A Survey of Uncertainty in Deep Neural Networks

Jakob Gawlikowski (Student Member, IEEE), Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee,
Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung (Member, IEEE), Ribana Roscher
(Member, IEEE), Muhammad Shahzad (Member IEEE), Wen Yang (Senior Member, IEEE), Richard Bamler
(Fellow, IEEE), Xiao Xiang Zhu (Fellow, IEEE)

Abstract—Over the last decade, neural networks have reached almost every field of science and became a crucial part of various real world applications. Due to the increasing spread, confidence in neural network predictions became more and more important. However, basic neural networks do not deliver certainty estimates or suffer from over or under confidence, i.e. are badly calibrated. To overcome this, many researchers have been working on understanding and quantifying uncertainty in a neural network's prediction. As a result, different types and sources of uncertainty have been identified and a variety of approaches to measure and quantify uncertainty in neural networks have been proposed. This work gives a comprehensive overview of uncertainty estimation in neural networks, reviews recent advances in the field, highlights current challenges, and identifies potential research opportunities. It is intended to give anyone interested in uncertainty estimation in neural networks a broad overview and introduction, without presupposing prior knowledge in this field. For that, a comprehensive introduction to the most crucial sources of uncertainty is given and their separation into reducible model uncertainty and not reducible data uncertainty is presented. The modeling of these uncertainties based on deterministic neural networks, Bayesian neural networks, ensemble of neural networks, and test-time data augmentation approaches is introduced and different branches of these fields as well as the latest developments are discussed. For a practical application, we discuss different measures of uncertainty, approaches for the calibration of neural networks and give an overview of existing baselines and available implementations. Different examples from the wide spectrum of challenges in the fields of medical image analysis, robotic and earth observation give an idea of the needs and challenges regarding uncertainties in practical applications of neural networks. Additionally, the practical limitations of uncertainty quantification methods in neural networks for mission- and safety-critical real world applications are discussed and an outlook on the next steps towards a broader usage of such methods is given.

Index Terms—Bayesian deep neural networks, Ensembles, Test-time augmentation, Calibration, Uncertainty

I. INTRODUCTION

Within the last decade enormous advances on deep neural networks (DNNs) have been realized, encouraging their adaptation in a variety of research fields, where complex systems have to be modeled or understood, as for example earth observation, medical image analysis or robotics. Although DNNs have become attractive in high risk fields such as medical image analysis [1], [2], [3], [4], [5], [6] or autonomous vehicle control [7], [8], [9], [10], their deployment in mission-

and safety-critical real world applications remains limited. The main factors responsible for this limitation are

- the lack of expressiveness and transparency of a deep neural network's inference model, which makes it difficult to trust their outcomes [2],

- the inability to distinguish between in-domain and out-of-domain samples [11], [12] and the sensitivity to domain shifts [13],

- the inability to provide reliable uncertainty estimates for a deep neural network's decision [14] and frequently occurring overconfident predictions [15], [16],

- the sensitivity to adversarial attacks that make deep neural networks vulnerable for sabotage [17], [18], [19].

These factors are mainly based on an uncertainty already included in the data (data uncertainty) or a lack of knowledge of the neural network (model uncertainty). To overcome these limitations, it is essential to provide uncertainty estimates, such that uncertain predictions can be ignored or passed to human experts [20]. Providing uncertainty estimates is not only important for a safe decision-making in high-risks fields, but also crucial in fields where the data sources are highly inhomogeneous and labeled data is rare, such as in remote sensing [21], [22]. Also for fields where uncertainties form a crucial part of the learning techniques, such as for active learning [23], [24], [25], [26] or reinforcement learning [20], [27], [28], [29], uncertainty estimates are highly important.

In recent years, researchers have shown an increased interest in estimating uncertainty in DNNs [30], [20], [31], [32], [33], [34], [35], [36]. The most common way to estimate the uncertainty on a prediction (the predictive uncertainty) is based on separately modelling the uncertainty caused by the model (epistemic or model uncertainty) and the uncertainty caused by the data (aleatoric or data uncertainty). While the first one is reducible by improving the model which is learned by the DNN, the latter one is not reducible. The most important approaches for modeling this separation are Bayesian inference [30], [20], [37], [9], [38], ensemble approaches [31], [39], [40], test time data augmentation approaches [41], [42], or single deterministic networks containing explicit components to represent the model and the data uncertainty [43], [44], [45], [32], [46]. Estimating the predictive uncertainty is not sufficient for safe decision-making. Furthermore, it is crucial to assure that the uncertainty estimates are reliable. To this end, the calibration property (the degree of reliability) of DNNs has been investigated and re-calibration methods have been

proposed [15], [47], [48] to obtain reliable (well-calibrated) uncertainty estimates.

There are several works that give an introduction and overview of uncertainty in statistical modelling. Ghanem et al. [49] published a handbook about uncertainty quantification, which includes a detailed and broad description of different concepts of uncertainty quantification, but without explicitly focusing on the application of neural networks. The theses of Gal [50] and Kendall [51] contain a good overview about Bayesian neural networks, especially with focus on the Monte Carlo (MC) Dropout approach and its application in computer vision tasks. The thesis of Malinin [52] also contains a very good introduction and additional insights into Prior networks. Wang et al. contributed two surveys on Bayesian deep learning [53], [54]. They introduced a general framework and the conceptual description of the Bayesian neural networks (BNNs), followed by an updated presentation of Bayesian approaches for uncertainty quantification in neural networks with special focus on recommender systems, topic models, and control. In [55] an evaluation of uncertainty quantification in deep learning is given by presenting and comparing the uncertainty quantification based on the softmax output, the ensemble of networks, Bayesian neural networks, and autoencoders on the MNIST data set. Regarding the practicability of uncertainty quantification approaches for real life mission- and safety-critical applications, Gustafsson et al. [56] introduced a framework to test the robustness required in real-world computer vision applications and delivered a comparison of two popular approaches, namely MC Dropout and Ensemble methods. Hüllermeier et al. [57] presented the concepts of aleatoric and epistemic uncertainty in neural networks and discussed different concepts to model and quantify them. In contrast to this, Abdar et al. [58] presented an overview on uncertainty quantification methodologies in neural networks and provide an extensive list of references for different application fields and a discussion of open challenges.

In this work, we present an extensive overview over all concepts that have to be taken into account when working with uncertainty in neural networks while keeping the applicability on real world applications in mind. Our goal is to provide the reader with a clear thread from the sources of uncertainty to applications, where uncertainty estimations are needed. Furthermore, we point out the limitations of current approaches and discuss further challenges to be tackled in the future. For that, we provide a broad introduction and comparison of different approaches and fundamental concepts. The survey is mainly designed for people already familiar with deep learning concepts and who are planning to incorporate uncertainty estimation into their predictions. But also for people already familiar with the topic, this review provides a useful overview of the whole concept of uncertainty in neural networks and their applications in different fields.

In summary, we comprehensively discuss

- Sources and types of uncertainty (Section II),

- Recent studies and approaches for estimating uncertainty in DNNs (Section III),

- Uncertainty measures and methods for assessing the quality and impact of uncertainty estimates (Section IV),

- Recent studies and approaches for calibrating DNNs (Section V),

- An overview over frequently used evaluation data sets, available benchmarks and implementations[1] (Section VI),

- An overview over real-world applications using uncertainty estimates (Section VII),

- A discussion on current challenges and further directions of research in the future (Section VIII).

In general, the principles and methods for estimating uncertainty and calibrating DNNs can be applied to all regression, classification, and segmentation problems, if not stated differently. In order to get a deeper dive into explicit applications of the methods, we refer to the section on applications and to further readings in the referenced literature.

II. UNCERTAINTY IN DEEP NEURAL NETWORKS

A neural network is a non-linear function f_θ parameterized by model parameters θ (i.e. the network weights) that maps from a measurable input set $\mathbb{X}$ to a measurable output set $\mathbb{Y}$, i.e.

$$f_\theta : \mathbb{X} \to \mathbb{Y} \qquad f_\theta(x) = y \ . \tag{1}$$

For a supervised setting, we further have a finite set of training data $\mathcal{D} \subseteq \mathbb{D} = \mathbb{X} \times \mathbb{Y}$ containing N data samples and corresponding targets, i.e.

$$\mathcal{D} = (\mathcal{X}, \mathcal{Y}) = \{x_n, y_n\}_{n=1}^N \subseteq \mathbb{D} \ . \tag{2}$$

For a new data sample $x^* \in \mathbb{X}$, a neural network trained on $\mathcal{D}$ can be used to predict a corresponding target $f_\theta(x^*) = y^*$. We consider four different steps from the raw information in the environment to a prediction by a neural network with quantified uncertainties, namely

1) the *data acquisition* process:
   The occurrence of some information in the environment (e.g. a bird's singing) and a measured observation of this information (e.g. an audio record).

2) the *DNN building* process:
   The design and training of a neural network.

3) the *applied inference* model: The model which is applied for inference (e.g. a Bayesian neural network or an ensemble of neural networks).

4) the *prediction's uncertainty* model: The modelling of the uncertainties caused by the neural network and by the data.

In practice, these four steps contain several potential sources of uncertainty and errors, which again affect the final prediction of a neural network. The five factors that we think are the most vital for the cause of uncertainty in a DNN's predictions are

---

[1]The list of available implementations can be found in Section VI as well as within an additional GitHub repository under github.com/JakobCode/UncertaintyInNeuralNetworks_Resources.

- the variability in real world situations,
- the errors inherent to the measurement systems,
- the errors in the architecture specification of the DNN,
- the errors in the training procedure of the DNN,
- the errors caused by unknown data.

In the following, the four steps leading from raw information to uncertainty quantification on a DNN's prediction are described in more detail. Within this, we highlight the sources of uncertainty that are related to the single steps and explain how the uncertainties are propagated through the process. Finally, we introduce a model for the uncertainty on a neural network's prediction and introduce the main types of uncertainty considered in neural networks.

The goal of this section is to give an accountable idea of the uncertainties in neural networks. Hence, for the sake of simplicity we only describe and discuss the mathematical properties, which are relevant for understanding the approaches and applying the methodology in different fields.

A. Data Acquisition

In the context of supervised learning, the data acquisition describes the process where measurements x and target variables y are generated in order to represent a (real world) situation ω from some space Ω. In the real world, a realization of ω could for example be a bird, x a picture of this bird, and y a label stating *'bird'*. During the measurement, random noise can occur and information may get lost. We model this randomness in x by

$$x|\omega \sim p_{x|\omega} \ . \tag{3}$$

Equivalently, the corresponding target variable y is derived, where the description is either based on another measurement or is the result of a labeling process[2]. For both cases the description can be affected by noise and errors and we state it as

$$y|\omega \sim p_{y|\omega} \ . \tag{4}$$

A neural network is trained on a finite data set of realizations of $x|\omega_i$ and $y|\omega_i$ based on N real world situations $\omega_1, ..., \omega_N$,

$$\mathcal{D} = \{x_i, y_i\}_{i=1}^N \ . \tag{5}$$

When collecting the training data, two factors can cause uncertainty in a neural network trained on this data. First, the sample space Ω should be sufficiently covered by the training data $x_1, ..., x_N$ for $\omega_1, ..., \omega_N$. For that, one has to take into account that for a new sample x^* it in general holds that $x^* \neq x_i$ for all training situations x_i. Following, the target has to be estimated based on the trained neural network model, which directly leads to the first factor of uncertainty:

---

[2]In many cases one can model the labeling process as a mapping from $\mathbb{X}$ to $\mathbb{Y}$, e.g. for speech recognition or various computer vision tasks. For other tasks, as for example earth observation, this is not always the case. Data is often labeled based on high resolutional data while low resolutional data is utilized for the prediction task.

> **Factor I: Variability in Real World Situations**
> Most real world environments are highly variable and almost constantly affected by changes. These changes affect parameters as for example temperature, illumination, clutter, and physical objects' size and shape. Changes in the environment can also affect the expression of objects, as for example plants after rain look very different to plants after a drought. When real world situations change compared to the training set, this is called a distribution shift. Neural networks are sensitive to distribution shifts, which can lead to significant changes in the performance of a neural network.

The second case is based on the measurement system, which has a direct effect on the correlation between the samples and the corresponding targets. The measurement system generates information x_i and y_i that describe ω_i but might not contain enough information to learn a direct mapping from x_i to y_i. This means that there might be highly different real world information ω_i and ω_j (e.g. city and forest) resulting in very similar corresponding measurements x_i and x_j (e.g. temperature) or similar corresponding targets y_i and y_j (e.g. label noise that labels both samples as forest). This directly leads to our second factor of uncertainty:

> **Factor II: Error and Noise in Measurement Systems**
> The measurements themselves can be a source of uncertainty on the neural network's prediction. This can be caused by limited information in the measurements, as for example the image resolution, or by not measuring false or insufficiently available information modalities. Moreover, it can be caused by noise, for example sensor noise, by motion, or mechanical stress leading to imprecise measures. Furthermore, false labeling is also a source of uncertainty that can be seen as error and noise in the measurement system. It is referenced as label noise and affects the model by reducing the confidence on the true class prediction during training.

B. Deep Neural Network Design and Training

The design of a DNN covers the explicit modeling of the neural network and its stochastic training process. The assumptions on the problem structure induced by the design and training of the neural network are called inductive bias [59]. We summarize all decisions of the modeler on the network's structure (e.g. the number of parameters, the layers, the activation functions, etc.) and training process (e.g. optimization algorithm, regularization, augmentation, etc.) in a structure configuration s. The defined network structure gives the third factor of uncertainty in a neural network's predictions:

> **Factor III: Errors in the Model Structure**
> The structure of a neural network has a direct effect on its performance and therefore also on the uncertainty of its

prediction. For instance, the number of parameters affects the memorization capacity, which can lead to under- or over-fitting on the training data. Regarding uncertainty in neural networks, it is known that deeper networks tend to be overconfident in their softmax output, meaning that they predict too much probability on the class with highest probability score [15].

For a given network structure s and a training data set $\mathcal{D}$, the training of a neural network is a stochastic process and therefore the resulting neural network f_θ is based on a random variable,

$$\theta | D, s \sim p_{\theta|D,s}. \tag{6}$$

The process is stochastic due to random decisions as the order of the data, random initialization or random regularization as augmentation or dropout. The loss landscape of a neural network is highly non-linear and the randomness in the training process in general leads to different local optima θ^* resulting in different models [31]. Also, parameters as batch size, learning rate, and the number of training epochs affect the training and result in different models. Depending on the underlying task these models can significantly differ in their predictions for single samples, even leading to a difference in the overall model performance. This sensitivity to the training process directly leads to the fourth factor for uncertainties in neural network predictions:

Factor IV: Errors in the Training Procedure
The training process of a neural network includes many parameters that have to be defined (batch size, optimizer, learning rate, stopping criteria, regularization, etc.) and also stochastic decisions within the training process (batch generation and weight initialization) take place. All these decisions affect the local optima and it is therefore very unlikely that two training processes deliver the same model parameterization. A training data set that suffers from imbalance or low coverage of single regions in the data distribution also introduces uncertainties on the network's learned parameters, as already described in the data acquisition. This might be softened by applying augmentation to increase the variety or by balancing the impact of single classes or regions on the loss function.

Since the training process is based on the given training data set $\mathcal{D}$, errors in the data acquisition process (e.g. label noise) can result in errors in the training process.

C. Inference

The inference describes the prediction of an output y^* for a new data sample x^* by the neural network. At this time, the network is trained for a specific task. Thus, samples which are not inputs for this task cause errors and are therefore also a source of uncertainty:

Factor V: Errors Caused by Unknown Data
Especially in classification tasks, a neural network that is trained on samples derived from a world $\mathcal{W}_1$ can also be capable of processing samples derived from a completely different world $\mathcal{W}_2$. This is for example the case, when a network trained on images of cats and dogs receives a sample showing a bird. Here, the source of uncertainty does not lie in the data acquisition process, since we assume a world to contain only feasible inputs for a prediction task. Even though the practical result might be equal to too much noise on a sensor or complete failure of a sensor, the data considered here represents a valid sample, but for a different task or domain.

D. Predictive Uncertainty Model

As a modeller, one is mainly interested in the uncertainty that is propagated onto a prediction y^*, the so-called *predictive uncertainty*. Within the data acquisition model, the probability distribution for a prediction y^* based on some sample x^* is given by

$$p(y^*|x^*) = \int_\Omega p(y^*|\omega)p(\omega|x^*)d\omega \tag{7}$$

and a maximum a posteriori (MAP) estimation is given by

$$y^* = \arg\max_y p(y|x^*) \ . \tag{8}$$

Since the modeling is based on the unavailable latent variable ω, one takes an approximative representation based on a sampled training data set $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ containing N samples and corresponding targets. The distribution and MAP estimator in (7) and (8) for a new sample x^* are then predicted based on the known examples by

$$p(y^*|x^*) = \int_D p(y^*|, x^*) \tag{9}$$

and

$$y^* = \arg\max_y p(y|\mathcal{D}, x^*) \ . \tag{10}$$

In general, the distribution given in (9) is unknown and can only be estimated based on the given data in D. For this estimation, neural networks form a very powerful tool for many tasks and applications.

The prediction of a neural network is subject to both model-dependent and input data-dependent errors, and therefore the predictive uncertainty associated with y^* is in general separated into *data uncertainty* (also statistical or aleatoric uncertainty [57]) and *model uncertainty* (also systemic or epistemic uncertainty [57]). Depending on the underlying approach, an additional explicit modeling of *distributional uncertainty* [32] is used to model the uncertainty, which is caused by examples from a region not covered by the training data.

1) Model- and Data Uncertainty: The model uncertainty covers the uncertainty that is caused by shortcomings in the model, either by errors in the training procedure, an insufficient model structure, or lack of knowledge due to unknown samples or a bad coverage of the training data set.

In contrast to this, data uncertainty is related to uncertainty that directly stems from the data. Data uncertainty is caused by information loss when representing the real world within a data sample and represents the distribution stated in (7). For example, in regression tasks noise in the input and target measurements causes data uncertainty that the network cannot learn to correct. In classification tasks, samples which do not contain enough information in order to identify one class with 100% certainty cause data uncertainty on the prediction. The information loss is a result of the measurement system, e.g. by representing real world information by image pixels with a specific resolution, or by errors in the labelling process. Considering the five presented factors for uncertainties on a neural network's prediction, model uncertainty covers Factors I, III, IV, and V and data uncertainty is related to Factor II. While model uncertainty can be (theoretically) reduced by improving the architecture, the learning process, or the training data set, the data uncertainties cannot be explained away [60]. Therefore, DNNs that are capable of handling uncertain inputs and that are able to remove or quantify the model uncertainty and give a correct prediction of the data uncertainty are of paramount importance for a variety of real world mission- and safety-critical applications.

The Bayesian framework offers a practical tool to reason about uncertainty in deep learning [61]. In Bayesian modeling, the model uncertainty is formalized as a probability distribution over the model parameters θ, while the data uncertainty is formalized as a probability distribution over the model outputs y^*, given a parameterized model f_θ. The distribution over a prediction y^*, the predictive distribution, is then given by

$$p(y^*|x^*, D) = \int \underbrace{p(y^*|x^*, \theta)}_{\text{Data}} \underbrace{p(\theta|D)}_{\text{Model}} d\theta \ . \qquad (11)$$

The term $p(\theta|D)$ is referenced as posterior distribution on the model parameters and describes the uncertainty on the model parameters given a training data set D. The posterior distribution is in general not tractable. While ensemble approaches seek to approximate it by learning several different parameter settings and averaging over the resulting models [31], Bayesian inference reformulates it using Bayes Theorem [62]

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \ . \qquad (12)$$

The term $p(\theta)$ is called the prior distribution on the model parameters, since it does not take any information but the general knowledge on θ into account. The term $p(D|\theta)$ represents the likelihood that the data in D is a realization of the distribution predicted by a model parameterized with θ. Many loss functions are motivated by or can be related to the likelihood function. Loss functions that seek to maximize the log-likelihood (for an assumed distribution) are for example the cross-entropy or the mean squared error [63].

Even with the reformulation given in (12), the predictive distribution given in (11) is still intractable. To overcome this, several different ways to approximate the predictive distribution were proposed. A broad overview on the different concepts and some specific approaches is presented in Section III.

2) Distributional Uncertainty: Depending on the approaches that are used to quantify the uncertainty in y^*, the formulation of the predictive distribution might be further separated into data, distributional, and model parts [32]:

$$p(y^*|x^*, D) = \int \int \underbrace{p(y|\mu)}_{\text{Data}} \underbrace{p(\mu|x^*, \theta)}_{\text{Distributional}} \underbrace{p(\theta|D)}_{\text{Model}} d\mu d\theta \ . \qquad (13)$$

The distributional part in (13) represents the uncertainty on the actual network output, e.g. for classification tasks this might be a Dirichlet distribution, which is a distribution over the categorical distribution given by the softmax output. Modeled this way, distributional uncertainty refers to uncertainty that is caused by a change in the input-data distribution, while model uncertainty refers to uncertainty that is caused by the process of building and training the DNN. As modeled in (13), the model uncertainty affects the estimation of the distributional uncertainty, which affects the estimation of the data uncertainty.

While most methods presented in this paper only distinguish between model and data uncertainty, approaches specialized on out-of-distribution detection often explicitly aim at representing the distributional uncertainty [32], [64]. A more detailed presentation of different approaches for quantifying uncertainties in neural networks is given in Section III. In Section IV, different measures for measuring the different types of uncertainty are presented.

E. Uncertainty Classification

On the basis of the input data domain, the predictive uncertainty can also be classified into three main classes:

- *In-domain uncertainty* [65]
  In-domain uncertainty represents the uncertainty related to an input drawn from a data distribution assumed to be equal to the training data distribution. The in-domain uncertainty stems from the inability of the deep neural network to explain an in-domain sample due to lack of in-domain knowledge. From a modeler point of view, in-domain uncertainty is caused by design errors (model uncertainty) and the complexity of the problem at hand (data uncertainty). Depending on the source of the in-domain uncertainty, it might be reduced by increasing the quality of the training data (set) or the training process [57].

- *Domain-shift uncertainty* [13]
  Domain-shift uncertainty denotes the uncertainty related to an input drawn from a shifted version of the training distribution. The distribution shift results from insufficient coverage by the training data and the variability inherent to real world situations. A domain-shift might increase the uncertainty due to the inability of the DNN to explain the domain shift sample on the basis of the seen samples at training time. Some errors causing domain shift uncertainty can be modeled and can therefore be reduced. For example, occluded samples can be learned by the deep neural network to reduce domain shift uncertainty caused by occlusions [66]. However, it is difficult if
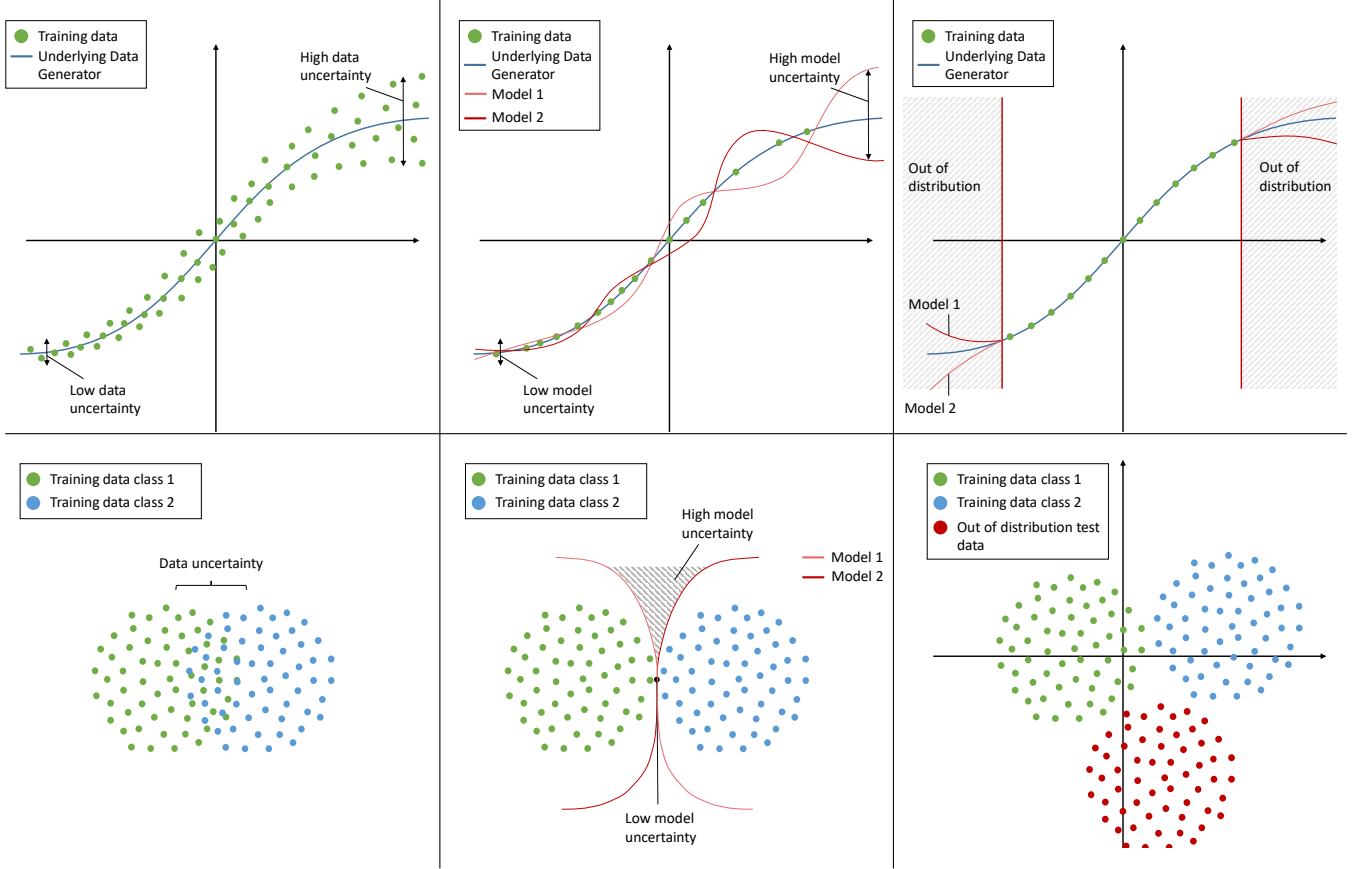
Fig. 1: Visualization of the data, the model, and the distributional uncertainty for classification and regression models.

not impossible to model all errors causing domain shift uncertainty, e.g., motion noise [60]. From a modeler point of view, domain-shift uncertainty is caused by external or environmental factors but can be reduced by covering the shifted domain in the training data set.

