Calibrated and Sharp Uncertainties in Deep Learning via Density Estimation

Volodymyr Kuleshov [1] Shachi Deshpande [1]

Abstract

Accurate probabilistic predictions can be characterized by two properties—calibration and sharpness. However, standard maximum likelihood training yields models that are poorly calibrated and thus inaccurate—a 90% confidence interval typically does not contain the true outcome 90% of the time. This paper argues that calibration is important in practice and is easy to maintain by performing low-dimensional density estimation. We introduce a simple training procedure based on recalibration that yields calibrated models without sacrificing overall performance; unlike previous approaches, ours ensures the most general property of distribution calibration and applies to any model, including neural networks. We formally prove the correctness of our procedure assuming that we can estimate densities in low dimensions and we establish uniform convergence bounds. Our results yield empirical performance improvements on linear and deep Bayesian models and suggest that calibration should be increasingly leveraged across machine learning.

1. Introduction

Accurate probabilistic predictions can be characterized by two properties—calibration and sharpness (Gneiting et al., 2007). Intuitively, calibration means that a 90% confidence interval contains the true outcome 90% of the time. Sharpness means that these confidence intervals are narrow. These properties are grounded in statistical theory and are used to evaluate forecasts in domains such as meteorology and medicine (Gneiting & Raftery, 2005; 2007).

This paper argues for reasoning about uncertainty in deep learning in terms of calibration and sharpness and proposes

[1]Department of Computer Science, Cornell Tech and Cornell University, New York, NY. Correspondence to: Volodymyr Kuleshov <kuleshov@cornell.edu>, Shachi Deshpande <ssd86@cornell.edu>.
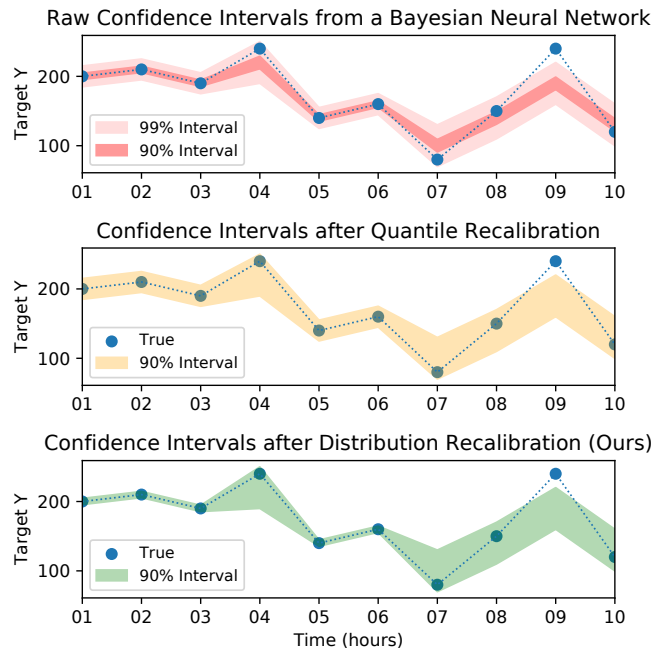
Figure 1. Top: Credible intervals (90% & 99%) from a Bayesian neural network on a time series forecasting task. The 90% credible interval around the forecast (red) is miscalibrated and inaccurate: half of the points are outside of it. Middle: Quantile recalibration (Kuleshov et al., 2018) relabels the 99% credible interval as a 90% interval, which now correctly contains 9/10 points (orange). Bottom: Our recalibration method enforces distribution calibration—the 90% interval is unchanged at points where it is narrow and expands where it is wide. This yields an improved and narrower calibrated 90% interval (green).

simple algorithms for enforcing these properties. Standard maximum likelihood training yields models that are poorly calibrated—a 90% confidence interval typically does not contain the true outcome 90% of the time (Guo et al., 2017). Popular approaches based on dropout or ensembling (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017) improve model probabilities, yet do not enforce calibration (Kuleshov et al., 2018). Recalibration techniques (Platt, 1999) yield calibrated and sharp forecasts, but enforce a limited notion of quantile calibration (Kuleshov et al., 2018), require complex variational approximations (Song et al., 2019), and are not well understood theoretically.

This paper argues that calibration is important in practice

and is easy to maintain by performing low-dimensional density estimation. We introduce a simple recalibration procedure based on quantile function regression (Si et al., 2021), which yields calibrated models without sacrificing overall performance. Unlike previous approaches, ours guarantees the general property of distribution calibration (Song et al., 2019) and applies to any model, including neural networks. We prove the correctness of our procedure assuming that we can estimate densities in low dimensions and we establish uniform convergence bounds.

Empirically, we find that our method consistently outputs well-calibrated predictions in linear and deep Bayesian models, and improves performance on downstream tasks with minimal implementation overhead. A key takeaway is that distribution calibration may be simpler to obtain than previously thought, and we argue that it should be enforced in predictive models and leveraged across machine learning.

Contributions. In summary, we make three contributions. We show that probability calibration (including distribution calibration) can be maintained via density estimation in low dimensions. We then introduce a simple algorithm based on recalibration that is simpler and more broadly applicable than previous approaches. We complement this algorithm with a theoretical analysis establishing guarantees on calibration and vanishing regret. A key takeaway is that calibration may be easier to maintain than previously thought and should be leveraged more broadly throughout machine learning.

2. Background

2.1. Predictive Uncertainty in Machine Learning

Supervised machine learning models predict a probability distribution over the target variable—e.g. class membership probabilities or the parameters of a normal distribution. We seek to produce models with accurate probabilistic outputs.

Notation. We predict a target $y \in \mathcal{Y}$—where $\mathcal{Y}$ is either discrete (it's a classification problem) or $\mathcal{Y} = \mathbb{R}$ (it's a regression problem)—using input features $x \in \mathcal{X}$. We are given a forecaster $H : \mathcal{X} \to \Delta_{\mathcal{Y}}$, which outputs a probability distribution $F(y) : \mathcal{Y} \to [0, 1]$ within the set $\Delta_{\mathcal{Y}}$ of distributions over $\mathcal{Y}$; the probability density function of F is f. We are also given a training set $\mathcal{D} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^n$ and a calibration set $\mathcal{C} = \{(x_j, y_j) \in \mathcal{X} \times \mathcal{Y}\}_{j=1}^m$, each consisting of i.i.d. realizations of random variables $X, Y \sim \mathbb{P}$, where $\mathbb{P}$ is the data distribution.

