm 0.000}$	0.182 ± 0.000	0.097 ± 0.002	0.092 ± 0.002
dssdenest	$\mathbf{0.179 \pm 0.003}$	0.273 ± 0.003	0.180 ± 0.001	0.184 ± 0.002
betan	$\mathbf{0.146 \pm 0.001}$	0.253 ± 0.005	0.153 ± 0.002	0.150 ± 0.001
li	$\mathbf{0.097 \pm 0.001}$	0.166 ± 0.003	0.102 ± 0.001	0.104 ± 0.001
volume	$\mathbf{0.051 \pm 0.001}$	0.107 ± 0.007	0.053 ± 0.001	0.052 ± 0.001
ip	$\mathbf{0.068 \pm 0.000}$	0.199 ± 0.011	0.077 ± 0.002	0.076 ± 0.001
kappa	$\mathbf{0.072 \pm 0.001}$	0.150 ± 0.004	0.079 ± 0.002	0.078 ± 0.001
R0	$\mathbf{0.120 \pm 0.001}$	0.208 ± 0.002	0.126 ± 0.002	0.130 ± 0.005
tribot	$\mathbf{0.084 \pm 0.001}$	0.184 ± 0.020	0.092 ± 0.001	0.096 ± 0.005
tritop	$\mathbf{0.102 \pm 0.001}$	0.200 ± 0.018	0.107 ± 0.004	0.107 ± 0.002

Figure 3.15: **Fusion Check Score** Check score results from fusion experiments in Section 3.4.1. Mean score across 5 trials is given, along with ± 1 standard error. The best mean has been bolded. *SQR* achieves the best check score.

range of the mean values, i.e $\epsilon_i \sim \text{Unif}[-0.025 * (\max_i y_i - \min_i y_i), 0.025 * (\max_i y_i - \min_i y_i)]$. Thus synthetic dataset is $\{(x_i, \tilde{y}_i)\}_{i=1}^N$.

Procedure: The training procedure follows exactly that of the main experiments, which is described in detail in Appendix 3.C.1. The only difference is that the models were trained for the full 1000 epochs, instead of halting training according to the validation loss.

3.E.2 Regularizing the Pinball Loss

At first glance, regularization may appear to be the answer to what seems like an overfitting problem with the pinball loss. In fact, regularization has shown to be effective in the "single quantile learning setting", where the quantile level p is fixed and the p^{th} conditional quantile is learned from data [Steinwart and Christmann, 2011, Takeuchi et al., 2006]. This setting is concerned with learning $\mathbb{Q}(x)$ for a *given* p, which is different from the setting of this work, which considers learning a single model $\mathbb{Q}_p(x)$, which takes both x and p as input and outputs the full predictive distribution by specifying conditional quantiles predictions for all quantiles levels.

In this section, we empirically demonstrate the effect of applying regularization when minimizing

	SQR	*mPAIC*	***Interval***	***Cali***
aminor	1.181 ± 0.007	3.225 ± 0.000	$\mathbf{1.090 \pm 0.017}$	1.207 ± 0.035
dssdenest	2.387 ± 0.060	4.352 ± 0.109	$\mathbf{1.995 \pm 0.011}$	2.369 ± 0.054
betan	1.970 ± 0.013	4.301 ± 0.124	$\mathbf{1.725 \pm 0.021}$	2.002 ± 0.046
li	1.354 ± 0.018	2.923 ± 0.088	$\mathbf{1.146 \pm 0.010}$	1.410 ± 0.015
volume	0.711 ± 0.023	2.036 ± 0.154	$\mathbf{0.602 \pm 0.012}$	0.697 ± 0.026
ip	0.906 ± 0.005	3.203 ± 0.295	$\mathbf{0.845 \pm 0.017}$	0.978 ± 0.026
kappa	1.010 ± 0.010	2.639 ± 0.101	$\mathbf{0.891 \pm 0.015}$	1.066 ± 0.020
R0	1.599 ± 0.011	3.310 ± 0.064	$\mathbf{1.414 \pm 0.012}$	1.709 ± 0.057
tribot	1.211 ± 0.009	3.185 ± 0.242	$\mathbf{1.074 \pm 0.012}$	1.421 ± 0.115
tritop	1.481 ± 0.022	3.564 ± 0.351	$\mathbf{1.232 \pm 0.034}$	1.444 ± 0.031

Figure 3.16: **Fusion Interval Score** Interval score results from fusion experiments in Section 3.4.1. Mean score across 5 trials is given, along with ± 1 standard error. The best mean has been bolded. *Interval* achieves the best interval score.

	SQR	*mPAIC*	***Interval***	***Cali***
aminor	0.081 ± 0.006	0.268 ± 0.000	$\mathbf{0.055 \pm 0.007}$	0.058 ± 0.004
dssdenest	0.072 ± 0.003	0.162 ± 0.003	$\mathbf{0.054 \pm 0.006}$	0.071 ± 0.008
betan	$\mathbf{0.051 \pm 0.001}$	0.160 ± 0.012	0.087 ± 0.009	0.056 ± 0.007
li	0.071 ± 0.011	0.214 ± 0.017	0.069 ± 0.013	$\mathbf{0.043 \pm 0.003}$
volume	0.073 ± 0.007	0.315 ± 0.022	0.063 ± 0.007	$\mathbf{0.056 \pm 0.002}$
ip	0.051 ± 0.003	0.227 ± 0.020	0.077 ± 0.015	$\mathbf{0.036 \pm 0.003}$
kappa	0.069 ± 0.008	0.254 ± 0.014	0.078 ± 0.011	$\mathbf{0.053 \pm 0.004}$
R0	0.068 ± 0.007	0.168 ± 0.009	$\mathbf{0.058 \pm 0.007}$	0.066 ± 0.004
tribot	0.087 ± 0.006	0.227 ± 0.016	$\mathbf{0.074 \pm 0.008}$	0.080 ± 0.007
tritop	0.077 ± 0.006	0.231 ± 0.017	0.058 ± 0.015	$\mathbf{0.037 \pm 0.004}$

Figure 3.17: **Fusion Centered Interval Calibration** Full centered interval calibration results from fusion experiments in Section 3.4.1. Mean score across 5 trials is given, along with ± 1 standard error. The best mean has been bolded. *Cali* and *Interval* achieves the best centered interval calibration in 9 out of 10 datasets.

the pinball loss to learn $\mathbb{Q}_p(x)$, and show how regularization does not adequately address the issues with the pinball loss.

With the *SQR* method (which optimizes the pinball loss simultaneously for random batches of quantile levels), we applied L1, L2, and dropout, by cross-validating the regularization coefficients in $\{2^i, i \in \text{np.linspace}(-13, 1, 40)\}$ for L1 and L2, and dropout probability p in $\{2^i, i \in \text{np.linspace}(-13, -1, 40)\}$, for the pinball loss criterion (i.e. cross-validate among 40 different regularization coefficients, between 2^{-13} and 2^{-1} on the log scale). We show the best calibrated regularization result, across all regularization methods and cross-validation (table in same format as Figure 2 in paper).

Counter-intuitively, regularization tends to further the bias towards sharpness, and upon reflection, this may not be surprising because the range of quantile predictions shrinks: given a quantile model $f : \mathcal{X} \times \mathcal{P} \to \mathcal{Y}$, where $\mathcal{P}$ is the space of quantile levels in $(0, 1)$, regularization affects the

	SQR	*SQR w/Reg*
concrete	$\mathbf{9.3} \pm 1.5(7.0 \pm 1.0)$	$10.9 \pm 1.0(\underline{6.9} \pm 0.6)$
power	$2.6 \pm 0.4(\underline{13.4} \pm 0.2)$	$\mathbf{1.0} \pm 0.1(14.8 \pm 0.1)$
wine	$\mathbf{4.2} \pm 0.2(29.5 \pm 0.4)$	$5.1 \pm 0.8(\underline{26.0} \pm 0.5)$
yacht	$\mathbf{9.4} \pm 0.9(\underline{1.0} \pm 0.1)$	$12.3 \pm 2.6(\underline{1.0} \pm 0.1)$
naval	$\mathbf{9.7} \pm 1.6(\underline{3.5} \pm 0.4)$	$11.5 \pm 2.8(\underline{3.5} \pm 0.3)$
energy	$9.8 \pm 0.8(2.0 \pm 0.1)$	$\mathbf{9.4} \pm 1.3(\underline{1.9} \pm 0.2)$
boston	$\mathbf{9.0} \pm 0.8(9.3 \pm 0.7)$	$11.6 \pm 1.1(\underline{8.6} \pm 0.8)$
kin8nm	$4.4 \pm 0.1(11.4 \pm 0.2)$	$\mathbf{3.5} \pm 0.3(\underline{11.2} \pm 0.2)$

Figure 3.18: **Applying Regularization with SQR: Average Calibration and Sharpness.** The table shows *SQR*'s mean ECE and sharpness (in parentheses) and their standard error with and without regularization. Among the 3 regularization methods (L1, L2, dropout), the method resulting in the best calibration is shown, for each dataset. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined. All values have been multiplied by 100 for readability.

smoothness not only in $\mathcal{X}$, but also in $\mathcal{P}$, hence for any fixed $x \in \mathcal{X}$, the range in predictions for different quantile level inputs also shrinks.

3.E.3 Comparison on 95% Prediction Interval Task

In Section 6.3, we have presented an experiment that is targeted at evaluating the *full predictive distribution* on the 8 UCI datasets.

In this section, we present an experiment that is targeted at constructing a 95% *centered prediction interval (PI)* on the same UCI datasets, for the purposes of comparing against other quantile-based algorithms that are designed to output only a single quantile level. This experiment setup is exactly the same as the prediction intervals experiments in Section 4.1 of the work by Tagasovska and Lopez-Paz [2019], and we follow the exact same experiment procedure for a direct comparison against their reported results.

The comparison algorithms here are:

- Dropout [Gal and Ghahramani, 2016]: a NN that uses a dropout layer during testing for multiple predictions
- QualityDriven [Pearce et al., 2018b]: a NN that optimizes a Binomial likelihood approximation as a surrogate loss for calibration and sharpness
- GradientBoosting [Meinshausen, 2006]: a decision tree based model that optimizes the pinball loss with gradient boosting
- QuantileForest [Meinshausen, 2006]: a decision tree based model that predicts the quantiles based on the trained output of a random forest
- ConditionalGaussian [Lakshminarayanan et al., 2017]: a probabilistic NN that optimizes the Gaussian NLL to output the parameters of a Gaussian distribution

We show the performance of *MAQR* to represent our proposed methods in this experiment, since *MAQR* performs the best on the full predictive distribution evaluations in the UCI experiments of Section 3.4.1. We also omit the results of *SQR* Tagasovska and Lopez-Paz [2019] in this experiment since we perform a full evaluation comparison with *SQR* in Section 3.4.1 and Appendix 3.D.1, 3.D.2, and the purpose here is to compare our proposed method against the additional baselines.

In this experiment, since we only output two quantile levels $(0.025, 0.975)$ to construct a single 95% prediction interval, we do not assess calibration (which requires the predictions for all quantile levels) and assess only the observed proportion of test points within the PI (also referred to as "prediction interval coverage probability" or "PICP") and sharpness represented by the width of the PI (also referred to as "mean prediction interval width" or "MPIW"). We refer the reader to Tagasovska and Lopez-Paz [2019] for exact details on the experiment setup and the hyperparameters tuned. For *MAQR*, we used the exact same NN architecture (1 layer of 64 hidden units, ReLU non-linearities) as the NN based baselines (*Dropout, QualityDriven, ConditionalGaussian*) and the same training procedure as detailed in Tagasovska and Lopez-Paz [2019]. The hyperparameters tuned for *MAQR* are detailed in Appendix 3.C.1.

	Dropout	*QualityDriven*	*GradientBoostingQR*
concrete	none	none	0.93 ± 0.00 (0.71 ± 0.00)
power	0.94 ± 0.00 (0.37 ± 0.00)	0.93 ± 0.02 (0.34 ± 0.19)	none
wine	none	none	none
yacht	0.97 ± 0.03 (0.10 ± 0.01)	0.92 ± 0.05 (0.04 ± 0.01)	0.95 ± 0.02 (0.79 ± 0.01)
naval	0.96 ± 0.01 (0.23 ± 0.00)	0.94 ± 0.02 (0.21 ± 0.11)	none
energy	0.91 ± 0.04 (0.17 ± 0.01)	0.91 ± 0.04 (0.10 ± 0.05)	none
boston	none	none	0.89 ± 0.00 (0.75 ± 0.00)
kin8nm	none	0.96 ± 0.00 (0.84 ± 0.00)	none

	QuantileForest	*ConditionalGaussian*	***MAQR***
concrete	0.96 ± 0.01 (0.37 ± 0.02)	0.94 ± 0.03 (0.32 ± 0.09)	0.93 ± 0.01 (0.26 ± 0.01)
power	0.94 ± 0.01 (0.18 ± 0.00)	0.94 ± 0.01 (0.18 ± 0.00)	0.95 ± 0.01 (0.28 ± 0.03)
wine	none	0.94 ± 0.02 (0.49 ± 0.03)	0.95 ± 0.02 (0.56 ± 0.06)
yacht	0.97 ± 0.04 (0.28 ± 0.11)	0.93 ± 0.06 (0.03 ± 0.01)	0.92 ± 0.03 (0.03 ± 0.01)
naval	0.92 ± 0.01 (0.22 ± 0.00)	0.96 ± 0.01 (0.15 ± 0.25)	0.94 ± 0.01 (0.03 ± 0.00)
energy	0.95 ± 0.02 (0.15 ± 0.01)	0.94 ± 0.03 (0.12 ± 0.18)	0.94 ± 0.02 (0.05 ± 0.01)
boston	0.95 ± 0.03 (0.37 ± 0.02)	0.94 ± 0.03 (0.55 ± 0.20)	0.95 ± 0.02 (0.34 ± 0.09)
kin8nm	none	0.93 ± 0.01 (0.20 ± 0.01)	0.93 ± 0.00 (0.20 ± 0.01)

Figure 3.19: **95% PI PICP and MPIW** The test average and standard deviation PICP of models with validation PICP in $[92.5\%, 97.5\%]$ is shown, and the test average and standard deviation MPIW is shown in parantheses. "none" indicates the method could not find a model with validation PICP in $[92.5\%, 97.5\%]$.

Summarizing the experimental result in Figure 3.19:

- Our proposed method (*MAQR*) is capable of consistently producing PIs that have the correct desired coverage (0.95), even in cases when some of the baseline algorithms are not able to.
- Even when the baseline algorithms do produce PIs with correct desired coverage, *MAQR* is able to produce PI's that are much sharper (e.g. mean PI width for naval dataset is an order of magnitude sharper than all other baselines)

On this limited output and evaluation setting, our proposed method is still competitive in its performance. However, it should be noted that this experiment tells *only one facet* of overall UQ quality. Inspecting and evaluating the full predictive uncertainty is necessary for a more thorough evaluation of UQ quality, as done in our main experiments in Section 6.3.

3.E.4 Discussion on Recalibration [Kuleshov et al., 2018]

The recalibration algorithm by Kuleshov et al. [2018] utilizes isotonic regression with a validation set to fine-tune predictive uncertainties from a UQ model. We have applied this recalibration as a post-processing step on the methods presented in Section 6.3, and the empirical results indicate that its effect on overall improvement in UQ quality is inconclusive. Here, we show its effect on one of our methods, *Interval*, because we observe the same pattern across all the methods, including the baseline algorithms.

Recalibration tends to improve average calibration. This is expected, because recalibration specifically targets average calibration. However, it does so at a cost in sharpness. This is evident in the recalibrated output moving upper left in the average calibration-sharpness plot in Figure 3.21.

However, there is little to no improvement in adversarial group calibration (except for the Naval dataset) as shown in Figure 3.22, which seems to indicate that the improvement in average calibration was not meaningful (i.e. the recalibrated result is not closer to individual calibration). This is also the observation made by Zhao et al. [2020].

At the same time, the proper scoring rules improved on average (Figures 3.23, 3.24), but interval calibration tended to worsen (Figure 3.25).

Based on these metrics, it is difficult to conclude on whether recalibration by Kuleshov et al. [2018] is a beneficial step or not for overall UQ quality. If a practitioner is primarily concerned with average calibration, the results indicate that recalibration *is* a beneficial step, but if converging to the true conditional distribution is the primary objective, recalibration does not seem to be a robust remedy.

	Interval	***Interval Recalibrated***
concrete	$3.7 \pm 0.6(\underline{18.1} \pm 0.6)$	$\mathbf{3.1} \pm 0.4(26.3 \pm 1.8)$
power	$2.2 \pm 0.4(21.0 \pm 1.0)$	$\mathbf{1.0} \pm 0.1(\underline{20.6} \pm 0.5)$
wine	$5.0 \pm 0.8(\underline{41.4} \pm 2.5)$	$\mathbf{2.6} \pm 0.2(50.5 \pm 1.8)$
yacht	$7.5 \pm 0.9(\underline{4.5} \pm 1.0)$	$\mathbf{4.7} \pm 0.5(5.6 \pm 1.1)$
naval	$4.7 \pm 1.4(28.4 \pm 3.6)$	$\mathbf{1.3} \pm 0.1(\underline{21.9} \pm 3.3)$
energy	$4.3 \pm 0.6(\underline{5.1} \pm 0.9)$	$\mathbf{3.8} \pm 0.6(6.9 \pm 1.2)$
boston	$6.9 \pm 1.1(\underline{20.3} \pm 0.5)$	$\mathbf{5.4} \pm 0.9(30.8 \pm 2.6)$
kin8nm	$2.9 \pm 0.4(\underline{16.9} \pm 0.5)$	$\mathbf{1.1} \pm 0.1(20.6 \pm 0.3)$

Figure 3.20: **UCI Average Calibration-Sharpness Table with Recalibration.** Recalibration tends to trade off sharpness for average calibration. Better mean average calibration has been bolded, and better sharpness has been underlined. All values have been multiplied by 100 for readability.

3.F Ablation Study

3.F.1 Ablation Study Details

The ablation study from Section 3.4.2 (with full results in Appendix 3.F.2) investigates the effect of group batching on two methods: *Cali* (combined calibration loss, one of our proposed methods) and *SQR* (a baseline method).

For each method, we applied group batching by tuning the group batching frequency hyperparameter with cross-validation according to the details in Appendix 3.C.1.

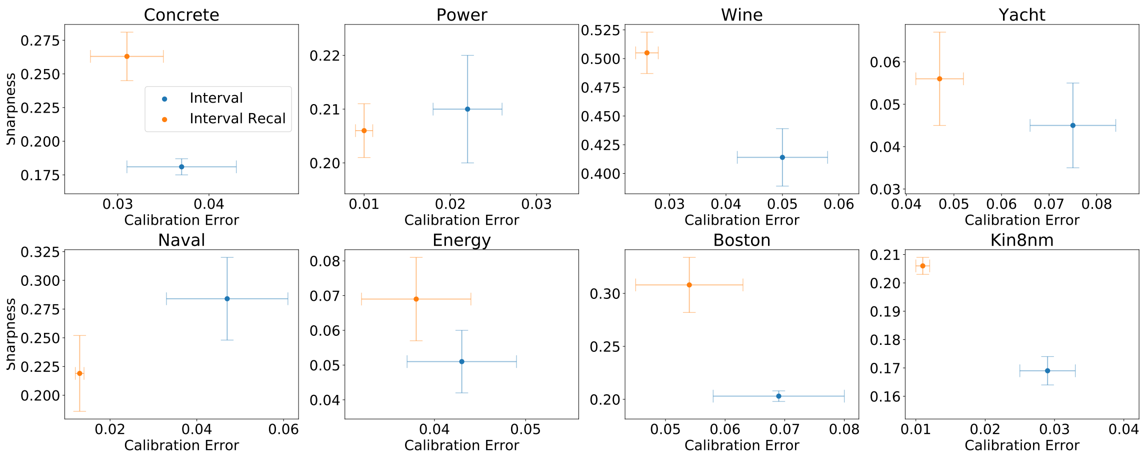


Figure 3.21: **UCI Average Calibration-Sharpness Plots with Recalibration.** Recalibration tends to trade off sharpness for average calibration. This is evident as the recalibrated predictions move *upper left*.

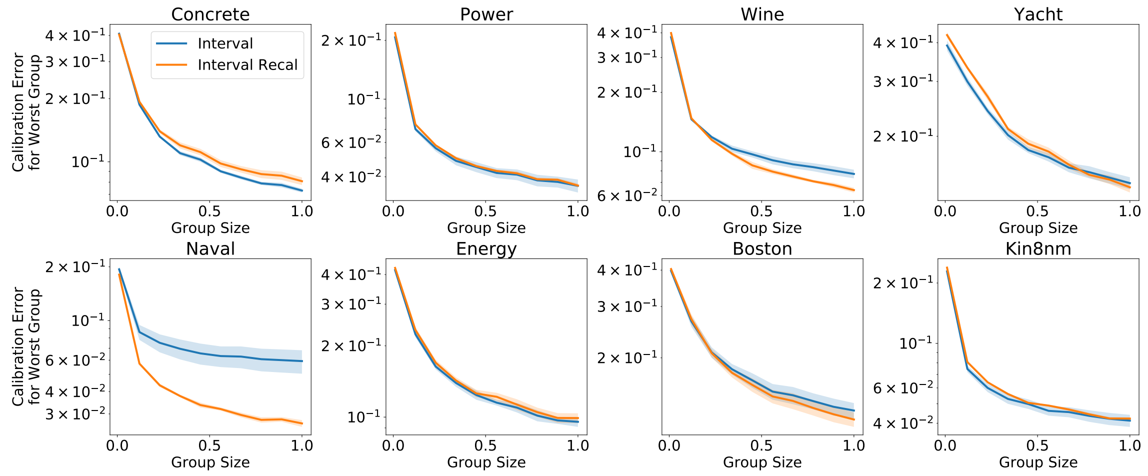


Figure 3.22: **UCI Adversarial Group Calibration with Recalibration.** Recalibration in general shows little improvement in adversarial group calibration.

When we did not apply group batching, each batch was a uniform draw from the training dataset, which is the default setting in most batch optimization procedures.

3.F.2 Full Ablation Study Experiment Results

The full set of results from the ablation study presented in Section 3.4.2 is provided here. To re-iterate the purpose of this study: we show how group batching affects UQ performance on two methods: *Cali* (combined calibration loss, which is one of our proposed methods) and *SQR* (a baseline method).

We present the effect of group batching via all of the evaluation metrics (average calibration, sharpness, adversarial group calibration, check score, interval score, and centered interval calibration).

	Interval	*Interval Recalibrated*
concrete	0.086 ± 0.004	$\mathbf{0.077 \pm 0.004}$
power	0.062 ± 0.001	$\mathbf{0.061 \pm 0.001}$
wine	0.214 ± 0.006	$\mathbf{0.209 \pm 0.013}$
yacht	$\mathbf{0.018 \pm 0.003}$	$\mathbf{0.018 \pm 0.004}$
naval	0.066 ± 0.013	$\mathbf{0.062 \pm 0.012}$
energy	0.017 ± 0.003	$\mathbf{0.016 \pm 0.003}$
boston	0.094 ± 0.009	$\mathbf{0.076 \pm 0.014}$
kin8nm	$\mathbf{0.077 \pm 0.001}$	$\mathbf{0.077 \pm 0.002}$

Figure 3.23: **UCI Check Score with Recalibration.** Recalibration tends to improve the check score.

	Interval	*Interval Recalibrated*
concrete	0.943 ± 0.053	$\mathbf{0.778 \pm 0.064}$
power	0.620 ± 0.010	$\mathbf{0.616 \pm 0.006}$
wine	2.197 ± 0.045	$\mathbf{1.921 \pm 0.024}$
yacht	0.190 ± 0.021	$\mathbf{0.158 \pm 0.025}$
naval	$\mathbf{3.112 \pm 0.053}$	3.150 ± 0.050
energy	0.182 ± 0.026	$\mathbf{0.148 \pm 0.028}$
boston	1.010 ± 0.118	$\mathbf{0.931 \pm 0.107}$
kin8nm	0.776 ± 0.017	$\mathbf{0.754 \pm 0.023}$

Figure 3.24: **UCI Interval Score with Recalibration.** Recalibration tends to improve the interval score.

	Interval	*Interval Recalibrated*
concrete	$\mathbf{0.061 \pm 0.008}$	0.068 ± 0.015
power	$\mathbf{0.023 \pm 0.003}$	0.028 ± 0.008
wine	$\mathbf{0.079 \pm 0.014}$	$\mathbf{0.079 \pm 0.019}$
yacht	$\mathbf{0.121 \pm 0.005}$	0.136 ± 0.025
naval	$\mathbf{0.043 \pm 0.014}$	0.105 ± 0.016
energy	$\mathbf{0.060 \pm 0.010}$	0.066 ± 0.005
boston	0.079 ± 0.015	$\mathbf{0.078 \pm 0.012}$
kin8nm	$\mathbf{0.048 \pm 0.006}$	0.061 ± 0.010

Figure 3.25: **UCI Centered Interval Calibration with Recalibration.** Recalibration tends to worsen centered interval calibration.

3.G Considerations for Epistemic Uncertainty

3.G.1 Sources for Epistemic Uncertainty

The primary focus of this paper is on learning a quantile model. For any single setting of the parameters of the quantile model, the model outputs the current best estimate of the *true underlying distribution* of the dataset. Following the notation laid out in Section 4.2, the learned quantile model

	Cali	
	Random Batch	*Group Batch*
Concrete	$6.6 \pm 0.9(17.6 \pm 2.3)$	$\mathbf{5.6} \pm 0.8(\underline{17.3} \pm 1.5)$
Power	$\mathbf{1.7} \pm 0.2(14.2 \pm 0.3)$	$2.0 \pm 0.1(\underline{13.1} \pm 0.1)$
Wine	$4.4 \pm 0.5(\underline{25.6} \pm 0.8)$	$\mathbf{4.2} \pm 0.4(26.0 \pm 0.8)$
Yacht	$11.1 \pm 1.8(\underline{1.8} \pm 0.1)$	$\mathbf{8.3} \pm 0.6(2.0 \pm 0.4)$
Naval	$2.8 \pm 0.2(\underline{12.1} \pm 3.1)$	$\mathbf{2.4} \pm 0.3(50.6 \pm 8.6)$
Energy	$9.2 \pm 0.3(\underline{2.8} \pm 0.1)$	$\mathbf{5.8} \pm 0.4(3.6 \pm 0.3)$
Boston	$9.7 \pm 1.3(\underline{10.2} \pm 0.7)$	$\mathbf{8.5} \pm 1.5(10.9 \pm 0.6)$
Kin8nm	$\mathbf{3.4} \pm 0.3(\underline{13.7} \pm 0.4)$	$3.5 \pm 0.3(\underline{13.7} \pm 0.7)$

	SQR	
	Random Batch	*Group Batch*
Concrete	$9.8 \pm 1.3(\underline{7.0} \pm 0.6)$	$\mathbf{7.1} \pm 0.9(8.5 \pm 0.6)$
Power	$\mathbf{2.5} \pm 0.3(14.0 \pm 0.5)$	$2.9 \pm 0.5(\underline{13.6} \pm 0.8)$
Wine	$4.5 \pm 0.4(29.7 \pm 0.5)$	$\mathbf{4.0} \pm 0.4(\underline{28.5} \pm 0.8)$
Yacht	$9.0 \pm 0.9(\underline{0.9} \pm 0.1)$	$\mathbf{8.9} \pm 0.9(2.3 \pm 0.2)$
Naval	$8.6 \pm 1.6(\underline{3.6} \pm 0.1)$	$\mathbf{5.3} \pm 0.8(6.0 \pm 0.5)$
Energy	$10.2 \pm 0.8(\underline{1.8} \pm 0.1)$	$\mathbf{6.9} \pm 1.1(2.4 \pm 0.2)$
Boston	$10.9 \pm 1.0(\underline{8.8} \pm 1.1)$	$\mathbf{9.8} \pm 1.2(9.5 \pm 0.9)$
Kin8nm	$4.7 \pm 0.3(\underline{11.1} \pm 0.1)$	$\mathbf{3.9} \pm 0.4(11.3 \pm 0.2)$

Figure 3.26: **Group Batching Ablation: Average Calibration and Sharpness.** The table shows mean ECE and sharpness (in parentheses) and their standard error with and without group batching. The best mean ECE for each dataset has been bolded and the best mean sharpness has been underlined for *Cali* and *SQR* separately. All values have been multiplied by 100 for readability. This is the same table as Figure 3.4 from Section 3.4.2, which is repeated here for completeness.

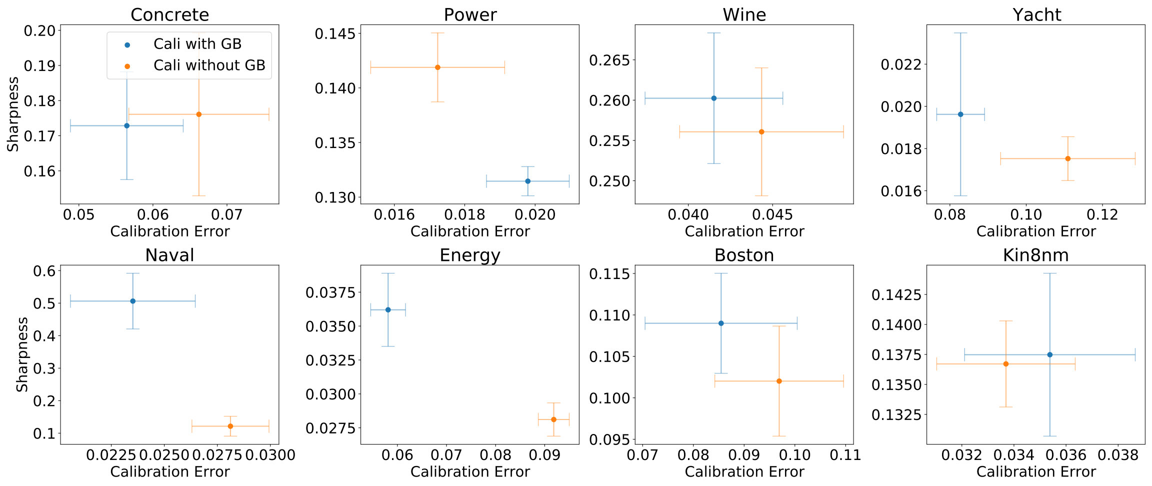


Figure 3.27: **Group Batching Ablation with *Cali* Method: Average Calibration and Sharpness Plot.** GB in legend refers to group batching. Group batching tends to improve calibration and worsen sharpness for the *Cali* method.

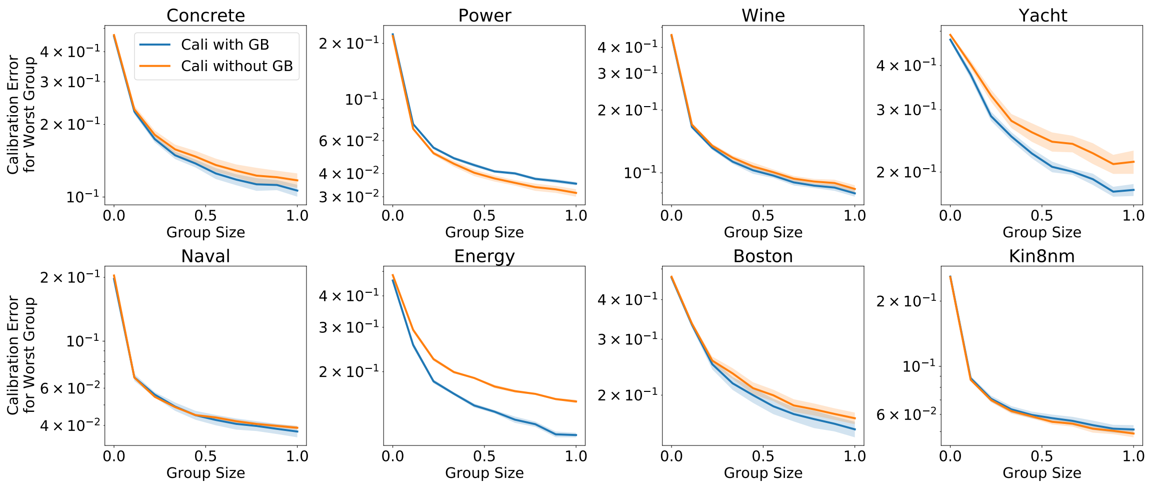


Figure 3.28: **Group Batching Ablation with *Cali* Method: Adversarial Group Calibration.** GB in legend refers to group batching. Group batching tends to improve adversarial group calibration for the *Cali* method.

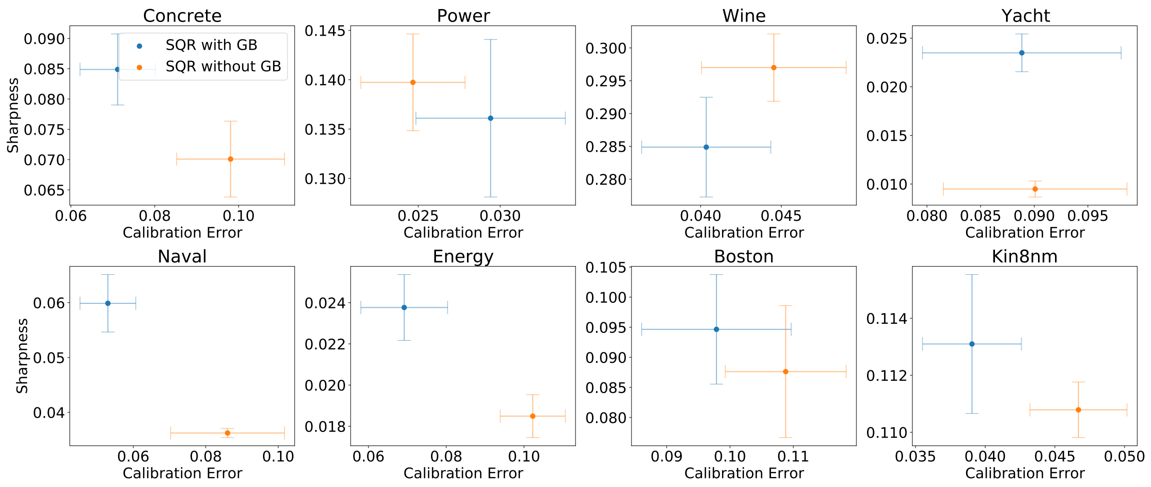


Figure 3.29: **Group Batching Ablation with *SQR* Method: Average Calibration and Sharpness Plot.** GB in legend refers to group batching. Group batching tends to improve calibration and worsen sharpness for *SQR* method.

$\hat{Q}$ is a best approximation to $\mathbb{Q}$, the quantile function of the true distribution.

Meanwhile, epistemic uncertainty refers to the uncertainty *in making the distributional prediction, $\hat{Q}$*. Pearce et al. [2018b] provides one method of decomposing the sources of epistemic uncertainty in a regression setting:

- *Model misspecification:* $\hat{Q}$ may lack the flexibility to accurately model $\mathbb{Q}$, leading to systematic bias.
- *Data uncertainty:* $\hat{Q}$ may not be estimated using a representative sample $\{x_i, y_i\}$ from the assumed underlying distribution.
- *Parameter uncertainty:* $\hat{Q}$ may not be estimated using a large enough quantity of samples, leading to uncertainty about the estimated quantity.

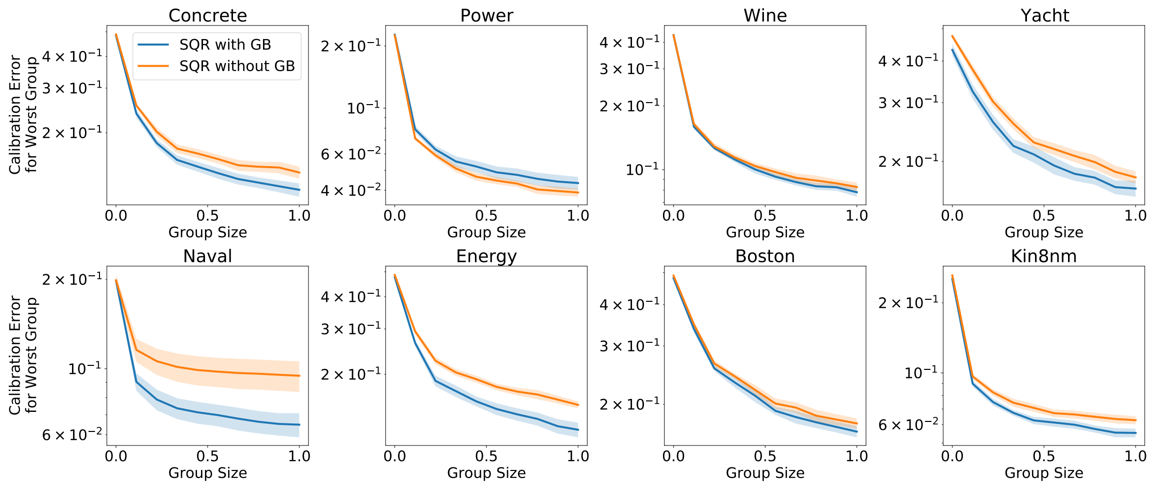


Figure 3.30: **Group Batching Ablation with *SQR* Method: Adversarial Group Calibration.** GB in legend refers to group batching. Group batching tends to improve adversarial group calibration for *SQR* method.

	Cali		*SQR*	
	Random Batch	*Group Batch*	*Random Batch*	*Group Batch*
Concrete	0.120 ± 0.007	$\mathbf{0.118 \pm 0.006}$	0.083 ± 0.006	$\mathbf{0.077 \pm 0.004}$
Power	$\mathbf{0.063 \pm 0.002}$	0.064 ± 0.001	$\mathbf{0.056 \pm 0.001}$	0.057 ± 0.001
Wine	$\mathbf{0.209 \pm 0.007}$	0.210 ± 0.008	$\mathbf{0.206 \pm 0.008}$	$\mathbf{0.206 \pm 0.008}$
Yacht	$\mathbf{0.018 \pm 0.002}$	0.019 ± 0.004	$\mathbf{0.011 \pm 0.002}$	0.013 ± 0.002
Naval	$\mathbf{0.042 \pm 0.009}$	0.159 ± 0.029	$\mathbf{0.007 \pm 0.000}$	0.014 ± 0.001
Energy	$\mathbf{0.016 \pm 0.000}$	0.017 ± 0.002	$\mathbf{0.013 \pm 0.000}$	$\mathbf{0.013 \pm 0.001}$
Boston	$\mathbf{0.100 \pm 0.013}$	0.103 ± 0.013	0.091 ± 0.007	$\mathbf{0.089 \pm 0.008}$
Kin8nm	$\mathbf{0.094 \pm 0.002}$	0.096 ± 0.005	0.079 ± 0.001	$\mathbf{0.077 \pm 0.000}$

Figure 3.31: **Group Batching Ablation: Check Score.** This table shows mean test check score and their standard error with and without group batching for both *Cali* and *SQR*. The best mean for each dataset has been bolded for *Cali* and *SQR* separately. While the general pattern is that group batching worsens the check score, this is expected because group batching tends to worsen sharpness and the check score favors sharpness (Proposition 1 in main paper). Still, the change in the mean of the check score tends to be insignificant when considering the standard errors (except for Naval dataset).

Pearce et al. [2018b] has argued that, given the rich class of function approximators at hand today (NN, deep trees, ensembles), model misspecification can be ignored, which we believe is reasonable. In modeling the remaining sources of uncertainties in $\hat{Q}$, we can incorporate common standard methods to quantify epistemic uncertainty, including boostrapping the data, creating an ensemble of estimates for $\hat{Q}$ with random parameter initializations, or fitting a residual process [Liu et al., 2019]. Here, we describe one combination of these methods: an ensemble of estimates of the learned quantile function $\{\hat{Q}^{(1)}, \hat{Q}^{(2)}, ...\}$ each trained with random initialization (to address parameter uncertainty), on a bootstrapped sample of the training data (to address data uncertainty). The uncertainty over this set of models is the epistemic uncertainty.

	Cali		*SQR*	
	Random Batch	*Group Batch*	*Random Batch*	*Group Batch*
Concrete	1.498 ± 0.083	$\mathbf{1.465 \pm 0.086}$	1.254 ± 0.120	$\mathbf{1.079 \pm 0.066}$
Power	$\mathbf{0.667 \pm 0.025}$	0.699 ± 0.019	$\mathbf{0.603 \pm 0.016}$	0.615 ± 0.014
Wine	$\mathbf{2.495 \pm 0.130}$	2.498 ± 0.135	$\mathbf{2.325 \pm 0.107}$	2.362 ± 0.117
Yacht	$\mathbf{0.276 \pm 0.040}$	0.298 ± 0.063	0.177 ± 0.033	$\mathbf{0.164 \pm 0.029}$
Naval	$\mathbf{0.479 \pm 0.098}$	1.560 ± 0.268	$\mathbf{0.069 \pm 0.003}$	0.144 ± 0.014
Energy	0.218 ± 0.009	$\mathbf{0.204 \pm 0.018}$	0.191 ± 0.006	$\mathbf{0.172 \pm 0.012}$
Boston	$\mathbf{1.437 \pm 0.255}$	1.449 ± 0.259	1.284 ± 0.151	$\mathbf{1.217 \pm 0.152}$
Kin8nm	$\mathbf{1.102 \pm 0.031}$	1.121 ± 0.072	0.914 ± 0.016	$\mathbf{0.871 \pm 0.011}$

Figure 3.32: **Group Batching Ablation: Interval Score.** This table shows mean test interval score and their standard error with and without group batching for both *Cali* and *SQR*. The best mean for each dataset has been bolded for *Cali* and *SQR* separately. The general pattern is that the interval score worsens for *Cali* and improves for *SQR*. However, the change tends to be insignificant when considering the standard errors (except for Naval dataset).

	Cali		*SQR*	
	Random Batch	*Group Batch*	*Random Batch*	*Group Batch*
Concrete	0.102 ± 0.014	$\mathbf{0.096 \pm 0.013}$	0.188 ± 0.029	$\mathbf{0.127 \pm 0.014}$
Power	$\mathbf{0.033 \pm 0.003}$	0.037 ± 0.002	$\mathbf{0.040 \pm 0.008}$	0.047 ± 0.006
Wine	0.072 ± 0.007	$\mathbf{0.065 \pm 0.007}$	0.057 ± 0.005	$\mathbf{0.055 \pm 0.005}$
Yacht	0.139 ± 0.018	$\mathbf{0.129 \pm 0.016}$	0.128 ± 0.027	$\mathbf{0.119 \pm 0.020}$
Naval	0.048 ± 0.006	$\mathbf{0.034 \pm 0.002}$	0.114 ± 0.032	$\mathbf{0.066 \pm 0.009}$
Energy	0.146 ± 0.013	$\mathbf{0.090 \pm 0.011}$	0.171 ± 0.018	$\mathbf{0.104 \pm 0.020}$
Boston	0.142 ± 0.032	$\mathbf{0.138 \pm 0.028}$	0.195 ± 0.021	$\mathbf{0.173 \pm 0.027}$
Kin8nm	$\mathbf{0.061 \pm 0.004}$	0.067 ± 0.005	0.079 ± 0.003	$\mathbf{0.069 \pm 0.009}$

Figure 3.33: **Group Batching Ablation: Centered Interval Calibration.** This table shows mean test centered interval calibration (measured by ECE for centered intervals) score and their standard error with and without group batching for both *Cali* and *SQR*. The best mean for each dataset has been bolded for *Cali* and *SQR* separately. The general pattern is that group batching improves centered interval calibration for both *Cali* and *SQR*. While the change tends to be insignificant when considering the standard errors, the improvement is significant in numerous cases (e.g. Naval and Energy for both methods, Concrete with *SQR*).

3.G.2 Expressing and Utilizing Epistemic Uncertainty

Once we decide on methods to quantify the epistemic uncertainty, the next question is *how to express the epistemic uncertainty*, especially when combining it with the current prediction of the aleatoric uncertainty. This is still an open research question, especially in the regression setting, and methods of combining aleatoric with epistemic uncertainty will differ for how the uncertainty is represented (e.g. density estimates, quantiles, prediction intervals).

One method of combination is to consider the utility of quantifying epistemic uncertainty. Intuitively, for a given input, if we have high epistemic uncertainty, the combined uncertainty should be higher (i.e. less confident prediction), and vice versa. If we consider a single quantile, it is unclear whether lower confidence (due to epistemic uncertainty) would equate to a lower or higher quantile estimate. However, if we consider constructing a centered prediction interval for total uncertainty, it is straightforward to see that lower confidence should widen the interval, by raising the upper bound (quantile level above 0.5) and lowering the lower bound (quantile level below 0.5). This conservative upper and lower bound can be constructed with the bootstrap distribution of each quantile according to each ensemble member prediction. This is also the method utilized in Pearce et al. [2018b].

