Project Final Report

Yuchen Zhuang[1,2], Wenqi Shi[1,2], Tianyu Zhan[1,2], Jincheng Zhu[1,2], Yuechen Wu[1,2]
[1]Team TBD,
[2]Georgia Institute of Technology

1 Project summary

With the rapid development of machine learning, highly-advanced algorithms have gradually moved towards black boxes, which leads to a new problem, explain-ability. Although the highly-advanced models and algorithms are capable of learning powerful representations of mappings between high dimensional given input and expected output, at occasions even their designers cannot explain why they arrived at some mappings. In addition, current measurements could hardly explain the results from the perspective of confidence. A well-known example is the accident caused by the auto-pilot system of Tesla, where it mistook the white side of a trailer as bright sky after a difficult decision. Considering the fact that machine learning models can make mistakes [1], we still need to know the uncertainty of mappings to support our decision making. In other words, rather than training a dog-cat classifier with 100% validation accuracy, it is more important to train it to say "I don't know" when given a picture of fish.

We will address the problem of uncertainty in two aspects - model uncertainty and data uncertainty. Bayesian neural networks (BNN) is one of the approaches to quantify uncertainty associated with models. Data uncertainty inherits the noise captured by observation and measurement uncertainty. With sufficient analysis of uncertainties, decision making problems mentioned above will be better handled. In the project, we have also explored the quantification of both uncertainties in several scenarios, including simple data regression, natural language processing (NLP) and computer vision (CV).

2 Detailed project description

2.1 Problem description

Nowadays, machine learning has earned its popularity in multiple areas with its magical performance. However, before applying machine learning techniques to circumstances where reliability is the priority, we still need to quantify their uncertainty to reduce the risk. Thanks to recent progress in Bayesian deep learning, such quantification has been made possible, which significantly contributes to improving the decision making process. For our project, we will look into the mathematical deductions of quantifying uncertainty in Bayesian models, and conduct several experiments in different application scenarios.

For this problem, we mainly focus on the dropout variation Inference methods in Bayesian Neural Network. We study the mathematical deduction and knowledge base of the dropout variation inference methods. And separately do the different experiments on different Corresponding applications and finally composed all the experiments and works together. Despite the mathematical deduction part was accomplished by all of the members. More in details, Yuchen Zhuang is responsible for the NLP application; Jincheng Zhu and Yuechen Wu are responsible for the toy dataset regression application; Wenqi Shi and Tianyu Zhan are responsible for the Computer Vision Task. Finally, Yuchen Zhuang is responsible for the final polish and summarization of the report.

2.2 Main thought

Many factors in machine learning can affect the uncertainties, when we apply the different models. In this paper we separate the uncertainties into 2 categories, epistemic uncertainty and aleatoric uncertainty [2], and they can also be named as model uncertainty and data uncertainty.

First of all, we start with the law of total variance. Given a input variable x and its corresponding output variable y, the variance in y can be decomposed as:

$$\mathrm{Var}(y) = \mathrm{Var}(\mathbb{E}[y|x]) + \mathbb{E}[\mathrm{Var}(y|x)] \tag{1}$$

We mathematically define model uncertainty and data uncertainty as:

$$U_m(y|x) = \text{Var}(\mathbb{E}[y|x]) \quad U_d(y|x) = \mathbb{E}[\text{Var}(y|x)]. \tag{2}$$

where U_m and U_d are model and data uncertainties respectively. We can see that both uncertainties partially explain the variance in the observation. In particular, model uncertainty explains the part related to the mapping process $\mathbb{E}[y|x]$ and data uncertainty describes the variance inherent to the conditional distribution $\text{Var}(y|x)$. By quantifying both uncertainties, we essentially are trying to explain different parts of the observation noise in y. We may explain and deduce the equations in details in the following sections:

2.2.1 Model Uncertainty (Epistemic Uncertainty)

With the aid of the Bayesian neural network we may find the posterior distribution of $\mathbf{W}$ given the dataset $D = \{(x_i, y_i)\}_{i=1}^N$. We can also specify the data generating process in the regression case as:

$$y|\mathbf{w} \sim \mathcal{N}(f^W(x), \sigma^2). \tag{3}$$

With the posterior distribution $p(\mathbf{W}|D)$, given a new input vector x^*, the prediction is obtained by marginalizing over the posterior:

$$p(y^*|\mathbf{x}^*, D) = \int_{\mathbf{W}} p(y^*|f^{\mathbf{W}}(x))p(\mathbf{W}|D)d\mathbf{W} \tag{4}$$