2.2. Calibration and Sharpness

Accurate probabilistic predictions are characterized by two properties—calibration and sharpness. Calibration means

that an 80% confidence interval contains the target y 80% of the time. Sharpness means that the intervals are tight.

Quantile Calibration in Regression Formally, Kuleshov et al. (2018) define calibration in a regression setting as

$$\mathbb{P}(Y \leq \mathrm{CDF}_{F_X}^{-1}(p)) = p \text{ for all } p \in [0, 1], \quad (1)$$

where $F_X = H(X)$ is the forecast at X, itself a random variable that takes values in $\Delta_{\mathcal{Y}}$. Intuitively, for each X, Y, the Y is contained in the p-th confidence interval $(-\infty, \mathrm{CDF}_{F_X}^{-1}(p)]$ predicted at X a fraction p of the time.

Distribution Calibration Song et al. (2019) defines a stronger notion of distribution calibration as

$$\mathbb{P}(Y = y \mid F_X = F) = f(y) \text{ for all } y \in \mathcal{Y}, F \in \Delta_{\mathcal{Y}}, \quad (2)$$

where $F_X = H(X)$ is the random forecast at X and f is its probability density or probability mass function. When $\mathcal{Y} = \{0, 1\}$ and F_X is Bernoulli with parameter p, we can write (2) as $\mathbb{P}(Y = 1 \mid F_X = p) = p$. Intuitively, the true probability of $Y = 1$ is p conditioned on predicting it as p.

Equation 2 extends to regression as well. For example, if F is a Gaussian with variance σ^2, this definition asks that the data distribution conditioned on predicting F also has variance σ^2. Distribution calibration also implies that quantile calibration holds (Song et al., 2019).

Calibration vs. Sharpness Calibration by itself is not sufficient to produce a useful forecast. For example, a binary classifier that always predicts the marginal data probability $\mathbb{P}(Y = 1)$ is calibrated; however it does not even use the features X and thus cannot be accurate.

In order to be useful, forecasts must also be *sharp*. Intuitively, this means that the confidence intervals should be as tight as possible. Formally, sharpness is quantified via the entropy of F_X; see Table 1 and the

2.3. Evaluating Probabilistic Predictions

We evaluate probabilistic predictions using the framework of proper scoring rules (Gneiting & Raftery, 2007). Formally, let $L : \Delta_{\mathcal{Y}} \times \mathcal{Y} \to \mathbb{R}$ denote a loss between a probabilistic forecast $F \in \Delta_{\mathcal{Y}}$ and a realized outcome $y \in \mathcal{Y}$. Given a distribution $G \in \Delta_{\mathcal{Y}}$ over y, we use $L(F, G)$ to denote the expected loss $L(F, G) = \mathbb{E}_{y \sim G} L(F, y)$.

We say that L is a *proper loss* if it is minimized by G when G is the true distribution for y: $L(F, G) \geq L(G, G)$ for all F. One example is the log-likelihood $L(F, y) = -\log f(y)$. Another example is the check score for $\tau \in [0, 1]$:

$$\rho_\tau(F, y) = \begin{cases} \tau(y - F^{-1}(\tau)) & \text{if } y \geq f \\ (1 - \tau)(F^{-1}(\tau) - y) & \text{otherwise.} \end{cases} \quad (3)$$
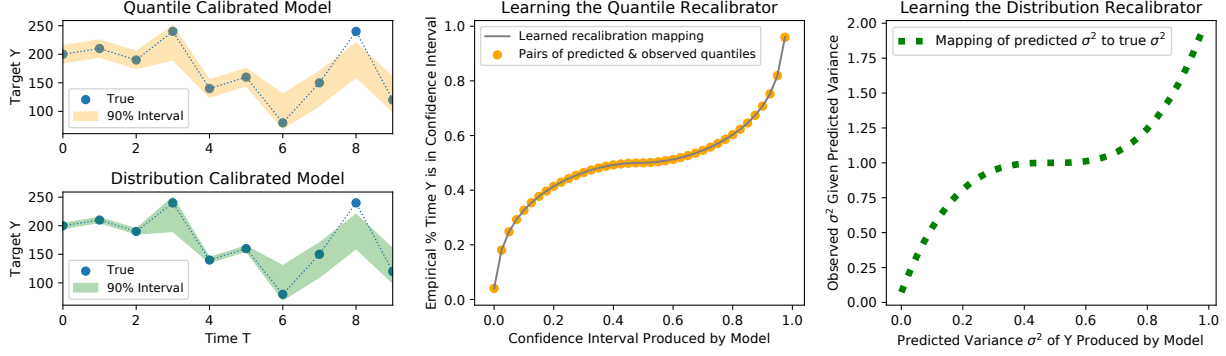
Figure 2. Quantile vs. distribution calibration.. Left: A Bayesian neural network outputs probabilistic forecasts recalibrated using quantile and distribution recalibration. Middle: In quantile recalibration, we observe each confidence interval and count how many times y falls in that interval—this mapping is learned by R and is an estimate of $\mathbb{P}(F_X(Y) \leq p)$. Right: In distribution recalibration, we represent forecasts by features (e.g., their variance σ^2) and learn a density $\mathbb{P}(Y \mid F)$ conditioned on these features. In this example, we learn to remap predicted to observed variance—this allows adjusting different variances differently, yielding tighter intervals (bottom left).

Proper Loss	Loss $L(F,G)$	Calibration $L_c(F,Q)$	Refinement $L_r(Q)$
Logarithmic	$\mathbb{E}_{y \sim G} \log f(y)$	$\mathrm{KL}(q \| f)$	$H(q)$
CRPS	$\mathbb{E}_{y \sim G}(F(y) - G(y))^2$	$\int_{-\infty}^{\infty}(F(y) - Q(y))^2 \mathrm{dy}$	$\int_{-\infty}^{\infty} Q(y)(1 - Q(y))dy$
Quantile	$\mathbb{E}_{y \sim G}^{\tau \in U[0,1]} \rho_\tau(F,y)$	$\int_0^1 \int_{Q^{-1}(\tau)}^{F^{-1}(\tau)}(Q(y) - \tau)dyd\tau$	$\mathbb{E}_{y \sim Q}^{\tau \in U[0,1]} \rho_\tau(Q,y)$

Table 1. Examples of three proper losses: the log-loss, the continuous ranked probability score (CRPS), and the quantile loss. A proper loss $L(F,G)$ between distributions F, G—assumed here to be cumulative distribution functions (CDFs)—decomposes into a calibration loss term $L_c(F,Q)$ (also known as reliability) plus a refinement term $L_r(Q)$ (which itself decomposes into a sharpness and an uncertainty term). Here, $Q(y)$ denotes the CDF of $\mathbb{P}(Y = y \mid F_X = F)$, and $q(y), f(y)$ are the probability density functions of Q and F.

