entropy

MDPI

Article

Winsorization for Robust Bayesian Neural Networks

Somya Sharma [1] **and Snigdhansu Chatterjee** [2,*]

[1] Department of Computer Science and Engineering, University of Minnesota-Twin Cities, 200 Union Street SE, Minneapolis, MN 55455, USA; sharm636@umn.edu
[2] School of Statistics, University of Minnesota-Twin Cities, 313 Ford Hall, 224 Church St. SE, Minneapolis, MN 55455, USA
* Correspondence: chatt019@umn.edu

Abstract: With the advent of big data and the popularity of black-box deep learning methods, it is imperative to address the robustness of neural networks to noise and outliers. We propose the use of Winsorization to recover model performances when the data may have outliers and other aberrant observations. We provide a comparative analysis of several probabilistic artificial intelligence and machine learning techniques for supervised learning case studies. Broadly, Winsorization is a versatile technique for accounting for outliers in data. However, different probabilistic machine learning techniques have different levels of efficiency when used on outlier-prone data, with or without Winsorization. We notice that Gaussian processes are extremely vulnerable to outliers, while deep learning techniques in general are more robust.

Keywords: Bayesian neural network; uncertainty quantification; variational Gaussian process; Winsorization; concrete dropout; flipout; mixture density networks

check for updates

Citation: Sharma, S.; Chatterjee, S. Winsorization for Robust Bayesian Neural Networks. *Entropy* **2021**, *23*, 1546. https://doi.org/10.3390/e23111546

Academic Editors: Eric Nalisnick and Dustin Tran

Received: 22 October 2021
Accepted: 18 November 2021
Published: 20 November 2021

1. Introduction

Machine learning (ML) and artificial intelligence (AI) techniques have met astounding success in different industries and research problems. Conventionally, these techniques have the singular focus of improving prediction accuracy in complex data analysis problems. Despite the mass applicability and popularity of ML prediction methods, many of the related architectures fail to account for the fact that in many large datasets, there are potential outlying observations in both the target variable and the features. Unlike classical statistical frameworks involving relatively small datasets with few features, it is not possible in big data to carefully select and then either drop or modify observations in a pre-processing step prior to the main data analysis. In any case, such ad hoc pre-processing steps can lead to a violation of standard regularity conditions that are required for a proper probabilistic analysis [1,2]. This is essentially a result of using the data twice, once for outlier detection and then again for constructing the predictive model, whereby there is a false sense of accuracy and precision for the second step. Similar issues have been noted in the context of model selection and other problems also, see [3] and related literature for deep theoretical discussions and results.

The problem of outliers in the data is exacerbated when such data are used with deep learning (DL) or related black-box techniques that are supremely versatile. Because of the inherent strengths of these techniques, they may yield excellent numeric summaries such as mean squared errors even on data with outliers by simply overfitting near such aberrant observations. Such aspects of DL fitting have been observed earlier and are of considerable interest in studies on the properties of DL [4,5]. In essence, standard outlier detection techniques such as studies on residuals are not operative owing to localized overfitting by the DL architecture, and the use of robust model fitting procedures are not viable because they scale poorly with data size or parameter size and hence pose extremely burdensome computational requirements.

In this paper, we propose a probabilistic Winsorization step on the training data, to mitigate the adversarial effects of model learning on noisy and outlier-prone data. Consider, for a moment, a numeric dataset on a single variable. In this case, the data can be ordered, say, in an increasing order. Winsorization is a process where the highest and lowest α-fraction of the observations are replaced by their nearest neighbors in the remaining "central" $(1 - 2\alpha)$-fraction of the data. For example, if there are 100 observations in an increasing order and if $\alpha = 0.05$, we replace each of the smallest five values with the sixth lowest value and each of the highest five values with the 95-the highest value. Thus, the original data size remains intact, outliers are dropped from the data, human-centric pre-processing of the data is not needed, and complex mathematical formulations and optimizations are not required to ensure robustness. The value of α can be chosen using a trade-off between efficiency and robustness, or some other criteria, and it is trivial to generalize to the case where different fractions of observations are selected to be replaced from the upper and lower tails. A minor generalization is achieved when some random noise is added to each of the 2α replacement observations that are used in place of the highest and lowest actual data or some other systematic transformations used on these. The statistical properties of hyperparameter estimators, predictions, and inferences from Winsorized data are not substantially different from the case where the upper and lower α-fractions of the data are not used in model fitting, for example, as in the case of using *trimmed least squares* instead of ordinary least squares. Very importantly for big data studies, Winsorization in each variable leaves the database architecture and structure of data tensors unaltered throughout, which leads to computational simplicity.

In this paper, several different probabilistic and Bayesian ML and AI methods are studied—each of which derive their probabilistic nature from different aspects of a neural network architecture. We primarily conduct thorough empirical studies on several datasets in this project. For each probabilistic ML technique, we conduct a four-fold study of each dataset. First, we use the data as it is, unaltered. Second, we introduce independent and identically distributed Cauchy random variables in each of the target observation, thus creating an extremely noisy target with potentially several outliers of different magnitudes. The feature set is left intact. Third, we introduce independent and identically distributed Cauchy random variables in each observation in each feature, but leave the target intact. Fourth, we introduce independent and identically distributed Cauchy random variables in each of the target observations as well as each observation of all the features. Thus, the four versions of each dataset represent a (*i*) *no noise* scenario, (*ii*) a *noise in target* scenario, (*iii*) a *noise in features* scenario, and (*iv*) a *noise in target and features* scenario. Then, we analyze the dataset versions both as they are and when Winsorization is used to eliminate outliers. The goal of this study is to understand how probabilistic outliers affect the results from black-box ML techniques and how Winsorization may be used to greatly ameliorate the problem.

The rest of this paper is organized as follows: In Section 2, we discuss the related methods to introduce perturbations and train in the presence of adversarial noise. In Section 3, we discuss the different Bayesian deep learning methods that are employed for a comparative analysis. Section 4 outlines the experimental setup and results from the experiments. We discuss our findings in Sections 5 and 6.

2. Related Literature

In several applications in computer vision, natural language processing and machine perception, deep neural networks have achieved remarkable performance. However, in the presence of even small perturbations in the training samples, the performance deteriorates quickly [6–8]. This instability makes it imperative to study the robustness of deep learning methods.

For instance, robustness analysis in domains such as natural language processing (NLP) often focuses on introduction of adversarial examples [9] while training neural networks. These multi-scale adversarial perturbations range from character-level to sentencelevel perturbations. Several studies [10,11] also compute forward gradients of input sequences to guide the search for modifications that would introduce perturbation. Studies [10,12–15] on which a word's addition or omission adversely affects model performance have shed light on the importance of different words in recovering from the adversarial attacks. Attacks based on gradient computation and insertion of perturbation in embedding space are susceptible to vanishing or exploding gradient problems. While, there are several black-box methods for adversarial attacks, their reproducibility in the NLP domain is also known to be limited [9]. To overcome these adversarial attacks, either the adversarial examples are identified and separated from the training set or the model architecture is modified to accommodate for the additional difficulty in learning. Adversarial example detection relies on recognizing known words and treating perturbations as unknown words that are not used for training [16,17]. Model modification based methods, on the other hand, generally include the adversarial examples during training [18–20]. The way similarity and dissimilarity are computed between adversarial examples and perturbation-free samples can also shed light on how can we optimally distinguish between the two. Through clustering of the embeddings of input words, shared encoding among similar embeddings can be used to differentiate the noise from the words [21]. Studies on constraining the perturbation in input have helped in development of certifiable defenses [22–24]. This certification can also be incorporated as an objective to create an adaptive regularizer that enhances the robustness and stability of the model [25]. Several studies also focus on Interval Bound Propagation (IBP) that propagate some verifiable input-output properties [26,27]. There is also evidence suggesting that overfitting on training data in overparameterized regimes adversely affects the performance of neural networks in presence of adversarial perturbations. The performance gains owing to all the adversarial training frameworks can be achieved by early stopping as well [28].

Similar to the application in the NLP domain, perturbation-based robustness in deep learning also utilizes adversarial training [29]. For a loss function of the form $l(\mathbf{x}, \mathbf{y}; \mathbf{W})$, $\mathbb{E}$ refers to the expectation value, where $\mathbf{W}$ is the weight vector parameterizing a neural network, the optimization problem can be stated as follows [30]:

$$\mathbf{W}^* \in \arg\min_{\mathbf{W}} \mathbb{E}[\max_{\delta \in \Delta} l(\mathbf{x} + \delta, \mathbf{y}; \mathbf{W})] \tag{1}$$

where the perturbation is norm-bounded as, $\Delta = \{\delta : ||\delta|| \leq \epsilon\}$. For a worst case perturbation problem, we want to find δ such that it maximizes the loss function, and we also want to find the $\mathbf{W}$ array that minimizes the empirical risk.

Apart from artificially induced perturbations as listed above, there are several forms of natural perturbations in real world data as well. While it is important to study the synthetically generated perturbations to understand neural networks better, these are limited in their applicability due to their norm-bound restrictions and dearth in real world datasets, especially when data distribution shifts as it does in scenarios such as changing weather conditions [30]. In computer vision as well, naturally occurring noise in the form of blurring effects and distorted lights are less likely to be fully accounted for using norm-bounded perturbations [31]. More recently, model-based robust optimization methods are being employed to incorporate a natural variation that may take the form $G(\mathbf{x}, \delta)$ and may already be a part of the training examples in the form of blurring and distortion, for instance. The objective, in this case, will be of the following form [30]:

$$\min_{\mathbf{W}} \mathbb{E}[\max_{\delta \in \Delta} l(G(\mathbf{x}, \delta), \mathbf{y}; \mathbf{W})] \tag{2}$$

$G(\mathbf{x}, \delta)$ is called the model of natural variation and can be non-linear in δ. The model can be of encoder–decoder form as well allowing for learning an encoding for G.

Similar to perturbations, outliers can potentially contaminate any data analysis and provide misleading results. A lot of the recent attention has been placed on removing the outliers by truncating or trimming them. While this omission may be useful in removing the influence of extreme values, it still leads to a loss of information. Alternatively, the dataset can be "clipped" or "Winsorized" by replacing the extreme values with more central samples. Winsorization aids in managing the adverse effects of outliers in the data by clipping the extreme values. There have been several studies into efficiency and bias comparisons of Winsorized mean estimators [32,33] and Winsorized regression [34] where the residuals are Winsorized. Due to its simplicity, Winsorization may hold more appeal as compared to other robust regression techniques. There are also studies that attribute instability caused by perturbations not just as a shortcoming of model deep learning frameworks but also an attribute of adversarial training examples and the data itself. Outliers and perturbations in data make it difficult for the model to learn as the model regards both the clean samples and perturbed samples as equally important for training at the start of the learning process [35,36]. Winsorization as a data treatment addresses this concern and focuses on eliminating the outliers before the training begins. This way the association between target and features is preserved for the model to learn.

While the quantiles and median of error taken from the observed target might be robust towards outliers and make it favorable to incorporate in the learning objective, these functions are not differentiable, making them an intractable choice for gradient based learning. Recent advances in the space of robust neural networks learning include the use of M-averaging functions over the mean in the empirical risk estimation [37]. As an approximation of quantiles, a differentiable parametric family of M-average functions can be used such that they satisfy certain differentiability constraints and can act as surrogate for quantiles of loss, independent of outliers.

Winsorization has been used on regression problems using deterministic neural networks [38,39]. Asymptotic properties of trimmed and Winsorized M and Z estimators have been investigated and trimmed M-estimators have been used for robust estimation in neural networks [40]. As opposed to trimming or Winsorization of residuals, we seek to understand Winsorization as a treatment method on the training dataset. The aim of our study is to understand the effect of Winsorization on perturbed input data that are used to train probabilistic neural networks and investigate if Winsorization can aid in producing stable prediction results in probabilistic neural networks.

3. Methodology

We now briefly discuss the different probabilistic machine learning methods we study in this paper.

3.1. Exact Gaussian Processes

One of the most prominent and relatively simpler technique to use in prediction and inference in the presence of unknown functional relations between variables is the *Gaussian Process* (GP) modeling approach. This is a Bayesian approach which models unknown functions as Gaussian stochastic processes, that is, the evaluation of the unknown function on any finite collection of points in the feature space is assigned a multivariate Gaussian prior, and a posterior prediction is obtained by coupling the observed data with this prior. GP modeling is a non-parameteric regression approach with uncertainty quantification. GP regression can be fully specified using a mean and a covariance function that can be used to define a Gaussian probability distribution from which the predictions are drawn. GP is adept at capturing non-linear relations between the feature set and the prediction function using a non-linear covariance function kernel. A function $f : \mathcal{X} \to \mathbb{R}$ is modeled as a Gaussian Process with mean function m and covariance function k and can be written as follows [41]:

$$f \sim \mathcal{GP}(m(x), k(x)), \tag{3}$$

if for any finite integer $k \geq 1$ and any collection of points $x_1, x_2, \ldots, x_k \in \mathcal{X}$, the k-dimensional random variable $\big(f(x_1), \ldots, f(x_k)\big)$ has a multivariate Gaussian distribution with mean $\big(\mu(x_1), \ldots, \mu(x_k)\big)$ and a covariance matrix Σ whose (i, j)-th element is $k(x_i, x_j)$.

If $\mathbf{f}$ are the function values for the training set and $\mathbf{f}_*$ is the set of function values on the test set $\mathbf{X}_* \subset \mathcal{X}$, the joint distribution can be specified as follows:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^T & \Sigma_{**} \end{bmatrix}). \tag{4}$$

Here, Σ_* represents the train-test set covariance while Σ_{**} represents the test set covariance. The conditional distribution of $\mathbf{f}_*$ given $\mathbf{f}$ is as follows:

$$\mathbf{f}_*|\mathbf{f} \sim \mathcal{N}(\mu_* + \Sigma_*^T \Sigma^{-1}(\mathbf{f} - \mu), \Sigma_{**} - \Sigma_*^T \Sigma^{-1} \Sigma_*) \tag{5}$$

More often than not, the mean function m and covariance function k involve hyperparameters, θ, that are tuned by maximizing the logarithm of marginal likelihood. The marginal likelihood gives us the probability of observing the data samples given hyperparameter values and is of the following form:

$$log\, p(\mathbf{y}|\mathbf{x}, \theta) = -\frac{1}{2} log|\Sigma| - \frac{1}{2}(\mathbf{y} - \mu)^T \Sigma^{-1}(\mathbf{y} - \mu) - \frac{n}{2} log(2\pi). \tag{6}$$

Partial derivatives of Equation (6) give us the gradient estimate update rules for the hyperparameters of the mean and covariance functions whose values can be calculated using an iterative numerical optimization technique. The exact GP has been proven to be very successful in several empirical use cases. Depending on the different kernel functions, the definition and shape of similarity that is encoded through the kernel function can be changed. Exact GP, however, becomes intractable for extremely big datasets as the computation cost scales by $O(n^3)$, while storage scales by $O(n^2)$ as n, the number of training samples, increases [41]. Therefore, several approximation methods have been devised recently to improve the scalability of Gaussian processes.

3.2. Variational Gaussian Processes

Exact Gaussian processes models are capable of utilizing high-dimensional feature sets for the modeling response. However, with increasing sample sizes in the wake of the advent of big data, the computational burden involved in defining the kernel function may increase in a super-linear way and as a function of the sample size. Therefore, it may be of interest to explore a more sparse kernel function [42,43] that can be defined in a more efficient manner, such as in case of Variational Gaussian Processes (VGP). Variational inference [44–46] also improves the efficiency in approximating the posterior predictive function. In this paper, we use a variational Gaussian process approach that renders the output of a deterministic deep neural network (DNN) as probabilistic.

Based on the mean field variation inference theory and use of hidden variables to encode representations from the observational data to obtain the posterior conditional distribution [47], a practical method utilizing sparser covariance structure is proposed to obtain a variational Gaussian process framework for big data [45]. Let $\mathbf{u}$ represent a vector of function values at a subset of samples $\mathbf{Z} = \{z_i\}_{i=1}^m$ from $\mathbf{x}$ called inducing variables. These inducing variables can be utilized to create a model that is consistent with the application of stochastic variational inference [47].

Marginalizing the inducing variables in the work [48], the lower bound on $log\, p(\mathbf{y}|\mathbf{x})$ can be obtained as follows [45]:

$$\mathcal{L} = log\, \mathcal{N}(\mathbf{y}|, \mathbf{0}, \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn} + \beta^{-1}\mathbf{I}) - \frac{1}{2}\beta\, tr(\tilde{\mathbf{K}}) \tag{7}$$

where β is the precision of the original probability distribution of response conditioned on function $\mathbf{f}$. Inducing variables $\mathbf{u}$ perform the role of global variables in applying a stochastic variational inference to a Gaussian process model. These are used to further lower the bound on $p(\mathbf{y}|\mathbf{x})$. The updated lower bound becomes the following:

$$\mathcal{L}' = \sum_{i}^{n} \{ log \mathcal{N}(y_i | \mathbf{k}_i^T \mathbf{K}_{mm}^{-1} \mathbf{m}, \beta^{-1}) - \frac{1}{2} \beta \tilde{k}_{i,i} - \frac{1}{2} tr(\mathbf{S} \mathbf{\Lambda}_i) \} - KL[q(\mathbf{u}) || p(\mathbf{u}))]. \tag{8}$$

The partial derivatives of $\mathcal{L}'$ provide us with estimates for the kernel hyperparameters and the noise precision β. Furthermore, a stochastic gradient estimate can be performed to obtain the optimal values for the variational parameters. The factorization of $\mathcal{L}'$ enables performing stochastic gradient methods on $q(\mathbf{u})$ and also the use of non-Gaussian likelihoods for inference.

3.3. Concrete Dropout

Unlike the conventional use of dropout [49] to improve generalization power by sampling neurons during training, we can derive an empirical predictive distribution by using the layer-wise dropout relaxation during the testing process [50,51] where the variance of the predictive distribution is generated by randomly dropping neurons at test time using the optimal dropout rate. The optimal dropout rate can be found either by a grid-search over dropout probabilities [50], which can be prohibitive when computational resources are constrained. In this work [50], dropout is used to obtain an approximation to a probabilistic deep Gaussian process [52].

Similar to an L_2 regularization objective for dropout, variational parameters vector θ_i for each layer i can be regularized to achieve a model that reduces the Kullback–Leibler (KL) divergence of the weight distribution with the true Bayesian posterior as follows:

$$\begin{aligned} \mathcal{F}_{GP-MC} &\propto \quad KL[q(\mathbf{W}|\theta)||P(\mathbf{W})] - \mathbb{E}_{q(\mathbf{W}|\theta)}[logP(\mathcal{D}|\mathbf{W})] \\ &\propto \quad \tfrac{l^2}{2\tau N} \textstyle\sum_i^L (p_i ||\theta_i||_2^2 + ||m_i||_2^2) - \tfrac{1}{\tau N} \textstyle\sum_n^N logp(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}) \end{aligned}$$

where m_i refers to the bias terms in each hidden layer i, p_i is the dropout probability, l is length-scale, and τ is the precision hyperparameter.

Alternatively, a more efficient way of learning p_i instead of doing a grid search is by setting layer-wise dropout rates as trainable and by learning them via the standard backpropagation process along with other neural network parameters. This method is called concrete dropout [51]. Similar to the KL divergence term in the grid-search scenario [50], the KL divergence term for the dropout rate estimation by variational free energy optimization may also include the variational parameters, thus:

$$KL[q(\mathbf{W}|\theta)||P(\mathbf{W})] \propto \frac{l^2(1-p)}{2} ||\theta||^2 - K\mathcal{H}(p) \tag{9}$$

where K is the dimension of weight vector for each layer and $\mathcal{H}(p)$ is the entropy of dropout probability, which is a Bernoulli random variable in this case:

$$\mathcal{H}(p) = -plog(p) - (1-p)log(1-p). \tag{10}$$

Moreover, a concrete relaxation of dropout masks makes it possible to obtain the optimal dropout probability value for each layer by pathwise derivative estimation [51]. If $u \sim Unif(0,1)$ and t is a temperature value, then the concrete distribution random variable will be of the following form:

$$\tilde{\mathbf{z}} = \text{sigmoid}(\frac{1}{t}(log\,p - log(1-p) + log\,u - log(1-u))) \tag{11}$$

Concrete dropout does not require a lot of additional compute as compared to a standard dropout implementation and is more efficient than a grid-search to find optimal dropout probability value.

3.4. Flipout Estimator

Historically, there have been several advances in the field of regularization to overcome overfitting [49,53–56]. Some of these methods include Gaussian perturbations [57] and DropConnect [58] as methodologies for perturbing weights for regularization. Interestingly, while these methods were originally formulated for regularizing artificial neural networks (ANNs), the resulting stochasticity in weights also allows for the quantification of uncertainty in weights to some extent. Bayesian neural networks have been created by perturbing the weights in different hidden layers and trained using variational inference after applying reparameterization trick as elucidated in [59] that makes use of *Bayes by backprop* possible [60]. For an unknown weight parameter $\mathbf{W}_{ij}$ for i^{th} layer and j^{th} node, $\mathbf{W}_{ij}$ is drawn from $\mathcal{N}(\mu_{ij}, log(1+exp(\Sigma_{ij}))^2)$, where $\theta_{ij} = (\mu_{ij}, \Sigma_{ij})$ are variational posterior parameters that are trained via standard backpropagation. This is often done through the introduction of a non-parametric noise distribution, $\epsilon \sim \mathcal{N}(0,1)$, that is then scaled and shifted as follows:

$$\mathbf{W} = \mu + log(1+exp(\Sigma)) \odot \epsilon. \tag{12}$$

Here, using $log(1+exp(\Sigma))$ ensures that this term remain positive and differentiable. Variational free energy [57,61–64] is minimized to estimate the parameters θ of the weight distribution such that the Kullback–Leibler (KL) divergence with the true weights posterior is minimized [57,60]. Similar to previous work on weight-based uncertainty [60], the objective function remains as follows:

$$\mathcal{F}_{Flipout} = KL[q(\mathbf{W}|\theta)||P(\mathbf{W})] - \mathbb{E}_{q(\mathbf{W}|\theta)}[logP(\mathcal{D}|\mathbf{W})], \tag{13}$$

which intends to minimize the negative log likelihood based on the data and the complexity cost of fully encoding the functional relation as a very complex $\mathbf{W}$. This cost not only optimizes for the best weights for predicting our target but also optimizes for the simplest weight representation that we can get. The cost wants to reduce the number of bits required to transmit weights to a receiver, and this is the complexity cost of the weights and the second component of the loss function wants to minimize the number of bits required to transmit the errors in the model. These additive costs are together called compression cost [60] or minimum description length [57]. Through this cost function, we can ensure that the weight distribution is not too complex and does not overfit. Another form of Equation (13) uses an approximate cost function for complexity cost that makes the computation more efficient. Introduction of the perturbation ensures that the gradient estimates of the cost are unbiased [60]. Monte Carlo samples of $\mathbf{W}$ are drawn from the variation posterior distribution can be used for calculating the approximate cost.

