Can Bayesian Neural Networks Explicitly Model Input Uncertainty?

Matias Valdenegro-Toro[a] and Marco Zullich[b]
Department of Artificial Intelligence, University of Groningen, Nijenborgh 9, 9747AG, Groningen, Netherlands

Keywords: Uncertainty Estimation, Input Uncertainty, Feature Uncertainty.

Abstract: Inputs to machine learning models can have associated noise or uncertainties, but they are often ignored and not modelled. It is unknown if Bayesian Neural Networks and their approximations are able to consider uncertainty in their inputs. In this paper we build a two input Bayesian Neural Network (mean and standard deviation) and evaluate its capabilities for input uncertainty estimation across different methods like Ensembles, MC-Dropout, and Flipout. Our results indicate that only some uncertainty estimation methods for approximate Bayesian NNs can model input uncertainty, in particular Ensembles and Flipout.

1 INTRODUCTION

In the last two decades, Neural Networks (NNs) have become state-of-the-art applications in many different domains, such as computer vision and natural language processing. Despite this, these models are known as being notoriously bad at modelling uncertainty, especially when considering the *frequentist* setting (Valdenegro-Toro, 2021a), in which fixed parameters are *trained* to minimize a loss function. Indeed, while NNs for regression lack a direct way to estimate uncertainty, Deep NNs for classification are often found to be extremely overconfident in their predictions (Guo et al., 2017), even when running inference with random data (Nguyen et al., 2015). Bayesian Neural Networks (BNNs), which consider parameters as probability distributions, provide a natural way to produce uncertainty estimates, both in the regression and in the classification setting (Papadopoulos et al., 2001). They have been proven to be substantially better at producing more reliable uncertainty estimates, albeit the quality depends on the techniques which are used to approximate these models (Ovadia et al., 2019). A model whose uncertainty estimates are reliable is also called *calibrated*, and one of the main metrics for calibration is called the Expected Calibration Error (ECE) (Naeini et al., 2015).

Uncertainty, in the context of Machine Learning, is split in two categories (Hüllermeier and Waegeman, 2021): (a) *epistemic* or *model* uncertainty, and (b) *aleatoric* or *data* uncertainty—here also called *input* uncertainty. These two types of uncertainty are usually implicitly modelled together in a single concept, called *predictive* uncertainty, and the process of recovering the epistemic and aleatoric components is called uncertainty *disentanglement* (Valdenegro-Toro and Mori, 2022). An effective modeling of aleatoric and epistemic uncertainty by a machine learning model is crucial whenever this model needs to (a) be deployed in-the-wild, or (b) be used (in assisting) for decision-making in safety-critical situations. In these cases, it is paramount that model is well calibrated: if it is presented with anomalous or unknown data—for

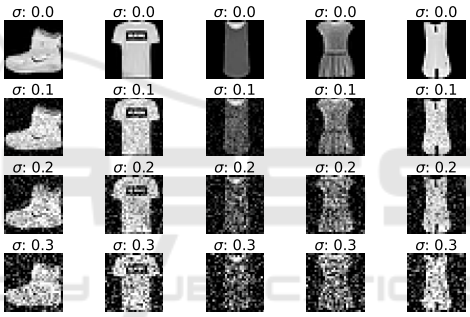


Figure 1: Sample of data from the Fashion-MNIST dataset with Gaussian noise with increasing standard deviation (σ in the figure) added. The first row ($\sigma = 0.0$) represents the original, unperturbed data. Natural data are often captured by means of digital sensors, which are prone to be noisy and can sporadically fail. Training NNs which can effectively model input uncertainty, especially when the noise is anomalously high, is important in having reliable predictions, which can be discarded whenever the predictive uncertainty of the model is too high.

[a] https://orcid.org/0000-0001-5793-9498
[b] https://orcid.org/0000-0002-9920-9095

Valdenegro-Toro, M. and Zullich, M.
Can Bayesian Neural Networks Explicitly Model Input Uncertainty?.
DOI: 10.5220/0013313300003912

In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) - Volume 3: VISAPP*, pages 188-199
ISBN: 978-989-758-728-3; ISSN: 2184-4321

