Dual Accuracy-Quality-Driven Neural Network for Prediction Interval Generation

Giorgio Morales, *Member, IEEE*, and John W. Sheppard, *Fellow, IEEE*

***Abstract*— Accurate uncertainty quantification is necessary to enhance the reliability of deep learning (DL) models in real-world applications. In the case of regression tasks, prediction intervals (PIs) should be provided along with the deterministic predictions of DL models. Such PIs are useful or "high-quality (HQ)" as long as they are sufficiently narrow and capture most of the probability density. In this article, we present a method to learn PIs for regression-based neural networks (NNs) automatically in addition to the conventional target predictions. In particular, we train two companion NNs: one that uses one output, the target estimate, and another that uses two outputs, the upper and lower bounds of the corresponding PI. Our main contribution is the design of a novel loss function for the PI-generation network that takes into account the output of the target-estimation network and has two optimization objectives: minimizing the mean PI width and ensuring the PI integrity using constraints that maximize the PI probability coverage implicitly. Furthermore, we introduce a self-adaptive coefficient that balances both objectives within the loss function, which alleviates the task of fine-tuning. Experiments using a synthetic dataset, eight benchmark datasets, and a real-world crop yield prediction dataset showed that our method was able to maintain a nominal probability coverage and produce significantly narrower PIs without detriment to its target estimation accuracy when compared to those PIs generated by three state-of-the-art neural-network-based methods. In other words, our method was shown to produce higher quality PIs.**

***Index Terms*— Companion networks, deep regression, prediction intervals (PIs), uncertainty quantification.**

I. Introduction

DEEP learning has gained a great deal of attention due to its ability to outperform alternative machine learning methods in solving complex problems in a variety of domains. In conjunction with the availability of large-scale datasets and modern parallel hardware architectures (e.g., GPUs), convolutional neural networks (CNNs), as one popular deep learning (DL) technique, have attained unprecedented achievements in fields such as computer vision, speech recognition, natural language processing, medical diagnosis, and others [1].

Manuscript received 1 November 2022; revised 29 March 2023 and 9 August 2023; accepted 1 December 2023. This work was supported in part by the United States Department of Agriculture (USDA)-National Institute of Food and Agriculture (NIFA)-Agriculture and Food Research Initiative (AFRI) Food Security Program Coordinated Agricultural Project under Grant 2016-68004-24769 and in part by the USDA-Natural Resources Conservation Service (NRCS) Conservation Innovation Grant from the On-farm Trials Program under Award NR213A7500013G021. *(Corresponding author: John W. Sheppard.)*

The authors are with the Gianforte School of Computing, Montana State University, Bozeman, MT 59717 USA (e-mail: giorgiol.moralesluna@student.montana.edu; john.sheppard@montana.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3339470.

Digital Object Identifier 10.1109/TNNLS.2023.3339470

While the undeniable success of DL has impacted applications that are used on a daily basis, many theoretical aspects remain unclear, which is why these models are usually referred to as "black boxes" in the literature [2]. In addition, numerous reports suggest that current DL techniques typically lead to unstable predictions that can occur randomly and not only in worst-case scenarios [3]. As a consequence, they are considered unreliable for applications that deal with uncertainty in the data or in the underlying system, such as weather forecasting [4], electronic manufacturing [5], or precision agriculture [6]. Note that, in this context, reliability is defined as the ability for a model to work consistently across real-world settings [7].

One of the limitations of conventional neural networks (NNs) is that they only provide deterministic point estimates without any additional indication of their approximate accuracy [8]. Reliability and accuracy of the generated point predictions are affected by factors such as the sparsity of training data or target variables affected by probabilistic events [9]. One way to improve the reliability and credibility of such complex models is to quantify the uncertainty in the predictions they generate [10]. This uncertainty (σ_y^2) can be quantified using prediction intervals (PIs), which provide an estimate of the upper and the lower bounds within which a prediction will fall according to a certain probability [11]. Hence, the amount of uncertainty for each prediction is provided by the width of its corresponding PI. PIs account for two types of uncertainty: model uncertainty (σ_{model}^2) and data noise variance (σ_{noise}^2) [11], where $\sigma_y^2 = \sigma_{\text{model}}^2 + \sigma_{\text{noise}}^2$. Model uncertainty arises due to model selection, training data variance, and parameter uncertainty [12]. Data noise variance measures the variance of the error between observable target values and the outputs produced by the learned models.

Recently, some NN-based methods have been proposed to solve the PI generation problem [11], [12], [13], [14], [15], [16]. These methods aim to train NNs using loss functions that aim to balance at least two of the following three objectives: minimizing mean PI width, maximizing PI coverage probability, and minimizing the mean error of the target predictions. Although the aforementioned works have achieved promising results, there exist some limitations that need to be addressed. For instance, they rely on the use of deep ensembles; however, training several models may become impractical when applied to complex models and large datasets [17]. Furthermore, their performance is sensitive to the selection of multiple tunable hyperparameters whose values may differ substantially depending on the application. Therefore, fine-tuning an ensemble of deep NNs becomes a computationally expensive task. Finally, methods that generate

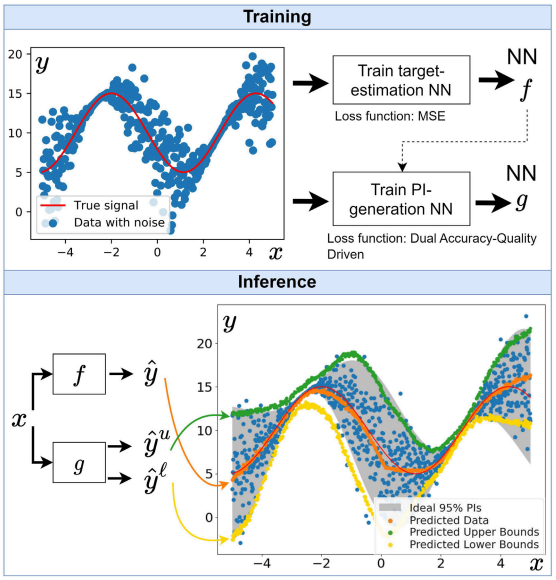

Fig. 1. Example of our PI-generation method on a synthetic dataset.

PI bounds and target estimations simultaneously have to deal with a trade-off between the quality of generated PIs and the accuracy of the target estimations.

Pearce et al. [12] coined the term high-quality (HQ) principle, which refers to the requirement that PIs be as narrow as possible while capturing some specified proportion of the predicted data points. Following this principle, we pose the PI generation problem for regression as a multiobjective optimization problem. In particular, our proposal involves training two NNs: one that generates accurate target estimations and one that generates narrow PIs (see Fig. 1).

The first NN is trained to minimize the mean squared error (MSE) of the target estimations. Our main contribution is the design of a loss function for the second NN that, besides the generated PI bounds and the target, considers the output of the first NN as an additional input. It minimizes the mean PI width and uses constraints to ensure the integrity of the generated PIs while implicitly maximizing the probability coverage (Section III-A). Our second contribution is a method that updates the coefficient that balances the two optimization objectives of our loss function automatically throughout training (Section III-C). Our method avoids generating unnecessarily wide PIs by using a technique that sorts the mini-batches at the beginning of each training epoch according to the width of the generated PIs (Section III-B). Then, we apply a Monte Carlo-based approach to account for the uncertainty of the generated upper and lower bounds (Section III-E). Finally, when compared to three state-of-the-art NN-based methods, we show that our method is able to produce PIs that maintain the target probability coverage while yielding better mean width without detriment to its target estimation accuracy (Section IV).

Our specific contributions are summarized as follows.

1) Our main contribution is a novel loss function called dual accuracy-quality-driven (DualAQD) used to train a PI-generation NN. It is designed to solve a multiobjective optimization problem: minimizing the mean PI width while ensuring PI integrity using constraints that maximize the probability coverage implicitly.
2) We present a new PI-generation framework that consists of two companion NNs: one that is trained to produce accurate target estimations, and another that generates HQ PIs; thus, avoiding the common trade-off between target estimation accuracy, and quality of PIs.
3) We introduce a self-adaptive coefficient that balances the two objectives of our DualAQD loss function. This differs from previous approaches that consider this balancing coefficient as a tunable hyperparameter with a fixed value throughout the training process.
4) We present a method called batch-sorting that sorts the mini-batches according to their corresponding PI width and, as such, avoids generating unnecessarily wide PIs.
5) Our method is shown to generate higher quality PIs and better reflects varying levels of uncertainty within the data than the compared methods.

II. Related Work

One of the more common approaches to uncertainty quantification for regression tasks is via Bayesian approaches, such as those represented by Bayesian NNs (BNNs), which model the NN parameters as distributions. As such, they have the advantage that they allow for a natural quantification of uncertainty. In particular, uncertainty is quantified by learning a posterior weight distribution [18], [19]. The inference process involves marginalization over the weights, which in general is intractable, and sampling processes such as Markov chain Monte Carlo (MCMC) can be computationally prohibitive. Thus, approximate solutions have been formulated using variational inference (VI) [20]. However, Wu et al. [21] argued that VI approaches are fragile since they require careful initialization and tuning. To overcome these issues, they proposed approximating moments in NNs to eliminate gradient variance. They also presented an empirical Bayes procedure for selecting prior variances automatically. Moreover, Izmailov et al. [22] discussed scaling BNNs to deep NNs by constructing low-dimensional subspaces of the parameter space. By doing so, they were able to apply elliptical slice sampling and VI, which struggle in the full parameter space. In addition, Lut et al. [23] presented a Bayesian-learning-based sparse stochastic configuration network that replaces the Gaussian distribution with a Laplace one as the prior distribution for output weights.