In a similar fashion, adding parametric noise to weights has also aided in efficient exploration of optimal agent policies in reinforcement learning [65–67]. Depending on the network size, Gaussian noise added to the weights may be either independently used or factorized as well [67]. This means that for p inputs to a layer and q outputs, adding independent Gaussian noise to each weight will yield $(p+1)q$ noise variables, while using factorized Gaussian noise would lead to individual noise weights for the noise and for the outputs such that only $p+q$ noise variables are employed for a hidden layer. However, having the same Gaussian perturbation for a mini-batch may lead to some correlation in gradients and lead to high variance in the gradient estimates [68].

the flipout gradient estimation technique [68] can be used to de-correlate the gradient estimates. Like all the previously proposed methods, flipout also has a base noise that affects the $\triangle\mathbf{W}$ where the base noise is drawn from a unit Gaussian distribution. This is similar to the previous weight based uncertainty [60]. Any other method that uses weight perturbation for weight-based uncertainty can be used as the method to obtain the gradient estimates that can be further de-correlated using flipout. Let the gradient estimates obtained from any of the previous work be $\triangle\hat{\mathbf{W}}$. $\triangle\hat{\mathbf{W}}$ in a mini-batch are still correlated. In order to de-correlate, we make use of randomly drawn sign matrices. If r_n

and s_n are two random vectors drawn from ±1 uniformly, the flipout perturbation can be given as follows:

$$\triangle \mathbf{W}_n = \triangle \hat{\mathbf{W}} \odot r_n s_n^T. \tag{14}$$

Here, the n subscript represents each training example in a mini-batch. These additional matrix multiplications with sign matrices have been shown to decorrelate the gradient estimates and lead to variance reduction. Therefore, the convergence to optimal variational parameter values happens earlier. However, having the additional matrix multiplication with the sign matrices also leads to higher computational cost which may be offset by parallelizing computations through evolution strategies, similar to the ones used in reinforcement learning [66,69,70]. Similar to [57], evidence lower bound (ELBO) is optimized to obtain estimates for θ as is done in Equation (13). Through empirical evidence, it has been shown that flipout is able to reduce the variance in stochastic gradient estimates [68]. However, it is also stated that as compared to vanilla implementation of *Bayes by backprop*, flipout is 60 times more computationally expensive [68].

3.5. Mixture Density Networks

A mixture density network [71] allows for the response distribution to be a mixture of distributions, essentially allowing for multiple distributions to weigh in with varied degrees such that the response is multimodal. The parameters of each of these distributions are estimated using a fully connected artificial neural network. For solving inverse problems that often involve one-to-many mappings or for estimating multimodal distributions, a mixture density network (MDN) proves to be an effective regime. The conditional probability of obtaining a response value by mixing k Gaussian components can be given as follows [72]:

$$p(\mathbf{y}|\mathbf{x}) = \sum_k \pi_k(\mathbf{x}) \mathcal{N}(\mathbf{y}|\mu_k(\mathbf{x}), \sigma_k^2(\mathbf{x})). \tag{15}$$

Here, for each Gaussian component k, the parameters π_k, μ_k, and σ_k^2 are estimated via a neural network with k outputs. π_k is the contribution of the k^{th} mixing component, μ_k, and σ_k are the mean and standard deviation of the Gaussian distribution that decide the mixing component's density distribution. In addition, since σ_k^2 depends on the input $\mathbf{x}$, this is considered a heteroscedastic model. If $\mathbf{W}$ represents the weight vector that specifies the neural network, Equation (15) can be written as follows:

$$p(\mathbf{y}|\mathbf{x}) = \sum_k \pi_k(\mathbf{x}, \mathbf{W}) \mathcal{N}(\mathbf{y}|\mu_k(\mathbf{x}, \mathbf{W}), \sigma_k^2(\mathbf{x}, \mathbf{W})). \tag{16}$$

The weights $\mathbf{W}$ can be estimated by minimizing the negative logarithm of likelihood which will be our loss function that we will optimize for. The loss function is of the following form [72]:

$$\mathcal{F}_{MDN} = -\sum_i log\{\sum_k \pi_k(\mathbf{x}_i, \mathbf{W}) \mathcal{N}(\mathbf{y}_i|\mu_k(\mathbf{x}_i, \mathbf{W}), \sigma_k^2(\mathbf{x}_i, \mathbf{W}))\}. \tag{17}$$

Minimizing Equation (17) gives us the conditional density function of the response. Varying the number of mixing components k may also be an additional tuning hyperparameter as it decides the number of modalities in the dataset.

3.6. Winsorization

In Winsorization, the extreme values are replaced with more centrally located representative values. The samples within the first α percentile are replaced with the α^{th} percentile sample, and the samples beyond the $1-\alpha$ percentile are replaced with $(1-\alpha)^{th}$ percentile value. This is similar to clipping that is used in gradient vanishing/explosion problems. By bounding the dataset to exclude the outliers, the adverse effects of perturbations on the training are attenuated.

For a given input random variable X, the values in n training examples can be written in the form of ordered statistics as below:

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(\alpha n)} \ldots \leq X_{((1-\alpha)n)} \ldots \leq X_{(n-1)} \leq X_{(n)}. \tag{18}$$

After Winsorization, the sequence will be modified as follows to include more of centrally located samples:

$$X_{(\alpha)} \leq \ldots \leq X_{(\alpha)} \leq \ldots \leq X_{(1-\alpha)} \leq \ldots \leq X_{(1-\alpha)}. \tag{19}$$

As a methodology for limiting the effect of outliers, Winsorization can be applied on the training dataset for more robust learning in probabilistic neural networks. To test for robustness, we induce artificial perturbations in the training and validation set in features and targets by adding standard Cauchy noise. The model performance is evaluated on an untouched test set with no perturbation and a test set with perturbations. Winsorizing the training data ensures that the model is able to learn on more central values in feature and response space.

In the presence of outliers, we have the following:

$$sup\,|Y - \hat{Y}| \geq sup\,|Y - \hat{Y}_W| \tag{20}$$

where $\hat{Y}_W = f(X_W, Y)$ or $\hat{Y}_W = f(X, Y_W)$ or $\hat{Y}_W = f(X_W, Y_W)$. Here, f is the learned neural network based estimator for the response Y based on the input X. In the training dataset, if either the input space or the response space are clipped, it affects the learning of the neural network model.

4. Results

The performance of the models are evaluated on five datasets. Apart from comparing the performance of different probabilistic modeling methods, the experiments also provide insights into the use of Winsorization in the training stage of neural network modeling.

4.1. Datasets

4.1.1. Precision Agriculture Case Study: Crop Yield Prediction in the US Midwest

Historically, process-based biophysical models and classical statistical models have been employed for crop yield prediction. Process-based models [73–77] study physiological and physical processes to simulate crop yield. Often, simpler statistical models [75,78,79] are used owing to their straightforward reporting of goodness of fit metrics. Many of these models may be constrained by their strict and often unrealistic assumptions to control multi-collinearity and spatio-temporal dependence [80]. Additionally, process-based and simple statistical models often miss or exclude non-linear terms, which may prevent them for being useful for yield predictions under extreme climatic conditions. Machine learning methods present the opportunity to model agricultural data using more complex architectures, using fewer assumptions, and on larger datasets. Artificial neural networks [81–89], linear regression [87,88,90–93], tree-based models [87,92–98], and support vector machines [98–100] are some of the most used machine learning algorithms [101,102] for crop yield modeling. In particular, ANNs have been used for tasks such as species recognition, weed detection, or crop quality assessment in [81–89] and elsewhere, using a variety of complex features including satellite data.

In applications of statistical machine learning modeling to climate sciences and precision agriculture, it is important to incorporate spatio-temporal dependencies between multiple samples. Spatial and spatio-temporal statistical models may be used to capture such dependencies explicitly but are very sensitive to the stringent assumptions made for such models, and the computations do not scale with data size in these cases. Ignoring possible non-linear, complex functional relations might lead to considerable over-estimation of the uncertainty in prediction and a loss of statistical power for feature selection and risk bounds, which has severe consequences for downstream industries such as that of crop insurance. More alarmingly, under-estimating uncertainty may lead to misleadingly narrow uncertainty bounds that may be centered at an inaccurate and biased estimate. In view of these issues, this work focuses on using easily available within-season meteorological variables for rapid and efficient usability and ensures that ANNs capturing non-linear functional components are used to tackle all of the above listed concerns. Several probabilistic modeling techniques are explored in this section.

Studying the effect of variations in weather on human crop yield is important in mitigating production and economic losses in agrarian economies. This observed effect is conspicuous and well studied in [103,104]. Extreme weather events in the US midwest affect crop yields, food price hikes, and can lead to production losses—the effect being as high as 75% for some Minnesota counties [103]. Formulating data-driven methods is imperative for more efficient planning in precision agriculture and for public policy. In view of this, our experiments are performed on a climate dataset comprising of county-level end-of-year crop yields in the Midwest USA and daily meteorological variable readings. The data include daily maximum temperature readings, minimum temperature readings, and average precipitation readings. These help us in gauging how the end-of-year yield changes in a county depending on the climatological features and location. The feature set includes daily maximum temperature readings (365 features), daily minimum temperature readings (365 features), daily average precipitation readings (365 features), longitude and latitude (2 features), and cosine and sine transformation of these location coordinates (4 features). The meteorological data are from the PSL public dataset [105], and the corn crop yield data are from the USDA public dataset [106]. The dataset is limited to Minnesota and Illinois for clarity of presentation.

4.1.2. California Housing Data

The California housing data [107] are a well-known public dataset. The response or target variable is the median house value for different California districts collected in the 1990 U.S Census. There are eight features including median income of the block, median house age, average number of rooms, spatial coordinates, and population in the block, where a block is the smallest geographical unit in the Census. Algorithmic bias can unduly affect the model performance of predictive models. Especially, there may be confounding variables that lead to the model favoring certain sections of the population, and it is imperative to understand the extent of influence of this bias. Using probabilistic deep learning models ensures that the some transparency is lent to the black-box model in making predictions.

4.1.3. Data on Forest Fires in Portugal

The dataset on forest fires [108] contain information on the area burned by forest fires in Portugal using meteorological data. The response variable is a scaled burned area variable. The features include spatial coordinates, temperature, humidity, rain, wind, information from the Fire Weather Index (FWI) system, and time-based variables. Many times, the burned area is an estimate based on surveying methods. It is also possible that the feature values such as humidity and temperature are often measured in a metropolitan location instead of at the center of the forest area. These may make our estimates more biased due to lack of accurate feature information. Skewed observed target values and noisy data can have critical implications on the human lives and wildlife that are affected by wild fires each year. Obtaining uncertainty estimates will allow policy makers to analyze all probable prediction scenarios.

4.1.4. Mauna Loa CO_2 Data

This dataset consists of time series atmospheric CO_2 concentration variations data as collected by NOAA at Mauna Loa, Hawaii [109]. The data consists of monthly atmospheric CO_2 concentration readings from the year 1958 through August 2021. Climate change has severe implications on the green house gas concentration in the environment. Bayesian deep learning methods enable the revision of our known knowledge and provide better forecasts for climate-change-driven planning.

4.1.5. Data on Genomics of Drug Sensitivity in Cancer

Genomics of Drug Sensitivity in Cancer (GDSC) [110,111] study involves studying the sensitivity of cancer cells to anti-cancer drug as affected by the gene expressions of different patients. The GDSC database from which the data is derived curates over thousands of genetically characterized human cancer cells and information about biomarkers.

Each cancer patient responds differently to anti-cancer drug treatment. The objective is to explore therapeutic biomarkers that may be used in identifying patients that are more likely to respond to anti-cancer drug treatment. In precision oncology, we are interested in providing the correct drug treatment to the right cancer patients based on these biomarkers of drug sensitivity.

In the dataset, the response variable is the sensitivity to YM155 (Sepantronium bromide) as the natural logarithm of the fitted IC50 and the feature information consists of expression of 238 genes. The dataset samples are drawn from 4 cancer types and 148 cell lines. Precision in medicine can improve human mortality and uncertainty-guided model inference can lead to more reliable research outcomes in healthcare.

4.2. Architecture

We use a fully-connected deterministic neural network as the baseline model. It has 14 layers and ReLU activation for the first 13 layers. The number of hidden layers are 52, 250, 300, 450, 202, 452, 50, 300, 200, 52, 400, 350, 450, and 450 and 1 for the output layer. This network is chosen via Bayesian optimization. This involves using expected improvement as a surrogate acquisition function. This function is the objective function for neural network hyperparameter optimization. Over several trials using the surrogate function with different hyperparameter sets, we choose the model with optimal hyperparameter set, and this chosen hyperparameter set can be expected to accurately correspond with the maximized original objective function. The expected improvement for $(x_i, y_i)_{i=1}^{n}$ samples and $[a]^+ = max(0, a)$ is given by:

$$\text{Expected Improvement}_n(x) = E_n[[f(x) - f_n^*]^+].$$

More information about Bayesian hyperparameter tuning can be found in [112].

Concrete Dropout. One of the Bayesian neural networks we experiment with implements the concrete dropout methodology on all layers. The dropout rate is learned during the training stage, and the contribution of different hidden neurons is varied by dropping neurons during the testing stage. Mean and variance of prediction distribution are the outputs of the neural network. The ELBO function is modified to include the concrete relaxation for dropout rate. This allows for a more efficient search for optimal dropout rate and probabilistic model training as opposed to Monte Carlo dropout learning based on grid search.

Variational Gaussian processes (VGP). Another implementation uses a variational Gaussian process to estimate the target. The first 14 layers learn weights in a deterministic manner. The VGP learns the posterior predictive distribution of the target and receives its input from the sequence of fourteen fully connected layers. Instead of full covariance we use a sparser representation using inducing variables. An RBF kernel is used with amplitude and lengthscale hyperparameters. The method [45] is different from previous variational GP implementations in that the inducing variables and the kernel hyperparameters are jointly optimized using gradient descent-based updates.

Flipout Gradient Estimator. In the flipout implementation, gradients in all 14 layers are estimated using the flipout method [68]. The KL divergence of the bias and kernel surrogate posteriors and priors are minimized during training. Flipout enables variance reduction in the gradient estimates by minimizing the KL divergence of the weights' variational posterior with the prior on the weights. Apart from minimizing the complexity cost, we are interested in improving the expected logarithm of the likelihood of observing the data when the weights are sampled from the variational posterior distribution. The objective function comprises of both these terms and is called the compression cost. The gradient estimates are taken with respect to an approximate version of the variational free energy cost where the weight values are sampled using Monte Carlo sampling from the variational posterior.

Mixture Density Networks. Here, the mixture components are estimated via the fully connected neural network. The estimated parameters are used to estimate the response using Gaussian mixture models. The mixing coefficients as well as the mean and variance of the Gaussian components' density are estimated using the neural network. The gradient estimates of the negative log likelihood function are used to update the parameter values.

Evaluation Metrics

For training purposes, the datasets are split into training, validation, and test sets, which comprise of 70%, 10%, and 20% of the data. Since all the experiments essentially relate to regression problems, we use mean squared error as our metric of comparison. We also report the median absolute error which provides insight into the model performance in presence of outliers. The total time for training one instance of the model is reported. We also report the coefficient of determination and the total number of parameters in the network architecture. Apart from the methodology-based change in number of model parameters, the number of parameters also vary depending on the number of input features for the different datasets. The effects of adding artificial perturbations to the dataset and using Winsorization to mitigate the effect of the noise is also studied. The evaluation metrics are reported for different sites for noise and for different degrees of Winsorization. The effect is tested on a test set that is free from noise (original, untouched test set) and a test set that has standard Cauchy noise at different sites, the same way as the training/validation data sets (contaminated test set). Relative efficiencies are also evaluated to compare the Winsorized results relative to the non-Winsorized prediction results. We compare the efficiencies for different sites of contamination using standard Cauchy noise and at different degrees of Winsorzation.

4.3. Results

4.3.1. Crop Yield Estimation

Table 1 outlines the results of crop yield prediction based on conventional deterministic learning methods with the exception of a shallow concrete dropout model. First, we implement a LASSO [113] regression approach, which imitates the generalized linear regression models that are popularly used in many of the related domain science outlets. The regularization coefficient at 0.0781 is learned adaptively through fivefold cross validation. Based on the LASSO regularization and resampling inference on deep, semi-parametric modeling (not shown here), we can infer a strong negative correlation between agricultural yields and harvest-period precipitation. Next, we use a random forest model with 100 trees and a max depth of 100 per tree. Next, a support vector regression with a radial-basis kernel and $C = 10$ and a margin of error $\epsilon = 0.1$ is learned. To enable comparison with [79], we also fit a support vector model of 8^{th} degree, with $C = 1.0$. A shallow ANN model using three layers, with concrete dropout, is also fitted to the data. The deterministic deep ANN architecture is then used, which results in the best MSE and R^2 values on test data.

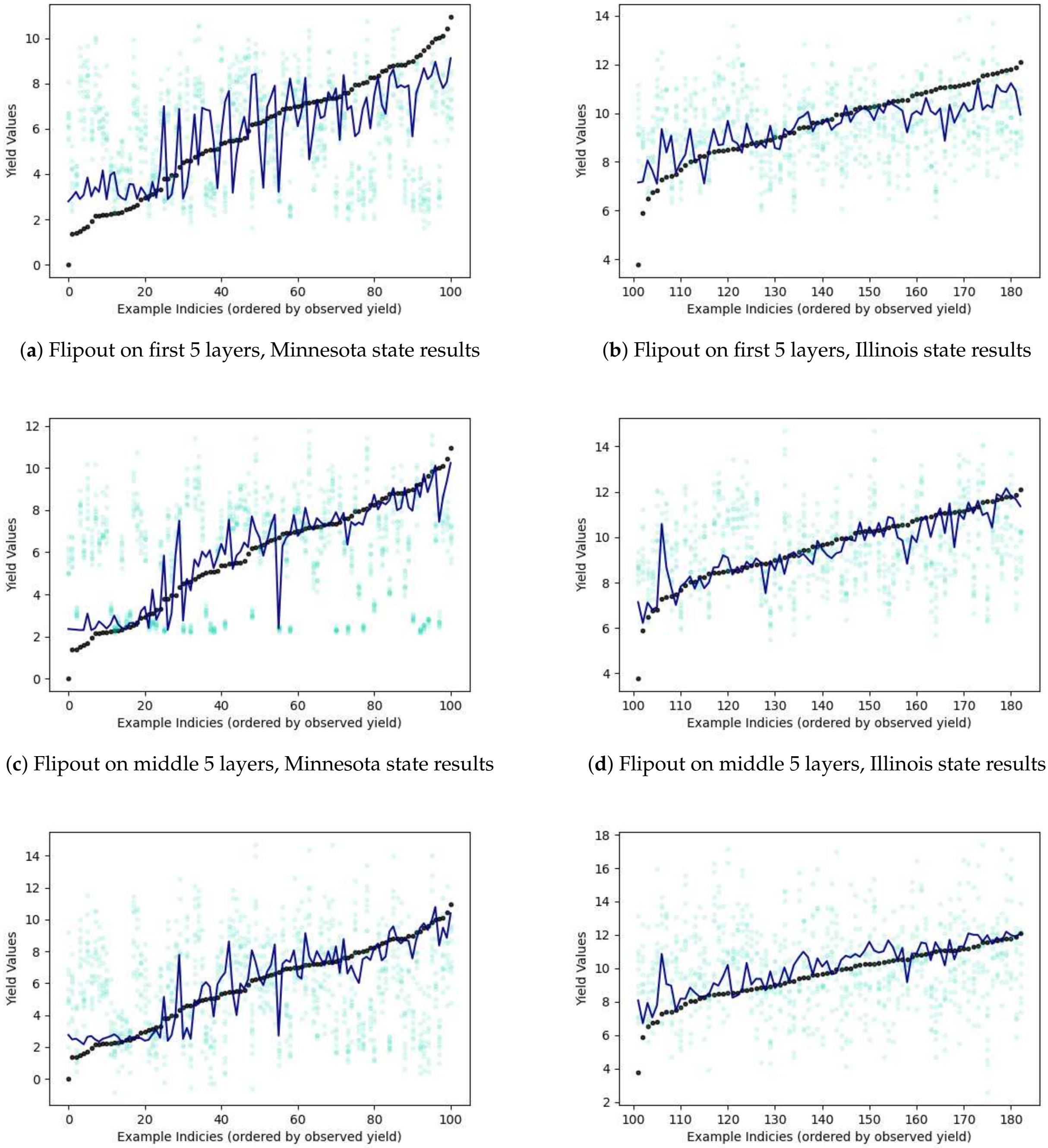
Table 2 reports the probabilistic frameworks that were tested on crop yield estimation problem. The results include variations of flipout based and MDN based networks. Apart from experiments with flipout gradient estimation on all layers, we also try the gradient estimation technique on specific layers in neural network. The flipout implementation on "early 5 layers" focuses on applying the estimation technique only on the first 5 layers of the model. Similarly, "mid 5 layers" focuses on applying flipout on the middle 5 layers while "final 5 layers" focuses on applying the method to the final 5 layers closer to the output. Flipout estimation for all layers not only leads to more number of floating point operations per second but also fails to estimate optimal parameters that may improve predictive performance. By applying flipout on subset of layers, we allow for the network to estimate the output in a more efficient manner and still retain the probabilistic behavior of the model (and the unbiased gradient estimates in the presence of weight perturbations). Figure 1 displays the prediction results for the flipout experiments. The colder counties in Minnesota have lower yield while warmer counties in Illinois have higher yields. For these relatively colder counties and relatively warmer counties, the model is still unable to accurately predict crop yield. This is visible in the lower left-hand-side tail of prediction in Minnesota result sub-plots and the upper right-hand-side tail of predictions in Illinois sub-plots. Similarly, variational GP is unable to achieve the same level of performance as exact GP. In the MDN, we vary the number of mixing components. As we increase the number of components, we add enough complexity to model multiple geographical regions using different mixing densities such that the multiple modalities of the data are sufficiently captured. This helps in improving the performance in terms of the test mean squared error and coefficient of determination. Figure 2 shows us the different predictions results on a geographical map. Flipout and VGP-based predictions are unable to cover the full range of the observed target. In Figure 3 as well, the prediction results can be noted by state. Bayesian neural network methods have lower predictive performance on Minnesota as opposed to Illinois. Epistemic uncertainty is higher for the concrete dropout and exact GP and lower for the mixture density network variants. While our inadequate knowledge about the best dataset and predictive model for supervised learning is displayed in the epistemic uncertainty, more discussion sources of uncertainties can be found in Appendix C.

Table 1. Comparison of different machine learning models on the test data.

Model	Test MSE	R^2
Linear Regression (Lasso)	2.3432	0.7205
Random Forest	2.1113	0.7481
Support Vector Regression (rbf kernel)	2.3	0.7246
Support Vector Regression (polynomial kernel, degree: 8)	4.2943	0.4878
Concrete Dropout, 3-layer ANN	3.0001	0.6379
Neural Network	1.9224	0.7684

Table 2. Probabilistic methods: Crop yield estimation.

