Journal of the Franklin Institute 362 (2025) 107548

Contents lists available at ScienceDirect

Journal of the Franklin Institute

journal homepage: www.elsevier.com/locate/fi

Shedding light on uncertainties in machine learning: *formal derivation and optimal model selection*

Giulio Del Corso, Sara Colantonio, Claudia Caudai *

Institute of Information Science and Technologies, National Research Council, Pisa, 56127, Italy

ARTICLE INFO

Keywords:
Uncertainty quantification
Bayesian learning
Probabilistic modeling
Deep neural network
Reliable artificial intelligence

ABSTRACT

The concept of uncertainty has always been important in the field of mathematical modeling. In particular, the growing application of Machine Learning and Deep Learning methods in many scientific fields has led to the implementation and use of new uncertainty quantification techniques aimed at distinguishing between reliable and unreliable predictions. However, the novelty of this discipline and the plethora of articles produced, ranging from theoretical results to purely applied experiments, has resulted in a very fragmented and cluttered literature. In this review, we have attempted to combine the well-established mathematical background of the Bayesian framework with the practical aspect of modern state-of-the-art emerging techniques in order to meet the urgent need for clarity on key concepts related to uncertainty quantification. First, we introduced the different sources of uncertainty, ranging from epistemic/reducible to aleatoric/irreducible, providing both a rigorous mathematical derivation and several examples to facilitate understanding. The review then details some of the most important techniques for uncertainty quantification. These methods are compared in terms of their advantages and drawbacks and classified in terms of their intrusiveness, in order to provide the practitioner with a useful vademecum for selecting the optimal model depending on the application context.

1. Introduction

Since ancient times, uncertainty has been seen as synonymous with disorder, and therefore as something to be limited or even eliminated from scientific experiments. In fact, uncertainty is the aspect that makes the calculations inconclusive, leaving us with a feeling of precariousness in the interpretation of an event. The attempt of this review is to reconcile with the concept of uncertainty, to recognize its usefulness, and to realize that it is an indispensable aspect of mathematical modeling anything that has to do with the perceivable and the thinkable. From the earliest philosophical definitions of knowable and uncertainty, through to modern Bayesian formalism or the use of credal sets, there has been an attempt to subdivide the types of uncertainty into more comprehensible classes. In particular, reducible uncertainties (i.e., dependent on the amount of information available) and irreducible uncertainties were introduced, which in turn specialize in a plethora of subclasses not clearly disjointed, such as epistemic uncertainties, model uncertainties, noise, and numerous others [1,2].

With the advent of modern Machine Learning techniques, it became increasingly important to determine how uncertainty in the model inputs (e.g., epistemic variability) changed the prediction (forward uncertainty quantification) and could affect its use. Similarly, it was possible to investigate which variables could play a dominant role in the model (local/global sensitivity analysis) or which were the optimal parameters of a model given information affected by uncertainties (inverse sensitivity analysis). The

* Corresponding author.
E-mail address: claudia.caudai@isti.cnr.it (C. Caudai).

https://doi.org/10.1016/j.jfranklin.2025.107548
Received 7 August 2024; Received in revised form 10 January 2025; Accepted 15 January 2025
Available online 21 January 2025

study of the combination of these problems has given rise to a flourishing line of research in classical Machine Learning called Uncertainty Quantification [3]. However, the advent of Neural Networks, and in particular their deep derivatives, has changed the classic paradigm. The models were no longer tools for understanding the underlying statistical relationships, but powerful predictors subject to high risk of over-fitting that could be used in a wide range of applications, from autonomous driving [4] to automatic identification of the correct Medical Equipment in emergency settings [5], or medical diagnosis [6–8]. Uncertainty analysis, especially in biomedical contexts, has thus taken on a completely different role, with increasing interest in the ability of the model to recognize its own limitations (reliability) or to detect a malfunction in radically different types of data from those used to train it (out-of-distribution analysis).

Many recent papers and reviews have focused on providing details of individual families of methods related to Deep Learning [9–11], proposing alternatives to the probabilistic/Bayesian formulation [12] or offering a systematic description of the application domains of different UQ techniques [13]. However, to the best of our knowledge, few attempts have been made to reconcile the classical formalism derived from the well-structured theory of Bayesian statistics and uncertainty quantification with the formalism typical of supervised Machine Learning (especially Deep). Similarly, it is easy for Machine Learning practitioners wishing to use uncertainty estimation techniques to get lost in the plethora of methods available, and it can be an overwhelming task to work out which method is best suited to the study situation, especially in terms of computational resources required and implementation difficulties.

Therefore, after a historical/mathematical introduction to the evolution of the concept of uncertainty (Section 2), our focus in this paper has been to provide a theoretically rigorous yet illustrative vademecum to guide a practitioner in the selection of the optimal Bayesian ANN model. We are also committed to providing precise nomenclatures and definitions, with the aim of providing disclaimers where the state of the art is sometimes imprecise, vague, or contaminated by transversal concepts. In Section 3 we introduce step-by-step the different kinds of uncertainties (Aleatoric Uncertainty and Epistemic Uncertainty) and further subdivide them into Aleatoric Inherent, Aleatoric Experimental, Aleatoric Model and Epistemic Model, and Epistemic Approximation uncertainties. We also discuss the difference between irreducible and reducible uncertainties and show how the out-of-distribution problem arises as a special case of approximation uncertainty. In Section 4 we briefly introduce the Bayesian formalism and make a comparison between the state-of-the-art techniques for dealing with uncertainty. In particular, we divide the techniques into Intrusive (Section 5: ANN with distributional outputs, Bayesian Neural Networks, Deep Gaussian Processes), Semi-intrusive (Section 7: Ensemble Methods), and Non-intrusive (Section 6: Monte Carlo Dropout, Internal Score, Trust Score, etc.) to help practitioners choose the appropriate method for the task. Given the breadth and complexity of the topic, we decided to move some important but mathematically challenging related topics to the Appendices, such as a formal derivation of variational inference for Bayesian Neural Networks, Markov Chain Monte Carlo posterior estimation, and the theory of Classical Gaussian Processes.

2. An historical introduction: *The road to modern uncertainty conceptualization*

Uncertainty, from disturbing factor to useful modeling component

Since the 17th century, with the efforts of great scientists such as Descartes, Newton, Galileo, and Laplace, there has been an attempt to use mathematics to model reality with the highest possible degree of precision. Several theories were developed, with the intent of modeling frequency probabilities. In particular, in 1763 the Reverend Thomas Bayes developed a powerful theorem for conditional probabilities [14], later used by Bruno De Finetti [15] to define the subjectivist approach to probability. Until the last decades of the 19th century, uncertainty was considered a disturbing factor, to be tried to reduce as much as possible. Over time, however, many studies, including primarily those of the theoretical physicists who, in the first decades of the twentieth century, theorized quantum mechanics, such as Heisenberg, Bohr, Schrödinger, Pauli, etc. have made it clear that uncertainty was an indispensable component for the modeling of physical and biological processes in particular, but more generally of any phenomenon inherent to reality [16,17]. This awareness has begun to decrease the credibility of the analytical infinitesimal calculus, in which few variables are related to each other in a deterministic way, in favor of approaches such as statistical mechanics, in which is much more complicated to eliminate randomness and decide in a deterministic way the relationships between variables. Statistical mechanics made it possible to deal with non-linear problems with a large number of components, taking into consideration the relationships between sensitive indicators of the variables (such as mean and variance) instead of the variables themselves [18]. Towards the first decades of the 20th century, awareness grew about the importance of incorporating uncertainty into mathematical approaches as a necessary element for understanding and modeling reality. Illustrious mathematicians, economists, logicians, and philosophers dedicated themselves to the problem of decoding, recognition, and rigorous formalization of uncertainty (see Fig. 1 for a road map of contributions). Scientists began to understand the inverse correlation between uncertainty and complexity, in the sense that allowing greater uncertainty in a system can contribute to simultaneously decreasing its complexity; this consideration opened the doors to the challenges of determining the optimal level of tolerable uncertainty within a problem of modeling to make it as faithful and accurate as possible [16].

The arduous task of defining what uncertainty is

In 1923 Bertrand Russell published an essay entitled “Vagueness” [19], in which he attempted to define the limits in the perception of reality, both at a sensory level in the perception field and at a representation level through the logical-mathematical instruments. According to Russell, logic is the highest possible tool for representing reality, but faithful representation still remains an ineffable concept: *“All traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life but only to an imagined celestial existence... logic takes us closer to heaven than other studies.”* [19]. Bertrand Russell,

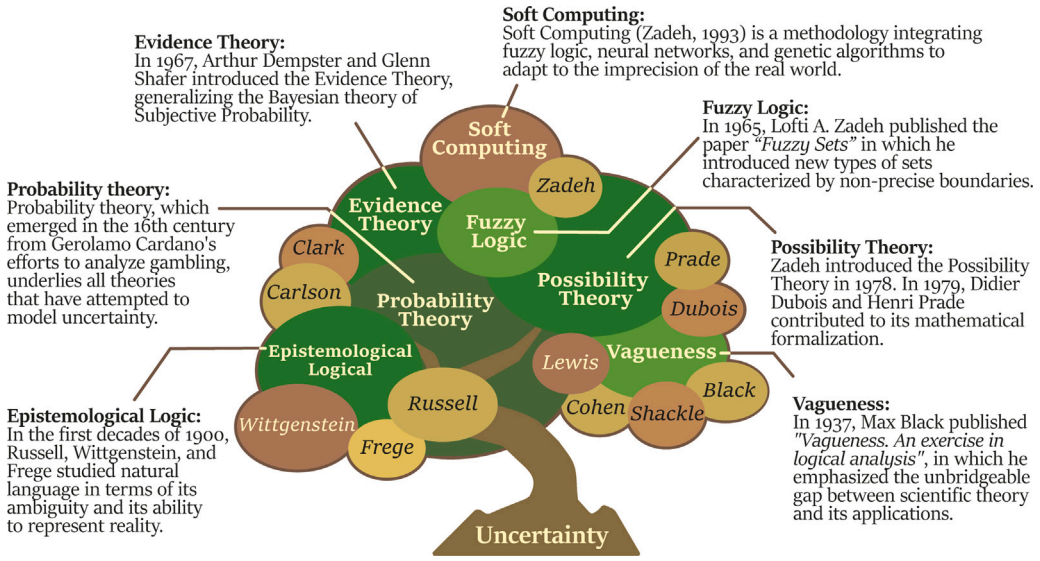

Fig. 1. Road map of evolution of Uncertainty conceptualization in the 20th century, throw years and currents of thought.

together with the contemporary philosophers Gottlob Frege and Ludwig Wittgenstein, studied natural language for a long time in relation to its ambiguity and its ability to represent reality [20,21], laying the foundations for the Logical Atomism, an important philosophical view according to which reality is thought of as decomposition into non-reducible propositions, i.e., atomic facts, to which the principles of classical logic can be applied [22]. Both Russell and Wittgenstein came to consider vagueness as a "degree" to be attributed to reality based on the number of "differences" that the various systems derived from its representation (through language) can have.

Another philosopher who studied the concept of Vagueness was Max Black, who in his 1937 essay entitled "*Vagueness. An exercise in logical Analysis*" [23] underlined the unbridgeable gap between scientific theory and its application. However, Black contested Russell's hypothesis of the non-applicability of logic principles to "vague" concepts, because he saw it as too great a threat to reasoning with ordinary language (epistemological logic) [24]. Two other philosophers who dealt with the concept of uncertainty were Michael Cohen, who considered the problem of legal reasoning, introducing the so-called Baconian Probabilities, understood as degrees of provability of an evidence [25], and David Lewis, who introduced a graded notion of possibility in the form of relations between subjective plausibility estimated for counterfactual statements in different possible worlds [26]. Another step forward in the long road of the formalization of the concept of uncertainty, as we know it today, was made by the economist G. L. S. Shackle who between 1940 and 1970 wrote many essays on the conceptualization of the degree of possibility of an event, understood as the inverse of the degree of surprise that this event would arouse if it occurred. Shackle considered the possibility as closely linked to the uncertainty in the decision and inverse to the impossibility, understood as the necessity for the opposite event to occur. He formulated the Principle of Decisional Cruciality, later used in the theory of financial risk analysis [27].

Probability, possibility, and evidence theory

The most important turning point in the formulation of modern uncertainty theory was provided by Lofti A. Zadeh, who in 1965 published the paper "Fuzzy Sets" [28] in which he introduced new types of sets characterized by non-precise boundaries. Membership to a fuzzy set cannot be established with certainty, but is defined through a continuous function, in this way Zadeh questioned the foundations of Aristotelian dichotomous logic and introduced a logic with continuous values that actually contains and does not exclude the binary logic (true-false) used by classical Bayesian Probability Theory. Fuzzy logic has greatly facilitated the representation of uncertainty in its various forms such as ambiguity, measurement error, incomplete information, and natural variability. Zadeh, like Russell and Wittgenstein, deeply reflected on natural language, on the fact that natural language represents the most accurate tool possible for the description of reality perceived by human beings because it is the result of an evolution of dozens of thousands of years. In natural language there are terms such as *"something"*, *"some"*, *"about"*, *"little"*, *"very"*, *"essentially"*, *"more or less"*, which are essentially "vague" terms, too complex to be processed with a logical-mathematical language, but which perfectly represent concepts specific to areas such as biology, physiology, psychology, sociology, economics and more generally human and natural sciences. Zadeh called these terms "hedges" [29]. Hedges are considered untractable by classical computing, based on a binary architecture. According to Lofti Zadeh, the difference between human and mechanical intelligence lies in the human brain's ability to reason in imprecise and non-quantitative terms. Fuzzy Set Theory aims to facilitate the codification, by the machine, of concepts, vague and imprecise from a quantitative point of view, but very significant from a qualitative one. *"… to effectively deal with such [biological] systems, which are generally orders of magnitude more complex than artificial systems, we need a radically different kind of mathematics, the mathematics of fuzzy or nebulous quantities that are not describable in terms of probability*

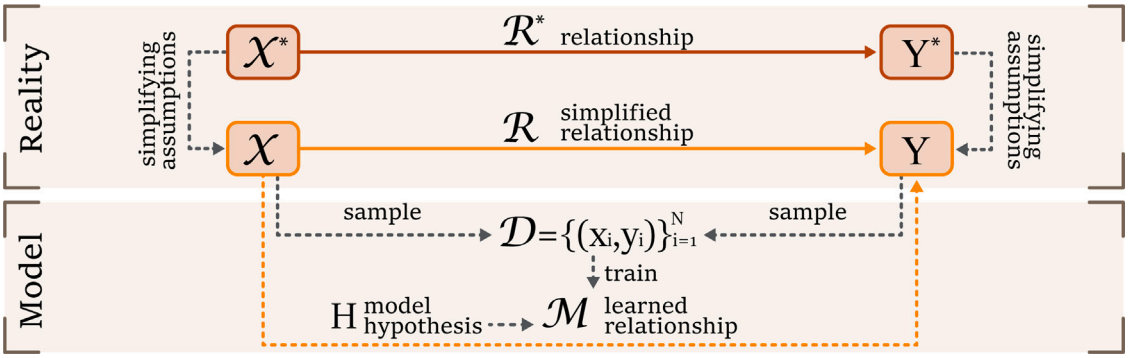

Fig. 2. Graphical scheme showing the relationship/simplified relationship between the real input/output and the corresponding trained model.

distributions" [28]. For fuzzy logic, the proposition "x is a member of A" is not necessarily either true or false, but it may be true only to some degree, the degree to which x is actually a member of A. In 1978 Zadeh introduced the Theory of Possibility [30], an important mathematical theory for dealing with the propagation of uncertainty with limited statistical or subjective information and for managing incomplete information. In subsequent years Didier Dubois and Henri Prade contributed to the mathematical formalization and development of this theory [31,32]. At a formal level, there is a broad correspondence between the theory of possibility and the theory of probability both are based on set functions, and in a certain sense possibility theory may be interpreted in terms of interval-valued probabilities. Important differences consist of the fact that in the possibility theory, a pair of functions are used (the possibility operator and the necessity operator) instead of just one and the addition operator corresponds to the maximum operator, in fact in the probability theory the distributions have a sum 1, while in the possibility theory, the distributions have max value 1 [33]. In his 1995 essay *"Discussion: Probability Theory and Fuzzy Logic Are Complementary Rather Than Competitive"* [34] Zadeh theorized that probability theory alone is not sufficient to comprehensively deal with the concept of uncertainty, but rather probability theory and possibility theory are complementary and very useful for jointly dealing with different aspects of uncertainty. Probability theory offers the possibility of representing with numbers, while fuzzy logic offers the possibility of representing with words, and estimating fuzzy probabilities. Classical probability theory is not very effective in those fields in which knowledge is incomplete and the dependencies between variables are not well defined or in those fields in which human reasoning, perceptions, and emotions play an important role. From this perspective, contamination of fuzzy logic and classical probability theory results in a significant enrichment in terms of representative capacity: probability fundamentally measures the frequency of an event, while possibility is used to quantify the meaning of an event [17].

There is also a strong correspondence between probability theory and Evidence theory, also called Dempster-Shafer Theory (DST), introduced by Arthur Dempster in 1967 [35] in the field of statistical inference and later developed by Glenn Shafer [36]. DST generalizes the Bayesian Theory of Subjective Probability [37]. The main difference with classical probability theory is that in probability theory the evidence (degree of belief) is associated with only one possible event, while in DST the evidence can be associated with multiple possible events, i.e., sets of events. Degrees of belief derived by different events can be combined through Dempster's rule of combination, making DST an important tool for mathematically dealing with uncertainty.

Teach to machines uncertainty reasoning

In 1981 Clark and Carlson focused on the computational context associated with Machine Learning algorithms, observing that ambiguity in a computational context can lead to significant uncertainty [38,39]. They proposed a disclaimer between stochastic (also called aleatoric) uncertainty, which occurs because the Machine Learning algorithm can behave in many different ways due to variable input data, and subjective (also called epistemic) uncertainty, which arises from a lack of information, from an inadequacy of the parameters or from problems concerning the implementation of the algorithm. From the 80 s other fuzzy measures had been introduced [40] and attempts to give an algebraic structure to fuzzy spaces have been made, introducing new ideas of fuzzy arithmetic [41] and hybrid arithmetic [42,43]. An interesting methodological approach introduced by Zadeh in 1993 is represented by soft computing [44], a methodology that integrates fuzzy logic, Neural Networks, and genetic algorithms to adapt to the imprecision of the real world. Soft computing aims to manage uncertainty, incompleteness, tolerance for errors, and biases to obtain tractability, robustness, scalability, and optimization of computational costs. Soft computing has led to the birth of Neuro-Fuzzy Systems [45] and Fuzzy Neural Networks [46], which play a particularly important role in the process of inducing rules from observation and in the creation of adaptive systems. In conclusion, many possible approaches to the formalization and management of uncertainty in Deep Learning have been introduced, capable of tackling the problem from different perspectives. For our purposes, however, the probabilistic approach, based on the Bayesian paradigm introduced and formalized by Pearl in 1988 [47] and currently the predominant uncertainty analysis mechanism in ML methods, is the most appropriate because it allows greater explanatory clarity and pedagogical potential.

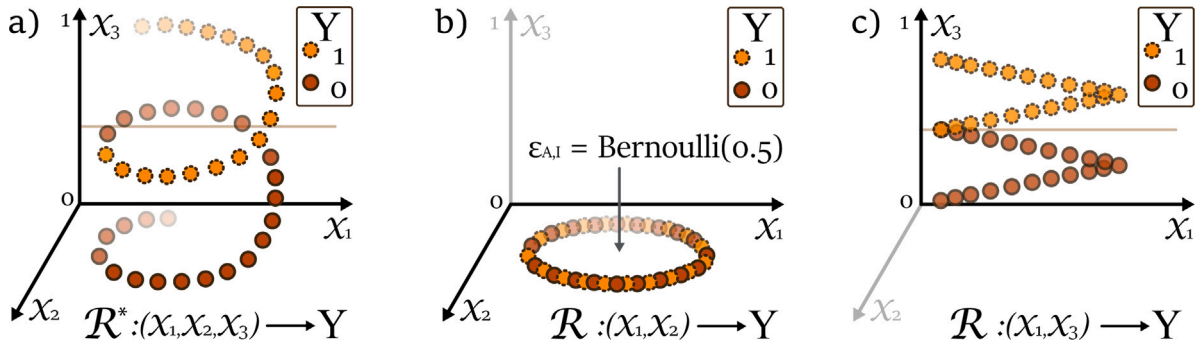

Fig. 3. Examples of how the choice of dimensionality affects the perception of the relationship under study, $\mathcal{R}$, and the uncertainties associated with it. (Panel a): $\mathcal{X}^* = (\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3) = [0, 2\pi]^2 \times [0, 1]$, $Y = \{0, 1\}$, $\mathcal{R}^*(x_1, x_2, x_3) = 1$ if $x_3 > 0.5$. (Panel b): Limiting the input space to two dimensions can hide the real relationship; to an observer, the y values appear to be randomly sampled from a Bernoulli distribution (Inherent Aleatoric Uncertainty). Therefore, an approximate model $\mathcal{M}$ emulates a simplified $\mathcal{R}$ affected by aleatoric uncertainty. (Panel c): Conversely, removing confounding variables can reduce the complexity of the problem.

3. Uncertainties: *From intuition to formal derivation*

Uncertainty analysis in Machine Learning begins with the formal definition of the uncertain quantities involved in the modeling problem. In fact, uncertainties can be of different types (including aleatoric, model, and approximation uncertainties) and can be intrinsic to the relationship under study or derived from an attempt to approximate reality with a numerical model. Although most techniques derived from the Bayesian approach fail to disentangle reducible (epistemic) and irreducible (aleatoric) uncertainties, providing a formal definition allows the practitioner to understand how a method provides an appropriate estimate of random variability. In addition, each type of uncertainty requires an appropriate strategy to mitigate it, so it is fundamental to have a complete overview of the sources of randomness in order to properly manage the modeling of the phenomena.