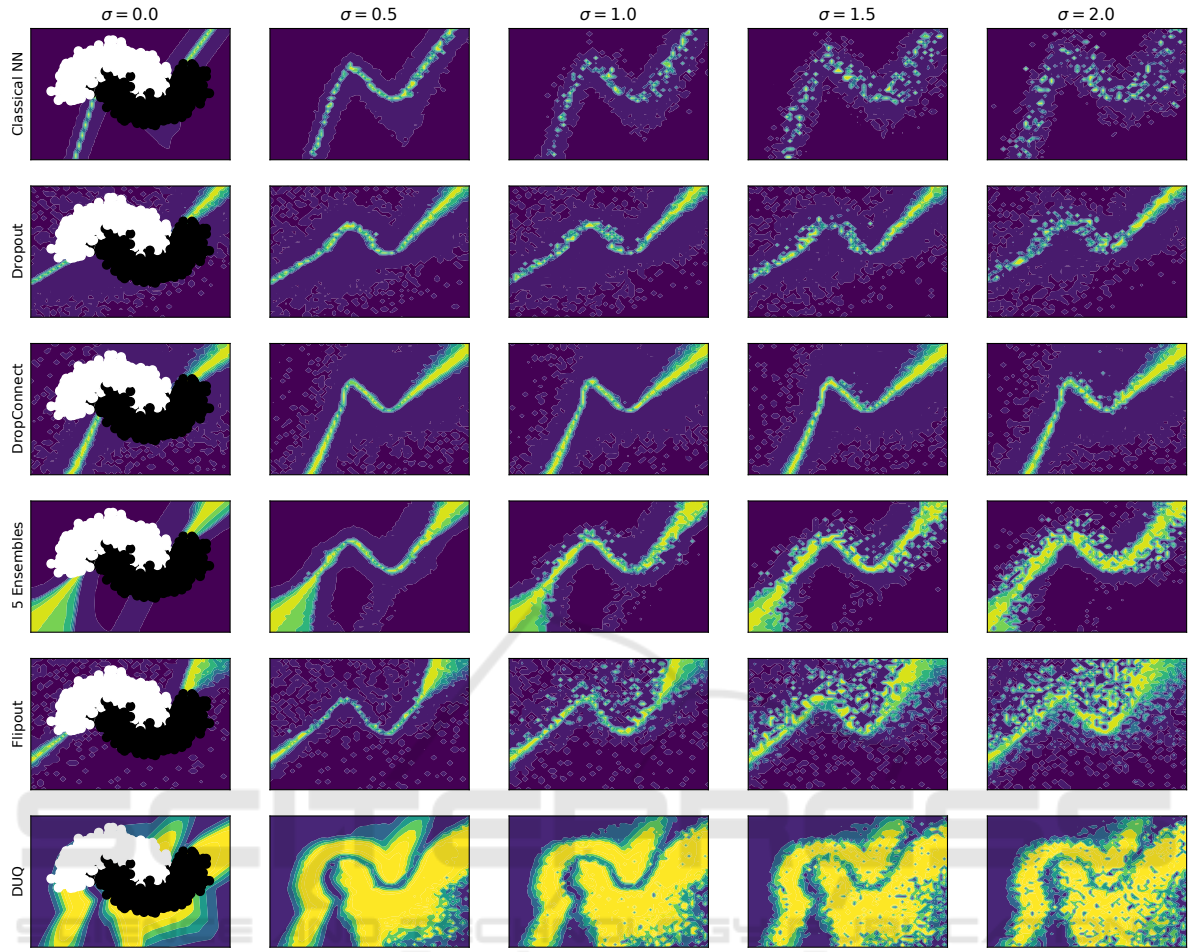


Figure 2: Comparison on the Two Moons dataset with training $\sigma = 0.2$, as the testing standard deviation is varied. Each heatmap indicates predictive entropy (low blue to high yellow) and the first column includes the training data points, With larger test standard deviation, some UQ methods do not significantly change their output uncertainty (DropConnect, Dropout, DUQ), while Flipout and Ensembles do have significant changes, indicating that they are able to model input uncertainty and propagate it to the output.

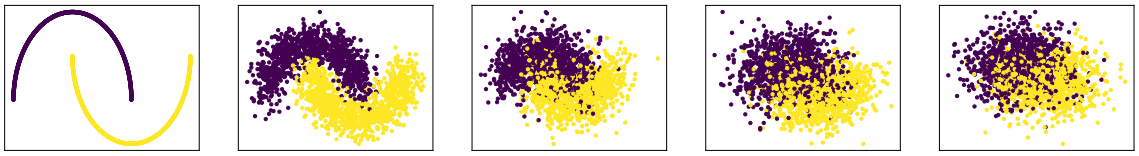
which it effectively behaves randomly—we want this reflected in the prediction uncertainty. In this sense, a highly unconfident prediction can be discarded *a priori* because it has a high chance of being inaccurate.

The digitization of natural data requires capturing it with either manual measurements or sensors, both procedures which are subjects to noise: this represents aleatoric uncertainty; recapturing the data within the same condition several times will lead to different measurements. We can thus summarize each data point as a *mean* data and the corresponding *standard deviation*.

In the present work, instead of letting the NNs implicitly model predictive uncertainty, we provide the input uncertainty as *input*, in addition to the *mean* value of the data. We call these models *two-input NNs*.

We provide our results on two small-scale classification tasks: the Two Moons toy example and the Fashion-MNIST dataset (Xiao et al., 2017). We train classical NNs and five approximate classes of BNNs (MC-Dropout, MC-DropConnect, Ensembles, Flipout, Direct Uncertainty Quantification—DUQ) on these tasks. We observe the behavior of the uncertainty and ECE when different values of noise are injected into the data and conclude that often these models fail to correctly estimate input uncertainty.

The investigation of the quality of predictive uncertainty estimates for machine learning models is a long-studied subject and is usually associated with Bayesian modeling (Roberts, 1965): given the fact that these models output a probability distribution, its deviation can be used as a natural estimate of uncertainty. Deterministic NNs predict point estimates,



(a) $\sigma = 0.0$ (b) $\sigma = 0.2$ (c) $\sigma = 0.4$ (d) $\sigma = 0.6$ (e) $\sigma = 0.8$

Figure 3: The version of the Two Moons dataset (with 1000 data points) used in the present work, the two colors representing the two categories. From left to right, we add an increasingly higher level of zero-mean Gaussian noise. The standard deviation is denoted by σ.

thus they lack a natural expression of uncertainty, except for the case of classification, where the output—after the application of the softmax function—is interpretable as a probability distribution. Initial attempts at computation of probability intervals on the output of NNs include the usage of *two-headed models* which output a mean prediction and the standard deviation (Nix and Weigend, 1994) and early-day BNNs (MacKay, 1992). None of these attempts, though, propose a direct modeling of data uncertainty.

Nonetheless, there are more recent efforts for achieving this. (Wright, 1998) and (Wright, 1999) use the Laplace approximation to train a BNN with input uncertainty, but this is not a modern BNN and it is only tested on simplistic regression settings. (Tzelepis et al., 2017) introduce a variation of Support Vector Machines which include a Gaussian noise formulation for each data point, which is directly taken into consideration in the hinge loss for determining the separating hyperplane. (Rodrigues et al., 2023) introduce the concept of two-input NNs by crafting a simple toy classification problem, showing how, by providing more information as input to the models, their NNs perform better than the regular, "single-input" counterparts. (Hüllermeier, 2014), instead, focuses on producing *fuzzy* loss functions to utilize in a deterministic setting. This allows to incorporate input uncertainty in the empirical risk minimization paradigm. All of these three works limit their investigations to toy problems and, moreover, do not provide insights into the evaluation of uncertainty estimates.

To the best of our knowledge, we are the first to investigate the capability of modern BNNs to explicitly model input uncertainty, by providing an analysis on the quality of the uncertainty that these models produce. Our hypothesis is that models being presented with aleatoric uncertainty as input will not be able to effectively reflect it in the predictive uncertainty, exhibiting high levels of confidence even when the input is anomalously noisy.

The contributions of this work are an evaluation of the capability of BNNs to explicitly model uncertainty in their inputs, we evaluate several uncertainty estimation methods and approximate BNNs, and conclude that only Ensembles are—to a certain extent—reliable when considering explicit uncertainty in its input.

