Uncertainty-Aware Spiking Neural Networks for Regression

Tao Sun*, Sander Bohté *†
*†Machine Learning Group, Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
† Cognitive and Systems Neuroscience Group, University of Amsterdam, Amsterdam, The Netherlands
Email: *tao.sun@cwi.nl, ‡sbohte@cwi.nl

***Abstract*—Uncertainty estimation is a key component for quantifying the reliability of modern deep learning models, and is crucial for many real-world applications. However, efficient methods for uncertainty estimation in spiking neural networks (SNNs), particularly for regression tasks, remain underexplored. In this work, we demonstrate that uncertainty estimation in SNN-based regression tasks can be effectively achieved using recent frameworks originally developed for classification. Specifically, we adapt these frameworks to regression with two uncertainty-aware approaches: (1) a heteroscedastic Gaussian method, in which the SNN predicts both the mean and variance of the target variable; and (2) a Regression-as-Classification (RAC) method, which reformulates regression as a classification task to enable probabilistic modeling. We evaluate our approaches on a toy dataset and several benchmark regression datasets, showing that these approaches deliver efficient and high-quality uncertainty estimates, comparable to or surpassing state-of-the-art deep neural network baselines. Our findings underscore the potential of SNNs for uncertainty estimation in regression tasks, offering a biologically inspired and energy-efficient solution for applications requiring both accuracy and robustness, while also paving the way for broader adoption of SNNs in sequential and event-driven domains.**

***Index Terms*—Spiking Neural Networks, Uncertainty Estimation, Regression, AOT-SNN, Heteroscedastic Gaussian, Regression-as-Classification, Energy Efficiency**

I. Introduction

Uncertainty estimation plays a pivotal role in machine learning, especially in high-stakes applications such as autonomous systems, healthcare, and financial forecasting, where the confidence in predictions is as vital as the predictions themselves [1]. In these high-stake models, uncertainty is often represented through probabilities, with high-quality uncertainty estimation ensuring that predicted probabilities accurately reflect the true likelihood of a prediction being correct [2]. While classification models typically provide a probability distribution over discrete classes, which facilitates uncertainty estimation, regression models produce continuous outputs, making uncertainty estimation significantly more challenging [3]. In deep neural networks (DNNs), uncertainty estimation has been extensively studied for both classification and regression tasks, thereby enhancing model reliability and robustness [4], [5]. However, the methods developed for uncertainty estimation in DNNs are generally not directly applicable to event-driven neural networks such as spiking neural networks (SNNs) [6].

Inspired by biological computation, SNNs are increasingly studied for real-time, resource-constrained applications [7]. The sparse and asynchronous event-driven nature of SNNs makes them highly suitable for tasks requiring low-latency decision-making and energy-efficient processing [8]. However, these characteristics also pose challenges for integrating uncertainty estimation techniques originally developed for DNNs. For DNNs, Monte Carlo dropout (MC-dropout) [9] and deep ensembles [3] are state-of-the-art uncertainty estimation methods that have demonstrated high-quality uncertainty estimation. Both MC-dropout (Fig. 1a) and deep ensembles (Fig. 1b) rely on multiple forward passes through their network(s), making inference inefficient when directly applied to SNNs [6]. This inefficiency stems from the need for SNNs to propagate through all their layers multiple times to achieve accurate predictions for each input. Moreover, uncertainty estimation for regression tasks faces additional challenges due to the absence of existing solutions for SNN-based regression and the inherent difficulty of event-based SNNs to create continuous outputs, further complicating the process.

Previous research has investigated uncertainty estimation in SNNs using MC-dropout, primarily for classification tasks [6]. Conventional SNNs typically rely on the model's output at the final time step of a sample to train and evaluate models — a framework we refer to as the Last-Time-Step SNN (LTS-SNN). As illustrated in Fig. 1c, the Average-Over-Time SNN (AOT-SNN) framework [6] leverages the temporal processing capabilities of SNNs to approximate multiple stochastic forward passes, thereby reducing computational cost while maintaining precise uncertainty estimation. However, unlike deep neural networks (DNNs), SNNs inherently generate event-based, discrete outputs rather than continuous values, which introduces an added layer of complexity when adapting SNN-based uncertainty estimation techniques — including both the AOT-SNN and LTS-SNN frameworks — to regression tasks.

For classification tasks, DNNs typically compute a value for each class, which is subsequently converted into a normalized probability using the softmax function. In regression, estimating uncertainty is more challenging because a regular regression model only predicts a point estimate of the target variable as the outcome, rather than a probability distribution conditioned on an input. Deep ensembles [3], often regarded as

Tao Sun is supported by NWO-NWA grant NWA.1292.19.298.