Despite the aforementioned improvements in Bayesian approaches, they still suffer from various limitations. Namely, the high dimensionality of the parameter space of deep NNs, including complex models such as CNNs, makes the cost of characterizing uncertainty over the parameters prohibitive [24]. Attempts to scale BNNs to deep NNs are considerably more expensive computationally than VI-based methods and have been scaled up to low-complexity problems only, such as MNIST [25]. Conversely, non-Bayesian methods do not require the use of initial prior distributions and biases to train the models [11]. Recent works have demonstrated that non-Bayesian approaches provide better or competitive uncertainty estimates than their Bayesian counterparts [11], [12], [26]. In addition, they are scalable to complex problems and can handle millions of parameters.

MC-Dropout was proposed by Gal and Ghahramani [8] to quantify model uncertainty in NNs. They cast dropout

training in deep NNs as approximate Bayesian inference in deep Gaussian processes. The method uses dropout repeatedly to select subsamples of active nodes in the network, turning a single network into an ensemble. Hence, model uncertainty is estimated by the sample variance of the ensemble predictions. MC-Dropout is not able to estimate PIs themselves, as it does not account for data noise variance. Therefore, Zhu and Laptev [27] proposed estimating PIs by quantifying the model uncertainty through MC-Dropout, coupled with estimating the data noise variance as the MSE calculated over an independent held-out validation set.

Recently, several non-Bayesian approaches have been proposed for approximate uncertainty quantification. Such approaches use models whose outputs provide estimations of the predictive uncertainty directly. For instance, Schupbach et al. [28] proposed a method that estimates confidence intervals in NN ensembles based on the use of U-statistics. Other techniques estimate PIs by using ensembles of feedforward networks [29] or stochastic configuration networks [30] and bootstrapping. Lakshminarayanan et al. [26] presented an ensemble approach based on the mean-variance estimation (MVE) method introduced by Nix and Weigend [31]. Here, each NN has two outputs: one that represents the mean (or target estimation) and the other that represents the variance of a normal distribution, which is used to quantify the data noise variance. Other approaches use models that generate PI bounds explicitly. Khrosavi et al. [11] proposed a lower upper bound estimation (LUBE) method that uses a NN and a loss function to minimize the PI width while maximizing the probability coverage using simulated annealing.

Similar approaches have attempted to optimize the LUBE loss function using methods such as genetic algorithms [13] and particle swarm optimization [14]. Pearce et al. [12] proposed a method called QD-Ens that consists of a quality-driven loss function similar to LUBE but that is compatible with gradient descent. Then, Salem et al. [16] proposed QD+ which is based on QD-Ens, which uses exactly the same two penalty functions to reduce the PI width and maximize the probability coverage. They used three-output NNs and included a third penalty term that aims to decrease the MSE of the target predictions and a fourth penalty term to enforce the point predictions to lay inside the generated PIs. In our work, we use only three penalty terms; the differences are explained in Section III-F. Finally, both QD-Ens and QD+ used an ensemble approach to estimate the model uncertainty while we use a Monte Carlo approach on a single network.

III. Proposed Methodology

A. DualAQD Loss Function

Let $\mathbf{X}^b = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a training batch with N samples where each sample $\mathbf{x}_i \in \mathbb{R}^z$ consists of z covariates. Furthermore, let $\mathbf{y}^b = \{y_1, \ldots, y_N\}$ be a set of corresponding target observations where $y_i \in \mathbb{R}$. We construct a NN regression model that captures the association between $\mathbf{X}^b$ and $\mathbf{y}^b$. More specifically, $f(\cdot)$ denotes the function computed by the NN, and $\boldsymbol{\theta}_f$ denotes its weights. Hence, given an input $\mathbf{x}_i$, $f(\mathbf{x}_i, \boldsymbol{\theta}_f)$ computes the target estimate $\hat{y}_i$. This network is trained to generate accurate estimates $\hat{y}_i$ with respect to y_i. We quantify this accuracy by calculating the MSE of the estimation $MSE_{\text{est}} = (1/N)\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$. Thus, f is conventionally optimized as follows:

$$\boldsymbol{\theta}_f = \underset{\boldsymbol{\theta}_f}{\operatorname{argmin}} \, \text{MSE}_{\text{est}}.$$

Once network $f(\cdot)$ is trained, we use a separate NN whose goal is to generate PIs for $\mathbf{y}^b$ given data $\mathbf{X}^b$. Let $g(\cdot)$ denote the function computed by this PI-generation NN, and $\boldsymbol{\theta}_g$ denotes its weights. Given an input $\mathbf{x}_i$, $g(\mathbf{x}_i, \boldsymbol{\theta}_g)$ generates its corresponding upper and lower bounds, $\hat{y}_i^u$ and $\hat{y}_i^\ell$, such that $[\hat{y}_i^\ell, \hat{y}_i^u] = g(\mathbf{x}_i, \boldsymbol{\theta}_g)$. Note that there is no assumption of $\hat{y}_i^\ell$ and $\hat{y}_i^u$ being symmetric with respect to the target estimate $\hat{y}_i$ produced by network $f(\cdot)$. We describe its optimization procedure below.

We say that a training sample $\mathbf{x}_i \in \mathbf{X}^b$ is covered (i.e., we set $k_i = 1$) if both the predicted value $\hat{y}_i$ and the target observation y_i fall within the estimated PI

$$k_i = \begin{cases} 1, & \text{if } \hat{y}_i^\ell < \hat{y}_i < \hat{y}_i^u \text{ and } \hat{y}_i^\ell < y_i < \hat{y}_i^u \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, using k_i, we define the PI coverage probability (PICP) for $\mathbf{X}^b$ as the percent of covered samples with respect to the batch size N: $\text{PICP} = \sum_{i=1}^{N} k_i/N$.

The HQ principle suggests that the width of the PIs should be minimized as long as they capture the target observation value. Thus, Pearce et al. [12] considered the mean PI width of captured points ($\text{MPIW}_{\text{capt}}$) as part of their loss function

$$\text{MPIW}_{\text{capt}} = \frac{1}{\epsilon + \sum_i k_i} \sum_{i=1}^{N} \left(\hat{y}_i^u - \hat{y}_i^\ell\right) k_i \tag{2}$$

where ϵ is a small number used to avoid dividing by zero. However, we argue that minimizing $\text{MPIW}_{\text{capt}}$ does not imply that the width of the PIs generated for the noncaptured samples will not decrease along with the width of the PIs generated for the captured samples.[1]

Furthermore, consider the case where none of the samples are captured by the PIs, as likely happens at the beginning of the training. Then, the penalty is minimum (i.e., $\text{MPIW}_{\text{capt}} = 0$). Hence, the calculated gradients of the loss function will force the weights of the NN to remain in the state where $\forall i, \; k_i = 0$, which contradicts the goal of maximizing PICP.

Instead of minimizing $\text{MPIW}_{\text{capt}}$ directly, we let

$$\text{PI}_{\text{pen}} = \frac{1}{N} \sum_{i=1}^{N} \left(|\hat{y}_i^u - y_i| + |y_i - \hat{y}_i^\ell|\right) \tag{3}$$

which we minimize instead. This function quantifies the width of the PI as the sum of the distance between the upper bound and the target and the distance between the lower bound and the target. We argue that PI_{pen} is more suitable than $\text{MPIW}_{\text{capt}}$

[1] We provide a toy example demonstrating this behavior in the following link https://github.com/GiorgioMorales/PredictionIntervals/blob/master/models/QD_toy_example.ipynb

given that it forces $\hat{y}_i^u$, y_i, and $\hat{y}_i^\ell$ to be closer together. For example, suppose that the following case is observed during the first training epoch: $y_i = 24$, $\hat{y}_i = 25$, $\hat{y}_i^u = 0.2$, and $\hat{y}_i^\ell = 0.1$. Then, $\text{MPIW}_{\text{capt}} = 0$ given that the target is not covered by the PI, while $\text{PI}_{\text{pen}} = 47.7$. As a result, PI_{pen} will penalize this state while $\text{MPIW}_{\text{capt}}$ will not. Thus, we define our first optimization objective as

$$\min_{\boldsymbol{\theta}_g} \mathcal{L}_1 = \min_{\boldsymbol{\theta}_g} \text{PI}_{\text{pen}}.$$

However, minimizing $\mathcal{L}_1$ is not enough to ensure the integrity of the PIs. Their integrity is given by the conditions that the upper bound must be greater than the target and the target estimate ($\hat{y}_i^u > y_i$ and $\hat{y}_i^u > \hat{y}_i$) and that the target and the target estimate, in turn, must be greater than the lower bound ($y_i > \hat{y}_i^\ell$ and $\hat{y}_i > \hat{y}_i^\ell$). Note that if the differences $(\hat{y}_i^u - y_i)$ and $(y_i - \hat{y}_i^\ell)$ are greater than the maximum estimation error within the training batch $\mathbf{X}^b$ (i.e., $(\hat{y}_i^u - y_i) > \max_i |\hat{y}_i - y_i|$ and $(\hat{y}_i^u - y_i) > \max_i |\hat{y}_i - y_i|$, $\forall i \in [1, \ldots, N]$), it is implied that all samples are covered ($k_i = 1$, $\forall i \in [1, \ldots, N]$).

