A Survey on Uncertainty Estimation in Deep Learning Classification Systems from a Bayesian Perspective

JOSÉ MENA, Eurecat, Centre Tecnològic de Catalunya and Departament de Matemàtiques i Informàtica, Universitat de Barcelona, Spain
ORIOL PUJOL, Departament de Matemàtiques i Informàtica, Universitat de Barcelona, Spain
JORDI VITRIÀ, Departament de Matemàtiques i Informàtica, Universitat de Barcelona, Spain

Decision-making based on machine learning systems, especially when this decision-making can affect human lives, is a subject of maximum interest in the Machine Learning community. It is, therefore, necessary to equip these systems with a means of estimating uncertainty in the predictions they emit in order to help practitioners make more informed decisions. In the present work, we introduce the topic of uncertainty estimation, and we analyze the peculiarities of such estimation when applied to classification systems. We analyze different methods that have been designed to provide classification systems based on deep learning with mechanisms for measuring the uncertainty of their predictions. We will take a look at how this uncertainty can be modeled and measured using different approaches, as well as practical considerations of different applications of uncertainty. Moreover, we review some of the properties that should be borne in mind when developing such metrics. All in all, the present survey aims to provide a pragmatic overview of the estimation of uncertainty in classification systems that can be very useful for both academic research and deep learning practitioners.

CCS Concepts: • **Computing methodologies → Neural networks**; **Supervised learning by classification**; • **Mathematics of computing → Bayesian networks**.

Additional Key Words and Phrases: Classification, Machine Learning, Deep Learning, Uncertainty

ACM Reference Format:
José Mena, Oriol Pujol, and Jordi Vitrià. 2022. A Survey on Uncertainty Estimation in Deep Learning Classification Systems from a Bayesian Perspective. 1, 1 (February 2022), 36 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine learning (ML) is currently present in all kinds of applications and areas. Object recognition, automatic captioning, and machine translation represent just some of the multiple fields in which machine learning, and especially Deep Learning (DL), is imposing itself in the service of competitive business. In some areas of application, such as autonomous driving or automated patient diagnosis support systems, the level of performance required is very high. Failures in prediction can result in severe economic losses, or even the loss of human life. Hence the need for ways of managing the risks that automatic decision-making entails, and for these types of applications in particular.

Authors' addresses: José Mena, Eurecat, Centre Tecnològic de Catalunya, 72, Bilbao Street, Barcelona, Departament de Matemàtiques i Informàtica, Universitat de Barcelona, 585, Gran Via de les Corts Catalanes, Barcelona, Spain, jose.mena@eurecat.org; Oriol Pujol, Departament de Matemàtiques i Informàtica, Universitat de Barcelona, 585, Gran Via de les Corts Catalanes, Barcelona, Spain, oriol_pujol@ub.edu; Jordi Vitrià, Departament de Matemàtiques i Informàtica, Universitat de Barcelona, 585, Gran Via de les Corts Catalanes, Barcelona, Spain, jordi.vitria@ub.edu.

Managing this risk is especially pertinent when applying Deep Learning systems, a subfield of Machine Learning. Deep Learning is based on the use of rich architectures of Artificial Neural Networks (ANN). One of its main differences from traditional Machine Learning systems is the assumption that these ANNs can capture meaningful features of the input data and tailor them to the learning task. By delegating the feature engineering in these models, DL makes the design of classification systems simpler.

However, this delegating of the feature engineering in the model, and the high number of model parameters present in many modern DL architectures, make these systems hard to interpret. This lack of interpretability can be overcome if we accompany the predictions obtained with a metric of their uncertainty that allows us to manage the risk of their use in decision-making. However, the notion of uncertainty is not monolithic, and does not even has an agreed definition due to the fact that it is present in every phase of the machine learning process. Sources of uncertainty can be found in data acquisition and pre-processing, in the design of the model, the selection stage, and even the training process. This gives rise to many different definitions of uncertainty, depending on which specific aspects researchers and practitioners are focusing on.

According to the machine learning literature, Gal [28], one common way of understanding uncertainty is to rely on its sources. In this case, we can consider aleatoric uncertainty - which is associated with the uncertainty inherent in the data - and epistemic uncertainty - which is related to the confidence of the model. Another approach, suggested in [88], is to categorize it into four types: randomness - a kind of objective uncertainty related to random variables; fuzziness - a kind of cognitive uncertainty due to the absence of strict or precise boundaries of concepts; roughness - representing the degree to which knowledge is accurate; and non-specificity or ambiguity - which results from choosing one from two or more unclear objects. Additionally, we can consider other different types, depending on whether the uncertainty can be reduced or not. As we shall see, it is not always possible to maintain these sharp divisions, as the same model can suffer from different types of uncertainty during its lifespan.

In view of the above, we can state that uncertainty is a complex concept that needs to be **represented, measured, and applied**. The goal of the present work is to survey the different approaches present in the literature for each of these three stages employed in **classification systems**. Although the procedure followed to estimate uncertainty in deep learning classification models does not fundamentally differ from that used for traditional classification models, it does have some specific features. In deep learning, the most widely used loss function is softmax cross-entropy, which was never very popular in traditional models. This function operates on the unscaled output of earlier layers (logits), meaning that the relative scale for understanding the units is linear. Such an approach provides a direct probabilistic interpretation in terms of class scores, which serve as a basis for defining aleatoric uncertainty measures. Moreover, the flexibility inherent in neural networks can be used to enrich classification models by adding some components to compute these uncertainty measures. For example, additional layers can be used to apply hierarchical Bayesian models for uncertainty estimation. Regarding epistemic uncertainty measures, the classical approach is to consider each of the model parameters as a set of parameter estimates that define a random variable rather than a point estimate. In this case, the challenge relies on the number of parameters to be estimated, which can be in the order of millions.

The added value of this paper is three-fold. First, unlike Gal [28], who focused on the estimation of uncertainty in regression problems, we deal with classification, surveying a range of approaches, from the first ones based on Dropout to more novel techniques such as modeling continuous distributions on the simplex. Second, a wide variety of methods are presented by means of a unified Bayesian perspective that facilitates their understanding. And last, we present not only the most recent advances in estimation techniques, but also some considerations regarding their properties and use in practical scenarios.

The following two sections of the article formalize the deep learning classification scenario, which is the focus of this survey, while also reviewing the definition of uncertainty.

Section 4 surveys how uncertainty is **represented** in deep learning classification systems. In most cases, these systems produce a result that comes in the form of a probability distribution over a set of classes. One of the most basic ways of representing uncertainty in a system's predictions is to rely on these probabilities to decide whether or not to trust the outcome of the system. However, as the article shows, these probabilities can lead to errors, since they may not be well calibrated, their interpretation may not be intuitive, or, worse still, they may be falsely considered to be safe predictions.

Instead of relying on these point estimates, uncertainty estimation methods may generate uncertainty metrics from an approximation of the posterior distribution of the classifier output. We shall see how various works have proposed different approaches to modeling this posterior distribution by using different probability distributions and focusing on different terms for its definition.

After reviewing different ways of estimating the posterior distribution of the classifier, in Section 5 we present different ways of **measuring** uncertainty. This is a necessary step if we are to turn the concept of uncertainty into an actionable value that allows risk to be managed in the classification system. In this section, we will see how these systems present additional difficulty when trying to extract a single metric for uncertainty. This is due to the fact that classification systems usually return multiple values, with their corresponding uncertainties for each category. Thus, there is a need for mechanisms that combine these multiple outputs into a single value. As our review has shown, this additional complexity leads to multiple ways of summarizing uncertainty, contrasting with uncertainty metrics in regression systems, which usually output a single value.

Section 6 introduces **applications** that illustrate the use of uncertainty in different aspects related to classification systems.

Finally, Section 7 provides a summary of this article and describes a set of challenges facing uncertainty estimation methods. The aim of this is to provide the reader with some criteria and good practices and help practitioners choose the method that best fits their problem when including uncertainty in the design of a classification system.

2 PROBLEM FORMULATION

Before addressing the concept of uncertainty, we first introduce the general problem setting that will serve as a guide for the discourse in the forthcoming sections.

Given a training dataset, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}, i = 1 \dots N, \mathcal{D} \in X \times Y$, where $y \in \{1, \dots, C\}$ with C the number of categories, and $\mathbf{x}_i \in R^d$, and a new sample of data points $\mathbf{x}^*$, we want to predict their corresponding labels y^*. Our focus is on discriminative classification models, where the goal is to capture the distribution of the target outputs given the input data. Thus, from the observed data X, Y and $\mathbf{x}^*$, our goal is to compute the following conditional distribution:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) \equiv p(y^*|\mathbf{x}^*, X, Y) \tag{1}$$

In order to estimate this output distribution, we can use an intermediate model with parameters $\mathbf{W}$. By using this model with parameters $\mathbf{W}$, we can then marginalize as follows,

$$p(y^*|\mathbf{x}^*, X, Y) = \int_{\mathbf{W}} p(y^*|\mathbf{W}, \mathbf{x}^*, X, Y) p(\mathbf{W}|\mathbf{x}^*, X, Y) d\mathbf{W} \tag{2}$$

At this point, we can consider two assumptions:

- Assumption 1: When learning the parameter $\mathbf{W}$, we do not have access to the test data $\mathbf{x}^*$, so we approximate:
$$p(\mathbf{W}|\mathbf{x}^*, X, Y) \approx p(\mathbf{W}|X, Y) \tag{3}$$
- Assumption 2: We assume that the model parameterized by $\mathbf{W}$ is able to capture the true distributions over X, Y, so we can approximate the conditional output distribution as,
$$p(y^*|\mathbf{W}, \mathbf{x}^*, X, Y) \approx p(y^*|\mathbf{W}, \mathbf{x}^*) \tag{4}$$

Bearing these assumptions in mind, we end up with:
$$p(y^*|\mathbf{x}^*, X, Y) \approx \int_{\mathbf{W}} p(y^*|\mathbf{W}, \mathbf{x}^*)p(\mathbf{W}|X, Y)d\mathbf{W} \tag{5}$$

At this point, the evaluation of the integral may be complex due to the need to estimate the distribution of those models that are compatible with our dataset. The most common approach to solve this step is to collapse $p(\mathbf{W}|X, Y)$ into a Dirac delta function as follows
$$p(\mathbf{W}|X, Y) = \delta(\mathbf{W} - \mathbf{W}^*) = \begin{cases} 1 & \text{if} \quad \mathbf{W} = \mathbf{W}^* \\ 0 & \quad \text{otherwise} \end{cases} \tag{6}$$

The selection of $\mathbf{W}^*$ may define different strategies for this simplification. A common approach in this regard is to use a maximum a posteriori probability approximation, as follows:
$$\mathbf{W}^* = \operatorname*{argmax}_{\mathbf{W}} p(\mathbf{W}|X, Y) \tag{7}$$

Thus, in prediction time, the target probability is approximated as follows:
$$p(y^*|\mathbf{x}^*, X, Y) = p(y^*|\mathbf{x}^*, \mathbf{W}^*) \tag{8}$$

With these equations in mind, we may recall that any prediction intended to be used as a decision in an undetermined scenario should be complemented with an additional modelling of the uncertainty associated [56]. We now turn our attention to Eq. 5 to see how the different parts of this integral link with different aspects of the uncertainty found in the predictions.

3 DEFINITION OF UNCERTAINTY

Uncertainty is a polysemic word related to concepts like doubt, hesitancy, unpredictability, or indeterminacy. This potential set of multiple meanings can be mapped into the machine learning scenario, where the most widely used meaning refers to the lack of confidence in the predictions output by a model. In line with this, the literature offers one possible taxonomy of uncertainty [24, 46, 78]:

- **Epistemic uncertainty** corresponds to the uncertainty originating from a lack of knowledge ("episteme" is the Ancient Greek word for knowledge). This lack of knowledge can come from two different sources. One is the lack of evidence to learn. In this case, the model does not have access to enough cases to have a proper knowledge to output confident predictions. Another source for this lack of knowledge comes from the ignorance of what kind of model better suits a given problem.
- **Aleatoric uncertainty**. Aleatoric, or aleatory uncertainty, refers to statistical variability and effects that are inherently random in the data generation process, i.e. due to measurement noise or inherent ambiguity of the data. There are two types of aleatoric uncertainty, according to the following assumptions:
  - **Homocedastic uncertainty** measures the level of noise derived from the measurement process. Homoscedastic uncertainty is invariant to different inputs, meaning it remains consistent regardless of inputs.

– **Heteroscedastic uncertainty** measures the intrinsic noise in the observations in a way that depends on the inputs of the model.

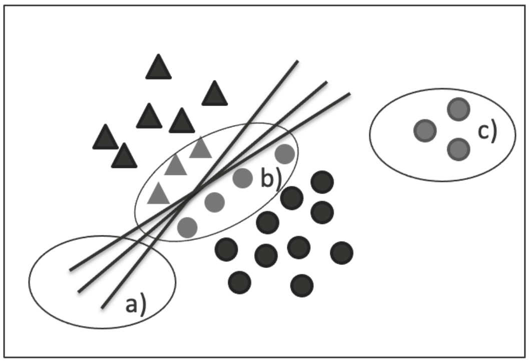
Another criterion used to categorize uncertainty is the extent to which it can be reduced. Accordingly, uncertainty can be,

- **Reducible**. This type of uncertainty decreases as we incorporate more examples into our training set. Hence, by getting more information about the problem, we reduce the derived uncertainty. The same is true of the type of model applied. By increasing the complexity and tuning architectures and hyper-parameters, the uncertainty associated with the capacity of the model shrinks.
- **Irreducible**. In this case, the uncertainty is inherent in the data. Even after adding more examples or increasing the capacity of the models, this type of uncertainty will not decrease. The assumption is that uncertainty is caused by processes that are inherently random when acquiring the data or in the data generation itself.

In many works, epistemic uncertainty is associated with reducible and aleatoric with irreducible uncertainties. By increasing the amount of information, we reduce epistemic uncertainty, while aleatoric uncertainty will never get smaller, even if we master the problem. It is worth noting that these distinctions are relative and can be subject to interpretation [21, 24]. For example, Der Kiureghian and Ditlevsen [21] show situations where aleatoric uncertainty turns into epistemic uncertainty as the model evolves.

