Probabilistic Regression via Neural Network Ensembles

Leon Winheim[*†], **Hayk Amirkhanian**[*‡§], **Markus Walker**[†], **Uwe D. Hanebeck**[†], and **Marco F. Huber**[‡§]

[†]Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
leon.winheim@kit.edu, markus.walker@kit.edu, uwe.hanebeck@kit.edu

[‡]Research Unit Artificial Intelligence and Machine Vision
Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Stuttgart, Germany
hayk.amirkhanian.namagerdi@ipa.fraunhofer.de, marco.huber@ieee.org

[§]Institute of Industrial Manufacturing and Management IFF
University of Stuttgart, Germany

***Abstract*—Probabilistic neural networks (PNNs) often target conditional mean and variance in regression tasks. When data exhibits multi-modality, many methods fail to reproduce this behavior. Existing methods that tackle such problems, like mixture density networks (MDNs) and similar approaches, require rigid assumptions or are complicated to implement. After investigating the characteristics of PNN-models that enable them to reproduce arbitrary densities, we propose a new method that is simple to implement and deploy. We call it DENSE, which is an ensemble of multi layer perceptrons (MLPs) that is trained to minimize a modified Cramér–von Mises distance in the joint space of inputs and outputs. It enables non-parametric representations of arbitrary densities without (complex) likelihood design. We demonstrate that DENSE effectively captures heteroscedastic and multi-modal datasets by evaluating synthetic examples and comparing them with baselines from literature.**

***Index Terms*—probabilistic neural networks, regression, density estimation, LCD distance**

I. Introduction

Input-output relationships in regression tasks can be arbitrarily complicated, e.g., skewed, heteroscedastic, or, like the training dataset in Fig. 1, multi-modal. Such multi-modal situations could arise from, e.g., having a confounding variable that influences the output, but is not modeled, or an ill-posed inversion of a nonlinear mapping from state to measurement with probabilistic states. Scenarios could arise where such a relationship needs to be reproduced by a queryable model, and the framework of probabilistic regression should be capable of capturing such relationships.

Especially neural networks (NNs) are popular and generally capable function approximators [1] that can be used for regression tasks. While alternative frameworks exist—such as Gaussian processes [2], normalizing flows [3] or kernel-based methods [4]—we focus exclusively on NNs to keep the scope of this publication compact. While deterministic versions target the conditional mean $\mathrm{E}\{f(y|x)\}$, probabilistic versions often target the conditional mean and the conditional variance under assumptions of a uni-modal and often Gaussian likelihood function [5], [6]. Models from the class of probabilistic neural networks (PNNs) can generally quantify model uncertainty (epistemic) by having random parameters or encode an uncertainty assumption on the output values (aleatoric), or both. A popular example of a PNN from the subgroup of Bayesian neural networks (BNNs) are NNs trained with Markov Chain Monte Carlo (MCMC). It is a Bayesian algorithm that is guaranteed to converge to the given posterior (prior $\times$ likelihood) [7] and results in models that quantify both model and output uncertainty by having random parameters and an assumed likelihood function [8]. In Fig. 1, the result of a BNN with one hidden layer of 50 neurons and ReLU-activation, trained with MCMC on the basis of a Gaussian likelihood with a single learned likelihood variance parameter (as described in, e.g., [8]), is displayed. The model fails to reproduce the multi-modal part of the data in $x \in (-0.5, 0.5)$. This is due to the model being limited to uni-modal explanations of the observed data by the likelihood function that was used in formulating the target posterior.

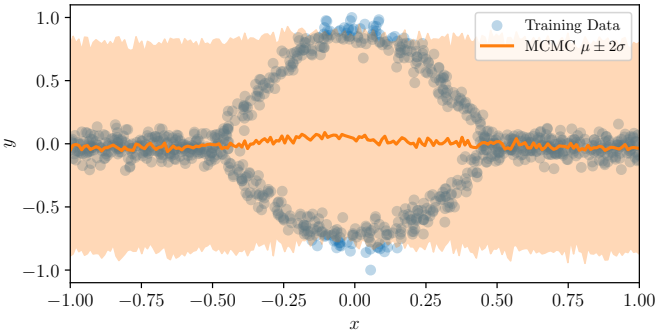

Fig. 1: Multi-modal training data and predictions of a MCMC-BNN with a Gaussian likelihood.

This work was funded by the Federal Ministry for Economic Affairs and Climate Action as part of a project of the Central Innovation Programme for SMEs (ZIM).

[*]These authors contributed equally.

The probabilistic regression framework can be seen as performing conditional density estimation and is theoretically unconstrained w.r.t. the shape of the dataset. In the mentioned case, the restriction is a result of the inappropriate assumption in the likelihood function. However, methods that can solve the problem of general and potentially multi-modal datasets (we call them *flexible*) either involve a complex likelihood design including assumptions about shape and mode count, as in mixture density networks (MDNs) [9], or the solution of association problems and clustering in the case of set distance methods such as LipNet [10]. We aim for a simpler method and an improved understanding of the effects that typical assumptions during method design have.