Suppose we have an ensemble of M quantile models: $\{\hat{Q}^{(1)}, \hat{Q}^{(2)}, \ldots, \hat{Q}^{(M)}\}$. For any test point x^* and test coverage level $(1 - \alpha^*)$, the total uncertainty represented by a centered prediction interval with upper bound $\hat{u}$ and lower bound $\hat{l}$ is constructed as:

$$\hat{l} = \bar{\mu}(x^*, \alpha^*/2) - z\frac{s(x^*, \alpha^*/2)}{\sqrt{M}}$$

$$\hat{u} = \bar{\mu}(x^*, 1 - \alpha^*/2) + z\frac{s(x^*, 1 - \alpha^*/2)}{\sqrt{M}}$$

$$\bar{\mu}(x^*, p^*) = \frac{1}{M}\sum_{i=1}^{M} \hat{Q}^{(i)}_{p^*}(x^*)$$

$$s(x^*, p^*) = \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(\hat{Q}^{(i)}_{p^*}(x^*) - \bar{\mu}(x^*, p^*))^2}$$

and z is the chosen critical value (e.g. 1.96 for a conservative bound that takes the 95% confidence interval of the bootstrap distribution). In words, the construction of $\hat{u}, \hat{l}$ equates to constructing a conservative prediction interval that depends on how dispersed or concentrated each ensemble member's predictions are.

3.G.3 Metrics for Total Uncertainty Evaluation

After choosing a method to express epistemic uncertainty and combining it with aleatoric uncertainty for a prediction of total uncertainty, next comes the question of *how to evaluate the combined uncertainty*.

The critical point here is that the calibration, sharpness and proper scoring rule metrics we have discussed thus far *are not applicable here*. This is because these metric only judge how close the prediction is to the true underlying distribution. In fact, ECE (a measure of average calibration) can be shown to be identical to the Wasserstein distance between distributions under the $L1$ distance metric [Zhao et al., 2020]. In a hypothetical setting where we have very few datapoints and hence very high epistemic uncertainty throughout the whole data support, if one distributional prediction was extremely lucky and predicted a distribution that adheres exactly to the true underlying distribution, the aforementioned metrics will consider this prediction a perfect prediction – however, a lucky guess is not at all a useful UQ, and to a practitioner, a less confident prediction (by quantifying high epistemic uncertainty) is much more useful, rather than a very confident prediction that can be correct if it is lucky, but confidently very incorrect otherwise.

We also emphasize here that, while standard evaluation experiments and metrics exist for the *classification* setting (e.g. by training an image classifier on the MNIST dataset and testing on the Non-MNIST dataset to assess the entropy of the predicted class probabilities or the output of a trained out-of-distribution detector), there does not exists standard experiments and metrics to evaluate epistemic uncertainty in the *regression setting*.

Therefore, we propose one evaluation metric to assess combined total uncertainty, which is *sharpness subject to sufficient coverage* and we will refer to this metric as "epistemic coverage". Epistemic coverage measures average calibration of a centered prediction interval, but the difference in observed probabilities from expected probabilities is penalized only when the observed probability is less than the expected probability (i.e. do not penalize the prediction if the observed probability is higher than the expected probability), and if this sufficient coverage condition is met, then we evaluate sharpness.

By this metric, only over-confidence is penalized and under-confidence is considered acceptable. However, since infinitely wide prediction intervals are also not useful, we also consider sharpness after sufficient coverage is met.

3.G.4 Demonstrating Effect of Bootstrap Ensembling and the Epistemic Coverage Metric

We design an "epistemic experiment" to show the effect of incorporating epistemic uncertainty via bootstrapped ensembles. In this experiment, we swap the train and test sets, such that the training set is much smaller than the test set (roughly $\frac{1}{7}$ of test set size), hence, the model should have very high epistemic uncertainty in making predictions on the test set. It is expected that by not incorporating epistemic uncertainty in such a setting, the model will produce *overconfident* predictions that are *too sharp*. This overconfidence will be penalized heavily by the epistemic coverage metric we described above in Appendix 3.G.3, and sharpness should also indicate that the predictions are too tight. Producing conservative distributional predictions via the bootstrapped ensembling technique described above in Appendix 3.G.2 is expected to mitigate these overconfidence issues by incorporating epistemic uncertainty.

We show the effect on one of our methods, *Interval*, because the same effect can be observed for any of the quantile methods, including the baseline algorithms. The results are shown in Figure 3.34.

When a conservative PI is constructed with the bootstrapped ensemble (labelled *Interval Boot-Ens*), the epistemic coverage error decreases significantly to or near zero, which is expected given that the conservative bounds only work to widen the PI, and the epistemic coverage error only penalizes PI that are not wide enough. The increase in width is also evident in the increase in sharpness with bootstrap ensembling.

Therefore, the ensembling technique does work to incorporate epistemic uncertainty from a practical standpoint by imbuing more underconfidence into the distributional predictions. Still, quantifying and evaluating epistemic uncertainty in a regression setting is an open problem, and we leave for future work developing alternative methods of quantifying epistemic uncertainty in regression.

3.H Additional Discussions

Potential negative social impacts This work proposes methods in uncertainty quantification (UQ), with a focus on the notion of calibration. UQ is a field that is becoming more and more important as many autonomous systems are being deployed in various real-life applications (self-driving cars, security devices, object recognition systems). While we believe the development of robust and accurate methods in UQ will accelerate deployment of autonomous systems and expand real-world use-cases, we also acknowledge the potential disruptive effects such change can have in the relevant industries.

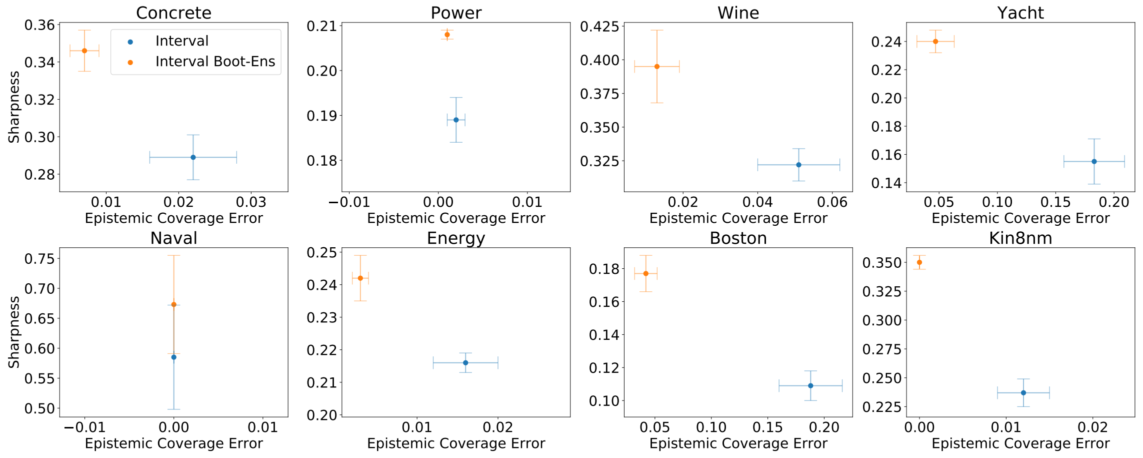


Figure 3.34: **Epistemic Experiments on UCI Datasets.** We evaluate epistemic coverage in the epistemic experiment setting where the train set is much smaller than the test set. Epistemic coverage only penalizes overconfidence. Sharpness is the average width of the 95% PI.

Futher, calibration is a relevant notion in fairness [Kleinberg et al., 2016]. Though we propose methods to achieve better calibration when making probabilistic forecasts, we note the potential for using such information with ill intent, e.g. to intentionally avoid calibrated (or fair) decisions.

Assets used in this work In implementing our work, we have referenced the implementation of one of the baseline methods (*SQR*), which is publicly available under the Creative Commons Attribution-NonCommercial 4.0 International Public License.

We also state that no data from human subjects were used in this work and thus there is no personal identifying information.

4 | Sampling-based Multi-dimensional Recalibration

Calibration of probabilistic forecasts in the regression setting has been widely studied in the single dimensional case, where the output variables are assumed to be univariate. In many problem settings, however, the output variables are multi-dimensional, and in the presence of dependence across the output dimensions, measuring calibration and performing recalibration for each dimension separately can be both misleading and detrimental. In this work, we focus on representing predictive uncertainties via samples, and propose a recalibration method which accounts for the joint distribution across output dimensions to produce calibrated samples. Based on the concept of highest density regions (HDR), we define the notion of HDR calibration, and show that our recalibration method produces samples which are HDR calibrated. We demonstrate the performance of our method and the quality of the recalibrated samples on a suite of benchmark datasets in multi-dimensional regression, a real-world dataset in modeling plasma dynamics during nuclear fusion reactions, and on a decision-making application in forecasting demand.

4.1 Introduction

Calibration in probabilistic forecasting, in general terms, refers to the alignment between the predicted probabilities and empirical frequencies of the true observations. Alongside other quantitative metrics to assess predictive distributions, e.g. negative log-likelihood (NLL) or simply accuracy of the mean, calibration is considered an important and desirable quality of probabilistic forecasts, and many works have appraised the utility of calibration in various application settings [Gneiting et al., 2007, Malik et al., 2019, Deshpande and Kuleshov, 2021, Chung et al., 2023b].

Within the general principle of "aligning predicted and empirical probabilities", various notions of calibration exist, and these definitions also vary slightly between classification and regression settings. In this work, we focus on the regression setting where both the inputs, X, and targets, Y, are continuous.

We begin our discussion from the observation that the most widely studied notions of calibration in regression are usually confined to the setting where the targets are single dimensional [Gneiting et al., 2007, Pearce et al., 2018b, Kuleshov et al., 2018, Song et al., 2019, Cui et al., 2020, Zhao et al., 2020, Sahoo et al., 2021, Kuleshov and Deshpande, 2022]. While multi-dimensional regression models are widely used in machine learning, especially in applications such as model-based control [Chua et al., 2018, Malik et al., 2019, Yu et al., 2020, Kidambi et al., 2020] or modeling in the physical sciences [Sexton et al., 2012, Duraisamy et al., 2019, Abbate et al., 2021, Char et al.,

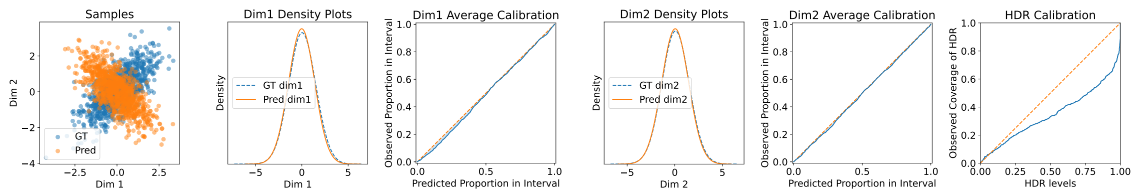


Figure 4.1: We demonstrate a pitfall of assessing the calibration of each dimension independently for multi-dimensional distributional predictions. **(From Left to Right)** Consider a 2-dimensional target space where samples from the predictive distribution (labeled *Pred*) and ground truth distribution (labeled *GT*) are displayed as a scatter plot. The predictive distribution exhibits the opposite correlation in the output dimensions compared to the ground truth, but each of the marginal distributions are accurate. Assessing calibration of each dimension separately suggests a well-calibrated predictive distribution. Highest density regions (HDRs), on the other hand, are able to account for the dependence in the dimensions. Assessing HDR calibration, which considers the output dimensions *jointly*, reveals the miscalibration of the full joint distribution.

2023a], we find that methods which account for the joint multi-dimensional distribution in *assessing* calibration and *recalibrating* the prediction is generally lacking. In lieu of more sophisticated methods, calibration is often considered for each output dimension independently. However, failing to account for interplay among the output dimensions may be problematic when dependence does exist. In this case, the collection of marginals is not sufficient to provide an accurate assessment of the prediction quality (see Figure 4.1 for an example).

In this work, we address the problem of calibration in multi-dimensional regression by first formalizing a notion of calibration which *can* account for dependence among the output dimensions and further proposing a recalibration algorithm for the joint predictive distribution. We summarize our main contributions as follows:

- By leveraging existing ideas in highest density regions (HDR), we propose the notion of *HDR calibration*, which accounts for dependence in the output dimensions in defining and evaluating calibration for multi-dimensional distributional predictions.
- We develop a recalibration algorithm for multi-dimensions which produces HDR calibrated predictive distributions via a sampling procedure.
- We provide extensive demonstrations of the merits of the notion of HDR calibration and the efficacy of the recalibration algorithm on a suite of benchmark datasets in multi-dimensional regression, and two real-world datasets: a dynamics modeling task in nuclear fusion, and a downstream decision-making application in forecasting customer demand.

We continue our discussion by first describing the problem setting and relevant concepts to motivate the definition of HDR calibration in Section 4.2. Based on this notion of calibration, we present our proposed HDR recalibration algorithm in Section 4.3. We provide empirical evaluations in Section 6.3[1].

[1]Code is available at: `https://github.com/YoungseogChung/multi-dimensional-recalibration`

4.2 Preliminaries and Related Works

4.2.1 Setting and Notation

Upper case letters X, Y denote random variables (r.v.), and lower case letters x, y denote their observed values. We consider the regression setting with an input feature space $\mathcal{X} \subseteq \mathbb{R}^n$ and a target space $\mathcal{Y} \subseteq \mathbb{R}^D$. We use x^d, X^d, y^d, and Y^d to denote the d^{th} dimension of input and target vectors. f and F denote the true probability density function (PDF) and cumulative distribution function (CDF), and when it exists, we denote the true quantile function with F^{-1}. Estimates of these functions are denoted with $\hat{f}$, $\hat{F}$ and $\hat{F}^{-1}$. We use subscripts to indicate the corresponding random variable of the PDFs and CDFs (e.g. f_X and F_X are the marginal PDF and CDF of X, and $f_{Y|X}$ and $F_{Y|X}$ are the PDF and CDF of Y conditioned on X). When conditioning on a specific value $X = x$, we denote the conditional distribution functions as $f_{Y|x}$ and $F_{Y|x}$. Lastly, we assume that new target samples can be drawn from the distribution estimate, and we denote the random variable corresponding to these new target samples as $\hat{Y}$. In particular, this can be done by sampling $X \sim f_X$ from the dataset and subsequently sampling $\hat{Y}|X \sim \hat{f}_{Y|X}$. Importantly, note that the distribution of $\hat{Y}$ is still tied to the distribution of X.

4.2.2 Calibration in Univariate Regression

Before discussing the multi-dimensional setting, we first provide a brief review of notions of calibration in the univariate setting. A widely accepted notion of calibration in univariate regression is *probabilistic calibration* [Gneiting et al., 2007]. A predictive distribution $\hat{F}_{Y|X}$ is probabilistically calibrated if

$$P(Y \leq \hat{F}^{-1}_{Y|X}(p)) = p, \forall p \in (0, 1). \tag{4.1}$$

This notion is also referred to as simply *calibration* [Kuleshov et al., 2018], *quantile calibration* [Song et al., 2019], or *average calibration* [Zhao et al., 2020, Chung et al., 2021b] since it focuses on the average validity of the predictive quantile function $\hat{F}^{-1}_{Y|X}$. We henceforth refer to this notion as *average calibration*. Here, we note that the true distribution $F_{Y|X}$ trivially satisfies Eq. 8.1 since $F_{Y|X}(Y) \sim \mathcal{U}(0, 1)$ by the probability integral transform and $P(\hat{F}_{Y|X}(Y) \leq p) = p$ is the CDF of $\mathcal{U}(0, 1)$.

From this general definition, subsequent works have derived various notions of calibration, usually by placing different conditions in assessing the empirical probability (LHS of Eq. 8.1). For example, *distribution calibration* [Song et al., 2019] assesses average calibration conditioned on the predictive distribution; *individual calibration* [Zhao et al., 2020] requires average calibration conditioned on each input point, $x \in \mathcal{X}$; and *group calibration* [Kleinberg et al., 2016, Hébert-Johnson et al., 2017] requires average calibration conditioned on specific subsets of the input space with non-zero measure.

In all of the aforementioned notions, Y is assumed to be univariate (i.e $\mathcal{Y} \subseteq \mathbb{R}$), and predictive conditional quantiles $\hat{F}^{-1}_{Y|X} : \mathcal{X} \times (0, 1) \rightarrow \mathcal{Y}$ are utilized to measure the discrepancy between predicted and empirical probabilities (RHS and LHS of Eq. 8.1).

4.2.3 The Multi-dimensional Setting

While a naive application of the notions of univariate calibration to multi-dimensional distribution functions may seem plausible, in the multivariate setting, the quantile function is not well-defined [Belloni and Winkler, 2009], and further, $F_Y(Y)$ for $Y \in \mathbb{R}^D$ when $D > 1$ is no longer uniformly distributed [Barbe et al., 1996, Genest and Rivest, 2001]. To circumvent these issues, prior works have suggested utilizing projections of the target variable Y in order to *define* and *assess* calibration of multi-dimensional distributional predictions. We formalize such methods as follows.

Consider a mapping $g : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, where $\mathcal{Z} \subseteq \mathbb{R}$. Furthermore, we let Z and $\hat{Z}$ be the r.v.s over the projection outputs when using target labels Y and $\hat{Y}$, respectively. Concretely, $Z := g(X, Y)$ and $\hat{Z} := g(X, \hat{Y})$. Since sampling from the predicted distribution is cheap, we can estimate the CDF $F_{Z|X}$ using the empirical distribution of $\hat{Z}|X$. We refer to this empirical CDF as $\hat{F}_{Z|X}$.

Then, following the definition of average calibration (Eq. 8.1), we can define calibration in the projected space as satisfying, $\forall p \in (0, 1)$,

$$P(Z \leq \hat{F}^{-1}_{Z|X}(p)) = p \tag{4.2}$$

$$\text{or equivalently, } P(\hat{F}_{Z|X}(Z) \leq p) = p. \tag{4.3}$$

We can easily show that the optimal prediction $\hat{F}_{Y|X} = F_{Y|X}$ satisfies this definition of calibration in the projected space.

Proposition 1. *The optimal distributional prediction, i.e.* $\hat{F}_{Y|X} = F_{Y|X}$, *satisfies calibration in the projected space, Eq. 4.3. (proof in Section 4.A)*

Several prior works have proposed specific versions of Eq. 4.3 with specific projection functions. Ziegel and Gneiting [2014] introduced *copula calibration* by utilizing the predictive CDF as the projection function, i.e. $g(X, \cdot) = \hat{F}_{Y|X}$. In this specific case, the distribution of the projections is called the Kendall distribution [Nelsen et al., 2003].

One can also utilize the predictive PDF for the projection function such that $g(X, \cdot) = \hat{f}_{Y|X}$, in which case Eq. 4.3 bears intrinsic relationships to existing concepts of highest predictive density (HPD) values [Harrison et al., 2015, Dalmasso et al., 2020, Zhao et al., 2021a] and highest density regions (HDR) [Hyndman, 1996].

While there are several candidates for projection functions, in this work, we choose to focus on using the predictive PDF. In particular, we leverage its connections with HPD and HDR to formalize a notion of calibration in multi-dimensions (Defn. 1) and propose a recalibration procedure that achieves this notion of calibration (Section 4.3). Hence, in the rest of this work, we always assume $Z := \hat{f}_{Y|X}(Y)$ and $\hat{Z} := \hat{f}_{Y|X}(\hat{Y})$.

For any given (x, y), $\texttt{HPD}_x(y)$ is a measure of how plausible y is w.r.t $\hat{f}_{Y|x}$ and is defined as

$$\texttt{HPD}_x(y) = \int_{y' : \hat{f}_{Y|x}(y') \geq \hat{f}_{Y|x}(y)} \hat{f}_{Y|x}(y') dy'. \tag{4.4}$$

In words, $\texttt{HPD}_x(y)$ is the predicted probability of observing $\hat{Y}$ that is more likely than y, where the likelihood is determined by $\hat{f}_{Y|x}$. Considering the definition of Z and $\hat{Z}$, we see that

$$\texttt{HPD}_x(y) \tag{4.5}$$

$$= P(\hat{f}_{Y|x}(\hat{Y}) \geq \hat{f}_{Y|x}(y) \mid X = x) \tag{4.6}$$

$$= 1 - \hat{F}_{Z|x}(\hat{f}_{Y|x}(y)). \tag{4.7}$$

Further, HPD values, which are *probabilities*, have a direct relationship to HDRs, which are *prediction sets*. For clarity, we provide a definition of HDR below using our notation, and we refer the reader to Section 4.C for the original notation by Hyndman [1996]. For a fixed x and constant $\lambda \in \mathbb{R}$, we define the λ-density region as $\texttt{DR}_x(\lambda) := \{y : \hat{f}_{Y|x}(y) \geq \lambda\}$. Then for a given coverage level p, the p-HDR is the smallest density region with probability greater than or equal to p. Concretely,

$$\texttt{HDR}_x(p) := \texttt{DR}_x(\lambda^*)$$
$$\text{where} \quad \lambda^* = \sup\{\lambda : P(\hat{Y} \in \texttt{DR}_x(\lambda) | X = x) \geq p\}.$$

By their definitions, the following equivalence holds:

$$\texttt{HPD}_x(y) \leq p \iff y \in \texttt{HDR}_x(p) \tag{4.8}$$

We note that calibration is generally defined in terms of prediction sets of a distribution, and drawing on the intrinsic relationships between HDR, HPD and Eq. 4.7, we formalize Eq. 4.3 with the notion of HDR calibration:

Definition 1. *A predictive PDF $\hat{f}_{Y|X}$ is HDR calibrated if, $\forall p \in (0, 1)$,*

$$P(Y \in \texttt{HDR}_X(p)) = p \tag{4.9}$$
$$\text{or equivalently, } P(\texttt{HPD}_X(Y) \leq p) = p. \tag{4.10}$$

Proposition 2. *HDR calibration holds if and only if Equation 4.3 holds. (proof in Section 4.A)*

Similar to average calibration (Eq. 8.1), which requires Y to be contained in the p-prediction set (defined by the p^{th} quantile) with probability p, HDR calibration requires the p-HDR to contain Y with probability p.

Utilizing projections allows one to *define* notions of calibration in the multi-dimensional setting which can account for dependence in the output dimension, granted that the projection function models the dependence. Further, based on the definitions, one can *assess* calibration (or miscalibration) via the discrepancy between the predicted and empirical probabilities. Following the commonly used notion of expected calibration error (ECE) [Guo et al., 2017, Cui et al., 2020, Tran et al., 2020, Chung et al., 2021b] we can measure the (L1-)ECE w.r.t the general notion of calibration defined in Eq. 4.3 as

$$\mathbb{E}_{p \sim \mathcal{U}(0,1)} \left| P(\hat{F}_{Z|X}(Z) \leq p) - p \right|. \tag{4.11}$$

Not only do these metrics allow one to evaluate the quality of uncertainty for multi-dimensional predictions, but they can also be used to improve a model's predictive distribution via a recalibration procedure.

4.2.4 Recalibration

Probabilistic models are usually trained by optimizing a loss function which may not be necessarily aligned with calibration. This can often lead to models being miscalibrated at the end of the training [Guo et al., 2017, Kuleshov et al., 2018, Chung et al., 2021b]. A *post-hoc recalibration* step can be applied on top of the trained model to adjust for its level of miscalibration observed on a held-out calibration or validation dataset.

Post-hoc recalibration is well-studied in classification, and there are several methods which have proven to be effective in producing well-calibrated (discrete) class probabilities [Platt, 1999, Zadrozny and Elkan, 2001, 2002, Guo et al., 2017, Gupta and Ramdas, 2021].

This problem is not as widely studied in regression, however, and to the best of our knowledge, the most popular method is that of Kuleshov et al. [2018], which learns an isotonic mapping between expected and observed quantile levels. Crucially, this method readily applies only to the case when the targets Y are univariate, and we henceforth refer to this algorithm as "single dimensional (SD) recalibration". In Section 4.3, we propose a recalibration procedure for the multivariate setting.

While also proposed for the single dimensional setting, it is worth mentioning that Izbicki et al. [2022] proposes a conformal prediction method which bears relevance as their method utilizes HPD values as the conformity score. However, there are key differences: while they are focused on producing prediction *sets* for a fixed coverage level (as is the goal of conformal prediction), we are focused on expressing the full predictive *distribution*. Crucially, since Izbicki et al. [2022] does not consider multi-dimensional target spaces, their method does not account for dependence in the target dimensions, and the algorithm relies on constructing a finite grid of the target space, which is ill-suited for higher dimensions. As we will discuss in Section 4.3, our recalibration procedure explicitly addresses dependence in the target dimensions and is more scalable as it focuses on *sampling* from a predictive distribution in the multi-dimensional space. We refer the reader to Appendix 4.B for additional details on related works.

4.2.5 Predictive Uncertainty and Sampling

Ensuring the calibration of predictive uncertainties becomes important when deploying probabilistic models in downstream applications. The application setting will dictate how the uncertainties are utilized. For example, in the context of Bayesian optimization, depending on the acquisition function, the uncertainties may be used to construct confidence bounds [Auer, 2002, Srinivas et al., 2009], compute probabilities or expectations [Jones et al., 1998], or be used to sample from [Thompson, 1933, Kandasamy et al., 2018, Char et al., 2019]. For model-based control where a probabilistic dynamics model is learned, the uncertainty-aware model is most often used to sample plausible transitions and trajectories [Chua et al., 2018, Janner et al., 2019, Mehta et al., 2021b, Char et al., 2023c].

In this work, we focus on *sampling* to represent and utilize the predictive uncertainties and aim to produce samples from a well-calibrated predictive distribution. Applying SD recalibration [Kuleshov et al., 2018] to multi-dimensional settings will necessitate recalibrating each dimension separately, which renders each dimension independent in the recalibrated samples. However, we note that this, in fact, is how recalibration is utilized in practice to multi-dimensional settings (e.g. Malik et al. [2019]). The algorithm we propose in the next section performs recalibration *jointly* across all output dimensions and is able to consider dependence across the dimensions.

4.3 Method

In this section, we describe our proposed recalibration procedure which aims to achieve HDR calibration (Defn. 1). We describe the procedure in two parts. Section 4.3.1 details the recalibration algorithm that aims to optimize for Eq. 4.3, which is equivalent to HDR calibration by Proposition 2. Afterwards, Section 4.3.2 describes a pre-conditioning step that can modify the predictive PDF to account for dependence in the output dimensions when applying the recalibration algorithm.

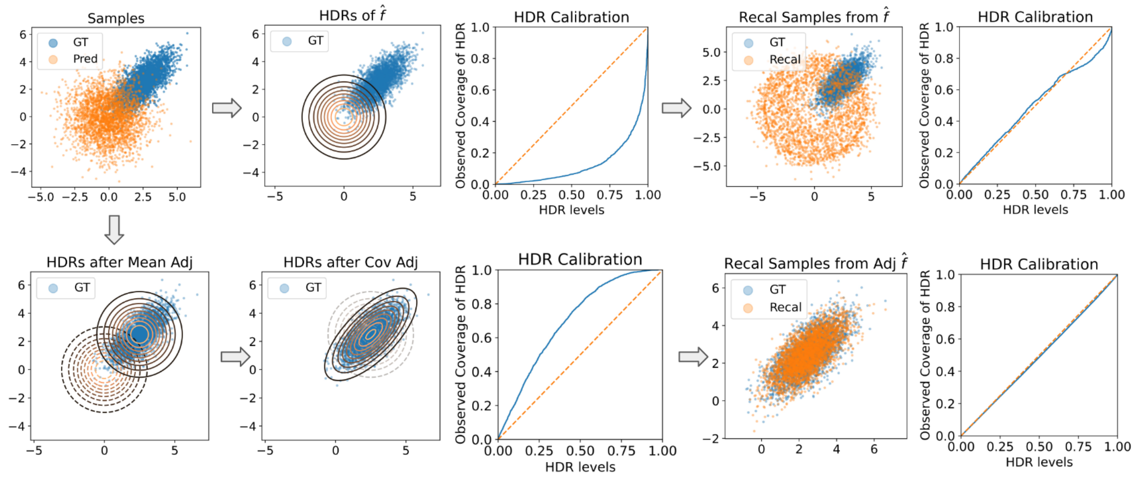


Figure 4.2: Demonstration of HDR recalibration on a marginal distributional prediction. **(Top Left)** The initial prediction (labeled *Pred*) displays bias in the mean prediction and fails to model the correlation in the ground truth distribution (labeled *GT*). **(Top Row)** Without the PDF adjustment step, we observe that observations (GT points) fall more often in the higher level HDRs (level sets defined by darker boundaries) than lower level HDRs (level sets bounded by lighter colors). HDR recalibration re-samples from each HDR according to the observed frequencies (i.e. the learned recalibration mapping), hence when producing recalibrated samples, the higher level HDRs (i.e. outer level sets of $\hat{f}$) are over-sampled and the lower level HDRs (inner level sets of $\hat{f}$) are under-sampled. The resulting recalibrated samples are HDR calibrated (right-most plot), but we can visually assess that the samples are suboptimal and in particular, fail to model the correlation in the dimensions. **(Bottom Row)** Before the recalibration procedure, we can estimate the bias in the mean on the calibration dataset and correlation in the dimensions with the correlation matrix of the mean prediction error. After applying these two adjustments, HDR calibration reveals that each p-HDR contains more than p proportion of the observations (which also indicates that the level sets are too wide). Hence, HDR recalibration proportionately under-samples from each HDR, which results in well-calibrated samples that also reflect the correlation in the output dimensions.

4.3.1 HDR Recalibration Algorithm

The proposed recalibration algorithm is comparable to that of Kuleshov et al. [2018] for univariate settings, but with key differences – the recalibration occurs in the projected space $\mathcal{Z}$, and the recalibration output must be translated back into the target space $\mathcal{Y}$.

First, we estimate a recalibration mapping in the projected space by using observations of the random variable $\hat{F}_{Z|X}(Z)$ with a calibration dataset $\{(x_i, y_i)\}_{i=1}^N$, i.e. the observed values are $\{\hat{F}_{Z|x_i}(z_i)\}_{i=1}^N$ where $z_i = \hat{f}_{Y|x_i}(y_i)$. To elaborate more on this procedure, note that z_i is a scalar value produced by evaluating the PDF $\hat{f}_{Y|x_i}$ at y_i, where $\hat{f}_{Y|x_i}$ is the PDF of the predictive distribution. $\hat{F}_{Z|x_i}(z_i)$ is also a scalar value produced by evaluating the CDF $\hat{F}_{Z|x_i}$ at z_i, however, $\hat{F}_{Z|x_i}$ is an empirical CDF over the projected space that is estimated by producing samples from the predictive distribution $\hat{f}_{Y|x_i}$. Again, we note that sampling from the predictive distribution is cheap, thus estimating this empirical CDF is also cheap. Algorithm 6 provides exact details on this estimation step.

Algorithm 4 HDR Recalibration: Training

```
1: Input: Calibration dataset $\{(x_i, y_i)\}_{i=1}^N$, predictive PDF $\hat{f}_{Y|X}$.
2: $\hat{f}_{Y|X} \leftarrow$ ADJUST($\hat{f}_{Y|X}$) (Algorithms 8, 9, 10).
3: Construct the dataset $\mathcal{C} = \{\hat{F}_{Z|x_i}(z_i)\}_{i=1}^N$, where $z_i = \hat{f}_{Y|x_i}(y_i)$ (see Algorithm 6).
4: Sort values in $\mathcal{C}$ to construct $\{c_{(i)}\}_{i=1}^N$, construct the recalibration dataset $\mathcal{C}' = \{i/N, c_{(i)}\}_{i=1}^N$.
5: Learn the recalibration mapping $R$ on $\mathcal{C}'$.
6: Output: Recalibration mapping $R$.
```

Algorithm 5 HDR Recalibration: Sampling

```
1: Input: Test point $x$, predictive PDF $\hat{f}_{Y|X}$, recalibration mapping $R$, number of samples $M$.
2: $\hat{f}_{Y|X} \leftarrow$ ADJUST($\hat{f}_{Y|X}$). (Algorithms 8, 9, 10).
3: Construct $\mathcal{D} = \{(\hat{y}_j, \hat{z}_j)\}_{j=1}^M$ by producing $M$ samples $\hat{y}_j \sim \hat{f}_{Y|x}$ and setting $\hat{z}_j = \hat{f}_{Y|x}(\hat{y}_j)$.
4: Re-sample from $\mathcal{D}$ to construct $\mathcal{D}' = \{(\hat{y}_k, \hat{z}_k)\}_{k=1}^M$ s.t. $\{\hat{z}_k\}_{k=1}^M$ approximately follows $R \circ \hat{F}_{Z|x}$ (see Algorithm 7).
5: Output: Recalibrated samples at $x$, $\{\hat{y}_k\}_{k=1}^M$.
```

Afterwards, we learn the monotonic mapping $R : [0, 1] \to [0, 1]$ where $R(p) := P(\hat{F}_{Z|X}[Z] \leq p)$. R is then applied to the predictive distribution at each x, $\hat{F}_{Z|x}$, to produce the recalibrated predictive distribution $R \circ \hat{F}_{Z|x}$.

Proposition 3. *Consider $R \circ \hat{F}_{Z|X}$ for an invertible mapping R. Then $R \circ \hat{F}_{Z|X}$ satisfies Eq. 4.3, i.e.*

$$P(R \circ \hat{F}_{Z|X}(Z) \leq p) = p \quad \forall p \in (0, 1).$$

(proof in Section 4.A)

One can therefore use such a recalibration map, R, to draw new, calibrated samples in $\mathcal{Z}$ space. However, it remains unclear how to relate these samples back to their counterparts in $\mathcal{Y}$ space. To address this issue, we present a sampling algorithm that operates over samples of r.v. $\hat{Y}$. The key idea is to re-sample from the set of samples generated from $\hat{f}_{Y|X}$ according to what the distribution should look like in $\mathcal{Z}$ space. In particular, for any fixed x, we can draw many samples from the predictive PDF, $\{\hat{y}_j\}_{j=1}^M \sim \hat{f}_{Y|x}$, then apply the projection $\hat{f}_{Y|x}(\cdot)$ to produce the dataset of tuples $\mathcal{D} = \{(\hat{y}_j, \hat{z}_j)\}$, where $\hat{z}_j = \hat{f}_{Y|x}(\hat{y}_j)$, and note that by definition, $\hat{z}_j \sim \hat{f}_{Z|x}, \hat{F}_{Z|x}$. We then re-sample from $\mathcal{D}$ to produce $\{(y_k, z_k)\} \subseteq \mathcal{D}$ such that the distribution of $\{z_k\}$ is more closely aligned with $R \circ \hat{F}_{Z|x}$. Concretely, this is done by forming an empirical CDF of the $\hat{Z}$ samples $\{\hat{z}_j\}$ using binning, re-weighting each bin to match $R \circ \hat{F}_{Z|x}$, then re-sampling from each bin according to the adjusted weights. The full algorithm is summarized in Algorithms 4 and 5: Algorithm 4 describes the procedure for learning the recalibration map R, and Algorithm 5 describes the test time sampling procedure. We provide more details on each of the steps in Section 4.D. Crucially, the corresponding $\{y_k\}$ are HDR calibrated.

Proposition 4. *Suppose that $\hat{Z} \sim R \circ \hat{F}_{Z|X}$ and that R is an invertible mapping. Then the distribution of $\hat{Y}$ is HDR calibrated. (proof in Section 4.A)*

4.3.2 Adjusting the Predictive PDF

The HDR recalibration algorithm from Section 4.3.1 produces a predictive distribution (via samples) s.t. the p-HDR contains p proportion of the target observations, on average, $\forall p \in (0, 1)$. However, this predictive distribution can still fail to address dependencies among the output dimensions. This is because, for any fixed x, the HDRs are constructed with *level sets* of $\hat{f}_{Y|x}$, and, if $\hat{f}_{Y|x}$ fails to model dependencies, then the recalibrated samples will also express independence among the output dimensions. We provide an illustration in Figure 4.2. The top row shows that the pre-hoc predictive distribution assumes independence in the output dimensions, which is reflected in the spherical boundaries of the HDRs. After HDR recalibration, the shape of the recalibrated distribution is still spherical, even though the calibration dataset (i.e. ground truth (GT) observations in blue) displays correlation among the dimensions.

This highlights the importance of the projection function $\hat{f}_{Y|X}$, and ideally, $\hat{f}_{Y|X}$ should better reflect the true distribution in order for the recalibration procedure to produce more accurate samples. Further, if we can estimate the errors in $\hat{f}_{Y|X}$ (e.g. correlation, bias) with a held-out dataset, it can be beneficial to adjust $\hat{f}_{Y|X}$ for these factors prior to recalibration.

As a concrete instantiation of this adjustment, we propose a simple procedure to adjust the PDF of multivariate Gaussian distributions by estimating the bias in the predicted mean (i.e. the *location* of the HDRs), standard deviation (i.e. the *width* of the HDRs in each dimension), and the correlation in output dimensions (i.e. the *shape* of the HDRs) with a held-out dataset and correcting the PDF for each of these aspects. We provide details on each adjustment in Section 4.D, and we suggest applying the composition of these adjustments prior to recalibration, as indicated with the `ADJUST` step in Line 2 of Algorithms 4 and 5. The bottom row of Figure 4.2 provides an illustration of the mean adjustment and correlation adjustment. We can observe that the resulting recalibrated samples more closely reflect the ground truth distribution. In our experiments, we always apply the composition of adjustments, and we provide an ablation study of performing HDR recalibration with and without adjustments in Section 4.E.5.

Lastly, we note that the `ADJUST` steps in Line 2 of Algorithms 4 and 5 are meant to be a general procedure that depends on the type of predictive distribution used, and the adjustments provided in Algorithms 8, 9, and 10 in Section 4.D are examples that are specific to the case when the predictive distribution is Gaussian. When either the predictive or ground truth distributions are complex, these specific adjustment steps may be insufficient to adequately model the relationships among the output dimensions, and more sophisticated adjustments may be necessary.

4.4 Experiments

We demonstrate the efficacy of the proposed method on two sets of modeling tasks (Section 4.4.1) and one downstream decision-making task (Section 4.4.2). Across all experiments, we compare the performance of model predictions with no recalibration (i.e. pre-hoc), with SD recalibration, and with HDR recalibration.

4.4.1 Modeling Tasks

The two modeling tasks are comprised of 1) a suite of benchmark regression datasets and 2) a real-world dataset from the physical sciences – modeling plasma dynamics in a nuclear fusion device called a tokamak.

	Pre-hoc			SD Recalibration			HDR Recalibration		
Dataset	Energy	HDR-ECE	SD-ECE	Energy	HDR-ECE	SD-ECE	Energy	HDR-ECE	SD-ECE
scpf	3.97 (0.37)	0.30 (0.00)	0.15 (0.00)	4.00 (0.37)	**0.04 (0.00)**	**0.02 (0.00)**	**−0.79 (0.42)**	0.07 (0.01)	0.17 (0.00)
rf1	0.11 (0.01)	0.08 (0.00)	0.05 (0.00)	0.11 (0.01)	0.03 (0.00)	**0.01 (0.00)**	**0.08 (0.01)**	**0.01 (0.00)**	0.05 (0.00)
rf2	1.06 (0.28)	0.07 (0.00)	0.05 (0.00)	1.06 (0.28)	**0.04 (0.00)**	**0.01 (0.00)**	**1.04 (0.28)**	0.09 (0.01)	0.06 (0.00)
scm1d	1.13 (0.00)	0.48 (0.00)	0.11 (0.00)	1.13 (0.00)	0.24 (0.00)	0.02 (0.00)	**0.36 (0.05)**	**0.04 (0.00)**	**0.01 (0.00)**
scm20d	1.29 (0.01)	0.48 (0.00)	0.11 (0.00)	1.31 (0.01)	0.25 (0.00)	**0.02 (0.00)**	**0.81 (0.09)**	**0.04 (0.00)**	**0.02 (0.00)**
Fusion1	2.48 (0.03)	0.34 (0.00)	0.11 (0.00)	2.48 (0.03)	0.14 (0.00)	**0.05 (0.00)**	**−3.73 (0.14)**	**0.13 (0.00)**	0.06 (0.00)
Fusion2	1.95 (0.01)	0.45 (0.00)	0.17 (0.00)	1.93 (0.01)	0.31 (0.00)	0.09 (0.00)	**−1.85 (0.05)**	**0.05 (0.00)**	**0.01 (0.00)**
Fusion3	4.79 (0.07)	0.35 (0.00)	0.09 (0.00)	5.03 (0.08)	0.17 (0.00)	**0.03 (0.00)**	**−5.01 (0.39)**	**0.09 (0.00)**	0.05 (0.00)

Table 4.1: Results from multi-dimensional regression and recalibration experiments. The mean is shown with 1 standard error in parentheses (0.00 indicates that the values were smaller than 2 decimal places). The lowest mean value for each metric is bolded. **(Top)** Results from the benchmark experiments. **(Bottom)** Results from the nuclear fusion dynamics model experiments.

Datasets. The "mulan" benchmark [Tsoumakas et al., 2011] is a set of prediction tasks with multi-dimensional targets of up to 16 dimensions. Among these tasks, we take regression datasets with at least 1000 datapoints, which result in the following 5 datasets: **scpf** (3D), **rf1** (8D), **rf2** (8D), **scm1d** (16D), **scm20d** (16D). On each dataset, we make train-validation-test splits of proportions $[65\%, 20\%, 15\%]$, and use the train set to learn a probabilistic neural network (PNN), which is a neural network that predicts a multivariate Gaussian distribution with a diagonal covariance matrix. Section 4.E.2 provides the full set of details on the experiment setup for this benchmark experiment.

In the nuclear fusion experiment, we take three different pre-trained dynamics models of plasma evolution during nuclear fusion reactions as the pre-hoc models. These models were learned from recorded data of nuclear fusion experiments conducted on the DIII-D tokamak [Luxon, 2002], a magnetic confinement nuclear fusion device. Controlling these devices is meticulously difficult, and these dynamics models were used to optimize model-based control policies for deployment on DIII-D. Each model takes in the current plasma state and tokamak actuators, then predicts a multi-dimensional predictive distribution over several key plasma state variables for the next time step. All three models predict a multivariate Gaussian distribution with a diagonal covariance matrix. Two of the models (**Fusion1** and **Fusion2**) predict a 3-dimensional state target, and the third model, (**Fusion3**) predicts one additional state variable to predict a 4-dimensional target. Section 4.E.3 provides the full set of details on the experiment setup for the nuclear fusion experiment.

Evaluation. We perform evaluations for both the benchmark and fusion modeling tasks as follows. For every test input x_i, samples are drawn from the predictive distribution, and we denote this set of samples as $\mathcal{S}_i = \{\hat{y}_j\}_{j=1}^{M}$. We report evaluation metrics based on the predictions, $\mathcal{S}_i$, and the true target datapoint, y_i, for each (x_i, y_i) in the test set. When applying recalibration (SD or HDR), we use the validation set to learn the recalibration mapping R and apply the mapping in producing $\mathcal{S}_i$. Sections 4.E.2 and 4.E.3 provides full details on the evaluation procedure for each set of experiments.

Metrics. As evaluation metrics, we report one proper scoring rule and two measures of calibration. Proper scoring rules [Gneiting and Raftery, 2007] are summary statistics of overall performance of a distributional prediction, and they serve as both an optimization objective as well as evaluation metrics. Because we represent the predictive distribution via *samples*, we use the energy score as our core evaluation metric. The energy score is defined in terms of expectations w.r.t the predictive distribution and hence, is amenable to estimation with samples.

Given a test datapoint (x_i, y_i) and the predictive distribution at x_i, $\hat{f}_{Y|x_i}$, the (negatively-oriented) energy score, ES($\hat{f}_{Y|x_i}, y_i$), is defined as

$$\text{ES}(\hat{f}_{Y|x_i}, y_i) = \mathbb{E}_{\hat{f}_{Y|x_i}} \left\| \hat{Y} - y_i \right\|_2^\beta - \frac{1}{2} \mathbb{E}_{\hat{f}_{Y|x_i}} \left\| \hat{Y} - \hat{Y}' \right\|_2^\beta,$$

where each of $\hat{Y}$ and $\hat{Y}'$ are independent r.v.s that are both distributed $\sim \hat{f}_{Y|x_i}$, and β is a hyperparameter $\in (0, 2)$.