Additionally, some authors [11, 58] refer to an alternative type of uncertainty: distributional uncertainty, also known as "out-of-distribution". This type corresponds to changes in the distribution of the data in prediction time. Such changes may introduce uncertainty due to a potential lack of generalization of the model, as it requires that the model predicts data points that are different from those seen during training. Figure 1 shows the different types of uncertainty described using a simple linear model to illustrate how a) the uncertainty on the model parameters is related to the set of different models that solve the problem, b) how points close to the decision boundary show high aleatoric uncertainty and c) data points not included in the training set can induce high uncertainty in prediction time.

Fig. 1. Simple linear model to illustrate different sources of uncertainty: a) model or epistemic, b) data or aleatoric, c) distributional or out-of distribution



In order to illustrate these concepts, let us consider, for example, the case of detecting a type of lung tumour. Based on a set of images obtained for patients in a hospital, the goal is to analyze lung images to detect whether there is a tumour or not. The sensors used to obtain these images are known and calibrated. The hospital's data scientist trains a Deep Learning model with these data, testing different models and using uncertainty to compare them and select the best performing one. In this situation, the uncertainty analyzed is epistemic, and it can be reduced by increasing the number of images.

Later, another hospital is interested in the prediction system and starts using it on its premises. As the sensors used in this new setting differ from the former hospital, a new source of aleatoric uncertainty is introduced. As soon as the data scientists at the new hospital realize this, they contact the original data scientist and fine-tune the system by incorporating images from the new sensors. Thus, after this process is complete, the aleatoric uncertainty becomes epistemic once again, and can be reduced as more data are added to the training set. Additionally, there are situations in which changes in the measurement equipment generating the data, e.g. adding more precision to the tools used to acquire the data, will reduce the aleatoric uncertainty.

3.1 Bayesian perspective on uncertainty

At this point we can establish a direct relationship between these concepts and the probabilistic terms introduced in Equation 5. We observe two terms contributing to the integral:

- $p(y^*|\mathbf{W}, \mathbf{x}^*)$: given a model with parameters $\mathbf{W}$, this term defines the probability of the output for the inputs $\mathbf{x}^*$ received during "test" time. Let us remember that, in this case, we assumed that model $\mathbf{W}$ is able to capture the distributions in X, Y. Here, the distribution captures the effect of the noise inherent in $\mathbf{x}^*$ on the target y^*.
- $p(\mathbf{W}|X, Y)$: this term estimates the probability of the model with parameters $\mathbf{W}$ conditioned by the training dataset $\mathcal{D}$. Also, as commented in Section 2, we assume that we train the model using only X, Y, considering that we do not have access to $\mathbf{x}^*$ or y^* during training.

According to the above definitions of different uncertainties, we can identify the epistemic and aleatoric elements. When focusing on $p(y^*|\mathbf{W}, \mathbf{x}^*)$ for a fixed $\mathbf{W}$, we can derive from this the statistical elements representing the aleatoric component of uncertainty, that stemming from the actual data. This should answer questions regarding whether the expressivity of the model is enough to capture the original distributions or whether $\mathcal{D}$ is a representative sample of the entire data distribution.

On the other hand, when considering $P(\mathbf{W}|X, Y)$, we are focusing on the distribution of models being compatible with the data. This distribution allows us to characterize epistemic uncertainty, which should allow us to answer such questions as the extent to which the model is able to truly capture the variability of the training set, or to characterize the different models that are compatible with the data at hand.

Moreover, aleatoric and epistemic uncertainties are not independent categories. As reflected in [41], the two types are related, the former often being conditioned by the underlying epistemic uncertainty. This is again reflected in Equation 5, where we observe that aleatoric and epistemic uncertainty are linked by $\mathbf{W}$.

Said link between the two is especially relevant in the case of using black-box models. By black-box models, we mean the application of pre-trained classification systems, delivered through software libraries or application programming interfaces (APIs). In this case, the model parameters are given and fixed, and the practitioner has no access to the internals of the model nor to the data used for training it. In such a scenario, measuring the uncertainty of the predictions of these classifiers when applied to a target application might be more relevant. However, in the case of APIs, there is no way of reducing the epistemic uncertainty, as the model cannot be trained further. Therefore, we have to focus on the term $p(y^*|\mathbf{W}, \mathbf{x}^*)$ and measure the aleatoric uncertainty associated with applying the classifier to the target dataset.

In this section, we have seen how each component of Equation 5 is linked to each type of uncertainty, i.e. aleatoric and epistemic. This means that when summarizing uncertainty, we will be able to map each type of uncertainty to the corresponding term.

4 MODELLING POSTERIOR DISTRIBUTIONS FOR UNCERTAINTY ESTIMATION

This section surveys different ways of estimating the posterior distributions involved in Equation 5. We will see how some methods focus on modeling the parameters **W**, and thus tackle the more epistemic part of the uncertainty. Other methods will take the model as granted and focus on modeling the output, thereby handling the aleatoric component of uncertainty. Figure 2 shows a taxonomy of the works surveyed in this article and how the compilation of articles has been organized around the different categories.

4.1 Epistemic Uncertainty

In this section, we focus on the $p(\mathbf{W}|X, Y)$ term in Equation 5. The section name, epistemic uncertainty, refers to a common aspect of the methods included, which consider the parameters of the network **W** to be unknown and model them as random variables. A neural network implemented by considering a prior over its parameters and that approximates the posterior using Bayesian inference is called a Bayesian Neural Network. Once we have a classifier modeled as a Bayesian Neural Network, we are able to analyze the random variables that describe its parameters to study the associated uncertainty. For example, if we model the weights **W** of the network as Gaussian distributions, we can study how the scale parameter of these distributions diminishes as we incorporate more data into the training set.

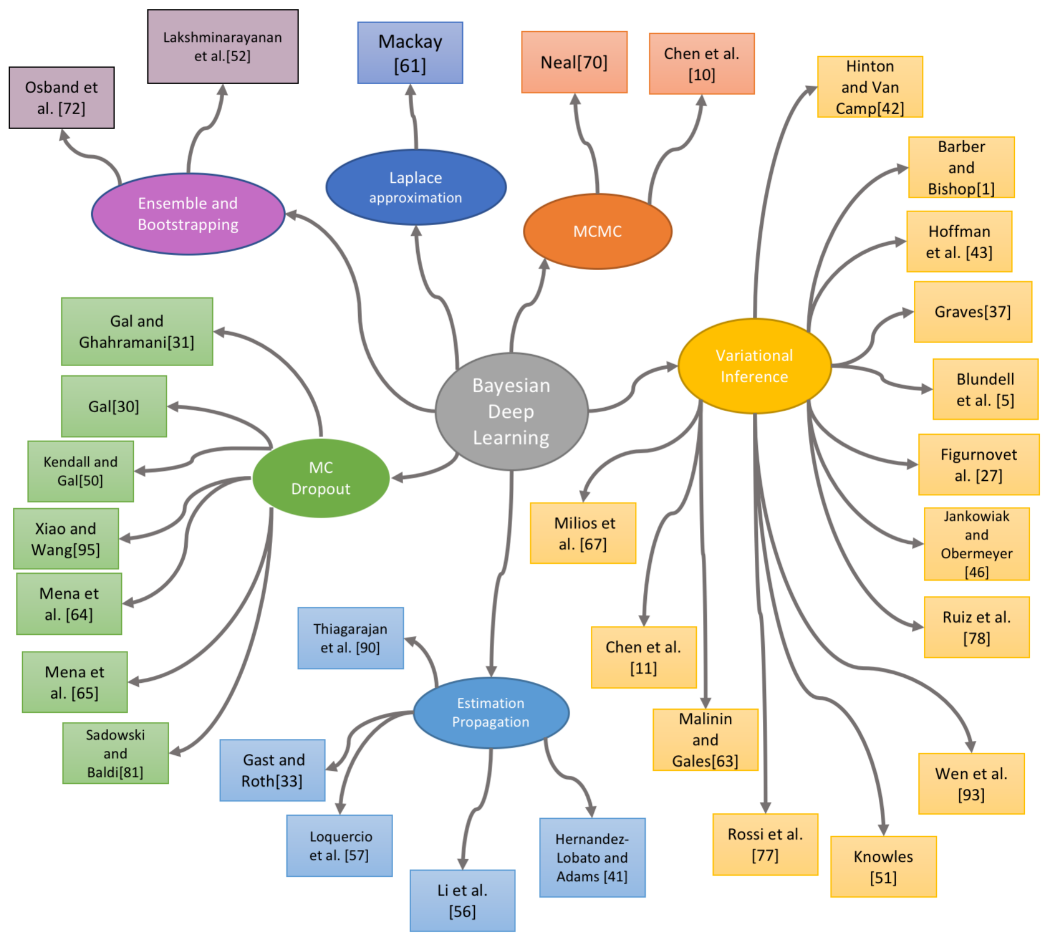
In this section, we review various approaches proposing different ways of modeling the parameters using different probability distributions and methods to solve the posterior. Some model the weights with simple distributions that allow the equation to be solved analytically. Others try to discretize the normalization over **W** by using different sampling methods. Others still propose variational posteriors and then use optimization techniques to reduce the difference between the variational and the true posterior.

4.1.1 Laplace approximations. One of the first approaches to Bayesian Neural Networks was proposed in MacKay [57] and focused on model selection for regression problems. In this case, Bayesian inference was not applied only to the weights of the model but also to the regularization terms. By restricting the type of random variables used for modeling the parameters, the method was able to solve the posterior analytically and by using backpropagation methods for optimization.

Beyond the commonly used approach of hold-out data or cross-validation, MacKay [57] proposes incorporating the optimization of hyper-parameters, such as regularization terms, as part of the optimization process, including the task of model selection in the Bayesian inference process. This model selection also includes the selection of hyper-parameters, such as regularization terms, for example: those that are included in the loss function in order to penalize large weights and achieve a smoother or simpler mapping [74]. This additional step is part of a Bayesian inference schema, where it establishes a Gaussian prior over the parameters and uses Bayes to estimate the posterior of the *evidence*, i.e. $p(D|\mathbf{W}, \mathcal{R})$, where $\mathcal{R}$ corresponds to the set of regularization parameters. It then uses the value of this evidence to compare models.

More relevant to this work is the extension for classification systems proposed by MacKay [56]. In this paper, each classifier produces two kinds of outputs. In essence, the Maximum a Posteriori, MAP, used to train the models by minimizing a loss function, comparing it to the ground truth, and a *moderated* one, which consists of the prediction together with a measure of its uncertainty.

Fig. 2. Taxonomy of Bayesian Neural Network methods for estimating uncertainty



In prediction time, this *moderated* output can be used to make decisions based on the uncertainty, compare models, and even use it for active learning schemes.

In both of the aforementioned works, the evidence is evaluated using the Laplace method with a Gaussian approximation based on the properties of the *most probable* fit $\mathbf{W}_{MAP}$, and the error bars A^{-1}, which corresponds to the inverse of the Hessian matrix of the parameters from the loss function. Some assumptions are made for this Gaussian approximation, such as considering the surface around the minimum as a quadratic function, something that rarely occurs in real problems.

Moreover, approximating the Hessian, especially when using complex architectures with millions of parameters like those used in Deep Learning nowadays, is not trivial and, as reported by Bishop in [3], "the accurate evaluation of the evidence can prove to be very difficult. One of the reasons for this is that the determinant of the Hessian is given by the product of the eigenvalues and so is very sensitive to errors in the small eigenvalues". In addition, in order to obtain the moderated output resulting from the combination of the Gaussian and the sigmoid non-linearity, these authors introduce a new approximation based on a function of the variance parameter applied to the output activations.

4.1.2 Markov chain Monte Carlo approximations. MacKay [56] solved the integral in Equation 5 by employing Gaussian approximations for the prior and posterior distributions, which allows these integrals to be solved analytically. Because of the limitations posed in the previous section, this Gaussian approximation might not always be recommendable, especially in modern architectures with millions of parameters.

An alternative approach for solving the integral in Equation 5 is to approximate it with the sum of L models, similarly to that proposed by Lakshminarayanan et al. [48] in ensemble learning:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \frac{1}{L}\sum_{i=1}^{L} p(y^*|\mathbf{W}_i, \mathbf{x}^*) \tag{9}$$

In this case, the vector of parameters in each model, $\mathbf{W}_i$, is sampled from the distribution $p(\mathbf{W}|\mathcal{D})$. In [66], Neal reviews this approach and different ways of conducting this sampling. The proposed method is a combination of different sampling policies, known as hybrid Monte Carlo. In short, the method combines a Gibbs sampling method for selecting the hyper-parameters together with a Metropolis-like sampling method for selecting the parameters or the network. By using the hybrid Monte Carlo algorithm, suitable sampling vectors of the parameters $\mathbf{W}_i$ can be generated in reasonable computational times. Once the method reaches an equilibrium in prediction time, it is possible to sample predictions from the posterior. By analyzing this sampling, it is possible to assess the uncertainty of the predictions given an input $\mathbf{x}_i$, which we call the aleatoric uncertainty of the model.

The problem with this kind of solutions is that they can take longer to converge because of the stochasticity inherent in the process, even when using smart approaches like hybrid Monte Carlo. Besides that, it is difficult to obtain insights by directly observing the parameters of the model, as was the case with the aforementioned Gaussian approaches, which burden the capacity of these algorithms when it comes to model selection.

In [10], Chen et al. work on the problem of convergence by using second-order Langevin dynamics with a friction term that counteracts the effects of the noisy gradient. The article studies the effect of applying this method on the classification of MNIST images, and observes how sampling-based methods are better than optimization-based ones, showing how Bayesian inference has an advantage in this setting.

4.1.3 Variational Inference. A different approach for estimating the posterior is that followed by Hinton and Van Camp [39], called *ensemble learning*. In this article, they tackle the model selection problem by following Occam's razor law, as in MacKay [56]. In this case, however, they apply the Minimum Description Length Principle (MDLP), which is used to determine the appropriate complexity of a model for a given dataset. Adding unnecessary structure to a model can result in overfitting and high variance, whereas insufficient model complexity can lead to a high bias.

Hinton and Van Camp [39] provide a way to derive the error together with the amount of information in the weights. This assumption is especially relevant in problems with a small dataset, however, it is limited to models with a linear output. Here, the MDLP asserts that the best model for some data is that which minimizes the combined cost of describing the model and minimizing the error. In supervised Neural Networks, the model cost is the error in the number of bits it takes to describe the discrepancy between the correct output and the output of the neural network in each training case and the number of bits it takes to describe the weights. If we remember Equation 5, this corresponds to the left and right terms, respectively.

