DeepUQ: Assessing the Aleatoric Uncertainties from two Deep Learning Methods

Rebecca Nevin[1]
rnevin@fnal.gov

Aleksandra Ćiprijanović[1,2]
aleksand@fnal.gov

Brian D. Nord[1,2,3]
nord@fnal.gov

[1]Fermi National Accelerator Laboratory, Batavia, IL 60510
[2]Department of Astronomy and Astrophysics, University of Chicago, Chicago, IL 60637
[3]Kavli Institute for Cosmological Physics, University of Chicago, Chicago, IL 60637

Abstract

Assessing the quality of aleatoric uncertainty estimates from uncertainty quantification (UQ) deep learning methods is important in scientific contexts, where uncertainty is physically meaningful and important to characterize and interpret exactly. We systematically compare aleatoric uncertainty measured by two UQ techniques, Deep Ensembles (DE) and Deep Evidential Regression (DER). Our method focuses on both zero-dimensional (0D) and two-dimensional (2D) data, to explore how the UQ methods function for different data dimensionalities. We investigate uncertainty injected on the input and output variables and include a method to propagate uncertainty in the case of input uncertainty so that we can compare the predicted aleatoric uncertainty to the known values. We experiment with three levels of noise. The aleatoric uncertainty predicted across all models and experiments scales with the injected noise level. However, the predicted uncertainty is miscalibrated to $\mathrm{std}(\sigma_{\mathrm{al}})$ with the true uncertainty for half of the DE experiments and almost all of the DER experiments. The predicted uncertainty is the least accurate for both UQ methods for the 2D input uncertainty experiment and the high-noise level. While these results do not apply to more complex data, they highlight that further research on post-facto calibration for these methods would be beneficial, particularly for high-noise and high-dimensional settings.

1 Introduction

Physically and statistically interpretable uncertainties are critical for applications in science and industry. Uncertainty quantification (UQ) in deep neural networks has gained significant attention, with recent work exploring taxonomies of uncertainties, including domain, epistemic, and aleatoric uncertainties, e.g., [4, 5, 10]. Aleatoric uncertainty, σ_{al}, is significant because, unlike epistemic uncertainty, it is not a result of model limitations but rather an inherent property of the data. In many cases, aleatoric uncertainty is exactly known because it is produced by a well-understood physical process, allowing us to anticipate not only its expected amplitude but also its distributional characteristics. For instance, in astrophysics, the Poisson distribution[1] characterizes 'shot' or photon noise, while the Normal distribution characterizes read and other forms of thermal or electronic noise. Developing a coherent framework for benchmarking aleatoric uncertainty estimates from deep

[1]The Poisson distribution can be approximated by a Gaussian when the photon rate λ is large.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

networks and assessing their calibration is needed to ensure that the predicted aleatoric uncertainty aligns with our scientific expectations.

UQ methods broadly fall under the categories of Bayesian methods (e.g., Bayesian Neural Networks (BNNs) [15, 24, 19]), Bayesian model averaging (e.g., Deep Ensembles [14], MC Dropout [9]), and Evidential Deep Learning [26] (e.g., Deep Evidential Regression [1, 16]). In addition, a class of methods for uncertainty calibration [18] exist separately in the statistical literature and have recently gained popularity as post-processing tools (e.g., conformal prediction [2]). Other work has explored formalized comparison of UQ methods (e.g., [6, 21, 3, 25, 7]). [21, 25] compare aspects of predictive uncertainty distributions, and [7] present an uncertainty toolbox for comparing predictive uncertainties; all of these methods do so without access to true uncertainty values. Of the few studies testing the exact calibration of predicted uncertainties [6, 3], some do not vary data dimensionalities or uncertainty injection types [6], while others vary these factors but do not report mean aleatoric uncertainty or propagate input uncertainty, preventing direct comparison to expected values [3].

Quantifying how noise on the input variable affects the predictions of aleatoric uncertainty on the output variable from deep learning methods is of critical importance, especially in computer vision, and has not yet received much attention in the literature (e.g., [20, 29]). The bulk of previous work on aleatoric uncertainty has focused mostly on assessing the predicted aleatoric uncertainty on the output variable y via injecting uncertainty directly on y (for a review, see [11]). Recently, the statistical field of input uncertainty has intersected with the deep learning literature under the umbrella of UQ (for a review, see [27]). Experiments have focused on propagating input uncertainty through a neural network for regression using a Laplace Approximation [29] as well as through a Taylor series expansion and Monte Carlo sampling approach with a multi-layer perception [27]. Assessing input uncertainty is inherently more complex, requiring tractable functional relationships between input and output variables when propagating the uncertainty onto the output variables.