2 EVALUATING BAYESIAN NEURAL NETWORKS AGAINST INPUT UNCERTAINTY

2.1 Datasets

We base our experiments on two datasets, the Two Moons dataset and Fashion-MNIST.

Two Moons. Two Moons is a toy binary classification problem available in the Python library scikit-learn (Kramer and Kramer, 2016). It is composed by a variable number of 2d data points generated in forming two interleaving half circles. Due to the ease of visualization, it is often being used in research on uncertainty estimation for visualizing the capability of the models to produce reliable uncertainty values in and around the domain of the dataset. Notice that, due to its toy nature, this dataset only comes with a training set, i.e., there are no validation or test splits. Some examples of unperturbed and perturbed Two Moons dataset with 1000 data points are visible in Figure 3.

Fashion-MNIST. Fashion-MNIST is a popular benchmark for image classification introduced by (Xiao et al., 2017) as a more challenging version of MNIST (LeCun et al., 1998). It features 70 000 grayscale, 28×28 images of clothing items from 10 different categories. The images come pre-split into a training set of 60 000 and a test set of 10 000 images. A sample of unperturbed and perturbed images from Fashion-MNIST is showcased in Figure 1.

Toy Regression. For a regression setting we use a commonly used sinusoid with variable amplitude and both homoscedatic (ε_2) and heteroscedatic (ε_1) aleatoric uncertainty, defined by:

$$f(x) = x\sin(x) + \varepsilon_1 x + \varepsilon_2 \quad (1)$$

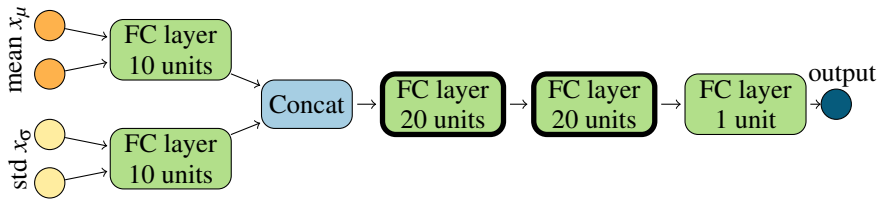


Figure 4: Diagram of the MLP for the Two Moons dataset. The mean and std input pass through two parallel fully-connected ("FC") layers of 10 units, whose output is concatenated. Then, two 20-units fully-connected layers and the final classification layer are applied, which produce the final output. The two 20-units layers (depicted with bold borders) are made Bayesian—depending on the specific technique used.

Where $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0, 0.3)$. We produce 1000 samples for $x \in [0, 10]$ as a training set, and an out-of-distribution dataset is built with 200 samples for $x \in [10, 15]$.

2.2 Predictive Uncertainty in NNs

As previously stated, there is no direct way to compute predictive uncertainty on *standard* deterministic NNs for regression. In the case of c-way classification, instead, the output $f_\theta(x) \doteq \hat{y}$ is a vector of c scalars, each scalar representing the *confidence* that the model assigns to the input x belonging to the corresponding category. If softmax is applied to the output, we can see it as a probability distribution, and we can define two notions of uncertainty:

- Entropy of the distribution (the *flatter* the distribution, the more the model is uncertain)

$$H(\hat{y}) = -\sum_{k=1}^{c} \hat{y}_k \log \hat{y}_k$$

- Maximum of the distribution (the less confident the model is on assigning the model to the category with the maximum value, the more the model is uncertain).

$$\text{Confidence}(\hat{y}) = \max\{\hat{y}_k\} \tag{2}$$
$$\text{Unconfidence} = 1 - \text{Confidence} \tag{3}$$

In the present work, we make use of both definitions of uncertainty. For a regression setting, we use the predictive mean $\mu(x)$ as a prediction and predictive standard deviation $\sigma(x)$ as uncertainty of that prediction.

$$\mu(x) = M^{-1} \sum_i f_{\theta_i}(x) \tag{4}$$
$$\sigma^2(x) = M^{-1} \sum_i [f_{\theta_i}(x) - \mu(x)]^2 \tag{5}$$

Where f_{θ_i} is an stochastic bayesian model or ensemble members (via index i, see Section 2.3) and M is the number of forward passes or ensemble members, we usually use $M = 50$ for stochastic bayesian models.

In addition, the quality of the uncertainty estimates provided by the models can be assessed using *calibration*. The main idea is that, given a data point x, the model confidence should correspond to the accuracy attained on x. By gathering the results on confidence and accuracy on a dataset, the confidence values can be divided in B bins. Then, the mean accuracy on each bin can be computed. Given a bin b, we call confidence_b the reference confidence on the bin; accuracy_b is then the mean accuracy value. Finally, the calibration can be measured by means of the ECE:

$$ECE = \sum_{b=1}^{B} N_b \frac{|\text{confidence}_b - \text{accuracy}_b|}{N}$$

where N is the size of the dataset, and N_b indicates the number of data points belonging to bin b.

2.3 Bayesian Neural Networks

BNNs provide a paradigm shift, in which the parameters of the model are not scalars, but probability distributions. This allows for a more reliable estimate of the predictive uncertainty (Naeini et al., 2015; Ovadia et al., 2019) due to the more noisy nature of the prediction. As in all Bayesian models, the driving principle behind BNNs is the computation of the posterior density $p(\theta|\mathcal{D})$, which is obtained via Bayes' theorem:

$$p(\theta|\mathcal{D}) = \frac{\overbrace{p(\mathcal{D}|\theta)}^{\text{likelihood}} \cdot \overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{marginal likelihood}}}. \tag{6}$$

The goal of Bayesian models is to start from a prior distribution defined on the parameters and updating the knowledge over these parameters by means of the evidence—the likelihood. The updated probability distribution of the parameter is the posterior. The computation of the marginal likelihood (the denomi-