From the experience and the basis of Bayesian neural network, we may have gotten to know that the posterior distribution is very difficult to compute, because it is very difficult to get the true data distribution, unless all the weights $\mathbf{W}$ are traversed and this is obviously impossible. Thus, the exact inference is intractable in this kind of model. We use variational inference approach to find an approximation $q_\theta(\mathbf{W})$ to the true posterior $p(\mathbf{W}|D)$ parameterized by a different sets of θ, where the Kullback-Leibler (KL) divergence of the two distributions is minimized. Existing literatures have introduced several variational inference methods proposed for Bayesian neural networks [3–7]. Among them, the dropout variational inference method, when applied to models with dropout layers, requires no retraining and can be applied with minimum changes. The only requirement is that dropouts have to be added between nonlinear layers. The Monte Carlo method is that we utilize limited times of samples to generate a estimation towards our posterior distribution and we can use the approximate distribution as $P(\mathbf{W}|D)$. With this, we can know whether the dataset D is in the distributions we have learnt or not and get the epistemic uncertainty. To avoid that the whole system generate the same results everytime we send the data into it, we will activate the dropout layer in the testing stage. In general, we set dropout layer as a natural random generator. Thus, we do not need to make any changes to out network to obtain the $P(\mathbf{W}|D)$.

At test time, we have optimized approximated posterior $q(\mathbf{W})$. Prediction distribution can be approximated by switching $p(\mathbf{W}|D)$ to $q(\mathbf{W})$ and perform Monte Carlo integration as follows:

$$\mathbb{E}(y^*|\mathbf{x}^*) \approx \frac{1}{M}\sum_{j=1}^{M} f^{\mathbf{W}_j}(\mathbf{x}^*), \ \text{Var}(y^*) \approx \sigma^2 + \frac{1}{M}\sum_{j=1}^{M} f^{\hat{\mathbf{W}}_j}(x^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2. \tag{5}$$

where $\hat{\mathbf{W}}_j$ is sampled from $q(\mathbf{W})$.

Note here σ^2 is the inherent noise associated with the inputs which is homogeneous across the input space. This is often considered by adding a weight decay term in the loss function. We will discuss the modeling of input-dependent data uncertainty in the next section. The rest part of the variance arises because of the uncertainty about the model parameters $\mathbf{W}$. We use this to quantify model uncertainty in the study:

$$U_m(y^*|\mathbf{x}^*) = \frac{1}{M}\sum_{j=1}^{M} f^{\hat{\mathbf{W}}_j}(x^*)^2 - \mathbb{E}(y^*|\mathbf{x}^*)^2. \tag{6}$$

2.2.2 Data Uncertainty (Aleatoric Uncertainty)

Data uncertainty can be either modeled homogeneous across input space or input-dependent. We choose the second one and make the assumption that data uncertainty is dependent on the input. To achieve this, we

need to have a model that not only predicts the output values but also estimates the output variances given some input. In other words, the model needs to give an estimation of $\text{Var}(y|x)$.

Denote $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ as functions parameterized by $\mathbf{W}$ that calculate output mean and standard deviation for input $\mathbf{x}$. Commonly, the logarithm operation is applied for stability of the loss function. We make the following assumption on the data generating process: $y \sim \mathcal{N}(\mu(x), \sigma(x)^2)$.

Given the setting and the assumption, the negative data log-likelihood can be written as follows:

$$L(\mathbf{W}) = -\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|\mu(\mathbf{x}_i), \sigma(x_i)) = \frac{1}{N}\sum_{i=1}^{N} \left(\frac{\|y_i - f(x_i)\|^2}{2\sigma(x_i)^2} + \frac{1}{2}\log(\sigma(x_i)^2) + \frac{1}{2}\log 2\pi\right). \tag{7}$$

Among the function, our model is represented as $\mathbf{f}$, the corresponding output is $\{f(x_i), \sigma^2\}$. In the loss function, the $\sigma(x_i)^2$ is to describe the aleatoric uncertainty of the model on data x_i, which is exactly the variance of data.