Contribution: In this paper, we investigate the requirements for a PNN's capability to accurately represent diverse datasets. We propose a flexible, non-parametric ensemble method that utilizes a set distance (the LCD distance (LCDd)), bypassing the need for (complex) likelihood design, called DENSE ensemble. DENSE is easily deployable and demonstrates competitive performance on heteroscedastic and multi-modal regression tasks. The method is provided as a PyTorch-based implementation.[1]

II. Problem Statement

For simplicity, we focus on the univariate case of the problem. Based on a given dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, the objective is to obtain a model that approximates the underlying conditional density $f(y|x)$ of outputs y given inputs x. For any given query point x_q it must provide an approximation of $f(y|x_q)$. In the following, we call the output values from the training dataset *targets* to keep them separate from predicted outputs of a model. Training seeks to find a distribution $f(\underline{\theta}|\mathcal{D})$ over the model's random parameters $\underline{\boldsymbol{\theta}}$ (i.e., the network weights) that best describe the observed data with the given model architecture, or the vector $\underline{\theta}$ in case the model's parameters are deterministic. Notably, this formulation avoids a priori assumptions regarding the distribution's shape, modality, or homoscedasticity. Also, we do not require the dataset to include multiple y_i for any single x_i, as is required in, e.g., [11].

III. Relevant Model Class

We constrain our analysis to feed-forward multi layer perceptrons (MLPs). Within this class of models, a fundamental distinction lies in the treatment of the model's parameters (i.e., network weights): they may be collected in a deterministic vector $\underline{\theta}$ or in a random vector $\underline{\boldsymbol{\theta}}$ that is characterized by a probability distribution. Each of these cases can typically be implemented using one of two architectural paradigms regarding the number of output neurons (sometimes also called network *heads*):

- The number of output neurons n_{out} corresponds to the number of dimensions of the training targets n_{targets}.
- There is a higher number of output neurons n_{out}, which are then used to parametrize a probability density over target dimensions in the training data (e.g., a Gaussian parametrized by mean and covariance network outputs).

Besides the architecture and nature of the parameters, the training philosophy is an orthogonal attribute of any method. We differentiate between Bayesian and non-Bayesian approaches. We call an approach Bayesian if the concept of a prior distribution over the parameters exists, therefore Bayes's law is involved in the training [12], and if the model is kept on a distributional basis, meaning it will not be reduced to a point estimate such as maximum a posteriori [13]. All other methods are considered non-Bayesian.

Another orthogonal attribute that characterizes a method is the way in which we assess the closeness of prediction and target values. Both could exist either as samples or as continuous density. The important aspect of this dimension is choosing a distance function that is helpful in computing updates for the parameters that should be inferred. We neglect the cases of continuous targets (i.e. where a continuous density for any query is given as target) and describe the remaining cases involving discrete training datasets:

- For the combination of *continuous prediction–sample target*, a model outputs a continuous density for each given input that is then evaluated against the training samples. The shape of this density is a model assumption. Standard mean squared error loss is equivalent to maximizing a Gaussian log-likelihood with a fixed covariance. From a state-space perspective, the assumed variance in the likelihood represents the observation noise that has to be added during prediction with the model [5].
- When the combination of *sample prediction–sample target* arises, we obtain a predicted sample set for given inputs and need to compare this sample representation to the training targets. A set distance measure like Wasserstein [10] or the LCDd [14] has to be chosen to compare both sample sets directly.

For probabilistic regression tasks, NNs capable of producing probabilistic outputs are required. We call this class PNNs. This excludes models with a deterministic $\underline{\theta}$ and $n_{\text{out}} = n_{\text{target}}$, which amounts to a standard deterministic MLP. We also exclude the case where an independent noise generator could make the output of such a deterministic model probabilistic, as nothing about this probabilistic nature would be subject to the learning procedure.

One member of the PNN class are methods with deterministic parameters $\underline{\theta}$ where $n_{\text{out}} > n_{\text{target}}$ (e.g., Mean-Variance Networks [15] or MDNs [9]).

Apart from that, methods with random parameters $\underline{\boldsymbol{\theta}}$ where $n_{\text{out}} = n_{\text{target}}$ (e.g., Progressive Net (PN) [5], probabilistic backpropagation (PBP) [16] and other common BNNs [17]), which often have subsequent additive noise generators on the output values, are considered PNNs.

Finally, combination methods with random parameters $\underline{\boldsymbol{\theta}}$ where $n_{\text{out}} > n_{\text{target}}$ (e.g., deep ensembles (DEs) [18] or Bayesian MDNs [19]) are also considered PNNs. To provide an overview, Fig. 2 contains visualizations of the included types of models.

[1] Find code at https://github.com/leonwinheim/dense_FUSION26

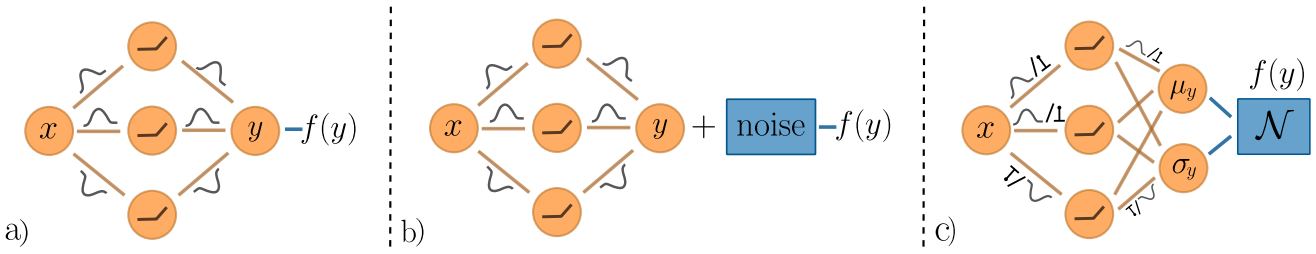

Fig. 2: Different PNN architectures. In a), a model's predictive distribution is a direct result of its uncertain parameters. In b), the total predictive uncertainty is additionally influenced by an additive noise generator. In c) a model with either deterministic (visualized by the Dirac-symbols on the connections) or random parameters yield the parameters of a continuous density as its outputs.