Model	Test MSE	R^2	Run Time	Test Median Absolute Error	Number of Parameters
Concrete Dropout	2.33	0.62	19 s	1.09	1,106,952
Variation GP	51.09	−7.38	75 s	6.50	1,108,167
Flipout	10,349	−15,544	464 s	73.63	2,213,872
Flipout (early 5 layers)	2.72	0.47	215 s	1.30	1,481,959
Flipout (mid 5 layers)	2.53	0.60	166 s	0.75	1,309,563
Flipout (final 5 layers)	2.70	0.31	232 s	0.85	1,635,316
MDN (2 components)	2.95	0.66	56 s	1.01	1,108,748
MDN (3 components)	2.24	0.69	52 s	0.73	1,110,107
MDN (4 components)	2.15	0.71	54 s	0.71	1,111,466
Exact GP	2.22	0.69	3.6 s	0.69	2



(**a**) Flipout on first 5 layers, Minnesota state results

(**b**) Flipout on first 5 layers, Illinois state results

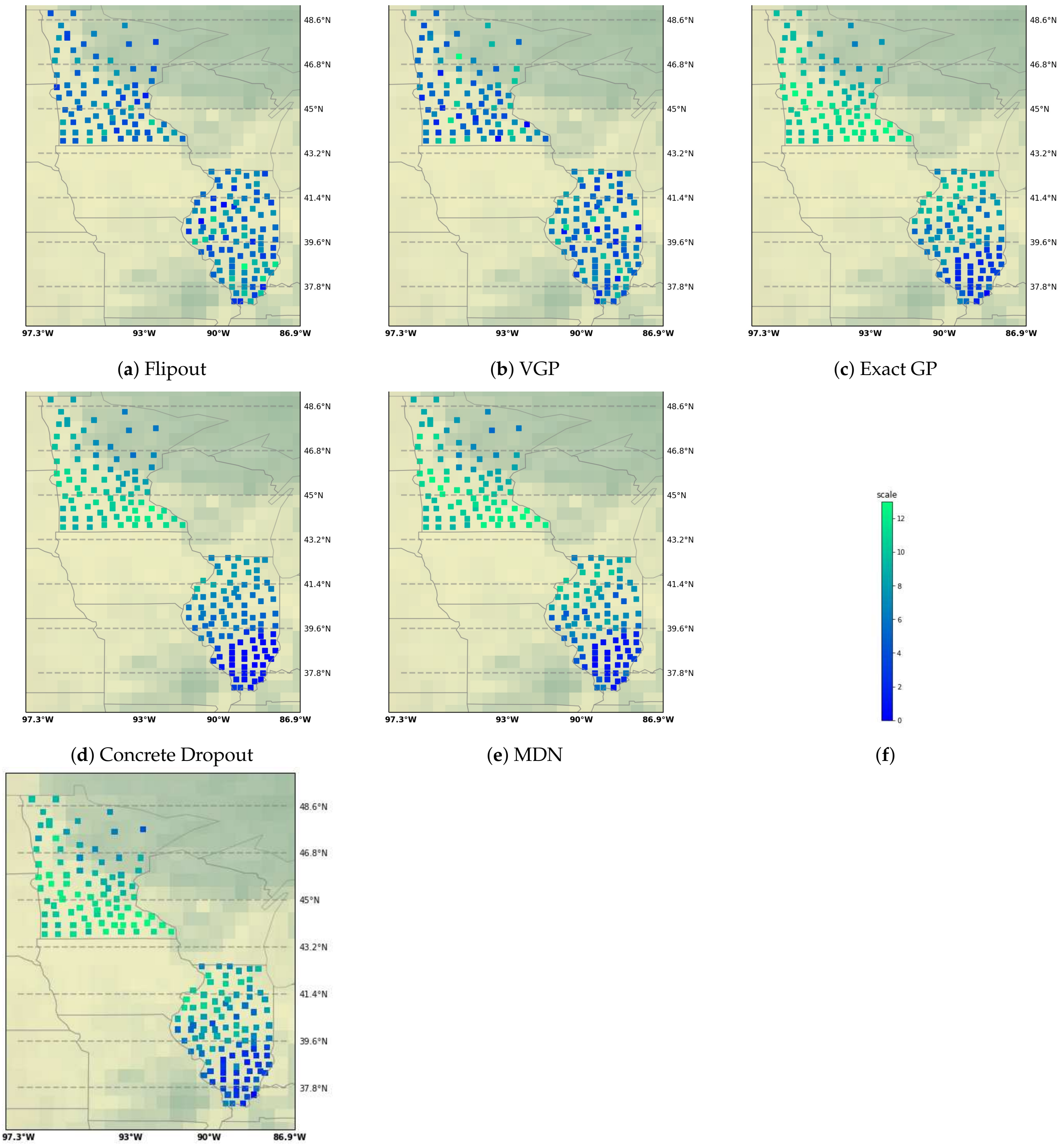
(**c**) Flipout on middle 5 layers, Minnesota state results

(**d**) Flipout on middle 5 layers, Illinois state results

(**e**) Flipout on last 5 layers, Minnesota state results

(**f**) Flipout on last 5 layers, Illinois state results

Figure 1. Crop yield predictions. X-axis shows arbitrary county indices which are sorted by the observed yield in ascending order. Y-axis represents the yield value. Black points are the observed yield. Navy blue line is the mean prediction and light blue points are the predictions in several individual runs.

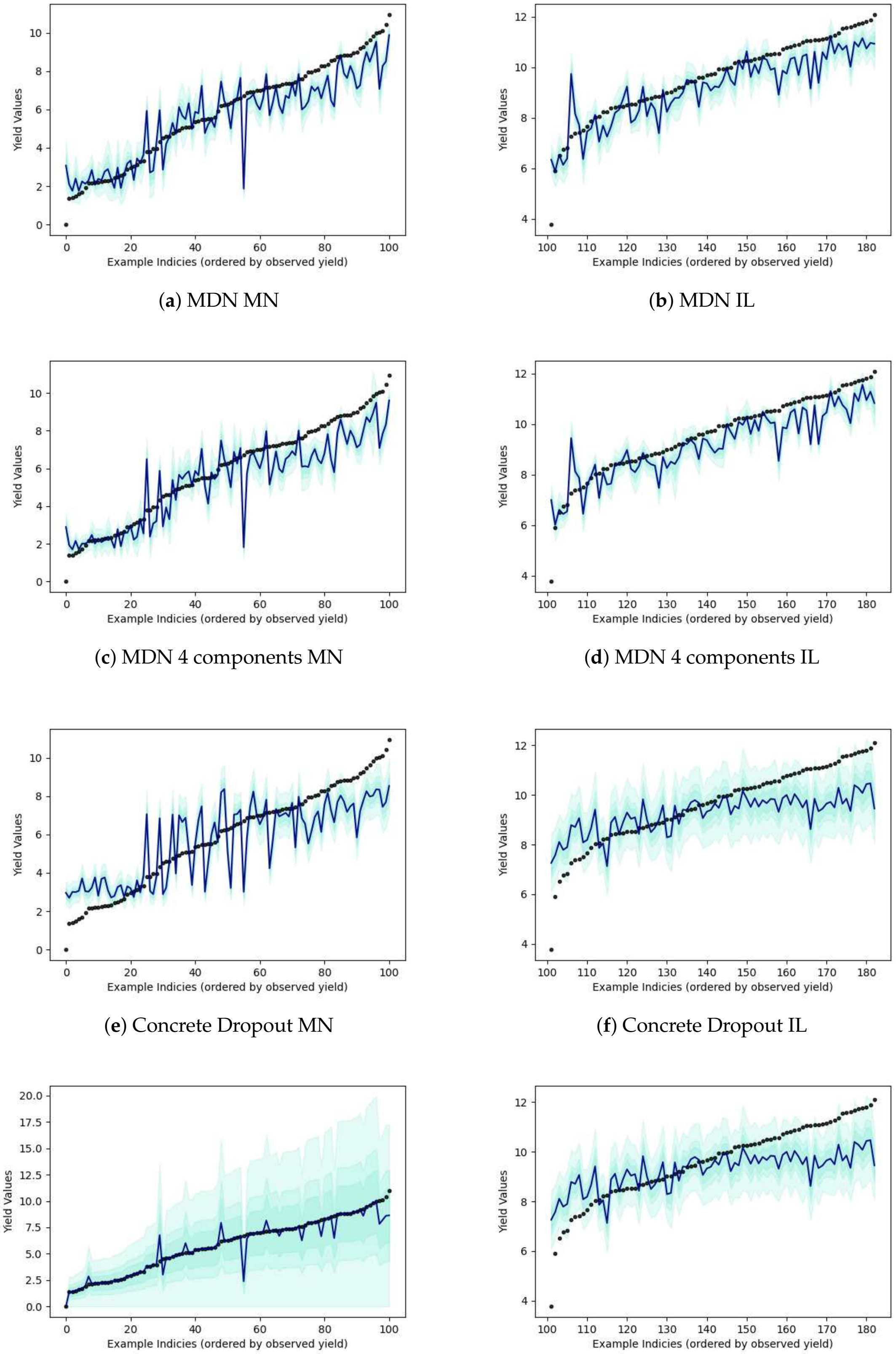


(**a**) Flipout (**b**) VGP (**c**) Exact GP

(**d**) Concrete Dropout (**e**) MDN (**f**)

(**g**) Observed Yield

Figure 2. Crop yield predictions for Minnesota and Illinois. Sub-plot (**f**) shows us the legend. Darker blue shade represents lower yield predictions and lighter shade represents higher yield predictions. Methods for better predictive performance (concrete dropout, mixture density network, and exact gp) are able to correctly predict the whole range of observed yield. Flipout and VGP-based Bayesian neural networks are unable to predict well especially in Minnesota counties.



(**a**) MDN MN

(**b**) MDN IL

(**c**) MDN 4 components MN

(**d**) MDN 4 components IL

(**e**) Concrete Dropout MN

(**f**) Concrete Dropout IL

(**g**) Exact GP MN

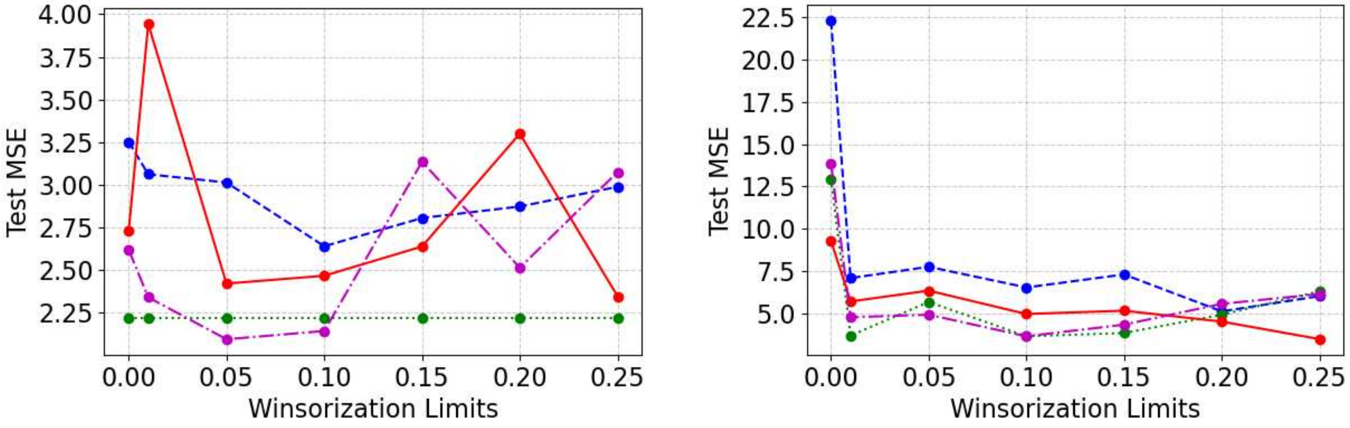
(**h**) Exact GP IL

Figure 3. Crop yield predictions. X-axis shows arbitrary county indices which are sorted by the observed yield in ascending order. Y-axis represents the yield value. Black points are the observed yield. Navy blue line is the mean prediction and epistemic uncertainty estimates is shown in turquoise.

Winsorization Results

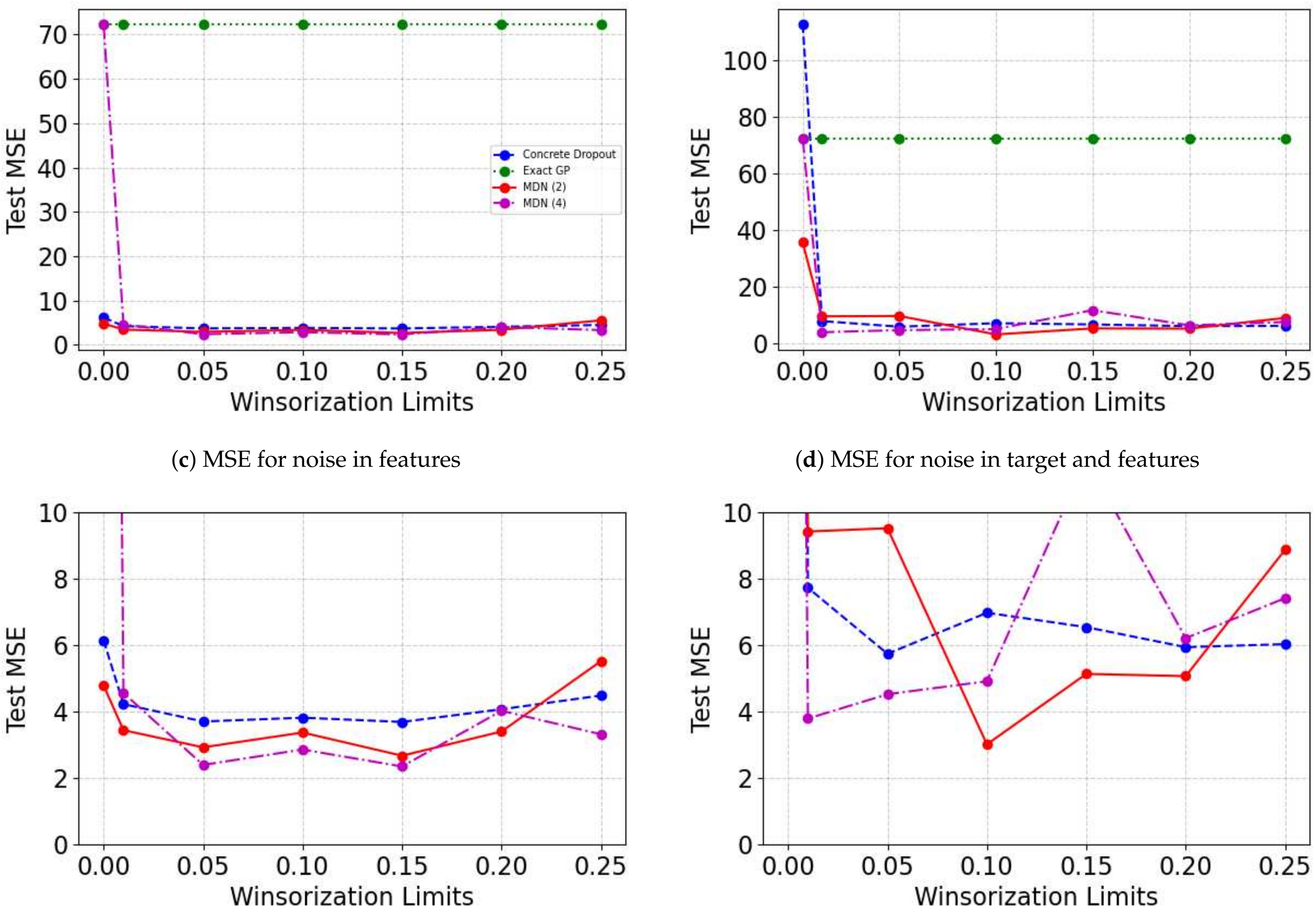
Experiments to test the effectiveness of Winsorization in removing Cauchy noise from the data are performed. In four different variations, we add standard Cauchy noise to *(a)* target, *(b)* features, *(c)* target and features, *(d)* neither target nor features in the training and the validation set. We experiment with two scenarios—when the test set contains no Cauchy noise in none of the four variations listed before and when the test set contains Cauchy noise according to the first three variations listed before. We compute the mean squared error, median absolute error, mean absolute error, and the coefficient of determination as our evaluation metrics. Table 3 shows the values of the metric before Winsorization as MSE, Median AE, MAE, and R^2, respectively. The metric values after Winsorization are shown as MSE_W, Median AE_W, MAE_W, and R^2_W. For each noise site variation, an optimal Winsorization limit that minimizes the mean MSE_W is chosen to be displayed. The untouched test set is more similar to the central data distribution in the training set. With the exception of exact GP, the model performance on the test set unconditionally improves when the training and validation sets are Winsorized. In the contaminated test set, there is no apparent trend in model performance as Winsorization is applied. This is consistent with the fact that the model is training to learn the noise as well, increasing the coefficient of determination and reducing MSE values in the contaminated case as compared to the untouched test set case.

Figure 4 displays the Test MSE_W values on the untouched test set after Winsorizing the training and validation data set. The figure showcases results when perturbation is added to separate noise sites. Figure 4e,f show the results from Figure 4c,d only for the test MSE range 0 to 10. It is evident that, in general, the presence of noise using some degree of Winsorization in the training and validation set improves the model performance as opposed to when Winsorization is not used. The benefits of Winsorization are not clear when there is no artificial perturbation in the data. The marginal improvement in Figure 4a could be the result of the removal of naturally occurring outliers. The benefits of Winsorization are more apparent in Figure 4b–d. In addition, while exact GP was the best performing model before the addition of noise, the performance degrades after the addition of noise, especially when added to the feature set. Mixture Density networks perform relatively better for all the noisy cases. Probabilistic neural network models are able to overcome the effects of perturbation in features at relatively lower Winsorization levels but require a higher degree of winsorization to reach the optimal Winsorization limits for achieving the best test MSE.



(**a**) MSE for noise-free case (**b**) MSE for noise in target

Figure 4. *Cont.*



(**c**) MSE for noise in features

(**d**) MSE for noise in target and features

(**e**) MSE for noise in features, y axis between 0 and 10

(**f**) MSE for noise in target and features, y axis between 0 and 10

Figure 4. Winsorization results from 0 to 25 percentile limits on crop yield dataset. Mean Squared Error is shown on the y-axis and the Winsorization limits are shown on the x-axis. Different lines represent different methods: Concrete dropout is shown as blue dashed line, exact GP is shown as green dotted line, mixture density network with 2 components is shown in red solid line, and mixture density network with 4 components is shown in magenta dashed-dotted line. As Winsorization limit increases on the training set, the model performance in terms of mean squared error for the untouched test set is shown in the sub-plots.

Table 3. Winsorization results on test set for crop yield dataset.

Noise Site	Optimal Limit	Model	MSE	MSE_W	R^2	R^2_W	Median AE	Median AE_W	MAE	MAE_W
None	0.1	Concrete Dropout	3.25	2.63	0.55	0.64	1.55	1.16	1.51	1.33
None	0.1	Exact GP	2.21	2.21	0.69	0.69	0.69	0.69	1.13	1.13
None	0.1	MDN	2.72	2.46	0.63	0.66	1.05	0.72	1.28	1.15
None	0.1	MDN (4)	2.61	2.14	0.64	0.70	0.91	0.50	1.19	1.01
Untouched Test Set										
Target	0.25	Concrete Dropout	22.27	6.02	−2.02	0.18	4.37	1.55	4.16	1.97
Target	0.25	Exact GP	12.94	6.31	−0.75	0.14	1.74	1.26	2.45	1.89
Target	0.25	MDN	9.29	3.49	−0.26	0.52	1.49	1.40	2.25	1.56
Target	0.25	MDN (4)	13.84	6.15	−0.87	0.17	2.44	1.12	2.91	1.79
Features	0.15	Concrete Dropout	6.15	3.69	0.16	0.50	1.98	1.26	2.07	1.52
Features	0.15	Exact GP	72.27	72.27	−8.79	−8.79	8.6	8.6	8.05	8.05
Features	0.15	MDN	4.78	2.67	0.35	0.63	1.87	0.98	1.86	1.32
Features	0.15	MDN (4)	72.27	2.34	−8.79	0.68	8.60	0.96	8.05	1.21
Target and Features	0.25	Concrete Dropout	112.62	6.03	−14.26	0.18	9.93	1.17	10.33	1.82
Target and Features	0.25	Exact GP	72.27	72.27	−8.79	−8.79	8.60	8.60	8.05	8.05
Target and Features	0.25	MDN	35.55	8.88	−3.81	−0.20	5.44	2.48	5.52	2.46
Target and Features	0.25	MDN (4)	72.27	7.42	−8.79	−0.01	8.60	1.31	8.05	1.98
Contaminated Test Set										
Target	0.25	Concrete Dropout	2.85	3.04	0.61	0.58	1.37	1.47	1.40	1.49
Target	0.25	Exact GP	2.21	2.91	0.69	0.60	0.69	1.49	1.13	1.46
Target	0.25	MDN	3.05	2.73	0.58	0.62	1.49	1.69	1.44	1.44
Target	0.25	MDN (4)	2.37	2.93	0.67	0.60	0.81	0.95	1.14	1.32
Features	0.05	Concrete Dropout	2.51	2.27	0.65	0.69	1.19	1.08	1.33	1.17
Features	0.05	Exact GP	2.21	2.39	0.69	0.68	1.13	1.23	0.62	0.61
Features	0.05	MDN	2.39	2.71	0.67	0.63	0.70	1.02	1.09	1.28
Features	0.05	MDN (4)	2.47	2.55	0.66	0.65	1.02	0.84	1.18	1.26
Target and Features	0.25	Concrete Dropout	3.08	3.38	0.58	0.54	1.35	1.64	1.52	1.58
Target and Features	0.25	Exact GP	2.21	5.89	0.69	0.20	0.69	1.98	1.13	2.12
Target and Features	0.25	MDN	2.60	2.09	0.64	0.71	0.87	1.23	1.20	1.20
Target and Features	0.25	MDN (4)	4.12	2.69	0.44	0.63	1.44	1.07	1.63	1.36

4.3.2. California Housing Data

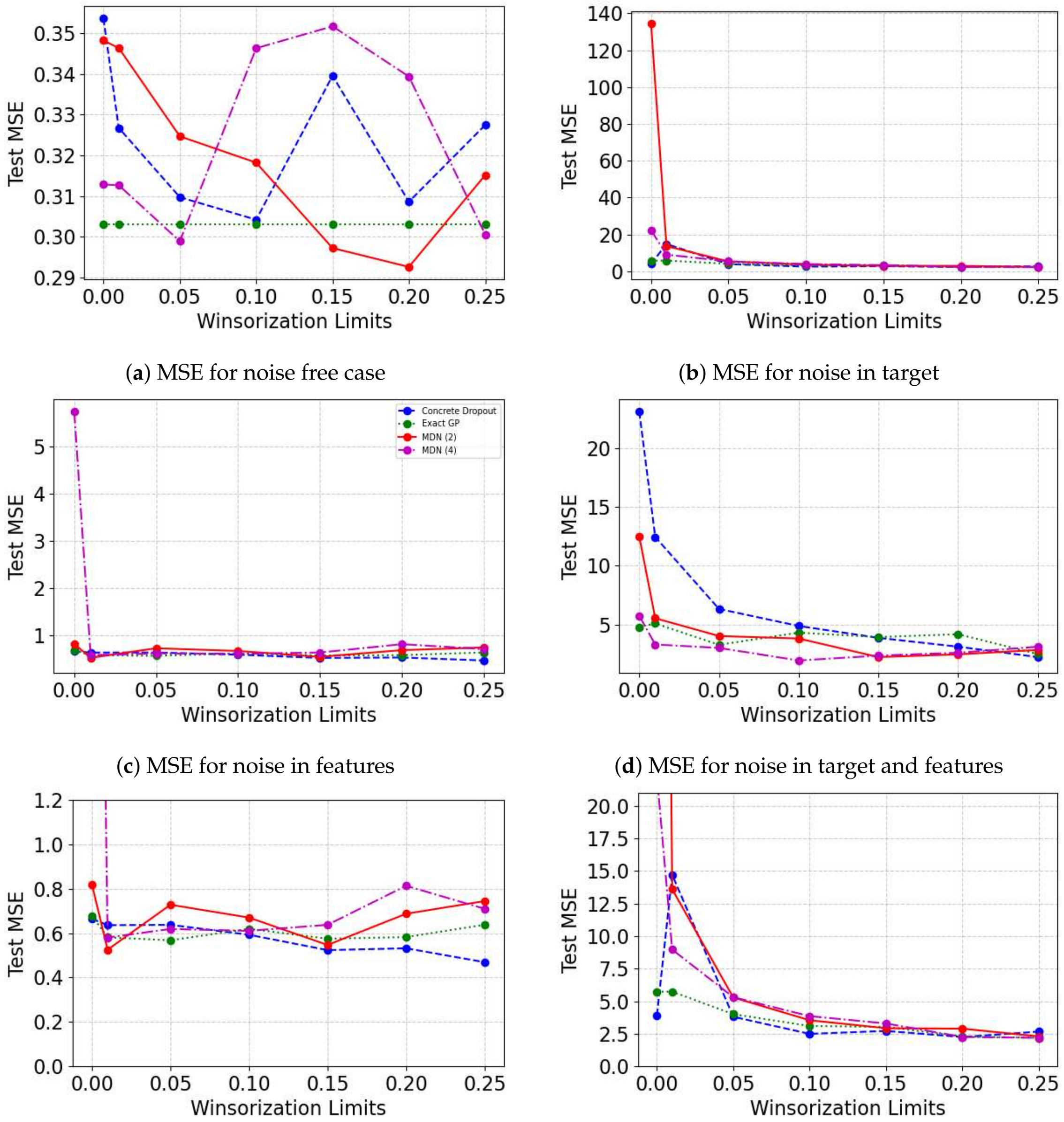
For the California house price dataset, Table 4 shows the results for different probabilistic models. Similar to the previous case study, the variational GP and flipout gradient estimations in the presence of weight perturbation are unable to perform well. The mixture density network with two mixing components is able to perform well, but exact GP provides the best performance in the absence of any noise.

