Uncertainty Quantification and Benchmarking in Deep Learning

Omkar Nitsure
Electrical Engineering Department
Indian Institute of Technology Bombay
Mumbai, India
210070057@iitb.ac.in

***Abstract*—With the advent of Deep Learning, powerful network architectures, clever optimization techniques and large amounts of pre-training data are readily available. It has led to unprecedented accuracies for various tasks in Computer Vision, Natural Language Processing, and Automatic Speech Recognition. These results have led to the motivation to deploy these models to solve real-life problems. The main challenge in their deployment in critical applications is the unreliable nature of their predictions. This unreliability/uncertainty comes from many different sources. The field of uncertainty quantification tries to understand and reduce this unreliability to make model predictions reliable, generalizable and robust to noise. There are a lot of recent advancements in this field to incorporate uncertainty-aware training in state-of-the-art network architectures.**

***Index Terms*—Deep Learning, optimization, Uncertainty Quantification**

I. Introduction

Uncertainty Quantification in the predictions of Deep Learning models is a very active area of research in Machine Learning as it allows for commercial deployment of these models, which will improve the quality of life of humans. These techniques will have applications in safety-critical areas like healthcare, self-driving vehicles, robotics, military and space exploration. By integrating Uncertainty Quantification (UQ) with model training, it is possible to quantify the level of confidence that the model has in its predictions and how much the predictions can be trusted. It also quantifies the amount of noise in the training or test data. This information will guide the improvements in data acquisition tools, further improving the model's accuracy.

Large deep-learning models require huge amounts of pre-training data to learn to solve the desired problem. Most of the time, this data is collected manually and is prone to mistakes during acquisition. Data acquisition is also prone to noise addition as the sensors used to acquire this data are imperfect and add some random noise to every sample. Usually, clean and accurate data is scarce and cannot be used to train large models as they can directly remember that data(overfit) and thus do not generalize well to new data. In mathematical terms, the data comes from an underlying distribution where the individual instances are samples drawn from this distribution. In practical applications, the test data used for evaluating the models might come from a different distribution than the training data. The lack of clean and static data is one of the sources of uncertainty that propagates till the final model predictions and thus needs to be addressed carefully. As an example, a lung MRI scan might be given as input to a brain-tumour classification system. In such cases, it is desirable if the model rejects such test samples and does not provide any classification output. This is an example of Out of Distribution (OOD) detection.

II. Sources of Uncertainty

Various sources contribute to model prediction uncertainty. It motivates researchers to aim to build a system that can flag warnings in such cases and make predictions only for samples where it is confident that the prediction is correct. This approach can allow us to reach 100 % accuracy (which is otherwise theoretically impossible) with human intervention. So, the model makes predictions for only those samples where it is confident and gives uncertain samples to humans for prediction.

Various sources of uncertainty can be classified into the following 2 types -

A. Aleatoric Uncertainty

In Statistical terms, Aleatoric Uncertainty [1] is defined as the uncertainty arising from the inherent randomness of an event (stochastic relationship between input and output variables $\mathrm{Var}(Y\|x)$). Usually, even though the mapping from X to Y is deterministic to some extent, it is theoretically more advantageous to consider it as a stochastic process and admit that Y cannot be explained/predicted from X without error. More data might not be able to reduce this kind of uncertainty while information rich data samples can help in this regard. True data might have some additional features(Z) other than the ones which were acquired(X) and they can help predict the output(Y) with much less uncertainty. Labels can also be noisy (wrong labelling) leading to models with inherent bias.

Model optimization can sometimes increase the robustness of models to errors in data. For large models, parameter optimization using gradient descent ensures that more error-prone features of the training data(X) have much less influence over model prediction than the other cleaner features.

B. Epistemic Uncertainty

Epistemic Uncertainty [1] arises due to a lack of knowledge. Every model architecture can express a set of function space which is searched through during gradient descent to find the parameters which lead to the minimum loss. Even if the data distribution lies in this function space, optimal parameters cannot be learned because of noise. This is called the Epistemic Uncertainty. In most practical cases, the data distribution doesn't even belong to the function space of the architecture making it impossible to learn the distribution and generalize to unseen data. It can be seen as the inability of the model (because of constraints on the number of learnable parameters) to learn to solve the problem. This is the Epistemic Uncertainty.

