Non-Parametric Calibration for Classification

Jonathan Wenger  
University of Tuebingen  
TU Munich  
KTH Royal Institute of Technology  
jonathan.wenger@uni-tuebingen.de

Hedvig Kjellström  
KTH Royal Institute of Technology  
hedvig@kth.se

Rudolph Triebel  
German Aerospace Center (DLR)  
TU Munich  
rudolph.triebel@dlr.de

Abstract

Many applications of classification methods not only require high accuracy but also reliable estimation of predictive uncertainty. However, while many current classification frameworks, in particular deep neural networks, achieve high accuracy, they tend to incorrectly estimate uncertainty. In this paper, we propose a method that adjusts the confidence estimates of a general classifier such that they approach the probability of classifying correctly. In contrast to existing approaches, our calibration method employs a non-parametric representation using a latent Gaussian process, and is specifically designed for multi-class classification. It can be applied to any classifier that outputs confidence estimates and is not limited to neural networks. We also provide a theoretical analysis regarding the over- and underconfidence of a classifier and its relationship to calibration, as well as an empirical outlook for calibrated active learning. In experiments we show the universally strong performance of our method across different classifiers and benchmark data sets, in particular for state-of-the-art neural network architectures.

1 INTRODUCTION

With the recent achievements in machine learning, in particular in the area of deep learning, the application range for learning methods has increased significantly. Especially in challenging fields such as computer vision or speech recognition, important advancements have been made using powerful and complex network architectures, trained on very large data sets. Most of these techniques are used for classification tasks, e.g. object recognition. We also consider classification in our work. However, in addition to achieving high classification accuracy, our goal is to provide reliable prediction uncertainty estimates. This is particularly relevant in safety-critical applications, such as autonomous driving and robotics (Amodei et al., 2016). Reliable uncertainties can be used to increase a classifier’s precision by reporting only class labels that are predicted with low uncertainty or for information theoretic analyses of what was learned and what was not. The latter is especially interesting in active learning, where the model actively selects the most relevant data samples for training via a query function based on the predictive uncertainty of the model (Settles, 2010).

Unfortunately, current probabilistic classification approaches that inherently provide good uncertainty estimates, such as Gaussian processes (GP), often suffer from lower accuracy and higher computational complexity on high-dimensional classification tasks compared to state-of-the-art convolutional neural networks (CNN). It was recently observed that many modern CNNs are overconfident (Lakshminarayanan et al., 2017, Hein et al., 2019) and miscalibrated (Guo et al., 2017). Calibration refers to how well confidence estimates of a classifier match the probability of the associated prediction being correct. Originally developed in the context of forecasting (Murphy, 1973, DeGroot and Fienberg, 1983), uncertainty calibration has seen an increased interest in recent years (Naeini et al., 2015, Guo et al., 2017, Vaicenavicius et al., 2019), partly because of the popularity of CNNs which generally lack an inherent uncertainty representation. Earlier studies show that also classical methods such as decision trees, boosting, SVMs and naive Bayes classifiers tend to be miscalibrated (Zadrozny and Elkan, 2001, Niculescu-Mizil and Caruana, 2005a,b, Naeini et al., 2015). Based on these observations, we claim that training and calibrating a classifier can be two different objectives that benefit...
from being considered separately, as shown in a toy example in Figure 1. Here, a simple neural network continually improves its accuracy on the test set during training, but eventually overfits in terms of NLL and calibration error. A similar phenomenon was observed by Guo et al. (2017) for more complex models.

Calibration methods approach this problem by performing a post-hoc improvement to uncertainty estimation using a small subset of the training data. Our goal in this paper is to develop a multi-class calibration method for arbitrary classifiers, to provide reliable predictive uncertainty estimates in addition to maintaining high accuracy. In contrast to recent approaches, which strive to improve uncertainty estimation only for neural networks, including Bayesian neural networks (MacKay 1992, Gal 2016) and Laplace approximations (LA) (Martens and Grosse 2015, Ba et al. 2017), our aim is a framework that is not based on tuning a specific classification method. This has the advantage that our method operates independently of the training process.

**Contribution** In this work we develop a new multi-class and model-agnostic approach to calibration, based on a latent Gaussian process inferred using variational inference. We replicate and extend previous findings that popular classification models are generally not calibrated and demonstrate the superior performance of our method for deep neural networks. Finally, we study the relationship between active learning and calibration from a theoretical perspective and give an empirical outlook.