IV. Representational Limits in Probabilistic NNs

Existing work on NN-based probabilistic regression on multi-modal datasets includes specialized architectures and training procedures (see [10], [20] or [21]). We want to generally analyze within the model class of PNNs what enables models to solve flexible probabilistic regression tasks and deliver an overview of limiting assumptions and modeling choices. As any NN method is based on a forward step that computes an output (predictive capability) and a backward step that performs inference on the parameters of the model (learning capability), we split the analysis into these two sections.

A. Predictive Capability

What determines the predictive capability of a method depends on the nature of its parameters, i.e., whether they are deterministic or random or to which case in Fig. 2 they belong.

1) Deterministic Parameters: If a model has deterministic parameters $\underline{\theta}$, the predictive capability for any point is governed by the distribution the model output parametrizes as there is no uncertainty about the model parameters. This model type is visualized by an example in Fig. 2 c), where the parametrized density is a Gaussian, and the model would have $n_{\text{out}} = 2 \cdot n_{\text{target}}$ output neurons (network heads), because for the mean of any dimension an additional output variance is needed [15]. This enables the model to learn heteroscedastic data uncertainty because the predictive variance depends on the input. In case the dataset is multi-modal, like the training data in Fig. 1, the network needs to (at least partially) parametrize a mixture model with a sufficient amount of modes to represent the nature of the dataset, like in MDNs [9]. In any case, a distribution shape over the output and a maximum number of modes that can be reproduced during prediction has to be assumed. The model does not output y directly, but a parametrized distribution over it.

2) Random Parameters: If the model parameters $\underline{\boldsymbol{\theta}}$ are random variables, three distinct cases arise for the mechanisms that determine the resulting output distribution.

First, if $n_{\text{out}} = n_{\text{target}}$ and no additional noise generation exists, the predictive capability is completely determined by the propagation of uncertainties of $\underline{\boldsymbol{\theta}}$ through the network architecture, which corresponds to Fig. 2 a). While any input to the network is still deterministic, all intermediate values will become probabilistic upon interacting with the random weights. The randomness will be propagated through the layers towards the output, resulting in a predictive distribution. Methods with that sole source of predictive uncertainty are, e.g., Kalman Bayesian Neural Networks (KBNN) [22] or the proposed Variational Inference (VI) method in [23].

Second, if $n_{\text{out}} = n_{\text{target}}$ and an additional noise generator exists, the output distribution is a combination of the uncertainty propagation through the network and this additional noise (displayed in Fig. 2 b)). The additive noise is often Gaussian with zero-mean as in the example of a BNN trained by MCMC used in Fig. 1 or in PN [5]. It exists to represent the assumed data uncertainty, in this case homoscedastically, as the variance, though learnable by augmenting the state vector, is independent of the input [8].

Third, if instead of having an additive noise generator, the network outputs parametrize a density and $n_{\text{out}} > n_{\text{target}}$, the resulting predictive distribution is also a superposition of the model uncertainty and the parametrized density. We can imagine this as having a distribution over distributions, as it exists in the Bayesian MDN in [19] and corresponds to Fig. 2 c) with random weights.

3) Computational Aspect: In models with uncertain parameters $\underline{\boldsymbol{\theta}}$, the distribution over the weights could be arbitrary, and the complexity of the nonlinear MLP model could be high. Therefore, arbitrarily complex distributions should be possible at the output of the architecture in any of the mentioned variants. However, different aspects can break this general capability.

The first fundamental assumption regards the shape of the distribution over the weights. It is a common strategy to assume this to be Gaussian (see [5], [6] or [17]), but other distributions are possible. In such a strategy, one has to decide on a covariance structure. The most expressive is a full correlation over all weights and layers, as in PN [5], and the least expressive is the mean field approximation that enabled many methods to improve in scalability (see [6] or [16]). While strong weight correlations may preserve multi-modality in the network's hidden activations, the mean field case is subject to the central limit theorem. As the pre-activation is a weighted sum of independent random variables, the distribution tends toward a uni-modal Gaussian as the number of hidden units increases [24]. This independence assumption effectively collapses the activations toward uni-

modality, potentially inhibiting the model's ability to represent complex, multi-modal predictive densities at the output.

The second fundamental assumption arises during the actual computation of the predictive forward pass. Propagating the uncertainty through the nonlinearities of the network model is, in general, analytically intractable. Therefore, the computation of the forward pass has to be approximated. This can either happen based on samples [5] or based on moment propagation [22]. In the case of sample approximation, samples are drawn from the distribution over the weights. With each sample, a deterministic forward pass is computed and the output distribution exists as the union of the results of each deterministic forward pass. The approximation quality of this method depends on the sample count, but any shape of any distribution can be approximated without any assumptions on shape or mode count. If the approximation happens based on stochastic moments, assumptions on the pre- or post-activation values may be involved. Like in PBP [16] or KBNN [22], each value is assumed to be Gaussian in combination with either mean field or pairwise independent covariance structures (regarding pairs of neurons). As a result, any potential multi-modal value is reapproximated to be Gaussian, which is why no multi-modal predictive distribution can arise.