Motivated by this, we include an additional penalty function to ensure PI integrity and maximize the number of covered samples within the batch simultaneously. Let us denote the mean differences between the PI bounds and the target estimates as $d_u = \sum_{i=1}^{N}(\hat{y}_i^u - y_i)/N$ and $d_\ell = \sum_{i=1}^{N}(y_i - \hat{y}_i^\ell)/N$. Let $\xi = \max_i |\hat{y}_i - y_i|$ denote the maximum distance between a target estimate and its corresponding target value within the batch ($\xi > 0$). From this, our penalty function is defined as

$$P = e^{\xi - d_u} + e^{\xi - d_\ell}. \tag{4}$$

Here, if the PI integrity is not met (i.e., $d_u < 0$ or $d_\ell < 0$), then their exponent magnitude becomes larger than ξ, producing a large penalty value. Moreover, these terms encourage both d_u and d_ℓ not only to be positive but also to be greater than ξ. This implies that the distance between the target y_i and any of its bounds will be larger than the maximum error within the batch, ξ, thus the target y_i will lie within the PI. From this, we define our second optimization objective as

$$\min_{\boldsymbol{\theta}_g} \mathcal{L}_2 = \min_{\boldsymbol{\theta}_g} P.$$

Then, our proposed DualAQD loss function is given by

$$\text{Loss}_{\text{DualAQD}} = \mathcal{L}_1 + \lambda \mathcal{L}_2 \tag{5}$$

where λ is a self-adaptive coefficient that controls the relative importance of $\mathcal{L}_1$ and $\mathcal{L}_2$. Hence, our multiobjective optimization problem can be expressed as

$$\boldsymbol{\theta}_g = \underset{\boldsymbol{\theta}_g}{\text{argmin}}\, \text{Loss}_{\text{DualAQD}}.$$

For simplicity, we assume that $f(\cdot)$ and $g(\cdot)$ have L layers and the same network architecture except for the output layer. Network $f(\cdot)$ is trained first. Then, weights $\boldsymbol{\theta}_g$ are initialized using weights $\boldsymbol{\theta}_f$ except for those of the last layer: $\boldsymbol{\theta}_g^{(0)}[1 : L-1] = \boldsymbol{\theta}_f[1 : L-1]$. Note, that, in general, DualAQD can use different network architectures for $f(\cdot)$ and $g(\cdot)$.

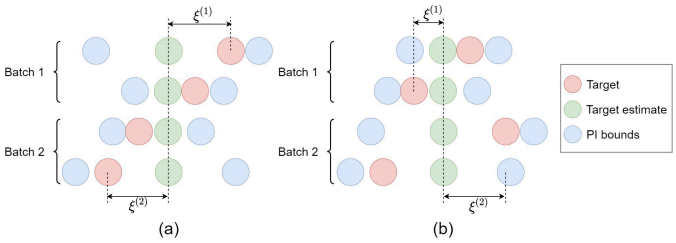

Fig. 2. $\mathcal{L}_3$ penalty calculation. (a) Without batch sorting. (b) With batch sorting.

B. Batch Sorting

The objective function $\mathcal{L}_2$ minimizes the term P [see (4)], forcing the distance between the target estimate of a sample and its PI bounds to be larger than the maximum absolute error within its corresponding batch. This term assumes that there exists a similarity among the samples within a batch. However, consider the case depicted in Fig. 2 where we show four samples that have been split randomly into two batches. In Fig. 2(a), the PIs of the second and third samples already cover their observed targets. Nevertheless, according to $\mathcal{L}_2$, these samples will yield high penalties because the distances between their target estimates and their PI bounds are less than $\xi^{(1)}$ and $\xi^{(2)}$, respectively, forcing their widths to increase unnecessarily.

For this reason, we propose a method called "batch sorting," which consists of sorting the training samples with respect to their corresponding generated PI widths after each epoch. By doing so, the batches will process samples with similar widths, avoiding unnecessary widening. For example, in Fig. 2(b), the penalty terms are low given that $d_u^{(1)}, d_\ell^{(1)} > \xi^{(1)}$ and $d_u^{(2)}, d_\ell^{(2)} > \xi^{(2)}$. Note that, during testing, the PI generated for a given sample is independent of other samples and, as such, batch sorting becomes unnecessary during inference.

C. Self-Adaptive Coefficient λ

The coefficient λ of (5) balances the two optimization objectives $\mathcal{L}_1$ and $\mathcal{L}_2$. In this section, we propose that, instead of λ being a tunable hyperparameter with a fixed value throughout training, it should be adapted throughout the learning process automatically.

Typically, the PICP value improves as long as the MPIW value increases; however, extremely wide PIs are not useful. We usually aim to obtain PIs with a nominal probability coverage no greater than $(1 - \alpha)$. A common value for the significance level α is 0.05, in which case we say that we are 95% confident that the target value will fall within the PI.

Let $\text{PICP}_{\text{train}}^{(t)}$ denote the PICP value calculated on the training set $\mathbf{X}_{\text{train}}$ after the tth training epoch. If $\text{PICP}_{\text{train}}^{(t)}$ is below the confidence target $(1-\alpha)$, more relative importance should be given to the objective $\mathcal{L}_2$ that enforces PI integrity (i.e., λ should increase). Likewise, if $\text{PICP}_{\text{train}}^{(t)}$ is higher than $(1 - \alpha)$, more relative importance should be given to the objective $\mathcal{L}_1$ that minimizes MPIW (i.e., λ should decrease).

We formalize this intuition by defining the cost $\mathcal{C}$ that quantifies the distance from $\text{PICP}_{\text{train}}^{(t)}$ to the confidence target $(1-\alpha)$: $\mathcal{C} = (1-\alpha) - PICP_{\text{train}}^{(t)}$. Then, we propose to increase or decrease λ proportionally to the cost function $\mathcal{C}$ after each

Algorithm 1 DualAQD Method

function TRAINNNWITHDUALAQD($\mathbf{X}_{train}$, Y_{train}, f, g, α, η)
  $\lambda \leftarrow 1$
  for each $t \in$ range(1, $maxEpochs$) **do**
    if $t > 1$ **then**
      $Batches \leftarrow batchSorting(\mathbf{X}_{train}, Y_{train}, widths)$
    else
      $Batches \leftarrow shuffle(\mathbf{X}_{train}, Y_{train})$
    for each $batch \in Batches$ **do**
      $\mathbf{x}, y \leftarrow batch$
      $\hat{y} \leftarrow f(\mathbf{x})$
      $\hat{y}^u, \hat{y}^\ell \leftarrow g(\mathbf{x})$
      $loss \leftarrow DualAQD(\lambda, y, \hat{y}, \hat{y}^u, \hat{y}^\ell)$
      $update(g, loss)$
    $PICP^{(t)}_{train}, widths^{(t)} \leftarrow metrics(\mathbf{X}_{train}, Y_{train})$
    // Update coefficient λ
    $\mathcal{C} \leftarrow ((1-\alpha) - PICP^{(t)}_{train})$
    $\lambda = \lambda + \eta \cdot \mathcal{C}$
  return g

training epoch as follows (see Algorithm 1):

$$\lambda^{(t)} = \lambda^{(t-1)} + \eta \cdot \mathcal{C} \tag{6}$$

where $\lambda^{(t)}$ is the value of the coefficient λ at the tth iteration (we consider that $\lambda^{(0)} = 1$), and η is a tunable scale factor.

Note that Algorithm 1 takes as inputs the data $\mathbf{X}_{\text{train}}$ and corresponding targets Y_{train} as well as the trained prediction network f, the untrained network g, the significance level α, and the scale factor η. Function `batchSorting`($\mathbf{X}_{\text{train}}, Y_{\text{train}}$, widths$^{(t-1)}$) returns a list of batches sorted according to the PI widths generated during the previous training epoch (see Section III-B). Function `DualAQD`($\lambda, y, \hat{y}, \hat{y}^u, \hat{y}^\ell$) represents the DualAQD loss function [see (5)] while `update`(g, loss) encompasses the conventional backpropagation and gradient descent processes used to update the weights of network g. Furthermore, function `metrics`($\mathbf{X}_{\text{train}}, Y_{\text{train}}$) passes $\mathbf{X}_{\text{train}}$ through g to generate the corresponding PIs and their widths, and to calculate compares the output to Y_{train} to calculate the $\text{PICP}^{(t)}_{\text{train}}$ value using Y_{train}.