- *Out-of-domain uncertainty* [67], [68], [69], [70]
  Out-of-domain uncertainty represents the uncertainty related to an input drawn from the subspace of unknown data. The distribution of unknown data is different and far from the training distribution. While a DNN can extract in-domain knowledge from domain-shift samples, it cannot extract in-domain knowledge from out-of-domain samples. For example, when domain-shift uncertainty describes phenomena like a blurred picture of a dog, out-of-domain uncertainty describes the case when a network that learned to classify cats and dogs is asked to predict a bird. The out-of-domain uncertainty stems from the inability of the DNN to explain an out-of-domain sample due to its lack of out-of-domain knowledge. From a modeler point of view, out-of-domain uncertainty is caused by input samples, where the network is not meant to give a prediction for or by insufficient training data.

Since the model uncertainty captures what the DNN does not know due to lack of in-domain or out-of-domain knowledge, it captures all, in-domain, domain-shift, and out-of-domain uncertainties. In contrast, the data uncertainty captures in-

domain uncertainty that is caused by the nature of the data the network is trained on, as for example overlapping samples and systematic label noise.

III. UNCERTAINTY ESTIMATION

As described in Section II, several factors may cause model and data uncertainty and affect a DNN's prediction. This variety of sources of uncertainty makes the complete exclusion of uncertainties in a neural network impossible for almost all applications. Especially in practical applications employing real world data, the training data is only a subset of all possible input data, which means that a miss-match between the DNN domain and the unknown actual data domain is often unavoidable. However, an exact representation of the uncertainty of a DNN prediction is also not possible to compute, since the different uncertainties can in general not be modeled accurately and are most often even unknown. Therefore, methods for estimating uncertainty in a DNN prediction is a popular and vital field of research. The data uncertainty part is normally represented in the prediction, e.g. in the softmax output of a classification network or in the explicit prediction of a standard deviation in a regression network [60]. In contrast to this, several different approaches which model the model uncertainty and seek to separate it from the data uncertainty in order to receive an accurate representation of the data uncertainty were introduced [60], [32], [31].
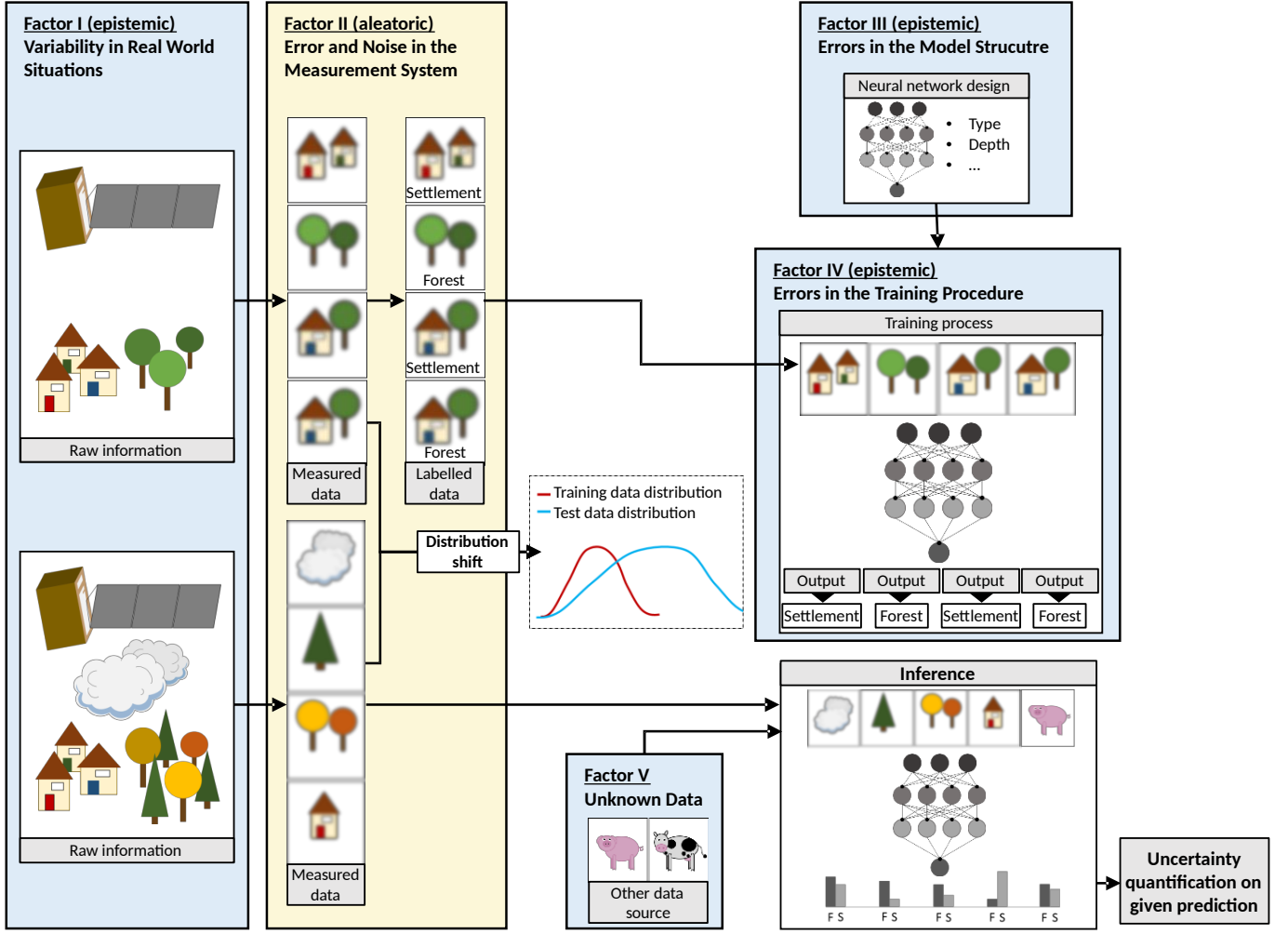
Fig. 2: The illustration shows the different steps of a neural network pipeline, based on the earth observation example of land cover classification (here settlement and forest) based on optical images. The different factors that affect the predictive uncertainty are highlighted in the yellow boxes. Factor I is shown as changing environments by cloud covered trees, different types and colors of trees. Factor II is shown by insufficient measurements, that can not directly be used to separate between settlement and forest and by label noise. In practice, the resolution of such images can be low and which would also be part of Factor II. Factor II and Factor III represent the uncertainties caused by the network structure and the stochastic training process. Factor V in contrast is represented by feeding the trained network with unknown types of images, namely cows and pigs.
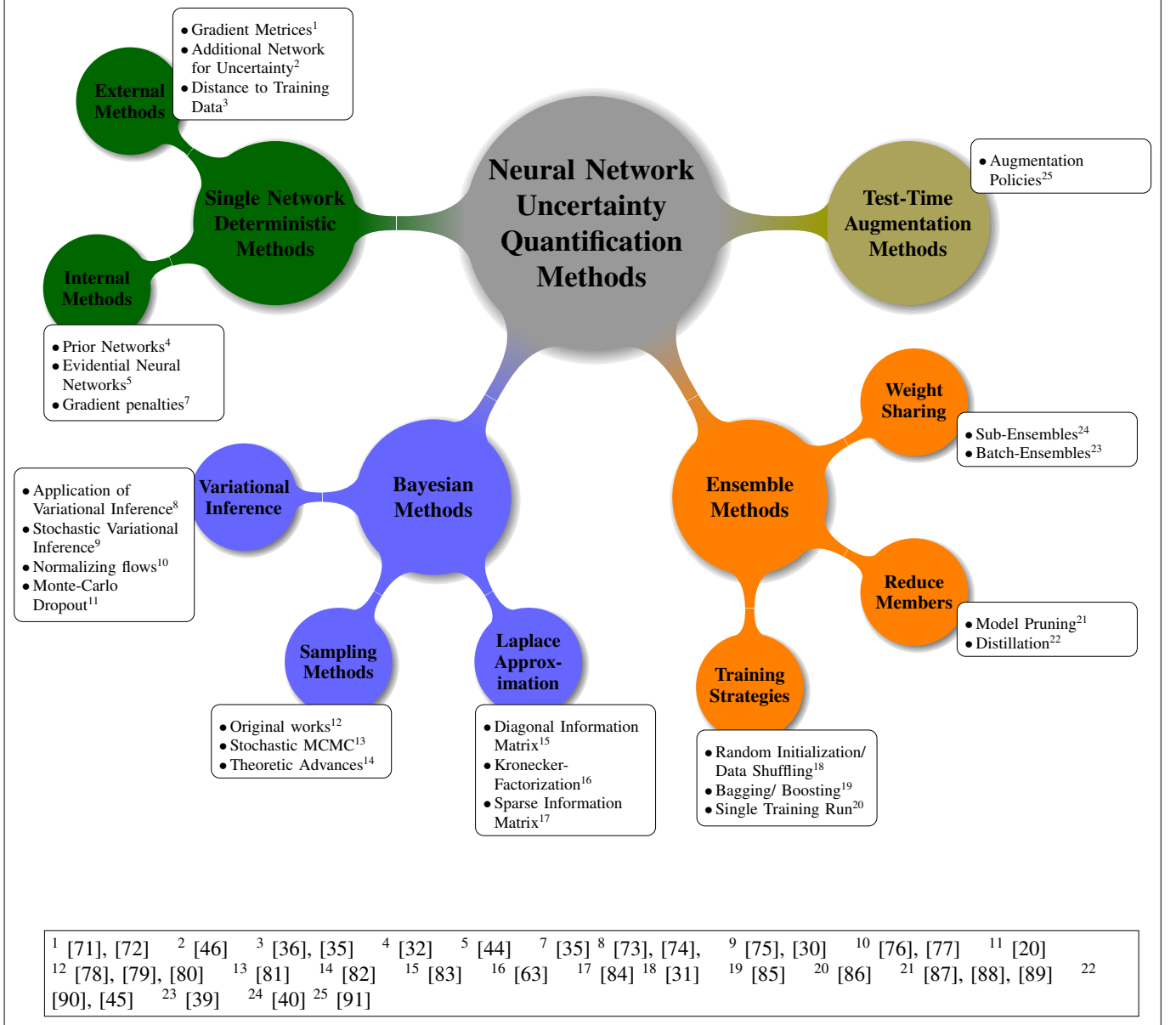
In general, the methods for estimating the uncertainty can be split in four different types based on the number (single or multiple) and the nature (deterministic or stochastic) of the used DNNs.

- **Single deterministic methods** give the prediction based on one single forward pass within a deterministic network. The uncertainty quantification is either derived by using additional (external) methods or is directly predicted by the network.
- **Bayesian methods** cover all kinds of stochastic DNNs, i.e. DNNs where two forward passes of the same sample generally lead to different results.
- **Ensemble methods** combine the predictions of several different deterministic networks at inference.

- **Test-time augmentation methods** give the prediction based on one single deterministic network but augment the input data at test-time in order to generate several predictions that are used to evaluate the certainty of the prediction.

In the following, the main ideas and further extensions of the four types are presented and their main properties are discussed. In Figure 3, an overview of the different types and methods is given. In Figure 4, the different underlying principles that are used to differentiate between the different types of methods are presented. Table I summarizes the main properties of the methods presented in this work, such as complexity, computational effort, memory consumption, flexibility, and others.

Fig. 3: Visualization of the four different types of uncertainty quantification methods presented in this paper.

^{1} [71], [72] ^{2} [46] ^{3} [36], [35] ^{4} [32] ^{5} [44] ^{7} [35] ^{8} [73], [74], ^{9} [75], [30] ^{10} [76], [77] ^{11} [20]
^{12} [78], [79], [80] ^{13} [81] ^{14} [82] ^{15} [83] ^{16} [63] ^{17} [84] ^{18} [31] ^{19} [85] ^{20} [86] ^{21} [87], [88], [89] ^{22} [90], [45] ^{23} [39] ^{24} [40] ^{25} [91]

A. Single Deterministic Methods

For deterministic neural networks the parameters are deterministic and each repetition of a forward pass delivers the same result. With single deterministic network methods for uncertainty quantification, we summarize all approaches where the uncertainty on a prediction y^* is computed based on one single forward pass within a deterministic network. In the literature, several such approaches can be found. They can be roughly categorized into approaches where one single network is explicitly modeled and trained in order to quantify uncertainties [44], [32], [92], [64], [93] and approaches that use additional components in order to give an uncertainty estimate on the prediction of a network [46], [36], [71], [72]. While for the first type, the uncertainty quantification affects the training procedure and the predictions of the network, the latter type is in general applied on already trained networks. Since trained networks are not modified by such methods, they have no effect on the network's predictions. In the following, we call these two types *internal* and *external* uncertainty quantification approaches.

1) Internal Uncertainty Quantification Approaches: Many of the internal uncertainty quantification approaches followed the idea of predicting the parameters of a distribution over the predictions instead of a direct pointwise maximum-a-posteriori estimation. Often, the loss function of such networks takes the expected divergence between the true distribution and the predicted distribution into account as e.g., in [32], [94]. The distribution over the outputs can be interpreted as a quantification of the model uncertainty (see Section II), trying to emulate the behavior of a Bayesian modeling of the network

TABLE I: An overview about the four general methods presented in this paper, namely Bayesian Neural Networks, Ensembles, Single Deterministic Neural Networks, and Test-Time Data Augmentation. The labels high and low are given relative to the other approaches and based on the general idea behind them.

	Single Deterministic Networks	Bayesian Methods	Ensemble Methods	Test-Time Data Augmentation
Description	The predictions of several models are combined into one prediction. A variety among the single models is crucial.	Approaches that receive an uncertainty quantification on a prediction of a deterministic neural network.	Model parameters are explicitly modeled as random variables. For a single forward pass the parameters are sampled from this distribution. Therefore, the prediction is stochastic and each prediction is based on different model weights.	The prediction and uncertainty quantification at inference is based on several predictions resulting from different augmentations of the original input sample.
Description of Model Uncertainties	No	Yes	No	No
Need changes on existing networks	Depends on method	Yes	Yes (retrain several times)	No
Sensitivity to initialization and parameters of training process	High (in general)	Low (Usage of uninformative priors possible)	Low	Low
Number of networks trained	1	1	Several	1
Computational effort during training	Low	High	High	Low
Memory consumption during training	Low	Low	High	Low
Number of inputs per prediction	1	1	1	Several
Forward passes per prediction	1	Several	Several	Several
Evaluated modes	Single	Single	Multiple	Single
Computational effort during inference	Low (One forward pass, possibly some minor additional effort for uncertainty quantification)	High (sampling is either needed for explicit approach or for the approximation of intractable formulas)	High (Several models need to be evaluated)	High (Several augmentations and forward passes are performed)
Memory Consumption - Inference	Low	Low	High	Low

parameters [64]. The prediction is then given as the expected value of the predicted distribution.

For classification tasks, the output in general represents class probabilities. These probabilities are a result of applying the softmax function

$$\text{softmax} : \mathbb{R}^K \to \left\{ z \in \mathbb{R}^K | z_i \geq 0, \sum_{k=1}^{K} z_k = 1 \right\}$$

$$\text{softmax}(z)_j = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)} \tag{14}$$

for multiclass settings and the sigmoid function

$$\text{sigmoid} : \mathbb{R} \to [0, 1]$$

$$\text{sigmoid}(z) = \frac{1}{1 + \exp(-z)} \tag{15}$$

for binary classification tasks on the logits z. These probabilities can be already interpreted as a prediction of the data

uncertainty. However, it is widely discussed that neural networks are often over-confident and the softmax output is often poorly calibrated, leading to inaccurate uncertainty estimates [95], [67], [44], [92]. Furthermore, the softmax output cannot be associated with model uncertainty. But without explicitly taking the model uncertainty into account, out-of-distribution samples could lead to outputs that certify a false confidence. For example, a network trained on cats and dogs will very likely not result in 50% dog and 50% cat when it is fed with the image of a bird. This is, because the network extracts features from the image and even though the features do not fit to the cat class, they might fit even less to the dog class. As a result, the network puts more probability on cat. Furthermore, it was shown that the combination of rectified linear unit (ReLu) networks and the softmax output leads to settings where the network becomes more and more confident as the distance between an out-of-distribution sample and the
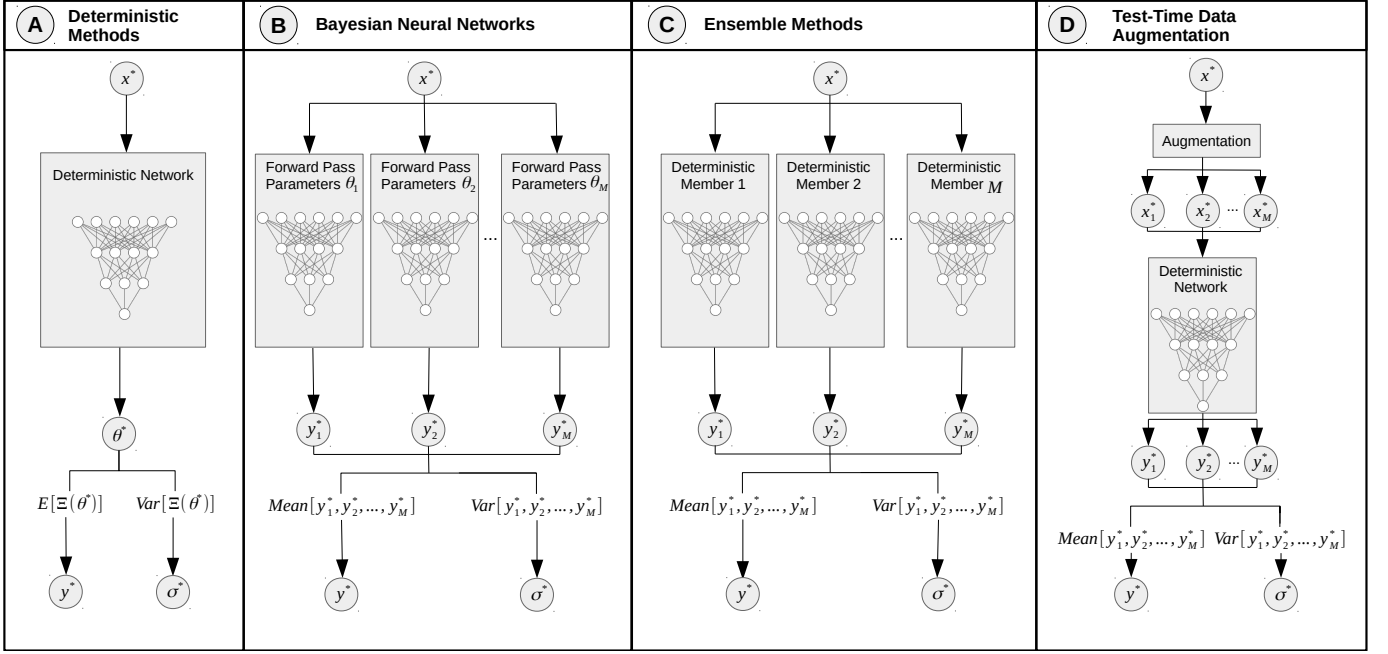
Fig. 4: A visualization of the basic principles of uncertainty modeling of the four presented general types of uncertainty prediction in neural networks. For a given input sample x^* each approach delivers a prediction y^*, a representation of model uncertainty σ_{model} and a value of data uncertainty σ_{model}. **A) single deterministic model**, **B) Bayesian neural network**, **B) ensemble approach**, and **D) test-time data augmentation**. The mean and the standard deviation are only used to keep the visualization simple. In practice other methods, could be utilized. For the deterministic approaches the idea of predicting the parameters of an probability distribution Ξ is visualized, other approaches which base on tools additional to the prediction network are not visualized here.

TABLE II: Overview over the properties of internal and external deterministic single network methods. For a comparison of single deterministic network approaches with Bayesian, ensemble, and test-time augmentation methods, see Table I.

	Internal Methods	External Methods
Description	Estimate uncertainty using one evaluation of a single network without external components.	Estimate uncertainty using one evaluation of the network while relying on additional external components.
Implementation effort	Relatively low, but depends on explicit approach, often only loss and network output has to be fixed.	Relatively low, but depends on explicit approach.
Application on already trained networks possible	No	Yes
Separated prediction and uncertainty estimation	No	Yes

learned training set becomes larger [96]. Figure 5 shows an example where the rotation of a digit from MNIST leads to false predictions with high softmax values. This phenomenon is described and further investigated by Hein et al. [96] who proposed a method to avoid this behaviour, based on enforcing a uniform predictive distribution far away from the training data.

Several other classification approaches [44], [32], [94], [64] followed a similar idea of taking the logit magnitude into account, but make use of the Dirichlet distribution. The Dirichlet distribution is the conjugate prior of the categorical distribution and hence can be interpreted as a distribution over categorical distributions. The density of the Dirichlet distribution is defined by

$$\text{Dir}(\mu|\alpha) = \frac{\Gamma(\alpha_0)}{\prod_{c=1}^{K} \Gamma(\alpha_c)} \prod_{c=1}^{K} \mu_c^{\alpha_c - 1}, \quad \alpha_c > 0, \; \alpha_0 = \sum_{c=1}^{K} \alpha_c \quad,$$

where Γ is the gamma function, $\alpha_1, ..., \alpha_K$ are called the concentration parameters, and the scalar α_0 is the precision of the distribution. In practice, the concentrations $\alpha_1, ..., \alpha_K$ are derived by applying a strictly positive transformation, as for example the exponential function, to the logit values. As visualized in Figure 6, a higher concentration value leads to a sharper Dirichlet distribution.

Fig. 5: Predictions received from a LeNet network trained on MNIST's handwritten digits from 0 to 9 and evaluated on different rotations of test samples. One can clearly see, that for some rotations the network gives a high confidence on the false class due to confusion (e.g.: 3 is confused with 8) or representations not seen at training. These examples represent a simple case of how a basic classification network can lead to overconfident wrong predictions under data distribution shifts.

(a) Low uncertainty (b) Data uncertainty (c) Out-of-dist.
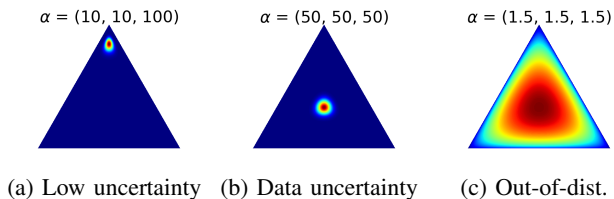
Fig. 6: The desired behaviors of a Dirichlet distribution over categorical distributions. The visualizations show three Dirichlet distributions over three classes. Each node of the simplex represents one class. In (a) the sharp Dirichlet distribution with its expectation close to the upper node represents a certain prediction of a categorical distribution. In (b) the sharp Dirichlet distribution in the center of the simplex represents high data uncertainty but low distributional uncertainty. In (c) the flat Dirichlet distribution indicates high distributional uncertainty.

The set of all class probabilities of a categorical distribution over k classes is equivalent to a $k - 1$-dimensional standard or probability simplex. Each node of this simplex represents a probability vector with the full probability mass on one class and each convex combination of the nodes represents a categorical distribution with the probability mass distributed over multiple classes. Malinin et al. [32] argued that a high model uncertainty should lead to a lower precision value and therefore to a flat distribution over the whole simplex, since the network is not familiar with the data. In contrast to this, data uncertainty should be represented by a sharper but also centered distribution, since the network can handle the data, but cannot give a clear class preference. In Figure 6 the different desired behaviors are shown. The Dirichlet distribution is utilized in several approaches as *Dirichlet Prior Networks* [43], [32] and *Evidential Neural Networks* [97], [44]. Both

of these network types output the parameters of a Dirichlet distribution from which the categorical distribution describing the class probabilities can be derived. The general idea of prior networks [32] is already described above and is visualized in Figure 6. Prior networks are trained in a multi task way with the goal of minimizing the expected Kullback-Leibler (KL) divergence between the predictions of in-distribution data and a sharp Dirichlet distribution and between a flat Dirichlet distribution and the predictions of out-of-distribution data [32]. Besides the main motivation of a better separation between in-distribution and OOD samples, these approaches also improve the separation between the confidence of correct and incorrect predictions, as was shown by [98]. As a follow up, [94] discussed that for the case that the data uncertainty is high, the forward definition of the KL-divergence can lead to an undesirable multi-model target distribution. In order to avoid this, they reformulated the loss using the reverse KL-divergence. The experiments showed improved results in the uncertainty estimation as well as for the adversarial robustness. Zhao et al. [99] extended the Dirichlet network approach by a new loss function that aims at minimizing an upper bound on the expected error based on the $\mathcal{L}_\infty$-norm, i.e. optimizing an expected worst-case upper bound. [34] argued that using a mixture of Dirichlet distributions gives much more flexibility in approximating the posterior distribution. Therefore, an approach where the network predicts the parameters for a mixture of K Dirichlet distributions was suggested. For this, the network logits represent the parameters for M Dirichlet distributions and additionally M weights $\omega_i, i = 1, .., M$ with the constraint $\sum_{i=1}^{M} \omega_i = 1$ are optimized. Nandy et al. [64] analytically showed that for in-domain samples with high data uncertainty, the Dirichlet distribution predicted for a false prediction is often flatter than for a correct prediction. They argued that this makes it harder to differentiate between in- and out-of-distribution predictions and suggested a regularization term towards maximizing the gap between in- and out-of-distribution samples.

Evidential neural networks [44] also optimize the parameterization of a single Dirichlet network. The loss formulation is derived by using subjective logic and interpret the logits as multinomial opinions or beliefs, as introduced in Evidence or Dempster-Shafer theory [100]. Evidential neural networks set the total amount of evidence in relation with the number of classes and conclude a value of uncertainty from this, i.e. receiving an additional "I don't know class". The loss is formulated as expected value of a basic loss, as for example categorical cross entropy, with respect to a Dirichlet distribution parameterized by the logits. Additionally, a regularization term is added, encouraging the network to not consider features that provide evidence for multiple classes at the same time, as for example a circle is for 6 and 8. Due to this, the networks do not differentiate between data uncertainty and model uncertainty, but learn whether they can give a certain prediction or not. [33] extended this idea by differentiating between acuity and dissonance in the collected evidence in order to better separate in- and out-of-distribution samples. For that, two explicit data sets containing overlapping classes and out-of-distribution samples are needed to learn a

regularization term. Amini et al. [101] transferred the idea of evidential neural networks from classification tasks to regression tasks by learning the parameters of an evidential normal inverse gamma distribution over an underlying Normal distribution. Charpentier et al. [102] avoided the need of OOD data for the training process by using normalizing flows to learn a distribution over a latent space for each class. A new input sample is projected onto this latent space and a Dirichlet distribution is parameterized based on the class wise densities of the received latent point.