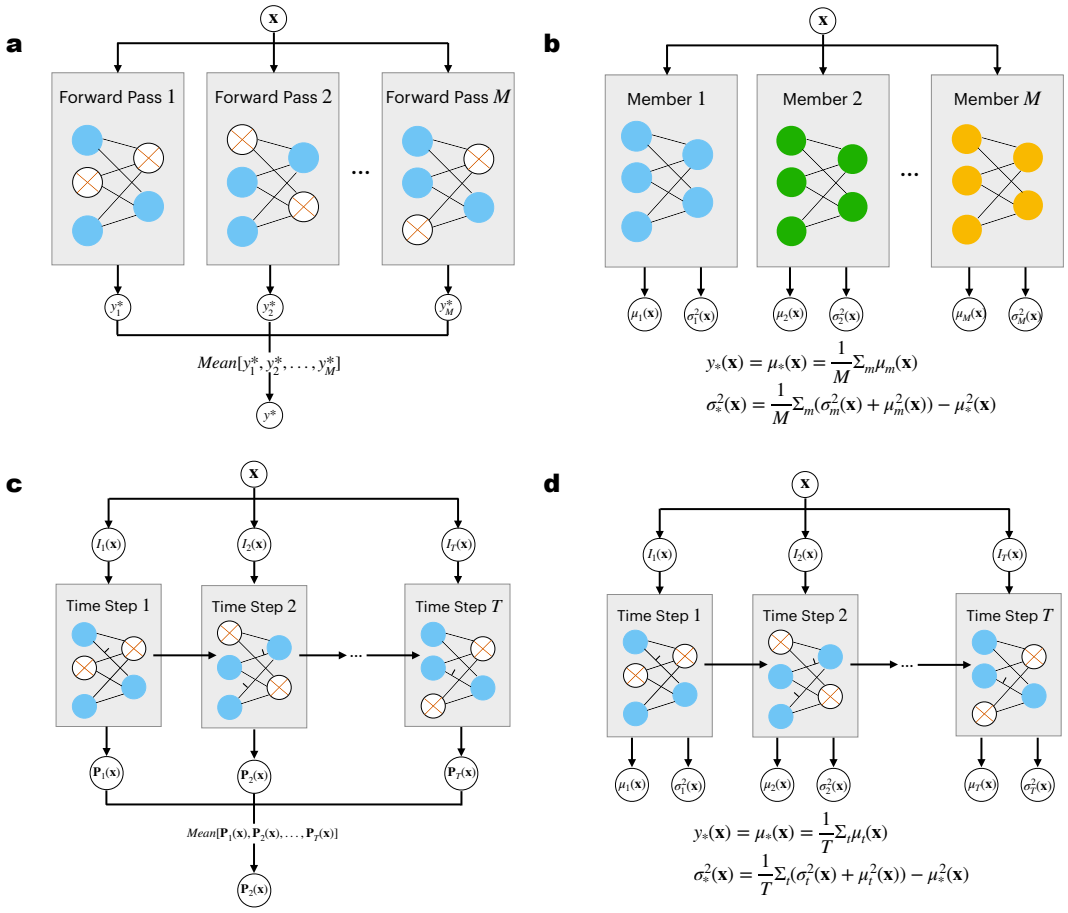

Fig. 1. **a**, The MC-dropout method estimates uncertainty by averaging the outputs of multiple forward passes through a dropout-enabled DNN. **b**, Deep ensembles train an ensemble of DNNs with identical architectures. For regression tasks, each member in an ensemble assumes a heteroscedastic Gaussian distribution and outputs two variables per input: the mean $\mu_m(\mathbf{x})$ and variance $\sigma_m^2(\mathbf{x})$. The predictive mean $\mu_*(\mathbf{x})$ and variance $\sigma_*^2(\mathbf{x})$ of the ensemble are derived from these two variables. **c**, In classification, AOT-SNNs apply continual MC-dropout in SNNs by averaging the probabilities across time steps to generate the predictive distribution. **d**, In regression, at each time step, an AOT-SNN outputs the the mean $\mu_t(\mathbf{x})$ and variance $\sigma_t^2(\mathbf{x})$, with the final predictive mean $\mu_*(\mathbf{x})$ and variance $\sigma_*^2(\mathbf{x})$ calculated from these values.

'the gold standard for accurate and well-calibrated predictive distributions' [10], address this challenge by estimating both the mean and variance for a given input and subsequently generating a conditional probability distribution (CPD) of the target variable, assuming a heteroscedastic Gaussian distribution [11]. Mixture density networks [12] predict more flexible CPDs through the mixtures of Gaussians assumption, with the mixing coefficients and parameters of the individual Gaussians being output by a neural network.