D. Parameter and Hyperparameter Selection

We train a NN on the training set $\mathbf{X}_{\text{train}}$ during T epochs using $\text{Loss}_{\text{DualAQD}}$ as the loss function. After the tth training epoch, we calculate the performance metrics $z_t = \{\text{PICP}^{(t)}_{\text{val}}, \text{MPIW}^{(t)}_{\text{val}}\}$ on the validation set $\mathbf{X}_{\text{val}}$. Thus, we consider that the set of optimal weights of the network, θ_g, will be those that maximize performance on the validation set. The remaining question is what are the criteria to compare two solutions z_i and z_j.

Taking this criterion into account, we consider that a solution z_i dominates another solution z_j ($z_i \preceq z_j$) if.

1) $\text{PICP}^{(i)}_{\text{val}} > \text{PICP}^{(j)}_{\text{val}}$ and $\text{PICP}^{(i)}_{\text{val}} \leq (1-\alpha)$.
2) $\text{PICP}^{(i)}_{\text{val}} == \text{PICP}^{(j)}_{\text{val}} < (1-\alpha)$ and $\text{MPIW}^{(i)}_{\text{val}} < \text{MPIW}^{(j)}_{\text{val}}$.
3) $\text{PICP}^{(i)}_{\text{val}} \geq (1-\alpha)$ and $\text{MPIW}^{(i)}_{\text{val}} < \text{MPIW}^{(j)}_{\text{val}}$.

In other words, if $\alpha = 0.05$, we seek a solution whose PICP_{val} value is at least 95%. After exceeding this value, a solution z_i is said to dominate another solution z_j only if it produces narrower PIs.

We use a grid search to tune the hyperparameter η for training [see (6)]. For each value, we train a NN using ten-fold cross-validation and calculate the average performance metrics on the validation sets. Then, the hyperparameters are selected using the dominance criteria explained above.

E. PI Aggregation Using MC-Dropout

In Section I, we explained that both the model uncertainty (σ^2_{model}) and the data noise variance (σ^2_{noise}) have to be taken into account when generating PIs. A model trained using $\text{Loss}_{\text{DualAQD}}$ generates PI estimates based on the training data; that is, it accounts for σ^2_{noise}. However, we still need to quantify the uncertainty of those estimates due to σ^2_{model}.

Unlike previous work that used explicit NN ensembles to quantify σ^2_{model} [12], [26], we propose to use a Monte Carlo-based approach. Specifically, we use MC-Dropout [32], which consists of using dropout layers that ignore each neuron of the network according to some probability or dropout rate. Then, during each forward pass with active dropout layers, a slightly different network architecture is used and, as a result, a slightly different prediction is obtained. According to Gal and Ghahramani [8], this process can be interpreted as a Bayesian approximation of the Gaussian process.

Our approach consists of using M forward passes through the network with active dropout layers. Given an input $\mathbf{x}_i$, the estimates $\hat{y}^{(m)}_i$, $\hat{y}^{u(m)}_i$, and $\hat{y}^{\ell(m)}_i$ are obtained at the mth iteration. Hence, the expected target estimate $\bar{y}_i$, the expected upper bound $\bar{y}^u_i$, and the expected lower bound $\bar{y}^\ell_i$ are calculated as: $\bar{y}_i = (1/M)\sum_{m=1}^{M} \hat{y}^{(m)}_i$, $\bar{y}^u_i = (1/M)\sum_{m=1}^{M} \hat{y}^{u(m)}_i$, $\bar{y}^\ell_i = (1/M)\sum_{m=1}^{M} \hat{y}^{\ell(m)}_i$.

F. Comparison to QD-Ens and QD+

Here, we consider the differences between our method (DualAQD) and the two methods QD-Ens [12] and QD+ [16]. For reference, we include the loss functions used by QD-Ens and QD+

$$\begin{aligned}
\text{Loss}_{\text{QD}} &= \text{MPIW}_{\text{capt}} + \delta \frac{N}{\alpha(1-\alpha)} \max(0, (1-\alpha) - \text{PICP})^2 \\
\text{Loss}_{\text{QD+}} &= (1-\lambda_1)(1-\lambda_2)\text{MPIW}_{\text{capt}} \\
&\quad + \lambda_1(1-\lambda_2)\max(0, (1-\alpha) - \text{PICP})^2 + \lambda_2 \text{MSE}_{\text{est}} \\
&\quad + \frac{\xi}{N}\sum_{i=1}^{N}\left[\max\left(0, \left(\hat{y}^u_i - \hat{y}_i\right)\right) + \max\left(0, \left(\hat{y}_i - \hat{y}^\ell_i\right)\right)\right]
\end{aligned}$$

where δ, λ_1, λ_2, and ξ are hyperparameters used by QD-Ens and QD+ to balance the learning objectives. The differences compared to our method are listed in order of importance from highest to lowest as follows.

1) QD-Ens and QD+ use objective functions that maximize PICP directly aiming to a goal of $(1-\alpha)$ at the batch level. We maximize PICP indirectly through $\mathcal{L}_2$, which encourages the model to produce PIs that cover as many training points as possible. This is achieved by producing PIs whose widths are larger than the maximum absolute error within each training batch. Then, the optimal weights of the network are selected as those that produce a coverage probability on the validation set of at least $(1-\alpha)$.

2) Note that PICP is not directly differentiable as it involves counting the number of samples that lay within the predicted PIs. However, QD-Ens and QD+ force its differentiation by including a sigmoid operation and a softening factor (i.e., an additional hyperparameter). On the other hand, the loss functions of DualAQD are already differentiable.
3) Our objective $\mathcal{L}_1$ minimizes PI_{pen}, which is a more suitable penalty function than $\text{MPIW}_{\text{capt}}$ (see Section III-A).
4) Our objective $\mathcal{L}_2$ maximizes PICP and ensures PI integrity simultaneously. QD+ uses a truncated linear constraint and a separate function to maximize PICP.
5) NN-based PI generation methods aim to balance three objectives: 1) accurate target prediction; 2) generation of narrow PIs; and 3) high coverage probability. QD-Ens uses a single coefficient δ within its loss function that balances objectives 2) and 3) and does not optimize objective 1) explicitly, while QD+ uses three coefficients λ_1, λ_2, and ξ to balance the three objectives. All of the coefficients are tunable hyperparameters. Our loss function, $\text{Loss}_{\text{DualAQD}}$, uses a balancing coefficient whose value is not fixed but is adapted throughout the training process using a single hyperparameter (i.e., the scale factor η).
6) Our approach uses two companion NNs $f(\cdot)$ and $g(\cdot)$ that optimize objective 1) and objectives 2) and 3), respectively, to avoid the trade-off between them. Conversely, the other approaches optimize a single NN architecture.
7) We use MC-Dropout to estimate the model uncertainty. By doing so, we need to train only a single model instead of using an explicit ensemble of models, as in QD-Ens and QD+. Also, QD+ requires fitting a split normal density function [33] for each data point to aggregate the PIs produced by the ensemble, thus increasing the complexity of the learning process.

IV. Experiments

A. Experiments With Synthetic Data

Previous approaches have been tested on datasets with similar uncertainty levels across all their samples, or on synthetic datasets with a single region of low uncertainty surrounded by a gradual increase of noise. This is a limitation as it does not allow testing the ability of the PI's to adapt to rapid changes of uncertainty within the data. Therefore, we test all of the methods on a more challenging synthetic dataset with more fluctuations and extreme levels of uncertainty. The code is available at https://github.com/GiorgioMorales/PredictionIntervals.

We created a synthetic dataset with varying PI widths that consists of a sinusoid with Gaussian noise. Specifically, the dataset contains 1000 points generated using the equation $y(x) = 5\cos(x) + 10 + \epsilon$, where $x \in [-5, 5]$ and ϵ is Gaussian noise whose magnitude depends on x: $\epsilon = (2\cos(1.2\,x) + 2)\,v$ where $v \sim \mathcal{N}(0, 1)$. For these experiments, we trained a feed-forward NN with two hidden layers, each with 100 nodes with ReLU activation. A 5×2-fold cross-validation design was used to train and evaluate all networks.

TABLE I

PI Metrics MSE_{val}, MPIW_{val}, PICP_{val}, and $\text{PI}_{\delta\,\text{val}}$ Evaluated on the Synthetic Dataset Using 5×2 Cross-Validation

Method	MSE_{val}	$MPIW_{val}$	$PICP_{val}(\%)$	$PI_{\delta_{val}}$
DualAQD	5.27 ± 0.27	7.30 ± 0.29	95.5 ± 0.48	1.52 ± 0.13
DualAQD_noBS	5.27 ± 0.27	9.16 ± 0.35	96.3 ± 0.77	3.08 ± 0.19
QD+	5.28 ± 0.29	8.56 ± 0.14	95.5 ± 0.31	3.12 ± 0.24
QD-Ens	5.31 ± 0.26	10.17 ± 0.79	94.0 ± 1.57	4.88 ± 0.17
MC-Dropout-PI	5.22 ± 0.30	9.31 ± 0.27	93.3 ± 0.63	5.04 ± 0.08

Knowing the probability distribution of the noise at each position x allows us to calculate the ideal 95% PIs ($\alpha = 0.05$), $[y^u, y^\ell]$, as follows:

$$y^u(x) = y(x) + 1.96\,\epsilon, \quad \text{and } y^\ell(x) = y(x) - 1.96\,\epsilon$$

where 1.96 is the approximate value of the 95% confidence interval of the normal distribution. Therefore, we define a new metric we called PI_δ that sums the absolute differences between the estimated bounds and the ideal 95% bounds for all the samples within a set $\mathbf{X}$

$$\text{PI}_\delta = \frac{1}{|\mathbf{X}|}\sum_{x\in\mathbf{X}}\left(|y^u(x) - \hat{y}^u(x)| + |y^\ell(x) - \hat{y}^\ell(x)|\right).$$

We compared the performance of DualAQD using batch sorting and without using batch sorting (denoted as "DualAQD_noBS" in Table I). All networks were trained using a fixed mini-batch size of 16 and the Adadelta optimizer. Table I gives the average performance for the metrics calculated on the validation sets, MSE_{val}, MPIW_{val}, PICP_{val}, and $\text{PI}_{\delta\,\text{val}}$, and corresponding standard deviations.