**Related Work** Estimation of uncertainty, in particular in deep learning (Kendall and Gal 2017), is of considerable interest in the machine learning community. There are two main approaches in classification. The first chooses a model and a (regularized) loss function for a particular problem to inherently learn a good representation, and the second performs post-hoc calibration by transforming the output of the underlying model. For example, Pereyra et al. (2017) propose to penalize low-entropy output distributions, Kumar et al. (2018) suggest a trainable measure of calibration as a regularizer and Maddox et al. (2019) employ an approximate Bayesian inference technique using stochastic weight averaging. Milios et al. (2018) approximate Gaussian process classifiers by GP regression on transformed labels for better scalability and Wilson et al. (2016) combine additive Gaussian processes with deep neural network architectures. Research on calibration goes back to statistical forecasting (Murphy 1973, DeGroot and Fienberg 1983) and approaches to provide uncertainty estimates for non-probabilistic binary classifiers (Platt 1999, Lin et al. 2007, Zadrozny and Elkan 2002). More recently, Bayesian binning into quantiles (Naeini et al. 2015) and beta calibration (Kull et al. 2017a) for binary classification and temperature scaling (Guo et al. 2017) for multi-class problems were proposed. A theoretical framework for evaluating calibration in classification was suggested by Vaičenavicius et al. (2019). Calibration was also previously considered in the online setting with potentially adversarial input (Kuleshov and Ermon 2017). Calibration in a broader sense is also of interest outside of the classification setting, e.g. in regression (Kuleshov et al. 2018, Song et al. 2019), in the discovery of causal Bayesian network structure from observational data (Jabbari et al. 2017) and in the algorithmic fairness literature (Pleiss et al. 2017, Kleinberg 2018).

2 **BACKGROUND**

**Notation** Consider a data set \( D = \{(x_n, y_n)\}_{n=1}^N \) assumed to consist of independent and identically distributed realizations of the random variable \((x, y) \in \mathcal{X} \times \mathcal{Y} \) with \( K := |\mathcal{Y}| \) classes. If not stated otherwise,
any expectation is taken with respect to the law of $(x, y)$. Let $f : \mathcal{X} \rightarrow \mathbb{R}^K$ be a classifier with output $z = f(x)$, prediction $\hat{y} = \arg\max_{c} (z_c)$ and associated confidence score $\hat{z} = \max_{c} (z_c)$. Lastly, $v : \mathbb{R}^K \rightarrow \mathbb{R}^K$ denotes a calibration method.

2.1 Calibration

A classifier is called calibrated if the confidence in its class prediction matches the probability of its prediction being correct, i.e. $\mathbb{E}[1_{\hat{y} = y} | \hat{z}] = \hat{z}$. In order to measure calibration, we define the expected calibration error following Naeini et al. (2015) for $1 \leq p < \infty$ by

$$ECE_p = \mathbb{E}[|\hat{z} - \mathbb{E}[1_{\hat{y} = y} | \hat{z}]|^p]^{\frac{1}{p}} \tag{1}$$

and the maximum calibration error by $ECE_\infty = \max_{z \in [0,1]} |\hat{z} - \mathbb{E}[1_{\hat{y} = y} | \hat{z}]|$. In practice, we estimate the calibration error using a fixed binning for $\hat{z}$ as described by Naeini et al. (2015). However, calibration alone is not sufficient for useful uncertainty estimates. A classifier on a balanced binary classification problem that always returns a confidence of 0.5 is perfectly calibrated, because this equals the probability of making a correct prediction. However, intuitively prediction confidence should be sufficiently close to 0 and 1 to be informative. This notion is known as sharpness or refinement (DeGroot and Fienberg 1983, Murphy and Winker 1992, Cohen and Goldszmidt 2004).

2.2 Over- and Underconfidence

We build on the notions of over- and underconfidence, as introduced previously in the context of active learning by Mund et al. (2015). The idea is to measure the average confidence of a classifier on its false predictions and the average uncertainty on its correct predictions:

$$o(f) = \mathbb{E}[1_{\hat{y} \neq y}] \quad u(f) = \mathbb{E}[1 - \hat{z} | \hat{y} = y] \tag{2}$$

Over- and underconfidence are properties of a classifier independent of accuracy. They relate to query efficiency in active learning (Settles 2010). Counter to intuition, both can be present to varying degrees simultaneously. We refer to Section S1.1 of the supplementary material for more details. We demonstrate that there is a direct link between calibration and these two notions.

**Theorem 1** (Calibration, Over- and Underconfidence) Let $1 \leq p < q < \infty$, then the following relationship between overconfidence, underconfidence and the expected calibration error holds:

$$|o(f)\mathbb{P}(\hat{y} \neq y) - u(f)\mathbb{P}(\hat{y} = y)| \leq ECE_p \leq ECE_q.$$

A proof is given in Section S1.2 of the supplementary material. We see that the expected calibration error bounds the weighted absolute difference of over- and underconfidence. This implies that for perfect calibration, the odds of making a correct prediction equal the ratio between over- and underconfidence. Comparable statements exist in algorithmic fairness, where over- and underconfidence were termed generalized false positive and negative rates (Pleiss et al. 2017, Kleinberg 2018).

