Exploring the Potential of Bilevel Optimization for Calibrating Neural Networks

Gabriele Sanguin[1], Arjun Pakrashi[2], Marco Viola[3,*] and Francesco Rinaldi[1]

[1]*Dipartimento di Matematica, Università degli Studi di Padova, Via 8 Febbraio 2, 35122 Padova (Italy)*
[2]*School of Computer Science, University College Dublin, Belfield, Dublin 4 (Ireland)*
[3]*School of Mathematical Sciences, Dublin City University, Collins Avenue Ext, Dublin 9 (Ireland)*

Abstract

Handling uncertainty is critical for ensuring reliable decision-making in intelligent systems. Modern neural networks are known to be poorly calibrated, resulting in predicted confidence scores that are difficult to use. This article explores improving confidence estimation and calibration through the application of bilevel optimization, a framework designed to solve hierarchical problems with interdependent optimization levels. A self-calibrating bilevel neural-network training approach is introduced to improve a model's predicted confidence scores. The effectiveness of the proposed framework is analyzed using toy datasets, such as Blobs and Spirals, as well as more practical simulated datasets, such as Blood Alcohol Concentration (BAC). It is compared with a well-known and widely used calibration strategy, isotonic regression. The reported experimental results reveal that the proposed bilevel optimization approach reduces the calibration error while preserving accuracy.

Keywords

Bilevel optimization, confidence, calibration, neural networks

1. Introduction

Machine learning classification algorithms with increasingly higher discriminative power, especially neural networks, have rapidly developed in the last decade. Such advanced models are generally supposed to help humans make decisions by assisting. Although these models have a high discriminative power, sometimes they will predict something completely incorrect with a fairly high confidence score. This creates a huge problem in highly regulated and sensitive real-world applications (e.g. medical field, autonomous vehicles, healthcare diagnostics, financial forecasting, etc.) [1]. Therefore, it is important for these models to provide a meaningful confidence, based on which it can say "I don't know", when they are not confident enough, so that the human expert can inspect, and make further decisions. This is sometimes called *learning to reject* or *abstention* [2, 3].

Confidence is a probabilistic score that a model assigns to each prediction of a data point, and it determines how certain the model is about the prediction. One straightforward approach to use such a confidence score to understand if the model is confident *enough* or not is to define an interval, a *rejection window*, within which if the prediction falls, it will be marked as *reject*. It is however hard to fix such a confidence window to decide the rejection window for such models, especially modern neural network models, because such models are known to have poor model confidence calibration [1, 4].

Confidence *calibration* is the process of adjusting the confidence scores to better align with the actual likelihood of correctness. Well-calibrated confidence is crucial for effective decision making and is important for the interpretation of the model, since humans have a natural cognitive intuition for probabilities [5]. Accurate confidence scores make it easier for users to comprehend how confident the model was during prediction and to establish trust on the decisions being made. Moreover, accurate confidence scores are essential if such a rejection window needs to be defined by a human expert.

There are two types of confidence calibration. Firstly, *post-calibration*, where the output scores/probabilities of the main model are re-adjusted by another external calibration model. For such methods,

the main model does not need to be modified. In [4], the authors have demonstrated how well-known post-calibration methods can be used to calibrate existing models. On the other hand, *self-calibration* method are algorithms which integrate the confidence calibration process into the model training itself. These methods aim to ensure that the model's probabilities are calibrated during the training phase, without the need for a separate calibration step.

The objective of this article is to present an initial study on whether it is possible to self-calibrate neural network models by exploiting *bilevel optimization* (BO), a mathematical framework specifically designed to solve hierarchical two-level decision-making optimization problems. BO has recently gained importance in machine learning, particularly in hyperparameter optimization and metalearning [6, 7, 8]. The choice of such a framework is obvious where the inner-level optimization problem tackles the training of the model, while the outer-level optimization problem addresses the model confidence calibration.

To the best of our knowledge, in the context of uncertainty scores, the only work in the literature that uses BO is [9]. Here, we define a BO framework to train two different architectures, one for classification and one for the uncertainty score (which is then tested as a rejection function), thus leading to a significant increase of the parameters to be trained. Then we focus on the study on the selective potential of such a score. The work presented in the current article is the first of its kind, which is quite distinct from the one in [9]. The objective of the current work is to propose a BO framework to train a single self-calibrated deep neural network, *BO4SC*, and provide an initial, but crucial analysis of the applicability of the approach.

The article is organized as follows. Section 2 discusses the mathematical foundations of confidence estimation, calibration methods, and bilevel optimization (BO). Section 3 introduces BO4SC, a BO framework for confidence estimation in neural networks. In Section 4 the initial experiments and analysis are shown and discussed. Finally, Section 5 concludes the article.

2. Theoretical Background

This section will briefly introduce and discuss the relevant parts of confidence estimation, model calibration evaluation, calibration methods, and bilevel optimization.