We also compared our DualAQD PI generation methodology to three other NN-based methods: QD+ [16], QD-Ens [12], and a PI generation method based on MC-Dropout alone [27] (denoted MC-Dropout-PI). For the sake of consistency and fairness, we used the same configuration (i.e., network architecture, optimizer, and batch size) for all the networks trained in our experiments. In our preliminary experiments, for the case of QD+, QD-Ens, and MC-Dropout-PI, we found that batch sorting either helped to improve their performance or there was no significant change. Thus, for the sake of fairness and consistency, we decided to use batch sorting for all compared methods. In addition, we tested Dropout rates between 0.1 and 0.5. The obtained results did not indicate a statistically significant difference; thus, we used a Dropout rate of 0.1 for all networks and datasets.

Note that the only difference between the network architecture used by the four methods is that QD+ requires three outputs, QD-Ens requires two (i.e., the lower and upper bounds), and MC-Dropout-PI requires one. For DualAQD and MC-Dropout-PI, we used $F = 100$ forward passes with active dropout layers. For QD+ and QD-Ens, we used an ensemble of five networks and a grid search to choose the hyperparameter values. Fig. 3 shows the PIs generated by the four methods from the first validation set together with the ideal 95% PIs.

B. Benchmarking Experiments

We experimented with eight open-access datasets from the UC Irvine Machine Learning Repository [34]. Note that even though our experiments use scalar and 2-D regression tasks

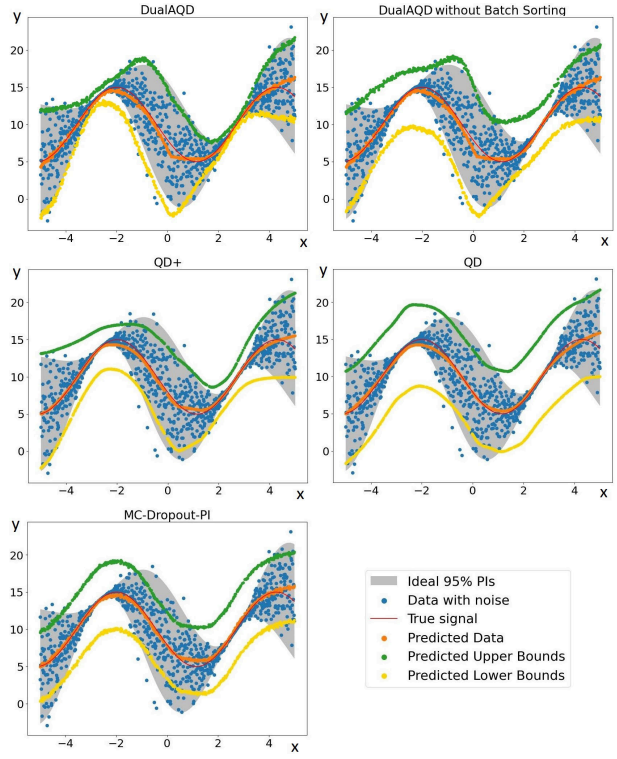

Fig. 3. Performance of PI generation methods on the synthetic dataset.

(Section IV-C), our proposed method can be extended to other tasks such as classification. For each dataset, we used a feed-forward NN whose architecture was the same as that described in Section IV-A. We used ten-fold cross-validation to train and evaluate all networks. Table II gives the average performance for the metrics calculated on the validation sets, MSE_{val}, MPIW_{val}, and PICP_{val}, and corresponding standard deviations. We applied z-score normalization (mean equal to 0 and standard deviation equal to 1) to each feature in the training set while the exact same scaling was applied to the features in the validation and test sets. Likewise, min-max normalization was applied to the response variable; however, Table II shows the results after rescaling to the original scale. Similar to Section IV-A, all networks were trained using a fixed mini-batch size of 16, except for the Protein and Year datasets that used a mini-batch size of 512 due to their large size.

The bold entries in Table II indicate the method that achieved the lowest average MPIW_{val} value and that its difference with respect to the values obtained by the other methods is statistically significant according to a paired t-test performed at the 0.05 significance level. The results obtained by DualAQD were significantly narrower than the compared methods while having similar MSE_{val} and PICP_{val} of at least 95%. Furthermore, Fig. 4 depicts the distribution of the scores achieved by all the compared methods on all the datasets, where the line through the center of each box indicates the median F1 score, the edges of the boxes are the 25th and 75th percentiles, whiskers extend to the maximum and minimum points (not counting outliers), and outlier points are those past the end of the whiskers (i.e., those points greater than $1.5 \times$ IQR plus the third quartile or less than $1.5 \times$ IQR minus the first quartile, where IQR is the inter-quartile range).

TABLE II

PI METRICS MSE_{val}, MPIW_{val}, AND PICP_{val} EVALUATED ON THE BENCHMARK DATASETS USING TEN-FOLD CROSS-VALIDATION

Dataset	Metric	DualAQD	QD+	QD-Ens	MC-Dropout-PI
Boston	$MPIW_{val}$	**9.99±2.26**	12.14±2.05	16.13±0.67	12.52±2.28
	MSE_{val}	8.91±3.90	11.91±5.24	15.29±5.07	8.94±3.87
	$PICP_{val}(\%)$	95.0±1.6	95.6±1.9	97.2±1.3	96.0±0.9
Concrete	$MPIW_{val}$	**15.72±1.42**	18.57±2.06	25.42±1.30	20.52±1.74
	MSE_{val}	22.45±4.79	26.65±8.02	29.30±5.25	22.71±4.96
	$PICP_{val}(\%)$	95.2±0.5	95.2±1.3	97.9±1.6	95.7±1.2
Energy	$MPIW_{val}$	**1.41±0.12**	2.94±0.05	10.99±1.47	3.81±0.21
	MSE_{val}	0.25±0.05	0.31±0.08	0.35±0.25	0.26±0.05
	$PICP_{val}(\%)$	96.5±0.6	99.0±1.0	100.0±0.0	99.5±0.6
Kin8nm	$MPIW_{val}$	**0.280±0.01**	0.311±0.01	0.502±0.01	0.336±0.01
	MSE_{val}	0.005±0.00	0.007±0.00	0.009±0.00	0.005±0.00
	$PICP_{val}(\%)$	95.1±0.1	96.6±0.4	98.5±0.3	97.5±0.4
Power	$MPIW_{val}$	**14.60±0.35**	15.31±0.44	27.57±1.54	16.08±0.63
	MSE_{val}	15.23±1.34	16.43±1.34	17.14±1.11	15.26±1.31
	$PICP_{val}(\%)$	95.2±0.1	95.7±0.3	99.6±0.2	96.4±0.5
Protein	$MPIW_{val}$	**13.02±0.26**	**13.05±0.14**	15.79±0.24	15.95±0.20
	MSE_{val}	14.79±0.40	17.51±0.59	18.35±0.87	15.05±0.42
	$PICP_{val}(\%)$	95.0±0.1	95.4±0.4	95.1±0.5	94.8±0.1
Yacht	$MPIW_{val}$	**1.56±0.42**	4.10±0.17	10.99±1.47	4.74±1.20
	MSE_{val}	0.51±0.53	0.72±0.70	0.35±0.25	0.53±0.54
	$PICP_{val}(\%)$	97.1±0.9	98.4±2.2	100.0±0.0	100.0±0.0
Year	$MPIW_{val}$	**29.68±0.29**	32.68±0.25	37.03±0.13	34.25±0.16
	MSE_{val}	73.26±0.76	104.8±8.1	78.12±0.87	73.13±0.69
	$PICP_{val}(\%)$	95.1±0.1	95.4±0.9	37.03±0.1	93.82±0.0

Note that even though QD-Ens uses only one hyperparameter (see Section III-F), it is more sensitive to small changes. For example, a hyperparameter value of $\delta = 0.021$ yielded poor PIs with $\text{PICP}_{\text{val}} < 40\%$ while a value of $\delta = 0.02105$ yielded too wide PIs with $\text{PICP}_{\text{val}} < 100\%$. For this reason, the hyperparameter δ of the QD-Ens approach was chosen manually while the scale factor η of DualAQD was chosen using a grid search with values $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. Fig. 5 shows the difference between the learning curves obtained during one iteration of the cross-validation for the Power dataset using two different η values (i.e., $\eta = 0.01$ and $\eta = 0.1$). The dashed lines indicate the training epoch at which the optimal weights $\boldsymbol{\theta}_g$ were selected according to the dominance criteria explained in Section III-D. On the other hand, the hyperparameters λ_1 and λ_2 of QD+ were chosen using a random search since it requires significantly higher training and execution time.