B. Learning Capability

A core building block of the training method is the way we determine how close the current model prediction is to the training target. In probabilistic methods, training target points need to be related to the predicted output distributions of the model. In many methods, this is done by defining a likelihood function (continuous prediction–sample target case). This function evaluates how likely it is to observe the training data under given model parameters. It is critical to recognize that the choice of the likelihood function prescribes a specific inductive bias onto the optimization process. When confronted with bimodal data, a Gaussian likelihood function results in the model showing an averaging behavior between both modes, just as is shown in Fig. 1. This is not an error, but the consequence of an imposed assumption one has to be aware of. The most flexible PNN w.r.t. its predictive power cannot break out of the scope of the assumed likelihood. Choosing a multi-modal likelihood function like in MDNs enables reproducing multi-modal data in the assumed shape and specified maximum number of modes, which is restrictive. Also, mode collapse during training is a practical issue [25].

Alternatively, if a method yields predictive samples, a set distance can be used to evaluate prediction quality. Again, the set distance has to be meaningful in multi-modal situations to be suitable for these types of datasets. A method from literature that employs this principle is called LipNet and uses a Wasserstein distance (in form of the Sinkhorn approximation) to compare predictive samples with target samples [10]. The method can be categorized as a model with deterministic parameters that outputs the parameters of a Dirac mixture distribution, i.e., sample positions. Evaluating the Wasserstein distance between two sample sets requires solving an association problem, which can be computationally expensive, especially in higher dimensions. Additionally, the proposed method needs to specify a number of clusters that should represent the mode count in a neighborhood of any given input value. The method also uses special network layers that induce Lipschitz continuity into the predictions. Overall, many steps are required to implement it and the method is quite complex.

There are other set distances to consider that may simplify potential methods, like the LCDd [14]. An advantage is that this measure is association-free and differentiable. Thus, we will use it to derive a new method, which will be proposed in the next section.

C. Summarized Requirements

To be able to solve the targeted problem class including multi-modal datasets, a method requires:

1) Appropriate predictive capability without uni-modal collapse or reapproximations.
2) A capable and flexible distance function between target samples and predictions, either shape-based with modes or a set distance.
3) A suitable optimization or inference method for adjusting network parameters.

Existing methods that generally exhibit these attributes are:

i) MDNs [25] or Bayesian MDNs [19], although they assume shape and mode count of the data.
ii) Special extensions of VI methods like in [20], with the same limitations as in i).
iii) Specialized BNN-models with additional noise generators like [21], which complicates the network structure.
iv) LipNet, a method with deterministic parameters and a Dirac mixture at the output, which includes clustering and solving an association problem that can be problematic in high dimensions.

In the next section, we want to propose a method that is as simple as possible, makes no assumptions on shape or mode count and is easy to implement and deploy, therefore mitigating some issues that the aforementioned methods exhibit.

V. DENSE Ensemble

We introduce DENSE ensemble, which stands for "Density Estimation by Neural Sample Embedding". It is a model that is described by a discrete random parameter vector $\underline{\boldsymbol{\theta}}$ (i.e. samples), and therefore is an ensemble of MLPs. No additional uncertainty modeling is present, which makes it correspond to the model architecture in Fig. 2 a). Predictions and targets are compared by employing a set distance. This and further details are described in the following subsections.

A. LCD Distance

The LCDd is a flexible method to compare probability distributions [14] and builds the core of the training step of the proposed method. It allows comparing samples to samples, continuous to continuous distributions and hybrid situations. It is based on a modified Cramér–von Mises distance where the

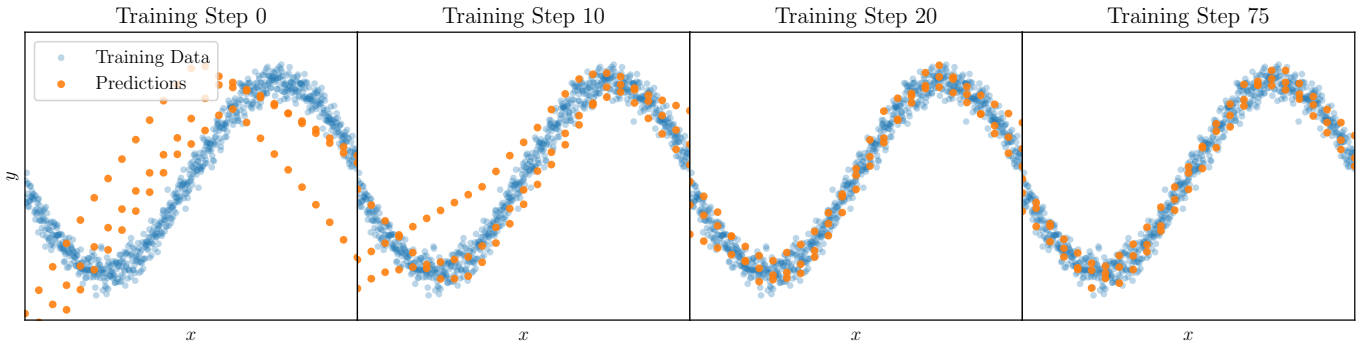

Fig. 3: Training evolution. With increasing amount of optimization steps, the predictive distribution approximates the training distribution better and better.

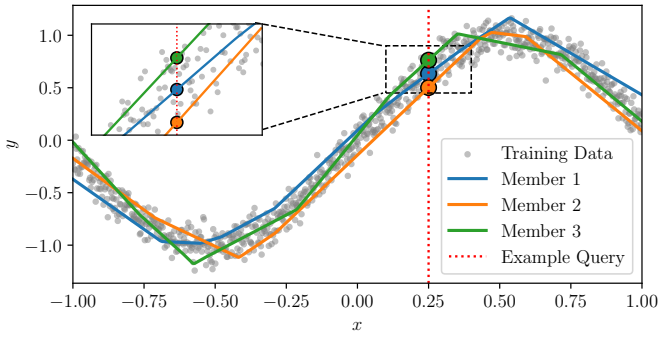

Fig. 4: Prediction in sample-based ensembles. For a specific input query, multiple prediction samples are returned.

cumulative distributions are replaced by localized cumulative distributions, which is a generalization of the cumulative distribution that is also unique in the multivariate case [26].