3.1. Reality and model: The map isn't the territory

The first step, with reference to Fig. 2, is to define the inputs/instance space of the problem (denoted by $\mathcal{X}^*$) and the corresponding outputs/outcomes (denoted by $\mathcal{Y}^*$). These two quantities are associated by a relationship $\mathcal{R}^* : \mathcal{X}^* \to \mathcal{Y}^*$ that returns a value $y^* \in \mathcal{Y}^*$ for each input $\mathbf{x}^* \in \mathcal{X}^*$. This relationship represents **reality**, the true output once the input is chosen, and is independent of measurement error or model approximation. However, in several real cases, the relationship is too complex to be handled, and therefore we work with a simplified relationship $\mathcal{R}$ which relates an input space $\mathcal{X}$ with the corresponding outcomes $\mathcal{Y}$.

The goal of a mathematical modeling process is to estimate the relationship $\mathcal{R}$ using an approximating **model** $\mathcal{M}$ (i.e., model induction). The model $\mathcal{M}$ is constructed from a sample/**data-set** $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$ and a set of hypothesis $h = (\theta, \hat{h}) \in H$:

$$\mathcal{M} := \mathcal{M}_h = \mathcal{M}_{\theta,\hat{h}} : \mathcal{X} \to \mathcal{Y} \tag{1}$$

The $\mathcal{D}$ data set includes the data available to the experimenter. Instead, a choice of hypothesis $h = (\theta, \hat{h}) \in H$ of a model includes both the *trainable parameters* θ of a model (e.g., usually weights and biases for an ANN) and all those choices $\hat{h}$ made on the basis of prior knowledge (e.g., use of a linear regressor or an ANN, choice of training hyperparameters, etc.).

3.2. Aleatoric uncertainty: A random world

The first type of uncertainty that can affect the relationship under consideration is called **Aleatoric Uncertainty** (ε_A, from *Alea*, dice in Latin). It is often referred to as Stochastic Uncertainty, Statistical Uncertainty, or even Irreducible Uncertainty. Random uncertainty may be inherent in the relationship under consideration ($\varepsilon_{A,I}$, **Aleatoric Inherent Uncertainty**), the result of noise in the analysis and measurement process ($\varepsilon_{A,N}$, **Aleatoric Experimental Uncertainty/Noise**) or to random alterations in the modeling process ($\varepsilon_{A,M}$, **Aleatoric Model Uncertainty**). Random uncertainty can vary with the input $\mathbf{x}$ (*Heteroscedastic Uncertainty*) or be constant over the input space (*Homoscedastic Uncertainty*). Aleatoric uncertainty represents the inherently random nature of the problems studied, which cannot be explained away. Aleatoric uncertainty is inherently **irreducible** (i.e., it cannot be eliminated by improving the model/providing more data) and therefore, unless properly modeled, will always result in an unexplained discrepancy between model predictions $\mathcal{M}(x)$ and reality $\mathcal{R}(x)$.

Remark 1 (*Does Aleatoric Uncertainty Exist?*)**.** Referring to Fig. 3, aleatoric uncertainty associated with the relationship $\mathcal{R}$ is often a consequence of the simplifying assumptions, which reduce the more complex relationship $\mathcal{R}^*$ to the simpler one $\mathcal{R}$. As an example, consider an input space $\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3) = [0, 2\pi]^2 \times [0, 1]$ and a relationship $\mathcal{R}^*(x_1, x_2, x_3) = 1 \iff x_3 > 0.5$. This relationship is fully deterministic. However, working with a limited amount of information (i.e., reducing $\mathcal{X}$ to $(\mathcal{X}_1, \mathcal{X}_2)$) leads to a relationship

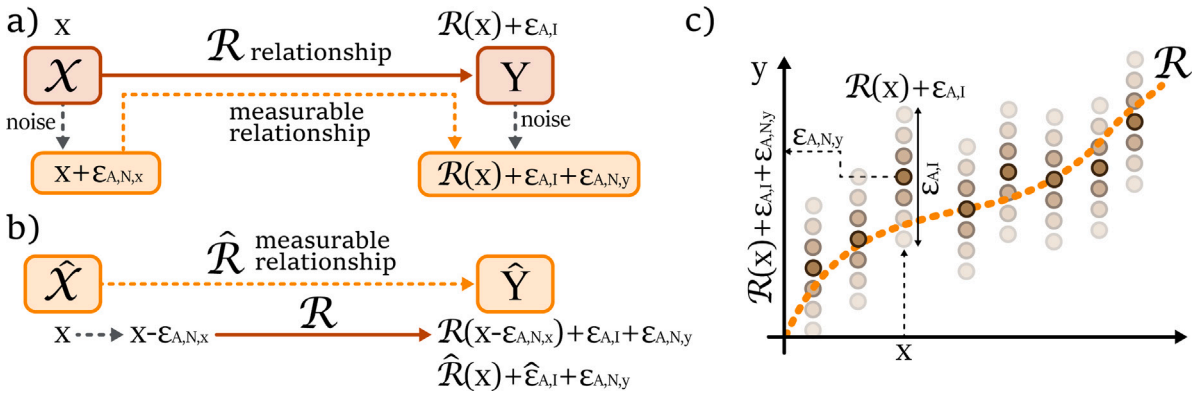

Fig. 4. Model of aleatoric uncertainty. (Panel a): The non-deterministic relationship $\mathcal{R}$ relates inputs/outputs which are affected by Experimental Aleatoric Uncertainty/Noise ($\varepsilon_{A,N,\mathbf{x}}$, $\varepsilon_{A,N,y}$). (Panel b): With a change of perspective, $\mathcal{R}$ can be seen as a relationship between a deterministic input and an output influenced by Aleatoric Inherent ($\varepsilon_{A,I}$) and Aleatoric Experimental Uncertainty/Noise. (Panel c): Shows the effect of Aleatoric Uncertainties. The deterministic value $\mathbf{x}$ corresponds to a random output affected by the Aleatoric Inherent Uncertainty $\mathcal{R}(\mathbf{x})+\varepsilon_{A,I}$, which is further contaminated by the Aleatoric Experimental Uncertainty/Noise on y.

$\mathcal{R}(x)$ corresponding to a fully random Bernoulli distribution $p = 0.5$. It is worth noting that often the aleatoric uncertainty is just a consequence of the limited knowledge of the problem, which forces us to approximate the effects of the unknown/unmeasured parameters and call it "Aleatoric Uncertainty". The approximating model $\mathcal{M}$ emulates the simplified relationship $\mathcal{R}$ and therefore will never be able to match the reality $\mathcal{R}^*$, perceiving the discrepancy as aleatoric uncertainty.

Aleatoric inherent uncertainty: A non-deterministic relationship

The relationship $\mathcal{R}$ is usually not deterministic (i.e., for a given input $\mathbf{x} \in \mathcal{X}$, $\mathcal{R}(x)$ is a random variable). It is crucial to emphasize that this source of uncertainty is a property of the relationship/reality and therefore will never be reduced by providing more data or improving the models. Referring to Fig. 4(a), **Aleatoric Inherent Uncertainty** can be introduced using the following formalism: $\mathbf{x} \to \mathcal{R}(\mathbf{x}) + \varepsilon_{A,I}$, where $\varepsilon_{A,I} := \varepsilon_{A,I}(\mathbf{x}, \mathcal{R})$ is a function of both the input value $\mathbf{x}$ and the relationship $\mathcal{R}$ itself.

Example 1. Consider a relationship $\mathcal{R} : \mathcal{X} = \mathbb{N} \to \mathcal{Y} = \mathbb{N}$ which associates an integer $\mathbf{x}$ with the value given by the sum of x six-sided dice. $\mathcal{R}$ is inherently random, since the same input $\mathbf{x}$ may correspond to different outputs y (Aleatoric Inherent Uncertainty). In this particular example, the Inherent Aleatoric Uncertainty depends on the input value $\mathbf{x}$ (Heteroscedastic Uncertainty).

Aleatoric Experimental Uncertainty/Noise: The observer effect

A second type of uncertainty (**Aleatoric Experimental Uncertainty/Noise**) refers to variations in information content caused by the acquisition process itself. Examples include the choice of experimental settings, measurement errors, incorrect labeling of data, or visual artifacts introduced into the input image (blur, motion artifacts, degradation of image quality, etc.). Referring to Fig. 4(a), this uncertainty affects both the input and output quantities ($\mathbf{x} \to \mathbf{x} + \varepsilon_{A,N,\mathbf{x}}$ and $y \to y + \varepsilon_{A,N,y}$), making the problem almost mathematically intractable. However, to simplify the discussion, it is possible to change the perspective of the problem slightly (Fig. 4(b)).

If the original study relationship $\mathcal{R}$ links a deterministic input $\mathbf{x}$ to its output y and the observer analyses a perturbed version of it ($\mathbf{x} + \varepsilon_{A,N,\mathbf{x}}, y + \varepsilon_{A,N,y}$), the same problem can be seen as a different relationship $\hat{\mathcal{R}}$ between the perturbed input and output. $\hat{\mathcal{R}}$ takes as input the (perturbed) value $\mathbf{x} \in \hat{\mathcal{X}}$, and produces the output $\hat{\mathcal{R}}(\mathbf{x}) := \mathcal{R}\left(\mathbf{x} - \varepsilon_{A,N,\mathbf{x}}\right)$. This change of perspective shifts the Experimental Aleatoric Uncertainty on $\mathbf{x}$ to the relation itself (i.e., to the Aleatoric Inherent Uncertainty $\varepsilon_{A,I}(\mathbf{x}, \hat{\mathcal{R}})$) thus condensing all uncertainties on y:

$$\text{Original} \quad \mathbf{x} + \varepsilon_{A,N,\mathbf{x}} \to \mathcal{R}(x) + \varepsilon_{A,I}(\mathbf{x}, \mathcal{R}) + \varepsilon_{A,N,y}$$

$$\text{Shift} \quad \mathbf{x} \to \mathcal{R}(\mathbf{x} - \varepsilon_{A,N,\mathbf{x}}) + \varepsilon_{A,I} + \varepsilon_{A,N,y} := \hat{\mathcal{R}}(\mathbf{x}) + \varepsilon_{A,I}(\mathbf{x}, \hat{\mathcal{R}}) + \varepsilon_{A,N,y}$$

To improve readability, we will always assume the perspective shift in the following. Thus we will use a relationship between a deterministic input $\mathbf{x}$ and the corresponding output $\mathcal{R}(\mathbf{x}) + \varepsilon_{A,I} + \varepsilon_{A,N}$ by omitting $\hat{\cdot}$ and $\cdot_y$ from the notation.

Aleatoric model uncertainty: The effect of random inference

Aleatoric uncertainty can also be generated by the model $\mathcal{M}$ itself during the inference process. This can be due to the pseudo-random propagation of rounding errors on a machine, or to the numerical instability of approximating certain physical phenomena. However, **Aleatoric Model Uncertainty** ($\varepsilon_{A,M}$) is often negligible compared to the other components of aleatoric uncertainty. Therefore, for simplicity, we will assume that the model is not subject to aleatoric uncertainty, which therefore remains a property of the true relationship under study (Aleatoric Inherent Uncertainty) and the data collection process (Aleatoric Experimental Uncertainty).

3.3. Epistemic uncertainty: The lack of knowledge

While Aleatoric Uncertainty encompasses the intrinsically random nature of the underlying relationship $\mathcal{R}$, the variability caused by lack of knowledge about the problem is called **Epistemic Uncertainty**. This uncertainty is also known as Systematic Uncertainty, Subjective Uncertainty, or Reducible Uncertainty. Epistemic uncertainty can be induced by the *a priori* choice of hyperparameters $\hat{h}$ characterizing the model ($\varepsilon_{E,M}$, Epistemic Model Uncertainty) or, conversely, by a finite amount of available data $\mathcal{D}$ ($\varepsilon_{E,Ap}$, Epistemic Approximation Uncertainty). The inherent randomness of the relationship $\mathcal{R}$ is encoded in the Aleatoric Uncertainty and does not directly affect the Epistemic one. Since this uncertainty is due to a lack of knowledge, it can be reduced by improving the choice of model hyperparameters h or by increasing the size of the data-set $\mathcal{D}$ (i.e., is the **reducible** part of total uncertainty).

Epistemic model uncertainty: How to train the optimal model

In supervised learning, to obtain a numerical approximation of the relationship $\mathcal{R}$, we need to define a proper model $\mathcal{M}_h = \mathcal{M}_{\theta,\hat{h}} : \mathcal{X} \to \mathcal{Y}$ where $h = (\theta, \hat{h}) \in H$ is a set of hypotheses. The set of hypotheses can be divided into hyperparameters $\hat{h}$ and trainable parameters θ. Hyperparameters (often referred to as meta-hypotheses) correspond to the choices made a priori by the experimenter based on previous experience, available resources, or literature. These include the type of model used (classical linear/logistic/random Forest models, Deep Learning models, etc.), the model structure (number of Random Trees for random forests, number and type of layers for Neural Networks, etc.), and even the rule needed to define how well the model fits (i.e., the loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$). Once the hyperparameters $\hat{h}$ are established, we have to choose the optimal model (fully characterized by its trainable parameters θ).

The goal of the learner is to induce an optimal hypothesis $(\theta^*, \hat{h}) \in H$ with low **expected loss/risk** (the cost function):

$$\theta^* := \underset{(\theta,\hat{h})\in H}{\arg\min}\, \mathbb{E}_\ell(\theta,\hat{h}) \quad ; \quad \mathbb{E}_\ell(\theta,\hat{h}) := \int_{\mathcal{X}\times\mathcal{Y}} \ell(\mathcal{M}_{\theta,\hat{h}}(\mathbf{x}), y)\; p(\mathbf{x}, y)\; d\mathbf{x}dy \tag{2}$$

where p is a probability measure on $\mathcal{X} \times \mathcal{Y}$, ℓ is the chosen loss, $\mathbb{E}_\ell$ is the expected loss/risk (the cost function), and θ^* are the optimal trainable parameters (i.e., the true expected loss minimizer). $\mathcal{M}_{\theta^*,\hat{h}}$ is called the **Optimal Bayesian Predictor**. To provide an intuitive insight, the optimal trainable parameters θ^* are obtained as those that minimize a measure of model-reality discrepancy (i.e., the loss ℓ) over the entire space of possibilities $\mathcal{X} \times \mathcal{Y}$. The model obtained depends both on these optimized parameters and on the a priori hyperparameters $\hat{h}$, hence: *$\mathcal{M}_{\theta^*,\hat{h}}$ is the optimal model given a set of hyperparameters $\hat{h}$ and goodness of fit measure ℓ.*

Remark 2 (*A Given Parameter is Trainable or Fixed?*)**.** In most cases, it is difficult to separate trainable parameters from hyperparameters, which can often be switched from one type to another (e.g., the number of layers in a Deep Neural Network can be fixed or automatically adapted to the available data). There are several strategies (**hyperparameter tuning/optimization**) to identify an optimal set of hyperparameters $\hat{h}$, including: Grid Search, Random Search, Bayesian optimization, and Nested Cross Validation. Note that this can be seen as a shift of all hyperparameters to the trainable set.

Even with the optimal trainable parameters θ^* and even assuming a null aleatoric uncertainty ε_A, there could be a discrepancy between the reality and the model outcome: $\mathcal{R}(\mathbf{x}) \neq \mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x})$. The discrepancy between the Optimal Bayesian Predictor and the reality is called **Epistemic Model Uncertainty** ($\varepsilon_{E,M}$):

$$\mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x}) = \mathcal{R}(\mathbf{x}) + \varepsilon_{E,M} \tag{3}$$

This uncertainty is often referred to as Model Inadequacy, Model Bias, Model Discrepancy, or Structural Uncertainty, and thus underlies the dependence on the limit of the model itself to approximate the relationship $\mathcal{R}$.

Epistemic approximation uncertainty: Training on real data

The minimization problem described in Eq. (2) requires full knowledge of the space $\mathcal{X} \times \mathcal{Y}$, which is usually not available to the experimenter. In fact, in most supervised learning applications, the information about the space $\mathcal{X} \times \mathcal{Y}$ is provided by a sample $\mathcal{D} := \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\} \subset \mathcal{X} \times \mathcal{Y}$, the so-called **Training data-set**. The Optimal Bayesian Predictor $\mathcal{M}_{\theta^*,\hat{h}}$ is therefore replaced by the **Empirical Model** $\mathcal{M}_{\hat{\theta},\hat{h}}$ trained on the data-set $\mathcal{D}$, whose optimal trainable parameters $\hat{\theta}$ are obtained as:

$$\begin{aligned} \hat{\theta} &:= \underset{(\theta,\hat{h})\in H}{\arg\min}\, \hat{\mathbb{E}}_\ell(\theta,\hat{h}) \approx \theta^* \\ \hat{\mathbb{E}}_\ell(\theta,\hat{h}) &:= \frac{1}{N}\sum_{i=1}^{N} \ell\left(\mathcal{M}_{\theta,\hat{h}}(\mathbf{x}_i), y_i\right) \approx \mathbb{E}_\ell(\theta,\hat{h}) \\ \mathcal{D} &:= \left\{\left(\mathbf{x}_i, y_i\right)\right\}_{i=1}^{N} \subset \mathcal{X}\times\mathcal{Y} \end{aligned} \tag{4}$$

where the empirical expected loss/risk $\hat{\mathbb{E}}_\ell$ approximates the expected loss/risk $\mathbb{E}_\ell$.

Remark 3 (*Cost, Loss and Objective Functions*)**.** The Loss Function ℓ quantifies how much the model prediction deviates from the corresponding ground truth y_i ($\ell\left(\mathcal{M}(\mathbf{x}_i), y_i\right)$). Common choices for regression task include Absolute Loss (L_1-loss, $\ell\left(\mathcal{M}(\mathbf{x}_i), y_i\right) = \left|\mathcal{M}(\mathbf{x}_i) - y_i\right|$), Square Loss ($L_2$-loss, $\ell\left(\mathcal{M}(\mathbf{x}_i), y_i\right) = \left(\mathcal{M}(\mathbf{x}_i) - y_i\right)^2$), or Variance Attenuation loss for two headed models ($\ell(\mathcal{M}(\mathbf{x}_i), y_i) =$

$\frac{\log \sigma^2_{\mathcal{M}}(\mathbf{x}_i)}{2} + \frac{(y_i - \mu_{\mathcal{M}}(\mathbf{x}_i))^2}{2\sigma^2_{\mathcal{M}}(\mathbf{x}_i)}$). For classification task, a frequent choice is the Binary Cross entropy ($\ell(\mathcal{M}(\mathbf{x}_i), y_i) = y_i \cdot \log(\mathcal{M}(\mathbf{x}_i)) + (1 - y_i) \cdot \log(1 - \mathcal{M}(\mathbf{x}_i))$). The Cost Function is a measure of how much the model's prediction of a group of objects deviates from the ground truth. It is usually derived as a proper combination of the loss function values as the mean (Expected loss), median, etc. (e.g., $\text{Cost}(\mathcal{M}, D) = \frac{1}{\#D} \sum_{i=1}^{\#D} \ell(\mathcal{M}(\mathbf{x}_i), y_i)$). Most cost functions can be derived as a special case of Maximum Likelihood Estimation, see Appendix A. Whenever a regularization term is added to the cost function, the function to be minimized is called the Objective Function ($\text{Obj}(\mathcal{M}, D) = \text{Cost}(\mathcal{M}, D) + \text{Regularisation}(\mathcal{M})$).

The discrepancy between θ^* and $\hat{\theta}$ depends on the quality and the number N of data available and is called the **Epistemic Approximation Uncertainty** ($\varepsilon_{E,Ap}$):

$$\mathcal{M}_{\hat{\theta},\hat{h}}(\mathbf{x}) = \mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x}) + \varepsilon_{E,Ap} \tag{5}$$

Epistemic Approximation Uncertainty is sometimes referred to as Interpolation Uncertainty.

Remark 4 (*Why Do We Need More Data?*)**.** Increasing the data-set size $N \to \infty$ improves the approximation $\hat{\mathbb{E}}_{\ell}(\theta, \hat{h}) \approx \mathbb{E}_{\ell}(\theta, \hat{h})$ and thus reduces the Epistemic Approximation Uncertainty $\varepsilon_{E,Ap} \to 0$. However, the increasing amount of data cannot reduce the epistemic model uncertainty, in fact:

$$\lim_{N\to\infty} \mathcal{M}_{\hat{\theta},\hat{h}}(\mathbf{x}) = \mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x}) \approx \mathcal{R}(\mathbf{x}) + \varepsilon_{E,M}$$

and therefore "more data" does not automatically mean a perfect approximation of reality.

It is worth noting that D is a sample from the space $\mathcal{X} \times \mathcal{Y}$. Therefore, the empirical distribution $p(D)$ obtained from D and used to train the model, imperfectly mimics the real underlying distribution on $\mathcal{X} \times \mathcal{Y}$ (i.e., $p(D) \approx p(\mathcal{X} \times \mathcal{Y})$). This can often occur with a light-tailed $p(\mathcal{X} \times \mathcal{Y})$ distribution, which can lead to a data set D with little or no training data in the tails. This discrepancy can lead to an Epistemic Approximation Uncertainty strongly dependent on the input $\mathbf{x}$. The inputs $\mathbf{x}$ which are relatively under-represented in $p(D)$ are called **out-of-distribution** values and are a major cause of reduced model generalization.

Remark 5 (*Universal Approximation Theorem*)**.** Increasing the complexity of the model (i.e., increasing the number of trainable parameters θ) improves the model's ability to approximate a deterministic relationship $\mathcal{R}$. In particular, for Artificial Neural Networks, the Universal Approximation Theorem ensures that a sufficiently complex Neural Network can perfectly approximate every relationship $\mathcal{R}$ ($\varepsilon_{E,M} \to 0$). However, the increasing complexity of the models requires a huge amount of data to be trained on, resulting in a dramatic increase in Epistemic Approximation Uncertainty ($\varepsilon_{E,Ap} \to \infty$).