Additionally, we report two measures of calibration: "HDR Expected Calibration Error (HDR-ECE)" and "single dimensional ECE (SD-ECE)". We estimate both metrics by first drawing K probability values: $0 \leq p_1 < p_2 \cdots < p_K \leq 1$ as the predicted probabilities.

HDR-ECE is computed as ECE (Eq. 4.11) using the notion of HDR calibration, and this produces one value for the full joint predictive distribution:

$$\widehat{\text{HDR-ECE}} = \frac{1}{K} \sum_{k=1}^{K} |\hat{p}_k - p_k|,$$

where $\hat{p}_k$ is an estimate of the empirical probability term, $P(\hat{F}_{Z|X}(Z) \leq p_k)$.

To compute SD-ECE, we first compute ECE using the notion of average calibration (Eq. 8.1) for each output dimension separately: $\widehat{\text{SD-ECE}}^d$, $d \in [D]$. Since this produces D values, and we take the average to summarize SD-ECE as a single scalar:

$$\widehat{\text{SD-ECE}} = \frac{1}{D} \sum_{d=1}^{D} \widehat{\text{SD-ECE}}^d$$

We point out that SD-ECE is simply a point of reference since it computes miscalibration as the sum of calibration error from each output dimension. Thus, this metric may not provide an accurate representation of miscalibration of the full joint distribution and may display pathologies, as described in Figure 4.1.

We refer the reader to Section 4.E.1 for the full set of details on how each metric is estimated. Lastly, note that all three metrics (Energy score, HDR-ECE, SD-ECE) are negatively oriented, i.e. lower values are more desirable.

Results. Table 4.1 provides results on both the benchmark (Top) and nuclear fusion tasks (Bottom). We see that across all 5 benchmark datasets and the 3 nuclear fusion tasks, the energy score indicates that the samples produced by HDR recalibration are the highest quality and has best approximated the ground truth distribution. HDR recalibration also improves HDR-ECE compared to the pre-hoc model on 4 out of 5 benchmark datasets and on all three fusion tasks, which is expected given HDR recalibration aims to minimize HDR-ECE. Likewise, SD recalibration aims to minimize SD-ECE, which it achieves on 4 out of 5 benchmark datasets and 2 out of 3 fusion tasks. However, the fact that SD recalibration does not improve the energy score further supports the argument that SD-ECE is *not an adequate metric* for assessing multi-dimensional predictions.

4.4.2 Decision-making with Demand Forecasts

We apply the proposed recalibration algorithm in a decision-making setting, where a decision-maker (in this case a grocery store manager) must forecast future demand for store items and stock the items accordingly. Similar to the inventory management experiments in Kuleshov et al. [2018], Malik et al. [2019], we take the "Corporacion Favorita" Kaggle dataset [Favorita et al., 2017], which records the historical sales of items from a supermarket chain in Ecuador. We take the top three most sold items between the dates 2015-01-01 to 2017-08-11 and set up the modeling problem s.t. the grocery store forecasts the demand for the three products in the next business day, given the recent four day history of the sales and variables that indicate the day of the week and week of the year. Given that sales of items in a store may display dependence (e.g. seasonality or cannibalization), modeling the dependencies across the three items will be important.

We set up the decision-making problem s.t. the grocery store attributes very high loss to under-stocking (e.g. loss of reputation and future demand) and also incurs a small loss for over-stocking (e.g. possible spoilage and waste):

$$\text{Loss}_t = 10 * Q_{t,\text{under-stock}} + 1 * Q_{t,\text{over-stock}},$$

where Loss_t denotes the loss on day t, $Q_{t,\text{under-stock}}$ denotes the quantity of under-stocked items on day t, and $Q_{t,\text{over-stock}}$ denotes the quantity of over-stocked items on day t. The demand forecasts are generated as samples from the predictive distribution (i.e. possible realizations of the next day demand), and we use a cautious decision policy that makes the decisions based on the sample that forecasts the highest demand. With a total of 795 days in the dataset, we train the probabilistic model with the first 559 days while using the subsequent 159 days as the validation set, and we use this same validation set for recalibration. With the remaining 77 days, we simulate the decision-making problem and record the accumulated loss. We repeat the simulation across 5 different seeds and report the average loss and standard error. We refer the reader to Section 4.E.4 for more details on the experiment setup.

	Pre-hoc	**SD Recal**	**HDR Recal**
Loss	916.81 (4.16)	910.82 (5.19)	**865.34** (7.83)

Table 4.2: Total loss incurred by each method in sales simulation experiment. The mean loss is shown with 1 standard error in parentheses.

Table 4.2 displays the accumulated loss based on each method. While SD recalibration marginally outperforms the distribution with no recalibration (Pre-hoc), HDR recalibration significantly improves the loss, demonstrating that the joint calibration of predictions provides utility in this decision-making setting.

4.5 Discussion

In this work, we addressed the problem of recalibrating multi-dimensional distributional predictions. Prior recalibration methods consider each target dimension separately and fail to take into account dependencies that may exist in the dimensions. Bridging ideas in calibration of projections of multivariate targets, HPD values, and HDRs, we defined the notion of HDR calibration and proposed the HDR recalibration algorithm.

HDR calibration leverages the property that for any distributional prediction setting, the p-HDR of the optimal prediction will contain the true observations with empirical frequency p. As HDRs consider the full joint distribution, it is a more adequate notion for assessment of calibration of multi-dimensional distributional predictions. The proposed HDR recalibration algorithm aims to achieve HDR calibration by performing recalibration in the projected space, sampling from the recalibrated projection distribution, and mapping the projection samples back to the target space. This produces a sampling-based representation of the HDR calibrated distribution. Across the suite of benchmark multi-dimensional regression tasks, plasma dynamics prediction tasks, and decision-making task, HDR recalibration consistently improves the quality of the predictive samples compared to the baseline methods.

We note that HDR calibration is a specific instance of calibration in the projected space with the predictive PDF as the projection function. Other projection functions which capture various aspects of the data distribution can also be used, and we leave for future work exploring such projections.

Appendices for Chapter 4

4.A Theoretical Statements

Proposition 1 The optimal distributional prediction, i.e. $\hat{F}_{Y|X} = F_{Y|X}$, satisfies calibration in the projected space, Eq. 4.3.

Proof. Recall the definition of calibration in the projected space, which we restate here:

$$P(Z \leq \hat{F}^{-1}_{Z|X}(p)) = p, \forall p \in (0,1), \tag{4.12}$$

where $Z \coloneqq g(X,Y), X \sim f_X, Y \sim f_{Y|X}$ and $\hat{F}_{Z|X}$ is the CDF of the r.v. $\hat{Z}|X$.

For any $p \in (0,1)$,

$$P(Z \leq \hat{F}^{-1}_{Z|X}(p)) \tag{4.13}$$
$$= P(\hat{F}_{Z|X}(Z) \leq p) \tag{4.14}$$
$$= \int_{\mathcal{X}} P\left(\hat{F}_{Z|x}(Z) \leq p \mid X = x\right) dF_X(x) \tag{4.15}$$

By the condition of the statement, we have $\hat{F}_{Y|X} = F_{Y|X}$, thus $\hat{F}_{Y|x} = F_{Y|x}$.

$$\hat{F}_{Y|x} = F_{Y|x} \tag{4.16}$$
$$\Rightarrow \hat{Y}|x \overset{d}{=} Y|x \qquad (\hat{Y}|x \sim \hat{F}_{Y|x}, Y|x \sim F_{Y|x}, \text{ and “} \overset{d}{=} \text{”denotes “equal in distribution"}) \tag{4.17}$$
$$\Rightarrow g(x,\hat{Y})|x \overset{d}{=} g(x,Y)|x \tag{4.18}$$
$$\Rightarrow \hat{Z}|x \overset{d}{=} Z|x \qquad (\hat{Z}|x \coloneqq g(x,\hat{Y})|x \text{ and } Z|x \coloneqq g(x,Y)|x) \tag{4.19}$$
$$\Rightarrow \hat{F}_{Z|x} = F_{Z|x} \qquad (\hat{Z}|x \sim \hat{F}_{Z|x} \text{ and } Z|x \sim F_{Z|x}) \tag{4.20}$$

Then, by the probability integral transform, $\hat{F}_{Z|x}(Z|x) \sim \mathcal{U}(0,1)$ and $P\left(\hat{F}_{Z|x}(Z) \leq p \mid X = x\right) = p$.

Thus we have

$$\begin{aligned}
&\int_{\mathcal{X}} P\left(\hat{F}_{Z|x}(Z) \leq p \mid X = x\right) dF_X(x) && (4.21)\\
&= \int_{\mathcal{X}} p dF_X(x) && (4.22)\\
&= \int_{\mathcal{X}} p f_X(x) dx && (4.23)\\
&= p \int_{\mathcal{X}} f_X(x) dx && (4.24)\\
&= p. && (4.25)
\end{aligned}$$

□

Proposition 2 HDR calibration holds if and only if Equation 4.3 holds.

Proof. We first prove “HDR calibration holds” $\implies$ Equation 4.3.

For any given $p \in (0, 1)$.

$$\begin{aligned}
&P(Y \in \texttt{HDR}_X(p)) && (4.26)\\
&= P(\texttt{HPD}_X(Y) \leq p) && (4.27)\\
&= \int_{\mathcal{X}} P(\texttt{HPD}_x(Y) \leq p \mid X = x) dF_X(x) && (4.28)\\
&= \int_{\mathcal{X}} \int_{\mathcal{Y}} \mathbb{I}\{\texttt{HPD}_x(y) \leq p\} dF_{Y|x}(y) dF_X(x) && (4.29)\\
&= \int_{\mathcal{X}} \int_{\mathcal{Y}} \mathbb{I}\{1 - \hat{F}_{Z|x}(\hat{f}_{Y|x}(y)) \leq p\} dF_{Y|x}(y) dF_X(x) && (4.30)\\
&= \int_{\mathcal{X}} \int_{\mathcal{Y}} \mathbb{I}\{1 - p \leq \hat{F}_{Z|x}(\hat{f}_{Y|x}(y))\} dF_{Y|x}(y) dF_X(x) && (4.31)\\
&= \int_{\mathcal{X}} P(1 - p \leq \hat{F}_{Z|x}(\hat{f}_{Y|x}(Y)) \mid X = x) dF_X(x) && (4.32)\\
&= P(1 - p \leq \hat{F}_{Z|X}(\hat{f}_{Y|X}(Y))) && (4.33)\\
&= 1 - P(\hat{F}_{Z|X}(\hat{f}_{Y|X}(Y)) \leq 1 - p) && (4.34)\\
&= p \quad \text{(by condition that HDR calibration holds)} && (4.35)
\end{aligned}$$

Thus, $\forall p \in (0, 1)$,

$$\begin{aligned}
&1 - P(\hat{F}_{Z|X}(\hat{f}_{Y|X}(Y)) \leq 1 - p) = p && (4.36)\\
&\iff P(\hat{F}_{Z|X}(\hat{f}_{Y|X}(Y)) \leq 1 - p) = 1 - p && (4.37)\\
&\iff P(\hat{F}_{Z|X}(\hat{f}_{Y|X}(Y)) \leq p) = p \quad \text{(Equation 4.3)}, && (4.38)
\end{aligned}$$

proving that “HDR calibration holds” $\implies$ Equation 4.3.

We can reverse all of the steps to prove the other direction, which completes the proof.

□

Proposition 3 Consider $R \circ \hat{F}_{Z|X}$ for an invertible mapping R. Then $R \circ \hat{F}_{Z|X}$ satisfies Eq. 4.3, i.e.

$$P(R \circ \hat{F}_{Z|X}(Z) \leq p) = p \quad \forall p \in (0,1).$$

Proof. For any fixed $p \in (0,1)$, let $q = R^{-1}(p)$.

Note the following equality

$$\begin{aligned} & P(\hat{F}_{Z|X}(Z) \leq p) && (4.39)\\ & = \int_{\mathcal{X}} P(\hat{F}_Z(Z|x) \leq p \mid X = x) dF(x). && (4.40) \end{aligned}$$

Applying R to $\hat{F}_{Z|X}$, we have

$$\begin{aligned} & P(R \circ \hat{F}_{Z|X}(Z) \leq p) && (4.41)\\ & \int_{\mathcal{X}} P(R \circ \hat{F}_{Z|x}(Z) \leq p \mid X = x) dF(x) && (4.42)\\ & = \int_{\mathcal{X}} P(\hat{F}_{Z|x}(Z) \leq R^{-1}(p) \mid X = x) dF(x) && (4.43)\\ & = \int_{\mathcal{X}} P(\hat{F}_{Z|x}(Z) \leq q \mid X = x) dF(x) && (4.44)\\ & = P(\hat{F}_{Z|X}(Z) \leq q) && (4.45)\\ & = p \quad \text{(by definition of } R(q)) && (4.46) \end{aligned}$$

□

Proposition 4 Suppose that $\hat{Z} \sim R \circ \hat{F}_{Z|X}$ and that R is an invertible mapping. Then the distribution of $\hat{Y}$ is HDR calibrated.

Proof. To first clarify the notation in the proposition statement, in writing $\hat{Z} \sim R \circ \hat{F}_{Z|X}$, the notation for $\hat{Z}$ has been overloaded as in the main text, we have stated that $\hat{Z}$ is the r.v. of the distribution function $\hat{F}_{Z|X}$.

Following the context of Section 4.3.1, the $\hat{Y}$ and $\hat{Z}$ in this proposition statement should be taken as any arbitrary r.v. that is generated as follows: $\hat{Y}$ is an arbitrary r.v. in $\mathcal{Y}$, and $\hat{Z}$ is the corresponding r.v. in the projected space induced by the projection function $\hat{f}_{Y|X}$ and the r.v. $\hat{Y}$.

By Proposition 3, we know that if $\hat{Z} \sim R \circ \hat{F}_{Z|x}$, then Eq. 4.3 holds.

By Proposition 2, we know that Eq. 4.3 is equivalent to HDR calibration, i.e. the distribution function of $\hat{Y}$ satisfies HDR calibration.

□

4.B Additional Details on Related Works

Izbicki et al. [2022] proposes "HPD-split" as a conformal prediction method that utilizes HPD values (Eq. 4.4) as the split residuals, and simply by the method of split conformal prediction, given a fixed level $\alpha \in (0,1)$, the prediction set produced by HPD-split is guaranteed at least $1-\alpha$ coverage on average over the data distribution, i.e. marginal validity (Definition 1, Izbicki et al. [2022]) holds.

As discussed in Section 4.2.4, while their method does not consider multi-dimensional target spaces, because of the intrinsic relationship between HPD values and HDRs (Eq. 4.8), it is interesting to consider the relationship between the prediction set produced by HPD-split, HDR calibration (Defn. 1), and our proposed recalibration algorithm (Algorithms 4, 5).

Since HDR calibration is a notion of calibration defined for a predictive *distribution* function $\hat{f}, \hat{F}$, strictly speaking, one cannot state that the prediction *set* by HPD-split is HDR calibrated since there is no notion of a probability distribution function: given a fixed $p \in (0,1)$, HPD-split produces a set $\mathcal{S} \subseteq \mathcal{Y}$ s.t. $P(Y \in \mathcal{S}) \geq p$, but following Definition 1, one cannot construct $\texttt{HDR}(p)$ as the concept of HDR is intrinsically tied to a distribution function. This is not particular to HPD-split, but a key feature of all conformal prediction methods that differentiates it from calibration (or recalibration) methods: conformal prediction methods output prediction *sets* for a specified α-level, while calibration methods output full predictive *distributions*.

However, conversely, one can construct prediction sets from predictive distributions, and we can show that a prediction set constructed from the HDR recalibration procedure satisfies marginal validity that holds for HPD-split, i.e. our recalibration procedure shares the conformal guarantees of HPD-split.

Proof. Assume an invertible interpolation algorithm for the recalibration mapping R in Algorithm 4 s.t. $\forall \alpha \in \{\frac{1}{N}, \frac{2}{N}, \dots \frac{N-1}{N}, 1\}$, $R^{-1}(\alpha) = c_{(N\alpha)}$ (Recall that $c_{(i)}$ is the ith order statistic of the recalibration dataset $\mathcal{C}'$ in Algorithm 4). Assume a fixed level $\alpha \in \{\frac{1}{N}, \frac{2}{N}, \dots \frac{N-1}{N}, 1\}$.

Recall that the CDF of the recalibrated distribution in $\mathcal{Z}$ space is $R \circ \hat{F}_{Z|X}$ (Section 4.3.1). Given a test input point x, consider constructing a 1-sided prediction interval in $\mathcal{Z}$ with expected coverage equal to $1-\alpha$, i.e. $\{z : R \circ \hat{F}_{Z|x}(z) \geq \alpha)\}$.

$$\begin{aligned}
&\{z : R \circ \hat{F}_{Z|x}(z) \geq \alpha)\} \\
&= \{z : \hat{F}_{Z|x}(z) \geq R^{-1}(\alpha)\} \\
&= \{z : \hat{F}_{Z|x}(z) \geq c_{(N\alpha)}\}
\end{aligned}$$

Following the definition $Z := \hat{f}_{Y|X}(Y)$, we have that the pre-image of this set in $\mathcal{Y}$ space is the following set

$$\mathcal{C}(x) = \{y : \hat{F}_{Z|x}(\hat{f}_{Y|x}(y)) \geq c_{(N\alpha)}\}$$

Note that this set is identical to the conformal prediction set in Definition 15 of Izbicki et al. [2022], and following their Theorem 20, this set satisfies marginal validity, i.e.

$$P(Y \in C(X)) \geq 1 - \alpha$$

□

Feldman et al. [2023] is another conformal prediction method, but which aims to produce prediction sets in multi-dimensional target spaces. Their method relies on training a VAE on the dataset to learn a representation that is amenable to performing quantile regression. In addition to the point that it is a conformal method which produces prediction sets instead of distributions, we note that their work is somewhat orthogonal to our setting as it is a *pre-hoc* method, whereas we are focused on *post-hoc* methods which can be applied on top of pre-hoc trained models (as discussed in Section 4.2.4).

4.C Additional Definitions

Average Calibration. We provide additional notes on average calibration (Eq. 8.1), which we restate here:

$$P(Y \leq \hat{F}^{-1}_{Y|X}(p)) = p, \forall p \in [0, 1].$$

We can rewrite this expression by conditioning on $X = x$ and applying the law of total probability:

$$P(Y \leq \hat{F}^{-1}_{Y|X}(p)) = \mathbb{E}_{x \sim f_X} \left[P\left(Y \leq \hat{F}^{-1}_{Y|x}(p) \middle| X = x \right) \right].$$

Copula Calibration and Kendall Distribution. We first re-state copula calibration using our notation.

Copula calibration requires Eq. 4.3 with $Z := \hat{F}_{Y|X}(Y)$, $\hat{Z} := \hat{F}_{Y|X}(\hat{Y})$, and $\hat{F}_{Z|X} : [0, 1] \to [0, 1]$ is the Kendall distribution: $\hat{F}_{Z|X}(p) = P(\hat{Z} \leq p)$.

Utilizing the notation from Ziegel and Gneiting [2014] the Kendall distribution of any CDF F is denoted as $\mathcal{K}_F : [0, 1] \to [0, 1]$, where $\mathcal{K}_F(p) = P(F(Y) \leq p)$ for $p \in [0, 1]$, and copula calibration is defined as

$$P(\mathcal{K}_{\hat{F}_X}[\hat{F}_X(Y)] \leq p) = p, \forall p \in [0, 1].$$

Highest Density Region Hyndman [1996] defines highest density regions as follows. Given a coverage level p, the p-HDR of the predictive PDF $\hat{f}_{Y|x}$, $R_{\hat{f}_{Y|x}}(p)$ is defined as

$$R_{\hat{f}_{Y|x}}(p) = \{y : \hat{f}_{Y|x}(y) \geq \hat{f}_p\}, \tag{4.47}$$

$$\text{where } \hat{f}_p = \arg\sup_{\hat{f}_p} \{P(\hat{Y} \in R_{\hat{f}_{Y|x}}(p) \mid X = x) \geq p\}. \tag{4.48}$$

We note that this definition, which is stated here nearly verbatim from the original definitions from Hyndman [1996], is recursive. To clarify, $R_{\hat{f}_{Y|x}}(p)$ is the prediction set which has the highest density values w.r.t. $\hat{f}_{Y|x}$ and which has p integrated probability. We refer the reader to Figure 1 of Hyndman [1996] or Figure 2 of Zhao et al. [2021a] for helpful visualizations.

4.D Additional Details on Algorithms

4.D.1 Subroutine Algorithms

This section provides details on the algorithms that are used as subroutines of the main algorithms (Algorithms 4 and 5 from Section 4.3).

Algorithm 6 is used in Algorithm 4 to construct the recalibration mapping dataset.

Algorithm 7 is used in Algorithm 5 to re-sample from a set of predictive samples to construct a set of recalibrated samples. We point out that Algorithm 7 provides an implementation of using binning to apply the recalibration mapping R during re-sampling. However, there can be other ways of applying R during re-sampling (e.g. isotonic regression).

Algorithms 8, 9, 10 comprise the PDF adjustment step described in Section 4.3.2, which is used in Line 2 of both Algorithms 4 and 5. While the PDF adjustment step is meant to be a general procedure to correct for biases or correlations that are evident based on the predictions on a held-out dataset, because diagonal Gaussians are commonly used for multi-dimensional distributional predictions, we provide an instantiation of the adjustment procedure specifically for diagonal Gaussian distributions. In our experiments, we used a composition of all three algorithms in order - i.e. the PDF was first adjusted for the mean, then the standard deviation, and lastly the covariance. We note that Algorithm 9 requires a choice of loss function defined for univariate Gaussians, and in our experiments, we used SD-ECE (single dimensional expected calibration error) as the loss function.

Lastly, note that in practice, Algorithms 8, 9, 10 are actually run only once during Training (Line 2 of Algorithm 4). During testing, the learned adjustment functions are simply applied to the distributional predictions (Line 2 of Algorithm 5).

Algorithm 6 Constructing the Recalibration Mapping Dataset

```
1: **Input**: Calibration dataset $\{(x_i, y_i)\}_{i=1}^N$, predictive PDF $\hat{f}_{Y|X}$, number of samples $M$.
2: $\mathcal{C} \leftarrow \varnothing$
3: **for** $i \in [N]$ **do**
4:     Draw $M$ samples from $\hat{f}_{Y|x_i}$ to construct $\{\hat{y}_{i,j}\}_{j=1}^M$ and apply $\hat{f}_{Y|x_i}$ to each sample to construct $\{\hat{z}_{i,j}\}_{j=1}^M$ where $\hat{z}_{i,j} = \hat{f}_{Y|x_i}(\hat{y}_{i,j})$
5:     $z_i \leftarrow \hat{f}_{Y|x_i}(y_i)$
6:     Estimate empirical CDF $\hat{F}_{Z|x_i}$ with $\{\hat{z}_{i,j}\}_{j=1}^M$ and evaluate $z_i$: $r_i = \hat{F}_{Z|x_i}(z_i)$
7:     $\mathcal{C} \leftarrow \mathcal{C} \cup \{r_i\}$
8: **end for**
9: **Output**: $\mathcal{C}$
```

Algorithm 7 Producing HDR Recalibrated Samples via Binning

```
1: **Input**: Dataset of tuples of predictive samples and projections $\mathcal{D} = \{(\hat{y}_j, \hat{z}_j)\}_{j=1}^M$, recalibration mapping $R$, number of bins $B$.
2: Place $\mathcal{D}$ into $B$ equal width bins $\mathcal{M} = \{b_i = [l_i, u_i)\}_{i=1}^B$ w.r.t $\{\hat{z}_j\}_{j=1}^M$, s.t. $\min_{\hat{z}_j}(\hat{y}_j, \hat{z}_j) \in b_1$, $\max_{\hat{z}_j}(\hat{y}_j, \hat{z}_j) \in b_B$, and the number of elements in each bin $|b_i| = \lfloor \frac{M}{B} \rfloor$.
3: Recalibrated samples $\mathcal{D}' \leftarrow \varnothing$
4: **for** $i \in [B]$ **do**
5:     Sampling rate for bin $s_i = R\left(i * \lfloor \frac{M}{B} \rfloor / M\right) - R\left((i-1) * \lfloor \frac{M}{B} \rfloor / M\right)$.
6:     Sample from $b_i$ with probability $s_i$ to construct the dataset $\{(\hat{y}_k, \hat{z}_k)\}_{k=1}^{K_i}$ where $K_i = \lfloor (s_i * \lfloor \frac{M}{B} \rfloor) \rfloor$
7:     $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{\hat{y}_k\}_{k=1}^{K_i}$.
8: **end for**
9: **Output**: $\mathcal{D}'$.
```

Algorithm 8 Mean Adjustment for Gaussian Distributions

```
1: **Input**: Calibration dataset $\{(x_i, y_i)\}_{i=1}^N$, predictive Gaussian PDF $\hat{f}_{Y|X} := (\hat{\mu}_X, \hat{\sigma}_X)$.
2: Predict the conditional mean at each $x_i$, $_i$, and compute the bias: $\texttt{bias} = \frac{1}{N} \sum_{i=1}^N (y_i - _i)$.
3: Define the mean adjustment function: $\mathcal{A}((,)) = (+\texttt{bias},)$
4: **Output**: Mean adjusted Gaussian distribution $\mathcal{A}((_X, _X))$.
```

Algorithm 9 Standard Deviation Adjustment for Diagonal Gaussian Distributions

```
1: Input: Calibration dataset $\{(x_i, y_i)\}_{i=1}^N$, predictive Gaussian PDF $\hat{f}_{Y|X} \coloneqq (\hat{\mu}_X, \hat{\sigma}_X)$, loss
   function for univariate Gaussian distributions : $(\mu \times \sigma, \mathcal{Y}) \to \mathbb{R}$
2: Predict the conditional mean and standard deviation at each $x_i$: $(_i, _i)$.
3: Optimized standard deviation ratios $\mathcal{S} \leftarrow$ [ ].
4: for $d \in [D]$ do
5:     $c^d = \arg\min_{c \in \mathbb{R}_+} \frac{1}{N} \sum_{i=1}^N \mathcal{L}\left((_i^d, c*_i^d), y_i^d\right)$.
6:     Append $c^d$ to $\mathcal{S}$.
7: end for
8: Concatenate $\mathcal{S}$ into a vector and denote it $\hat{s} \in \mathbb{R}^D$.
9: Define the standard deviation adjustment function: $\mathcal{A}((,)) = (, \hat{s} \odot)$, where $\odot$ indicates
   element-wise product.
10: Output: Standard deviation adjusted Gaussian distribution $\mathcal{A}((_X, _X))$.
```

Algorithm 10 Covariance Adjustment for Diagonal Gaussian Distributions

```
1: Input: Calibration dataset $\{(x_i, y_i)\}_{i=1}^N$, predictive Gaussian PDF $\hat{f}_{Y|X} \coloneqq (\hat{\mu}_X, \hat{\sigma}_X)$.
2: Predict the conditional mean and standard deviation at each $x_i$: $(_i, _i)$.
3: Compute the error in mean prediction $\{\epsilon_i\}_{i=1}^N$, where $\epsilon_i = y_i - _i$.
4: Compute the error correlation matrix from $\{\epsilon_i\}_{i=1}^N$: $\hat{\rho} \in \mathbb{R}^{D \times D}$.
5: Define the covariance adjustment function, $\mathcal{A}((,)) = (, \hat{\Sigma}(, \hat{\rho}))$, where $\hat{\Sigma}(, \hat{\rho}) = \mathrm{Diag}() \cdot \hat{\rho} \cdot$
   $\mathrm{Diag}()$, $\mathrm{Diag}()$ denotes the $D \times D$ diagonal matrix with elements of  in the diagonal, and $\cdot$ is
   the standard matrix-matrix product operation.
6: Output: Covariance adjusted Gaussian distribution $\mathcal{A}((_X, _X))$.
```

4.D.2 Algorithm Analysis

We provide an analysis of the computational complexity of the main algorithms, Algorithms 4 and 5. Because the `ADJUST` step (Line 2) is an auxiliary procedure, we first analyze the run time of both algorithms excluding this step.

Algorithm 4

- Line 3: When constructing the dataset $\mathcal{C}$, computing the value $\hat{F}_{Z|x_i}(z_i)$ involves drawing M samples from $\hat{f}_{Y|x_i}$ and sorting M values of z_i, hence takes $O(M \log M)$ time for each $i \in [N]$, and thus takes $O(NM \log M)$ time.
- Line 4: Sorting N values takes $O(N \log N)$ time.
- Line 5: Learning R depends on the algorithm used. If one uses binning, this step takes no additional time since the bins are already defined by $\mathcal{C}'$.

Therefore, the whole procedure takes $O(NM \log M)$ time, where N is the number of datapoints in the calibration dataset, and M is the number of samples drawn to estimate the empirical CDF $\hat{F}_{Z|X}$.

Algorithm 5

- Line 3: Drawing M samples and applying $\hat{f}_{Y|x}$ on each sample takes $O(M)$ time.
- Line 4: Following Algorithm 7:
  - Line 2 of Algorithm 7: The most expensive step is sorting M values, which takes $O(M \log M)$ time.
  - Lines 4-8 for-loop of Algorithm 7: In each iteration, we re-sample from each bin of $\lfloor \frac{M}{B} \rfloor$ points, which takes $O(M/B)$ time. Since there are B iterations, the whole for-loop takes $O(M)$ time.

Therefore, the whole procedure takes $O(M \log M)$ time, where M is the number of samples drawn to express the predictive distribution $\hat{f}_{Y|x}$.

PDF Adjustment Step (`ADJUST`)

- Algorithm 8 takes $O(N)$ time.
- The run time of Algorithm 9 depends on the optimization algorithm used in Line 5, but since the optimization is repeated for D dimensions with a dataset of N points, it takes at least $\Omega(ND)$ time.
- In Algorithm 10, correlation estimation (Line 4) takes $O(ND^2)$ time, and the covariance matrix estimation involves two matrix multiplication of size $D \times D$, which takes $O(D^3)$ time, hence the overall complexity is $O(ND^2 + D^3)$.

4.E Additional Details on Experiments

4.E.1 Evaluation Metrics

Energy score Given a test datapoint (x_i, y_i) and the predictive distribution at x_i, $\hat{f}_{Y|x_i}$, the (negatively-oriented) energy score, ES$(\hat{f}_{Y|x_i}, y_i)$, is defined as

$$\text{ES}(\hat{f}_{Y|x_i}, y_i) = \mathbb{E}_{\hat{f}_{Y|x_i}} \left\| \hat{Y} - y_i \right\|_2^{\beta} - \frac{1}{2}\mathbb{E}_{\hat{f}_{Y|x_i}} \left\| \hat{Y} - \hat{Y}' \right\|_2^{\beta},$$

where each of $\hat{Y}$ and $\hat{Y}'$ are independent r.v.'s following the distribution $\hat{f}_{Y|x_i}$, and β is a hyperparameter $\in (0, 2)$. We estimate this score with

$$\widehat{\text{ES}}(\hat{f}_{Y|x_i}, y_i) = \frac{1}{|\mathcal{S}|}\sum_{\hat{y}\in\mathcal{S}} \|\hat{y} - y_i\|_2^{\beta} - \frac{1}{2|\mathcal{S}'||\mathcal{S}''|}\sum_{\hat{y}'\in\mathcal{S}', \hat{y}''\in\mathcal{S}''} \|\hat{y}' - \hat{y}''\|_2^{\beta}$$

by drawing finite sets of samples $\mathcal{S}, \mathcal{S}'$, and $\mathcal{S}''$ independently from the distribution $\hat{f}_{Y|x_i}$. The exact number of samples drawn for each of $\mathcal{S}, \mathcal{S}', \mathcal{S}''$ is provided in the following sections on each of the experiments. We set $\beta = 1.7$ for all of our experiments, but other values also result in similar trends as reported.

Given the test set, $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we report the mean energy score over $\mathcal{D}$:

$$\widehat{ES} = \frac{1}{N}\sum_{i=1}^{N} \widehat{\text{ES}}(\hat{f}_{Y|x_i}, y_i).$$

HDR-ECE HDR-ECE (highest density region expected calibration error) is computed following the equation for expected calibration error (ECE) in Eq. 4.11 with the notion of HDR calibration, specifically Eq. 4.3, which is equivalent to Definition 1 by Proposition 2. We estimate HDR-ECE with the test set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ as follows. At a test input x_i, since we do not know the closed form of $\hat{F}_{Z|x_i}$, we estimate $\hat{F}_{Z|x_i}(z_i)$ by first drawing a set of samples from $\hat{f}_{Y|x_i}$: $\{\hat{y}_j\}_{j=1}^M$, then applying $\hat{f}_{Y|x_i}$ to each sample to construct $\mathcal{S}_i = \{\hat{z}_j = \hat{f}_{Y|x_i}(\hat{y}_j)\}_{j=1}^M \sim \hat{F}_{Z|x_i}$, and finally produce the estimate for $\hat{F}_{Z|x_i}(z_i)$ as

$$p_i = \frac{1}{M}\sum_{j=1}^{M} \mathbb{I}\{\hat{z}_j \le z_i\} = \frac{1}{M}\sum_{j=1}^{M} \mathbb{I}\{\hat{f}_{Y|x_i}(\hat{y}_j) \le \hat{f}_{Y|x_i}(y_i)\}.$$

Then for any $p_k \in [0, 1]$ we estimate the empirical probability term, $P(\hat{F}_{Z|X}(Z) \le p_k)$, as

$$\hat{p}_k = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left\{p_i \le p_k\right\}.$$

Finally, we choose K probability values: $0 \le p_1 < p_2 \cdots < p_K \le 1$ and estimate HDR-ECE as the empirical estimate of Eq. 4.11:

$$\widehat{\text{HDR-ECE}} = \frac{1}{K}\sum_{k=1}^{K} |\hat{p}_k - p_k|.$$

The exact number of samples drawn to estimate $\hat{F}_{Z|x_i}$ is provided in the following sections on each of the experiments. For the probability values, we set $K = 99$ and used the following grid of probability values: $[0.01, 0.02, 0.03, \ldots 0.98, 0.99]$.

SD-ECE SD-ECE (single dimensional expected calibration error) is computed as the mean of ECE measured for each output dimension independently following the notion of univariate calibration (Eq. 8.1). For each dimension d, given a test input x_i, we first estimate $\hat{F}_{Y^d|x_i^d}(y_i^d)$ with a set of predictive samples $\mathcal{S}_i = \{\hat{y}_j\}_{j=1}^M \sim \hat{f}_{Y|x_i}$ as

$$p_i^d = \frac{1}{M}\sum_{j=1}^{M} \mathbb{I}\{\hat{y}_j^d \leq y_i^d\}.$$

Then for a given probability $p_k \in [0, 1]$, we estimate the empirical coverage term, $P(\hat{F}_{Y^d|X^d}(Y^d) \leq p_k)$, as

$$\hat{p}_k^d = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left\{p_i^d \leq p_k\right\}.$$

Thus, we estimate the "SD-ECE at dimension d" (SD-ECE$^{\mathrm{d}}$) as $\widehat{\text{SD-ECE}}^{\mathrm{d}} = \frac{1}{K}\sum_{k=1}^{K} |\hat{p}_k^d - p_k|$, and take the average of $\widehat{\text{SD-ECE}}^{\mathrm{d}}$ over all of the output dimensions to estimate SD-ECE:

$$\widehat{\text{SD-ECE}} = \frac{1}{D}\sum_{d=1}^{D} \widehat{\text{SD-ECE}}^{\mathrm{d}},$$

where D is the number of dimensions of Y.

Again, the exact number of samples drawn to estimate $\hat{F}_{Y|x_i}$ is provided in the following sections on each of the experiments. Just as with HDR-ECE, for the probability values, we set $K = 99$ and used the following grid of probability values: $[0.01, 0.02, 0.03, \ldots 0.98, 0.99]$.

Lastly, we note that calibration plots (a.k.a. reliability diagrams) in Figures 4.1 and 4.2 were produced with the *Uncertainty Toolbox* [Chung et al., 2021a].

4.E.2 Benchmark Regression Tasks

We use the following 5 datasets from the "mulan" benchmark [Tsoumakas et al., 2011]: **scpf** (3D), **rf1** (8D), **rf2** (8D), **scm1d** (16D), **scm20d** (16D). With each dataset, we make train-validation-test splits of proportions $[65\%, 20\%, 15\%]$, and train a probabilistic neural network (PNN) [Lakshminarayanan et al., 2017, Nix and Weigend, 1994] that predicts a diagonal Gaussian distribution.

For all of the datasets, the PNN trained has 5 fully connected layers, each with 200 hidden units, and the output parametrizes a diagonal Gaussian with a mean and a log-variance prediction. The Gaussian likelihood loss was used for training, with a learning rate of 0.001 and no weight decay was used. Training was halted early if the validation loss did not improve for more than 100 epochs, for a maximum of 1000 epochs. All of the models early-stopped their training.

After training a PNN on the train set, we learn the recalibration mapping on the validation split and produce 20 independent sets of recalibrated predictive samples on the test split (for methods *SD Recalibration* and *HDR Recalibration*). For the *Pre-hoc* method, samples were drawn from the PNN model without any recalibration. For **scpf**, each set contained 5000 samples of $\hat{y} \sim \hat{f}_{Y|x}$ at each test point x, for **rf1, rf2**, 8000 samples, and for **scm1d, scm20d**, 10000 samples. We compute each evaluation metric (Energy score, HDR-ECE, SD-ECE) on each of these sets of samples, then take the average for each metric across the 20 sets of samples. We then repeat this process (data splits, model training, recalibration and repeated sampling) with 5 different seeds, and report the mean and standard error of the metrics in Table 4.1 (Top).

4.E.3 Dynamics Modeling in Nuclear Fusion

We first provide some background information on the problem setup of modeling plasma dynamics for tokamaks. A tokmak is a device that magnetically confines a toroidal plasma, and it is one of the most promising devices for making nuclear fusion energy a reality. With the potential of providing an abundant source of safe and clean power generation, nuclear fusion, which is the physical process during which atomic nuclei combine together to form heavier atomic nuclei, is regarded as the power source of the future. However, fusion reactions are difficult to control, and recently, there has been increasing interest in both learning dynamics models [Boyer et al., 2021, Abbate et al., 2021] and applying those models for control of tokamaks [Mehta et al., 2021b, 2022, Char et al., 2023a, Seo et al., 2021, 2022]. In model-based control, a model of the system dynamics is learned and used e.g. to optimize a control policy offline, or the model is deployed online for model predictive control [Rawlings, 2000]. In either case, the learned model is sampled repeatedly to optimize a control sequence, hence obtaining well-calibrated samples which reflect the intricacies of the true system dynamics is crucial [Malik et al., 2019, Chua et al., 2018].

We take 3 different pre-trained dynamics models that were used to optimize control policies for deployment on the DIII-D tokamak, a nuclear fusion device in San Diego that is operated by General Atomics [Luxon, 2002]. All three models were trained with logged data from past experiments (referred to as "shots") on this device. As input, the models take in the current state of the plasma and the actuators from the tokamak. They then output a multi-dimensional predictive distribution of several key plasma state variables in the next timestep. Two of the models, which we refer to as **Fusion1** and **Fusion2**, predict a 3-dimensional target: β_N (the ratio of plasma pressure over magnetic pressure), *density* (the line-averaged electron density), and *li* (internal inductance). The third model, **Fusion3**, predicts one additional variable, *dr* (differential rotation of the plasma), to predict a 4-dimensional target. For the actuators, the model takes in the amount of power and torque injected from the neutral beams, the current, the magnetic field, and four shape variables (elongation, a_{minor}, triangularity top, and triangularity bottom). This, along with the state space, make for an input dimension of 11 (for **Fusion1** and **Fusion2**) or 12 (for **Fusion3**).

All three pre-trained models have the same model architecture. It is a recurrent probabilistic neural network (RPNN), which features an encoding layer by an RNN with 64 hidden units followed by a fully connected layer with 256 units, and a decoding layer of fully connected layers with [128, 512, 128] units, which finally outputs a diagonal Gaussian parameterized by the mean and a log-variance prediction. The Gaussian likelihood loss was used for training, with a learning rate of 0.0003 and weight decay of 0.0001. In using dynamics models to sample trajectories and train policies, the key metric practitioners are concerned with is explained variance, hence explained variance on a held out validation set of 1000 shots was monitored during training and training was halted early if there was no improvement for more than 250 epochs. **Fusion2** and **Fusion3** were trained with a non-smoothed dataset consisting of 12000 shots in the training dataset, and **Fusion2** explains on average 57% of the variance in the outputs, and **Fusion3** 40%. **Fusion1** was trained with a smoothed version of the dataset and explains on average 63% of variance in the outputs.

For each of these models, we learn the recalibration mapping with a validation dataset consisting of 1000 shots' worth of data for methods *SD Recalibration* and *HDR Recalibration*. For the *Pre-hoc* method, samples were drawn from the pre-trained models without any recalibration. For all models and methods, we drew 3000 samples at each test datapoint. We compute the average of each evaluation metric across 10 sets of predictive samples on a single set of 20 held-out test shots, and repeat this process for 10 different sets of 20 test shots. We report the mean and standard error across these 10 sets in Table 4.1 (Bottom).

4.E.4 Decision-making with Demand Forecasting

The demand forecasting model takes in the recent 4-day history of sales of each of the top three most sold items, and categorical variables which indicate the day of the week and week of the year, which makes for a total of 14 input dimensions. The model is then trained to predict a distribution over how much of each item was sold in the next business day. i.e. over 3-dimensional targets.

To demonstrate versatility of the proposed HDR recalibration method, in this experiment, we used NGBoost (natural gradient boosting [Duan et al., 2020]) for the demand forecasting model. We used the NGBRegressor, which predicts a diagonal Gaussian distribution. The model was trained with the CRPScore with a learning rate of 0.005, and number of estimators set to 1000. Training was stopped early if the CRPScore on the validation set did not improve for 20 iterations. The same validation set was used for recalibration.

During testing, we use the demand forecasting model to produce a set of samples that reflect possible realizations of demand for each of the three products in the next business day. Among these samples, we filter out samples which are over the budget. Then, with the remaining samples, we select the sample with the maximum sum of demand across the three products, and take this sample as the action – i.e. this sample is how much quantity of each product the store will prepare for the next business day. With the true sales data of the next business day, we compute the loss as described in Section 4.4.2. To set the budget, we took the mean of sales in the validation set.

For each method (Pre-hoc, HDR recalibration, SD recalibration), the simulation across the test set was repeated 10 times and the average of the cumulative loss across these 10 times was recorded. This full pipeline (model training, sampling, 10 repeated simulations) was done with 5 different seeds, and the mean and standard error of the cumulative loss across the 5 seeds is reported in Table 4.2.

4.E.5 Ablation Study on the Effects of PDF Adjustment in HDR Recalibration

We present an ablation study which demonstrates the effect of the PDF adjustment step, described in Section 4.3.2. For the 5 benchmark datasets, we run HDR recalibration *without* the adjustment step, and compare evaluation metrics against running HDR recalibration *with* the adjustment. The results are shown in the table below. The "With Adjustment" columns have been reproduced here from Table 4.1 for convenience. The energy score indicates that the adjustment step improves the quality of the predictive samples.

	***Without* Adjustment**			***With* Adjustment**		
Dataset	Energy	HDR-ECE	SD-ECE	Energy	HDR-ECE	SD-ECE
scpf	$3.55\ (0.37)$	$\mathbf{0.03\ (0.00)}$	$\mathbf{0.15\ (0.00)}$	$\mathbf{-0.79\ (0.42)}$	$0.07\ (0.01)$	$0.17\ (0.00)$
rf1	$0.09\ (0.02)$	$\mathbf{0.01\ (0.00)}$	$0.06\ (0.00)$	$\mathbf{0.08\ (0.01)}$	$\mathbf{0.01\ (0.00)}$	$\mathbf{0.05\ (0.00)}$
rf2	$1.05\ (0.29)$	$\mathbf{0.01\ (0.00)}$	$\mathbf{0.05\ (0.00)}$	$\mathbf{1.04\ (0.28)}$	$0.09\ (0.01)$	$0.06\ (0.00)$
scm1d	$1.11\ (0.01)$	$0.40\ (0.00)$	$0.07\ (0.00)$	$\mathbf{0.36\ (0.05)}$	$\mathbf{0.04\ (0.00)}$	$\mathbf{0.01\ (0.00)}$
scm20d	$1.27\ (0.01)$	$0.42\ (0.00)$	$0.07\ (0.00)$	$\mathbf{0.81\ (0.09)}$	$\mathbf{0.04\ (0.00)}$	$\mathbf{0.02\ (0.00)}$

Table 4.3: Results from comparing the HDR recalibration method without and with the PDF adjustment step. The energy score indicates that the adjustment step significantly improves the sample quality when performing HDR recalibration.