While the equation above works desirably for regression, it is based on the previous assumption. This assumption clearly does not hold in the classification context. We can however adapt the same formulation in the logit space. In detail, define $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ as functions that maps input $\mathbf{x}$ to the logit space. Logit vector is sampled and thereafter transformed into probabilities using softmax operation. This process can be described as:

$$\begin{aligned} \mathbf{u} &\sim \mathcal{N}(\mu(\mathbf{x}), \text{diag}(\sigma(\mathbf{x})^{\mathbf{2}})), \\ \mathbf{p} &= \text{softmax}(\mathbf{u}), \\ y &\sim \text{Categorical}(\mathbf{p}). \end{aligned} \tag{8}$$

where diag() function takes a vector and output a diagonal matrix by putting the elements on the main diagonal. Note here that y is a single label.

During training, we seek to maximize the expected data likelihood. Here we approximate the expected data likelihood. Here, we approximate the expected distribution for $\mathbf{p}$ using Monte Carlo approximation:

$$\mathbf{u}^{(k)} \approx \mathcal{N}(\mu(\mathbf{x}), \text{diag}(\sigma(\mathbf{x}))),\ E[\mathbf{p}] \approx \frac{1}{K}\sum_{k=1}^{K} \text{softmax}(\mathbf{u}^{(k)}). \tag{9}$$

The negative log-likelihood for the dataset can be written as:

$$\mathbf{L}_{clf}(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp(u_{i,y_i}^{(k)} - \log \sum_{c} \exp u_{i,c}^{(k)}) - \log K. \tag{10}$$

where $u_{i,c}$ is the c-th element in $\mathbf{u}_i$.

After the model is optimized, we use $\sigma(\mathbf{x}^*)$ to estimate the data uncertainty. Thiscan reflect to a certain extent the variance caused by the input.

3 Experiments and Result

3.1 Toy Model

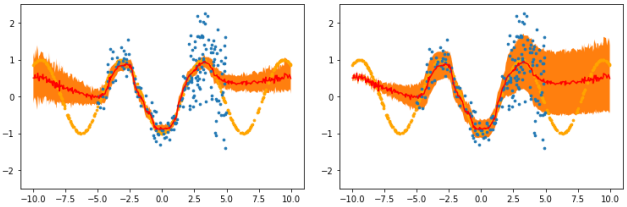
3.1.1 Sinusoid Function Regression

We first experimented on a simple dataset generated by ourselves - discrete sinusoid data points (shown in Figure 1a and 1b). Blue points represent the training data, which is based on discrete sinusoid function defined over $[-5, 5]$. We added Gaussian Noise $\mathcal{N}(\mu = 0, \sigma^2)$ over $[-5, -2.5]$ and $\mathcal{N}(\mu = 0, 4\sigma^2)$ over $[2.5, 5]$. Orange points represent the testing data, which is generated using the same sinusoid function but defined over $[-10, 10]$. Obviously, testing data was defined over a broader range than training data.

We trained a 7-layer neural network for regression. Each layer consists of a linear layer with 64 nodes, a dropout layer and a ReLU layer. For training, we used Adam optimizer and set parameters as follow. $\text{lr} = 0.001$, $\text{weight\_decay} = 1e-4$, $\text{drop\_out} = 0.2$. We tried 3 different Gaussian Noise variance 0.01, 0.05 and 0.1, and here we only place the $\sigma = 0.1$ case here as an example.

The red lines in Figure 1a and 1b show the regression prediction of function value. The orange area in Figure 1a represents model uncertainty (epistemic uncertainty), while in Figure 1b represents data uncer-

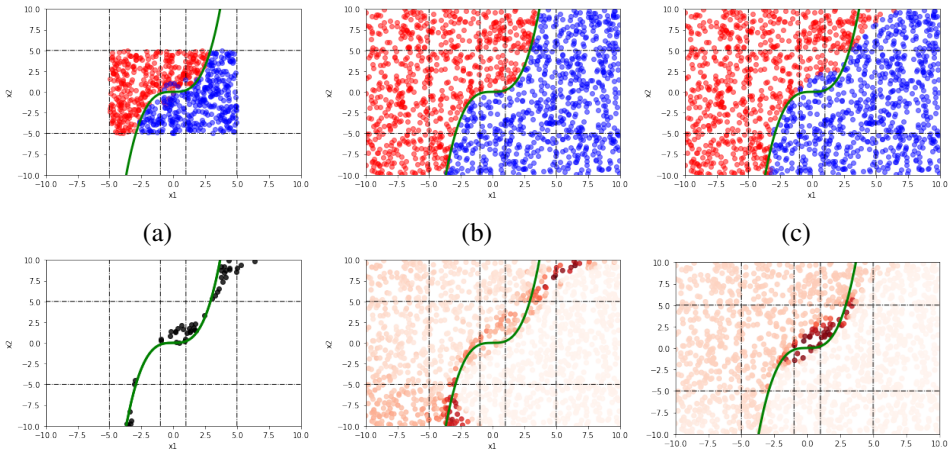
tainty (aleatoric uncertainty). As can be inferred from the plots, model uncertainty increases when testing data falls out of the range where training data is defined. As the variance of noise gets larger, data uncertainty also increases.