2.3 Calibration Methods

The aim of calibrating a classifier is to transform its output to be closer to the true correctness probability. This is typically done by fitting a calibration method $v$ on a small hold-out set called the calibration data (see Figure 2). In the following, we describe the most prevalent methods.

2.3.1 Binary Calibration

**Platt Scaling** (Platt 1999, Lin et al. 2007) was originally introduced to provide probabilistic output for SVMs. It is a parametric calibration method, where a logistic regressor is fit to the confidence scores of the positive class such that

$$v(z)_2 = (1 + \exp(-az_2 - b))^{-1}$$

for $a, b \in \mathbb{R}$. This parametric assumption is justified if the scores of each class are normally distributed with identical variance (Kull et al. 2017b).

**Isotonic Regression** (Zadrozny and Elkan 2002) is a non-parametric approach. It assumes a non-decreasing relation between the model confidence $z_2$ of the positive class and its correctness probability $v(z)_2$. A piecewise-constant isotonic function $m$ is found by

![Figure 2: Schematic diagram of calibration. A fraction of the training data is split off and the remaining data is used for training. The split-off calibration data is classified by the trained model and subsequently used to fit the calibration method (blue). Confidence estimates from the classifier for new data are then adjusted by the calibration method (orange).](image-url)
minimizing a squared loss function, resulting in the calibration function \( v(z)_2 = m(z_2) + \epsilon \).

**Beta Calibration** ([Kull et al. 2017a](#)) was designed for probabilistic classifiers with output range \( z_2 \in [0, 1] \). A family of calibration maps is defined based on the likelihood ratio between two Beta distributions. The calibration map is given by

\[
v(z)_2 = \frac{1 + \exp(-c)(1 - z_2)^b z_2^{-a}}{1 + \exp(-c)(1 - z_2)^b z_2^{-a} - 1},
\]

where \( a, b, c \in \mathbb{R} \) are fit on the calibration data.

**Bayesian Binning into Quantiles (BBQ)** ([Naeini et al. 2015](#)) scores multiple equal-frequency binning models and uses a score weighted average of the accuracy in each bin as a calibration map. A binning model \( M \) is weighted by \( \mathbb{P}(M)\mathbb{P}(D \mid M) \), where \( \mathbb{P}(M) \) is uniform and the marginal likelihood \( \mathbb{P}(D \mid M) \) can be computed in closed form given parametric assumptions on data generation.

### 2.3.2 Multi-class Calibration

**One-vs-All** In order to extend binary calibration methods to multi-class problems, [Zadrozny and Elkan (2002)](#) suggest a one-vs-all approach, training a binary classifier on each split and calibrating subsequently. As most modern classifiers are inherently multi-class, this approach is not feasible anymore. We instead use a one-vs-all approach for the output \( z \) of the multi-class classifier, train a calibration method on each split and average their predictions.

**Temperature Scaling** ([Guo et al. 2017](#)) was introduced as a multi-class extension to Platt scaling for neural networks. For an output logit vector \( z \) of a neural network and a *temperature* parameter \( T > 0 \), the calibrated confidence is defined as

\[
v(z) = \sigma \left( \frac{z}{T} \right) = \frac{\exp \left( \frac{z}{T} \right)}{\sum_{i=1}^{K} \exp \left( \frac{z_i}{T} \right)}.
\]

The parameter \( T \) is determined by optimizing the NLL. By construction, the accuracy of the classifier is unchanged after scaling. Variants of this method where the factor is replaced by an affine map were shown to be ineffective ([Guo et al. 2017](#)).

### 3 GAUSSIAN PROCESS CALIBRATION

In the following section, we will outline our non-parametric calibration approach. Our aim is to develop a calibration method, which is inherently multi-class, suitable for arbitrary classifiers, makes as few assumptions as possible on the shape of the calibration map and can take prior knowledge into account. These desired properties motivate the use of a latent GP.

**Definition** Assume a one-dimensional Gaussian process prior over the latent function \( g : \mathbb{R} \to \mathbb{R} \), i.e.

\[
g \sim \mathcal{GP}(\mu(\cdot), k(\cdot, \cdot \mid \theta))
\]

with mean function \( \mu \), kernel \( k \) and kernel parameters \( \theta \) ([Rasmussen and Williams 2005](#)). Further, let the calibrated output be given by the **softargmax** inverse link function applied to the latent process evaluated at the model output

\[
v(z)_k = \sigma(g(z_1), \ldots, g(z_K))_k = \frac{\exp(g(z_k))}{\sum_{j=1}^{K} \exp(g(z_j))}.
\]

Note the similarity to multi-class Gaussian process classification, which has \( K \) latent functions ([Rasmussen and Williams 2005](#)). In contrast, we consider one **shared latent function** applied to each component \( z_k \). We use the categorical likelihood

\[
\text{Cat}(y \mid v(z)) = \prod_{k=1}^{K} \sigma(g(z_1), \ldots, g(z_K))^{[y=k]}_k
\]

to obtain a prior on the class prediction. We make the prior assumption that the classifier is already calibrated. This corresponds to either \( \mu(z_k) = \ln(z_k) \) if the inputs are confidence estimates, or to \( \mu(z_k) = z_k \) if the inputs are logits. For specific models other choices may be beneficial, e.g. a linear prior. An example of a latent function for a synthetic data set is shown in Figure 3.