See Table 1 for additional examples.

Every proper loss decomposes into a sum of the following terms (Gneiting et al., 2007):

$$\text{proper loss} = \text{calibration} \underbrace{-\text{sharpness} + \text{irreducible term}}_{\text{refinement term}}.$$

Thus, calibration and sharpness are both necessary and sufficient characteristics of an accurate probabilistic forecast.

2.4. Training Calibrated Models

Typically, machine learning models do not output accurate and calibrated probabilities out-of-the-box (Niculescu-Mizil & Caruana, 2005; Guo et al., 2017). Recalibration is an alternative training strategy in which a model $R : \Delta_{\mathcal{Y}} \to \Delta_{\mathcal{Y}}$ is fit on the calibration set $\mathcal{C}$ such that the forecasts $R \circ F$ are calibrated (Platt, 1999; Vovk et al., 2005)

In the setting of quantile calibration, one can see that choosing $R : [0,1] \to [0,1]$ to be $R(p) = \mathrm{CDF}_{F_X}^{-1}(p)$ yields a calibrated $R \circ F$ as in (1). Thus, recalibration can be understood as estimating the above distribution.

3. Training Distribution Calibrated Models

In this section, we introduce methods for training probabilistic classification and regression models that ensure the strong property of distribution calibration. Our approach is an instance of recalibration (Platt, 1999; Kuleshov et al., 2018), in which we learn an auxiliary model $R : \Delta_{\mathcal{Y}} \to \Delta_{\mathcal{Y}}$ such that the joint model $R \circ H$ is calibrated.

3.1. Recalibration as Density Estimation

When x_t, y_t are sampled i.i.d. from $\mathbb{P}$, choosing

$$R(F) = \mathbb{P}(Y \mid H(X) = F) \qquad (4)$$

yields a distribution calibrated model $R \circ H$; we provide a formal proof in Section 4. Thus, our task is to learn the density (4); this requires solving two challenges: (1) conditioning R on arbitrary forecasts F; (2) choosing a learning objective for R.

Challenge 1: Conditioning on F Our approach is to define a featurization $\phi : \Delta_{\mathcal{Y}} \to \mathbb{R}^p$ of F, such that $\phi(F)$ is represented by a small number of parameters p. Hence, learning $\mathbb{P}(Y \mid \phi(F))$ involves a tractable low-dimensional

density estimation problem for which there exist efficient and provably correct algorithms (Wasserman, 2006).

In practice, most models (especially neural networks) already assume a parametric form for F (e.g., μ, σ^2 for a Gaussian). Alternatively, we may featurize F via some of its quantiles. We provide several additional examples of features in Section 3.2.

Challenge 2: A Learning Objective for R Our approach will optimize a proper scoring rule L. Specifically, we choose a recalibrator R that minimizes the objective $\sum_{x,y \in \mathcal{C}} L(R(F_x), y)$ over a calibration dataset $\mathcal{C} = \{x_j, y_j\}_{j=1}^m$ sampled i.i.d. from $\mathbb{P}$:. Observe that

$$\sum_{x,y \in \mathcal{C}} L(R(F_x), y) \approx \mathbb{E}_{F_X \sim \mathbb{P}} \mathbb{E}_{Y \sim \mathbb{P}(Y|F_X)} [L(R(F_X), Y)]$$
$$= \mathbb{E}_{F_X \sim \mathbb{P}} [L(R(F_X), \mathbb{P}(Y|F_X))],$$

where $F_X = H(X)$. The first line follows by Monte Carlo approximation and the second line follows from the definition of a proper loss.

Thus, minimizing $L(R(F_x)), y)$ over $\mathcal{C}$ yields an estimate of $\mathbb{P}(Y|F_X)) = \mathbb{P}(Y|H(X) = F))$, which is the probability we seek to learn. Overall, our high-level approach can be summarized by the following algorithm; we describe specific instantiations of L, ϕ, and R below.

Algorithm 1 Distribution Recalibration Framework

Input: Pre-trained model $H : \mathcal{X} \to \Delta_{\mathcal{Y}}$, featurizer $\phi : \Delta_{\mathcal{Y}} \to \mathbb{R}^p$, recalibrator $R : \mathbb{R}^p \to \Delta_{\mathcal{Y}}$, calibration set $\mathcal{C}$
Output: Recalibrated model $R \circ H : \mathcal{X} \to \Delta_{\mathcal{Y}}$

1. Create a training set for recalibrator:
   $\mathcal{S} = \{(\phi(H(x)), y) \mid x, y \in \mathcal{C}\}$

2. Fit the recalibrator R on $\mathcal{S}$ using a proper loss:
   $\min_R \sum_{(\phi,y) \in \mathcal{S}} L(R(\phi), y)$

3.2. Density Estimation via Quantile Function Regression

Next, we introduce specialized versions of Algorithm 1 for the settings of probabilistic regression and classification and we define additional details of the method—the model R, the features ϕ, and the objective L.

Recall that our goal is to estimate the distribution $\mathbb{P}(Y \mid H(X) = F)$ (Equation 4). We choose to represent this distribution via its cumulative distribution function (CDF) or, equivalently, its inverse, the quantile function (QF). Learning a model of the CDF or of the QF facilitates computing confidence intervals and yields more numerically stable algorithms.

Without loss of generality, we train R to fit the QF; the density can be obtained from the CDF or QF by via a derivative. This approach also yields the following equivalent notion of distribution calibration:

$$\mathbb{P}(Y \leq y \mid F_X = F) = F(Y \leq y) \text{ for all } y \in \mathcal{Y}, F \in \Delta_{\mathcal{Y}},$$

Our approach for estimating the QF relies on quantile function regression (Si et al., 2021). We define this method below in terms of three components—the model R, the features ϕ, and the objective L. The resulting method is shown in Algorithm 2.

