Uncertainty Quantification

This is mainly a paper reading note of A survey of uncertainty in deep neural networks, complemented with some other sources of knowledge.

References:

- Artificial Intelligence Review - A survey of uncertainty in deep neural networks (2023) ⭐ ⭐
- A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges (2021)
- ICML - Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (2015)
- NIPS - Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles (2017)
- ICML - Weight Uncertainty in Neural Networks (2015)

1 Introduction

Over the last decade, neural networks have reached almost every field of science and become a crucial part of various real world applications. However, basic neural networks only give **point estimates** rather than **certainty estimates**. In high-risk fields such as medical image analysis or autonomous vehicle control, their deployment in mission- and safety-critical real world applications remains limited. Major limitations include:

- The **lack of expressiveness and transparency** of a deep neural network's inference model, which makes it difficult to trust their outcomes;
- The **inability to distinguish between in-domain and out-of-domain samples and the sensitivity to domain shifts**;
- The **inability to provide reliable uncertainty estimates** for a deep neural network's decision;
- The **sensitivity to adversarial attacks**.

To overcome these limitations, it is essential to provide uncertainty estimates.

2 Uncertainty in deep neural networks

2.1 Sources of Uncertainty

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.,

$$f_\theta : \mathbb{X} \to \mathbb{Y}, \quad x \mapsto f_\theta(x) \tag{1}$$

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

$$\mathcal{D} = (\mathcal{X}, \mathcal{Y}) = \{x_i, y_i\}_{i=1}^{N} \subseteq \mathbb{D} \tag{2}$$

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 is applied for inference (e.g. a BNN or an ensemble of neural networks).
4. the **prediction's uncertainty model**: The modelling of the uncertainties caused by the neural network and/or by the data.

The five factors that are the most vital for the cause of uncertainty in a DNN's predictions are

1. **The variability in real world situations**: Real world environment condition such as temperature and illumination may change, which can affect the expression of objects (such as plants after rain look very different from 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 model parameters and then the performance of a neural network.

2. **The errors inherent to the measurement systems**: The measurements themselves can be a source of uncertainty. This can be caused by clipped information, sensor noise, false labeling, ... Depending on the intensity, this type of noise and errors can be used to regularize the training process and to improve robustness and generalization.
3. **The errors in the architecture specification of the DNN**: The structure of a neural network has a direct effect on its performance and therefore also on the uncer
tainty 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.
4. **The errors in the training procedure of the DNN**: 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 introduces variability in model parameters.
5. **The 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.

2.2 Predictive uncertainty model

The uncertainty that is propagated onto a prediction y^* is called **predictive uncertainty**. Within the data acquisition model, the probability distribution for a prediction y^* based on some sample x^* is given by

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

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 for a new sample x^* is then given by

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

and the maximum a posteriori (MAP) estimator is given by

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

In general, the distribution given in (9) is unknown and can only be estimated based on the given data in $\mathcal{D}$. 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**) and **model uncertainty** (also systemic or **epistemic uncertainty**). Additionally, for examples from a region not covered by the training data, an explicit modeling of **distributional uncertainty** is adopted.

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

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

The term $p(\theta|D)$ is referenced as **posterior distribution** on the model parameters. It is in general not tractable. While ensemble approaches seek to approximate it by aggregating multiple models, Bayesian inference reformulates it using Bayes Theorem,

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

The term $p(\theta)$ is called the **prior distribution** on the model parameters. 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. The term $p(D)$ is often referred to as the **evidence**. Many loss functions are motivated by or can be related to the likelihood function, for example the cross-entropy or the mean squared error.

If taking distributional uncertainty into account, the predictive distribution can be written as

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

Modeled this way, distributional uncertainty refers to uncertainty that is caused by a change in the input-data distribution. As modeled in (13), the model uncertainty affects the estimation of the distributional uncertainty, which then affects the estimation of the data uncertainty.

2.3 Uncertainty classification

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

- **In-domain Uncertainty**:
  - 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 **a lack of in-domain knowledge**.
  - From a modeler's 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.
- **Domain-shift Uncertainty**:
  - 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.
  - From a modeler's 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**:
  - 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. A DNN cannot extract in-domain knowledge from **out-of-domain samples**. For example, when a network that learns to classify cats and dogs is asked to predict a bird, it's the case of out-of-domain uncertainty.
  - From a modeler's 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.