It can be reduced by adding more training data. Data used for training has many missing features for some data entries. Most of the standard Machine Learning packages discard those samples, which reduces training data size and increases Epistemic Uncertainty. An alternative approach could be to fill missing entries with feature means. This approach maintains the training data size and doesn't increase Epistemic Uncertainty. It is important to note that data extrapolation should be employed only when there are just a few samples with missing entries compared to the overall training data size.

III. Methods

Some standard methods are being used to quantify uncertainty and report highly uncertain input samples for human supervision. Different methods can be classified into the following 2 categories -

A. Bayesian Methods

In standard machine learning, Gradient Descent is used to learn the "right" weights of neural networks and those weights are used as it is for inference. In Bayesian methods, weights are sampled from a predetermined distribution (like Gaussian). The parameters of those distributions are learned during gradient descent. During inference, samples are drawn from the learned weight distributions. Multiple forward passes are done for a test example, each time sampling new weight instances from the distributions. All corresponding model outputs are then computed. If the outputs are similar, then the model is confident in the prediction and if they are different, then the model is uncertain about its prediction.

Some popular methods in this category are the Monte Carlo Dropout [2] and Variational Inference [3] in Graphical Neural Networks.

B. Non-Bayesian Methods

Non-Bayesian methods are also called Frequentist Methods. It involves training completely disjoint models in parallel for the same task and with the same training data. These models have different weight initializations, hyperparameters and sometimes even different model architectures. During inference, the same test input is given to all the models in parallel and the outputs are compared. If the outputs of these models (which generally have similar training loss) are similar then the prediction is confident else the model is uncertain. If the model is confident, then the output can be the mean of the outputs of the ensemble.

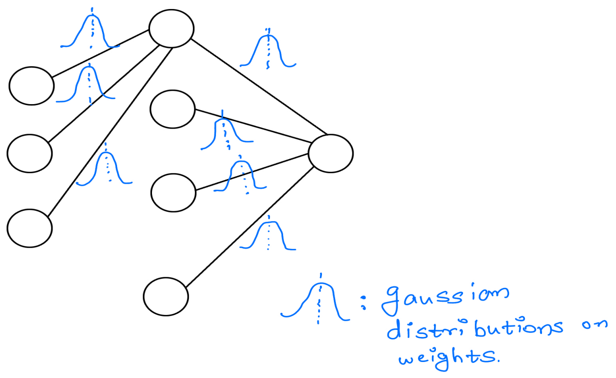

Fig. 1. Distributions of weights in Bayesian Methods

One of the drawbacks of this method is the large amount of compute required both for training and inference. It also requires a large amount of memory to load all models in parallel. Some popular methods in this category are Ensemble Learning [4] and Bootstrap [5].

IV. Problems with SoftMax and Cross Entropy

In most classification tasks, the final layer gives the probability of an example belonging to the different possible classes. SoftMax activation is used for this purpose. The sample is assigned that label which has the largest probability value. During training, Cross Entropy loss penalizes high probability values assigned to the wrong classes. This training approach can potentially lead to very confident but wrong predictions. It happens because minimizing cross-entropy loss encourages the probability values to be very high for the correct class while being very low for the other classes. Thus, even if the model is unsure about its prediction, it tends to output a very high probability value for one class. This is why it sometimes makes the wrong prediction with confidence. Thus, researchers are trying to discover novel loss functions and activations which can compete with SoftMax and cross-entropy pair in terms of accuracy and still give good uncertainty performance.

V. Current Research ideas

A. Evidential Deep Learning

Evidential Deep Learning [6] tries to overcome the disadvantages of SoftMax activation for the final output layer of the neural network. Evidential deep learning defines a different output layer for the neural networks. It defines an evidential prior based on the amount of evidence available from a particular input for each of the classes. It trains the model to "collect" more evidence for the correct class than other classes.