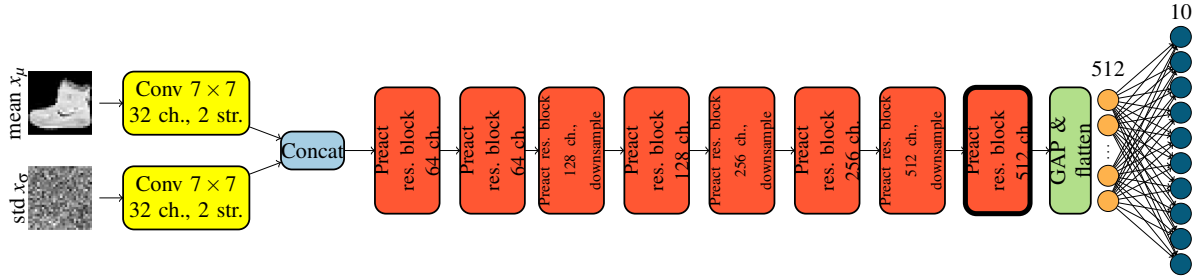


Figure 5: Diagram depicting the two-input Preact-ResNet18 used on Fashion-MNIST. The input mean and standard deviation are passed through two 7×7 convolutions with 32 channels and stride 2, whose outputs are concatenated. The data is then passed sequentially through a series of residual blocks ("Preact res. block") with increasing number of channels. Some blocks operate downsampling of the spatial dimensions. A detailed depiction of the residual blocks is shown in Figure 11. Following the last residual block, global average pooling ("GAP") is applied to return a vector of size 512. This vector is passed through a fully-connected layer which produces the final output of 10 units. The last residual block (depicted with thick borders) can be rendered Bayesian by turning its convolutional layers into the corresponding Bayesian version, depending on the method used.

nator in Equation (6)) is often computationally unfeasible, thus Bayesian Machine Learning often resort to approximations based on variations of Markov-Chain Monte-Carlo methods. However, these methods are still too computationally demanding for BNNs (Blundell et al., 2015), hence a number of techniques for approximating BNNs have been proposed in the last decade. In the present work, we make use of a handful of these.

MC-Dropout. MC-Dropout (Gal and Ghahramani, 2016) is a simple modification of the Dropout algorithm for NN regularization (Hinton et al., 2012). During the training phase, at each forward pass, some intermediate activations are randomly zeroed-out with a given probability value p. During inference, the dropout behavior is turned off. MC-Dropout maintains the dropout behavior active during the inference phase, thus allowing for the model to become stochastic. A probability distribution over the output can hence be obtained by repeatedly running inference on the same data point—a process called *sampling*.

MC-DropConnect. DropConnect (Wan et al., 2013) is a conceptual variation of Dropout: instead of suppressing activations, it acts by randomly zeroing-out some parameters with a given probability value p. As for Dropout, DropConnect is also meant as a regularization technique to be activated during training. MC-DropConnect, analogously to MC-Dropout, allows this method to be active also during inference, hence making the model stochastic.

Direct Uncertainty Quantification. DUQ (van Amersfoort et al., 2020) is a method for creating a deterministic NN which incorporates reliable uncertainty estimates in its prediction. It is designed only for classification tasks. Its main idea is to redefine the final classification layer: instead of a vector of c scalars, the model thus produces c embeddings in the same space $\mathbb{R}^m$. The model is trained to pull the embeddings of the same categories closer to each other: the goal is to produce c clusters corresponding to the classes. The data point x is then assigned to the category whose corresponding cluster centroid is nearest; similarly, uncertainty can be defined as the RBF distance to the nearest cluster centroid μ_k:

$$\text{Uncertainty}_{DUQ} = \max_{k \in \{1,\dots,c\}} \exp\left[\frac{\frac{1}{m}||f_\theta(x) - \mu_k||_2^2}{2\sigma^2}\right], \tag{7}$$

with σ being a hyperparameter. (van Amersfoort et al., 2020) suggest to train DUQ models using gradient penalty (Drucker and Le Cun, 1992), a regularization method which rescales the gradient by a hyperparameter λ.

Flipout. vBayes By Backprop is a Variational Inference–inspired technique introduced by (Blundell et al., 2015). It allows to directly model the parameters of a BNN as Gaussian distributions, while introducing a technique to enable the backpropagation-based training typical of deterministic NNs. It can be seen as a proper Bayesian method, since it directly models the probability distribution of the parameters, which are explicitly given a prior distribution. The authors propose to use, as prior, a mixture of two zero-mean Gaussian with standard deviations σ_1 and σ_2 respectively and a mixture weight π. BayesByBackprop makes use of a variational loss based on the Kullback-Leibler divergence between the approximate posterior learnt by the model and the true posterior. BayesByBackprop is, though, computationally intensive and unstable; (Wen et al., 2018) introduced a scheme, called *Flipout*, to add perturbations to the training procedure, allowing to reduce training time and increasing stability.

Ensembles. Ensembles are groups of non-

Table 1: Hyperparameters used in the implementation and training of the NNs. "FMNIST" is short for Fashion-MNIST and "TM" corresponds to the Two Moons dataset.

	# epochs		Batch size		Other hyperparameters		# samples for inference	
	TM	FMNIST	TM	FMNIST	TM	FMNIST	TM	FMNIST
Deterministic NN	100	15	32	256			—	—
MC-Dropout	100	15	32	256	$p = 0.2$	$p = 0.1$	100	25
MC-DropConnect	100	15	32	256	$p = 0.05$		100	25
Ensembles	100	15	32	256	# components= 5		5	5
DUQ	100	—	32	—	$\sigma = 0.1; \lambda = 0.5$	—		—
Flipout	300	15	32	256	$\sigma_1 = 5; \sigma_2 = 2; \pi = 0.5$		100	25

Bayesian NNs with the same architecture and trained on the same data, but with different random initialization of their parameters. They are not inherently Bayesian—their output is not stochastic—but the fact of having multiple outputs for a single data point allows us to make considerations on the predictive uncertainty. Moreover, it has been shown (Lakshminarayanan et al., 2017) that ensembles are producing uncertainty estimates which are often superior in reliability to other methods here presented.

2.4 Two-Input NNs for Input Uncertainty

In the *deterministic* paradigm for NNs, the input is evaluated one-by-one, i.e., the data is passed one sample at the time without explicitly passing input uncertainty as input to the model. Given the data space R^d, the model is hence seen as a function $f_\theta : \mathrm{R}^d \to \mathcal{Y} \subseteq \mathcal{R}^k$, where k is dependent on the task that the model needs to solve and θ indicates the parameters of the model (which are probability distributions in the case of BNNs). In graphical terms, for both deterministic and Bayesian NNs, the model is represented with an *input layer* with n neurons.