Model We learn a model $R_\theta(\tau; \phi(F)) : \mathbb{R} \times \mathbb{R}^p \to \mathbb{R}$ of the inverse of the CDF of $\mathbb{P}(Y \mid H(X) = F)$. Specifically, $R_\theta(\tau; \phi(F))$ takes in a scalar τ and features ϕ, and outputs an estimate of the τ-th quantile of $\mathbb{P}(Y \mid H(X) = F)$. In our experiments, R is parameterized by a fully-connected neural network with inputs τ and ϕ.

Features We form a p-dimensional representation of F by featurizing it via its quantiles $\phi(F) = (F^{-1}(\alpha_i))_{i=1}^p$ for some sequence of p levels α_i, typically uniform in $[0, 1]$. This parameterization works across diverse types of F, including parametric functions (e.g., Gaussians) or F represented via a set of samples.

Objective We train R using the quantile proper loss L_q (see Table 1). Specifically, when $R(\tau; \phi)$ is an estimate of the τ-th quantile, our objective becomes

$$\mathbb{E}_{\phi,y} L_q(\phi, y) = \mathbb{E}_{\phi,y} \mathbb{E}_{\tau \in U[0,1]} \rho_\tau(R^{-1}(\tau; \phi), y), \quad (5)$$

where ρ_τ is the check score (Equation 3). We fit the objective using gradient descent, approximating the expectations using Monte Carlo.

Algorithm 2 Distribution Calibrated Regression

Input: Pre-trained model $H : \mathcal{X} \to \Delta_{\mathcal{Y}}$, recalibrator $R : [0, 1] \times \Phi(\Delta_{\mathcal{Y}}) \to \mathbb{R}$, training set $\mathcal{D}$, calibration set $\mathcal{C}$
Output: Recalibrated model $R \circ H : \mathcal{X} \to ([0, 1] \to \mathbb{R})$

1. Create a training set for recalibrator:
   $\mathcal{S} = \{(\phi(H(x)), y) \mid x, y \in \mathcal{C}\}$

2. Fit the recalibrator R on $\mathcal{S}$ using a quantile loss:
   $\min_R \sum_{(\phi,y) \in \mathcal{S}} \mathbb{E}_{\tau \in U[0,1]} \rho_\tau(R^{-1}(\tau; \phi), y)$

3.3. Distribution Calibrated Classification

We also propose an improved algorithm for the case when Y is discrete. Our approach uses the fact that the distribution of Y can be represented via its probability mass function (PMF). We specify this algorithm below in terms of the model R, features ϕ, and objective L.

Features, Model, and Objective In classification, each $F \in \Delta_{\mathcal{Y}}$ is categorical and can be represented as a vector $p_F \in \Delta_K$ of $K \geq 2$ class membership probabilities living in a simplex Δ_K over K-dimensional probability vectors.

A natural architecture for R is a sequence of dense layers mapping the simplex Δ_K into Δ_K. Such network recalibrators can be implemented easily within deep learning frameworks and work well in practice despite being non-convex, as we later demonstrate empirically.

A natural choice of proper scoring rule is the log-loss $L_{\log}$. Optimizing it is a standard supervised learning problem.

Algorithm 3 Distribution Calibrated Classification

Input: Pre-trained model $H : \mathcal{X} \to \Delta_K$, recalibrator $R : \Delta_K \to \Delta_K$, training set $\mathcal{D}$, calibration set $\mathcal{C}$
Output: Recalibrated model $R \circ H : \mathcal{X} \to \Delta_K$.

1. Create a recalibrator training set:
   $\mathcal{S} = \{(p_{H(x)}, y) \mid x, y \in \mathcal{C}\}$

2. Fit the recalibration model R on $\mathcal{S}$:
   $\min_R \sum_{(p,y) \in \mathcal{S}} L_{\log}(R(p), y)$

3.4. Practical Considerations

Diagnostic Tools We propose assessing calibration in a regression setting using the check score ρ_τ aggregated over q quantiles $\{\tau_i\}_{j=1}^q$ (typically uniform in $[0, 1]$) and a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ as $\frac{1}{n} \sum_{i=1}^n \frac{1}{q} \sum_{j=1}^q \rho_{\tau_j}(F_{x_i}, y_i)$, where F_{x_i} is the forecast at x_i. Each ρ_τ is a consistent estimator for τ-th quantile—small ρ_τ indicate that the target y falls below the τ-th quantile a fraction τ of the time.

We recommend assessing the overall performance of F via a proper loss—this measures both calibration and sharpness and oftentimes corresponds to standard notions of accuracy (e.g., the log-loss often reduces to the mean squared error). Sharpness can be assessed heuristically via the variance of F, although mathematically, this normally does not correspond to a component of a proper loss.

Calibrating Bayesian Models When the model H is Bayesian or consists of an ensemble, the forecasts F may not have a closed form expression—instead, we may only sample from F. In such cases, we may compute features ϕ on samples from F—e.g. by computing empirical quantiles.

Non-Parametric Recalibrators Algorithms 1 represents a general framework that admits a range of recalibration models and objectives. One possible additional class of model families for R are non-parametric kernel density estimators over a low-dimensional space induced by features $\phi(F)$. Such non-parametric models can be combined with

parameterized neural networks via architectures such as mixture density networks (MDN; Bishop (1994)).

4. Theoretical Analysis

Next, we formally establish the correctness of our procedure. Our main claim is that constructing a distribution calibrated model is equivalent to estimating the density $\mathbb{P}(Y = y \mid F_X = F)$. When F is represented via low-dimensional features ϕ, this problem is tractable—hence calibration may be easier to enforce than previously thought.

Notation We start with some notation. We have a calibration dataset $\mathcal{C}$ of size m sampled from $\mathbb{P}$ and train a recalibrator $R : \Delta_{\mathcal{Y}} \to \Delta_{\mathcal{Y}}$ over the outputs of a base model H to minimize a proper loss L. We denote the Bayes-optimal recalibrator by $B := \mathbb{P}(Y = y \mid H(X))$; the distribution of Y conditioned on the forecast $(R \circ H)(X)$ is $Q := \mathbb{P}(Y = y \mid (R \circ H)(X))$. We are interested in expectations of various losses over X, Y; to simplify notation, we omit the variable X, e.g. $\mathbb{E}[L(R \circ H, Y)] = \mathbb{E}[L(R(H(X)), Y)]$.

4.1. Distribution Calibration

Our first claim is that distribution calibration reduces to a density estimation problem. We first formally define the task of density estimation.