If the focus is only on the classification tasks, then by employing Evidential deep learning, we train the model to output parameters of a Dirichlet distribution. So, given an input sample, the neural network outputs values of parameters of a Dirichlet distribution. The model prediction and the variance(uncertainty) are computed using standard formulae involving the parameter values. In this way, there is no need for the cross-entropy loss and SoftMax activation, which ensures reliable and robust predictions. This approach might sometimes reduce the model prediction accuracy, but that is a trade-off for much better uncertainty quantification.

VI. Setup, Model and Datasets

The setup I used for doing the experiments is as follows -

- Training data - 10000 randomly sampled images from train split of MNIST dataset
- Base classification model - LeNet (No of trainable parameters - 60000)
- In distribution (ID) test examples - 1000 randomly sampled images from test split of MNIST
- Out-of-distribution (OOD) examples - 500 images of digits other than MNIST + 1000 images of Imagenet

VII. Distance-aware Neural Networks

Modern Neural Networks extract rich information from an input sample in terms of a vectorized representation, but these representations are usually not distance-aware [7]. In these techniques, the model is trained to output distance-aware representations. The training data distribution is usually just a subset of the overall data-generating distribution, but we want the model to generalize well to the unseen OOD data as well. Thus, the model should be able to recognize whether the test point belongs to the training data manifold or not. They impose a L2 Lipschitz constraint while training, which ensures that the distance between 2 points in the representation space is proportional to the distance between them in the input space (inputs belonging to different classes should have distant representations). This allows us to restrict decision-making to only the In distribution (ID) points.

VIII. Deterministic Uncertainty Quantification

In classification tasks, the neural networks usually consist of 2 parts - 1) Feature extractor 2) Output probability distribution projector. It means that the neural network creates a representation for every input and then maps this representation to one of the class labels. Thus, these representation clusters must be distant for different classes. Deterministic Uncertainty Quantification [8] (DUQ) exploits this idea to estimate the uncertainty of a particular example.

It defines a kernel value per class, which is defined as -

$$K_c(f_\theta(x), e_c) = \exp[-\frac{\frac{1}{n}||W_c f_\theta(x) - e_c||_2^2}{2\sigma^2}]$$

The training process tries to maximize the kernel value for the correct class and minimize it for all the wrong classes. The training objective and loss-function are as follows -

$$\arg\max_c K_c(f_\theta(x), e_c)$$

$$L(x, y) = -\sum_c y_c \log(K_c) + (1 - y_c)\log(1 - K_c)$$

It ensures that the cluster centers are separated in the higher dimensional space. Cluster centers are computed again after each iteration. Uncertainty score is then defined as the minimum of all kernel values. If this value is small, then it means that the input example belongs to that class, but if this value is large, then it means that the input doesn't belong to any of the classes. It can be either an OOD example or the model is uncertain about its prediction.

A. *Results of Experiments on DUQ*

I trained a LeNet model on the MNIST dataset for 100 epochs. Following is the T-SNE visualization of the features extracted by the feature extractor before the final classification layer -

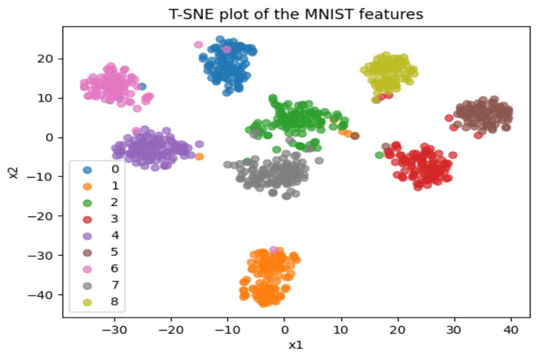

Fig. 2. Feature Clusters for the various classes

Training was robust and loss reduced uniformly but the model accuracy is a bit lower than the standard SoftMax-based models. A point to note is that the uncertainty quantification scores are very good. The average uncertainty score for ID examples is 14.27 while for OOD examples is 24313.89. The following plot shows the accuracy of the model after rejecting certain percentage of highly uncertain points (tested on the MNIST dataset) during inference -