C. PIs for Crop Yield Prediction

We assert our approach is general in applicability. To test this assertion, we decided to experiment with a difficult, real-world application of 2-D regression using spatially correlated data to convey the usefulness of our method. Specifically, we focused on the crop yield prediction problem, which has an important impact on society and is one of the main tasks of precision agriculture. Accurate and reliable crop yield prediction, along with careful uncertainty management strategies, enables farmers to make informed management decisions, such as determining the nitrogen fertilizer rates needed in specific regions of their fields to maximize profit while minimizing environmental impact [35].

We use an early-yield prediction dataset of winter wheat we curated and presented in a previous work [36]. The early-yield prediction is posed as a regression problem where

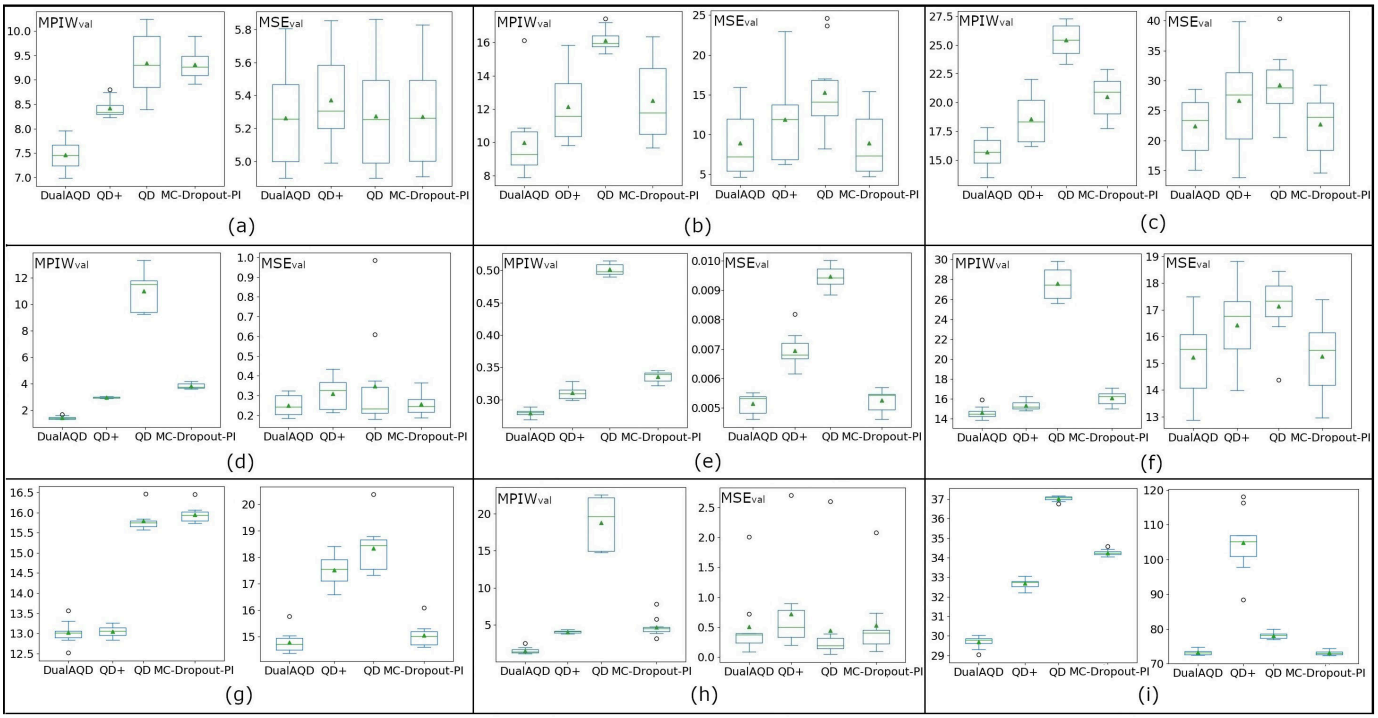

Fig. 4. Box plots of the MPIW_{val} and MSE_{val} scores of DualAQD, QD+, QD-Ens, and MC-Dropout-PI PI generation methods on the synthetic and benchmarking datasets. (a) Synthetic. (b) Boston. (c) Concrete. (d) Energy. (e) Kin8nm. (f) Power. (g) Protein. (h) Yacht. (i) Year.

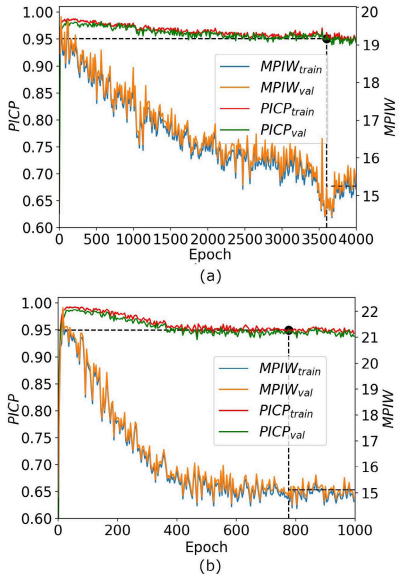

Fig. 5. MPIW and PICP learning curves obtained for the Power dataset using DualAQD. (a) $\eta = 0.01$. (b) $\eta = 0.1$.

the explanatory variables are represented by a set of eight features obtained during the growing season (March). These features consist of nitrogen rate applied, precipitation, slope, elevation, topographic position index (TPI), aspect, and two backscattering coefficients obtained from synthetic aperture radar (SAR) images from Sentinel-I. The response variable corresponds to the yield value in bushels per acre (bu/ac), measured during the harvest season (August). In other words, the data acquired in March is used to predict crop yield values in August of the same year.

The yield prediction problem requires 2-D inputs and 2-D outputs. As such, it can be viewed as a 2-D regression task. To tackle this problem, we trained a CNN using the Hyper3DNetReg 3-D–2-D network, architecture we presented in [36], which was specifically designed to predict the yield values of small spatial neighborhoods of a field simultaneously. We then modified this architecture to produce three output patches of 5×5 pixels (i.e., the estimated yield patch and two patches containing the upper and lower bounds of each pixel, respectively) instead of one.

For our experiments, we used data collected from three winter wheat fields, which we refer to as "A," "B," and "C," respectively. Three crop years of data were collected for each field. The information from the first two years was used to create the training and validation sets (90% of the data is used for training and 10% for validation). The four methods, AQD, QD+, QD-Ens, and MC-Dropout-PI, were compared using the results from the test set of each field, which consists of data from the last observed year and whose ground-truth yield map is denoted as Y. The test set was used to generate a predicted yield map of the entire field, $\hat{Y}$, and its corresponding lower and upper bounds, $\hat{Y}_L$ and $\hat{Y}_U$, respectively.

Fig. 6 shows the ground-truth yield map for field "A" (darker colors represent lower yield values) along with the uncertainty maps obtained by the four compared methods and their corresponding PICP and MPIW values. Field "A" is used as a representative field for presenting our results, since we obtained similar results on the other fields. Here, we define the uncertainty map $U = \hat{Y}^u - \hat{Y}^\ell$ as a map that contains the PI width of each point of the field (darker colors represent lower PI width and thus lower uncertainty). That is, the wider the PI of a given point, the more uncertain its yield prediction.

We used four metrics to assess the behavior of the four methods (Table III). First, we calculated the root mean square error ($\text{RMSE}_{\text{test}}$) between the ground-truth yield map Y and the estimated yield map $\hat{Y}$. Then, we considered the mean PI

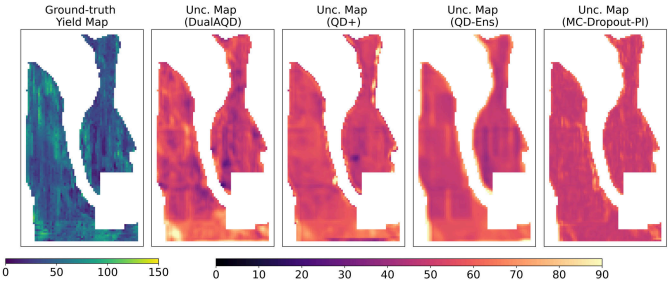

Fig. 6. Uncertainty maps comparison for field A.