4.E.6 Computing Infrastructure

All of the model training was done with 4 NVIDIA GeForce RTX 2080 Ti GPUs.

All of the evaluation was done on a CPU machine with Intel(R) Xeon(R) Gold 6238 CPU @ 2.10GHz.

Part III

Using Uncertainties for Decision-making

5 | Proper Scoring Rules for Decision-making

Uncertainty-aware predictive models are commonly used in decision-making tasks. Specifically, given high-stakes decision-making settings, the model's quantified uncertainty provides a notion of risk when considering the set of possible outcomes that the model predicts as probable. When training such models along with quantified uncertainties, the optimization objective usually does not consider the downstream application task, and a standard loss function is used to train the model. In lieu of a mechanism to inform the model during training of the decision-making task at hand, the model is often post-hoc "recalibrated" to the task at hand. In this work, we propose a method of fine-tuning the model to calibrate the model to the downstream decision-making task. We leverage proper scoring rules in order to construct an optimization objective that is proper and is aligned with the utility of the decision-making objective.

5.1 Introduction

Consider a doctor who is tasked with making a diagnosis for heart disease based on a set of measurement collected from a patient. While the doctor can make a subjective judgment based on their experience, it is likely that a more reliable decision can be drawn if there existed a model which has been trained on as much diagnosis data that exists which far exceed the number of patients any single doctor can see in the span of a career and the doctor can access this model. Suppose the doctor has access to such an expert model which, given the set of measurement features, outputs a prediction for the existence of the disease. Based on the prediction, the doctor then chooses an action, which could be to administer disease treatment, prescribe medicine, or do nothing. Based on this action, the doctor and patient will be feedback that reflects whether the action was optimal for the true state of the patient.

While such a decision-making system is functional as described, given that the potential cost of an erroneous action can be catastrophic (e.g. no treatment administered for a positive case of disease can lead to loss of life), having a notion of uncertainty in the model prediction can greatly benefit the doctor (i.e. the decision-maker) in terms of gauging what the potential risk is, and the doctor can then choose an action would be optimal for the risk profile, where the notion of optimality can be task specific (e.g. a more conservative action can be optimal in this case).

As such, in high-stakes decision-making settings, a decision-maker can leverage uncertainty-aware models in an attempt to make optimal decisions for the downstream task. In such a setting, a pertinent question is, *"are the model's uncertainties informative and beneficial to the decision-maker's performance on the downstream task?"*

In many cases, the predictive models are produced by a modeling expert who has their own model training infrastructure, data sources, and evaluation pipelines. For example, weather forecasting is performed by designated (usually public) entities and the forecasts are released and consumed by many different downstream entities, which includes power traders speculating the supply and demand for electricity, air traffic control in deciding to ground flights, construction sites proceeding with or pausing work, and even down to very local and private entities such as a beach that decides to close the beach due to lightning risk, or a tennis academy training the next generation of athletes which is deciding to schedule outdoor sessions. The key point we raise in this work is that, to each of the parties involved in their own decision-making task, the set of actions they are considering, and the utility of an action given the true observation are all different. A power trader may face significant financial losses with wrong actions, whereas to a beach-goer, watching spectacular lightning indoors in a beach-side cafe can be as enjoyable and thus incurs no loss. Still, all of these actors consume the same probabilistic forecasts about the weather. Even if all of the actors chose the best action based on their utility (or loss) functions, given that the cost of an error for certain predictions can vary drastically across actors, it certainly may be the case that the probabilistic forecasts hurt one actor more than another even when it achieves "good validation performance" during model training and development.

In this work, we focus on this downstream decision-making utility of uncertainty. Following the Bayes decision-making perspective of proper scoring rules, we show that all proper scoring rules are inherently designed for a single decision-making task, which may be at odds with other decision-making tasks with different utility functions. We further propose a method of fine-tuning (or calibrating) a probabilistic machine learning model for a specific downstream task with a specific utility function.

5.2 Preliminaries

In this work, we focus on the classification setting where, given input features $x \in \mathcal{X}$, a probabilistic classifier $f : \mathcal{X} \to \Delta$ outputs a probability vector $p \in \Delta$. For a classification problem with k total number of classes, the output space will be the k-dimensional probability simplex: $p \in \Delta^k$ where $p = [p_0, p_1, \dots p_{k-1}]$, $p_i \geq 0, \sum_i p_i = 1, i \in [k]$. The true outcome is denoted $y \in \Delta^k$ which is a one-hot vector with value 1 at the index of the true class.

We consider decision-makers who use the predicted probability from f to make a decision by choosing an action from a *finite* action space $\mathcal{A}$. Based on an action $a \in \mathcal{A}$ and true class label $y \in \mathcal{Y}$, the decision-maker incurs utility $U(y, a)$, where $U : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$ is the utility function of the decision-making task. We note that the set of $\{\mathcal{Y}, \mathcal{A}, U\}$ specify a single decision-making task. For example, given the same target space and action space $\mathcal{Y}$ and $\mathcal{A}$, a different utility function U' specifies a different decision-making task.

We consider a decision-maker who, based on the predicted probabilities of the targets p, decides to take the action that will maximize their *expected* utility. Such an action is referred to as the Bayes optimal action, which we denote with a_p.

$$a_p = \arg\max_a \mathbb{E}_{y \sim p}[U(y, a)] \tag{5.1}$$

Note here that the expectation is taken w.r.t. the predicted distribution p. Thus, the decision-maker tries "their best" given the information available (i.e. p).

We summarize the notation in the following bullet points:

- Outcome: $y \in \mathcal{Y}$

- Predicted probability: $p \in \Delta$
- Action space: $a \in \mathcal{A}$
- Utility function: $U(y, a) : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$
- Bayes optimal action: $a_p = \arg\max_a \mathbb{E}_{y \sim p}[U(y, a)]$

In the next section, we introduce proper scoring rules, and their relationship to decision-making tasks. Based on the relationship, we elucidate how the training objective (i.e. the loss function) is often misaligned with the downstream decision-making task.

5.3 Proper Scoring Rules and Decision-making

Proper scoring rules are summary statistics of the quality of predictive distributions [Gneiting and Raftery, 2007]. By definition, the optimum of the expectation of any proper scoring rule is achieved by the ground truth, data-generating distribution. Proper scoring rules can be used to *evaluate* predictive distributions against a single test datapoint, or a test distribution: i.e a proper scoring rule S is either a function $S : \Delta \times \mathcal{Y} \to \mathbb{R}$ or $S : \Delta \times \Delta \to \mathbb{R}$ (which admittedly abuses notation). Given a test datapoint (x, y) and predictive distribution over y, $p = f(x)$, the proper scoring rule S can be evaluated as $S(p, y)$, and the expected proper scoring rule is evaluated by taking the expectation $S(p, y)$ over the ground truth distribution over $Y|x$, which we denote with q: $S(p, q) = \mathbb{E}_{y \sim q} S(p, y)$. By definition of being a proper scoring rule, $S(q, q) \geq S(p, q), \forall p \in \Delta$.

Proper scoring rules are also commonly used as loss functions in machine learning: negative log-likelihood (a.k.a. log score), cross entropy loss (a.k.a. log score), squared loss (a.k.a. quadratic score), pinball loss (a.k.a. check score) are primary examples.

Proper scoring rules have an inherent correspondence to decision-making tasks. Dawid [2007] elucidates this relationship, which we summarize as follows. Given an outcome $y \in \mathcal{Y}$, action $a \in \mathcal{A}$ and utility function $U : \mathcal{Y} \times \mathcal{A} \to \mathbb{R}$, let $U(y, a)$ be the utility of having taken action a when the observed outcome is y, and let Δ be a space of distributions over the outcomes $\mathcal{Y}$. Denoting the Bayes action given the distribution $p \in \Delta$ as a_p, the function $U(y, a_p)$ is a proper scoring rule, i.e. $U(y, a_p) = S(p, y)$ for some proper scoring rule S.

This relationship indicates that when we train a model using a proper scoring rule as the loss function and optimization objective, we are in fact training the model in solving a specific decision-making task with a fixed utility function. This raises the question, what exactly is the decision-making task of common loss functions? As it turns out, each proper scoring rule assumes the same function as the utility function in their corresponding decision making task.

5.3.1 All proper scoring rules assume their own function as the utility function in their decision-making task

Repeating the setting for the general classification case:

- Outcome space: $y \in \mathcal{Y} \equiv [k]$
- Action space: $a \in \mathcal{A} \equiv \Delta$, i.e. k-dimensional probability simplex

Since we have fixed the action and outcome space, deriving the “decision making task” is now equivalent to deriving the “utility function”. We show that if we

1. Set the action space $\mathcal{A}$ identical to the probability space, Δ (i.e. $(k-1)$ probability simplex)
2. Set the utility function as the proper scoring rule itself, i.e. $U(y, a) = S(a, y)$, where $a \in \Delta$

then all of the equations related to the utility function are satisfied.

Recall that $S(p, y) = U(y, a_p)$, where a_p is the Bayes optimal action given predicted probability p.

$$
\begin{aligned}
a_p &= \arg\max_{a \in \mathcal{A}} \mathbb{E}_{y \sim p}[U(y, a)] && (5.2) \\
&= \arg\max_{a \in \mathcal{A}} \mathbb{E}_{y \sim p}[S(a, y)] && (5.3) \\
&= \arg\max_{a \in \mathcal{A}} S(a, p) && (5.4) \\
& && (5.5)
\end{aligned}
$$

By the definition of proper scoring rules, we know that $\forall q \in \Delta, S(q, p) \leq S(p, p)$.

Therefore,

$$
a_p = \arg\max_{a \in \mathcal{A}} S(a, p) = p. \qquad (5.6)
$$

Hence, $S(p, y) = U(y, a_p) = U(y, p)$, which adheres to our assumptions. □

What this signifies is the following. If we take the commonplace practice of training a probabilistic model with log-likelihood, which is equivalent to training with the cross-entropy loss in classification, we are training the model to perform well in a decision-making task where given an outcome y and an action $a = p \in \Delta$, the model is *rewarded* with utility equal to $\log p_y$, where y denotes the index of the true class.

We note the following observations about the above decision-making problem:

- **Action space is identical to the prediction space over outcomes**: Note that this designates a *continuous* action space.
- **Symmetry across the true label** y: As long as the same amount of predicted probability was allocated to the label y, the utility is identical, regardless of what y is.
- **Independence across labels**: The utility function is only concerned with how much probability was allocated to the true label, and is agnostic about *which other* wrong labels probability was allocated to. If we take CIFAR10 as an example, on a datapoint with `airplane` as the true label, if the model placed 0.7 probability to the true label, the score does not care if the residual probability was allocated to `dog` or `horse`. Either case provides identical utility.

However, in the real world, there are many decision-making tasks where the utility function is at odds with both aspects listed. First of all, the action space may be a completely different space than the prediction space, and it may be discrete. Further, consider the example of heart disease detection and treatment. A false negative error and action may result in a much higher loss than a false positive error. Also, on a positive case of heart disease, administering treatment may yield the highest utility, prescribing symptom causing drugs may yield lower utility, and applying no treatment may result in the lowest utility.

Given these considerations, it seems natural to expect that the standard cross-entropy loss function may not be optimal in training a heart disease classification system, when considering the fact that the downstream task this system will be used for displays such asymmetry and correlation. In the next section, we introduce a method of constructing a proper scoring rule which *does* take into consideration the utility function of the downstream task. This results in an optimization objective which is directly aligned with the downstream decision-making task, thus training the model with this loss function is identical to training the model to perform better on the specific downstream task.

5.4 Method

In introducing our method, we focus on a simplified but practical setting that we term *Finite Action Decision-making*. The only condition this setting introduces is the fact that the number of actions the decision-maker can take is finite, i.e. $|\mathcal{A}| = n, n \in \mathbb{N}$.

5.4.1 Finite Action Decision-making

In the finite action decision-making setting, the utility function can be summarized with a single utility matrix, C, where the (a, y) entry specifies the utility of having taking action a when the label was y: $U(y, a)$. We summarize and derive the relevant quantities in this setting as follows:

- Utility matrix C, $(n \times k)$:
  - Rows are indexed by action, columns are indexed by the label (n total actions, k labels)
  - $(k \times k)$ matrix when #(actions) equals #(labels).
  - $C[i, j]$ is the utility gained by taking action i when label is j.

The mathematical quantities of interest are as follows:

- Action space $\mathcal{A} : \{a_1, a_2, \dots a_n\}$ where $a_i = e_i$ (one-hot vector at index i)
- Utility function $U(y, a) = a^T C y$
- Score function $S(p, y) = U(y, \arg\max_a \mathbb{E}_{y\sim p}[U(y, a)]) = U(y, a_p) = a_p^T C y$
- Expected utility of action a assuming $y \sim p$: $\mathbb{E}_{y\sim p}[U(y, a)] = a^T C p$
- Expected utility $G(p) = S(p, p) = \mathbb{E}_{y\sim p}[S(p, y)] = \mathbb{E}_{y\sim p}[a_p^T C y] = a_p^T C p$

Consider the equation $a_p = \arg\max_a \mathbb{E}_{y\sim p}[U(y, a)]$.

$$\arg\max_a \mathbb{E}_{y\sim p}[U(y, a)] \tag{5.7}$$

$$= \arg\max_a a^T C p \tag{5.8}$$

$$= \text{index of max among } \{(Cp)_i\}_{i=1}^n \tag{5.9}$$

Note that $Cp \in \mathbb{R}^{n\times 1}$ and $(Cp)_i$ is the utility of taking action i when $y \sim p$.

Since $\arg\max_a a^T C p$ is a one-hot vector which can be interpreted as a probability vector with probability 1 at the index of the max value of $(Cp)_i$, we can "soften" the argmax operation with the softmax: $\texttt{softmax}(Cp) \in \mathbb{R}^{n\times 1}$.

Therefore, we can approximate $S(p, y)$ with the following:

$$S(p, y) = U(y, a_p) \tag{5.10}$$

$$= a_p^T C y \tag{5.11}$$

$$\approx \hat{S}(p, y) = \texttt{softmax}(Cp)^T C y \tag{5.12}$$

$$\tag{5.13}$$

Lastly, note that p is the predicted probability vector over the labels, which is derived from the logits $X \in \mathbb{R}^{k\times 1}$ outputted by a classification model, i.e. $p = \texttt{softmax}(X)$.

In contrast to $S(p, y)$, $\hat{S}(p, y)$ is now fully differentiable w.r.t. p and amenable to optimization with gradient-based methods. We empirically verify that optimizing classifiers with this objective leads to much higher utility for the downstream task compared to standard loss functions or other baseline calibration methods.

5.5 Empirical Results

We evaluate the performance of our method on two different datasets: they are both from medical applications and both involve producing probabilistic predictions on the state of a patient, and then taking an action to decide the treatment for the patient.

In all cases, the experimental procedure is as follows. We first pre-train a classifier on the train split of the dataset with cross entropy loss, and early stop based on the validation cross entropy loss. This checkpoint is regarded as the best likelihood model. Afterwards, we fine-tune or recalibrate the model on the train split while performing validation on the validation set. We consider the following methods:

- `prehoc`: We simply use the pretrained model to perform the decision-making task.
- `ce`: Fine-tuning with the cross entropy loss.
- `weighted_ce`: Fine-tuning with weighted cross entropy loss, where the class weights are optimal for the decision-making task
- `decision_calibration`: Recalibration method by Zhao et al. [2021b]
- `soft_score`: Fine-tuning with the soft objective constructed in Section 5.4.

For each of these methods, we evaluate performance with the decision-making task. Given a test datapoint, the model (prehoc, fine-tuned, or recalibrated) outputs predicted probabilities of the true label, and based on these class probabilities, we assume a decision-maker takes the Bayes optimal action according to the utility function of the task. We then observe the true label and record the utility achieved by having taken the Bayes optimal action. The evaluation metric is the utility achieved on the test dataset.

5.5.1 Heart Disease Detection

This a binary classification dataset where given 13-dimensional input features, the prediction task is to classify the datapoint as a positive or negative case of heart disease. We assume there are two actions a decision-maker can take: {`treatment`, `no treatment`}. We define the utility function as follows: We set k s.t. $0 < k < 1$. A realistic loss function for this problem would be

	Treatment	No Treatment
Positive	0	$-10 * k$
Negative	$-10 * (1 - k)$	0

when k is large, i.e. when the negative utility of choosing action `no treatment` for a positive case is much larger than the negative utility of having administered treatment on a negative case. Regardless, to test the methods on a variety of loss functions, we vary k between 0.1 and 0.9 in 0.1 increments.

Table 5.1 and 5.2 display the average test utility across 9 different settings of k (i.e. 9 different utility functions). We see that the method `soft_score` is able to consistently produce higher utility, with the exception for $k = 0.6$, where training on any other objective than cross entropy results in deterioration. We are not sure why this is the case with $k = 0.6$.

Method	0.1	0.2	0.3	0.4	0.5
prehoc	-1.03 ± 0.24	-1.39 ± 0.31	-1.71 ± 0.30	-1.82 ± 0.26	-1.84 ± 0.31
ce	-1.04 ± 0.24	-1.39 ± 0.31	-1.71 ± 0.30	-1.82 ± 0.26	-1.84 ± 0.31
weighted_ce	$\mathbf{-1.02 \pm 0.29}$	-1.33 ± 0.31	-1.60 ± 0.31	-1.78 ± 0.26	-1.84 ± 0.31
decision_calibration	$\mathbf{-1.02 \pm 0.26}$	-1.36 ± 0.32	-1.63 ± 0.32	$\mathbf{-1.63 \pm 0.28}$	-1.80 ± 0.32
soft_score	$\mathbf{-1.02 \pm 0.29}$	$\mathbf{-1.26 \pm 0.31}$	$\mathbf{-1.42 \pm 0.28}$	-1.75 ± 0.26	$\mathbf{-1.75 \pm 0.31}$

Table 5.1: Heart disease detection: test utility values (mean $\pm$ stderr) $\times 100$ across 20 seeds, for $k \in \{0.1, \ldots, 0.5\}$

Method	0.6	0.7	0.8	0.9
prehoc	$\mathbf{-1.71 \pm 0.29}$	-1.28 ± 0.24	-0.99 ± 0.10	-0.81 ± 0.18
ce	$\mathbf{-1.71 \pm 0.29}$	-1.28 ± 0.24	-0.99 ± 0.19	-0.81 ± 0.18
weighted_ce	-1.93 ± 0.30	-1.31 ± 0.26	-0.94 ± 0.20	-0.69 ± 0.19
decision_calibration	-1.80 ± 0.32	$\mathbf{-1.21 \pm 0.22}$	-0.93 ± 0.20	-0.75 ± 0.19
soft_score	-1.76 ± 0.27	$\mathbf{-1.21 \pm 0.26}$	$\mathbf{-0.87 \pm 0.22}$	$\mathbf{-0.64 \pm 0.20}$

Table 5.2: Heart disease detection: test utility values (mean $\pm$ stderr) $\times 100$ across 20 seeds, for $k \in \{0.6, \ldots, 0.9\}$

5.5.2 Skin Lesion Dataset

We also perform evaluations on the HAM10000 dataset [Tschandl et al., 2018], which is another dataset in medicine, specifically on skin lesion classification. We follow the exact same experimental protocol as Zhao et al. [2021b] and the specific utility function that is most appropriate for this dataset. We refer the reader to Zhao et al. [2021b] for details on the label space, action space, and the utility function.

Table 5.3 displays the test utility on this dataset, and we again see that `soft_score` is the most effective in adjusting the model to output the highest utility on the decision-making task that we are concerned with.

Method	Value
prehoc	1.819 ± 0.004
ce	1.817 ± 0.003
weighted_ce	1.819 ± 0.004
decision_calibration	1.822 ± 0.004
soft_score	$\mathbf{1.824 \pm 0.005}$

Table 5.3: Skin lesion classification: test utility values (mean $\pm$ stderr) $\times 10$ across 8 seeds

5.6 Discussion

In this work, we focused on *using* uncertainties (or probabilistic forecasts) for decision-making. We made the connection that when uncertainty-aware models are trained with proper scoring rules, they are in fact being trained on a specific decision-making task that may not align with a downstream decision-making task that one may actually be faced with. To remedy this mismatch, we proposed a method of constructing a proper scoring rule which directly reflects the task in question, and also proposed a soft approximation which allows gradient based optimization.

While the empirical results in this work are promising, there are many improvements that can be made to the discussion made thus far in this work. For one, while the soft approximation (i.e. the `soft_score`) is differentiable, it is not convex. While this does not prevent one from still using gradient-based methods for optimization since deep neural network optimization is non-convex anyway, having an optimization objective that is either convex in the predicted probability space or the logit space has been empirically shown to be beneficial for training. Deriving approximations which are still smooth but achieve convexity would be an exciting direction for future directions.

6 | Parity Calibration

In a sequential regression setting, a decision-maker may be primarily concerned with whether the future observation will increase or decrease compared to the current one, rather than the actual value of the future observation. In this context, we introduce the notion of parity calibration, which captures the goal of calibrated forecasting for the increase-decrease (or "parity") event in a timeseries. Parity probabilities can be extracted from a forecasted distribution for the output, but we show that such a strategy leads to theoretical unpredictability and poor practical performance. We then observe that although the original task was regression, parity calibration can be expressed as binary calibration. Drawing on this connection, we use an online binary calibration method to achieve parity calibration. We demonstrate the effectiveness of our approach on real-world case studies in epidemiology, weather forecasting, and model-based control in nuclear fusion.

6.1 Introduction

Many tasks in the scope of prediction and decision making are sequential in nature. A weather forecaster who uses some procedure to make predictions for tomorrow, may find that tomorrow's events falsify these predictions. A good forecaster must then update their model before using it on the following days. In this paper we study the sequential forecasting setting where the goal is to make predictions about a sequence of real-valued outcomes $y_1, y_2, \ldots \in \mathcal{Y} \subseteq \mathbb{R}$ using informative covariates $_1, _2, \ldots \in$. In the presence of inherent stochasticity or insufficient data, forecasters who provide rich predictions in the form of complete distributions over the output allow us to reason about the inherent uncertainties in the data stream [Gneiting et al., 2007]. If a distributional prediction is available, a downstream decision-maker can account for risks that were unknown at the time of forecasting.

Often, a distributional forecast for the real-valued y_t takes the form of a predictive cdf (cumulative distribution function) for y_t, which in this paper we typically denote as $\hat{F}_t : \mathcal{Y} \to [0, 1]$. We sometimes write $\hat{F}_t$ as $\hat{F}_t(\cdot|_t)$ or $\hat{F}_t(\cdot|_t, y_{t-1}, _{t-1}, \ldots, y_1, _1)$; this overloaded notation allows us to be succinct when defining what it means for $\hat{F}_t$ to be calibrated, but explicit when it is necessary to stress that $\hat{F}_t$ depends on all available knowledge. We also refer to $\hat{F}_t$'s as regression forecasts, as it models a continuous distribution over the real-valued output.

In this paper, we are interested in the question: can we forecast whether the future outcome y_{t+1} will be greater or less than the current outcome y_t? To motivate this question, consider a hospital in the midst of a fast moving pandemic such as COVID-19. It may be difficult for the hospital to comprehend absolute numbers of patients requiring hospitalization. However, relative numbers are perhaps easier to interpret: hospitals know the situation today, and would like to know if it is going to worsen or improve tomorrow.

A domain expert (e.g. epidemiologist) may have produced a regression forecast $\hat{F}_t$ for y_t. The downstream user (e.g. hospital) can then extract from $\hat{F}_t$ a natural implied probability of the next observation decreasing:

$$\text{for } t \geq 2, \;\; \hat{p}_t = \hat{F}_t(y_{t-1} \mid_t). \tag{6.1}$$

The hope of the hospital is that the forecasted probabilities $\hat{p}_t$ are parity calibrated, as defined next.

Definition 2 (Parity calibration)**.** *The forecasts* $\{\hat{p}_t \in [0,1]\}_{t=2,\ldots,T}$ *are said to be parity calibrated if*

$$\frac{\sum_{t=2}^{T} y_t \leq y_{t-1} \hat{p}_t = p}{\sum_{t=2}^{T} \hat{p}_t = p} \to p, \forall p \in [0,1]. \tag{6.2}$$

In words, whenever a parity calibrated forecaster predicts with probability p that $y_t \leq y_{t-1}$, the event $y_t \leq y_{t-1}$ actually occurs with empirical frequency p (in the long run). To avoid confusion with usage of the term “parity" in fairness literature, we remark that our context is purely in comparing two consecutive values.

Our first contribution is showing that even if $\hat{F}_t$ is calibrated (based on some accepted notions of calibration), the seemingly reasonable strategy mentioned above (6.1) can have devastating and unpredictable behavior (Section 6.1.1). Yet, it stands to reason that the expert's rich forecast $\hat{F}_t$ should be used in some way. Our second contribution is a methodology for doing this (Sections 6.2 and 6.3). Our main methodology described in Section 6.2.2 is based on the key observation that although the parity calibration problem is derived from a regression problem, it naturally reduces to a problem of forecasting binary events.

6.1.1 Regression calibration does not give parity calibration

A popular notion of calibration in regression is *probabilistic calibration* [Gneiting et al., 2007]. The sequence $\hat{F}_1, \hat{F}_2, \ldots$ is said to be probabilistically calibrated if

$$\frac{1}{T} \sum_{t=1}^{T} F_t(\hat{F}_t^{-1}(p)) \to p, \; \forall p \in [0,1], \tag{6.3}$$

where F_t denotes the ground truth distribution. Probabilistic calibration is also referred to as *quantile calibration*, since it focuses on the quantile function being valid. In other works, it has also been referred to as average calibration [Zhao et al., 2020, Chung et al., 2021b, Sahoo et al., 2021], or simply calibration [Kuleshov et al., 2018, Cui et al., 2020, Charpentier et al., 2022, Marx et al., 2022]. We will henceforth refer to this notion as *quantile calibration*.

Another notion of calibration in regression is *distributional calibration* [Song et al., 2019], which assesses the convergence of the full distribution of the observations to the predictive distribution. A distribution calibrated forecaster satisfies $\forall p \in [0,1], \; \forall F \in \mathcal{F}$,

$$\frac{\sum_{t=1}^{T} \hat{F}_t = F F_t(\hat{F}_t^{-1}(p))}{\sum_{t=1}^{T} \hat{F}_t = F} \approx p \tag{6.4}$$

where $\mathcal{F}$ is the space of distributions predicted by $\hat{F}_t$. However, distributional calibration is an idealistic notion that cannot be achieved in practice [Song et al., 2019].

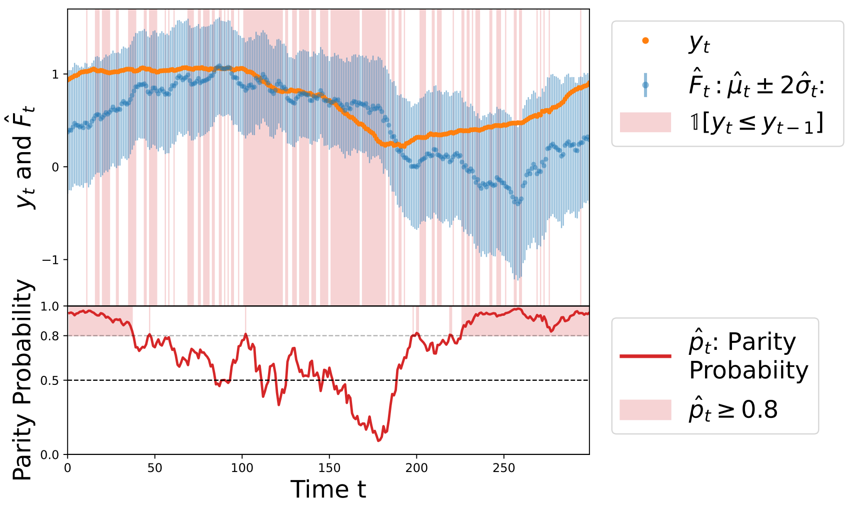


Figure 6.1: Snapshot of the first 300 points from one of our experiment datasets (Pressure from Section 6.3.2) shows a quantile calibrated forecaster that is highly parity miscalibrated. (**top**) The expert forecasts $\hat{F}_t$ are Gaussians, expressed in the plot as prediction intervals $[\hat{\mu}_t - 2\hat{\sigma}_t, \hat{\mu}_t + 2\hat{\sigma}_t]$. This prediction interval almost always contains y_t and its reliability diagram in Figure 6.6 (plot titled "Quantile Calibration") confirms that $\hat{F}_t$ is in fact quantile calibrated when considering the full timeseries. (**bottom**) For $t \in [0, 40]$ and $t \in [230, 300]$, the parity probabilities $\hat{p}_t = \hat{F}_t(y_{t-1})$ assign ≥ 0.8 probability (red shaded areas) to $y_t \leq y_{t-1}$. But y_t actually decreases with much lower frequency during these timesteps as can be seen from the top figure. The parity miscalibration when considering the full timeseries is confirmed by Figure 6.6 (plot titled "Prehoc").

Recently, Sahoo et al. [2021] paired calibration with the notion of threshold decisions and proposed *threshold calibration*. Forecasts are said to be threshold calibrated if,

$$\frac{\sum_{t=1}^{T} \hat{F}_t(y_0) \leq \alpha F_t(\hat{F}_t^{-1}(p))}{\sum_{t=1}^{T} \hat{F}_t(y_0) \leq \alpha} \to p,$$
$$\forall y_0 \in \mathcal{Y},\ \forall \alpha \in [0,1], \forall p \in [0,1].$$

Sahoo et al. [2021] show that distribution calibration implies threshold calibration, but the converse may not hold.

A common aspect of the aforementioned notions of calibration is that they all assess how well-aligned the predictive quantiles are to their empirical counterparts. The key difference among the notions is the conditioning over which this assessment is performed.

Since calibration is regarded as a desirable quality of distributional forecasts, one may wonder whether a calibrated $\hat{F}_t$ is sufficient for parity calibration of the implied probabilities as per Eq. (6.1). We show that this is *not* the case with the following examples.

Synthetic example. Let $_-$ and $_+$ denote the standard normal distributions truncated at 0, with density functions $f_-(x) = x < 0\sqrt{2/\pi}e^{-x^2/2}$ and $f_+(x) = x \geq 0\sqrt{2/\pi}e^{-x^2/2}$ respectively. Let F_- and F_+ be the cdfs of $_-$ and $_+$. Suppose the target sequence $(Y_t)_{t=1}^{\infty}$ is distributed as

$$Y_t \sim \begin{cases} _- & \text{if } t \text{ is odd,} \\ _+ & \text{if } t \text{ is even.} \end{cases}$$

Consider the following predictive cdf targeting Y_t,

$$\hat{F}_t = \frac{1}{2}F_- + \frac{1}{2}F_+ = \begin{cases} \frac{1}{2}F_-(y), \text{ if } y < 0, \\ 0.5 + \frac{1}{2}F_+(y), \text{ if } y \geq 0. \end{cases}$$

We note that when $y < 0, \frac{1}{2}F_-(y) \in [0, 0.5)$, and when $y \geq 0.5, 0.5 + \frac{1}{2}F_+(y) \in [0.5, 1]$. It can be verified that the corresponding quantile function is

$$\hat{F}_t^{-1}(p) = \begin{cases} F_-^{-1}(2p), \text{ if } p < 0.5 \\ F_+^{-1}(2p - 1), \text{ if } p \geq 0.5. \end{cases}$$

We verify that $\hat{F}_t$ is quantile calibrated (following Eq. (6.3)).

When t is odd, $F_t = F_-$.

- $\forall p \in [0, 0.5)$, $F_t(\hat{F}_t^{-1}(p)) = F_-(F_-^{-1}(2p)) = 2p$.
- $\forall p \in [0.5, 1]$, $\hat{F}_t^{-1}(p) = F_+^{-1}(2p - 1) \geq 0$, thus $F_t(\hat{F}_t^{-1}(p)) = F_-(F_+^{-1}(2p - 1)) = 1$.

When t is even, $F_t = F_+$.

- $\forall p \in [0, 0.5)$, $\hat{F}_t^{-1}(p) = F_-^{-1}(2p) < 0$, thus $F_t(\hat{F}_t^{-1}(p)) = F_+(F_-^{-1}(2p)) = 0$.
- $\forall p \in [0.5, 1]$, $F_t(\hat{F}_t^{-1}(p)) = F_+(F_+^{-1}(2p - 1)) = 2p - 1$.

Therefore, for $p \in [0, 0.5)$, $\frac{1}{T}\sum_{t=1}^{T} F_t(\hat{F}_t^{-1}(p)) = \frac{1}{T}\sum_{t \text{ is odd}} 2p = p + o(\frac{1}{T}) \to p$, and the same can be verified for $p \in [0.5, 1]$, showing that $\hat{F}_t$ *is quantile calibrated*.

We can easily show that $\hat{F}_t$ is also distribution and threshold calibrated. Since $\hat{F}_t$ is constant for all t, following Eq. (6.4), the space of predicted distributions is a singleton. Thus, measuring distribution calibration is equivalent to measuring quantile calibration, and $\hat{F}_t$ *is distribution calibrated*. Since distribution calibration implies threshold calibration [Sahoo et al., 2021], $\hat{F}_t$ *is threshold calibrated*.

However, as we show next, $\hat{F}_t$ is not parity calibrated.

When t is odd, $Y_t \sim F_-$ and $Y_{t-1} \sim F_+$. Thus $Y_t < Y_{t-1}$ whereas $\hat{p}_t = \hat{F}_t(Y_{t-1}) \geq 0.5$.

When t is even, $Y_t \sim F_+$ and $Y_{t-1} \sim F_-$. Thus $Y_t > Y_{t-1}$ whereas $\hat{p}_t = \hat{F}_t(Y_{t-1}) < 0.5$.

Therefore, $\forall \hat{p}_t \geq 0.5$, $y_t \leq y_{t-1} = 1$ and $\forall \hat{p}_t < 0.5$, $y_t \leq y_{t-1} = 0$, thus $\hat{F}_t$ *is parity miscalibrated* for all $\hat{p}_t \in (0, 1)$, i.e. all $\hat{p}_t \neq 0$ or 1.

Intuitively, the sequential aspect of predictions and observations is central to the notion of parity calibration, whereas traditional notions of calibration effectively treat the datapoints as an i.i.d. or exchangeable batch of points. Figure 6.1 provides a visualization of how this pitfall can be manifested in a practical example.

The implication is that methods designed to achieve traditional notions of calibration in regression cannot be expected to provide parity calibration. The following section introduces the posthoc binary calibration framework that can instead be used to achieve parity calibrated forecasts.

6.2 Parity calibration via binary calibration

Define the *parity outcomes* as

$$\text{for } t \geq 2,\ \widetilde{y}_t := y_t \leq y_{t-1}, \tag{6.5}$$

and observe that the parity calibration condition (Eq. (6.2)) is equivalently written as,

$$\frac{\sum_{t=2}^{T} \widetilde{y}_t \hat{p}_t = p}{\sum_{t=2}^{T} \hat{p}_t = p} \to p, \forall p \in [0, 1]. \tag{6.6}$$

Thus parity calibration is in fact targeting the binary sequence $\widetilde{y}_t$, instead of y_t. In this section, we show how this connection allows us to leverage powerful techniques from the rich literature of binary calibration that goes back four decades [DeGroot and Fienberg, 1981, Dawid, 1982, Foster and Vohra, 1998]. Of specific interest to us will be a class of methods that have been proposed for *posthoc calibration* of machine learning (ML) classifiers, which we review next.

6.2.1 Posthoc binary calibration

Let $f : \mathcal{X} \to [0, 1]$ be a binary classifier that takes as input a feature vector in feature space $\mathcal{X}$ and outputs a score in $[0, 1]$. Suppose a feature-label pair (X, Y) is drawn from some distribution P over $\times\{0, 1\}$. Then, f is said to be calibrated (in the binary sense) if

$$P(Y = 1 \mid f(X)) = f(X). \tag{6.7}$$

The terms on either side of the equal sign are random variables and the equality is understood almost-surely. The connection between (6.6) and (6.7) is evident: $\hat{p}_t$ is like $f(X)$, conditioning on the random variable $f(X)$ is akin to using indicators in the numerator/denominator, and $\widetilde{y}_t$ is like Y.

We do not expect ML models to be calibrated "out-of-the-box". So, if f is a logistic regression or neural network trained on some training data, it is unlikely to satisfy an approximate version of (6.7) on unseen data. Posthoc calibration techniques transform f to a function that is better calibrated by using a so-called *calibration dataset* $\mathcal{D}_{\text{cal}} = \{(_1, y_1), (_2, y_2), \ldots, (_c, y_c)\}$. $\mathcal{D}_{\text{cal}}$ is a set of points on which f was not trained—in practice $_{\text{cal}}$ is often just the validation dataset. $\mathcal{D}_{\text{cal}}$ is used to a learn a mapping $m : [0, 1] \to [0, 1]$ so that $m \circ f$ is better calibrated than f. By way of an example, we now introduce the popular Platt scaling technique [Platt, 1999] that will be central to this paper (henceforth, Platt scaling is referred to as PS). Given a pair of real numbers $(a, b) \in \mathbb{R}^2$, the PS mapping $m^{a,b} : [0, 1] \to [0, 1]$ is defined as,

$$m^{a,b}(z) = \text{sigmoid}(a \cdot \text{logit}(z) + b).$$

Here $\text{logit}(z) = \log(\frac{z}{1-z})$ and $\text{sigmoid}(z) = 1/(1 + e^{-z})$ are inverses of each other. Thus PS is a logistic model on top of the f-induced one-dimensional feature $\text{logit}(f(x)) \in [0, 1]$, instead of on the raw feature $x \in$. In the posthoc setting, (a, b) are set to the values that minimize log-loss (equivalently cross entropy loss) on $\mathcal{D}_{\text{cal}}$:

$$(\widehat{a}, \widehat{b}) =_{(a,b)\in\mathbb{R}^2} \sum_{(_s, y_s)\in\mathcal{D}_{\text{cal}}} l(m^{a,b}(f(_s)), y_s), \tag{6.8}$$

where $l(p, y) = -y \log p - (1 - y) \log(1 - p)$.

We briefly note some other popular posthoc calibration methods. These broadly fall under two categories: parametric scaling methods such as beta scaling [Kull et al., 2017], temperature scaling [Guo et al., 2017], and PS [Platt, 1999]; and nonparametric methods such as binning [Zadrozny and Elkan, 2001, Gupta et al., 2020, Gupta and Ramdas, 2021], isotonic regression [Zadrozny and Elkan, 2002], and Bayesian binning [Naeini et al., 2015].

6.2.2 Parity calibration using online versions of Platt Scaling (PS)

To achieve parity calibration using posthoc techniques, we start with a base cdf predictor $G : \mathcal{X} \to \Delta(\mathcal{Y})$ derived from an expert—such as an epidemiologist, a weather forecaster, or a stock trader. Here, $\Delta(\mathcal{Y})$ refers to the space of distributions over $\mathcal{Y}$. If the expert is an ML engineer, such a G

Algorithm 11 Platt scaling (PS) variants for parity calibration

```
1:  Input: Any base forecaster $G :\to \Delta()$, covariate-outcome pairs $(_1, y_1), (_2, y_2), \ldots \in \times$,
    update-frequency uf, moving-window-size ws.
2:  Output: PS forecasts $(\hat{p}_t, \hat{p}_t, \hat{p}^{)\infty}_{tt=2}$
3:  Initialize IW, MW, OPS parameters:
    $(a^{\text{IW}}, b^{\text{IW}}) = (a^{\text{MW}}, b^{\text{MW}}) = (a^{\text{OPS}}, b^{\text{OPS}}) \leftarrow (1, 0)$
4:  for $t = 2$ to $T$ do
5:      $\widetilde{y}_t = y_t \leq y_{t-1}$
6:      $\hat{p}_t = G(_t)[y_{t-1}]$
7:      $\hat{p}_t^{\text{IW}} \leftarrow (a^{\text{IW}} \cdot (\hat{p}_t) + b^{\text{IW}})$
8:      $\hat{p}_t^{\text{MW}} \leftarrow (a^{\text{MW}} \cdot (\hat{p}_t) + b^{\text{MW}})$
9:      $\hat{p}_t^{\text{OPS}} \leftarrow (a^{\text{OPS}} \cdot (\hat{p}_t) + b^{\text{OPS}})$
10:     if $t$ is a multiple of uf then
11:         $(a^{\text{IW}}, b^{\text{IW}}) \leftarrow$ optimal PS parameters
                based on (6.8) setting $_{\text{cal}} = (_s, \widetilde{y}_s)_{s=1}^t$
12:         $(a^{\text{MW}}, b^{\text{MW}}) \leftarrow$ optimal PS parameters
                based on (6.8) setting $_{\text{cal}} = (_s, \widetilde{y}_s)_{s=t-\texttt{ws}+1}^t$
13:     end if
14:     $(a^{\cdot}b^{)} \leftarrow \text{OPS}((_1, \widetilde{y}_1), \ldots, (_t, \widetilde{y}_t))$
15:     (OPS is Algorithm 2 in Appendix D)
16: end for
```

can be obtained using Gaussian processes [Rasmussen, 2004] or probabilistic neural networks [Nix and Weigend, 1994, Lakshminarayanan et al., 2017], among other methods. The test-stream occurs after G has been trained and fixed. This G gives us a $\hat{F}_t$ as described in the introduction: $\hat{F}_t = G(_t)$. Recall that the strategy Eq. (6.1) is to forecast $\hat{p}_t = \hat{F}_t(y_{t-1})$. If $\hat{F}_t$ were the true cdf of y_t given the past, the above $\hat{p}_t$ would be the true probability of $\widetilde{y}_t = 1$, and thus the most useful parity forecast possible.

However, in Section 6.1.1 we showed that we must modify $\hat{p}_t$ in order to achieve parity calibration. We propose using PS to perform this modification (any posthoc calibration method can be used; we focus on PS in this paper). A natural possibility would be to use an initial part of the test-stream to learn fixed PS parameters once, as described in the previous subsection. However, real-world regression sequences (weather, stocks, etc) have non-stationary shifting behavior across time. Therefore, a fixed model is unlikely to remain calibrated over time.

In Algorithm 11 we outline three ways to mitigate this. Increasing Window (IW) updates the PS parameters using all datapoints until some recent time step, such as every 100 timesteps ($t = 100, 200$, etc). A related alternative, Moving Window (MW) is to use only the most recent datapoints when updating the PS parameters (instead of all the points). The third alternative is Online Platt Scaling (OPS) based on our own recent work [Gupta and Ramdas, 2023].

In the following section, we compare these online versions of Platt scaling on three real-world sequential prediction tasks. We find that OPS performs better than the base model, MW, and IW, across multiple settings. Further, while MW and IW involve re-fitting the PS parameters from scratch, OPS makes a constant time update at each step, hence the overall computational complexity of OPS is $O(T)$.

Brief note on theory and limitations of OPS. OPS satisfies a regret bound with respect to the

Platt scaling class for log-loss [Gupta and Ramdas, 2023, Theorem 2.1]. This means that the OPS forecasts do as well as forecasts of the single best Platt scaling model in hindsight. However, we note that OPS could fail if the best Platt scaling model is itself not good. This limitation can be overcome by combining OPS with a method called calibeating, as discussed in Gupta and Ramdas [2023]. We do not pursue calibeating in this paper since OPS already performs well on the data we considered.

6.3 Real-world case studies

We study parity calibration in three real-world scenarios: 1) forecasting COVID-19 cases in the United States, 2) forecasting weather, and 3) predicting plasma state evolution in nuclear fusion experiments. This diverse set of domains, datasets, and expert forecasters provides an attractive test-bed to demonstrate the parity calibration concept and the performance of the calibration methods from Section 6.2.2.

In each setting, the prediction target is real-valued, and we assume an expert forecaster provides regression forecasts $\hat{F}_t$ for the target. We also refer to $\hat{F}_t :\rightarrow [0,1]$ as the *base regression model*. The expert forecaster implicitly provides parity probabilities $\hat{p}_t$ (following Eq. (6.1)). We refer to $\hat{p}_t$ as the *prehoc* probabilities, in contrast to the *posthoc* probabilities that the calibration methods produce. We calibrate $\hat{p}_t$ with the calibration methods from Section 6.2.2 to produce the posthoc probabilities $\hat{p}'_t$. Each calibration method requires a set of hyperparameters, which we tune with a validation set. Details regarding hyperparameter tuning are provided in Appendix C.