For encoding misclassifications in the output, the method described in [39] proposes discretizing the distribution in t-wide bins and modeling each output unit as a Gaussian with zero mean and a standard deviation of σ. Assigning a value for this σ to the root square deviation of the misclassifications from zero results in a misclassifications cost function of a constant. To encode the weights again, they are modeled as Gaussian random variables with zero mean and σ variance. The cost function is then equal to an RMSE (Root Mean Square Error) with quadratic weight decay. These authors complain, however, that this approach uses the same σ for all of the weights without taking into account different levels of contribution. This is translated into a waste of bits when encoding the information. Compared to [55], which also looked at variations in the output once the model is trained, in [39] the variance during the training is also taken into account.

The method based on the MDLP uses information theory to encode the weights by adding Gaussian noise to each one. Thus, although noisy weights can be more straightforward to communicate, they may lead to high variance on the output, which would imply more cost in communicating the misclassifications. To measure the cost, it proposes using the Kullback-Leibler divergence ($\mathcal{KL}$) as a measure of the difference between the likelihood $\mathcal{P}$ and the posterior $\mathcal{Q}$. To determine the amount of bits it costs to communicate the weights, $C(w)$, it takes each weight's posterior and binarizes it. For each bin, it computes the information using the prior. It then uses the weights to encode the noisy weights and obtain the exact posterior $\mathcal{Q}$. Next, it uses Q and binarizes the misclassifications to compute the transmission cost of the weights, $\mathcal{R}(w)$. The difference between one and the other is, again, the $\mathcal{KL}$, which can be solved analytically when using Gaussian distributions.

To learn the communication cost of erroneous predictions, the expected value of the Root Mean Square Error, RMSE, is required. Considering the noisy weights obtained, the root squared error might be hard to compute. The calculations are performed considering linear outputs and a single hidden layer. By having one single layer, a matrix can be created with the mean and variance for the outputs by means of Monte Carlo sampling. Then, to compute the RMSE for each output unit, the sum of the mean is used to subtract the ground truth, adding a variance term obtained from the table. The number of computations required for this approach when using Deep Learning makes it unfeasible. A useful feature of the proposal is that it does not require a unimodal Gaussian approach for the prior. To adapt to cases like a spike around 0 or 1, a mixture of Gaussians is proposed. This approach to computing the posterior therefore constitutes an alternative to restrictive Gaussian approximations by using more adaptive functions.

Moreover, by employing the $\mathcal{KL}$ divergence, this method can approximate the integral of the posterior analytically, avoiding the high computational requirements associated with the use of Monte Carlo Markov Chains. We would also like to highlight the fact that it addresses estimation of the epistemic uncertainty when modeling the distribution for the weights and aleatoric when modeling the distribution of the outputs. In addition, the architectures used to illustrate the method are very simple when compared to current Deep Learning models. A significant limitation of this work when applied to the classification problem is that its assumes a linear output, this not being the case for probabilistic classifiers, which use a softmax no-linearity

To tackle the issues presented by Hinton and Van Camp [39], Barber and Bishop [1] extend the model to allow a Gaussian approximating distribution with a general covariance matrix, meaning it can capture the posterior correlations between parameters while still leading to a tractable algorithm. Also, to illustrate the method, they overcome the stated limitation of using linear output functions only and apply a variation of a sigmoid function (cumulative Gaussian), while also making use of the $\mathcal{KL}$ divergence to approximate the integral defined in Equation 5. In order to capture the variance in weights and output, this method still applies a Gaussian approximation for both concepts. In the case of the output, it models the variance with a parameter σ and the weights with a covariance matrix $\mathbf{A}$. Hence, the marginal likelihood is given by

$$p(D|\sigma,\mathbf{A}) = \int_{\mathbf{W}} p(D|\mathbf{W},\sigma)p(\mathbf{W}|\mathbf{A})dW. \tag{10}$$

This integration is, in general, analytically intractable. Instead of applying a Laplace approximation as in [57], here the authors consider the logarithm of the marginal likelihood defined in Equation 10. By introducing a variational posterior $Q(\mathbf{W})$, and making use of Jensen's inequality together with the convexity of the logarithmic function, they end up with the following decomposition of the marginal likelihood:

$$\log p(D|\sigma,\mathbf{A}) = \mathcal{KL}(Q||P) + \mathcal{F}(Q) \tag{11}$$

The main idea is that this posterior $Q(\mathbf{W})$ represents a lower bound on the log marginal likelihood, as $\mathcal{KL}(Q||P) \geq 0$. Maximizing the lower bound $\mathcal{F}(Q)$ for the parameters of Q is equivalent to minimizing the Kullback-Leibler divergence. When taken to its limit, if $Q(\mathbf{W})$ is equal to the true posterior, the $\mathcal{KL}(Q||P)$ will be zero. The trick here is to choose a form for the $Q(\mathbf{W})$ distribution so that the lower bound $\mathcal{F}(Q)$ can be evaluated efficiently.

Again, they choose a Gaussian model for the posterior $Q(\mathbf{W}) \sim \mathcal{N}(\bar{\mathbf{W}},\mathbf{C})$, where $\mathbf{C}$ covariance matrix to approximate the posterior by playing with its decompositions, which enables the use of analytical techniques. Hence, it is possible to reduce the different non-tractable integrals to one-dimensional ones that can then be evaluated using standard numerical techniques. Similar analytical techniques are employed to compute the gradients of the $\mathcal{KL}$ divergence used to optimize the posterior, which allows for application of a gradient optimizer.

However, the variance parameters,σ, $\mathbf{A}$, have so far been considered as constants. To include these parameters in the Bayesian framework, a standard isotropic prior covariance matrix of the form $\mathbf{A} = \alpha\mathbf{I}$ is considered and hyperpriors introduced in the form of Gamma distributions. Then, by modeling the joint posterior $p(W,\alpha,\sigma|D)$ by a factorized approximating distribution of the form $Q(w)\mathcal{R}(\alpha)\mathcal{S}(\sigma)$, an alternate optimization process can be used to approximate these posterior distributions.

We therefore see how the proposed method still has a strong dependency on Gaussian distributions to take advantage of analytical operations that simplify the calculation of the integrals. The covariance matrix used to model the variance on the weights may pose a problem with modern architectures as it correlates with the number of parameters. While it is true that the authors provide an approximation for using a diagonal matrix, this still means a significant amount of extra parameters to be optimized. Also, the method and problems used to illustrate it focus on regression problems, leaving the classification problem unsolved.

One of the main problems of the methods described so far is that they are hard to implement in modern Deep Learning architectures, where the number of parameters is high. Graves tackles this problem in [34] by introducing a variational stochastic method. This paper proposes working with expectations of variational distributions that approximate the true posterior instead of trying to apply analytical solutions to find it.

To implement the variational stochastic method that estimates these posteriors, the solution returns to the MDLP introduced in [39], applying it to the loss function to find a trade-off between prediction accuracy and the complexity of the model. This variational distribution takes the shape of a diagonal Gaussian $Q(\mathbf{W}|\sigma)$. By applying the MDLP, the loss is:

$$\mathcal{L} = \mathcal{L}^N(\mathbf{W},\mathcal{D})_{\mathbf{W}\sim Q(\sigma)} + \mathcal{KL}(Q(\sigma) \parallel P(\alpha)), \tag{12}$$

where the left term corresponds to the accuracy objective, i.e. the Negative Log-Likelihood (NLL) of the network, and the right term controls the complexity by trying to minimize the distance between the posterior and the prior. The Negative Log-Likelihood is computed by obtaining samples from the posterior $Q(\sigma)$. This paper describes how the authors ran experiments with different combinations of posterior distributions, such as Gaussian and Delta distributions, and priors, such as Uniform, Laplace and Gaussian priors. They obtained the best results when both were Gaussian. To optimize the loss function, they applied an optimizer that can tolerate noisy gradient estimates, experimenting with Steepest descent with momentum and Resilient Backpropagation (RPROP). This method has been criticized over a bias effect on the gradients caused by the Monte Carlo samplings.

In the same line, Hoffman et al. also develops the concept of Stochastic Variational Inference in [40] to approximate the posterior of predictive models, emphasizing the fact that the stochastic version of Variational Inference is better suited to massive datasets. By making use of the meanfield assumption, which considers all of the parameters as independent variables, it achieves noisy but unbiased gradients and applies a stochastic optimization process to estimate the posteriors, similarly to that described in [34]. In this case, however, the priors and posteriors used are modeled as Dirichlet distributions applied to topic modeling problems.

Based on the work in [34] on Variational Inference, we find the Bayes by Backprop method described in [5], which is aimed at learning a probability distribution on the weights of a neural network. This focuses on the regularization effect when attributing a compression cost over the weights while obtaining the uncertainty associated with the predictions. In this case, the novelty lies in the reparametrization trick, which consists of decomposing the parameters of Gaussian variational posteriors, as follows,

$$w = \mu + \log(1 + \exp(\rho)) \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1). \tag{13}$$

Hence, by drawing Monte Carlo samples from this distribution, the resulting mean is used. Blundell et al. derive the gradients of the mean, μ, and standard deviation, ρ, and use them to update the weights, again obtaining noisy but unbiased gradients. Note how, in this case, by doubling the number of parameters, two per weight, they handle an infinite number of models in an ensemble. As for the prior distribution, they propose using a mixture of Gaussians with zero mean but different variances. An interesting aspect worth highlighting in this work is that the method is illustrated by means of a classification problem, MNIST, a database of handwritten digits commonly used to train computer vision systems.

Beyond these seminal works, Variational Inference (VI) is a very active research topic. Different research lines have addressed problems such as creating black-box versions of the Variational Inference or using other distributions rather than Gaussians, and how to apply the reparametrization trick to them [25, 43, 47, 73].

A recent approach used in modern architectures is Flipout [89], which integrates over the weights by performing a Monte Carlo approximation sampling from the weight distributions. Flipout uses roughly twice as many floating-point operations as the reparameterization estimator, but has the advantage of significantly lower variance. By substituting the layers of a classifier with the equivalent Flipout ones, it is easy to apply Variational Inference to a classification system, including Recurrent or Convolutional Neural Networks architectures. The Flipout gradient estimator is then used to minimize the $\mathcal{KL}$ divergence up to a constant, also known as the negative Evidence Lower Bound (ELBO). This consists of the sum of two terms: the expected Negative Log-Likelihood, which is approximated via Monte Carlo; and the $\mathcal{KL}$ divergence, which is added via regularizer terms to the layer.

Another approach for applying Variational Inference in the case of the classification problem is that proposed in [72]. In this work, Rossi et al. extend the Variational Inference regression setting to k-class classification problems. The proposed method assumes a one-hot encoding of the labels, so that Y is a matrix of zeros and ones. It then models the posterior over classification functions by applying regression on a transformation of the labels, as proposed in [62].

4.1.4 Expectation propagation. In this section, we review an alternative to Variational Inference, introduced in [38] by Hernandez-Lobato and Adams and called Probabilistic Backpropagation (PBP). Again, the goal of the method is to tackle the scalability problems of Bayesian Neural Networks, similarly to stochastic Variational Inference, while preserving their benefits, i.e. hyper-parameter tuning, less overfitting, a measure for uncertainty (by building credible intervals based on the predictions obtained), and smaller volumes of training data. The main idea is to compute a forward propagation of probabilities throughout the network to obtain the marginal likelihood. Then we compute the gradients of this marginal likelihood and use them to approximate the posterior by performing a backward computation.

Probabilistic Backpropagation assumes Gaussian distribution for the output conditioned to the input, the weights and the noise precision σ with a mean equal to the output of the NN, and deviation equal to σ^{-1}. The method also assumes priors over the weights modeled as Gaussians with mean equal to 0 and deviation α^{-1}. Hyper priors of both σ and α are Gamma distributions. The posterior distribution for the parameters W, σ and α can then be obtained by applying Bayes' rule. However, the exact computation of this posterior is not tractable.

The Probabilistic Backpropagation method is proposed to approximate the aforementioned posterior. Similar to traditional backpropagation, in the forward phase, the weights are applied to the input to produce the activations. As the weights are now random variable distributions, the resulting activations are also random and have intractable distributions. These activations are then approximated with onedimensional Gaussians that match their marginal means and variances. Even though the example used to illustrate this is again a regression problem, the logarithm for the marginal probability of the target variable is evaluated at the output rather than the error. Based on this, the gradients of this quantity are updated for the means and variances of the approximate Gaussian posterior.

Updating of the weights is complex in this case. Applying the Bayes' rule to update the beliefs based on the data requires an approximation with a more straightforward distribution function $Q(\mathbf{W})$. We use this $Q(\mathbf{W})$ function, together with the original update function, to minimize the $\mathcal{KL}$. The parameters of this new function derive from an update based on the original mean and variances. All in all, the posterior is approximated with a factored distribution of these $Q(\mathbf{W})$ and the Gamma hyper-priors, ensuring that both the true posterior and the approximation share the same moments.

The Expectation Propagation method (EP) improves this assumed density filtering implementation by iteratively incorporating each factor multiple times. On each pass, Expectation Propagation removes one factor from the posterior, re-estimates it, and then reincorporates it afterwards. One drawback of Expectation Propagation is that it requires all of the factors to be retained in memory for each sample; however, since they operate in mini-batches, the computational cost is reduced.

In [52], Li et al. propose a variation of the method, called Stochastic Expectation propagation. The aforementioned paper illustrates how the method can be applied to different classification problems

One of the main advantages of using Probabilistic Back-propagation is that, unlike traditional back-propagation approaches, it does not require hyper-parameters like learning rates, for example.

4.1.5 Monte Carlo Dropout. All of the methods presented thus far belong to the category of Bayesian Neural Networks. They mainly present different approaches for solving the integral defined in Equation 5: by solving it analytically assuming some approximations, by using Monte Carlo by sampling different models, or by approximating the posterior with a simpler distribution. The weights or parameters of the network are assumed to follow a given distribution and we have seen different ways of learning the parameters of those distributions with variants of the backpropagation process for optimizing the neural network.

In this section, we present an alternative that does not require modifying the architecture of the neural network or assume distributions over the network parameters. Rather, it builds on the concept of Dropout introduced by Srivastava et al. in [84]. Dropout was conceived as a method for reducing over-fitting on the training of neural networks. The main idea is that some neurons are deactivated during training, forcing the active ones to assume the training effort of those that have been deactivated.

In a neural network using Dropout, each unit will have a probability of going offline. On every training pass, based on this probability, some units will turn off, and the rest will be retained for the examples of that given batch during the training process. Thus, by avoiding specialization of the neurons, the neural network will be more robust against overfitting problems.