We present a study of regression on tabular (0D) and imaging (2D) data that investigates aleatoric uncertainty for cases where uncertainty is injected on either the input x or the output y variables, providing a more comprehensive understanding of aleatoric uncertainty in regression tasks. By injecting uncertainty onto the input variable and propagating it to the output variable, we can assess the exact calibration of the predicted uncertainty estimate. We design a set of desiderata for how the predicted aleatoric uncertainty should behave: i) the predicted uncertainty should scale with the injected uncertainty; ii) the aleatoric uncertainty should be well-calibrated (within $\mathrm{std}(\sigma_{\mathrm{al}})$ of the true uncertainty value); and iii) these desiderata should hold for both data dimensionalities and both uncertainty injection types (input and output). We do this all for a very simple set of experiments; we caution the reader against applying the conclusions here to all types of data, including real-world datasets.

2 Deep Learning Methods for Predicting Uncertainty

Deep Ensemble: Ensembling mean-variance estimation networks (MVEs) produces a set of predicted mean and variance values – 'Deep Ensembles' [DE; 14]. We build upon the DE framework from [14] incorporating several modifications inspired by previous work, including a softplus activation for the σ^2 output neuron to enforce a positive output value and a β modification to the loss function, as introduced in [22]. The modified loss function we use is: $\mathcal{L}_{\beta-\mathrm{NLL}} = \frac{1}{N}\sum_{i=0}^{N}\left[\sigma^{2\beta}(x_i)\left[\frac{1}{2}\log\sigma^2(x_i) + \frac{(y_i-\mu(x_i))^2}{2\sigma^2(x_i)} + \mathrm{C}\right]\right]$. The β-modified loss function helps ensure convergence of the network predictions, avoiding a commonly observed problem in MVEs, where the variance artificially enlarges resulting in a poor estimate of the mean. We use a β value of 0.5, which is recommended by [22], and described in more depth in Appendix D. The aleatoric uncertainty is the mean of the predicted standard deviations for the ensemble of $K = 10$ models: $\sigma_{\mathrm{al}} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\sigma_k^2}$, where σ_k^2 is the variance predicted by the k-th network.

Deep Evidential Regression: Deep Evidential Regression (DER) estimates aleatoric uncertainties via evidential distributions that are directly incorporated into the loss function [1]. Instead of requiring an ensemble of networks, it places evidential priors over a Gaussian likelihood function, and the network is trained to learn the hyperparameters of the evidential distribution.

We use the normal-inverse-gamma (NIG) loss from [16], which includes an additional term weighted by the width of the $t-$distribution. This formulation improves the efficiency and accuracy of training:

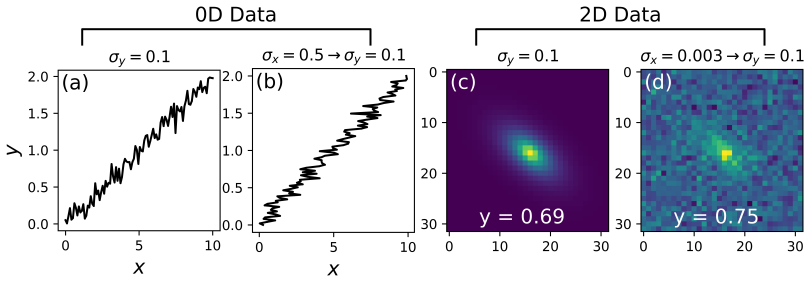

Figure 1: Data examples for the four experimental designs: a) output uncertainty for the 0D linear regression, b) input uncertainty for 0D, c) output uncertainty for the 2D imaging data, and d) input uncertainty for the 2D data. The noise level is high for all panels: $\sigma_y = 0.1$. For the case of the input uncertainty panels (b and d), the uncertainty is injected on the input variable, σ_x, and uncertainty propagation results in a σ_y value of 0.1.