We use the LCDd to compare predicted samples to target samples, so it is employed as a set distance. The formulation of the squared distance D amounts to

$$D^2 = K_1\left(D_{\underline{y}} - 2D_{\underline{xy}} + D_{\underline{x}}\right) + K_2 D_E \ ,$$

where the definition of $D_{\underline{y}}, D_{\underline{xy}}, D_{\underline{x}}, C_b$ and D_E can be found in [14]. Here, $\underline{x}$ and $\underline{y}$ describe two sample sets, not inputs and outputs. Together with constants K_1 and K_2, each component of the measure has a specific meaning. $D_{\underline{y}}$ and $D_{\underline{x}}$ are the parts that prevent clustering by penalizing closeness of samples to each other in any set. $D_{\underline{xy}}$ penalizes differences between the two sample sets sample positions and ensures good coverage. Finally, D_E enforces that the means of each marginal are matching in both sample sets.

Although alternative set distances can be used, there are two attributes that make the LCDd a good fit for our case. First, it is association-free. Other than for set distances like the Wasserstein distance, every sample is compared to every sample. No association problem has to be solved, and the distance can be applied directly. Second, the distance is differentiable and an analytic gradient is available, which is advantageous for the NN training case. It has no explicit mode assumption, but is minimized when the density of the target sample set is optimally covered by the predicted sample set.

B. Representation Aspect

To avoid limitations by an assumed shape of $f(\underline{\theta})$, it is kept non-parametric and represented as a set of samples. This also determines the computation of the forward pass as sample-based. The predictive distribution for a given input x is given by

$$\hat{f}(y|x) = \frac{1}{M}\sum_{j=1}^{M} g(\underline{\theta}_j, x) \ , \tag{1}$$

where $g(.)$ is the network architecture and M is the number of ensemble members. Each j corresponds to evaluating a single sample of $f(\underline{\theta})$ in a deterministic forward pass. The "$\hat{}$" indicates that this is a result of the model, not the true underlying conditional density. The combination of sample-based computation and no assumptions on the distribution shape enables arbitrarily strong correlations between different weights and arbitrary predictive shapes within the representational capability of the sample count. To avoid confusion, it is important to note that the model is continuous on the input domain, as any MLP is. The output for any input consists of discrete points, one of each corresponding to one ensemble member (a sample from the weight distribution). This situation is visualized in Fig. 4. We do not model any explicit additional uncertainty in our model, so the only uncertainty in the output is that resulting from the marginalization over the weights, which makes the architecture correspond to Fig. 2 a) and methods like the one proposed in [23].

C. Training Aspect

To learn a conditional density on specific query points is hard, as training data points are relatively sparse, and for each training input in general, only a single target is available. This could be tackled by binning data points together, as is done in [10]. Alternatively, we propose to view the training as approximating the complete joint density of all training inputs and outputs $f(x, y)$ with a predicted joint density of the network inputs and outputs $\hat{f}(x, y)$. In our case, both are sample sets. For a single-input single-output system, this means we view each training data target together with its input as a two-dimensional sample, and do the same with the network prediction and its corresponding input. We obtain the approximated joint density

$\hat{f}(x, y)$ through inserting the conditional (1) (which is the NN-model and weights) into the definition of the joint distribution $\hat{f}(x, y) = \hat{f}(y|x)\hat{f}(x)$ as

$$\hat{f}(x, y) = \frac{1}{M} \sum_{j=1}^{M} g(\underline{\theta}_j, x)\hat{f}(x)$$

and obtain the final expression by inserting the marginal distribution $\hat{f}(x) = \frac{1}{N_\mathrm{q}} \sum_{k}^{N_\mathrm{q}} \delta(x - x_k)$ as Dirac mixture distribution (i.e., the input query points x_k)

$$\hat{f}(x, y) = \frac{1}{M} \frac{1}{N_\mathrm{q}} \sum_{j=1}^{M} \sum_{k=1}^{N_\mathrm{q}} g(\underline{\theta}_j, x)\delta(x - x_k)$$

$$\hat{f}(x, y) = \frac{1}{MN_\mathrm{q}} \sum_{j=1}^{M} \sum_{k=1}^{N_q} g(\underline{\theta}_j, x_k) \ ,$$

where M is the number of ensemble members and N_q the number of input query points used to build up the joint density. The last transformation is enabled by the sifting property of the Dirac distribution. We use the LCDd between the complete joint training dataset and the complete joint prediction dataset as loss and minimize it during the training. With gradient tracking, we can trace each network's parameters contribute to the LCDd and use standard optimizers to minimize it.

The optimization process over multiple steps is visualized in Fig. 3. With each iteration, the orange predictive sample set approximates the blue training set better. All ensemble members are optimized jointly, which is a distinction to methods like DE [18]. Experiments found a performance decrease in DE if members are trained jointly [27]. This is due to diversity in the ensemble members being an emergent property, which is very different to our joint training situation, where the interaction is actively influenced by the loss.