(a) Model Uncertainty (b) Data Uncertainty

Figure 1: When the Gaussian Noise is $\mu = 0, \sigma = 0.1$.

3.1.2 Binary Classification



(a) (b) (c)

(d) (e) (f)

Figure 2: a) Training Data-set. b) Testing Data-set. c) Classified Results. d) Misclassified points. e) Epistemic Uncertainty. The darker the color, the higher the uncertainty. f) Aleatoric Uncertainty. The darker the color, the higher the uncertainty.

To further illustrate the significance of uncertainty in practical problems, we trained a similar 7-layer neural network for a binary classification task. We generate the 700 points training point uniformly in range $[-5,5] \times [-5,5]$ with relatively high noise in range $x_1 \in [-1,1]$ and 2100 test points without noise uniformly in range $[-10,10] \times [-10,10]$. The green lines in the figures mark the real decision boundary.

From the Figure 2, we can see the epistemic uncertainty is higher near the dividing line outside the training data, while the aleatoric uncertainty is higher near the dividing line where the noise of training data is greater. The results have a good correspondence with our intuitive description of the two uncertainties.

In general, we can find that the misclassified points usually have at least one type of uncertainty which is very high. Therefore, when faced with the classification problem, the error points can be well located by observing the uncertainty of the network output. At this time, the introduction of human intervention or error correction mechanism can greatly avoid misclassification.

3.2 Computer Vision Application

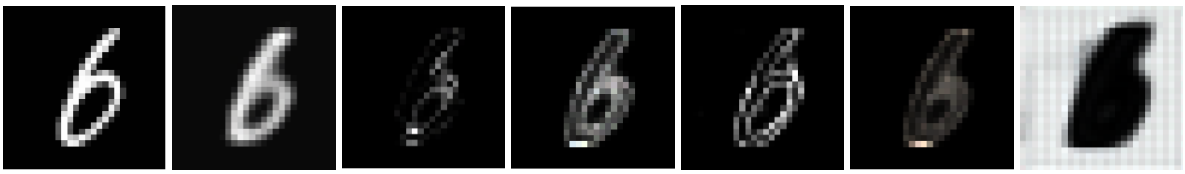
In this section, we develop the model which can study the effects of modeling either aleatoric uncertainty, epistemic uncertainty, or modeling both uncertainties together in the regression task [8]. For aleatoric un-

certainty, we observed that it can be interpreted as learned loss attenuation in regression tasks. For epistemic uncertainty, we model it using Monte Carlo dropout which is an approach repeated random sampling to obtain posterior distribution $P(W|D)$. That is implemented with dropout at test time(We sampled 20 nets here). The predictive uncertainty for pixel y in the combined model can be approximated using the formula as below:

$$\text{Var}(\mathbf{y}) \approx \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t^2 - (\frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t)^2 + \frac{1}{T}\sum_{t=1}^{T}\hat{\sigma}_t^2) \tag{11}$$

$\{\hat{\mathbf{y}}_t, \hat{\sigma}_t{}^2\}_{t=1}^{T}$ is a set of T sampled outputs: $\hat{\mathbf{y}}_t, \hat{\sigma}_t{}^2 = f^{\widehat{\mathbf{W}}_t}(x)$ for randomly masked weights $\widehat{\mathbf{W}}_t \sim q(\mathbf{W})$.

To simplify the problem, we train a simple auto-encoder (regression) to reconstruct MNIST digits and analyze the performance with single uncertainty and combination uncertainty model as well. Figure 3a-3c shows the original and the reconstructed picture of the network. To study the effects of the uncertainty, we design three experiments. Firstly, we only model the aleatoric uncertainty. Secondly, we only model the epistemic uncertainty and finally we model both uncertainties together. Figure 3d-3g show the results.



(a) (b) (c) (d) (e) (f) (g)

Figure 3: a) Original Image. b) Reconstructed Image. c) L2 error. d) Only aleatoric. e) Only epistemic. f) Combined aleatoric. g) Combined epistemic.

