Journal of Machine Learning Research 26 (2025) 1-7 Submitted 12/24; Published 3/25

Lightning UQ Box: Uncertainty Quantification for Neural Networks

Nils Lehmann N.LEHMANN@TUM.DE
Data Science in Earth Observation, Technical University of Munich

Nina Maria Gottschling NINA-MARIA.GOTTSCHLING@DLR.DE
MF-DAS OP - EO Data Science, German Aerospace Center (DLR)

Jakob Gawlikowski JAKOB.GAWLIKOWSKI@DLR.DE
MF-DAS OP - EO Data Science, German Aerospace Center (DLR)

Adam J. Stewart ADAM.STEWART@TUM.DE
Data Science in Earth Observation, Technical University of Munich

Stefan Depeweg STEFAN.DEPEWEG@SIEMENS.COM
Siemens AG

Eric Nalisnick NALISNICK@JHU.EDU
Johns Hopkins University

Editor: Joaquin Vanschoren

Abstract

Although neural networks have shown impressive results in a multitude of application domains, the "black box" nature of deep learning and lack of confidence estimates have led to scepticism, especially in domains like medicine and physics where such estimates are critical. Research on uncertainty quantification (UQ) has helped elucidate the reliability of these models, but existing implementations of these UQ methods are sparse and difficult to reuse. To this end, we introduce Lightning UQ Box, a PyTorch-based Python library for deep learning-based UQ methods powered by PyTorch Lightning. Lightning UQ Box supports classification, regression, semantic segmentation, and pixelwise regression applications, and UQ methods from a variety of theoretical motivations. With this library, we provide an entry point for practitioners new to UQ, as well as easy-to-use components and tools for scalable deep learning applications.

Keywords: Uncertainty Quantification, Bayesian Deep Learning, Conformal Prediction, Deep Learning, PyTorch

1. Introduction

Deep learning is increasingly being applied in a variety of application domains that require decision making under uncertainty. Examples include medical applications like tumor segmentation (Abdullah et al., 2022), Earth observation cases, in particular natural disaster response (Schumann et al., 2016), robotics (Sanket et al., 2023), and healthcare (Seoni et al., 2023). These applications demand reliable predictive uncertainty estimates which neural networks usually do not provide by default. Numerous uncertainty quantification (UQ) approaches for neural networks have been proposed (Gawlikowski et al., 2023). However, to adequately evaluate the efficacy of these methods for various applications, a common modeling framework is necessary to foster the reproducibility of experiments, provide a fair evaluation, and make UQ methods more easily accessible to various research domains. Multiple open-source implementations and frameworks for uncertainty quantification in deep learning are available (Lee et al., 2022; Esposito, 2020; Krishnan et al., 2022; Detommaso et al., 2024; Lafage and Laurent, 2024), often focusing on specific tasks or leaving out specific types of methods, e.g., Bayesian Deep Learning, or without the modularity and flexibility provided by a framework such as PyTorch Lightning. In a recent position paper, Papamarkou et al. (2024) state that "There is a demand for user-friendly software that facilitates the integration of BDL [Bayesian Deep Learning] into various projects". Lightning UQ Box [1] aims to fill this gap, but simultaneously it does not limit itself to Bayesian frameworks but instead includes UQ methods from a diverse set of theoretical underpinnings across current research focuses, for example, conformal prediction (Angelopoulos et al., 2023).

2. Library Design

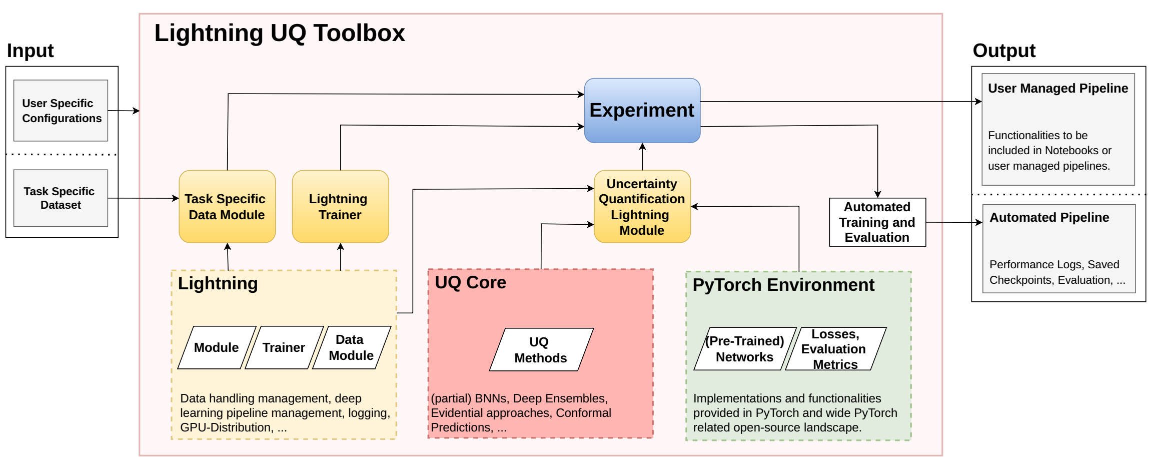

Figure 1: The structure of Lightning UQ Box. The experiments can be built and evaluated at scale or manually tailored to specific use cases. For large experiments at scale, only a dataset and a configuration file have to be provided.