If the latent function \( g \) is monotonically increasing in its domain, the accuracy of the underlying classifier is unchanged after calibration.
Inference In order to infer the calibration map, we need to find the underlying GP based on the confidence scores or logits and labels in the calibration set. Given the likelihood \( [y, n] \), the posterior is not analytically tractable. We use variational inference to approximate the posterior \( \left( \text{Girolami and Rogers 2006} \right) \) Paul et al. (2012). For our method to scale to large data sets we only retain a sparse representation of the inputs, making inference computationally less intensive. We extend an approach by Hensman et al. (2015) to our choice of likelihood. The joint distribution of the data \((z_n, y_n)\) and latent variables \( g \) is given by

\[
p(y, g) = p(y \mid g)p(g) = \prod_{n=1}^{N} p(y_n \mid g_n)p(g)
\]

where \( y \in \{1, \ldots, K\}^N \), \( g = (g_1, g_2, \ldots, g_N)^\top \in \mathbb{R}^{NK} \) and \( g_n = (g(z_n, 1), \ldots, g(z_n, K)) \) \( \in \mathbb{R}^K \). The covariance matrix \( \Sigma_g \) has block-diagonal structure by independence of the calibration data. If performance is important, a further diagonal assumption can be made. Note that we drop the explicit dependence on \( z_n \) and \( \theta \) throughout to lighten the notation. We want to compute the posterior \( p(g \mid y) \). In order to reduce the computational complexity \( O((NK)^3) \), we define \( M \) inducing inputs \( w \in \mathbb{R}^M \) and inducing variables \( u \in \mathbb{R}^M \). The joint distribution is given by

\[
p(g, u) = \mathcal{N} \left( \begin{bmatrix} g \\ u \end{bmatrix} \mid \begin{bmatrix} \mu_g \\ \mu_u \end{bmatrix}, \begin{bmatrix} \Sigma_g & \Sigma_{g, u} \\ \Sigma_{g, u}^\top & \Sigma_u \end{bmatrix} \right) .
\]

The joint distribution factorsizes as \( p(y, g, u) = p(y \mid g)p(g \mid u)p(u) \). We aim to find a variational approximation \( q(u) = \mathcal{N}(u \mid m, S) \) to the posterior \( p(u \mid y) \). For general treatments on variational inference we refer to Blei et al. (2017), Zhang et al. (2018). We find the variational parameters \( m \) and \( S \), the locations of the inducing inputs \( w \) and the kernel parameters \( \theta \) by maximizing a lower bound to the marginal log-likelihood

\[
\ln p(y) \geq \text{ELBO}(q(u))
\]

\[
= \mathbb{E}_{q(u)} \left[ \ln p(y \mid u) \right] - \text{KL} \left[ q(u) \mid \mid p(u) \right]
\]

\[
\geq \mathbb{E}_{q(u)} \left[ \mathbb{E}_{q(g \mid u)} \left[ \ln p(y \mid g) \right] \right] - \text{KL} \left[ q(u) \mid \mid p(u) \right]
\]

\[
= \mathbb{E}_{q(g)} \left[ \ln p(y \mid g) \right] - \text{KL} \left[ q(u) \mid \mid p(u) \right] \quad (6)
\]

\[
= \sum_{n=1}^{N} \mathbb{E}_{q(g_n)} \left[ \ln p(y_n \mid g_n) \right] - \text{KL} \left[ q(u) \mid \mid p(u) \right]
\]

where \( q(g) := \int p(g \mid u)q(u) \, du \) is Gaussian and only its \( K \)-dimensional marginals \( q(g_n) = \mathcal{N}(g_n \mid \phi_n, C_n) \) are required to compute the expectation terms. To do so, we use a second order Taylor approximation for \( \ln p(y_n \mid g_n) \) and obtain

\[
\mathbb{E}_{q(g_n)} \left[ \ln p(y_n \mid g_n) \right] \approx \ln p(y_n \mid \phi_n)
\]

\[
+ \frac{1}{2} \left( \sigma(\phi_n)^\top C_n \sigma(\phi_n) - \text{diag}(C_n)^\top \sigma(\phi_n) \right)
\]

which can be computed in \( O(K^2) \). Computing the KL-divergence term is in \( O(M^3) \). Therefore, computing the objective \((6) \) has complexity \( O(NK^2 + M^3) \). Note that this can be remedied through parallelization as all \( N \) expectation terms can be computed independently. The optimization is performed via a gradient-based optimizer and automatic differentiation. We refer to Section S2 of the supplementary material for a more detailed treatment of inference.