In the present work, instead, inspired by (Rodrigues et al., 2023), we take a different approach and craft a NN architecture, which we call *two-input NN*. As the name suggests, this model has two input channels: (a) the *mean* data x_μ (of dimension d), and (b) the *standard deviation* of the data x_σ, also of dimension d. Thus, a two-input NN is represented as a function $f_\theta : \mathrm{R}^d \times \mathrm{R}^d \to \mathcal{Y}$. This setting allows the NN to directly model input uncertainty. We can see this process as feeding *multiple versions* of the same data to the model, by accounting for the uncertainty—encoded in the standard deviation—which is intrinsic in the process of capturing this data.

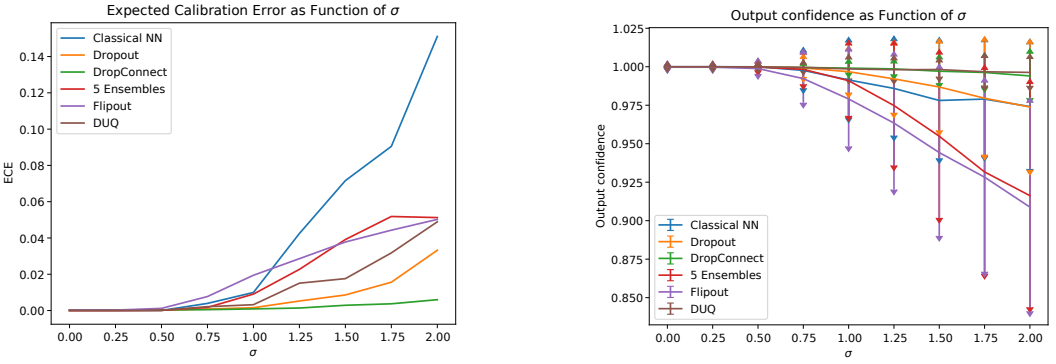
We created three different versions of two-input NNs, one per dataset.

NN for Two Moons. For the Two Moons dataset, we make use of a Multilayer Perceptron (MLP) with four input neurons (2 neurons for x_μ, 2 neurons for x_σ) and three hidden layers of, respectively, 10, 20 and 20 hidden units and ReLU activation and a final, one-unit classification layer. The first hidden layer is duplicated so that the information of the mean and standard deviation flows parallely through them, after which the outputs are concatenated. The two 20-units layers are all Bayesian, which means they implement either MC-Dropout, MC-DropConnect, or Flipout. Ensembles and DUQ use regular MLPs, with DUQ replacing the classification layer with its custom implementation mentioned in Section 2.3. A diagram of the architecture is depicted in Figure 4.

NN for Fashion-MNIST. Inspired by (Harris et al., 2020), for Fashion-MNIST we use a custom PreactResNet18 (He et al., 2016) with two modifications with respect to the original implementation.

1. We turn this model into a two-input NN by modifying the first convolutional layer. Instead of a convolution with 64 output channels, we operate two convolutions in parallel with 32 output channels: the first one operates on x_μ, the second one on x_σ.
2. The second modification instead turns the NN into a BNN: we modify the two convolutional layers of the last residual block by implementing MC-Dropout, MC-DropConnect, or Flipout on them. Notice that the ensemble uses regular convolutions. Due to computational constraints, we don't train a model with DUQ for Fashion-MNIST.

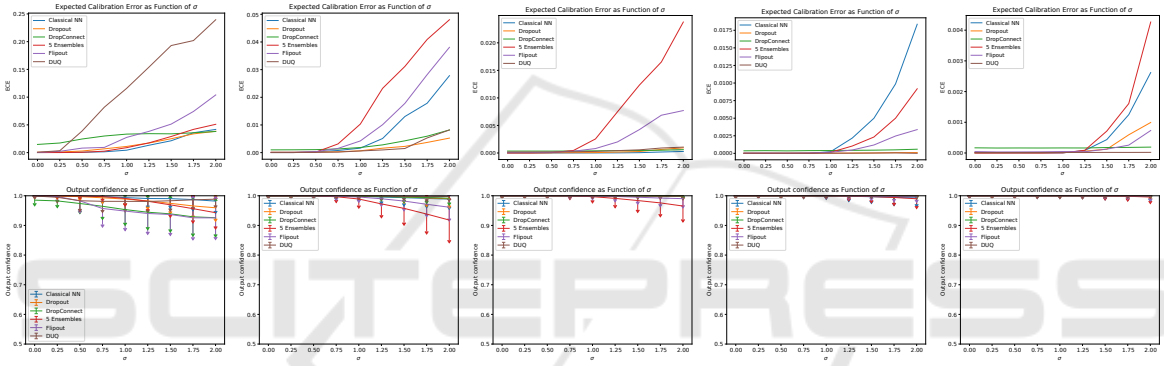
NN for Toy Regression. We use a similar architecture than the NN for two moons. A MLP with four input neurons (2 neurons for x_μ, 2 neurons for x_σ) and four hidden layers of, respectively, 10, 10, 20 and 20 hidden units and ReLU activation and a final, one-unit regression layer. There are separate hidden layers for input mean and input standard deviation, concatenated to connect to the final set of hidden layers.



(a) Expected Calibration Error

(b) Output confidence as function of input uncertainty σ.

Figure 6: Comparison of Expected Calibration Error and Output Confidences on the Two Moons dataset as input uncertainty σ varies. The smallest variation in ECE is with DropConnect while Ensembles and Flipout have the largest decrease of output confidence.



(a) $\sigma = [0.0]$ (b) $\sigma = [0.0, 0.2]$ (c) $\sigma = [0.0, 0.2, 0.4]$ (d) $\sigma = [0.0, 0.2, 0.4, 0.6]$ (e) $\sigma = [0.0, 0.2, 0.4, 0.6, 0.8]$

Figure 7: Results for the Two Moons dataset, setting when training set contains multiple values of sigma. ECE (top) and input/output uncertainty (bottom) are compared. Training on additional σ values increases generalization for testing $\sigma > 1.0$, but makes most models except Ensembles to be insensitive to input uncertainty σ by producing high confidences.