Metrics. Given a test dataset $\mathcal{D}_{\text{test}} = \{t, y_t\}_{t=1}^{T}$, we initially assess the quantile calibration of $\hat{F}_t$ and the parity calibration of $\hat{p}_t$ and $\hat{p}'_t$ by visualizing the reliability diagrams and measuring calibration errors.

To assess quantile calibration of $\hat{F}_t$, we produce the reliability diagram using the Uncertainty Toolbox [Chung et al., 2021a], which takes a finite set of quantile levels $\mathcal{P} = \{p_i \in [0,1]\}$, computes the empirical coverage of the predictive quantile $\hat{F}_t^{-1}(p_i)$ as $p_{i,\text{obs}} = \frac{1}{T}\sum_{t=1}^{T} y_t \leq \hat{F}_t^{-1}(p_i)$, and plots each p_i against $p_{i,\text{obs}}$. Calibration error is then summarized into a single scalar with Quantile Calibration Error (QCE), which is computed as $\frac{1}{|\mathcal{P}|}\sum_i \mid p_{i,\text{obs}} - p_i \mid$. In our experiments, we set $\mathcal{P}$ to be 100 equi-spaced quantile levels in $[0,1]$.

To assess parity calibration of a parity probability $\hat{p}_t$, we follow the standard method of producing reliability diagrams in binary calibration [DeGroot and Fienberg, 1981, Niculescu-Mizil and Caruana, 2005]. Noting that $\hat{p}_t$ is a predicted probability of the binary parity outcome $\widetilde{y}_t := y_t \leq y_{t-1}$, we first bin $\hat{p}_t$ into a finite set of fixed width bins $\mathcal{B} = \{B_m\}$, then for each bin B_m, we compute the average outcome as $\text{obs}(B_m) = \frac{1}{|B_m|}\sum_{t:\hat{p}_t \in B_m} \widetilde{y}_t = 1$ and the average prediction as $\text{pred}(B_m) = \frac{1}{|B_m|}\sum_{t:\hat{p}_t \in B_m} \hat{p}_t$, and finally, we plot $\text{pred}(B_m)$ against $\text{obs}(B_m)$ to produce the reliability diagram. Parity Calibration Error (PCE) summarizes the diagram following the standard definition of (ℓ_1-)expected calibration error (ECE): $\sum_m \frac{|B_m|}{T} \mid \text{obs}(B_m) - \text{pred}(B_m) \mid$. In our experiments, we set $\mathcal{B}$ to be 30 fixed-width bins: $[0, \frac{1}{30}), [\frac{1}{30}, \frac{2}{30}), \ldots [\frac{29}{30}, 1]$.

For the parity probabilities $\hat{p}_t$ and $\hat{p}'_t$, we additionally report sharpness and two metrics for accuracy: binary accuracy and area under the ROC curve. Sharpness (Sharp) is computed as $\sum_m \frac{|B_m|}{T} \cdot \text{obs}(B_m)^2$ and measures the degree to which the forecaster can discriminate events with different outcomes [Bröcker, 2009]. Binary accuracy (Acc) and area under the ROC curve (AUROC) are computed following their standard definitions in binary classification. Appendix A provides the

Code is available at `https://github.com/YoungseogChung/parity-calibration` full set of details on how each metric is computed. Lastly, in reporting the metrics in numeric tables, we denote each metric with their orientation, e.g. ↑ indicates that a higher value is more desirable and vice versa.

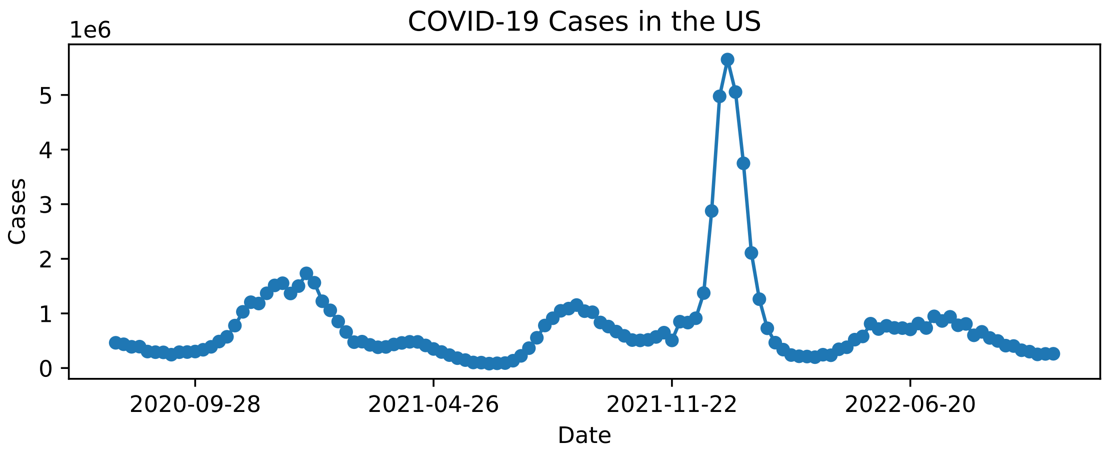


Figure 6.2: Total COVID-19 cases in the US displays high non-stationarity.

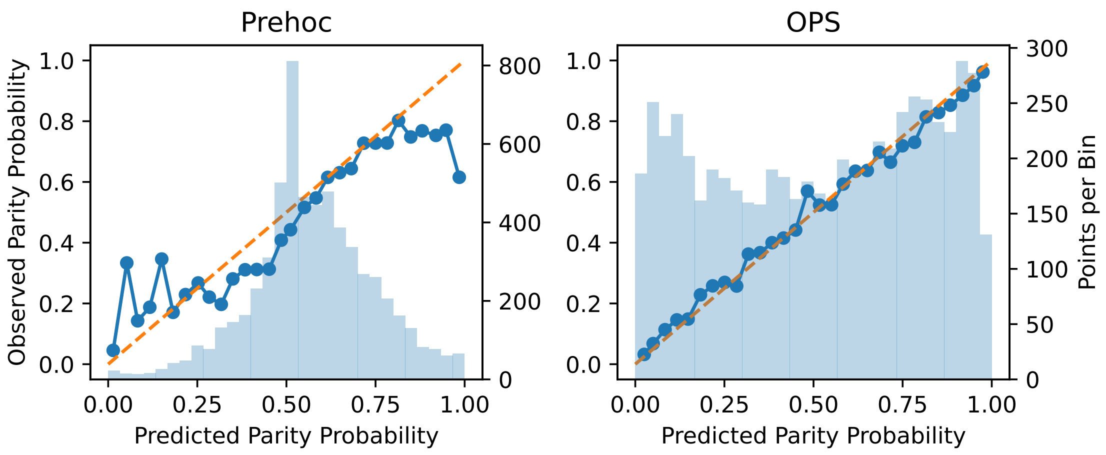


Figure 6.3: Reliability diagrams for the prehoc parity probabilities from the expert forecasts (**left**) and OPS calibrated probabilities (**right**). **Blue bars** denote the frequency of predictions in each bin.

Figure 6.4: The prehoc parity probabilities for the COVID-19 single-timeseries setting are miscalibrated and un-sharp. Posthoc calibration via OPS improves both aspects.

6.3.1 Case Study 1: COVID-19 cases in the US

In response to the COVID-19 pandemic, research groups across the world have created models to predict the short-term future of the pandemic. The COVID-19 Forecast Hub [Cramer et al., 2021] solicits and collects quantile forecasts of weekly incident COVID-19 cases in each US state (plus Washington D.C.), among other targets. Each week, the Hub generates an ensemble forecast from the dozens of submitted forecasts. This ensemble has proven to be more reliable and accurate than

	Prehoc	OPS$_{\text{alpha-order}}$	OPS$_{\text{rand100}}$
PCE $\downarrow$	0.0599	0.0216	0.0246 ± 0.0002
Sharp $\uparrow$	0.2953	0.3087	0.3090 ± 0.00002
Acc $\uparrow$	0.6309	0.6727	0.6737 ± 0.0001
AUROC $\uparrow$	0.6922	0.7355	0.7357 ± 0.00002

Table 6.1: In the COVID-19 single-timeseries setting, OPS improves the prehoc parity probabilities w.r.t all metrics. $\pm$ indicates mean $\pm$ 1 standard error across 100 state orders.

	Prehoc	MW	IW	OPS
PCE $\downarrow$	0.0599	0.0748	0.0406	**0.0328**
Sharp $\uparrow$	0.2953	0.2882	0.2839	**0.2993**
Acc $\uparrow$	0.6309	0.6237	0.6055	**0.6522**
AUROC $\uparrow$	0.6922	0.6622	0.6403	**0.7035**

Table 6.2: In the COVID-19 sequential-batch setting, OPS outperforms prehoc and alternative PS methods. Best value for each metric is in bold.

any constituent individual forecast in predicting other targets of interest (e.g. mortality [Cramer et al., 2022]). Thus, we take the ensemble forecast as the expert forecast and use its historical forecasts made between 2020-07-20 and 2022-10-24, which span a total of 119 weeks. Denoting the target y as the number of cases, there are effectively 51 timeseries, $\{y_{s,t}\}$: one for each US state $s \in \{$Alabama, Alaska, Arizona, ..., Wisconsin, Wyoming$\}$, and $t \in \{1, \ldots, 119\}$. For any given s, t, the expert forecast is provided by the Hub as seven forecasted quantiles for the distribution of $y_{s,t}$. Therefore, we must interpolate the quantiles to produce $\hat{F}_t$ (see Appendix B.1 for details).

The observed targets $y_{s,t}$ are the incident number of cases actually reported from each state, for each week. Figure 6.2 visualizes a summary of the target timeseries: the total incident number of cases in the US $(= \sum_s y_{s,t})$. We can observe high non-stationarity, with periods of rapid increases and falls, and other periods of long monotonic trends.

Parity calibration of expert forecasts and OPS

Note that the underlying timeseries $\{y_{s,t}\}$ is indexed by both state and time. We transform this to a fully sequential timeseries by concatenating $\{y_{s,t}\}$ chronologically across t and in alphabetical order across s. In other words, within a given week, we observe the number of cases for the states in alphabetical order. We refer to this experiment setting as the *single-timeseries* setting.

The reliability diagram in Figure 6.3 (left) shows that the prehoc probabilities implied by the expert forecast ($\hat{p}_t$) are parity calibrated in the $[0.25, 0.75]$ region (i.e. higher predicted probabilities result in higher empirical frequencies), but are miscalibrated otherwise. The distribution of $\hat{p}_t$ displayed by the blue bars further indicate that $\hat{p}_t$ is centered around 0.5, an uninformative or less sharp prediction.

Figure 6.3 (right) displays the reliability diagram of $\hat{p}_t$. We observe significant improvements in both parity calibration and sharpness, i.e. $\hat{p}_t$ is much more dispersed compared to $\hat{p}_t$. The second column of Table 6.1 (labeled OPS$_{\text{alpha-order}}$) show these improvements via the PCE and Sharp metrics, and we can also observe improvement in accuracy.

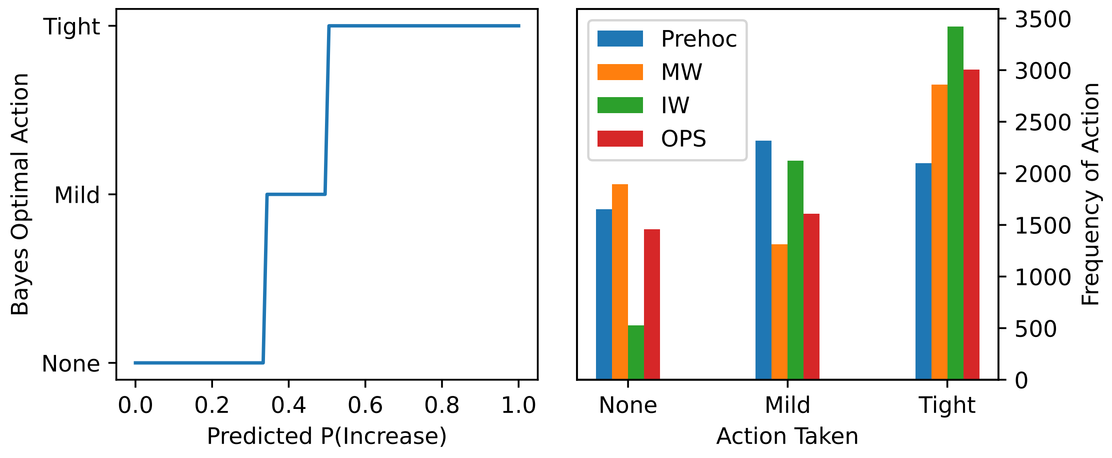


Figure 6.5: (Decision making on the COVID-19 dataset) (**left**) The Bayes optimal action for each predicted probability of increase in number of cases. (**right**) Frequency of each action taken by each method.

One may question whether this improvement by OPS is specific to the alphabetical order of states. In the third column of Table 6.1 (labeled $\text{OPS}_{\text{rand100}}$), we show the mean and standard error of each of the metrics across 100 different random orders of the states, and observe that the improvements provided by OPS over prehoc are fairly robust.

Comparing calibration methods

We perform an additional experiment to compare the performance of MW, IW and OPS. In this experiment, we assume a more realistic test setting for the data-stream. At each timestep t, we assume we observe cases from all 51 states, $\{y_{s,t}\}_{s=1}^{51}$, and update the PS parameters with this batch of data. We then fix the PS parameters and calibrate the next full batch of predictions for timestep $t+1$. This settings assumes that PS parameters are updated once per week based on all the data observed during the week. We refer to this experiment setting as the *sequential-batch* setting.

The first 20 weeks of data (i.e. 20 weeks $\times$ 51 states = 1020 datapoints) were used to tune the hyperparameters of each method. The subsequent 99 weeks of data was used for testing. Table 6.2 displays the results of the sequential batch setting (note that the prehoc values are the same for this setting as in Table 6.1). OPS is the best performing method on all metrics when compared with MW, IW, and prehoc.

Decision-making with parity probabilities

In this section, we demonstrate the utility of OPS in a decision-making setting where parity outcomes (Eq. (6.5)) dictate the loss incurred. Using the same COVID-19 dataset, we assume a setting where a policymaker (i.e. the decision-maker) at each timestep must decide among three levels of restrictions for disease spread prevention: Tight, Mild, or None. For any chosen level of restriction, the loss is dictated by the parity outcome in the number of cases, and the policymaker's goal is to minimize cumulative loss. A *Bayes optimal* policymaker will always choose an action which minimizes the expected loss, calculated with a predictive distribution over the loss [Lehmann and Casella,

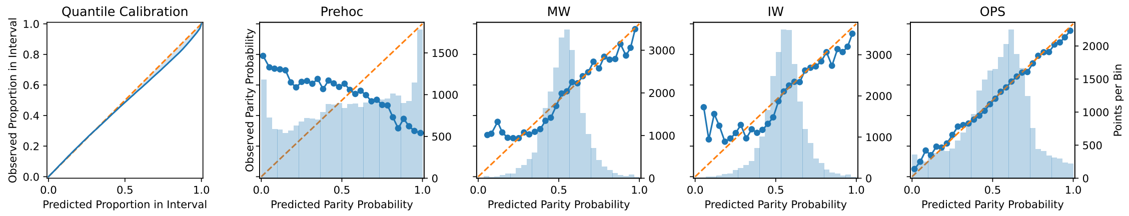


Figure 6.6: OPS significantly improves both parity calibration and sharpness of the base regression model predicting Pressure. The left two plots display the quantile calibration and parity calibration of the base model (Prehoc): it is nearly perfectly quantile calibrated, but terribly parity calibrated. **Blue bars** denote the frequency of predictions in each bin.

2006]. Hence the policymaker will assess the optimality of each action based on predicted parity probabilities.

We design an exemplar loss function $l_{\text{truth, decision}}$ as follows:

# Cases	Tight = 1	Mild = 2	None = 3
Increase = 1	$l_{1,1} = 0.3$	$l_{1,2} = 0.6$	$l_{1,3} = 1$ (max)
Decrease = 2	$l_{2,1} = 0.5$	$l_{2,2} = 0.2$	$l_{2,3} = 0$ (min)

Given this loss function, the Bayes optimal action is visualized in Figure 6.5 (left). On computing the the cumulative loss incurred with the predicted parity probabilities, we find that OPS incurs the lowest cumulative loss.

	Prehoc	MW	IW	OPS
Loss $\downarrow$	2119	2177	2196	**2050**

Figure 6.5 (right) shows the frequency of each action chosen by each method. We observe that OPS chooses Mild with relatively low frequency, which is a result of sharper and more accurate parity probabilities. We further note that IW results in a worse loss than prehoc despite being better parity calibrated (Table 6.2). To understand this, notice that IW is also less sharp and less accurate than Prehoc. Thus calibration, while a desirable quality, is not the only aspect to assess for good uncertainty quantification—sharpness and accuracy could also affect decision making.

6.3.2 Case Study 2: Weather forecasting

Our second case study examines weather forecasting using the benchmark Jena climate modeling dataset [2016], which records the weather conditions in Jena, Germany, with 14 different measurements, in 10 minute intervals, for the years 2009—2016. We did not have access to historical predictions from an expert weather forecaster, so instead we trained our own base regression model.

We follow the Keras tutorial on *Timeseries Forecasting for Weather Prediction*[1] to define our specific problem setup and train our base regression model. In summary, the regression model is implemented with an LSTM network [Hochreiter and Schmidhuber, 1997] which predicts the mean

[1]https://keras.io/examples/timeseries/timeseries_weather_forecasting/

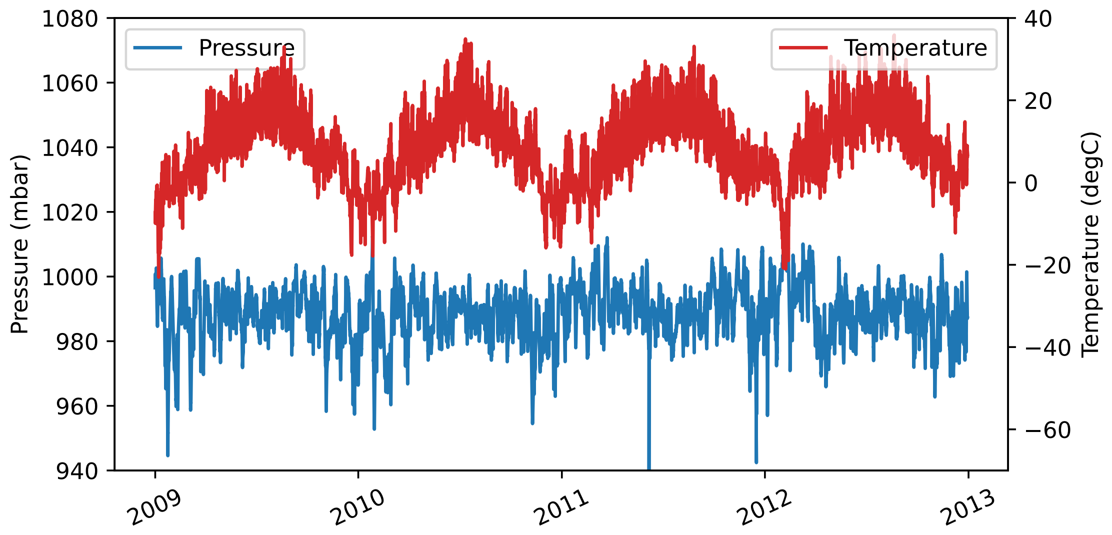


Figure 6.7: Snapshots of 4 years from the Temperature and Pressure timeseries display noise around a cyclical trend.

and variance of a Gaussian distribution. We trained 7 different models that each predict one of 7 weather features: Pressure, Temperature, Saturation vapor pressure, Vapor pressure deficit, Specific humidity, Airtight, and Wind speed. Appendix B.2.1 provides more details on the problem setup.

Lastly, we note that unlike the COVID-19 data, the weather data (Figure 6.7) displays high levels of noise around a cyclical, repeating trend.

	QCE ↓	PCE ↓	Sharp ↑	Acc ↑	AUROC ↑
Prehoc	**0.0181±0.0026**	0.3493±0.0015	0.3019±0.0004	0.4044±0.0006	0.3525±0.0012
MW	N/A	0.0278±0.0005	0.3005±0.0004	0.6124±0.0008	0.6410±0.0012
IW	N/A	0.0322±0.0005	0.3013±0.0004	0.6147±0.0009	0.6450±0.0013
OPS	N/A	**0.0148±0.0002**	**0.3172±0.0004**	**0.6525±0.0007**	**0.7056±0.0010**

Table 6.3: OPS improves the overall quality of parity probabilities from the base regression model predicting Pressure. ± indicates mean ± 1 standard error, across 50 test trials. Best value for each metric is in bold.

Results on Pressure timeseries. We first examine results from one of the 7 models predicting Pressure. Figure 6.6 displays quantile calibration (i.e. probabilistic calibration) of the base model, and parity calibration before and after MW, IW and OPS are applied to the prehoc parity probabilities. We first note that the base model is almost perfectly quantile calibrated, but terribly parity calibrated, which corroborates our argument from Section 6.1.1, that calibration in regression does not imply parity calibration. In the same plot, we can see that MW, IW and OPS are all able to improve parity calibration, but the numerical results in Table 6.3 show that OPS produces superior parity probabilities w.r.t. all of the metrics considered.

Binary classifiers as expert forecasters. While we have so far assumed that the expert forecaster

	PCE ↓	Sharp ↑	Acc ↑	AUROC ↑
Prehoc	0.0258±0.0005	0.3008±0.0007	0.6069±0.0011	0.6474±0.0016
MW	0.0201±0.0005	0.3002±0.0007	0.6050±0.0012	0.6439±0.0017
IW	0.0166±0.0003	0.3003±0.0008	0.6068±0.0010	0.6456±0.0016
OPS	**0.0150±0.0001**	**0.3232±0.0006**	**0.6665±0.0007**	**0.7275±0.0007**

Table 6.4: While MW, IW, OPS all improve parity calibration of the base classification model for Pressure (Prehoc), OPS is the only method that improves all metrics simultaneously. ± indicates mean ± 1 standard error, across 50 test trials. Best value for each metric is in bold.

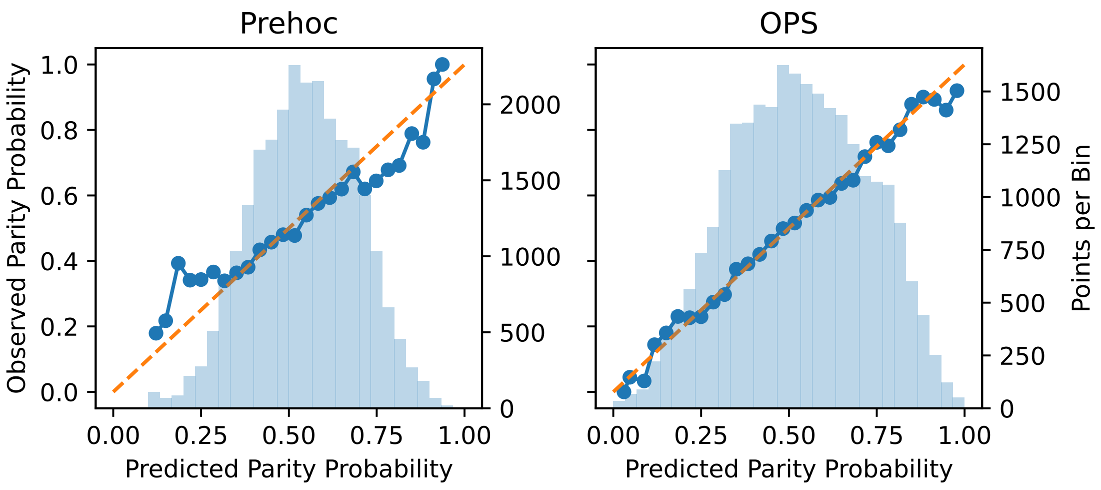


Figure 6.8: The base classification model for Pressure (Prehoc) is better parity calibrated than the base regression model (Figure 6.6 Prehoc), but OPS still improves its parity calibration and sharpness.

provides regression models $\hat{F}_t$, one may argue that an expert forecaster may be well-aware that the downstream user is primarily concerned with parity probabilities. Accordingly, the expert may choose to directly model parity probabilities in the context of a binary classification problem.

In Figure 6.8 and Table 6.4, we show results from training a base binary classifier with parity outcome labels and applying posthoc calibration. As expected, the prehoc parity probabilities of the binary classification model is significantly better calibrated than the regression model. Posthoc calibration still improves parity calibration further, especially in the case of OPS. In fact, OPS is the only method which improves all of the metrics simultaneously, while MW and IW notably worsen sharpness and AUROC. The full set of reliability diagrams is provided in Figure 10 in Appendix B.2.2.

Results across all 7 timeseries. Table 6 in Appendix B.2.2 shows each metric averaged across all 7 prediction targets: Table 6a displaying results with the base regression model, and 6b that of the base classification model. The pattern observed for the Pressure timeseries tend to hold on average across all 7 timeseries.

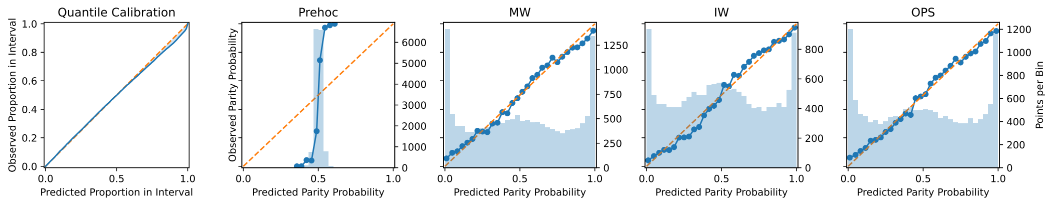


Figure 6.9: All methods (MW, IW, OPS) perform equally well in calibrating the Prehoc parity probabilities of the nuclear fusion dynamics model. The left two plots display the quantile calibration and parity calibration of the base dynamics model.

	QCE ↓	PCE ↓	Sharp ↑	Acc ↑	AUROC ↑
Prehoc	**0.0108±0.0003**	0.2571±0.0003	0.3243±0.0002	**0.7727±0.0003**	**0.8536±0.0002**
MW	N/A	0.0266±0.0002	0.3345±0.0002	0.7665±0.0003	0.8463±0.0002
IW	N/A	0.0291±0.0002	**0.3385±0.0002**	**0.7726±0.0003**	**0.8533±0.0002**
OPS	N/A	**0.0261±0.0002**	0.3334±0.0002	0.7629±0.0002	0.8440±0.0002

Table 6.5: MW, IW, and OPS all improve parity calibration and sharpness of the Prehoc fusion dynamics model predicting β_N, while maintaining roughly the same level of accuracy. $\pm$ indicates mean $\pm$ 1 standard error, across 50 test trials. Best value for each metric is in bold.

6.3.3 Case Study 3: Model-based Control for nuclear fusion

Nuclear fusion is the physical process during which atomic nuclei combine together to form heavier atomic nuclei, while releasing atomic particles and energy. Although fusion is possibly a safe, clean, and fuel-abundant technology for the future [Morse, 2018], there are various challenges to realizing fusion power, one of which is controlling nuclear fusion reactions [Humphreys et al., 2015].

Recently, model-based control methods, where a dynamics model of the system is learned and used to optimize control policies, has emerged as an effective control method for fusion devices [Abbate et al., 2023]. To the experimenter utilizing the dynamics model, it is of significant interest to know when certain signals will increase, and whether the dynamics model assigns correct probabilities to the events [Char et al., 2021]. In this section, we consider the problem of predicting the parity of β_N, which is a signal indicating reaction efficiency in a fusion device called a tokamak.

To this end, we design our empirical case study as follows. We take a pretrained dynamics model which was trained with a logged database of 10294 fusion experiments (referred to as "shots") conducted on the DIII-D tokamak [Luxon, 2002], a device in San Diego, CA, USA. This pretrained model has been used for model-based policy optimization for deployment in actual fusion experiments on this device [Char et al., 2021, Seo et al., 2021, Abbate et al., 2021]. The model architecture is a recurrent probabilistic neural network (RPNN), which is a recurrent neural network with a Gaussian output head. We refer the reader to Appendix B.3.1 for more details of the dynamics model and dataset. For testing, we allocate a set of 900 held-out test shots. On this test set, we produce the model's distributional predictions for β_N as the expert forecast. We concatenate the forecasts and the actual observed β_N values across the 900 test shots in chronological order into a single timeseries to assess parity calibration.

Figure 6.9 and Table 6.5 indicate that the expert forecast (Prehoc) is quantile calibrated but parity miscalibrated. The accuracy metrics in Table 6.5 indicate that despite prehoc's poor parity

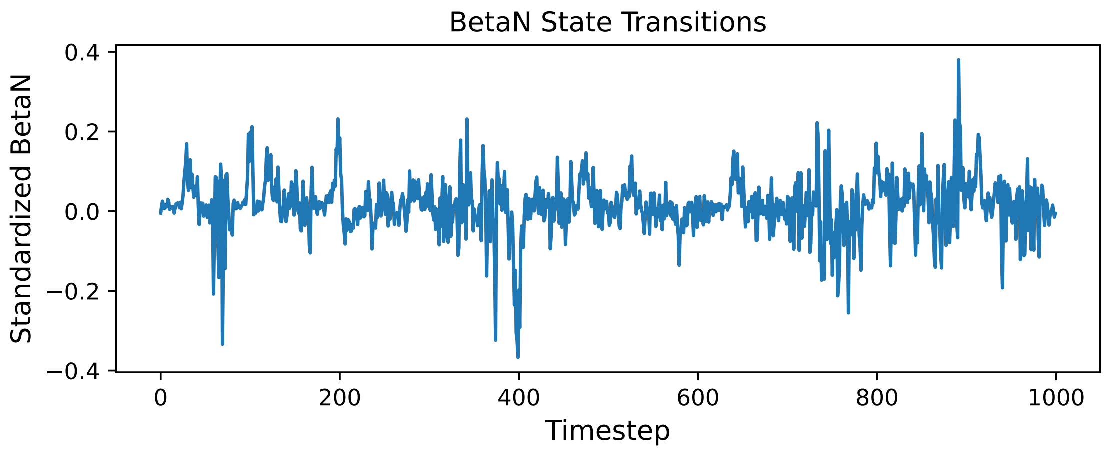


Figure 6.10: State transitions of the β_N signal during nuclear fusion experiments ("shots") concatenated across 50 training shots resemble trend-less noise.

calibration, the model is still highly predictive, with an AUROC > 0.85. MW, IW and OPS significantly improve parity calibration and sharpness, while maintaining roughly the same level of accuracy.

We note that the β_N timeseries, as displayed in Figure 6.10, tends to fluctuate rapidly, between timesteps and between shots, almost resembling white noise. The pretrained model still manages to model the signal well, and assigns correct tendencies of increases/decreases in β_N: the relibility diagram of prehoc in Figure 6.9 shows that although the parity probabilities are not aligned with the empirical frequencies, they predict higher probabilities for actually higher frequency events. We believe this provides for a relatively easy posthoc calibration problem, thus all methods (MW, IW, OPS) perform equally well. Hence, this case study highlights the significance of the base model's initial parity probabilities, especially in alleviating the difficulty of posthoc calibration.

6.4 Conclusion

We considered the problem of forecasting whether a continuous-valued sequence is going to increase or decrease at the next time step. Such scenarios, where relative changes are more interpretable than actual values, are ubiquitous: COVID-19 cases per day, weather, or stock prices. In this context, we proposed the notion of parity calibration. To be parity calibrated, a forecaster must predict probabilities for the outcome increasing at the next time step, and these probabilities should be calibrated in the binary sense.

A decision-maker may attempt to achieve parity calibration by using regression forecasts produced by an expert forecaster. However, this is unlikely to give parity calibration. Instead, we proposed the usage of posthoc binary calibration techniques to achieve parity calibration. Specifically, we advocated for a recently proposed online Platt scaling algorithm (OPS) in this setting. In three real-world empirical case studies, OPS consistently improves the overall quality of parity probabilities compared to the expert forecaster.

Appendices for Chapter 6

6.A Details on Evaluation: reliability diagrams and metrics

We provide details on how we assess a sequence of distributional forecasts $\{\hat{F}_t\}_{t=1}^T$ and parity probabilities $\{\hat{p}_t\}_{t=1}^T$, given a test dataset $\mathcal{D}_{\text{test}} = \{_t, y_t\}_{t=1}^T$. We assess distributional forecasts via Quantile Calibration, and the parity probabilities via Parity Calibration, Sharpness, and Accuracy metrics.

- **Quantile Calibration: reliability diagram and calibration error**

  To assess the quantile calibration of the distributional forecast $\hat{F}_t$, we produce the reliability diagram using the *Uncertainty Toolbox* [Chung et al., 2021a]. This process works as follows. We take 100 equi-spaced quantile levels in $[0, 1]$: $p_i \in$ `np.linspace(0, 1, 100)`, and for each p_i, we compute the empirical coverage of the predictive quantile $\hat{F}_t^{-1}(p_i)$ with $\frac{1}{T}\sum_{t=1}^T y_t \leq \hat{F}_t^{-1}(p_i)$, and we denote this quantity as $p_{i,\text{obs}}$. Note that $p_{i,\text{obs}}$ is an empirical estimate of the term $\frac{1}{T}\sum_{t=1}^T F_t(\hat{F}_t^{-1}(p_i))$, from Eq. (6.3). The reliability diagram is produced by plotting $\{p_i\}$ on the x-axis against $\{p_{i,\text{obs}}\}$ on the y-axis. Quantile Calibration Error (QCE) is then computed as the average of the absolute difference between p_i and $p_{i,\text{obs}}$ over the 100 values of p_i: $\frac{1}{100}\sum_{i=1}^{100} \mid p_{i,\text{obs}} - p_i \mid$.

- **Parity Calibration: reliability diagram and calibration error**

  For parity calibration, we produce the reliability diagram following the standard method in binary classification [DeGroot and Fienberg, 1981, Niculescu-Mizil and Caruana, 2005]. Note that the parity probability $\hat{p}_t$ is a prediction for the parity outcome $\widetilde{y}_t := y_t \leq y_{t-1}$ (Eq. (6.5)). Specifically, we first take 30 fixed-width bins of the predicted parity probabilities: $\{B_m\}_{m=1}^{30}$, where $B_m = [\frac{m-1}{30}, \frac{m}{30})$ for $m < 30$ and $B_{30} = [\frac{29}{30}, 1]$. The average outcome in bin B_m is computed as $\text{obs}(B_m) = \frac{1}{|B_m|}\sum_{t:\hat{p}_t \in B_m} \widetilde{y}_t = 1$, and the average prediction of bin B_m is computed as $\text{pred}(B_m) = \frac{1}{|B_m|}\sum_{t:\hat{p}_t \in B_m} \hat{p}_t$. The reliability diagram is then produced by plotting $\text{pred}(B_m)$ on the x-axis against $\text{obs}(B_m)$ on the y-axis. The blue bars in the background of each parity calibration reliability diagram represents the size of the bin: $|B_m|$. Parity Calibration Error (PCE) is then computed with this reliability diagram following the standard definition of (ℓ_1-)expected calibration error (ECE): $\sum_{m=1}^{30} \frac{|B_m|}{T} \mid \text{obs}(B_m) - \text{pred}(B_m) \mid$.

- **Sharpness**

  Assuming the same notation as above, sharpness is computed as: $\sum_{m=1}^{M} \frac{|B_m|}{T} \cdot \text{obs}(B_m)^2$, where M is the total number of bins. As indicated above, we use $M = 30$ in all of our experiments. We provide some additional intuition on this metric. A perfectly knowledgeable forecaster which outputs $\hat{p}_t = \widetilde{y}_t$ will place all predictions in either B_1 or B_M and achieve sharpness $= \frac{|B_1|}{T} \cdot \text{obs}(B_1)^2 + \frac{|B_M|}{T} \cdot \text{obs}(B_M)^2 = \frac{|B_1|}{T} \cdot 0^2 + \frac{|B_M|}{T} \cdot 1^2 = \frac{|B_M|}{T} = \frac{\sum_{t=1}^{T} \widetilde{y}_t}{T}$. On the other hand, if the forecaster places all predictions into a single bin B_k, then its sharpness will be $\text{obs}(B_k)^2 = \left(\frac{\sum_{t=1}^{T} \widetilde{y}_t}{T}\right)^2$. It can be shown that sharpness is always within the closed interval $\left[\left(\frac{\sum_{t=1}^{T} \widetilde{y}_t}{T}\right)^2, \frac{\sum_{t=1}^{T} \widetilde{y}_t}{T}\right]$ [Bröcker, 2009]. Intuitively, sharpness measures the degree to which the forecaster attributes different valued predictions to events with different outcomes (i.e. labels). Hence, a sharper, or more precise, forecaster has more discriminative power, and this is reflected in a higher sharpness metric.

- **Accuracy metrics (Acc and AUROC)**

  Accuracy is measured in the binary classification sense, where the true labels are the observed parity outcomes: $y_t \leq y_{t-1}$ (Eq. (6.5)).

  - **Binary accuracy (Acc)** is computed by regarding $\hat{p}_t \geq 0.5$ as the positive class prediction, and the opposite case as the negative class prediction.
  - **Area under the ROC curve (AUROC)** is computed using the `scikit-learn` Python package, which implements the standard definition of the score. Specifically, we called the function `sklearn.metrics.roc_auc_score` with the predictions $\{\hat{p}_t\}$ and labels $y_t \leq y_{t-1}$.

6.B Additional Details on Case Studies

6.B.1 Additional Details on COVID-19 Case Study

Details on Interpolating Expert Forecasts for COVID-19 Case Study

The expert forecast provided by the COVID-19 Forecast Hub is represented as a set of quantiles. To derive the parity probabilities $\hat{p}_{s,t}$, we need to interpolate the expert forecast, as the forecast contains predicted quantiles at only 7 quantile levels : $\{0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 0.975\}$. We interpolate under the assumption that the density between two adjacent quantiles τ_k and τ_{k+1} are defined by the normal distribution specified by those two quantiles. Specifically, for two quantiles τ_k and τ_{k+1} and forecast values $x_k^{(s,t)}$ and $x_{k+1}^{(s,t)}$, we compute

$$\sigma_k^{(s,t)} = \frac{x_{k+1}^{(s,t)} - x_k^{(s,t)}}{\Phi^{-1}(\tau_{k+1}) - \Phi^{-1}(\tau_k)},$$

$$\mu_k^{(s,t)} = x_k^{(s,t)} - \sigma_k^{(s,t)} \Phi^{-1}(\tau_k),$$

where Φ is the standard normal cdf. For each forecast, if $x_k^{(s,t)} \leq y_{s,t-1} < x_{k+1}^{(s,t)}$, then the parity probability

$$\hat{p}_{s,t} = \Phi\left(\frac{y_{s,t-1} - \mu_k^{(s,t)}}{\sigma_k^{(s,t)}}\right).$$

If $y_{s,t-1} < x_1^{(s,t)}$, we can extrapolate using $\mu_1^{(s,t)}$ and $\sigma_1^{(s,t)}$, and if $y_{s,t-1} >= x_7^{(s,t)}$, we can extrapolate using $\mu_6^{(s,t)}$ and $\sigma_6^{(s,t)}$. However, this never occurs with the forecasts and observations in this dataset. Figure 6.11 provides a visualization of this interpolation scheme.

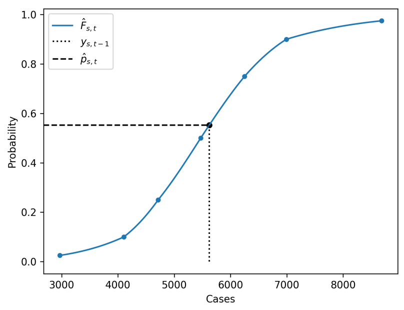


Figure 6.11: We use a piece-wise Gaussian interpolation of the expert forecast quantiles to estimate the predictive cdf, from which we then calculate the parity probabilities.

Details on Experiment Setup for COVID-19 Case Study

Section 6.3.1 compares the expert forecaster, its parity probabilities and posthoc calibration by OPS. We did not tune OPS hyperparameters in this experiment, so the full 119 weeks' worth of data was used for testing and reporting the results.

For Section 6.3.1, the first 20 weeks' worth of data was used for tuning hyperparameters, and the reported results are based on the remaining 99 weeks' worth of data as the test set.

For the decision-making experiment in Section 6.3.1, we used the parity probabilities produced from Section 6.3.1.
Although the chosen loss function is just one example, we observe that similar results hold with any loss function that satisfies: $l_{2,3} \leq l_{2,2} \leq l_{1,1} \leq l_{2,1} \leq l_{1,2} \leq l_{1,3}$.

6.B.2 Additional Details on Weather Forecasting Case Study

Details on Experiment Setup for Weather Forecasting Case Study

We used the modeling and training infrastructure provided by the Keras tutorial on *Timeseries Forecasting for Weather Prediction*[2] which models this same dataset with an LSTM network [Hochreiter and Schmidhuber, 1997]. We made one change to the model provided by the tutorial: since we are interested in probabilistic forecasts instead of point forecasts, we changed the head of the model and the loss function from a point output trained with mean squared error loss to a mean and variance output that parameterizes a Gaussian distribution and trained it with the Gaussian likelihood loss. Such a model is also referred to as a mean-variance network or a probabilistic neural network [Lakshminarayanan et al., 2017, Nix and Weigend, 1994], and it is one of the most popular methods currently used in probabilistic regression.

While the tutorial's setup takes as input the past 120 hours' window of 7 features to predict the value of one feature (Temperature) 12 hours into the future, we expand the setting to predict all 7 features: Pressure, Temperature, Saturation vapor pressure, Vapor pressure deficit, Specific humidity, Airtight, and Wind speed. We thus train 7 separate base regression models, one for each prediction target.

For the in-text experiment **Binary classifers as expert forecasts**, we trained binary classification base models with parity outcomes (Eq. (6.5)) as the labels and took this model as the expert forecaster. We adopted the same model architecture as the base regression model and changed the last layer to

[2]`https://keras.io/examples/timeseries/timeseries_weather_forecasting/`

output a logit. We then trained the model with the cross entropy loss.

The full Jena dataset spans from the beginning of January 2009 to the end of December 2016, with $420,551$ datapoints in total. In chronological order, we set $272,638$ datapoints to train the base models (both the regression and classification model) and the subsequent $83,390$ datapoints for validation. Following the same model training procedure as the tutorial, training was stopped early if the validation loss did not increase for 20 training epochs.

Afterwards, in running the posthoc calibration methods (MW, IW, and OPS), we used the last $8,640$ datapoints of the validation set to tune the hyperparameters of each calibration method, and used subsequent windows of $8,640 \times 3 = 25,920$ datapoints for testing.

We run 50 test trials with a moving test timeframe to produce the mean and standard errors reported in Tables 6.3 and 6.4. Denoting the first test window as $[t+1, t+H]$ (i.e. H is set to $25,920$), we move this frame by a multiple of a fixed offset c into the future, and repeat this 50 times, to create a new set of 50 test sets. The resulting new test timeframes are $[t+1+(ck), t+H+(ck)]$, where $k = 0, 1, 2, \ldots 49$, and c was set to 336.

Additional Results on Weather Forecasting Case Study

We shows additional plots and tables from the experimental results in Section 6.3.2 of the main paper.

Figure 6.12 displays the full set of reliability diagrams for Figure 6.8, which corresponds to the in-text experiment **Binary classifiers as expert forecasts** in Section 6.3.2.

Table 6.8 displays the numerical results from the weather forecasting case study when averaged across all 7 prediction target settings. This corresponds to the in-text experiment **Results across all 7 timeseries** in Section 6.3.2. To produce these results, we fixed the test timeframe to be the first test timeframe $[t+1, t+H]$ for all prediction target settings, then computed the mean and standard errors across the 7 sets of metrics produced (one set for each prediction target).

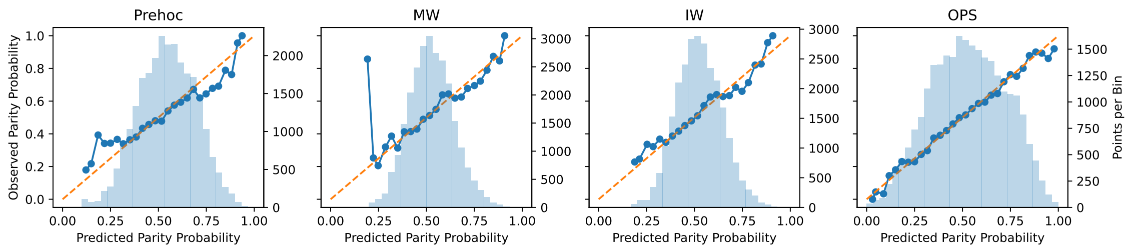


Figure 6.12: Reliability diagrams with a binary classification base model predicting Pressure. This is the full set of reliability diagrams for Figure 6.8 from Section 6.3.2. The left-most plot shows parity calibration of the base classification model (Prehoc), and the next three plots show the effects of MW, IW and OPS in calibrating the Prehoc parity probabilities. OPS produces the most calibrated and sharp parity probabilities.