1. Lightning UQ Box GitHub repository and documentation

The PyTorch ecosystem (Paszke et al., 2019) has enabled tremendous progress in various application domains. Lightning UQ Box is built on top of PyTorch Lightning (Falcon, 2019) with a focus on reproducibility and scalability of deep learning experiments under a modular design. PyTorch Lightning asks the user to organize code in a more structured way regarding training and evaluation steps, additionally removes boilerplate code, and separates dataset and model logic under different modules. To this end, every supported UQ method and task combination in Lightning UQ Box is a `LightningModule` that can leverage the training capabilities of PyTorch Lightning, such as automatic logging, mixed precision training, multi-GPU training, etc. to run experiments. The modular design is visualized in Figure 1. Through its enforced code organization, a `LightningModule` clearly defines what a given UQ method does during training, validation, and prediction and is easy to follow in the code files. This can more clearly highlight differences and commonalities not just between different methods, but also among different prediction tasks for any particular method. The modular design allows a straightforward extension to new tasks or new UQ methods that arise in this active research field and further simplifies community contributions.

2.1 Feature Highlights

Breadth of Methods Uncertainty Quantification in neural networks has been approached from various theoretical underpinnings (Gawlikowski et al., 2023) and Lightning UQ Box reflects this through the variety of methods it supports from the various categorized perspectives, such as Bayesian, conformal prediction, evidential deep learning, generative models, or post-hoc calibration methods.[2]

Backbone Agnostic A core design principle of Lightning UQ Box is that the implemented models are "backbone" agnostic, meaning that users can bring their custom PyTorch architecture or pretrained models from libraries like timm (Wightman, 2019), on top of which the selected UQ method will be applied without the user having to customize their model for different UQ methods. Selected model parts can also be frozen during training, which has potential applications of equipping large scale foundation models with UQ, for example through last-layer UQ fine tuning (Papamarkou et al., 2024).

Minimization of Boilerplate Code The modular design of PyTorch Lightning significantly reduces the amount of boilerplate code and allows fast experiment setup and iteration. The UQ Box can further seamlessly be used with existing Lightning pipelines.

Modern Bayesian Methods A common criticism of BNNs is that they are expensive to train and do not scale to large data problems (Papamarkou et al., 2024). The supported Bayesian UQ methods are made scalable to larger data regimes with partially stochastic variants (Sharma et al., 2023), that are also supported for Laplace (Daxberger et al.,

2. See the documentation page or repository README for a complete overview of supported methods

2021a), SWAG (Maddox et al., 2019), and MC-Dropout (Gal and Ghahramani, 2016). This allows for flexible hybrid approaches like last-layer or subnetwork approximations (Daxberger et al., 2021b). Furthermore, we support Deep Kernel Learning (DKL) (Wilson et al., 2016), Spectral-Normalized Gaussian Processes (SNGP) (Liu et al., 2020), and Deep Deterministic Uncertainty (DDU) (Van Amersfoort et al., 2020) as hybrid approaches.

Reproducibility Several works postulated that machine learning finds itself in a reproducibility crisis across application domains (Kapoor and Narayanan, 2023). In a related article in life sciences, Heil et al. (2021) state “The gold standard for reproducibility requires the entire analysis to be reproducible with a single command”. Lightning UQ Box works towards this goal by supporting configuration of experiments with simple configuration files, as well as the Lightning command line interface (CLI). For example, the required configurations to run a partially stochastic BNN or Deep Kernel Learning model based on the timm library ResNet18 implementation on the EuroSAT dataset from torchgeo is shown in Figure 2. For more examples, see the Lightning-UQ-Box documentation page.

```yaml
uq_method:
  _target_: BNN_VI_ELBO_Classification
  model:
    _target_: timm.create_model
    model_name: resnet18
    in_chans: 13
    num_classes: 10
  criterion:
    _target_: torch.nn.CrossEntropyLoss
  num_mc_samples_train: 3
  num_mc_samples_test: 25
  stochastic_module_names: ['layer4.1.conv1',
       'layer4.1.conv2', 'fc']

datamodule:
  _target_: torchgeo.datamodules.EuroSATDataModule
  batch_size: 64
  download: True

trainer:
  _target_: lightning.Trainer
  max_epochs: 40
```

```yaml
model:
  _target_: DKLClassification
  feature_extractor:
    _target_: timm.create_model
    model_name: resnet18
    num_classes: 8 # num latent features
  gp_kernel: "RBF"
  n_inducing_points: 5
  input_size: 64
  num_classes: 10

datamodule:
  _target_: torchgeo.datamodules.EuroSATDataModule
  batch_size: 64
  download: True

trainer:
  _target_: lightning.Trainer
  max_epochs: 40
  gradient_clip_val: 1.0
  accumulate_grad_batches: 2
```

Figure 2: Example YAML files that configure (left) a partially stochastic BNN based on a timm ResNet18 model implementation and (right) the same ResNet18 as Deep Kernel Learning model for training on the EuroSAT classification dataset from the geospatial PyTorch domain library TorchGeo (Stewart et al., 2022).