Task 4.1 (Density Estimation). *The model R approximates the distribution $B := \mathbb{P}(Y = y \mid H(X))$ if the proper loss of R tends to that of B as $m \to \infty$ and w.h.p. for all m:*

$$\mathbb{E}[L(B \circ H, Y)] \leq \mathbb{E}[L(R \circ H, Y)] < \mathbb{E}[L(B \circ H, Y)] + \delta_m$$

where $\delta_m > 0$, $\delta_m = o(m)$ is a bound decreasing with m.

When the features $\phi(F)$ are low-dimensional (e.g., F is Gaussian), this task can be solved efficiently. We show via experiments that parametric neural network density estimators are effective at Task 4.1, and their performance can be verified on a held-out dataset. Alternatively, non-parametric kernel density estimation is formally guaranteed to estimate this density given enough data (Wasserman, 2006).

Our next proposition states that minimizing the recalibration objective defined in Algorithm 1 yields a model R that performs density estimation (Task 4.1).

Proposition 4.2. *Suppose that R minimizes the objective in Step 2 of Algorithm 1. Then R estimates the density (4) and performs Task 4.1.*

Proof (Sketch). Algorithm 1 outputs a recalibrator R that minimizes the objective $\sum_{x,y \in \mathcal{C}} L(R(F_x), y)$ over a cali-

bration dataset $\mathcal{C}$ sampled i.i.d. from $\mathbb{P}$. Observe that

$$\sum_{x,y \in \mathcal{C}} L(R(F_x), y) \approx \mathbb{E}_{F_X \sim \mathbb{P}} \mathbb{E}_{Y \sim \mathbb{P}(Y|F_X)} [L(R(F_X), Y)]$$

$$= \mathbb{E}_{F_X \sim \mathbb{P}} [L(R(F_X), \mathbb{P}(Y|F_X))]. \quad (6)$$

where $F_X = H(X)$. The first line follows by Monte Carlo and the second line follows from the definition of a proper loss. It follows from empirical risk minimization and the properties of Monte Carlo estimation that minimizing the empirical loss $\sum_{x,y \in \mathcal{C}} L(R(F_x), y)$ yields a minimizer of (6), which is $\mathbb{P}(Y|F_X))$ since L is a proper loss. $\square$

Next, we prove that using an R that estimates density (4) (Task 4.1) yields models that are asymptotically calibrated, i.e. their calibration error tends to zero as $m \to \infty$.

Proposition 4.3. *Let R be a model solving Task 4.1. Then $R \circ H$ is asymptotically calibrated and there exists $\delta_m = o(m)$ such that $\mathbb{E}[L_c(R \circ H, Q)] < \delta_m$ for all m w.h.p.*

Proof. Recall that the loss $\mathbb{E}[L(R \circ H, Y)]$ decomposes into a sum of calibration and refinement terms $\mathbb{E}[L_c(R \circ H, Q)] + \mathbb{E}[L_r(Q)]$ where $Q(y) := \mathbb{P}(Y = y \mid (R \circ H)(X))$.

As shown by Kull & Flach (2015), refinement further decomposes into a group loss and an irreducible term: $\mathbb{E}[L_r(Q)] = \mathbb{E}[L_g(Q, B \circ H)] + \mathbb{E}[L(B \circ H, Y)]$, where $B(Y = y \mid H(X))$ is the Bayes-optimal recalibrator. The form of the group loss L_g is the same as that of L_c. We may then write:

$$\underbrace{\mathbb{E}[L(B \circ H, Y)]}_{\text{Bayes-Optimal Loss}}$$

$$\leq \underbrace{\mathbb{E}[L_c(R \circ H, Q)]}_{\text{Calibration Loss}} + \underbrace{\mathbb{E}[L_g(Q, B \circ H)]}_{\text{Group Loss}} + \underbrace{\mathbb{E}[L(B \circ H, Y)]}_{\text{Bayes-Opt Loss}}$$

$$= \underbrace{\mathbb{E}[L(R \circ H, Y)]}_{\text{Proper Loss}} < \underbrace{\mathbb{E}[L(B \circ H, Y)]}_{\text{Bayes-Optimal Loss}} + \delta$$

where $\delta_m > 0, \delta_m = o(m)$. In the first equality we used the decomposition of Kull & Flach (2015) and in the last inequality we used Assumption 4.1. It follows that $\mathbb{E}[L_c(R \circ H, Q)] < \delta_m$, i.e. the calibration loss is small. $\square$

Practical Implications. Assumption 4.1 suggests that we want to use a model family that can minimize the expected risk $\mathbb{E}[L(H(X), Y)]$ well. Thus, in practice we want to select highly flexible algorithms for which we can control overfitting and underfitting. This motivates our earlier advice of using density estimation algorithms—which have provable guarantees—and neural networks—which are expressive and feature effective regularization techniques.

4.2. Vanishing Regret

Our second result shows that our approach yields calibration without degrading the performance of the baseline model. We define performance using a proper loss L; for example, if L is the log-likelihood we cover most standard performance metrics such as the L2 or the cross-entropy losses.

Proposition 4.4. *The recalibrated model has asymptotically vanishing regret relative to the base model: $\mathbb{E}[L(R \circ H, Y)] \leq \mathbb{E}[L(H, Y)] + \delta_m$, where $\delta_m > 0, \delta = o(m)$ is a bound that decreases with m.*

Proof. If we solve Task 4.1, then $\mathbb{E}[L(R \circ H, Y)] \leq \mathbb{E}[L(B \circ H, Y)] + \delta_m \leq \mathbb{E}[L(H, Y)] + \delta_m$, where the second inequality holds because the Bayes-optimal recalibrator B achieves an equal or lower loss than an identity mapping. $\square$

Thus, given enough data, we are guaranteed to produce calibrated forecasts and preserve base model performance (as measured by L). Note that there is no guarantee that sharpness will not worsen (it likely will)—our claims only pertain to the proper loss.

Finite-Sample Bounds. Note that our analysis provides *finite-sample* and not only asymptotic bounds on the regret and calibration error—the bounds are stated in terms of δ, which is $o(m)$. The bound δ on the calibration error directly depends on the finite-sample bound on the generalization error of the algorithm used as the recalibrator.

5. On Calibration in Machine Learning

In this section, we discuss the role of calibration in machine learning. We argue that calibration may be simpler to obtain than previously thought and enforcing it in predictive models unlocks benefits across downstream applications.