Calibration Given the approximate posterior \( p(g, u \mid y) \approx p(g \mid u)q(u) \), calibrated predictions at new inputs \((z_1, \ldots, z_K^2) \) \( \in \mathbb{R}^{LN} \) are obtained via

\[
p(g, u \mid y) = \int p(g, u \mid g, u \mid y) \, dg \, du
\]

\[
\approx \int p(g, u \mid u)q(u) \, du
\]

which is Gaussian. Mean \( \mu_g \), and variance of a latent value \( g \in \mathbb{R}^K \) can be computed in \( O(KM^2) \). The class predictions \( y_* \) are then obtained by marginalization

\[
p(y_* \mid y) = \int p(y_* \mid g, p(g_* \mid y) \, dg,
\]

via Monte-Carlo integration. While inference and calibration have higher computational cost than in other methods, it is orders of magnitude less than the training cost of the classifier. Furthermore, calibration can be performed in parallel with training in the online setting. We can speed up calibration by approximating the predictive distribution via the GP mean, i.e. \( p(y_* \mid y) \approx p(y_* \mid \mu_g) = \sigma(\mu_g) \).

4 EXPERIMENTS

We experimentally evaluate our approach against the calibration methods presented in Section 2.3 applied to different classifiers on a range of binary and multiclass computer vision benchmark data sets. Besides CNNs, we are also interested in ensemble methods. All methods and experiments were implemented in Python 3.6. GPcalib was developed based on gpflow (Matthews et al. 2017). Any results reported used a sum kernel consisting of an RBF and a white noise kernel and a diagonal covariance matrix \( \Sigma_g \).

1 An implementation of GP calibration and code replicating the experiments is available at

https://github.com/jonathanwenger/pycalib
4.1 Calibration Results

We report the average ECE\textsubscript{1} estimated with 100 bins over 10 Monte-Carlo cross validation runs. We chose a larger number of bins than in previous works, as too few bins typically underestimate the ECE\textsubscript{1} (Kumar et al., 2019). See Section S3.1 of the supplementary material for details. We used the following data sets with indicated train, calibration and test splits:

- **KITTI** (Geiger et al., 2012; Narr et al., 2016): Stream-based urban traffic scenes with features (Himmelsbach et al., 2009) from segmented 3D point clouds. 8 or 2 classes, train: 16000, calibration: 1000, test: 9000.
- **PCam** (Veeling et al., 2018): Histopathologic scans of (metastatic) tissue from lymph node sections converted to grayscale. 2 classes, train: 22768, calibration: 1000, test: 9000.
- **CIFAR-100** (Krizhevsky, 2009): Image database of tiny color images from the web. 100 classes, train: 50000, calibration: 1000, test: 9000.

**Binary Classification** We trained two boosting variants, AdaBoost (Freund and Schapire, 1997; Hastie et al., 2009) and XGBoost (Chen and Guestrin, 2016), two forest variants, Mondrian Forests (Lakshminarayanan et al., 2014) and Random Forests (Breiman, 2001), and a one layer neural network on the binary KITTI and PCam data sets. We report the average ECE\textsubscript{1} in Table S3 in the supplementary material. For binary problems all calibration methods perform similarly with the exception of isotonic regression, which has particularly low calibration error on the KITTI data set. However, due to its piecewise constant calibration map the resulting confidence distribution has a set of singular peaks instead of a smooth distribution. While GPcalib is competitive across data sets and classifiers, it does not outperform isotonic regression. Hence, if exclusively binary problems are of interest a simple calibration method should be preferred. Interestingly, the 1-layer NN trained on KITTI is already well-calibrated, however all calibration methods except isotonic regression and GPcalib increase the ECE\textsubscript{1}.