2.1. Confidence Estimation

For a machine learning model, the standard process (referred to as *standard* method later in Section 4) to accurately predict an output involves learning a mapping function that can generalize well from training data to unseen samples. Let $\{X, Y\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ be a labeled dataset, where $\mathbf{x}_i$ represents a data point and $y_i \in \{1, 2, \ldots, C\}$ is its corresponding class label, where C being the total number of classes and n the number of data points. The objective is usually to learn a function f that maps an unseen data point $\mathbf{x}_t$ to a predicted output $\hat{y}_t = f(\mathbf{x}_t)$. This function f can be efficiently and effectively trained by minimizing the empirical loss over all training data.

Confidence estimation involves assigning a probabilistic score to each prediction which reflects the model's certainty about the predicted output. *Confidence* is a score function $\hat{p}_i = g(f, \mathbf{x}_i, \hat{y}_i)$, which measures the likelihood of the prediction $\hat{y}_i$ being correct given the features $\mathbf{x}_i$ and the classifier f. Ideally, the confidence score should be continuous and fall within the range $[0, 1]$.

A variety of methods have been developed to enable confidence estimation across different model types. Among these approaches, we find *distance-based* methods, which use the distance of a data point from other points, decision boundaries, or centroids of classes to estimate confidence [10, 11, 12, 13, 14]. *Bayesian uncertainty* methods use Bayesian principles to model uncertainty, providing a probabilistic interpretation of confidence [15, 16, 17, 18]. *Reconstruction error* techniques rely on the error of reconstructing input data to obtain a confidence score, often used in models with an encoder-decoder framework, such as autoencoders. The idea is that a high reconstruction error indicates a lower confidence in the prediction of the model, [19, 20]. *Ensemble methods* utilize the variance among predictions from multiple models to estimate confidence [21, 22]. *Extreme value theory (EVT)* approaches

are based on EVT that assess confidence by modeling the tail distributions of prediction scores [23]. Finally, *logits-based* techniques involve the use of logits, which include probabilistic outputs and other mechanisms derived from the raw scores produced especially by neural networks. These scores can be transformed or analyzed to estimate the confidence of the predictions [24].

Logits-based methods are the ones that are gaining more attention for the recent extensive use of neural networks. The experiments in the current work use a smooth version of *maximum class probability* (MCP), which is a common approach in many classification tasks. We define the MCP as follows

$$\hat{p}(x) = \max_c P(y = c \mid \mathbf{x}), \tag{1}$$

where $P(y = c \mid \mathbf{x})$ represents the predicted probability of class c for input $\mathbf{x}$ after applying a softmax function to the logits.

2.2. Evaluating Model Calibration

Unlike classification functions, which can be efficiently learned from labeled data, there is no available supervisory information for directly learning a *confidence function* and the first challenge is how to estimate it from pre-trained models (e.g. with MCP). Unfortunately, in many cases, especially in modern deep neural networks, the calculated confidences tend to be overestimated, meaning that the models are *over-confident* (see [2]). This can be formally described as

$$P(\hat{y} = y \mid \hat{p} = p) < p, \tag{2}$$

where p represents the true probability.

To address this issue, it is necessary to employ methods to calibrate the confidence. A model is considered *calibrated* if $\hat{p}_i$ accurately reflects the true likelihood of correctness:

$$P(\hat{y} = y \mid \hat{p} = p) = p, \quad \forall p \in [0, 1]. \tag{3}$$

To understand whether a model is well-calibrated one can exploit metrics quantifying the degree to which the model's predicted probabilities align with the actual outcomes or its likelihood. It is important to note that there is no single, universally accepted metric for assessing calibration.

In this work, we will use two of the most common calibration metrics in the literature, namely: *reliability diagrams* and *expected calibration error* (ECE).

Reliability diagrams are a visual tool used to assess model calibration [25, 26] by plotting the expected accuracy of samples against their predicted confidence levels. The predictions are grouped into M interval bins, each of size $\frac{1}{M}$. By letting B_m represent the set of indices of samples whose predicted confidence falls within the interval $I_m = \left(\frac{m-1}{M}, \frac{m}{M} \right]$, the accuracy for bin B_m is calculated as

$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i),$$

where $\hat{y}_i$ and y_i are the predicted and true class labels for data point $\mathbf{x}_i$, respectively. According to basic probability theory, $\mathrm{acc}(B_m)$ serves as an unbiased and consistent estimator of $P(\hat{y} = y \mid \hat{p} \in I_m)$.

The average confidence within the bin B_m is given by

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i,$$

where $\hat{p}_i$ represents the confidence of the sample i. For a perfectly calibrated model, the relationship $\mathrm{acc}(B_m) = \mathrm{conf}(B_m)$ should hold for all $m \in \{1, \ldots, M\}$, i.e., the plot should follow the identity line. It is important to note that reliability diagrams do not display the proportion of samples in each bin. This is also why they are often paired with a density plot of confidence prediction, called *confidence histograms*.

The ECE represents the weighted average of the absolute difference between accuracy and confidence over all prediction bins. Formally, it is defined as:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \mathrm{acc}(B_m) - \mathrm{conf}(B_m) \right|, \tag{4}$$

where n is the total number of samples. Although ECE is widely adopted due to its simplicity and interpretability, it is sensitive to the choice of the number of bins M, which can affect the accuracy of the measurement.