2.5 Experimental Setup

We implement the models mentioned in the previous sections on Python, making use of the Keras library with Tensorflow backend. For the Bayesian layers, we utilize Keras-Uncertainty (Valdenegro-Toro, 2021b). For the dataset Two moons, we run our experiments with all of the approximate BNN methods we introduced. Due to computational reasons, we do not train a NN with DUQ on Fashion-MNIST. In addition, for both datasets, we train a deterministic NN to allow for comparing results with respect to the frequentist setting.

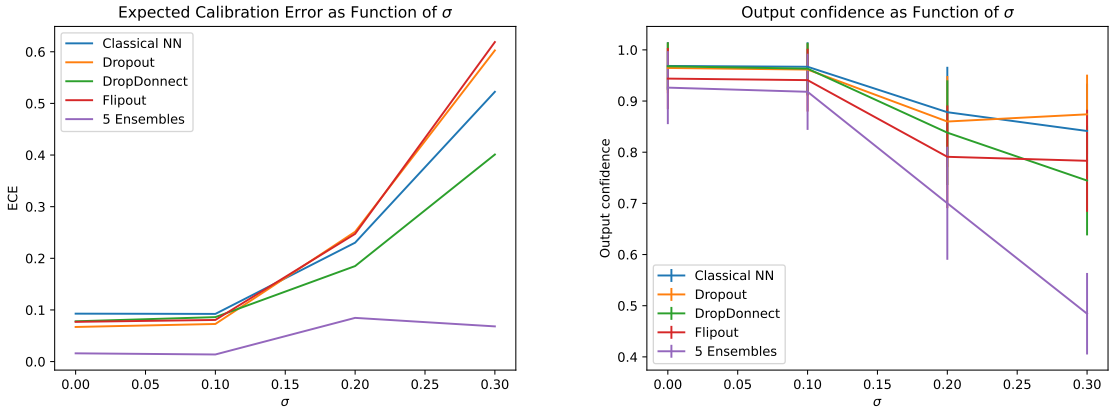
The hyperparameters used for the implementation and training are showcased in Table 1. In addition to what there indicated, we trained all of the models using the Adam optimizer (Kingma and Ba, 2014) with the Keras-default hyperparameters (learning rate of 0.001, β_1 9f 0.9, and β_2 of 0.999).

For what concerns the injection of noise in the images, we simulate the process by passing the original data point in the x_μ input. For the standard deviation input, we pass a structure x_σ with the same size of the x_μ sampled from a normal distribution $x_\sigma \sim N(0, \sigma)$ with a given input noise standard deviation σ. For Two moons, we fix σ at 0.5. For Fashion-MNIST, instead, we first normalize the images in the 0-1 range, then we generate the normal noise with $\sigma = 0.1$.

Evaluation of Uncertainty. For what concerns the evaluation of uncertainty, we test our models with increasing values of Gaussian noise. We then provide two uncertainty-related comparison:

(a) ECE in function of σ. For DUQ, ECE is calculated considering the specific Uncertainty_{DUQ} metric from Equation (7).

(b) Output confidence in function of σ. The confidence is computed as per Equation (2). For DUQ, the confidence is computed as the complemen-



(a) Expected Calibration Error

(b) Output confidence as function of input uncertainty σ.

Figure 8: Comparison of Expected Calibration Error and Output confidences for Fashion MNIST as input uncertainty σ is varied. Note how Ensembles has little variation in calibration error and the largest decrease in confidence with increasing σ.

tary of the normalized uncertainty obtained from Equation (7).

Finally, due to the ease of visualization provided by the 2D nature of Two Moons, we plot the dataset and the uncertainty, calculated in terms of entropy, for a lattice of points around the dataset.

3 EXPERIMENTAL RESULTS

We perform experiments on two datasets. The purpose of these experiments is to evaluate if a Bayesian neural network and other models with uncertainty estimation, can learn to model input uncertainty from two inputs (mean and standard deviation). We test this with a simple setup, we train models with fixed levels of input uncertainty, and then test with increasing levels of input uncertainty.

Our expectation is, if a model properly learns the relationship between input and output uncertainty, then increasing input uncertainty should lead to increases in output uncertainty. We measure output uncertainty via entropy and maximum softmax confidence, and quality of uncertainty via the expected calibration error.

3.1 Two Moons Toy Example

We first evaluate on a toy example, the Two Moons dataset, available in scikit-learn, as it allows for easy control of input uncertainty and to visualize its effects. We perform two experiments, first we train a model with a single σ value during training, and then train a model with multiple σ values.

We first examine the case for a single training uncertainty, we use $\sigma = 0.2$. We plot and compare the output entropy distribution over the input domain, keeping the mean fixed but varying the input uncertainty σ from $\sigma = 0.0$ to $\sigma = 2.0$. These results are presented in Figure 2 and detailed plots for two metrics in Figure 6.

These results show that only Ensembles and Flipout significantly decrease their output confidence as the input uncertainty σ increases, while a classical NN without uncertainty estimation becomes highly miscalibrated, and other methods only produce minor decreases in output confidence. No variations in ECE and output confidence while σ increases indicates that the model might be ignoring the input uncertainty, which is exactly the behavior we wanted to test.

We secondly examine the case for multiple training input uncertainties, using $\sigma \in [0.0, 0.2, 0.4, 0.6, 0.8]$, and testing with $\sigma \in [0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.50, 1.75, 2.0]$ progressively. These results are presented in Figure 7.

These results indicate that Flipout is always miscalibrated relative to other methods, and that all uncertainty estimation methods minus Flipout seem to be insensitive to input uncertainty, always producing high output uncertainty. At the end of the spectrum, training with five different σ values (Figure 7e), most methods have learned to ignore the input uncertainty as output confidence barely varies.

