KN-RUE: Key Nodes based Resampling Uncertainty Estimation

1st Xiang Li
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
xianglinwpu@mail.nwpu.edu.cn

2nd Wen Jiang*
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
jiangwen@nwpu.edu.cn

3rd Xinyang Deng
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
xinyang.deng@nwpu.edu.cn

4th Jie Geng
School of Electronics and Information
Northwestern Polytechnical University
Xi'an, China
gengjie@nwpu.edu.cn

***Abstract*—With the continuous development and advancement of neural networks, in the application of neural networks, users not only require neural networks to be able to complete a given task but also want to know when they can trust the network's prediction results and when they need to be cautious about the prediction results. In response to the need for uncertainty estimation of neural networks, many researchers have invested in the study of uncertainty estimation. Existing uncertainty evaluation methods are difficult to apply to deep neural networks with large parameter scales, complex internal structures, and mappings between inputs and outputs that are hard to express. This paper proposes a key nodes based resampling uncertainty estimation method ((KN-RUE), which achieves uncertainty estimation of prediction results for arbitrarily given large-scale neural networks. In this method, the first step involves analyzing the differences in feature space between adversarial and clean samples, identifying the main nodes affected by adversarial samples, and determining the critical nodes within the network. Next, by resampling the parameters of key nodes, the model is extended while ensuring model performance as much as possible, thus completing the measurement of uncertainty in prediction results. Through experiments, the effectiveness of the extended model and the superiority of uncertainty estimation performance in KN-RUE have been verified.**

***Index Terms*—deep learning, resampling uncertainty estimation, network node analysis**

I. Introduction

In the past decade, with the booming development of deep learning, neural networks have become absolutely dominant in computer vision tasks by virtue of their superior performance. It has enabled neural networks to be used in more and more intelligent applications, such as autonomous driving [1], object detection [2], medical image processing [3], and so on. Although neural networks have excellent performance in various fields, they also have defects that can not be ignored, 1. Neural networks often exhibit excessive confidence in their predictions and struggle to provide a quantified assessment of the uncertainty associated with those predictions. 2. When the neural network receives data out of the distribution of the training set, it is unable to make a judgment, and it will still give the results in the category of the training set [4]. The weaknesses of neural networks make them unable to be trusted by the user. Therefore, in scenarios with high-security requirements, uncertainty quantification for the predictions of neural networks is necessary because it can help end users determine when they can trust the model's predictions and when they need to be extra cautious in making decisions based on those predictions.

Many researchers have been working on understanding and quantifying uncertainty in neural network prediction and have proposed various methods to measure and quantify uncertainty in neural networks. Among them, GPR (Gaussian Process Regression) can be regarded as generalized Bayesian inference, extending from inference about a finite set of random variables to inference about a function [5], [6] shows Gaussian processes and prior with a single infinitely wide hidden layer and network parameters, which allows to perform Bayesian inference on neural networks in its exact form using simple matrix operations. As deep learning has grown in popularity in recent years, GPR has made significant extensions to standard DNNs [7] and deep convolutional neural networks [8], [9]. Secondly, uncertainty modeling in Bayesian neural networks combines the scalability, expressiveness, and predictive performance of neural networks with Bayesian learning, rather than learning via the maximum likelihood principle. The Bayesian approach considers the network parameter θ as a random variable, and the goal is to find the entire distribution of reasonable θ values that might generate the observed data D [10]. Finally, neural network integration involves independently training multiple models and aggregating predictions from these separate models. When constructing neural network model integration, a high degree of diversity among the individual models needs

The work was supported by National Natural Science Foundation of China (Program No. 62173272).
*Corresponding author

to be maintained [11].

Although these methods have made effective progress in neural network uncertainty quantitative, the shortcomings of each method are also very obvious, firstly, GPR usually does not scale well to large training datasets (large N) as it has a training complexity of $\mathcal{O}(N^3)$ [12]. Secondly, the uncertainty quantitative of Bayesian networks has requirements on the structure of the neural network and the training process, and cannot be directly supposed to a given arbitrary neural network. Finally, neural network integration likewise requires multiple models in the training process, cannot be used for a given neural network, and greatly increases the training cost.

To achieve uncertainty estimation for any given large-scale neural network, in this paper, we propose a key node resampling uncertainty estimation (KN-RUE) method. In this method, the uncertainty assessment of the prediction results for a neural network is divided into two parts, the first one is the network node contribution measure method based on the adversarial attack. This part analyses the difference between the activation values of the adversarial samples generated by the adversarial attack algorithm and the clean samples in the network to determine the main attack nodes of the adversarial attack algorithm on the neural network, so as to determine the key nodes in it. Then there is an uncertainty estimation based on key node resampling, which achieves the expansion of the neural network by resampling the parameters of key nodes, to achieve the generation of multiple prediction results for a single sample, and to estimate the uncertainty measure of the prediction results. The main contributions of this paper are shown in the following three aspects:

- We propose a network node contribution metric method based on adversarial attacks. By analyzing the difference between the activation values of adversarial samples and clean samples inside the neural network, we search for the reasons that lead to the recognition error of the adversarial samples, and achieve the network node contribution metric and key node extraction.
- We propose an uncertainty estimation method based on the resampling of key nodes, which simulates the Bootstrap training process by resampling the key nodes to achieve model expansion. While greatly reducing the computational effort, the performance of the model is preserved as much as possible.
- We conduct experiments on mainstream datasets and networks to demonstrate the applicability and effectiveness of the proposed method.