2.3. Calibration Methods

Calibration methods can be categorized into two types, namely post-calibration and self-calibration.

Post-calibration: These methods involve adjusting the output probabilities of a pretrained model using a separate calibration model, applied after the initial model has been trained. This adjustment aims to align the predicted probabilities with the true likelihood of events.

Among the most common approaches we can find histogram binning [27], Bayesian binning into quantiles (BBQ) [28], Platt scaling [29, 26] and its derivative matrix and vector scaling and temperature scaling [4]. Other recent methods include beta calibration [30], Shape-Restricted Polynomial Regression [31] and neural calibration [32].

The experiments in the current work make use of the *isotonic regression* [33], because of its simplicity and effectiveness. Isotonic regression learns a piecewise constant function f to transform uncalibrated outputs into calibrated ones, by minimizing the squared loss subject to the constraint that f is a non-decreasing function.

Self-calibration: These methods integrate the calibration process into the model training itself. These methods aim to ensure that the model's probabilities are calibrated during the training phase, without the need for a separate calibration step. Self-calibration often requires modifying the loss function or the training procedure to directly incorporate the calibration objectives. Techniques such as Bayesian neural networks, which incorporate uncertainty directly into the model predictions through probabilistic inference, inherently produce better-calibrated probabilities [16, 34].

The main objective of this article is to explore a new self-calibration strategy for neural networks that makes use of a bilevel optimization framework.

2.4. Bilevel Optimization

Bilevel optimization (BO) is a mathematical approach designed to address hierarchical decision-making processes, where decisions made at an outer level influence the outcomes of an inner level, which in turn affects the outer level. This hierarchical structure is prevalent in many real-world scenarios, such as economics, engineering, management, and various public and private sector operations. The distinctive feature of bilevel optimization lies in its two interconnected levels of optimization. Each level has its own objectives and constraints, and there are two classes of decision vectors: the leader's (outer-level) decision vectors and the follower's (inner-level) decision vectors. The inner-level optimization is a parametric optimization problem solved with respect to the inner-level decision vectors, while the outer-level decision vectors act as parameters. The inner-level optimization problem acts as a constraint to the outer-level optimization problem, such that only those solutions are considered feasible that are optimal for the inner level.

By denoting the outer and inner parameters as $\mathbf{w}$ and θ, respectively, we can define an unconstrained BO problem as

$$\min_{\mathbf{w}} \ f(\mathbf{w}, \theta^*) \qquad \text{s.t.} \ \theta^* \in \arg\min_{\theta} g(\mathbf{w}, \theta), \tag{5}$$

where θ^* is one of the minimizers of g.

Gradient-based approaches are now the most commonly used methods for solving bilevel optimization problems. The most compelling approach to gradient-based bilevel optimization is to replace the inner problem with a *dynamical system.* This idea, discussed, e.g., in [7, 35, 36], involves approximating the bilevel problem with a sequence of optimization steps, which allows for efficient gradient computation.

Specifically, consider a prescribed positive integer T and let $[T] = \{1, 2, \ldots, T\}$. We now rewrite the bilevel problem Eq.(5) with the following approximation:

$$\min_{\mathbf{w}} \quad f(\mathbf{w}, \theta^T(\mathbf{w}))$$
$$\text{s.t. } \theta^0(\mathbf{w}) = \Phi_0(\mathbf{w}), \tag{6}$$
$$\theta^t(\mathbf{w}) = \Phi_t(\theta^{t-1}(\mathbf{w}), \mathbf{w}), \quad t \in [T],$$

where $\Phi_0 : \mathbb{R}^n \to \mathbb{R}^m$ is a smooth initialization mapping and for each $t \in [T]$, $\Phi_t : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ represents the operation performed by the t-th step of an optimization algorithm. For example, if the optimization dynamics is gradient descent, we might have:

$$\Phi_t(\theta^{t-1}, \mathbf{w}) = \theta^{t-1} - \eta_t \nabla_\theta g(\theta^{t-1}, \mathbf{w}), \tag{7}$$

where $(\eta_t)_{t \in [T]}$ is a sequence of step sizes.

This approach approximates the bilevel problem and gives the possibility to use gradient descent also to solve the outer objective. To this end, one has to compute an *hypergradient*, which is the gradient of the outer objective $f(\mathbf{w}, \theta^T(\mathbf{w}))$ with respect to the hyperparameters $\mathbf{w}$, i.e.,

$$\nabla_{\mathbf{w}} f(\mathbf{w}, \theta^T(\mathbf{w})) = \nabla_{\mathbf{w}} f(\mathbf{w}, \theta^T) + [J_{\theta^T(\mathbf{w})}(\mathbf{w})]^\top \nabla_\theta f(\mathbf{w}, \theta^T), \tag{8}$$

where rows in the Jacobian matrix $J_{\theta^T(\mathbf{w})}(\mathbf{w})$ contain gradients of the entries of θ^T with respect to $\mathbf{w}$.

The reformulation (6) allows for efficient computation of the hypergradient using reverse or forward mode algorithmic differentiation.