3.4. Summary of the uncertainties

The uncertainties introduced in this section are summarized in the following set of equations:

$$y = \mathcal{R}(\mathbf{x}) + \varepsilon_{A,I} + \varepsilon_{A,N} \qquad \text{Aleatoric Inherent+Experimental Unc.} \tag{6}$$

$$\mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x}) \approx y + \varepsilon_{E,M} \qquad \text{Epistemic Model Uncertainty} \tag{7}$$

$$\mathcal{M}_{\hat{\theta},\hat{h}}(\mathbf{x}) \approx \mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x}) + \varepsilon_{E,Ap} \qquad \text{Epistemic Approximation Uncertainty} \tag{8}$$

The simplified relationship $\mathcal{R}$ relates the input $\mathbf{x}$ to the output y with the inclusion of two non-deterministic components, the Aleatoric Inherent Uncertainty ($\varepsilon_{A,I}$) and the Aleatoric Experimental Uncertainty ($\varepsilon_{A,N}$), which represent the **irreducible** part of the total variability (Eq. (6)).

The Optimal Bayesian Predictor $\mathcal{M}_{\theta^*,\hat{h}}(\mathbf{x})$ is the best approximation of the relationship given the experimenter's *a priori* knowledge. The discrepancy between prediction and reality is called Epistemic Model Uncertainty (Eq. (7)). Working with a data-set D of finite size leads to another difference between the actual Empirical Model $\mathcal{M}_{\hat{\theta},\hat{h}}$ and the Optimal Bayesian Predictor, the so-called Epistemic Approximation Uncertainty ($\varepsilon_{E,Ap}$, Eq. (8)). The epistemic uncertainties represent the **reducible** part of the total variability and can be reduced by improving the quality of the *a priori* hypotheses ($\varepsilon_{E,M} \to 0$) and by increasing the data-set size ($\varepsilon_{E,Ap} \to 0$).

Therefore, the equation that contains the full relationship between the actual model and the reality is the following:

$$\mathcal{M}_{\hat{\theta},\hat{h}}(\mathbf{x}) \approx \mathcal{R}(\mathbf{x}) + \varepsilon_{A,I} + \varepsilon_{A,N} + \varepsilon_{E,M} + \varepsilon_{E,Ap} := \mathcal{R}(\mathbf{x}) + \varepsilon_A + \varepsilon_E \tag{9}$$

where ε_A and ε_E represent the aleatoric and epistemic uncertainty, respectively. The combination of ε_A and ε_E is the **Predictive Posterior Uncertainty**.

Remark 6 (*Only Additive Uncertainties?*)**.** The formalism used (i.e., $y + \varepsilon$) may erroneously suggest that the uncertainties act only as an additive term on the values. However, $\varepsilon = \varepsilon(y)$ must be considered as a (nonlinear) function of the value itself. A description of how to model this uncertainty as an additive plus multiplicative component (i.e., $y + \varepsilon := \left(y \cdot \varepsilon_{mult}\right) + \varepsilon_{add}$) is reported in [48,49].

4. Techniques to handle uncertainties

The optimal balance between efficiency, reliability, complexity and computational cost of the different UQ approaches is difficult to find and depends strongly on the task under investigation. To assist practitioners in the challenging task of selecting the most appropriate method for their application, we opted to divide the techniques into three main macro-categories: *Intrusive*, *Non-intrusive* and *Semi-intrusive*.

The *Intrusive* (or by-design) techniques are effective but complex and usually computationally expensive methods that require careful integration with the model (Section 5). Among the numerous intrusive techniques, we focused on a straightforward distributional output extension of standard ANN (Section 5.1), on Bayesian Neural Networks (Section 5.2), and on Deep Gaussian Processes (Section 5.3). Conversely, *Non-intrusive* (or post-hoc) approaches are designed for a posteriori evaluation of a pre-trained model. These methods have inferior performance compared to intrusive methods but are also easier to implement, require fewer model modifications, and are usually less computationally expensive (Section 6). We introduced the Monte Carlo dropout (Section 6.1) as the main post-hoc technique and discussed the limitations of using the internal score as a measure of reliability (Section 6.2). Finally, semi-intrusive methods combine the effectiveness of intrusive techniques with the ease of the non-intrusive approaches (Section 7). The main semi-intrusive technique is the Ensemble method, which combines multiple deterministic models to quantify uncertainty, improve performance, and avoid over-fitting (Section 7.1).

Table 1 summarizes the characteristics of the main UQ techniques.

4.1. Bayesian framework

Most techniques for dealing with uncertainty rely significantly on Bayesian probability to provide a strong theoretical framework. In addition, most of the standard concepts behind classical (deterministic) ANNs (such as regularization functions, loss selection, dropout regularization, etc.) can be interpreted using the Bayesian formalism [64]. The Bayesian paradigm states that $p(x)$ is a measure of belief in the occurrence of an event; prior beliefs influence posterior beliefs:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)\cdot p(\theta)}{p(\mathcal{D})} := \frac{p(\mathcal{D},\theta)}{\int_\Theta p(\mathcal{D},\theta')\,d\theta'} \tag{10}$$

where $p(\theta)$ is the **prior** knowledge on trainable parameters, $p(\mathcal{D}|\theta)$ is the **likelihood** (the measure of belief that $\mathcal{D}$ can be generated by the hypothesis θ), $p(\mathcal{D})$ is the **evidence**, and $p(\theta|\mathcal{D})$ is the **posterior** (i.e., the belief in the hypothesis values given the original prior belief updated with the data $\mathcal{D}$). The choice of the prior $p(\theta)$ strongly depends on the available knowledge about the shape and distribution of the trainable parameters and can significantly influence the final quality of the posterior. An uninformative prior (e.g., chosen using Shannon's theory as the maximum entropy distribution under a given hypothesis) can provide less biased results but requires a lot of data to correctly update the real posterior. Conversely, a highly informative prior (associated with a strong belief in the parameter distribution) may provide a better posterior if correct, or pollute the result if incorrect. Induction (i.e., replacing observations by a general model) in the Bayesian setting can be seen as determining a posterior distribution on the set of trainable parameters $\theta \in \Theta$, given the data-set $\mathcal{D}$. In the Bayesian framework, making a prediction (given a prior set of hyperparameters $\hat{h}$ and the trainable parameters θ that fully characterize the model) can be written as $p(y|\mathbf{x},\theta) = p\left(y|\mathbf{x},\mathcal{M}_{\theta,\hat{h}}\right) = p(y|\mathbf{x},\mathcal{D})$, the so-called **Predictive Posterior Distribution**.

4.2. Aleatoric and epistemic uncertainty disentanglement

Disentangling the two components of prediction uncertainty (aleatoric and epistemic) is a challenging task that has been addressed by various authors for specific cases, including: regression/reinforcement learning [65], MC-Dropout [50], and Flipout/DropConnect/Ensemble [66]. It should be noted that the main limitation of the whole Bayesian approach is the difficulty of efficiently disentangling between the two types of uncertainty [1] and other approaches are based on non-Bayesian methods such as the fuzzy framework [67].

Remark 7 (*Are We Identifying ε_E and ε_A Correctly?*)**.** Epistemic and aleatoric disentanglement should be evaluated with numerical experiments [68] and ad hoc datasets [69], as disentanglement performance is typically evaluated on their respective downstream tasks [70]. In particular, Epistemic uncertainty should be reduced by increasing the amount and quality of the dataset and must be related to the ability of the model to discriminate OOD elements. On the contrary, aleatoric uncertainty should reach a plateau independent of the amount of data and must be related to the ability to reject ambiguous cases (e.g., label noise).

Among the Bayesian approaches, one possibility is the so-called Gaussian Logits Disentanglement [68]. Assuming that the aleatoric uncertainty is modeled as an explicit distribution provided by the model (i.e., multi-head ANNs, usually a simplified Gaussian distribution) and the epistemic is included in the trainable parameter distributions, a formula can be derived to obtain epistemic-aleatoric uncertainties [50].

Table 1
Characteriztics, advantages and drawbacks of state-of-the-art UQ techniques.

Technique	Description	Advantages	Drawbacks
Intrusive/By-design			
ANN with distributional outputs	An extension of the classical ANN that provides distributional output [50,51]	-Integrates standard predictions with ϵ_A uncertainty estimates -At inference time, only one model evaluation is required for each input -Minor model modifications in terms of architecture and training process	-Requires ad hoc losses derived from MLE theory -Overconfident estimates -Low ϵ_E approximation capability
BNN	Stochastic ANN with Bayesian weights/activation functions [52,53]. Special intrusive case of ensemble learning.	-Mitigates over-fitting -Integrates standard predictions with reliability estimates -Can be combined with ANN with distributional outputs on the simplified hypothesis to disentangle ϵ_A and ϵ_E -**MCMC Inference**: Exact posterior approximation -**VI**: Fast posterior approximation -**LA**: Post-hoc approximation of VI	-Increased number of degrees of trainable parameters -Simplified assumptions on posterior distribution, such as Gaussian i.i.d. marginals -Assumption on prior $p(\theta)$ can affect the posterior -At inference time, multiple model evaluations are required for each input x -**MCMC Inference**: Very high computational cost and lack of scalability -**VI/LA**: Poor approximation of true posterior distribution -**LA**: Local approximation of VI around MAP.
DGP	-**Stacked GP**: Several GPs combined in series [54,55] -**Deep Kernel Learning**: GP with a kernel parameterized by a DNN [56]	-Reliability predictions even with small data-sets -High-dimensional feature learning capability -**Deep Kernel Learning**: State-of-the-art predictions	-Complex mathematical framework and implementation -Require careful extension to overcome limitations -High computational/memory costs -**Stacked GP**: No longer a GP
Semi-intrusive			
Ensemble	Aggregation of multiple ML models [57,58]	-Embarrassingly parallelisable -The number of sub-models can be chosen according to memory/computing power. -General case of BNNs -Can be tailored to the problem by selecting specific ML sub-models	-High computational and memory costs -Sub-models/hyper-parameter choices can greatly influence models -At inference time, multiple model evaluations are required for each input **x**
Non-Intrusive/Post-hoc			
MC Dropout	Prediction based on Bernoulli dropping of weights at inference time [59,60]	-Low computational cost (only at inference time) -Mitigates over-fitting and provides improved predictions -Easy to implement and parallelisable -Theoretically developed framework	-No guaranteed convergence to the true posterior if GP approximations are not satisfied -Sometimes unable to fully capture of predictive uncertainty
Internal Model Score	Approximation using last layer score [61]	-Null computational cost -No need to change the model training process	-Overconfident predictive posterior -Poorly calibrated measures of reliability
Trust Score	Clustering score describing the level of confidence in the output [62,63]	-Solid mathematical background -Low computational cost -Easy to implement	-Suitable for classification only -No upper bound -Reliability calibration difficulties -Less informative on high-dimensional feature spaces

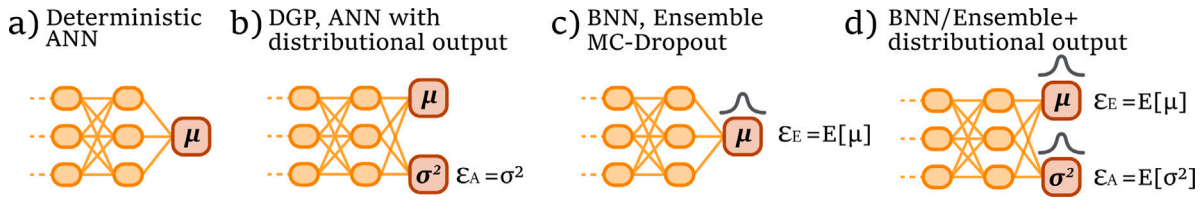

Fig. 5. Example of uncertainty disentanglement based on predictive posterior Gaussian assumptions. (Panel a): Deterministic ANNs can only provide a point estimate. (Panel b): Distributional output ANNs or DGPs output both a prediction and a reliability estimate (aleatoric uncertainty under the Gaussian hypothesis). (Panel c): BNNs, Ensembles, or MC-dropout provide a random value sampled from a distribution and can therefore approximate epistemic uncertainty. (Panel d): BNNs (or Ensembles) coupled to a distributional output provide a Gaussian Mixture Model output and can therefore approximate both epistemic and aleatoric uncertainty.

Regression

Uncertainty disentanglement can be derived by assuming that the predictive posterior distribution can be approximated by a Gaussian Mixture model [50]. Under this simplifying assumption, it holds:

$$\begin{aligned} &p(y|x) \sim \mathcal{N}(\mu_*(\mathbf{x}), \sigma_*^2(\mathbf{x})) \\ &\mu_*(\mathbf{x}) = \frac{1}{S}\sum_{j=1}^{S} \mu_j(\mathbf{x}) \quad ; \quad \sigma_*^2(\mathbf{x}) = \frac{1}{S}\sum_{j=1}^{S}\left(\sigma_j^2(\mathbf{x}) + \mu_j^2(\mathbf{x})\right) - \mu_*^2(\mathbf{x}) \end{aligned} \tag{11}$$

where S is the number of samples drawn by the model $\mathcal{M}$ (the number of outputs corresponding to different weight samples for Bayesian ANN, or the S predictions of an Ensemble/MC-Dropout approach), $\{\mu_j(x)\}_{j=1}^{S}$ are the predictions and $\{\sigma_i(\mathbf{x})\}_{j=1}^{S}$ are the corresponding variances (see Fig. 5). Therefore, under the simplified Gaussian assumption, the predictive variance can be decomposed into aleatoric and epistemic uncertainty:

$$\begin{aligned} \sigma_*^2(\mathbf{x}) &= \frac{1}{S}\sum_{j=1}^{S}\sigma_j^2(\mathbf{x}) + \frac{1}{S}\sum_{j=1}^{S}\mu_j^2(\mathbf{x}) - \mu_*^2(\mathbf{x}) \\ &= \mathbb{E}_j\left[\sigma_j^2(\mathbf{x})\right] + \mathbb{E}_j\left[\mu_j^2(\mathbf{x})\right] - \mathbb{E}_j\left[\mu_*(\mathbf{x})\right]^2 = \underbrace{\mathbb{E}_j\left[\sigma_j^2(\mathbf{x})\right]}_{\varepsilon_A} + \underbrace{\mathrm{Var}_j\left[\mu_j(\mathbf{x})\right]}_{\varepsilon_E} \end{aligned} \tag{12}$$

Intuitively, the epistemic uncertainty corresponds to the variance of the predictions $\mu_j(\mathbf{x})$ due to differences among the models. Conversely, the aleatoric uncertainty is derived as the average between the reliability $\sigma_j^2(\mathbf{x})$ of each prediction.

Remark 8 (*Does It Work?*)**.** It should be noted that if the normal assumptions about the predictive posterior distribution are not satisfied, this simplified decomposition can lead to non-intuitive results regarding the interaction between aleatoric and epistemic uncertainty [66].

Classification

For classification, a similar procedure can be applied to derive aleatoric/epistemic decomposition. The output of a (multiclass, $\mathcal{K}$ classes) classification model is usually a vector $(p_c)_{c=1}^{\#\mathcal{K}} \in [0,1]^{\#\mathcal{K}}$ of probabilities obtained by applying a softmax function to the so-called *logit layer* $z = (z_c)_{c=1}^{\#\mathcal{K}} \in \mathbb{R}^{\#\mathcal{K}}$. The output of the classification model is defined as: $\mathrm{argmax}_c(p_c)$.

Following the regression subsection, the logit layer can be converted to a vector of Gaussian distributions $(\mu_c(\mathbf{x}), \sigma_c(\mathbf{x}))_{c=1}^{\#\mathcal{K}}$ to include aleatoric uncertainty. However, there is no closed formula for applying a softmax function to Gaussian variables. One solution is to define a *sampling softmax function* that samples T times from the Gaussian logit layer $z_i \sim \mathcal{N}\left(\mu_c(\mathbf{x}), \sigma_c(\mathbf{x})\right)$ and approximates the softmax as $\frac{1}{T}\sum_{i=1}^{T}\mathrm{softmax}(z_i)$. For the logit layer, the results are the same as in the regression case:

$$\begin{aligned} &p(z|x) \sim \mathcal{N}(\mu_*(\mathbf{x}), \sigma_*^2(\mathbf{x})) \\ &\mu_*(\mathbf{x}) = \frac{1}{S}\sum_{j=1}^{S}\mu_j(\mathbf{x}) \quad ; \quad \sigma_*^2(\mathbf{x}) = \frac{1}{S}\sum_{j=1}^{S}\left(\sigma_j^2(\mathbf{x}) + \mu_j^2(\mathbf{x})\right) - \mu_*^2(\mathbf{x}) \\ &\sigma_*^2(\mathbf{x}) = \underbrace{\mathbb{E}_j\left[\sigma_j^2(\mathbf{x})\right]}_{\varepsilon_A} + \underbrace{\mathrm{Var}_j\left[\mu_j(\mathbf{x})\right]}_{\varepsilon_E} \end{aligned} \tag{13}$$

where S is the number of samples drawn by the model $\mathcal{M}$. By disentangling the logit layer and propagating the aleatoric/epistemic contributions through the sampling softmax function, we can derive the corresponding probabilities:

$$\begin{aligned} &p_{\varepsilon_A}(y|\mathbf{x}) = \text{sampling softmax}\left(\mu_*(\mathbf{x}), \mathbb{E}_j\left[\sigma_j^2(\mathbf{x})\right]\right) \\ &p_{\varepsilon_E}(y|\mathbf{x}) = \text{sampling softmax}\left(\mu_*(\mathbf{x}), \mathrm{Var}_j[\mu_j(\mathbf{x})]\right) \end{aligned} \tag{14}$$

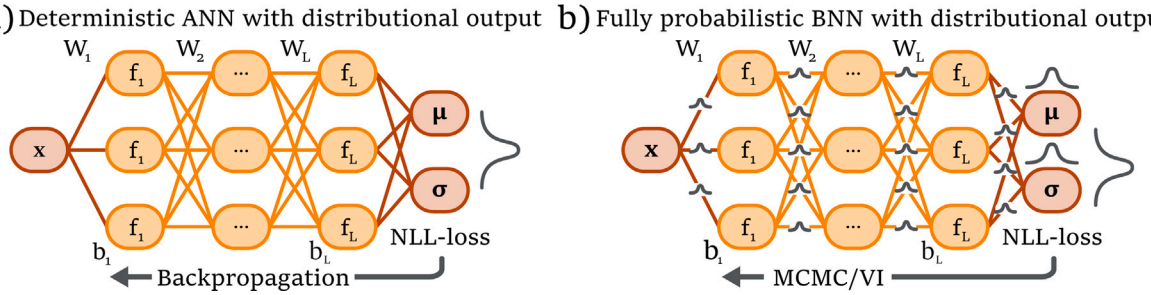

Fig. 6. (a) Deterministic ANN that outputs a distribution (e.g., a Gaussian distribution fully characterized by mean μ and std σ). The network uses a Negative Log-Likelihood loss and can train weights using classical backpropagation. (b) Fully probabilistic Bayesian Neural Network that outputs a (distributional) μ and a (distributional) σ. It is usually trained using a NLL-loss and using ad hoc techniques such as Markov Chain Monte Carlo or Variational Inference.

A measure of aleatoric/epistemic uncertainty can be computed as the Shannon entropy of $p_{\varepsilon_A}(y|\mathbf{x})$ / $p_{\varepsilon_E}(y|\mathbf{x})$ respectively:

$$\text{Shannon entropy}(p) = -\sum_c p_c \cdot \log(p_c)$$

Remark 9. Unlike in the case of regression, the transformation by the sampling softmax implies that the two entropies do not sum to the total entropy.

5. Intrusive methods: *Define a model from scratch*

Intrusive methods for uncertainty quantification, also known as **methods by-design**, are a broad family of techniques that involve proper model selection, training technique, software implementation, and even accurate data-set preparation. These approaches are typically complex to implement and require careful study design. However, they provide full insight into the uncertainties involved and are more powerful than their non-intrusive counterparts. Furthermore, even though intrusive methods usually require higher computational resources, they can work with smaller dataset sizes if a strong prior is well embedded in the experiment. Conversely, a wrong/weak prior or an inappropriate choice of method can lead to a disruptive increase in the amount of data required to converge to the solution. Among the several examples of intrusive methods for uncertainty quantification, some of the most relevant are: Artificial Neural Networks with distributional output [71], Bayesian Neural Networks [47], and Deep Gaussian Processes [72].

5.1. ANN with distributional output

The most straightforward way to estimate the variability of a prediction is to replace a deterministic output (i.e., a point estimate) with a probabilistic one (i.e., a distribution) [50,51,71]. In the same way that Gaussian regression provides a mean and variance that indicate the reliability of the model, Neural Networks can be modified to provide outputs that match the parameters for a given probability distribution. For example, if the output is assumed to be a Gaussian distribution, the Neural Network will have a final node corresponding to the mean and a second node representing the standard deviation (see Fig. 6).

Remark 10 (*How to Get Reasonable Variance Parameter?*)**.** The activation functions of these nodes can be used to impose conditions on the free parameters. For instance, a softplus activation function ($\log(1+\exp(\cdot))$ with the addition of a minimum variance $\sim 10^{-6}$ for numerical stability can force the node corresponding to the standard deviation to be positive, thus ensuring consistent results [73].