Table 4. Probabilistic methods: California house prices.

Model	Test MSE	R^2	Run Time	Test Median Absolute Error	Number of Parameters
Concrete Dropout	0.44	0.63	28 s	0.26	1,050,168
Variational GP	1.74	−2.39	36 s	0.71	1,050,143
Exact GP	0.28	0.69	73.34 s	0.21	2
Flipout	0.64	−0.54	392 s	0.45	2,100,304
MDN (2 components)	0.31	0.66	52 s	0.23	1,051,964

Figure 5 shows the Test MSE_W on the untouched test data as the degree of Winsorization on the training and validation sets are varied. Similar to the results obtained in the previous case study, there is no apparent certain improvement with varying Winsorization limits in test MSE_W when there is no noise in the data. When Winsorization limit is increased from 0 to 1 percentile and 5 percentile, we notice improvement in the performance for the probabilistic neural network models. The mixture density network is able to perform marginally better than exact GP when the Winsorization limit is 5 percentile. Beyond the 5 percentile mark, we notice that mixture density network with 2 mixing components are able to perform better than exact GP when the Winsorization limit is 15 percentile and 20 percentile. As noise is added to the features, the model performance for concrete dropout and mixture density network (with four components) improves from the pre-Winsorization case. It is visible in Figure 5b,d,f, that as the degree of Winsorization increases in the training/validation sets, the model performance of most probabilistic neural network models increases. In Figure 5c,e as the Winsorization limit increases from 0 to 1 percentile, we see the most model performance improvement. While the model performance continues to improve with the increasing Winsorization limit for the concrete dropout Bayesian neural network, we are unable to notice the same for other models. We notice that noise adversely affects exact GP's performance.

In Table 5, the model evaluation metrics are provided on the untouched and contaminated test sets before and after Winsorization is applied. The optimal Winsorization results are shown. We notice similar trends as the previous cases study results. The optimal Winsorization limit is usually well below the 25 percentile threshold for the case when we introduce noise in the features before model training. For the untouched test set case, the model performance improves for all models including exact GP.



(**a**) MSE for noise free case

(**b**) MSE for noise in target

(**c**) MSE for noise in features

(**d**) MSE for noise in target and features

(**e**) MSE for noise in features; y-axis between 0 and 1.2

(**f**) MSE for noise in target; y axis between 0 and 21

Figure 5. Winsorization results from 0 to 25 percentile limits on California housing dataset. Mean Squared Error is shown on the y-axis and the Winsorization limits are shown on the x-axis. Different lines represent different methods: Concrete dropout is shown as blue dashed line, exact GP is shown as green dotted line, mixture density network with 2 components is shown in red solid line, and mixture density network with 4 components is shown in magenta dashed-dotted line. As Winsorization limit increases in the training dataset, the model performance in terms of mean squared error for the untouched test set is shown in the picture.

Table 5. Winsorization results on test set for the California Housing dataset.

Noise Site	Optimal Limit	Model	MSE	MSE$_W$	R^2	R^2_W	Median AE	Median AE$_W$	MAE	MAE$_W$
None	0.05	Concrete Dropout	0.35	0.30	0.62	0.67	0.25	0.24	0.38	0.36
None	0.05	Exact GP	0.30	0.30	0.67	0.67	0.20	0.20	0.34	0.34
None	0.05	MDN	0.35	0.32	0.62	0.65	0.23	0.22	0.36	0.35
None	0.05	MDN (4)	0.31	0.29	0.66	0.68	0.18	0.20	0.33	0.34
				Untouched Test Set						
Target	0.25	Concrete Dropout	3.91	2.65	−3.18	−1.84	0.80	1.58	1.26	1.51
Target	0.25	Exact GP	5.72	2.18	−5.12	−1.33	1.97	1.32	2.18	1.34
Target	0.25	MDN	134.64	2.29	−142.85	−1.44	2.45	1.15	6.89	1.27
Target	0.25	MDN (4)	22.29	2.18	−22.82	−1.33	1.45	1.33	2.56	1.32
Features	0.1	Concrete Dropout	0.66	0.59	0.28	0.36	0.56	0.40	0.64	0.57
Features	0.1	Exact GP	0.67	0.62	0.27	0.33	0.56	0.42	0.62	0.55
Features	0.1	MDN	0.82	0.67	0.12	0.28	0.51	0.45	0.67	0.58
Features	0.1	MDN (4)	5.72	0.61	−5.12	0.34	1.97	0.36	2.18	0.54
Target and Features	0.25	Concrete Dropout	23.10	2.21	−23.68	−1.37	4.55	1.43	4.48	1.33
Target and Features	0.25	Exact GP	4.77	2.47	−4.10	−1.64	2.07	1.53	2.01	1.42
Target and Features	0.25	MDN	12.47	2.82	−12.32	−2.01	2.32	1.31	2.84	1.37
Target and Features	0.25	MDN (4)	5.72	3.09	−5.12	−2.31	1.97	1.18	2.18	1.36
				Contaminated Test Set						
Target	0.25	Concrete Dropout	0.31	0.60	0.65	0.35	0.23	0.40	0.36	0.55
Target	0.25	Exact GP	0.30	0.57	0.67	0.38	0.20	0.37	0.34	0.51
Target	0.25	MDN	0.32	0.58	0.65	0.37	0.19	0.36	0.33	0.53
Target	0.25	MDN (4)	0.32	0.56	0.65	0.39	0.17	0.35	0.32	0.52
Features	0.01	Concrete Dropout	0.28	0.33	0.69	0.64	0.23	0.27	0.34	0.37
Features	0.01	Exact GP	0.30	0.30	0.67	0.67	0.20	0.20	0.34	0.34
Features	0.01	MDN	0.32	0.27	0.65	0.71	0.20	0.21	0.35	0.33
Features	0.01	MDN (4)	0.33	0.30	0.64	0.67	0.21	0.22	0.35	0.34
Target and Features	0.25	Concrete Dropout	0.30	0.38	0.67	0.59	0.22	0.33	0.34	0.44
Target and Features	0.25	Exact GP	0.30	0.60	0.67	0.35	0.20	0.39	0.34	0.54
Target and Features	0.25	MDN	0.31	1.41	0.66	−0.51	0.21	0.60	0.34	0.80
Target and Features	0.25	MDN (4)	0.35	1.37	0.62	−0.41	0.19	0.63	0.34	0.81

4.3.3. Mauna Loa CO_2 Data

The CO_2 concentration in the atmosphere increases as a function of time, measured in months and years. A naive fitting on the CO_2 concentration as response will lead to sub-optimal training. The model performance results and the Winsorization results on unadjusted responses are given in Appendix B. We therefore adjust for the long-term linear trend and monthly seasonal trend in CO_2 concentration. Therefore the response, CO_2 concentration, can be decomposed as follows:

$$Y_{i,k} = \mu_i + \mu_{i,k} + f(X'_{i,k}) \tag{21}$$

where μ_i is the linear long term trend for the i^{th} observation. This component can be fit as a linear regression model on X_i. $\mu_{i,k}$ is the seasonal mean for the i^{th} observation and $f(X'_i)$ is the non-parametric fitting on the residuals using a probabilistic neural network. Here, $X'_{i,k} = Y_{i,k} - \mu_i - \mu_{i,k}$. In addition, the linear long-term trend can be fit using the original features as follows:

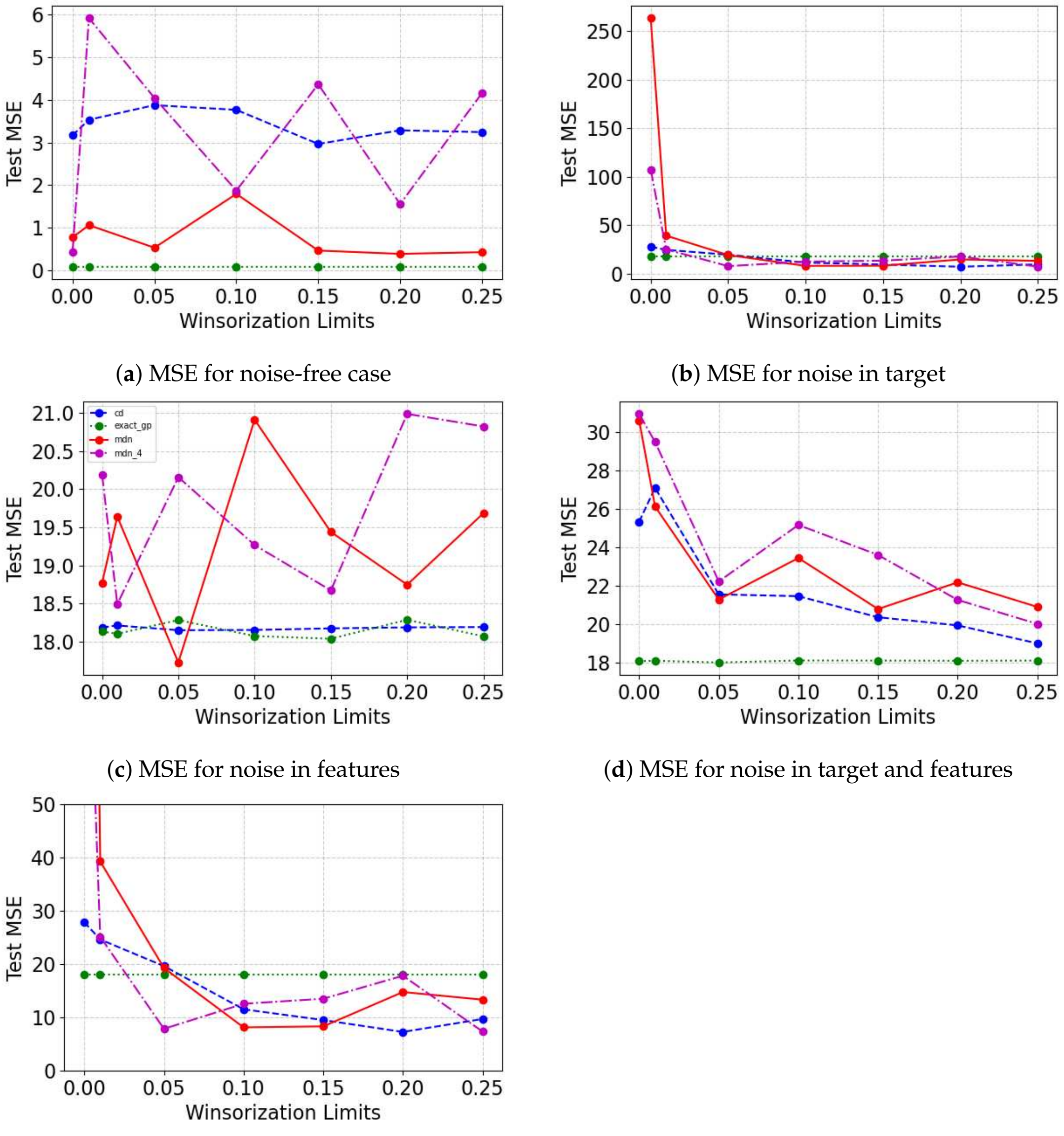
$$\mu_i = X_i\beta \tag{22}$$

where the $\hat{\beta}$ estimator can be obtained by ordinary least squared for this dataset. Using this semi-parameteric model allows for the non-parameteric component of the model to focus on fitting to the more complex relationships without having to accommodate for the linear long term trend and the monthly seasonal trend. Table 6 shows the performance results for various probabilistic methods on the Mauna CO_2 concentration dataset. Flipout is unable to converge to a representative posterior predictive distribution. Variational GP performs slightly better. Concrete dropout, mixture density networks, and Gaussian processes perform the best where MDN has slightly faster training time.

Table 6. Probabilistic methods: Mauna CO_2 concentration.

Model	**Test MSE**	R^2	**Run Time**	**Test Median Absolute Error**
Concrete Dropout	3.24	0.82	25 s	0.99
VGP	18.44	−0.04	3 s	31.45
Exact GP	0.08	0.99	34 s	0.17
MDN	0.42	0.97	23 s	0.38
Flipout	19.68	−0.85	4 s	211.83

Figure 6 show the Test MSE_W on the untouched test set as the Winsorization limits on the training and validation set are increased from 0 to 25 percentile. In the noise-free case, as the degree of Winsorization is increased, test set MSE remains the same for the exact GP and changes marginally for MDN with 2 components and concrete dropout. For the case when noise is introduced in features, test MSE increases with the increasing degree of Winsorization. This may be due the fact that the feature set dimension is one and adding perturbations greater than feature values to all training examples adversely impairs the models from learning. When the noise is added at least to the target, there is a decline in untouched test set MSE. In the case, where there is noise only in target, the MDN performance improves drastically. The exact GP remains unchanged while concrete dropout improves marginally. For the case where there is noise in both target and feature, the exact GP remain the same throughout while the model performance for other models only improves to a certain point, even at a higher degree of Winsorization.



(**a**) MSE for noise-free case

(**b**) MSE for noise in target

(**c**) MSE for noise in features

(**d**) MSE for noise in target and features

(**e**) MSE for noise in target; y axis 0 to 50

Figure 6. Winsorization results from 0 to 25 percentile limits on Mauna dataset. Mean Squared Error is shown on the y-axis and the Winsorization limits are shown on the x-axis. Different lines represent different methods: Concrete dropout is shown as blue dashed line, exact GP is shown as green dotted line, mixture density network with 2 components is shown in red solid line, and mixture density network with 4 components is shown in magenta dashed-dotted line. As Winsorization limit increases in the training set, the model performance in terms of mean squared error in the untouched test set is shown in the sub-plots.

Table 7 displays the change in model performance. Similar to previous data sets, the model performance improves with Winsorization for the untouched test set while it does not for the contaminated test set. For the untouched test set case, while there is a Winsorization limit for which most methods see an improvement in performance, the model performance improves only slightly for the concrete dropout case.The optimal limit remains high for all noise cases on the untouched test set.

Table 7. Winsorization results on test set for the Mauna CO_2 dataset.

Noise Site	Optimal Limit	Model	MSE	MSE_W	R^2	R^2_W	Median AE	Median AE_W	MAE	MAE_W
None	0.2	Concrete Dropout	3.16	3.28	0.82	0.81	1.03	1.46	1.35	1.52
None	0.2	Exact GP	0.08	0.08	0.99	0.99	0.17	0.17	0.22	0.22
None	0.2	MDN	0.77	0.38	0.95	0.98	0.50	0.31	0.67	0.46
None	0.2	MDN (4)	0.42	1.55	0.97	0.91	0.33	0.57	0.47	0.88
Untouched Test Set										
Target	0.25	Concrete Dropout	27.89	9.71	−0.54	0.46	5.24	2.84	4.84	2.80
Target	0.25	Exact GP	18.08	18.08	−0.001	−0.001	3.15	3.15	3.58	3.58
Target	0.25	MDN	263.51	13.34	−13.58	0.26	3.37	3.12	8.24	3.14
Target	0.25	MDN (4)	107.36	7.35	−4.94	0.59	5.59	2.49	6.92	2.43
Features	0.25	Concrete Dropout	18.18	18.19	−0.006	−0.007	2.97	2.95	3.53	3.53
Features	0.25	Exact GP	18.06	18.06	−0.003	−0.0003	3.15	3.15	3.57	3.57
Features	0.25	MDN	18.76	19.68	−0.03	−0.08	3.20	2.98	3.60	3.65
Features	0.25	MDN (4)	20.19	20.82	−0.11	−0.15	2.88	3.09	3.64	3.75
Target & Features	0.25	Concrete Dropout	25.33	18.99	−0.40	−0.05	5.33	4.17	4.61	3.92
Target & Features	0.25	Exact GP	18.06	20.89	−3e−3	−0.002	3.15	3.15	3.57	3.58
Target & Features	0.25	MDN	30.60	20.89	−0.69	−0.15	4.66	4.13	4.72	4.12
Target & Features	0.25	MDN (4)	30.94	20.01	−0.71	−0.11	4.72	3.98	4.75	3.93
Contaminated Test Set										
Target	0.05	Concrete Dropout	3.13	2.95	0.82	0.83	0.87	0.98	1.27	1.27
Target	0.05	Exact GP	0.08	0.15	0.99	0.99	0.17	0.17	0.22	0.26
Target	0.05	MDN	0.70	0.82	0.96	0.95	0.48	0.49	0.64	0.68
Target	0.05	MDN (4)	1.77	0.49	0.90	0.97	0.42	0.35	0.84	0.51
Features	0.05	Concrete Dropout	3.46	3.21	0.80	0.82	0.89	1.11	1.34	1.37
Features	0.05	Exact GP	0.08	1.65	0.99	0.90	0.17	0.19	0.22	0.59
Features	0.05	MDN	1.03	1.14	0.94	0.93	0.45	0.49	0.68	0.74
Features	0.05	MDN (4)	0.80	4.21	0.95	0.76	0.47	0.78	0.63	1.33
Target and Features	0.01	Concrete Dropout	3.48	3.51	0.80	0.80	1.18	0.93	1.42	1.36
Target and Features	0.01	Exact GP	0.08	0.13	0.99	0.99	0.17	0.17	0.22	0.24
Target and Features	0.01	MDN	0.44	0.43	0.97	0.98	0.41	0.34	0.50	0.43
Target and Features	0.01	MDN (4)	3.46	6.33	0.80	0.64	0.95	1.56	1.29	1.87

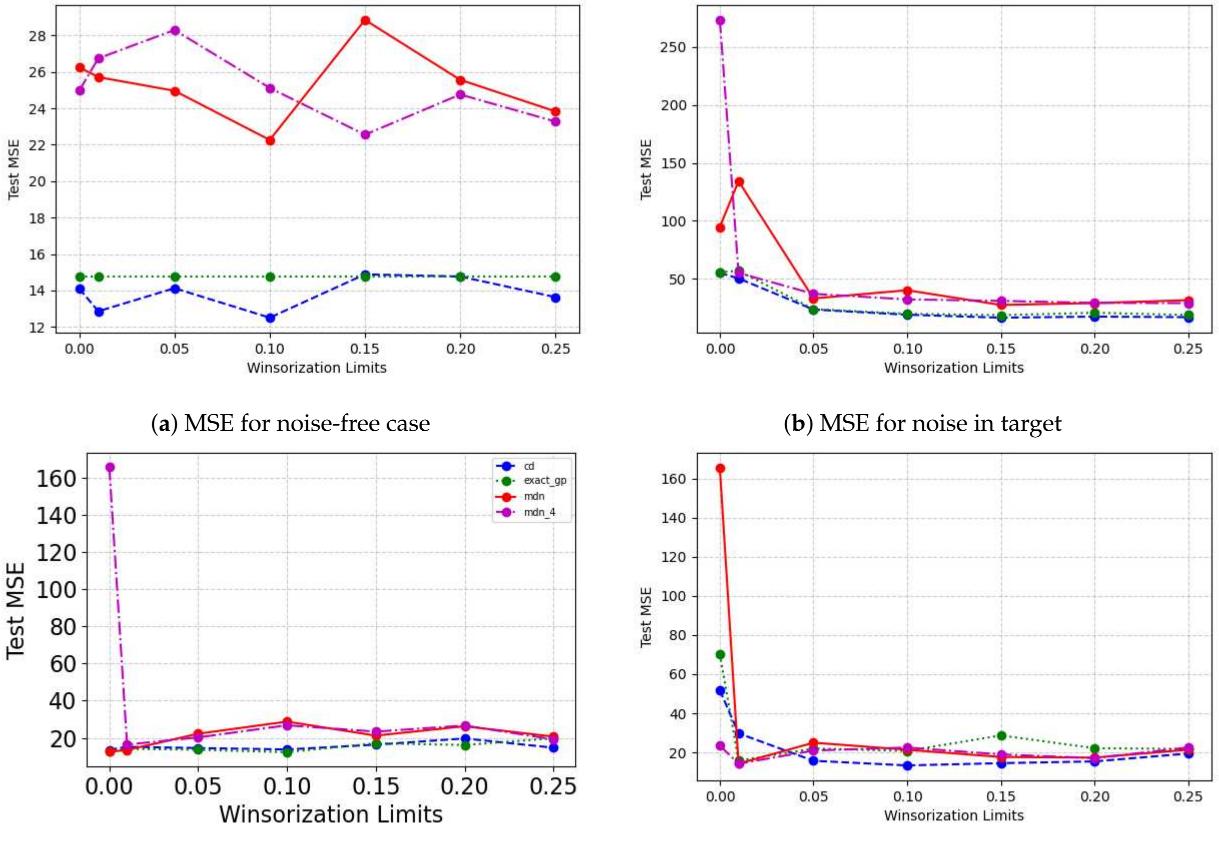
4.4. Forest Fires Data

Table 8 compares the model performance without any noise in the dataset and without Winsorization in the training and validation set. The forest fires dataset is a challenging dataset for all methods. None of the methods are able to achieve reasonable model performance on this dataset, suggesting that there is a need for exploring more methodologies and architectures in further work. Among the methods that were tested, similar trends as before were noticed. Flipout is the most expensive and takes more time. Exact GP and MDN achieve the best model performance and run time. VGP performs slightly worse and takes more time to train than MDN and Concrete dropout.