3 Uncertainty Estimation

As described in Section 2, several factors may cause model and data uncertainty and affect a DNN's prediction. Methods for estimating uncertainty in a DNN prediction is a popular and vital field of research.

In general, the methods for estimating the uncertainty can be split into 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 net works 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.

3.1 Single Deterministic Methods

Compared to many other principles, **single deterministic methods** are **computationally efficient** in training and evaluation. For training, only one network has to be trained and often the approaches can even be applied to pre-trained networks. For evaluation, usually only a single or at most two forward passes have to be fulfilled.

Single deterministic methods can be roughly categorized into **internal** and **external approaches**. In internal approaches, one single network is explicitly modeled and trained in order to quantify uncertainties; in external approaches, additional components are used in order to give an uncertainty estimate on the prediction of a network.

3.1.1 Internal Uncertainty Quantification Approaches

Many of the internal uncertainty quantification approaches followed the idea of **predicting the parameters of a distribution** intnstead 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. The distribution over the outputs can be interpreted as a quantification of the model uncertainty, and 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 can be already interpreted as a prediction of the data uncertainty. However, it is widely discussed that neural networks are often over-confident. For exam
ple, 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. That's because even though the features do not fit to the cat class, they might fit even less to the dog class.

Another popular approach is to make use of the **Dirichlet distribution**. It is the conjugate prior of the categorical distribution and hence can be interpreted as a distribution over categorical distributions. 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.

The Dirichlet distribution is utilized in several approaches as **Dirichlet Prior Networks** and **Evidential Neural Networks**. Both of these network types output the parameters of a Dirichlet distribution from which the categorical distribution describing the class probabilities can be derived.

Prior networks are trained in a multi-task way with the goal of minimizing the expected KL divergence between the predictions of **in-distribution data** and a **sharp Dirichlet distribution**, and between the predictions of **out-of-distribution data** and a **flat Dirichlet distribution**. There are also variants that use the mixture of M Dirichlet distributions to give much more flexibility in approximating the (possibly multi-modal) posterior distribution.

Evidential neural networks also optimize the parameterization of a single Dirichlet network. The loss formulation is derived by interpreting the logits as multinomial opinions or beliefs. Evidential neural networks set the total amount of evidence in relation to 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 the expected value of a basic loss, such as categorical cross entropy, with respect to a Dirichlet distribution parameterized by the logits. Additionally, a regularization term is added, encouraging the network predict the "I don't know state" if no evidence for an improvement in the data fit is found.

For **regression tasks**, one of the popular approaches is Interval Neural Network (INN). In contrast to Gaussian representations of uncertainty given by a standard deviation, this approach can give non-symmetric values of uncertainty. In addition, Deep Evidential Regression 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. Simultaneous Quantile Regression (SQR) is another approach where out-of-distribution samples are mapped to a non-zero value and thus can be recognized.

3.1.2 External Uncertainty Quantification Approaches

External uncertainty quantification approaches do not affect the models' predictions, since the uncertainty evaluation is **separated from the underlying prediction task**. Raghu et al. (2019) 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 training **two neural networks**, one for the **actual prediction**

task and a second one for the **uncertainty prediction** on the first network's predictions. Furthermore, some other works take the idea of **adversarial attack** and uses model's **gradients** to evaluate the model's sensitivity to changes in the input space.

3.2 Bayesian Neural Networks

3.2.1 Principles of Bayesian Neural Networks

Bayesian Neural Networks (BNNs) have the ability to combine the scalability, expressiveness, and 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** $\theta = (w_1, \ldots, w_K)$. More specifically, given a training input-target pair (x, y), the posterior distribution of the parameters, $p(\theta|x, y)$, is modeled by assuming a prior distribution over the parameters $p(\theta)$ and applying Bayes theorem:

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

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

$$p(y|x) = \int p(y|x,\theta)p(\theta)d\theta \tag{17}$$

Once the posterior distribution over the weights has been estimated, the prediction of 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:

$$p(y^*|x^*,x,y) = \int p(y^*|x^*,\theta)p(\theta|x,y)d\theta \tag{18}$$

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 stochastic networks, $f_{\theta_1}, \ldots, f_{\theta_N}$, parameterized by N samples, $\theta_1, \theta_2, \ldots, \theta_N$, from the posterior distribution of the weights, i.e.

