EC4

Methodological Guideline for CAUTIONER

Document reference number for ANR

contact@confiance.ai | www.confiance.ai

Document reference: XXX

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Lawrence ADU-GYAMFI	CEA	Researcher
Scientific responsible	Lawrence ADU-GYAMFI	CEA	Researcher
Co-authors	Eiji KAWASAKI	CEA	Researcher

Document Control

Revision	Date	Commentary	Author
v1.0	17/04/24	Revision	Lawrence ADU-GYAMFI

Contents

A. Introduction and abstract

A.1. General introduction to trustworthy AI challenges

Trustworthiness in AI within critical systems (systems that can directly or indirectly affect human life and moral entities) is essential for its widespread adoption (by the industry, the decision makers, the general public, etc.) and poses the following significant challenges.

- First, how to design AI models, so that, by construction, they satisfy trustworthy properties (accuracy, robustness. . .).
- Secondly, how to characterize these AI models, for example to understand and explain their behavior and their adequacy to the operational domain.
- Then, how to implement and embed those AI models on hardware, by making them fit for the target without losing their trustworthy properties.
- Another question is, what methods of data engineering to apply in order to, among other topics, manage important volumes of data and adapt to the evolution of the operational domain.
- At system level, what verification and certification processes to consider specifically for AI-based systems.
- Finally, a federation of all these matters is necessary to build an end-to-end methodological approach, supported by a consistent engineering environment compatible with industrial practices.

These are the challenges, among others, that the Confiance.ai program addresses.

A.1.1 Rationale for this Methodology

The objective of this guideline is to provide a general overview of the needed information and best practices regarding the methodology of uncertainty quantification (UQ) for machine learning models. It also serves as a guide to understand the use of the CAUTIONER component as part of the Confiance.ai ecosystem to accomplish uncertainty quantification.

A.1.2 Scientific Challenges

The Confiance.ai scientific challenges that are addressed by this method include:

- Predicting the performance of a learning system without using a test base.
- Ensure robustness to variations from the environment (development vs industrial) or from input data.

A.1.3 Trustworthiness Attributes

The Trustworthiness Attributes that are addressed by this method include reliability and robustness.

A.1.4 Target audience and disclaimer

This document is written with the following audience in mind:

- Systems Engineer
- Data Engineer
- ML-Algorithm Engineer

A.1.5 Document organization

The rest of the document is organised as follows:

- Section B.1 presents the motivation and some background for uncertainty quantification in deep learning.
- Section B.2 presents a description of the CAUTIONER library as a tool for UQ.
- Section B.4 shows how to use CAUTIONER for uncertainty quantification for different supervised learning tasks.
- The conclusion highlights some limitations of the method and future work.

B. Description of the method

B.1. Overall vision, context, motivation, objective, added value of the proposed method

Supervised learning is a machine-learning task that aims at learning a mapping between input and output data. Learning this input-output relation enables the possibility to predict a future output data (yet unknown) based only on the input data. In practice, there is always a gap between a model's predictions and the actual observed data. We want to characterize this gap which we call uncertainty. The following sections show how to estimate the prediction uncertainty in order to design robust and trustworthy machine learning models. This uncertainty has an epistemic and an aleatoric component.

B.1.1 Epistemic Uncertainty

Epistemic uncertainty comes from a lack of knowledge of the real/physical model that generated the data. This error can therefore be reduced by accumulating more data and by increasing the flexibility of the model (the number of parameters for example). These two approaches roughly correspond to a reduction of the variance and the bias of the AI model. However, the reduction of this uncertainty cannot replace its measurement and estimation. Bayesian methods and ensemble methods are currently the most popular approaches for quantifying epistemic uncertainty.

B.1.2 Aleatoric Uncertainty

The predictive uncertainty is not exclusively determined by the bias and variance introduced by the representation of the AI model. Indeed, if the phenomenon under study includes a random component, it is impossible to establish an exact deterministic correspondence between input and output data. In this case, classical supervised models are doomed to provide an average prediction whose deviation from the observed value will be non-zero. This aleatoric uncertainty can sometimes be neglected. In the case where the stochastic component completely dominates the physical phenomenon, it is no longer possible to neglect it.
For instance, within the framework of meteorological predictions, models often a single temperature prediction to the user (random component neglected) whereas they offer a percentage of precipitation (random component taken into account).
In the provided method, the aleatoric uncertainty is modelled as a Gaussian unpredictable noise associated to the prediction target. We thus design a Neural Network that estimates both the prediction target mean value and the value of the variance of this aleatoric noise. This type of deep learning model name is Mixture Density Network (Bishop, 1994).