II. Methodology

A. Overview

The traditional resampling uncertainty measure makes it difficult to measure uncertainty for deep neural network prediction results with a huge number of parameters. In this paper, we propose a critical node resampling uncertainty metric technique based on adversarial attack network path analysis. As shown in Fig. 1, in this method we propose the key node analysis algorithm of deep neural network based on anti-attack to complete the extraction of key nodes of deep neural network and combine the uncertainty metric technique on the key nodes to achieve the uncertainty metric of the prediction results of deep neural network.

B. Node Contribution Analysis

In a neural network, each neuron node represents a pattern of features contained in a sample, and the contribution of these nodes is differentiated. Some neuron nodes have large contributions and are known as key nodes and they occupy an important position in the network [13]. The adversarial samples and the clean samples without adding small perturbations are fed as input samples to the trained neural network to get the activation level of each neuron node within the neural network for the adversarial samples and the clean samples, respectively. Since the small perturbation causes the neural network to misjudge, it can be assumed that the small change mainly affects the neuron nodes that have a larger contribution within the neural network, i.e., the key nodes. By comparing and analyzing the activation level of the adversarial samples and the clean samples, the contribution of each neuron node can be obtained, and the network nodes with a large contribution can be considered as key nodes in the process of network operation.

In generating adversarial samples with malicious perturbations, we use the commonly used iterative adversarial attack method Projected Gradient Descent (PGD) [14].

$$\begin{cases} \mathrm{X}^0 = \mathrm{X} \\ \mathrm{X}^{\mathrm{N}+1} = Clip_{\mathrm{X}+\mathrm{S}}\left\{\mathrm{X}^{\mathrm{N}} + \alpha \mathrm{sign}[\nabla_{\mathrm{x}} \mathrm{J}\left(\theta, \mathrm{X}^{\mathrm{N}}, \mathrm{y}_{\mathrm{true}}\right)]\right\} \end{cases} \tag{1}$$

where X is the clean sample, X^{N} is the sample after N rounds of attack, α represents the step size of each attack perturbation, S represents the upper limit of the attack perturbation, $sign(\cdot)$ is the sign function and $Clip(\cdot)$ is the truncation function.

The clean and adversarial samples are sequentially fed into the deep neural network to be analyzed, and the feature maps F corresponding to the samples in each intermediate hidden layer of the network to be analyzed are obtained. For the convolutional layer, the feature map corresponding to a node is a two-dimensional vector with dimension h×w. In order to measure the activation of the node, we perform global max-pooling on the feature map of the convolutional layer to obtain the most significant features of the feature map and use this as the activation value of the node. For the fully connected layer, the feature map of each node is the activation value of that node, so no additional operation is needed. By doing this, the activation of the convolutional layer and the fully connected layer are measured uniformly.

$$A_i = \begin{cases} GMP(F_i) & layer_i \text{ is convolutional layer} \\ F_i & layer_i \text{ is fully connected layer} \end{cases} \tag{2}$$

where F_i represents the feature map of the $layer_i$ and A_i represents the activation value of the $layer_i$, $GMP(\cdot)$ represents the global max-pooling operation.