3.2 Fashion-MNIST Image Classification

We then proceed to evaluate our hypothesis on Fashion-MNIST. We train the models on a fixed standard deviation value $\sigma = 0.1$ and report the corre-

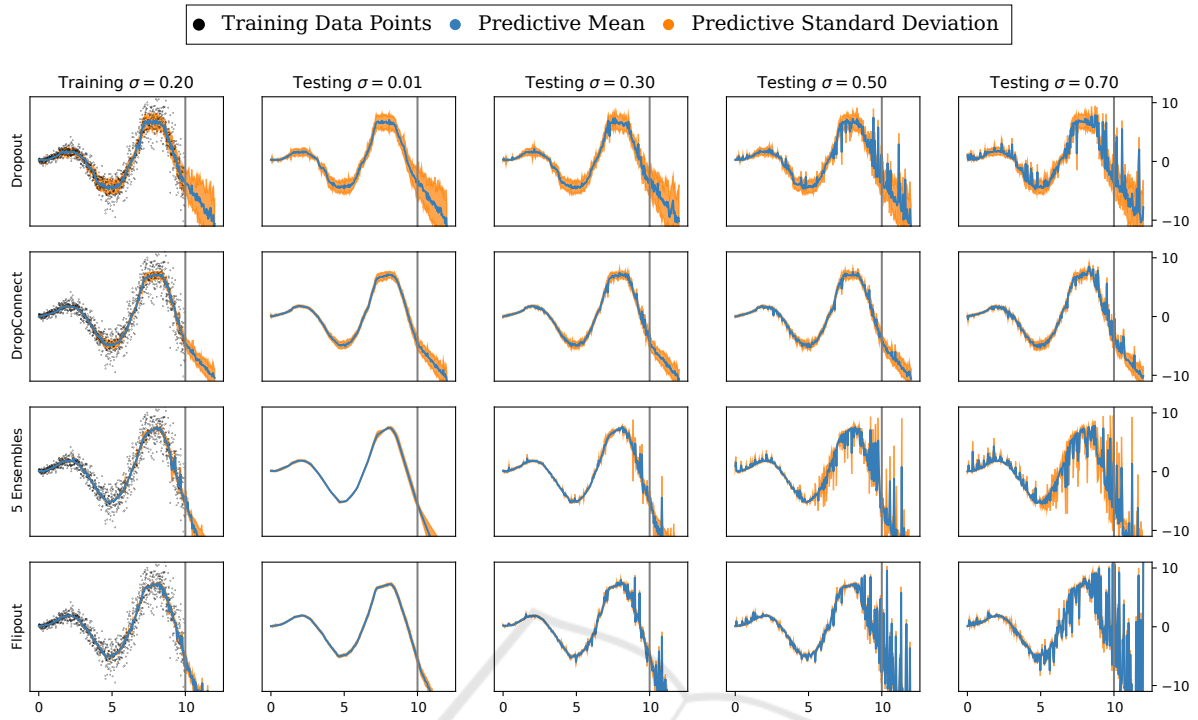


Figure 9: Comparison on a toy regression setting with training $\sigma = 0.2$ and variable testing standard deviation. Consistent with classification results, Ensembles and Dropout have the highest sensitivity to input uncertainty σ.

Table 2: Train– and test–set accuracy attained by our NNs trained on Fashion-MNIST.

Model	Train accuracy	Test accuracy
Deterministic NN	98.6%	88.6%
MC-Dropout	98.7%	88.7%
MC-DropConnect	98.7%	87.7%
Ensemble	98.5%	88.3%
Flipout	95.5%	85.9%

sponding test-set accuracy in Table 2. ECE and output confidence (computed on the test-set) as function of input uncertainty are presented in Figure 8. In this case, we restrict the range of standard deviation for the testing to σ =0.0, 0.1, 0.2 and 0.3 While we train on a single input σ, Ensembles and Flipout decrease their output confidence while input uncertainty σ increases, as expected, while other methods do not. The results are similar to what we observed on the Two Moons dataset, indicating that our results and experiments generalize to a more complex image classification setting.

3.3 Toy Regression Example

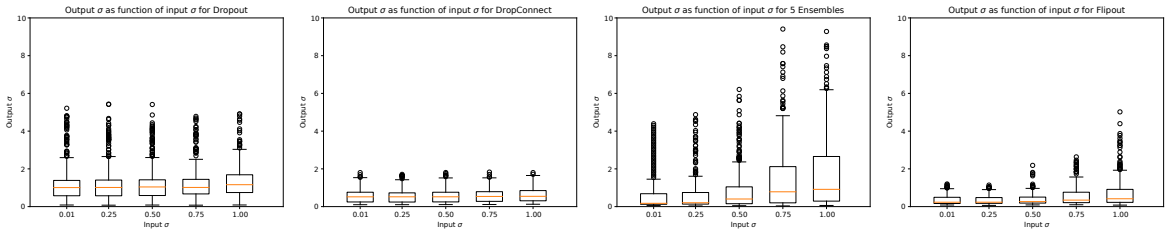
Finally we evaluate results on the toy regression example, these results are shown in Figure 9 in terms of predictions with epistemic uncertainty, and Figure 10 by comparing input and output uncertainties.

Dropout and DropConnect are insensitive to changes in input uncertainty, mostly by producing large uncertainties that do not vary with the input uncertainty, while Ensembles and Flipout do have varying output uncertainty with the input uncertainty, in a monotonic way, Flipout has increasing uncertainty mostly as variations in the predictive mean, while Ensembles has variation mostly on the standard deviation, so we consider the Ensemble results to be more representative of our expectations on how output uncertainty should behave as functions of input uncertainty.

Results on this regression example are consistent with our previous classification results, indicating that the results are general enough across tasks.

4 CONCLUSIONS

In the present work, we investigated the quality of modeling aleatoric uncertainty by classical Neural Networks (NNs) and Bayesian Neural Networks (BNNs) when the input uncertainty is fed directly to the models in addition to the *canonical* input. We proposed a simple setting in which we artificially injected Gaussian noise in two famous benchmark datasets—



(a) Dropout (b) Dropconnect (c) 5 Ensembles (d) Flipout

Figure 10: Comparison of output standard deviation as function of input standard deviation for the toy regression setting via boxplots. Flipout seems barely sensitive to changes in input uncertainty, while Ensembles has the highest reaction to increased input uncertainty.