Towards addressing the challenge of uncertainty quantification, particularly in deep learning, we have developed the CAUTIONER library as part of the CONFIANCE.ai program. CAUTIONER provides Bayesian Neural Networks (BNNs) that are able to compute their prediction uncertainty by modeling the distribution over the parameters (weights and biases) given the training data set. BNN cannot be trained by a standard minimization : this component provides several methods to compute a prediction:

- a random walk MCMC
- a noisy gradient descent (Stochastic Gradient Langevin Dynamic)
- and lastly a batched MCMC that is beyond state of the art.

The details of the batched MCMC are further described in Kawasaki and Holzmann (2022). In this we show that naively using sub-sampled mini-batches to train BNN results in poor predictive performances.
The library or component allows the estimation of the prediction uncertainty in order to design robust and trustworthy machine learning models. It relies on Monte Carlo estimation of this uncertainty. CAUTIONER achieves this estimation by sampling the parameters of the learning model, a method that remains the gold standard in uncertainty estimation.

B.2. CAUTIONER as a UQ library

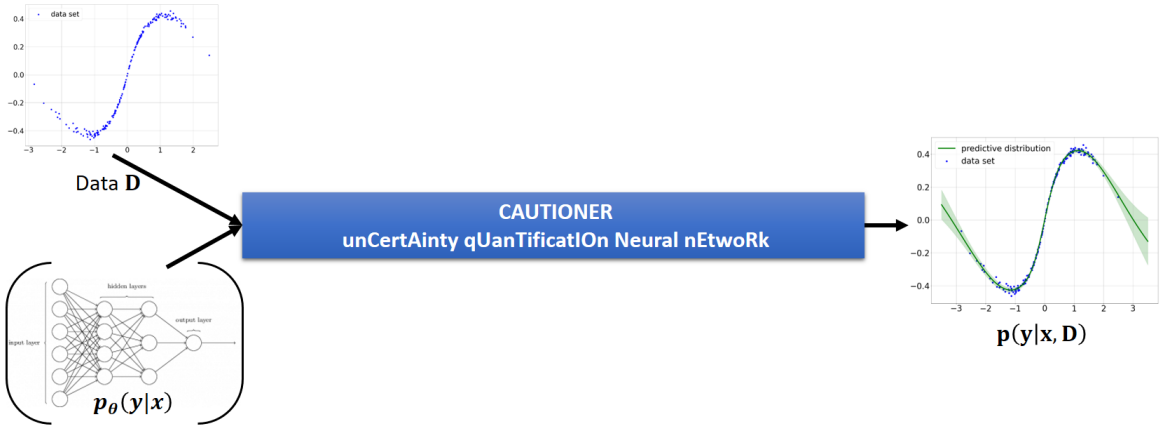

Figure B.1: Overview of Bayesian inference using BNNs as implemented in CAUTIONER. Inputs are a neural network architecture, prior or stochastic model(which is provided in CAUTIONER in this case) and the training data. The BNN can be used for inference on some test data to generate the posterior predictive distributions.

The CAUTIONER library developed as part of the CONFIANCE program contains:

- an MCMC module that is able to sample a distribution in high dimension that is defined up to a constant
- a Mixture Density Network module to take into account the aleatoric component of the uncertainty
- an Uncertainty Quantification module that samples a Bayesian Neural Network posterior using MCMC

B.2.1 Prerequisites for using the method

To perform uncertainty quantification using bayesian inference as per the method described, one will need a deep neural network architecture represented by a trained model, a stochastic model or a prior distribution over the model parameters and a training dataset. Samples from the posterior distribution of the BNN can then be generated and used for inference or prediction on a test set. This setup as implemented by CAUTIONER is summarized by Figure B.1.

In the following sections we detail the requirements to use CAUTIONER for designing BNNs for bayesian inference.