Recently, the regression-as-classification (RAC) approach, which reformulates regression as a classification problem, has demonstrated excellent performance in regression tasks [13]. Specifically, RAC divides the range of the target variable into equal-interval bins, treating each bin as a class [14]. The RAC approach can often outperforms regular regression models [13] and has been applied to various regression tasks, such as image colorization [15], depth estimation [16], age estimation [17], and pose estimation [18]. However, the discussion of DNN-based uncertainty estimation using the RAC approach is very limited: an RAC-based method is used to obtain conformal prediction sets for regression in [14], and in [19], deep ensembles or an auxiliary-network based method is proposed to estimate the uncertainty of depth estimation. To the best of our knowledge, there have been no discussions in the literature on uncertainty estimation for SNNs in regression using the RAC approach.

Here, we present two uncertainty-aware approaches for adapting both the AOT-SNN and LTS-SNN frameworks to regression tasks. The first approach employs SNNs that calculate both the mean and variance at each time step as separate outputs. As illustrated in Fig. 1d, the final predicted mean and variance in AOT-SNNs are computed by aggregating the outputs across all time steps, following a strategy similar to that used in classification tasks. In contrast, LTS-SNNs rely solely on the outputs from the final time step. In both frameworks, the CPD of the target variable is modeled as a (heteroscedastic) Gaussian distribution parameterized by the predicted mean and variance. The second approach is based

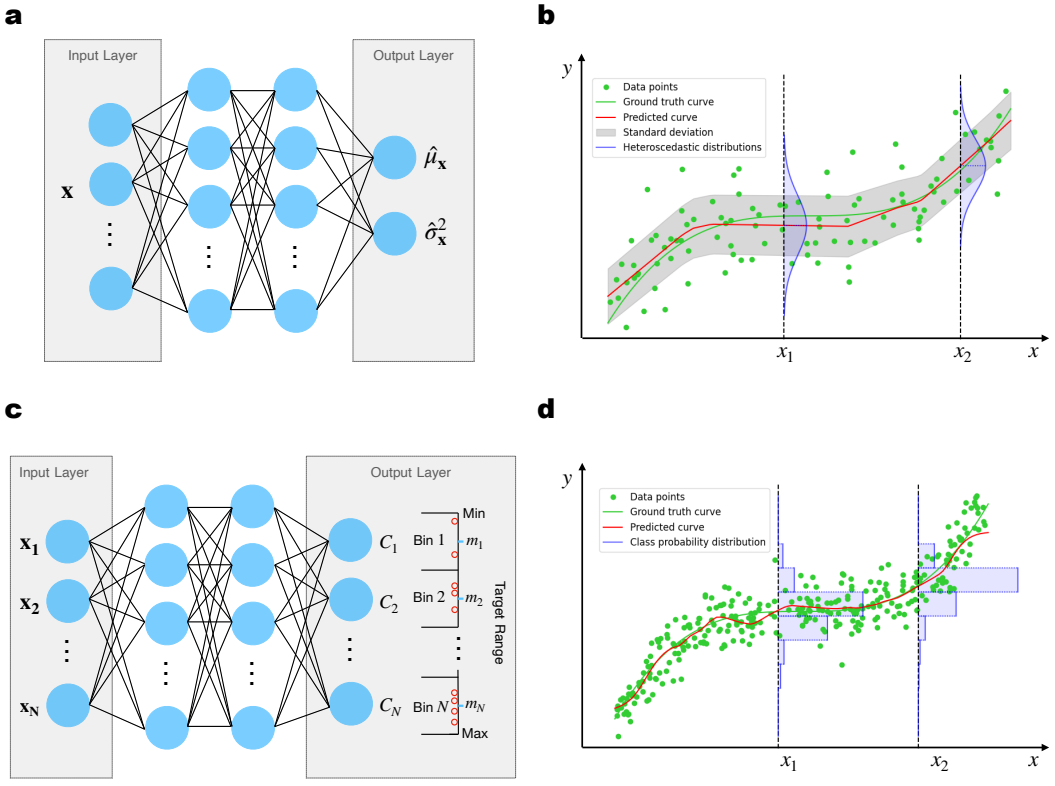

Fig. 2. Illustration of the heteroscedastic Gaussian and Regression-as-Classification (RAC) approaches for uncertainty estimation in regression tasks. (a) Neural network architecture for predicting mean and variance. (b) Data fitting with the heteroscedastic Gaussian assumption. (c) RAC approach, discretizing the target variable into bins and predicting conditional probabilities. (d) Data fitting with the RAC approach under the uniform distribution assumption.

on the RAC approach, which allows the direct application of classification-based AOT-SNNs and LTS-SNNs for uncertainty estimation. Assuming a uniform distribution within each bin, we formulate CPDs for regression by interpreting the bin probabilities as a piecewise-constant probability density. Both methods were validated using a toy example dataset as well as experiments on standard benchmark regression datasets. The results demonstrate that both proposed SNN methods achieve uncertainty performance and accuracy on par with state-of-the-art DNN approaches.

II. Background

In this section, we formalize the regression problem within a probabilistic framework. We then introduce two uncertainty-aware approaches for regression: the heteroscedastic Gaussian method [3], [11] and the regression-as-classification (RAC) method [13]. Lastly, we briefly outline the evaluation techniques for assessing the quality of uncertainty estimation.