6.B.3 Additional Details on Control in Nuclear Fusion Case Study

Details on Experiment Setup for Control in Nuclear Fusion Case Study

The expert forecaster for the nuclear fusion experiment in Section 6.3.3 is provided by a pretrained dynamics models that was used to optimize control policies for deployment on the DIII-D toka-

	QCE ↓	PCE ↓	Sharp ↑	Acc ↑	AUROC ↑
Prehoc	$\mathbf{0.0266 \pm 0.0052}$	0.2794 ± 0.0161	0.2915 ± 0.0117	0.4902 ± 0.0159	0.4806 ± 0.0249
MW	N/A	0.0233 ± 0.0048	0.2913 ± 0.0117	0.5610 ± 0.0106	0.5419 ± 0.0195
IW	N/A	0.0188 ± 0.0047	0.2913 ± 0.0118	0.5630 ± 0.0099	0.5403 ± 0.0209
OPS	N/A	$\mathbf{0.0159 \pm 0.0009}$	$\mathbf{0.2961 \pm 0.0122}$	$\mathbf{0.5790 \pm 0.0122}$	$\mathbf{0.5830 \pm 0.0217}$

Table 6.6: Numerical results averaged across all 7 prediction settings where the base model is a Gaussian regression model. The base regression model (Prehoc) tends to be well quantile calibrated (QCE) but terribly parity calibrated (PCE). All methods (MW, IW, OPS) improve parity calibration, but OPS is the only method which improves all metrics simultaneously. Best value for each metric is in bold.

	PCE ↓	Sharp ↑	Acc ↑	AUROC ↑
Prehoc	0.0247 ± 0.0016	0.3049 ± 0.0074	0.6078 ± 0.0099	0.6348 ± 0.0136
MW	0.0170 ± 0.0018	0.3049 ± 0.0075	0.6061 ± 0.0102	0.6340 ± 0.0143
IW	0.0156 ± 0.0012	0.3047 ± 0.0074	0.6075 ± 0.0098	0.6340 ± 0.0136
OPS	$\mathbf{0.0135 \pm 0.0013}$	$\mathbf{0.3134 \pm 0.0075}$	$\mathbf{0.6278 \pm 0.0121}$	$\mathbf{0.6643 \pm 0.0183}$

Table 6.7: Numerical results averaged across all 7 prediction settings where the base model is a binary classification model trained with parity outcome labels. The base classification model (Prehoc) tends to be much better parity calibrated than when a regression base model is used (above Table 6.6). All methods (MW, IW, OPS) improve parity calibration further, but OPS is the only method which improves all metrics simultaneously. Notably, MW and IW tends to decrease the accuracy of the parity probabilities. Best value for each metric is in bold.

Table 6.8: Numerical results from the weather forecasting case study (Section 6.3.2), averaged across all 7 forecasting targets. Table 6.6 displays results with the Gaussian regression base model, and Table 6.7 displays results with the binary classification base model. $\pm$ indicates mean $\pm$ 1 standard error, across the 7 prediction target settings.

mak [Luxon, 2002], a nuclear fusion device in San Diego that is operated by General Atomics. The dynamics model was trained with logged data from past experiments (referred to as “shots”) on this device. Each shot consists of a trajectory of (state, action, next state) transitions, and one trajectory consists of $\sim$ 20 transitions (i.e. 20 timesteps).

As input, the model takes the current state of the plasma and the actuator settings (i.e. actions). The model outputs a multi-dimensional predictive distribution over the state variables in the next timestep. The state is represented by three signals: β_N (the ratio of plasma pressure over magnetic pressure), *density* (the line-averaged electron density), and *li* (internal inductance). For the actuators, the model takes in the amount of power and torque injected from the neutral beams, the current, the magnetic field, and four shape variables (*elongation*, a_{minor}, *triangularity-top*, and *triangularity-bottom*). This, along with the states, makes for an input dimension of 11 and output dimension of 3 for the states.

The model was implemented with a recurrent probabilistic neural network (RPNN), which features an encoding layer by an RNN with 64 hidden units followed by a fully connected layer with 256 units, and a decoding layer of fully connected layers with [128, 512, 128] units, which finally outputs a 3-dimensional isotropic Gaussian parameterized by the mean and a log-variance prediction.

The training dataset consisted of trajectories from 10294 shots, and the model was trained with the Gaussian likelihood loss, with a learning rate of 0.0003 and weight decay of 0.0001. In using dynamics models to sample trajectories and train policies, the key metric practitioners are concerned with is explained variance, hence explained variance on a held out validation set of 1000 shots was monitored during training. Training was stopped early if there was no improvement in explained variance over the validation set for more than 250 epochs. The test dataset consisted of another held-out set of 900 shots, with which we report all results presented in Section 6.3.3.

In all of our experiments, since β_N is the key signal of interest in our problem setting, we just examine the predictive distribution for β_N in the model outputs and ignore the other dimensions of the outputs.

In running the posthoc calibration methods (MW, IW, and OPS), we used the same validation set to tune the hyperparameters of each calibration method, and used windows of $15,000$ datapoints from the concatenated test shot data for testing.

We run 50 test trials with a moving test timeframe to produce the mean and standard errors reported in Table 6.5. Denoting the first test window as $[t+1, t+H]$ (i.e. H is set to $15,000$), we move this frame by a multiple of a fixed offset c into the future, and repeat this 50 times, to create a set of 50 test datasets. The resulting test timeframes are $[t+1+(ck), t+H+(ck)]$, where $k = 0, 1, 2, \dots 49$, and c was set to 100.

6.C Details on Hyperparameters

Each of the three calibration methods we consider in Section 6.2.2, which we use in our experiments in Section 6.3, requires a set of hyperparameters.

- **MW** requires `uf` and `ws`.
  - `uf` determines how often the PS parameters ($a^{\text{MW}}, b^{\text{MW}}$) are updated.
  - `ws` determines the size of the calibration set that is used to update the PS parameters
- **IW** requires `uf`.
  - `uf` determines how often the PS parameters ($a^{\text{IW}}, b^{\text{IW}}$) are updated.
    Note that IW always uses all of the data seen so far to update the PS parameters.
- **OPS** requires γ and `D`.
  - γ can be understood as step size for the OPS updates.
  - D can be understood as regularization for the OPS updates.

We provide details on how these hyperparameters were tuned for each of the three case studies.

6.C.1 Hyperparameters for COVID-19 Case Study

We observed that OPS performed well with the default hyperparameters, so we did not tune hyperparameters for OPS for the COVID-19 case study. The default hyperparameter values used for OPS were $\gamma = 0.001$ and `D` $= 10$.

For MW and IW, we tuned hyperparameters by optimizing parity calibration error (PCE, Section 6.3) on the first 20 weeks' worth of data as the validation set, over the following grids:

- `uf` $\in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$, separately for MW and IW
- `ws` $\in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$, for MW.

The COVID-19 dataset records data for each week, so the grid size of 1 represents 1 week.

The tuned hyperparameters we used for MW and IW are as follows:

- MW: $\texttt{uf} = 1, \texttt{ws} = 10$
- IW: $\texttt{uf} = 5$

6.C.2 Hyperparameters for Weather Forecasting Case Study

For each calibration method, the hyperparameters were tuned by optimizing parity calibration error (PCE, Section 6.3) on the validation dataset over the following grids:

- $\texttt{uf} \in [1, 24, 168, 336, 720, 2160]$, separately for MW and IW
- $\texttt{ws} \in [24, 168, 336, 720, 2160, 4320, 8640]$, for MW
- $\gamma \in$ [1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2], for OPS
- $\texttt{D} \in [1, 10, 30, 50, 70, 100, 150, 200]$, for OPS.

The hyperparameters were tuned separately for each base model setting (regression and classification), for each method (MW, IW, and OPS), and for each base model predicting one of 7 targets (Pressure, Temperature, Saturation vapor pressure, Vapor pressure deficit, Specific humidity, Airtight, and Wind speed).

The tuned hyperparameters we used are as follows:

- **Base Regression Model**
  - Pressure Model
    - MW: $\texttt{uf} = 2160, \texttt{ws} = 8640$
    - IW: $\texttt{uf} = 2160$
    - OPS: $\gamma =$ 1e-5, $\texttt{D} = 50$
  - Temperature Model
    - MW: $\texttt{uf} = 336, \texttt{ws} = 8640$
    - IW: $\texttt{uf} = 168$
    - OPS: $\gamma =$ 1e-5, $\texttt{D} = 30$
  - Saturation Vapor Pressure Model
    - MW: $\texttt{uf} = 2160, \texttt{ws} = 2160$
    - IW: $\texttt{uf} = 336$
    - OPS: $\gamma =$ 1e-4, $\texttt{D} = 10$
  - Vapor Pressure Deficit Model
    - MW: $\texttt{uf} = 1, \texttt{ws} = 4320$
    - IW: $\texttt{uf} = 1$
    - OPS: $\gamma =$ 1e-3, $\texttt{D} = 1$
  - Specific Humidity Model
    - MW: $\texttt{uf} = 1, \texttt{ws} = 4320$
    - IW: $\texttt{uf} = 168$
    - OPS: $\gamma =$ 1e-5, $\texttt{D} = 30$

    - Airtight Model
        - MW: $\texttt{uf} = 2160, \texttt{ws} = 2160$
        - IW: $\texttt{uf} = 720$
        - OPS: $\gamma = $ 5e-5, $\texttt{D} = 10$
    - Wind Speed Model
        - MW: $\texttt{uf} = 1, \texttt{ws} = 168$
        - IW: $\texttt{uf} = 24$
        - OPS: $\gamma = $ 1e-4, $\texttt{D} = 10$
- **Base Classification Model**
    - Pressure Model
        - MW: $\texttt{uf} = 2160, \texttt{ws} = 8640$
        - IW: $\texttt{uf} = 720$
        - OPS: $\gamma = $ 5e-5, $\texttt{D} = 30$
    - Temperature Model
        - MW: $\texttt{uf} = 1, \texttt{ws} = 4320$
        - IW: $\texttt{uf} = 168$
        - OPS: $\gamma = $ 1e-5, $\texttt{D} = 150$
    - Saturation Vapor Pressure Model
        - MW: $\texttt{uf} = 336, \texttt{ws} = 4320$
        - IW: $\texttt{uf} = 720$
        - OPS: $\gamma = $ 1e-4, $\texttt{D} = 30$
    - Vapor Pressure Deficit Model
        - MW: $\texttt{uf} = 1, \texttt{ws} = 168$
        - IW: $\texttt{uf} = 1$
        - OPS: $\gamma = $ 1e-5, $\texttt{D} = 70$
    - Specific Humidity Model
        - MW: $\texttt{uf} = 1, \texttt{ws} = 2160$
        - IW: $\texttt{uf} = 2160$
        - OPS: $\gamma = $ 1e-5, $\texttt{D} = 50$
    - Airtight Model
        - MW: $\texttt{uf} = 24, \texttt{ws} = 4320$
        - IW: $\texttt{uf} = 336$
        - OPS: $\gamma = $ 1e-3, $\texttt{D} = 10$
    - Wind Speed Model
        - MW: $\texttt{uf} = 24, \texttt{ws} = 2160$
        - IW: $\texttt{uf} = 1$

    – OPS: $\gamma = 1\text{e-}5, \texttt{D} = 10$.

6.C.3 Hyperparameters for Control in Nuclear Fusion Case Study

The nuclear fusion dataset records measurements in 25 millisecond intervals. Therefore, in tuning hyperparameters, we design the search grid to represent lengths of time during which evolution of various plasma states are expected to be observable.

For each calibration method, the hyperparameters were tuned by optimizing parity calibration error (PCE, Section 6.3) on a validation dataset consisting of 1000 shot's worth of data, over the following grids:

- $\texttt{uf} \in [1, 2, 4, 8, 24]$, separately for MW and IW
- $\texttt{ws} \in [2, 8, 16, 24, 48, 60, 80, 100, 200]$, for MW
- $\gamma \in$ [1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2], for OPS
- $\texttt{D} \in [1, 10, 30, 50, 70, 100, 150, 200]$, for OPS

The tuned hyperparameters we used are as follows:

- MW: $\texttt{uf} = 1, \texttt{ws} = 60$
- IW: $\texttt{uf} = 8$
- OPS: $\gamma = 5\text{e-}3, \texttt{D} = 150$.

6.D Online Platt Scaling Algorithm

Algorithm 12 Online Platt Scaling (based on Gupta and Ramdas [2023])

```
**Input:** $\mathcal{K} = \{(x, y) : \|(x, y)\|_2 \leq 100\}$, time horizon $H$, and initialization parameter $(a_1^, b_1^) = (1, 0) =: \theta_1 \in \mathcal{K}$
**Hyperparameters and default values:** $\gamma = 0.1$, $D = 1$, $A_0 = (1/\gamma D)^2\ \mathbf{I}_2$
**for** $t = 1$ **to** $H$ **do**
    Play $\theta_t$, observe log-loss $l(m^{\theta_t}(f(_t)), y_t)$ and its gradient $\nabla_t := \nabla_{\theta_t} l(m^{\theta_t}(f(_t)), y_t)$
    $A_t = A_{t-1} + \nabla_t \nabla_t^\intercal$
    Newton step: $\widetilde{\theta}_{t+1} = \theta_t - \frac{1}{\gamma} A_t^{-1} \nabla_t$
    Projection: $(a_{t+1}^, b_{t+1}^) = \theta_{t+1} =_{\theta \in \mathcal{K}} (\widetilde{\theta}_{t+1} - \theta)^\intercal A_t (\widetilde{\theta}_{t+1} - \theta)$
**end for**
```

7 | Offline Contextual Bayesian Optimization

7.1 Introduction

Nuclear fusion is regarded as the energy of the future since it presents the possibility of unlimited clean energy [Chen, 2011]. The most widespread method of realizing fusion reactions requires heating up isotopes of hydrogen to temperatures of hundreds of millions of degrees using a magnetic device called a *tokamak*. In this state, the nuclei of two nearby atoms may overcome electrostatic repulsion force between them to form a single nucleus, releasing energy. In contrast to nuclear fission, this process creates much less hazardous byproducts and has a plentiful fuel source [Chen, 2011, Clery, 2014]. One obstacle in utilizing fusion as a feasible energy source, however, is the stability of the reaction. At such high temperatures, the hydrogen atoms are in a plasma state, and any instability in the plasma can give rise to events called *disruptions*. If disruptions occur, the plasma is quickly lost and the nuclear reaction is halted. Ideally, one would have a controller for the tokamak that makes actions in response to the current state of the plasma in order to prolong the reaction as long as possible; however, such a controller is unknown as of now.

In this work, we make preliminary steps to learning such a controller. Since learning on a real world tokamak is infeasible, we tackle this problem by attempting to learn optimal controls offline via a simulator. In our experiments, we make the simplifying assumption that the plasma will only ever be in one of eight states. Finding the best action upon seeing one of the eight states (or contexts) bares similarities to the contextual bandit setting bandits [Krause and Ong, 2011, Agrawal and Goyal, 2013, Auer, 2002]; however, literature usually assumes that the context is decided by nature and cannot be set by the algorithm. Intuitively speaking, being able to decide how many queries are made in each context is advantageous because some contexts may be harder to learn the optimal action for or may occur more frequently in nature. Luckily, our tokamak simulator allows for contexts to be set, and since each simulation is expensive, it is crucial that we use an algorithm that prudently picks both actions and contexts to evaluate.

In this paper we introduce an algorithm that recommends a context and action pair to evaluate at every iteration. Our algorithm falls under the umbrella of *Bayesian optimization* (BO) since we introduce a prior over the reward structure of the context-action space and leverage this model for optimization. Unlike many other problems under the BO setting, our algorithm searches for an optimal action for each context rather than a single optimal action. This serves as a contrasting feature from other problems in multi-task BO [Swersky et al., 2013, Toscano-Palmerin and Frazier, 2018, Shah and Ghahramani, 2016], in which a single action that performs optimally across all objectives simultaneously is sought. Furthermore, our algorithm is theoretically grounded and can broadly be applied to "offline" problems in which context can be set explicitly. In the remainder of the paper, we briefly describe the algorithm and apply it to the nuclear fusion problem.

7.2 Algorithm

Generally, one can think of the optimization problem for each context as a task; we are then faced with a multi-task optimization problem. Let $\mathcal{X}$ be the finite collection of tasks and let $\mathcal{A}$ be the compact set of possible actions. For our application, we assume that the set of actions is the same for each task. Let $f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ be the reward function, where $f(x, a)$ is the reward for performing action a in task x. It is assumed that this reward function is always bounded. Let $\hat{h} : \mathcal{X} \to \mathcal{A}$ be our estimated mapping from task to action. Our goal is then to find an $\hat{h}$ which maximizes $\sum_{x\in\mathcal{X}} f\left(x, \hat{h}(x)\right) \omega(x)$, where $\omega(x) \geq 0$ is some weighting on x (e.g. probability of seeing x at evaluation time or the importance of x). At round t of optimization, we pick a task x_t and an action a_t to perform a query (x_t, a_t) and observe a noisy estimate of the function $y_t = f(x_t, a_t) + \epsilon_t$, where $\epsilon_t \sim N(0, \sigma_\epsilon^2)$ and is iid. Let D_t be the sequence of queried tasks, actions, and rewards up to time t, i.e. $D_t = \{(x_1, a_1, y_1,), \dots, (x_t, a_t, y_t)\}$. Additionally, define $\hat{y}_t(x)$ to be the best reward observed for task x up to time t, $\hat{a}_t(x)$ to be the action made to see this corresponding reward, and $\mathcal{A}_t(x)$ to be the set of all actions made for task x up to time t. In this work, we use Gaussian Processes (GP) to model the reward function. When tasks are correlated, one can use a single GP to jointly model tasks and actions; however, for this paper we only consider a fixed finite set of tasks and opt to model each task, x, with an independent GP with mean function μ_x and covariance function σ_x. For more information about GPs, we refer the reader to Rasmussen and Williams [2005].

Our proposed algorithm (shown in Algorithm 13), named Multi-Task Thompson Sampling (MTS), extends the classic Thompson sampling strategy [Thompson, 1933] to the multi-task setting. The algorithm, simply put, acts optimally with respect to samples drawn from the posterior. That is, at every round a GP sample of the reward function is drawn for each task, and these samples are used as if they were the ground truth reward function to identify the task in which the most improvement can be made. After repeating this for T iterations, we return the estimated mapping $\hat{h}$ such that $\hat{h}(x) = \hat{a}_T(x)$ if an evaluation was made for task x; if no evaluations have been made for the task, $\hat{h}(x)$ maps to an $a \in \mathcal{A}$ drawn uniformly at random.

Algorithm 13 Multi-Task Thompson Sampling (MTS)

```
**Input:** capital $T$, initial capital $t_{init}$, mean functions $\{\mu_x\}_{x\in\mathcal{X}}$, kernel functions $\{\sigma_x\}_{x\in\mathcal{X}}$.
Do random search on tasks in round-robin fashion until $t_{init}$ evaluations are expended.
**for** $t = t_{init} + 1$ **to** $T$ **do**
    Draw $\widetilde{f}(x, \cdot) \sim GP(\mu_x, \sigma_x) | D_{t-1} \quad \forall x \in \mathcal{X}$.
    Set $x_t = \underset{x\in\mathcal{X}}{\text{argmax}} \left[\left(\max_{a\in\mathcal{A}} \widetilde{f}(x, a) - \max_{a\in\mathcal{A}_t(x)} \widetilde{f}(x, a)\right) \omega(x)\right]$.
    Set $a_t = \underset{a\in\mathcal{A}}{\text{argmax}} \widetilde{f}(x_t, a)$.
    Observe $y_t = f(x_t, a_t)$.
    Update $D_t = D_{t-1} \cup \{(x_t, a_t, y_t)\}$.
**end for**
**Output:** $\hat{h}$
```

One benefit of this algorithm is that it comes with theoretic guarantees. For the following, define

$a_t^*(x)$ to be the past action played for task x up to time t that yields the largest expected reward. That is,

$$a_t^*(x) := \begin{cases} \underset{a \in \mathcal{A}_t(x)}{\operatorname{argmax}} f(x,a) & \mathcal{A}_t(x) \neq \emptyset \\ \underset{a \in \mathcal{A}}{\operatorname{argmin}} f(x,a) & \text{else} \end{cases}$$

Note that $a_t^*(x)$ has an implicit dependence on f.

Theorem 1. *Define the maximum information gain to be $\gamma_T := \max_{D_T} I(D_T; f)$, where $I(\cdot;\cdot)$ is the Shannon mutual information. Assume that $\mathcal{X}$ and $\mathcal{A}$ are finite. Then if Algorithm 13 is played for T rounds where $t_{init} = 0$*

$$\mathbb{E}\left[R_{T,f}\right] \leq |\mathcal{X}| \left(\frac{1}{T} + \sqrt{\frac{|\mathcal{X}||\mathcal{A}|\gamma_T}{2T}} \right)$$

where the expectation is with respect to the data sequence collected and f, and where $R_{T,f}$ is defined to be

$$R_{T,f} := \frac{\sum_{x \in \mathcal{X}} \omega(x) \left(\max_{a \in \mathcal{A}} f(x,a) - f(x, a_T^*(x))\right)}{\sum_{x \in \mathcal{X}} \omega(x) \left(\max_{a \in \mathcal{A}} f(x,a) - \min_{a \in \mathcal{A}} f(x,a)\right)}$$

when the denominator is not 0. Otherwise, $R_{T,f}$ takes the value of 0.

The proof relies on ideas from Kandasamy et al. [2019], and the details can be found in Appendix A of Char et al. [2019]. The result gives a bound on the normalized simple regret summed across tasks where the the $\sqrt{|\mathcal{X}||\mathcal{A}|}$ factor in the theorem accounts for the number of actions that can be taken at every step, and the $\sqrt{\gamma_T}$ factor characterizes the complexity of the prior over the tasks. Intuitively, this result shows that there is no task in which we will have especially bad results, and when $\gamma_T = o(T)$, the normalized simple regret converges to 0 in expectation for every task. Finally, we note that these types of results can usually be generalized to infinite action spaces via known techniques [Russo and Van Roy, 2016, Bubeck et al., 2011]

7.3 Application to Nuclear Fusion

7.3.1 Tokamak Simulator (TRANSP) and Overview of Problem

We use the TRANSP program [Grierson et al., 2018] to simulate fusion reactions on the DIII-D tokamak, a tokamak in San Diego that is operated by General Atomics. TRANSP is a time-dependent transport code used for interpretive analysis and predictive simulations of tokamaks. Access to TRANSP and running TRANSP experiments were possible thanks to our collaborators at Princeton Plasma Physics Lab. TRANSP operates by simulating real-world experiments (referred to as "shots") that were conducted on DIII-D. By running the predictive module of TRANSP, we are able to predict how changes in controls would affect the plasma. When simulating a given shot (a simulation on TRANSP is referred to as a "run"), we can identify variables at each time step that correspond to the state of the plasma. One such variable that we focus on is β_n, which is a ratio of the pressure of the plasma to the magnetic energy density. β_n serves as a proxy for the economic output of the reaction. Besides this quantity, we also consider the *total energy eigenvalues*, which represent the amount of change in energy within and outside the plasma due to certain perturbations. In particular, we focus on the minimum value of the total energy eigenvalues, which we will refer to as $\Delta\omega$. $\Delta\omega$ serves as a proxy for the stability of the plasma. When conducting a simulation, we apply controls that specify parameters of the neutral beams, which include power, energy, full energy fraction and half energy fraction. The DIII-D tokamak has a total of 8 neutral beams, 6 of which are co-current beams (inject in the same direction as the plasma current) and 2 of which are counter-current beams (inject in the opposite direction of the plasma current). In our experiments, we confine the action space to 2 dimensions: power coefficient of co-current beams and counter-current beams, each with domain [0.001, 1.0]. These power coefficients are applied by multiplying the maximum power of the set of beams by the coefficient. By ranging the power coefficient from 0.001 to 1.0, we essentially scale the beam powers from the minimum to the maximum power level possible.

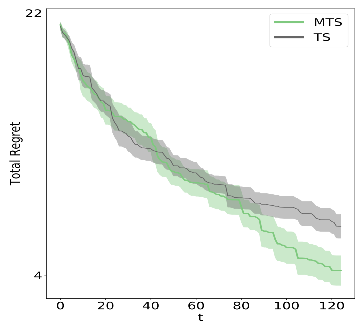
In our experiments, we consider 8 distinct states of the plasma, which are represented by 8 shots. In all 8 shots, a common instability called *tearing* occurred. Ideally, we would like to perform preventative measures once we sense a tear is about to occur. Therefore, we start the simulation 150 ms before time of tearing and run the simulator until 150 ms after the tearing. After the run completes, we extract the $\Delta\omega$ and β_n values at 5ms increments throughout the duration of the run (total 300 ms) and average them to produce $\overline{\Delta\omega}$ and $\overline{\beta}_n$. In order to balance between stability and the pressure in the tokamak, we set our reward to be $10\overline{\beta}_n + 100\overline{\Delta\omega}$, where we chose coefficients based on the scales of each value to make the two objective components roughly equal. In summary, we optimize a combination of pressure and stability of the plasma, for each of the 8 different plasma states (8 tasks or contexts) simultaneously, by changing the power level of the co-current and counter-current beams (2D controls).

7.3.2 Tokamak Control Optimization

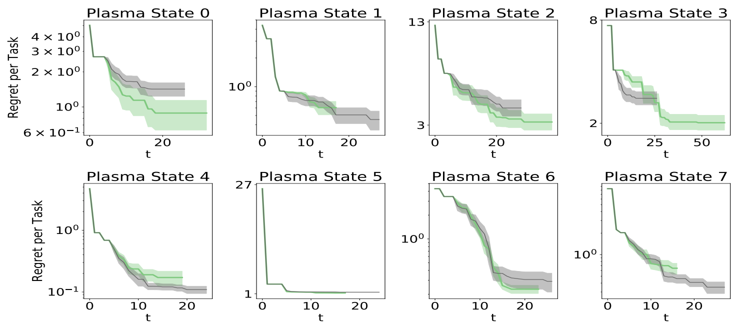
We optimize the reward produced from TRANSP simulations using both MTS and a method that chooses shots (or plasma states) uniformly at random then performs the standard Thompson sampling procedure. The optimization experiment results presented in Figure 7.1 are averaged over 10 trials, each with 125 query capital. In each trial and for each shot, 5 initial points are drawn uniformly at random for evaluation to form an initial GP. Each task is modeled by an independent GP with an RBF kernel and hyperparameters are tuned for each GP every time an observation is seen for its corresponding shot by maximizing the marginal likelihood. We parallelized the experiments by having a batch of 20 workers, each making evaluations according to the respective algorithms using a shared pool of collected data. This process is suboptimal since workers operate asynchronously (i.e. they do not wait to see the data other workers will collect); however, Kandasamy et al. [2018] showed that this approach is not unreasonable for the standard Thompson sampling setting. The results demonstrate how MTS is able to achieve better performance by focusing its resources intelligently. Looking at Figure 7.1 (b), in contexts where reward (and hence regret) levels off quickly (e.g. plasma states 4 and 5), MTS is able to recognize that resources should be allocated in other contexts where higher improvement is expected. This is reflected with more queries and better optimization in plasma states 2 and 3.

7.3.3 Discussion of Results and Future Directions

Figure 7.2 shows the total reward and reward component surfaces for one of the 8 shots from the experiments conducted in Section 7.3.2. From the total reward surface (Figure 7.2 (a)), we can see that beam power should be scaled down for maximum total reward. However, the surfaces of each reward component indicate the negative correlation between plasma stability (Figure 7.2 (b)) and pressure (Figure 7.2 (c)): as beam power is scaled up, stability increases but pressure decreases.

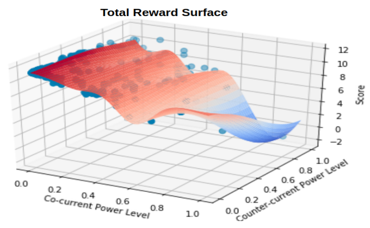


(a)

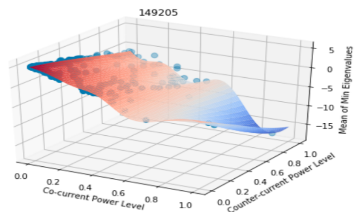


(b)

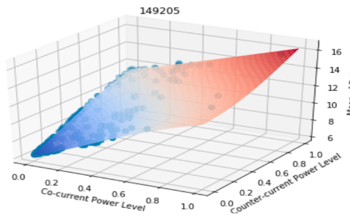
Figure 7.1: **Fusion Simulation Experiments.** Each of the above show average values and standard error from 10 trials. **(a)** shows the log total regret summed across all states and **(b)** log regret achieved in each state. Note that curves differ in length for (b) since different amounts of resources were allocated for each state. Note that regret was approximated by treating the optimum as the maximum value observed for each state, plus a small ϵ term.



(a)



(b)



(c)

Figure 7.2: **Reward and Reward Component Surfaces.** The surfaces have been estimated by fitting a GP to the queried points from shot 149205, which corresponds to plasma state 1. Each point in the space shows a value returned by the simulator.

Hence, there is a fine balance in optimizing a combination of these components, which is dependent on the weighting between the two objectives. This raises further questions about exactly what weighting would be optimal for actual plasma behavior over a longer period of time. In addition, these results provide interesting insights to the fusion community. Although there has been some previous work applying machine learning to fusion in the past [Cannas et al., 2013, Tang et al., 2016, Montes et al., 2019, Kates-Harbeck et al., 2019, Baltz et al., 2017], to the best of our knowledge this is the first work towards learning a tokamak controller offline with no human intervention.

In the future, we hope to discover more interesting results by increasing our action space and forming different reward functions. We are also working to expand this work to finding a policy over a continuous set of plasma states, rather than just a subset of eight. From an algorithmic standpoint, this problem has been explored by Ginsbourger et al. [2014] and Pearce and Branke [2018]. We have preliminary evidence showing that a variation of our algorithm performs competitively with theirs. Lastly, readers may remark that the problem of tokamak control is actually a reinforcement learning problem, since we should be searching for an optimal policy that makes a sequence of actions rather than a single action. Because the simulator is expensive (it takes approximately 2 hours to simulate one control evaluation lasting 300 ms), we chose to limit the scope of the problem in this work; however, we wish to revisit this in the future.

8 | Correlated Trajectory Uncertainty for Adaptive Sequential Decision Making

One of the great challenges with decision making tasks on real world systems is the fact that data is sparse and acquiring additional data is expensive. In these cases, it is often crucial to make a model of the environment to assist in making decisions. At the same time, limited data means that learned models are erroneous, making it just as important to equip the model with good predictive uncertainties. In the context of learning sequential decision making policies, these uncertainties can prove useful for informing which data to collect for the greatest improvement in policy performance [Mehta et al., 2021b, 2022] or informing the policy about unsure regions of state and action space to avoid during test time [Yu et al., 2020]. Additionally, assuming that realistic samples of the environment can be drawn, an adaptable policy can be trained that attempts to make optimal decisions for any given possible instance of the environment [Ghosh et al., 2022, Chen et al., 2021]. In this work, we examine the so-called "probabilistic neural network" (PNN) model that is ubiquitous in model-based reinforcement learning (MBRL) works. We argue that while PNN models may have good marginal uncertainties, they form a distribution of non-smooth transition functions. Not only are these samples unrealistic and may hamper adaptability, but we also assert that this leads to poor uncertainty estimates when predicting multiple step trajectory estimates. To address this issue, we propose a simple sampling method that can be implemented on top of pre-existing models.We evaluate our sampling technique on a number of environments, including a realistic nuclear fusion task, and find that, not only do smooth transition function samples produce more calibrated uncertainties, but they also lead to better downstream performance for an adaptive policy.

8.1 Introduction

One of the great challenges with decision making tasks on real world systems is the fact that data is sparse and acquiring additional data is expensive. In these cases, it is often crucial to make a model of the environment to assist in making decisions. At the same time, limited data means that learned models are erroneous, making it just as important to equip the model with good predictive uncertainties. In the context of learning sequential decision making policies, these uncertainties can prove useful for informing which data to collect for the greatest improvement in policy performance [Mehta et al., 2021b, 2022] or helping the policy identify and avoid regions of state and action space that are uncertain during test time [Yu et al., 2020]. Additionally, assuming that realistic samples of the environment can be drawn, an adaptable policy can be trained that attempts to make optimal decisions for any given possible instance of the environment [Ghosh et al., 2022, Chen et al., 2021].

In this work, we examine the so-called "probabilistic neural network" (PNN) model that is ubiquitous for learning a transition function in model-based reinforcement learning (MBRL) works. We argue that while PNN models may have good marginal uncertainties, they form a distribution of non-smooth transition functions. Not only are these function samples unrealistic and may hamper adaptability, but we also assert that this leads to poor uncertainty estimates when predicting multiple step trajectories. To address this, we propose a simple sampling method that can be implemented on top of pre-existing models. We evaluate our sampling technique on a number of control environments, including a realistic nuclear fusion task. Not only do smooth transition function samples produce more calibrated uncertainties, but they also lead to better downstream performance for an adaptive policy.

8.2 Method

Preliminaries. In this work, we focus on finding optimal policies for infinite-horizon Markov Decision Processes (MDPs). We define the following MDP, $\mathcal{M} := (\mathcal{S}, \mathcal{A}, r, T, T_0, \gamma)$. $\mathcal{S}$ is the set of states; $\mathcal{A}$ is the set of actions that can be played at any state; $r : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ is the reward function over current state, action, and next state; $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function; $T_0 \subset \Delta(\mathcal{S})$ is the initial state distribution; and γ is the discount factor. $\Delta(\mathcal{S})$ and $\Delta(\mathcal{A})$ denotes the class of probability distributions over the state and action space, respectively, and we assume that $\mathcal{S} \subset \mathbb{R}^D$. Also, in this work we constrain the transition function to be deterministic. Our goal is to learn a policy function, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the objective $J(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t, s_{t+1})\right]$, where $a_t \sim \pi(s_t)$, $s_{t+1} = T(s_t, a_t)$, and the expectation is over randomness in initial state and policy actions. We focus on deep reinforcement learning algorithms in which π is a neural network.

One difficulty that arises with deep reinforcement learning methods is the large number of samples needed to learn a good policy. Model-based reinforcement learning (MBRL) methods alleviates this by also learning a model of the environment, $\hat{T}$. One can then reduce the number of samples needed from the true MDP by supplementing the data with fictitious samples generated using $\hat{T}$ and π. For notational convenience, let $\mathcal{X} := \mathcal{S} \times \mathcal{A}$ be the space of concatenated state-action pairs. A trajectory in the MDP of length H can then be written as $(x_1, x_2, \ldots, x_H)$, where $x_t := (s_t, a_t) \in \mathcal{X}$, $s_t = T(x_{t-1})$, $a_t \sim \pi(s_t)$, and $s_1 \sim T_0$. Instead of learning the transition to the next state $x_t \to s_{t+1}$, in practice one usually learns the state delta: $x_t \to s_{t+1} - s_t$, such that the learned model $f_\theta : \mathcal{X} \to \Delta(\mathbb{R}^D)$ can predict the next state as $\hat{T}(x_t) = s_t + f_\theta(x_t)$, where θ is the parameters of the model. One can then use the learned model to "rollout" a fictitious trajectory $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_H)$, where $\hat{x}_t = (\hat{s}_t, \hat{a}_t)$, $\hat{s}_t \sim \hat{T}(\hat{x}_{t-1})$, $\hat{a}_t \sim \pi(\hat{s}_t)$, and $\hat{s}_1 \sim T_0$. As a last piece of notational convenience, we use superscripts to denote particular dimensions of vectors. For example, $s_t^{(d)}$ is the d^{th} dimension of s_t and $f_\theta^{(d)}(x)$ is the d^{th} dimension of the model's output.

8.2.1 The "Probabilistic Neural Network"

One of the first works to emphasize the importance of uncertainty in MBRL was Chua et al. [2018]. To model the transition function, this work proposes using an ensemble of so-called "probabilistic neural networks" (PNN), which are models that output predictive distributions instead of point predictions. In particular, they propose learning a PNN that outputs a mean vector and diagonal

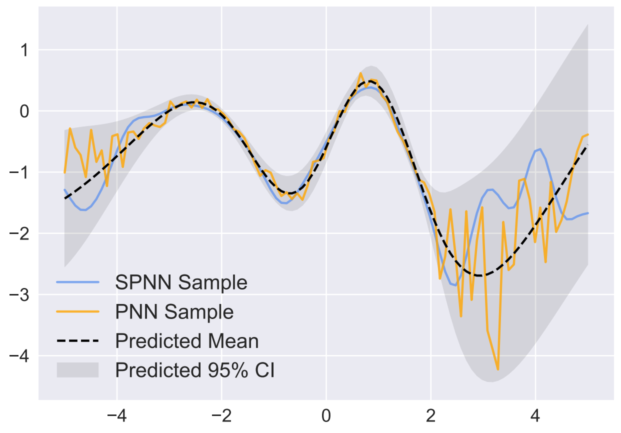


Figure 8.1: **A visual example of function samples**. The vanilla PNN sample can be seen in orange and a sample from our proposed SPNN method can be seen in blue. Crucially both samples stem from the same mean and standard deviation predictions.

covariance matrix which parameterize a multivariate Gaussian distribution. In the context of our work, when f_θ is such a PNN, we write $f_\theta(x) = (\mu_\theta(x), \sigma_\theta(x))$, where $\mu_\theta : \mathcal{X} \to \mathbb{R}^D$ and $\sigma_\theta : \mathcal{X} \to \mathbb{R}^D$ are the mean and standard deviation functions respectively. We will always assume this form of predictive distribution when referring to a PNN. Since this work, PNN's have been used in many MBRL works, including works from both the online [Janner et al., 2019] and offline [Yu et al., 2020, Chen et al., 2021, Yu et al., 2021] settings.

In their paper, Chua et al. [2018] motivate the ensemble of PNNs by claiming that the predicted variance of a PNN, σ_θ, captures aleatoric uncertainty (i.e. the randomness inherent in the true dynamics of the MDP) and ensembling accounts for epistemic uncertainty (i.e. estimation uncertainty stemming from a finite number of datapoints from the system). However, we claim that in reality this breakdown is not so clear. For one thing, the main experiments presented in Chua et al. [2018] as well as subsequent works that leverage this architecture [Janner et al., 2019, Yu et al., 2020, Chen et al., 2021] test their methods in environments with no aleatoric uncertainty. Despite this absence of aleatoric uncertainty, σ_θ seems to play an important role and is even used for penalization in Yu et al. [2020] and Chen et al. [2021] to indicate that the policy is leaving the support of the data. It seems that PNNs can play an important role in helping capture epistemic uncertainty, and we demonstrate this empirically in Appendix 8.B with a toy example.

That being said, a PNN can only capture *marginal* uncertainty (i.e. the uncertainty for any single input $x \in \mathcal{X}$), it has no notion of *joint* uncertainty over the input space $\mathcal{X}$.

As a result, it is unable to draw smooth samples of the transition function (see Figure 8.1). Why does this matter? Consider predicting a full trajectory and assume that the true dynamics, T, and the predicted mean function, μ_θ, are Lipschitz smooth. Then, if consecutive inputs x_{t-1} and x_t are close (i.e. $\|x_{t-1} - x_t\|$ is small), we expect the residuals $T(x_{t-1}) - \mu_\theta(x_{t-1})$ and $T(x_t) - \mu_\theta(x_t)$ to also be close. In other words, we often expect there to be temporal correlations in the residuals that stem from the smoothness in the dynamics and policy (we empirically show these correlations exist in Appendix 8.H). This may be problematic for two reasons: first, these correlations in residuals can lead to miscalibrated uncertainty predictions (see Appendix 8.G) and over-confident behavior in downstream policy learning. Second, assuming that an adaptive policy is being trained, these temporal correlations may be key in distinguishing and adapting to different environments.

Learning Residual Correlation An ideal model should therefore model uncertainty jointly over $\mathcal{X}$ space, and as a result, be able to sample smooth transition functions that can be used to generate trajectory predictions. For the following, we fix an output dimension d and a time step in the dynamics rollout t. To motivate our method, we first note that the predictive distribution of a PNN can be rewritten as $\mu_\theta^{(d)}(x_t) + \sigma_\theta^{(d)}(x_t)Z_t$ where all of the stochasticity comes from $Z_t \sim \mathcal{N}(0, 1)$. In the vanilla PNN, $Z_1, \ldots, Z_t$ are i.i.d random variables; however, one can instead learn a joint distribution over these random variables. Intuitively, this joint distribution should be such that, when $\|x_i - x_j\|$ is small, the correlation between Z_i and Z_j is high. To achieve this, for each dimenions d, we leverage a kernel function $\kappa_d : \mathcal{X} \times \mathcal{X} \to [0, 1]$. This function indicates the similarity between two points, and we assume that $\kappa_d(x, x) = 1$ in our work. Then, $Z_1, \ldots, Z_t$ are distributed as a multivariate Gaussian distribution with mean 0 and covariance matrix Σ, where $\Sigma_{i,j} = \kappa_d(x_i, x_j)$. The parameters of the kernel function can then be tuned by optimizing the likelihood of the data via a gradient-based optimizer. Because this technique results in smooth transition function samples, we refer to it as the Smooth Probabilistic Neural Network (SPNN).

There are several important details to note about this method. First, because of our assumption on the kernel function, the marginal distribution remains the same as the vanilla PNN; only the smoothness of the function samples will change. Second, since every collection of $Z_1, \ldots, Z_t$ is a multivariate Gaussian random variable, this implies that we are modelling the function of standardized residuals (i.e. $\frac{T(x)-\mu_\theta(x)}{\sigma_\theta(x)}$) as a Gaussian Process (GP) parameterized by a constant mean function of 0 and kernel function κ_d. Although GPs are often used as a prior over functions, our method is purely frequentist as the GP is fit to the residuals in the dataset by optimizing kernel parameters via maximum likelihood, and a posterior is never computed. That being said, we can take advantage of GP computational efficiencies in our method. As discussed in Rahimi and Recht [2007] and Wilson et al. [2020], function samples from a GP can be approximated as Fourier series. Following this, for every trajectory, we can sample a function $g : \mathcal{X} \to \mathbb{R}^D$ where $g^{(d)}(x) = \sqrt{\frac{2}{B}} \sum_{b=1}^{B} \cos(\phi_{b,d}^T x + \tau_{b,d})$. Here, B is the number of bases in the approximation, $\tau_{b,d} \sim \mathcal{U}(0, 2\pi)$, and $\phi_{b,d} \sim p_{\kappa_d}$ where p_{κ_d} is the spectral density corresponding to kernel κ_d. One can then simply use $g^{(d)}(x_t)$ in place of Z_t. See Appendix 8.C for more details.

8.3 Experiments

We empirically evaluate the impact of drawing smooth transition function samples. We measure these effects in two ways: first, by assessing how smooth samples affect modeling metrics such as likelihood and calibration under a test set, and secondly, by assessing how smooth samples affect the downstream training of policies.