B.2.1.1 Input Variables

- **data_loader**: (`torch.DataLoader`): DataLoader containing the training input and output data for regression or labels for classification.
- **test_input_data**: (`torch.DataLoader`) : DataLoader containing the test input data for inference.
- **models**: (`List[torch.nn.Module]`) : List of pretrained neural network models. Currently works only with PyTorch models.
- **samples**: (`List[torch.Tensor]`) : (Optional) List of pre-computed samples from the posterior distribution of the model parameters. If not provided, samples are generated using the MCMC sampler.

B.2.1.2 Parameters

- **model_type**: (`str`): Regression specific; either of ['heteroscedastic', 'homoscedastic']. Optional; for creating / training models with CAUTIONER.
- **number_of_processes**: (`int`) : Number of models to create. Optional for creating / training models with CAUTIONER.
- **weight_decay**: (`float`) : Regularization parameter to prevent overfitting.
- **total_number_iterations**: (`float`) : The total number of iterations to run the MCMC algorithm for.
- **random_walk_size**: (`float`) : Step size for the random walk in sampling. Optional.

B.2.1.3 Output

Regression Specific Outputs

- **prediction_means**: (`numpy.ndarray`) : Average value(s) of the predicted outputs of the models on the input data.
- **prediction_stds**: (`numpy.ndarray`) : Standard deviation(s) around the mean(s) of the predicted outputs of the models, for the input data.

Classification Specific Outputs

- **test_output_predictions**: (`numpy.ndarray`) : Array containing log probability distributions over classes for the input data.

B.2.2 Hardware Requirements

The CAUTIONER library is optimized for performance on contemporary CPU architectures. It is engineered to handle its computational tasks effectively without imposing a heavy burden on the CPU. This means that even standard CPUs can manage the library's operations smoothly and efficiently.
There is no specific need for GPU resources in cautioner. It operates entirely within CPU capabilities, ensuring full functionality on systems that may not have a dedicated or high-end GPU.

B.3. Level of Maturity

According to the maturity level requirements, this method is at a maturity level of 3. The method has been validated on the following tasks:

- Regression: using a synthetic dataset.
- Classification: using the MNIST hand digit recognition dataset
- Time-Series: using the Air Liquide Demand forecast use case from CONFIANCE project.

B.4. Uncertainty Quantification with CAUTIONER

In the following sections, we demonstrate the application of uncertainty quantification on regression and classification tasks using the following datasets respectively;

- Synthetic regression
- MNIST hand written digits dataset

We explore the time series task in detail in a separate document: CAUTIONER application to Air Liquide demand forecast. In the examples provided below, we use the CAUTIONER library to generate samples from the posterior distribution of the model parameters for bayesian inference.

B.4.1 Regression Use Case

In this section we use a synthetic non-linear regression use case to demonstrate the concept described in this document, particularly how to use the CAUTIONER library to compute uncertainty quantification.
An accompanying notebook is available as part of the documentation of CAUTIONER to demonstrate how to perform uncertainty quantification on a regression task.

```
# import required libraries
from cautioner import UncertaintyQuantifier
from matplotlib import pyplot as plt
from data import RegressionData
from data import plot_caliberation, plot_posterior
```

The training data is generated using a helper function not shown here. The training dataset consists of 300 pairs of input and output data. The inputs are randomly generated from a standard normal distribution. The outputs are calculated using the function

$$y = \sin(x)\left(\frac{1}{|x|+1}\right) + \varepsilon$$

, with ε as additional random noise having a standard deviation of 0.04. This design models a non-linear relationship with noise, ideal for evaluating the performance of learning algorithms.

```
train_loader, test_loader =RegressionData.get_data_loaders()
train_input_data, train_output_data =next(iter(train_loader))
test_input_data, test_output_data =next(iter(test_loader))

config ={
"task": "regression",
"weight_decay": 0.01,
"model_type": "heteroscedastic",
"random_walk_step_size" :0.1,
```

```
"temperature" :0.1,
}
```

As mentioned in B.2, to perform UQ with CAUTIONER, one needs to provide a neural network architecture. Though the main features of CAUTIONER are for uncertainty quantification, the current version of the library allows the design of some simple neural network models. This is shown in the listing below. Ideally, the user will provide their own model(s).

```
test_uq =UncertaintyQuantifier(train_loader, config)
# create models and generate samples
models =test_uq.create_models(number_of_processes=3, number_of_epochs=200)
samples =test_uq.generate_samples(models=models, total_number_iterations=10000);
predictions =test_uq.predict(test_input_data, module=models[0], samples=samples)
```

```
# helper method for plotting
plot_posterior(predictions, test_loader, train_loader)
```