$\mathcal{L}_{\text{NIG}} = \frac{1}{N}\sum_{i=1}^{N}\left[-\log L_i^{\text{NIG}} + \lambda|\frac{y_i-\gamma}{w_{\text{St}}}|\Phi\right]$, where λ is a tunable regularization hyperparameter (we use $\lambda = 0.01$ as in [16]), w_{St} is the width of the $t-$distribution, and Φ is the total evidence $\Phi = 2\nu + \alpha$. For a full derivation, see [16], which we also summarize in Appendix E.

We use the modified definitions of aleatoric uncertainty from [16]. The aleatoric uncertainty is the width of the $t-$distribution, which resembles a normal distribution: $\sigma_{\text{al}} \equiv w_{\text{St}} = \sqrt{\frac{\beta(1+\nu)}{\alpha\nu}}$.

Experimental Design: Figure 1 illustrates the experimental setup for an example of high-noise data.

The 0D data are from a simple linear regression model: $y = mx$. The values of x are linearly spaced between 0 and 10. The data are designed so that the y distribution is uniform, $\mathcal{U}[0, 2]$. The training/validation/test set size is 90k/10k/10k. To create data for the output uncertainty experiments, we inject noise directly on the prediction or label, such that $y_{\text{noisy}} = y + \mathcal{N}(0, \sigma_y^2)$. The models are trained on (x, y_{noisy}) pairs. In the input uncertainty experiments, we inject noise via the input variable $x_{\text{noisy}} = x + \mathcal{N}(0, \sigma_x^2)$ and the models are trained on (x_{noisy}, y) pairs.

For the 2D data, we use the software package `DeepBench` ([28] in prep) to generate 32×32-pixel galaxy images by varying the Sérsic radius, amplitude, and position angle within ranges $[0, 0.01]$, $[1, 10]$, and $[-1.5, 1.5]$, respectively. We are motivated by real-world uncertainty examples in astronomical imaging to generate a 2D dataset in addition to the 0D tabular dataset. The output variable y is the sum of the pixel values. The dataset is designed to be uniform in y over a range $[0, 2]$ for a training/validation/test set size of 4500/500/500. For the output uncertainty experiments, we inject a random normal distribution with mean zero and standard deviation σ_y directly on y. For the input uncertainty experiments, we inject a random normal distribution with mean zero and standard deviation σ_x on each pixel, which results in a normal distribution in σ_y after propagation. We use a random normal distribution because the DE and the DER methods assume that the output variable is distributed as a random normal distribution.

We distinguish between the predicted aleatoric uncertainty from the methods, σ_{al}, which is measured as an uncertainty on the output variable, and the true uncertainty on the output variable, σ_y. The true uncertainty on the output variable is either directly known in the case of the output uncertainty experiments or is known through uncertainty propagation for the input uncertainty experiments. The true output uncertainty has low, medium, and high values: $\sigma_y \in [0.01, 0.05, 0.1]$. These noise levels are chosen such that the high noise value datasets have an uncertainty level that is on average 10% of the output variable y. For input uncertainty, we inject noise σ_x and calculate the expected σ_y uncertainty value via standard rules of error propagation described in Appendix C. The relationship between σ_y and σ_x for the 0D data is $\sigma_y = |m|\sigma_x$, where m is the slope of the line, and the relationship for the 2D data is $\sigma_y = 32\sigma_x$.

UQ Model Architectures: We use the `DeepUQ` package to define the model architecture and perform our experiments. We also present the `DeepUQ-neurIPS-WS-2024` repository as an accompaniment to the paper, with notebook examples of how to reproduce the exact models, figures, and tables in this paper. Both UQ methods use the same fully connected layer network architecture, which is two hidden layers of 64 neurons each. The hidden layer neurons utilize a ReLu activation function. For the 0D experiments, the networks use two input neurons (the m value and the x value for a single point), while the 2D experiments use the 32×32 pixel input. We use five convolutional layers for the

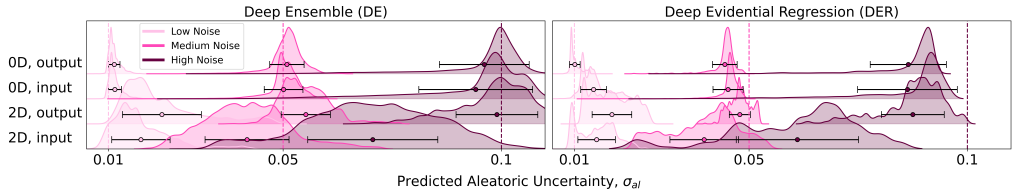

Figure 2: Distribution of predicted σ_{al} values. The circular point for each distribution is the sample mean and the black error bars show the $\text{std}(\sigma_{\text{al}})$ confidence range, the standard deviation of the σ_{al} distribution. The vertical dashed lines demonstrate the true output uncertainty values, σ_y, which vary by noise level. The light pink, medium pink, and purple distributions correspond to the low-, medium-, and high-noise models.