Apart from that, the LCDd enables useful evaluation independent of the shape of the training data distribution, and also with different amounts of samples. As we try to match the shape of the joint training density, but only have the distribution over the model parameters available as source for uncertainty, there is a catch: From a model perspective, we propagate model uncertainty. From a training perspective, we optimize this output uncertainty to match the uncertainty in the data. If sparsity exists in some regions of the input domain, the loss will lead to less focus being laid there, which leads to model disagreement and therefore also encoding a true epistemic part. However, as in [23], it cannot be disentangled anymore.

D. Summary

The DENSE ensemble is a simple PNN method that consists of a set of MLPs being trained by optimizing the LCDd via a common optimization library. It is able to reproduce heteroscedastic and multi-modal datasets without any need for prior specification of shape or mode count. The most similar approach from literature is LipNet [10]. The main differences are that DENSE is an ensemble method, while LipNet produces its output samples (called "atoms") by specifying a network output for every atom. Furthermore,

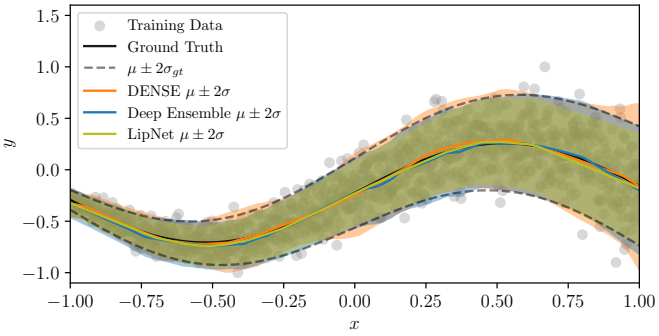

Fig. 5: DENSE, DE and LipNet on a heteroscedastic regression task.

DENSE performs optimization of the complete joint input-output density, while LipNet uses clustering methods and partitions the input space into neighborhoods, which involves a hyperparameter for the number of clusters that is not needed for DENSE. Finally, instead of the LCDd as it is used in DENSE, LipNet uses a Sinkhorn approximation of the Wasserstein distance between target and predictions.

An intriguing aspect of DENSE is its simplicity. It demonstrates the requirements for representing diverse datasets, without clustering, association, or specifying mode count and shape—providing a minimum working method.

However, there are currently also some limitations of the proposed method. The most important one is the stability of the optimization. During development, we noticed an issue with local minima. This is a standard problem in the research of NN methods. Because we compute a single objective for the complete dataset that is influenced by multiple ensemble members, this issue becomes rather worse. Incorporating restarts into the optimization showed possible improvement, however, we expect to gain more stability through, e.g., smarter regularization. Additionally, naïve evaluation of the distance is relatively slow, scaling with $\mathcal{O}(n^2)$ with n being the number of samples involved. This could be improved with batch-wise processing of the training data, which we aim to investigate in the future. Also, the method is not Bayesian and it is currently not possible to introduce a prior over the weights. Obviously, a sufficient number of ensemble members is needed to represent a specific density, but choosing many will not result in mode collapse, but in a better uncertainty representation.

VI. Evaluation

We evaluate a heteroscedastic and a multi-modal, single-input single-output regression task to showcase the flexibility of the proposed method and compare it with DE [18] and LipNet [10], depending on task and capabilities. All experiments were conducted on a Laptop with Intel Core Ultra 7 155H CPU.

A. Heteroscedastic Regression

We consider a single-input single-output regression example with heteroscedastic noise given by

$$\begin{aligned} y &= \sin(3x) + \nu \\ \nu &= \big(0.1 + 0.25(x + 1)\big)\varepsilon \\ \varepsilon &\sim \mathcal{N}(0, \sigma^2) \ . \end{aligned}$$

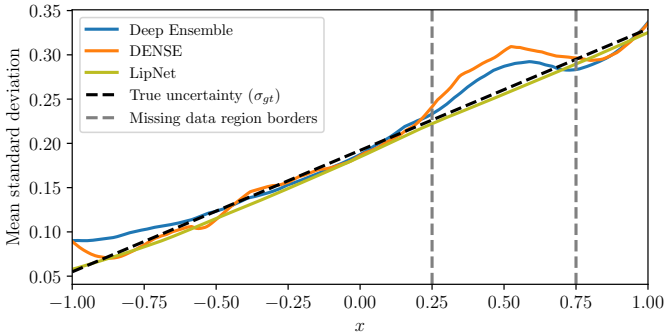

Fig. 6: Predictive heteroscedastic uncertainty. DENSE shows a significant increase in predictive uncertainty in areas where no training data was available.

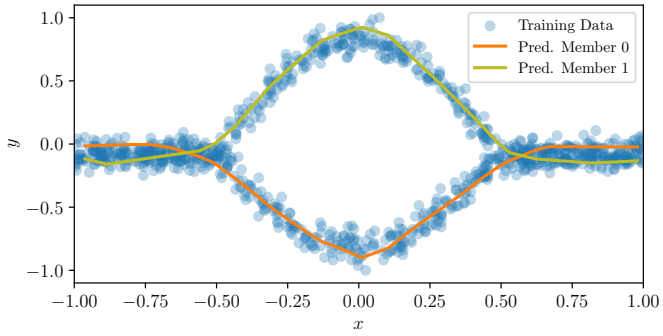

Fig. 7: Two Ensemble Members of DENSE approximate a bimodal dataset. One member covers one mode.