Table 8. Probabilistic methods: Forest fires.

Model	Test MSE	R^2	Run Time	Test Median Absolute Error
Concrete Dropout	12.93	−0.11	100 s	3.25
Flipout	2015.67	−192.80	1162 s	3.66
VGP	21.93	−0.73	372 s	3.71
Exact GP	14.66	−0.19	4 s	3.46
MDN	21.44	−0.74	95 s	3.52

Figure 7 shows the change the model performance on the forest fires dataset when the training and validation datasets are Winsorized. In the noise-free case, there is no affect on the exact GP performance as the degree of Winsorization is increased. MDN performance remains worse than Concrete dropout and exact GP despite the changing Winsorization limits. For other cases, where noise is introduced, we notice that some degree of Winsorization helps improve model performance as opposed to the no Winsorization case, especially when there is noise in the response variable. In the cases where noise is introduced, Concrete dropout and exact GP are able to achieve the lowest Test MSE_W.



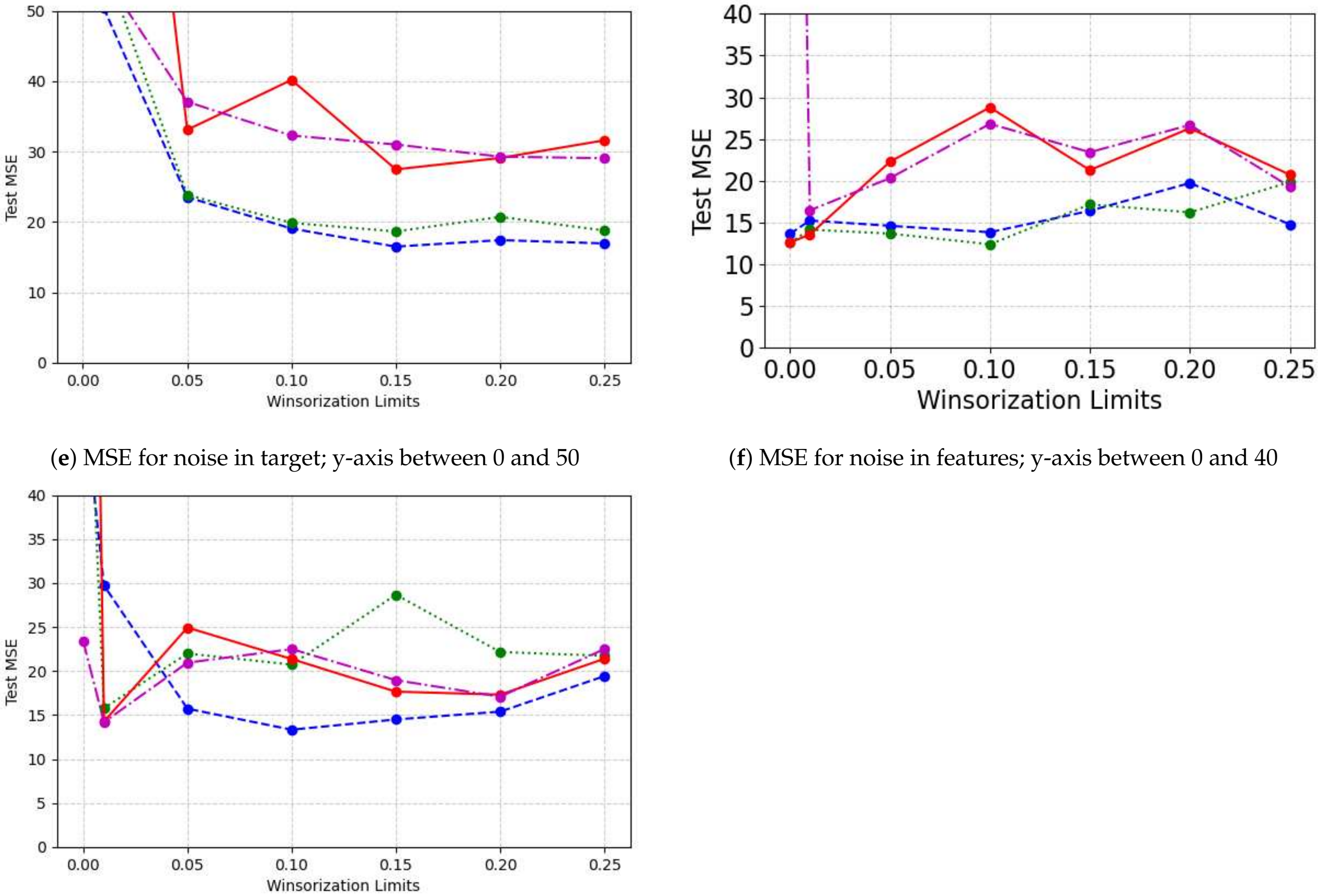
(**a**) MSE for noise-free case

(**b**) MSE for noise in target

(**c**) MSE for noise in features

(**d**) MSE for noise in target and features

Figure 7. *Cont.*



(**e**) MSE for noise in target; y-axis between 0 and 50

(**f**) MSE for noise in features; y-axis between 0 and 40

(**g**) MSE for noise in target and features; y-axis between 0 and 40

Figure 7. Winsorization results from 0 to 25 percentile limits on forest fires dataset. Mean Squared Error is shown on the y-axis and the Winsorization limits are shown on the x-axis. Different lines represent different methods: Concrete dropout is shown as blue dashed line, exact GP is shown as green dotted line, mixture density network with 2 components is shown in red solid line, and mixture density network with 4 components is shown in magenta dashed-dotted line. As Winsorization limit increases in the training set, the model performance in terms of Mean Squared Error in the untouched test set is show in the sub-plots.

Table 9 showcases similar trends as the previous datasets. The model performance improves with Winsorization on the untouched test set and there is no definite improvement for the contaminated test set. The optimal Winsorization limit for noise in the features case is low, similar to all datasets.

Table 9. Winsorization results on test set for forest fires dataset.

Noise Site	Optimal Limit	Model	MSE	MSE$_W$	R^2	R^2_W	Median AE	Median AE$_W$	MAE	MAE$_W$
None	0.01	Concrete Dropout	14.10	12.85	−0.15	−0.04	3.39	3.25	3.40	3.22
None	0.01	Exact GP	14.75	14.75	−0.2	−0.20	3.51	3.51	3.39	3.39
None	0.01	MDN	26.24	25.71	−1.14	−1.09	3.76	3.57	4.28	4.03
None	0.01	MDN (4)	24.97	26.74	−1.03	−1.18	3.91	3.83	4.05	4.17
				Untouched Test Set						
Target	0.25	Concrete Dropout	55.19	16.95	−3.51	−0.38	4.86	2.96	5.98	3.51
Target	0.25	Exact GP	55.73	18.78	−3.55	−0.53	3.84	3.04	5.34	3.66
Target	0.25	MDN	94.78	31.60	−6.73	−1.57	5.27	3.99	6.99	4.70
Target	0.25	MDN (4)	273.36	29.05	−21.31	−1.37	5.16	4.21	8.55	4.62
Features	0.15	Concrete Dropout	13.61	16.39	−0.11	−0.33	2.97	2.68	3.41	3.31
Features	0.15	Exact GP	12.56	17.15	−0.02	−0.40	3.16	3.51	3.29	3.62
Features	0.15	MDN	12.59	21.26	−0.02	−0.73	3.23	3.39	3.08	3.75
Features	0.15	MDN (4)	165.53	23.41	−12.51	−0.91	11.18	3.56	12.38	3.91
Target and Features	0.25	Concrete Dropout	51.78	19.42	−3.22	−0.58	6.57	3.40	6.04	3.77
Target and Features	0.25	Exact GP	70.21	21.73	−4.73	−0.77	6.55	3.95	7.06	8.06
Target and Features	0.25	MDN	165.53	21.39	−12.51	−0.74	11.18	3.42	12.38	3.71
Target and Features	0.25	MDN (4)	23.44	22.49	−0.91	−0.65	3.89	3.55	4.01	3.81
				Contaminated Test Set						
Target	0.25	Concrete Dropout	13.68	12.95	−0.11	−0.05	3.33	3.04	3.36	3.23
Target	0.25	Exact GP	14.75	14.91	−0.20	−0.21	3.50	3.41	3.39	3.45
Target	0.25	MDN	23.89	14.91	−0.95	−0.84	3.72	3.41	4.06	3.45
Target	0.25	MDN (4)	22.49	20.38	−0.83	−0.66	3.74	3.11	3.93	3.68
Features	0.05	Concrete Dropout	14.23	14.71	−0.16	−0.20	3.10	3.16	3.40	3.37
Features	0.05	Exact GP	14.75	15.80	−0.65	−0.29	3.50	3.43	3.39	3.45
Features	0.05	MDN	20.30	23.54	−0.65	−0.92	3.05	3.72	3.64	4.02
Features	0.05	MDN (4)	27.00	29.38	−1.20	−1.39	4.01	3.85	4.25	4.40
Target and Features	0.25	Concrete Dropout	13.78	46.52	−0.12	−2.79	3.29	5.17	3.37	5.73
Target and Features	0.25	Exact GP	14.75	39.99	−0.20	−2.26	3.50	4.08	3.39	4.08
Target and Features	0.25	MDN	25.54	202.67	−1.08	−15.54	3.85	7.58	4.21	10.52
Target and Features	0.25	MDN (4)	21.82	221.33	−0.78	−17.06	3.91	8.15	3.94	11.31

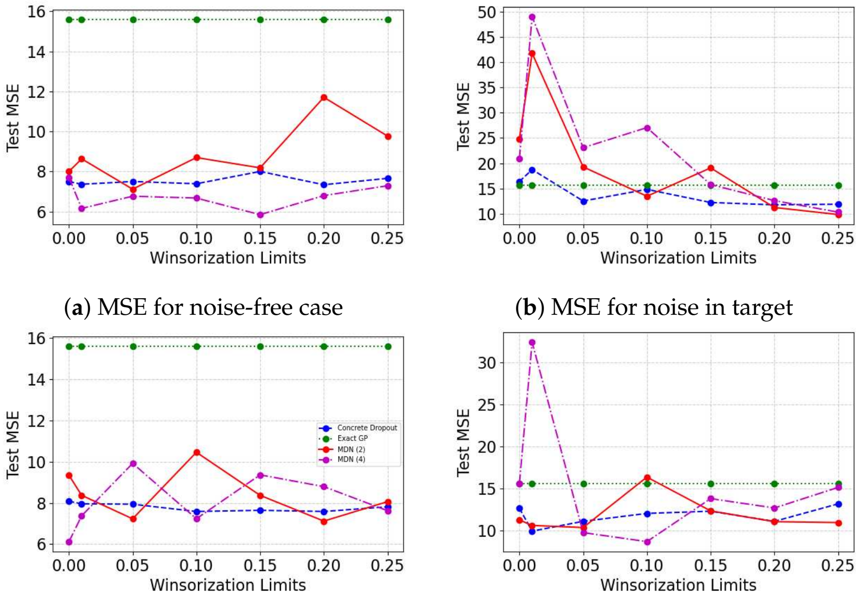
GDSC Data

As can be seen in Table 10, the GDSC dataset also proves to be a difficult dataset for the probabilistic models studied in this work. Among the models that were tested, VGP is able to achieve the best model performance followed by MDN with two mixing components.

Table 10. Probabilistic methods: GDSC.

Model	Test MSE	R^2	Run Time	Test Median Absolute Error
Concrete Dropout	7.77	−0.10	32 s	1.46
Exact GP	15.59	−1.25	3.8 s	3.75
MDN	7.78	−0.13	119 s	1.26
Flipout	80.67	−26.07	970 s	5.95
VGP	6.15	0.08	295 s	1.46

Figure 8 shows the model performance on the untouched test set, and the Winsorization limit is varied on the training and validation set. Apart from the noise in the target case, all cases show no definite improvement in model performance as the Winsorization limit is increased. For the noise in target case, the model performance improves as the degree of Winsorization is increased. However, in all cases, the exact GP performance is not affected by the varying Winsorization limits.



(**a**) MSE for noise-free case

(**b**) MSE for noise in target

(**c**) MSE for noise in features

(**d**) MSE for noise in target and features

Figure 8. Winsorization results from 0 to 25 percentile limits on GDSC dataset. Mean Squared Error is shown on the y-axis and the Winsorization limits are shown on the x-axis. Different lines represent different methods: Concrete dropout is shown as blue dashed line, exact GP is shown as green dotted line, mixture density network with 2 components is shown in red solid line, and mixture density network with 4 components is shown in magenta dashed-dotted line. As Winsorization limit increases in the training set, the model performance in terms of mean squared error in the untouched test set is shown in the sub-plots.

Table 11 shows us the model performance on the untouched test set and the contaminated test set for the optimal degree of Winsorization for different noise sites. For the untouched test set case, we notice marginal improvement in model performance for all probabilistic neural networks. For the contaminated test set case, we notice that the model performance improves or remains the same for concrete dropout and exact GP. For the mixture density networks, performance improvements are noticed for only a subset of cases.

Table 11. Winsorization results on test set for the GDSC dataset.

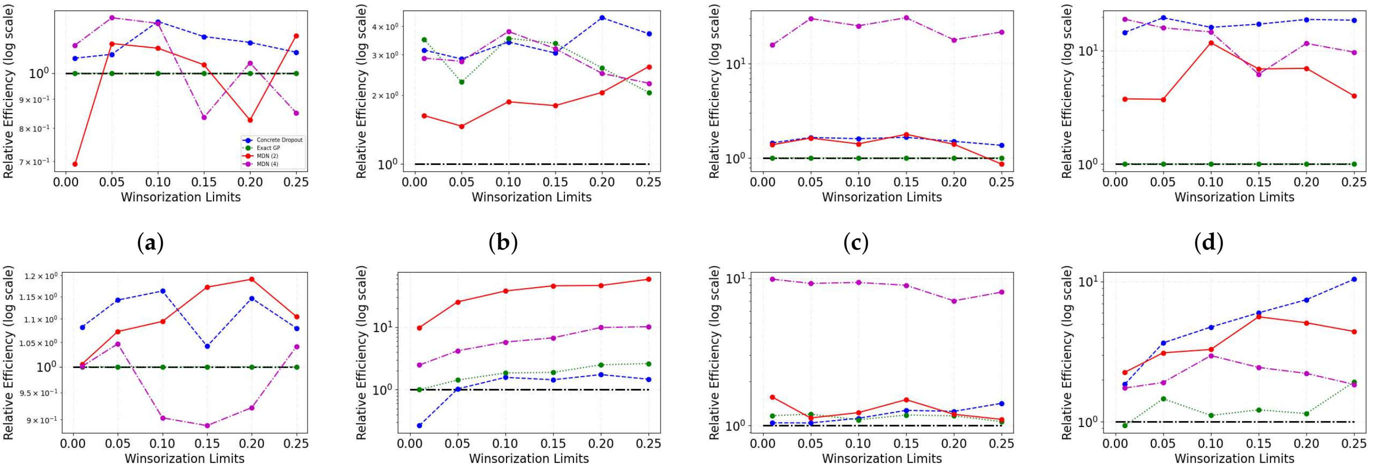
Noise Site	Optimal Limit	Model	MSE	MSE$_W$	R^2	R^2_W	Median AE	Median AE$_W$	MAE	MAE$_W$
None	0.05	Concrete Dropout	7.49	7.49	−0.08	−0.08	2.57	2.35	2.36	2.40
None	0.05	Exact GP	15.59	15.59	−1.25	−1.25	3.75	3.75	3.40	3.40
None	0.05	MDN	7.99	7.11	−0.15	−0.02	1.64	1.60	2.12	2.11
None	0.05	MDN (4)	7.71	6.76	−0.11	0.02	1.73	1.62	2.06	1.98
Untouched Test Set										
Target	0.25	Concrete Dropout	16.38	11.89	−1.37	−0.72	3.88	3.51	3.48	2.97
Target	0.25	Exact GP	15.59	15.59	−1.25	−1.25	3.75	3.75	3.40	3.40
Target	0.25	MDN	24.82	9.85	−2.59	−0.42	2.08	2.03	3.36	2.43
Target	0.25	MDN (4)	20.86	10.32	−2.01	−0.49	2.91	2.80	3.58	2.79
Features	0.2	Concrete Dropout	8.08	7.58	−0.16	−0.09	2.46	2.50	2.51	2.45
Features	0.2	Exact GP	15.59	15.59	−1.25	−1.25	3.75	3.75	3.40	3.40
Features	0.2	MDN	9.34	7.11	−0.35	−0.03	1.44	1.95	2.33	2.22
Features	0.2	MDN (4)	6.11	8.78	0.11	−0.27	1.57	2.26	2.00	2.54
Target and Features	0.25	Concrete Dropout	12.65	13.18	−0.83	−0.90	3.54	3.69	3.08	3.15
Target and Features	0.25	Exact GP	15.59	15.59	−1.25	−1.25	3.75	3.75	3.40	3.40
Target and Features	0.25	MDN	11.26	10.95	−0.63	−0.58	3.25	3.28	2.96	2.69
Target and Features	0.25	MDN (4)	15.59	15.13	−1.25	−1.19	3.75	3.51	3.40	3.37
Contaminated Test Set										
Target	0.25	Concrete Dropout	7.55	5.99	−0.09	0.13	2.31	2.19	2.32	2.12
Target	0.25	Exact GP	15.59	15.59	−1.25	−1.25	3.75	3.75	3.40	3.40
Target	0.25	MDN	12.23	6.67	−0.77	0.03	2.14	1.68	2.78	2.08
Target	0.25	MDN (4)	5.96	6.36	0.13	0.07	1.49	1.93	1.84	2.16
Features	0.25	Concrete Dropout	7.79	7.61	−0.12	−0.10	2.52	2.43	2.40	2.41
Features	0.25	Exact GP	15.59	8.70	−1.25	−0.26	3.75	2.55	3.40	2.54
Features	0.25	MDN	7.99	8.14	−0.15	−0.17	2.08	1.99	2.37	2.34
Features	0.25	MDN (4)	10.77	20.3	−0.55	−1.94	2.09	2.42	2.77	3.52
Target and Features	0.25	Concrete Dropout	7.22	6.71	−0.04	0.02	2.41	1.95	2.33	2.22
Target and Features	0.25	Exact GP	15.59	7.18	−1.25	−0.03	3.75	2.29	3.4	2.38
Target and Features	0.25	MDN	5.24	7.45	0.24	−0.07	1.76	1.62	1.95	2.11
Target and Features	0.25	MDN (4)	9.03	7.32	−0.30	−0.05	1.95	2.53	2.44	2.40

5. Discussion

Winsorization of the noise site in the presence of noise is able to aid in mitigating the adverse effects of outliers on the model performance for probabilistic neural networks. This can also be elucidated by measuring the relative efficiency (RE) of the Winsorized model performance with the pre-Winsorized model performance. RE can act as a metric that concisely conveys the impact of Winsorization on the test MSE. Figure 9 shows the logarithm of RE values, where RE is computed as follows:

$$\text{RE}_W \text{ (Relative Efficiency)} = \frac{\text{Test MSE}}{\text{Test MSE}_W} \tag{23}$$

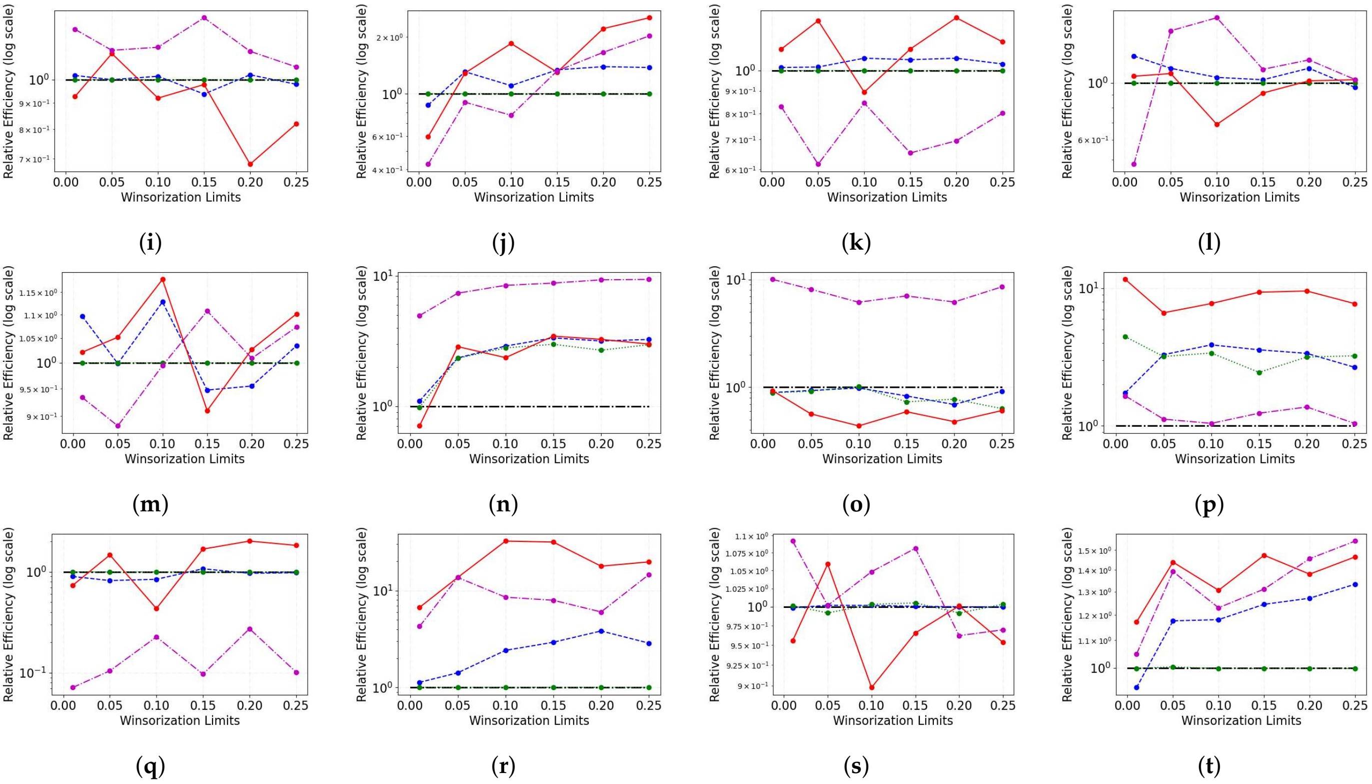
An RE value of one represents the same level of MSE while an RE value greater than one signifies model improvement with Winsorization. In most cases, Winsorization leads to an improvement in model performance. In the crop yield dataset, where we suspect that there are natural variations in the data apart from artificial noise that may be interfering with the models' ability to learn effectively, the RE values are always greater than one for all the noisy cases, barring exact GP. Exact GP is able to retain the same level of performance as pre-Winsorised learning when the noise is introduced in features but has improved performance when the noise is introduced in target. In other datasets as well, exact GP is not always able to see drastic model performance improvement. For more difficult datasets such as crop yield, GDSC, and forest fires, exact GP is not always the best performing in the presence of noise. For all datasets, the model performance is able to improve with Winsorization, especially when the noise site is target, especially at higher degrees of Winsorization. We see similar results when the noise site is target and features. The results on GDSC dataset are not indicative of definite improvement with the Winsorization of the training and validation data. On the other hand, the crop yield dataset shows the most improvement for all noise sites. This might be due to the fact that the architecture of the underlying fully connected neural network is optimized for the crop yield dataset using hyperparameter search by Bayesian optimization. Therefore, it may be meaningful in future work to investigate the effects of changing the architecture to be more suited for particular datasets.