Environments. We test our method on a number of different environments where there exists some safety critical limit that the agent must avoid. We give brief descriptions of each environment here, but more details can be found in Appendix 8.D. **(Fusion)** We first consider controlling a tokamak for nuclear fusion, an application that has gained interest in the RL community [Degrave et al., 2022, Char et al., 2023b, Seo et al., 2021, 2022]. In our environment, adapted from Char and Schneider [2023], the goal is to push β_N (the normalized ratio between plasma and magnetic

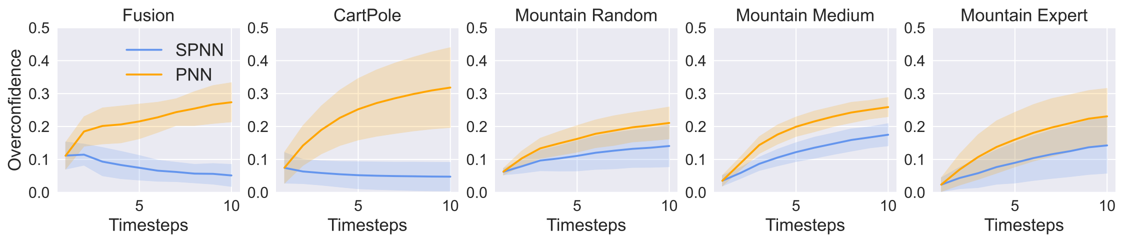


Figure 8.2: **Overconfidence vs. Timestep**. The plots above show the computed overconfidence by timestep in the rollouts when a single PNN or SPNN is used. The solid line shows the mean overconfidence computed over five random seeds, and the shaded region shows the standard error. We use the in-distribution dataset to compute the metrics for the plots shown here.

pressure) to be as close to a fixed limit as possible by adjusting the amount of power injected; however, if the limit is exceeded, the episode ends and the agent is given a large negative reward. The static dataset is comprised of five trajectories in which a PID controller was used to hit a static β_N target well below the limit. **(Mountain)** Next, we consider an environment where an agent is tasked with traversing a mountain ridge in a two-dimensional space. Not only can the agent fall off either side of the ridge, but there is also a cliff at the end of the ridge that the agent must get as close as possible to without falling over. To move, the agent has a thruster which can be angled to accelerate the agent in different directions. To form the dataset for this environment, we train an agent on the true environment and collect data at different stages of performance. In total, we consider three variants of datasets which we name Random, Medium, and Expert. Note that Medium is a supser set of Random, and Expert is a super set of Medium. **(Cart Pole)** Finally, we consider the standard environment in which an agent controls a cart with a pole attached and is tasked with balancing the pole while ensuring the cart does not go out of bounds. For this environment, we collect 250 data points with a poor performing policy.

Training For each of the environments listed above, we train an ensemble of five PNNs. Although forming an ensemble by itself does not give the ability to sample smooth transition functions, it can introduce correlation in residuals assuming one chooses and fixes an ensemble member for each trajectory prediction. We found that for all environments (except the mountain ridge environment), each trained PNN was miscalibrated. Thus, we use the Uncertainty Toolbox [Chung et al., 2021a] to recalibrate each member by finding a constant scaling for the standard deviation of each output dimension. To learn a policy, we use the Soft Actor Critic (SAC) [Haarnoja et al., 2018] algorithm. Following Ni et al. [2021], Chen et al. [2021], we use a recurrent network for the policy to enable the policy to adapt to different instances of the dynamics. See Appendix 8.E for more details.

Modeling Metric Results We first use uncertainty metrics to evaluate the quality of multi-step trajectory samples generated using our sampling technique. Specifically, we measure the *average calibration of centered prediction intervals at each timestep*. Average calibration (also referred to as probabilistic or quantile calibration) [Gneiting et al., 2007, Kuleshov et al., 2018, Song et al., 2019, Chung et al., 2021b] is a standard metric in uncertainty quantification which measures the average discrepancy between the expected and observed proportion of data covered by a predictive interval. We also report the component of miscalibration (i.e. error in average calibration) stemming from overconfidence - where prediction intervals are too narrow and the observed proportion within the interval is thus less than the expected proportion. We call this metric the *overconfidence*, and we

	Method	Fusion	Cart Pole	Mountain Random	Mountain Medium	Mountain Expert
ID	PNN	0.22 ± 0.01 (0.22)	0.27 ± 0.01 (0.24)	0.16 ± 0.01 (0.16)	0.19 ± 0.01 (0.18)	0.16 ± 0.02 (0.15)
	SPNN	**0.10** ± **0.01** (0.08)	**0.14** ± **0.02** (0.05)	0.11 ± 0.01 (0.11)	0.12 ± 0.01 (0.12)	0.12 ± 0.01 (0.09)
	PNN Ensemble	0.13 ± 0.01 (0.03)	0.23 ± 0.01 (0.02)	0.06 ± 0.00 (0.04)	0.10 ± 0.01 (0.08)	**0.10** ± **0.01** (0.05)
	SPNN Ensemble	0.17 ± 0.01 (**0.01**)	0.29 ± 0.01 (**0.00**)	**0.05** ± **0.00** (**0.02**)	**0.09** ± **0.00** (**0.05**)	0.11 ± 0.01 (**0.03**)
OOD	PNN	0.35 ± 0.02 (0.35)	0.32 ± 0.01 (0.29)	**0.15** ± **0.02** (0.08)	0.37 ± 0.01 (0.37)	0.23 ± 0.02 (0.23)
	SPNN	0.27 ± 0.02 (0.26)	0.18 ± 0.02 (0.13)	0.17 ± 0.01 (0.05)	0.33 ± 0.01 (0.33)	0.16 ± 0.02 (0.15)
	PNN Ensemble	0.18 ± 0.01 (0.16)	**0.16** ± **0.01** (0.03)	0.18 ± 0.01 (0.06)	0.28 ± 0.00 (0.28)	0.14 ± 0.02 (0.11)
	SPNN Ensemble	**0.17** ± **0.01** (**0.14**)	0.19 ± 0.01 (**0.01**)	0.20 ± 0.01 (**0.04**)	**0.26** ± **0.00** (**0.26**)	**0.12** ± **0.01** (**0.07**)

Table 8.1: **Miscalibrations and Overconfidences.** The table shows the miscalibration averaged over time steps for each method of sampling and environment. In addition we show the standard error over the five seeds and the proportion of the miscalibration due to overconfidence (in parentheses). All of these quantities are rounded to two digits. We bold the lowest mean miscalibration and overconfidence for each of the environments. The top and bottom blocks show the metrics computed over in-distribution (ID) and out-of-distribution (OOD) datasets, respectively.

focus on it since it is often preferable to be underconfident, especially in safety-critical tasks.

To compute these metrics, we start by splitting a test set of data into sub-trajectories of length 10 (a sub-trajectory is a contiguous segment of a full trajectory in the dataset that need not include the start of the trajectory). Following this our evaluation procedure is as follows: first, a number of "replay" samples are drawn for each sub-trajectory (i.e. the model is used to predict the sub-trajectory using the same action sequence), these samples are then used to form centered prediction intervals for each sub-trajectory, and finally miscalibration and overconfidence metrics can be computed for each step of the replays. We compute these metrics for both an in-distribution (ID) dataset, collected using the same policy as the train set, and an out-of-distribution (OOD) dataset, collected using an expert policy. Concrete definitions of metrics and more details on evaluation procedure can be found in Appendix 8.F.

Table 8.1 shows the miscalibrations and overconfidences averaged across time steps for each environment. Especially in the single PNN case, it is clear that adding smoothness dramatically decreases the overconfidence in the PNN. This is apparent in Figure 8.2, where it is clear that without smoothness, overconfidence grows much faster over time. We also observe that ensembling often helps with uncertainty, especially when evaluating on OOD datasets. While adding smoothness to the samples can cause miscalibration due to underconfidence, we see that the least overconfident models are the ones which couple ensembling and smoothness.

Method	Fusion	Cart Pole	Mountain Random	Mountain Medium	Mountain Expert	Average
NN	65.95 ± 1.63	**100** ± **0.00**	63.57 ± 9.80	26.79 ± 0.62	49.73 ± 2.50	61.21
PNN	65.34 ± 12.94	99.98 ± 0.02	64.29 ± 4.01	23.39 ± 1.34	44.34 ± 13.45	59.47
SPNN	**91.46** ± **2.50**	98.78 ± 1.22	**65.93** ± **1.72**	**39.08** ± **8.24**	**84.72** ± **4.73**	**75.99**

Table 8.2: **Normalized Policy Returns.** Each of the reported numbers is averaged over the last 20% of recorded evaluation episodes during training and five random seeds. We also report the standard errors from the five seeds, and we bold the result with the highest mean. All numbers are normalized using the performance of a poor and expert policy (i.e. a normalized score of 100 is the same performance as the expert policy).

Policy Performance We now turn to examining the performance of an adaptive policy trained with and without smooth samples. For this section we include an additional baseline: an ensemble of neural networks outputting a point prediction (we refer to this as NN). Table 8.3 shows the returns achieved by the policy averaged over the last 20% of training steps. For the fusion environment and all configurations of the Mountain Ridge environment, we see that it is beneficial to have smooth samples and that this better sampling prevents the final policy from crossing the β_N limit or falling off a cliff for each respective environment. Interestingly, we see that the medium version of the mountain ridge environment is the most difficult to optimize. We hypothesize this may be because the medium version of the dataset has a greater spread of data so the dynamics are more confident yet do not have enough data to fully model the dynamics. This is therefore a case where it is especially important that there are smooth samples of the dynamics, and we visually show the difference in policy performance in Figure 8.21 in the Appendix. Lastly, while there seems to be little difference between the final scores in Cart Pole, in Figure 8.17, we show that the returns while training are much more reliable when using smooth dynamics samples. We hypothesize this may be due to the smooth dynamics being more controllable and easier to adapt to.

Discussion. To summarize, in this work we highlighted the existence of correlated errors in dynamics models and how sampling smooth trajectory functions is key to capture this phenomenon in uncertainty estimates. Experimentally, we show that with more intelligent sampling we can achieve less overconfident uncertainty predictions and better performing policies. In future works, we hope to extend our results to more complex and higher dimensional environments.

Appendices for Chapter 8

8.A Related Works

Reinforcement Learning The experimental setting considered in this work falls under the so-called "offline" reinforcement learning setting [Levine et al., 2020]. In this setting, a policy must be learned from a fixed dataset of environment interactions and additional environment queries cannot be made. In this setting, Yu et al. [2020] the MOPO algorithm that uses an ensemble of PNNs and relies on penalties to make sure the policy does not steer out of the offline dataset's support. Chen et al. [2021] extend this idea in the algorithm MAPLE, which incorporates an adaptive policy into the learning procedure. Our reinforcement learning experiments are similar to the set up in MAPLE, except we do full episodes from start states (drawn from the start distribution which is assumed to be known). In contrast, MAPLE does short rollouts of 10 steps starting states randomly selected in the offline dataset.

Gaussian Processes in RL Hypothetically, one could use a GP for the dynamics model and draw posterior samples when generating rollouts with the policy, and this has been done for several low dimensional tasks by previous works [Deisenroth and Rasmussen, 2011, Mehta et al., 2021b, 2022] However, the non-parametric nature of GPs makes scaling up to higher dimensional tasks a challenge. Perhaps an even greater problem comes from the fact that these posterior samples are much more computationally expensive than their PNN alternative. Because algorithms such as MBPO, MOPO, and MAPLE require millions if not billions of model samples to train a neural network policy to convergence, GPs are often too big of a computational burden to use.

8.B PNN Toy Example

To help guide intuition on why the PNN is useful even in deterministic environments, consider a toy regression problem in which we wish to model the function $f(x) = \cos(3x)$. Our training dataset consists of 100 (X, Y) data points where X is distributed as an exponential random variable. As such, there will be a high concentration of X data around 0, but the concentration of training data quickly tapers off. We train a PNN on this toy problem and show the results in Figure 8.3. As one would hope, the PNN is confident in regions where data is plentiful, and the model produces wide predictive distributions in regions lacking data. Why does this happen instead of the network producing highly confident predictive distribution where the mean goes through each training point? We hypothesize that this is due to both the capacity of the network and the property of neural networks to produce generally smooth solutions. Similar observations were also made in Seitzer et al. [2022], although they consider the setting in which aleatoric noise truly does exist.

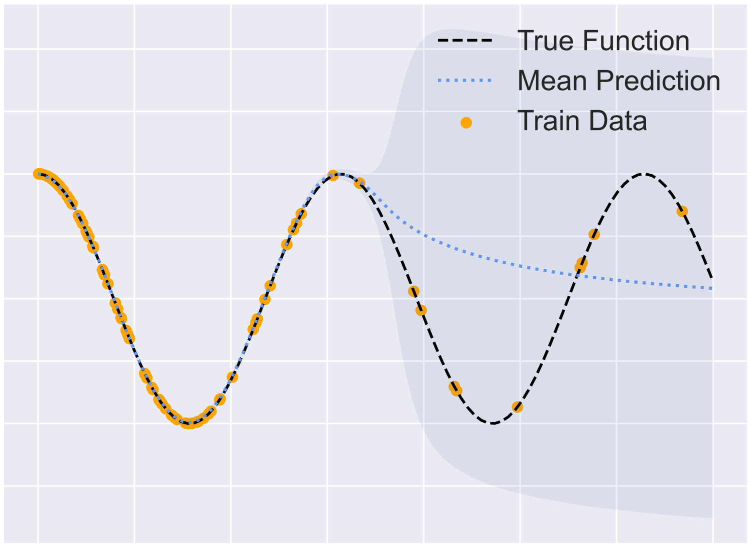


Figure 8.3: **PNN Trained on a Toy Example**. The orange points make up the training dataset, the black dashed line is the true function, and the dotted blue line shows predicted mean. The blue shaded region shows three standard deviations of the predicted Gaussian distribution. We also use a validation set with 20 points to know when to stop training the model.

8.C Algorithm Details

Algorithm 14 SPNN Trajectory Sampling

```
**Input**: Policy $\pi$, initial state $s_1$, kernels $\{\kappa_d\}_{d=1}^{D}$, horizon $H$, and number of bases $B$. Sample function $g$ by sampling $\phi_{b,d} \sim p_{\kappa_d}$ and $\tau_{b,d} \sim \mathcal{U}(0, 2\pi)$ for $b \in \{1, \dots, B\}$ and $d \in \{1, \dots, D\}$.
$t \leftarrow 1, \dots H$ $a_t \sim \pi(s_t)$ $x_t \leftarrow (s_t, a_t)$ $s_{t+1} \leftarrow s_t + \mu_\theta(x_t) + \sigma_\theta(x_t) g(x_t)$ **Return** $(x_1, \dots, x_H)$
```

8.D Environment Details

Nuclear Fusion Environment As stated in the main body of the paper, this environment is adapted from Char and Schneider [2023]. This environment uses equations described in Boyer et al. [2019a] and Scoville et al. [2007]. In particular, we use the following relations for stored energy, E, and rotation, v_{rot}:

$$\dot{E} = P - \frac{E}{\tau_E}$$
$$\tau_E = C_E I^{0.95} B^{0.15} P^{-0.69}$$
$$\beta_N = C_\beta \left(\frac{aB}{I}\right) E$$

where P is the total power, τ_E is the energy confinement time, I is the plasma current, a is the minor radius, B is the magnetic field, and C_E, C_β are constants set to 200 and 5, respectively. The second of these equations is ITERH-98 scaling [Transport et al., 1999]. For our version of the environment, we also fix $I = 10^6$, $a = 0.589$, and $B = 2.75$. Similar to Char and Schneider [2023], we include momentum in the energy update. The equation describing the evolution of the energy is

$$\dot{E}_t = 0.5\left(P_t - \frac{E_t}{\tau_E}\right) + 0.5\dot{E}_{t-1}$$

The observation space for the environment is three dimensional and consists of the current β_N measurement, the rate of change of β_N, and the current amount of power being injected into the system. The action space is one-dimensional and is simply the change in the power. We set the β_N limit to be 2.2 , and the reward function is

$$r(\beta_N, a) := \begin{cases} \left(\frac{2.5-|\beta_N-2.2|}{2.5}\right)^2 - \frac{\|a\|}{10} & \beta_N \leq 2.2 \\ -100 & \beta_N > 2.2 \end{cases}$$

where a is the action scaled to be between -1 and 1. We use a horizon length of 100 for each episode.

Mountain Ridge Environment The mountain ridge environment has five-dimensional observations space: $s_t = (x_t, \dot{x}_t, y_t, \dot{y}_t, \theta)$, where x_t is the x position, y_t is the y position, and θ is the angle of the thruster used to propel the agent. The action space is two-dimensional and consists of a^{thrust} and a^{angle}, which controls the amount of thrust and the change in angle of the thruster, respectively. The updates are as follows:

$$\begin{aligned} x_{t+1} &= x_t + \dot{x}_t \Delta t \\ \dot{x}_{t+1} &= \dot{x}_t + \left(\text{sign}(x_t)x_t^2 + a^{\text{thrust}}\sin(\theta_t)\right)\Delta t \\ y_{t+1} &= y_t + \dot{y}_t\Delta t \\ \dot{y}_{t+1} &= \dot{y}_t + \left(a^{\text{thrust}}\cos(\theta_t) + 0.05\exp(y_t)\right)\Delta_t \\ \theta_{t+1} &= \text{clip}\left(\theta_t + \frac{\pi}{6}f(a^{\text{angle}}), -\pi, \pi\right) \end{aligned}$$

where the function f is defined as

$$f(x) = \begin{cases} 1 + \exp\left[-12.5\,(x - 0.5)\right] & x > 0 \\ 1 + \exp\left[12.5\,(x + 0.5)\right] & x \leq 0 \end{cases}$$

The reward function is simply $\frac{6+y_t-\left\|a^{\text{thrust}}\right\|}{10}$ while the agent is on the cliff. The episode ends and the agent recieves a reward of -100 if $|x_t| > 3$, $y_t < -6$, or $y_t > 5$. We use a horizon length of 200 for each episode.

8.E Additional Training Details

Model Training For each of the dynamics models, we use a network with 2 hidden layers, each with 512 units. We use two separate heads for the mean and standard deviation predictions. We find that we can get better uncertainty by adding an additional hidden layer with 256 units to the standard deviation head. Besides the change in architecture, the learning procedure follows what is done in Chua et al. [2018], and we use the Adam [Kingma and Ba, 2014] optimizer with a learning rate of 3×10^{-4} and a batch size of 64. Lastly, we use 10% of the data as a validation set and early stop based on MSE (although we pick the checkpoint that achieves the best negative log likelihood).

Reinforcement Learning Training Our implementation of SAC with recurrent policies closely follows the implementation given by Ni et al. [2021] and uses a Gated Recurrent Unit (GRU) [Cho et al., 2014]. We give hyperparameter settings in Table 8.3. We train for 100, 250, and 1000 epochs for the Cart Pole, Fusion, and Mountain environments, respectively. Following other offline model-based reinforcement learning works [Yu et al., 2020, Chen et al., 2021], we add a penalty to the reward to indicate when the policy is going out of distribution. When using PNN models, we use the maximum predicted standard deviation among the ensemble members, and, when using neural networks with point predictions, we use the standard deviation among the mean predictions. As per [Chen et al., 2021] we scale this uncertainty by 0.25.

When simulating episodes with the model during training time, the trajectory can sometimes blow up and predict large values. While this is rare, we find that it helps training stability to cap the velocity components of the state space to reasonable values. Finally, for all methods of sampling, we choose a member of the ensemble to make predictions each episode, and we fix this member for the entire episode.

Hyperparameter	**Value**
Discount Factor	0.99
Learning Rate	3×10^{-4}
Batch Size	256
Target Soft Update Weight	5×10^{-3}
History Lookback Size	64
Exploration Steps per Epoch	1000
Gradient Steps per Epoch	1000

Table 8.3: **Reinforcement learning hyperparameters.**

8.F Model Metric Details

A widely accepted metric in uncertainty quantification to evaluate the validity of distributional predictions is *average calibration*. Given input covariates X, target variables Y, a predictive distribution with CDF $F_X : \mathcal{X} \to (\mathcal{Y} \to [0,1])$ and its corresponding quantile function $F_X^{-1} : \mathcal{X} \to ([0,1] \to \mathcal{Y})$, F is said to be average calibrated if

$$P\left(Y \leq F_X^{-1}(p)\right) = p, \forall p \in [0,1]. \tag{8.1}$$

Note that Eq. 8.1 assesses the validity of the predictive quantile function F_X^{-1}, which is identical to a prediction interval between the probabilities $[0,p]$. We note that centered prediction intervals (e.g. a 95% prediction interval that spans the probabilities $[0.025, 0.975]$) can be more useful in practice, and we assess the average of centered prediction intervals, which is defined as:

$$P\left(F_X^{-1}(0.5 - p/2) \leq Y \leq F_X^{-1}(0.5 + p/2)\right) = p, \forall p \in [0,1]. \tag{8.2}$$

Miscalibration, i.e. error in average calibration of centered intervals, is then measured as

$$\int_0^1 \mid P\left(F_X^{-1}(0.5 - p/2) \leq Y \leq F_X^{-1}(0.5 + p/2)\right) - p \mid dp. \tag{8.3}$$

Given a dataset $\{x_i, y_i\}_{i=1}^N$, and a uniform draw of probabilities $\{p_k\}_{k=1}^K \in [0,1]$, miscalibration of centered intervals can be estimated as

$$\frac{1}{K}\sum_{k=1}^{K}\left|\texttt{(empirical coverage at } p_k\texttt{)} - p_k\right|, \tag{8.4}$$

where $\texttt{(empirical coverage at } p_k\texttt{)}$ is defined as $\frac{1}{N}\sum_{i=1}^N \mathbb{I}\{F_{x_i}^{-1}(0.5 - p_k/2) \leq y_i \leq F_{x_i}^{-1}(0.5 + p_k/2)\}$ and $\mathbb{I}$ is the indicator function.

We compute miscalibration of centered intervals at each timestep of a trajectory, where the inputs are the current state-action pairs, and the targets are the state delta: i.e. from Eq. 8.4, x_i would be the tuple $(s_{i,t}, a_{i,t})$ and y_i would be $s_{i,t+1} - s_{i,t}$. We used 19 equi-spaced probabilities: $\{p_k = \frac{k}{20}\}_{k=1}^{19}$. Since an ensemble does not provide a closed form quantile function, we use empirical quantiles for $F_{x_i}^{-1}$ by generating many trajectories for a single test sequence of states and actions.

To measure overconfidence, we performed the outer summation over probabilities in Eq. 8.4 only if the empirical coverage was lower than p_k:

$$\frac{1}{K}\sum_{k=1}^{K}\min\left(0, \texttt{(empirical coverage at } p_k\texttt{)} - p_k\texttt{)}\right), \tag{8.5}$$

8.G Ignoring Error Correlation Can Lead to Overconfidence

In this section, we show that under certain assumptions, ignoring the correlation between consecutive residuals leads to overconfident predictions. While these assumptions make major simplifications to the problem, this result still gives insight into why overconfidence may grow over time. In what follows, assume that there is a fixed action sequence $a_1, \ldots, a_N$. The corresponding rollout using the true transition function, T, is then $x_1 \ldots, x_{N+1}$.

Ideally, we would compare this to the distribution of rollouts created by sampling autoregressively from $\hat{T}$, which we assume to be a PNN. However since this distribution is difficult to characterize, we focus on analyzing one-step errors. Towards this end, let $\delta_t := \mu_\theta(x_t) - T(x_t)$ and let $\Delta_N := \sum_{t=1}^N \delta_t$. Although the true amount of error after N steps is hard to reason about because of the predicted sequence's autoregressive nature, Δ_N can be thought of as a proxy. We also make the following assumptions:

1. The sequence of residuals is Markovian, i.e. $p(\delta_t|\delta_1, \ldots, \delta_{t-1}) = p(\delta_t|\delta_{t-1})$.
2. The distribution between consecutive residuals is

$$\begin{bmatrix}\delta_t \\ \delta_{t-1}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0 \\ 0\end{bmatrix}, \begin{bmatrix}1 & \rho \\ \rho & 1\end{bmatrix}\right)$$

Note that these are assumptions that are made to the true underlying transition function (i.e. that there are correlations between residuals). We will now see that the standard deviation of Δ_N grows faster if positive correlation is present versus if an independence assumption is made (as is usually done with sampling procedure in PNNs).

Proposition 5. *(Deserno [2002]) Following the above assumptions*

$$\Delta_{N+1} = \sqrt{\frac{1+\rho}{1-\rho}}G_{N+1} - \frac{\rho}{1-\rho}\delta_{N+1} + \frac{1-\sqrt{1-\rho^2}}{1-\rho}g_1$$

where $N > 1$, $g_1, \ldots, g_{N+1} \overset{i.i.d}{\sim} \mathcal{N}(0,1)$, *and* $G_{N+1} := \sum_{t=1}^{N+1} g_t$.

Note that the first term of Δ_N has standard deviation $\sqrt{\frac{1+\rho}{1-\rho}N}$, and the standard deviation for the rest of the terms does not grow as N increases. In contrast, if one were to model residuals as independent, the standard deviation of the sample trajectories from that model after N steps would be $\sqrt{N}$. Again, the assumptions made here prevent any statement from being made in the actual setting in which trajectories are autoregressively predicted; however, it does give intuition as to why PNNs may produce overconfidence predictive distributions over time.

We now restate the proof from Deserno [2002] for completeness.

Proof. Using the fact that $p(\delta_t|\delta_{t-1} = d) \sim \mathcal{N}(\rho d, 1-\rho^2)$, we can express the sequence in terms of IID samples as follows:

$$\delta_1 = g_1; \qquad \delta_t = \rho\delta_{t-1} + \sqrt{1-\rho^2}g_t$$

We can expand the right definition of δ_t to get the following,

$$\delta_t = \rho^{t-1}g_1 + \sqrt{1-\rho^2}\sum_{i=2}^{t} g_i\rho^{t-i}$$

Summing this up to get Δ_N,

$$\begin{aligned}
\Delta_{N+1} &= \sum_{t=1}^{N+1}\left[\rho^{t-1}g_1 + \sqrt{1-\rho^2}\sum_{i=2}^{t} g_i\rho^{t-i}\right] \\
&= g_1\frac{1-\rho^{N+1}}{1-\rho} + \sqrt{1-\rho^2}\sum_{i=2}^{N+1} g_i \sum_{n=i}^{N+1}\rho^{n-i} \\
&= g_1\frac{1-\rho^{N+1}}{1-\rho} + \frac{\sqrt{1-\rho^2}}{1-\rho}\left(\sum_{i=2}^{N+1} g_i - \rho\sum_{i=2}^{N+1} g_i\rho^{N+1-i}\right).
\end{aligned}$$

The first term in the bracket is $G_{N+1} - g_1$, and the second term can be rewritten with δ_{N+1}.

$$\Delta_{N+1} = \sqrt{\frac{1+\rho}{1-\rho}}G_{N+1} - \frac{\rho}{1-\rho}x_N + \frac{1-\sqrt{1-\rho^2}}{1-\rho}g_1$$

$\square$

8.H Empirical Correlations

We empirically compute temporal correlation between the residuals for models, specifically ensembles of PNNs trained on all our environments. Consider a rollout in the true dynamics - $(s_0, a_0, r_0), \ldots, (s_n, a_n, r_n)$. For a given ensemble member, let $b_0 = \frac{T(s_0,a_0)-\mu_\theta(s_0,a_0)}{\sigma_\theta(s_0,a_0)}, \ldots, b_n = \frac{T(s_n,a_n)-\mu_\theta(s_n,a_n)}{\sigma_\theta(s_n,a_n)}$ be the corresponding sequence of standardized residuals. We make the assumption that successive residuals b_i, b_{i+1} are sampled from a bivariate gaussian with correlation coefficient ρ, that is $\begin{bmatrix} b_i \\ b_{i+1}\end{bmatrix} \sim \mathcal{N}(\mu, \Sigma)$, where $\mu = \begin{bmatrix} 0 \\ 0\end{bmatrix}$ and $\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1\end{bmatrix}$. Then, we compute the maximum likelihood estimator, $\hat{\rho}$, based on the observed residuals $b_1, \ldots, b_n$. The estimates in Table 8.4 are averaged over five seeds and five ensemble members in each corresponding ensemble.

Method	Fusion	Cart Pole	Mountain Random	Mountain Medium	Mountain Expert
Dimension 0	0.69 ± 0.13	0.96 ± 0.04	0.95 ± 0.02	0.94 ± 0.01	0.93 ± 0.02
Dimension 1	0.70 ± 0.04	0.95 ± 0.05	0.93 ± 0.01	0.97 ± 0.01	0.98 ± 0.01
Dimension 2	0.85 ± 0.08	0.96 ± 0.04	0.62 ± 0.10	0.57 ± 0.07	0.51 ± 0.11
Dimension 3	-	0.93 ± 0.07	0.45 ± 0.03	0.66 ± 0.02	0.68 ± 0.07
Dimension 4	-	-	-0.02 ± 0.03	0.13 ± 0.02	0.13 ± 0.03

Table 8.4: **Empirical Correlations** Each value is the average over five seeeds and five ensemble members. Note entries in the table that are entry are due to the environment being lower dimensional (e.g. Fusion only has three dimensions).

8.I Additional Experimental Results

Additional Calibration Results To better understand how uncertainty estimates change with different sampling procedures, we provide additional plots of the miscalibration and overconfidence metrics. Figures 8.4- 8.11 show how both metrics change with respect to time for single models and an ensemble. Figures 8.12- 8.15 show how the metrics change with respect to ensemble size. In all of these, it is clear that smooth samples mitigate against overconfidence over time; however, this can also cause uncertainty predictions to be slightly underconfident.

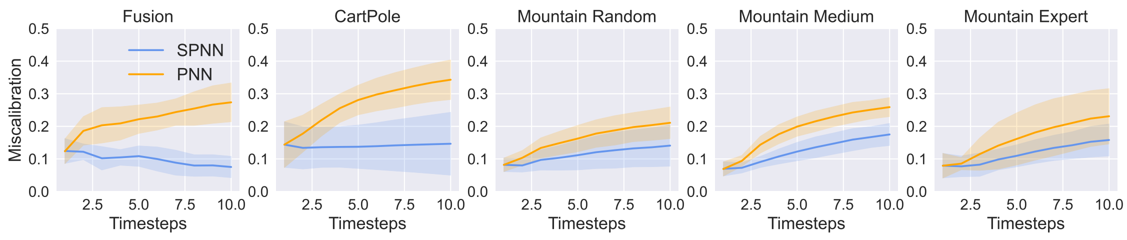


Figure 8.4: **Average miscalibration for ID data using a single PNN with respect to rollout step.** The regions shows the standard error over the five seeds.

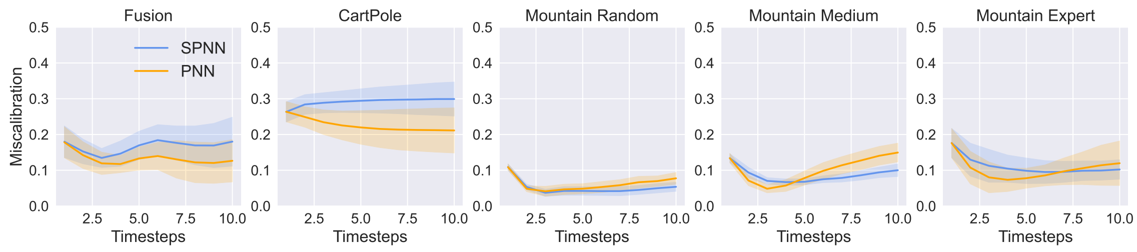


Figure 8.5: **Average miscalibration for ID data using an ensemble of PNNs with respect to rollout step.** The regions shows the standard error over the five seeds.

Reinforcement Learning Training Curves We also provide plots of the average returns during training of the policy in Figures 8.16- 8.20. In general, we see that training with SPNN is less prone to overfitting and often more stable.

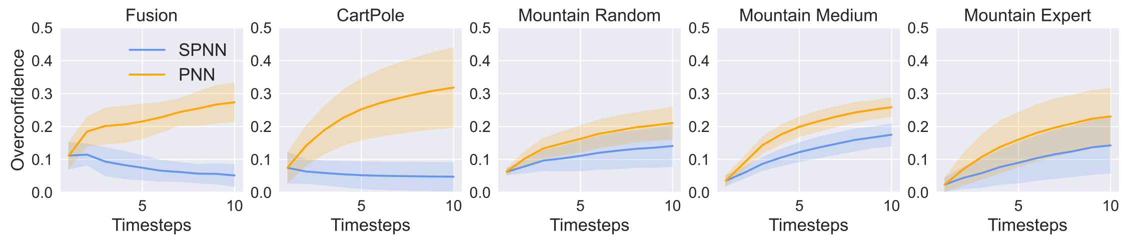


Figure 8.6: **Average overconfidence for ID data using a single PNN with respect to rollout step.** The regions shows the standard error over the five seeds.

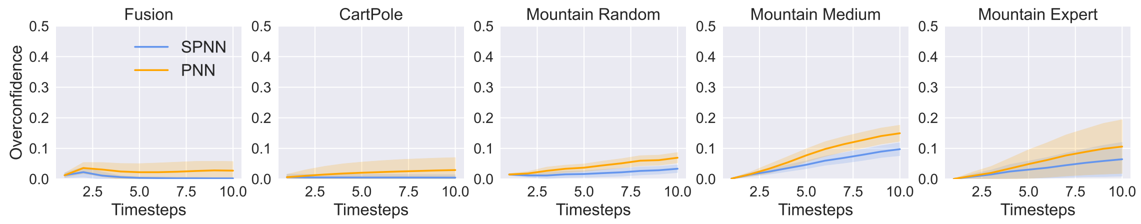


Figure 8.7: **Average overconfidence for ID data using an ensemble of PNNs with respect to rollout step.** The regions shows the standard error over the five seeds.

Can additional penalty help in the Mountain environment? To prevent the agent from falling off the cliff, it is possible that a higher scale on the penalty could be beneficial. In Table 8.5, we test what happens when the penalty scaling is changed to 0.0 or 1.0. We find that greatly increasing the penalty $4\times$ does not have same affect on policy performance as intelligent sampling.

Method	Mountain Random	Mountain Medium	Mountain Expert	Average
SPNN Penalty=0.0	61.41 ± 4.47	29.67 ± 0.87	**88.00 ± 1.60**	59.69
PNN Penalty=0.0	63.64 ± 11.41	23.13 ± 1.15	68.77 ± 11.97	51.84
NN Penalty=0.0	31.26 ± 5.68	27.25 ± 0.67	40.71 ± 5.11	33.07
SPNN Penalty=0.25	65.93 ± 1.72	39.08 ± 8.24	84.72 ± 4.73	**63.24**
PNN Penalty=0.25	64.29 ± 4.01	23.39 ± 1.34	44.34 ± 13.45	44.01
NN Penalty=0.25	63.57 ± 9.80	26.79 ± 0.62	49.73 ± 2.50	46.7
SPNN Penalty=1.0	62.05 ± 1.07	**40.45 ± 8.91**	81.78 ± 3.63	61.42
PNN Penalty=1.0	53.06 ± 7.93	30.57 ± 2.33	46.28 ± 4.63	43.3
NN Penalty=1.0	**67.47 ± 10.56**	25.71 ± 0.07	55.07 ± 12.17	49.42

Table 8.5: **Normalize policy performances for different penalties on the Mountain environment.** Each result is averaged over the last 20% of evaluations during training. Five seeds were used to compute the average scores, and we show the standard errors.

Mountain Environment Visualization To better understand what is happening in the Mountain environment, we plot the average path taken by each type of policy (see Figure 8.21). While all policies are overconfident and have episdoes where the agent falls off the cliff, on average policies trained with SPNN stay within the support of the dataset and avoid falling off the cliff.

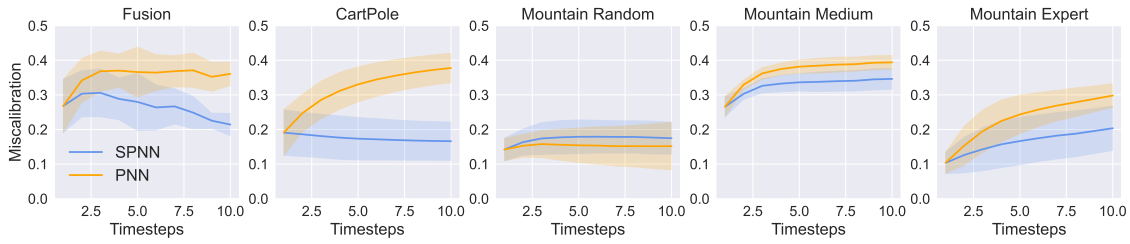


Figure 8.8: **Average miscalibration for OOD data using a single PNN with respect to rollout step.** The regions shows the standard error over the five seeds.

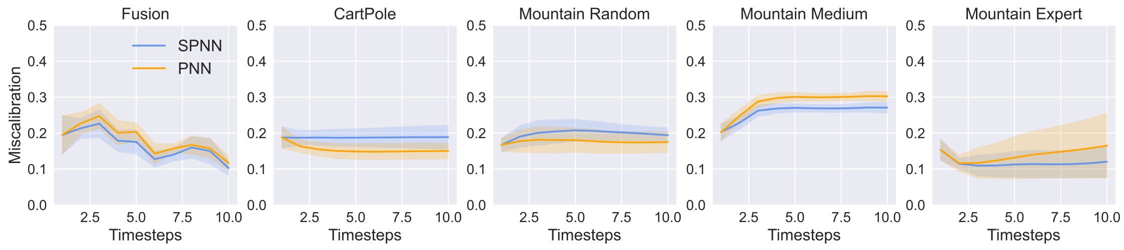


Figure 8.9: **Average miscalibration for OOD data using an ensemble of PNNs with respect to rollout step.** The regions shows the standard error over the five seeds.

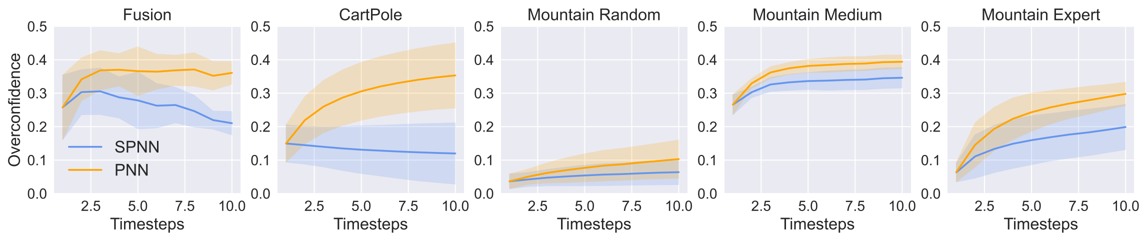


Figure 8.10: **Average overconfidence for OOD data using a single PNN with respect to rollout step.** The regions shows the standard error over the five seeds.

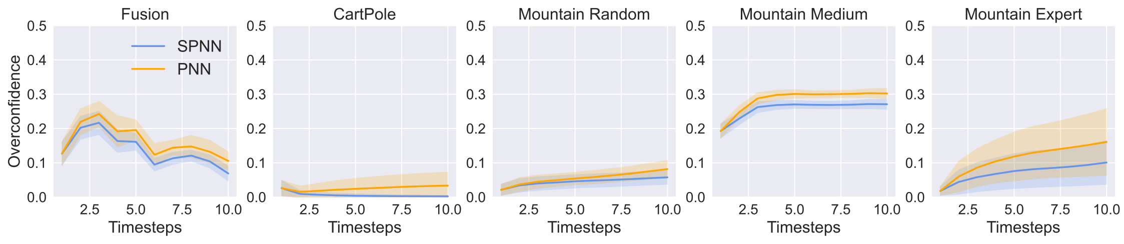


Figure 8.11: **Average overconfidence for OOD data using an ensemble of PNNs with respect to rollout step.** The regions shows the standard error over the five seeds.

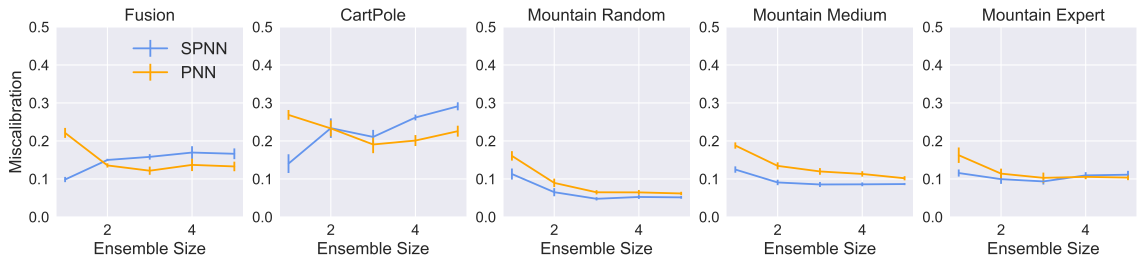


Figure 8.12: **Average miscalibration over rollout for ID data with respect to ensemble size.** The error bars shows the standard error over the five seeds.

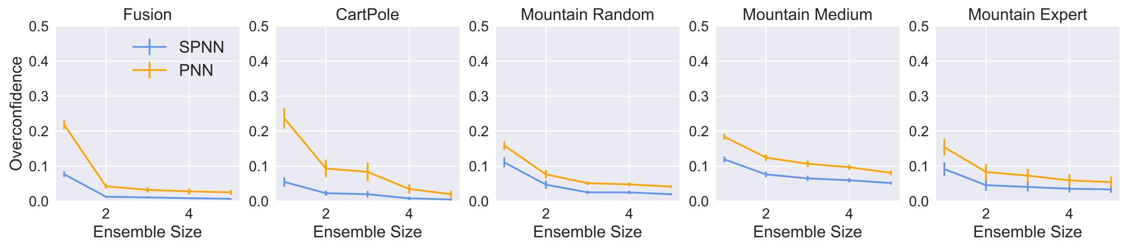


Figure 8.13: **Average overconfidence over rollout for ID data with respect to ensemble size.** The error bars shows the standard error over the five seeds.

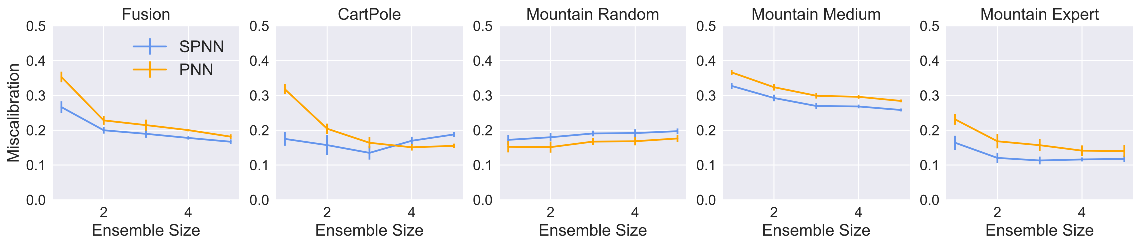


Figure 8.14: **Average miscalibration over rollout for OOD data with respect to ensemble size.** The error bars shows the standard error over the five seeds.

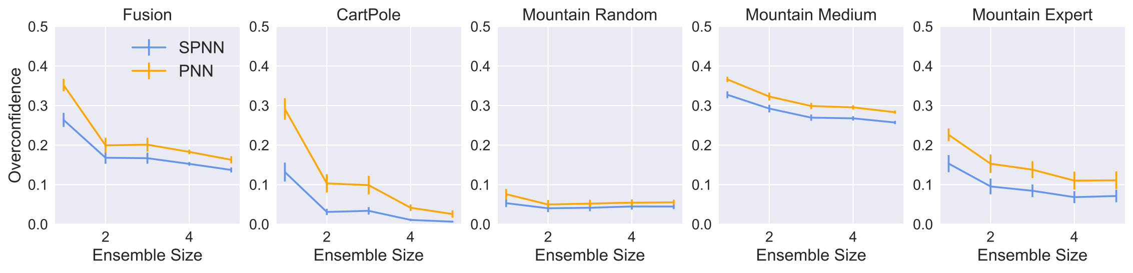


Figure 8.15: **Average overconfidence over rollout for OOD data with respect to ensemble size.** The error bars shows the standard error over the five seeds.

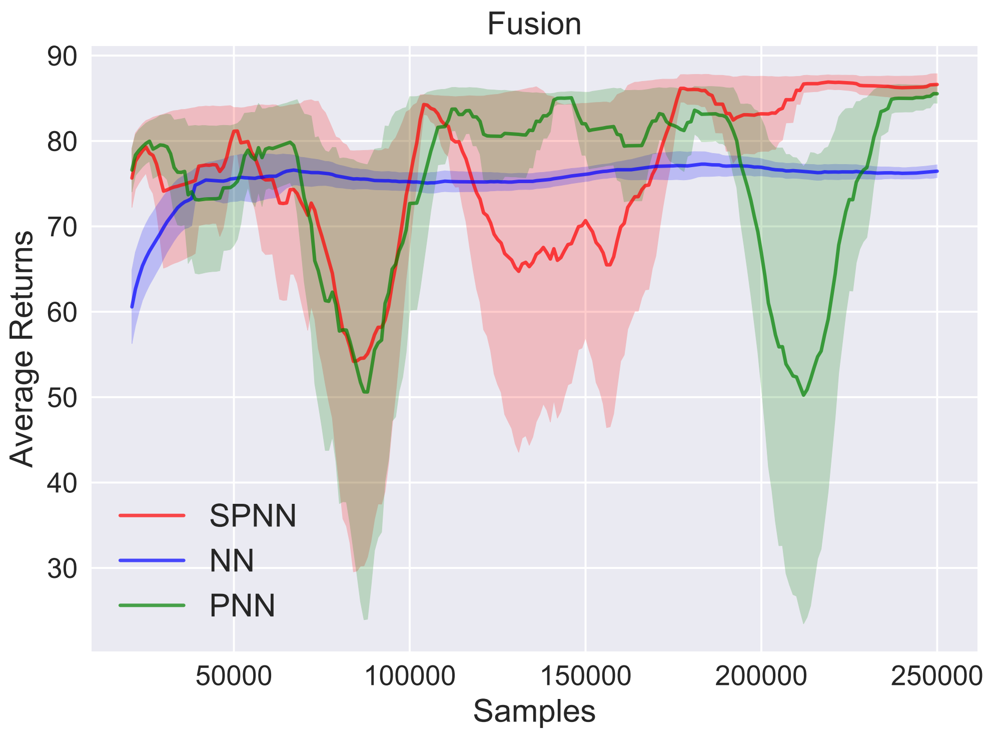


Figure 8.16: **Average returns for the Fusion environment during training.** The regions shows the standard error over the five seeds.