3. BO4SC: A Bilevel Optimization Framework for Self-Calibration

We introduce here the bilevel optimization framework we designed to enhance confidence estimation, which we will name *BO4SC*.

We here assume that the prediction models are characterized by a *dual-output structure*: one output to provide the prediction for the data point, the other to estimate the confidence of that prediction. This is essential because we want both the class predictions and the confidence estimation to be dependent on the same model parameters. For the model m, parametrized by θ, we will denote the output relative to the sample $\mathbf{x}_i$ with

$$m(\mathbf{x}_i, \theta) = (\hat{y}(\mathbf{x}_i, \theta), \hat{p}(\mathbf{x}_i, \theta)) = (\hat{y}_i, \hat{p}_i), \tag{9}$$

where $\hat{y}_i$ is the class prediction and $\hat{p}_i$ is his confidence estimation.

Now consider the optimization problem in Eq. (5), with the outer parameters the weights $\mathbf{w}$ and the inner parameters θ of the model m_θ. The inner loss function g is trained on the training set (D^{train}), focusing on minimizing the *weighted cross-entropy* (CE) loss over the model's prediction output with the objective of minimizing the model's parameters θ:

$$g(\mathbf{w}, \theta) = \frac{1}{|D^{\text{train}}|} \sum_{i \in D_{\text{train}}} w_i \cdot \text{CE}(\hat{y}(\mathbf{x}_i, \theta), y_i) \tag{10}$$

supposing θ^* to be unique and where the CE loss is defined as:

$$\text{CE}(\hat{y}(\mathbf{x}_i, \theta), y_i) = -\sum_{c=1}^{C} y_{i,c} \log(\hat{y}(\mathbf{x}_i, \theta)_c) \tag{11}$$

Here, C represents the number of classes, $y_{i,c}$ is the binary indicator (0 or 1) if the class label c is the correct classification for input $\mathbf{x}_i$, and $\hat{y}(\mathbf{x}_i, \theta)_c$ is the final logit for class c given input $\mathbf{x}_i$ according to the model $\hat{y}(\cdot, \theta)$.

The outer loss function f, on the other hand, is evaluated on the validation set (D^{val}), where it aims to minimize a *binary cross-entropy* (BCE) loss on the model's confidence output $\hat{p}(\cdot, \theta)$. The objective is to learn weights for each sample in the training set that can effectively balance the trade-off between prediction accuracy and confidence calibration:

$$f(\mathbf{w}, \theta^*) = \frac{1}{|D^{\mathrm{val}}|} \sum_{j \in D^{\mathrm{val}}} \mathrm{BCE}(\hat{p}(\mathbf{x}_j, \theta_{\mathbf{w}}^*), y_j), \tag{12}$$

where $\theta_{\mathbf{w}}^*$ are the model parameters found by the inner problem and that depend on the weights $\mathbf{w}$ assigned to the training samples. $\hat{p}(\cdot, \theta)$ is the confidence output of the model.

The binary cross-entropy (BCE) loss is defined as:

$$\mathrm{BCE}(\hat{p}(\mathbf{x}_j, \theta^*), y_j) = - \left[y_j^B \log(\hat{p}(\mathbf{x}_j, \theta^*)) + (1 - y_j^B) \log(1 - \hat{p}(\mathbf{x}_j, \theta^*)) \right] \tag{13}$$

In this equation, y_j^B is the true *binary label* (0 or 1) for the sample $\mathbf{x}_j$, indicating whether $\mathbf{x}_j$ has been correctly classified (i.e. $\hat{y}_j = y_j$); $\hat{p}(\mathbf{x}_j, \theta^*)$ represents the predicted confidence (probability) that $\mathbf{x}_j$ belongs to the positive class according to the model.

The difficulty in solving this bilevel optimization problem usually lies in the accurate computation of the hypergradient $\nabla_{\mathbf{w}} \mathcal{L}_{\mathrm{outer}}(\mathbf{w}) = \nabla_{\mathbf{w}} f(\mathbf{w}, \theta_{\mathbf{w}}^*)$, which necessitates sophisticated approaches requiring a large cost in time and memory performance.

We schematize as Algorithm 1 the approximate hypergradient descent algorithm we implemented to solve the BO4SC problem.

Algorithm 1 BO4SC via Approximate Hypergradient Descent

Initialize: Set initial weights $\mathbf{w}^0$ and model parameters θ^0.
for $j = 0, 1, \ldots$ **do**
  for $k = 0$ to $T - 1$ **do**
    {Inner loop: gradient descent on inner loss}
    Compute the gradient of the inner loss w.r.t. θ^k:

$$\nabla_\theta g(\mathbf{w}^j, \theta^k) = \frac{1}{|D^{\mathrm{train}}|} \sum_{i \in D^{\mathrm{train}}} w_i^j \cdot \nabla_\theta \mathrm{CE}(\hat{y}(\mathbf{x}_i, \theta^k), y_i)$$