2D networks before the fully connected layers: the architecture is a series of convolutional filters that increase in depth and decrease in size further into the neural network. These layers are interspersed with 2D pooling. For all models, we use the Adam optimizer with an initial learning rate of 0.001.

For the DE method, two output neurons correspond to μ and σ^2, such that $y \sim \mathcal{N}(\mu, \sigma^2)$. For the DER method, four output neurons correspond to the parameters γ, ν, α, and β. The output neurons utilize a softplus activation if a positive value is required (i.e., for σ^2 for the DE, and for ν, α, and β for the DER) and a linear activation for all other outputs (i.e., for μ for the DE, and γ for the DER). For more details of the software package `DeepUQ`, see Appendix F.

3 Results

We run both UQ methods for all four experimental setups and all three noise levels and find that all models converge, with final mean-square error (MSE) values at epoch 99 ranging from 0.0001 to 0.01 (Appendix G). Furthermore, the DE and DER methods have comparable final MSE values for each noise level, and the NIG loss and $\beta-$NLL loss values are similar for each UQ method across experiments for a fixed noise level. This indicates that the different methods of uncertainty injection and the different data dimensionalities are all equally adequately learning to predict the relationship between input and output values. To test desideratum (i), we display the distribution of predicted aleatoric uncertainties for the test set for the different noise levels in Figure 2. We use the standard deviation of the predicted uncertainty values, $\text{std}(\sigma_{\text{al}})$, to assess desiderata (ii) and (iii): whether the predicted uncertainty value σ_{al} is consistent with the true value σ_y for all four experiments and all three noise levels.

4 Discussion

The predicted aleatoric uncertainty increases proportionately with the true injected uncertainty. The models are sensitive to the true uncertainty, which bolsters confidence in these UQ methods. It further confirms the findings of [8], where the automated Deep Ensemble method (AutoDEUQ) produces predicted aleatoric uncertainty that scales with uncertainty injected on the output variable. Additionally, [3] find that the predicted aleatoric uncertainty from DER models increases for 0D and 2D experiments where uncertainty is increased on both the input and output variables.

When we require that the predicted uncertainty falls within $\text{std}(\sigma_{\text{al}})$ of the true uncertainty to be considered 'well calibrated', we find that only seven of the twelve DE experiments and two of the twelve of the DER experiments satisfy this requirement. Furthermore, for both methods, the degree of miscalibration depends on the experiment's dimensionality and the type of uncertainty injection. Desiderata (ii) and (iii) are therefore violated for both experiments.

For the DE method, the 0D experiments are calibrated for the medium- and high-noise models (Figure 2, left). Both of the 0D low-noise experiments slightly overestimate the uncertainty. Overall, the uncertainty estimates are the least calibrated for the most complex experimental setup (input, 2D; bottom row). For the DER method (Figure 2, right), the majority of experiments across all noise levels produce miscalibrated uncertainty estimates, over-estimating (low-noise) and under-estimating (medium- and high-noise) the predictive uncertainty. The exceptions are the 0D output low-noise model and the 2D output medium-noise model, which are calibrated. For both methods, the most

inaccurate experiment is the 2D input uncertainty experiment. Within this experiment, the high-noise models are the least calibrated.

In [3], the authors perform regression experiments for a DER model; the network is well calibrated for the 0D dataset but underestimates the true uncertainty for the 2D dataset injected with output uncertainty. They suggest a recalibration step in more complex domains (i.e., 2D data) to ameliorate this concern, where recalibration involves training an auxiliary isotonic regression model so that the predicted uncertainties are calibrated to the cumulative density function of the data. We expand upon the work of [3] to also include uncertainty injected on the input variable. For both types of uncertainty injection, we identify a miscalibration in aleatoric uncertainty for the DER models and identify that the effect is worse for high-dimensionality and high-noise experiments where the uncertainty injection is on the input variable.