Interpreting Dropout from another point of view, we might consider the architecture and parameters to be different in every training batch. The probabilities associated with each unit allow different models to be sampled, similarly to the Monte Carlo method proposed in [66]. In [29], Gal and Ghahramani take advantage of this sampling ability and propose using Dropout as an approximation of the probabilistic deep Gaussian process. With regular Dropout layers, as found in many architectures, Dropout minimizes the Kullback–Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process.

As with Variational Inference methods, the idea is to reduce the $\mathcal{KL}$ divergence between the true posterior and a variational posterior $Q(\mathbf{W})$. In this case, however, $Q(\mathbf{W})$ is a distribution over the matrices $\mathbf{W}_i = \mathbf{M}_i \cdot diag([z_{i,j}j = 1])$, and $z_{i,j} \sim Bernoulli(p_i)$. The binary variable $z_{i,j} = 0$ then corresponds to unit j in layer $i - 1$ being dropped out as an input to layer i.

In the prediction phase, they sample T sets of vectors from the Bernoulli distribution $z_{i,j} \sim Bernoulli(p_i)$, thus obtaining different T values for y^*, and $E(y^*|\mathbf{x}^*)_{(Q)} \approx \frac{1}{T}\sum_{t=1}^{T} f^{\hat{\mathbf{W}}_j}(\mathbf{x}^*)$. This Monte Carlo estimate is what the authors refer to as Monte Carlo Dropout. The main advantage of the Monte Carlo Dropout is the fact that it operates directly with Dropout layers present in the definition of many neural network architectures, so there is no need for extra parameters, modifications on the backpropagation process or the computing of complex Hessian matrices.

4.1.6 Ensemble and Bootstrapping. In a similar vein to Monte Carlo Dropout, where the approach is to have multiple models available at the same time and then be able to sample from them and thus obtain an approximation for the posterior, we find Ensemble [48], and Bootstrapping [68]. The idea here is to approximate the integral of the posterior with a set of models.

In Ensemble methods, the approach is to train M models using different weight initialization for each of them. The models are then trained together sharing a joint loss function. In the prediction phase, the outcome of the ensemble model is the average of the individual models. By observing the variance of the outputs, it is also possible to obtain a measure of uncertainty in the models.

Bootstrapping uses a similar strategy, but instead of replicating the same model M times, it uses a shared network with different heads. The training data, $\mathcal{D}$, is divided into different subsamples $\tilde{\mathcal{D}}$ sampled uniformly with a replacement of $\mathcal{D}$. This can be understood as a data dropout approach, in the sense that each data point is included in the training of a particular head depending on a probability $\sim Bernoulli(p)$. Thus, each head is trained only on its bootstrapped $\tilde{\mathcal{D}}$, allowing the shared network to learn a joint feature representation across all of the data.

Both Ensemble and Bootstrapping represent methods that do not require significant changes to the network architecture while providing means for obtaining the uncertainty of predictions. Specifically, according to [83], despite its simplicity, uncertainties obtained using the Ensemble method manifest higher robustness to posterior domain shifts.

4.2 Heteroscedastic Aleatoric Uncertainty

The previous section provided a review of different methods for approximating the posterior in Equation 5, the first step in the process of modeling uncertainty in classification systems. The methods surveyed so far propose considering the parameters of the classifier as unknowns and modeling them with random variables. Thus, these models focus on the $p(\mathbf{W}|x, y)$ component of the equation.

By contrast, in this section we focus on another type of uncertainty: heteroscedastic aleatoric uncertainty. Taking Equation 3, we now analyze the term $p(y^*|\mathbf{W}, \mathbf{x}^*)$, where we evaluate the probability of the output of a given model with parameters $\mathbf{W}$ predicting testing input data, otherwise known as data uncertainty.

Although the methods in Section 4.1 mostly studied epistemic uncertainty, and were mainly based on regression problems, some already considered data uncertainty. In the introduction to Variational Inference, when describing the work in [39], we saw how they divided the contribution of the noisy weights and misclassifications. In this case, they focused on a regression task and were approximating the output distribution of the square error with a Gaussian posterior distribution. Although the uncertainty in the output activations may be analyzed by observing their posterior distribution, it might be difficult to isolate the aleatoric part, as output activations are trained together with the posteriors over the weights.

Moreover, there would also be the problem of adapting the method to classifications were the output is not a single unit and includes non-linearities, like the standard softmax function. In recent Variational Inference approaches, Zhang et al. [91] tackled this problem by using categorical distribution, rather than Gaussian ones, at the output. Nevertheless, the training process includes both the weights and outputs as one.

4.2.1 Gaussian output models.

Relevant contributions [28, 46] can be found with regard to this separation of concerns. In these works, the authors extend the application of Dropout as a method for estimating the model uncertainty in neural networks introduced in [29], considering the peculiarities of the classification problem and the estimation of its aleatoric uncertainty

In their proposal for uncertainty in classification tasks, instead of modeling the output distribution, $y^* \sim Categorical(p)$, they model the noise at the logits u activation, thus avoiding dealing with a softmax non-linearity

$$\mathbf{u} \sim \mathcal{N}(f^{\mathbf{W}}(\mathbf{x}^*), diag(\sigma^2(\mathbf{x}^*))) \tag{14}$$

Here, each logit is modeled with a Gaussian distribution, with a mean equal to the output for the corresponding class, as obtained from the model $f^{\mathbf{W}}(\mathbf{x}^*)$ with parameters $\mathbf{W}$, and the variance modeled as a diagonal matrix with a parameter σ , learned from the input data. The resulting output of the model is:

$$p = softmax(\mathbf{u}) \tag{15}$$

The associated likelihood is obtained from the expectation of this probability by applying Monte Carlo to these distributions, drawing M samples from the Gaussian modeling the logits:

$$\mathbb{E}[p] = \frac{1}{M}\sum_{m=1}^{M} softmax(\mathbf{u}_m) \tag{16}$$

Next, the model is trained using a cross-entropy loss function that includes the resulting loglikelihood. To be able to apply backpropagation to the training including these new Gaussian random variables, the authors apply the reparametrization-trick introduced in [5]:

$$\mathbf{u} = f^{\mathbf{W}}(\mathbf{x}^*) + \sqrt{diag(\sigma^2(\mathbf{x}^*))} \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0,1) \tag{17}$$

And finally, to obtain the model parameters, W, they combine the Monte Carlo Dropout method with this aleatoric cross-entropy loss, allowing the joint modeling of both aleatoric and epistemic uncertainty.

In the original paper, Kendall and Gal [46] focused on Computer Vision, analyzing the role of uncertainty in image classification and segmentation. Later, Xiao and Wang [90] extended this work by studying its impact in the natural language processing field. It is interesting how they studied the different contributions of epistemic and aleatoric uncertainties and a combination of both. These authors ran different natural language processing experiments - language modeling, sentiment analysis and Named Entity Recognition - observing how including a combination of both epistemic and aleatoric uncertainties in training contributes to achieving better results in the classification tasks addressed. Moreover, focusing on black-box classification systems, and Sentiment Analysis in particular, we find [59]. In this work, the authors tackle the problem of estimating uncertainty in pre-trained models, where there is no option of adapting the model with variational layers, or assuming the existence of Dropout layers in the model, showing how, even in this scenario it is still possible to compute the uncertainty of such classification models.

4.2.2 Dirichlet models. In the method described in the previous section, two limitations can be observed. First of all, contrary to what happens in regression, where the output distribution is directly approximated, in classification, the aleatoric uncertainty is modeled using random variables at the logits, not the output, level. Each logit is modeled using a separate Gaussian random variable. In doing this, possible interactions between classes are not considered, and these random variables are tied together by applying a softmax at the output.

Moreover, using Gaussian modeling as a generic recipe might impose unnecessary constraints in some problems. In the regressions setting, this means imposing a uni-modality that might not reflect the actual condition of the data. In this type of problem, some alternatives consider more elaborated approximations, as detailed in [7].

In a classification problem, we would ideally like to model the Categorical output distribution, as composed of the probabilities associated with each output class. A natural interpretation of the Categorical output may be to consider the natural conjugates of those distributions. Thinking of this output as elements in a simplex, a Beta may be contemplated in the binary case or a Dirichlet in the Multinomial case.

The approach suggested by Sadowski and Baldi in [76] proposes using these distributions. In their work, these authors consider different applications of Beta and Dirichlet for different solutions. The article suggests using the Beta distribution for problems with a real output with upper and lower bounds. More relevant to us is the solution provided for multinomial classification systems. In these scenarios, the paper proposes two alternatives. One is to model the output as a multi-headed Dirichlet, where each output unit corresponds to an α control parameter of the overall distribution. This approach resembles the one posited in [46], but instead of modeling at the logits level, it models each output unit. Alternatively, it is also proposed that the softmax output be modeled directly with a Dirichlet and the parameters learned at once. For each option, the corresponding gradients are derived to enable training using standard optimizers. In the paper, the authors detected stability problems due to the digamma terms in the gradient of the Dirichlet, which may become large for small α_i, resulting in large gradients. To fix these problems, the article introduces a variety of activation functions, i.e. Inverse-Linear (IL), Exponential-Linear (EL) and Exponential-Tanh (ET). However, even after including these activation units, it was still necessary to add some regularizers to prevent very small α that generated overflow errors.

Another work that uses a Dirichlet as a probabilistic output layer is [31]. Here, Gast and Roth include uncertainty in Convolutional Neural Networks by two means. First, they propose changing the output layer for a probabilistic one, similarly to [76]. Second, by modeling the input as a Gaussian random variable, they apply Expectation Propagation, similarly to [38]. To implement Assumed Density Filtering to a Convolutional Neural Network, they provide a variational approximation for all the layers involved: pool and max-pool layers, and Relu and leaky Relus. Thus, they manage to model each output activation as a Gaussian random variable.

For classification purposes, the paper proposes using a Dirichlet as the output layer, building on top of the Assumed Density Filtering outputs. To convert these Gaussian activations to the α_j control parameters that govern the Dirichlet distribution, they propose a decomposition of these parameters based on the Gaussian output activations

$$p(y|\mathbf{x}^*, \mathbf{W}) = Dir(\alpha(\mu, v)), \quad \alpha(\mu, v) = \frac{m}{s}, \quad s = \sum_j \alpha_j^{-1} \tag{18}$$

and

$$m = softmax(\mu), \quad s = c_1 + c_2 \sqrt{\sum_j m_j v_j}, \tag{19}$$

where c_1 controls how spiky the distribution is and c_2 is an amplification factor for converting uncertainties into the Dirichlet scale. By including the m term in the variance, the corresponding uncertainty mainly comes from the class contributing most to the softmax activation. Hence, as the output is already a distribution, they are able to apply the maximum conditional likelihood learning that finds the parameters by maximizing the conditional likelihood of the data under a predictive model.

Similarly to [46], in [31] we find a way of modeling aleatoric uncertainty by using a Dirichlet at the output, and epistemic uncertainty by including Gaussian noise at the input and propagating the expectation with Assumed Density Filtering. By combining both types of uncertainty while including minimal changes in the network architecture, they achieve the goal of converting a Convolutional Neural Network that outputs point estimates to a probabilistic Neural Network that outputs the predictions together with a measure of uncertainty.

In [85], we find another extension of [31], although in this case adding an uncertainty attribution method. This attribution means identifying which features in the input data have a higher impact on the aleatoric uncertainty of the model. In this article, Thiagarajan et al. focus on regression problems with tabular input data, meaning it is easy to illustrate the influence at the attribute level. To measure this attribution, they suggest decomposing the function value $f(x)$ as a sum of relevance scores, obtained using a first-order Taylor expansion of the function. Another benefit of this decomposition is that it can act as a regularizer when added to the loss, increasing the model's generalizability.

In [11, 58], we find new approaches that build on top of Dirichlet distributions, focused in this case on the detection of out-of-distribution data. Both methods model *distributional uncertainty*, i.e. a distributional mismatch between the test and training data distributions. Although this type of uncertainty is different from the aleatoric one analysed in this section, we include them here because of their usage of Dirichlet distributions for modelling the data. In [58], Malinin and Gales explicitly separate the three types of uncertainty:

In [11, 58], we find new approaches that build on top of Dirichlet distributions, focusing in this case on the detection of out-of-distribution data. Both methods model distributional uncertainty, i.e. a distributional mismatch between the test and training data distributions. Although this type of uncertainty differs from the aleatoric one analyzed in this section, we include these aproaches here because of their use of Dirichlet distributions to model the data. In [58], Malinin and Gales explicitly separate the three types of uncertainty:

$$p(y^*|\mathbf{x}^*, X, Y) = \int_W \int_\mu \underbrace{p(y^*|\mu)}_{data}\ \underbrace{p(\mu|\mathbf{W}, \mathbf{x}^*)}_{distributional}\ \underbrace{p(\mathbf{W}|X, Y)}_{model}\, d\mu\mathbf{W} \tag{20}$$

The point estimate for data uncertainty is determined by the distributional uncertainty. To model the new term $p(\mu|\mathbf{W}, \mathbf{x}^*)$, the paper proposes using a Dirichlet distribution $Dir(\mu|\alpha)$, where $\alpha = f(\mathbf{x}^*; \mathbf{W}^*)$. To train the posterior, these authors explicitly train in a multi-task fashion so as to minimize the KL divergence between the model and a sharp Dirichlet distribution focusing on the appropriate class for in-distribution data, and between the model and a flat Dirichlet distribution for out-of-distribution points:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}_{\mathrm{p_{in}}(\mathbf{x})}[KL[\mathrm{Dir}(\mu|\hat{\alpha})||\mathrm{p}(\mu|\mathbf{x}; \mathbf{W})]] + \mathbb{E}_{\mathrm{p_{out}}(\mathbf{x})}[KL[\mathrm{Dir}(\mu|\tilde{\alpha})||\mathrm{p}(\mu|\mathbf{x}; \mathbf{W})]] \tag{21}$$

Where we find a decomposition of the loss composed of a term for the in-distribution and out-of-distribution data samples. The multi-task part arises from the need to have in- and out-of-distribution points, which are simulated by generating points on the boundary of the in-domain region using a generative model or using a different, real dataset as a set of samples from the out-of-domain distribution.

Following a similar approach, Chen et al. [11] propose to parameterize a posterior model $p_{\mathbf{W}}(Z|X)$ and optimizing its parameters to estimate $p_{\mathbf{W}}(Z|X)$ given a pairwise input (X, Y) by minimizing their $\mathcal{KL}$ divergence:

$$D_{\mathrm{KL}}\left(P_{\mathbf{W}}(Z|X)||P(Z|Y)\right) \tag{22}$$

To minimize the divergence, they employ Variational Inference, using a Dirichlet as the variational posterior. However, in this case they do not explicitly separate the out-of-distribution part. To demonstrate the out-of-distribution detection, they train the network using one dataset and then analyze the uncertainties found when applying the model to a different one.