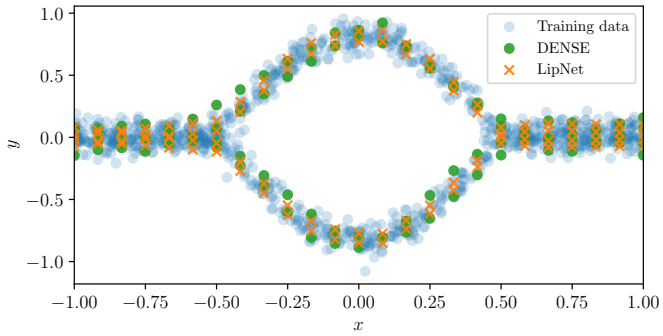

Fig. 8: Four Ensemble Members of DENSE vs. LipNet, scattered.

We compare our method's results to those of DE and LipNet. For DE and DENSE, we use a model architecture of one hidden layer with 50 neurons and ReLU activation. Both ensemble methods have 5 members and LipNet has 5 output atoms. To compare the ensemble methods with deterministic LipNet, we chose a comparable number of parameters and set up LipNet with 100 hidden neurons.

The DE method has two outputs per ensemble member (one for variance and one for mean), while DENSE only has one. This highlights the different approaches again: The heteroscedasticity needs to be modeled explicitly in DE, while DENSE encodes the total uncertainty in the distribution over its weights. LipNet encodes the total variance in the distribution of the output atoms, which is akin to DENSE, but resulting from a deterministic network instead of an ensemble.

An exemplary result comparison can be seen in Fig. 5. All methods capture the heteroscedasticity well. Averaged results over 10 trials are listed in Tab. I. The proposed method can keep up with the state-of-the-art method of DE, and further algorithmic improvement will also enable faster execution times. Execution time of DE corresponds to sequentially training the members, in a parallelized case this would be improved. Also, LipNet provides similar prediction quality at great speed.

In regions where no training data exists (out of distribution), one would like to see this situation quantified in the predictive uncertainty. We repeat the experiment with a modified dataset where data is missing between $x = 0.25$ and $x = 0.75$ and evaluate the predictive uncertainty of the three models. We can see in Fig. 6 that in the example case, only DENSE and DE are able to quantify the epistemic uncertainty in the out of distribution region. This is an important feature when it comes to quantifying the reliability of predictions and DENSE is able to provide it even without strict division between model and data uncertainty.

TABLE I: Results of heteroscedastic task.

Method	NLL	RMSE	Training time in s
DENSE	-1.33 $\pm$ 0.05	0.19 $\pm$ 0.01	25.38 $\pm$ 3.55
DE	-1.21 $\pm$ 0.28	0.20 $\pm$ 0.02	11.09 $\pm$ 0.11
LipNet	-1.33 $\pm$ 0.04	0.19 $\pm$ 0.01	2.47 $\pm$ 0.15

B. *Multi-Modal Regression*

The data generating process of our example is given by

$$f(y \mid x) = \mathcal{N}\big(y; g_1(x), \sigma^2\big) + \mathcal{N}\big(y; g_2(x), \sigma^2\big)$$

$$g_1(x) = \begin{cases} 0 & \forall x < 0.25 \text{ or } x > 0.75 \\ \sin(2\pi(x - 0.25)) & \text{otherwise} \end{cases}$$

$$g_2(x) = \begin{cases} 0 & \forall x < 0.25 \text{ or } x > 0.75 \\ -\sin(2\pi(x - 0.25)) & \text{otherwise} \end{cases}$$

For DENSE, the training leads to different members being responsible for representing different modes of the dataset, as is displayed for the extreme case of two ensemble members in Fig. 7. If more members than modes are available, multiple members represent a mode and its spread.

In practice, more members will be used to enable flexible and sufficient coverage of the complete dataset including its uncertainties. To keep the visualizations clear, Fig. 8 shows the case with four members of DENSE and in comparison four atoms in the predictions of LipNet. DE is not evaluated here, because it would fail as in Fig. 1 with its Gaussian assumed outputs. For both methods we kept the architecture from the heteroscedastic example. For LipNet, we specify the number of clusters to two. The comparison is shown in Fig. 8. We query both models for multiple input points and visualize the result as scatterplot. It can be seen that both predictions capture the underlying dataset well.

We evaluate the expected negative log-likelihood between predictions and the data-generating Gaussian mixture with two modes. Results can be seen in Tab. II. While performance is

TABLE II: Results of DENSE vs. LipNet on multi-modal task.

Method	NLL	Training time in s
DENSE	$3.54 \cdot 10^2 \pm 3.0$	29.12 ± 8.91
LipNet	$3.50 \cdot 10^2 \pm 2.9$	2.70 ± 0.05

similar, LipNet shows a big advantage in training time. As our implementation is not yet optimized, we expect to close up on that gap with improved implementation. However, we note that the performance and speed of LipNet is remarkable. Despite having a different background with respect to implementation and derivation, the method is a close relative to DENSE.

VII. Conclusion

Navigating the landscape of PNNs is challenging, and underlying assumptions have to be carefully considered. In this work, we addressed this problem by systematically categorizing PNN methods based on their ability to handle complex, multi-modal distributions. By decomposing the structural and probabilistic components of these models, we derived a simple and easy to deploy MLP-based PNN method. Our approach leverages a specialized loss based on the LCDd, achieving high predictive flexibility with minimal architectural overhead. While the current implementation demonstrates flexible performance on multi-modal and heteroscedastic datasets, it serves as a foundation for future improvements. Potential directions include enhancing numerical stability and execution speed, as well as scalability improvement by using subparts of the joint input-output space. Furthermore, extending this framework into a fully Bayesian method by including priors could enhance the quality of uncertainty quantification. Ultimately, this work describes the class of PNNs in terms of their predictive and learning capabilities and shows how an ablative approach can lead to methods that are easy to understand yet performant.