TABLE III
PI METRICS $\text{RMSE}_{\text{test}}$, $\text{MPIW}_{\text{test}}$, $\text{PICP}_{\text{test}}$, AND μ EVALUATED ON THE YIELD PREDICTION DATASETS

Field	Method	$RMSE_{test}$	$MPIW_{test}$	$PICP_{test}$ (%)	μ
A	DualAQD	15.44	53.75	92.8	**.350**
	QD+	17.73	54.27	89.5	.397
	QD-Ens	15.55	53.99	92.3	.359
	MC-Dropout-PI	15.27	51.68	91.8	.355
B	DualAQD	11.16	43.45	94.9	**.221**
	QD+	11.83	50.17	93.7	.261
	QD-Ens	12.95	73.09	95.6	.306
	MC-Dropout-PI	10.83	47.18	94.4	.241
C	DualAQD	18.48	59.96	96.6	**.279**
	QD+	22.27	62.02	93.9	.336
	QD-Ens	17.75	39.93	63.8	.490
	MC-Dropout-PI	17.15	50.61	89.3	.349

width ($\text{MPIW}_{\text{test}}$) and PI probability coverage ($\text{PICP}_{\text{test}}$). Note that k-fold or $k \times 2$ cross-validation cannot be used in this experimental setting. Thus, to help us explain the advantages of our method over the others in the context of the HQ principle, we introduce a new metric that summarizes the $\text{MPIW}_{\text{test}}$ and $\text{PICP}_{\text{test}}$ metrics shown in Table III. Let $\overline{\text{MPIW}}_{\text{test}}$ represent the mean PI width after min-max normalization using as upper bound the maximum $\text{MPIW}_{\text{test}}$ value among the four methods in each field. Let μ_ω denote the weighted geometric mean between $\overline{\text{MPIW}}_{\text{test}}$ and $(1 - \text{PICP}_{\text{test}})$ (i.e., the complement of the PI coverage probability) with $\omega \in [0, 1]$ being the relative importance between both terms. Then,

$$\mu_\omega = (\overline{\text{MPIW}}_{\text{test}})^{\omega}(1 - \text{PICP}_{\text{test}})^{(1-\omega)}.$$

According to the HQ principle that aims to obtain narrow PIs and high probability coverage, low μ_ω values are preferable when comparing the performance of different PI-generation methods. Fig. 7 shows the comparison of the μ_ω metric obtained for each method on the three tested fields for different ω values. In order to summarize the behavior shown in Fig. 7 into a single metric, we calculated the integral $\mu = \int_0^1 \mu_\omega \, d\omega$. Since we seek to obtain low μ_ω values for various ω, low μ values are preferable. Bold entries in Table III indicate the method with the lowest μ.

V. DISCUSSION

Our loss function $\text{Loss}_{\text{DualAQD}}$ was designed to minimize the estimation error and produce narrow PIs simultaneously while using constraints that maximize the coverage probability inherently. From Tables I and II, we note that DualAQD consistently produced significantly narrower PIs than the compared methods, according to the paired t-test performed at the 0.05 significance level, except for the Protein dataset, where QD+ obtained comparable PI widths. Simultaneously,

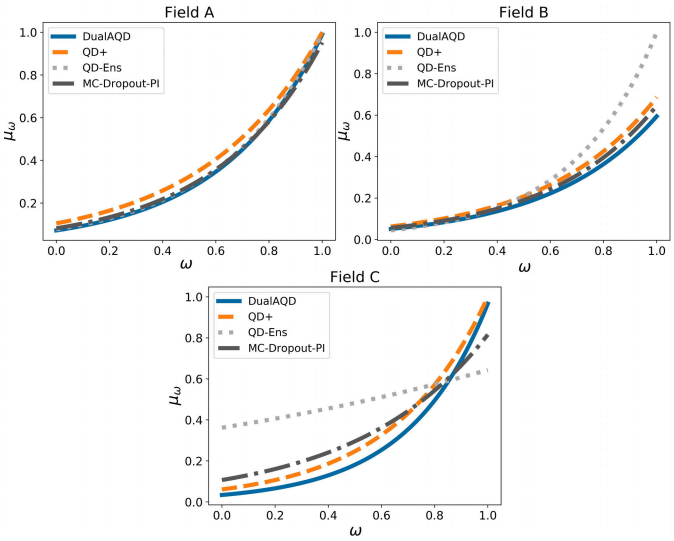

Fig. 7. μ_ω versus ω comparison on yield prediction datasets.

we yielded PICP_{val} values of at least 95% and better or comparable MSE_{val} values. In addition, the $\text{PI}_{\delta\,\text{val}}$ values reported in Table I demonstrate that DualAQD is the method that best adapted to the highly varying uncertainty levels of our synthetic dataset. Thus, the PI bounds generated by DualAQD were the closest to the ideal 95% PIs.

Notice that DualAQD obtains lower MSE_{val} values than QD+ consistently despite the fact that QD+ also includes an objective function that minimizes the error of the target predictions. The reason is that our method uses a NN [i.e., $f(\cdot)$] that is specialized in generating accurate target predictions, and its optimization objective does not compete with others. Conversely, QD+ uses a loss function that balances four objective functions: minimizing the PI widths, maximizing PI coverage probability, minimizing the target prediction errors, and ensuring PI integrity. The NN used by QD-Ens, on the other hand, only generates the upper and lower bounds of the PIs. The target estimate is then calculated as the central point between the PI bounds. As a consequence of not using a NN specialized in minimizing the target prediction error, QD-Ens achieved the worst MSE_{val} values of the compared methods, except for the Year dataset.

It is worth mentioning that one of the advantages of using DualAQD over QD+ and QD-Ens is that we achieved better PIs while requiring less computational complexity. That is, our method requires training only two NNs and uses MC-Dropout to account for the model uncertainty while QD+ and QD-Ens require training ensembles of five NNs. In addition, QD+ requires extra complexity given that it uses a split normal aggregation method that involves an additional fitting process for each data point during testing. Note that using deep ensembles of M models is expected to perform better or similar to MC-Dropout when using M forward passes [37]. In other words, using an ensemble of five NNs, as QD and QD+ do, is expected to perform better than using five forward passes through the NN using MC-Dropout. Nevertheless, during inference, we are able to perform not only five but 100 passes through the NN without significantly adding computationally cost. Our method becomes more practical in the sense that, even when it uses the rough estimates of model

uncertainty provided by MC-Dropout, it is still able to generate significantly higher-quality PIs.

In Fig. 5, we see the effect of using different scale factors η to update the balancing coefficient λ of $\text{Loss}_{\text{DualAQD}}$. Notice that DualAQD produced wide PIs at the beginning of the training process in order to ensure PI integrity; as a consequence, the $\text{PICP}_{\text{train}}$ and PICP_{val} values improved drastically. Once the generated PIs were wide enough to cover most of the samples in the training set (i.e., $\text{PICP}_{\text{train}} \approx 1$), DualAQD focused on reducing the PI widths until $\text{PICP}_{\text{train}}$ reached the nominal probability coverage α. The rate at which PICP and MPIW were reduced was determined by the scale factor η.

Furthermore, Fig. 5(a) ($\eta = 0.01$) and Fig. 5(b) ($\eta = 0.1$) show that both models converged to a similar MPIW_{val} value ($\sim$15) despite having improved at different rates. It is worth noting that we did not find a statistical difference between the results produced by the different η values that were tested on all the datasets (i.e., $\eta \in [0.001, 0.1]$), except for the case of Kin8nm. When various η values were considered equally as good for a given dataset, we selected the η value that yielded the lowest average MPIW_{val}, which was $\eta = 0.01$ for Boston, Concrete, and Yacht, $\eta = 0.005$ for Kin8nm, and $\eta = 0.05$ for the rest of the datasets. This is significant because it shows that the sensitivity of our method to the scale factor η is low, unlike the hyperparameters required by QD-Ens, as explained in detail in Section IV-B. What is more, our method requires a single hyperparameter, η, while QD-Ens requires two: λ and a softening factor used to enforce differentiability of its loss function; and QD+ requires four: λ_1, λ_2, and λ_3, and the same softening factor used by QD-Ens. Note that our method does not need an additional softening factor given that the functions of DualAQD are already differentiable.

We see in Table III that DualAQD yielded better $\text{PICP}_{\text{test}}$ values than the other methods, except for field "B" where QD-Ens had the highest $\text{PICP}_{\text{test}}$ value, albeit at the expense of generating excessively wide PIs. What is more, Fig. 7 shows that, in general, DualAQD obtained lower μ_ω values; as a consequence, it achieved the lowest μ value in each of the three fields (Table III), which implies that it offers a better width-coverage trade-off in comparison to the other methods. Notice that Table III shows $\text{PICP}_{\text{test}}$ values lower than 95% for field A. During training and validation, the coverage probability did reach the nominal value of 95%. Note that, since the distribution of the test set (2020) differs from the one seen during training (2016 and 2018), the $\text{PICP}_{\text{test}}$ values may not be equal to those obtained during training. This illustrates the ability to show increased uncertainty when insufficient data are available for making reliable predictions.

Fig. 6 shows that DualAQD was able to produce better distributed PIs for field "A" (i.e., with a wider range of values) while achieving slightly better $\text{PICP}_{\text{test}}$ and $\text{MPIW}_{\text{test}}$ values than QD-Ens. This means that DualAQD is more dynamic in the sense that it outputs narrower PIs when it considers there is more certainty and wider PIs when there is more uncertainty (recall the behavior in Fig. 3). As a consequence, 54.4%, 44.3%, and 40.3% of the points processed by DualAQD on field "A" have smaller PI width than QD+, QD, and MC-Dropout, respectively, while still being able to cover the observed target values. Similarly, 88.7%, 65.3%, and 49.9% of the points processed by DualAQD on field "B" have smaller PI width than QD+, QD, and MC-Dropout while still covering the observed target values and 62.5%, 6.0%, and 8.8% of the points processed by DualAQD on field "C" have smaller PI width than QD+, QD, and MC-Dropout while still covering the observed target values.