In Mena et al. [60], a Dirichlet distribution is proposed for modeling the uncertainty of black-box classification systems when applied to target applications. In this work, the authors propose a decomposition of the concentration parameters of a Dirichlet distribution that models the output of the black box:

$$p(y^w|\mathbf{x}, m(y^{bb}), w) \sim Dir(\boldsymbol{\alpha}). \tag{23}$$

These α parameters are decomposed in:

$$\boldsymbol{\alpha} = \beta y^{bb}, \tag{24}$$

where y^{bb} belongs to the output of the original black-box classifier, and β corresponds to a parameter that regulates the variance of the data and is estimated by training a deep learning wrapper. In [61], Mena et al. extend this method by including an out-of-distribution term in the loss that allows detection of potential shifts in the distribution of the prediction data. In this case, the method is showcased using classification systems of both natural language processing and computer vision applications

To wrap up, Table 1 contains a summary of the different methods analyzed including the type of uncertainty studied in each. It also provides a categorization of the methods based on some scalability issues reported. Thus, we can see how methods like Laplace approximations, [56, 57], where there is the need to approximate the Hessian matrix, Monte Carlo Markov Chains methods, [66], that present convergence problems when the number of parameters increases, or initial Variational Inference methods, [39], that assume linear models or small architectures, display the highest level of scalability as they do not easily adapt to modern Deep Learning architecture, where the number of parameters can reach 8 or 9 figures. For Variational Inference methods optimized using Stochastic Gradient Descent approximations, we consider that scalability would depend on the complexity of the architecture, as it would require modification of the layers to introduce the variational equivalents. Finally, we consider that methods based on ensembles, [48], or bootstrapping, [68], or those based on the use of Dropout layers, do not present large scalability issues beyond those inherent in the original Deep Learning models they are based on. Additionally, we have included a column detailing whether the article analyzes desirable properties that any uncertainty modeling method should consider

Table 1. Summary of uncertainty estimation methods. Scalability issues of each method are reported as LOW, MID, HIGH and desirable properties reviewed are highlighted using the following icons, ♣ Calibration ♠ Robustness ◇ Interpretability ♡ Explainability

Family	Method	Refs.	Type of uncertainty	Scalability	Props.	Section
Laplace	Bayesian Neural Networks using Laplace approximation	[56, 57]	Epistemic	HIGH		4.1.1
MCMC	Markov chain Monte Carlo approximations	[66]	Epistemic Aleatoric	HIGH		4.1.2
	Stochastic gradient Hamiltonian Monte Carlo	[10]	Epistemic Aleatoric	MID	♠	4.1.2
Variational Inference	Minimum Description Length Principle	[39]	Epistemic Aleatoric	HIGH		4.1.3
	Ensemble Learning	[1]	Epistemic Aleatoric	MID		4.1.3
	Variational Inference	[34]	Epistemic	MID		4.1.3
	Stochastic Variational Inference	[40]	Epistemic	MID	♠	4.1.3
	Bayes by Backprop	[5]	Epistemic	MID		4.1.3
	Stochastic gradient variational Bayes for gamma approximating distributions	[47]	Epistemic	MID	♡	4.1.3
	The generalized reparameterization gradient	[73]	Epistemic	MID		4.1.3
	Implicit reparameterization gradients	[25]	Epistemic	MID		4.1.3
	Pathwise derivatives beyond the reparameterization trick	[43]	Epistemic	MID		4.1.3
	Flipout	[89]	Epistemic	MID		4.1.3
	Variational Dirichlet Framework	[11]	Aleatoric Distributional	MID		4.2.2
Expectation Propagation	Expectation propagation	[38, 52]	Epistemic	MID	♠	4.1.4
	Lightweight probabilistic deep networks (LPN)	[31]	Epistemic Aleatori	MID	♠	4.2.2
	A general framework for uncertainty estimation in deep learning	[53]	Epistemic Aleatoric	MID		4.2.2
	Predictive Uncertainty Estimation via Prior Networks	[58]	Epistemic Aleatoric Distributional	MID		4.2.2
	Understanding DNN through input uncertainties	[85]	Aleatoric	MID	♡ ◇	4.2.2
Monte Carlo Dropout	Monte Carlo Dropout	[29]	Epistemic	LOW	♠	4.1.5
	Aleatoric Monte Carlo Dropout	[28, 46]	Aleatoric	LOW	♠ ◇	4.2.1
	Neural Network Regression with Beta, Dirichlet, and Dirichlet-Multinomial Outputs	[76]	Aleatoric	MID	♠	4.2.2
	Dirichlet Uncertainty Wrapper	[61]	Aleatoric Distributional	LOW	♣ ♠	4.2.2
Ensemble and bootstrapping	Network Ensemble	[48]	Epistemic	LOW	♠	4.1.6
	Bootstrapped DQN	[68]	Epistemic	LOW		4.1.6

5 SUMMARIZING UNCERTAINTY

In the previous section, we reviewed some works designed to approximate the posterior over the weights and the outcome of Deep Learning classification systems. By applying Bayesian inference, these methods manage to turn traditional point estimates of Deep Learning models into random variables following probability distributions. This transformation enriches system outcomes and allows practitioners to make more informed decisions based on the predictions of these classification systems. However, this constitutes only the first step in uncertainty modeling. For these outcomes to become actionable, there is a need to define a way of summarizing this uncertainty based on these probability distributions. In the case of regression problems, for example, if the output of the model is a Gaussian random variable, the scale parameter of the distribution can be used as a measure of this uncertainty.

5.1 Summarizing uncertainty for probabilistic classifiers

Using the scale parameter of the distribution as a measure of uncertainty in classification problems is not that straightforward, however. In some of the reviewed approaches [46] we have seen that the random variables model the logits of the network's output. In others, such as [31], the random variables directly model the prediction's probability distribution, but there remains the need to define how to obtain a numerical score from these random variables. In this section, we review different ways of obtaining this uncertainty score from the corresponding probability distributions.

In classification systems, the subject of this work, and specifically in probabilistic classifiers, the output of the model already has the shape of a probability distribution for the predicted classes, resulting from the application of a softmax operation at the output logits. Thus, a first approach to obtaining a measure of the uncertainty is by observing the output distribution itself. At this point, we wish to highlight two points. On the one hand, that even when the probability associated with the predictions is high, it can correspond to a confident but erroneous prediction. And on the other, that the output of a DL might suffer from miscalibration, as detailed in [35].

Assuming a calibrated and correct prediction, Figure 3 illustrates the following metrics derived directly from observing the predictions of the original model:

- Softmax response: in probabilistic classifiers, it corresponds to the probability of the predicted class.
- Predictive entropy: the difference between all predictions, understood as the entropy of the softmax probability distribution, as defined by information theory [19].

$$U_{x^*} = \frac{-\sum_y P_{\mathbf{W}}(\mathbf{y}_1^*|\mathbf{x}^*)\log_2 P_{\mathbf{W}}(\mathbf{y}_1^*|\mathbf{x}^*)\Big)}{\log_2(n)} \tag{25}$$

- Least confidence: the difference between the most confident prediction and 100% confidence, as defined in [18].

$$U_{x^*} = \frac{n\left(1 - P_{\mathbf{W}}\left(\mathbf{y}_1^*|\mathbf{x}^*\right)\right)}{n-1} \tag{26}$$

- Margin of confidence: difference between the top two most confident predictions.

$$U_{x^*} = 1 - \left(P_{\mathbf{W}}\left(\mathbf{y}_1^*|\mathbf{x}^*\right) - P_{\mathbf{W}}\left(\mathbf{y}_2^*|\mathbf{x}^*\right)\right) \tag{27}$$

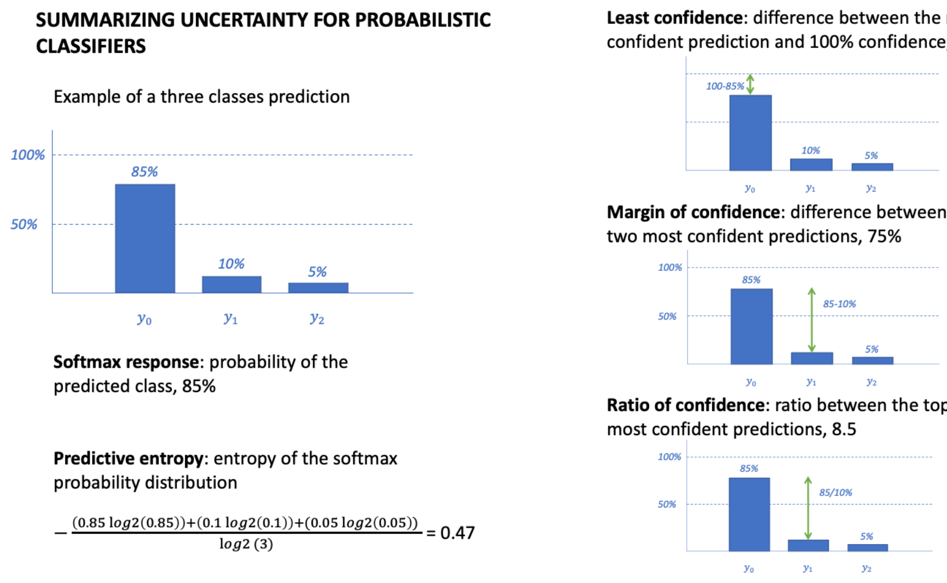
- Ratio of confidence: ratio between the top two most confident predictions, as defined in [77], understood as the margin and ratio respectively between the first and second more confident output units, respectively.

$$U_{x^*} = \frac{P_\theta\left(\mathbf{y}_2^*|\mathbf{x}\right)}{P_\theta\left(\mathbf{y}_1^*|\mathbf{x}\right)} \tag{28}$$

5.2 Summarizing uncertainty for Bayesian Deep Learning classifiers

All of the metrics introduced so far, obtain an uncertainty score from point estimates. However, taking advantage of the posteriors modeled in the previous section rather than the point estimates, we can now use a proper output probability distribution. The literature [28] contains different ways of obtaining a numerical uncertainty score by summarizing these distributions:

Fig. 3. How to obtain an uncertainty measure from the output of probabilistic classifiers


SUMMARIZING UNCERTAINTY FOR PROBABILISTIC CLASSIFIERS
Example of a three classes prediction
100%
50%
85%
10%
5%
y_0
y_1
y_2
Softmax response: probability of the predicted class, 85%
Predictive entropy: entropy of the softmax probability distribution
$-\frac{(0.85\ log2(0.85))+(0.1\ log2(0.1))+(0.05\ log2(0.05))}{log2\ (3)} = 0.47$
Least confidence: difference between the most confident prediction and 100% confidence, 15%
100-85%
Margin of confidence: difference between the top two most confident predictions, 75%
85-10%
Ratio of confidence: ratio between the top two most confident predictions, 8.5
85/10%

Variation ratios measures the variability of the predictions obtained from sampling [27] by computing the fraction of samples with the correct output. This heuristic is a measure of the dispersion of the predictions around its mode. In this respect, for a given data point x we can sample M output vectors from the output distribution y_x^t, $t = 1 \ldots M$, and compute the variation ratios as follows,

$$VR = 1 - \frac{f_{c^*}}{M} \tag{29}$$

where $f_{c^*} = \sum_t 1[\mathbf{y}_x^t = c^*]$, and c^* corresponds to the sampled majority class,

$$c^* = \arg \max_{c=1,\ldots,C} \sum_{t=1}^{M} 1[\mathbf{y}_x^t = c]. \tag{30}$$

Predictive entropy considers the average amount of information contained in the predictive distribution [80]. Results with low entropy values correspond to confident predictions, whereas high entropy leads to significant uncertainty. This score takes into account the variability of the predicted value by sampling what we call the sampled predictive entropy from the distribution inferred by the wrapper, defined as

$$\mathbb{H} = -\sum_c \mathbb{E}[\mathbf{y}^*]_c \log \mathbb{E}[\mathbf{y}^*]_c. \tag{31}$$

Mutual information (MI) considers the relationship between two random variables, X, Y and measures the mutual dependence between the two. Again, it is related to Information theory, as it can be understood as the amount of information obtained about one random variable by observing the other. MI determines how different the joint distribution of the pair (X, Y) is to the product of the marginal distributions of X and Y:

$$I(X;Y) = D_{\mathrm{KL}}\left(P_{(X,Y)} \| P_X \otimes P_Y\right) \tag{32}$$

Where $\mathcal{KL}$ divergence is used to measure this difference, which when considering continuous variables translates to:

$$\mathrm{I}(X;Y) = \int_{y}\int_{X} p_{(X,Y)}(x,y)\log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)}\right)dxdy \tag{33}$$

Or viewed from the Information theory perspective:

$$\mathrm{I}(X;Y) = \mathrm{H}(Y) - \mathrm{H}(Y|X) \tag{34}$$

When measuring uncertainty in neural network models, the two variables to consider are the prediction y^* and the posterior over the model parameters $\mathbf{W}$. Usually, this metric relates more to epistemic uncertainty, as it compares the posterior of $\mathbf{W}$ with Y, as approximated in [28]:

$$\mathbb{I}\left[Y, \mathbf{W}|X, \mathcal{D}_{\text{train}}\right] \approx \underset{T\to\infty}{\longrightarrow} \mathbb{H}\left[Y|X, \mathcal{D}_{\text{train}}\right] - \mathbb{E}_{q_\theta^*(\mathbf{W})}\left[\mathbb{H}[y|X, \mathbf{W}]\right] \tag{35}$$

by sampling T models for the posterior $q_\theta^*(\mathbf{W})$. In [42], Houlsby et al. state that test points $\mathbf{x}^*$ that maximize mutual information are points on which the model is uncertain on average, as the model parameters $\mathbf{W}$ erroneously produce predictions with a high degree of confidence.

Additional methods for summarizing uncertainty are present in some of the works that make use of Dirichlet distributions reviewed in the previous section [11, 58].