Changing the output of a network also means changing the loss used. The parameters (weights and biases) of the network can be learned by Maximum Likelihood Estimation (MLE), i.e., by minimizing the Negative Log-Likelihood (NLL) criterion using Stochastic Gradient Descent [74]. For instance, the Negative Log-Likelihood loss for standard regression is the mean squared error: $\ell(\mathbf{x}, y) = \frac{(y-\mu(\mathbf{x}))^2}{2}$, with $\mu(\mathbf{x}) = \mathcal{M}(\mathbf{x})$. Vice versa, for a two-headed (μ, σ) Gaussian ANN (assuming that the output can be modeled using a mean $\mu_\theta(\mathbf{x})$ plus a zero-mean Gaussian noise with variance $\sigma^2_\theta(\mathbf{x})$ depending on trainable parameters θ, see Fig. 5(b)) The Negative Log-Likelihood is the Variance Attenuation Loss:

$$-\log p_\theta(y|\mathbf{x}) = \frac{1}{2}\log\left(\sigma^2_\theta(\mathbf{x})\right) + \frac{\left(y-\mu_\theta(\mathbf{x})\right)^2}{2\sigma^2_\theta(\mathbf{x})} + \text{constant} \tag{15}$$

Intuitively, this loss rewards solutions close to the original ground truth ($\left(y-\mu_\theta(\mathbf{x})\right)^2 \to 0$). However, this loss tends to increase the variance $\sigma_\theta(\mathbf{x})$ to reduce the Mean Squared Error (MSE) contribution each time a good solution cannot be derived. More complex distributions, both for regression and classification, can be used by deriving the corresponding maximum likelihood estimation, see Appendix A for more details.

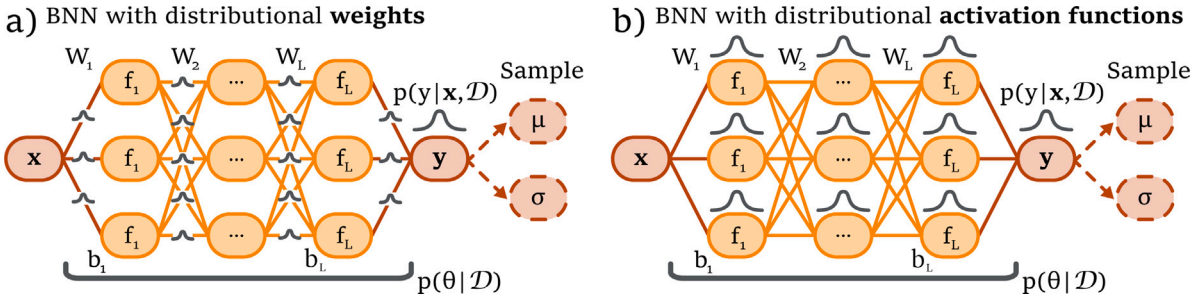

Fig. 7. (Panel a): BNN with distributional weights, the output is the predictive posterior distribution $p(y|\mathbf{x}, D)$. By sampling the output, the point estimate can be collected as the mean μ, coupled with a measure of dispersion σ (for regression tasks) or the mean probability for classification tasks. (Panel b): BNN with distributional activation functions.

Remark 11 (*Two-heads Classification*)**.** As briefly introduced in Section 4.2, for multiclass classification, the last layer of the pre-softmax network function (i.e., the so-called logit layer) can be converted to a multi-head network to approximate aleatoric uncertainties. Formally, the logit layer $\mu_\theta(\mathbf{x})$ (i.e., the vector of the raw value for each class) is substituted by a multivariate Gaussian distribution with means $\mu_\theta(\mathbf{x})$ and variances $\sigma^2_\theta(\mathbf{x})$ [50]. To derive the prediction, an empirical approximation of the softmax function (sampling softmax function) is obtained by sampling the multivariate Gaussian distribution $\mathcal{N}(\mu_\theta(\mathbf{x}), \sigma^2_\theta(\mathbf{x}))$, applying the softmax function, and averaging the result. The selected class is the arg max of this average, while a measure of aleatoric uncertainty is the Shannon entropy of the sampled softmax probabilities.

Unfortunately, this approach tends to produce overconfident variance estimates and can additionally lead to below-average mean fits. There are a few extensions to mitigate both overconfidence [51,75] and underperformance [74].

Remark 12 (*Aleatoric or Epistemic?*)**.** The simplified Gaussian assumption on the output (i.e., fully described by a mean plus a zero-mean Gaussian noise $y \sim \mu(\mathbf{x}) + \mathcal{N}\left(0, \sigma^2\right)$) is an input-dependent heteroscedastic (aleatoric) uncertainty. Hence, assuming a null epistemic uncertainty $\varepsilon_E = 0$, the variance tends to increase towards the inputs affected by a greater aleatoric uncertainty ε_A. Therefore, under simplified Gaussian assumptions, the variance can be used as an approximation for aleatoric uncertainty $\varepsilon_A \approx \mathbb{E}[\sigma_\theta(\mathbf{x})] = \sigma_\theta(\mathbf{x}) \cdot \mathbb{E}[1] = \sigma_\theta(\mathbf{x})$, see Eq. (11).

In summary, this technique is a straightforward, intrusive extension of the standard deterministic ANN capable of integrating standard prediction with reliability estimates which tends to approximate aleatoric uncertainties. Furthermore, this technique can be combined with more advanced approaches such as Bayesian Neural Networks or Ensemble to produce more complex predictive posterior distributions. However, the overconfident and poorly calibrated reliability estimates encourage the search for alternatives.

5.2. Bayesian neural networks

A Bayesian Neural Network (BNN) is a stochastic ANN trained using Bayesian inference. ANNs can incorporate stochasticity by introducing non-deterministic elements into the original architecture, either stochastic weights or activation functions [47,76–78], see Fig. 7. Bayesian Neural Networks are not strictly a category of architectures, but rather a way of approaching training and inference that interprets the learning process from a probabilistic point of view, using probability distributions instead of deterministic weights and activation functions [79]. In recent years, many architectures have been reinterpreted and implemented with a Bayesian approach, such as Adversarial Variational Bayes [80], Bayesian Graph Neural Networks [81], Bayesian Recurrent Neural Networks [82], and Bayesian Long Short-Term Memory [83].

Among the several advantages, BNNs mitigate over-fitting [52] as they are difficult to overtrain [84], allow learning from small data sets [85], and are a powerful tool for providing better calibrated uncertainty estimates compared to the model's internal score [86–88]. The main disadvantage of using BNNs over their deterministic alternatives is the increase in the number of degrees of freedom (e.g., each Gaussian weight is described by two parameters, mean and variance, instead of one). In addition, standard training methods such as backpropagation cannot be applied to stochastic networks, which rely on much more complex (and computationally expensive) methods such as Markov Chain Monte Carlo, Variational Inference, or Laplace Approximation (e.g., local post-hoc approximation of VI), see Appendix C.

BNN functional/stochastic model: A standard feedforward ANN architecture (the so-called "functional model" in the Bayesian framework) can be seen as:

$$\mathcal{M}_\theta(\mathbf{x}) := \begin{cases} v_0 = \mathbf{x} & \text{[Input]} \\ v_i = f_i\left(W_i \cdot v_{i-1} + b_i\right) & i \in [1, L] \\ y = v_L & \text{[Output]} \end{cases} \quad ; \quad \theta = \{(W_i, b_i)\}_{i=1}^{L} \tag{16}$$

where L is the number of layers, v_i is the ith layer, $\{(W_i, b_i)\}_{i=1}^{L}$ is the set of trainable parameters corresponding to the matrix of weights and the vector of biases of each layer, and f_i is an activation function (such as ReLU, Sigmoid, TanH, etc.). The BNN counterpart shares the original hyperparameters ($\hat{h}$: number of layers, learning rate, etc...) and is obtained by replacing the deterministic weights W_i with a suitable distribution or by introducing stochasticity in the activation functions f_i. Several other ways to incorporate stochasticity can be derived from Probabilistic Graphical Theory [89].

Remark 13 (*Stochastic Weights vs. Stochastic Activation Functions*)**.** Stochastic activation functions can be used to compress variational parameters during variational inference [90], but are less commonly used in practice due to their more complex formulation and implementation compared to stochastic weights. Moreover, in several cases, the use of stochastic activation functions has equivalent formulation using stochastic weights [9].

It should be noted that even the use of elementary weight/activation function distributions can lead to a highly non-Gaussian predictive posterior due to the non-linearity of the network. Similarly, the degrees of freedom of the BNN can be reduced by making only a few weights/activation functions stochastic instead of every element in the network.

Model induction: For standard ANNs, training is usually done by minimizing a cost function (e.g., the log-likelihood of the training set plus a regularization term) to obtain the best model trainable parameters $\hat{\theta}$ using efficient methods such as backpropagation (with a stochastic gradient descent algorithm, ADAM, etc.). However, standard sampling approaches fail to determine the posterior $p(\theta|\mathcal{D})$, which is highly non-convex and high-dimensional [91] (e.g., $|\theta| \sim 2.3 \cdot 10^7$ for ResNet50) and does not admit a closed-form solution. To approximate the posterior (i.e., to be able to sample from $p(\theta|\mathcal{D})$), various numerical approaches have been adopted. The most effective are represented by proper conditional distribution sampling, such as **Markov Chain Monte Carlo** [92,93], and approximate distribution approaches, such as **Variational Inference** [53,94] (see Appendix C). According to Eq. (10), the posterior can only be derived after choosing a prior $p(\theta)$. This choice is a major drawback of the Bayesian approach, as the posterior distribution is strongly dependent on the prior. When no strong knowledge of the prior distribution can be provided (as in most real-world DL scenarios), one possibility is to choose a non-informative (weak) prior that can be washed out by a small amount of data, such as a uniform distribution over the entire range of parameter variation or a diffuse normal distribution (i.e., high variance) [95]. It should be noted that even in the deterministic setting, the prior is implicitly chosen when the objective function is defined, so BNNs simply make the choice of prior explicit.

Remark 14 (*Explicit Prior and Regularization*)**.** The regularization function in the deterministic setting acts as a prior in the Bayesian framework [9]. Indeed, given a cost function, if the loss is the Negative Log-Likelihood, it holds:

$$\begin{aligned}\hat{\theta} &= \operatorname{argmin}_\theta \operatorname{Cost}\left(\mathcal{M}_\theta, \mathcal{D}\right) = \operatorname{argmax}_\theta \log p\left(\mathcal{D}_y|\mathcal{D}_x, \theta\right) \approx \operatorname{argmax}_\theta \log p\left(\mathcal{D}_y|\mathcal{D}_x, \theta\right) \cdot p(\theta) \\ &= \operatorname{argmin}_\theta \operatorname{Cost}\left(\mathcal{M}_\theta, \mathcal{D}\right) + \operatorname{Regularisation}(\theta)\end{aligned}$$

If we are refining an existing model, the prior can be derived from the trained parameters θ. One possible approach is to define the prior as a multivariate diffuse normal distribution with means equal to the (deterministic) trained parameters and high variances (weak prior). However, for new models, a common choice is to set $p(\theta) = \mathcal{N}(0, \sigma \cdot I)$, which corresponds to the L_2 regularization in the deterministic setting, and thus reduces the difficulties of weight space and scaling symmetry [96]. The use of the normal law is dictated by the fact that it has a reduced increase in the number of parameters (from 1 to 2 trainable parameters) and due to the good theoretical properties of Gaussian distributions, including the simple formulation of its log. However, there are no strong theoretical arguments for preferring it to other more complex distributions corresponding to different regularization strategies [97].

Model Prediction: Given the data-set $\mathcal{D}$ and the posterior distribution $p(\theta|\mathcal{D})$, the predictive posterior distribution

$$p(y|\mathbf{x}, \mathcal{D}) = \int \underbrace{p\left(y|\mathbf{x}, \theta'\right)}_{prediction} \underbrace{p\left(\theta'|\mathcal{D}\right)}_{density} d\theta'$$

on unseen data $\mathbf{x}$ can be derived by sampling from the posterior distribution $\hat{\theta}^{(j)} \sim p(\hat{\theta}) = p(\theta|\mathcal{D})$ and collecting the results $\{y^{(j)}\}_j = \{\mathcal{M}_{\hat{\theta}^{(j)}}(\mathbf{x})\}_j$. In fact, due to the random components, a forward pass through a BNN can be seen as sampling a set of values for the weights/activation functions and applying the associated deterministic ANNs. This can be interpreted as considering $\mathcal{M}_{\hat{\theta},\hat{h}}$ as an (infinite) collection (ensemble) of models $\{\mathcal{M}_{\hat{\theta}^{(1)},\hat{h}}, \mathcal{M}_{\hat{\theta}^{(2)},\hat{h}}, \dots\}$ whose trainable parameters $\hat{\theta}^{(j)}$ are realizations of $\hat{\theta}$ [98]. Thus, BNNs can be seen as a special case of ensemble learning (where the stochastic parameters are forced to follow a simplified, usually Gaussian, distribution during VI) [99]. The **predictive posterior distribution** $y|\mathbf{x}, \hat{\theta} \sim \mathcal{M}_{\hat{\theta},\hat{h}} + \varepsilon_A + \varepsilon_E$ can be approximated from multiple model evaluations $\{\mathcal{M}_{\hat{\theta}^{(1)},\hat{h}}(\mathbf{x}), \mathcal{M}_{\hat{\theta}^{(2)},\hat{h}}(\mathbf{x}), \dots\}$. The output of the model can be the overall approximate distribution or an average point estimate (e.g., mean/median/mode) coupled with a measure of dispersion (e.g., variance/interquartile range). In particular for regression task, following Eq. (11), the predictions $\mu_j = y^{(j)} = \mathcal{M}_{\hat{\theta}^{(j)}}(\mathbf{x})$ can be averaged to obtain a point estimate $\hat{y} = \frac{1}{S}\sum_{j=1}^{S} \mu_j(\mathbf{x}) \approx \mathbb{E}_j\left[\mu_j(\mathbf{x})\right]$. A measure of reliability is the variance of the predictions $\frac{1}{S-1}\sum_{j=1}^{S}\left(\mu_j(\mathbf{x}) - \hat{y}\right)^2 \approx \operatorname{Var}_j\left[\mu_j(\mathbf{x})\right]$ which, under the normal assumptions on the predictive posterior distribution, represents the Epistemic Uncertainty ε_E.

For a (multi-class) classification task, the last layer of the BNN contains one node for each possible outcome $k \in \mathcal{K}$. Each time the model $\mathcal{M}_{\hat{\theta}^{(j)}}(\mathbf{x})$ is called for inference, it returns a probability vector $\{(p_i^{(j)})_{i=1}^{\#\mathcal{K}}\}_{j=1}^{S}$, where S is the number of samples used for inference. The predictions are averaged ($\hat{p} = (\hat{p}_i)_{i=1}^{\#\mathcal{K}} = \frac{1}{S}\sum_{j=1}^{S}\mathcal{M}_{\hat{\theta}^{(j)}}(\mathbf{x})$) and the point estimate is defined as the most likely averaged

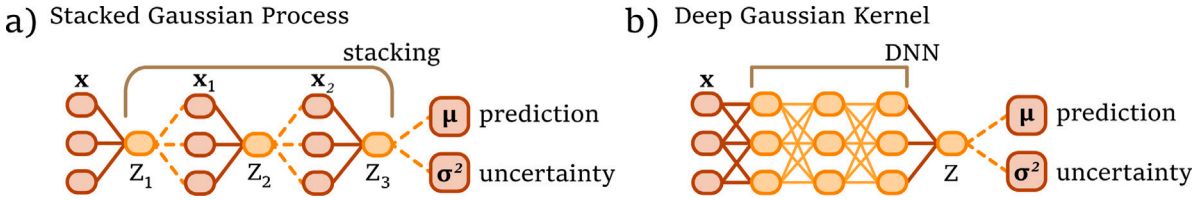

Fig. 8. Deep Gaussian Processes. (Panel a): Stacked Gaussian Process obtained with multiple layers of classical GPs. (Panel b): Stacked Gaussian Process, which uses a Deep Neural Network as the kernel function of a classical GP.

class: $\hat{y} = \text{argmax}_i\, \hat{p}_i$. A possible measure of reliability can be derived from the variance of the averaged probability $\text{Var}_i\left[\hat{p}_i\right]$ or as the variance of the probabilities of the predicted class among the samples $\text{Var}_j[p_{\hat{y}}^{(j)}]$.

Remark 15 (*This Evaluation Seems Really Expensive!*)**.** Iterating thousands of model evaluations to extrapolate the distribution $y|\mathbf{x}, \hat{\theta}$ can be described as forward uncertainty propagation [100], which relies on an appropriate sampling strategy to be computationally efficient. Unfortunately, most efficient sampling strategies suffer from the curse of dimensionality (including quasi-Monte Carlo low discrepancy sequence [101]) and are therefore unsuitable for ANNs whose number of parameters m is usually very large. Therefore, in most cases, inference on y must be done using standard Monte Carlo/Latin Hypercube approaches, possibly coupled with an appropriate halting criterion to avoid unnecessary model evaluations [102].

As described in Section 5.1, Bayesian Neural Networks can be modified to explicitly output a distribution (e.g., Gaussian, LogNormal, Bernoulli, etc.) rather than a pointwise output. In such cases, the NLL loss should be adapted to the multihead output and the induction should vary accordingly (by using BNN training strategies such as MCMC and VI). Following Section 5.1, the combination of multihead output with Bayesian weights/activation functions is used as a starting point for uncertainty disentanglement [2,65,103]. Indeed, under Gaussian assumptions on the predictive posterior distribution, the distributional output $\mu_j(\mathbf{x}), \sigma_j(\mathbf{x})$ can be used to provide an approximation of both epistemic ($\varepsilon_E \approx \text{Var}_j\left[\mu_j\right]$) and aleatoric uncertainty ($\varepsilon_A \approx \mathbb{E}_j\left[\sigma_j^2(\mathbf{x})\right]$).

In conclusion, BNNs are powerful intrusive models that provide insight into the shape of the predictive posterior distribution, using a more complete and computationally intensive theoretical framework than the deterministic method, and associating a reliability index with the pointwise estimate.

5.3. Deep Gaussian processes

Classical Gaussian Process (GP) Regression [104,105] is a powerful metamodeling technique that interpolates the data to provide an estimate of the predictive posterior distribution even with extremely small data-sets [106]. Classical GP (see Appendix B) is often used to provide a mathematical background to other Bayesian techniques, such as MC-Dropout (Section 6.1), since it can be proved that under certain conditions a GP is a multilayer perceptron with infinite units in the hidden layer [107]. A GP $Z(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$ is a stochastic process fully characterized by a mean function $m(\mathbf{x})$ and a parameterized covariance (or kernel) function $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}'|\varphi)$.

However, classical GPs are ineffective at handling high-dimensional data, such as complex input vectors or images, because most commonly used kernel functions are based on simplistic similarity metrics [72]. Furthermore, the choice of an optimal kernel function capable of extracting useful features for computing the similarity between samples is sometimes challenging, especially for structured data where hierarchical feature extraction needs to be considered [54]. For this reason, **Deep Gaussian Processes** were introduced to combine the ability of Deep Neural Networks to learn high-dimensional features with the flexible framework of classical GPs [72]. The two main paradigms for integrating DNNs with GPs are: **Stacked Gaussian Processes** [54] and **Deep Kernel Learning** [56].

A strategy for improving Gaussian Processes to capture complex relationships is represented by the **Stacked Gaussian Processes** (sometimes called Deep Gaussian Process) [54,55]. This involves combining several GPs so that the output of the previous GP is used as the input of the next GP, similar to stacking perceptrons in a multilayer perceptron (Fig. 8(a)). From the original work of Lawrence et al. [55], with a GP using a second GP as an input prior, it was clear that the main difficulty with stacked GPs was the lack of an analytical solution. To overcome the risk of over-fitting of the Maximum A Posteriori estimation (MAP) approach, Damianou et al. proposed a variational approach that can be applied to a Stacked GP with an arbitrary number of layers [54,108] and a back constraint to limit the number of variational parameters to be learned [109]. However, even with these improvements, the assumption of GP layer independence leads to solutions with hidden GP layers turned off that are equivalent to single-layer GP. This major drawback can be overcome by using a variational bound, which preserves the exact posterior of the model while maintaining the correlation within and between adjacent layers [110]. Unfortunately, this approach requires the use of costly MCMC

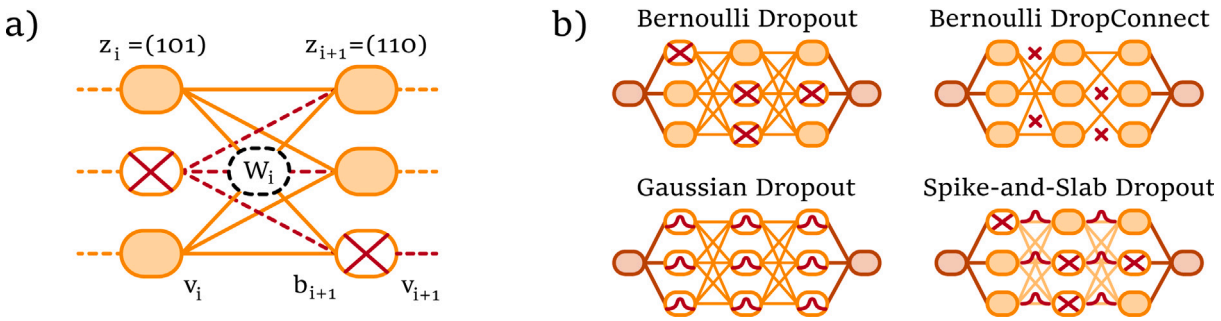

Fig. 9. (Panel a): Dropout of units corresponding to 0 outcomes of a Bernoulli distribution z. (Panel b): Alternatives to the standard Bernoulli dropout that act on the connection (Bernoulli DropConnect), use a continuous dropout function (Gaussian Dropout), or act on both units and connections (Spike-and-Slab Dropout).

strategies, making the use of Stacked GP extremely computationally expensive. It should also be noted that a stacked GP would no longer be a GP since the posterior distribution can be any (non-Gaussian) distribution.

An alternative to the Stacked Gaussian Process is **Deep Kernel Learning** [56], which aims to construct kernels that encapsulate the expressive power of deep architectures to overcome the limitations of simple classical ones. Formally, given a parameterized (classical) kernel $k(\mathbf{x}, \mathbf{x}'|\varphi)$ and a DNN $\mathcal{N}_\theta$, a Deep Kernel function is defined as $k(\mathcal{N}_\theta(\mathbf{x}), \mathcal{N}_\theta(\mathbf{x}')|\varphi)$, which can be seen as a GP applied on the last layer of a Deep Neural Network (Fig. 8(b)). This allows a natural extension of standard GP to images/videos by using Convolutional Neural Networks to define kernels that evaluate shape and contours.