(**a**) (**b**) (**c**) (**d**)

(**e**) (**f**) (**g**) (**h**)

Figure 9. *Cont.*



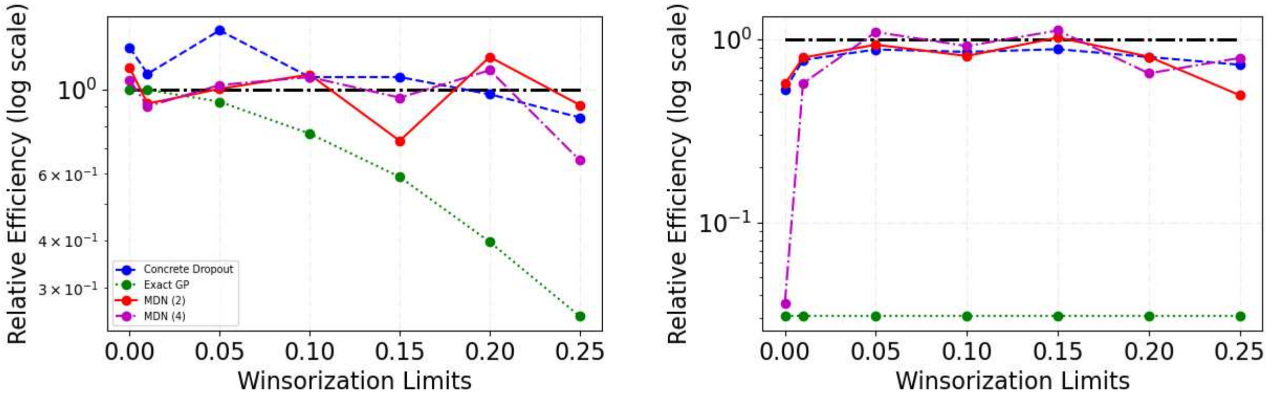
(**i**) (**j**) (**k**) (**l**)

(**m**) (**n**) (**o**) (**p**)

(**q**) (**r**) (**s**) (**t**)

Figure 9. Relative efficiencies (REs) of Winsorized MSE with non-Winsorized MSE for different noise sites. The black dashed line represents an RE of one. RE values greater than one represent improvement in performance with Winsorized training and validation data and vice versa. (**a**) RE for noise free case in crop yield data. (**b**) RE for noise in target in crop yield data. (**c**) RE for noise in features in crop yield data. (**d**) RE for noise in target and features in crop yield data. (**e**) RE for noise free case in California data. (**f**) RE for noise in target in California data. (**g**) RE for noise in features in California data. (**h**) RE for noise in target and features in California data. (**i**) RE for noise free case in GDSC data. (**j**) RE for noise in target in GDSC data. (**k**) RE for noise in features in GDSC data. (**l**) RE for noise in target and features in GDSC data. (**m**) RE for noise free case in forest fires data. (**n**) RE for noise in target in forest fires data. (**o**) RE for noise in features in forest fires data. (**p**) RE for noise in target and features in forest fires data. (**q**) RE for noise free case in Mauna data. (**r**) RE for noise in target in Mauna data. (**s**) RE for noise in features in Mauna data. (**t**) RE for noise in target and features in Mauna data.

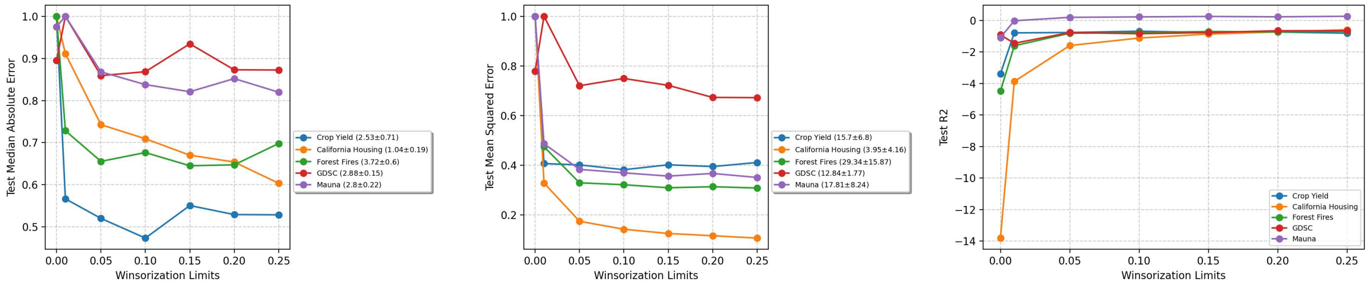
We can also compare the Winsorized results in the noisy cases with the noise free, non-Winsorized results. The case where there is contamination only in feature set allows for comparison of MSE on target variable in contaminated data with noise-free data. In Figure 10, the black dashed line is the unattainable gold standard of achieving the same results as the noise-free training in the presence of noise. Our objective is to come as close to it as possible. In the results shown here, we make the comparison for crop yield dataset, which has optimized architecture design and the most stable of all data use cases presented here. In Figure 10a, it is shown that lower degree of Winsorization in the training and validation datasets helps improve performance over noise free data for the probabilistic neural networks. As the degree of Winsorization increases, the loss of information adversely affects the exact GP performance. Figure 10b displays the relative efficiencies on the untouched dataset for the noise in features case as well. For all the neural networks, the relative efficiency increases as we Winsorize the training and validation datasets. For exact GP, the relative efficiency does not change as Winsorization limit is increased. However, the model performance as indicated by relative efficiency does not remain at the same level as in the noise-free, non-Winsorized scenario. The neural networks are able to recover similar level of performance as the noise-free, non-Winsorized case. However, a higher degree of Winsorization lead to a loss of information that adversely affects the model performance for all probabilistic neural networks. For the untouched test set, the RE for feature noise site as compared to the RE when noise site is target (not shown here) indicate that for different datasets, Winsorization helps the most when the noise is in the feature itself. The RE results on feature noise site (not shown here) indicate that a lower degree of Winsorization (up to 10 percentile in our experiment results) aids in recovering the original model performance.



(**a**) RE for noise in feature case in contaminated test set (**b**) RE for noise in feature case in untouched test set

Figure 10. Crop yield dataset result: Relative Efficiencies (RE) comparing performance of Winsorized results with standard Cauchy noise in the features with original performance on noise free data without Winsorization. Black dashed line represents RE of one. REs above one represent improvement in performance due to Winsorization on contaminated data.

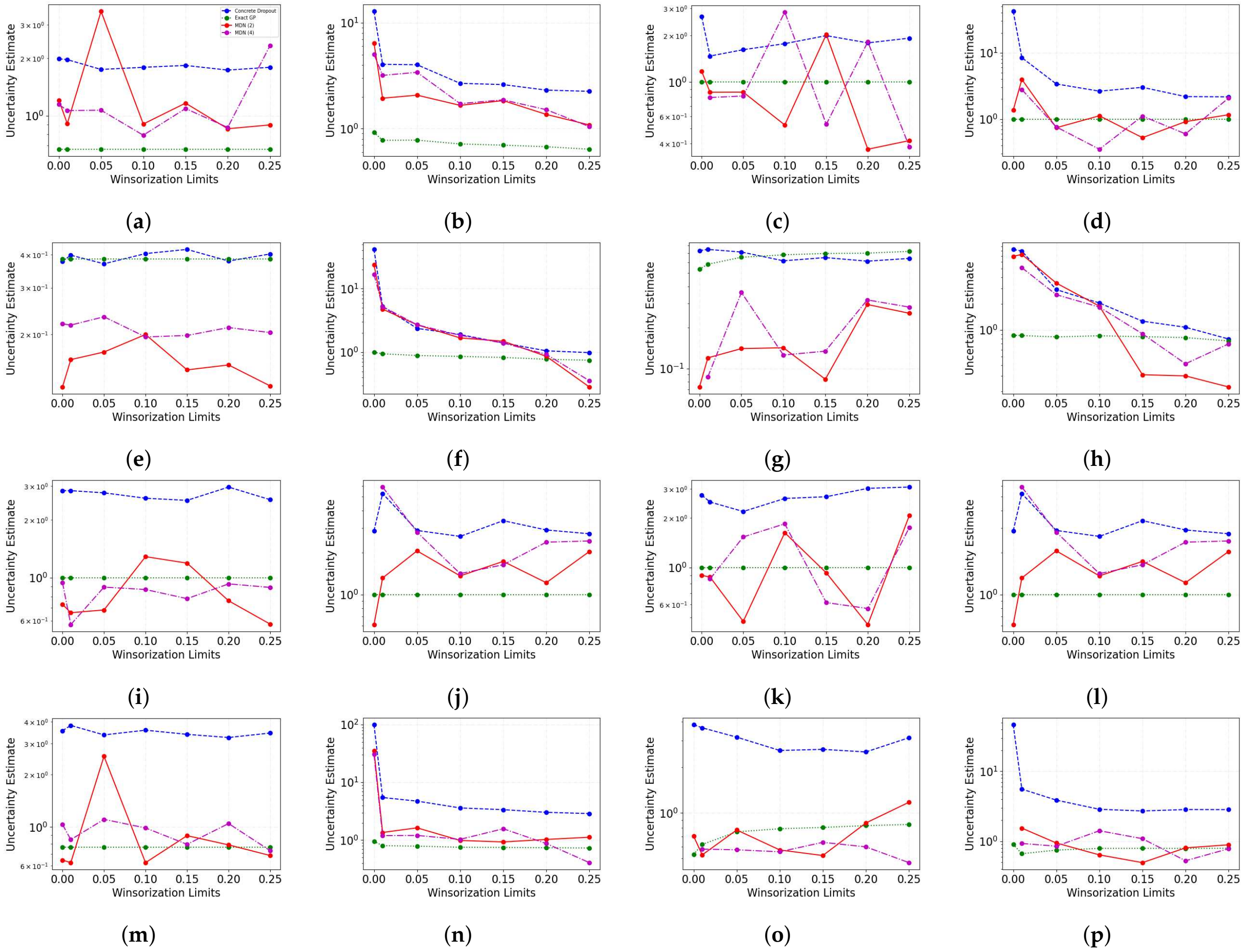
Figure 11 shows the average model performance over all noise sites (feature, target, and both feature and target) for different Winsorization limits on the untouched test set. The median absolute error and mean squared error are scaled by the maximum value of the respective metrics for each dataset. The mean and variance of the metrics are mentioned in the legend for reference. We notice a general trend of model improvement up to a certain limit. We also notice that GDSC is unable to achieve model performance improvement. For the more challenging datasets, it may be material to optimize the architecture to obtain more stable models for further experimentation.



(**a**) Average Median Absolute Error (**b**) Average Mean Squared Error (**c**) Average Coefficient of Determination

Figure 11. Summarizing Winsorization results: The subplots show average of evaluation metrics over all methodologies used for cases when artificial perturbation is introduced in the datasets. The MSE and Median AE plot legends also convey the mean and standard error of the evaluation metric in the respective sub-plots for each dataset.

Similar to comparison of model performance in terms of the mean squared error, the uncertainty of different probabilistic models can also be compared. Figure 12 shows how uncertainty in prediction changes as the degree of Winsorization is increased. While exact GP uncertainty estimates are mildly affected by change in Winsorization limits, there are discernible slight changes in uncertainty for other methods. As noise is added to target, the uncertainty estimates increase for a subset of datasets. For the datasets where this is visible in the pre-Winsorization uncertainty estimate values, with Winsorization, uncertainty estimates drop to a lower level as the degree of Winsorization is increased. Adding noise in features does not heavily influence the uncertainty estimates. Even in cases where there is noise in target and features, we can see a decrease in uncertainty estimates as the training and validation data are Winsorized. However, the erratic behavior of uncertainty estimates for mixture density networks requires further investigation into the source of volatility.



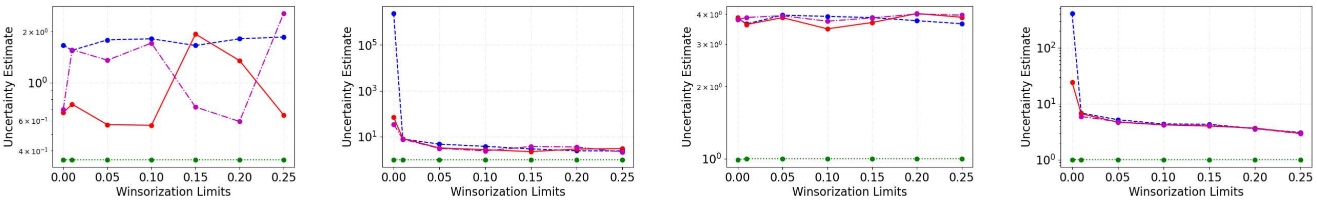
(**a**) (**b**) (**c**) (**d**)

(**e**) (**f**) (**g**) (**h**)

(**i**) (**j**) (**k**) (**l**)

(**m**) (**n**) (**o**) (**p**)

Figure 12. *Cont.*



(**q**) (**r**) (**s**) (**t**)

Figure 12. Apart from predictive performance in terms of accurate prediction, the precision can also be compared in terms of uncertainty estimates. On the y-axis, we measure the average standard error in prediction. (**a**) Uncertainty estimate for noise free case in crop yield data. (**b**) Uncertainty estimate for noise in target in crop yield data. (**c**) Uncertainty estimate for noise in features in crop yield data. (**d**) Uncertainty estimate for noise in target and features in crop yield data. (**e**) Uncertainty estimate for noise free case in California data. (**f**) Uncertainty estimate for noise in target in California data. (**g**) Uncertainty estimate for noise in features in California data. (**h**) Uncertainty estimate for noise in target and features in California data. (**i**) Uncertainty estimate for noise free case in GDSC data. (**j**) RE for noise in target in GDSC data. (**k**) Uncertainty estimate for noise in features in GDSC data. (**l**) Uncertainty estimate for noise in target and features in GDSC data. (**m**) Uncertainty estimate for noise free case in forest fires data. (**n**) Uncertainty estimate for noise in target in forest fires data. (**o**) Uncertainty estimate for noise in features in forest fires data. (**p**) Uncertainty estimate for noise in target and features in forest fires data. (**q**) Uncertainty estimate for noise free case in Mauna data. (**r**) Uncertainty estimate for noise in target in Mauna data. (**s**) Uncertainty estimate for noise in features in Mauna data. (**t**) Uncertainty estimate for noise in target and features in Mauna data.

6. Conclusions

We compare different probabilistic neural networks in terms of model performance and time taken for training. Among the different methods that we employ, VGP based neural network, flipout, and concrete dropout solely rely on variational free energy for learning the variational posterior distribution (optimizing variational hyperparameters). MDN uses the negative log likelihood for estimating parameters that define response distribution. Exact GP also depends on optimizing for the marginal log likelihood to estimate the hyperparameters. In terms of the model performance, we see varied results for the different methods on different datasets. In general, when the flipout gradient estimation is used for all layers, the model performance and training time are adversely affected. The additional matrix computations in flipout make it more expensive. Recent literature also sheds light on the heave-tailed distribution of deep Gaussian processes [114,115]. Further experimentation on methods such as concrete dropout that try to provide approximation of deep Gaussian processes can be conducted to understand the properties of the predictive distribution arising from the implementation of such methods.

We experimented with the use of Winsorization to make probabilistic deep learning models more robust against outliers. Over several data sets, we obtained several model performance results for different noise sites and degrees of Winsorization. Through the results that we observed on the untouched test dataset, we are able to observe the effects of introducing noise and Winsorization on the training and validation dataset. Using an untouched test set enables an easy comparison with the original performance of the models on noise-free, pre-Winsorized datasets. We also obtained model performance results on noisy test set data. This helps us in exploring more realistic use cases where it is known that the whole data set is contaminated from noise or perturbations. For our case study on the crop yield dataset, it is shown that in the presence of noise in the features, Winsorization helps the models in recovering model performance, both when tested on untouched test set, and contaminated test set. Noise in the dataset drastically degrades the Exact GP performance. It further worsens by the loss of information as the degree of Winsorization increases.

We notice that for several experiments, Winsorization produces unfavorable results due to a loss of information. This much has been noticed for linear regression problems [116] as well. It has been shown that the Winsorization of the features and response increases the Mean Squared Error of the regression and also increases the variance in the estimates of the coefficients. As we have seen in the noise-free scenario in our experiments, this is not always the case for the Bayesian neural networks. Through the results shown in contaminated test data case, we see that the models learn the noise along with other signals in the data, as is evident from MSE values and coefficient of determination on the pre-Winsorized data. Winsorization is unable to improve the performance as the i.i.d. Cauchy perturbations degrade the quality of training that happens for the neural networks. On the other hand, the untouched test set does not have the artificial perturbations in the hold-out set, enabling us a direct comparison of the evaluation metric results with the noise-free dataset results. For most cases, Winsorization in the training and validation dataset clearly improves the ability of neural networks to learn the more centrally located values in the dataset. The performance change on the untouched test set suggests that a lower degree of Winsorization on the dataset might be beneficial in training. Meanwhile, the evaluation results on the contaminated test set may point towards the non-existence of universally stable neural networks in today's deep learning frameworks that can withstand any perturbations [36]. These results also follow the deeper cause of the instability—the functional relation between the input and target are often based on correlations existing in the observed, real-world data. These functional relations are often brittle and disrupted by the perturbations. From the model's perspective, the noisy and non-noisy samples in the data are potentially equally important for learning effectively at the start of the training process. These perturbations are learned by models, and often several models that learn independently on the same data learn the adversarial perturbations the same way. This perturbation learning can be transferred from one model to another, where each is trained independently. This is called adversarial transferability [35]. Similar to our results, this is proven on image classification problems where the evaluation metric results on the untouched test set yields improved accuracy. This suggests that robustness as a challenge is not only tied to the training of the models but is also a property of the dataset, making treatments such as Winsorization amenable to more effective learning.

Despite the vast literature focusing on the robustness of neural networks against adversarial attacks, producing a robust framework is still a daunting problem. It is therefore meaningful to study the effects of Winsorization on the instability that perturbations cause in neural network training. Instead of introducing perturbations to all samples in the data, as we did in our experiments, studying the effects of the sparser addition of perturbations to certain samples or certain features may be closer to the experimental setups in the current studies. In our study, we saw that Winsorization on the feature set in the presence of noise aided in obtaining good evaluation results. Choosing a Winsorization limit more adaptively for individual features may enable more effective learning as certain features may be more prone to instability arising from perturbations than the other features. A comparative analysis of the effects of Winsorization with other methodologies to deal with noise and outliers would also provide insights into the relative efficacy of these methods [35,117].

It is also a well-known result that training for several epochs does not adversely affect the generalization capabilities of neural networks, especially in overparameterized regimes. The same has been proven to not be true when training robust networks that deal against adversarial attacks. In fact, it has been shown that early stopping during the learning process can be more beneficial than several of the adversarial training algorithms that have been proposed to deal with adversarial examples. It would be interesting to explore how tuning the number of epochs for early stopping in our experimental setup would change the results.

Author Contributions: S.S. conducted the data analyses reported here, under the guidance of S.C.The planning and writing of this paper were done jointly by the authors. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially supported by the US National Science Foundation (NSF) under grants 1737918, 1939916, 1939956, and a grant from Cisco Systems Inc.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We have used publicly available datasets in this paper.

Acknowledgments: We would like to thank Deepak Ray for providing the crop yield dataset and also for providing insights about it. We would also like to thank the editors and reviewers for providing detailed comments.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Loss Functions for Mixture Density Networks

Apart from experimenting with different probabilistic models and data treatments, we can also focus on how training is affected in presence of noise when different loss functions are used. We can experiment with different loss functions depending on how they measure distance differently and how the model learning is affected by perturbations in all samples as noise site is varied. In our vanilla implementation of mixture density networks, we use the negative log likelihood for guiding the training of the model. We define the different loss functions that we use in our experiments in Appendix A.1. On the crop yield dataset, over 100 repetitions, we average the loss function values as shown in Table A1.

Appendix A.1. Loss Function Definitions

Appendix A.1.1. Negative Log Likelihood

$$\mathcal{F}_{NLL} = \frac{-1}{N} log(\text{likelihood} \times \text{prior}) \tag{A1}$$

$$= \frac{-1}{N} log(P(y|x, W)p(W))$$

$$\mathcal{F}_{NLL} \approx \frac{-1}{N} \sum_i log(\sum_k \pi_k(x_i, W)\mathcal{N}(y_i|mu_k(x_i, W), \sigma_k^2(x_i, W))) \tag{A2}$$

Appendix A.1.2. KL Divergence

$$\mathcal{F}_{KL} = \sum_i y_i \quad (log y_i - log(\sum_k \pi_k(x_i, W)\mathcal{N}(y_i|\mu_k(x_i, W), \sigma_k^2(x_i, W)))) \tag{A3}$$

Appendix A.1.3. Heteroscedastic Loss

$$\mathcal{F}_{HL} = \sum_i exp(-log(var))(y_i - \text{mean})^2 + log(var) \tag{A4}$$

$$= \sum_i exp(-log(\sum_k \pi_k \sigma_k^2 + \sum_k \pi_k \mu_k^2 - (\sum_k \pi_k \mu_k)^2))+$$

$$(y_i - \sum_k \pi_k \mu_k)^2 +$$

$$log((\sum_k \pi_k \sigma_k^2 + \sum_k \pi_k \mu_k^2 - (\sum_k \pi_k \mu_k)^2)$$

where each μ_k, π_k, σ_k depend on x and W.

Appendix A.1.4. Logarithm of Cos h

$$\mathcal{F}_{LC} = \sum_i log(\frac{exp(Z_i) + exp(-Z_i)}{2}) \tag{A5}$$

where $Z_i = y_i - \sum_k \pi_k(x_i, W)\mathcal{N}(y_i|\mu_k(x_i, W), \sigma_k^2(x_i, W))$.

Appendix A.2. Experiment Results

As can be seen, the negative log likelihood loss function is adversely affected by Cauchy noise. Other loss functions seem to be more robust against noise, as is shown in the mean squared error values. However, several of these loss functions focus on minimising the distance of the observed response with the realizations of the conditional mean predictive distribution instead of learning a complex posterior predictive distribution. Therefore, changing the loss function will require more careful analysis into which loss functions are able to preserve the complexity of the conditional target distribution.