Introduction and Tutorials A great emphasis has been placed on providing an entry point to UQ for practitioners from various domains. To this end, we include more than 30 tutorials as Jupyter Notebooks (Kluyver et al., 2016) in the accompanying documentation page that explain the theoretical framework and demonstrate their application to common toy datasets.

3. Conclusion

We introduce Lightning UQ Box, a PyTorch framework built on PyTorch Lightning for UQ in deep learning. By offering a comprehensive set of methods across theoretical frameworks that can be scaled to common problems from different domains, the toolbox allows practitioners to easily and fairly compare these approaches. Together with detailed tutorials, we hope that the library can provide an entry point for people interested in UQ, support large scale experiments across methods and perhaps foster new research ideas.

References

Abdullah A. Abdullah, Masoud M. Hassan, and Yaseen T. Mustafa. A review on Bayesian deep learning in healthcare: Applications and challenges. *IEEE Access*, 10:36538–36562, 2022.

Anastasios N. Angelopoulos, Stephen Bates, et al. Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591, 2023.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless Bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021a.

Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, pages 2510–2521. PMLR, 2021b.

Gianluca Detommaso, Alberto Gasparin, Michele Donini, Matthias Seeger, Andrew Gordon Wilson, and Cedric Archambeau. Fortuna: A library for uncertainty quantification in deep learning. *Journal of Machine Learning Research*, 25(238):1–7, 2024.

Piero Esposito. BLiTZ - Bayesian Layers in Torch Zoo (a Bayesian deep learing library for Torch). `https://github.com/piEsposito/blitz-bayesian-deep-learning/`, 2020.

William A. Falcon. PyTorch Lightning. `https://github.com/Lightning-AI/pytorch-lightning`, 2019.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56 (Suppl 1):1513–1589, 2023.

Benjamin J. Heil, Michael M. Hoffman, Florian Markowetz, Su-In Lee, Casey S. Greene, and Stephanie C. Hicks. Reproducibility standards for machine learning in the life sciences. *Nature Methods*, 18(10):1132–1135, 2021.

Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9), 2023.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks - a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, Netherlands, 2016. IOS Press. URL `https://eprints.soton.ac.uk/403913/`.

Ranganath Krishnan, Pi Esposito, and Mahesh Subedar. Bayesian-Torch: Bayesian neural network layers for uncertainty estimation. `https://github.com/IntelLabs/bayesian-torch`, January 2022. URL `https://doi.org/10.5281/zenodo.5908307`.

Adrian Lafage and Olivier Laurent. Torch Uncertainty. `https://github.com/ENSTA-U2IS-AI/torch-uncertainty`, 2024.

Sungyoon Lee, Hoki Kim, and Jaewook Lee. GradDiv: Adversarial robustness of randomized neural networks via gradient diversity regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in neural information processing systems*, 33: 7498–7512, 2020.

Wesley J. Maddox, Pavel Izmailov, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, Aliaksandr Hubin, et al. Position paper: Bayesian deep learning in the age of large-scale AI. *arXiv preprint arXiv:2402.00809*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. Ajna: Generalized deep uncertainty for minimal perception on parsimonious robots. *Science Robotics*, 8(81):eadd5139, 2023.

Guy Schumann, Dalia Kirschbaum, Eric Anderson, and Kashif Rashid. Role of Earth observation data in disaster response and recovery: From science to capacity building. *Earth Science Satellite Applications: Current and Future Prospects*, pages 119–146, 2016.

Silvia Seoni, Vicnesh Jahmunah, Massimo Salvi, Prabal Datta Barua, Filippo Molinari, and U Rajendra Acharya. Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013–2023). *Computers in Biology and Medicine*, page 107441, 2023.

Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do Bayesian neural networks need to be fully stochastic? In *International Conference on Artificial Intelligence and Statistics*, pages 7694–7722. PMLR, 2023.

Adam J Stewart, Caleb Robinson, Isaac A Corley, Anthony Ortiz, Juan M Lavista Ferres, and Arindam Banerjee. Torchgeo: deep learning with geospatial data. In *Proceedings of the 30th international conference on advances in geographic information systems*, pages 1–12, 2022.

Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR, 2020.

Ross Wightman. PyTorch Image Models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378. PMLR, 2016.