    Update model parameters θ^k using gradient descent: $\theta^{k+1} = \theta^k - \eta_\theta \cdot \nabla_\theta g(\mathbf{w}^j, \theta^k)$
  end for
  Set $\theta_{\mathbf{w}}^j = \theta^T$ {Final inner solution after T iterations, in function on outer parameters $\mathbf{w}$}
  Compute the *hypergradient*, i.e. the gradient of the outer loss w.r.t. $\mathbf{w}$, using the approximated $\theta_{\mathbf{w}}^j$:

$$\nabla_{\mathbf{w}} f(\mathbf{w}, \theta_{\mathbf{w}}^j) = \frac{1}{|D^{\mathrm{val}}|} \sum_{j \in D^{\mathrm{val}}} \nabla_{\mathbf{w}} \mathrm{BCE}(\hat{p}(\mathbf{x}_j, \theta_{\mathbf{w}}^j), y_j) \tag{14}$$

  Update the outer-parameters $\mathbf{w}^j$ using gradient descent: $\mathbf{w}^{j+1} = \mathbf{w}^j - \eta_{\mathbf{w}} \cdot \nabla_{\mathbf{w}} f(\mathbf{w}^j, \theta_{\mathbf{w}}^j)$
end for

---

4. Experiments and Results

In this section we present our experiment process and the results. First, we give an overview on the training approaches we compared and the datasets we used. Then we analyse the experiment results.

This work mainly focuses on the proposed method along with two others which are described as follows:

- *Standard*: this refers to the standard training procedure in which the model's parameters are updated using backpropagation based on a *single* loss function.
- *Isotonic Regression (IsoReg)*: it is the non-parametric method used to calibrate confidence scores after the initial training phase of a model with the *Standard* method (see Section 2.3).
- *BO4SC*: the proposed method in this work Algorithm 1).

In the implementation of the BO4SC algorithm, particularly for the explicit calculation of the outer loss gradient with respect to $\mathbf{w}$ (that is, the gradient of $\theta_{\mathbf{w}}^T$ with respect to $\mathbf{w}$), the Python package `torchopt` [37] was used. `torchopt` is a library that extends PyTorch [38] by providing tools for higher-order optimization, specifically tailored for problems involving complex optimization hierarchies such as bilevel optimization. It enables efficient computation of the hypergradients. By leveraging `torchopt`, we can accurately and efficiently compute the required gradients in Eq. (14), thereby facilitating the optimization process in our experiments.

To facilitate the initial investigation, some toy datasets were built which were used as a diagnostic tool to understand BO4SC behavior, as well as how it compares with others. These datasets have two features to facilitate visual inspection.

The first two datasets are *Blobs 1.3* and *Blobs 1.7*, each of which has two dimensions and five classes, where the blobs are generated from a normal distribution with standard deviations of 1.3 and 1.7, respectively. The third and fourth are two class datasets named *Spiral 2.5* and *Spiral 3.5*, consisting of two interlocking spiral-shaped regions, each corresponding to one class, with the values 2.5 and 3.5 indicating the standard deviation from the center of the spiral, thus controlling the amount of overlap between the regions. These datasets are used for diagnostic purposes to understand the behaviour of the algorithm. Finally, we used the *Blood Alcohol Concentration (BAC)* dataset, which is commonly utilized in decision-making and confidence estimation tasks. Data were first collected by Nugent and Cunningham [39] and can be used for regression and binary classification, depending on whether a threshold is set on the BAC level to distinguish between classes. Both the toy datasets, Blobs and Spirals, and BAC are made of 2000 samples in total, 700 are used for training, 300 for validation and 1000 in the test set.
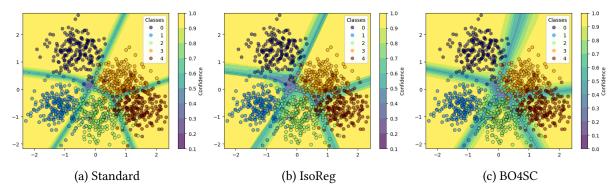
For each dataset a feed forward neural network has been implemented, with a softmax function applied to the final logits. The MCP is extracted with a smooth maximum function, namely the *Boltzman operator* [40], to keep the confidence score differentiable with respect to the model parameters. The Adam [41] optimizer was used in the *standard* training and in the inner loop of BO4SC (to optimize the model parameters θ). All hyperparameters has been selected through a grid search. Besides the number of epochs, in the bilevel approach it is important to adjust the number of inner iterations (T) and the learning rate $\eta_{\mathbf{w}}$ for the update of the outer parameters.

4.1. Confidence Estimation and Calibration

What interests us in our experiments is assessing how well the methods predict calibrated confidence estimations. We begin with a analysis using one of the toy datasets, where we can observe how well the models differentiate between high-confidence and low-confidence regions.

The toy dataset *Blobs 1.7* provides an excellent case for this analysis. In Figure 1 below, we present an image made up of three different plots, each representing the confidence estimation results. These plots visually demonstrate the predicted confidence levels across the entire input space, highlighting areas where each model is more or less confident in its predictions.