A. Problem Setup

A regression task aims to predict a continuous target variable y given an input vector $\mathbf{x}$. Formally, let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote a dataset of N independent observations, where $\mathbf{x}_i \in \mathbb{R}^d$ represents the d-dimensional vector for the i-th observation, and $y_i \in \mathbb{R}$ is the corresponding target value. A regression model learns a mapping function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes a loss function, typically the mean squared error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2. \tag{1}$$

If we assume that given an input $\mathbf{x}$ in the training dataset D, the corresponding target value follows a (homoscedastic) Gaussian distribution that has a constant variance, the MSE loss is equivalent to a negative log-likelihood (NLL) loss.

B. Regression with the heteroscedastic Gaussian assumption

The constant variance assumption in the homoscedastic Gaussian distribution of the MSE loss may not be satisfied in many real regression problems. To tackle this issue, as depicted in Fig. 2a, a neural network is commonly employed to compute two values: the mean $\mu(\mathbf{x})$ and the variance $\sigma^2(\mathbf{x})$ of the target variable y [11]. In this framework, $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ are assumed to follow heteroscedastic Gaussian distributions (Fig. 2b). The CPD of the corresponding target value y can be written as:

$$\mathbf{p}(y|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x})). \tag{2}$$

The NLL loss used to train such a regression model is

$$\text{NLL} = \frac{1}{2}\log(2\pi\sigma^2(\mathbf{x})) + \frac{(y-\mu(\mathbf{x}))^2}{2\sigma^2(\mathbf{x})}. \tag{3}$$

Deep ensembles [3] carry out uncertainty estimation in regression based on this method. Specifically, deep ensembles train a specific number (M) of DNNs with identical architecture (Fig. 1b). For an input $\mathbf{x}$, each of those DNNs outputs a mean $\mu_m(\mathbf{x})$ and a variance $\sigma_m^2(\mathbf{x})$. The predictive mean $\mu_*(\mathbf{x})$ and variance $\sigma_*(\mathbf{x})^2$ of an ensemble are written as:

$$y_*(\mathbf{x}) = \mu_*(\mathbf{x}) = \frac{1}{M}\Sigma_m \mu_m(\mathbf{x}). \tag{4}$$

$$\sigma_*^2(\mathbf{x}) = \frac{1}{M}\Sigma_m(\sigma_m^2(\mathbf{x}) + \mu_m^2(\mathbf{x})) - \mu_*^2(\mathbf{x}). \tag{5}$$

C. Regression-as-Classification (RAC)

As shown in Fig. 2c, the RAC approach [13] discretizes the range space $[y_{min}, y_{max}]$ of the target variable in a regression task into K equal-sized intervals, called *bins*. The boundaries and midpoints of bins are written as $b_1, b_2, \cdots, b_{K+1}$ and $m_1, m_2, \cdots, m_K$, respectively, with

$$m_k = \frac{b_k + b_{k+1}}{2}. \tag{6}$$

Treating each bin as a class C_k, where $k = 1, 2, \cdots, K$, a regression problem can be reformulated as a classification problem. The continuous target variables can be converted into discrete class labels $\tilde{y}_i = \text{argmin}_j |y_i - m_j|$, where the smallest j is taken in case of ties. By minimizing the cross-entropy loss, a neural network can be trained for the reformulated classification problem with the discretized dataset $\tilde{D} = \{(\mathbf{x}_i, \tilde{y}_i)\}_{i=1}^N$. The RAC approach facilitates uncertainty estimation, as RAC models yield a normalized CPD for each input, written as $p_1, p_2, \cdots, p_K$, via the Softmax function. The predicted target value can be computed as the expected value over those probabilities $y_*(\mathbf{x}) = \sum_k p_k m_k$.

D. Quality of Uncertainty Estimation

Uncertainty estimation aims to produce predictive distributions that are as sharp as possible, while remaining well-calibrated [20]. Calibration refers to the statistical alignment between predicted probabilities and observed frequencies, whereas sharpness describes the concentration or specificity of the predictive distributions. Metrics that jointly assess both aspects are known as proper scoring rules [20], [21], with commonly used examples including the Brier score (BS) and negative log-likelihood (NLL). A properly minimized score from such a rule indicates a high-quality prediction that is not the result of trivial solutions. Proper scoring rules are also frequently used as loss functions for model training. In our evaluation, we report both the Root Mean Squared Error (RMSE), which reflects the accuracy of point predictions, and NLL, which quantifies the quality of uncertainty estimates.

III. Method

In this section, we explain the AOT-SNN method and explain how we apply it to the heteroscedastic Gaussian regression model and the RAC model.