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

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. Hence, approximate Bayesian inference techniques are often needed to compute the posterior probabilities. Yet, directly using approximate Bayesian inference techniques is difficult as the **size of the data and the 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 of model parameters θ is inferred:

- **Variational inference** (Hinton and Van Camp 1993; Barber and Bishop 1998)
  - Variational inference approaches approximate the (in general intractable) posterior distribution by optimizing over a family of tractable distributions.
- **Sampling approaches** (Neal 1992)
  - 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** (Denker and LeCun 1991; MacKay 1992c)
  - 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.

3.2.2 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, this 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 is given by the **Kullback–Leibler (KL) divergence**

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

As the above KL divergence can not be minimized directly, it is the **evidence lower bound (ELBO)** that is usually optimized. For a given prior distribution on the parameters $p(\theta)$, the ELBO is given by

$$\mathrm{ELBO}=\mathbb{E}_q\Big[\log\frac{p(y|x,\theta)}{q(\theta)}\Big]=\log p(y|x)-\mathrm{KL}(q\|p)\leq\underbrace{\log p(y|x)}_{\text{Evidence}} \tag{21}$$

Maximizing the ELBO is equivalent to minimizing the KL divergence.

An evident direction with the current methods is the use of **stochastic variational inference** (or Monte-Carlo variational inference), where the optimization of ELBO is performed using a mini-batch of data. One of the key concepts is to reformulate the loss function of the neural network as the ELBO. As a result, the intractable posterior distribution is indirectly optimized.

3.2.3 Sampling Approaches

Sampling methods, also often called **Monte Carlo methods**, are another family of Bayesian inference algorithms that represent uncertainty without a parametric model. Popular algorithms within this domain are particle filtering, rejection sampling, importance sampling, and **Markov Chain Monte Carlo sampling (MCMC)**. In the case of neural networks, MCMC is often used due to the dimensionality. Important extensions include **Hamiltonian Monte Carlo (HMC)** and **Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC)**.

3.2.4 Laplace Approximation

The goal of the Laplace Approximation is to estimate the posterior distribution over the parameters of neural networks $p(\theta|x,y)$ 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

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

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:

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

where H is the Hessian of $\log p(\theta|x,y)$. This means that the model uncertainty is represented by the Hessian H.

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. Existing works obtain Laplace Approximation using various approximations of the Hessian in the line of fidelity-complexity trade-offs. Recently there are also efforts that extend the Laplace approximation beyond the Hessian Approximation.

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.

3.3 Ensemble Methods

3.3.1 Principles of ensemble methods

Ensembles derive a prediction based on the predictions received from multiple so-called ensemble members. They target 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. For an ensemble $f : X \to Y$ with members $f_i : X \to Y$ for $i \in 1, 2, \ldots, M$, this could be for example implemented by simply averaging over the members' predictions,

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

Besides the improvement in accuracy, ensembles give an intuitive way of representing the model uncertainty on a prediction by evaluating the variety among the members' 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 ensemble training strategies that target efficiency.
- Second, ensemble methods were originally not introduced to explicitly handle and quantify uncertainties, but to improve the accuracy on a prediction. 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.

3.3.2 Single- and Multi-mode Evaluation

One major difference of ensemble methods compared to other methods is the number of local optima that are considered, i.e. the differentiation into **single-mode** and **multi-mode** evaluation.

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. Deterministic neural networks converge to **one single local optimum** in the solution space. Other approaches, e.g. BNNs, still converge to one single optimum, but additionally, take **the uncertainty on this local optimum** into account. 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**. 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.

3.3.3 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. In order to increase the variety, several different approaches can be applied:

- Random initialization and data shuffle
- Bagging and boosting
- Data augmentation
- Ensemble of different network architecture

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. It is important to note that if not explicitly stated, the works and approaches presented so far targeted improvements in predictive accuracy and did not explicitly consider uncertainty quantification.

3.3.4 Ensemble Methods and Uncertainty Quantification

Besides the improvement in the accuracy, ensembles are widely used for modelling uncertainty on predictions of complex models. An ensemble training pipeline is introduced 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. In all tests, the method performs at least equally well as the BNN approaches, namely Monte Carlo Dropout and Probabilistic Backpropagation.