Remark 16 (*Sparse Formalism and Speed Up*)**.** There are several extensions of naive DGP that use a sparse matrix of interpolation weights on the covariance matrix generated by the Deep Kernel function, thus providing an approximation of the original Deep Kernel (i.e., $K\gamma \approx K_{\text{sparse}}$) [111]. This allows for fast computations both in the training phase (almost linear scaling vs. potentially cubic scaling of classical GP/DGP [112]) and $\mathcal{O}(1)$ time per test point prediction.

In summary, Deep Gaussian Processes are highly intrusive methods that can lead to state-of-the-art models by combining the probabilistic properties of standard Gaussian Processes with the shape (Stacked GP) or flexibility (Deep Kernel GP) of DNNs. However, these techniques have several drawbacks in terms of memory, computational cost, and the a priori decision on the shape of the underlying DNN architecture. Although most of these limitations can be addressed, this family of methods is among the most complex to handle for uncertainty quantification tasks.

6. Non intrusive methods: *Post-Hoc approaches*

A more practical approach to quantifying the uncertainty of a pre-trained model is represented by the so-called "post-hoc methods", which are non-intrusive and cannot affect the predictive performance of the models, but uniquely deal with quantifying their uncertainty. Post-hoc methods have the great advantage of not having to rerun the models, thus saving computationally expensive data processing, but conversely are usually barely acceptable for providing reliable uncertainty estimates. Some post-hoc approaches can be derived from local approximations of intrusive methods (e.g., Laplace approximation, see Appendix C).

6.1. Monte Carlo (MC) dropout

Dropout (also known in the UQ setting as Monte Carlo Dropout or Model Averaging) was originally introduced as a technique to reduce model over-fitting [59,60]. Given an arbitrary Neural Network, the output of the ith layer can be written as a non-linear function of a linear combination of the values of the layer $v_i = f\left(W_i \cdot v_{i-1} + b_i\right)$, where W_i is the matrix of weights between the layers, b_i is the vector of biases and f is the activation function (see Fig. 9). With Dropout regularization, each layer i (made up of s units) except the last is coupled to a multivariate binary random variable (usually following a Bernoulli distribution $\mathcal{B}(q_i) \mid q_i \in [0,1]$). During the training phase, one realization $z_i = (z_{i,1}, \ldots, z_{i,s})$ of this distribution is sampled for each backward pass. Units corresponding to a score of 0 are dropped, while units corresponding to a score of 1 are retained: $v_i = f\left(W_i \cdot z_i^T \cdot \text{diag}\left(v_{i-1}\right) + b_i\right)$.

Formally, dropping a unit is equivalent to setting its value to 0 (even during backward propagation of derivatives). By alternately resampling the binary variable for each input point and each forward pass, using the same values for backpropagation, the network converges smoothly with a reduced risk of over-fitting [59,60].

Remark 17 (*Can We Accidentally Disconnect the Network?*)**.** By using a high dropout value, it appears that parts of the ANN can be accidentally disconnected. However, dropout is applied at the layer level (i.e., $q = 0.5$ means that 50% of the units in each layer will be dropped, not that 50% of all units will be dropped). Furthermore, most Dropout implementations do not sample from a Bernoulli distribution (with the risk of dropping every single unit in a layer), but instead drop a proportion of the layer weights exactly equal to the dropout parameter. Therefore, no layers can be disconnected following the dropout procedure.

A cost function consisting of an NLL loss combined with a L_2 regularization can be used to provide a theoretical framework for Dropout regularization:

$$\text{cost}_{dropout} = \frac{1}{N}\sum_{i=1}^{N} \text{NLL}(y_i, \mathcal{M}(\mathbf{x}_i)) + \lambda \sum_{j=1}^{L} \left(\|W_i\|_2^2 + \|b_i\|_2^2\right)$$

where N is the size of the training set, L is the number of layers, and NLL is a given loss (e.g., softmax for classification tasks, square "Euclidean" loss for regression tasks, etc.). It can be proved that minimizing this loss function is equivalent to minimizing the Kullback–Leibler divergence between an approximate distribution and the posterior of a Deep Gaussian process. Therefore, a network trained with Dropout using a cost function that includes a regularization term that acts as a prior (e.g. L_2 normalization) turns a standard ANN into an approximated BNN [9].

Remark 18 (*Why Does It Work?*)**.** The reason why Dropout regularization can drastically reduce over-fitting is that it can perform as stochastic gradient descent on a regularized error function, and is equivalent to a L_2 regularizer applied after proper scaling of the features [113,114]. In addition, an arbitrary ANN with dropout applied before each weight layer is equivalent to an approximation of the probabilistic Deep Gaussian Process (marginalized over its covariance function parameters) [115]. The Deep Gaussian Process inherits the ability to avoid over-fitting of classical Gaussian Processes [54] and therefore the aforementioned approximation provides an explanation for the use of Dropout as a practice to avoid over-fitting.

Changing the dropout parameter q_i leads to a variation in the magnitude of the output of the layer (e.g., applying a $q_i \sim 50\%$ will drop half the units and therefore reduce the magnitude of the outputs). If a network layer is trained to receive values from only a fraction $(1 - q_i)$ of the previous layer due to dropout, then at inference time, when the entire previous layer contributes to the result, model performance may degrade. To avoid this pitfall, whenever a Bernoulli dropout is applied (either in the training phase or in the inference phase), it is sufficient to apply a scaling factor to the weights equal to $1/(1 - q_i)$ to compensate for the missing nodes.

As a remark, Dropout can also use other distributions than Bernoulli, such as Gaussian Dropout, which uses multiplicative random normal values with unitary mean $\epsilon \sim \mathcal{N}(1, \sigma^2)$.

In addition, the dropout can be applied to the connection (Bernoulli/Gaussian DropConnect) instead of to the units, or to both connections and units (Spike-and-Slap Dropout) [116].

Remark 19 (*Optimal Dropout Parameter*)**.** There is no consensus on the optimal dropout parameter, which is strongly dependent on the size of the training data set. In fact, even if a dropout value of 0.5 is reported to be optimal for various large data-set tasks [60,117], even smaller values can lead to a significant increase in performance compared to naive ANN [118]. Conversely, it should be noted that higher dropout rates slow down convergence [117] and may be suboptimal for small data sets [118].

Dropout as post-hoc method: While standard dropout is only a technique to prevent overfitting during training, a slight modification of the above procedure can be used at inference time to provide a post hoc approximation of the posterior predictive distribution $p(y|\mathbf{x})$. Moment matching can be used to derive formulae for higher order statistics [10]. An empirical approximation of the first two moments (mean, used as a more stable point estimate, and variance, used as a measure of reliability) can be derived from these model evaluations. The mean can be computed by performing S stochastic forward passes through the network and averaging the results. Similarly, the model predictive variance can be approximated as the sample variance of the same S stochastic forward passes through the ANN (plus a bias term equal to the inverse of the model precision [115]). With respect to this bias, the model precision can be approximated using Bayesian optimization over validation log-likelihood [119] or otherwise dropped while maintaining a biased estimate of the variance.

Therefore, at inference time, S model evaluations are performed by resampling the Bernoulli distribution and performing a forward pass. By applying a Bernoulli dropout at a given layer (of factor q_i), the weights must also be rescaled by a factor of $1/(1-q_i)$. This very large number of forward evaluations (i.e. the convergence rate of the standard MC approximation is $\mathcal{O}(1/\sqrt{\#Sample})$, so an accuracy of $\sim$1% requires about 10^4 samples) can be time-consuming, even though the task is embarrassingly parallelizable.

Remark 20 (*How to Calibrate the Reliability Score?*)**.** Gaussian Processes, and consequently MC-Dropout, are not calibrated. In fact, MC Dropout is an approximation of an underlying Gaussian process whose uncertainty depends on the prior choice of hyperparameters and covariance function. Different activation functions (ReLU, TanH, etc.) or different dropout distributions (Bernoulli, $\mathcal{N}(1, \sigma^2)$, etc.) lead to a MC Dropout that approximates a different GP. A standard approach to scale uncertainty and understand if the prediction is reliable is to fit an elementary distribution (Gaussian, LogNormal, etc.) to the uncertainty of the training set and evaluate if the prediction belongs to a high percentile of the cumulative distribution function.

In conclusion, MC Dropout is a post-hoc technique that approximates VI on an associated Deep Gaussian Process, thus providing a simple and embarrassingly parallelizable method to derive an estimate of Epistemic Uncertainty [120] and, with proper extension, Aleatoric Uncertainty as well [121]. The main drawback of MC dropout is that the Bayesian approximation of the predictive posterior

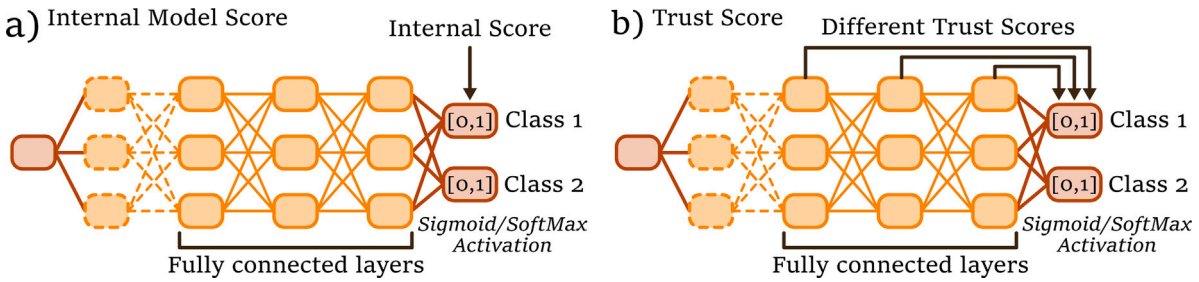

Fig. 10. (Panel a): Internal Model reliability score calculated from the Sigmoid/SoftMax activation values for classification tasks. (Panel b): Trust Score calculated from one of the fully connected layers.

distribution relies on the Gaussian Process equivalence, which may not always be satisfied [10]. Indeed, there is evidence that MC Dropout does not always fully capture the predictive uncertainty [122].

6.2. *Internal score, trust score, et simila*

In several real-world scenarios, practitioners aim to derive an estimate of the predictive posterior probability without further training. Therefore, the literature provides a plethora of post-hoc scores that should allow an assessment of the reliability of the predictions. However, most of these scores lack a theoretical basis and therefore, although useful in practice as an indicator of overconfident results, they do not approximate the predictive posterior distribution (see Fig. 10).

A common practice in Machine Learning is to use the **Internal Model Score** as an approximation of the predictive posterior distribution. For example, a Deep Learning Classifier typically has a final layer that includes a *sigmoid* activation function (binary classification) or a *softMax* activation function (multiclass classification). These functions return a normalized value ($[0,1]$, the Internal Score) which is often referred to as a "probability". This score can be used as a measure of reliability (i.e. higher internal scores correspond to better predictions, while lower scores may indicate unreliable results). However, the use of these Internal Scores should be discouraged as they tend to produce poorly calibrated measures of reliability [61,123]. Confidence Calibration is a related line of work that aims to transform classifier outputs into probability values [123–126]. However, even after calibration, the internal scores tend to be overconfident [127–129], incorrectly reporting as 'reliable' predictions with a very high variance in the predicted posterior distribution.

Among the various post-hoc alternatives to the use of Internal Scores, a technique with a solid mathematical background and a few-shot learning fashion [63] is the **Trust Score** [62]. Formally, given a test sample x, a classifier $\mathcal{M}$ (with a set of possible classes $\mathcal{K}$), and the set $\mathrm{S}(K_i)$ of the training set corresponding to the given class K_i, the Trust Score is defined as the ratio between the distance of the test sample to $\mathrm{S}(K_i)$ of the nearest class other than the predicted class and the distance of the test sample to the set of the class predicted by the classifier $\mathcal{M}$:

$$\begin{cases} \mathrm{S}(K_i) := \{(\mathbf{x}_j, y_j) \in \mathcal{X} \times \mathcal{Y} \mid y_j \in K_i\} \\ O^{2\text{nd}}(\mathbf{x}) := \operatorname{argmin}_{K_i \in \mathcal{K} \mid K_i \neq \mathcal{M}(\mathbf{x})} d\left(\mathbf{x}, S(K_i)\right) \\ \mathbf{TS}(\mathbf{x}) := \frac{d\left(\mathbf{x}, S\left(O^{2\text{nd}}(\mathbf{x})\right)\right)}{d(\mathbf{x}, S(\mathcal{M}(\mathbf{x})))} \in [0, \infty) \end{cases} \tag{17}$$

where $O^{2\text{nd}}(x)$ is the second optimal closest class, d is an arbitrary distance (Euclidean k-nearest neighbor, distance from the centroid, etc.) computed on the last dense layer of the Artificial Neural Network, and $\mathbf{TS}(x)$ is the trust score of the input x.

It can be shown that for labeled data distributions with well-behaved class boundaries, the classifier is likely to agree with the optimal Bayesian Classifier when the Trust Score is large [62]. However, for high-dimensional feature spaces (such as those computed on ViT or VGGs), the reliability measure obtained is less informative, and the fact that the trust score has no upper bound makes the results difficult to interpret.

In conclusion, post-hoc methods should generally be replaced by other non-intrusive (MC dropout) or semi-intrusive (Ensemble) techniques. However, a few theoretically well-founded methods (such as Trust Score) can add a reliability score at almost null computational/memory cost, thus improving the standard deterministic classification.

7. Semi-intrusive methods: *From deterministic to Bayesian*

The intrusive methods described in Section 5 are accurate and complex techniques for uncertainty quantification but are often characterized by high computational costs. On the other hand, the non-intrusive methods, described in Section 6, are less accurate than the intrusive models but can save time and computational costs. Between intrusive and non-intrusive techniques, semi-intrusive

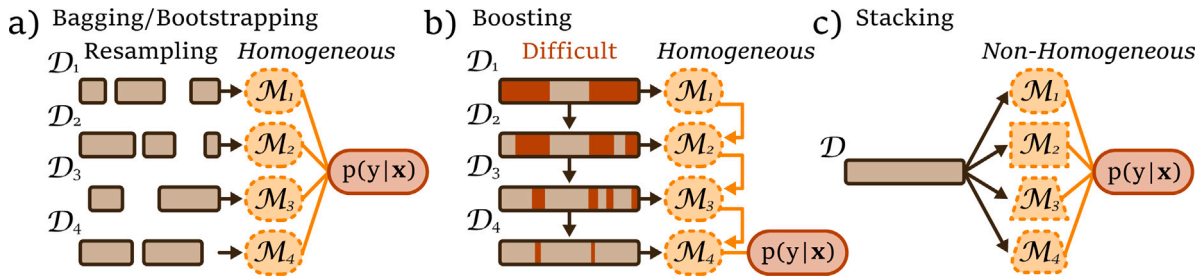

Fig. 11. Plots of the three main ensembling paradigms: Bagging (Panel a), based on random resampling of homogeneous models; Boosting (Panel b), based on sequential fitting of difficult data; and Stacking (Panel c): based on non-homogeneous/highly specialized models.

techniques represent an efficient alternative approach that combines the accuracy of intrusive methods with the ease of non-intrusive ones. In fact, semi-intrusive methods modify the architecture and training procedure of the original deterministic model with fewer modifications than intrusive approaches, while still obtaining a better reliability estimate compared to post-hoc strategies. Among them, the most relevant are the Ensemble methods, a family of approaches that combine multiple deterministic sub-models to avoid over-fitting and to quantify uncertainty [57].

7.1. Ensemble methods

Ensemble methods are among the best techniques for improving the predictive ability and providing an estimate of the uncertainty of ML models. **Ensemble Methods** represent a learning paradigm in which multiple ML models are trained to solve the same problem and combined to achieve better results [57]. The naïve idea behind ensemble learning is that multiple predictions from models trained on different data or with different hyperparameters can provide a combined estimate that is less prone to over-fitting than the disjoint ones. Formally, instead of training a single model $\mathcal{M}_{\hat{\theta},\hat{h}}$, we define a parameterized family of models $\mathcal{E} = \{\mathcal{M}_{\hat{\theta}^{(j)},\hat{h}^{(j)}}\}_{j=1}^{S}$. The point estimate $\mathcal{M}_{\hat{\theta},\hat{h}}(\mathbf{x})$ is substituted by an aggregate estimate (usually the mean across predictions for regression tasks: $\mathcal{E}(\mathbf{x}) = \frac{1}{S}\sum_{j=1}^{S}\mathcal{M}_{\hat{\theta}^{(j)},\hat{h}^{(j)}}(\mathbf{x})$ and the argmax of the average of the outputs of the softmax layer: $\mathcal{E}(\mathbf{x}) = \text{argmax}_{class}\left(\frac{1}{S}\sum_{j=1}^{S}\mathcal{M}_{\hat{\theta}^{(j)},\hat{h}^{(j)}}(\mathbf{x})\right)$ / the most frequent class for classification tasks). The main drawbacks of ensemble methods are: how to define an optimal strategy to derive the sub-models, and the computational/memory cost both at training and inference time when S is large. In fact, when ML or DL models are used as part of the ensemble, the memory, and computational cost are not suitable for most applications [58].

As reported in Fig. 11, there are three main ensembling paradigms to define an optimal family $\mathcal{E}$: **Bagging**, **Boosting**, and **Stacking**. Bagging is an example of a randomization-based approach where the (homogeneous) ensemble members can be trained in parallel without any interaction, while boosting-based approaches train the ensemble members sequentially, improving performance at a high computational cost.

In **Bagging** (also called **Bootstrapping**) S bootstrap sets $\mathcal{D}_j$ of randomly sampled observations are generated from the initial data-set $\mathcal{D}$. Each of these sets should be representative of the original data distribution and independent of each other [130]. Each model $\mathcal{M}_{\hat{\theta}_j,\hat{h}}$ is trained on a different sub-set $\mathcal{D}_j$ so that the $\mathcal{E}$ encompasses the variability of the original data distribution. Due to the independence of the bootstrap sets, the training process of the models can be performed in parallel (both through embarrassing parallelization and efficient parallelization), thus reducing the computational time. If the base learner lacks intrinsic randomization (e.g., can be obtained by solving a convex optimization problem, as in the case of random forests [131]), Bagging can introduce diversity. Conversely, if the base learner has multiple local optima (such as ANNs and especially DNNs), bootstrapping can actually reduce performance. In such cases, the entire data set $\mathcal{D}$ can be used for each $\mathcal{M}_{\hat{\theta}_j,\hat{h}}$, taking advantage of the multiple optima to provide differences between ensemble sub-models [73].

Remark 21 (*Deep Ensemble*)**.** Deep Ensemble is a semi-intrusive bootstrapping method that requires fewer changes to the model definition and training algorithms than other ensemble techniques [73]. Exploiting the inherent randomness of the Deep Learning training process and the multiple optima typical of ANNs, Deep Ensembles start from homogeneous models with random initialization and train them multiple times on the randomly reshuffled full data-set (or on a bootstrapping sub-set if enough data are available). By using an appropriate scoring rule (i.e., a measure of the quality of the prediction uncertainty, such as log-likelihood for ANNs) as the loss function, we can train the ensemble using standard backpropagation [132]. To smooth the predictive distributions and further differentiate the sub-models, an appropriate data augmentation strategy coupled with an adversarial smoothing technique can be employed [128].

Boosting methods aim to improve the accuracy of individual models using an adaptive sequential training approach [133]. In particular, each model $\mathcal{M}_{\hat{\theta}^{(j)},\hat{h}}$ focuses on observations that the previous model $\mathcal{M}_{\hat{\theta}^{(j-1)},\hat{h}}$ had difficulty dealing with. Because the calculations to fit the different models cannot be done in parallel (unlike bagging), boosting is considered a computationally

expensive technique. The most commonly used boosting methods are Adaptive Boosting (AdaBoost) [134] and Extreme Gradient Boosting (XGBoost) [135].

Remark 22 (*Hyperensemble*)**.** Hyperensemble is a variation of the standard bagging/boosting strategy. Instead of using homogeneous models (i.e., with the same initial parameters/hyperparameters $\hat{h}$), the sub-models $\{\mathcal{M}_{\hat{\theta}_j,\hat{h}_j}\}_{j=1}^{S}$ are also characterized by a slight variation of their hyperparameters $\hat{h}_j$, such as learning rate, kernel size or optimization strategy [136].

The third ensembling paradigm is **Stacking**. Stacking combines heterogeneous learning models (usually highly specialized for different facets of the problem) [137]. The combination is done by training a meta-model to make predictions based on the multiple predictions returned by different models. The metamodel, usually an ANN, takes the model outputs as inputs and is trained on them to make more robust predictions and assess reliability. A natural extension of stacking is Multilevel Stacking, which consists of multiple layers of ensembles stacked on sub-layers [138].

After training, the inference is done by set aggregation, and uncertainty is obtained from the variance of the prediction (regression) or the mean entropy (classification), see Eq. (11). In particular, if the ensemble $\mathcal{E}$ ($\#\mathcal{E} = S$) is treated as a uniformly-weighted mixture model, the combined prediction is simply $\hat{y} = \frac{1}{S}\sum_{j=1}^{S} M_{\hat{\theta}(j)}(\mathbf{x})$. Under the simplified assumptions on the predictive posterior distribution (see Section 4), Ensembles can be used to approximate epistemic uncertainty as $\varepsilon_E \approx \mathrm{Var}_j\left[M_{\hat{\theta}(j)}(\mathbf{x})\right]$. In addition, if the sub-models of the ensemble provide distributional outputs (i.e., are ANNs with distributional outputs or BNNs), the ensemble can also be used to approximate the aleatoric uncertainty (see Eq. (12)).

Remark 23 (*How Many Models Make Up an Ensemble?*)**.** Determining the optimal number of models to form an ensemble is still an open problem. Therefore, the available computing power is usually the main bottleneck for the number of trained models. Iterative approaches can be used to add more models until a convergence criterion is reached or the available resources are exceeded.