5.1. Are Models Calibrated Out-of-the-Box?

Machine learning models typically make uncalibrated probabilistic predictions (Niculescu-Mizil & Caruana, 2005; Guo et al., 2017). One reason for this is the limited expressivity of the model H—we cannot assign the correct probability to every level curve of the distribution. Another reason is the use of computational approximations (e.g., variational inference) to compute intractable predictive uncertainties.

While most models H are typically trained with a proper loss (the log-likelihood), this rarely yields calibration. The proper loss equals a sum of calibration and sharpness terms—not being able to perfectly optimize the loss, the model strikes a balance between suboptimal calibration and sharpness. Our method instead estimates densities in low dimensions (in the space of outputs of H), which tractably

approaches a Bayes-optimal loss and yields predictions with low calibration error.

5.2. On the Simplicity of Ensuring Calibration

Machine learning models are normally trained to minimize expected risk; we argue that in addition they should be calibrated. Calibration is achievable using the following procedure: (1) train a base model H on the main dataset $\mathcal{D}$; (2) train a recalibrator R using Algorithms 2 or 3.

Our results show that calibration can be provably achieved if we can perform density estimation in low dimensions (Task 4.1). Furthermore, the recalibrated forecasts do not degrade performance, as measured by a proper loss. In that sense, incorporating calibrated forecasting in machine learning systems may be easier than previously thought.

Our methods can be seen as an implementation of the principle of Gneiting et al. (2007), who argue that predictive uncertainties should be maximally sharp subject to being calibrated. While this approach is used in statistics for *evaluating* probabilistic models, our methods provides a way of enforcing this principle.

5.3. On the Importance of Calibrated Predictinos

Probabilistic models are key building blocks of machine learning systems in many areas—medicine, robotics, industrial automation, and more. We argue that maintaining calibration in predictive models will unlock benefits in downstream applications across may of these domains.

Safety and Interpretability. Good predictive uncertainties are important for model interpretability: in user-facing applications, humans make decisions based on model outputs and need to assess the confidence of the model, for example when interpreting an automated medical diagnosis. Calibration is also important for model safety: in areas such as robotics, we want to minimize the probability of adverse outcomes (e.g., a crash), and estimating their probability is a key step towards averting them (Berkenkamp et al., 2017).

Model-Based Planning. More generally, good predictive uncertainties also improve downstream decision-making applications such as model-based planning (Malik et al., 2019), a setting in which agents learn a model of the world to plan future decisions (Deisenroth & Rasmussen, 2011). Planning with a probabilistic model improves performance and sample complexity, especially when representing the model using a deep neural network. and improves the cumulative reward and the sample complexity of model-based agents (Rajeswaran et al., 2016; Chua et al., 2018).

Efficient Exploration. Balancing exploration and exploitation is a central challenge in reinforcement learning,

Bayesian optimization, and active learning. When probabilistic models are uncalibrated, inaccurate confidence intervals might incentivize the model to explore ineffective actions, degrading performance. Calibrated uncertainties have been shown to improve decision-making in bandits (Malik et al., 2019), Bayesian optimization (Deshpande & Kuleshov, 2021), and are likely to extend to active learning.

6. Experiments

6.1. Setup

Datasets. We use a number of UCI regression datasets varying in size from 194 to 8192 training instances; each training input may have between 6 and 159 continuous features. We randomly use 25% of each dataset for testing, and use the rest for training. We also perform classification on the following standard datasets: MNIST, SVHN, CIFAR10.

Models. Our first model is Bayesian Ridge Regression (MacKay, 1992). It uses a spherical Gaussian prior over the weights and a Gamma prior over the precision parameter. Posterior inference is performed in closed form as the prior is conjugate.

We also test a number of deep neural networks. We use variational dropout (Gal & Ghahramani, 2016) to produce probabilistic predictions. In our UCI experiments, we use fully-connected feedforward neural networks with two layers of 128 hidden units with a dropout rate of 0.5 and parametric ReLU non-linearities. We use convolutional neural networks (CNNs) on the image classification tasks. These are formed by fine-tuning a ResNet50 architecture on the training split for each dataset.

We additionally compare against a popular uncertainty estimation method recently developed specifically for deep learning models: deep ensembles (Lakshminarayanan et al., 2017). Deep ensembles average the predictive distributions of multiple models; we ensembled 5 neural networks, each having the same architecture as our standard model.

Our recalibrator R was also a densely connected neural network with two fully connected hidden layers of 20 units each and parametric ReLU non-linearities. We added dense skip connections between the layers. We held out 15% of the training set (up to max of 500 datapoints) for recalibration. On large datasets 500 points, is a relatively small quantity. When data is scarce, we can fit an ensemble of models calibrated on leave-one-out folds.

In regression experiments, we featurized input distributions F using nine quantiles $[0.1, ..., 0.9]$. We trained R using the quantile regression objective of Algorithm 2; we concatenated the quantile parameter $\tau \in [0, 1]$ to the featurization of F. In classification experiments, the inputs and the

	Bayesian Linear Regression Uncalibrated		Kuleshov et al.		Song et al.		Ours	
dataset	MAE	CHK	MAE	CHK	MAE	CHK	MAE	CHK
mpg	2.45 ± .00	0.92 ± .00	2.46 ± .00	0.91 ± .00	2.49 ± .01	**0.91 ± .00**	2.39 ± .03	**0.90 ± .01**
boston	3.45 ± .00	1.39 ± .00	3.39 ± .00	1.36 ± .00	3.38 ± .02	1.37 ± .01	3.34 ± .03	**1.31 ± .02**
yacht	6.17 ± .00	2.43 ± .00	5.96 ± .00	2.37 ± .00	1.50 ± .01	1.27 ± .01	0.90 ± .01	**0.35 ± .01**
wine	0.62 ± .00	0.24 ± .00	0.62 ± .00	0.23 ± .00	0.63 ± .00	**0.24 ± .00**	0.62 ± .01	0.24 ± .01
crime	0.52 ± .00	0.20 ± .00	0.51 ± .00	0.20 ± .00	0.52 ± .00	0.20 ± .00	0.51 ± .01	**0.19 ± .00**
auto	0.64 ± .00	0.25 ± .00	0.62 ± .00	0.25 ± .00	0.63 ± .00	**0.25 ± .00**	0.63 ± .01	0.25 ± .01
cpu	39.9 ± 0.00	15.8 ± 0.00	39.0 ± 0.00	15.4 ± 0.00	35.1 ± 0.24	**13.9 ± 0.05**	28.1 ± 0.58	**13.6 ± 0.45**
bank	39.5 ± 0.00	17.4 ± 0.00	39.1 ± 0.00	16.6 ± 0.00	34.1 ± 0.19	15.4 ± 0.04	29.3 ± 0.61	**14.2 ± 0.42**