Chen et al. [11] propose using the entropy of the resulting Dirichlet output distribution:

$$E(\alpha) = -C(\alpha) = -\int_z z\,\mathrm{Dir}(z|\alpha)dz = \log B(\alpha) + (\alpha_0 - K)\,\psi(\alpha_0) - \sum_i^k (\alpha_i - 1)\,\psi(\alpha_i) \tag{36}$$

The final uncertainty score corresponds to the negative of this entropy. This work also analyzes the resulting α parameters. The authors observe how these parameters tend to take large values $\alpha >> 1.0$, compromising the method's performance. To prevent such problems, the article suggests a smoothing factor $\hat{\alpha} = \log(\alpha + 1)$, while other smoothing functions can also be applied.

Finally, Malinin and Gales [58] propose different metrics of uncertainty. Let us recall that three different types of uncertainty are considered in this article: model, data and distributional. These different types can be calculated according to the type of marginalization applied to Eq. 20. To obtain the total uncertainty, the paper proposes the use of *max probability*, $\mathcal{P} = \max_c \mathrm{P}\left(\omega_c|\boldsymbol{x}^*; \mathcal{D}\right)$, and predictive entropy. For other uncertainties, it proposes to use mutual information between the total uncertainty and each individual type of uncertainty.

6 APPLICATIONS OF UNCERTAINTY

In this section, we analyze potential applications of uncertainty in classification tasks. Beyond the interpretability of the predictions introduced in the previous section, in the literature we find many examples of how to use the different types of uncertainty to help improve the quality of Machine Learning models. Using epistemic uncertainty, we will see how it is possible to estimate the extent to which a model adapts for a given problem or how to select new examples that boost training speed. We will also observe how aleatoric uncertainty can be used to discard examples that are more uncertain, increasing the quality of the preserved predictions or permitting the intervention of experts to decide on those specific cases.

6.1 Model Selection

A first application of uncertainty consists in model comparison or selection. When designing Deep Learning models, the choice of architectures, parameters and regularizers can be cumbersome. Working with held-out data or cross-validation to optimize these hyper-parameters imply sacrificing some of the training data available for this purpose. Such a sacrifice has led researchers to include optimization of these hyper-parameters in the model training. To this end, the use of epistemic uncertainty allows models to be compared effectively and includes the implicit optimization of hyper-parameters and regularizers.

Some of the first research to introduce this use of model uncertainty was conducted in [55, 56]. Here, the Bayes' theorem is applied so as to model the probability of the data given the model parameters and regularizers, $p(D|\mathbf{W}, \mathcal{R})$. By estimating this posterior to find its maximum a posteriori, it was possible to find the most optimal combination of parameters and regularizers

In [56], the model uncertainty was estimated in an additional step of the inference process called model selection. Later works such as Variational Inference, Expectation Maximization or Network Ensembles propose including the uncertainty in the training process. By considering the weights as random variables, these methods obtain not only one valid model but a collection of infinite (or finite in the case of ensembles) models that maximize the posterior of those models given the training data.

In [28, 46], the use of Dropout is proposed to sample models $\mathbf{W}$, also tackling the optimization of the regularizers by using aleatoric uncertainty in what is known as loss attenuation.

After modeling the uncertainty of the model by choosing one of the approaches presented in Section 4.1, it would be possible to sample parameters $\mathbf{W}^*$ from the distributions obtained. This sampling will allow the expectation of the output $\mathbb{E}_{p(\omega|\mathcal{D}_{\text{train}})}[Y|X, \omega]$ to be estimated. With this expectation, it is possible to then estimate the mutual information between the prediction Y and the posterior over the model parameters $\mathbf{W}$. As illustrated in [28], mutual information captures the model's confidence in its output, this constituting an excellent way of comparing the performance of different models for a given problem

6.2 Active Learning

Another recurrent application of uncertainty present in the literature on Machine Learning is Active Learning (AL) [13]. The goal of AL is to boost the training process by selecting meaningful and representative examples that allow the model to learn faster. This task is especially relevant in constrained scenarios, where obtaining training data requires the intervention of human experts to label the samples, implying extra effort in terms of time and cost. In such scenarios, it is very desirable to have a criterion for selecting new samples to label, as it speeds up training of the model.

Different approaches have been posited to tackle Active Learning, as surveyed in [79]. Among these, the simplest is uncertainty sampling [51]. By observing the output confidence or predictive entropy of the so-far-trained model when applied to unlabeled points, it is possible to select the more uncertain ones, these being expected to be the more challenging ones for the model, thus fostering the training process.

The problem with this approach is that sampling from the more uncertain unlabeled points is usually equivalent to sampling from the decision boundary of the classifier. As David Mackay suggested in [56]: "The strategy of sampling on decision boundaries is motivated by the argument that we are unlikely to gain information by sampling in a region where we are already confident of the correct classification. But similarly, if we have already sampled a great deal on one particular boundary then we do not gain useful information by repeatedly sampling there either, because the location of the boundary has been established!"

MacKay proposes another way to use uncertainty for Active Learning. The main idea is to divide the unlabeled dataset in what he calls regions of interest and obtain labels for points that maximize the mean marginal entropy. Intuition tells us that those points with a higher variance according to the entropy of the predictions obtained by drawing samples from the posterior over the models will translate to more representative points and better Active Learning.

Recently, this concept of regions of interest has been extended to the context of Thompson sampling (TS) and multi-armed bandits. In [75], Russo et al. propose using ensembles as the arms and include TS in the selection of new training inputs. In [6], Bouneffouf et al. propose creating clusters of the unlabeled points and considering each of these clusters as the arms. By applying TS, it is then possible to choose examples based on a reward function that looks at the cosine distance of a vector comprising the points before and after adding the selected input.

Other approaches to Active Learning are those based on variance reduction and Fisher information Ratio, as in [92]. In this case, they comprise a pool-based method that looks at the ratio $X^*_{FIR} = \underset{X}{\operatorname{argmin}} \operatorname{tr}\left(\mathcal{I}_X(\theta)^{-1}\mathcal{I}_{\mathcal{U}}(\theta)\right)$ between the selected point and the rest of the unlabeled examples. The Fisher information ratio is a measure of epistemic uncertainty that, in addition to the uncertainty in the model, also measures which model parameters are most responsible for this uncertainty.

In general, based on the uncertainty in the model, different strategies select new samples to label based on different criteria: reducing the expected error or maximizing the variance of the model or how much it changes after adding the sample to the training.

Regarding Active Learning, the use of aleatoric uncertainty is not indicated, as the same conclusions extracted from the selection of points in the decision boundaries would apply here. Aleatoric uncertainty measures the noise inherent in the data, so selecting points that are of themselves noisy would not help to improve the variance of the model or the expected error, as this type of uncertainty will not be reduced with more training, unlike what happens with epistemic uncertainty.

6.3 Classification with rejection

Classification systems with rejection comprise another application where the role of uncertainty is key. The process of not issuing or of discarding a prediction when the system is not confident enough can be found under different names in the machine learning literature, including Rejection Methods, Selective Classification, Abstention, or Three-Way Classification. The most common approach establishes a threshold based on a given metric and discards predictions accordingly. Following the ideas proposed in the seminal work [12], where the authors establish a threshold for rejection with which to minimize the classification risk, other works like [15, 20, 23, 32] consider different cost-based rejection models based on the output responses of deep learning models.

Alternatively, [2, 16, 20] include a term for the rejector in the loss that is learned together with the classifier. Following a similar approach, in [86], Thulasidasan et al. propose omitting noisy labels in order to improve the training process. These works share the approach of considering a fixed cost for the abstention, in which the classifier incurs a fixed cost every time the abstain option is invoked. In contrast, in [81], Shekhar et al. establish a fixed fraction of input samples, δ, in which the learner is allowed to abstain without incurring any costs. The paper proposes a plug-in classifier that employs unlabeled samples to decide the region of abstention and derive an upper-bound on the excess risk. Another strategy, followed by the authors in [33], is to train the model together with a particular loss function for rejection. However, in this case they fit a model specifically for each degree of coverture, chosen beforehand.

Finally, evaluations of the performance of classification with rejection models usually use standard metrics such as accuracy or the F1-score to obtain accuracy-rejection curves (ARC) [65] or the 3D receiver operating characteristic [49]. Other works focus on the rejection of multi-label classification systems [69]. These approaches present limitations, as they are not able to determine the optimal rejection rate by comparing the performance of the classifiers. Beyond the ROC space, some authors [36] have analyzed different representation spaces to build a rejection system, mainly the cost-reject (CR) and the error-reject (ER) space.

Aligned with this error-rejection approach, we find [14], where Condessa et al. propose three different performance measures for evaluating the best rejection point that overcomes the previous restrictions. In order to evaluate the rejection metric, they divide the dataset using two criteria: whether the method **R**ejects the data point or **N**ot; and whether the point is **A**ccurately classified, or **M**issclassified, called R, N, A or M respectively. Three quality metrics can be derived from these values (illustrated in Figure 4):

- **Non-rejected Accuracy** measures the ability of the classifier to classify non-rejected samples accurately. It is computed as follows,

$$NRA = \frac{|A \bigcap N|}{|N|}$$

- **Classification Quality** measures the ability of the classifier with rejection to classify non-rejected samples accurately and to reject misclassified samples. It is computed as follows,

$$CQ = \frac{|A \bigcap N| + |M \bigcap R|}{|N| + |R|}$$

- **Rejection Quality** measures the ability to concentrate all misclassified samples onto the set of rejected samples. It is computed as follows,

$$RQ = \frac{|M \bigcap R|\,|A|}{|A \bigcap R|\,|M|}$$

An optimal rejection point will show a trade-off between the three metrics, it being possible to divide misclassified predictions from correct ones and preserve only those points that provide useful information. The higher the value displayed, the better that metric performs for rejection.

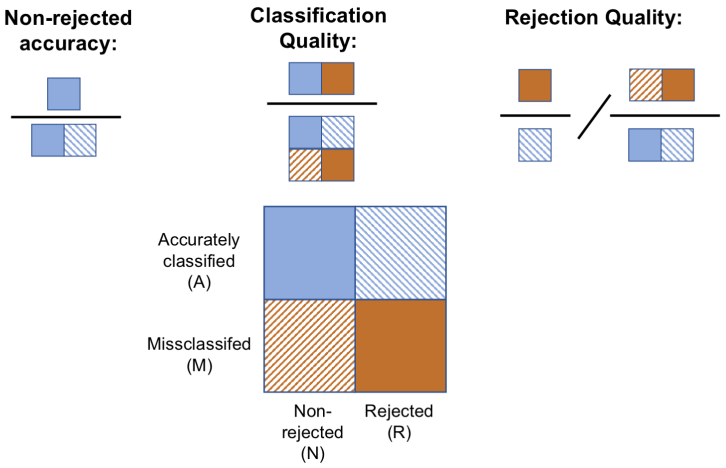
In a rejection setting, the model is enhanced with a new class, the *rejection* class. Thus, the final prediction becomes

$$y^* = \begin{cases} f(\mathbf{x}), & \text{if } \phi(\mathbf{x}) < \tau \\ reject, & \text{if } \phi(\mathbf{x}) \geq \tau \end{cases}$$

Here, $f(\mathbf{x})$ is the classifier without rejection, the function $\phi(\mathbf{x})$ is a function of the input that evaluates the confidence of the prediction model, and τ is a threshold value that indicates the rejection point. Those predictions with lower confidence than the given threshold are rejected, whereas the more confident ones are predicted using the original classifier. In probabilistic classifiers, risk can derive from the observation of the output probabilities employing different metrics, like Least Confidence [18], Margin of Confidence [77] and Variation Ratios and Predictive Entropy, as proposed in [28].

Moreover, an option for automatizing selection of the rejection point could be to use a method like [32], where Geifman and El-Yaniv employ a binary search to optimize selection.

Fig. 4. Rejection performance metrics as proposed in [14]



7 DISCUSSION AND FUTURE WORK

7.1 Review Methodology

The aim of the present survey is to provide a summary of the literature on methods for modeling posterior distributions for estimating and measuring uncertainty. The survey follows the evolution of these methods over time. We have attempted to illustrate how these methods have evolved along with the complexity of Deep Learning models and the availability of computational resources. The methodology used for this compilation followed a two-step process:

- First, we identified the main approaches used for the task: from Laplace approximations through variational approaches or the use of modern architectures like Dropout layers.
- Next, we applied the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) framework guidelines [63]. Following this framework, we looked for references of articles for each category, following the citations found in Google Scholar, ACM Digital Library, Springer, Elsevier and the IEEE Xplore databases, among others.

7.2 Overview of the Field

We have seen how uncertainty estimation requires two steps. First, by applying a Bayesian inference, it requires the approximation of the posterior as defined in Equation 5, to transform point estimates of the classifiers into probability distributions. Then, it needs an additional step of summarizing these distributions to produce a numerical score from the probability distributions obtained

In the papers reviewed in Section 4, we have seen different approaches to approximating the integral in Equation 5. Section 5 contains different strategies for summarizing the probability distributions to obtain a degree of confidence in the predictions. Even though the selection of these strategies designed to obtain a numerical score is very relevant to estimating uncertainty, approximation of the posterior in Equation 5 is the critical step.

Initially, efforts focused on approximating the integral analytically [56], either by sampling model parameters [66] or substituting it with tractable posteriors [39]. The problem with these initial works is their lack of adaptation to modern Deep Learning architectures, where the number of parameters might be huge

To deal with the high number of parameters and large datasets inherent in modern Deep Learning, the work in [34] developed the method known as Stochastic Variational Inference (SVI) by using a stochastic optimization process to estimate the posterior. This method was later extended in the so-called Bayes by Backprop method described in [5], where Blundell et al. introduce the use of the reparametrization trick to handle Gaussian random variables as regular parameters in the derivation of the gradients of the loss function. Recently, new works inspired in Stochastic Variational Inference have addressed different constraints of the original method, like including new versions of the reparametrization trick to support other distribution rather than just Gaussian random variables.

In parallel with Variational Inference, the work in [38] uses expectation maximization to tackle the scalability problems of Bayesian Neural Networks. By using assumed density filtering, the authors injected noise into the inputs and propagated this noise across the network, learning the parameters of distributions for weights that maximize the output probability. Recently, this work has been extended to Convolutional Neural Networks in [31], providing variational implementations for all layers of the architecture and using a Dirichlet output distribution.

Different approaches have been posited using the Ensemble [48] and Bootstrapping [68] methods, which do not propose altering the architecture of Deep Learning models, but rather measure the disagreement among different versions of these models. By training models with different initializations (ensemble) or different partitions of the training data (bootstrapping), these approaches can estimate the uncertainty of the models' predictions. Similarly, Monte Carlo Dropout [29] generates these different versions of the same model by taking advantage of Dropout layers, present in many modern Deep Learning models.