The *standard* model is highly confident in most regions, as indicated by the yellow areas. These regions reflect the areas where the model predicts class membership with high certainty (confidence value in $(0.9, 1]$). However, this confidence is sharply reduced in very narrow areas corresponding to the decision boundaries, represented by the green regions. These 'lines' of uncertainty appear consistently thin across different parts of the dataset, irrespective of the degree of overlap between classes. In contrast, the BO4SC model's confidence regions show a different pattern. Here, the uncertainty regions are considerably broader, especially in areas where the classes overlap more. This broader distribution of uncertainty better reflects the true complexity and intersections within the data, suggesting that the BO4SC approach is more sensitive to the nuances of the dataset's distribution. This ability, which also characterize the Isotonic Regression post-calibration method, represents a significant improvement over the Standard model, highlighting the advantages of a post- or self- calibration technique to address the confidence estimation challenge.

| (a) Standard | (b) IsoReg | (c) BO4SC |

Figure 1: Confidence region estimation on the Blobs 1.7 dataset for different approaches. Each plot represents the spatial distribution of confidence levels across the dataset. The color in the background represents the confidence value that the model associates to a point that would be found in that place.
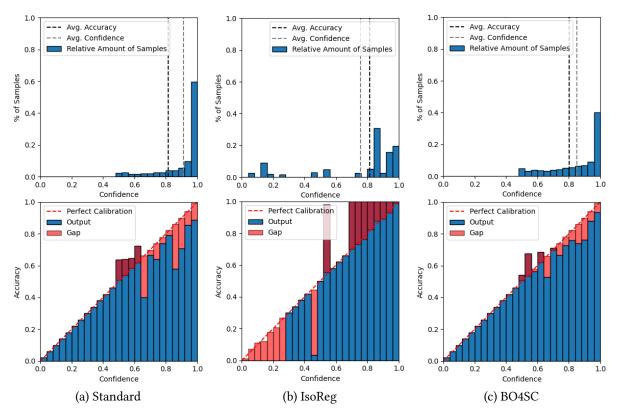
A more detailed examination using quantitative metrics is essential to rigorously evaluate the effectiveness of these methods and of bilevel optimization in producing well-calibrated models.

The first step is to examine the confidence calibration through *reliability diagrams* (Section 2.2) and *confidence histograms*. These visual tools provide a direct representation of the relationship between predicted probabilities and actual observed frequencies, allowing for a straightforward assessment of a model's calibration. The plots are consistently similar in all datasets, and we present in Figure 2 the reliability diagrams for the *Spiral 3.5* dataset, which emphasize a drawback of Isotonic Regression. A critical aspect to consider is the gap between the two dashed vertical lines in the confidence istogram: the darker line represents average accuracy, while the lighter grey line indicates the average confidence. For a model to be considered well-calibrated, these two lines should ideally overlap, or at least be very close to each other. The closer these lines are, the more aligned the model's predicted confidence is with its actual performance. When we examine the toy datasets, the gap between these two lines becomes particularly noticeable. The Standard model consistently displays the largest gap between the average accuracy and the average confidence across all datasets. This wide gap, with the darker line staying on the left side, implies that the model's confidence scores are overly optimistic and do not accurately reflect its true performance.

On the other hand, the BO4SC model shows the smallest gap, indicating that it has a more accurate alignment between confidence and accuracy. The IsoReg method also achieves a relatively close alignment between these two metrics. However, there is a nuanced difference between the confidence distribution obtained through bilevel optimization and the distribution achieved by post-calibration methods like IsoReg. Although IsoReg effectively narrows the gap between accuracy and confidence, it does not always appropriately adjust confidence predictions. In the spiral dataset for example, the IsoReg model produces confidence scores that fall within the $(0, 0.5]$ range. Since these datasets are binary classification tasks, the minimum reasonable confidence score should be around 0.5, reflecting the baseline probability of a random guess. The presence of lower confidence scores indicates an improper adjustment given by the IsoReg model, where it underestimates the confidence needed, thereby deviating from a reasonable calibration.

The reliability diagrams further reinforce the conclusions drawn from the confidence histograms. The Standard model demonstrates a clear tendency toward overconfidence. This is evident from the prevalence of orange gaps, especially in the higher confidence bins. In contrast, the bilevel optimization approach exhibits much better calibration, with reliability diagrams visually more balanced. Interestingly, while IsoReg effectively reduces the overconfidence seen in the Standard model, it introduces occasional calibration issues of its own. In particular, it may undercorrect or overcorrect certain confidence levels, leading to gaps that are not entirely aligned with the model's true accuracy.

With regard to confidence calibration metrics, we report in Table 1 the results for the *Expected Calibration Error (ECE)* and the Accuracy of the models. The Expected Calibration Error (ECE) shows that the Bilevel Optimization method generally achieves lower values compared to the traditional

Figure 2: Confidence Histograms (top) and Reliability Diagrams (bottom) for Spiral 3.5 test set. Orange sections represent overconfident gap, while red represents underconfidence.