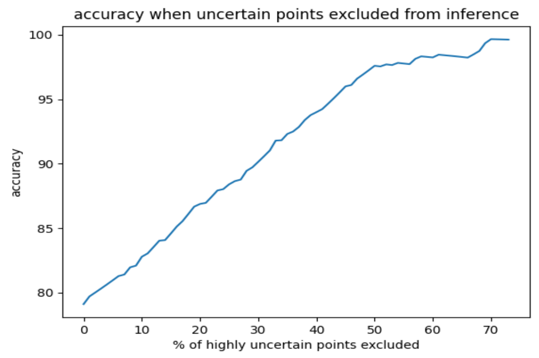

Fig. 3. Accuracy Vs % of highly uncertain points rejected for DUQ

B. *Effects of Noise Addition*

I added zero-mean white gaussian noise with increasing variance to the input at inference for some randomly selected images and then reported the average value of uncertainty. The average value of uncertainty increased as can be seen in the following figure. This analysis can be used in another context. We can use this technique of progressively adding more and more noise to the data and reporting the Uncertainty value given by a technique for benchmarking it against all other SOTA methods.

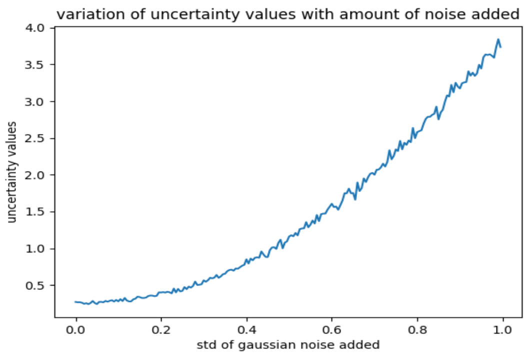

Fig. 4. Average uncertainty value with increasing noise variance

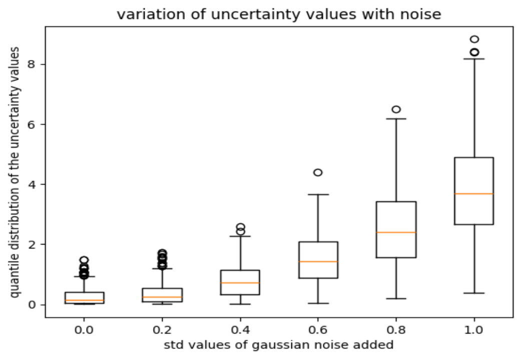

Fig. 5. Distribution of Uncertainty values for some noise values

IX. Quantification of Uncertainty with Adversarial Models

This approach is related to the idea of training different models in parallel and defining uncertainty as the variance of their prediction. Uncertainty Quantification with Adversarial Models [9] (QUAM) has only an additional criterion that the models must fulfil. This method starts with a base model trained on the task at hand where we can make use of large amounts of pre-training data using Self Supervised Learning (SSL). K different models are trained starting from the same base model where each model should have output probability distribution centred around some randomly picked class while maintaining similar loss on the training data. The optimization objective is -

$$\max_{\delta \in \Delta} D(p(y|x,w), p(y|x,w+\delta)) + c(\log p(w+\delta|D) - \log p(w|D) + \gamma)$$

Here, **D** is usually the **Kullback-Leibler** (KL) divergence.

Finally, the outputs of K finetuned models are computed. Uncertainty is the variance of the K predictions. It is computationally expensive as it finetunes K different models for each example, but the uncertainty value reported by this method is very accurate (better than all other methods). It justifies the time, space and computing required if it is deployed for a safety-critical application.