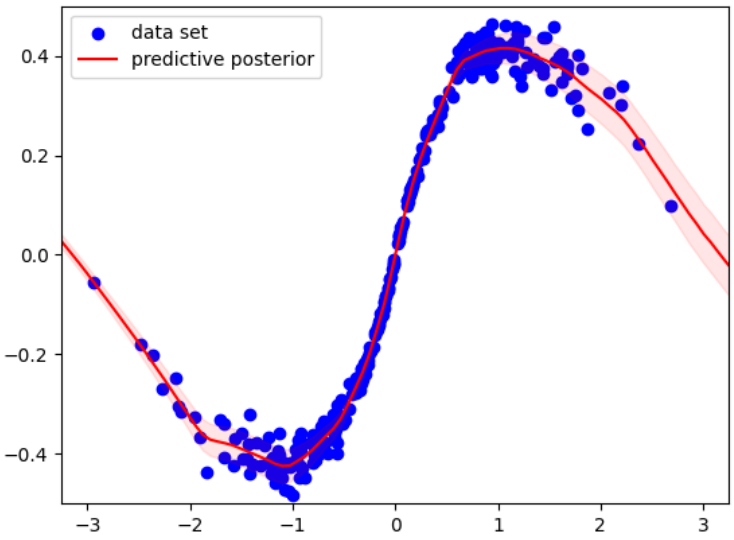

Figure B.2: Posterior predictive distribution computed for non-linear regression use case. Shaded areas indicate predictive means ± one standard deviation.

With CAUTIONER we are able to verify the calibration of the trained model and confirm that BNN's predictive posterior distribution is better calibrated as it takes into account both the aleatoric and epistemic uncertainties. A model is calibrated if, in the long-run, the proportion of forecast × percent credible intervals that succeed in covering the actual value of the predicted quantity turns out to be × percent (Degroot and Fienberg, 2018).

```
# helper method for plotting
plot_caliberation(test_uq, predictions, test_output_data)
```

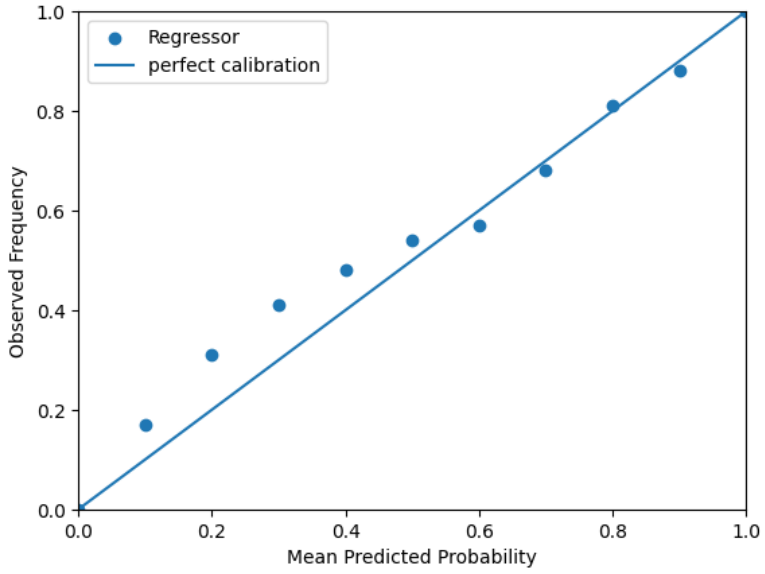

Figure B.3: Reliability diagram for regression task computed on test data.