**Multi-class Classification** Besides the aforementioned classification models, which were trained on MNIST, we also calibrated pre-trained CNN architectures\footnote{Pre-trained CNNs were obtained from https://github.com/bearpaw/pytorch-classification and https://github.com/Cadene/pretrained-models.pytorch} on CIFAR-100 and ImageNet. The following CNNs were used: AlexNet (Krizhevsky et al., 2012), VGG19 (Simonyan and Zisserman, 2014; Liu and Deng, 2015), ResNet-50, ResNet-152 (He et al., 2016), WideResNet (Zagoruyko and Komodakis, 2016), DenseNet-121, DenseNet-BC-190, DenseNet-201 (Huang et al., 2017), InceptionV4 (Szegedy et al., 2016), ResNeXt-29, SE-ResNeXt-50, SE-ResNeXt-101 (Xie et al., 2017; Hu et al., 2018), PolyNet (Zhang et al., 2017), SENet-154 (Hu et al., 2018), PNASNet-5-Large (Liu et al., 2018), NASNet-A-Large (Zoph et al., 2018). All binary calibration methods were extended to the multi-class setting in a one-vs-all manner. Temperature scaling and GPcalib were applied to logits for all CNNs and otherwise directly to probability scores. The average ECE\textsubscript{1} is shown in Table S1. While binary methods still perform reasonably well for 10 classes in the case of MNIST and CIFAR-100, they worsen calibration considerably in the case of 1000 classes on ImageNet. Moreover, they also skew the posterior distribution so much that accuracy is heavily affected, disqualifying them from use. Temperature scaling preserves the underlying accuracy of the classifier by definition. Even though GP calibration has no such guarantees, our experiments show little effect on accuracy (see Table S4 in the supplementary material). GP calibration performs comparably to other methods on MNIST except for the simple NN and AdaBoost. It does not improve upon calibration for the simple NN, but it is the only method able to handle the large ECE\textsubscript{1} of AdaBoost. GPcalib calibrates significantly better on CIFAR-100 than all other methods for all CNNs, except AlexNet. On ImageNet GPcalib demonstrates low ECE\textsubscript{1} within one to two standard deviations of temperature scaling on four CNNs, but outperforms all other calibration methods on the remaining nine evaluated architectures. In particular on higher accuracy CNNs (see Table S4), GPcalib calibrates better. For CNNs which already demonstrate low ECE\textsubscript{1}, such as InceptionV4 and SE-ResNeXt-50, most methods worsen calibration, whereas GPcalib does not. We attribute this desirable behavior, also seen in the binary case, to its prior assumption that the underlying classifier is already calibrated. The increased flexibility of the non-parametric latent map and its prior assumptions allow GPcalib to adapt to various classifiers and data sets.

**Latent Function Visualization** In order to illustrate the benefit of a non-linear latent function when calibrating, we show some latent functions from our experiments on ImageNet. We compare GPcalib with temperature scaling and no calibration corresponding to the
We can also see how the latent GP gives information on the calibration uncertainty.