A. *Results of Experiments on QUAM*

This method makes use of the power of pre-training. Thus, it gives very good classification accuracy while doing excellent uncertainty quantification. Uncertainty values are computed as the entropy of the output distribution. The average uncertainty values for different image types are -

- ID images(MNIST) - 0.232
- OOD but the same classes(Digits) - 0.456
- OOD and different classes - 2.251

The following plot shows how inference accuracy varies with certain percentage of highly uncertain points removed -

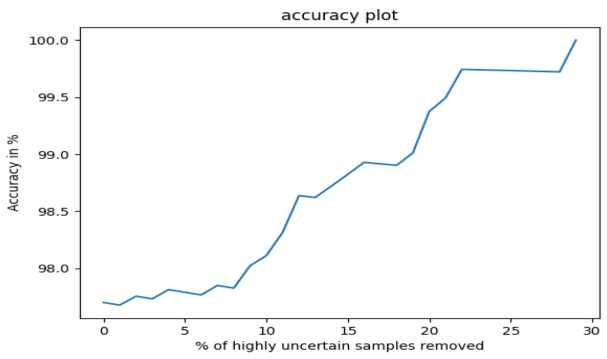

Fig. 6. accuracy Vs % of highly uncertaint points rejected for QUAM

I then plotted the accuracy curves for both DUQ and QUAM on the same plot and computed the area under the curve (AUC). The AUC values for DUQ and QUAM are 6773.83 and 7361.92, respectively.

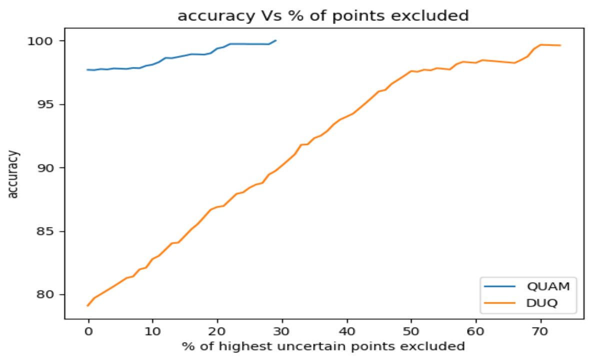

Fig. 7. AUC curve for both DUQ and QUAM

X. Conclusions

Uncertainty Quantification is a new and heavily researched topic in modern Machine Learning because the motivation of most of the research is to improve the model's accuracy and reliability so it can be deployed in real life. The model's accuracy falls after deployment as the input it receives is generally different from the training data. It causes the model to be uncertain about its prediction. The sources of this uncertainty are noise in input data, distribution shift of the test data from the training data, adversarial attacks [10] of different kinds, smaller model size and thus inability to generalize to a broader range of input types, inability of the model to detect a test example as OOD (coming from a distribution that the model has not seen during training).

Modern Deep Learning models are trained with at least some technique to mitigate the uncertainty in prediction or quantify the amount of uncertainty. Some traditional methods for uncertainty quantification are Ensemble learning [4], Bootstrap [5], Monte Carlo dropout [2]. Adversarial attacks are aimed to exploit the vulnerabilities in the model prediction. But this technique is being heavily used in modern machine learning research on techniques for uncertainty quantification. Some state-of-the-art techniques (SOTA) for uncertainty quantification are adversarial training [10], exploiting relative distance in representation space [8], adversarial model fine-tuning [9] for almost exact detection of out-of-distribution test samples, etc.

References

[1] C. Gruber, P. O. Schenk, M. Schierholz, F. Kreuter, and G. Kauermann, "Sources of uncertainty in machine learning–a statisticians' view," *arXiv preprint arXiv:2305.16703*, 2023.

[2] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, pp. 1050–1059, PMLR, 2016.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[4] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[5] B. Efron, *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

[6] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," *Advances in neural information processing systems*, vol. 31, 2018.

[7] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan, "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness," *Advances in neural information processing systems*, vol. 33, pp. 7498–7512, 2020.

[8] J. van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty estimation using a single deep deterministic neural network," 2020.

[9] K. Schweighofer, L. Aichberger, M. Ielanskyi, G. Klambauer, and S. Hochreiter, "Quantification of uncertainty with adversarial models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[10] I. Alarab and S. Prakoonwit, "Adversarial attack for uncertainty estimation: identifying critical regions in neural networks," *Neural Processing Letters*, vol. 54, no. 3, pp. 1805–1821, 2022.