A. Uncertainty Estimation for SNNs in classification

SNNs typically use similar types of network topologies as DNNs, but their computation is distinct. SNNs employ stateful and binary-valued spiking neurons, as opposed to the stateless, analog-valued neurons of DNNs. Consequently, unlike the synchronous computation in DNNs, inference takes the form of an iterative process through multiple time steps $t = 0, 1, \ldots, T$ in typical SNN models. During each time step t, the membrane potential of a spiking neuron U_t is influenced by the impinging spikes from connected neurons emitted at time step $t-1$ and the previous membrane potential U_{t-1}. When the membrane potential U_t reaches a threshold θ, the neuron emits a spike. This sparse, asynchronous communication between connected neurons is key to enabling SNNs to achieve high energy efficiency.

MC-dropout [9] is a popular technique for estimating predictive uncertainty in DNNs. It involves enabling dropout during both training and inference, where for a given input, multiple forward passes are used to obtain a distribution of outputs. Applying this method directly to an SNN can be computationally inefficient due to its inherent time-step mechanism, which requires running the SNN multiple times for a single forward pass. As a result, multiple forward passes substantially increase resource demands for uncertainty estimation. In [6], an efficient framework named *AOT-SNN* is introduced that leverages the time-step mechanism of SNNs for MC-dropout. Instead of running multiple forward passes, AOT-SNNs compute predictive distributions by averaging the outputs across time steps within a single forward pass, as illustrated in Fig. 1c. This approach significantly reduces computational costs while still enhancing the quality of uncertainty estimation. Additionally, AOT-SNNs introduce the average-over-time loss function, which calculates loss by averaging over multiple time steps rather than using only the final time step:

$$L = \frac{1}{T}\sum_{t=1}^{T} l(t). \tag{7}$$

In contrast, conventional SNNs typically rely only on the output at the final time step for training and evaluation. This setup, referred to as the LTS-SNN framework, computes the loss as

$$L = l(T), \tag{8}$$

where $l(T)$ denotes the loss evaluated at the final time step. In classification tasks, AOT-SNNs have been shown to outperform LTS-SNNs in both predictive accuracy and uncertainty estimation [6].

B. SNN Regression over the heteroscedastic Gaussian model

In regression based on the heteroscedastic Gaussian model, each time step of an SNN model has two outputs, representing the mean $\mu_t(\mathbf{x})$ and the variance $\sigma_t^2(\mathbf{x})$ of the target variable, respectively. In the implementation, a readout integrator layer that has two non-spiking neurons is employed as the output

layer, following the approach in [22], which effectively addresses the issue of continuous outputs in SNNs. As such, the membrane potentials of these two output neurons are taken as $\mu_t(\mathbf{x})$ and $\sigma_t^2(\mathbf{x})$. As shown in Fig. 1d, the predicted mean and variance for a sample $\mathbf{x}$ in AOT-SNNs are computed as $\mu_*(\mathbf{x}) = \frac{1}{T}\Sigma_t \mu_t(\mathbf{x})$ and $\sigma_*^2(\mathbf{x}) = \frac{1}{T}\Sigma_t(\sigma_t^2(\mathbf{x}) + \mu_t^2(\mathbf{x})) - \mu_*^2(\mathbf{x})$, respectively. In contrast, for LTS-SNNs, the predicted mean and variance for the same sample $\mathbf{x}$ are given by $\mu_*(\mathbf{x}) = \mu_T(\mathbf{x})$ and $\sigma_*^2(\mathbf{x}) = \sigma_T^2(\mathbf{x})$, respectively.

C. SNN Regression over RAC

1) Predictive value of the target variable: As discussed previously, the RAC approach assigns a probability p_i to each discretized bin, where $i = 1, 2, \cdots, K$. For uncertainty estimation, however, we need a probability distribution $\mathbf{p}(y|\mathbf{x})$ that spans the entire range of the continuous target variable y. To achieve this, we assume the predicted target value follows a uniform distribution within each bin (Fig. 2d), represented as f_k, which can be formulated as

$$f_k = \frac{p_k}{b_{k+1} - b_k}. \tag{9}$$

With this setup, the expected value $\mathbb{E}[y]$ of the target variable can be calculated as follows:

$$\mathbb{E}[y] = \Sigma_i \int_{b_k}^{b_{k+1}} f_i y \, dy \tag{10}$$

$$= \Sigma_i f_i \int_{b_k}^{b_{k+1}} y \, dy \tag{11}$$

$$= \Sigma_i f_i \cdot \left(\frac{b_{k+1}^2}{2} - \frac{b_k^2}{2} \right) \tag{12}$$

$$= \Sigma_i f_i \cdot (b_{k+1} - b_k) \left(\frac{b_k + b_{k+1}}{2} \right). \tag{13}$$

By substituting Equations 6 and 9, we obtain:

$$\mathbb{E}[y] = \Sigma_k p_k \cdot m_k. \tag{14}$$

We take $\mathbb{E}[y]$ as the predictive value $y_*(\mathbf{x})$ for regression.