In summary, Ensembles are semi-intrusive highly parallelizable methods that can be applied with a slight modification of the cost function (i.e., by using an appropriate scoring rule). The ease of training, the ability to provide an estimate of predictive uncertainty, and the improved performance compared to point estimation make this family of techniques the gold standard for UQ estimation. However, the high computational/memory cost of training and handling tens or even hundreds of models makes them difficult to apply to most real-world scenarios.

8. Conclusion

Uncertainty Quantification has become a crucial aspect in dealing with Machine Learning contexts, especially in Deep Learning ones. In fact, due to the high risk of Artificial Neural Networks over-specializing on the problem, leading to low generalization capability, the real-world application of these methods is strongly dependent on the assessment of the reliability of the predictions. With this work, we hope to provide the scientific community with a useful resource to better navigate the emergent, vast, and often challenging field of supervised Bayesian uncertainty analysis. First, we tried to define the different types of uncertainty, taking into account their relationship with the chosen model, the trainable parameters, and the learning context. In particular, we have shown the reader, in a mathematically rigorous yet illustrative manner, the fundamental difference between those types of variability that can be controlled simply by increasing the numerosity of the data set, and those that require active modification of the model used. These uncertainties can be integrated by using appropriate techniques beyond the use of internal scores. However, each of these methods has its own implementation difficulties and corresponding computational costs. Thus, the practitioner may feel overwhelmed by the different methods available and end up implementing deterministic techniques, even when the available resources are sufficient to use Bayesian techniques. Therefore, we have analyzed the frameworks (probabilistic/deterministic), intrusiveness, advantages, and disadvantages of some of the main approaches to quantifying uncertainty in Deep Learning that are currently state of the art: ranging from highly intrusive and efficient techniques (such as Bayesian Neural Networks with distributional output) to post-hoc methods with almost zero computational cost (MC Dropout, Trust Score, etc.). Formal derivation through consistent language and formulae between the different sections will hopefully assist in a more detailed understanding of the techniques presented, thus avoiding many of the errors found even in the technical literature.

The landscape of UQ techniques is diverse, ranging from Bayesian methods to several families of techniques that are not purely probabilistic. Where, for the sake of simplicity in the presentation of key concepts, it has been necessary to make a sharp selection of what we believe to be the cornerstones of Bayesian UQ methods, it is useful to remember that this article does not exhaust the plethora of modern techniques for uncertainty quantification. Examples of further intrusive techniques are the generative methods for UQ [139] or, among post-hoc approaches, Conformal prediction methods [140]. Future work will focus on investigating the critical issues identified in the review, both theoretically and experimentally. In particular, it is crucial to determine the capability of the different techniques to estimate each type of uncertainty, their ability to disentangle them, and the sample sizes that are considered appropriate for their use. Therefore, we will focus on providing a rigorous comparison between different techniques (in terms of OOD detection, posterior approximation, and uncertainty disentanglement) using standardized procedures (see [66] for disentanglement experiments) using ad hoc synthetic datasets for UQ investigation [69,141]. In addition, we associated this review with the git containing the open source codes of the aforementioned techniques and experiments [142].

In conclusion, this review stands as a vademecum of the state of the art of supervised Bayesian uncertainty analyses, specifying the drawbacks not only from a theoretical point of view but also in terms of implementation complexity and intrusiveness of use and leading practitioners to view Artificial Neural Networks as inextricably linked to the uncertainty they introduce.

Notation	Meaning
$\mathcal{X}^*$, $\mathcal{Y}^*$	Inputs/instance space of the problem, outputs/outcomes space
$\mathcal{R}^* : \mathcal{X}^* \to \mathcal{Y}^*$	Real relationship returning a value $y^* \in \mathcal{Y}^*$ for each input $x^* \in \mathcal{X}^*$
$\mathcal{R} : \mathcal{X} \to \mathcal{Y}$	Simplified relationship between simplified spaces $\mathcal{X}$ and $\mathcal{Y}$
$\hat{\mathcal{R}} : \hat{\mathcal{X}} \longrightarrow \hat{\mathcal{Y}}$	Measurable relationship between perturbed input and output
$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$	Loss function
$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$	Sample/data-set of dimension N of the space $\mathcal{X} \times \mathcal{Y}$
$h = (\theta, \hat{h}) \in H$	Set of hypotheses: θ are the trainable parameters (weights and biases for ANNs), $\hat{h}$ hyperparameters (prior knowledge choices such as type of architectures, learning rate, etc.)
$\mathcal{M} := \mathcal{M}_h = \mathcal{M}_{\theta,\hat{h}} : \mathcal{X} \to \mathcal{Y}$	Approximating Model
θ^*	Optimal trainable parameters (true expected loss minimizer)
$\mathcal{M}_{\theta^*,\hat{h}}$	Optimal Bayesian Predictor
$\hat{\theta}$	Optimal trainable parameters on a finite data-set $\mathcal{D}$
$\mathcal{M}_{\hat{\theta},\hat{h}}$	Empirical Model
ε_A	Aleatoric Uncertainty
$\varepsilon_{A,I}$	Aleatoric Inherent Uncertainty
$\varepsilon_{A,N}$	Aleatoric Experimental Uncertainty/Noise
$\varepsilon_{A,M}$	Aleatoric Model Uncertainty
ε_E	Epistemic Uncertainty
$\varepsilon_{E,M}$	Epistemic Model Uncertainty
ε_{E,A_p}	Epistemic Approximation Uncertainty
$v_i = f_i\left(W_i \cdot v_{i-1} + b_i\right)$	ith layer in a feedforward ANN
L	Number of layers
W_i, b_i	Weights/biases of the ith layer
$f_i(\cdot)$	Activation function of the ith layer
$p(\theta)$	Prior knowledge on trainable parameters
$p(\mathcal{D}\|\theta)$	Likelihood of the dataset given θ
$p(\mathcal{D})$	Evidence of the available data-set
$p(\theta\|\mathcal{D})$	Posterior distribution of trainable parameters
$p(y\|\mathbf{x},\theta) = p(y\|\mathbf{x},\mathcal{D})$	Predictive posterior distribution
$\mu_*(\mathbf{x})$	Mean of the output for regression task
$\sigma_*^2(\mathbf{x})$	Variance of the output for regression task
S	Number of sub-models/evaluations/forward passes

CRediT authorship contribution statement

Giulio Del Corso: Writing – review & editing, Writing – original draft, Investigation, Formal analysis, Conceptualization. **Sara Colantonio:** Writing – review & editing, Funding acquisition, Conceptualization. **Claudia Caudai:** Writing – review & editing, Writing – original draft, Supervision, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was partially funded by the European Union's Horizon 2020 Research and Innovation Program under Grant Agreement No 952159 (ProCAncer-I) and the European Union's Horizon Europe Research and Innovation Program under Grant Agreement No 101135932 (FAITH).

Appendix A. Negative log-likelihood and maximum likelihood estimation

The assumptions about the task determine the choice of loss ℓ. Classification problems require a loss that penalizes choosing the wrong class (e.g., binary cross-entropy), whereas regression problems need to value choices that are close to the ground truth (e.g., MSE). However, there are many cases where the choice of a loss straightforward, such as in multi-head networks corresponding to a distributed output, see Section 5.1. The Bayesian framework allows most known losses to be reinterpreted as the Negative Log-Likelihood (NLL) of the problem of interest.

Formally, defining the data-set D as realizations of N i.i.d. random variables $\{(X_i, Y_i)\}_{i=1}^N$ from a population Y with density/mass function f, **Likelihood function** is defined as:

$$\mathcal{L}\left(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N\right) := f\left(\{(\mathbf{x}_i, y_i)\}_{i=1}^N|\theta\right) \underbrace{=}_{i.i.d.} \Pi_{i=1}^N f((\mathbf{x}_i, y_i)|\theta) \tag{A.1}$$

Intuitively, it describes the plausibility that these values come from the distribution fully characterized by θ. Using the i.i.d. hypothesis, the likelihood can be calculated for each sample $(\mathbf{x}, y)$ and the combined likelihood for the whole data set D. Determining the optimal model means choosing θ values and therefore the Maximum Likelihood Estimator (MLE) is defined as:

$$\hat{\theta}_{MLE} := \text{argmax}\, \mathcal{L}\left(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N\right) \tag{A.2}$$

In optimization problems, it is preferable to minimize the sums of elements rather than maximizing their products, and therefore we can apply a logarithm to the likelihood (log-likelihood). The other advantage of using log-likelihood over likelihood is that it avoids problems of numerical precision (i.e., products of small numbers can lead to arithmetic underflow). Furthermore, logarithms transform products in summation, making it easier to calculate the derivative (both for theoretical calculation and for backpropagation). Therefore, by applying a logarithmic transformation to the original Likelihood, we get:

$$\begin{aligned} NLL(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N) &:= -\log \mathcal{L}(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N) = -\sum_{i=1}^N \log f((\mathbf{x}_i, y_i)|\theta) \\ \hat{\theta}_{MLE} &:= \text{argmax}\, \mathcal{L}\left(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N\right) = \text{argmin}\, NLL(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N) \end{aligned} \tag{A.3}$$

The first-order condition for minimizing the NLL is $\frac{\partial NLL(\theta|\{(\mathbf{x}_i, y_i)\}_{i=1}^N)}{\partial \theta} = 0$, which can lead to an analytical solution in some simplified statistical cases. Most of the cost functions commonly used in Machine Learning can be derived as Negative Log-Likelihood under appropriate assumptions. The introduction of regularization functions, which act as a prior in the Bayesian setting, leads to a different (less prone to over-fitting) maximization called Maximum A Posteriori estimate (MAP).

Remark 24 (*How to Compute the NLL Explicitly?*)**.** By assuming that the output of a neural network is a given probability distribution, $f(\cdot)$ can be written explicitly as the density/mass function of that probability distribution. This allows most standard (and non-standard) losses to be derived as a special case of NLL. Examples include: single-head Gaussian with fixed (homoscedastic) variance for MSE, two-head Gaussian with variable (heteroscedastic) variance for variance attenuation, k categorical distribution for cross-entropy, etc.

For regression tasks, the output can be modeled with a continuous distribution (e.g. Gaussian or more complex/skewed) to derive MSE or variance attenuation losses:

Example 2 (*One Head Regression - MSE*)**.** Mean Squared Error is a loss that can be derived from MLE under a few assumptions. Assume that there is a relationship between an input vector $\mathbf{x}$ and an outcome y, that can be described by a mean plus a variance term $\varepsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ (homoscedastic aleatoric uncertainty): $y = \mathbb{E}[y|\mathbf{x}] + \varepsilon \approx \mu_\theta(\mathbf{x}) + \varepsilon$. To derive the MLE approximation of the optimal θ, we can note that adding $\mu_\theta(\mathbf{x})$ to a normal distribution leads to: $y \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}), \sigma^2\right)$. Therefore, the likelihood of a couple $(\mathbf{x}, y)$ can be computed from the density of a Gaussian distribution:

$$\mathcal{L}(\theta|(\mathbf{x}, y)) = \frac{1}{\left(2\pi\sigma^2\right)^{\frac{1}{2}}} \exp\left(-\frac{\left(y - \mu_\theta(\mathbf{x})\right)^2}{2\sigma^2}\right) \tag{A.4}$$

and the NLL is:

$$\text{NLL}(\theta|(\mathbf{x}, y)) = \frac{1}{2}\log(2\pi) + \frac{1}{2}\log\sigma^2 + \frac{(y - \mu_\theta(\mathbf{x}))^2}{2\sigma^2} \tag{A.5}$$

by dropping the constant terms $\frac{1}{2}\log(2\pi)$ and $\frac{1}{2}\log(\sigma^2)$ and multiplying by the constant $2\sigma^2$, we get:

$$\text{argmin}\, \text{NLL}(\theta|(\mathbf{x}, y)) = \text{argmin}\left(y - \mu_\theta(\mathbf{x})\right)^2 \tag{A.6}$$

by computing this value on every $(\mathbf{x}_i, y_i) \in D$ and taking the average, we obtain the Mean Squared Error loss.

Example 3 (*Multi Head Regression - Variance Attenuation*)**.** Assume that we have an ANN $\mathcal{M}_\theta$ with two heads (see Section 5.1) corresponding to mean μ_θ and variance σ_θ^2 of a Gaussian output. In order to derive the optimal θ we need to define a cost function (i.e., the NLL). The likelihood for a Gaussian output is:

$$\mathcal{L}(\theta|(\mathbf{x}, y)) = \frac{1}{\left(2\pi\sigma_\theta^2(\mathbf{x})\right)^{\frac{1}{2}}} \exp\left(-\frac{\left(y - \mu_\theta(\mathbf{x})\right)^2}{2\sigma_\theta^2(\mathbf{x})}\right) \tag{A.7}$$

therefore the NLL is:

$$\text{NLL}(\theta|(\mathbf{x}, y)) = \frac{1}{2}\log(2\pi) + \frac{1}{2}\log\sigma_\theta^2(\mathbf{x}) + \frac{(y - \mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} \tag{A.8}$$

by dropping the constant term $\frac{1}{2}\log(2\pi)$, this is the **Variance Attenuation loss** introduced in Section 5.1

Classification tasks, on the other hand, assume that the output of the model can be written as a categorical distribution to derive classical cross-entropy or classification variance attenuation losses.

Example 4 (*One Head Classification - Cross Entropy*)**.** Standard cross entropy can be derived as the negative Log-Likelihood of the softmax layer of an ANN classifier ($\text{softmax}(z) = e^z / \sum_{c^*=1}^{\#\mathcal{K}} e^{z_c^*}$, $z = \mathcal{M}_\theta(\mathbf{x})$, with z the logit layer of the raw outputs of the network). Indeed, by assuming that the outputs of the softmax layer represent the parameters $(p_1, \dots, p_{\#\mathcal{K}})$ of a categorical distribution (i.e., $f(y = \text{one\_hot\_encoded}(i)) = p_i$), the likelihood is:

$$\mathcal{L}(\theta|(\mathbf{x}, y)) = f((y, \mathbf{x})|\theta) = \text{softmax}(\mathcal{M}_\theta(\mathbf{x}))_c \tag{A.9}$$

where $y = (y_{c^*})_{c^*=1}^{\#\mathcal{K}}$ is the one-hot encoded vector of the observed class, with $y_c = 1$ on the correct class index c and 0 otherwise (i.e, only the predicted probability of the given class c).

$$NLL(\theta|(\mathbf{x}, y)) = -\log \text{softmax}\left(\mathcal{M}_\theta(\mathbf{x})\right)_c = -\sum_{c^*=1}^{\#\mathcal{K}} y_c^* \log \frac{e^{\mathcal{M}_\theta(\mathbf{x})_c^*}}{\sum_{c'=1}^{\#\mathcal{K}} e^{\mathcal{M}_\theta(\mathbf{x})_{c'}}} \tag{A.10}$$

where the last term is the **Cross entropy loss**.

Example 5 (*Multi Head Classification - Variance Attenuation*)**.** Multi-head classification is not straightforward as the softmax layer of the categorical distribution parameters cannot be directly substituted with a Gaussian vector. One possible approach to circumvent this limitation is to substitute the last raw (logit) layer z of the network with a Gaussian distribution (i.e., $z = \mathcal{M}_\theta(\mathbf{x}) = \mathcal{N}\left(\mu_\theta(\mathbf{x}), \sigma_\theta^2(\mathbf{x})\right)$ The Negative Log Likelihood for a couple $(\mathbf{x}, y)$, with y the one hot encoded vector of the correct class c is:

$$\begin{aligned} NLL(\theta|(\mathbf{x}, y)) = -\log \text{softmax}\left(\mathcal{N}(\mu_\theta(\mathbf{x}), \sigma_\theta^2(\mathbf{x}))\right)_c &\approx -\frac{1}{T}\sum_{t=1}^{T}\sum_{c^*=1}^{\#\mathcal{K}} y_c^* \log \frac{e^{z_{c^*}^{(t)}}}{\sum_{c'=1}^{\#\mathcal{K}} e^{z_{c'}^{(t)}}} \\ &\text{where } z^{(t)} \sim \mathcal{N}\left(\mu_\theta(\mathbf{x}), \sigma_\theta^2(\mathbf{x})\right) \end{aligned} \tag{A.11}$$

where the Monte Carlo approximation (with T samples $z^{(t)}$ sampled from the logit Gaussian distribution) is necessary due to the lack of an analytical formulation of the density of a softmax function applied to a Gaussian vector. This is called the (**classification**) **variance attenuation loss**.

Appendix B. Classic Gaussian process regression

Gaussian Process (GP) **Regression** (also known as Kriging or Wiener Kolmogorov prediction) is a classic metamodel technique that interpolates the data using a Gaussian process to provide a Bayesian estimate of the posterior probability even with extremely small data-sets [106]. The mathematical theory behind GPs dates back to 1940, with the works of Kolmogorov [104] and Wiener [105], and it is arguably one of the first effective Bayesian metamodeling techniques.

A GP $\{Z(\mathbf{x})|\mathbf{x} \in \mathcal{X}\}$ is a stochastic process such that, given any finite number of random variables $(Z(\mathbf{x}_1), \dots, Z(\mathbf{x}_N))$, it has a joint Gaussian probability distribution. $Z(\mathbf{x})$ is uniquely identified by its mean and covariance functions [12], defined as:

$$\begin{cases} m(\mathbf{x}) = \mathbb{E}[Z(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') = \text{cov}(Z(\mathbf{x}), Z(\mathbf{x}')) = \mathbb{E}\left[(Z(x) - m(\mathbf{x}))(Z(\mathbf{x}') - m(\mathbf{x}'))\right] \end{cases} \tag{B.1}$$

and therefore can be denoted as: $Z(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \sim \mathcal{GP}(m, k)$.

Remark 25 (*How to Choose Mean and Covariance Functions?*)**.** The mean and covariance functions are not uniquely determined. Each family of functions has its own properties that strongly modify the underlying GP. The ability of the metamodel to approximate the original model can be strongly influenced by the choice of these functions. The choice of these two functions corresponds to an a priori hypothesis on the GP process.

The **mean function** $m(\mathbf{x})$ of the GP $Z(\mathbf{x})$ can lead to different model predictions. It is common to consider GPs with a zero mean function. It should be noted that the posterior distribution obtained by conditioning $Z(\mathbf{x})$ on the training data set can have a non-zero mean even with the zero mean assumption on the prior distribution. Therefore, the obtained posterior mean is not constrained to be zero regardless of the choice of $m(\mathbf{x})$ [143]. Common choices for $m(\mathbf{x})$ include: stationarity of the first moment over the domain (i.e., $\mathbb{E}[Z(\mathbf{x})] = constant$, *simple Kriging*), assuming constant unknowns over the search neighborhood of $\mathbf{x}$ (*ordinary Kriging*), or assume a general polynomial trend model (*universal Kriging*).

The **covariance function** is a positive definite kernel (i.e., a general symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $\sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j\ k(\mathbf{x}_i, \mathbf{x}_j) > 0$ holds for any $\mathbf{x}_1, \dots, \mathbf{x}_m \in \mathcal{X}$, given $m \in \mathbb{N}$ and $c_1, \dots, c_m \in \mathbb{R}$). For a GP $Z(\mathbf{x})$, a covariance function $k(\mathbf{x}, \mathbf{x}')$ gives the covariance of the values of the random field at the two locations $\mathbf{x}$ and $\mathbf{x}'$.

Remark 26 (*Stationary Kernel*)**.** The covariance functions used in many practical applications are stationary. A kernel is said to be stationary if it is a function of the distance $x - x'$, i.e. $k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x} - \mathbf{x}')$ (i.e., it is invariant with respect to translation in the input space).

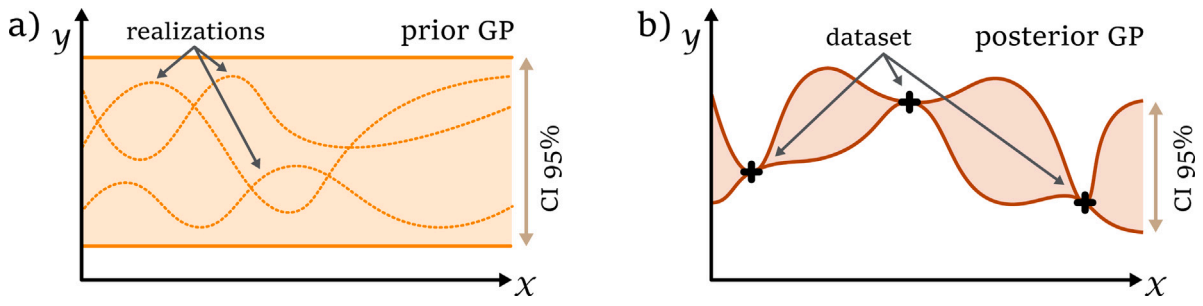

Fig. B.12. (Panel a): a priori Gaussian Process, no data have been provided and therefore the GP is consistent with the prior hypothesis. (Panel b): Training data points are provided and therefore the posterior GP provides a prediction according to the new information.

Commonly used covariance functions include: *Constant* covariance ($k(\mathbf{x},\mathbf{x}') = \text{constant}$), *Linear* covariance ($k(\mathbf{x},\mathbf{x}') = \mathbf{x}^T \cdot \mathbf{x}'$), and *Squared Exponential* (SE) covariance ($k(\mathbf{x},\mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2\varphi^2}\|\mathbf{x}-\mathbf{x}'\|^2\right)$ parameterized by φ (correlation length) and the variance parameter σ^2). The latter is a common stationary choice in Kriging [144], with $m(\mathbf{x})$ that is intuitively the trend around which the realizations vary, φ the oscillation frequencies, and σ^2 the range of variations.

Once the prior GP $Z(\mathbf{x})$ with its own mean function $m(x)$ and covariance function $k(\mathbf{x},\mathbf{x}')$ and given a training data-set $(\mathcal{X},\mathcal{Y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ have been defined, **regression** consists of describing the predictive posterior probability of an unknown output y^* for an input $\mathbf{x}^*$.