Table 1: Multi-class calibration experiments. Average ECE\(_1\) of 10 Monte-Carlo cross validation folds on multi-class benchmark data sets. Calibration errors (ECE\(_1\)) within one standard deviation of lowest per data set and model are printed in bold.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Model</th>
<th>Uncal.</th>
<th>Platt</th>
<th>Isotonic</th>
<th>Beta</th>
<th>BBQ</th>
<th>Temp.</th>
<th>GPcalib</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>AdaBoost</td>
<td>.6121</td>
<td>.2267</td>
<td>.1319</td>
<td>.2222</td>
<td>.1384</td>
<td>.1567</td>
<td>.0414</td>
</tr>
<tr>
<td>MNIST</td>
<td>XGBoost</td>
<td>.0740</td>
<td>.0449</td>
<td>.0176</td>
<td>.0184</td>
<td>.0207</td>
<td>.0222</td>
<td>.0180</td>
</tr>
<tr>
<td>MNIST</td>
<td>Mondrian Forest</td>
<td>.2163</td>
<td>.0357</td>
<td>.0282</td>
<td>.0333</td>
<td>.0762</td>
<td>.0208</td>
<td>.0213</td>
</tr>
<tr>
<td>MNIST</td>
<td>Random Forest</td>
<td>.1178</td>
<td>.0273</td>
<td>.0207</td>
<td>.0259</td>
<td>.1233</td>
<td>.0121</td>
<td>.0148</td>
</tr>
<tr>
<td>MNIST</td>
<td>1 layer NN</td>
<td>.0622</td>
<td>.0126</td>
<td>.0140</td>
<td>.0168</td>
<td>.0186</td>
<td>.0195</td>
<td>.0239</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>AlexNet</td>
<td>.2751</td>
<td>.0720</td>
<td>.1232</td>
<td>.0784</td>
<td>.0478</td>
<td>.0365</td>
<td>.0369</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>WideResNet</td>
<td>.0664</td>
<td>.0838</td>
<td>.0661</td>
<td>.0539</td>
<td>.0384</td>
<td>.0444</td>
<td>.0283</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>ResNetXt-29 (8x64)</td>
<td>.0495</td>
<td>.0882</td>
<td>.0599</td>
<td>.0492</td>
<td>.0392</td>
<td>.0424</td>
<td>.0251</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>ResNetXt-29 (16x64)</td>
<td>.0527</td>
<td>.0900</td>
<td>.0620</td>
<td>.0520</td>
<td>.0365</td>
<td>.0465</td>
<td>.0266</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>DenseNet-BC-190</td>
<td>.0717</td>
<td>.0801</td>
<td>.0665</td>
<td>.0543</td>
<td>.0376</td>
<td>.0377</td>
<td>.0237</td>
</tr>
<tr>
<td>ImageNet</td>
<td>AlexNet</td>
<td>.0353</td>
<td>1.132</td>
<td>.2937</td>
<td>.2290</td>
<td>.1307</td>
<td>.0342</td>
<td>.0357</td>
</tr>
<tr>
<td>ImageNet</td>
<td>VGG19</td>
<td>.0377</td>
<td>.0965</td>
<td>.2810</td>
<td>.2416</td>
<td>.1617</td>
<td>.0342</td>
<td>.0364</td>
</tr>
<tr>
<td>ImageNet</td>
<td>ResNet-50</td>
<td>.0441</td>
<td>.0875</td>
<td>.2724</td>
<td>.2250</td>
<td>.1635</td>
<td>.0341</td>
<td>.0335</td>
</tr>
<tr>
<td>ImageNet</td>
<td>ResNet-152</td>
<td>.0545</td>
<td>.0879</td>
<td>.2761</td>
<td>.2201</td>
<td>.1675</td>
<td>.0323</td>
<td>.0283</td>
</tr>
<tr>
<td>ImageNet</td>
<td>DenseNet-121</td>
<td>.0380</td>
<td>.0949</td>
<td>.2682</td>
<td>.2297</td>
<td>.1512</td>
<td>.0329</td>
<td>.0357</td>
</tr>
<tr>
<td>ImageNet</td>
<td>DenseNet-201</td>
<td>.0410</td>
<td>.0898</td>
<td>.2706</td>
<td>.2189</td>
<td>.1614</td>
<td>.0324</td>
<td>.0367</td>
</tr>
<tr>
<td>ImageNet</td>
<td>InceptionV4</td>
<td>.0318</td>
<td>.0865</td>
<td>.2900</td>
<td>.1653</td>
<td>.1593</td>
<td>.0462</td>
<td>.0269</td>
</tr>
<tr>
<td>ImageNet</td>
<td>SE-ResNeXt-101</td>
<td>.0574</td>
<td>.0853</td>
<td>.2844</td>
<td>.1631</td>
<td>.1496</td>
<td>.0415</td>
<td>.0250</td>
</tr>
<tr>
<td>ImageNet</td>
<td>PolyNet</td>
<td>.0823</td>
<td>.0806</td>
<td>.2590</td>
<td>.2006</td>
<td>.1787</td>
<td>.0369</td>
<td>.0283</td>
</tr>
<tr>
<td>ImageNet</td>
<td>SENet-154</td>
<td>.0612</td>
<td>.0809</td>
<td>.3003</td>
<td>.1582</td>
<td>.1502</td>
<td>.0497</td>
<td>.0309</td>
</tr>
<tr>
<td>ImageNet</td>
<td>PNASNet-5-Large</td>
<td>.0702</td>
<td>.0796</td>
<td>.3063</td>
<td>.1430</td>
<td>.1355</td>
<td>.0486</td>
<td>.0270</td>
</tr>
<tr>
<td>ImageNet</td>
<td>NASNet-A-Large</td>
<td>.0530</td>
<td>.0826</td>
<td>.3265</td>
<td>.1437</td>
<td>.1268</td>
<td>.0516</td>
<td>.0255</td>
</tr>
</tbody>
</table>

Computation Time We give a complexity analysis for temperature scaling and GPcalib in Table 2. The cost of evaluation of the optimization objective and calibration for different variants of GP calibration are shown. For a diagonal covariance matrix, evaluating the optimization objective is of similar complexity to temperature scaling, since in general \( NK \) dominates the cubed number of inducing points. However, in our experiments the optimizer converged more slowly for GPcalib than for temperature scaling. We provide wall-clock inference and prediction runtime averaged across models per benchmark data set in Section S3.6.1 of the supplementary material. In practice, for most classifiers the training time is orders of magnitude larger than the time for inferring the latent GP in our calibration method. Calibration is computationally more expensive for GPcalib compared to other methods in part due to the marginalization of the calibration uncertainty. This can be reduced by one order of magnitude for data sets with a large number of classes via the mean approximation with practically no effect on the ECE\(_1\). The resulting time taken for calibration is about one-order of magnitude more than temperature scaling. In our experiments this was at least two orders of magnitude less than the prediction time of the classifier.