	Bayesian Neural Network Uncalibrated		Kuleshov et al.		Song et al.		Ours	
dataset	MAE	CHK	MAE	CHK	MAE	CHK	MAE	CHK
mpg	2.73 ± .11	1.19 ± .07	2.97 ± .13	1.17 ± .07	2.67 ± .15	**1.10 ± .08**	2.60 ± .18	**1.08 ± .08**
boston	2.97 ± .15	1.23 ± .07	3.00 ± .16	1.20 ± .08	3.30 ± .18	**1.18 ± .09**	2.98 ± .17	**1.19 ± .10**
yacht	3.54 ± .19	1.53 ± .06	3.77 ± .22	1.51 ± .07	3.37 ± .25	1.51 ± .07	3.17 ± .21	**1.41 ± .07**
wine	0.62 ± .03	0.25 ± .01	0.63 ± .03	0.24 ± .02	0.62 ± .04	**0.23 ± .01**	0.62 ± .04	0.23 ± .02
crime	0.50 ± .01	0.19 ± .00	0.48 ± .01	0.19 ± .01	0.48 ± .01	0.19 ± .01	0.49 ± .01	**0.18 ± .00**
auto	0.62 ± .02	0.25 ± .01	0.62 ± .02	0.24 ± .01	0.64 ± .02	0.24 ± .01	0.64 ± .02	**0.23 ± .00**
cpu	74.0 ± 1.85	35.5 ± .92	71.0 ± 1.83	28.6 ± .93	68.4 ± 1.85	**31.8 ± 1.22**	66.4 ± 1.91	**31.3 ± 1.10**
bank	96.0 ± 1.89	46.0 ± .94	90.8 ± 1.87	44.1 ± .95	87.0 ± 1.88	41.1 ± 1.18	85.0 ± 1.90	**39.2 ± 1.09**

	Deep Ensemble Uncalibrated		Kuleshov et al.		Song et al.		Ours	
dataset	MAE	CHK	MAE	CHK	MAE	CHK	MAE	CHK
mpg	7.66 ± .32	3.55 ± .12	11.8 ± .41	5.20 ± .21	9.57 ± .39	3.93 ± .23	9.01 ± .40	**3.53 ± .18**
boston	8.42 ± .31	3.82 ± .18	12.8 ± .52	5.49 ± .27	8.13 ± .44	3.91 ± .21	8.00 ± .46	**3.62 ± .20**
yacht	9.40 ± .38	4.60 ± .11	10.8 ± .49	4.93 ± .23	9.45 ± .42	4.41 ± .15	9.36 ± .51	**4.27 ± .15**
wine	0.71 ± .02	0.30 ± .01	0.70 ± .03	0.27 ± .01	0.69 ± .03	**0.26 ± .01**	0.69 ± .03	0.26 ± .01
crime	0.68 ± .02	0.25 ± .01	0.77 ± .03	0.34 ± .02	0.68 ± .02	**0.25 ± .01**	0.68 ± .03	0.25 ± .01
auto	0.86 ± .02	0.30 ± .01	0.86 ± .03	0.31 ± .01	0.87 ± .03	0.35 ± .02	0.86 ± .04	**0.30 ± .02**
cpu	58.1 ± 1.77	20.0 ± 1.01	57.3 ± 1.82	18.3 ± 1.00	57.9 ± 1.81	**18.3 ± 0.94**	55.4 ± 1.89	**17.4 ± 1.08**
bank	59.0 ± 1.81	21.2 ± 0.99	58.7 ± 1.85	21.0 ± 0.98	57.4 ± 1.87	**20.0 ± 0.96**	55.8 ± 1.92	**19.1 ± 1.02**

Table 2. Calibration and accuracy on UCI regression datasets. We evaluate Bayesian linear regression, Bayesian neural networks, and deep ensembles using the mean average error (MAE) and check score (CHK), comparing against Kuleshov et al. (2018) and Song et al. (2019). These metrics precisely capture the two properties studied in our paper: the ability of our methods to ensure good calibration while having a very small regret relative to the baseline model as measured by a supervised learning loss.

ouputs of R are class probabilities, and R is trained using log-likelihood maximization defined in Algorithm 3. All other architectural details are unchanged.

We did not observe significant overfitting in our experiments. We believe overfitting is mitigated by the fact that we perform quantile regression and thus learn a complex distribution function that is not easy to overfit.

6.2. Regression Experiments on UCI Data

We report the results of our regression experiments on the UCI datasets in Table 2. As in Song et al. (2019), we evaluate the quality of forecasts using a check score $\rho_\tau(f, y) = \tau(y - f)$ if $y \geq f$ and $(1 - \tau)(f - y)$ as in Song et al. (2019); we average it over nine quantile levels $\tau \in 0.1, ..., 0.9$. We measure regression performance using the mean average error. These metrics precisely capture the two properties studied in our paper: the ability of our methods to ensure good calibration while having a very small regret relative to the baseline model as measured by a supervised learning loss.

Our method improves over the accuracies and uncertainties of Kuleshov et al. (2018), and in many cases over those of Song et al. (2019) on Bayesian linear regression, Bayesian neural networks, and deep ensembles, without ever being worse. Note that also that out method is simpler and easier to implement than that of Song et al. (2019) (it does not require implementing variational inference), and applies to any input distribution, not just Gaussians.

6.3. Classification Experiments on MNIST, SVHN, CIFAR10

We report the results of the image classification experiments in Table 3. We measure performance using accuracy and calibration error of Kuleshov et al. (2018) on the test set. We report these metrics for baseline and calibrated versions of convolutional neural network classifier. We perform recalibration with a simple softmax regression (a multi-class generalization of Platt scaling) and with the neural network recalibrator. The best uncertainties are produced by our method, while accuracies are similar.