Beside the Dirichlet distribution based approaches described above, several other internal approaches exist. In [68], a relatively simple approach based on small pertubations on the training input data and the temperature scaling calibration is presented leading to an efficient differentiation of in- and out-of-distritbuion samples. Możejko et al. [92] made use of the inhibited softmax function. It contains an artificial and constant logit that makes the absolute magnitude of the single logits more determining in the softmax output. Van Amersfoort et al. [35] showed that Radial Basis Function (RBF) networks can be used to achieve competitive results in accuracy and very good results regarding uncertainty estimation. RBF networks learn a linear transformation on the logits and classify inputs based on the distance between the transformed logits and the learned class centroids. In [35], a scaled exponentiated L_2 distance was used. The data uncertainty can be directly derived from the distances between the centroids. By including penalties on the Jacobian matrix in the loss function, the network was trained to be more sensitive to changes in the input space. As a result, the method reached good performance on out-of-distribution detection. In several tests, the approach was compared to a five members deep ensemble [31] and it was shown that this single network approach performs at least equivalently well on detecting out-of-distribution samples and improves the true-positive rate.

For regression tasks, Oala et al. [93] introduced an uncertainty score based on the lower and an upper bound output of an interval neural network. The interval neural network has the same structure as the underlying deterministic neural network and is initialized with the deterministic network's weights. In contrast to Gaussian representations of uncertainty given by a standard deviation, this approach can give non-symmetric values of uncertainty. Furthermore, the approach is found to be more robust in the presence of noise. Tagasovska and Lopez-Paz [103] presented an approach to estimate data and model uncertainty. A simultaneous quantile regression loss function was introduced in order to generate well-calibrated prediction intervals for the data uncertainty. The model uncertainty is quantified based on a mapping from the training data to zero, based on so called Orthonormal Certificates. The aim was that out-of-distribution samples, where the model is uncertain, are mapped to a non-zero value and thus can be recognized. Kawashima et al. [104] introduced a method which computes virtual residuals in the training samples of a regression task based on a cross-validation like pre-training step. With original training data expanded by the information of these residuals, the actual predictor is trained to give a prediction and a value of certainty. The experiments indicated that the virtual residu-

als represent a promising tool in order to avoid overconfident network predictions.

2) External Uncertainty Quantification Approaches: External uncertainty quantification approaches do not affect the models' predictions, since the evaluation of the uncertainty is separated from the underlying prediction task. Furthermore, several external approaches can be applied to already trained networks at the same time without affecting each other. Raghu et al. [46] argued that when both tasks, the prediction and the uncertainty quantification, are done by one single method, the uncertainty estimation is biased by the actual prediction task. Therefore, they recommended a "direct uncertainty prediction" and suggested to train two neural networks, one for the actual prediction task and a second one for the prediction of the uncertainty on the first network's predictions. Similarly, Ramalho and Miranda [36] introduced an additional neural network for uncertainty estimation. But in contrast to [46], the representation space of the training data is considered and the density around a given test sample is evaluated. The additional neural network uses this training data density in order to predict whether the main network's estimate is expected to be correct or false. Hsu et al. [105] detected out-of-distribution examples in classification tasks at test-time by predicting total probabilities for each class, additional to the categorical distribution given by the softmax output. The class wise total probability is predicted by applying the sigmoid function to the network's logits. Based on these total probabilities, OOD examples can be identified as those with low class probabilities for all classes.

In contrast to this, Oberdiek et al. [71] took the sensitivity of the model, i.e. the model's slope, into account by using gradient metrics for the uncertainty quantification in classification tasks. Lee et al. [72] applied a similar idea but made use of back propagated gradients. In their work they presented state of the art results on out-of-distribution and corrupted input detection.

3) Summing Up Single Deterministic Methods: Compared to many other principles, single deterministic methods are computational efficient in training and evaluation. For training, only one network has to be trained and often the approaches can even be applied on pre-trained networks. Depending on the actual approach, only a single or at most two forward passes have to be fulfilled for evaluation. The underlying networks could contain more complex loss functions, which slows down the training process [44] or external components that have to be trained and evaluated additionally [46]. But in general, this is still more efficient than the number of predictions needed for ensembles based methods (Section III-C), Bayesian methods (Section III-B), and test-time data augmentation methods (Section III-D). A drawback of single deterministic neural network approaches is the fact that they rely on a single opinion and can therefore become very sensitive to the underlying network architecture, training procedure, and the training data.

B. Bayesian Neural Networks

Bayesian Neural Networks (BNNs) [106], [107], [108] have the ability to combine the scalability, expressiveness, and

TABLE III: Overview over the properties different types of Bayesian neural networks approaches. The properties are stated relatively among the approaches. The properties can not used as comparison to other uncertainty methods as ensembles, single deterministic models, and test-time augmentation methods. For a comparison of Bayesian methods to these methods see Table ??

	Variational Inference	Sampling Approaches	Laplace Approximation
Description	Variational inference approaches approximate the (in general intractable) posterior distribution by optimizing over a family of tractable distributions. Achieved by minimizing the KL divergence.	Representation of the target random variable from which realizations can be sampled. Such methods are based on Markov Chain Monte Carlo and further extensions.	Simplifies the target distribution by approximating the log-posterior distribution and then, based on this approximation, deriving a normal distribution on the network weights.
Analytic expression	Yes	No	Yes
Can be applied to pretrained networks	No	No	Yes
Deterministic?	Yes	No	Yes
Unbiased?	No	Yes	No
Optimum	Local optimum	Hard to mix between modes	Local optimum
Convergence	Easy to assess convergence	Hard to assess convergence	Easy to assess convergence
Computational effort at training time	Medium - Convergence may be slowed down by regularization. Additional parameters for uncertainty representation.	High - M forward passes based on sampled parameters, otherwise intractable	Low - Training of one deterministic model and Laplace approximation
Computational effort at inference	High - M forward passes based on sampled parameters, otherwise intractable	High - M forward passes based on sampled parameters, otherwise intractable	High - M forward passes based on sampled parameters, otherwise intractable

predictive performance of neural networks with the Bayesian learning as opposed to learning via the maximum likelihood principles. This is achieved by inferring the probability distribution over the network parameters $\theta = (w_1, ..., w_K)$. More specifically, given a training input-target pair (x, y) the posterior distribution over the space of parameters $p(\theta|x, y)$ is modelled by assuming a prior distribution over the parameters $p(\theta)$ and applying Bayes theorem:

$$p(\theta|x,y) = \frac{p(y|x,\theta)p(\theta)}{p(y|x)} \propto p(y|x,\theta)p(\theta). \quad (16)$$

Here, the normalization constant in (16) is called the model evidence $p(y|x)$ which is defined as

$$p(y|x) = \int p(y|x,\theta)p(\theta)d\theta. \quad (17)$$

Once the posterior distribution over the weights have been estimated, the prediction of an output y^* for a new input data x^* can be obtained by Bayesian Model Averaging or Full Bayesian Analysis that involves marginalizing the likelihood $p(y|x, \theta)$ with the posterior distribution:

$$p(y^*|x^*, x, y) = \int p(y^*|x^*, \theta)p(\theta|x, y)d\theta. \quad (18)$$

This Bayesian way of prediction is a direct application of the law of total probability and endows the ability to compute the principled predictive uncertainty. The integral of (18) is intractable for the most common prior posterior pairs and approximation techniques are therefore typically applied. The most widespread approximation, the *Monte Carlo Approximation*, follows the law of large numbers and approximates

the expected value by the mean of N deterministic networks, $f_{\theta_1}, ..., f_{\theta_N}$, parameterized by N samples, $\theta_1, \theta_2, ..., \theta_N$, from the posterior distribution of the weights, i.e.

$$y^* \approx \frac{1}{N}\sum_{i=1}^{N} y_i^* = \frac{1}{N}\sum_{i=1}^{N} f_{\theta_i}(x^*). \quad (19)$$

Wilson and Izmailov [16] argue that a key advantage of BNNs lie in this marginalization step, which particularly can improve both the accuracy and calibration of modern deep neural networks. We note that the use-cases of BNNs are not limited for uncertainty estimation but open up the possibility to bridge the powerful Bayesian tool-boxes within deep learning. Notable examples include Bayesian model selection [109], [110], [111], [112], model compression [76], [113], [114], active learning [115], [23], [116], continual learning [117], [118], [119], [120], theoretic advances in Bayesian learning [121] and beyond.

While the formulation is rather simple, there exist several challenges. For example, no closed form solution exists for the posterior inference as conjugate priors do not typically exist for complex models such as neural networks [62]. Hence, approximate Bayesian inference techniques are often needed to compute the posterior probabilities. Yet, directly using approximate Bayesian inference techniques have been proven to be difficult as the size of the data and number of parameters are too large for the use-cases of deep neural networks. In other words, the integrals of above equations are not computationally tractable as the size of the data and number of parameters grows. Moreover, specifying a meaningful prior for deep neural networks is another challenge that is less understood.

In this survey, we classify the BNNs into three different types based on how the posterior distribution is inferred to approximate Bayesian inference:

- *Variational inference* [73], [74]
  Variational inference approaches approximate the (in general intractable) posterior distribution by optimizing over a family of tractable distributions.

- *Sampling approaches* [78]
  Sampling approaches deliver a representation of the target random variable from which realizations can be sampled. Such methods are based on Markov Chain Monte Carlo and further extensions.

- *Laplace approximation* [122], [123]
  Laplace approximation simplifies the target distribution by approximating the log-posterior distribution and then, based on this approximation, deriving a normal distribution over the network weights.

While limiting our scope to these three categories, we also acknowledge several advances in related domains of BNN research. Some examples are (i) approximate inference techniques such as alpha divergence [124], [125], [126], expectation propagation [127], [128], assumed density filtering [129] etc, (ii) probabilistic programming to exploit modern Graphical Processing Units (GPUs) [130], [131], [132], [133], (iii) different types of priors [134], [135], (iv) advancements in theoretical understandings of BNNs [136], [121], [137], (iv) uncertainty propagation techniques to speed up the marginalization procedures [138] and (v) computations of aleatoric uncertainty [139], [140], [141].

1) Variational Inference: The goal of variational inference is to infer the posterior probabilities $p(\theta|x, y)$ using a pre-specified family of distributions $q(\theta)$. Here, these so-called variational family $q(\theta)$ is defined as a parametric distribution. An example is the Multivariate Normal distribution where its parameters are the mean and the covariance matrix. The main idea of variational inference is to find the settings of these parameters that make $q(\theta)$ to be close to the posterior of interest $p(\theta|x, y)$. This measure of closeness between the probability distributions are given by the Kullback-Leibler (KL) divergence

$$\mathrm{KL}(q||p) = \mathbb{E}_q \left[\log \frac{q(\theta)}{p(\theta|x, y)} \right]. \tag{20}$$

Due to the posterior $p(\theta|x, y)$ the KL-divergence in (20) can not be minimized directly. Instead, the evidence lower bound (ELBO), a function that is equal to the KL divergence up to a constant, is optimized. For a given prior distribution on the parameters $p(\theta)$, the ELBO is given by

$$L = \mathbb{E}_q \left[\log \frac{p(y|x, \theta)}{q(\theta)} \right] \tag{21}$$

and for the KL divergence

$$\mathrm{KL}(q||p) = -L + \log p(y|x) \tag{22}$$

holds.

Variational methods for BNNs have been pioneered by Hinton and Van Camp [73] where the authors derived a diagonal Gaussian approximation to the posterior distribution of neural networks (couched in information theory - a minimum description length). Another notable extension in 1990s has been proposed by Barber and Bishop [74], in which the full covariance matrix was chosen as the variational family, and the authors demonstrated how the ELBO can be optimized for neural networks. Several modern approaches can be viewed as extensions of these early works [73], [74] with a focus on how to scale the variational inference to modern neural networks.

An evident direction with the current methods are the use of stochastic variational inference (or Monte-Carlo variational inference), where the optimization of ELBO is performed using mini-batch of data. One of the first connections to stochastic variational inference has been proposed by Graves et al. [75] with Gaussian priors. In 2015, Blundell et al. [30] introduced Bayes By Backprop, a further extension of stochastic variational inference [75] to non-Gaussian priors and demonstrated how the stochastic gradients can be made unbiased. Notable, Kingma et al. [142] introduced the local reparameterization trick to reduce the variance of the stochastic gradients. One of the key concepts is to reformulate the loss function of neural network as the ELBO. As a result the intractable posterior distribution is indirectly optimized and variational inference is compatible to back-propagation with certain modifications to the training procedure. These extensions widely focus on the fragility of stochastic variational inference that arises due to sensitivity to initialization, prior definition and variance of the gradients. These limitations have been addressed recently by Wu et al. [143], where a hierarchical prior was used and the moments of the variational distribution are approximated deterministically. Above works commonly assumed mean-field approximations as the variational family, neglecting the correlations between the parameters. In order to make more expressive variational distributions feasible for deep neural networks, several works proposed to infer using the matrix normal distribution [144], [145], [146] or more expressive variants [147], [148] where the covariance matrix is decomposed into the Kronecker products of smaller matrices or in a low rank form plus a positive diagonal matrix. A notable contribution towards expressive posterior distributions has been the use of normalizing flows [77], [149] - a hierarchical probability distribution where a sequence of invertible transformations are applied so that a simple initial density function is transformed into a more complex distribution. Interestingly, Farquhar et al. [137] argue that mean-field approximation is not a restrictive assumption, and the layer-wise weight correlations may not be as important as capturing the depth-wise correlations. While the claim of Farquhar et al. [137] may remain to be an open question, the mean-field approximations have an advantage on smaller computational complexities [137]. For example, Osawa et al. [150] demonstrated that variational inference can be scaled up to ImageNet size data-sets and architectures using multiple GPUs and proposed practical tricks such as data augmentation, momentum initialization

and learning rate scheduling.

One of the successes in variational methods have been accomplished by casting existing stochastic elements of deep learning as variational inference. A widely known example is Monte Carlo Dropout (MC Dropout) where the dropout layers are formulated as Bernoulli distributed random variables, and training a neural network with dropout layers can be approximated as performing variational inference [61], [20], [151]. A main advantage of MC dropout is that the predictive uncertainty can be computed by activating dropout not only during training, but also at test time. In this way, once the neural network is trained with dropout layers, the implementation efforts can be kept minimum and the practitioners do not need expert knowledge to reason about uncertainty - certain criteria that the authors are attributing to its success [20]. The practical values of this method has been demonstrated also in several works [152], [10], [21] and resulted in different extensions (evaluating the usage of different dropout masks for example for convolutional layers [153] or by changing the representations of the predictive uncertainty into model and data uncertainties [60]). Approaches that build upon the similar idea but randomly drop incoming activations of a node, instead of dropping an activation for all following nodes, were also proposed within the literature [37] and called drop connect. This was found to be more robust on the uncertainty representation, even though it was shown that a combination of both can lead to higher accuracy and robustness in the test predictions [154]. Lastly, connections of variation inference to Adam [155], RMS Prop [156] and batch normalization [157] have been further suggested in the literature.

2) Sampling Methods: Sampling methods, or also often called Monte Carlo methods, are another family of Bayesian inference algorithms that represent uncertainty without a parametric model. Specifically, sampling methods use a set of hypotheses (or samples) drawn from the distribution and offer an advantage that the representation itself is not restricted by the type of distribution (e.g. can be multi-modal or non-Gaussian) - hence probability distributions are obtained non-parametrically. Popular algorithms within this domain are particle filtering, rejection sampling, importance sampling and Markov Chain Monte Carlo sampling (MCMC) [62].

In case of neural networks, MCMC is often used since alternatives such as rejection and importance sampling are known to be inefficient for such high dimensional problems. The main idea of MCMC is to sample from arbitrary distributions by transition in state space where this transition is governed by a record of the current state and the proposal distribution that aims to estimate the target distribution (e.g. the true posterior). To explain this, let us start defining the Markov Chain: a Markov Chain is a distribution over random variables $x_1, \cdots, x_T$ which follows the state transition rule:

$$p(x_1, \cdots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1}), \qquad (23)$$

i.e. the next state only depends on the current state and not on any other former state. In order to draw samples from the true posterior, MCMC sampling methods first generate samples in an iterative and the Markov Chain fashion. Then, at each iteration, the algorithm decides to either accept or reject the samples where the probability of acceptance is determined by certain rules. In this way, as more and more samples are produced, their values can approximate the desired distribution.

Hamiltonian Monte Carlo or Hybrid Monte Carlo (HMC) [158] is an important variant of MCMC sampling method (pioneered by Neals [78], [79], [80], [159] for neural networks), and is often known to be the gold standards of Bayesian inference [159], [160], [125]. The algorithm works as follows: (i) start by initializing a set of parameters θ (either randomly or in a user-specific manner). Then, for a given number of total iterations, (ii) instead of a random walk, a momentum vector - an auxiliary variable ρ is sampled, and the current value of parameters θ is updated via the Hamiltonian dynamics:

$$H(\rho, \theta) = -\mathrm{log}p(\rho, \theta) = -\mathrm{log}p(\rho|\theta) - \mathrm{log}p(\theta). \qquad (24)$$

Defining the potential energy ($V(\theta) = -logp(\theta)$ and the kinetic energy $T(\rho|\theta) = -\mathrm{log}p(\rho|\theta)$, the update steps via Hamilton's equations are governed by,

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial \rho} = \frac{\partial T}{\partial \rho} \text{ and} \qquad (25)$$

$$\frac{d\rho}{dt} = -\frac{\partial H}{\partial \theta} = -\frac{\partial T}{\partial \theta} - \frac{\partial V}{\partial \theta}. \qquad (26)$$

The so-called leapfrog integrator is used as a solver [161]. (iii) For each step, a Metropolis acceptance criterion is applied to either reject or accept the samples (similar to MCMC). Unfortunately, HMC requires the processing of the entire data-set per iteration, which is computationally too expensive when the data-set size grows to million to even billions. Hence, many modern algorithms focus on how to perform the computations in a mini-batch fashion stochastically. In this context, for the first time, Welling and Teh [81] proposed to combine Stochastic Gradient Descent (SGD) with Langevin dynamics (a form of MCMC [162], [163], [159]) in order to obtain a scalable approximation to MCMC algorithm based on mini-batch SGD [164], [165]. The work demonstrated that performing Bayesian inference on Deep Neural Networks can be as simple as running a noisy SGD. This method does not include the momentum term of HMC via using the first order Langevin dynamics and opened up a new research area on Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC).

Consequently, several extensions are available which include the use of 2nd order information such as preconditioning and optimizing with the Fisher Information Matrix (FIM) [166], [167], [168], the Hessian [169], [170], [171], adapting preconditioning diagonal matrix [172], generating samples from non-isotropic target densities using Fisher scoring [173], and samplers in the Riemannian manifold [174] using the first order Langevin dynamics and Levy diffusion noise and momentum [175]. Within these methods, the so-called parameter dependent diffusion matrices are incorporated

with an intention to offset the stochastic perturbation of the gradient. To do so, the "thermostat" ideas [176], [177], [178] are proposed so that a prescribed constant temperature distribution is maintained with the parameter dependent noise. Ahn et al. [179] devised a distributed computing system for SG-MCMC to exploit the modern computing routines, while Wang et al. [180] showed that Generative Adversarial Models (GANs) can be used to distill the samples for improved memory efficiency, instead of distillation for enhancing the run-time capabilities of computing predictive uncertainty [181]. Lastly, other recent trends are techniques that reduce the variance [160], [182] and bias [183], [184] arising from stochastic gradients.

Concurrently, there have been solid advances in theory of SG-MCMC methods and their applications in practice. Sato and Nakagawa [185], for the first time, showed that the SGLD algorithm with constant step size weakly converges; Chen et al. [186] showed that faster convergence rates and more accurate invariant measures can be observed for SG-MCMCs with higher order integrators rather than a 1st order Euler integrator while Teh et al. [187] studied the consistency and fluctuation properties of the SGLD. As a result, verifiable conditions obeying a central limit theorem for which the algorithm is consistent, and how its asymptotic bias-variance decomposition depends on step-size sequences have been discovered. A more detailed review of the SG-MCMC with a focus on supporting theoretical results can be found in Nemeth and Fearnhead [82]. Practically, SG-MCMC techniques have been applied to shape classification and uncertainty quantification [188], empirically study and validate the effects of tempered posteriors (or called cold-posteriors) [189] and train a deep neural network in order to generalize and avoid over-fitting [190], [191].

3) Laplace Approximation: The goal of the Laplace Approximation is to estimate the posterior distribution over the parameters of neural networks $p(\theta \mid x, y)$ around a local mode of the loss surface with a Multivariate Normal distribution. The Laplace Approximation to the posterior can be obtained by taking the second-order Taylor series expansion of the log posterior over the weights around the MAP estimate $\hat{\theta}$ given some data (x, y). If we assume a Gaussian prior with a scalar precision value $\tau > 0$, then this corresponds to the commonly used L_2-regularization, and the Taylor series expansion results in

$$\log p(\theta \mid x, y) \approx \log p(\hat{\theta} \mid x, y) \\ + \frac{1}{2}(\theta - \hat{\theta})^T (H + \tau I)(\theta - \hat{\theta}),$$

where the first-order term vanishes because the gradient of the log posterior $\delta\theta = \nabla \log p(\theta \mid x, y)$ is zero at the maximum $\hat{\theta}$. Taking the exponential on both sides and approximating integrals by reverse engineering densities, the weight posterior is approximately a Gaussian with the mean $\hat{\theta}$ and the covariance matrix $(H + \tau I)^{-1}$ where H is the Hessian of $\log p(\theta \mid x, y)$. This means that the model uncertainty is represented by the

Hessian H resulting in a Multivariate Normal distribution:

$$p(\theta \mid x, y) \sim \mathcal{N}\left(\hat{\theta}, (H + \tau I)^{-1}\right). \tag{27}$$

In contrast to the two other methods described, the Laplace approximation can be applied on already trained networks, and is generally applicable when using standard loss functions such as MSE or cross entropy and piece-wise linear activations (e.g RELU). Mackay [123] and Denker et al. [122] have pioneered the Laplace approximation for neural networks in 1990s, and several modern methods provide an extension to deep neural networks [192], [193], [63], [84].

The core of the Laplace Approximation is the estimation of the Hessian. Unfortunately, due to the enormous number of parameters in modern neural networks, the Hessian matrices cannot be computed in a feasible way as opposed to relative smaller networks in Mackay [123] and Denker et al. [122]. Consequently, several different ways for approximating H have been proposed in the literature. A brief review is as follows. Instead of diagonal approximations (e.g. [194], [83]), several researchers have been focusing on including the off-diagonal elements (e.g. [195], [196] and [197]). Amongst them, layer-wise Kronecker Factor approximation of [198], [193], [192] and [199] have demonstrated a notable scalability [200]. A recent extension can be found in [201] where the authors propose to re-scale the eigen-values of the Kronecker factored matrices so that the diagonal variance in its eigenbasis is accurate. The work presents an interesting idea as one can prove that in terms of a Frobenius norm, the proposed approximation is more accurate than that of [193]. However, as this approximation is harmed by inaccurate estimates of eigenvectors, Lee et al. [84] proposed to further correct the diagonal elements in the parameter space.

Existing works obtain Laplace Approximation using various approximation of the Hessian in the line of fidelity-complexity trade-offs. For several works, an approximation using the diagonal of the Fisher information matrix or Gauss Newton matrix, leading to independently distributed model weights, have been utilized in order to prune weights [202] or perform continual learning in order to avoid catastrophic forgetting [203]. In Ritter et al. [63], the Kronecker factorization of the approximate block-diagonal Hessian [193], [192] have been applied to obtain scalable Laplace Approximation for neural networks. With this, the weights among different layers are still assumed to be independently distributed, but not the correlations within the same layer. Recently, building upon the current understandings of neural network's loss landscape that many eigenvalues of the Hessian tend to be zero, [84] developed a low rank approximation that leads to sparse representations of the layers' co-variance matrices. Furthermore, Lee et al. [84] demonstrated that the Laplace Approximation can be scaled to ImageNet size data-sets and architectures, and further showed that with the proposed sparsification technique, the memory complexity of modelling correlations can be made similar to the diagonal approximation. Lastly, Kristiadi et al. [204] proposed a simple procedure to compute the last-layer Gaussian approximation (neglecting the model uncertainty in all other layers of neural

networks), and showed that even such a minimalist solution can mitigate overconfidence predictions of ReLU networks. Recent efforts have extended the Laplace Approximation beyond the Hessian approximation. To tackle the widely known assumption that the Laplace Approximation is for the bell shaped true posterior and thus resulting in under-fitting behavior [63], Humt et al. [205] proposed to use Bayesian Optimization and showed that hyperparameters of the Laplace Approximation can be efficiently optimized with increaed calibration performance. Another work in this domain is by Kristiadi et al. [206], who proposed uncertainty units - a new type of hidden units that changes the geometry of the loss landscape so that more accurate inference is possible. While Shinde et al. [207] demonstrated practical effectiveness of the Laplace Approximation to the autonomous driving applications, Feng et al. [208] showed the possibility to (i) incorporate contextual information and (ii) domain adaptation in a semi-supervised manner within the context of image classification. This is achieved by designing unary potentials within a Conditional Random Field. Several real-time methods also exist that do not require multiple forward passes to compute the predictive uncertainty. So-called linearized Laplace Approximation has been proposed in [209], [210] using the ideas of Mackay [115] and have been extended with Laplace bridge for classification [211]. Within this framework, Daxberger et al. [212] proposed inferring the sub-networks to increase the expressivity of covariance propagation while remaining computationally tractable.