Table 2: Computational complexity. Complexity analysis of evaluation of the optimization objective for parameter inference of the calibration methods and complexity of calibration. \( N \) denotes the size of the calibration data, \( K \) the number of classes, \( M \) the number of inducing points and \( Q \) the number of MC samples. For the mean approximation \( a = 1 \) with a diagonal covariance and \( a = 2 \) for the full covariance. Implementation choices are \( M = 10 \) and \( Q = 100 \).

<table>
<thead>
<tr>
<th>Method</th>
<th>Optim. obj.</th>
<th>Calibration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp. scal.</td>
<td>( O(NK) )</td>
<td>( O(K) )</td>
</tr>
<tr>
<td>GPcalib</td>
<td></td>
<td></td>
</tr>
<tr>
<td>diag. cov.</td>
<td>( O(NK + M^3) )</td>
<td>( O(K(M^2 + Q)) )</td>
</tr>
<tr>
<td>full cov.</td>
<td>( O(NK^2 + M^3) )</td>
<td>( O(K^2(M^2 + Q)) )</td>
</tr>
<tr>
<td>mean appr.</td>
<td>( O(NK^a + M^3) )</td>
<td>( O(K^aM^2) )</td>
</tr>
</tbody>
</table>
Non-Parametric Calibration for Classification

Figure 4: Non-linear calibration maps in logit-space. The plot shows latent functions of temperature scaling and GPcalib from a single CV run of our experiments on ImageNet. For PolyNet and PNASNet GPcalib shows a significant decrease in ECE$_1$ in Table 1, corresponding to a higher degree of non-linearity in the latent GP.

Figure 5: Active learning and calibration. ECE$_1$ and classification error for two Mondrian forests trained online on labels obtained via an entropy query strategy on the KITTI data set. One forest is calibrated in regularly spaced intervals with GPcalib (gray). A GP regression up to the average number of queried samples across folds is shown. The calibrated forest queries $\sim 10\%$ less labels, while reaching comparable accuracy.

5 CONCLUSION

In this paper we proposed a novel multi-class calibration method for arbitrary classifiers based on a latent Gaussian process, inferred via variational inference. We evaluated different calibration methods for a range of classifiers often employed in computer vision and robotics on a collection of benchmark data sets. Our method demonstrated strong performance across different model and data set combinations and performed particularly well for large-scale neural networks. In Theorem 1 we linked calibration to concepts from active learning. We conclude with a motivating example for the possible impact of calibration on active learning and outline future research directions.

Active Learning We hypothesize that calibration could improve active learning when querying based on uncertainty. We trained two Mondrian forests on the multi-class KITTI data set. These are well-suited for the online setting as they have the same distribution whether trained online or in batch. We randomly shuffled the data 10 times and requested samples based on an entropy query strategy with a threshold of 0.25. Any samples above the threshold are used for training or calibration. Both forests are trained for 500 samples and subsequently one uses 250 samples exclusively for calibration in regularly spaced intervals. We report the ECE$_1$ and classification error in Figure 5. The calibration initially incurs a penalty on accuracy, as fewer samples are used for training. This is remedied over time through more efficient querying. The same accuracy is reached after a pass through the entire data set while querying less samples overall. This can be explained by calibration adjusting over- and underconfidence to the ratio determined by Theorem 1. Here, underconfidence is reduced, leading to less querying of uninformative samples. This is a promising result, but further research questions arise, regarding the size of the pre-training batch, the condition when calibration should be done, and the number of samples used for calibration. We believe that there is a trade-off, similar to an explore-exploit strategy, between classifier training and uncertainty calibration.

Future Directions Our proposed calibration approach is worth extending in the following directions: (a) forcing a monotone latent Gaussian process (Rihimäki and Vehtari, 2010, Agrell, 2019) provably preserves the accuracy of the underlying classifier; (b) extending our method to the online setting (Bui et al., 2017) allows for continuous calibration; (c) using our method for what we call active calibration, the concept of using an active learning query strategy, which switches between requesting samples for model training and uncertainty calibration based on the uncertainty of the latent Gaussian process.
Acknowledgements

JW gratefully acknowledges financial support by the European Research Council through ERC StG Action 757275 / PANAMA; the DFG Cluster of Excellence “Machine Learning - New Perspectives for Science”, EXC 2064/1, project number 390727645; the German Federal Ministry of Education and Research (BMBF) through the Tübingen AI Center (FKZ: 01IS18039A); and funds from the Ministry of Science, Research and Arts of the State of Baden-Württemberg. JW is grateful to the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for support. RT gratefully acknowledges the Helmholtz Artificial Intelligence Cooperation Unit (HAICU), which partly supported this work.

The authors thank the anonymous reviewers for helpful comments on an earlier version of this manuscript. In particular we appreciate the suggestions on inference using the mean estimate of the latent Gaussian process and regarding the definition of over- and underconfidence.

References


Non-Parametric Calibration for Classification