Table 1

Comparison of Expected Calibration Error (ECE) and Accuracy across different datasets for Standard, IsoReg, and BO4SC methods. The best performance for each dataset and metric is highlighted in bold.
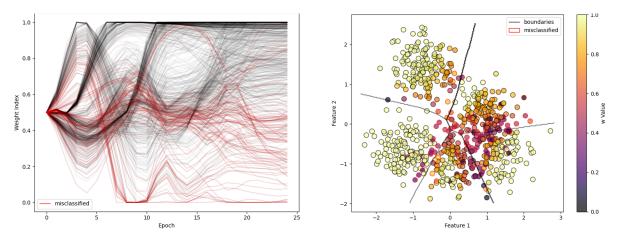
Method	Blobs 1.3	Blobs 1.7	Spiral 2.5	Spiral 3.5	BAC
Expected Calibration Error (ECE)					
Standard	0.026	0.074	0.064	0.109	0.018
IsoReg	0.023	0.039	0.039	0.143	**0.004**
BO4SC	**0.017**	**0.016**	**0.025**	**0.067**	0.012
Accuracy					
Standard	**0.94**	**0.876**	0.91	**0.815**	0.989
IsoReg	**0.94**	**0.876**	0.91	**0.815**	0.989
BO4SC	0.931	0.859	**0.923**	0.801	**0.994**

Standard and IsoReg methods, indicating better calibration and more reliable confidence scores that are closer to the true probabilities, while keeping good accuracies overall.

4.2. Training Weights

We can make some additional comments regarding the training approach that exploits bilevel optimization. One of these relates to the role of the *weights* assigned to each training sample. The weighted approach used allows the model to prioritize certain samples over others during training, potentially leading to better calibration and improved performance on more challenging or ambiguous classification. By studying the evolution of these weights in BO4SC, we can better understand how the method operates.

In Figure 3 the history of the weights values (left panel) and their final distribution (right panel) are reported for the Blobs 1.7 dataset. In the left panel, the red lines indicate the weights associated

Figure 3: *Left:* evolution of training *weights* found by the BO4SC method for the Blobs 1.7 dataset (1 epoch unit = 10 training epochs). *Right:* Final weight distribution.

with those samples that at the end result to be misclassified. One can clearly see that the weights often move in groups, creating bundles of lines that follow the same trend. They might represent groups of samples close to each other that have the same characteristic or close in the variable space. The main observation is that most of the red lines end between 0 and 0.5, while the darker lines are mainly above the middle value.

Looking at the right panel of Figure 3 one can observe that the BO4SC approach assigns a weight value of 1.0 to samples that are clearly and confidently classified into a single class, typically those located near the center of each cluster, far from the decision boundaries. As samples approach these boundaries, their weights decrease, converging towards 0.5 or even lower. This trend reflects BO4SC's strategy to diminish the influence of samples that are ambiguous or more likely to be misclassified. This is visually evident, as many of these samples are marked with a red contour to indicate their misclassification, so belonging to a cluster of a different class, and appear as dark-colored (black) points, indicative of their low weight.

5. Conclusions

In this article, we explored a novel bilevel optimization approach to address the challenge to self-calibrate a neural network in classification tasks. The objective was to improve the *confidence* predicted by a model in such a way that it better reflects the actual accuracy and that it would be more meaningful in ambiguous scenarios. We made experimentation and analysis across a variety of datasets, ranging from toy datasets like Blobs and Spirals to more complex ones like BAC, and demonstrated the effectiveness of bilevel methods, particularly in their ability to refine confidence by dynamically adjusting sample weights during training.

We used the Expected Calibration Error (ECE) to quantitatively assess the models' performance. The consistent superiority of the bilevel approach over traditional methods highlights its ability to enhance classifier reliability while maintaining good accuracies overall.

The bilevel approach behaves well also when compared with post-calibration techniques. They present better results and, more importantly, they do not suffer from typical issues that show up when post-calibrating the confidence. In fact, we found that fine-tuning with post-calibration methods, like isotonic regression, occasionally leads to over-adjustments, resulting in overly cautious confidence estimates. For this reason, the confidence produced by the bilevel optimization methods would be more trustworthy in a real-world scenario.

While the results are promising, future research should focus on further refining these techniques. There is indeed still room for improvements on the computational side, i.e., executional time and memory performance of our bilevel approach are not always competitive with traditional training.

Another future research direction lies towards *reject-option classification*, which allows models to refrain from making uncertain predictions.

Acknowledgements

References

[1] M. Minderer, J. Djolonga, R. Romijnders, F. Hubis, X. Zhai, N. Houlsby, D. Tran, M. Lucic, Revisiting the calibration of modern neural networks, in: Advances in Neural Information Processing Systems, volume 34, Curran Associates, Inc., 2021, pp. 15682–15694.

[2] X.-Y. Zhang, G.-S. Xie, X. Li, T. Mei, C.-L. Liu, A survey on learning to reject, Proceedings of the IEEE 111 (2023) 185–215.

[3] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, J. Davis, Machine learning with a reject option: A survey, Machine Learning 113 (2024) 3073–3110.

[4] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, On calibration of modern neural networks, in: International conference on machine learning, PMLR, 2017, pp. 1321–1330.

[5] L. Cosmides, J. Tooby, Are humans good intuitive statisticians after all? rethinking some conclusions from the literature on judgment under uncertainty, cognition 58 (1996) 1–73.