Finally, Fig. 6 shows that DualAQD indicates higher uncertainty in the lower (southern) region of the field, which received a nitrogen rate value that was not used in previous years (i.e., it was not available for training). Similarly, regions of high yield values are related to high nitrogen rate values; however, there exist considerably fewer training samples of this type, which logically would lead to greater uncertainty. Thus, there is more uncertainty when predicting regions that received high nitrogen rate values, and this is represented effectively by the uncertainty map generated by DualAQD but not the compared methods. It is worth mentioning that even though DualAQD showed some degree of robustness empirically when given previously unseen samples, NN-based PI generation methods do not offer any guarantee for the behavior of the model for out-of-distribution samples.

VI. Conclusion

Accurate uncertainty quantification is important to increase the reliability of DL models in real-world applications that require uncertainty to be addressed. In this work, we focus on methods that generate PIs using conventional deep NNs for regression tasks. As such, we presented a method that uses two companion NNs: one that specializes in generating accurate target estimations and another that has two outputs and is trained using a novel loss function designed to generate accurate and narrow PIs.

We tested our method, DualAQD, with a challenging synthetic dataset and seven benchmark datasets using feedforward NNs. We also experimented with a real-world application of 2-D regression using spatially correlated data to convey the usefulness and applicability of our PI generation method. Therefore, we conclude that by using our loss function $\text{Loss}_{\text{DualAQD}}$, we were able to produce higher quality PIs in comparison to QD+, QD-Ens, and MC-Dropout-PI; that is, our method generated significantly narrower PIs while maintaining a nominal probability coverage without detriment to its target estimation accuracy. DualAQD was also shown to be more dynamic in the sense that it better reflects varying levels of uncertainty within the data. It is important to point out that we achieved better performance metrics than the competing algorithms using less computational complexity and fewer tunable hyperparameters. In the future, we plan to adapt our loss function for its use in BNNs.

Acknowledgment

The authors would like to thank the members of the Data Intensive Farm Management project (USDA-NIFA-AFRI and USDA-NRCS) for their comments through the development of this work, especially Dr. Paul Hegedus for collecting and curating the site-specific data. They also would like to thank Jordan Schupbach for providing advice on the experimental design.

REFERENCES

[1] D. Ghimire, D. Kil, and S.-H. Kim, "A survey on efficient convolutional neural networks and hardware acceleration," *Electronics*, vol. 11, no. 6, p. 945, Mar. 2022.

[2] V. Buhrmester, D. Münch, and M. Arens, "Analysis of explainers of black box deep neural networks for computer vision: A survey," *Mach. Learn. Knowl. Extraction*, vol. 3, no. 4, pp. 966–989, Dec. 2021.

[3] M. J. Colbrook, V. Antun, and A. C. Hansen, "The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and Smale's 18th problem," *Proc. Nat. Acad. Sci. USA*, vol. 119, no. 12, Mar. 2022, Art. no. e2107151119.

[4] A. Zarnani, S. Karimi, and P. Musilek, "Quantile regression and clustering models of prediction intervals for weather forecasts: A comparative study," *Forecasting*, vol. 1, no. 1, pp. 169–188, Oct. 2019.

[5] A. Ruospo and E. Sanchez, "On the reliability assessment of artificial neural networks running on AI-oriented MPSoCs," *Appl. Sci.*, vol. 11, no. 14, p. 6455, Jul. 2021.

[6] E. D. Meenken et al., "Bayesian hybrid analytics for uncertainty analysis and real-time crop management," *Agronomy J.*, vol. 113, no. 3, pp. 2491–2505, May 2021.

[7] D. Tran et al., "Plex: Towards reliability using pretrained large model extensions," 2022, *arXiv:2207.07411*.

[8] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1050–1059.

[9] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Trans. Neural Netw.*, vol. 22, no. 9, pp. 1341–1356, Sep. 2011.

[10] D. L. Shrestha and D. P. Solomatine, "Machine learning approaches for estimation of prediction interval for the model output," *Neural Netw.*, vol. 19, no. 2, pp. 225–235, Mar. 2006.

[11] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Lower upper bound estimation method for construction of neural network-based prediction intervals," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 337–346, Mar. 2011.

[12] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4072–4081.

[13] X. Zhang, Z. Shu, R. Wang, T. Zhang, and Y. Zha, "Short-term load interval prediction using a deep belief network," *Energies*, vol. 11, no. 10, p. 2744, Oct. 2018.

[14] I. M. Galván, J. M. Valls, A. Cervantes, and R. Aler, "Multi-objective evolutionary optimization of prediction intervals for solar energy forecasting with neural networks," *Inf. Sci.*, vols. 418–419, pp. 363–382, Dec. 2017.

[15] E. Simhayev, G. Katz, and L. Rokach, "PIVEN: A deep neural network for prediction intervals with specific value prediction," 2020, *arXiv:2006.05139*.

[16] T. Salem, H. Langseth, and H. Ramampiaro, "Prediction intervals: Split normal mixture from quality-driven deep ensembles," in *Proc. 36th Conf. Uncertainty Artif. Intell.*, J. Peters and D. Sontag, Eds., vol. 124, Aug. 2020, pp. 1179–1187.

[17] M. Ganaie, M. Hu, A. Malik, M. Tanveer, and P. Suganthan, "Ensemble deep learning: A review," *Eng. Appl. Artif. Intell.*, vol. 115, Oct. 2022, Art. no. 105151.

[18] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118. New York, NY, USA: Springer, 2012.

[19] L. R. Chai, "Uncertainty estimation in Bayesian neural networks and links to interpretability," M.S. thesis, Dept. Eng., Univ. Cambridge, Cambridge, U.K., 2018.

[20] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, Apr. 2017.

[21] A. Wu, S. Nowozin, E. Meeds, R. Turner, J. Hernández-Lobato, and A. Gaunt, "Deterministic variational inference for robust Bayesian neural networks," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–24.

[22] P. Izmailov, W. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. Wilson, "Subspace inference for Bayesian deep learning," in *Proc. 35th Uncertainty Artif. Intell. Conf.*, Jul. 2020, pp. 1169–1179.

[23] J. Lu, J. Ding, C. Liu, and T. Chai, "Hierarchical-Bayesian-based sparse stochastic configuration networks for construction of prediction intervals," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3560–3571, Aug. 2022.

[24] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez, "Quality of uncertainty quantification for Bayesian neural network inference," 2019, *arXiv:1906.09686*.

[25] S. Farquhar, M. A. Osborne, and Y. Gal, "Radial Bayesian neural networks: Beyond discrete support in large-scale Bayesian deep learning," in *Proc. 23rd Int. Conf. Artif. Intell. Statist.*, in Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108, Aug. 2020, pp. 1352–1362.

[26] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.

[27] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 103–110.

[28] J. Schupbach, J. W. Sheppard, and T. Forrester, "Quantifying uncertainty in neural network ensembles using U-statistics," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[29] A. Khosravi, S. Nahavandi, D. Srinivasan, and R. Khosravi, "Constructing optimal prediction intervals by using neural networks and bootstrap method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1810–1815, Aug. 2015.

[30] J. Lu, J. Ding, X. Dai, and T. Chai, "Ensemble stochastic configuration networks for estimating prediction intervals: A simultaneous robust training algorithm and its application," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5426–5440, Dec. 2020.

[31] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, vol. 1, Jun. 1994, pp. 55–60.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[33] K. F. Wallis, "The two-piece normal, binormal, or double Gaussian distribution: Its origin and rediscoveries," *Stat. Sci.*, vol. 29, no. 1, pp. 106–112, Feb. 2014.

[34] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/index.php

[35] P. B. Hegedus et al., "Towards a low-cost comprehensive process for on-farm precision experimentation and analysis," *Agriculture*, vol. 13, no. 3, p. 524, Feb. 2023.

[36] G. Morales, J. W. Sheppard, P. B. Hegedus, and B. D. Maxwell, "Improved yield prediction of winter wheat using a novel two-dimensional deep regression neural network trained via remote sensing," *Sensors*, vol. 23, no. 1, p. 489, Jan. 2023.

[37] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6405–6416.

Giorgio Morales (Member, IEEE) received the B.S. degree in mechatronic engineering from the National University of Engineering, Lima, Peru, in 2015, and the M.S. degree in computer science from Montana State University, Bozeman, MT, USA, in 2021, where he is currently pursuing the Ph.D. degree.

He is currently a member of the Numerical Intelligent Systems Laboratory (NISL), Montana State University. His research interests include deep learning, explainable machine learning, computer vision, and precision agriculture.

John W. Sheppard (Fellow, IEEE) received the Ph.D. degree in computer science from Johns Hopkins University, Baltimore, MD, USA, in 1997.

He is currently a fellow of the Institute of Electrical and Electronics Engineers, Johns Hopkins University. He is also a Distinguished Professor of computer science with the Norm Asbjornson College of Engineering, Gianforte School of Computing, Montana State University, Bozeman, MT, USA. His research interests include extending and applying algorithms in deep learning, probabilistic graphical models, and evolutionary optimization to a variety of application areas, including electronic prognostics and health management, precision agriculture, and medical diagnostics.