Remark 27 (*GP as Surrogate Model*)**.** In a Machine Learning fashion, GP can be thought of as an original prior GP $Z(\mathbf{x})$ conditioned on a data-set D to define a posterior GP fully characterized by its mean and variance functions. These functions can be computed using a training strategy, such as the maximum likelihood method [145,146], gradient-based approaches [147], or with cross-validation strategy [148].

For the properties of the GP, it holds:

$$\begin{bmatrix}\mathcal{Y}\\ y^*\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}m(\mathcal{X})\\ m(\mathbf{x}^*)\end{bmatrix}, \begin{bmatrix}k(\mathcal{X},\mathcal{X}) & k(\mathcal{X},\mathbf{x}^*)\\ k(\mathbf{x}^*,\mathcal{X}) & k(\mathbf{x}^*,\mathbf{x}^*)\end{bmatrix}\right) \tag{B.2}$$

therefore, the predictive posterior random variable can be calculated as follows:

$$\begin{aligned}(y^* \mid \mathbf{x}^*, \mathcal{X}\times\mathcal{Y}) &\sim \mathcal{N}(\tilde{m}(\mathbf{x}^*), \tilde{k}(\mathbf{x}^*,\mathbf{x}^*))\\ \tilde{m}(\mathbf{x}^*) &:= m(\mathbf{x}^*) + k(\mathbf{x}^*,\mathcal{X})k(\mathcal{X},\mathcal{X})^{-1}(\mathcal{Y} - m(\mathcal{X}))\\ \tilde{k}(\mathbf{x}^*,\mathbf{x}^*) &:= k(\mathbf{x}^*,\mathbf{x}^*) - k(\mathbf{x}^*,\mathcal{X})k(\mathcal{X},\mathcal{X})^{-1}k(\mathcal{X},\mathbf{x}^*)\end{aligned} \tag{B.3}$$

Therefore, given a new input value $\mathbf{x}^*$, an estimate of the output can be obtained from the prior $Z(\mathbf{x})$ in terms of: *predicted mean* $\tilde{m}(\mathbf{x}^*)$ (a weighted combination of the training data) and *predicted variance* $\tilde{k}(\mathbf{x}^*,\mathbf{x}^*)$ (the posterior variability in the prediction).

Remark 28 (*Why Is the Mean a Good Point Estimate?*)**.** In most applications, the result must be a point estimate, obtained as the argmin of a given loss/risk function $\ell(y_{true}, y^*)$. In the absence of y_{true}, the point estimate is obtained by minimizing the average expected loss $y^*_{optimal}|\mathbf{x}^* = \text{argmin}_{y^*}\int \mathcal{L}(y', y^*)p(y'|\mathbf{x}^*, D)dy'$. In general, the risk-minimizing value is the median. However, for a Gaussian predictive posterior distribution, the mean and median coincide. Therefore, for any symmetric loss function and symmetric predictive distribution, we always get y^* as the mean of the predictive distribution [143].

The variance $\tilde{k}(\mathbf{x}^*,\mathbf{x}^*)$ of the conditional distribution above deals with its uncertainty and is made up of two parts: $k(\mathbf{x}^*,\mathbf{x}^*)$ can be read as the variance due to the test data, while the second term represents the variance induced by the training data [72]. The total variance is the difference between the two terms as if the information in the training data reduces the original variance inherent in the test data (see Fig. B.12).

In conclusion, GP regression is one of the best-known classical Machine Learning approaches to define a surrogate model capable of producing a prediction coupled with a probabilistic reliability score (i.e., the predictive variance $\tilde{k}(\mathbf{x}^*,\mathbf{x}^*)$, which is the model's mean-square error) that can be used both for regression and for global sensitivity analysis [106]. The given self-assessment of model performance can be used to define a sub-set of optimal points $\{x_i, y_i\}$ to enrich the original data set D (Sequential Design). This can be done by selecting new points as those with the highest predicted variance or by implementing more efficient strategies [149–151]. These sequential designs are often used in Deep Learning settings to efficiently explore the hyperparameter space (Bayesian optimization) [152,153]. The main drawbacks of this method are that the entire data set must be stored in order to generate the predictive posterior, and that GP cannot handle problems with a very high dimensionality of the input space. However, these difficulties are partially addressed by a deep extension of GP called Deep Gaussian Processes [72].

Appendix C. Training a Bayesian model

As described in Section 4, training a model (i.e., model induction) in the Bayesian framework can be seen as determining the posterior distribution of the trainable parameters given the data-set D (i.e., $\hat{\theta} = \theta|D$) or the full population $\mathcal{X}\times\mathcal{Y}$ (i.e., $\theta^* = \theta|\mathcal{X}\times\mathcal{Y}$). This applies directly to several of the methods discussed in this review, most notably the intrusive ones: BNNs, DGPs, and BNNs with distributional output.

Formally, model induction is merely sampling from the posterior distribution (followed, eventually, by model averaging to produce a point estimate). However, the posterior is a high-dimensional and highly non-convex conditional distribution [91]. Therefore, most of the classical approaches (such as inverse transform sampling, acceptance-rejection method, rejection sampling, etc.) fail to sample from it. Two of the most commonly used methods are Markov Chain Monte Carlo (MCMC) sampling [92] and Variational Inference (VI) [53].

C.1. Markov Chain Monte Carlo

MCMC approaches are methods that can approximate true complex conditional distributions by constructing a sequence of random samples (i.e., a Markov Chain such that each sample depends on the state of the previous sample) that converges to the desired distribution [92]. MCMC has several drawbacks, including: an initial burn-in time before convergence, autocorrelated samples after a certain number of iterations (hence the need to subsample a much larger set of samples to obtain uncorrelated values), and required hypothesis on the target distribution.

Among the various MCMC alternatives, the one best suited to the Bayesian framework is the Metropolis–Hastings algorithm which simply requires a function $f(x)$ proportional to the target posterior [154] (i.e., when computing the posterior distribution, it can be chosen as a combination of the terms in the numerator of Eq. (10)). Formally, the Metropolis–Hastings algorithm starts with a guess θ_0 on the possible distribution. The subsequent candidate θ_{i+1} is obtained by sampling a proposal distribution $Q\left(\theta'|\theta_i\right)$. If θ' is more likely, is retained (e.g., $\theta_{i+1} = \theta'$), otherwise is dropped with an acceptance probability $p = \min\left(1, \frac{Q(\theta'|\theta_i)}{Q(\theta_i|\theta')} \cdot \frac{f(\theta')}{f(\theta_i)}\right)$.

Remark 29 (*How to Choose MCMC Hyperparameters?*)**.** MCMC parameters have to be properly chosen. If the spread is too large too many samples are rejected while if it is too small the samples are highly autocorrelated. Similarly, the number of epochs must be defined to avoid burn-in time and the size of the full sample should be chosen to avoid excessively autocorrelated values. Therefore, the state-of-the-art MCMC algorithms for Bayesian statistics is the No-U-Turn Sampler (NUTS) [155], a Hamiltonian Monte Carlo (characterized by low rejection rate and short burn-in time) [156] combined with an automatic hyperparameter tuning strategy.

C.2. Variational inference

To reduce the computational burden associated with the use of MCMC strategies, an alternative approach is **Variational Inference** (VI) [53]. Variational Inference aims to define an easy (analytical and parameterized by ϕ, usually multivariate Gaussian on the weights of the model) distribution $q_\phi(\theta|D)$ which approximates the true posterior one: $q_\phi = q_\phi(\theta|D) \approx p(\theta|D)$

Remark 30 (*Going Beyond Gaussians*)**.** While the multivariate Gaussian distribution is a typical choice for approximating the posterior, a broader family of distributions with particular conjugacy properties can be used to tailor the posterior to the given task [157]. More advanced techniques overcome the limitation of choosing a given distribution by providing a sequence of increasingly complex approximations to capture multiple modes of the posterior, such as nonparametric VI [158] or normalizing flow VI [159].

Given the data set D and an a priori hypothesis about the trainable parameters θ, VI aims to minimize the difference between the real posterior $p(\theta|D)$ and approximate q_ϕ distributions, thus obtaining a simplified model $\mathcal{M}_{q_\phi} \approx \mathcal{M}_{\theta|D}$. To define a similarity between two distributions (with density), we need a suitable measure, like the **Kullback–Leibler Divergence** d_{KL} (also called Relative Entropy). If $d_{KL} \to 0$, the two distributions are close and, therefore, we can use q_ϕ as an approximation of $p(\theta|D)$. Given $q_\phi(\theta|D)$ and $p(\theta|D)$, the KL divergence is defined as:

$$
\begin{aligned}
d_{\mathrm{KL}}(q_\phi(\theta|D) \;\|\; p(\theta|D)) &:= \int_{\mathbb{R}^M} \log\left(\frac{q_\phi(\theta|D)(z)}{p(\theta|D)(z)}\right) \cdot q_\phi(\theta|D)(z) \cdot dz \\
&= \mathbb{E}_{q_\phi(\theta|D)}\left[\log\left(\frac{q_\phi(\theta|D)}{p(\theta|D)}\right)\right] \\
&= \mathbb{E}_{q_\phi(\theta|D)}\left[\log q_\phi(\theta|D) - \log p(\theta|D)\right] \\
&= \mathbb{E}_{q_\phi(\theta|D)}\left[\log q_\phi(\theta|D) - \log\left(p(\theta, D)/p(D)\right)\right] \\
&= \mathbb{E}_{q_\phi(\theta|D)}\left[\log q_\phi(\theta|D) - \log p(\theta, D) + \log p(D)\right] \\
&= \underbrace{\mathbb{E}_{q_\phi}\left[\log q_\phi(\theta|D)\right] - \underbrace{\mathbb{E}_{q_\phi}\left[\log p(\theta, D)\right]}_{\mathbb{E}_{q_\phi}\left[\log p(D|\theta)\cdot p(\theta)\right]}}_{-\mathrm{ELBO}} + \underbrace{\mathbb{E}_{q_\phi}\left[\log p(D)\right]}_{\log p(D)\cdot\mathbb{E}_{q_\phi}[1]}_{intractable}
\end{aligned}
\tag{C.1}
$$

where M is the number of trainable parameters. Indeed, by reordering the terms:

$$\begin{aligned}
&\underbrace{\log p(D)}_{evidence} = -\mathbb{E}_{q_\phi}\left[\log q_\phi(\theta|D)\right] + \mathbb{E}_{q_\phi}\left[\log p(D|\theta)\cdot p(\theta)\right] + \underbrace{d_{KL}(q_\phi(\theta|D) \;\|\; p(\theta|D))}_{\geq 0} \\
&\log p(D) \geq \underbrace{-\mathbb{E}_{q_\phi}\left[\log q_\phi(\theta|D)\right] + \mathbb{E}_{q_\phi}\left[\log p(\theta)\right]}_{} + \mathbb{E}_{q_\phi}\left[\log p(D|\theta)\right] \\
&\qquad = \underbrace{\mathbb{E}_{q_\phi}\underbrace{\left[\log p(D|\theta)\right]}_{Log\,Likelihood}}_{Reconstruction\;\;Error} - \underbrace{\mathbb{E}_{q_\phi}\left[\log \frac{q_\phi(\theta|D)}{p(\theta)}\right]}_{d_{\mathrm{KL}}(q_\phi(\theta|D)\;\|\;p(\theta))} := \mathrm{ELBO}\left(q_\phi(\theta|D) \;\|\; p(\theta|D)\right)
\end{aligned} \tag{C.2}$$

The ELBO (Evidence Lower BOund) is a lower bound of the evidence $\log p(D)$, and is given by the tractable **Reconstruction Error Term** ($\mathbb{E}_{q_\phi}\left[\log p(D|\theta)\right]$), which represents the Negative Log-Likelihood loss (Appendix A) minus the tractable KL divergence of the approximate posterior distribution Q_ϕ and the prior distribution θ [160].

Remark 31 (*ANN with Distributional Output*)**.** In Section 5.1 we introduced ANNs with distributional outputs. The loss discussed in that section is the Reconstruction Error Term. Obviously, the ANNs can be trained by adding also the KL divergence term (which remains constant by assuming fully deterministic weights and biases).

Remark 32 (*ELBO vs. KL*)**.** Following the previous equations, it holds:

$$d_{\mathrm{KL}}(q_\phi(\theta|D) \;\|\; p(\theta|D)) = \underbrace{\log p(D)}_{intractable\;\;(constant)} - \mathrm{ELBO}\left(q_\phi(\theta|D) \;\|\; p(\theta|D)\right) \tag{C.3}$$

Therefore, maximizing the ELBO is equivalent to minimizing the KL divergence. However, optimization problems are usually set up as minimization tasks rather than maximization ones, so the objective function in VI is taken to be the negative ELBO, also called the Variational Free Energy.

Variational Inference translates the computation of the marginal likelihood into the optimization of its lower bound. The natural choice for optimizing the lower bound with Neural Networks is the Stochastic Gradient Descent algorithm (on mini-batches). The adaptation of the Stochastic Gradient Descent algorithm to Variational Inference is called the **Stochastic Variational Inference** (SVI) algorithm. This allows scaling to large/computationally demanding data sets as the ELBO can be computed on a single mini-batch at each iteration. Concerning optimization using Neural Networks, for every mini batch $\mathcal{B}$ a Gradient Descent Algorithm aims to update the trainable parameters θ. An issue arises for SVI, because back propagation does not work on random sampling and therefore Gradient Descent Algorithm cannot be used directly. A smart solution for this problem has been proposed by Kingma et al. under the name of Reparametrization trick [161].

Remark 33 (*Reparametrization Trick*)**.** To use gradient base optimization, the reparametrization assumes that the trainable values z follow $q_\phi \sim \mathcal{N}(\mu, \sigma^2)$ (i.e., a multivariate normal distribution). Instead of sampling directly from it (and thus stopping the gradient flow through the model), the values z can be reparametrized as $\mu + \sigma \cdot \mathcal{N}(0,1)$ by assuming Gaussian noise over each trainable parameter. The need to use the reparametrization trick and the need to easily compute the reconstruction error leads to the choice of simplified q_ϕ, usually taken from the exponential family (such as multivariate normal, gamma, Dirichlet, etc.).

In summary, given a model with variational parameters (e.g., weights, biases, activation functions, etc.), a choice for the objective function is the Variational Free Energy (i.e., - ELBO). The reconstruction error term is computed in a supervised manner using the available data set, while Kullback–Leibler divergence is computed cumulatively on **each variational layer**. In particular, for each layer, d_{KL} is computed between q_ϕ (usually chosen as a Gaussian distribution) and the prior hypothesis on θ (typically $\theta \sim \mathcal{N}(0,1)$). Therefore, in the Gaussian case is computed again as a proper combination of the Variance Attenuation Loss (see Appendix A).

Remark 34 (*What About the $\mathbb{E}_{q_\phi}$ Term?*)**.** Formally, ELBO must be computed as the mean with respect to the q_ϕ distribution. However, approximating this average can drastically increase the computational training effort, as it requires multiple samples from the approximate distribution q_ϕ, recalculating the terms of the ELBO each time. Therefore, the practice is to keep only one sampled value and average over the minibatch to smooth the solution. It should be noted that this can lead to noisy solutions and therefore the optimal (but costly) way to compute ELBO would be to resample from the variational distribution.

C.3. Laplace approximation

A post hoc approach to evaluate the posterior distribution is the Laplace approximation [79]. This method relies on a local approximation of the posterior distribution around the "maximum a posterior" (MAP) estimator. This is similar to Variational Inference (i.e., is an optimization approach), but VI is a costly global approximation procedure (during training), while the Laplace approximation is a non-intrusive a posteriori local MAP estimate that can be applied to a pre-trained ANN. As for VI, the Laplace Approximation relies on strong assumptions about the form of the posterior distribution (usually assumed to be Gaussian). This, along

with the local nature of the approximation, implies that the Laplace approximation poorly captures the multimodal distribution and induces low accuracy UQ.

The Laplace approximation aims to find a Gaussian approximation of a complex probability density defined over a set of continuous variables [162]. In the DL framework, this translates to: finding a Gaussian approximation of the posterior $p(\theta|D)$ given the weights of the network.

The Negative Log-Likelihood loss is then estimated using a second-order Taylor expansion around the MAP:

$$\mathcal{L}(\theta; D) \approx \underbrace{\mathcal{L}(\theta_{MAP}; D)}_{0} + \frac{1}{2}(\theta - \theta_{MAP})^T \mathbf{H} (\theta - \theta_{MAP}) \tag{C.4}$$

where the first term is zero by definition and $\mathbf{H}$ is the Hessian matrix (i.e., the second derivative of the ANN likelihood function regarding the model weights $\mathbf{H} := \nabla^2_\theta \mathcal{L}(\theta|D)|_{\theta_{MAP}}$). By assuming a multivariate Gaussian posterior distribution (of the weights), this leads to:

$$p(\theta|D) \approx \mathcal{N}(\theta_{MAP}, \mathbf{H}^{-1}) := p(\theta_{MAP}|D) \cdot e^{-\frac{1}{2}(\theta-\theta_{MAP})^T \mathbf{H} (\theta-\theta_{MAP})} \tag{C.5}$$

where θ_{MAP} are the weights of the pre-trained ANN.

Remark 35 (*How to Compute* $\mathbf{H}^{-1}$*?*)**.** Computing $\mathbf{H}^{-1}$ requires storing n^2 partial derivatives (where n is the number of trainable weights) and then inverting the matrix, which is computationally infeasible for standard ANNs. Common methods to speed up computations include: (1) Reduce $\mathbf{H}$ to a diagonal matrix (i.e., assuming independent weights) (2) Reduce to a L block diagonal matrix (one block for each layer, assuming independent layers) (3) Select a subset of weights to update (to Gaussian distribution) and leave all other weights fixed at their MAP estimate [163]. However, all of these methods can lead to an overestimation of the uncertainty (variance) of the predictive posterior distribution.

This approximation can be used to obtain the predictive posterior distribution for a given input $\mathbf{x}^*$ by applying an MC approach to the simplified posterior $\mathcal{N}(\theta_{MAP}, \mathbf{H}^{-1})$ and then averaging the models with the sampled weights $\{\theta^{(s)}\}_{s=1}^{S}$.

References

[1] S.C. Hora, Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management, Reliab. Eng. Syst. Saf. 54 (2–3) (1996) 217–223.
[2] A. Der Kiureghian, O. Ditlevsen, Aleatory or epistemic? Does it matter? Struct. Saf. 31 (2) (2009) 105–112.
[3] T.J. Sullivan, Introduction to Uncertainty Quantification, Vol. 63, Springer, 2015.
[4] Q. Rao, J. Frtunikj, Deep learning for self-driving cars: Chances and challenges, in: Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems, 2018, pp. 35–38.
[5] E. Pachetti, G. Del Corso, S. Bardelli, S. Colantonio, Few-shot conditional learning: Automatic and reliable device classification for medical test equipment, J. Imaging 10 (7) (2024) 167.
[6] F. Yang, H.-z. Wang, H. Mi, C.-d. Lin, W.-w. Cai, Using random forest for reliable classification and cost-sensitive learning for medical diagnosis, BMC Bioinformatics 10 (2009) 1–14.
[7] A. Lambrou, H. Papadopoulos, A. Gammerman, Reliable confidence measures for medical diagnosis with evolutionary algorithms, IEEE Trans. Inf. Technol. Biomed. 15 (1) (2010) 93–99.
[8] G. Del Corso, D. Germanese, C. Caudai, G. Anastasi, P. Belli, A. Formica, A. Nicolucci, S. Palma, M.A. Pascali, S. Pieroni, et al., Adaptive machine learning approach for importance evaluation of multimodal breast cancer radiomic features, J. Imaging Inform. Med. (2024) 1–10.
[9] L.V. Jospin, H. Laga, F. Boussaid, W. Buntine, M. Bennamoun, Hands-on Bayesian neural networks—A tutorial for deep learning users, IEEE Comput. Intell. Mag. 17 (2) (2022) 29–48.
[10] M. Magris, A. Iosifidis, Bayesian learning for neural networks: an algorithmic survey, Artif. Intell. Rev. 56 (10) (2023) 11773–11823.
[11] W. He, Z. Jiang, A Comprehensive Survey on Uncertainty Quantification for Deep Learning, 2023, URL https://api.semanticscholar.org/CorpusID:257219242.
[12] E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods, Mach. Learn. 110 (2019) 457–506, URL https://api.semanticscholar.org/CorpusID:216465307.
[13] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U.R. Acharya, et al., A review of uncertainty quantification in deep learning: Techniques, applications and challenges, Inf. Fusion 76 (2021) 243–297.
[14] T. Bayes, An essay towards solving a problem in the doctrine of chances. 1763., M. D. Comput. : Comput. Med. Pr. 8 3 (1763) 157–171, URL https://api.semanticscholar.org/CorpusID:263568682.
[15] B. de Finetti, Sul significato soggettivo della probabilità, Fund. Math. 17 (1931) 298–329, URL https://api.semanticscholar.org/CorpusID:117887848.
[16] G.J. Klir, B.J.C. Yuan, Fuzzy Sets and Fuzzy Logic, 1995, URL https://api.semanticscholar.org/CorpusID:115906608.
[17] J.T. Bradley, R. Seising, The gap between scientific theory and application: Black and Zadeh - Vagueness and fuzzy sets, in: NAFIPS 2006 - 2006 Annual Meeting of the North American Fuzzy Information Processing Society, 2006, pp. 408–413, URL https://api.semanticscholar.org/CorpusID:17713995.
[18] J.M. Booker, T.J. Ross, An evolution of uncertainty assessment and quantification, Sci. Iran. 18 (2011) 669–676, URL https://api.semanticscholar.org/CorpusID:122228714.
[19] B. Russell, Vagueness, Australas. J. Philos. 1 (2) (1923) 84–92, http://dx.doi.org/10.1080/00048402308540623.
[20] L. Wittgenstein, Tractatus logico-philosophicus, Nord. Wittgenstein Rev. (1914) URL https://api.semanticscholar.org/CorpusID:170285990.
[21] G. Frege, Grundgesetze der Arithmetik, 1903, URL https://api.semanticscholar.org/CorpusID:190946830.
[22] B. Russell, The Philosophy of Logical Atomism, 1918, URL https://api.semanticscholar.org/CorpusID:170808846.
[23] M. Black, Vagueness. An exercise in logical analysis, Philos. Sci. 4 (1937) 427–455, URL https://api.semanticscholar.org/CorpusID:120376519.
[24] M. Black, Reasoning with loose concepts, Dialogue 2 (1963) 1–12, URL https://api.semanticscholar.org/CorpusID:170424742.
[25] M. Cohen, Action theory, Philos. Books 18 (3) (1977) 115–117, http://dx.doi.org/10.1111/j.1468-0149.1977.tb01727.x, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0149.1977.tb01727.x, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0149.1977.tb01727.x.
[26] D. Lewis, Counterfactuals and comparative possibility, J. Philos. Logic 2 (1973) 418–446, URL https://api.semanticscholar.org/CorpusID:122802088.
[27] J. Lécaillon, G.L.S. Shackle, J. Lécaillon, Decision Order and Time in Human Affairs, 1962, URL https://api.semanticscholar.org/CorpusID:143586227.

[28] L.A. Zadeh, Fuzzy Sets, 1965, URL https://api.semanticscholar.org/CorpusID:205883170.
[29] L.A. Zadeh, A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges, 1972, URL https://api.semanticscholar.org/CorpusID:120547398.
[30] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets and Systems 100 (1978) 9–34, URL https://api.semanticscholar.org/CorpusID:13928072.
[31] D. Dubois, H. Prade, Operations in a fuzzy-valued logic, Inf. Control. 43 (1979) 224–240, URL https://api.semanticscholar.org/CorpusID:5734042.
[32] D. Dubois, H. Prade, Possibility theory, Scholarpedia 2 (1988) 2074, URL https://api.semanticscholar.org/CorpusID:10448969.
[33] P. Agarwal, D.H.S. Nayal, Possibility theory versus probability theory in fuzzy measure theory, 2015, URL https://api.semanticscholar.org/CorpusID:6269180.
[34] L.A. Zadeh, Discussion: Probability Theory and Fuzzy Logic are Complementary Rather Than Competitive, 1995, URL https://api.semanticscholar.org/CorpusID:120593999.
[35] A.P. Dempster, Upper and lower probabilities induced by a multivalued mapping, in: Classic Works of the Dempster-Shafer Theory of Belief Functions, 1967, URL https://api.semanticscholar.org/CorpusID:1305116.
[36] G. Shafer, A mathematical theory of evidence, A Math. Theory Evid. (1976) URL https://api.semanticscholar.org/CorpusID:27862354.
[37] A.P. Dempster, A generalization of Bayesian inference, in: Classic Works of the Dempster-Shafer Theory of Belief Functions, 1968, URL https://api.semanticscholar.org/CorpusID:44440896.
[38] B. Jalaeian, S. Russell, Uncertain context: Uncertainty quantification in machine learning, AI Mag. 40 (2019) 40–49, URL https://api.semanticscholar.org/CorpusID:213331028.
[39] H.H. Clark, T.B. Carlson, I I ' Context for Comprehension, 2012, URL https://api.semanticscholar.org/CorpusID:201863058.
[40] G.J. Klir, M.J. Wierman, J. Kacprzyk, Uncertainty-Based Information: Elements of Generalized Information Theory (Studies in Fuzziness and Soft Computing), 1998, URL https://api.semanticscholar.org/CorpusID:67394188.
[41] M. Ma, M. Friedman, A. Kandel, A new fuzzy arithmetic, Fuzzy Sets and Systems 108 (1999) 83–90, URL https://api.semanticscholar.org/CorpusID:42467917.
[42] S. Ferson, L. Ginzburg, Hybrid arithmetic, in: Proceedings of 3rd International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society, 1995, pp. 619–623, URL https://api.semanticscholar.org/CorpusID:124809291.
[43] A. Kaufmann, M.M. Gupta, Introduction to Fuzzy Arithmetic : Theory and Applications, 1986, URL https://api.semanticscholar.org/CorpusID:120239309.
[44] L.A. Zadeh, Fuzzy logic, neural networks, and soft computing, Commun. ACM 37 (3) (1994) 77–84, http://dx.doi.org/10.1145/175247.175255.
[45] D.D. Nauck, F. Klawonn, R. Kruse, Foundations Of Neuro-Fuzzy Systems, 1997, URL https://api.semanticscholar.org/CorpusID:56820012.
[46] M.M. Gupta, E. Sanchez, Fuzzy information and decision processes, IFAC Proc. Vol. 15 (1981) 409–411, URL https://api.semanticscholar.org/CorpusID:118619232.
[47] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan kaufmann, 1988.
[48] R.J. Carroll, D. Ruppert, L.A. Stefanski, C.M. Crainiceanu, Measurement Error in Nonlinear Models: a Modern Perspective, Chapman and Hall/CRC, 2006.
[49] E. Saccenti, M.H. Hendriks, A.K. Smilde, Corruption of the pearson correlation coefficient by measurement error and its estimation, bias, and correction under different error models, Sci. Rep. 10 (1) (2020) 1–19.
[50] A. Kendall, Y. Gal, What uncertainties do we need in bayesian deep learning for computer vision? Adv. Neural Inf. Process. Syst. 30 (2017).
[51] N.S. Detlefsen, M. Jørgensen, S. Hauberg, Reliable training and estimation of variance networks, 2019, arXiv preprint arXiv:1906.03260.
[52] D. Hendrycks, K. Gimpel, A baseline for detecting misclassified and out-of-distribution examples in neural networks, 2016, arXiv preprint arXiv:1610.02136.
[53] D. Barber, C. Bishop, Variational learning in Bayesian neural networks, Neural Netw. Mach. Learn. (1998) 215–237.
[54] A. Damianou, N.D. Lawrence, Deep gaussian processes, in: Artificial Intelligence and Statistics, PMLR, 2013, pp. 207–215.
[55] N.D. Lawrence, A.J. Moore, Hierarchical Gaussian process latent variable models, in: Proceedings of the 24th International Conference on Machine Learning, 2007, pp. 481–488.
[56] A.G. Wilson, Z. Hu, R. Salakhutdinov, E.P. Xing, Deep kernel learning, in: Artificial Intelligence and Statistics, PMLR, 2016, pp. 370–378.
[57] D. Van, Ensemble Methods: Foundations and Algorithms, 2012, URL https://api.semanticscholar.org/CorpusID:15096071.
[58] A. Malinin, B. Mlodozeniec, M. Gales, Ensemble distribution distillation, 2019, arXiv preprint arXiv:1905.00076.
[59] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv preprint arXiv:1207.0580.
[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[61] V. Kuleshov, P.S. Liang, Calibrated structured prediction, Adv. Neural Inf. Process. Syst. 28 (2015).
[62] H. Jiang, B. Kim, M. Guan, M. Gupta, To trust or not to trust a classifier, Adv. Neural Inf. Process. Syst. 31 (2018).
[63] L. Fei-Fei, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. 28 (4) (2006) 594.
[64] N.G. Polson, V. Sokolov, Deep learning: A Bayesian perspective, 2017.
[65] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, S. Udluft, Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 1184–1193.
[66] M. Valdenegro-Toro, D.S. Mori, A deeper look into aleatoric and epistemic uncertainty disentanglement, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPRW, IEEE, 2022, pp. 1508–1516.
[67] R. Senge, S. Bösner, K. Dembczyński, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, E. Hüllermeier, Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty, Inform. Sci. 255 (2014) 16–29.
[68] I.P. de Jong, A.I. Sburlea, M. Valdenegro-Toro, How disentangled are your classification uncertainties?, 2024, arXiv preprint arXiv:2408.12175.
[69] G. Del Corso, F. Volpini, C. Caudai, D. Moroni, S. Colantonio, NADA: A synthetic shape benchmark for testing probabilistic deep learning models, 2024, http://dx.doi.org/10.5281/zenodo.14361220.
[70] B. Mucsányi, M. Kirchhof, S.J. Oh, Benchmarking uncertainty disentanglement: Specialized uncertainties for specialized tasks, 2024, arXiv preprint arXiv:2402.19460.
[71] D.A. Nix, A.S. Weigend, Estimating the mean and variance of the target probability distribution, in: Proceedings of 1994 Ieee International Conference on Neural Networks, ICNN'94, Vol. 1, IEEE, 1994, pp. 55–60.
[72] K. Jakkala, Deep Gaussian processes: A survey, 2021, arXiv abs/2106.12135. URL https://api.semanticscholar.org/CorpusID:235606388.
[73] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Adv. Neural Inf. Process. Syst. 30 (2017).
[74] M. Seitzer, A. Tavakoli, D. Antic, G. Martius, On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks, 2022, arXiv preprint arXiv:2203.09168.
[75] A. Stirn, D.A. Knowles, Variational variance: Simple, reliable, calibrated heteroscedastic noise variance parameterization, 2020, arXiv preprint arXiv:2006.04910.
[76] Tishby, Solla, Consistent inference of probabilities in layered networks: predictions and generalizations, in: International 1989 Joint Conference on Neural Networks, IEEE, 1989, pp. 403–409.
[77] J. Denker, Y. LeCun, Transforming neural-net output levels to probability distributions, Adv. Neural Inf. Process. Syst. 3 (1990).
[78] S. Geman, et al., Buntine, w., and weigend, a.(1991).“Bayesian back-propagation”. complex systems, 5, 603-643., 1991.

[79] D.J. MacKay, A practical Bayesian framework for backpropagation networks, Neural Comput. 4 (3) (1992) 448–472.
[80] L.M. Mescheder, S. Nowozin, A. Geiger, Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks, 2017, arXiv abs/1701.04722. URL https://api.semanticscholar.org/CorpusID:605416.
[81] H. Shi, X.-M. Zhang, S. Sun, L. Liu, L. Tang, A survey on Bayesian graph neural networks, in: 2021 13th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2021, pp. 158–161, URL https://api.semanticscholar.org/CorpusID:238478843.
[82] M. Fortunato, C. Blundell, O. Vinyals, Bayesian recurrent neural networks, 2017, arXiv abs/1704.02798. URL https://api.semanticscholar.org/CorpusID:4026237.
[83] C. Chen, X. Lin, G. Terejanu, An approximate Bayesian long short- term memory algorithm for outlier detection, in: 2018 24th International Conference on Pattern Recognition, ICPR, 2017, pp. 201–206, URL https://api.semanticscholar.org/CorpusID:2255761.
[84] F. Burden, D. Winkler, Bayesian regularization of neural networks, Artif. Neural Netw.: Methods Appl. (2009) 23–42.
[85] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with Bernoulli approximate variational inference, 2015, arXiv preprint arXiv:1506.02158.
[86] J. Mitros, B. Mac Namee, On the validity of Bayesian neural networks for uncertainty estimation, 2019, arXiv preprint arXiv:1912.01530.
[87] A. Kristiadi, M. Hein, P. Hennig, Being bayesian, even just a bit, fixes overconfidence in relu networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 5436–5446.
[88] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, J. Snoek, Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, Adv. Neural Inf. Process. Syst. 32 (2019).
[89] W.L. Buntine, Operations for learning with graphical models, J. Artificial Intelligence Res. 2 (1994) 159–225.
[90] Y. Wen, P. Vicol, J. Ba, D. Tran, R. Grosse, Flipout: Efficient pseudo-independent weight perturbations on mini-batches, 2018, arXiv preprint arXiv:1803.04386.
[91] P. Izmailov, S. Vikram, M.D. Hoffman, A.G.G. Wilson, What are Bayesian neural network posteriors really like? in: International Conference on Machine Learning, PMLR, 2021, pp. 4629–4640.
[92] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, 1970.
[93] R. Neal, Bayesian learning via stochastic dynamics, Adv. Neural Inf. Process. Syst. 5 (1992).
[94] G.E. Hinton, D. Van Camp, Keeping the neural networks simple by minimizing the description length of the weights, in: Proceedings of the Sixth Annual Conference on Computational Learning Theory, 1993, pp. 5–13.
[95] N.P. Lemoine, Moving beyond noninformative priors: why and how to choose weakly informative priors in Bayesian analyses, Oikos 128 (7) (2019) 912–928.
[96] A.A. Pourzanjani, R.M. Jiang, B. Mitchell, P.J. Atzberger, L.R. Petzold, Bayesian inference over the stiefel manifold via the Givens representation, Bayesian Anal. 16 (2) (2021) 639–666.
[97] D. Silvestro, T. Andermann, Prior choice affects ability of Bayesian neural networks to identify unknowns, 2020, arXiv preprint arXiv:2005.04987.
[98] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural network, in: International Conference on Machine Learning, PMLR, 2015, pp. 1613–1622.
[99] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, 2012.
[100] G. Del Corso, Uncertainty analysis of biological systems: towards a digital twin of the human heart, 2022.
[101] R.E. Caflisch, Monte carlo and quasi-monte carlo methods, Acta Numer. 7 (1998) 1–49.
[102] M.J. Gilman, A brief survey of stopping rules in Monte Carlo simulations, 1968.
[103] A. Mobiny, P. Yuan, S.K. Moulik, N. Garg, C.C. Wu, H. Van Nguyen, Dropconnect is effective in modeling uncertainty of bayesian deep networks, Sci. Rep. 11 (1) (2021) 5458.
[104] A. Kolmogoroff, Interpolation und extrapolation von stationaren zufalligen folgen, Izv. Ross. Akad. Nauk. Seriya Mat. 5 (1) (1941) 3–14.
[105] N. Wiener, N. Wiener, C. Mathematician, N. Wiener, N. Wiener, C. Mathématicien, Extrapolation, Interpolation, and Smoothing of Stationary Time Series: with Engineering Applications, Vol. 113, MIT press Cambridge, MA, 1949.
[106] L. Le Gratiet, S. Marelli, B. Sudret, Metamodel-based sensitivity analysis: polynomial chaos expansions and Gaussian processes, in: Handbook of Uncertainty Quantification, Springer, 2017, pp. 1289–1325.
[107] R.M. Neal, Bayesian Learning for Neural Networks, vol. 118, Springer Science & Business Media, 2012.
[108] A. Damianou, Deep Gaussian Processes and Variational Propagation of Uncertainty (Ph.D. thesis), University of Sheffield, 2015.
[109] Z. Dai, A. Damianou, J. González, N. Lawrence, Variational auto-encoded deep Gaussian processes, 2015, arXiv preprint arXiv:1511.06455.
[110] H. Salimbeni, M. Deisenroth, Doubly stochastic variational inference for deep Gaussian processes, Adv. Neural Inf. Process. Syst. 30 (2017).
[111] A. Wilson, H. Nickisch, Kernel interpolation for scalable structured Gaussian processes (KISS-GP), in: International Conference on Machine Learning, PMLR, 2015, pp. 1775–1784.
[112] J. Quinonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. 6 (2005) 1939–1959.
[113] S. Wager, S. Wang, P.S. Liang, Dropout training as adaptive regularization, Adv. Neural Inf. Process. Syst. 26 (2013).
[114] P. Baldi, P.J. Sadowski, Understanding dropout, Adv. Neural Inf. Process. Syst. 26 (2013).
[115] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1050–1059.
[116] P. McClure, N. Kriegeskorte, Representing inferential uncertainty in deep neural networks through sampling, 2016.
[117] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012).
[118] A. Pauls, J. Yoder, Determining optimum drop-out rate for neural networks, in: Midwest Instructional Computing Symposium, MICS, 2018.
[119] J.M. Hernández-Lobato, R. Adams, Probabilistic backpropagation for scalable learning of bayesian neural networks, in: International Conference on Machine Learning, PMLR, 2015, pp. 1861–1869.
[120] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, T. Vercauteren, Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks, Neurocomputing 338 (2019) 34–45.
[121] H. Liu, R. Ji, J. Li, B. Zhang, Y. Gao, Y. Wu, F. Huang, Universal adversarial perturbation via prior driven uncertainty approximation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2941–2949.
[122] A. Chan, A. Alaa, Z. Qian, M. Van Der Schaar, Unlabelled data improves bayesian uncertainty calibration under covariate shift, in: International Conference on Machine Learning, PMLR, 2020, pp. 1392–1402.
[123] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1321–1330.
[124] J. Platt, et al., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, Adv. Large Margin Classif. 10 (3) (1999) 61–74.
[125] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 694–699.
[126] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, in: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 625–632.
[127] F. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms 1998, in: Proceedings of the 15th International Conference on Machine Learning ICML-98 Morgan Kaufmann, San Mateo, CA.

[128] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.
[129] A. Nguyen, J. Yosinski, J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.
[130] S.B. Kotsiantis, Bagging and boosting variants for handling classifications problems: a survey, Knowl. Eng. Rev. 29 (2013) 78–100, URL https://api.semanticscholar.org/CorpusID:27301684.
[131] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32, URL https://api.semanticscholar.org/CorpusID:89141.
[132] T. Gneiting, A.E. Raftery, Strictly proper scoring rules, prediction, and estimation, J. Amer. Statist. Assoc. 102 (477) (2007) 359–378.
[133] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Neural Information Processing Systems, 2016, URL https://api.semanticscholar.org/CorpusID:6294674.
[134] R.E. Schapire, Explaining AdaBoost, in: Empirical Inference, 2013, URL https://api.semanticscholar.org/CorpusID:7122892.
[135] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, URL https://api.semanticscholar.org/CorpusID:4650265.
[136] F. Wenzel, J. Snoek, D. Tran, R. Jenatton, Hyperparameter ensembles for robustness and uncertainty quantification, 2020, arXiv abs/2006.13570. URL https://api.semanticscholar.org/CorpusID:220042034.
[137] D. Horng, Chau, G. Tech, M. Roozbahani, J. Heer, J.T. Stasko, C. Faloutsos, Ensemble methods, Mach. Learn. with Spark™ Python® (2019) URL https://api.semanticscholar.org/CorpusID:60385584.
[138] S. Singh, A. Yassine, R. Benlamri, Internet of energy: Ensemble learning through multilevel stacking for load forecasting, in: 2020 IEEE Intl Conf Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/ PiCom/ CBDCom/ CyberSciTech), 2020, pp. 658–664, URL https://api.semanticscholar.org/CorpusID:226845927.
[139] R. McAllister, G. Kahn, J. Clune, S. Levine, Robustness to out-of-distribution inputs via task-aware generative uncertainty, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 2083–2089.
[140] G. Shafer, V. Vovk, A tutorial on conformal prediction., J. Mach. Learn. Res. 9 (3) (2008).
[141] F. Volpini, C. Caudai, G. Del Corso, S. Colantonio, NA database: Generator of probabilistic synthetic geometrical shape dataset.
[142] G. Del Corso, Shedding light on ucertainties: an open source python implementation of UQ techniques, 2025, https://github.com/GDelCorso/ShedLight_UQ.
[143] C.E. Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[144] M.L. Stein, Interpolation of Spatial Data: Some Theory for Kriging, Springer Science & Business Media, 1999.
[145] D.A. Harville, Maximum likelihood approaches to variance component estimation and to related problems, J. Amer. Statist. Assoc. 72 (358) (1977) 320–338.
[146] T.J. Santner, B.J. Williams, W.I. Notz, B.J. Williams, The Design and Analysis of Computer Experiments, Vol. 1, Springer, 2003.
[147] C.E. Rasmussen, C.K. Williams, Model selection and adaptation of hyperparameters, in: Gaussian Processes for Machine Learning, MIT Press, 2005, pp. 105–128.
[148] F. Bachoc, Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification, Comput. Statist. Data Anal. 66 (2013) 55–69.
[149] R. Bates, R. Buck, E. Riccomagno, H. Wynn, Experimental design and observation for large systems, J. R. Stat. Soc. Ser. B Stat. Methodol. 58 (1) (1996) 77–94.
[150] W.C. Van Beers, J.P. Kleijnen, Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping, European J. Oper. Res. 186 (3) (2008) 1099–1113.
[151] L. Le Gratiet, C. Cannamela, Cokriging-based sequential design strategies using fast cross-validation techniques for multi-fidelity computer codes, Technometrics 57 (3) (2015) 418–427.
[152] B. Betrò, Bayesian methods in global optimization, J. Global Optim. 1 (1) (1991) 1–14.
[153] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, N. De Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE 104 (1) (2015) 148–175.
[154] S. Chib, E. Greenberg, Understanding the metropolis-hastings algorithm, Amer. Statist. 49 (4) (1995) 327–335.
[155] M.D. Hoffman, A. Gelman, et al., The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo., J. Mach. Learn. Res. 15 (1) (2014) 1593–1623.
[156] R.M. Neal, MCMC using Hamiltonian dynamics, 2012, arXiv preprint arXiv:1206.1901.
[157] C. Zhang, J. Bütepage, H. Kjellström, S. Mandt, Advances in variational inference, IEEE Trans. Pattern Anal. Mach. Intell. 41 (8) (2018) 2008–2026.
[158] S. Gershman, M. Hoffman, D. Blei, Nonparametric variational inference, 2012, arXiv preprint arXiv:1206.4665.
[159] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: International Conference on Machine Learning, PMLR, 2015, pp. 1530–1538.
[160] T. Salimans, D.P. Kingma, M. Welling, Markov Chain Monte Carlo and Variational Inference: Bridging the Gap, 2014, URL https://api.semanticscholar.org/CorpusID:216078910.
[161] D.P. Kingma, M. Welling, Auto-encoding variational bayes, 2013, arXiv preprint arXiv:1312.6114.
[162] C.M. Bishop, N.M. Nasrabadi, Pattern Recognition and Machine Learning, vol. 4, Springer, 2006.
[163] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, P. Hennig, Laplace redux-effortless bayesian deep learning, Adv. Neural Inf. Process. Syst. 34 (2021) 20089–20103.