When looking at the application of uncertainty, we observe that the most common way of estimating it is still to analyze the softmax response or predictive uncertainty of the output, together with calibration techniques that aim to turn this output into meaningful probabilities.

Beyond observation of the output confidence, a model that is often included as a baseline to compare with new models or in the literature on applications like Active Learning [30, 54, 79], or rejection systems [33] is that known as Monte Carlo Dropout. Recently, Ensemble methods [70] have also become more relevant, perhaps because of their ease of implementation, only requiring minor changes to existing models and robustness [45, 83].

In addition, many recent works have also included Stochastic Variational Inference in comparisons between available models. This may be due to the addition of methods like Flipout [89] in popular Deep Learning libraries like Tensorflow Probability footnotehttps://www.tensorflow.org/probability or implementations of Stochastic Variational Inference included in Edward [1], which greatly simplify the application of this type of Bayesian models.

7.3 Open Challenges

Despite all the efforts analyzed in previous sections, estimating uncertainty in classification systems is still an active research topic. Some of the challenges that remain relate to the desirable properties that any uncertainty estimation model should bear in mind, such as calibration, robustness, explainability, and interpretability.

7.3.1 Calibration. As mentioned in Section 5, the most straightforward way of measuring uncertainty in classification systems is to observe the probability associated with the predicted class. Take, for instance, a binary classifier — a prediction of 0.9 would mean that the classifier is confident in assigning a positive prediction. In contrast, a prediction of 0.51 would mean that the uncertainty of the prediction is high.

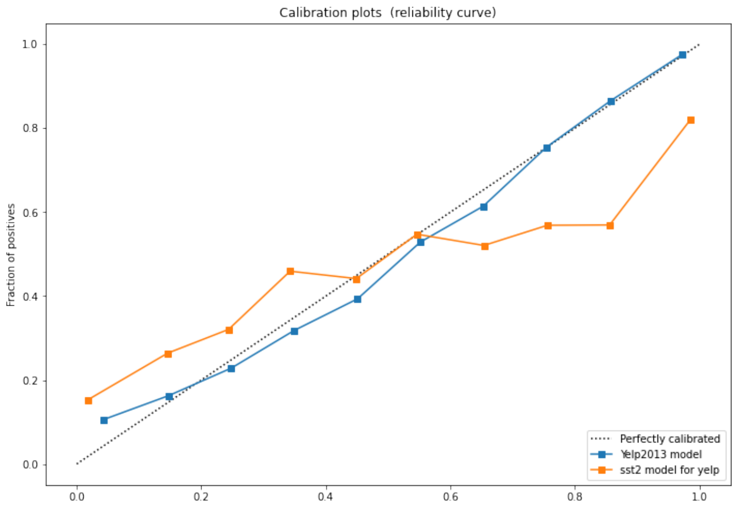
[1]http://edwardlib.org/

In order to use these output probabilities to measure their level of confidence, one key aspect is the fact that they need to be calibrated. Usually, the output of a classifier derives from the application of a sigmoid or a softmax function. These functions transform the output into values from 0 to 1. In the case of softmax, by normalizing the outputs, the results describe sets of C values, one for each class, which summed together add up to 1, resembling a probability distribution.

To consider a classification system as calibrated, its output must match the expected distribution of probabilities for each class. The problem is that most of the time this is not the case, especially with modern Deep Learning architectures. As reported by Gast and Roth in [31], the more complex the architecture, the less calibrated the output will be. The miscalibration problem also extends to other types of predictive systems, as analyzed in [67], like Support Vector Machines, Random Forests or boosted trees.

The main method used to measure the calibration of a classification system is the reliability diagram, also known as the calibration curve. A further analysis of reliability diagrams can be found in [8], but the main idea is that a proper output probability distribution should adequately reflect the actual likelihood of the predicted data. For example, given 100 predictions, each with a probability of 0.8, we expect 80 to be correctly classified. Figure 5 shows an example of reliability diagrams.

Fig. 5. Example of reliability diagrams



Even though the reliability diagram is a handy tool for diagnosing calibration issues, a numerical score is sometimes required. One of the most common metrics to this end is Expected Calibration Error (ECE). Very much related to reliability diagrams, ECE measures the difference between the average accuracy and confidence, as calculated in the diagrams. A variation of this measure is the Maximum Calibration Error (MCE), which computes the maximum rather than the average. Furthermore, the Negative Log-Likelihood of the model can also sometimes be used.

[31] describes the main methods for tackling calibration issues, i.e. isotonic regression, Platt scaling and a variation of the later temperature scaling.

Calibration is an essential property of classification systems, especially when evaluating the output probabilities to measure the confidence of predictions. As reported in [31, 67], methods that rely on measuring probabilities, like logistic regression, seem to be more calibrated than others.

Thus, when developing methods that transform the outputs of classifiers as a random variable, we should expect the outputs of such systems to be better calibrated than the original ones.

In this regard, the work described in [46] is interesting, where the authors analyze the calibration of uncertainties obtained for both regression and classification computer vision problems. Moreover, [61] analyzes the relationship between calibration and uncertainty estimation for NLP problems, verifying that previous calibration of the inputs does not have a big impact on estimation of the corresponding uncertainty.

7.3.2 Robustness. Another aspect that any uncertainty modeling technique must bear in mind is the robustness of its metric. By robustness of an uncertainty metric, we mean its capability to adapt to new data distribution in the prediction phase. These data distributions might differ from the original ones used to train the models, but the uncertainty estimator must continue to efficiently detect uncertainty.

In respect of this, we find the work presented in [83] to be very relevant to the robustness of uncertainty methods. Here, Snoek et al. analyze different uncertainty metrics and subject them to different types of distortion in the original training inputs, including a study of their behavior in relation to out-of-distribution data.

The methods analyzed in the above study include a vanilla implementation based on the Maximum softmax probability [37], the same metric but after applying the Temperature scaling calibration method introduced in the previous section, Monte Carlo Dropout [29], Stochastic Variational Inference [5], ensemble methods [48] and a last-layer-only variation of both Dropout and Stochastic Variational Inference [71].

To measure the quality of the uncertainty modeled, they use the metrics detailed in Section 7.3.1, namely the Expected Calibration Error and Negative Log-Likelihood, plus the Brier Score, which measures the accuracy of predicted probabilities as the squared error of a predicted probability vector, $\mathbf{y}^*$, and the one-hot encoded true response $\mathbf{y}$.

Snoek et al. include experiments that use datasets from different domains, i.e. image and text classification, tabular data or even genomics. Each dataset is composed of the original data plus two additional variations: one with gradual levels of distortion, and another with a completely different distribution to test the out-of-distribution case. In this latter case, as labels do not apply, the article includes a plot for the distribution of the predictive entropy in order to observe the detection capability of the model with regard to out-of-distribution points.

Snoek et al.'s paper provides the reader with some relevant insights when talking about the robustness of uncertainty measures. First, regardless of the method considered, uncertainty accuracy and quality both consistently degrade when the domain data distribution shifts. Although some methods, like Temperature scaling, seem to work well with test data, they do not adapt to shifts in the data distributions. For epistemic models, the article highlights Ensemble as the best performing, while indicating that even though Stochastic Variational Inference seems to work well on small datasets, it is hard to apply to datasets that require complex architectures.

All in all, from Snoek et al.'s analysis we can derive that modeling uncertainty for a given problem or dataset does not mean that this model adapts well to changes on the data once it is deployed in a real environment. At this point, we would like to highlight a couple of aspects relating to the present work.

The first thing to consider is that, except for Vanilla and Temperature scaling, the included methods all addressed epistemic uncertainty, measuring how the model adapts to the data, not the uncertainty of the data itself. Training the original models with data points from the additional distorted datasets would reduce this uncertainty. In addition, by way of example, after rotating an image that represents a horse, the image will still represent a horse. If the picture represents a horse appropriately, the associated aleatoric uncertainty will be low. The problem is when the picture of this horse is labeled as a dog. Then the model will be very confident predicting it as a horse, so epistemic uncertainty will be low, whereas aleatoric uncertainty will be high.

Furthermore, in its analysis, Snoek et al.'s article explicitly discards methods that specifically train for out-of-distribution detection. As mentioned above, they analyze the uncertainty of out-of-distribution points by observing the resulting predictive entropy. Looking at the results reported for the text models, since out-of-distribution is only reported for this kind of model, we observe how, except for the last-layer-only versions, all the methods used, including Vanilla, already report out-of-distribution points with high uncertainty. As found with the authors in [58], epistemic and distributional uncertainties are viewed as separate concepts, and they must therefore be addressed separately.

Nonetheless, the work presented in [83] delivers an appealing framework for evaluating uncertainty in scenarios that might mimic what one can expect when applying classification models to production, where data distributions may shift or new types of examples might appear.

Another interesting initiative that works on measuring how different uncertainty models deal with real datasets is the Bayesian Deep Learning Benchmarks framework or BDLB[2]. This is a framework for evaluating inference tools and for measuring how these tools scale to real-world settings. The idea is to provide diverse types of benchmarks, i.e. real-world downstream tasks, that allow for the comparison of different Bayesian Deep Learning implementations. To compare these, the framework provides well-tuned baselines and benchmarks with each model against existing baseline results and generates plots using expert metrics (such as the Area under the ROC Curve of retained data when referring 50% of most uncertain patients to an expert). In its current version, the framework supports the following implementations:

- Deterministic
- Monte Carlo Dropout (following [29])
- Mean-Field Variational Inference (following [89])
- Deep Ensembles (following [48])
- Ensemble Monte Carlo Dropout (following [82])

To date, four use cases have been implemented: a Diabetic Retinopathy Diagnosis tool [50], an Autonomous Vehicle Scene Segmentation [64], a Galaxy Zoo [87] and Fishyscapes tasks [4], providing results and a benchmark only for the first of these. In this benchmark, they show how the best scoring methods are Monte Carlo Dropout, Mean-Field Variational Inference and Deep Ensembles. These methods combine both epistemic and aleatoric uncertainties, estimated using predictive risk. However, Standard Dropout, which uses only aleatoric uncertainty, performs worse.

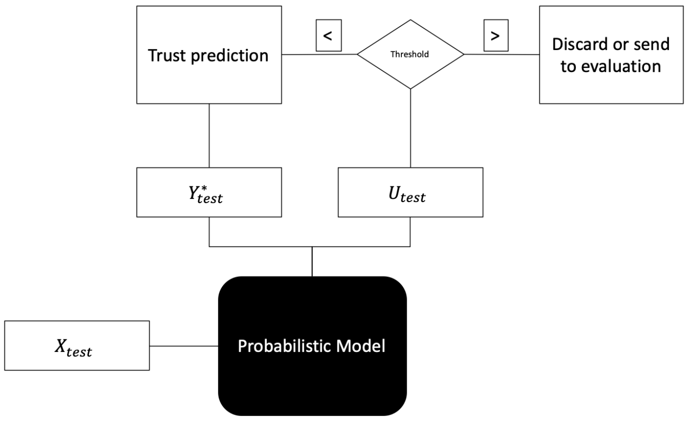
To evaluate the performance of various uncertainty estimation methods using different tasks, the authors adopt a similar approach to the one used in [60], which is based on applying the uncertainty as a rejector.

Figure 6 illustrates the mechanism used in both articles: the uncertainty obtained from the probabilistic model is used as a rejector. Predictions in which uncertainty scores are under a given threshold are taken into consideration, whereas those above are discarded. The authors then evaluate the quality of the uncertainty indirectly, by measuring the quality of the resulting classifications using different metrics: accuracy or AUC.

Beyond rejection-based methods, [60] and in [48] experiment using out-of-distribution examples to evaluate the measured uncertainty. By introducing out-of-distribution points, similar to what they did in [83], they analyze whether the uncertainty estimator can detect those points as very uncertain predictions.

[2]https://github.com/OATML/bdl-benchmarks

Fig. 6. Rejection method for evaluating robustness of uncertainty estimators



All in all, it is clear that robustness is a relevant issue that must be considered when developing and evaluating new uncertainty estimators. Initiatives like the work in [83] analyze the extent to which uncertainty models trained with a given dataset can adapt to changes in the distribution of data once deployed in the wild. From this study, Ensemble is observed to be the only method that shows an overall capacity to adapt to perturbations in the prediction data. The more data-dependent the methods are, the more they overfit the data used to train the uncertainty models. Thus models designed to handle aleatoric uncertainty suffer more from a lack of robustness than those that address purely epistemic uncertainty. Another aspect that can be observed from the study is that there is no all-in-one method. How each one performs depends heavily on the models used and the variability of the data. As reported in an analysis comparing aleatoric and epistemic uncertainties [46], some datasets used to compare uncertainty methods might bias the analysis: *"In practice I found the cifar10 dataset did not have many images that would in theory exhibit high aleatoric uncertainty. This is probably by design"*.

7.3.3 Interpretability. Another crucial aspect when estimating the uncertainty of a classification system is its interpretability. Uncertainty is usually considered to be a source of interpretability regarding a classifier's predictions: uncertain predictions can warn the practitioner about problematic inputs that might require the intervention of a human expert or complementary sources of information, like sensors in a self-driving car.

However, in this section we would like to focus on the interpretability of the uncertainty itself. In Section 5, we analyzed different ways of obtaining a numerical score for measuring uncertainty. Taking the simpler of these methods, the confidence or softmax response of the prediction, we observe that it can be treated as a probability (omitting calibration issues). In this case, the interpretation of the uncertainty is quite straightforward, as humans understand probabilities well [17].

The problem comes with more complex metrics like predictive entropy. Such measures might require further explanations to make them understandable and useful for practitioners. For example, predictive entropy scores will depend on the number of classes predicted, 0.69 being the maximum entropy for a binary classifier and 2.30 for a ten-class problem.

7.3.4 Explainability. Finally, a further property that uncertainty estimation methods should bear in mind is the capacity to explain the uncertainty scores obtained. These scores are summarizations taken from the modeled output probability layers, as detailed in Section 4.2. Their aim is to capture the variance of the distributions that describe the output, based on the given inputs. Depending on the distribution used, variance might be the σ parameter of a Gaussian random variable. In other cases [61], it may be a β parameter resulting from the decomposition of the α concentration parameters of a Dirichlet distribution.

Either way, we can consider this to be a regression problem whose goal is to estimate the scale parameter used to measure variance in each case. As such, we can apply explainability tools to search for explanations regarding which parts of the input contribute more to this uncertainty. Tools like LIME [73], or Shap values [56], among others, can help when attributing uncertainty to words in a text that increase uncertainty in a sentiment analysis tool. Other tools, like the What-if-tool [93], can help in identifying attributes in a tabular dataset that can be changed to diminish this uncertainty.