	MNIST	SVHN	CIFAR10
Base Model			
Accuracy	0.9952	0.9508	0.9179
Calibration	0.3166	0.5975	0.5848
Platt Scaling			
Accuracy	0.9952	0.9508	0.9181
Calibration	0.2212	0.3278	0.2233
Ours			
Accuracy	0.9951	0.9509	0.9163
Calibration	0.1030	0.2674	0.1091

Table 3. Performance on Image Classification

7. Related Work

Probabilistic Forecasting More modern discussions of probabilistic forecasting can be found in the literature on meteorology (Gneiting & Raftery, 2005). This influential work appears in methods weather forecasting applications systems (Raftery et al., 2005). Most previous work focuses on classification, but recent work (Gneiting et al., 2007; Kuleshov et al., 2018) extends to regression.

Probabilistic forecasting has been studied extensively in the statistics literature (Murphy, 1973; Dawid, 1984), mainly in the context of evaluation using proper scoring rules (Gneiting & Raftery, 2007). Proper scores measure calibration and sharpness in classification (Murphy, 1973) and regression (Hersbach, 2000).

Calibration Recalibration is a widely used approach for improving probabilistic forecasts. It has roots in the classification setting, starting with Platt scaling (Platt, 1999) and isotonic regression (Niculescu-Mizil & Caruana, 2005). These mehtods have been extended to multi-class (Zadrozny & Elkan, 2002), structured (Kuleshov & Liang, 2015), and online prediction (Kuleshov & Ermon, 2017b). There is significant recent interest in calibration in deep learning (Guo et al., 2017; Lakshminarayanan et al., 2017; Gal et al., 2017; Kuleshov et al., 2018). Beyond deep learning, accurate uncertainty estimation is important for the design of decision-making systems (Malik et al., 2019), crowdsourcing (Werling et al., 2015), data-efficient machine learning (Ratner et al., 2016; Kuleshov & Ermon, 2017a; Ren et al., 2018), machine translation (Kumar & Sarawagi, 2019), as well as other problems in natural language processing and beyond (Nguyen & O'Connor, 2015; Kuleshov et al., 2019).

Conformal prediction is a closely related line of work (Vovk et al., 2005; Shafer & Vovk, 2008). The recent methods by Romano et al. (2019) and Barber et al. (2021) provide similar guarantees, but only for a single confidence interval (i.e., two quantiles), not a full distribution. Romano et al. (2020) focuses on *classification*, while our work encom-

passes regression.

Compared to previous work, our methods target a different type of calibration (distribution vs. quantile) and estimate different distributions ($P(Y \leq F_X^{-1}(p))$ vs. $P(Y|H(X) = F)$) using different objectives (e.g., the quantile divergence vs. calibration error in Kuleshov et al. (2018)).

Distribution Calibration Unlike Song et al. (2019) our method can recalibrate any parametric distribution (not just Gaussians) while being also simpler. While Song et al. (2019) relies on variational inference in Gaussian processes (which is slow and complex to implement), our method uses a neural network that can be implemented in a few lines of code. Our method applies to both classification and regression and outperforms Song et al. (2019), Kuleshov et al. (2018), as well as Platt and temperature scaling.

Interestingly, the method of Song et al. (2019) is a special case of Algorithm 1 when $\phi(F)$ consists of Gaussian natural parameters, R is a Gaussian process, its outputs are parameters for the Beta link function, and L is the log-likelihood. However, the resulting problem requires variational inference; our framework instead admits simple solutions based on gradient descent.

8. Conclusion

Accurate predictive uncertainties are fully characterized by two properties—calibration and sharpness. We argue that predictive uncertainties should maximize sharpness subject to being calibrated (Gneiting et al., 2007) and propose a recalibration technique that achieves this goal. Out technique guarantees distribution calibration for a wide range of base models and is easy to implement in a few lines of code. It applies to both classification and regression and is formally guaranteed to produce asymptotically distributionally calibrated forecasts while minimizing regret. Finally, our analysis formalizes the well-known paradigm of Gneiting et al. (2007) and provides a simple method that provably achieves it. This lends strong support for this principle and influences how one should reason about uncertainty in machine learning. We believe that an important takeaway of our work is that calibration should be leveraged more broadly throughout machine learning.

References

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486–507, 2021.

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

Bishop, C. M. Mixture density networks. 1994.

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

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

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

Deshpande, S. and Kuleshov, V. Calibration improves bayesian optimization. *arXiv preprint arXiv:2112.04620*, 2021.

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

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3584–3593, 2017.

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

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

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

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

Hersbach, H. Decomposition of the continuous ranked probability score for ensemble prediction systems. *Weather and Forecasting*, 15(5):559–570, 2000.

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

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

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

Kuleshov, V., Fenner, N., and Ermon, S. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pp. 2796–2804. PMLR, 2018.

Kuleshov, V., Ding, J., Vo, C., Hancock, B., Ratner, A., Li, Y., Ré, C., Batzoglou, S., and Snyder, M. A machine-compiled database of genome-wide association studies. *Nature communications*, 10(1):1–8, 2019.

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

Kumar, A. and Sarawagi, S. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*, 2019.

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

MacKay, D. J. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Malik, A., Kuleshov, V., Song, J., Nemer, D., Seymour, H., and Ermon, S. Calibrated model-based deep reinforcement learning. In *International Conference on Machine Learning*, pp. 4314–4323. PMLR, 2019.

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

Nguyen, K. and O'Connor, B. Posterior calibration and exploratory analysis for natural language processing models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1587–1598, 2015.

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

Platt, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

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

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

Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29: 3567–3575, 2016.

Ren, H., Stewart, R., Song, J., Kuleshov, V., and Ermon, S. Learning with weak supervision from physics and data-driven constraints. *AI Magazine*, 39(1):27–38, 2018.

Romano, Y., Patterson, E., and Candes, E. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

Romano, Y., Sesia, M., and Candes, E. Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, 33:3581–3591, 2020.

Shafer, G. and Vovk, V. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

Si, P., Bishop, A., and Kuleshov, V. Autoregressive quantile flows for predictive uncertainty estimation, 2021.

Song, H., Diethe, T., Kull, M., and Flach, P. Distribution calibration for regression. In *International Conference on Machine Learning*, pp. 5897–5906. PMLR, 2019.

Vovk, V., Gammerman, A., and Shafer, G. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

Wasserman, L. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

Werling, K., Chaganty, A. T., Liang, P. S., and Manning, C. D. On-the-job learning with bayesian decision theory. *Neural Information Processing Systems*, 2015.

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