4) Sum Up Bayesian Methods: Bayesian methods for deep learning have emerged as a strong research domain by combining principled Bayesian learning for deep neural networks. A review of current BNNs has been provided with a focus on mostly, how the posterior $p(\theta|x, y)$ is inferred. As an observation, many of the recent breakthroughs have been achieved by performing approximate Bayesian inference in a mini-batch fashion (stochastically) or investigating relatively simple but scalable techniques such as MC-dropout or Laplace Approximation. As a result, several works demonstrated that the posterior inference in large scale settings are now possible [213], [150], [84], and the field has several practical approximation tools to compute more expressive and accurate posteriors since the revival of BNNs beyond the pioneers [73], [74], [78], [122], [123]. There are also emerging challenges on new frontiers beyond accurate inference techniques. Some examples are: (i) how to specify meaningful priors? [134], [135], (ii) how to efficiently marginalize over the parameters for fast predictive uncertainty? [181], [138], [211] (iii) infrastructures such as new benchmarks, evaluation protocols and software tools [214], [131], [132], [215], and (iv) towards better understandings on the current methodologies and their potential applications [137], [189], [216], [208].

C. Ensemble Methods

1) Principles of Ensemble Methods: Ensembles derive a prediction based on the predictions received from multiple so-called ensemble members. They target at a better generalization by making use of synergy effects among the different models, arguing that a group of decision makers tend to make better decisions than a single decision maker [217], [218]. For an ensemble $f : X \to Y$ with members $f_i : X \to Y$ for $i \in 1, 2, ..., M$, this could be for example implemented by simply averaging over the members' predictions,

$$f(x) := \frac{1}{M} \sum_{i=1}^{M} f_i(x) \ .$$

Based on this intuitive idea, several works applying ensemble methods to different kinds of practical tasks and approaches, as for example bioinformatics [219], [220], [221], remote sensing [222], [223], [224], or reinforcement learning [225], [226] can be found in the literature. Besides the improvement in the accuracy, ensembles give an intuitive way of representing the model uncertainty on a prediction by evaluating the variety among the member's predictions.

Compared to Bayesian and single deterministic network approaches, ensemble methods have two major differences. First, the general idea behind ensembles is relatively clear and there are not many groundbreaking differences in the application of different types of ensemble methods and their application in different fields. Hence, this section focuses on different strategies to train an ensemble and some variations that target on making ensemble methods more efficient. Second, ensemble methods were originally not introduced to explicitly handle and quantify uncertainties in neural networks. Although the derivation of uncertainty from ensemble predictions is obvious, since they actually aim at reducing the model uncertainty, ensembles were first introduced and discussed in order to improve the accuracy on a prediction [218]. Therefore, many works on ensemble methods do not explicitly take the uncertainty into account. Notwithstanding this, ensembles have been found to be well suited for uncertainty estimations in neural networks [31].

2) Single- and Multi-Mode Evaluation: One main point where ensemble methods differ from the other methods presented in this paper is the number of local optima that are considered, i.e. the differentiation into *single-mode* and *multi-mode* evaluation.

In order to create synergies and marginalise false predictions of single members, the members of an ensemble have to behave differently in case of an uncertain outcome. The mapping defined by a neural network is highly non-linear and hence the optimized loss function contains many local optima to which a training algorithm could converge to. Deterministic neural networks converge to one single local optimum in the solution space [227]. Other approaches, e.g. BNNs, still converge to one single optimum, but additionally take the uncertainty on this local optimum into account [227]. This means, that neighbouring points within a certain region around the solution also affect the loss and also influence the prediction of a test sample. Since these methods focus on single regions, the evaluation is called *single-mode* evaluation. In contrast to this, ensemble methods consist of several networks, which should converge to different local optima. This leads to a so called multi-mode evaluation [227].
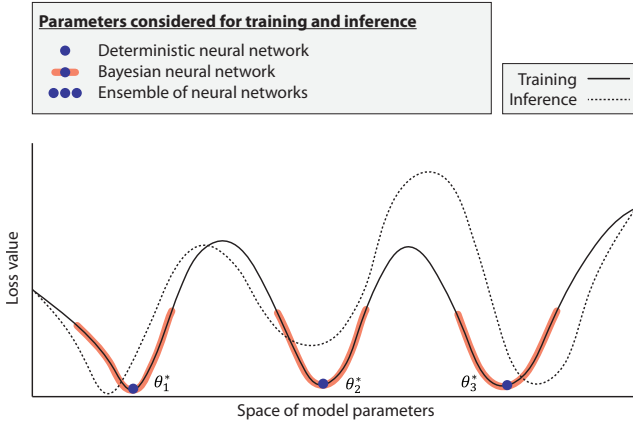
Fig. 7: A visualization of the different evaluation behaviours of deterministic neural networks, Bayesian neural networks and the ensemble of deterministic neural networks. The x-axis indicates the network parameters θ and the y-axis represents the loss value. While the deterministic network learns the parameters based on a pointwise estimation, the Bayesian neural network also takes the surrounding of the single point into account. The ensemble of deterministic methods optimizes pointwise but learns several different parameter settings.

In Figure 7, the considered parameters of a single-mode deterministic, single-mode probabilistic (Bayesian) and multi-mode ensemble approach are visualized. The goal of multi-mode evaluation is that different local optima could lead to models with different strengths and weaknesses in the predictions such that a combination of several such models brings synergy effects improving the overall performance.

3) Bringing Variety into Ensembles: One of the most crucial points when applying ensemble methods is to maximize the variety in the behaviour among the single networks [228], [31]. In order to increase the variety, several different approaches can be applied:

- **Random Initialization and Data Shuffle**
  Due to the very non-linear loss landscape, different initializations of a neural network lead in general to different training results. Since the training is realized on mini-batches, the order of the training data points also affects the final result.

- **Bagging and Boosting**
  Bagging (**B**ootstrap **agg**regat**ing**) and Boosting are two strategies that vary the distribution of the used training data sets by sampling new sets of training samples from the original set. Bagging is sampling from the training data uniformly and with replacement [62]. Thanks to the replacement process, ensemble members can see single samples several times in the training set while missing some other training samples. For boosting, the members are trained one after another and the probability of sampling a sample for the next training set is based

on the performance of the already trained ensemble [62].

- **Data Augmentation**
  Augmenting the input data randomly for each ensemble member leads to models trained on different data points and therefore in general to a larger variety among the different members.

- **Ensemble of different Network Architecture**
  The combination of different network architectures leads to different loss landscapes and can therefore also increase the diversity in the resulting predictions [229].

In several works, it has been shown that the variety induced by random initialization works sufficiently and that bagging could even lead to a weaker performance [230], [31]. Livieris et al. [231] evaluated different bagging and boosting strategies for ensembles of weight constrained neural networks. Interestingly, it is found that bagging performs better for a small number of ensemble members while boosting performs better for a large number. Nanni et al. [232] evaluated ensembles based on different types of image augmentation for bioimage classification tasks and compared those to each other. Guo and Gould [233] used augmentation methods within in an ensemble approach for object detection. Both works stated that the ensemble approach using augmentations improves the resulting accuracy. In contrast to this, [234], [235] stated with respect to uncertainty quantification that image augmentation can harm the calibration of an ensemble and post-processing calibration methods have to be slightly adapted when using ensemble methods. Other ways of inducing variety for specific tasks have been also introduced. For instance, in [236], the members are trained with different attention masks in order to focus on different parts of the input data. Other approaches focused on the training process and introduced learning rate schedulers that are designed to discover several local optima within one training process [86], [237]. Following, an ensemble can be built based on local optima found within one single training run. It is important to state that if not explicitly stated, the works and approaches presented so far targeted on improvements in the predictive accuracy and did not explicitly consider uncertainty quantification.

4) Ensemble Methods and Uncertainty Quantification: Besides the improvement in the accuracy, ensembles are widely used for modelling uncertainty on predictions of complex models, as for example in climate prediction [238], [239]. Accordingly, ensembles are also used for quantifying the uncertainty on a deep neural network's prediction, and over the last years they became more and more popular for such tasks [31], [228]. Lakshminarayanan et al. [31] are often referenced as a base work on uncertainty estimations derived from ensembles of neural networks and as a reference for the competitiveness of deep ensembles. They introduced an ensemble training pipeline to quantify predictive uncertainty within DNNs. In order to handle data and model uncertainty, the member networks are designed with two heads, representing the prediction and a predicted value of data uncertainty on the prediction. The approach

is evaluated with respect to accuracy, calibration, and out-of-distribution detection for classification and regression tasks. In all tests, the method performs at least equally well as the BNN approaches used for comparison, namely Monte Carlo Dropout and Probabilistic Backpropagation. Lakshminarayanan et al. [31] also showed that shuffling the training data and a random initialization of the training process induces a sufficient variety in the models in order to predict the uncertainty for the given architectures and data sets. Furthermore, bagging is even found to worsen the predictive uncertainty estimation, extending the findings of Lee et al. [230], who found bagging to worsen the predictive accuracy of ensemble methods on the investigated tasks. Gustafsson et al. [56] introduced a framework for the comparison of uncertainty quantification methods with a specific focus on real life applications. Based on this framework, they compared ensembles and Monte Carlo dropout and found ensembles to be more reliable and applicable to real life applications. These findings endorse the results reported by Beluch et al. [240] who found ensemble methods to deliver more accurate and better calibrated predictions on active learning tasks than Monte Carlo Dropout. Ovadia et al. [13] evaluated different uncertainty quantification methods based on test sets affected by distribution shifts. The excessive evaluation contains a variety of model types and data modalities. As a take away, the authors stated that already for a relatively small ensemble size of five, deep ensembles seem to perform best and are more robust to data set shifts than the compared methods. Vyas et al. [241] presented an ensemble method for the improved detection of out-of-distribution samples. For each member, a subset of the training data is considered as out-of-distribution. For the training process, a loss, seeking a minimum margin greater zero between the average entropy of the in-domain and the out-of-distribution subsets is introduced and leads to a significant improvement in the out-of-distribution detection.

5) Making Ensemble Methods more Efficient: Compared to single model methods, ensemble methods come along with a significantly increased computational effort and memory consumption [217], [45]. When deploying an ensemble for a real life application the available memory and computational power are often limited. Such limitations could easily become a bottleneck [242] and could become critical for applications with limited reaction time. Reducing the number of models leads to less memory and computational power consumption. *Pruning approaches* reduce the complexity of ensembles by pruning over the members and reducing the redundancy among them. For that, several approaches based on different diversity measures are developed to remove single members without strongly affecting the performance [88], [87], [243].

Distillation is another approach where the number of networks is reduced to one single model. It is the procedure of teaching a single network to represent the knowledge of a group of neural networks [244]. First works on the distillation of neural networks were motivated by restrictions when deploying large scale classification problems [244]. The original classification problem is separated into several sub-problems focusing on single blocks of classes that are difficult to differentiate. Several smaller trainer networks are trained on the sub-problems and then teach one student network to separate all classes at the same time. In contrast to this, *Ensemble distillation approaches* capture the behaviour of an ensemble by one single network. First works on ensemble distillation used the average of the softmax outputs of the ensemble members in order to teach a student network the derived predictive uncertainty [245]. Englesson and Azizpour [246] justify the resulting predictive distributions of this approach and additionally cover the handling of out-of-distribution samples. When averaging over the members' outputs, the model uncertainty, which is represented in the variety of ensemble outputs, gets lost. To overcome this drawback, researchers applied the idea of learning higher order distributions, i.e. distributions over a distribution, instead of directly predicting the output [90], [45]. The members are then distillated based on the divergence from the average distribution. The idea is closely related to the prior networks [32] and the evidential neural networks [44], which are described in Section III-A. [45] modelled ensemble members and the distilled network as prior networks predicting the parameters of a Dirichlet distribution. The distillation then seeks to minimize the KL divergence between the averaged Dirichlet distributions of the ensemble members and the output of the distilled network. Lindqvist et al. [90] generalized this idea to any other parameterizable distribution. With that, the method is also applicable to regression problems, for example by predicting a mean and standard deviation to describe a normal distribution. Within several tests, the distillation models generated by these approaches are able to distinguish between data uncertainty and model uncertainty. Although distillation methods cannot completely capture the behaviour of an underlying ensemble, it has been shown that they are capable of delivering good and for some experiments even comparable results [90], [45], [247].

Other approaches, as *sub-ensembles* [39] and *batch-ensembles* [40] seek to reduce the computation effort and memory consumption by sharing parts among the single members. It is important to note that the possibility of using different model architectures for the ensemble members could get lost when parts of the ensembles are shared. Also, the training of the models cannot be run in a completely independent manner. Therefore, the actual time needed for training does not necessarily decrease in the same way as the computational effort does.

Sub-ensembles [39] divide a neural network architecture into two sub-networks. The trunk network for the extraction of general information from the input data, and the task network that uses these information to fulfill the actual task. In order to train a sub-ensemble, first, the weights of each member's trunk network are fixed based on the resulting parameters of one single model's training process. Following, the parameters of each ensemble members' task network are trained independently from the other members. As a result, the members are built with a common trunk and an individual task sub-network. Since the training and the evaluation of

the trunk network have to be done only once, the number of computations needed for training and testing decreases by the factor $\frac{M \cdot N_{\text{task}} + N_{\text{trunk}}}{M \cdot N}$, where N_{task}, N_{trunk}, and N stand for the number of variables in the task networks, the trunk network, and the complete network. Valdenegro-Toro [39] further underlined the usage of a shared trunk network by arguing that the trunk network is in general computational more costly than the task network. In contrast to this, batch-ensembles [40] connect the member networks with each other at every layer. The ensemble members' weights are described as a Hadamard product of one shared weight matrix $W \in \mathbb{R}^{n \times m}$ and M individual rank one matrices $F_i \in \mathbb{R}^{n \times m}$, each linked with one of the M ensemble members. The rank one matrices can be written as a multiplication $F_i = r_i s_i^{\mathsf{T}}$ of two vectors $s \in \mathbb{R}^n$ and $r \in \mathbb{R}^m$ and hence the matrix F_i can be described by $n + m$ parameters. With this approach, each additional ensemble member increases the number of parameters only by the factor $\frac{n+m}{M \cdot (n+m) + n \cdot m} + 1$ instead of $\frac{M+1}{M} = 1 + \frac{1}{M}$. On the one hand, with this approach, the members are not independent anymore such that all the members have to be trained in parallel. On the other hand, the authors also showed that the parallelization can be realized similar to the optimization on mini-batches and on a single unit.

6) Sum Up Ensemble Methods: Ensemble methods are very easy to apply, since no complex implementation or major modification of the standard deterministic model have to be realized. Furthermore, ensemble members are trained independently from each other, which makes the training easily parallelizable. Also, trained ensembles can be extended easily, but the needed memory and the computational effort increases linearly with the number of members for training and evaluation. The main challenge when working with ensemble methods is the need of introducing diversity among the ensemble members. For accuracy, uncertainty quantification, and out-of-distribution detection, random initialization, data shuffling, and augmentations have been found to be sufficient for many applications and tasks [31], [232]. Since these methods may be applied anyway, they do not need much additional effort. The independence of the single ensemble members leads to a linear increase in the required memory and computation power with each additional member. This holds for the training as well as for testing. This limits the deployment of ensemble methods in many practical applications where the computation power or memory is limited, the application is time-critical, or very large networks with high inference time are included [45]. Many aspects of ensemble approaches are only investigated with respect to the performance on the predictive accuracy but do not take predictive uncertainty into account. This also holds for the comparison of different training strategies for a broad range of problems and data sets. Especially since the overconfidence from single members can be transferred to the whole ensemble, strategies that encourage the members to deliver different false predictions instead of all delivering the same false prediction should be further investigated. For a better understanding of ensemble behavior, further evaluations of the loss landscape, as done by Fort et al. [227], could offer interesting insights.

D. Test Time Augmentation

Inspired by ensemble methods and adversarial examples [14], the test time data augmentation is one of the simpler predictive uncertainty estimation techniques. The basic method is to create multiple test samples from each test sample by applying data augmentation techniques on it and then test all those samples to compute a predictive distribution in order to measure uncertainty. The idea behind this method is that the augmented test samples allow the exploration of different views and is therefore capable of capturing the uncertainty. Mostly, this technique of test time data augmentations has been used in medical image processing [248], [249], [14], [250]. One of the reasons for this is that the field of medical image processing already makes heavy use of data augmentations while using deep learning [251], so it is quite easy to just apply those same augmentations during test time to calculate the uncertainties. Another reason is that collecting medical images is costly, thus forcing practitioners to rely on data augmentation techniques. Moshkov et al. [250] used the test time augmentation technique for cell segmentation tasks. For that, they created multiple variations of the test data before feeding it to a trained UNet or Mask R-CNN architecture. Following, they used a majority voting to create the final output segmentation mask and discuss the policies of applying different augmentation techniques and how they affect the final predictive results of the deep networks.

Overall, test time augmentation is an easy method for estimating uncertainties because it keeps the underlying model unchanged, requires no additional data, and is simple to put into practice with off-the-shelf libraries. Nonetheless, it needs to be kept in mind that during applying this technique, one should only apply valid augmentations to the data, meaning that the augmentations should not generate data from outside the target distribution. According to [252], test time augmentation can change many correct predictions into incorrect predictions (and vice versa) due to many factors such as the nature of the problem at hand, the size of training data, the deep neural network architecture, and the type of augmentation. To limit the impact of these factors, Shanmugam et al. [252] proposed a learning-based method for test time augmentation that takes these factors into consideration. In particular, the proposed method learns a function that aggregates the predictions from each augmentation of a test sample. Similar to [252], Molchanov et al. [91] proposed a method, named "greedy Policy Search", for constructing a test-time augmentation policy by choosing augmentations to be include in a fixed-length policy. Similarly, Kim et al. [253] proposed a method for learning a loss predictor from the training data for instance-aware test-time augmentation selection. The predictor selects test-time augmentations with the lowest predicted loss for a given sample.

Although learnable test time augmentation techniques [252], [91], [253] help to select valid augmentations, one of the major open question is to find out the effect on uncertainty due to different kinds of augmentations. It can for example happen that a simple augmentation like reflection is not able to capture much of the uncertainty while some domain

specialized stretching and shearing captures more uncertainty. It is also important to find out how many augmentations are needed to correctly quantify uncertainties in a given task. This is particularly important in applications like earth observation, where inference might be needed on global scale with limited resources.

E. Neural Network Uncertainty Quantification Approaches for Real Life Applications

In order to use the presented methods on real life tasks, several different considerations have to be taken into account. The memory and computational power is often restricted while many real world tasks my be time-critical [242]. An overview over the main properties is given in Table I.
The presented applications all come along with advantages and disadvantages, depending on the properties a user is interested in. While ensemble methods and test-time augmentation methods are relatively easy to apply, Bayesian approaches deliver a clear description of the uncertainty on the models parameters and also deliver a deeper theoretical basis. The computational effort and memory consumption is a common restriction on real life applications, where single deterministic network approaches perform best, but distillation of ensembles or efficient Bayesian methods can also be taken into consideration. Within the different types of Bayesian approaches, the performance, the computational effort, and the implementation effort still vary strongly. Laplace approximations are relatively easy to apply and compared to sampling approaches much less computational effort is needed. Furthermore, there often already exist pretrained networks for an application. In this case, Laplace Approximation and external deterministic single network approaches can in general be applied to already trained networks.

Another important aspect that has to be taken into account for uncertainty quantification in real life applications is the source and type of uncertainty. For real life applications, out-of-distribution detection forms the maybe most important challenge in order to avoid unexpected decisions of the network and to be aware of adversarial attacks. Especially since many motivations of uncertainty quantification are given by risk minimization, methods that deliver risk averse predictions are an important field to evaluate. Many works already demonstrated the capability of detecting out-of-distribution samples on several tasks and built a strong fundamental tool set for the deployment in real life applications [254], [241], [255], [56]. However, in real life, the tasks are much more difficult than finding out-of-distribution samples among data sets (e.g., MNIST or CIFAR data sets etc.) and the main challenge lies in comparing such approaches on several real-world data sets against each other. The work of Gustafsson et al. [56] forms a first important step towards an evaluation of methods that better suits the demands in real life applications. Interestingly, they found for their tests ensembles to outperform the considered Bayesian approaches. This indicates, that the multi-mode

evaluation given by ensembles is a powerful property for real life applications. Nevertheless Bayesian approaches have delivered strong results as well and furthermore come along with a strong theoretical foundation [84], [211], [6], [23]. As a way to go, the combination of efficient ensemble strategies and Bayesian approaches could combine the variability in the model parameters while still considering several modes for a prediction. Also, single deterministic approaches as the prior networks [32], [64], [44], [33] deliver comparable results while consuming significantly less computation power. However, this efficiency often comes along with the problem that separated sets of in- and out-of-distribution samples have to be available for the training process [33], [64]. In general, the development of new problem and loss formulations as for example given in [64] leads to a better understanding and description of the underlying problem and forms an important field of research.

IV. Uncertainty Measures and Quality

In Section III, we presented different methods for modeling and predicting different types of uncertainty in neural networks. In order to evaluate these approaches, measures have to be applied on the derived uncertainties. In the following, we present different measures for quantifying the different predicted types of uncertainty. In general, the correctness and trustworthiness of these uncertainties is not automatically given. In fact, there are several reasons why evaluating the quality of the uncertainty estimates is a challenging task.

- First, the quality of the uncertainty estimation depends on the underlying method for estimating uncertainty. This is exemplified in the work undertaken by Yao et al. [256], which shows that different approximates of Bayesian inference (e.g. Gaussian and Laplace approximates) result in different qualities of uncertainty estimates.

- Second, there is a lack of ground truth uncertainty estimates [31] and defining ground truth uncertainty estimates is challenging. For instance, if we define the ground truth uncertainty as the uncertainty across human subjects, we still have to answer questions as "How many subjects do we need?" or "How to choose the subjects?".

- Third, there is a lack of a unified quantitative evaluation metric [257]. To be more specific, the uncertainty is defined differently in different machine learning tasks such as classification, segmentation, and regression. For instance, prediction intervals or standard deviations are used to represent uncertainty in regression tasks, while entropy (and other related measures) are used to capture uncertainty in classification and segmentation tasks.

A. Evaluating Uncertainty in Classification Tasks

For classification tasks, the network's softmax output already represents a measure of confidence. But since the raw softmax output is neither very reliable [67] nor can it represent all sources of uncertainty [19], further approaches and corresponding measures were developed.

1) Measuring Data Uncertainty in Classification Tasks:
Consider a classification task with K different classes and a probability vector network output $p(x)$ for some input sample x. In the following p is used for simplification and p_k stands for the k-th entry in the vector. In general, the given prediction p represents a categorical distribution, i.e. it assigns a probability to each class to be the correct prediction. Since the prediction is not given as an explicit class but as a probability distribution, (un)certainty estimates can be directly derived from the prediction. In general this pointwise prediction can be seen as estimated data uncertainty [60]. However, as discussed in Section II, the model's estimation of the data uncertainty is affected by model uncertainty, which has to be taken into account separately. In order to evaluate the amount of predicted data uncertainty, one can for example apply the maximal class probability or the entropy measures:

$$\text{Maximal probability:} \qquad p_{\max} = \max \left\{ p_k \right\}_{k=1}^{K} \qquad (28)$$

$$\text{Entropy:} \qquad \text{H}(p) = - \sum_{k=1}^{K} p_k \log_2(p_k) \qquad (29)$$

The maximal probability represents a direct representation of certainty, while entropy describes the average level of information in a random variable. Even though a softmax output should represent the data uncertainty, one cannot tell from a single prediction how large the amount of model uncertainty is that affects this specific prediction as well.

2) Measuring Model Uncertainty in Classification Tasks:
As already discussed in Section III, a single softmax prediction is not a very reliable way for uncertainty quantification since it is often badly calibrated [19] and does not have any information about the certainty the model itself has on this specific output [19]. An (approximated) posterior distribution $p(\theta|D)$ on the learned model parameters can help to receive better uncertainty estimates. With such a posterior distribution, the softmax output itself becomes a random variable and one can evaluate its variation, i.e. uncertainty. For simplicity, we denote $p(y|\theta, x)$ also as p and it will be clear from context whether p depends on θ or not. The most common measures for this are the mutual information (MI), the expected Kullback-Leibler Divergence (EKL), and the predictive variance. Basically, all these measures compute the expected divergence between the (stochastic) softmax output and the expected softmax output

$$\hat{p} = \mathbb{E}_{\theta \sim p(\theta|D)} \left[p(y|x, \theta) \right] . \qquad (30)$$

The MI uses the entropy to measure the mutual dependence between two variables. In the described case, the difference between the information given in the expected softmax output and the expected information in the softmax output is compared, i.e.

$$\text{MI} \left(\theta, y|x, D \right) = \text{H} \left[\hat{p} \right] - \mathbb{E}_{\theta \sim p(\theta|D)} \text{H} \left[p(y|x, \theta) \right] . \qquad (31)$$

Smith and Gal [19] pointed out that the MI is minimal when the knowledge about model parameters does not increase the information in the final prediction. Therefore, the MI can be interpreted as a measure of model uncertainty.
The Kullback-Leibler divergence measures the divergence

between two given probability distributions. The EKL can be used to measure the (expected) divergence among the possible softmax outputs,

$$\mathbb{E}_{\theta \sim p(\theta|D)} \left[KL(\hat{p} \mid\mid p) \right] = \mathbb{E}_{\theta \sim p(\theta|D)} \left[\sum_{i=1}^{K} \hat{p}_i \log \left(\frac{\hat{p}_i}{p_i} \right) \right] , \qquad (32)$$

which can also be interpreted as a measure of uncertainty on the model's output and therefore represents the model uncertainty.
The predictive variance evaluates the variance on the (random) softmax outpus, i.e.

$$\sigma(p) = \mathbb{E}_{\theta \sim p(\theta|D)} \left[(p - \hat{p})^2 \right] . \qquad (33)$$

As described in Section III, an analytically described posterior distribution $p(\theta|D)$ is only given for a subset of the Bayesian methods. And even for an analytically described distribution, the propagation of the parameter uncertainty into the prediction is in almost all cases intractable and has to be approximated for example with Monte Carlo approximation. Similarly, ensemble methods collect predictions from M neural networks, and test-time data augmentation approaches receive M predictions from M different augmentations applied to the original input sample. For all these cases, we receive a set of M samples, $\left\{ p^i \right\}_{i=1}^{M}$, which can be used to approximate the intractable or even undefined underlying distribution. With these approximations, the measures defined in (31), (32), and (33) can be applied straight forward and only the expectation has to be replaced by average sums. For example, the expected softmax output becomes

$$\hat{p} \approx \frac{1}{M} \sum_{i=1}^{M} p^i .$$

For the expectations given in (31), (32), and (33), the expectation is approximated similarly.
3) Measuring Distributional Uncertainty in Classification Tasks:
Although these uncertainty measures are widely used to capture the variability among several predictions derived from Bayesian neural networks [60], ensemble methods [31], or test-time data augmentation methods [14], they cannot capture distributional shifts in the input data or out-of-distribution examples, which could lead to a biased inference process and a falsely stated confidence. If all predictors attribute a high probability mass to the same (false) class label, this induces a low variability among the estimates. Hence, the network seams to be certain about its prediction, while the uncertainty in the prediction itself (given by the softmax probabilities) is also evaluated to be low. To tackle this issue, several approaches described in Section III take the magnitude of the logits into account, since a larger logit indicates larger evidence for the corresponding class [44]. Thus, the methods either interpret the total sum of the (exponentials of) the logits as precision value of a Dirichlet distribution (see description of Dirichlet Priors in Section III-A) [32], [94], [64], or as a collection of evidence that is compared to a defined constant [44], [92]. One can also

derive a total class probability for each class individually by applying the sigmoid function to each logit [105]. Based on the class-wise total probabilities, OOD samples might easier be detected, since all classes can have low probability at the same time. Other methods deliver an explicit measure how well new data samples suit into the training data distribution. Based on this, they also give a measure that a sample will be predicted correctly [36].