2) Distance loss: To estimate uncertainty effectively, it is crucial to assign accurate probabilities not only to the ground-truth bin but also to its neighboring bins. Specifically, the bins closer to the ground-truth should be assigned higher probabilities. A model trained using cross-entropy loss does not necessarily achieve this. Therefore, we employ the loss function introduced in [14], referred to here as the *distance loss*. The distance loss not only encourages the model to concentrate its predictions on bins closer to the actual values, thereby preserving the ordering of the output space, but also ensures that the model does not become overly confident in its predictions, leading to more reliable uncertainty estimation. In implementation, this loss function consists of two components:

a) Penalization for Distance: This component imposes a penalty on the predicted density for output bins that are distant from the true output value. It is defined as

$$\mathcal{L}_{\text{dis}} = \sum_{k=1}^{K} |k - j|^q p_k, \tag{15}$$

where $q > 0$ is a hyperparameter.

b) Entropy Regularization: This component encourages variability in the predictions by regularizing the model's output distribution. It is written as

$$\mathcal{H} = -\sum_{k=1}^{K} p_k \log p_k. \tag{16}$$

The total loss is defined as

$$\mathcal{L} = \mathcal{L}_{\textbf{dis}} - \tau \mathcal{H}, \tag{17}$$

where $\tau > 0$ is a hyperparameter.

3) Network configuration: Similar to regression over the heteroscedastic Gaussian model, the output layer consists of a readout integrator layer with K non-spiking neurons, corresponding to K classes. At each time step, the membrane potentials of these non-spiking neurons are treated as logits, which are then converted to probabilities using the Softmax function. As in [6] for classification, AOT-SNNs obtain the final prediction by averaging the predictive probabilities across all time steps, whereas LTS-SNNs compute the predictive probabilities solely from the output at the final time step.

IV. Experiments

In this section, we describe the experimental evaluation of the proposed uncertainty-aware regression methods for SNNs. We validate our approaches on both a toy dataset and the standard benchmark regression datasets to evaluate their uncertainty estimation performance and predictive accuracy, in comparison with DNN-based methods.

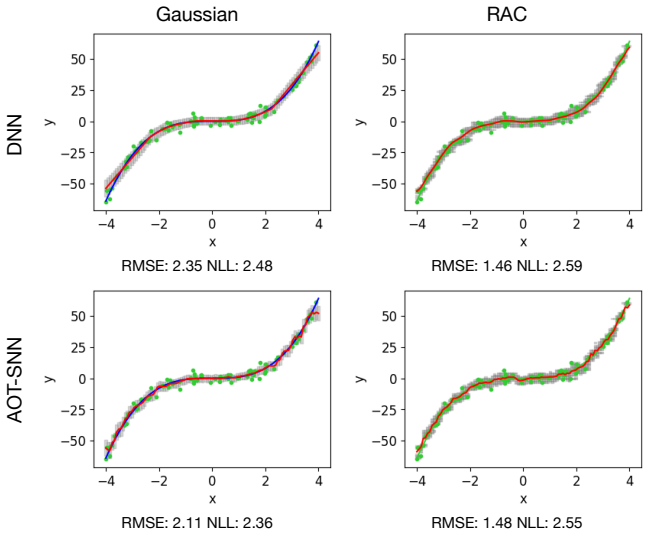

Fig. 3. Comparison of RMSE and NLL metrics for AOT-SNN and DNN models using Gaussian and RAC approaches on the toy dataset. The x-axis and y-axis represent the input and output variables, respectively. In each plot, the blue line shows the ground-truth curve, the red line indicates the predicted mean, and green dots denote training data points. The gray areas cover approximately 68.27% of the probability mass.

TABLE I
COMPARISONS OF **RMSE** ON REGRESSION BENCHMARK DATASETS. BEST RESULTS AMONG THE GAUSSIAN AND RAC MODELS ARE HIGHLIGHTED IN BOLD.

	DNNs		AOT-SNNs		LTS-SNNs	
Datasets	MC-dropout [9]	Deep Ensembles [3]	Gaussian	RAC	Gaussian	RAC
Boston housing	*2.97 ± 0.19*	*3.28 ± 1.00*	**3.35 ±0.17**	3.40 ±0.19	3.54 ±0.15	3.36 ±0.16
Concrete strength	*5.23 ± 0.12*	*6.03 ± 0.58*	5.79 ±0.11	**5.08 ±0.10**	6.18 ±0.14	5.21 ±0.13
Energy efficiency	*1.66 ± 0.04*	*2.09 ± 0.29*	2.09 ±0.13	**1.26 ±0.02**	2.26 ±0.14	1.28 ±0.02
Kin8nm	*0.10 ± 0.00*	*0.09 ± 0.00*	**0.08 ±0.00**	**0.08 ±0.00**	**0.08 ±0.00**	**0.08 ±0.00**
Naval propulsion	*0.01 ± 0.00*	*0.00 ± 0.00*	0.01 ±0.00	**0.00 ±0.00**	**0.00 ±0.00**	**0.00 ±0.00**
Power plant	*4.02 ± 0.04*	*4.11 ± 0.17*	5.13 ±0.08	4.39 ±0.03	5.21 ±0.05	**4.30 ±0.04**
Protein structure	*4.36 ± 0.01*	*4.71 ± 0.06*	4.31 ±0.02	4.13 ±0.03	4.25 ±0.01	**4.06 ±0.02**
Wine quality red	*0.62 ± 0.01*	*0.64 ± 0.04*	**0.62 ±0.01**	0.64 ±0.01	0.66 ±0.01	0.66 ±0.01