[6] F. Pedregosa, Hyperparameter optimization with approximate gradient, in: International conference on machine learning, PMLR, 2016, pp. 737–746.

[7] L. Franceschi, M. Donini, P. Frasconi, M. Pontil, Forward and reverse gradient-based hyperparameter optimization, in: International Conference on Machine Learning, PMLR, 2017, pp. 1165–1173.

[8] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, M. Pontil, Bilevel programming for hyperparameter optimization and meta-learning, in: International conference on machine learning, PMLR, 2018, pp. 1568–1577.

[9] N. Jain, P. Shenoy, Selective classification using a robust meta-learning approach, arXiv preprint arXiv:2212.05987 (2022).

[10] K. Q. Weinberger, L. K. Saul, Distance metric learning for large margin nearest neighbor classification., Journal of machine learning research 10 (2009).

[11] P. R. Mendes Júnior, R. M. De Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. De Almeida, O. A. Penatti, R. d. S. Torres, A. Rocha, Nearest neighbors distance ratio open-set classifier, Machine Learning 106 (2017) 359–386.

[12] H. Jiang, B. Kim, M. Guan, M. Gupta, To trust or not to trust a classifier, Advances in neural information processing systems 31 (2018).

[13] A. Mandelbaum, D. Weinshall, Distance-based confidence score for neural network classifiers, arXiv preprint arXiv:1709.09844 (2017).

[14] N. Papernot, P. McDaniel, Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning, arXiv preprint arXiv:1803.04765 (2018).

[15] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, PMLR, 2016, pp. 1050–1059.

[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural network, in: International conference on machine learning, PMLR, 2015, pp. 1613–1622.

[17] A. Kristiadi, M. Hein, P. Hennig, Being bayesian, even just a bit, fixes overconfidence in relu networks, in: International conference on machine learning, PMLR, 2020, pp. 5436–5446.

[18] C. Riquelme, G. Tucker, J. Snoek, Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling, arXiv preprint arXiv:1802.09127 (2018).

[19] Y. Xia, X. Cao, F. Wen, G. Hua, J. Sun, Learning discriminative reconstructions for unsupervised outlier removal, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1511–1519.

[20] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, T. Naemura, Classification-reconstruction learning for open-set recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4016–4025.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (2014) 1929–1958.

[22] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Advances in neural information processing systems 30 (2017).

[23] A. Bendale, T. E. Boult, Towards open set deep networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 1563–1572.

[24] C. De Stefano, C. Sansone, M. Vento, To reject or not to reject: that is the question-an answer in case of neural classifiers, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 30 (2000) 84–94.

[25] M. H. DeGroot, S. E. Fienberg, The comparison and evaluation of forecasters, Journal of the Royal Statistical Society: Series D (The Statistician) 32 (1983) 12–22.

[26] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, in: Proceedings of the 22nd international conference on Machine learning, 2005, pp. 625–632.

[27] B. Zadrozny, C. Elkan, Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers, in: Icml, volume 1, 2001, pp. 609–616.

[28] M. P. Naeini, G. Cooper, M. Hauskrecht, Obtaining well calibrated probabilities using bayesian binning, in: Proceedings of the AAAI conference on artificial intelligence, volume 29, 2015.

[29] J. Platt, et al., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, Advances in large margin classifiers 10 (1999) 61–74.

[30] M. Kull, T. Silva Filho, P. Flach, Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers, in: Artificial intelligence and statistics, PMLR, 2017, pp. 623–631.

[31] Y. Wang, L. Li, C. Dang, Calibrating classification probabilities with shape-restricted polynomial regression, IEEE transactions on pattern analysis and machine intelligence 41 (2019) 1813–1827.

[32] F. Pan, X. Ao, P. Tang, M. Lu, D. Liu, L. Xiao, Q. He, Field-aware calibration: a simple and empirically strong method for reliable probabilistic predictions, in: Proceedings of The Web Conference 2020, 2020, pp. 729–739.

[33] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 694–699.

[34] Y. Kwon, J.-H. Won, B. J. Kim, M. C. Paik, Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation, Computational Statistics & Data Analysis 142 (2020) 106816.

[35] J. Domke, Generic methods for optimization-based modeling, in: Artificial Intelligence and Statistics, PMLR, 2012, pp. 318–326.

[36] D. Maclaurin, D. Duvenaud, R. Adams, Gradient-based hyperparameter optimization through reversible learning, in: International conference on machine learning, PMLR, 2015, pp. 2113–2122.

[37] J. Ren*, X. Feng*, B. Liu*, X. Pan*, Y. Fu, L. Mai, Y. Yang, Torchopt: An efficient library for differentiable optimization, Journal of Machine Learning Research 24 (2023) 1–14.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: NIPS-W, 2017.

[39] C. Nugent, P. Cunningham, A case-based explanation system for black-box systems, Artif. Intell. Rev. 24 (2005) 163–178.

[40] K. Asadi, M. L. Littman, An alternative softmax operator for reinforcement learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 243–252.

[41] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2017).