4) Performance Measure on Complete Data Set:
While the measures described above measure the performance of individual predictions, others evaluate the usage of these measures on a set of samples. Measures of uncertainty can be used to separate between correctly and falsely classified samples or between in-domain and out-of-distribution samples [67]. For that, the samples are split into two sets, for example in-domain and out-of-distribution or correctly classified and falsely classified. The two most common approaches are the *Receiver Operating Characteristic* (ROC) curve and the *Precision-Recall* (PR) curve. Both methods generate curves based on different thresholds of the underlying measure. For each considered threshold, the ROC curve plots the true positive rate against the false positive rate[3], and the PR curve plots the precision against the recall[4]. While the ROC and PR curves give a visual idea of how well the underlying measures are suited to separate the two considered test cases, they do not give a qualitative measure. To reach this, the area under the curve (AUC) can be evaluated. Roughly speaking, the AUC gives a probability value that a randomly chosen positive sample leads to a higher measure than a randomly chosen negative example. For example, the maximum softmax values measure ranks of correctly classified examples higher than falsely classified examples. Hendrycks and Gimpel [67] showed for several application fields that correct predictions have in general a higher predicted certainty in the softmax value than false predictions. Especially for the evaluation of in-domain and out-of-distribution examples, the *Area Under Receiver Operating Curve* (AUROC) and the *Area Under Precision Recall Curce* (AUPRC) are commonly used [64], [32], [94]. The clear weakness of these evaluations is the fact that the performance is evaluated and the optimal threshold is computed based on a given test data set. A distribution shift from the test set distribution can ruin the whole performance and make the derived thresholds impractical.

B. Evaluating Uncertainty in Regression Tasks

1) Measuring Data Uncertainty in Regression Predictions:
In contrast to classification tasks, where the network typically outputs a probability distritution over the possible classes, regression tasks only predict a pointwise estimation without any hint of data uncertainty. As already described in Section

III, a common approach to overcome this is to let the network predict the parameters of a probability distribution, for example a mean vector and a standard deviation for a normally distributed uncertainty [31], [60]. Doing so, a measure of data uncertainty is directly given. The prediction of the standard deviation allows an analytical description that the (unknown) true value is within a specific region. The interval that covers the true value with a probability of α (under the assumption that the predicted distribution is correct) is given by

$$\left[\hat{y} - \frac{1}{2}\Phi^{-1}(\alpha)\cdot\sigma; \quad \hat{y} + \frac{1}{2}\Phi^{-1}(\alpha)\cdot\sigma\right] \qquad (34)$$

where Φ^{-1} is the quantile function, the inverse of the cumulative probability function. For a given probability value α the quantile function gives a boundary, such that $100 \cdot \alpha\%$ of a standard normal distribution's probability mass is on values smaller than $\Phi^{-1}(\alpha)$. Quantiles assume some probability distribution and interpret the given prediction as the expected value of the distribution.

In contrast to this, other approaches [259], [260] directly predict a so called prediction interval (PI)

$$PI(x) = [B_l, B_u] \qquad (35)$$

in which the prediction is assumed to lay. Such intervals induce an uncertainty as a uniform distribution without giving a concrete prediction. The certainty of such approaches can, as the name indicates, be directly measured by the size of the predicted interval. The *Mean Prediction Interval Width* (MPIW) can be used to evaluate the average certainty of the model [259], [260]. In order to evaluate the correctness of the predicted intervals the *Prediction Interval Coverage Probability* (PICP) can be applied [259], [260]. The PCIP represents the percentage of test predictions that fall into a prediction interval and is defined as

$$\text{PICP} = \frac{c}{n} , \qquad (36)$$

where n is the total number of predictions and c the number of ground truth values that are actually captured by the predicted intervals.

2) Measuring Model Uncertainty in Regression Predictions:

In Section II, it is described, that model uncertainty is mainly caused by the model's architecture, the training process, and underrepresented areas in the training data. Hence, there is no real difference in the causes and effects of model uncertainty between regression and classification tasks such that model uncertainty in regression tasks can be measured equivalently as already described for classification tasks, i.e. in most cases by approximating an average prediction and measuring the divergence among the single predictions [60].

C. Evaluating Uncertainty in Segmentation Tasks

The evaluation of uncertainties in segmentation tasks is very similar to the evaluation for classification problems. The uncertainty is estimated in segmentation tasks using approximates of Bayesian inference [1], [2], [4], [152], [261], [262], [263], [264] or test-time data augmentation techniques [249].

---

[3]The true positive rate is the number of samples, which are correctly predicted as positive divided by the total number of true samples. The false positive rate is the number of samples falsely predicted as positive divided by the total number of negative samples (see also [258])

[4]The precision is equal to the number of samples that are correctly classified as positive, divided by the total number of positive samples. The recall is equal to the number of samples correctly predicted as positive divided by the total number of positive samples (see also [258])

In the context of segmentation, the uncertainty in pixel wise segmentation is measured using confidence intervals [4], [152], the predictive variance [262], [264], the predictive entropy [2], [249], [261], [263] or the mutual information [1]. The uncertainty in structure (volume) estimation is obtained by averaging over all pixel-wise uncertainty estimates [264], [261]. The quality of volume uncertainties is assessed by evaluating the coefficient of variation, the average Dice score or the intersection over union [2], [249]. These metrics measure the agreement in area overlap between multiple estimates in a pair-wise fashion. Ideally, a false segmentation should result in an increase in pixel-wise and structure uncertainty. To evaluate whether this is the case, Nair et al. [1] evaluated the pixel-level true positive rate and false detection rate as well as the ROC curves for the retained pixels at different uncertainty thresholds. Similar to [1], McClure et al. [261] also analyzed the area under the ROC curve.

V. CALIBRATION

A predictor is called well-calibrated if the derived predictive confidence represents a good approximation of the actual probability of correctness [15]. Therefore, in order to make use of uncertainty quantification methods, one has to be sure that the network is well calibrated. Formally, for classification tasks a neural network f_θ is calibrated [265] if it holds that

$$\forall p \in [0,1]: \quad \sum_{i=1}^{N} \sum_{k=1}^{K} \frac{y_{i,k} \cdot \mathbb{I}\{f_\theta(x_i)_k = p\}}{\mathbb{I}\{f_\theta(x_i)_k = p\}} \xrightarrow{N\to\infty} p \ . \tag{37}$$

Here, $\mathbb{I}\{\cdot\}$ is the indicator function that is either 1 if the condition is true or 0 if it is false and $y_{i,k}$ is the k-th entry in the one-hot encoded groundtruth vector of a training sample (x_i, y_i). This formulation means that for example 30% of all predictions with a predictive confidence of 70% should actually be false. For regression tasks the calibration can be defined such that predicted confidence intervals should match the confidence intervals empirically computed from the data set [265], i.e.

$$\forall p \in [0,1]: \quad \sum_{i=1}^{N} \frac{\mathbb{I}\{y_i \in \text{conf}_p(f_\theta(x_i))\}}{N} \xrightarrow{N\to\infty} p, \tag{38}$$

where conf_p is the confidence interval that covers p percent of a distribution.

A DNN is called under-confident if the left hand side of (37) and (38) are larger than p. Equivalently, it is under-confident if the terms are smaller than p. The calibration property of a DNN can be visualized using a *reliability diagram*, as shown in Figure 8.

In general, calibration errors are caused by factors related to model uncertainty [15]. This is intuitively clear, since as discussed in Section II, data uncertainty represents the underlying uncertainty that an input x and a target y represent the same real world information. Following, correctly predicted data uncertainty would lead to a perfectly calibrated neural network. In practice, several works pointed out that deeper networks tend to be more overconfident than shallower ones [15], [266], [267].

Several methods for uncertainty estimation presented in Section III also improve the networks calibration [31], [20]. This is clear, since these methods quantify model and data uncertainty separately and aim at reducing the model uncertainty on the predictions. Besides the methods that improve the calibration by reducing the model uncertainty, a large and growing body of literature has investigated methods for explicitly reducing calibration errors. These methods are presented in the following, followed by measures to quantify the calibration error. It is important to note that these methods do not reduce the model uncertainty, but propagate the model uncertainty onto the representation of the data uncertainty. For example, if a binary classifier is overfitted and predicts all samples of a test set as class A with probability 1, while half of the test samples are actually class B, the recalibration methods might map the network output to 0.5 in order to have a reliable confidence. This probability of 0.5 is not equivalent to the data uncertainty but represents the model uncertainty propagated onto the predicted data uncertainty.

A. Calibration Methods

Calibration methods can be classified into three main groups according to the step when they are applied:

- *Regularization methods applied during the training phase* [268], [269], [11], [270], [271]
  These methods modify the objective, optimization and/or regularization procedure in order to build DNNs that are inherently calibrated.

- *Post-processing methods applied after the training process of the DNN* [15], [47]
  These methods require a held-out calibration data set to adjust the prediction scores for recalibration. They only work under the assumption that the distribution of the left-out validation set is equivalent to the distribution, on which inference is done. Hence, also the size of the validation data set can influence the calibration result.

- *Neural network uncertainty estimation methods*
  Approaches, as presented in Section III, that reduce the amount of model uncertainty on a neural network's confidence prediction, also lead to a better calibrated predictor. This is because the remaining predicted data uncertainty better represents the actual uncertainty on the prediction. Such methods are based for example on Bayesian methods [272], [209], [273], [274], [16] or deep ensembles [31], [275].

In the following, we present the three types of calibration methods in more detail.

1) Regularization Methods: Regularization methods for calibrating confidences manipulate the training of DNNs by modifying the objective function or by augmenting the training data set. The goal and idea of regularization methods is very similar to the methods presented in Section III-A where the methods mainly quantify model and data uncertainty separately within a single forward pass. However, the methods in Section III-A quantify the model and data uncertainty, while these calibration methods are regularized in order to
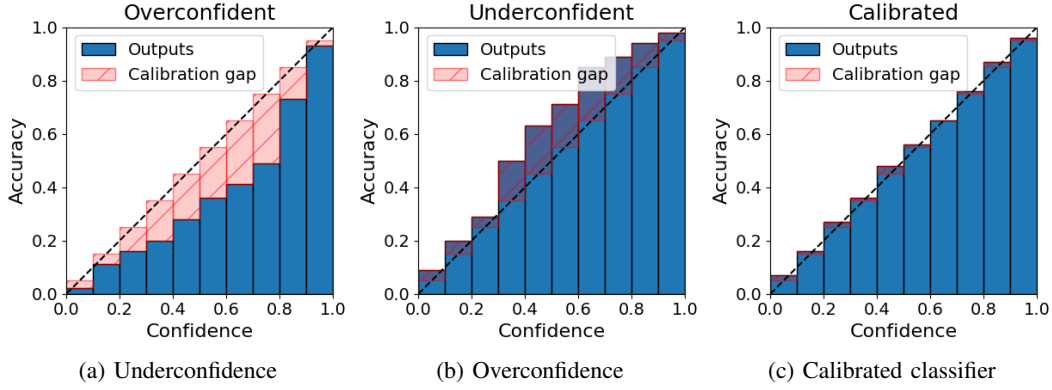
Fig. 8: (a) Reliability diagram showing an overconfident classifier: The bin-wise accuracy is smaller than the corresponding confidence. (b) Reliability diagram of an underconfident classifier: The bin-wise accuracy is larger than the corresponding confidence. (c) Reliability diagram of a well calibrated classifier: The confidence fits the actual accuracy for the single bins.

minimize the model uncertainty. Following, at inference, the model uncertainty cannot be obtained anymore. This is the main motivation for us to separate the approaches presented below from the approaches presented in Section III-A.

One popular regularization based calibration method is label smoothing [268]. For label smoothing, the labels of the training examples are modified by taking a small portion α of the true class' probability mass and assign it uniformly to the false classes. For hard, non-smoothed labels, the optimum cannot be reached in practice, as the gradient of the output with respect to the logit vector z,

$$\nabla_z \text{CE}(y, \hat{y}(z)) = \text{softmax}(z) - y$$
$$= \frac{\exp(z)}{\sum_{i=1}^{K} \exp(z_i)} - y , \qquad (39)$$

can only converge to zero with increasing distance between the true and false classes' logits. As a result, the logits of the correct class are much larger than the logits for the incorrect classes and the logits of the incorrect classes can be very different to each other. Label-smoothing avoids this and while it generally leads to a higher training loss, the calibration error decreases and the accuracy often increases as well [270].

Seo et al. [266] extended the idea of label smoothing and directly aimed at reducing the model uncertainty. For this, they sampled T forward passes of a stochastic neural network already at training time. Based on the T forward passes of a training sample (x_i, y_i), a normalized model variance α_i is derived as the mean of the Bhattacharyya coefficients [281] between the T individual predictions $\hat{y}_1, ..., \hat{y}_T$ and the average prediction $\bar{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}_t$,

$$\alpha_i = \frac{1}{T} \sum_{t=1}^{T} BC(\bar{y}_i, \hat{y}_t)$$
$$= \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} \sqrt{\bar{y}_{i,k} \cdot \hat{y}_{i,t,k}} . \qquad (40)$$

Based on this α_i, Seo et al. [266] introduced the variance-weighted confidence-integrated loss function that is a convex combination of two contradictive loss functions,

$$L^{\text{VWCI}}(\theta) = -\sum_{i=1}^{N} (1 - \alpha_i) L_{\text{GT}}^{(i)}(\theta) + \alpha_i L_{\text{U}}^{(i)}(\theta) , \qquad (41)$$

where $L_{\text{GT}}^{(i)}$ is the mean cross-entropy computed for the training sample x_i with given ground-truth y_i. L_{U} represents the mean KL-divergence between a uniform target probability vector and the computed prediction. The adaptive smoothing parameter α_i pushes predictions of training samples with high model uncertainty (given by high variances) towards a uniform distribution while increasing the prediction scores of samples with low model uncertainty. As a result, variances in the predictions of a single sample are reduced and the network can then be applied with a single forward pass at inference.

Pereyra et al. [269] combated the overconfidence issue by adding the negative entropy to the standard loss function and therefore a penalty that increases with the network's predicted confidence. This results in the entropy-based objective function L^H, which is defined as

$$L^H(\theta) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log \hat{y}_i - \alpha_i H(\hat{y}_i) , \qquad (42)$$

where $H(\hat{y}_i)$ is the entropy of the output and α_i a parameter that controls the strength of the confidence penalty and is computed equivalently as for the VWCI loss.

Instead of regularizing the training process by modifying the objective function, Thulasidasan et al. [278] regularized it by using a data-agnostic data augmentation technique named mixup [282]. In mixup training, the network is not only trained on the training data, but also on virtual training samples $(\tilde{x}, \tilde{y})$ generated by a convex combination of two random training pairs (x_i, y_i) and (x_j, y_j), i.e.

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \qquad (43)$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j . \qquad (44)$$

According to [278], the label smoothing resulting from mixup training can be viewed as a form of entropy-based regularization resulting in inherent calibration of networks trained
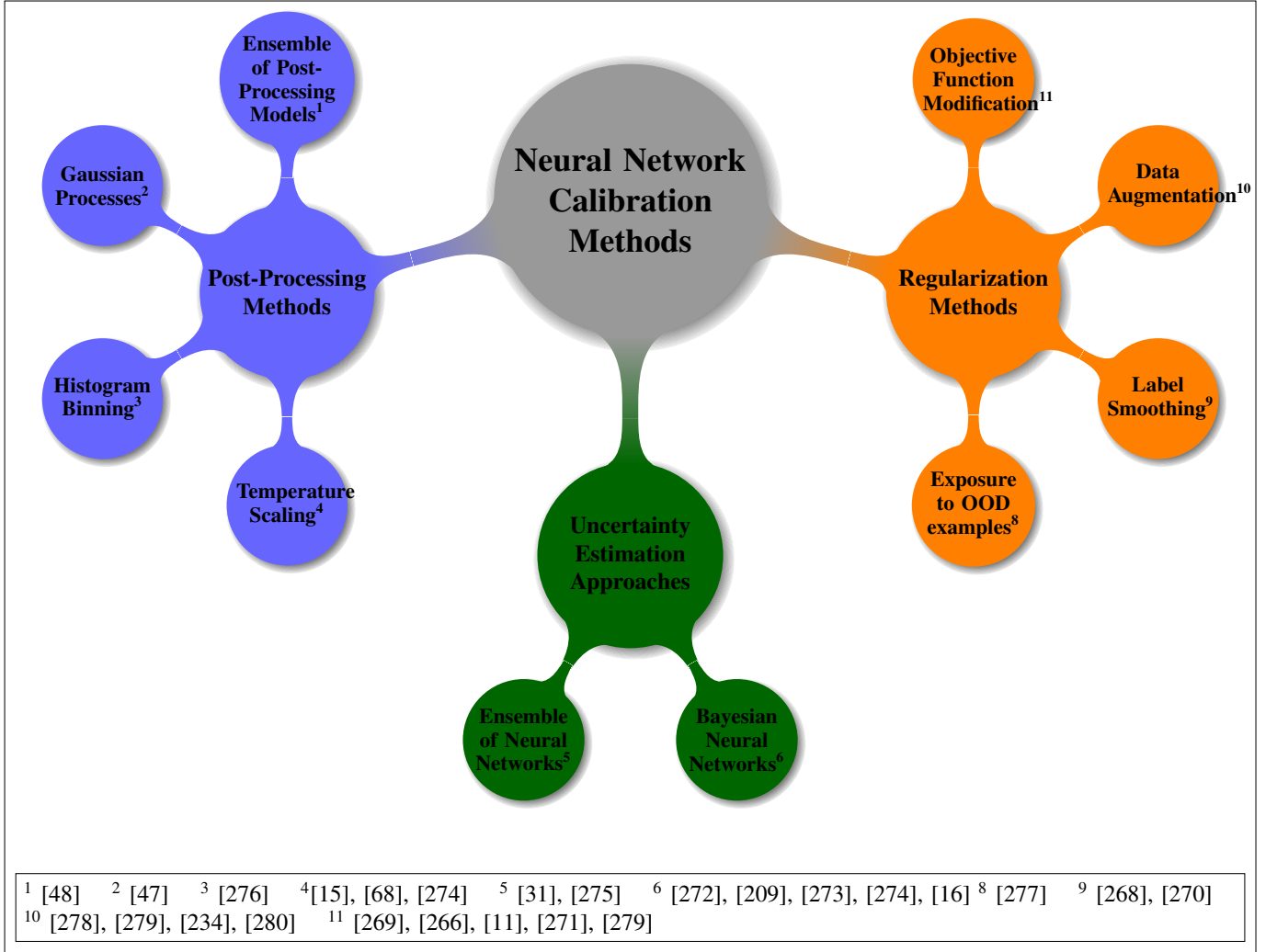
Fig. 9: Visualization of the different types of uncertainty calibration methods presented in this paper.

with mixup. Maroñas et al. [279] see mixup training among the most popular data augmentation regularization techniques due to its ability to improve the calibration as well as the accuracy. However, they argued that in mixup training the data uncertainty in mixed inputs affects the calibration and therefore mixup does not necessarily improve the calibration. They also underlined this claim empirically. Similarly, Rahaman and Thiery [234] experimentally showed that the distributional-shift induced by data augmentation techniques such as mixup training can negatively affect the confidence calibration. Based on this observation, Maroñas et al. [279] proposed a new objective function that explicitly takes the calibration performance on the unmixed input samples into account. Inspired by the expected calibration error (ECE, see Section V-B) Naeini et al. [283] measured the calibration performance on the unmixed samples for each batch b by the differentiable squared differences between the batch accuracy and the mean confidence on the batch samples. The total loss is given as a weighted combination of the original loss on mixed and unmixed samples and the calibration measure evaluated only on the unmixed samples:

$$L^{ECE}(\theta) = \frac{1}{B} \sum_{b \in B} L^b(\theta) + \beta ECE_b , \quad (45)$$

where $L^b(\theta)$ is the original unregularized loss using training and mixed samples included in batch b and β is a hyperparameter controlling the relative importance given to the batchwise expected calibration error ECE_b. By adding the batchwise calibration error for each batch $b \in B$ to the standard loss function, the miscalibration induced by mixup training is regularized.

In the context of data augmentation, Patel et al. [280] improved the calibration of uncertainty estimates by using on-manifold data augmentation. While mixup training combines training samples, on-manifold adversarial training generate out-of-domain samples using adversarial attack. They experimentally showed that on-manifold adversarial training outperforms mixup training in improving the calibration. Similar to [280], Hendrycks et al. [277] showed that exposing classifiers to out-of-distribution examples at training can help to improve the calibration.

2) Post-Processing Methods: Post-processing (or post-hoc) methods are applied after the training process and aim at learning a re-calibration function. For this, a subset of the training data is held-out during the training process and used as a calibration set. The re-calibration function is applied to the network's outputs (e.g. the logit vector) and yields an improved calibration learned on the left-out calibration set. Zhang et al. [48] discussed three requirements that should be satisfied by post-hoc calibration methods. They should

1) preserve the accuracy, i.e. should not affect the predictors performance.
2) be data efficient, i.e. only a small fraction of the training data set should be left out for the calibration.
3) be able to approximate the correct re-calibration map as long as there is enough data available for calibration.

Furthermore, they pointed out that none of the existing approaches fulfills all three requirements.

For classification tasks, the most basic but still very efficient way of post-hoc calibration is temperature scaling [15]. For temperature scaling, the temperature $T > 0$ of the softmax function

$$\text{softmax}(z_i) = \frac{\exp^{z_i/T}}{\sum_{j=1}^{K} \exp^{z_j/T}} \ , \qquad (46)$$

is optimized. For $T = 1$ the function remains the regular softmax function. For $T > 1$ the output changes such that its entropy increases, i.e. the predicted confidence decreases. For $T \in (0, 1)$ the entropy decreases and following, the predicted confidence increases. As already mentioned above, a perfect calibrated neural network outputs MAP estimates. Since the learned transformation can only affect the uncertainty, the log-likelihood based losses as cross-entropy do not have to be replaced by a special calibration loss. While the data efficiency and the preservation of the accuracy is given, the expressiveness of basic temperature scaling is limited [48]. To overcome this, Zhang et al. [48] investigated an ensemble of several temperature scaling models. Doing so, they achieved better calibrated predictions, while preserving the classification accuracy and improving the data efficiency and the expressive power. Kull et al. [284] were motivated by non-neural network calibration methods, where the calibration is performed classwise as a one-vs-all binary calibration. They showed that this approach can be interpreted as learning a linear transformation of the predicted log-likelihoods followed by a softmax function. This again is equivalent to train a dense layer on the log-probabilities and hence the method is also very easy to implement and apply. Obviously, the original predictions are not guaranteed to be preserved.

Analogous to temperature scaling for classification networks, Levi et al. [285] introduced standard deviation scaling (std-scaling) for regression networks. As the name indicates, the method is trained to rescale the predicted standard deviations of a given network. Equivalently to the motivation of optimizing temperature scaling with the cross-entropy loss, std-scaling can be trained using the Gaussian log-likelihood function as loss, which is in general also used for the training of regression networks, which also give a prediction for the data uncertainty.

Wenger et al. [47] proposed a Gaussian process (GP) based method, which can be used to calibrate any multi-class classifier that outputs confidence values and presented their methodology by calibrating neural networks. The main idea behind their work is to learn the calibration map by a Gaussian process that is trained on the networks confidence predictions and the corresponding ground-truths in the left out calibration set. For this approach, the preservation of the predictions is also not assured.

3) Calibration with Uncertainty Estimation Approaches: As already discussed above, removing the model uncertainty and receiving an accurate estimation of the data uncertainty leads to a well calibrated predictor. Following several works based on deep ensembles [31], [275] and BNNs, [272], [209], [204] also compared their performance to other methods based on the resulting calibration. Lakshminarayanan et al. [31] and Mehrtash et al. [275] reported an improved calibration by applying deep ensembles compared to single networks. However, Rahaman and Thiery[234] showed that for specific configurations as the usage of mixup-regularization, deep ensembles can even increase the calibration error. On the other side they showed that applying temperature scaling on the averaged predictions can give a significant improvement on the calibration.

For the Bayesian approaches, [204] showed that restricting the Bayesian approximation to the weights of the last fully connected layer of a DNN is already enough to improve the calibration significantly. Zhang et al. [273] and Laves et al. [274] showed that confidence estimates computed with MC dropout can be poorly calibrated. To overcome this, Zhang et al. [273] proposed structured dropout, which consists of dropping channel, blocks or layers, to promote model diversity and reduce calibration errors.

B. Evaluating Calibration Quality

Evaluating calibration consists of measuring the statistical consistency between the predictive distributions and the observations [286]. For classification tasks, several calibration measures are based on binning. For that, the predictions are ordered by the predicted confidence $\hat{p}_i$ and grouped into M bins $b_1, ..., b_M$. Following, the calibration of the single bins is evaluated by setting the average bin confidence

$$\text{conf}(b_m) = \frac{1}{|b_m|} \sum_{s \in b_m} \hat{p}_s \qquad (47)$$

in relation to the average bin accuracy

$$\text{acc}(b_m) = \frac{1}{|b_m|} \sum_{s \in b_m} \mathbb{1}(\hat{y}_s = y_s) \ , \qquad (48)$$

where $\hat{y}_s$, y_s and $\hat{p}_s$ refer to the predicted and true class label of a sample s. As noted in [15], confidences are well-calibrated when for each bin $\text{acc}(b_m) = \text{conf}(b_m)$. For a visual evaluation of a model's calibration, the reliability diagram introduced by [287] is widely used. For a reliability diagram, the $\text{conf}(b_m)$ is plotted against $\text{acc}(b_m)$. For a well-calibrated model, the plot should be close to the diagonal, as visualized

in Figure 8. The basic reliability diagram visualization does not distinguish between different classes. In order to do so and hence to improve the interpretability of the calibration error, Vaicenavicius et al. [286] used an alternative visualization named multidimensional reliability diagram.