TABLE II
COMPARISONS OF **NLL** ON REGRESSION BENCHMARK DATASETS. BEST RESULTS AMONG THE GAUSSIAN AND RAC MODELS ARE HIGHLIGHTED IN BOLD.

	DNNs		AOT-SNNs		LTS-SNNs	
Datasets	MC-dropout [9]	Deep Ensembles [3]	Gaussian	RAC	Gaussian	RAC
Boston housing	*2.46 ± 0.06*	*2.41 ± 0.25*	2.69 ±0.04	2.59 ±0.02	2.73 ±0.05	**2.58 ±0.02**
Concrete strength	*3.04 ± 0.02*	*3.06 ± 0.18*	3.17 ±0.02	**3.08 ±0.01**	3.09 ±0.01	**3.08 ±0.01**
Energy efficiency	*1.99 ± 0.02*	*1.38 ± 0.22*	1.93 ±0.08	**1.52 ±0.01**	1.87 ±0.08	1.53 ±0.01
Kin8nm	*-0.95 ± 0.01*	*-1.20 ± 0.02*	-1.13 ±0.00	-0.75 ±0.00	**-1.21 ±0.01**	-0.77 ±0.00
Naval propulsion	*-3.80 ± 0.01*	*-5.63 ± 0.05*	-3.76 ±0.23	-4.30 ±0.00	-4.10 ±0.05	**-4.32 ±0.01**
Power plant	*2.80 ± 0.01*	*2.79 ± 0.04*	3.19 ±0.02	3.05 ±0.00	3.08 ±0.01	**3.02 ±0.00**
Protein structure	*2.89 ± 0.00*	*2.83 ± 0.02*	2.79 ±0.03	2.32 ±0.01	2.85 ±0.06	**2.28 ±0.00**
Wine quality red	*0.93 ± 0.01*	*0.94 ± 0.12*	1.28 ±0.04	**0.21 ±0.03**	3.12 ±0.22	0.24 ±0.03

A. Toy Dataset

The toy dataset consists of 100 training examples randomly drawn from $y = x^3 + \epsilon$, where $x \in [-4, 4]$ and $\epsilon \sim \mathcal{N}(0, 3^2)$. We have 100 evenly spaced numbers over $[-4, 4]$ for the testing examples. Both the regression with a heteroscedastic Gaussian assumption (referred to as the *Gaussian* approach) and the RAC approach were applied to the MC-dropout-based AOT-SNNs. Regular DNNs, used as a baseline, and the SNNs each had a single hidden layer of 100 neurons. The DNNs employed the ReLU activation function. While Gaussian models were trained using the NLL loss, RAC models were trained using the distance loss with $\tau = 1$, optimized via the grid search. Both SNN models set their dropout rate as 0.05. The neuron model used was PLIF [23], where the time constant is learned and shared within the same layer. The number of time steps for the SNN models was set to 8. For AOT-SNN over RAC, we constructed an SNN with $K = 150$ outputs, corresponding to 150 classes, also optimized through grid search.

1) Results: Fig. 3 presents the results on the toy dataset. The RAC approach in the right column achieves better RMSEs, which is consistent with the observations in [13]. In contrast, the Gaussian approach in the left column slightly outperforms the RAC approach in NLL, likely due to the use of the NLL loss function. Comparing our AOT-SNN models in the second row to the baseline DNN models in the first row, the Gaussian-based AOT-SNN significantly outperforms its DNN counterpart in both RMSE and NLL, while the RAC-based AOT-SNN achieves comparable performance to the RAC DNN model. The strong performance of our AOT-SNN models can be attributed to the AOT-SNN framework, where MC-dropout is applied throughout the time steps of SNN models. Note that we assume that the zig-zags in the two SNN plots are attributed to the averaging operation within the AOT-SNN framework.