B.4.2 Classification Use Case

Below, we demonstrate a similar exercise for the case of the classification task using the MNIST handwritten digits dataset. The MNIST (Modified National Institute of Standards and Technology) database is a large database of handwritten figures commonly used for training various image processing systems. It has a training set of 60,000 examples and a test set of 10,000 examples of handwritten digits with a fixed size of 28X28 pixels.
We also provide a notebook as part of the documentation of CAUTIONER with further details of this use case.

```
import numpy as np
import torch
from matplotlib import pyplot as plt

from cautioner import UncertaintyQuantifier
from data import ClassificationData
from data import plot_caliberation, plot_idd_odd_comparison
```

```
train_loader, test_loader =ClassificationData.get_data_loaders()
test_input, test_output =next(iter(test_loader))

config ={
    "task": "classification",
    "weight_decay": 0.01,
    "random_walk_step_size" :0.1,
    "temperature" :0.1,
}
```

```
test_uq =UncertaintyQuantifier(train_loader, config)
models =test_uq.create_models(number_of_processes=3, number_of_epochs=100)
samples =test_uq.generate_samples(models=models, total_number_iterations=1000);
y_pred =test_uq.predict(test_input, models[0], samples)
```

```
# helper method for plotting
plot_caliberation(test_uq, predictions, test_output_data)
```

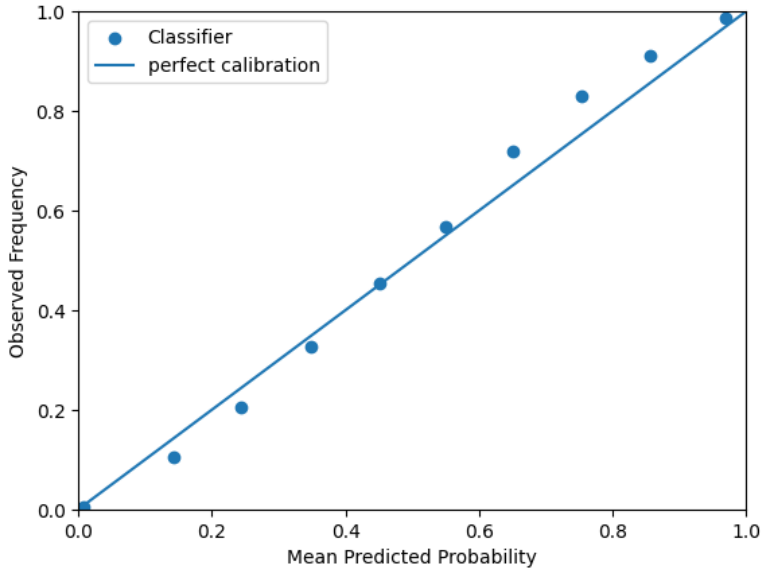

Figure B.4: Reliability diagram for classification task computed on test data.

Following the training of a classifier using the MNIST database (which contains images of handwritten digits), we construct a histogram that displays the distribution of predicted probability labels for two distinct datasets: a test dataset and an out-of-distribution dataset comprised entirely of random noise.
Figures B.5 and B.6 show how the epistemic uncertainty can help detect Out-Of-Distribution data when using a BNN, a feature which is not present when using a standard neural network.

```
bins =[i/10 for i in range(11)]
iod_maximums =[]
for idx, i in enumerate(y_pred):
    iod_maximums.append(float(np.max(np.exp(i))))

ood_maximums_mle =[]
for _ in range(10**3):
    ood_input =torch.randn(1,1,28,28)
    ood_input =ood_input *10
    predicted_probas =test_uq.predict(ood_input, models[0], [models[0].state_dict()])
    ood_maximums_mle.append(float(np.max(np.exp(predicted_probas))))

# helper method for plotting
plot_idd_odd_comparison(iod_maximums, ood_maximums_mle, bins)
```