For a quantitative evaluation of a model's calibration, different calibration measures can be considered.
The *Expected Calibration Error* (ECE) is a widely used binning based calibration measure [283], [15], [274], [275], [278], [47]. For the ECE, M equally-spaced bins $b_1, ..., b_M$ are considered, where b_m denotes the set of indices of samples whose confidences fall into the interval $I_m =]\frac{m-1}{M}, \frac{m}{M}]$. The ECE is then computed as the weighted average of the bin-wise calibration errors, i.e.

$$\text{ECE} = \sum_{m=1}^{M} \frac{|b_m|}{N} |\text{acc}(b_m) - \text{conf}(b_m)| . \qquad (49)$$

For the ECE, only the predicted confidence score (top-label) is considered. In contrast to this, the *Static Calibration Error* (SCE) [288], [289] considers the predictions of all classes (all-labels). For each class, the SCE computes the calibration error within the bins and then averages across all the bins, i.e.

$$\text{SCE} = \frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \frac{|b_{m_k}|}{N} |\text{conf}(b_{m_k}) - \text{acc}(b_{m_k})| . \qquad (50)$$

Here $conf(b_{m_k})$ and $acc(b_{m_k})$ are the confidence and accuracy of bin b_m for class label k, respectively. Nixon et al. [288] empirically showed that all-labels calibration measures such as the SCE are more effective in assessing the calibration error than the top-label calibration measures as the ECE.

In contrast to the ECE and SCE, which group predictions into M equally-spaced bins (what in general leads to different numbers of evaluation samples per bin), the adaptive calibration error [288], [289] adaptively groups predictions into R bins with different width but equal number of predictions. With this adaptive bin size, the *adaptive Expected Calibration Error* (aECE)

$$\text{aECE} = \frac{1}{R} \sum_{r=1}^{R} |\text{conf}(b_r) - \text{acc}(b_r)| , \qquad (51)$$

and the *adaptive Static Calibration Error* (aSCE)

$$\text{aSCE} = \frac{1}{KR} \sum_{k=1}^{K} \sum_{r=1}^{R} |\text{conf}(b_{r_k}) - \text{acc}(b_{r_k})| \qquad (52)$$

are defined as extensions of the ECE and the SCE.
As has been empirically shown in [280] and [288], the adaptive binning calibration measures aECE and aSCE are more robust to the number of bins than the corresponding equal-width binning calibration measures ECE and SCE.

It is important to make clear that in a multi-class setting, the calibration measures can suffer from imbalance in the test data. Even when then calibration is computed classwise, the computed errors are weighted by the number of samples in the classes. Following, larger classes can shadow the bad calibration on small classes, comparable to accuracy values in classification tasks [290].

VI. DATA SETS AND BASELINES

In this section, we collect commonly used tasks and data sets for evaluating uncertainty estimation among existing works. Besides, a variety of baseline approaches commonly used as comparison against the methods proposed by the researchers are also presented. By providing a review on the relevant information of these experiments, we hope that both researchers and practitioners can benefit from it. While the former can gain a basic understanding of recent benchmarks tasks, data sets and baselines so that they can design appropriate experiments to validate their ideas more efficiently, the latter might use the provided information to select more relevant approaches to start based on a concise overview on the tasks and data sets on which the approach has been validated.

In the following, we will introduce the data sets and baselines summarized in table IV according to the taxonomy used throughout this review.

The structure of the table is designed to organize the main contents of this section concisely, hoping to provide a clear overview of the relevant works. We group the approaches of each category into one of four blocks and extract the most commonly used tasks, data sets and provided baselines for each column respectively. The corresponding literature is listed at the bottom of each block to facilitate lookup. Note that we focus on methodological comparison here, but not the choice of architecture for different methods which has an impact on performance as well. Due to the space limitation and visual density, we only show the most important elements (task, data set, baselines) ranked according to the frequency of use in the literature we have researched.

The main results are as follows. One of the most frequent tasks for evaluating uncertainty estimation methods are regression tasks, where samples close and far away from the training distribution are studied. Furthermore, the calibration of uncertainty estimates in the case of classification problems is very often investigated. Further noteworthy tasks are out-of-distribution (OOD) detection and robustness against adversarial attacks. In the medical domain, calibration of semantic segmentation results is the predominant use case.

The choice of data sets is mostly consistent among all reviewed works. For regression, toy data sets are employed for visualization of uncertainty intervals while the UCI data sets are studied in light of (negative) log-likelihood comparison. The most common data sets for calibration and OOD detection are MNIST, CIFAR10 and 100 as well as SVHN while ImageNet and its tiny variant are also studied frequently. These form distinct pairs when OOD detection is studied where models trained on CIFAR variants are evaluated on SVHN and visa versa while MNIST is paired with variants of itself like notMNIST and FashionMNIST. Classification data sets are also commonly distorted and corrupted to study the effects on calibration, blurring the line between OOD detection and adversarial attacks.

Finally, the most commonly used baselines by far are Monte Carlo (MC) Dropout and deep ensembles while the softmax output of deterministic models is almost always employed as a kind of surrogate baseline. It is interesting to note that

TABLE IV: Overview of frequently compared benchmark approaches, tasks and their data sets among existing works organized according to the taxonomy of this paper.

	Tasks	Task index: Data sets	Baselines
Bayesian Neural Networks	1. Regression[1,3,4,7,8,11,12,15-17] 2. Calibration[6,10,13,14] 3. OOD Detection[4,6,8,10,12,13] 4. Adversarial Attacks[4,8,12] 5. Active Learning[7,12,14] 6. Continual Learning[10] 7. Reinforcement Learning (Intrinsic Motivation, Contextual Bandits)[1,14,15]	1: UCI 2, 3, 4: (not)MNIST, CIFAR10/100, SHVN, ImageNet 5: UCI 6: Permuted MNIST	Softmax[44], MCdropout[1], DeepEnsemble[2], BBB[3], NormalizingFlow[4], PBP[7], SWAG[6], KFAC[8], DVI[11] HMC[9], VOGN[10], INF[12]

[1][20] [3][30] [4][149] [5][75] [6][213] [7][129] [8][63] [9][80] [10][150] [11][143] [12][84] [13][235] [14][145] [15][135] [16][146] [17][272]

	Tasks	Task index: Data sets	Baselines
Ensembles	1. Regression[2,20] 2. Calibration[2,20,24-32] 3. OOD Detection[2,20,25,27-30,32-34] 4. Active Learning[31]	1: Toy[7], UCI 2, 3: Toy, (not)MNIST, SVHN, LSUN, CIFAR10/100, (Tiny)ImageNet, Diabetic Retinopathy 4: MNIST	Softmax[44], MFVI[5], SGLD[55], MCdropout[1], DeepEnsemble[2], BBB[3], PBP[7], NormalizingFlow[4], TemperatureScaling[38,54]

[2][31] [20][90] [24][234] [25][85] [26][86] [27][94] [28][39] [29][235] [30][40] [31][240] [32][13] [33][241] [34][246]

	Tasks	Task index: Data sets	Baselines
Single Deterministic Models	1. Regression[21-23] 2. Calibration[21-23,39,40,41,43] 3. OOD Detection[21,22,38,39,41-53] 4. Adversarial Attacks[21,41,48]	1. Toy[7], UCI, NYU Depth 2, 3: (E/Fashion/not)MNIST, Toy, CIFAR10/100, SVHN, LSUN, (Tiny)ImageNet, IMDB, Diabetic Retinopathy, Omniglot 4: MNIST, CIFAR10, NYU Depth, Omniglot	Softmax[44], GAN[27], Dirichlet[48], BBB[3], MCdropout[1], DeepEnsemble[2,57], Mahalanobis[56], TemperatureScaling[38,54] NormalizingFlow[4]

[21][101] [22][103] [23][104] [38][68] [39][143] [40][98] [41][43] [42][96] [43][95] [44][67] [45][64] [46][92] [47][72] [48][44] [49][94] [50][35] [51][36] [52][46] [53][71]

	Tasks	Task index: Data sets	Baselines
Test-Time Data Augmentation	1. Semantic Segmentation[36, 37] 2. Calibration[35] 3. OOD Detection[35-37]	1, 2, 3: Medical data, Diabetic Retinopathy	Softmax[44], MCdropout[1]

[35][14] [36][248] [37][249]

[54][15] [55][81] [56][291] [57][259]

inside each approach–BNNs, Ensembles, Single Deterministic Models and Input Augmentation–some baselines are preferred over others. BNNs are most frequently compared against variational inference methods like Bayes' by Backprop (BBB) or Probabilistic Backpropagation (PBP) while for Single Deterministic Models it is more common to compare them against distance-based methods in the case of OOD detection. Overall, BNN methods show a more diverse set of tasks considered while being less frequently evaluated on large data sets like ImageNet.

To further facilitate access for practitioners, we provide web-links to the authors' official implementations (marked by a star) of all common baselines as identified in the baselines column. Where no official implementation is provided, we instead link to the highest ranked implementations found on GitHub at the time of this survey. The list can be also found within our GitHub repository on available implementations[5]. The relevant baselines are Softmax[*] (TensorFlow, PyTorch), MCdropout (TensorFlow[*]; PyTorch: 1, 2), DeepEnsembles (TensorFlow: 1, 2, 3; PyTorch: 1, 2), BBB (PyTorch: 1, 2, 3,

4), NormalizingFlow (TensorFlow, PyTorch), PBP, SWAG (1[*], 2), KFAC (PyTorch: 1, 2, 3; TensorFlow), DVI (TensorFlow[*], PyTorch), HMC, VOGN[*], INF[*], MFVI, SGLD, TemperatureScaling (1[*], 2, 3), GAN[*], Dirichlet and Mahalanobis[*].

VII. APPLICATIONS OF UNCERTAINTY ESTIMATES

From a practical point of view, the main motivation for quantifying uncertainties in DNNs is to be able to classify the received predictions and to make more confident decisions. This section gives a brief overview and examples of the aforementioned motivations. In the first part, we discuss how uncertainty is used within active learning and reinforcement learning. Subsequently, we discuss the interest of the communities working on domain fields like medical image analysis, robotics, and earth observation. These application fields are used representatively for the large number of domains where the uncertainty quantification plays an important role. The challenges and concepts could (and should) be transferred to any application domain of interest.

1) Active Learning: The process of collecting labeled data for supervised training of a DNN can be laborious, time-consuming, and costly. To reduce the annotation effort, the

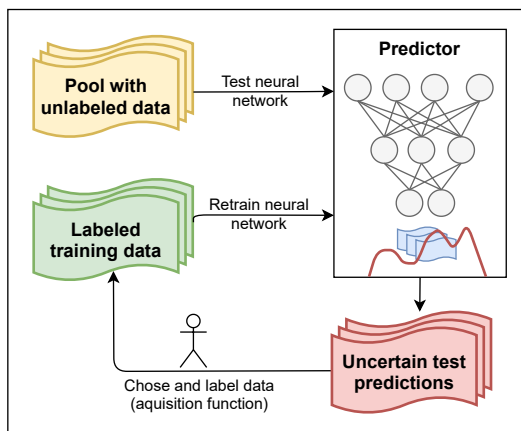[5]https://github.com/JakobCode/UncertaintyInNeuralNetworks_Resources

Fig. 10: The active learning framework: The acquisition function evaluates the uncertainties on the network's test predictions in order to select unlabelled data. The selected data are labelled and added to the pool of labelled data, which is used to train and improve the performance of the predictor.
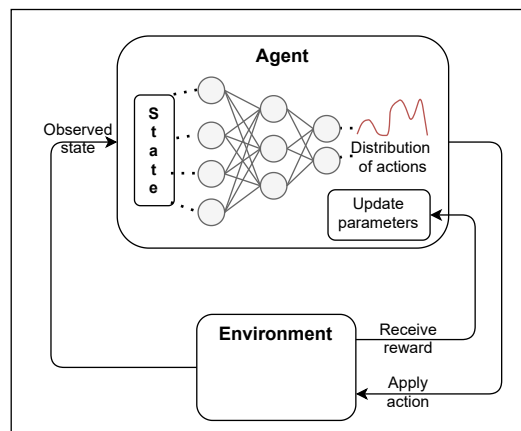

Fig. 11: The reinforcement learning framework: The agent interacts with the environment by executing a specific action influencing the next state of the agent. The agent observes a reward representing the cost associated with the executed action. The agent chooses actions based on a policy learned by a deep neural network. However, the predicted uncertainty associated with the action predicted by the deep neural network can help the agent to decide weather to execute the predicted action or not.

active learning framework shown in Figure 10 trains the DNN sequentially on different labelled data sets increasing in size over time [292]. In particular, given a small labelled data set and a large unlabeled data set, a deep neural network trained in the setting of active learning learns from the small labeled data set and decides based on the acquisition function, which samples to select from the pool of unlabeled data. The selected data are added to the training data set and a new DNN is trained on the updated training data set. This process is then repeated with the training set increasing in size over time. Uncertainty sampling is one most popular criteria used in acquisition functions [293] where predictive uncertainty determines which training samples have the highest uncertainty and should be labelled next. Uncertainty based active learning strategies for deep learning applications have been successfully used in several works [23], [24], [294], [25], [26].

2) Reinforcement Learning: The general framework of deep reinforcement learning is shown in Figure 11. In the context of reinforcement learning, uncertainty estimates can be used to solve the exploration-exploitation dilemma. It says that uncertainty estimates can be used to effectively balance the exploration of unknown environments with the exploitation of existing knowledge extracted from known environments. For example, if a robot interacts with an unknown environment, the robot can safely avoid catastrophic failures by reasoning about its uncertainty. To estimate the uncertainty in this framework, Huang et al. [27] used an ensemble of bootstrapped models (models trained on different data sets sampled with replacement from the original data set), while Gal and Ghahramani [20] approximated Bayesian inference via dropout sampling. Inspired by [20] and [27], Kahn et al. [28] and Lötjens et al. [29] used a mixture of deep Bayesian networks performing dropout sampling on an ensemble of bootstrapped models. For further reading, Ghavamzadeh et al. [295] presented a survey of Bayesian reinforcement learning.

A. Uncertainty in Real-World Applications

With increasing usage of deep learning approaches within many different fields, quantifying and handling uncertainties has become more and more important. On one hand, uncertainty quantification plays an important role in risk minimization, which is needed in many application fields. On the other hand, many fields offer only challenging data sources, which are hard to control and verify. This makes the generation of trust-worthy ground truth a very challenging task. In the following, three different fields where uncertainty plays an important role are presented, namely Autonomous Driving, medical image analysis, and earth observation.

1) Medical Analysis: Since the size, shape, and location of many diseases vary largely across patients, the estimation of the predictive uncertainty is crucial in analyzing medical images in applications such as lesion detection [1], [3], lung node segmentation [296], brain tumor segmentation [152], [248], [249], [2], [261], parasite segmentation in images of liver stage malaria [262], recognition of abnormalities on chest radiographs [297], and age estimation [6]. Here, uncertainty estimates in particular improve the interpretability of decisions of DNNs [298]. They are essential to understand the reliability of segmentation results, to detect false segmented areas and to guide human experts in the task of refinement [249]. Well-calibrated and reliable uncertainty estimates allow clinical experts to properly judge whether an automated diagnosis can be trusted [298]. Uncertainty was estimated in medical image segmentation based on Monte Carlo dropout [152], [296], [1], [2], [3], [263], [4], [5], [6], spike-and-slab dropout [261], and spatial dropout [262]. Wang et al. [248], [249] used test time data augmentation to estimate the data-dependent uncertainty in medical image segmentation.

2) Robotics: Robots are active agents that perceive, decide, plan, and act in the real-world – all based on their incomplete knowledge about the world. As a result, mistakes of the robots not only cause failures of their own mission, but can endanger human lives, e.g. in case of surgical robotics, self-driving cars, space robotics, etc. Hence, the robotics application of deep learning poses unique research challenges that significantly differ from those often addressed in computer vision and other off-line settings [299]. For example, the assumption that the testing condition come from the same distribution as training is often invalid in many settings of robotics, resulting in deterioration of the performance of DNNs in uncontrolled and detrimental conditions. This raises the questions how we can quantify the uncertainty in a DNN's predictions in order to avoid catastrophic failures. Answering such questions are important in robotics, as it might be a lofty goal to expect data-driven approaches (in many aspects from control to perception) to always be accurate. Instead, reasoning about uncertainty can help in leveraging the recent advances in deep learning for robotics.

Reasoning about uncertainties and the use of probabilistic representations, as oppose to relying on a single, most-likely estimate, have been central to many domains of robotics research, even before the advent of deep learning [300]. In robot perception, several uncertainty-aware methods have been proposed in the past, starting from localization methods [301], [302], [303] to simultaneous localization and mapping (SLAM) frameworks [304], [305], [306], [307]. As a result, many probabilistic methods such as factor graphs [308], [309] are now the work-horse of advanced consumer products such as robotic vacuum cleaners and unmanned aerial vehicles. In case of planning and control, estimation problems are widely treated as Bayesian sequential learning problems, and sequential decision making frameworks such as POMDPs [310], [311] assume a probabilistic treatment of the underlying planning problems. With probabilistic representations, many reinforcement learning algorithms are backed up by stability guarantees for safe interactions in the real-world [312], [313], [314]. Lastly, there have been also several advances starting from reasoning (semantics [315] to joint reasoning with geometry), embodiment (e.g. active perception [316]) to learning (e.g. active learning [317], [318], [319] and identifying unknown objects [320], [321], [322]).

Similarly, with the advent of deep learning, many researchers proposed new methods to quantify the uncertainty in deep learning as well as on how to further exploit such information. As oppose to many generic approaches, we summarize task-specific methods and their application in practice as followings. Notably, [323] proposed to perform novelty detection using auto-encoders, where the reconstructed outputs of auto-encoders was used to decide how much one can trust the network's predictions. Peretroukhin et al. [324] developed a SO(3) representation and uncertainty estimation framework for the problem of rotational learning problems with uncertainty. [325], [28], [326], [327] demonstrated uncertainty-aware, real world application of a reinforcement learning algorithm for robotics, while [328], [329] proposed to leverage spatial information, on top of MC-dropout. [207], [330], [331] developed

deep learning based localization systems along with uncertainty estimates. Other approaches also learn on the robots' past experiences of failures or detect inconsistencies of the predictors [332], [333]. In summary, the robotics community has been both, the users and the developers of the uncertainty estimation frameworks targeted to a specific problems.

Yet, robotics pose several unique challenges to uncertainty estimation methods for DNNs. These are for example, (i) how to limit the computational burden and build real-time capable methods that can be executed on the robots with limited computational capacities (e.g. aerial, space robots, etc); (ii) how to leverage spatial and temporal information, as robots sense sequentially instead of having a batch of training data for uncertainty estimates; (iii) whether robots can select the most uncertainty samples and update its learner online; (iv) Whether robots can purposefully manipulate the scene when uncertain. Most of these challenges arise due to the properties of robots that they are physically situated systems.

3) Earth Observation(EO): Earth Observation (EO) systems are increasingly used to make critical decisions related to urban planning [334], resource management [335], disaster response [336], and many more. Right now, there are hundreds of EO satellites in space, owned by different space agencies and private companies. Figure 12 shows the satellites owned by the European Space Agency (ESA). Like in many other domains, deep learning has shown great initial success in the field of EO over the past few years [337]. These early successes consisted of taking the latest developments of deep learning in computer vision and applying them to small curated earth observation data sets [337]. At the same time, the underlying data is very challenging. Even though the amount of data is huge, so is the variability in the data. This variability is caused by different sensor types, spatial changes (e.g. different regions and resolutions), and temporal changes (e.g. changing light conditions, weather conditions, seasons). Besides the challenge of efficient uncertainty quantification methods for such large amounts of data, several other challenges that can be tackled with uncertainty quantification exist in the field of EO. All in all, the sensitivity of many EO applications together with the nature of EO systems and the challenging EO data make the quantification of uncertainties very important in this field. Despite hundreds of publications in the last years on DL for EO, the range of literature on measuring uncertainties of these systems is relatively small.

Furthermore, due to the large variation in the data, a data sample received at test time is often not covered by the training data distribution. For example while preparing training data for a local climate zone classification, the human experts might be presented only with images where there is no obstruction and structures are clearly visible. When a model which is trained on this data set is deployed in real world, it might see the images with clouds obstructing the structures or snow giving them a completely different look. Also, the classes in EO data can have a very wide distribution. For example, there are millions of types of houses in the world and no training data can contain the examples for all of them. The question is where the OOD detector will draw the line and declare the following houses as OOD. Hence, OOD detection is important

in earth observation and uncertainty measurements play an important part in this [22].

Another common task in EO, where uncertainties can play an important role, is the data fusion. Optical images normally contain only a few channels like RGB. In contrast to this, EO data can contain optical images with up to hundreds of channels, and a variety of different sensors with different spatial, temporal, and semantic properties. Fusing the information from these different sources and channels propagates the uncertainties from different sources onto the prediction. The challenge lies in developing methods that do not only quantify uncertainties but also the amount of contribution from different channels individually and which learn to focus on the trustworthy data source for a given sample [338].

Unlike normal computer vision scenarios where the image acquisition equipment is quite near to the subject, the EO satellites are hundreds of kilometers away from the subject. The sensitivity of sensors, the atmospheric absorption properties, and surface reflectance properties all contribute to uncertainties in the acquired data. Integrating the knowledge of physical EO systems, which also contain information about uncertainty models in those systems, is another major open issue. However, for several applications in EO, measuring uncertainties is not only something good to have but rather an important requirement of the field. E.g., the geo-variables derived from EO data may be assimilated into process models (ocean, hydrological, weather, climate, etc) and the assimilation requires the probability distribution of the estimated variables.

VIII. CONCLUSION AND OUTLOOK

A. Conclusion - How well do the current uncertainty quantification methods work for real world applications?

Even though many advances on uncertainty quantification in neural networks have been made over the last years, their adoption in practical mission- and safety-critical applications is still limited. There are several reasons for this, which are discussed one-by-one as follows:

- **Missing Validation of Existing Methods over Real-World Problems**
  Although DNNs have become the defacto standard in solving numerous computer vision and medical image processing tasks, the majority of existing models are not able to appropriately quantify uncertainty that is inherent to their inferences particularly in real world applications. This is primarily because the baseline models are mostly developed using standard data sets such as Cifar10/100, ImageNet, or well known regression data sets that are specific to a particular use case and are therefore not readily applicable to complex real-world environments, as for example low resolutional satellite data or other data sources affected by noise. Although many researchers from other fields apply uncertainty quantification in their field [21], [10], [8], a broad and structured evaluation of existing methods based on different real world applications is not available yet. Works like [56] already built first steps towards a real life evaluation.

- **Lack of Standardized Evaluation Protocol**
  Existing methods for evaluating the estimated uncertainty are better suited to compare uncertainty quantification methods based on measurable quantities such as the calibration [340] or the performance on out-of-distribution detection [32]. As described in Section VI, these tests are performed on standardized sets within the machine learning community. Furthermore, the details of these experiments might differ in the experimental setting from paper to paper [214]. However, a clear standardized protocol of tests that should be performed on uncertainty quantification methods is still not available. For researchers from other domains it is difficult to directly find state of the art methods for the field they are interested in, not to speak of the hard decision on which sub-field of uncertainty quantification to focus. This makes the direct comparison of the latest approaches difficult and also limits the acceptance and adoption of current existing methods for uncertainty quantification.

- **Inability to Evaluate Uncertainty Associated to a Single Decision**
  Existing measures for evaluating the estimated uncertainty (e.g., the expected calibration error) are based on the whole testing data set. This means, that equivalent to classification tasks on unbalanced data sets, the uncertainty associated with single samples or small groups of samples may potentially get biased towards the performance on the rest of the data set. But for practical applications, assessing the reliability of a predicted confidence would give much more possibilities than an aggregated reliability based on some testing data, which are independent from the current situation [341]. Especially for mission- and safety-critical applications, pointwise evaluation measures could be of paramount importance and hence such evaluation approaches are very desirable.

- **Lack of Ground Truth Uncertainties**
  Current methods are empirically evaluated and the performance is underlined by reasonable and explainable values of uncertainty. A ground truth uncertainty that could be used for validation is in general not available. Additionally, even though existing methods are calibrated on given data sets, one cannot simply transfer these results to any other data set since one has to be aware of shifts in the data distribution and that many fields can only cover a tiny portion of the actual data environment. In application fields as EO, the preparation of a huge amount of training data is hard and expensive and hence synthetic data can be used to train a model. For this artificial data, artificial uncertainties in labels and data should be taken into account to receive a better understanding of the uncertainty quantification performance. The gap between the real and synthetic data, or estimated and real uncertainty further limits the adoption of currently existing methods for uncertainty quantification.
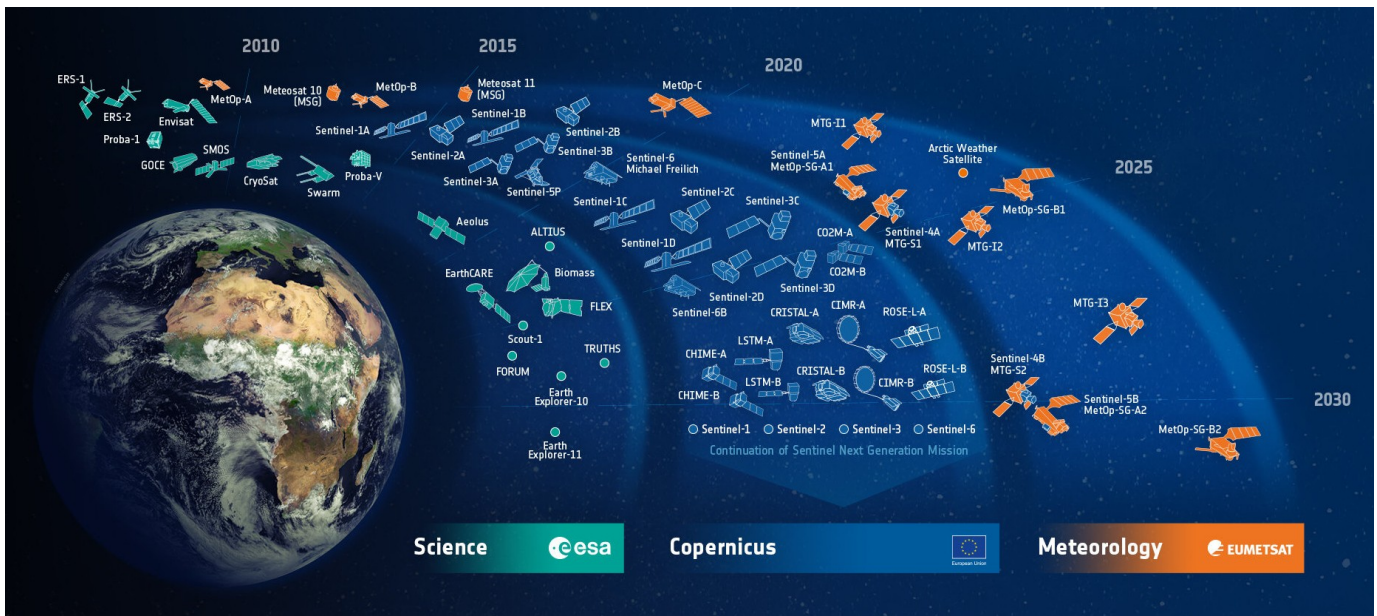
- **Explainability Issue:**

Fig. 12: European Space Agency (ESA) Developed Earth Observation Missions [339].

Existing methods of neural network uncertainty quantification deliver predictions of certainty without any clue about what causes possible uncertainties. Even though those certainty values often look *reasonable* to a human observer, one does not know whether the uncertainties are actually predicted based on the same observations the human observer made. But without being sure about the reasons and motivations of single uncertainty estimations, a proper transfer from one data set to another, and even only a domain shift, are much harder to realize with a guaranteed performance. Regarding safety critical real life applications, the lack of explainability makes the application of the available methods significantly harder. Besides the explainability of neural networks decisions, existing methods for uncertainty quantification are not well understood on a higher level. For instance, explaining the behavior of single deterministic approaches, ensembles or Bayesian methods is a current direction of research and remains difficult to grasp in every detail [227]. It is, however, crucial to understand how those methods operate and capture uncertainty to identify pathways for refinement, detect and characterize uncertainty, failures and important shortcomings [227].

B. Outlook

- **Generic Evaluation Framework**
  As already discussed above, there are still problems regarding the evaluation of uncertainty methods, as the lack of 'ground truth' uncertainties, the inability to test on single instances, and standardized benchmarking protocols, etc. To cope with such issues, the provision of an evaluation protocol containing various concrete baseline data sets and evaluation metrics that cover all types of uncertainty would undoubtedly help to boost research in uncertainty quantification. Also, the evaluation with

regard to risk-averse and worst case scenarios should be considered there. This means, that uncertainty predictions with a very high predicted uncertainty should never fail, as for example for a prediction of a red or green traffic light. Such a general protocol would enable researchers to easily compare different types of methods against an established benchmark as well as on real world data sets. The adoption of such a standard evaluation protocol should be encouraged by conferences and journals.

- **Expert & Systematic Comparison of Baselines**
  A broad and structured comparison of existing methods for uncertainty estimation on real world applications is not available yet. An evaluation on real world data is even not standard in current machine learning research papers. As a result, given a specific application, it remains unclear which method for uncertainty estimation performs best and whether the latest methods outperform older methods also on real world examples. This is also partly caused by the fact, that researchers from other domains that use uncertainty quantification methods, in general present successful applications of single approaches on a specific problem or a data set by hand. Considering this, there are several points that could be adopted for a better comparison within the different research domains. For instance, domain experts should also compare different approaches against each other and present the weaknesses of single approaches in this domain. Similarly, for a better comparison among several domains, a collection of all the works in the different real world domains could be collected and exchanged on a central platform. Such a platform might also help machine learning researchers in providing an additional source of challenges in the real world and would pave way to broadly highlight weaknesses in the current state of the art approaches. Google's repository on baselines in uncertainties in neural

networks [340]⁶ could be such a platform and a step towards achieving this goal.

- **Uncertainty Ground Truths**

    It remains difficult to validate existing methods due to the lack of uncertainty ground truths. An actual uncertainty ground truth on which methods can be compared in an ImageNet like manner would make the evaluation of predictions on single samples possible. To reach this, the evaluation of the data generation process and occurring sources of uncertainty, as for example the labeling process, might be investigated in more detail.

- **Explainability and Physical Models**

    Knowing the actual reasons for a false high certainty or a low certainty makes it much easier to engineer the methods for real life applications, which again increases the trust of people into such methods. Recently, Antorán et al. [342] claimed to have published the first work on explainable uncertainty estimation. Uncertainty estimations, in general, form an important step towards explainable artificial intelligence. Explainable uncertainty estimations would give an even deeper understanding of the decision process of a neural network, which, in practical deployment of DNNs, shall incorporate the desired ability to be risk averse while staying applicable in real world (especially safety critical applications). Also, the possibility of improving explainability with physically based arguments offers great potential. While DNNs are very flexible and efficient, they do not directly embed the domain specific expert knowledge that is mostly available and can often be described by mathematical or physical models, as for example earth system science problems [343]. Such physic guided models offer a variety of possibilities to include explicit knowledge as well as practical uncertainty representations into a deep learning framework [344], [345].

REFERENCES

[1] T. Nair, D. Precup, D. L. Arnold, and T. Arbel, "Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation," *Medical image analysis*, vol. 59, p. 101557, 2020.

[2] A. G. Roy, S. Conjeti, N. Navab, C. Wachinger, A. D. N. Initiative *et al.*, "Bayesian quicknat: Model uncertainty in deep whole-brain segmentation for structure-wise quality control," *NeuroImage*, vol. 195, pp. 11–22, 2019.

[3] P. Seeböck, J. I. Orlando, T. Schlegl, S. M. Waldstein, H. Bogunović, S. Klimscha, G. Langs, and U. Schmidt-Erfurth, "Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal oct," *IEEE transactions on medical imaging*, vol. 39, no. 1, pp. 87–98, 2019.

[4] T. LaBonte, C. Martinez, and S. A. Roberts, "We know where we don't know: 3d bayesian cnns for credible geometric uncertainty," *arXiv preprint arXiv:1910.10793*, 2019.

[5] J. C. Reinhold, Y. He, S. Han, Y. Chen, D. Gao, J. Lee, J. L. Prince, and A. Carass, "Validating uncertainty in medical image translation," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2020, pp. 95–98.

[6] S. Eggenreich, C. Payer, M. Urschler, and D. Štern, "Variational inference and bayesian cnns for uncertainty estimation in multi-factorial bone age prediction," *arXiv preprint arXiv:2002.10819*, 2020.

[7] D. Feng, L. Rosenbaum, and K. Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3266–3273.

[8] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 502–511.

[9] A. Amini, A. Soleimany, S. Karaman, and D. Rus, "Spatial uncertainty sampling for end-to-end control," *arXiv preprint arXiv:1805.04829*, 2018.

[10] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153–3160, 2020.

[11] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *International Conference on Learning Representations*, 2018.

[12] J. Mitros and B. Mac Namee, "On the validity of bayesian neural networks for uncertainty estimation," *arXiv preprint arXiv:1912.01530*, 2019.

[13] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 991–14 002.

[14] M. S. Ayhan and P. Berens, "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks," in *Medical Imaging with Deep Learning Conference*, 2018.

[15] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[16] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, 2020, pp. 4697–4708.

[17] M. Rawat, M. Wistuba, and M.-I. Nicolae, "Harnessing model uncertainty for detecting adversarial examples," in *NIPS Workshop on Bayesian Deep Learning*, 2017.

[18] A. C. Serban, E. Poll, and J. Visser, "Adversarial examples-a complete characterisation of the phenomenon," *arXiv preprint arXiv:1810.01185*, 2018.

[19] L. Smith and Y. Gal, "Understanding measures of uncertainty for adversarial example detection," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2018, pp. 560–569.

[20] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[21] M. Rußwurm, S. M. Ali, X. X. Zhu, Y. Gal, and M. Körner, "Model and data uncertainty for satellite time series forecasting with deep recurrent models," in *2020 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2020.

[22] J. Gawlikowski, S. Saha, A. Kruspe, and X. X. Zhu, "Out-of-distribution detection in satellite image classification," in *RobustML workshop at ICLR 2021*. ICRL, 2021, pp. 1–5.

[23] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1183–1192.

[24] K. Chitta, J. M. Alvarez, and A. Lesnikowski, "Large-scale visual active learning with deep probabilistic ensembles," *arXiv preprint arXiv:1811.03575*, 2018.

[25] J. Zeng, A. Lesnikowski, and J. M. Alvarez, "The relevance of bayesian layer positioning to model uncertainty in deep bayesian active learning," *arXiv preprint arXiv:1811.12535*, 2018.

[26] V.-L. Nguyen, S. Destercke, and E. Hüllermeier, "Epistemic uncertainty sampling," in *International Conference on Discovery Science*. Springer, 2019, pp. 72–86.

[27] W. Huang, J. Zhang, and K. Huang, "Bootstrap estimated uncertainty of the environment model for model-based reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3870–3877.

[28] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[29] B. Lötjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8662–8668.

[30] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, 2015, pp. 1613–1622.

[31] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in neural information processing systems*, 2017, pp. 6402–6413.

[32] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7047–7058.

[33] X. Zhao, Y. Ou, L. Kaplan, F. Chen, and J.-H. Cho, "Quantifying classification uncertainty using regularized evidential neural networks," *arXiv preprint arXiv:1910.06864*, 2019.

[34] Q. Wu, H. Li, W. Su, L. Li, and Z. Yu, "Quantifying intrinsic uncertainty in classification via deep dirichlet mixture networks," *arXiv preprint arXiv:1906.04450*, 2019.

[35] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty estimation using a single deep deterministic neural network," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, pp. 9690–9700.

[36] T. Ramalho and M. Miranda, "Density estimation in representation space to predict model uncertainty," in *Engineering Dependable and Secure Machine Learning Systems: Third International Workshop, EDSMLS 2020, New York City, NY, USA, February 7, 2020, Revised Selected Papers*, vol. 1272. Springer Nature, 2020, p. 84.

[37] A. Mobiny, H. V. Nguyen, S. Moulik, N. Garg, and C. C. Wu, "Dropconnect is effective in modeling uncertainty of bayesian deep networks," *arXiv preprint arXiv:1906.04569*, 2019.

[38] D. Krueger, C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville, "Bayesian hypernetworks," *arXiv preprint arXiv:1710.04759*, 2017.

[39] M. Valdenegro-Toro, "Deep sub-ensembles for fast uncertainty estimation in image classification," in *Bayesian Deep Learning Workshop at Neural Information Processing Systems 2019*, 2019.

[40] Y. Wen, D. Tran, and J. Ba, "Batchensemble: an alternative approach to efficient ensemble and lifelong learning," in *8th International Conference on Learning Representations*, 2020.

[41] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[42] Q. Wen, L. Sun, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," *arXiv preprint arXiv:2002.12478*, 2020.

[43] T. Tsiligkaridis, "Information robust dirichlet networks for predictive uncertainty estimation," *arXiv preprint arXiv:1910.04819*, 2019.

[44] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," in *Advances in Neural Information Processing Systems*, 2018, pp. 3179–3189.

[45] A. Malinin, B. Mlodozeniec, and M. Gales, "Ensemble distribution distillation," in *8th International Conference on Learning Representations*, 2020.

[46] M. Raghu, K. Blumer, R. Sayres, Z. Obermeyer, B. Kleinberg, S. Mullainathan, and J. Kleinberg, "Direct uncertainty prediction for medical second opinions," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5281–5290.

[47] J. Wenger, H. Kjellström, and R. Triebel, "Non-parametric calibration for classification," in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 178–190.

[48] J. Zhang, B. Kailkhura, and T. Y.-J. Han, "Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 117–11 128.

[49] R. Ghanem, D. Higdon, and H. Owhadi, *Handbook of uncertainty quantification*. Springer, 2017, vol. 6.

[50] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[51] A. G. Kendall, "Geometry and uncertainty in deep learning for computer vision," Ph.D. dissertation, University of Cambridge, 2019.

[52] A. Malinin, "Uncertainty estimation in deep learning with application to spoken language assessment," Ph.D. dissertation, University of Cambridge, 2019.

[53] H. Wang and D.-Y. Yeung, "Towards bayesian deep learning: A framework and some existing methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3395–3408, 2016.

[54] ——, "A survey on bayesian deep learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–37, 2020.

[55] N. Ståhl, G. Falkman, A. Karlsson, and G. Mathiason, "Evaluation of uncertainty quantification in deep learning," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer International Publishing, 2020, pp. 556–568.

[56] F. K. Gustafsson, M. Danelljan, and T. B. Schon, "Evaluating scalable bayesian deep learning methods for robust computer vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 318–319.

[57] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[58] 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," *Information Fusion*, 2021.

[59] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[60] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[61] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv preprint arXiv:1506.02158*, 2015.

[62] C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[63] H. Ritter, A. Botev, and D. Barber, "A scalable laplace approximation for neural networks," in *6th International Conference on Learning Representations*, vol. 6. International Conference on Representation Learning, 2018.

[64] J. Nandy, W. Hsu, and M. L. Lee, "Towards maximizing the representation gap between in-domain & out-of-distribution examples," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., 2020, pp. 9239–9250.

[65] A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov, "Pitfalls of in-domain uncertainty estimation and ensembling in deep learning," in *International Conference on Learning Representations*, 2020.

[66] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[67] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *5th International Conference on Learning Representations*, 2017.

[68] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *6th International Conference on Learning Representations*, 2018.

[69] A. Shafaei, M. Schmidt, and J. J. Little, "A less biased evaluation of out-of-distribution sample detectors," in *British Machine Vision Conference 2019*, 2019.

[70] M. Mundt, I. Pliushch, S. Majumder, and V. Ramesh, "Open set recognition through deep neural network uncertainty: Does out-of-distribution detection require generative classifiers?" in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[71] P. Oberdiek, M. Rottmann, and H. Gottschalk, "Classification uncertainty of deep neural networks based on gradient information," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer, 2018, pp. 113–125.

[72] J. Lee and G. AlRegib, "Gradients as a measure of uncertainty in neural networks," in *2020 IEEE International Conference on Image Processing*. IEEE, 2020, pp. 2416–2420.

[73] G. E. Hinton and 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.

[74] D. Barber and C. M. Bishop, "Ensemble learning in bayesian neural networks," *Nato ASI Series F Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.

[75] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, 2011, pp. 2348–2356.

[76] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Advances in neural information processing systems*, 2017, pp. 3288–3298.

[77] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, 2015, pp. 1530–1538.

[78] R. M. Neal, "Bayesian training of backpropagation networks by the hybrid monte carlo method," Citeseer, Tech. Rep., 1992.

[79] ——, "An improved acceptance procedure for the hybrid monte carlo algorithm," *Journal of Computational Physics*, vol. 111, no. 1, pp. 194–203, 1994.

[80] ——, "Bayesian learning for neural networks," Ph.D. dissertation, University of Toronto, 1995.

[81] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th international conference on machine learning*, 2011, pp. 681–688.

[82] C. Nemeth and P. Fearnhead, "Stochastic gradient markov chain monte carlo," *Journal of the American Statistical Association*, pp. 1–18, 2020.

[83] T. Salimans and D. P. Kingma, "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 901–909.

[84] J. Lee, M. Humt, J. Feng, and R. Triebel, "Estimating model uncertainty of neural networks in sparse information form," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5702–5713.

[85] O. Achrack, O. Barzilay, and R. Kellerman, "Multi-loss sub-ensembles for accurate classification with uncertainty estimation," *arXiv preprint arXiv:2010.01917*, 2020.

[86] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," in *International conference on learning representations*, 2017.

[87] G. D. Cavalcanti, L. S. Oliveira, T. J. Moura, and G. V. Carvalho, "Combining diversity measures for ensemble pruning," *Pattern Recognition Letters*, vol. 74, pp. 38–45, 2016.

[88] H. Guo, H. Liu, R. Li, C. Wu, Y. Guo, and M. Xu, "Margin & diversity based ordering ensemble pruning," *Neurocomputing*, vol. 275, pp. 237–246, 2018.

[89] W. G. Martinez, "Ensemble pruning via quadratic margin maximization," *IEEE Access*, vol. 9, pp. 48 931–48 951, 2021.

[90] J. Lindqvist, A. Olmin, F. Lindsten, and L. Svensson, "A general framework for ensemble distribution distillation," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[91] D. Molchanov, A. Lyzhov, Y. Molchanova, A. Ashukha, and D. Vetrov, "Greedy policy search: A simple baseline for learnable test-time augmentation," *arXiv preprint arXiv:2002.09103*, vol. 2, no. 7, 2020.

[92] M. Możejko, M. Susik, and R. Karczewski, "Inhibited softmax for uncertainty estimation in neural networks," *arXiv preprint arXiv:1810.01861*, 2018.

[93] L. Oala, C. Heiß, J. Macdonald, M. März, W. Samek, and G. Kutyniok, "Interval neural networks: Uncertainty scores," *arXiv preprint arXiv:2003.11566*, 2020.

[94] A. Malinin and M. Gales, "Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 14 547–14 558.

[95] V. T. Vasudevan, A. Sethy, and A. R. Ghias, "Towards better confidence estimation for neural models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7335–7339.

[96] M. Hein, M. Andriushchenko, and J. Bitterwolf, "Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 41–50.

[97] T. Joo, U. Chung, and M.-G. Seo, "Being bayesian about categorical probability," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4950–4961.

[98] T. Tsiligkaridis, "Failure prediction by confidence estimation of uncertainty-aware dirichlet networks," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2021, pp. 3525–3529.

[99] ——, "Information robust dirichlet networks for predictive uncertainty estimation," *arXiv preprint arXiv:1910.04819*, 2019.

[100] A. P. Dempster, "A generalization of bayesian inference," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 30, no. 2, pp. 205–232, 1968.

[101] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 14 927–14 937.

[102] B. Charpentier, D. Zügner, and S. Günnemann, "Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, 2020, pp. 1356–1367.

[103] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 6417–6428.

[104] T. Kawashima, Q. Yu, A. Asai, D. Ikami, and K. Aizawa, "The aleatoric uncertainty estimation using a separate formulation with virtual residuals," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 1438–1445.

[105] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 951–10 960.

[106] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield, "Large automatic learning, rule extraction, and generalization," *Complex systems*, vol. 1, no. 5, pp. 877–922, 1987.

[107] N. Tishby, E. Levin, and S. A. Solla, "Consistent inference of probabilities in layered networks: Predictions and generalization," in *International Joint Conference on Neural Networks*, vol. 2, 1989, pp. 403–409.

[108] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation," *Complex systems*, vol. 5, no. 6, pp. 603–643, 1991.

[109] D. J. C. MacKay, "Bayesian model comparison and backprop nets," in *Advances in neural information processing systems*, 1992, pp. 839–846.

[110] M.-A. Sato, "Online model selection based on the variational bayes," *Neural computation*, vol. 13, no. 7, pp. 1649–1681, 2001.

[111] A. Corduneanu and C. M. Bishop, "Variational bayesian model selection for mixture distributions," in *Artificial intelligence and Statistics*, vol. 2001. Morgan Kaufmann Waltham, MA, 2001, pp. 27–34.

[112] S. Ghosh, J. Yao, and F. Doshi-Velez, "Model selection in bayesian neural networks via horseshoe priors," *Journal of Machine Learning Research*, vol. 20, no. 182, pp. 1–46, 2019.

[113] M. Federici, K. Ullrich, and M. Welling, "Improved bayesian compression," *arXiv preprint arXiv:1711.06494*, 2017.

[114] J. Achterhold, J. M. Koehler, A. Schmeink, and T. Genewein, "Variational network quantization," in *International Conference on Learning Representations*, 2018.

[115] D. J. MacKay, "Information-based objective functions for active data selection," *Neural computation*, vol. 4, no. 4, pp. 590–604, 1992.

[116] A. Kirsch, J. van Amersfoort, and Y. Gal, "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 7026–7037.

[117] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," in *International Conference on Learning Representations*, 2018.

[118] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach, "Uncertainty-guided continual learning with bayesian neural networks," in *International Conference on Learning Representations*, 2020.

[119] S. Farquhar and Y. Gal, "A unifying bayesian view of continual learning," *arXiv preprint arXiv:1902.06494*, 2019.

[120] H. Li, P. Barnaghi, S. Enshaeifar, and F. Ganz, "Continual learning using bayesian neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[121] M. E. E. Khan, A. Immer, E. Abedi, and M. Korzepa, "Approximate inference turns deep networks into gaussian processes," in *Advances in neural information processing systems*, 2019, pp. 3094–3104.

[122] J. S. Denker and Y. LeCun, "Transforming neural-net output levels to probability distributions," in *Advances in neural information processing systems*, 1991, pp. 853–859.

[123] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[124] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner, "Black-box alpha divergence minimization," in *International Conference on Machine Learning*, 2016, pp. 1511–1520.

[125] Y. Li and Y. Gal, "Dropout inference in bayesian neural networks with alpha-divergences," in *International Conference on Machine Learning*, 2017, pp. 2052–2061.

[126] T. Minka *et al.*, "Divergence measures and message passing," Technical report, Microsoft Research, Tech. Rep., 2005.

[127] T. P. Minka, "Expectation propagation for approximate bayesian inference," in *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, 2001, pp. 362–369.

[128] J. Zhao, X. Liu, S. He, and S. Sun, "Probabilistic inference of bayesian neural networks with generalized expectation propagation," *Neurocomputing*, vol. 412, pp. 392–398, 2020.

[129] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International Conference on Machine Learning*, 2015, pp. 1861–1869.

[130] D. Tran, A. Kucukelbir, A. B. Dieng, M. Rudolph, D. Liang, and D. M. Blei, "Edward: A library for probabilistic modeling, inference, and criticism," *arXiv preprint arXiv:1610.09787*, 2016.

[131] D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei, "Deep probabilistic programming," in *International Conference on Machine Learning*, 2016.

[132] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 973–978, 2019.

[133] R. Cabañas, A. Salmerón, and A. R. Masegosa, "Inferpy: Probabilistic modeling with tensorflow made easy," *Knowledge-Based Systems*, vol. 168, pp. 25–27, 2019.

[134] Y. Ito, C. Srinivasan, and H. Izumi, "Bayesian learning of neural networks adapted to changes of prior probabilities," in *International Conference on Artificial Neural Networks*. Springer, 2005, pp. 253–259.

[135] S. Sun, G. Zhang, J. Shi, and R. Grosse, "Functional variational bayesian neural networks," in *International Conference on Learning Representations*, 2018.

[136] S. Depeweg, J. M. Hernández-Lobato, S. Udluft, and T. Runkler, "Sensitivity analysis for predictive uncertainty in bayesian neural networks," *arXiv preprint arXiv:1712.03605*, 2017.

[137] S. Farquhar, L. Smith, and Y. Gal, "Try depth instead of weight correlations: Mean-field is a less restrictive assumption for deeper networks," *arXiv preprint arXiv:2002.03704*, 2020.

[138] J. Postels, F. Ferroni, H. Coskun, N. Navab, and F. Tombari, "Sampling-free epistemic uncertainty estimation using approximated variance propagation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2931–2940.

[139] J. Gast and S. Roth, "Lightweight probabilistic deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3369–3378.

[140] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and 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.

[141] ——, "Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1184–1193.

[142] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in neural information processing systems*, 2015, pp. 2575–2583.

[143] A. Wu, S. Nowozin, E. Meeds, R. Turner, J. Hernández-Lobato, and A. Gaunt, "Deterministic variational inference for robust bayesian neural networks," in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[144] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix gaussian posteriors," in *International Conference on Machine Learning*, 2016, pp. 1708–1716.

[145] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse, "Noisy natural gradient as variational inference," in *International Conference on Machine Learning*, 2018, pp. 5852–5861.

[146] S. Sun, C. Chen, and L. Carin, "Learning structured weight uncertainty in bayesian neural networks," in *Artificial Intelligence and Statistics*, 2017, pp. 1283–1292.

[147] J. Bae, G. Zhang, and R. Grosse, "Eigenvalue corrected noisy natural gradient," *arXiv preprint arXiv:1811.12565*, 2018.

[148] A. Mishkin, F. Kunstner, D. Nielsen, M. Schmidt, and M. E. Khan, "Slang: Fast structured covariance approximations for bayesian deep learning with natural gradient," in *Advances in Neural Information Processing Systems*, 2018, pp. 6245–6255.

[149] C. Louizos and M. Welling, "Multiplicative normalizing flows for variational bayesian neural networks," in *International Conference on Machine Learning*, 2017, pp. 2218–2227.

[150] K. Osawa, S. Swaroop, M. E. E. Khan, A. Jain, R. Eschenhagen, R. E. Turner, and R. Yokota, "Practical deep learning with bayesian principles," in *Advances in neural information processing systems*, 2019, pp. 4287–4299.

[151] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in neural information processing systems*, 2017, pp. 3581–3590.

[152] Z. Eaton-Rosen, F. Bragman, S. Bisdas, S. Ourselin, and M. J. Cardoso, "Towards safe deep learning: accurately quantifying biomarker uncertainty in neural network predictions," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 691–699.

[153] C. R. N. Tassi, "Bayesian convolutional neural network: Robustly quantify uncertainty for misclassifications detection," in *Mediterranean Conference on Pattern Recognition and Artificial Intelligence*. Springer, 2019, pp. 118–132.

[154] P. McClure and N. Kriegeskorte, "Robustly representing uncertainty through sampling in deep neural networks," *arXiv preprint arXiv:1611.01639*, 2016.

[155] M. Khan, D. Nielsen, V. Tangkaratt, W. Lin, Y. Gal, and A. Srivastava, "Fast and scalable bayesian deep learning by weight-perturbation in adam," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2611–2620.

[156] M. E. Khan, Z. Liu, V. Tangkaratt, and Y. Gal, "Vprop: Variational inference using rmsprop," in *Advances in neural information processing systems*, 2017, pp. 3288–3298.