Acknowledgment

Thanks to Aaron Preus for his work on fast and accessible implementation of the LCDd, including C/C++ code and wrappers for Python and Julia. It is linked in the publication repository.

References

[1] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.

[2] Vincent Dutordoir, Hugh Salimbeni, James Hensman, and Marc Deisenroth. “Gaussian Process Conditional Density Estimation”. In: *Advances in neural information processing systems* 31 (2018).

[3] Lucas Berry and David Meger. “Normalizing Flow Ensembles for Rich Aleatoric and Epistemic Uncertainty Modeling”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.6 (June 2023), pp. 6806–6814.

[4] Michael P. Holmes, Alexander G. Gray, and Charles Lee Isbell. “Fast Nonparametric Conditional Density Estimation”. In: *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. UAI’07. Vancouver, BC, Canada: AUAI Press, 2007, pp. 175–182.

[5] Leon Winheim and Uwe D. Hanebeck. “BNN Training as State Estimation: A Progressive Filtering Approach”. In: *Proceedings of the 2025 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2025)*. College Station, Texas, Sept. 2025.

[6] Alex Graves. “Practical Variational Inference for Neural Networks”. In: *Advances in neural information processing systems* 24 (2011).

[7] Gareth O Roberts and Jeffrey S Rosenthal. “General State Space Markov Chains and MCMC Algorithms.” In: *Probability Surveys* 1 (2004), pp. 20–71.

[8] Rohitash Chandra and Joshua Simmons. “Bayesian Neural Networks via MCMC: A Python-Based Tutorial”. In: *IEEE Access* 12 (2024), pp. 70519–70549.

[9] Christopher M Bishop. *Mixture Density Networks*. Aston University, 1994.

[10] Cyril Benezet, Ziteng Cheng, and Sebastian Jaimungal. “Learning Conditional Distributions on Continuous Spaces”. In: *Journal of Machine Learning Research* 26.105 (2025), pp. 1–64.

[11] Bao Nguyen, Binh Nguyen, Hieu Trung Nguyen, and Viet Anh Nguyen. “Generative Conditional Distributions by Neural (Entropic) Optimal Transport”. In: *Proceedings of the 41st International Conference on Machine Learning*. ICML’24. Vienna, Austria: JMLR.org, 2024.

[12] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. “Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users”. In: *IEEE Computational Intelligence Magazine* 17.2 (May 2022), pp. 29–48.

[13] Andrew G Wilson and Pavel Izmailov. “Bayesian Deep Learning and a Probabilistic Perspective of Generalization”. In: *Advances in neural information processing systems* 33 (2020), pp. 4697–4708.

[14] Uwe D. Hanebeck. “Optimal Reduction of Multivariate Dirac Mixture Densities”. In: *at - Automatisierungstechnik* 63.4 (Apr. 2015), pp. 265–278.

[15] David Nix and Andreas Weigend. “Learning Local Error Bars for Nonlinear Regression”. In: *Advances in neural information processing systems* 7 (1994).

[16] José Miguel Hernández-Lobato and Ryan Adams. “Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks”. In: *International Conference on Machine Learning*. PMLR, 2015, pp. 1861–1869.

[17] Ethan Goan and Clinton Fookes. “Bayesian Neural Networks: An Introduction and Survey”. In: *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018* (2020), pp. 45–87.

[18] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles”. In: *Advances in neural information processing systems* 30 (2017).

[19] Alessandro Brusaferri, Matteo Matteucci, Stefano Spinelli, and Andrea Vitali. “Probabilistic Electric Load Forecasting through Bayesian Mixture Density Networks”. In: *Applied Energy* 309 (Mar. 2022), p. 118341.

[20] Warren Morningstar, Sharad Vikram, Cusuh Ham, Andrew Gallagher, and Joshua Dillon. “Automatic Differentiation Variational Inference with Mixtures”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 3250–3258.

[21] Daiwei Zhang, Tianci Liu, and Jian Kang. “Density Regression and Uncertainty Quantification with Bayesian Deep Noise Neural Networks”. In: *Stat* 12.1 (2023), e604.

[22] Philipp Wagner, Xinyang Wu, and Marco F. Huber. “Kalman Bayesian Neural Networks for Closed-Form Online Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.8 (June 2023), pp. 10069–10077.

[23] David J. Schodt, Ryan Brown, Michael Merritt, Samuel Park, Delsin Menolascino, and Mark A. Peot. *A Framework for Variational Inference of Lightweight Bayesian Neural Networks with Heteroscedastic Uncertainties*. Feb. 2024. Pre-published.

[24] James-A Goulet, Luong Ha Nguyen, and Saeid Amiri. “Tractable Approximate Gaussian Inference for Bayesian Neural Networks”. In: *Journal of machine learning research* 22 (2021), pp. 1–23.

[25] Riddhi Pratim Ghosh and Ian Barnett. *Bayesian Neural Networks vs. Mixture Density Networks: Theoretical and Empirical Insights for Uncertainty-Aware Nonlinear Modeling*. Oct. 2025. Pre-published.

[26] Uwe D. Hanebeck and Vesa Klumpp. “Localized Cumulative Distributions and a Multivariate Generalization of the Cramér-von Mises Distance”. In: *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. Aug. 2008, pp. 33–39.

[27] Alan Jeffares, Tennison Liu, Jonathan Crabbé, and Mihaela van der Schaar. “Joint Training of Deep Ensembles Fails Due to Learner Collusion”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 13559–13589.