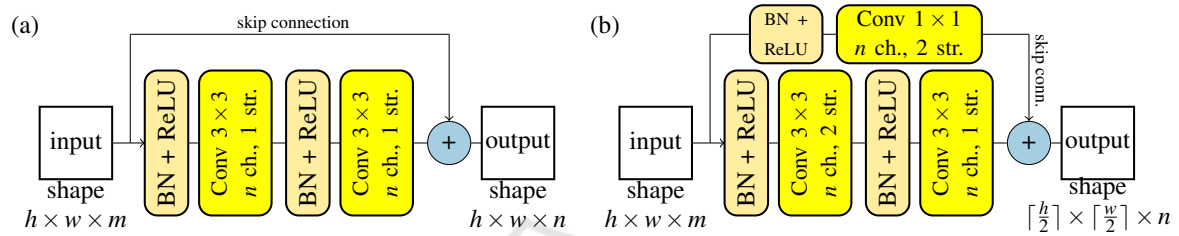


Figure 11: Diagrams depicting the two types of residual block used in the Preact-ResNet18 architecture: (a) standard residual block: a classic residual block with two 3×3 convolutions ("Conv") with a predefined number of output channels n, stride and padding of 1. The residual blocks are preceded by batch normalization ("BN") and ReLU activation. The input and output have the same spatial dimensions h and w. (b) residual block with downsample: it operates a downsampling on the spatial dimension by modifying the first convolution to have a stride of 2 instead of 1. In order to match the spatial dimension after the two convolutions, the skip connection presents a BN followed by ReLU and a 1×1 convolution with stride 2 and padding 1. The output has spatial dimensions which are half the size of the input's.

Two Moons and Fashion-MNIST—often used in uncertainty estimation studies. This simulates a natural environment in which the data are collected by means of sensors, which always exhibit a certain degree of noise. For having the models receive the input uncertainty directly, while this being separated from the data itself, we crafted a set of NN architectures—which we dubbed *two-input* NNs—with two input channels, one for the mean data and the other for the standard deviation corresponding to the added noise. We trained these models on the above mentioned datasets, with a fixed level of noise, using five approximate BNN techniques: MC-Dropout, MC-DropConnect, Flipout, Ensembles, and Direct Uncertainty Quantification (DUQ).

We tested these models with data with none to high-levels of noise and proceeded to compute the output confidence and the Expected Calibration Error (ECE) as a function of noise. Our hypothesis was that, generically, these models would exhibit a certain degree of *insensitivity* to added noise, where their confidence would still be high even when data with high noise—which are effectively out-of-distribution in the settings—are presented to them. The results are pointing in this direction: both on Two Moons and on Fashion-MNIST, the output confidence for most of the methods remains high, while the Ensembles show a pronounced drop in confidence as the input uncertainty increases. On the other hand, the results elicited by ECE are not conclusive, depicting a noisier scenario for what concerns the (mis)calibration of the models.

On Two Moons, where we conducted more extensive analyses, we noticed that, after injecting higher levels of noise in the training process, the models would essentially start ignoring the signal coming from the input uncertainty and always produce very confident predictions and being less miscalibrated. Despite this seemingly being an optimal behavior, in which robustness to noise is enforced, can cause the NNs to fail at recognizing anomalous data, which is one of the reasons for adopting BNNs: by providing more reliable confidence estimates, confidence can be thresholded to filter out outliers and avoid classifying them.

Thus, our analyses suggest that both *deterministic* NNs and BNNs fail, in a certain degree, to model data uncertainty when this one is provided explicitly as input, with ensembles—which are already known in the literature to being particularly powerful then other methods at producing good uncertainty estimates—and, to a lower extent, Flipout, showing the biggest

drop in confidence when presented with very noisy inputs.

Our work, despite being the first analysis on the uncertainty of the NNs when directly modeling input uncertainty, is still quite small scale and mostly observational, and could potentially benefit for more extensive analyses. For instance, larger-scale datasets might be used—although BNNs are notoriously difficult and slow to train on bigger datasets. Also, we could extend the selection of BNN-training schemes to other methods, like the more recent SWAG (Maddox et al., 2019), or Hamiltonian Monte–Carlo (Neal et al., 2011), which is still considered the golden standard for Bayesian modeling, albeit very unfeasible to apply in the large-scale datasets used in modern Deep Learning. Finally, our study could benefit from the addition on the analysis of *uncertainty disentanglement* by the BNNs, to understand to what extent the models are able to integrate aleatoric uncertainty into the input uncertainty.

REFERENCES

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France. PMLR.

Drucker, H. and Le Cun, Y. (1992). Improving generalization performance using double backpropagation. *IEEE transactions on neural networks*, 3(6):991–997.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Harris, E., Marcu, A., Painter, M., Niranjan, M., Prügel-Bennett, A., and Hare, J. (2020). Fmix: Enhancing mixed sample data augmentation. *arXiv preprint arXiv:2002.12047*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hüllermeier, E. (2014). Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization. *International Journal of Approximate Reasoning*, 55(7):1519–1534.

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kramer, O. and Kramer, O. (2016). Scikit-learn. *Machine learning for evolution strategies*, pages 45–53.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

MacKay, D. J. C. (1992). The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32.

Naeini, M. P., Cooper, G., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

Nix, D. A. and Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Papadopoulos, G., Edwards, P. J., and Murray, A. F. (2001). Confidence estimation methods for neural networks: A practical comparison. *IEEE transactions on neural networks*, 12(6):1278–1287.

Roberts, H. V. (1965). Probabilistic prediction. *Journal of the American Statistical Association*, 60(309):50–62.

Rodrigues, N. V., Abramo, L. R., and Hirata, N. S. (2023). The information of attribute uncertainties: what convolutional neural networks can learn about errors in input data. *Machine Learning: Science and Technology*, 4(4):045019.

Tzelepis, C., Mezaris, V., and Patras, I. (2017). Linear maximum margin classifier for learning from uncer-

tain data. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2948–2962.

Valdenegro-Toro, M. (2021a). I find your lack of uncertainty in computer vision disturbing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1263–1272.

Valdenegro-Toro, M. (2021b). Keras uncertainty. `https://github.com/mvaldenegro/keras-uncertainty`. GitHub repository.

Valdenegro-Toro, M. and Mori, D. S. (2022). A deeper look into aleatoric and epistemic uncertainty disentanglement. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1508–1516. IEEE.

van Amersfoort, J., Smith, L., Teh, Y. W., and Gal, Y. (2020). Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA. PMLR.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*.

Wright, W. (1998). Neural network regression with input uncertainty. In *Neural Networks for Signal Processing VIII. Proceedings of the 1998 IEEE Signal Processing Society Workshop (Cat. No. 98TH8378)*, pages 284–293. IEEE.

Wright, W. (1999). Bayesian approach to neural-network modeling with input uncertainty. *IEEE Transactions on Neural Networks*, 10(6):1261–1270.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.