[157] A. Atanov, A. Ashukha, D. Molchanov, K. Neklyudov, and D. Vetrov, "Uncertainty estimation via stochastic batch normalization," in *International Symposium on Neural Networks*. Springer, 2019, pp. 261–269.

[158] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[159] R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[160] K. A. Dubey, S. J. Reddi, S. A. Williamson, B. Poczos, A. J. Smola, and E. P. Xing, "Variance reduction in stochastic gradient langevin dynamics," in *Advances in neural information processing systems*, 2016, pp. 1154–1162.

[161] B. Leimkuhler and S. Reich, *Simulating hamiltonian dynamics*. Cambridge university press, 2004, vol. 14.

[162] P. J. Rossky, J. Doll, and H. Friedman, "Brownian dynamics as smart monte carlo simulation," *The Journal of Chemical Physics*, vol. 69, no. 10, pp. 4628–4633, 1978.

[163] G. O. Roberts and O. Stramer, "Langevin diffusions and metropolis-hastings algorithms," *Methodology and computing in applied probability*, vol. 4, no. 4, pp. 337–357, 2002.

[164] H. Kushner and G. G. Yin, *Stochastic approximation and recursive algorithms and applications*. Springer Science & Business Media, 2003, vol. 35.

[165] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[166] Y.-A. Ma, T. Chen, and E. Fox, "A complete recipe for stochastic gradient mcmc," in *Advances in Neural Information Processing Systems*, 2015, pp. 2917–2925.

[167] G. Marceau-Caron and Y. Ollivier, "Natural langevin dynamics for neural networks," in *International Conference on Geometric Science of Information*. Springer, 2017, pp. 451–459.

[168] Z. Nado, J. Snoek, R. B. Grosse, D. Duvenaud, B. Xu, and J. Martens, "Stochastic gradient langevin dynamics that exploit neural network structure," in *International Conference on Learning Representations (Workshop)*, 2018.

[169] U. Simsekli, R. Badeau, A. T. Cemgil, and G. Richard, "Stochastic quasi-newton langevin monte carlo," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, 2016, pp. 642–651.

[170] Y. Zhang and C. A. Sutton, "Quasi-newton methods for markov chain monte carlo," in *Advances in Neural Information Processing Systems*, 2011, pp. 2393–2401.

[171] T. Fu, L. Luo, and Z. Zhang, "Quasi-newton hamiltonian monte carlo." in *Conference on Uncertainty in Artificial Intelligence*, 2016.

[172] C. Li, C. Chen, D. Carlson, and L. Carin, "Preconditioned stochastic gradient langevin dynamics for deep neural networks," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1788–1794.

[173] S. Ahn, A. K. Balan, and M. Welling, "Bayesian posterior sampling via stochastic gradient fisher scoring," in *International Conference on Learning Representations*, 2012.

[174] S. Patterson and Y. W. Teh, "Stochastic gradient riemannian langevin dynamics on the probability simplex," in *Advances in neural information processing systems*, 2013, pp. 3102–3110.

[175] N. Ye and Z. Zhu, "Stochastic fractional hamiltonian monte carlo," in *IJCAI*, 2018, pp. 3019–3025.

[176] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven, "Bayesian sampling using stochastic gradient thermostats," in *Advances in neural information processing systems*, 2014, pp. 3203–3211.

[177] X. Shang, Z. Zhu, B. Leimkuhler, and A. J. Storkey, "Covariance-controlled adaptive langevin thermostat for large-scale bayesian sampling," in *Advances in Neural Information Processing Systems*, 2015, pp. 37–45.

[178] B. Leimkuhler and X. Shang, "Adaptive thermostats for noisy gradient systems," *SIAM Journal on Scientific Computing*, vol. 38, no. 2, pp. A712–A736, 2016.

[179] S. Ahn, B. Shahbaba, and M. Welling, "Distributed stochastic gradient mcmc," in *International conference on machine learning*, 2014, pp. 1044–1052.

[180] K.-C. Wang, P. Vicol, J. Lucas, L. Gu, R. Grosse, and R. Zemel, "Adversarial distillation of bayesian neural network posteriors," in *International Conference on Machine Learning*, 2018, pp. 5190–5199.

[181] A. K. Balan, V. Rathod, K. P. Murphy, and M. Welling, "Bayesian dark knowledge," in *Advances in Neural Information Processing Systems*, 2015, pp. 3438–3446.

[182] D. Zou, P. Xu, and Q. Gu, "Stochastic variance-reduced hamilton monte carlo methods," in *International Conference on Machine Learning*, 2018, pp. 6028–6037.

[183] A. Durmus, U. Simsekli, E. Moulines, R. Badeau, and G. Richard, "Stochastic gradient richardson-romberg markov chain monte carlo," in *Advances in Neural Information Processing Systems*, 2016, pp. 2047–2055.

[184] A. Durmus, E. Moulines *et al.*, "High-dimensional bayesian inference via the unadjusted langevin algorithm," *Bernoulli*, vol. 25, no. 4A, pp. 2854–2882, 2019.

[185] I. Sato and H. Nakagawa, "Approximation analysis of stochastic gradient langevin dynamics by using fokker-planck equation and ito process," in *International Conference on Machine Learning*, 2014, pp. 982–990.

[186] C. Chen, N. Ding, and L. Carin, "On the convergence of stochastic gradient mcmc algorithms with high-order integrators," in *Advances in Neural Information Processing Systems*, 2015, pp. 2278–2286.

[187] Y. W. Teh, A. H. Thiery, and S. J. Vollmer, "Consistency and fluctuations for stochastic gradient langevin dynamics," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 193–225, 2016.

[188] C. Li, A. Stevens, C. Chen, Y. Pu, Z. Gan, and L. Carin, "Learning weight uncertainty with stochastic gradient mcmc for shape classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5666–5675.

[189] F. Wenzel, K. Roth, B. Veeling, J. Swiatkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin, "How good is the bayes posterior in deep neural networks really?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 248–10 259.

[190] N. Ye, Z. Zhu, and R. K. Mantiuk, "Langevin dynamics with continuous tempering for training deep neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 618–626.

[191] R. Chandra, K. Jain, R. V. Deo, and S. Cripps, "Langevin-gradient parallel tempering for bayesian neural learning," *Neurocomputing*, vol. 359, pp. 315–326, 2019.

[192] A. Botev, H. Ritter, and D. Barber, "Practical gauss-newton optimisation for deep learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 557–565.

[193] J. Martens and R. Grosse, "Optimizing neural networks with kronecker-factored approximate curvature," in *Proceeding of the 32nd International conference on machine learning*, 2015, pp. 2408–2417.

[194] S. Becker and Y. LeCun, "Improving the convergence of back-propagation learning with second-order methods," in *Proceedings of the 1988 Connectionist Models Summer School, San Mateo*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. Morgan Kaufmann, 1989, pp. 29–37.

[195] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 08 1989.

[196] P. Hennig, "Fast probabilistic optimization from noisy gradients," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28-1. PMLR, 2013, pp. 62–70.

[197] N. L. Roux and A. W. Fitzgibbon, "A fast natural newton method," in *Proceedings of the International Conference on Machine Learning*, 2010.

[198] R. B. Grosse and J. Martens, "A kronecker-factored approximate fisher matrix for convolution layers," in *Proceedings of the 33nd International Conference on Machine Learning*, 2016.

[199] S.-W. Chen, C.-N. Chou, and E. Chang, "Bda-pch: Block-diagonal approximation of positive-curvature hessian for training neural networks," *CoRR, abs/1802.06502*, 2018.

[200] J. Ba, R. Grosse, and J. Martens, "Distributed second-order optimization using kronecker-factored approximations," in *International Conference on Learning Representations*, 2017.

[201] T. George, C. Laurent, X. Bouthillier, N. Ballas, and P. Vincent, "Fast approximate natural gradient descent in a kronecker factored eigenbasis," in *Advances in Neural Information Processing Systems*, 2018, pp. 9573–9583.

[202] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2*, 1990, pp. 598–605.

[203] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[204] A. Kristiadi, M. Hein, and P. Hennig, "Being bayesian, even just a bit, fixes overconfidence in relu networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5436–5446.

[205] M. Humt, J. Lee, and R. Triebel, "Bayesian optimization meets laplace approximation for robotic introspection," *arXiv preprint arXiv:2010.16141*, 2020.

[206] A. Kristiadi, M. Hein, and P. Hennig, "Learnable uncertainty under laplace approximations," *arXiv preprint arXiv:2010.02720*, 2020.

[207] K. Shinde, J. Lee, M. Humt, A. Sezgin, and R. Triebel, "Learning multiplicative interactions with bayesian neural networks for visual-inertial odometry," in *Workshop on AI for Autonomous Driving at the 37th International Conference on Machine Learning*, 2020.

[208] J. Feng, M. Durner, Z.-C. Marton, F. Balint-Benczedi, and R. Triebel, "Introspective robot perception using smoothed predictions from bayesian neural networks," in *International Symposium on Robotics Research*, 2019.

[209] A. Y. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner, "'in-between' uncertainty in bayesian neural networks," *arXiv preprint arXiv:1906.11537*, 2019.

[210] A. Immer, M. Korzepa, and M. Bauer, "Improving predictions of bayesian neural nets via local linearization," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 703–711.

[211] M. Hobbhahn, A. Kristiadi, and P. Hennig, "Fast predictive uncertainty for classification with bayesian deep networks," *arXiv preprint arXiv:2003.01227*, 2020.

[212] E. Daxberger, E. Nalisnick, J. U. Allingham, J. Antorán, and J. M. Hernández-Lobato, "Expressive yet tractable bayesian deep learning via subnetwork inference," *arXiv preprint arXiv:2010.14689*, 2020.

[213] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 153–13 164.

[214] J. Mukhoti, P. Stenetorp, and Y. Gal, "On the importance of strong baselines in bayesian deep learning," *arXiv preprint arXiv:1811.09385*, 2018.

[215] A. Filos, S. Farquhar, A. N. Gomez, T. G. Rudner, Z. Kenton, L. Smith, M. Alizadeh, A. de Kroon, and Y. Gal, "A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks," *arXiv preprint arXiv:1912.10481*, 2019.

[216] J. Mukhoti and Y. Gal, "Evaluating bayesian deep learning methods for semantic segmentation," *arXiv preprint arXiv:1811.12709*, 2018.

[217] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[218] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[219] Y. Cao, T. A. Geddes, J. Y. H. Yang, and P. Yang, "Ensemble deep learning in bioinformatics," *Nature Machine Intelligence*, pp. 1–9, 2020.

[220] L. Nannia, S. Ghidoni, and S. Brahnam, "Ensemble of convolutional neural networks for bioimage classification," *Applied Computing and Informatics*, 2020.

[221] L. Wei, S. Wan, J. Guo, and K. K. Wong, "A novel hierarchical selective ensemble classifier with bioinformatics application," *Artificial intelligence in medicine*, vol. 83, pp. 82–90, 2017.

[222] F. Lv, M. Han, and T. Qiu, "Remote sensing image classification based on ensemble extreme learning machine with stacked autoencoder," *IEEE Access*, vol. 5, pp. 9021–9031, 2017.

[223] X. Dai, X. Wu, B. Wang, and L. Zhang, "Semisupervised scene classification for remote sensing images: A method based on convolutional

neural networks and ensemble learning," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 6, pp. 869–873, 2019.

[224] E. Marushko and A. Doudkin, "Methods of using ensembles of heterogeneous models to identify remote sensing objects," *Pattern Recognition and Image Analysis*, vol. 30, no. 2, pp. 211–216, 2020.

[225] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *International Conference on Learning Representations*, 2018.

[226] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," in *International Conference on Learning Representations*, 2017.

[227] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv preprint arXiv:1912.02757*, 2019.

[228] A. Renda, M. Barsacchi, A. Bechini, and F. Marcelloni, "Comparing ensemble strategies for deep learning: An application to facial expression recognition," *Expert Systems with Applications*, vol. 136, pp. 1–11, 2019.

[229] E. J. Herron, S. R. Young, and T. E. Potok, "Ensembles of networks produced from neural architecture search," in *International Conference on High Performance Computing*. Springer, 2020, pp. 223–234.

[230] S. Lee, S. Purushwalkam, M. Cogswell, D. Crandall, and D. Batra, "Why m heads are better than one: Training a diverse ensemble of deep networks," *arXiv preprint arXiv:1511.06314*, 2015.

[231] I. E. Livieris, L. Iliadis, and P. Pintelas, "On ensemble techniques of weight-constrained neural networks," *Evolving Systems*, pp. 1–13, 2020.

[232] L. Nanni, S. Brahnam, and G. Maguolo, "Data augmentation for building an ensemble of convolutional neural networks," in *Innovation in Medicine and Healthcare Systems, and Multimedia*. Singapore: Springer Singapore, 2019, pp. 61–69.

[233] J. Guo and S. Gould, "Deep cnn ensemble with data augmentation for object detection," *arXiv preprint arXiv:1506.07224*, 2015.

[234] R. Rahaman and A. H. Thiery, "Uncertainty quantification and deep ensembles," *stat*, vol. 1050, p. 20, 2020.

[235] Y. Wen, G. Jerfel, R. Muller, M. W. Dusenberry, J. Snoek, B. Lakshminarayanan, and D. Tran, "Combining ensembles and data augmentation can harm your calibration," in *International Conference on Learning Representations*, 2021.

[236] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, "Attention-based ensemble for deep metric learning," in *Proceedings of the European Conference on Computer Vision*, 2018.

[237] J. Yang and F. Wang, "Auto-ensemble: An adaptive learning rate scheduling based deep learning model ensembling," *IEEE Access*, vol. 8, pp. 217 499–217 509, 2020.

[238] M. Leutbecher and T. N. Palmer, "Ensemble forecasting," *Journal of computational physics*, vol. 227, no. 7, pp. 3515–3539, 2008.

[239] W. S. Parker, "Ensemble modeling, uncertainty and robust predictions," *Wiley Interdisciplinary Reviews: Climate Change*, vol. 4, no. 3, pp. 213–223, 2013.

[240] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler, "The power of ensembles for active learning in image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9368–9377.

[241] A. Vyas, N. Jammalamadaka, X. Zhu, D. Das, B. Kaul, and T. L. Willke, "Out-of-distribution detection using an ensemble of self supervised leave-out classifiers," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 550–564.

[242] J. Kocić, N. Jovičić, and V. Drndarević, "An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms," *Sensors*, vol. 19, no. 9, p. 2064, 2019.

[243] G. Martínez-Muñoz, D. Hernández-Lobato, and A. Suárez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 245–259, 2008.

[244] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[245] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *stat*, vol. 1050, p. 9, 2015.

[246] E. Englesson and H. Azizpour, "Efficient evaluation-time uncertainty estimation by improved distillation," in *Workshop on Uncertainty and Robustness in Deep Learning at International Conference on Machine Learning*, 2019.

[247] S. Reich, D. Mueller, and N. Andrews, "Ensemble distillation for structured prediction: Calibrated, accurate, fast—choose three," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 5583–5595.

[248] G. Wang, W. Li, S. Ourselin, and T. Vercauteren, "Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation," in *International MICCAI Brainlesion Workshop*. Springer, 2018, pp. 61–72.

[249] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks," *Neurocomputing*, vol. 338, pp. 34–45, 2019.

[250] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Scientific reports*, vol. 10, no. 1, pp. 1–7, 2020.

[251] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[252] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "When and why test-time augmentation works," *arXiv preprint arXiv:2011.11156*, 2020.

[253] I. Kim, Y. Kim, and S. Kim, "Learning loss for test-time augmentation," in *Advances in Neural Information Processing Systems*, 2020, pp. 4163–4174.

[254] Q. Yu and K. Aizawa, "Unsupervised out-of-distribution detection by maximum classifier discrepancy," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9518–9526.

[255] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 707–14 718.

[256] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez, "Quality of uncertainty quantification for bayesian neural network inference," *arXiv preprint arXiv:1906.09686*, 2019.

[257] X. Huang, J. Yang, L. Li, H. Deng, B. Ni, and Y. Xu, "Evaluating and boosting uncertainty quantification in classification," *arXiv preprint arXiv:1909.06030*, 2019.

[258] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.

[259] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4075–4084.

[260] D. Su, Y. Y. Ting, and J. Ansel, "Tight prediction intervals using expanded interval minimization," *arXiv preprint arXiv:1806.11222*, 2018.

[261] P. McClure, N. Rho, J. A. Lee, J. R. Kaczmarzyk, C. Y. Zheng, S. S. Ghosh, D. M. Nielson, A. G. Thomas, P. Bandettini, and F. Pereira, "Knowing what you know in brain segmentation using bayesian deep neural networks," *Frontiers in neuroinformatics*, vol. 13, p. 67, 2019.

[262] A. P. Soleimany, H. Suresh, J. J. G. Ortiz, D. Shanmugam, N. Gural, J. Guttag, and S. N. Bhatia, "Image segmentation of liver stage malaria infection with spatial uncertainty sampling," *arXiv preprint arXiv:1912.00262*, 2019.

[263] R. D. Soberanis-Mukul, N. Navab, and S. Albarqouni, "Uncertainty-based graph convolutional networks for organ segmentation refinement," in *Medical Imaging with Deep Learning*. PMLR, 2020, pp. 755–769.

[264] P. Seebock, J. I. Orlando, T. Schlegl, S. M. Waldstein, H. Bogunovic, S. Klimscha, G. Langs, and U. Schmidt-Erfurth, "Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal oct," *IEEE Transactions on Medical Imaging*, vol. 39, p. 87–98, 2020.

[265] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2796–2804.

[266] S. Seo, P. H. Seo, and B. Han, "Learning for single-shot confidence calibration in deep neural networks through stochastic inferences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9030–9038.

[267] Z. Li and D. Hoiem, "Improving confidence estimates for unfamiliar examples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2686–2695.

[268] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[269] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[270] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Advances in Neural Information Processing Systems*, 2019, pp. 4694–4703.

[271] B. Venkatesh and J. J. Thiagarajan, "Heteroscedastic calibration of uncertainty estimators in deep learning," *arXiv preprint arXiv:1910.14179*, 2019.

[272] P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson, "Subspace inference for bayesian deep learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 1169–1179.

[273] Z. Zhang, A. V. Dalca, and M. R. Sabuncu, "Confidence calibration for convolutional neural networks using structured dropout," *arXiv preprint arXiv:1906.09551*, 2019.

[274] M.-H. Laves, S. Ihler, K.-P. Kortmann, and T. Ortmaier, "Well-calibrated model uncertainty with temperature scaling for dropout variational inference," *arXiv preprint arXiv:1909.13550*, 2019.

[275] A. Mehrtash, W. M. Wells, C. M. Tempany, P. Abolmaesumi, and T. Kapur, "Confidence calibration and predictive uncertainty estimation for deep medical image segmentation," *IEEE Transactions on Medical Imaging*, 2020.

[276] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *International Conference on Machine Learning*, vol. 1. Citeseer, 2001, pp. 609–616.

[277] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," in *International Conference on Learning Representations*, 2019.

[278] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 888–13 899.

[279] J. Maroñas, D. Ramos, and R. Paredes, "Improving calibration in mixup-trained deep neural networks through confidence-based loss functions," *arXiv preprint arXiv:2003.09946*, 2020.

[280] K. Patel, W. Beluch, D. Zhang, M. Pfeiffer, and B. Yang, "On-manifold adversarial data augmentation improves uncertainty calibration," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 8029–8036.

[281] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2000, pp. 142–149.

[282] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[283] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 2015, 2015, p. 2901.

[284] M. Kull, M. Perelló-Nieto, M. Kängsepp, T. de Menezes e Silva Filho, H. Song, and P. A. Flach, "Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration," in *Advances in Neural Information Processing Systems*, 2019, pp. 12 295–12 305.

[285] D. Levi, L. Gispan, N. Giladi, and E. Fetaya, "Evaluating and calibrating uncertainty prediction in regression tasks," *arXiv preprint arXiv:1905.11659*, 2019.

[286] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön, "Evaluating model calibration in classification," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 3459–3467.

[287] M. H. DeGroot and S. E. Fienberg, "The comparison and evaluation of forecasters," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 32, no. 1-2, pp. 12–22, 1983.

[288] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran, "Measuring calibration in deep learning," in *CVPR Workshops*, 2019, pp. 38–41.

[289] A. Ghandeharioun, B. Eoff, B. Jou, and R. Picard, "Characterizing sources of uncertainty to proxy calibration and disambiguate annotator and data bias," in *2019 IEEE/CVF International Conference on Computer Vision Workshop*. IEEE, 2019, pp. 4202–4206.

[290] F. J. Pulgar, A. J. Rivera, F. Charte, and M. J. del Jesus, "On the impact of imbalanced data in convolutional neural networks performance," in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, 2017, pp. 220–232.

[291] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7167–7177.

[292] M. L. Iuzzolino, T. Umada, N. R. Ahmed, and D. A. Szafir, "In automation we trust: Investigating the role of uncertainty in active learning systems," *arXiv preprint arXiv:2004.00762*, 2020.

[293] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[294] R. Pop and P. Fulop, "Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles," *arXiv preprint arXiv:1811.03897*, 2018.

[295] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, "Bayesian reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 8, no. 5-6, pp. 359–483, 2015.

[296] S. Hu, D. Worrall, S. Knegt, B. Veeling, H. Huisman, and M. Welling, "Supervised uncertainty quantification for segmentation with multiple annotations," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 137–145.

[297] F. C. Ghesu, B. Georgescu, E. Gibson, S. Guendel, M. K. Kalra, R. Singh, S. R. Digumarthy, S. Grbic, and D. Comaniciu, "Quantifying and leveraging classification uncertainty for chest radiograph assessment," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 676–684.

[298] M. S. Ayhan, L. Kuehlewein, G. Aliyeva, W. Inhoffen, F. Ziemssen, and P. Berens, "Expert-validated estimation of diagnostic uncertainty for deep neural networks in diabetic retinopathy detection," *Medical Image Analysis*, p. 101724, 2020.

[299] N. Sunderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford *et al.*, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[300] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[301] D. Fox, "Markov localization-a probabilistic framework for mobile robot localization and navigation." Ph.D. dissertation, Citeseer, 1998.

[302] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, no. 3, pp. 325–344, 2000.

[303] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[304] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[305] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[306] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[307] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, "The bayes tree: An algorithmic foundation for probabilistic robot mapping," in *Algorithmic Foundations of Robotics IX*. Springer, 2010, pp. 157–173.

[308] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[309] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004.

[310] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in Neural Information Processing Systems*, 2010.

[311] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[312] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*. PMLR, 2018, pp. 466–476.

[313] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 491–496.

[314] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems*, 2017.

[315] H. Grimmett, R. Triebel, R. Paul, and I. Posner, "Introspective classification for robot perception," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 743–762, 2016.

[316] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[317] R. Triebel, H. Grimmett, R. Paul, and I. Posner, "Driven learning for driving: How introspection improves semantic mapping," in *Robotics Research*. Springer, 2016, pp. 449–465.

[318] A. Narr, R. Triebel, and D. Cremers, "Stream-based active learning for efficient and adaptive classification of 3d objects," in *2016 IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 227–233.

[319] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[320] A. Nguyen, J. Yosinski, and 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.

[321] K. Wong, S. Wang, M. Ren, M. Liang, and R. Urtasun, "Identifying unknown instances for autonomous driving," in *Conference on Robot Learning*. PMLR, 2020, pp. 384–393.

[322] W. Boerdijk, M. Sundermeyer, M. Durner, and R. Triebel, ""What's this?"–learning to segment unknown objects from manipulation sequences," in *Intern. Conf. on Robotics and Automation*, 2021.

[323] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," *Robotics: Science and Systems Foundation*, 2017.

[324] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly, "A smooth representation of belief over so (3) for deep rotation learning with uncertainty," *arXiv preprint arXiv:2006.01031*, 2020.

[325] B. Lütjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation*. IEEE, 2019, pp. 8662–8668.

[326] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5129–5136.

[327] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5703–5708.

[328] V. Tchuiev and V. Indelman, "Inference over distribution of posterior class probabilities for reliable bayesian classification and object-level perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4329–4336, 2018.

[329] Y. Feldman and V. Indelman, "Bayesian viewpoint-dependent robust classification under model and localization uncertainty," in *2018 IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 3221–3228.

[330] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1281–1292.

[331] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[332] C. Gurău, C. H. Tong, and I. Posner, "Fit for purpose? predicting perception performance based on past experience," in *International Symposium on Experimental Robotics*. Springer, 2016, pp. 454–464.

[333] S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert, "Introspective perception: Learning to predict failures in vision systems," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1743–1750.

[334] M. Netzband, W. L. Stefanov, and C. Redman, *Applied remote sensing for urban planning, governance and sustainability*. Springer Science & Business Media, 2007.

[335] C. Giardino, M. Bresciani, P. Villa, and A. Martinelli, "Application of remote sensing in water resource management: the case study of lake trasimeno, italy," *Water resources management*, vol. 24, no. 14, pp. 3885–3899, 2010.

[336] C. J. Van Westen, "Remote sensing for natural disaster management," *International archives of photogrammetry and remote sensing*, vol. 33, no. B7/4; PART 7, pp. 1609–1617, 2000.

[337] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.

[338] J. Gawlikowski, M. Schmitt, A. Kruspe, and X. X. Zhu, "On the fusion strategies of sentinel-1 and sentinel-2 data for local climate zone classification," in *IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 2081–2084.

[339] ESA, "European space agency (esa) developed earth observation satellites," 2019. [Online]. Available: http://www.esa.int:8080/ESA_Multimedia/Images/2019/05/ESA-developed_Earth_observation_missions

[340] Z. Nado, N. Band, M. Collier, J. Djolonga, M. W. Dusenberry, S. Farquhar, A. Filos, M. Havasi, R. Jenatton, G. Jerfel *et al.*, "Uncertainty baselines: Benchmarks for uncertainty & robustness in deep learning," *arXiv preprint arXiv:2106.04015*, 2021.

[341] M. Kull and P. A. Flach, "Reliability maps: a tool to enhance probability estimates and improve classification accuracy," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 18–33.

[342] J. Antorán, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato, "Getting a clue: A method for explaining uncertainty estimates," in *International Conference on Learning Representations*, 2021.

[343] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, and N. Carvalhais, "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.

[344] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv:2003.04919*, 2020.

[345] E. De Bézenac, A. Pajot, and P. Gallinari, "Deep learning for physical processes: Incorporating prior scientific knowledge," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124009, 2019.