5 Conclusions and Outlook

We explore aleatoric uncertainties predicted by two deep learning UQ approaches — Deep Ensembles (DE) and Deep Evidential Regression (DER). We compare the aleatoric uncertainty predictions of these two methods to the true uncertainty for four different experiments and three noise levels. Both methods meet desideratum (i): the aleatoric uncertainties scale with the injected uncertainty.

However, for our experiments, the methods both fail to meet desiderata (ii) and (iii), that the predicted aleatoric uncertainties be well-calibrated, i.e., consistent to $\mathrm{std}(\sigma_{\mathrm{al}})$ with the true uncertainties, and that they meet this requirement across all experiments. Notably, most DER experiments underestimate the uncertainty for medium- and high-noise models and overestimate the uncertainty for low-noise models. The DE experiments deviate mostly for the more complex 2D input uncertainty experiments. The predicted uncertainties are the least accurate for both methods for the 2D, input uncertainty, and high-noise experiments. While these observations do not imply inherent deficiencies in DE and DER, they highlight that further research would be beneficial to assess whether these methods require post-facto calibration, particularly for high-noise and high-dimensional settings.

Some limitations of our work are: Our conclusions apply only to our toy datasets. We do not demonstrate performance of these methods on real world datasets with higher complexity. Additionally, our conclusions apply only to homoskedastic noise generated from a Gaussian distribution; expanding this to non-Gaussian distributions and heteroskedastic noise is a direction for future work.

References

[1] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep Evidential Regression. *arXiv e-prints*, page arXiv:1910.02600, October 2019.

[2] Anastasios N. Angelopoulos and Stephen Bates. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *arXiv e-prints*, page arXiv:2107.07511, July 2021.

[3] Lennart Bramlage, Michelle Karg, and Cristóbal Curio. Plausible uncertainties for human pose regression. In *EEE/CVF International Conference on Computer Vision (ICCV)*, pages 15087–15096, 10 2023.

[4] A. Brando. *Aleatoric uncertainty modelling in regression problems using deep learning*. PhD thesis, Universitat de Barcelona, 2022.

[5] Axel Brando, Isabel Serra, Enrico Mezzetti, Francisco Javier Cazorla Almeida, and Jaume Abella Ferrer. Standardizing the probabilistic sources of uncertainty for the sake of safety deep learning. In *Proceedings of the Workshop on Artificial Intelligence Safety 2023 (SafeAI 2023) co-located with the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023): Washington DC, USA, February 13-14, 2023.*, volume 3381. CEUR Workshop Proceedings, 2023.

[6] João Caldeira and Brian Nord. Deeply Uncertain: Comparing Methods of Uncertainty Quantification in Deep Learning Algorithms. *arXiv e-prints*, page arXiv:2004.10710, April 2020.

[7] Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. Uncertainty Toolbox: an Open-Source Library for Assessing, Visualizing, and Improving Uncertainty Quantification. *arXiv e-prints*, page arXiv:2109.10254, September 2021.

[8] Romain Egele, Romit Maulik, Krishnan Raghavan, Bethany Lusch, Isabelle Guyon, and Prasanna Balaprakash. AutoDEUQ: Automated Deep Ensemble with Uncertainty Quantification. *arXiv e-prints*, page arXiv:2110.13511, October 2021.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv e-prints*, page arXiv:1506.02142, June 2015.

[10] Yarin Gal, Petros Koumoutsakos, François Lanusse, et al. Bayesian uncertainty quantification for machine-learned models in physics. *Nature Reviews Physics*, 4:573–577, 2022.

[11] Eyke Hüllermeier and Willem Waegeman. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *arXiv e-prints*, page arXiv:1910.09457, October 2019.

[12] Alex Kendall and Yarin Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *arXiv e-prints*, page arXiv:1703.04977, March 2017.

[13] Harry H. Ku. Notes on the use of propagation of error formulas. 2010.

[14] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *arXiv e-prints*, page arXiv:1612.01474, December 2016.

[15] Jouko Lampinen and Aki Vehtari. Bayesian techniques for neural networks — review and case studies. In *2000 10th European Signal Processing Conference*, pages 1–8, 2000.

[16] Nis Meinert, Jakob Gawlikowski, and Alexander Lavin. The Unreasonable Effectiveness of Deep Evidential Regression. *arXiv e-prints*, page arXiv:2205.10060, May 2022.

[17] D.A. Nix and A.S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60 vol.1, 1994.

[18] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. Measuring Calibration in Deep Learning. *arXiv e-prints*, page arXiv:1904.01685, April 2019.

[19] Nicholas G. Polson and Vadim Sokolov. Deep Learning: A Bayesian Perspective. *Bayesian Analysis*, 12(4):1275 – 1304, 2017.

[20] Natália V. N. Rodrigues, L. Raul Abramo, and Nina S. T. Hirata. The information of attribute uncertainties: what convolutional neural networks can learn about errors in input data. *Machine Learning: Science and Technology*, 4(4):045019, December 2023.

[21] Gabriele Scalia, Colin A. Grambow, Barbara Pernici, Yi-Pei Li, and William H. Green. Evaluating Scalable Uncertainty Estimation Methods for DNN-Based Molecular Property Prediction. *arXiv e-prints*, page arXiv:1910.03127, October 2019.

[22] Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks. *arXiv e-prints*, page arXiv:2203.09168, March 2022.

[23] Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks. *arXiv e-prints*, page arXiv:2203.09168, March 2022.

[24] D. M. Titterington. Bayesian Methods for Neural Networks and Related Models. *Statistical Science*, 19(1):128 – 139, 2004.

[25] Kevin Tran, Willie Neiswanger, Junwoong Yoon, Qingyang Zhang, Eric Xing, and Zachary W. Ulissi. Methods for comparing uncertainty quantifications for material property predictions. *arXiv e-prints*, page arXiv:1912.10066, December 2019.

[26] Dennis Ulmer, Christian Hardmeier, and Jes Frellsen. Prior and Posterior Networks: A Survey on Evidential Deep Learning Methods For Uncertainty Estimation. *arXiv e-prints*, page arXiv:2110.03051, October 2021.

[27] Matias Valdenegro-Toro, Ivo Pascal de Jong, and Marco Zullich. Unified Uncertainties: Combining Input, Data and Model Uncertainty into a Single Formulation. *arXiv e-prints*, page arXiv:2406.18787, June 2024.

[28] M. Voetberg, Ashia Livaudais, Becky Nevin, Omari Paul, and Brian Nord. Deepbench: A simulation package for physical benchmarking data. Submitted to Journal of Open Source Software, 2024. Manuscript submitted for publication.

[29] W. Wright, Guillaume Ramage, Dan Cornford, and Ian Nabney. Neural network modelling with input uncertainty: Theory and application. *VLSI Signal Processing*, 26:169–188, 08 2000.

Acknowledgments and Disclosure of Funding

A Funding

We acknowledge the Deep Skies Lab as a community of multi-domain experts and collaborators who've facilitated an environment of open discussion, idea generation, and collaboration. This community was important for the development of this project.

Work supported by the Fermi National Accelerator Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

This material is based upon work supported by the Department of Energy under grant No FNAL-LDRD- L2021-004.

B Author Contributions

Nevin: Conceptualization, Methodology, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing

Ćiprijanović: Conceptualization, Methodology, Formal analysis, Writing - Review & Editing, Supervision, Project administration

Nord: Conceptualization, Methodology, Formal analysis, Resources, Writing - Original Draft, Writing - Review & Editing, Supervision, Project administration, Funding acquisition

We thank the following colleagues for their insights and discussions during the development of this work: Sreevani Jarugula.

C Uncertainty propagation

Here we describe our process for propagating uncertainty injected on the input variable σ_x onto uncertainty on the output variable σ_y. For a generic function $y = f(x_1, x_2, ...x_N)$, where y is the dependent variable and x_1, x_2, and so on are the independent variables, the standard deviation of y, σ_y, can be written in terms of uncertainty on the x variables [13]:

$$\sigma_y = \sqrt{(\frac{\partial f}{\partial x_1})^2 \sigma_{x_1}^2 + (\frac{\partial f}{\partial x_2})^2 \sigma_{x_2}^2 + 2(\frac{\partial f}{\partial x_1})(\frac{\partial f}{\partial x_2})\sigma_{x_1 x_2}}, \tag{1}$$

where the final covariance term ($\sigma_{x_1 x_2}$) can be dropped when the correlation between uncertainty terms is negligible, as is the case for both of our data dimensionalities.

For the 0D linear regression case, $y = mx$, the partial derivative only exists relative to x, which has associated uncertainty, so this equation reduces to:

$$\sigma_y = |m|\sigma_x. \tag{2}$$

For the case of the 2D image noise injection, we inject a standard normal value for each pixel and calculate the predicted value y as a sum of all pixel values. The partial derivative terms are all equal to 1 because this is a summation. Since we inject the same value of σ for all pixels, the formula then becomes:

$$\sigma_y = \sqrt{\sum_{i=1}^{N} \sigma_{x_i}^2} = 32\sigma_x \tag{3}$$

where i is the index of all (N) pixels, and the images are 32×32 pixels.

D Deep Ensembles

A common approach for quantifying aleatoric uncertainty in regression tasks with deep neural networks is to assume that the regression output y follows a distribution and to predict the parameters of this distribution. One standard technique is to assume that the errors are heteroskedastic[2] and to model the distribution of y as a Gaussian parameterized by mean μ and variance σ^2, where the predicted values $y_i \sim \mathcal{N}(\mu(x_i), \sigma^2(x_i))$. The model is trained using maximum likelihood estimation by minimizing the negative log likelihood loss under the training set distribution $p(X, Y)$:

$$\begin{aligned} \mathcal{L}_{\text{NLL}} &= -\log p(Y|X) \\ &= \frac{1}{N}\sum_{i=0}^{N}\left[\frac{1}{2}\log \sigma^2(x_i) + \frac{(y_i - \mu(x_i))^2}{2\sigma^2(x_i)} + \text{C}\right], \end{aligned} \tag{4}$$

where $\mu(x_i)$ and $\sigma^2(x_i)$ are the model outputs for each training data point using the model with the optimal set of internal parameters. This technique is known as mean-variance estimation [MVE; 17, 12].

We use a modified loss function for training, known as the β-NLL loss. This loss is proposed by [23] as a means for avoiding a commonly observed problem in MVEs, where the variance artificially enlarges resulting in a poor estimate of the mean. The β parameter helps ensure convergence of the network predictions for $\mu(x_i)$ and $\sigma^2(x_i)$:

$$\mathcal{L}_{\beta-\text{NLL}} = \frac{1}{N}\sum_{i=0}^{N}\left[\sigma^{2\beta}(x_i)\left[\frac{1}{2}\log \sigma^2(x_i) + \frac{(y_i - \mu(x_i))^2}{2\sigma^2(x_i)} + \text{C}\right]\right], \tag{5}$$

[2]The data we generate are homoskedastic, with a constant σ^2 for each noise level, while the Gaussian model in MVE is heteroskedastic, allowing the predicted σ^2 value to vary across points. These experiments, where we test MVE with homoskedastic data, do not break the model's assumption. Instead, we ensure that the model can still accurately predict constant uncertainty when the true distribution is homoskedastic. We are particularly interested here in the calibration of the uncertainty predictions; assessing the model's ability to return a distribution of uncertainty values is a compelling topic for future research.

where the contribution of each data point is weighted by its β-exponentiated variance estimate. This modified loss simplifies to the standard Gaussian negative log likelihood for $\beta = 0$ and the mean-squared error (MSE) loss for $\beta = 1$. We experiment with several prescriptions for β including constant values $\beta = 0.0, 0.5, 1.0$ and several situations where β changes throughout training, including a linearly decreasing β value, 1 to 0 and two step functions, where β decreases from 1 to 0.5 and 1 to 0.0 at half the total number of epochs. We ultimately select a β value of 0.5, which is recommended by [23].

E Deep Evidential Regression

Similar to MVE, we assume the training data are drawn from a Gaussian likelihood distribution $y_i \sim \mathcal{N}(\mu(x_i), \sigma^2(x_i))$. We also place a Gaussian prior on the mean μ and an Inverse-Gamma prior on the variance σ^2:

$$\begin{aligned} \mu_j &\sim \mathcal{N}(\gamma_j, \sigma_j^2/\nu_j), \\ \sigma_j^2 &\sim \Gamma^{-1}(\alpha_j, \beta_j), \end{aligned} \tag{6}$$

where j is a sample drawn from these hyperprior distributions; $\Gamma(\cdot)$ is the gamma function; and $\mathbf{m} = (\gamma, \nu, \alpha, \beta)$ are hyperparameters of these distributions, where $\gamma \in \mathbb{R}$, $\nu > 0$, $\alpha > 1$, and $\beta > 0$.

One can then formulate the conjugate prior distribution as a Normal-Inverse-Gamma (NIG) distribution; for a full derivation, see [1]. Drawing a sample from the NIG distribution yields a single instance j of the likelihood function: the NIG hyperparameters $\mathbf{m}$ control the location and dispersion (uncertainty) of the likelihood function $\mathcal{N}(\mu_j, \sigma_j^2)$. We will later use the hyperparameters of this higher-order evidential distribution to define the aleatoric uncertainty; these higher-order parameters determine the lower-order likelihood distribution from which observations are drawn.

To fit the model, we define a marginal likelihood $p(y_i|\mathbf{m})$, which is done using Bayesian probability theory in [1]: the conjugate prior defined above is combined with the Gaussian likelihood and integrated over the parameters μ and σ^2. An analytic solution to the marginal likelihood is the Student $t-$distribution:

$$L_{i,\mathrm{NIG}} = \mathrm{St}_{2\alpha}(y_i|\gamma, \frac{\beta(1+\nu)}{\nu\alpha}), \tag{7}$$

where $\mathrm{St}_{2\alpha}$ is a $t-$distribution with 2α degrees of freedom.

The negative log-likelihood loss of this distribution and the addition of an additional term weighted by the width of the $t-$distribution provides the $\mathcal{L}_{\mathrm{NIG}}$ loss that we use for training: $\mathcal{L}_{\mathrm{NIG}} = \frac{1}{N}\sum_{i=1}^{N}\left[-\log L_i^{\mathrm{NIG}} + \lambda|\frac{y_i-\gamma}{w_{\mathrm{St}}}|\Phi\right]$.

F Software package `DeepUQ`

`DeepUQ` is a software package that provides modules, utilities, and scripts for setting the hyperparameters and training both DE and DER models and analyzing the predicted aleatoric uncertainties. It is also designed to be tunable to insert additional UQ methods and/or to create additional noise profiles for uncertainty injection on the 0D or 2D data.

`DeepUQ` provides the following modules and scripts:

- `data.py` - A module for generating data with accompanying controllable injected aleatoric uncertainty on either the input or output variables.
- `model.py` - A module that provides tunable loss functions, network architecture, and hyperparameters for the DE and DER methods.
- `train.py` - A module for the training procedure for both models.
- `DeepEnsemble.py` and `DeepEvidentialRegression.py` - Scripts for generating data, initializing the methods, and training the models.

The DeepUQ-neurIPS-WS-2024 repository provides additional details for how run the DeepUQ scripts to exactly reproduce the results of the paper, including the models, figures, and tables.

G Model Loss

We report the values for the MSE metric and the NIG or $\beta-$NLL loss for the validation set for the low- and high-noise models for the final epoch of training in Tables 1 and 2, respectively.

Metric	0D Data				2D Data			
	Output Injection		Input Injection		Output Injection		Input Injection	
	DE	DER	DE	DER	DE	DER	DE	DER
MSE Metric	0.0001	0.0001	0.0001	0.0001	0.0006	0.0002	0.0002	0.0001
Loss	-0.0502	-3.0890	-0.0416	-2.9338	-0.0426	-2.7691	-0.0993	-3.0639

Table 1: Loss values for the final epoch of the low-noise experiments. We provide the mean-square error (MSE) metric and $\beta-$NLL/NIG loss for DE/DER methods in 0D and 2D for uncertainty injected on the output and input variables.

Metric	0D Data				2D Data			
	Output Injection		Input Injection		Output Injection		Input Injection	
	DE	DER	DE	DER	DE	DER	DE	DER
MSE Metric	0.0098	0.0097	0.0091	0.0092	0.0099	0.0086	0.0047	0.0042
Loss	-0.1724	-0.8728	-0.1678	-0.9018	-0.1767	-0.9202	-0.1330	-1.3358

Table 2: Loss values for the final epoch of the high-noise experiments. We provide the mean-square error (MSE) metric and $\beta-$NLL/NIG loss for DE/DER methods in 0D and 2D for uncertainty injected on the output and input variables.