B. Benchmark Datasets

To compare the predictive performance of our SNN models with other state-of-the-art methods, we conducted experiments on the standard benchmark datasets from the UCI Machine Learning Repository. Each dataset was split into 20 train-test folds, except for the protein dataset, which was divided into 5 folds. For each fold, we selected the dropout rate from the set $[0.005, 0.01, 0.05, 0.1]$ that performed best in terms of NLL on the validation set comprising 20% of the training data. This follows the same procedure as implemented in the code provided in [9]. For both AOT-SNNs and LTS-SNNs, we used a network architecture with a hidden layer of 200 PLIF neurons. The input data were encoded using a direct encoding scheme, in which each input feature was

applied as a constant current to the input neurons, following a common approach in PLIF-based models [6], [23]. Both the Gaussian and RAC models are trained with 8 time steps, corresponding to their respective best-performing durations. The Gaussian models are trained with the NLL loss, while the RAC models are trained using the distance loss with $\tau = 1$. For the RAC models, we set $K = 50$ for the output layer, following the choice made in [14] for the *Concrete Strength* and *Energy Efficiency* datasets. We found that this value provided satisfactory performance on these benchmark datasets. Further refinement of this hyperparameter could be explored to potentially enhance model performance for each dataset.

1) Results: The RMSE and NLL results for the benchmark datasets are summarized in Tables I and II, respectively. Compared to DNN baselines (MC-dropout and Deep Ensembles), the RAC-based AOT-SNN and LTS-SNN models consistently demonstrate strong performance in terms of RMSE and NLL, where AOT-SNNs generally slightly outperform LTS-SNNs. Specifically, our SNN-approaches often outperform the DNN-based MC-Dropout approach and even exceed the accuracy of Deep Ensembles for the *Concrete Strength*, *Energy Efficiency*, and *Protein Structure* datasets. Furthermore, for AOT-SNNs and LTS-SNNs, the RAC models consistently outperform Gaussian models on most datasets both in terms of RMSE and in terms of NLL, highlighting the effectiveness of the RAC approach for uncertainty estimation in regression tasks.

V. Conclusion

In this paper, we present two uncertainty-aware approaches for adapting both the AOT-SNN and LTS-SNN frameworks to regression tasks: one based on the assumption of heteroscedastic Gaussian distributions and the other leveraging the Regression-as-Classification (RAC) approach. Our experiments demonstrate that both proposed approaches achieve strong performance in uncertainty estimation, frequently surpassing traditional DNN-based methods in terms of RMSE and NLL metrics—particularly in the case of RAC-based models. These findings not only underscore the effectiveness of our regression methods for uncertainty estimation, but also lay the foundation for their extension to broader sequential scenarios, such as medical sensing or robotic control.

References

[1] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Department of Engineering, University of Cambridge, 2016.

[2] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[3] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 30, 2017, pp. 6402–6413.

[4] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.

[5] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information fusion*, vol. 76, pp. 243–297, 2021.

[6] T. Sun, B. Yin, and S. Bohté, "Efficient uncertainty estimation in spiking neural networks via mc-dropout," in *International Conference on Artificial Neural Networks (ICANN)*. Springer, 2023, pp. 393–406.

[7] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, 2022.

[8] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, 2023.

[9] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 1050–1059.

[10] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 4697–4708.

[11] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *International Conference on Neural Networks (ICNN)*, vol. 1. IEEE, 1994, pp. 55–60.

[12] C. M. Bishop, "Mixture density networks," Aston University, Tech. Rep., 1994. [Online]. Available: http://www.ncrg.aston.ac.uk/

[13] L. Stewart, F. Bach, Q. Berthet, and J.-P. Vert, "Regression as classification: Influence of task formulation on neural network features," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2023, pp. 11 563–11 582.

[14] E. K. Guha, S. Natarajan, T. Möllenhoff, M. E. Khan, and E. Ndiaye, "Conformal prediction via regression-as-classification," in *International Conference on Learning Representations (ICLR)*, 2024. [Online]. Available: https://openreview.net/forum?id=rulxyXjf46

[15] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 649–666.

[16] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," in *Conference on Computer Vision and Pattern Recognition*. IEEE/CVF, 2018, pp. 2002–2011.

[17] R. Rothe, R. Timofte, and L. Van Gool, "DEX: Deep expectation of apparent age from a single image," in *International Conference on Computer Vision (ICCV) workshops*. IEEE/CVF, 2015, pp. 10–15.

[18] G. Rogez, P. Weinzaepfel, and C. Schmid, "LCR-Net: Localization-classification-regression for human pose," in *Conference on Computer Vision and Pattern Recognition*. IEEE/CVF, 2017, pp. 3433–3441.

[19] X. Yu, G. Franchi, and E. Aldea, "On monocular depth estimation and uncertainty quantification using classification approaches for regression," in *International Conference on Image Processing (ICIP)*. IEEE, 2022, pp. 1481–1485.

[20] T. Gneiting and M. Katzfuss, "Probabilistic forecasting," *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 125–151, 2014.

[21] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[22] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.

[23] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2021, pp. 2661–2671.