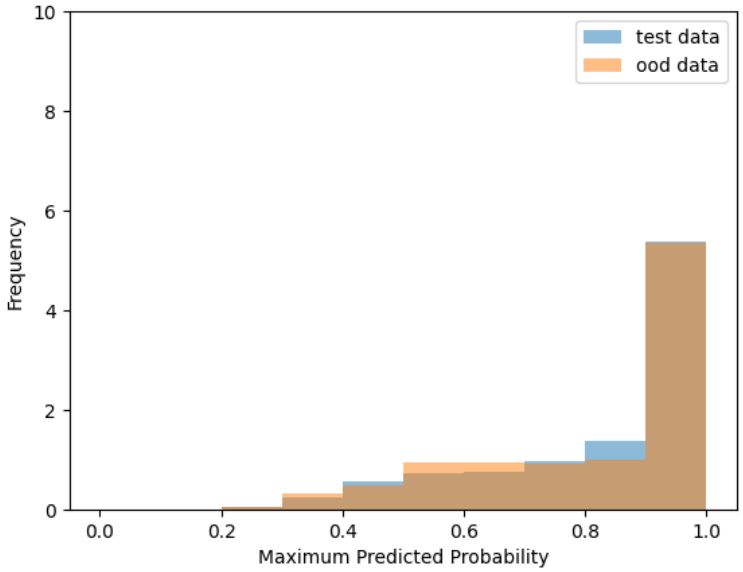

Figure B.5: Reliability diagram for classification task computed on test data.

```
odd_maximums_bnn =[]
for _ in range(10**3):
    ood_input =torch.randn(1,1,28,28)*10
    predicted_probas =test_uq.predict(ood_input, models[0], samples)
    odd_maximums_bnn.append(float(np.max(np.exp(predicted_probas))))
# helper method for plotting
plot_idd_odd_comparison(iod_maximums, odd_maximums_bnn, bins)
```

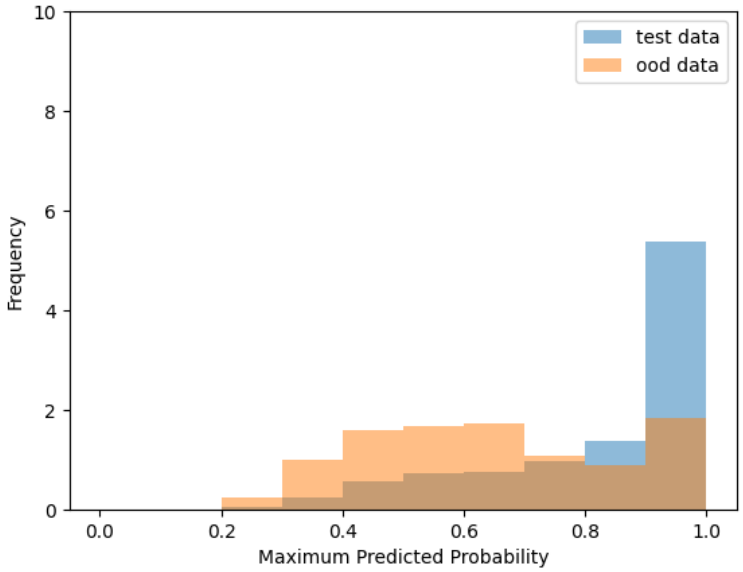

Figure B.6: Reliability diagram for classification task computed on test data.

C. Conclusion

We provide guarantees about a neural network model by quantifying the model's prediction uncertainty. We characterize the discrepancy between physical observations by studying the indeterminacy of the model's predictive parameters during training. We place ourselves in a Bayesian context in order to sampling the a posteriori distribution of neural network parameters. Among sampling techniques, Markov Chain Monte Carlo methods remain a gold standard, but they are very costly in terms of computing resources, which hinders their practical use. We have developed the CAUTIONER software – uncertainty qUanTificatIOn Neural nEtwoRk - that integrates these methodological developments and their applications.
We have demonstrated the application of this methodology to regression and classification supervised learning tasks using CAUTIONER. CAUTIONER was able to provide a calibrated prediction represented by a mean prediction and a standard deviation. This uncertainty contains both the aleatoric uncertainty due to the noise in the data and the epistemic uncertainty due to the limited training data set size.

C.1. Limitations and Perspectives

A major challenge lies in the tool's ability to quantify models with a large number of parameters (deep learning), as well as the ability to calibrate the predictions of these models.
As an outlook, we thus aim to apply CAUTIONER to data and use cases that require bigger models and study the impact of mini-batch in the training of BNNs

Bibliography

Bishop, C. M. (1994). Mixture density networks.

Degroot, M. H. and Fienberg, S. E. (2018). The Comparison and Evaluation of Forecasters. *Journal of the Royal Statistical Society Series D: The Statistician*, 32(1-2):12–22.

Kawasaki, E. and Holzmann, M. (2022). Data subsampling for bayesian neural networks.

Title: Methodological Guideline for CAUTIONER

Keywords: Bayesian Neural Network, Uncertainty Quantification

CAUTIONER provides Bayesian Neural Networks (BNNs) that are able to compute their prediction uncertainty by modelling the distribution over the parameters (weights and biases) given the training data set.

Our partners: