AI-Solver: Uncertainty in Prediction and Error Estimation for AI in Engineering

Ahmed Al-Jarro
Loic Beheshti
Serban Georgescu
ahmed.al-jarro@uk.fujitsu.com
loic.beheshti@uk.fujitsu.com
serban.georgescu@uk.fujitsu.com
Fujitsu Labs of Europe Ltd.,
Hayes UB4 8FE, London, United Kingdom

Koichi Shirahata
Yasumoto Tomita
Kouta Nakashima
k.shirahata@jp.fujitsu.com
tomita.yasumoto@jp.fujitsu.com
nakashima.kouta@jp.fujitsu.com
Fujitsu Laboratories Ltd.,
Kanagawa 211-8588, Kawasaki, Japan

ABSTRACT

In this work, we build on our recent efforts in developing a deep learning based AI platform that learns from simulation data to extract a general physics-based behavior, code-named the 'AI-Solver'. Importantly, we integrate uncertainty quantification mechanism into the prediction functionality of the AI-Solver and develop a bespoke error estimation methodology that is capable of providing an instant feedback on the confidence in its predictions. The uncertainty quantification in this work is obtained via exploiting the approximation of Bayesian deep learning. Thereafter, our custom developed error estimation method is applied to translate this uncertainty feedback into a reliable and easy to interpret error metric without relying on the availability of ground truth data. This uncertainty quantification to error estimation mechanism that we propose, integrated into the AI-Solver, can impact many design processes where many iterations of simulations are typically required. To our knowledge, the ability to estimate the discrepancy in deep learning predictions without labels is a first in the field of AI for Engineering.

CCS CONCEPTS

• Computer methodology → Artificial intelligence; Deep neural networks • Applied computing → Computer-aided engineering

KEYWORDS

Physics-based simulation, Approximation, Uncertainty, Bayesian Neural Networks, Error estimation

ACM Reference format:

Ahmed Al-Jarro, Loic Beheshti, Serban Georgescu, Yasumoto Tomita and Nakashima Kouta. 2019. AI-Solver: Uncertainty in Prediction and Error Estimation for AI in Engineering. *In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2019, Denver, Colorado, USA, 2 pages.*

https://doi.org/10.1145/XXXXXXXXXX

1 Introduction

On the back of the recent advancement in modern AI, much effort is observed in experimenting with deep neural networks (DNN) to predict the outcome of physical processes [1-3]. DNNs are considered black boxes. Therefore, much recent effort is also reported on mechanisms that can provide explainability, and/or human controllability and interpretability, over the source of prediction from DNNs. Typically, this is achieved at the cost of intrusive interventions into the DNN architectures that could lead to performance degradation, including that in the prediction accuracy [4]. On the other hand, one can invoke a DNN to provide an output on its (un)certainty/trust without degrading the quality in its performance. Depending on the method used, it is also likely to improve the accuracy of prediction [5, 6] at a small or no added computational cost [6]. The field of uncertainty is generally associated with Bayesian deep learning and Bayesian Neural Networks (BNN). We note that uncertainty is not to be confused with a standard classification output, for example, when a model hesitates between classes. Such metric does not consider the limit of knowledge for a given AI structure. Instead, it is necessary to obtain a credible uncertainty metric. In this work, we investigate methods that can provide quantification on the uncertainty in prediction, and integrate them into our AI-Solver [2, 3], to provide error estimation without relying on the availability of ground truth data. The methodology we adopt in this work is derived from pointwise visual uncertainty that is obtained from Bayesian deep learning concepts. We justify the need for a mechanism for producing error estimation in order to avoid misconceptions related to uncertainty in prediction, where it is not a direct representation of the final error. Our model does not only extract an error, but also likely margins of errors. In return, the final error estimation we propose can be easily interpreted and understood by the non-expert end-user.

2 Methodology & Results

Previous versions of the AI-Solver were presented under the name of DeepSim-HiPAC, *Deep Simulation High Performance Approximate Calculations*, in [2, 3]. The AI-Solver can handle a wide variety of classes of problems including those commonly identified in FEA, CFD and CEM, to name a few, with speedups of up to 250000X and extremely low error rate of 2-3% [2, 3]. For example in CFD based problems, we have demonstrated the generality and capabilities of the AI-Solver for airflow [2] and conjugate heat transfer (CHT) [3]. In this work, we report on a new and distinctive feature in the AI-Solver, where we enable it not only able to predict the target solution, but also provide reliable error estimation to its predictions. Therefore, it can assess the credibility of its output, given an arbitrary shaped 2D/3D object, as an input, and without relying on the availability of ground truth data, i.e. labels. To our knowledge, this new and challenging functionality, which is very important to have in practice, is a first in this field of AI for Engineering.