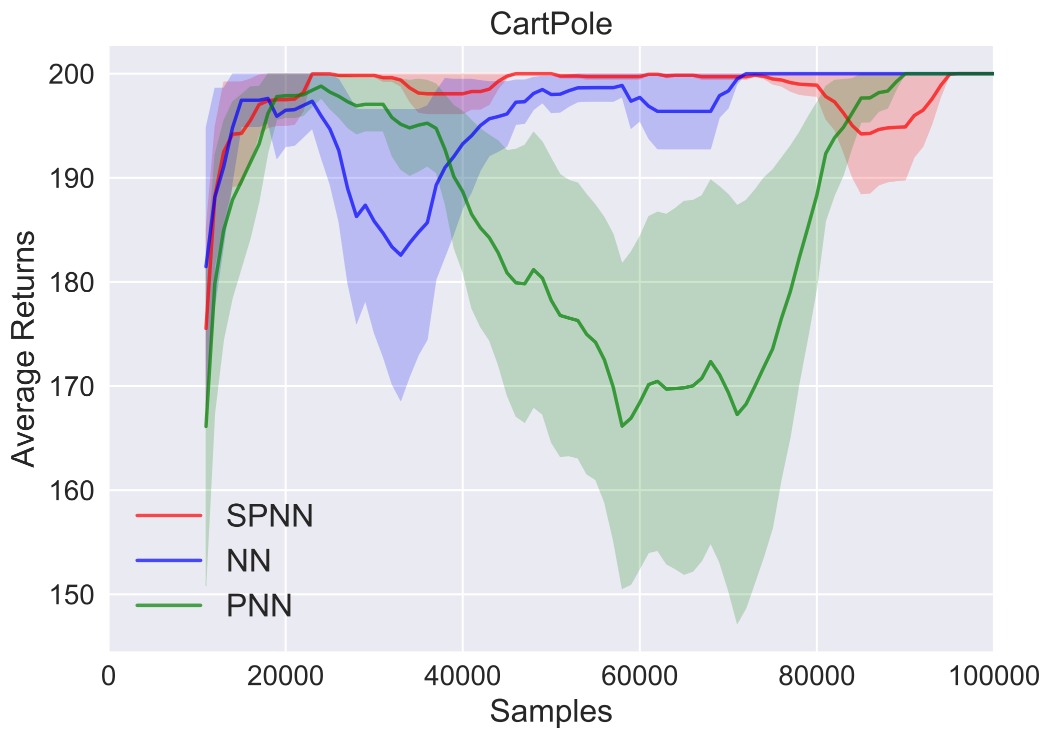


Figure 8.17: **Average returns for the Cart Pole environment during training.** The regions shows the standard error over the five seeds.

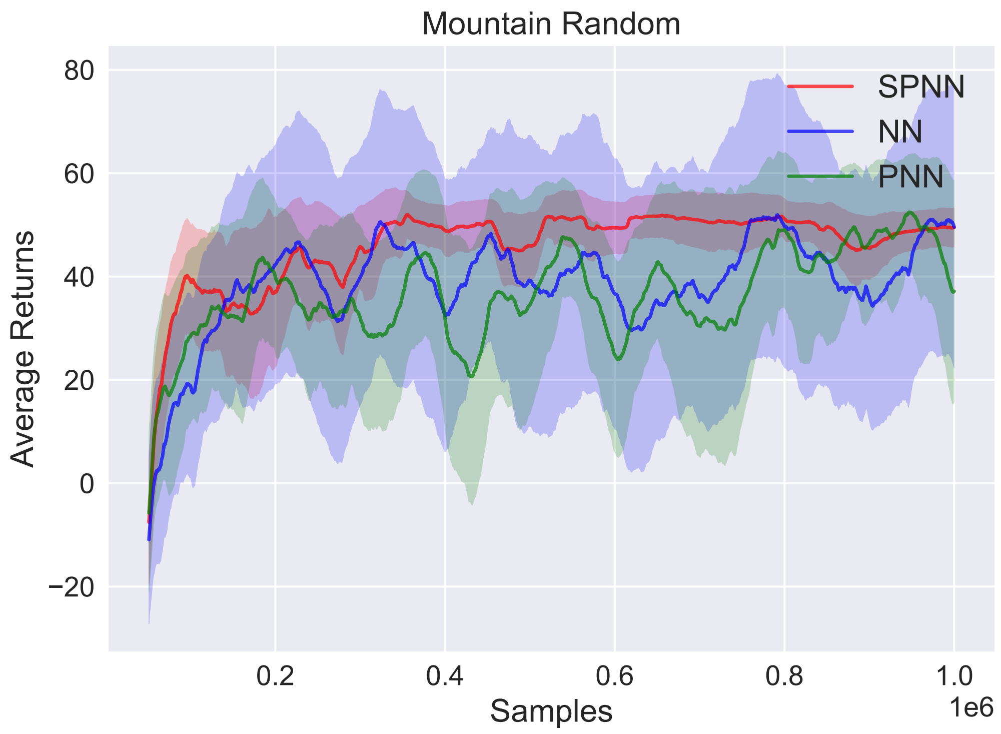


Figure 8.18: **Average returns for the Mountain Random environment during training.** The regions shows the standard error over the five seeds.

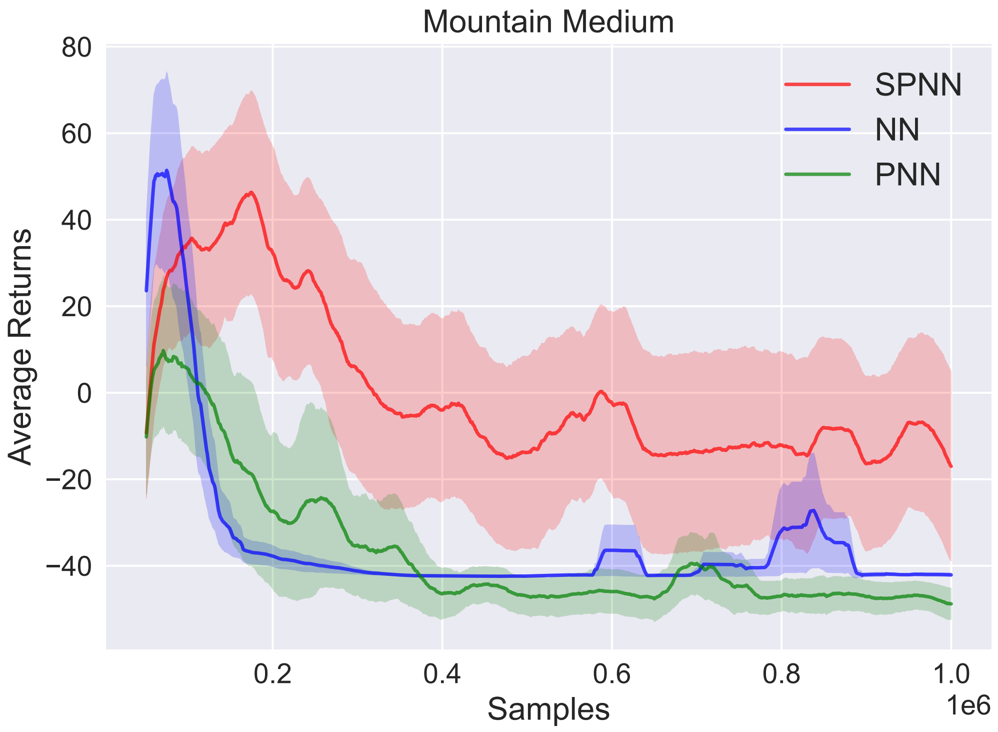


Figure 8.19: **Average returns for the Mountain Medium environment during training.** The regions shows the standard error over the five seeds.

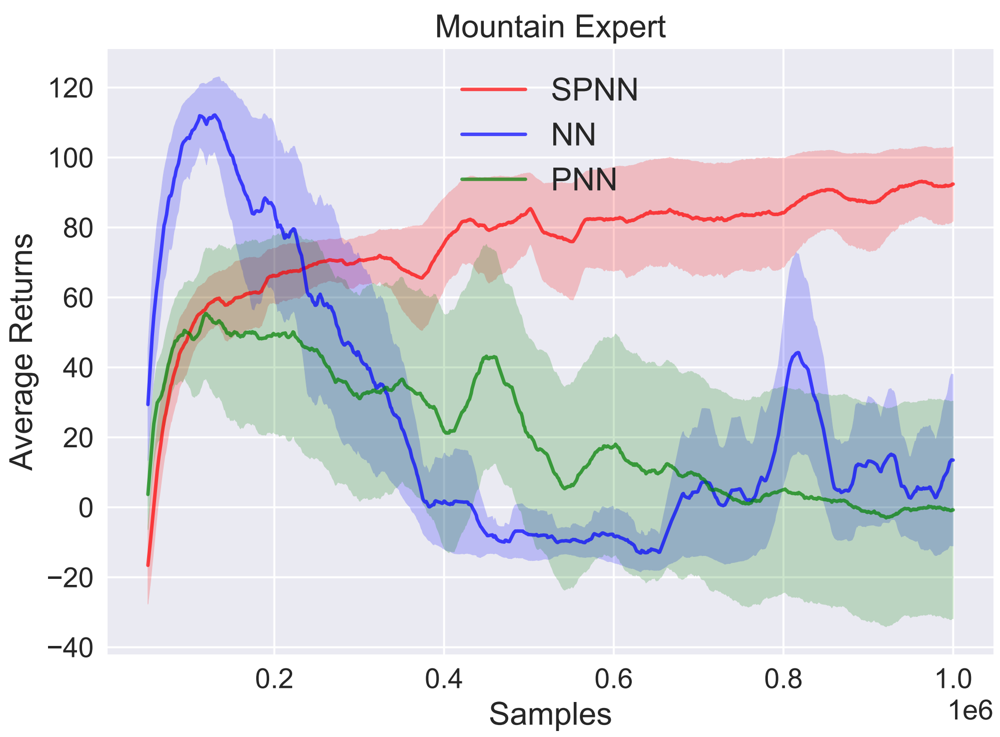


Figure 8.20: **Average returns for the Mountain Expert environment during training.** The regions shows the standard error over the five seeds.

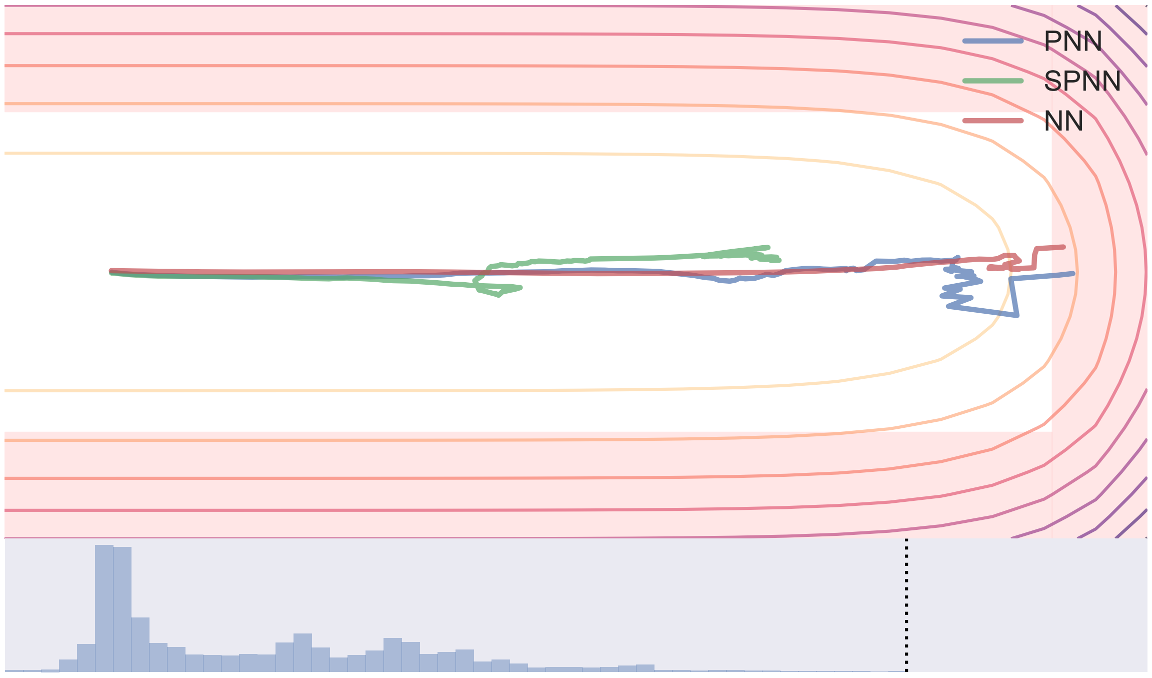


Figure 8.21: **Average trajectory in medium mountain ridge environment.** To create this figure, we collect 100 paths in the true environment with each of the five policies corresponding to the random seeds. We then average each path together, and the average path can be seen in the top plot. The red regions represent terminal regions where the agent falls off the cliff, and the contour lines show the contours of the mountain ridge. Here, the x-axis shows the y position of the agent as described in the environment definition (Appendix 8.D). The bottom plot shows a histogram of the y data in the Medium dataset, and the dashed black line shows the most extreme recorded y value.

Part IV

Conclusion

9 | Conclusion

In theory, uncertainty quantification has many practical applications. Estimates of uncertainty provide the primary mechanism by which exploration is executed for the purposes of data collection (e.g. Bayesian optimization, active learning, exploration in deep reinforcement learning), it serves as a crucial signal in decision-making (e.g. risk assessment, Bayes optimal decision-making, selective prediction, out-of-distribution detection), and sampling techniques (e.g. in planning, sequential sampling and uncertainty propagation, Monte Carlo estimation) rely on signals of uncertainty that are inherent to predictive distributions. This is obviously a non-exhaustive list of applications where uncertainty is crucial and readily used.

Most machine learning models are trained in a probabilistic manner, which gives rise to the expectation that the resulting predictions from the model should in fact behave like probabilities. Thus, calibration is considered a crucial evaluation criterion. Given this understanding, there are many methods which focus on the task of achieving calibration. Part II of this thesis discusses these methods, which can be grouped into pre-hoc and post-hoc methods. As discussed numerous times throughout this thesis, calibration by itself is hardly an unequivocal signal of the usefulness of the probabilistic predictions, and there are a variety of metrics one can additionally consider. Part I discusses the pertinent metrics in uncertainty quantification and practical open-source software which standardizes the usage and implementation of these metrics.

Moving forward, I believe the discussion in uncertainty should be centered around how they can be used and the utility they bring about, rather than the metrics and theoretical correctness of the uncertainty estimates. There are a plethora of calibration metrics. In classification, confidence calibration initiated much of the discussion around uncertainty calibration for deep learning [Guo et al., 2017], and since then, various notions of calibration have been proposed, e.g. classwise calibration, top label calibration, canonical calibration, threshold calibration, decision calibration. This thesis has also contributed the notion of parity calibration for the binary classification setting. Various notions have been proposed for the regression setting as well, including but not limited to quantile calibration (a.k.a. average calibration), group calibration, adversarial group calibration, and distribution calibration. The key question that practitioners and users of these uncertainty-aware models should ask is *why* do you want calibration, and *why* would you want certain notions of calibration? An analogous question can be asked of model providers: Why should you train models that are calibrated? What benefits can you offer to the downstream user with calibrated models?

Thus far, a vast constituent of work in uncertainty quantification has focused on providing probabilistic models that are well-calibrated. In the regression setting, many methods exist which output prediction intervals which are “well-calibrated and sharp”. However, it still remains unclear what the implications are when a downstream application uses these models for whichever task they are concerned with. A salient example is the seemingly conflicting findings by Foldager et al. [2023] an Deshpande et al. [2024]. Both works study the relationship between calibration of a probabilistic model and downstream performance of the model when used for Bayesian optimization (BO). The conclusion from Foldager et al. [2023] was that calibration provided little signal in terms of its utility in BO, whereas Deshpande et al. [2024] argue that calibration in fact does provide major advantages in BO performance. An important point to consider when evaluating these claims is the fact that the way in which calibration was measured was different: Foldager et al. [2023] assess quantile calibration in an offline manner by utilizing a held-out dataset drawn i.i.d. from the underlying data distribution, whereas Deshpande et al. [2024] assess calibration online *during* the BO process. Thus, Deshpande et al. [2024] suggests an online recalibration algorithm, and that calibration assessed online is an important signal of future performance. This work does not suggest a means of performing model selection prior to performing the BO procedure in case there were multiple probabilistic models one could choose to use. Meanwhile, the findings from Foldager et al. [2023] more closely align with the setting where there is a dichotomy between model providers and model users where the users will want to first perform model selection prior to deploying the model, and tells the user that calibration is not a pertinent metric to consider. In any case, neither work elucidates the reason why a model provider should aim to optimize for calibration during training.

A perfect model which recovers the true ground truth distribution will be perfectly calibrated and maximally sharp, regardless of the notion of calibration. However, models will have errors, and in performing evaluation based on quantifiable metrics of model performance, the UQ community has long advocated for achieving calibration and sharpness (instead of standard losses such as likelihood) without clearly enunciating why this should be more desirable, or what the downstream implications would be.

Zhao et al. [2021b] is one of the few works that elucidate the connection between different notions of calibration and their implications for downstream decision-making in the classification setting. They show that a decision calibrated model will provide accurate estimates of the utility incurred for the decision making task. They do not show nor claim, however, that a decision calibrated model will necessarily result in a higher utility. Chapter 5 aims to go a step further and directly optimize the downstream task utility w.r.t the model's predictive distribution, which is to say "I don't necessarily care about calibration or any other UQ metric, rather just give me a predictive distribution which I can make good decisions with".

In the same vein, I'd like to summarize a few additional directions that I personally believe uncertainty should move towards.

High dimensional uncertainty Many machine learning models deal with very high-dimensional target spaces, e.g. image or video models which model distributions that have hundreds of thousand or millions of dimensions, and the current modeling practice with language models treat the modeling problem as a classification problem with hundreds of thousands of classes. Dealing with uncertainty in high-dimensional target spaces is difficult, and even evaluation is also not straightforward. Chapter 4 discusses one method of evaluating and performing calibration for higher dimensional predictive distributions, but it is by no means a perfect measure. Because these models are still probabilistic in nature, it seems natural to expect probabilistic evaluations and interpretations of the model distribution. Connecting such evaluations and interpretation with downstream utility of *using* the models is also an under-explored question.

Sequential uncertainty and sampling Auto-regressive generation is a common strategy in generating predictions for long sequences where the potentially high-dimensional joint distribution is decomposed into a product of conditional distributions. This is the primary method of generating transitions in model-based reinforcement learning when the model learns single step transitions, as well as in language decoding where each token is generated (i.e. sampled) one at a time. Generation at each timestep involves reasoning about uncertainty since it involves sampling from the predictive distribution for the timestep, and ultimately, the generated sequence is a sample from the joint distribution over sequences, which is another distribution over which one should reason about uncertainty. Not only are sequence-level uncertainties difficult to evaluate, but recalibrating single-step uncertainties do not necessarily lead to desirable behavior at the sequence-level. Current methods for steering these distributions are fairly crude (e.g. adjusting the temperature of the softmax distribution at each step), and the connections between aspects of the joint distribution over sequences and the utility of the sampled sequences is unclear. Elucidating such connections would be immensely beneficial to understanding how to steer such models, especially as their usage becomes more and more widespread.

Exploration One major benefit of uncertainty-aware and stochastic models is their propensity for exploration, which is a crucial component in RL, and any system which hopes for novel discovery or emergence. However, if one were to ask, “what kind of distributions are most amenable to effective exploration”, or “what kind of uncertainty metrics are most indicative of curiosity of a model”, there wouldn’t be much of a clear cut answer. In some ways, much of the amazing results from RL stemmed from (almost) random exploration. It is exciting to imagine the progress that is possible with a better understanding of how predictive distributions can be better trained or utilized for even better exploration, especially as it relates to reasoning about out of distribution uncertainty.

Uncertainty quantification is personally an interesting field since at its core, it asks fundamental questions in probability and statistics, and also has extremely practical applications. Once the field is able to bridge the gap between its theory and practice, i.e. elucidate the theoretical properties of uncertainties used in practical applications of uncertainties, I am hopeful that there will be an even heightened appreciation and excitement in this field.

Bibliography

Joseph Abbate, R Conlin, and E Kolemen. Data-driven profile prediction for DIII-D. *Nuclear Fusion*, 61(4):046027, 2021.

Joseph Abbate, Rory Conlin, Ricardo Shousha, Keith Erickson, and Egemen Kolemen. A general infrastructure for data-driven control design and implementation in tokamaks. *Journal of Plasma Physics*, 89(1):895890102, 2023.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135, 2013.

Ross Askanazi, Francis X Diebold, Frank Schorfheide, and Minchul Shin. On the comparison of interval forecasts. *Journal of Time Series Analysis*, 39(6):953–965, 2018.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

EA Baltz, E Trask, M Binderbauer, M Dikovsky, H Gota, R Mendoza, JC Platt, and PF Riley. Achievement of sustained net plasma heating in a fusion experiment with the optometrist algorithm. *Scientific reports*, 7(1):6425, 2017.

Philippe Barbe, Christian Genest, Kilani Ghoudi, and Bruno Remillard. On kendall's process. *journal of multivariate analysis*, 58(2):197–229, 1996.

Alexandre Belloni and Robert L Winkler. On multivariate quantiles under partial ordering. Technical report, 2009.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

VE Bowman, DS Silk, U Dalrymple, and DC Woods. Uncertainty quantification for epidemiological forecasts of covid-19 through combinations of model predictions. *arXiv preprint arXiv:2006.10714*, 2020.

Mark Boyer, Josiah Wai, Mitchell Clement, Egemen Kolemen, Ian Char, Youngseog Chung, Willie Neiswanger, and Jeff Schneider. Machine learning for tokamak scenario optimization: combining accelerating physics models and empirical models. In *APS Division of Plasma Physics Meeting Abstracts*, volume 2021, pages PP11–164, 2021.

MD Boyer, KG Erickson, BA Grierson, DC Pace, JT Scoville, J Rauch, BJ Crowley, JR Ferron, SR Haskey, DA Humphreys, et al. Feedback control of stored energy and rotation with variable beam energy and perveance on diii-d. *Nuclear Fusion*, 59(7):076004, 2019a.

MD Boyer, S Kaye, and K Erickson. Real-time capable modeling of neutral beam injection on nstx-u using neural networks. *Nuclear Fusion*, 59(5):056008, 2019b.

Johannes Bracher, Evan L Ray, Tilmann Gneiting, and Nicholas G Reich. Evaluating epidemic forecasts in an interval format. *PLoS computational biology*, 17(2):e1008618, 2021.

Jochen Bröcker. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643):1512–1519, 2009.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

Barbara Cannas, Alessandra Fanni, A Murari, Alessandro Pau, Giuliana Sias, and JET EFDA Contributors. Automatic disruption classification based on manifold learning for real-time applications on jet. *Nuclear Fusion*, 53(9):093023, 2013.

Alex J Cannon. Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes. *Stochastic environmental research and risk assessment*, 32(11):3207–3225, 2018.

Ian Char and Jeff Schneider. Pid-inspired inductive biases for deep reinforcement learning in partially observable control tasks. *arXiv preprint arXiv:2307.05891*, 2023.

Ian Char, Youngseog Chung, Willie Neiswanger, Kirthevasan Kandasamy, Andrew Oakleigh Nelson, Mark D Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 4629–4640, 2019.

Ian Char, Youngseog Chung, Mark Boyer, Egemen Kolemen, and Jeff Schneider. A model-based reinforcement learning approach for beta control. In *APS Division of Plasma Physics Meeting Abstracts*, volume 2021, pages 11–150, 2021.

Ian Char, Joseph Abbate, László Bardóczi, Mark Boyer, Youngseog Chung, Rory Conlin, Keith Erickson, Viraj Mehta, Nathan Richner, Egemen Kolemen, et al. Offline model-based reinforcement learning for tokamak control. In *Learning for Dynamics and Control Conference*, pages 1357–1372. PMLR, 2023a.

Ian Char, Joseph Abbate, László Bardóczi, Mark Boyer, Youngseog Chung, Rory Conlin, Keith Erickson, Viraj Mehta, Nathan Richner, Egemen Kolemen, et al. Offline model-based reinforcement learning for tokamak control. In *Learning for Dynamics and Control Conference*, pages 1357–1372. PMLR, 2023b.

Ian Char, Youngseog Chung, Rohan Shah, Willie Neiswanger, and Jeff Schneider. Correlated trajectory uncertainty for adaptive sequential decision making. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023c.

Ian Char, Youngseog Chung, Joseph Abbate, Egemen Kolemen, and Jeff Schneider. Full shot predictions for the diii-d tokamak via deep recurrent networks. *arXiv preprint arXiv:2404.12416*, 2024.

Bertrand Charpentier, Oliver Borchert, Daniel Zügner, Simon Geisler, and Stephan Günnemann. Natural posterior network: Deep bayesian predictive uncertainty for exponential family distributions. In *International Conference on Learning Representations*, 2022.

Francis Chen. *An indispensable truth: how fusion power can save the planet*. Springer Science & Business Media, 2011.

Xiong-Hui Chen, Yang Yu, Qingyang Li, Fan-Ming Luo, Zhiwei Qin, Wenjie Shang, and Jieping Ye. Offline model-based adaptable policy learning. *Advances in Neural Information Processing*

Systems, 34:8432–8443, 2021.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Youngseog Chung, Ian Char, Willie Neiswanger, Kirthevasan Kandasamy, Andrew Oakleigh Nelson, Mark D Boyer, Egemen Kolemen, and Jeff Schneider. Offline contextual bayesian optimization for nuclear fusion. *arXiv preprint arXiv:2001.01793*, 2020a.

Youngseog Chung, Willie Neiswanger, Ian Char, and Jeff Schneider. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *arXiv preprint arXiv:2011.09588*, 2020b.

Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. Uncertainty toolbox: an open-source library for assessing, visualizing, and improving uncertainty quantification. *arXiv preprint arXiv:2109.10254*, 2021a.

Youngseog Chung, Willie Neiswanger, Ian Char, and Jeff Schneider. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *Advances in Neural Information Processing Systems*, 34:10971–10984, 2021b.

Youngseog Chung, Aaron Rumack, and Chirag Gupta. Parity calibration. In *Uncertainty in Artificial Intelligence*, pages 413–423. PMLR, 2023a.

Youngseog Chung, Aaron Rumack, and Chirag Gupta. Parity calibration. In Robin J. Evans and Ilya Shpitser, editors, *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, pages 413–423. PMLR, 31 Jul–04 Aug 2023b. URL https://proceedings.mlr.press/v216/chung23a.html.

Youngseog Chung, Ian Char, and Jeff Schneider. Sampling-based multi-dimensional recalibration. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 21–27 Jul 2024.

Daniel Clery. *A Piece of the sun: the quest for fusion energy*. Abrams, 2014.

Estee Y Cramer, Yuxin Huang, Yijin Wang, Evan L Ray, Matthew Cornell, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Katie House, Dasuni Jayawardena, Abdul H Kanji, Ayush Khandelwal, Khoa Le, Jarad Niemi, Ariane Stark, Apurv Shah, Nutcha Wattanachit, Martha W Zorn, Nicholas G Reich, and US COVID-19 Forecast Hub Consortium. The United States COVID-19 Forecast Hub dataset. *medRxiv*, 2021. doi: 10.1101/2021.11.04.21265886.

Estee Y. Cramer, Evan L. Ray, Velma K. Lopez, et al., and Nicholas G. Reich. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the united states. *Proceedings of the National Academy of Sciences*, 119(15):e2113561119, 2022. doi: 10.1073/pnas.2113561119.

Peng Cui, Wenbo Hu, and Jun Zhu. Calibrated reliable regression using maximum mean discrepancy. In *Advances in Neural Information Processing Systems*, 2020.

Niccolò Dalmasso, Taylor Pospisil, Ann B Lee, Rafael Izbicki, Peter E Freeman, and Alex I Malz.

Conditional density estimation tools in python and r with applications to photometric redshifts and likelihood-free cosmological inference. *Astronomy and Computing*, 30:100362, 2020.

A Philip Dawid. The well-calibrated Bayesian. *Journal of the American Statistical Association*, 77 (379):605–610, 1982.

A Philip Dawid. The geometry of proper scoring rules. *Annals of the Institute of Statistical Mathematics*, 59:77–93, 2007.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897): 414–419, 2022.

Morris H DeGroot and Stephen E Fienberg. Assessing probability assessors: Calibration and refinement. Technical report, Carnegie Mellon University, 1981.

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

Markus Deserno. How to generate exponentially correlated gaussian random numbers. *Department of Chemistry and Biochemistry UCLA, USA*, 2002.

Shachi Deshpande and Volodymyr Kuleshov. Calibration improves bayesian optimization. *arXiv preprint arXiv:2112.04620*, 2021.

Shachi Deshpande, Charles Marx, and Volodymyr Kuleshov. Online calibrated and conformal prediction improves bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 1450–1458. PMLR, 2024.

Nicki S Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. *arXiv preprint arXiv:1906.03260*, 2019.

Josip Djolonga, Frances Hubis, Matthias Minderer, Zachary Nado, Jeremy Nixon, Rob Romijnders, Dustin Tran, and Mario Lucic. Robustness Metrics, 2020. URL https://github.com/google-research/robustness_metrics.

Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In *International conference on machine learning*, pages 2690–2700. PMLR, 2020.

Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51:357–377, 2019.

Matteo Fasiolo, Simon N Wood, Margaux Zaffran, Raphaël Nedellec, and Yannig Goude. Fast calibrated additive quantile regression. *Journal of the American Statistical Association*, pages 1–11, 2020.

Corporación Favorita, inversion, Julia Elliot, and Mark McDonald. Corporación favorita grocery sales forecasting, 2017. URL https://kaggle.com/competitions/favorita-grocery-sales-forecasting.

Shai Feldman, Stephen Bates, and Yaniv Romano. Calibrated multiple-output quantile regression with representation learning. *Journal of Machine Learning Research*, 24(24):1–48, 2023.

Jonathan Foldager, Mikkel Jordahn, Lars K Hansen, and Michael R Andersen. On the role of model uncertainties in bayesian optimisation. In *Uncertainty in Artificial Intelligence*, pages 592–601.

PMLR, 2023.

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

Yichen Fu, David Eldon, Keith Erickson, Kornee Kleijwegt, Leonard Lupin-Jimenez, Mark D Boyer, Nick Eidietis, Nathaniel Barbour, Olivier Izacard, and Egemen Kolemen. Machine learning control for disruption and tearing mode avoidance. *Physics of Plasmas*, 27(2):022501, 2020.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international Conference on Machine Learning*, pages 1050–1059, 2016.

Luiz G. Galvão and M. Nazmul Huda. Pedestrian and vehicle behaviour prediction in autonomous vehicle system — a review. *Expert Systems with Applications*, 238:121983, 2024. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2023.121983. URL https://www.sciencedirect.com/science/article/pii/S0957417423024855.

Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

Christian Genest and Louis-Paul Rivest. On the multivariate probability integral transformation. *Statistics & probability letters*, 53(4):391–399, 2001.

Dibya Ghosh, Anurag Ajay, Pulkit Agrawal, and Sergey Levine. Offline rl policies should be trained to be adaptive. In *International Conference on Machine Learning*, pages 7513–7530. PMLR, 2022.

David Ginsbourger, Jean Baccou, Clément Chevalier, Frédéric Perales, Nicolas Garland, and Yann Monerie. Bayesian adaptive reconstruction of profile optima and optimizers. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):490–510, 2014.

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

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

BA Grierson, X Yuan, M Gorelenkova, S Kaye, NC Logan, O Meneghini, SR Haskey, J Buchanan, M Fitzgerald, SP Smith, et al. Orchestrating transp simulations for interpretative and predictive tokamak modeling with omfit. *Fusion Science and Technology*, 74(1-2):101–115, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

Chirag Gupta and Aaditya Ramdas. Distribution-free calibration guarantees for histogram binning without sample splitting. In *International Conference on Machine Learning*, 2021.

Chirag Gupta and Aaditya Ramdas. Online Platt scaling with calibeating. In *International Conference on Machine Learning*, 2023.

Chirag Gupta, Aleksandr Podkopaev, and Aaditya Ramdas. Distribution-free binary classification: prediction sets, confidence intervals and calibration. In *Advances in Neural Information Processing Systems*, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Lingxin Hao, Daniel Q Naiman, and Daniel Q Naiman. *Quantile regression*. Number 149. Sage, 2007.

Diana Harrison, David Sutton, Pedro Carvalho, and Michael Hobson. Validation of bayesian posterior distributions using a multidimensional kolmogorov–smirnov test. *Monthly Notices of the Royal Astronomical Society*, 451(3):2610–2624, 2015.

Ursula Hébert-Johnson, Michael P Kim, Omer Reingold, and Guy N Rothblum. Calibration for the (computationally-identifiable) masses. *arXiv preprint arXiv:1711.08513*, 2017.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Michael P Holmes, Alexander G Gray, and Charles Lee Isbell. Fast nonparametric conditional density estimation. In *Uncertainty in Artificial Intelligence*, 2007.

David Humphreys, G Ambrosino, Peter de Vries, Federico Felici, Sun H Kim, Gary Jackson, A Kallenbach, Egemen Kolemen, J Lister, D Moreau, et al. Novel aspects of plasma control in iter. *Physics of Plasmas*, 22(2):021806, 2015.

Rob J Hyndman. Computing and graphing highest density regions. *The American Statistician*, 50 (2):120–126, 1996.

Rob J Hyndman, David M Bashtannyk, and Gary K Grunwald. Estimating and visualizing conditional densities. *Journal of Computational and Graphical Statistics*, 5(4):315–336, 1996.

Rafael Izbicki, Gilson Shimizu, and Rafael B Stern. Cd-split and hpd-split: Efficient conformal regions in high dimensions. *The Journal of Machine Learning Research*, 23(1):3772–3803, 2022.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Jena Weather Station at Max Planck Institute for Biogeochemistry. Jena climate data. 2016. URL https://www.bgc-jena.mpg.de/wetter/.

Soyoung Jeon, Christopher J Paciorek, and Michael F Wehner. Quantile-based bias correction and uncertainty quantification of extreme event attribution statements. *Weather and Climate Extremes*, 12:24–32, 2016.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

HM Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE access*, 6: 36218–36234, 2018.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

Kirthevasan Kandasamy, Willie Neiswanger, Reed Zhang, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Myopic posterior sampling for adaptive goal oriented design of experiments. In *Proceedings of the 36th International Conference on Machine Learning*. JMLR. org, 2019.

Julian Kates-Harbeck, Alexey Svyatkovskiy, and William Tang. Predicting disruptive instabilities in controlled fusion plasmas through deep learning. *Nature*, page 1, 2019.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

Koenker. *Quantile Regression (Econometric Society Monographs; No. 38)*. Cambridge university press, 2005.

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

Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4): 143–156, 2001.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Grzegorz Kowal, Diego A Falceta-Gonçalves, Alex Lazarian, and Ethan T Vishniac. Kelvin–helmholtz versus tearing instability: What drives turbulence in stochastic reconnection? *The Astrophysical Journal*, 892(1):50, 2020.

Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

Volodymyr Kuleshov and Shachi Deshpande. Calibrated and sharp uncertainties in deep learning via density estimation. In *International Conference on Machine Learning*, pages 11683–11693. PMLR, 2022.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

Meelis Kull, Telmo M Silva Filho, and Peter Flach. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. *Electronic Journal of Statistics*, 11(2): 5052–5080, 2017.

Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. 1964.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning:

Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. Accurate uncertainty estimation and decomposition in ensemble learning. In *Advances in Neural Information Processing Systems*, 2019.

James L Luxon. A design retrospective of the DIII-D tokamak. *Nuclear Fusion*, 42(5):614, 2002.

Katarzyna Maciejowska, Jakub Nowotarski, and Rafał Weron. Probabilistic forecasting of electricity spot prices using factor quantile regression averaging. *International Journal of Forecasting*, 32 (3):957–965, 2016.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13153–13164, 2019.

Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated model-based deep reinforcement learning. *arXiv preprint arXiv:1906.08312*, 2019.

Charles Marx, Shengjia Zhao, Willie Neiswanger, and Stefano Ermon. Modular conformal calibration. In *International Conference on Machine Learning*, 2022.

Viraj Mehta, Ian Char, Willie Neiswanger, Youngseog Chung, Andrew Nelson, Mark Boyer, Egemen Kolemen, and Jeff Schneider. Neural dynamical systems: Balancing structure and flexibility in physical prediction. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3735–3742. IEEE, 2021a.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. *arXiv preprint arXiv:2112.05244*, 2021b.

Viraj Mehta, Ian Char, Joseph Abbate, Rory Conlin, Mark Boyer, Stefano Ermon, Jeff Schneider, and Willie Neiswanger. Exploration via planning for information about the optimal trajectory. *Advances in Neural Information Processing Systems*, 35:28761–28775, 2022.

Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7(Jun): 983–999, 2006.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

Kevin Joseph Montes, Cristina Rea, Robert Granetz, Roy Alexander Tinguely, Nicholas W Eidietis, O Meneghini, Dalong Chen, Biao Shen, Bingjia Xiao, Keith Erickson, et al. Machine learning for disruption warning on alcator c-mod, diii-d, and east. *Nuclear Fusion*, 2019.

Edward Morse. *Nuclear Fusion*. Springer, 2018.

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

Zachary Nado, Neil Band, Mark Collier, Josip Djolonga, Michael Dusenberry, Sebastian Farquhar, Angelos Filos, Marton Havasi, Rodolphe Jenatton, Ghassen Jerfel, Jeremiah Liu, Zelda Mariet, Jeremy Nixon, Shreyas Padhy, Jie Ren, Tim Rudner, Yeming Wen, Florian Wenzel, Kevin Murphy, D. Sculley, Balaji Lakshminarayanan, Jasper Snoek, Yarin Gal, and Dustin Tran. Uncertainty Baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In *AAAI Conference on Artificial Intelligence*, 2015.

Roger B Nelsen, José Juan Quesada-Molina, José Antonio Rodríguez-Lallena, and Manuel Úbeda-Flores. Kendall distribution functions. *Statistics & probability letters*, 65(3):263–268, 2003.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *International Conference on Machine Learning*, 2005.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *International Conference on Neural Networks*, 1994.

Michael Pearce and Juergen Branke. Continuous multi-task bayesian optimisation with correlation. *European Journal of Operational Research*, 270(3):1074–1085, 2018.

Tim Pearce, Felix Leibfried, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. Uncertainty in neural networks: Approximately bayesian ensembling. *arXiv preprint arXiv:1810.05546*, 2018a.

Tim Pearce, Mohamed Zaki, Alexandra Brintrup, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. *arXiv preprint arXiv:1802.07167*, 2018b.

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

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *ML Summer School 2003 (Canberra, Australia)*. Springer, 2004.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2005. ISBN 9780262182539.

James B Rawlings. Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52, 2000.

Filipe Rodrigues and Francisco C Pereira. Beyond expectation: deep joint mean and quantile regression for spatiotemporal problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016.

Roshni Sahoo, Shengjia Zhao, Alyssa Chen, and Stefano Ermon. Reliable decisions with threshold calibration. In *Advances in Neural Information Processing Systems*, 2021.

Tárik S Salem, Helge Langseth, and Heri Ramampiaro. Prediction intervals: Split normal mixture from quality-driven deep ensembles. In *Conference on Uncertainty in Artificial Intelligence*, pages 1179–1187. PMLR, 2020.

JT Scoville, DA Humphreys, JR Ferron, and P Gohil. Simultaneous feedback control of plasma rotation and stored energy on the diii-d tokamak. *Fusion engineering and design*, 82(5-14):

1045–1050, 2007.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*, 2022.

J Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Development of an operation trajectory design algorithm for control of multiple 0d parameters using deep reinforcement learning in kstar. *Nuclear Fusion*, 62(8):086049, 2022.

Jaemin Seo, Y-S Na, B Kim, CY Lee, MS Park, SJ Park, and YH Lee. Feedforward beta control in the kstar tokamak by deep reinforcement learning. *Nuclear Fusion*, 61(10):106010, 2021.

Burr Settles. Active learning literature survey. 2009.

David MH Sexton, James M Murphy, Mat Collins, and Mark J Webb. Multivariate probabilistic projections using imperfect climate models part i: outline of methodology. *Climate dynamics*, 38: 2513–2542, 2012.

Amar Shah and Zoubin Ghahramani. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pages 1919–1927, 2016.

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, 2015.

N Silver. When we say 70 percent, it really means 70 percent, 2019.

Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. Distribution calibration for regression. In *International Conference on Machine Learning*, pages 5897–5906. PMLR, 2019.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1):211–225, 2011.

Winfried Stute et al. On almost sure convergence of conditional empirical distribution functions. *The Annals of Probability*, 14(3):891–901, 1986.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. In *Advances in Neural Information Processing Systems*, 2019.

Ichiro Takeuchi, Quoc V Le, Timothy D Sears, and Alexander J Smola. Nonparametric quantile estimation. *Journal of machine learning research*, 7(Jul):1231–1264, 2006.

William Tang, Matthew Parsons, Eliot Feibush, A Murari, J Vega, A Pereira, and J Choi. Big data machine learning for disruption predictions. In *26th IAEA Fusion Energy Conference-IAEA CN-234, Paper Number EX/P6–47*, 2016.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Saul Toscano-Palmerin and Peter I Frazier. Bayesian optimization with expensive integrands. *arXiv preprint arXiv:1803.08661*, 2018.

Kevin Tran, Willie Neiswanger, Junwoong Yoon, Qingyang Zhang, Eric Xing, and Zachary W Ulissi. Methods for comparing uncertainty quantifications for material property predictions. *Machine Learning: Science and Technology*, 1(2):025006, 2020.

ITER Physics Expert Group on Confinement Transport, , ITER Physics Expert Group on Confinement Modelling Database, , and ITER Physics Basis Editors. Chapter 2: Plasma confinement and transport. *Nuclear Fusion*, 39(12):2175–2249, 1999.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *The Journal of Machine Learning Research*, 12:2411–2414, 2011.

Rebecca Wexler. When a computer program keeps you in jail: How computers are harming criminal justice. *New York Times*, 13, 2017.

James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pages 10292–10302. PMLR, 2020.

Robert L Winkler. A decision-theoretic approach to interval estimation. *Journal of the American Statistical Association*, 67(337):187–191, 1972.

Qifa Xu, Kai Deng, Cuixia Jiang, Fang Sun, and Xue Huang. Composite quantile regression neural network with applications. *Expert Systems with Applications*, 76:129–139, 2017.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *International Conference on Machine Learning*, 2001.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *International Conference on Knowledge Discovery and Data mining*, 2002.

David Zhao, Niccolò Dalmasso, Rafael Izbicki, and Ann B Lee. Diagnostics for conditional density models and bayesian inference algorithms. In *Uncertainty in Artificial Intelligence*, pages 1830–1840. PMLR, 2021a.

Shengjia Zhao, Tengyu Ma, and Stefano Ermon. Individual calibration with randomized forecasting. In *International Conference on Machine Learning*, 2020.

Shengjia Zhao, Michael Kim, Roshni Sahoo, Tengyu Ma, and Stefano Ermon. Calibrating predictions to decisions: A novel approach to multi-class calibration. In *Advances in Neural Information Processing Systems*, 2021b.

Johanna F Ziegel and Tilmann Gneiting. Copula calibration. *Electronic journal of statistics*, 8(2): 2619–2638, 2014.