Table A1. MDN 4 components, different loss functions.

Loss	NLL (Default)	MSE	LC	NLL + LC	HL	NLL + KL	Median AE
MSE (Noise-free)	2.88	2.73	2.38	4.07	2.51	2.50	4.77
MSE (Cauchy in features)	72.27	3.69	3.83	58.14	3.97	72.27	4.51
MSE (Cauchy in target)	23.19	422.82	7.20	10.41	10.45	60.58	5.73
MSE (Cauchy in target and features)	35.92	14.32	9.17	72.27	7.81	39.92	5.39

Appendix B. Adjusting for Long-Term Trend and Seasonality in the Mauna CO_2 Dataset

As addressed in the main text in Section 4, Mauna CO_2 atmospheric concentration is a time-series dataset that can be adjusted for the long-term trend and the seasonal trends. This adjustment becomes even more necessary when the perturbations that we are introducing are i.i.d Cauchy that can easily exacerbate model performance when the feature set is only one dimensional. Meanwhile, the main results are adjusted for these time series components, and we present the results before adjustment as follows.

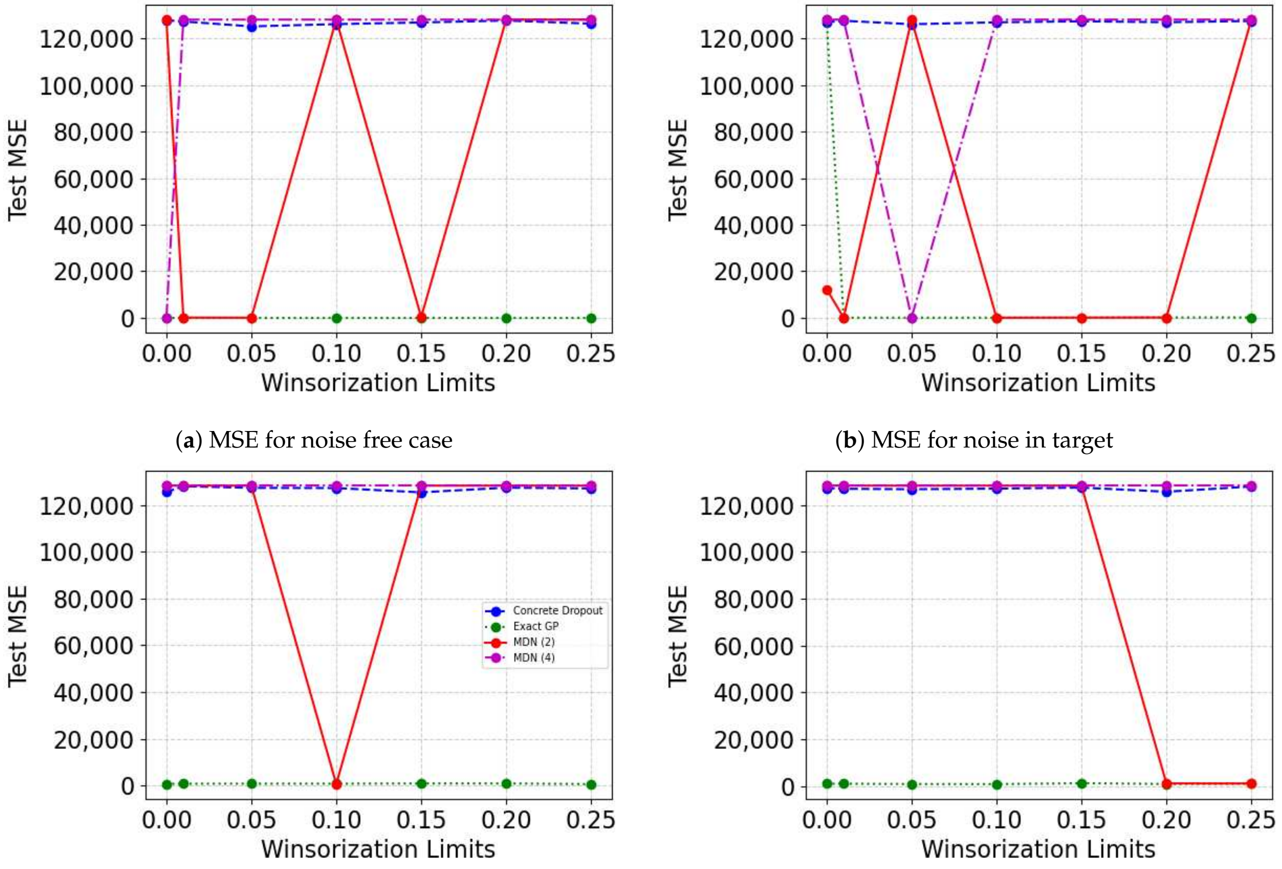
Before Adjustment

In Table A2, it can be seen that flipout is unable to converge to a representative posterior predictive distribution and takes the most time to train the model. Variational GP performs relatively well and has relatively shorter run time for model training. The dropout regularization in concrete dropout adversely affects the learning due to the low dimensionality of the feature set. Mixture density networks and Gaussian processes perform the best where MDN has slightly faster training time.

Table A2. Probabilistic methods: Mauna CO_2 concentration.

Model	Test MSE	R^2	Run Time	Test Median Absolute Error
Concrete Dropout	1318	−0.60	23 s	25
VGP	69	0.62	60 s	4.48
Exact GP	1.41	0.99	34 s	0.97
MDN	10.79	0.98	16 s	2.67
Flipout	1899	−12.9	225 s	29.74

Figure A1 show the Test MSE_W on the untouched test set as the Winsorization limits on the training and validation test are increased from 0 to 25 percentile. For all noise sites, including the case when there is no noise in the training and validation set, the probabilistic models are unable to perform consistently as Winsorization limits change. Exact GP produces the best test mean squared error values while concrete dropout always performs worse than other methods. Even before introducing noise, the concrete dropout and mixture density network (four mixing components) are unable to perform well. In the noise-free case, as the Winsorization limit is increased to 1 percentile, the mixture density network with two components and the mixture density network with four components have drastically different performance. When we introduce noise, we notice the same kind of instability in the mixture density networks for all noise cases.



(**a**) MSE for noise free case

(**b**) MSE for noise in target

(**c**) MSE for noise in features

(**d**) MSE for noise in target and features

Figure A1. Winsorization results from 0 to 25 percentile limits on Mauna dataset. Mean Squared Error is shown on the y-axis and the Winsorization limits are shown on the x-axis. Different lines represent different methods: Concrete dropout is shown as blue dashed line, exact GP is shown as green dotted line, mixture density network with 2 components is shown in red solid line, and mixture density network with 4 components is shown in magenta dashed-dotted line. As Winsorization limit increases in the training set, the model performance in terms of Mean Squared Error in the untouched test set is shown in the sub-plots.

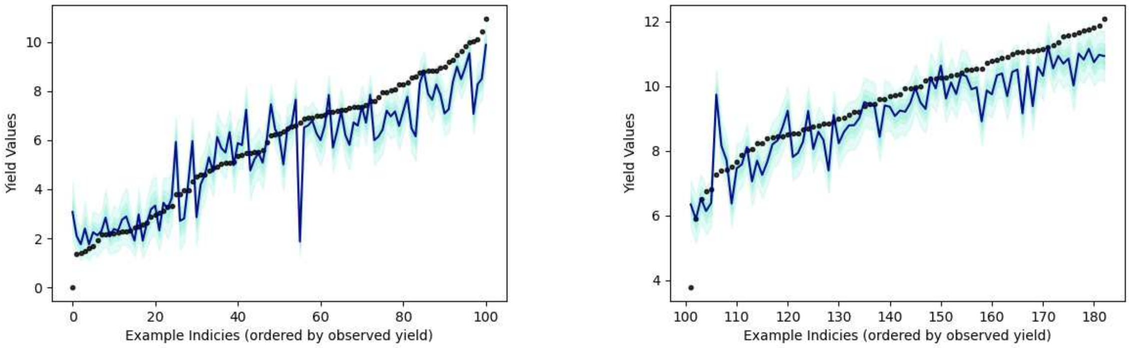
Table A3 displays the change in the model performance. Similar to previous data sets, the model performance improves with Winsorization for the untouched test set while it does not for the contaminated test set. For the untouched test set case, while there is a Winsorization limit for which most methods see an improvement in performance, the model performance improves only slightly for the concrete dropout case. The optimal Winsorization limit for the case when the noise is introduced in features is also higher than it was for the previous datasets.

Table A3. Winsorization results on test set for Mauna CO_2 dataset.

Noise Site	**Optimal Limit**	**Model**	**MSE**	**MSE$_W$**	**R^2**	**R^2_W**	**Median AE**	**Median AE$_W$**	**MAE**	**MAE$_W$**
None	0.05	Concrete Dropout	1.26×10^6	1.28×10^6	−140.24	−141.94	349.94	352.14	354.61	356.74
None	0.05	Exact GP	1.88	1.88	0.99	0.99	1.06	1.06	1.12	1.12
None	0.05	MDN	10.07	91.06	0.98	0.89	2.49	6.20	2.68	7.59
None	0.05	MDN (4)	1.24×10^6	8.48	−128.89	0.99	343.47	2.15	352.04	2.41
Untouched Test Set										
Target	0.01	Concrete Dropout	1.27×10^6	1.25×10^6	−141.10	−139.14	350.95	348.62	355.69	353.23
Target	0.01	Exact GP	1.28×10^6	21.31	−142.01	0.97	352.15	1.76	356.83	1.76
Target	0.01	MDN	2×10^5	24.40	−24.31	0.97	91.41	3.49	116.75	3.97
Target	0.01	MDN (4)	1.28×10^6	12.81	−142.01	0.98	352.14	2.44	356.83	2.89
Features	0.2	Concrete Dropout	1.27×10^6	1.25×10^6	−140.93	−139.08	350.74	348.55	355.48	353.14
Features	0.2	Exact GP	792.49	940.82	0.11	−0.04	18.76	19.01	22.22	23.77
Features	0.2	MDN	546.45	851.44	0.39	0.05	14.90	23.47	18.5	24.85
Features	0.2	MDN (4)	1.28×10^{-6}	827.14	−142.01	0.07	352.14	22.46	356.83	23.77
Target and Features	0.1	Concrete Dropout	1.27×10^6	1.26×10^6	−141.21	−140.01	351.28	349.29	355.82	354.31
Target and Features	0.1	Exact GP	947.30	763.98	−0.05	0.14	21.21	20.13	24.36	22.34
Target and Features	0.1	MDN	1.28×10^6	886.47	−142.01	0.01	352.15	23.63	356.83	25.3
Target and Features	0.1	MDN (4)	1.28×10^6	1.28×10^6	−142.01	−137.05	352.14	352.14	356.83	356.83
Contaminated Test Set										
Target	0.01	Concrete Dropout	1.25×10^6	1.27×10^6	−138.70	−141.29	348.10	351.31	352.65	355.93
Target	0.01	Exact GP	1.88	2.11	0.99	0.99	1.07	1.11	1.12	1.17
Target	0.01	MDN	101.57	6.25	0.88	0.99	9.37	1.67	9.38	1.97
Target	0.01	MDN (4)	8450.96	6.79	−8.42	0.99	88.47	1.71	87.14	2.06
Features	0.25	Concrete Dropout	1.27×10^6	1.23×10^6	−140.90	−127.07	350.82	349.42	355.44	354.32
Features	0.25	Exact GP	1.88	803.73	0.99	0.10	1.06	2.72	1.12	15.17
Features	0.25	MDN	313.79	723.44	0.65	0.19	12.66	8.97	14.33	18.65
Features	0.25	MDN (4)	1.28×10^6	503.59	−142.01	0.43	352.14	10.78	356.83	16.60
Target and Features	0.01	Concrete Dropout	1.25×10^6	1.27×10^6	−138.93	−141.44	348.23	351.39	352.96	356.11
Target and Features	0.01	Exact GP	1.88	3.26	0.99	0.99	1.06	1.11	1.12	1.27
Target and Features	0.01	MDN	22.92	1410.19	0.97	−0.57	3.58	36.79	3.90	36.80
Target and Features	0.01	MDN (4)	5.13	9.55	0.99	0.98	1.83	1.94	1.90	2.44

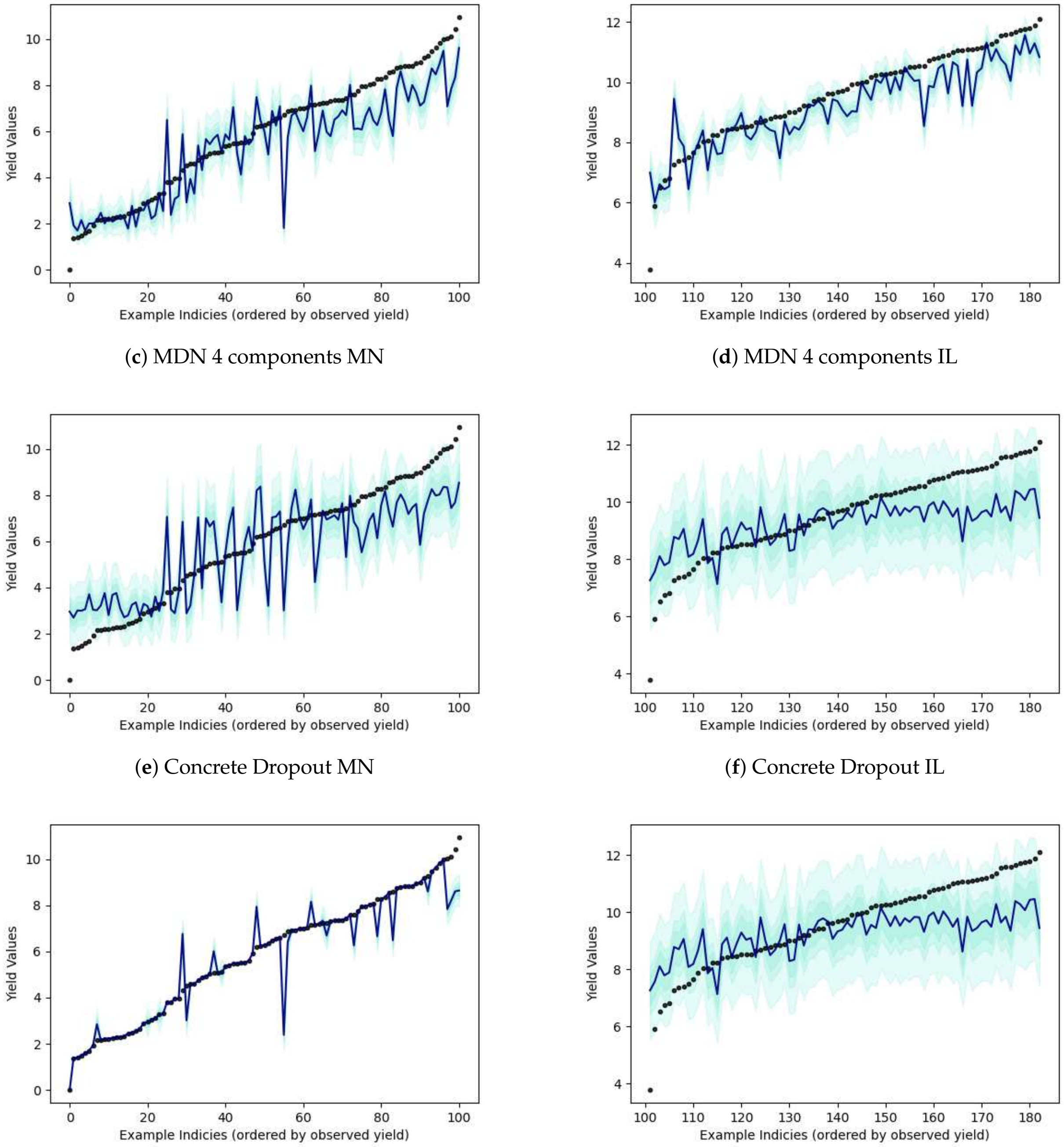
Appendix C. Aleatoric Uncertainty

Due to increasing relevance of uncertainty quantification, it is imperative to make a distinction between the sources of uncertainty. The recent literature in machine learning [51,118,119] also notes the difference in the source. The two types of uncertainty that we measure and show are epistemic and aleatoric uncertainty. Epistemic uncertainty arises due to inadequate knowledge about the optimal model to solve the task. It may arise due to insufficient data or due to an imperfect model structure. Adding more data or improving the model architecture can help mitigate this type of uncertainty, which is why it is reducible. Opposed to this, aleatoric uncertainty is the irreducible uncertainty. This type of uncertainty arises due to the stochastic behavior of the model rather than any insufficiency. In decision-making theory [120], epistemic uncertainty relates more to the inherent confidence of an event occurring while aleatoric uncertainty relates more to the understanding of the distributional behavior of outcomes. However, depending on the context and use case, the sources of uncertainty in the model may be defined differently. In our supervised learning case, epistemic uncertainty reduces with an increase in the amount of training data. In probabilistic models that are usually models over functions, epistemic uncertainty can be captured by the range of possible predictive functions. Aleatoric uncertainty can be explained by the amount of noise in the data [51], which does not change when the size of the training data set is varied. For a probabilistic deep learning model, epistemic uncertainty is measured as the variation in the function realizations for fixed input set, and the aleatoric uncertainty is measured through the model standard error, which can be estimated differently depending on the model that is being used. On the crop yield dataset, the epistemic and aleatoric uncertainties are computed and shown. Figure 3 shows the one standard deviation of epistemic uncertainty measured over 100 realizations for the fixed input data while the following figure, Figure A2, displays the one standard deviation of the aleatoric uncertainty. For Illinois counties, uncertainty is higher—except when measuring the epistemic uncertainty for exact GP implementation. For reference, Figure A3 shows the point predictions as well. The dispersion in the point predictions is closely evinced in the aleotoric uncertainty plots in Figure A2 as well. Concrete dropout results in Minnesota show higher uncertainty and variation as opposed to the mixture density results while exact GP uncertainty is very low. In Illinois, the dispersion in prediction is higher for concrete dropout and lowest for exact GP.



(**a**) MDN MN (**b**) MDN IL

Figure A2. *Cont.*



(**c**) MDN 4 components MN

(**d**) MDN 4 components IL

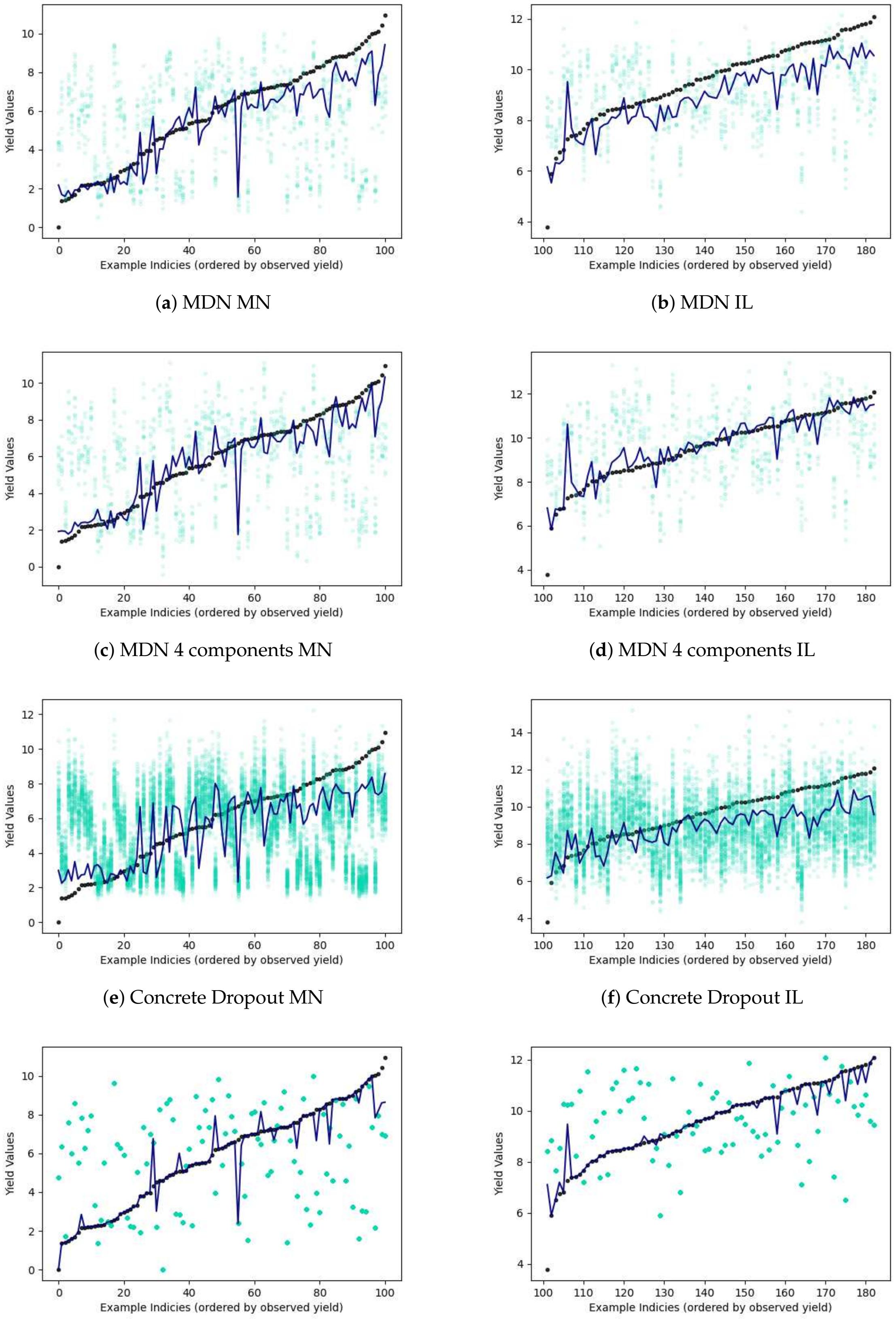
(**e**) Concrete Dropout MN

(**f**) Concrete Dropout IL

(**g**) Exact GP MN

(**h**) Exact GP IL

Figure A2. Crop yield predictions. X-axis shows arbitrary county indices which are sorted by the observed yield in ascending order. Y-axis represents the yield value. Black points are the observed yield. Navy blue line is the mean prediction, and aleatoric uncertainty estimates are shown in turquoise.



(**a**) MDN MN

(**b**) MDN IL

(**c**) MDN 4 components MN

(**d**) MDN 4 components IL

(**e**) Concrete Dropout MN

(**f**) Concrete Dropout IL

(**g**) Exact GP MN

(**h**) Exact GP IL

Figure A3. Crop yield predictions. X-axis shows arbitrary county indices which are sorted by the observed yield in ascending order. Y-axis represents the yield value. Black points are the observed yield. Navy blue line is the mean prediction and point predictions are shown in turquoise.