To the best of our knowledge, no previous work has been published on the explainability of uncertainty estimation models, beyond [85] for regression, and we firmly believe that such research may benefit the application of uncertainty in real problems. Having ways of identifying which elements of our data contribute more to uncertain predictions can enable actionable mechanisms that allow the quality of the predictions to increase by intervening in the data. An example of this might be an image of a horse behind a fence. This kind of pattern might hinder the results of a pre-trained image classifier, increasing the associated uncertainty. What if there was a way of detecting this issue? In this case, we could apply pre-processing to our target dataset images to remove fences (see [44]), thus increasing the quality of the resulting predictions.

Finally, another challenging aspect of uncertainty estimation is the type of problems and models where it is applied. Beyond the study included in [90], which includes experiments with named entity recognition and language modeling tasks, most of the articles that tackle uncertainty in classification tasks include experiments with binary classifiers or a small number of categories. Moreover, the type of architectures used are no more complicated than Recurrent Neural Networks [26], or Convolutional Neural Networks [31]. It is also important to analyze the performance of some methods involving problems with high dimensionality -like automatic translation- with changing outputs -like question answering- or with multiple labels. Additionally, it would be interesting to study how methods behave when dealing with huge modern models like those used in natural language processing tasks such as BERT [22] or GPT-3 [9], which have 175 billion parameters.

8 CONCLUSIONS

Estimating uncertainty in a classification system is essential when the predictions of said system are used for decision-making that may affect human lives. Even when this is not the case, however, having a notion of the uncertainty of predictions is very interesting for a better understanding of the problem, the data and how models behave. In this article, we present a thorough survey of uncertainty estimation applied to classification systems. As well as providing related definitions and describing some of its applications and desirable properties, the current report presents associated studies according to the classification of the approach used. We believe that this study will be beneficial for researchers who wish to understand the role of uncertainty in classification systems and Bayesian Neural Networks in general. Furthermore, the analysis of practical applications and issues related to implementing the different approaches analyzed may also be relevant to practitioners who wish to incorporate a measure of uncertainty in their classification systems.

ACKNOWLEDGMENTS

This work has been partially funded by the Spanish projects PID2019-105093GB-I00, TIN201566951-C2 (MINECO/FEDER, UE), RTI2018-095232-B-C21 and 2017 SGR 1742, and by an Industrial PhD grant from the Government of Catalonia agency AGAUR.

REFERENCES

[1] David Barber and Christopher M Bishop. 1998. Ensemble learning in Bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences* 168 (1998), 215–238.

[2] Peter L. Bartlett and Marten H. Wegkamp. 2008. Classification with a Reject Option Using a Hinge Loss. *J. Mach. Learn. Res.* 9 (June 2008), 1823–1840. http://dl.acm.org/citation.cfm?id=1390681.1442792

[3] Christopher M Bishop. 1995. Bayesian methods for neural networks. (1995).

[4] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2019. The Fishyscapes Benchmark: Measuring Blind Spots in Semantic Segmentation. arXiv:1904.03215 [cs.CV]

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight Uncertainty in Neural Networks. arXiv:1505.05424 [stat.ML]

[6] Djallel Bouneffouf, Romain Laroche, Tanguy Urvoy, Raphael Féraud, and Robin Allesiardo. 2014. Contextual bandit for active learning: Active thompson sampling. In *International Conference on Neural Information Processing*. Springer, 405–412.

[7] Axel Brando, Jose A Rodriguez, Jordi Vitria, and Alberto Rubio Muñoz. 2019. Modelling heterogeneous distributions with an Uncountable Mixture of Asymmetric Laplacians. In *Advances in Neural Information Processing Systems*. 8836–8846.

[8] Jochen Bröcker and Leonard A Smith. 2007. Increasing the reliability of reliability diagrams. *Weather and forecasting* 22, 3 (2007), 651–661.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[10] Tianqi Chen, Emily Fox, and Carlos Guestrin. 2014. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*. 1683–1691.

[11] Wenhu Chen, Yilin Shen, Hongxia Jin, and William Wang. 2018. A Variational Dirichlet Framework for Out-of-Distribution Detection. arXiv:1811.07308 [cs.LG]

[12] C Chow. 1970. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory* 16, 1 (1970), 41–46.

[13] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research* 4 (1996), 129–145.

[14] Filipe Condessa, Jelena Kovacevic, and José M. Bioucas-Dias. 2015. Performance measures for classification systems with rejection. *CoRR* abs/1504.02763 (2015). arXiv:1504.02763 http://arxiv.org/abs/1504.02763

[15] Luigi Pietro Cordella, Claudio De Stefano, Francesco Tortorella, and Mario Vento. 1995. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks* 6, 5 (1995), 1140–1147.

[16] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. 2016. Boosting with Abstention. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 1660–1668. http://papers.nips.cc/paper/6336-boosting-with-abstention.pdf

[17] Leda Cosmides and John Tooby. 1996. Are humans good intuitive statisticians after all? Rethinking some conclusions from the literature on judgment under uncertainty. *cognition* 58, 1 (1996), 1–73.

[18] Aron Culotta and Andrew McCallum. 2005. Reducing Labeling Effort for Structured Prediction Tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2* (Pittsburgh, Pennsylvania) *(AAAI'05)*. AAAI Press, 746–751. http://dl.acm.org/citation.cfm?id=1619410.1619452

[19] Ido Dagan and Sean P. Engelson. 1995. Committee-based Sampling for Training Probabilistic Classifiers. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning* (Tahoe City, California, USA) *(ICML'95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 150–157. http://dl.acm.org/citation.cfm?id=3091622.3091641

[20] Claudio De Stefano, Carlo Sansone, and Mario Vento. 2000. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30, 1 (2000), 84–94.

[21] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural safety* 31, 2 (2009), 105–112.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[23] Ran El-Yaniv and Yair Wiener. 2010. On the foundations of noise-free selective classification. *Journal of Machine Learning Research* 11, May (2010), 1605–1641.

[24] Michael Havbro Faber. 2005. On the treatment of uncertainties and probabilities in engineering decision analysis. (2005).

[25] Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. 2018. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*. 441–452.

[26] Meire Fortunato, Charles Blundell, and Oriol Vinyals. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798* (2017).

[27] L.C. Freeman. 1965. *Elementary applied statistics: for students in behavioral science*. Wiley. https://books.google.es/books?id=r4VRAAAAMAAJ

[28] Yarin Gal. 2016. *Uncertainty in deep learning*. Ph.D. Dissertation. PhD thesis, University of Cambridge.

[29] Yarin Gal and Zoubin Ghahramani. 2015. Dropout as a Bayesian approximation. *arXiv preprint arXiv:1506.02157* (2015).

[30] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1183–1192.

[31] Jochen Gast and Stefan Roth. 2018. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3369–3378.

[32] Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. In *Advances in neural information processing systems*. 4878–4887.

[33] Yonatan Geifman and Ran El-Yaniv. 2019. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. arXiv:1901.09192 [cs.LG]

[34] Alex Graves. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*. 2348–2356.

[35] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, International Convention Centre, Sydney, Australia, 1321–1330. http://proceedings.mlr.press/v70/guo17a.html

[36] Blaise Hanczar. 2019. Performance visualization spaces for classification with rejection option. *Pattern Recognition* 96 (2019), 106984.

[37] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[38] Jose Miguel Hernandez-Lobato and Ryan Adams. 2015. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis Bach and David Blei (Eds.). PMLR, Lille, France, 1861–1869. http://proceedings.mlr.press/v37/hernandez-lobatoc15.html

[39] Geoffrey E Hinton and Drew Van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*. 5–13.

[40] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research* 14, 1 (2013), 1303–1347.

[41] Stephen C Hora. 1996. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety* 54, 2-3 (1996), 217–223.

[42] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745* (2011).

[43] Martin Jankowiak and Fritz Obermeyer. 2018. Pathwise derivatives beyond the reparameterization trick. *arXiv preprint arXiv:1806.01851* (2018).

[44] Sankaraganesh Jonna, Krishna K Nakka, and Rajiv R Sahay. 2015. My camera can see through fences: A deep learning approach for image de-fencing. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE, 261–265.

[45] Alain Jungo and Mauricio Reyes. 2019. Assessing reliability and challenges of uncertainty estimations for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 48–56.

[46] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision?. In *Advances in neural information processing systems*. 5574–5584.
[47] David A Knowles. 2015. Stochastic gradient variational Bayes for gamma approximating distributions. *arXiv preprint arXiv:1509.01631* (2015).
[48] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*. 6402–6413.
[49] Thomas CW Landgrebe, David MJ Tax, Pavel Paclík, and Robert PW Duin. 2006. The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters* 27, 8 (2006), 908–917.
[50] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. 2017. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports* 7, 1 (2017), 1–14.
[51] David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*. Springer, 3–12.
[52] Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 2015. Stochastic expectation propagation. In *Advances in neural information processing systems*. 2323–2331.
[53] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. 2020. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3153–3160.
[54] David Lowell, Zachary C Lipton, and Byron C Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 21–30.
[55] David JC MacKay. 1992. Bayesian interpolation. *Neural computation* 4, 3 (1992), 415–447.
[56] David JC MacKay. 1992. The evidence framework applied to classification networks. *Neural computation* 4, 5 (1992), 720–736.
[57] David JC MacKay. 1992. A practical Bayesian framework for backpropagation networks. *Neural computation* 4, 3 (1992), 448–472.
[58] Andrey Malinin and Mark Gales. 2018. Predictive Uncertainty Estimation via Prior Networks. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7047–7058. http://papers.nips.cc/paper/7936-predictive-uncertainty-estimation-via-prior-networks.pdf
[59] José Mena, Axel Brando, Oriol Pujol, and Jordi Vitrià. 2019. Uncertainty Estimation for Black-Box Classification Models: A Use Case for Sentiment Analysis. In *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, 29–40.
[60] José Mena, Oriol Pujol, and Jordi Vitrià. 2020. Dirichlet uncertainty wrappers for actionable algorithm accuracy accountability and auditability. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 581–581.
[61] José Mena, Oriol Pujol, and Jordi Vitrià. 2020. Uncertainty-based Rejection Wrappers for Black-box Classifiers. *IEEE Access* (2020), 1–1.
[62] Dimitrios Milios, Raffaello Camoriano, Pietro Michiardi, Lorenzo Rosasco, and Maurizio Filippone. 2018. Dirichlet-based Gaussian processes for large-scale calibrated classification. In *Advances in Neural Information Processing Systems*. 6005–6015.
[63] David Moher, Douglas G Altman, Alesandro Liberati, and Jennifer Tetzlaff. 2011. PRISMA statement. *Epidemiology* 22, 1 (2011), 128.
[64] Jishnu Mukhoti and Yarin Gal. 2018. Evaluating Bayesian Deep Learning Methods for Semantic Segmentation. arXiv:1811.12709 [cs.CV]
[65] Malik Sajjad Ahmed Nadeem, Jean-Daniel Zucker, and Blaise Hanczar. 2009. Accuracy-rejection curves (ARCs) for comparing classification methods with a reject option. In *Machine Learning in Systems Biology*. 65–81.
[66] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.
[67] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*. 625–632.
[68] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*. 4026–4034.
[69] Ignazio Pillai, Giorgio Fumera, and Fabio Roli. 2013. Multi-label Classification with a Reject Option. *Pattern Recogn.* 46, 8 (Aug. 2013), 2256–2266. https://doi.org/10.1016/j.patcog.2013.01.035
[70] Remus Pop and Patric Fulop. 2018. Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles. *arXiv preprint arXiv:1811.03897* (2018).
[71] Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127* (2018).
[72] Simone Rossi, Pietro Michiardi, and Maurizio Filippone. 2018. Good Initializations of Variational Bayes for Deep Models. *arXiv preprint arXiv:1810.08083* (2018).

[73] Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. 2016. The generalized reparameterization gradient. In *Advances in neural information processing systems*. 460–468.

[74] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533–536.

[75] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning* 11, 1 (2018), 1–96.

[76] Peter Sadowski and Pierre Baldi. 2018. Neural Network Regression with Beta, Dirichlet, and Dirichlet-Multinomial Outputs. (2018).

[77] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active Hidden Markov Models for Information Extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis (IDA '01)*. Springer-Verlag, London, UK, UK, 309–318. http://dl.acm.org/citation.cfm?id=647967.741626

[78] Robin Senge, Stefan Bösner, Krzysztof Dembczyński, Jörg Haasenritter, Oliver Hirsch, Norbert Donner-Banzhoff, and Eyke Hüllermeier. 2014. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences* 255 (2014), 16–29.

[79] Burr Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[80] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal* 27, 3 (1948), 379–423.

[81] Shubhanshu Shekhar, Mohammad Ghavamzadeh, and Tara Javidi. 2019. Binary Classification with Bounded Abstention Rate. *arXiv preprint arXiv:1905.09561* (2019).

[82] Lewis Smith and Yarin Gal. 2018. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533* (2018).

[83] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. 2019. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*. 13969–13980.

[84] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[85] Jayaraman J Thiagarajan, Irene Kim, Rushil Anirudh, and Peer-Timo Bremer. 2019. Understanding deep neural networks through input uncertainties. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2812–2816.

[86] Sunil Thulasidasan, Tanmoy Bhattacharya, Jeff Bilmes, Gopinath Chennupati, and Jamal Mohd-Yusof. 2019. Combating Label Noise in Deep Learning Using Abstention. *arXiv preprint arXiv:1905.10964* (2019).

[87] Mike Walmsley, Lewis Smith, Chris Lintott, Yarin Gal, Steven Bamford, Hugh Dickinson, Lucy Fortson, Sandor Kruk, Karen Masters, Claudia Scarlata, and et al. 2019. Galaxy Zoo: probabilistic morphology through Bayesian CNNs and active learning. *Monthly Notices of the Royal Astronomical Society* 491, 2 (Oct 2019), 1554–1574. https://doi.org/10.1093/mnras/stz2816

[88] Xizhao Wang and Junhai Zhai. 2016. *Learning with uncertainty*. CRC Press.

[89] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. 2018. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386* (2018).

[90] Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7322–7329.

[91] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. 2018. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 2008–2026.

[92] Tong Zhang and Frank J. Oles. 2000. A probability analysis on the value of unlabeled data for classification problems. In *17th International Conference on Machine Learning*. http://www-cs-students.stanford.edu/~tzhang/papers/icml00-unlabeled.pdf