It is also found 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 datasets. Furthermore, bagging is even found to worsen the predictive uncertainty estimation. It is reported that that already for a relatively small ensemble size of five, deep ensembles seem to perform best and are more robust to dataset shifts than the compared methods.

3.3.5 Making Ensemble Methods More Efficient

Compared to single model methods, ensemble methods come along with a significantly increased **computational effort and memory consumption**. To boost the efficiency of ensemble methods, several approaches have been developed.

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.

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. 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.

Other approaches, as sub-ensembles and batch-ensembles, seek to reduce the computation effort and memory consumption by **sharing parts among the single members**.

Sub-ensembles divide a neural network architecture into two sub-networks. The trunk network is for the extraction of general information from the input data, and the task network uses this 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 member's 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.

Batch-ensembles 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 matrix $F_i \in \mathbb{R}^{n \times m}$, each linked with one of the M ensemble members. The rank one matrices can be written as a multiplication $F_i = r_i s_i^\top$ 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. 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 parallelization can be realized similarly to the optimization on mini-batches and on a single unit.

3.4 Test-Time Augmentation

Inspired by ensemble methods and adversarial examples, the **test-time augmentation** is one of the simpler predictive uncertainty estimation techniques. The basic method is to create **multiple samples from each test sample** by applying data augmentation techniques and then test all those samples to compute a predictive distribution in order to measure uncertainty. The idea is that the augmented test samples allow the exploration of different views and is therefore capable of capturing the uncertainty. In general, test-time augmentation can use the same augmentation techniques that can be used for regularization during training and has been shown to improve calibration to in-distribution data and out-of-distribution data detection. Mostly, this technique is used in situations where the training data is limited, e.g. medical image analysis.

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**. It is discovered that 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, the size of training data, the deep neural network architecture, and the type of augmentation. The data augmentation techniques should be carefully chosen to avoid introducing unwanted biases.

4 Uncertainty measures and quality

In order to **evaluate** different uncertainty modeling approaches, measures have to be applied to the derived uncertainties.

4.1 Evaluating uncertainty in classification tasks

4.1.1 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. In order to evaluate the amount of predicted data uncertainty, one can for example apply the **maximal class probability** or the **entropy** measures:

$$\text{Maximal probability:} \quad p_{\max} = \max\{p_k\}_{k=1}^{K} \tag{28}$$

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

The maximal probability represents a direct representation of certainty, while entropy describes the average level of information in a random variable.

4.1.2 Measuring model uncertainty in classification tasks

As already discussed in Sect. 3, a single softmax prediction is not a very reliable way for uncertainty quantification since it is often badly calibrated and does not have any information about the certainty of the model itself has on this specific output. 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 **mutual information (MI)**, the **expected Kullback–Leibler Divergence (EKL)**, and the **predictive variance**.

Basically, all these measures compute **the expected softmax output**

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

and measure some kind of divergence between it and the actual softmax output.

The **MI** uses 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.

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

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

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

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 outputs, i.e.

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

As described in Sect. 3, 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, $\{p^i\}_{i=1}^{M}$, 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.

4.2 Evaluating uncertainty in regression tasks

4.2.1 Measuring data uncertainty in regression predictions

In contrast to classification tasks, where the network typically outputs a probability distribution over the possible classes, regression tasks only predict a **pointwise estimation** without any hint of data uncertainty. As already described in Sect. 3, 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. In 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

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

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 directly predict a so-called **prediction interval (PI)**,

$$PI(x) = [B_l, B_u] \tag{35}$$

in which the prediction is assumed to lay. Such intervals induce 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 mode. In order to evaluate the correctness of the predicted intervals the **Prediction Interval Coverage Probability (PICP)** can be applied. The PICP represents the percentage of test predictions that fall into a prediction interval and is defined as

$$\text{PICP} = \frac{c}{n} \tag{36}$$

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

4.2.2 Measuring model uncertainty in regression predictions

In Sect. 2, 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**.

4.3 Evaluating uncertainty in segmentation tasks

The evaluation of uncertainties in segmentation tasks is very **similar to** the evaluation of **classification** problems. In the context of segmentation, the uncertainty in pixel-wise segmentation is measured using confidence intervals, the predictive variance, the predictive entropy, or the mutual information.