References

1. Yuan, K.H.; Bentler, P.M. Effect of outliers on estimators and tests in covariance structure analysis. *Br. J. Math. Stat. Psychol.* **2001**, *54*, 161–175. [CrossRef]
2. Huggins, R. A robust approach to the analysis of repeated measures. *Biometrics* **1993**, *49*, 715–720. [CrossRef]
3. Leeb, H.; Pötscher, B.M. Model selection and inference: Facts and fiction. *Econom. Theory* **2005**, *21*, 21–59. [CrossRef]
4. Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* **2021**, *64*, 107–115. [CrossRef]
5. Bartlett, P.L.; Long, P.M.; Lugosi, G.; Tsigler, A. Benign overfitting in linear regression. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 30063–30070. [CrossRef]
6. Wei, X.; Zhu, J.; Yuan, S.; Su, H. Sparse adversarial perturbations for videos. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 8973–8980.
7. Wallace, E.; Stern, M.; Song, D. Imitation attacks and defenses for black-box machine translation systems. *arXiv* **2020**, arXiv:2004.15015.
8. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; Song, D. Robust physical-world attacks on deep learning visual classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 1625–1634.
9. Wang, W.; Wang, R.; Wang, L.; Wang, Z.; Ye, A. Towards a robust deep neural network in texts: A survey. *arXiv* **2019**, arXiv:1902.07285.
10. Samanta, S.; Mehta, S. Towards crafting text adversarial samples. *arXiv* **2017**, arXiv:1707.02812.
11. Papernot, N.; McDaniel, P.; Swami, A.; Harang, R. Crafting adversarial input sequences for recurrent neural networks. In Proceedings of the MILCOM 2016—2016 IEEE Military Communications Conference, Baltimore, MD, USA, 1–3 November 2016; pp. 49–54.
12. Ren, S.; Deng, Y.; He, K.; Che, W. Generating natural language adversarial examples through probability weighted word saliency. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 1085–1097.
13. Jin, D.; Jin, Z.; Zhou, J.T.; Szolovits, P. Is bert really robust? A strong baseline for natural language attack on text classification and entailment. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 8018–8025.
14. Garg, S.; Ramakrishnan, G. Bae: Bert-based adversarial examples for text classification. *arXiv* **2020**, arXiv:2004.01970.
15. Li, L.; Ma, R.; Guo, Q.; Xue, X.; Qiu, X. Bert-attack: Adversarial attack against bert using bert. *arXiv* **2020**, arXiv:2004.09984.
16. Li, J.; Ji, S.; Du, T.; Li, B.; Wang, T. Textbugger: Generating adversarial text against real-world applications. *arXiv* **2018**, arXiv:1812.05271.
17. Zhou, Y.; Jiang, J.Y.; Chang, K.W.; Wang, W. Learning to discriminate perturbations for blocking adversarial attacks in text classification. *arXiv* **2019**, arXiv:1909.03084.
18. Wang, X.; Jin, H.; He, K. Natural language adversarial attacks and defenses in word level. *arXiv* **2019**, arXiv:1909.06723.
19. Shafahi, A.; Najibi, M.; Ghiasi, A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L.S.; Taylor, G.; Goldstein, T. Adversarial training for free! *arXiv* **2019**, arXiv:1904.12843.
20. Liu, H.; Zhang, Y.; Wang, Y.; Lin, Z.; Chen, Y. Joint character-level word embedding and adversarial stability training to defend adversarial text. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 8384–8391.
21. Jones, E.; Jia, R.; Raghunathan, A.; Liang, P. Robust encodings: A framework for combating adversarial typos. *arXiv* **2020**, arXiv:2005.01229.
22. Jia, R.; Raghunathan, A.; Göksel, K.; Liang, P. Certified robustness to adversarial word substitutions. *arXiv* **2019**, arXiv:1909.00986.
23. Katz, G.; Huang, D.A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. The marabou framework for verification and analysis of deep neural networks. In Proceedings of the International Conference on Computer Aided Verification, New York, NY, USA, 15–18 July 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 443–452.
24. Fazlyab, M.; Morari, M.; Pappas, G.J. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Trans. Autom. Control* **2020**. [CrossRef]
25. Raghunathan, A.; Steinhardt, J.; Liang, P. Certified defenses against adversarial examples. *arXiv* **2018**, arXiv:1801.09344.
26. Dvijotham, K.; Gowal, S.; Stanforth, R.; Arandjelovic, R.; O'Donoghue, B.; Uesato, J.; Kohli, P. Training verified learners with learned verifiers. *arXiv* **2018**, arXiv:1805.10265.
27. Huang, P.S.; Stanforth, R.; Welbl, J.; Dyer, C.; Yogatama, D.; Gowal, S.; Dvijotham, K.; Kohli, P. Achieving verified robustness to symbol substitutions via interval bound propagation. *arXiv* **2019**, arXiv:1909.01492.
28. Rice, L.; Wong, E.; Kolter, Z. Overfitting in adversarially robust deep learning. In Proceedings of the International Conference on Machine Learning, Virtual Event, 12–18 July 2020; pp. 8093–8104.
29. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2014**, arXiv:1412.6572.
30. Robey, A.; Hassani, H.; Pappas, G.J. Model-Based Robust Deep Learning: Generalizing to Natural, Out-of-Distribution Data. *arXiv* **2020**, arXiv:2005.10247.

31. Taori, R.; Dave, A.; Shankar, V.; Carlini, N.; Recht, B.; Schmidt, L. Measuring robustness to natural distribution shifts in image classification. *arXiv* **2020**, arXiv:2007.00644.
32. Rivest, L.P. Statistical properties of Winsorized means for skewed distributions. *Biometrika* **1994**, *81*, 373–383. [CrossRef]
33. Wu, M.; Zuo, Y. Trimmed and Winsorized means based on a scaled deviation. *J. Stat. Plan. Inference* **2009**, *139*, 350–365. [CrossRef]
34. Yale, C.; Forsythe, A.B. Winsorized regression. *Technometrics* **1976**, *18*, 291–300. [CrossRef]
35. Ilyas, A.; Santurkar, S.; Tsipras, D.; Engstrom, L.; Tran, B.; Madry, A. Adversarial examples are not bugs, they are features. *arXiv* **2019**, arXiv:1905.02175.
36. Bastounis, A.; Hansen, A.C.; Vlačić, V. The mathematics of adversarial attacks in AI–Why deep learning is unstable despite the existence of stable neural networks. *arXiv* **2021**, arXiv:2109.06098.
37. Shibzukhov, Z. Robust neural networks learning: New approaches. In Proceedings of the International Symposium on Neural Networks, Minsk, Belarus, 25–28 June 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 247–255.
38. Suleman, H.; Maulud, A.S.; Man, Z. Reconciliation of outliers in CO 2-alkanolamine-H 2 O datasets by robust neural network winsorization. *Neural Comput. Appl.* **2017**, *28*, 2621–2632. [CrossRef]
39. Nyitrai, T.; Virág, M. The effects of handling outliers on the performance of bankruptcy prediction models. *Socio-Econ. Plan. Sci.* **2019**, *67*, 34–42. [CrossRef]
40. Chen, Z. *Trimmed and Winsorized M- and Z-Estimators, with Applications to Robust Estimation in Neural Network Models*; The University of Texas at Dallas: Richardson, TX, USA, 2000.
41. Rasmussen, C.E. Gaussian processes in machine learning. In *Summer School on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 63–71.
42. Snelson, E.; Ghahramani, Z. Local and global sparse Gaussian process approximations. In Proceedings of the Artificial Intelligence and Statistics, San Juan, Puerto Rico, 21–24 March 2007; pp. 524–531.
43. Lawrence, N.; Seeger, M.; Herbrich, R. Fast sparse Gaussian process methods: The informative vector machine. In Proceedings of the 16th Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 9–11 December 2003; pp. 609–616.
44. Tran, D.; Ranganath, R.; Blei, D.M. The variational Gaussian process. *arXiv* **2015**, arXiv:1511.06499.
45. Hensman, J.; Fusi, N.; Lawrence, N.D. Gaussian processes for big data. *arXiv* **2013**, arXiv:1309.6835.
46. Titsias, M.; Lawrence, N.D. Bayesian Gaussian process latent variable model. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 844–851.
47. Hoffman, M.D.; Blei, D.M.; Wang, C.; Paisley, J. Stochastic variational inference. *J. Mach. Learn. Res.* **2013**, *14*, 303–1347.
48. Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In Proceedings of the Artificial Intelligence and Statistics, Clearwater, FL, USA, 16–18 April 2009; pp. 567–574.
49. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
50. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
51. Gal, Y.; Hron, J.; Kendall, A. Concrete dropout. *arXiv* **2017**, arXiv:1705.07832.
52. Damianou, A.; Lawrence, N.D. Deep gaussian processes. In Proceedings of the Artificial Intelligence and Statistics, Scottsdale, AZ, USA, 29 April–1 May 2013; pp. 207–215.
53. Hanson, S.; Pratt, L. Comparing biases for minimal network construction with back-propagation. *Adv. Neural Inf. Process. Syst.* **1988**, *1*, 177–185.
54. Kang, G.; Li, J.; Tao, D. Shakeout: A new regularized deep neural network training scheme. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
55. Li, Y.; Liu, F. Whiteout: Gaussian adaptive noise regularization in deep neural networks. *arXiv* **2016**, arXiv:1612.01490.
56. Goodfellow, I.; Warde-Farley, D.; Mirza, M.; Courville, A.; Bengio, Y. Maxout networks. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1319–1327.
57. Graves, A. Practical variational inference for neural networks. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 2348–2356.
58. Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; Fergus, R. Regularization of neural networks using dropconnect. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1058–1066.
59. Kingma, D.P.; Salimans, T.; Welling, M. Variational dropout and the local reparameterization trick. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2575–2583.
60. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural network. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1613–1622.
61. Friston, K.; Mattout, J.; Trujillo-Barreto, N.; Ashburner, J.; Penny, W. Variational free energy and the Laplace approximation. *Neuroimage* **2007**, *34*, 220–234. [CrossRef]
62. Jaakkola, T.S.; Jordan, M.I. Bayesian parameter estimation via variational methods. *Stat. Comput.* **2000**, *10*, 25–37. [CrossRef]
63. Yedidia, J.S.; Freeman, W.T.; Weiss, Y. Generalized belief propagation. In Proceedings of the Neural Information Processing Systems 2000 (NIPS 2000), Denver, CO, USA, 1 January 2000; Volume 13, pp. 689–695.
64. Neal, R.M.; Hinton, G.E. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 355–368.

65. Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R.Y.; Chen, X.; Asfour, T.; Abbeel, P.; Andrychowicz, M. Parameter space noise for exploration. *arXiv* **2017**, arXiv:1706.01905.
66. Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv* **2017**, arXiv:1703.03864.
67. Fortunato, M.; Azar, M.G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; et al. Noisy networks for exploration. *arXiv* **2017**, arXiv:1706.10295.
68. Wen, Y.; Vicol, P.; Ba, J.; Tran, D.; Grosse, R. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv* **2018**, arXiv:1803.04386.
69. Sun, Y.; Wierstra, D.; Schaul, T.; Schmidhuber, J. Efficient natural evolution strategies. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, Portland, OR, USA, 7–11 July 2009; pp. 539–546.
70. Wierstra, D.; Schaul, T.; Glasmachers, T.; Sun, Y.; Peters, J.; Schmidhuber, J. Natural evolution strategies. *J. Mach. Learn. Res.* **2014**, *15*, 949–980.
71. Bishop, C.M. Mixture Density Networks. 1994. Available online: https://publications.aston.ac.uk/id/eprint/373/1/NCRG_94_004.pdf (accessed on 1 September 2021).
72. Bishop, C.M. Pattern recognition. *Mach. Learn.* **2006**, *128*, 272–276.
73. Riha, S.J.; Wilks, D.S.; Simoens, P. Impact of temperature and precipitation variability on crop model predictions. *Clim. Chang.* **1996**, *32*, 293–311. [CrossRef]
74. McCown, R.L.; Hammer, G.L.; Hargreaves, J.N.G.; Holzworth, D.P.; Freebairn, D.M. APSIM: A novel software system for model development, model testing and simulation in agricultural systems research. *Agric. Syst.* **1996**, *50*, 255–272. [CrossRef]
75. Jones, P.G.; Thornton, P.K. The potential impacts of climate change on maize production in Africa and Latin America in 2055. *Glob. Environ. Chang.* **2003**, *13*, 51–59. [CrossRef]
76. McDermid, S.P.; Ruane, A.C.; Rosenzweig, C.; Hudson, N.I.; Morales, M.D.; Agalawatte, P.; Ahmad, S.; Ahuja, L.; Amien, I.; Anapalli, S.S.; et al. The AgMIP coordinated climate-crop modeling project (C3MP): Methods and protocols. In *Handbook of Climate Change and Agroecosystems: The Agricultural Model Intercomparison and Improvement Project Integrated Crop and Economic Assessments, Part 1*; World Scientific: Singapore, 2015; pp. 191–220.
77. Liu, X.; Chen, F.; Barlage, M.; Zhou, G.; Niyogi, D. Noah-MP-Crop: Introducing dynamic crop growth in the Noah-MP land surface model. *J. Geophys. Res. Atmos.* **2016**, *121*, 13–953. [CrossRef]
78. Lobell, D.B.; Burke, M.B. On the use of statistical models to predict crop yield responses to climate change. *Agric. For. Meteorol.* **2010**, *150*, 1443–1452. [CrossRef]
79. Schlenker, W.; Roberts, M.J. Nonlinear temperature effects indicate severe damages to US crop yields under climate change. *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 15594–15598. [CrossRef] [PubMed]
80. Sheehy, J.E.; Mitchell, P.L.; Ferrer, A.B. Decline in rice grain yields with temperature: Models and correlations can give different estimates. *Field Crops Res.* **2006**, *98*, 151–156. [CrossRef]
81. Lin, T.; Zhong, R.; Wang, Y.; Xu, J.; Jiang, H.; Xu, J.; Ying, Y.; Rodriguez, L.; Ting, K.; Li, H. DeepCropNet: A deep spatial-temporal learning framework for county-level corn yield estimation. *Environ. Res. Lett.* **2020**, *15*, 034016. [CrossRef]
82. Ruß, G.; Kruse, R.; Schneider, M.; Wagner, P. Data mining with neural networks for wheat yield prediction. In Proceedings of the Industrial Conference on Data Mining, Leipzig, Germany, 16–18 July 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 47–56.
83. Baral, S.; Tripathy, A.K.; Bijayasingh, P. Yield prediction using artificial neural networks. In Proceedings of the International Conference on Advances in Communication, Network, and Computing, Bangalore, India, 10–11 March 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 315–317.
84. Mkhabela, M.; Bullock, P.; Raj, S.; Wang, S.; Yang, Y. Crop yield forecasting on the Canadian Prairies using MODIS NDVI data. *Agric. For. Meteorol.* **2011**, *151*, 385–393. [CrossRef]
85. Fernandes, J.L.; Ebecken, N.F.F.; Esquerdo, J.C.D.M. Sugarcane yield prediction in Brazil using NDVI time series and neural networks ensemble. *Int. J. Remote Sens.* **2017**, *38*, 4631–4644. [CrossRef]
86. Pantazi, X.E.; Moshou, D.; Mouazen, A.M.; Kuang, B.; Alexandridis, T. Application of supervised self organising models for wheat yield prediction. In Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Rhodos, Greece, 19–21 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 556–565.
87. Rahman, M.M.; Haq, N.; Rahman, R.M. Machine learning facilitated rice prediction in Bangladesh. In Proceedings of the 2014 Annual Global Online Conference on Information and Computer Technology, Louisville, KY, USA, 3–5 December 2014; pp. 1–4.
88. Ahamed, A.M.S.; Mahmood, N.T.; Hossain, N.; Kabir, M.T.; Das, K.; Rahman, F.; Rahman, R.M. Applying data mining techniques to predict annual yield of major crops and recommend planting different crops in different districts in Bangladesh. In Proceedings of the 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Takamatsu, Japan, 1–3 June 2015; pp. 1–6.
89. Crane-Droesch, A. Machine learning methods for crop yield prediction and climate change impact assessment in agriculture. *Environ. Res. Lett.* **2018**, *13*, 114003. [CrossRef]
90. Matsumura, K.; Gaitan, C.F.; Sugimoto, K.; Cannon, A.J.; Hsieh, W.W. Maize yield forecasting by linear regression and artificial neural networks in Jilin, China. *J. Agric. Sci.* **2015**, *153*, 399–410. [CrossRef]

91. Kouadio, L.; Deo, R.C.; Byrareddy, V.; Adamowski, J.F.; Mushtaq, S. Artificial intelligence approach for the prediction of Robusta coffee yield using soil fertility properties. *Comput. Electron. Agric.* **2018**, *155*, 324–338. [CrossRef]
92. Goldstein, A.; Fink, L.; Meitin, A.; Bohadana, S.; Lutenberg, O.; Ravid, G. Applying machine learning on sensor data for irrigation recommendations: Revealing the agronomist's tacit knowledge. *Precis. Agric.* **2018**, *19*, 421–444. [CrossRef]
93. Zhong, H.; Li, X.; Lobell, D.; Ermon, S.; Brandeau, M.L. Hierarchical modeling of seed variety yields and decision making for future planting plans. *Environ. Syst. Decis.* **2018**, *38*, 458–470. [CrossRef]
94. Romero, J.R.; Roncallo, P.F.; Akkiraju, P.C.; Ponzoni, I.; Echenique, V.C.; Carballido, J.A. Using classification algorithms for predicting durum wheat yield in the province of Buenos Aires. *Comput. Electron. Agric.* **2013**, *96*, 173–179. [CrossRef]
95. Everingham, Y.; Sexton, J.; Skocaj, D.; Inman-Bamber, G. Accurate prediction of sugarcane yield using a random forest algorithm. *Agron. Sustain. Dev.* **2016**, *36*, 27. [CrossRef]
96. Shekoofa, A.; Emam, Y.; Shekoufa, N.; Ebrahimi, M.; Ebrahimie, E. Determining the most important physiological and agronomic traits contributing to maize grain yield through machine learning algorithms: A new avenue in intelligent agriculture. *PLoS ONE* **2014**, *9*, e97288. [CrossRef]
97. Jeong, J.H.; Resop, J.P.; Mueller, N.D.; Fleisher, D.H.; Yun, K.; Butler, E.E.; Timlin, D.J.; Shim, K.M.; Gerber, J.S.; Reddy, V.R.; et al. Random forests for global and regional crop yield predictions. *PLoS ONE* **2016**, *11*, e0156571. [CrossRef] [PubMed]
98. Ruß, G.; Kruse, R. Regression models for spatial data: An example from precision agriculture. In Proceedings of the Industrial Conference on Data Mining, Berlin, Germany, 12–14 July 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 450–463.
99. González Sánchez, A.; Frausto Solís, J.; Ojeda Bustamante, W. Predictive ability of machine learning methods for massive crop yield prediction. *Span. J. Agric. Res.* **2014**, *12*, 313–328. [CrossRef]
100. Gandhi, N.; Armstrong, L.J.; Petkar, O.; Tripathy, A.K. Rice crop yield prediction in India using support vector machines. In Proceedings of the 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), Khon Kaen, Thailand, 13–15 July 2016; pp. 1–5.
101. van Klompenburg, T.; Kassahun, A.; Catal, C. Crop yield prediction using machine learning: A systematic literature review. *Comput. Electron. Agric.* **2020**, *177*, 105709. [CrossRef]
102. Liakos, K.G.; Busato, P.; Moshou, D.; Pearson, S.; Bochtis, D. Machine learning in agriculture: A review. *Sensors* **2018**, *18*, 2674. [CrossRef]
103. Ray, D.K.; Gerber, J.S.; MacDonald, G.K.; West, P.C. Climate variation explains a third of global crop yield variability. *Nat. Commun.* **2015**, *6*, 1–9. [CrossRef] [PubMed]
104. Ray, D.K.; West, P.C.; Clark, M.; Gerber, J.S.; Prishchepov, A.V.; Chatterjee, S. Climate change has likely already affected global food production. *PLoS ONE* **2019**, *14*, e0217148. [CrossRef] [PubMed]
105. PSL. CPC Global Temperature Data Provided by the NOAA/OAR/ESRL PSL, Boulder, Colorado, USA, from Their Web Site. Available online: https://psl.noaa.gov/ (accessed on 1 September 2021).
106. USDA Maize Yield Data. Available online: https://www.nass.usda.gov/Statistics_by_Subject/index.php?sector=CROPS (accessed on 1 September 2021).
107. Pace, R.K.; Barry, R. Sparse spatial autoregressions. *Stat. Probab. Lett.* **1997**, *33*, 291–297. [CrossRef]
108. Cortez, P.; Morais, A.d.J.R. *A Data Mining Approach to Predict Forest Fires Using Meteorological Data*. In Proceedings of the 13th EPIA 2007—Portuguese Conference on Artificial Intelligence, Guimaraes, Portugal, 3–7 December 2007; pp. 512–523.
109. Keeling, C.D.; Bacastow, R.B.; Bainbridge, A.E.; Ekdahl, C.A., Jr.; Guenther, P.R.; Waterman, L.S.; Chin, J.F. Atmospheric carbon dioxide variations at Mauna Loa observatory, Hawaii. *Tellus* **1976**, *28*, 538–551.
110. Garnett, M.J.; Edelman, E.J.; Heidorn, S.J.; Greenman, C.D.; Dastur, A.; Lau, K.W.; Greninger, P.; Thompson, I.R.; Luo, X.; Soares, J.; et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature* **2012**, *483*, 570–575. [CrossRef]
111. Iorio, F.; Knijnenburg, T.A.; Vis, D.J.; Bignell, G.R.; Menden, M.P.; Schubert, M.; Aben, N.; Gonçalves, E.; Barthorpe, S.; Lightfoot, H.; et al. A landscape of pharmacogenomic interactions in cancer. *Cell* **2016**, *166*, 740–754.
112. Frazier, P.I. A tutorial on bayesian optimization. *arXiv* **2018**, arXiv:1807.02811.
113. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B (Methodol.)* **1996**, *58*, 267–288. [CrossRef]
114. Duvenaud, D.; Rippel, O.; Adams, R.; Ghahramani, Z. Avoiding pathologies in very deep networks. In Proceedings of the Artificial Intelligence and Statistics, Reykjavik, Iceland, 22–25 April 2014; pp. 202–210.
115. Lu, C.K.; Yang, S.C.H.; Hao, X.; Shafto, P. Interpretable deep Gaussian processes with moments. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Palermo, Italy, 26–28 August 2020; pp. 613–623.
116. Lien, D.; Balakrishnan, N. On regression analysis with data cleaning via trimming, winsorization, and dichotomization. *Commun. Stat. Comput.* **2005**, *34*, 839–849. [CrossRef]
117. Tarasov, I.E. A Mathematical Method for Determining the Parameters of Functional Dependencies Using Multiscale Probability Distribution Functions. *Mathematics* **2021**, *9*, 1085. [CrossRef]
118. Postels, J.; Ferroni, F.; Coskun, H.; Navab, N.; Tombari, F. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 2931–2940.

119. Hüllermeier, E.; Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Mach. Learn.* **2021**, *110*, 457–506.
120. Fox, C.R.; Ülkümen, G. Distinguishing two dimensions of uncertainty. In *Essays in Judgment and Decision Making*; Brun, W., Kirkebøen, G., Montgomery, H., Eds.; Universitetsforlaget: Oslo, Norway, 2011.