The performance of deep learning algorithms is highly dependent on the training data and its ability to generalize. Therefore, for a 'test case' that is considered relatively far from the training space, or totally unknown, it is typically unlikely that the trained DNN (predictor) will be able to produce accurate enough predictions. As such, we cannot be assured that the predictor is able to cover every possible 'test case'. To alleviate this burden, and therefore increase the reliability of our predictor in the AI-Solver, here, we investigate uncertainty in prediction. A metric from which a level of confidence our predictor has on its prediction can be derived. Uncertainty is a highly active and important field of machine learning. Bayesian learning and BNN are one of the most widely used mechanisms for quantifying uncertainty in deep learning, which relies on attaching probabilistic components to the predictor weights. BNN are able to give a measure of certainty to their output, where it considers the predictor as a probabilistic prior model and the output from it as its posterior. As such, we extract the distribution of prediction for each node of the outmost layer. We therefore do not evaluate an output, y*, to a given input, x*, but a probability distribution associated to that event, i.e. P(y*|x*, X, Y). Here, X and Y represent the inputs and associated outputs of the training dataset. The work presented in [5] contains one early approach where BNN are considered. Nevertheless, it is regarded niche due to its intensive training complexity. Later on, the concept of Bayesian approximation (BA) at inference was introduced in [6], where the global idea is that any network trained with dropout can be approximated as a BNN via a method called Monte Carlo Dropout (MC-Dropout). There exist other approaches demonstrating the effervescence of quantifying uncertainty in prediction. Some use different training mechanisms, while others apply different structural changes to the neural network. The work in [6] keeps the fixed dropout rate used during training active at both the training and inference stages. As a result, a number of predicted solutions are produced, which here we refer to them as the inference samples. From those samples, we extract the mean, μ, and the standard deviation, σ, which are now regarded as the posterior distribution output of the model for that input. The equation that represents the approximated posterior, via Monte Carlo Integration, is:

$$p(y^*|x^*, X, Y) \approx \log\left(\frac{1}{T}\sum_{t=1}^{T} p(y^*|x^*, \omega_t)\right).$$

Here, T defines the number of samples and ω_t represent the model as well as its trained weights for a given sample. We obtain the uncertainty in prediction by a MC-Dropout enabled AI-Solver. The evaluation of BA requires many samples at the inference stage; nevertheless, it is very cheap and produced in real time.

A couple of remarks on the uncertainty obtained from MC-Dropout are in order. We note that this uncertainty is the sum of the aleatoric uncertainty, which is heteroscedactic and is considered irreducible, plus the epistemic uncertainty, which is related to the DNN's quality and/or its lack of knowledge and generalization. This (combined) uncertainty metric is not a direct estimation of the final error, and thus it can be easily misinterpreted. To address this challenge, we first considered the possibility of obtaining a simple mapping between the uncertainty and the final error, given the strong correlation between them. However, due to heteroscedasticity, it becomes clear that a simple and direct mapping between them is not achievable. The heteroscedasticity represents a steady increase with respect to the mean and variance of the error as a function of uncertainty. This observation is consistent with expectations since the more a network is uncertain about its prediction, the higher we expect the average error and potential variance to be. To model a heteroscedactic relation, one generally uses Gaussian Processes (GP). However, our observation on the processed data distribution for our problem is fundamentally incompatible with the mean assumption of GP. Therefore, here we opted for a bespoke and custom developed approach. In this approach, we map a distribution of error to uncertainty, and provide a reliable metric of error as well as margins of confidence for that error. To derive and calibrate our error model, we make use of few data points. Here, we refer the reader to the co-submitted poster with this paper. In the results section of the poster, Figure 3 provides an illustration of the expected error with corresponding error margins as a function of uncertainty. We derive this plot for the uncertainty-error distribution obtained from a trained AI-Solver to predict the problem setup shown in Figure 2. In Figure 3, the red points are the set used for deriving our model and its calibration, and the green points represent unseen before and therefore new records, i.e. test data. Crosses and circles represent actual error and estimated error respectively, with excellent fitting.

REFERENCES

[1] X Guo, W. Li, and F. Iorio. Convolutional neural networks for steady flow approximation. *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 481–490, 2016.

[2] A. Al-Jarro, S Georgescu, Y. Tomita, and K. Nakashima. DeepSim-HiPAC: Deep learning high performance approximate calculation for interactive design and prototyping. *In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2018*, Dallas, TX, USA, 2018.

[3] A. Al-Jarro, L. Beheshti, S Georgescu, Y. Tomita, and K. Nakashima. DeepSim-HiPAC: Deep learning-based platform converts physics-based simulators into real-time ai simulators. *In NVIDIA GPU Technology Conference*, San Jose, California, USA, 2019.

[4] L. H Gilpin, D. Bau, B. Z Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. *In IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.

[5] J. S Denker and Y. Lecun. Transforming neural-net output levels to probability distributions. *In Advances in neural information processing systems*, pages 853-859, 1991.

[6] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *In international conference on machine learning*, pages 1050-1059, 2016.