From Figure 3d-3g, we can observe that epistecmic and aleatoric seem to capture the same phenomenon when modeled separately. Besides, we can know that they capture the uncertainty which is related to contours reconstruction. However, modeling the two uncertainties together generates different results than modeling separately. They capture complementary information when modeled together. In this case, the epistemic uncertainty is maximal on the background. Perhaps the uncertainty about the digits contours is already captured by the aleatoric uncertainty. Another phenomenon is that when we adjust the size of the training set, we find that epistemic uncertainty decreases as the training dataset gets larger while the aleatoric uncertainty remains relatively constant. So we can get the idea that epistemic uncertainty can be explained away with enough data while aleatoric uncertainty cannot.

3.3 Natural Language Processing Classification

Unlike the Toy dataset and the Computer Vision examples, the NLP examples are not that easy to establish. Because the sentences or words in the NLP tasks are high-dimensional data and are very difficult to visualize the uncertainty like the previous examples. We cannot explicitly see how the data points (sentences) on a high-dimensional space distribute. Thus, we need to reveal some new metrics and visualization methods to assess the influence of the uncertainty. Besides, we also need to design new settings for the NLP experiment.

For the setting, we get to know that estimating the classification model's uncertainty can be accomplished via two methods: **a)** Utilizing regression MSE metric in NLP classification tasks to see the error between golden score and the generated score, where if the uncertainty are detected and reduced, the performance will be improved to some extent; **b)** we can also utilize the model calibration to estimate the uncertainty of the model, where if the model is more calibrated, the model will have relatively less uncertainty in its decisions. For the metrics, accuracy is a standard metric to measure classification performances, while another metric is to see the model's calibration, which is the Expected Calibration Error (ECE) (The lower

the ECE score is, the more calibrated the model is [9]):

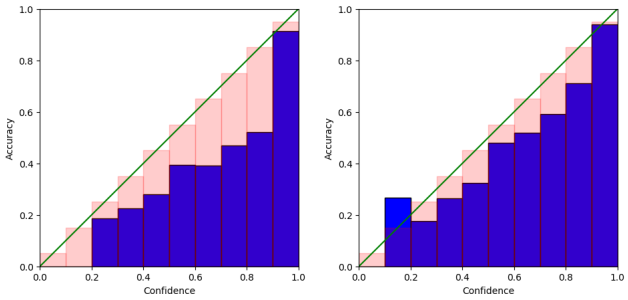
$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|, \ \text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\hat{y}_i = y_i), \ \text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i. \quad (12)$$

Based on the proposed setting **a)**, we test the regression MSE of the model TextCNN on the dataset IMDB movie review. The experiment platform is the Nvidia 2080. From the table, most of the performance gain is from quantifying model uncertainty. Modeling input-dependent uncertainty alone marginally hurts prediction performances. The performances for classification increase marginally with added uncertainty measures. We conjecture that this might be due to the limited output space in the classification setting.

Table 1: Experiments on TextCNN and IMDB dataset.

Model	**TextCNN**	**TextCNN+MU**	**TextCNN+DU**	**TextCNN+CU**
MSE	3.71	3.23	3.80	3.14

For the setting **b)**, we make experiments on the newly hottest NLP model BERT on 20news dataset, with 10 of the classes as the in-domain data and the left 10 as the out-domain data. We test the performance of the combined uncertainty method on the model on the calibration level. The accuracy of the BERT-baseline model is 85.7% and its ECE score is 9.404, while the accuracy of the BERT-combined uncertainty model is 85.8% and its ECE score is 5.830, which means that the combined uncertainty model truly improve the accuracy and the calibration to some extent via detecting and reducing the uncertainties in the decisions.



(a) (b)

Figure 4: Calibration figure of the a) Bert-base model and the b) Bert-combined model.

4 Conclusion

In the project, with the aid of the Bayesian Neural Network knowledge, we have managed to present mathematical deduction of the aleatoric uncertainty, epistemic uncertainty and the combined uncertainty in the deep learning algorithms via Monte Carlo Dropout inference method. Moreover, we also design different settings of experiments in different applications of Simple toy regression dataset, Computer Vision and Natural Language Processing. Besides, the valid and explanable results and analysis are offered as well.

References

[1] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[2] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?" *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[3] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, 2011, pp. 2348–2356.

[4] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International Conference on Machine Learning*, 2015, pp. 1861–1869.

[5] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

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

[7] L. Zhu and N. Laptev, "Deep and confident prediction for time series at uber," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 103–110.

[8] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" 2017.

[9] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1321–1330.