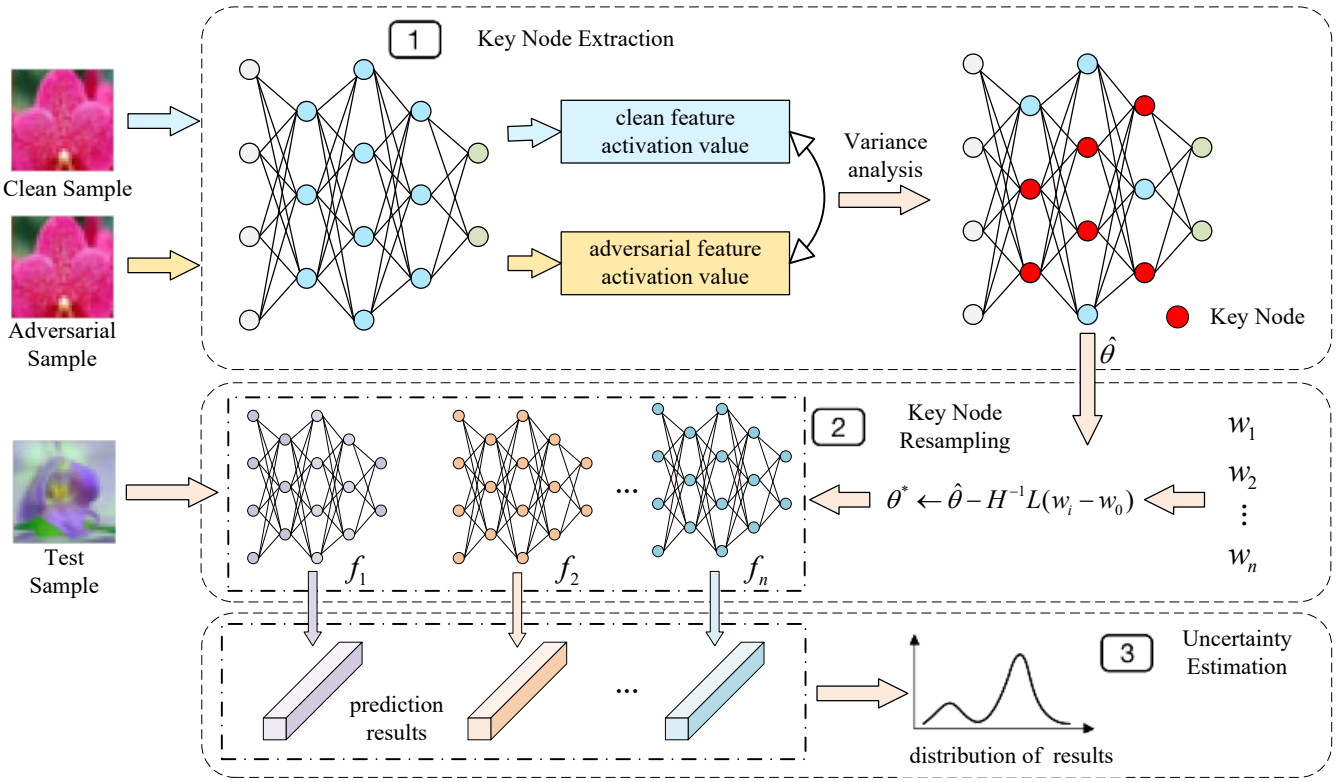

Fig. 1. The overall framework of the proposed KN-RUE. It contains key node extraction, key node resampling, and uncertainty estimation.

After obtaining the activation values for each node, the location of the model error caused by the adversarial sample can be localized by measuring the difference in activation values between the adversarial sample and the clean sample. Since there is only a slight difference between the adversarial sample and the clean sample in the data space, the same difference between the adversarial sample and the clean sample in the feature space is also concentrated only in some of the nodes. Since the change in some of the nodes leads to a change in the final result, we believe that the magnitude of the difference between the adversarial sample and the clean sample in the feature space can be a measure of how much the node contributes to the task of the neural network, and the nodes with a large difference play an important role in the neural network and can be regarded as the key nodes.

$$R_i = \mid A_i^c - A_i^a \mid \tag{3}$$

$$C_i = \frac{R_i}{\sum_{i=1}^{n} R_i} \tag{4}$$

where A^c and A^a represent the activation values of clean and adversarial samples on each node of the neural network respectively, R represents the importance degree of the node, and C represents the contribution degree of the node in the neural network.

After obtaining the contribution of each node, some of the nodes with the largest contribution can be filtered and considered as key nodes. In the subsequent resampling process, only the parameters of the key nodes are manipulated.

C. Key Node Resampling Uncertainty Estimation

Resampling uncertainty estimation (RUE) [15] is achieved by resampling the parameters of the neural network to obtain multiple extensions of the original model, thus achieving uncertainty estimation of the prediction results based on the prediction results of multiple extensions. The resampling uncertainty estimation technique can achieve the uncertainty measure of the prediction results of the neural network, but the computational amount and complexity of the resampling algorithm increase dramatically with the increase of the number of model parameters. In order to reduce the computational amount of resampling, we resample the key nodes to achieve network expansion, ensure the independence and effectiveness of the extended model as much as possible, and achieve the uncertainty measure of the prediction results of large-scale neural networks.

The function f of a deep neural network is defined by a vector of parameters $\theta \in \mathbb{R}^d$, and the deep neural network learns from the training data by minimizing the objective function:

$$J_D(\theta) = \sum_{i=1}^{n} \ell\big(y_i, f\big(x_i, \theta\big)\big) + R(\theta) \tag{5}$$

where ℓ is a loss function and $R(\theta)$ is a regularizer.

When learning on the original sample, let $\hat{\theta}$ represents the learning parameters that satisfy the following conditions:

$$\nabla J_D(\hat{\theta}) = \sum_{i=1}^{n} \nabla_\theta \ell_i(\hat{\theta}) + \nabla_\theta R(\hat{\theta}) = C \tag{6}$$

The resampling process for a trained deep neural network is an approximation of the Bootstrap process [16]. Bootstrap approximately estimates the sampling distribution of the samples by repetitively simulating Bootstrap samples, which create a new dataset by sampling with replacement from a uniform distribution on the original dataset. Let w_i denote the multiplicative weight of the ith training sample in the objective function, the objective function of the bootstrap training process can be rewritten in the following form:

$$J_D(\theta; w) = \sum_{i=1}^{n} w_i \ell_i(\theta) + R(\theta) \tag{7}$$

The bootstrap equivalent is to choose new weights to count the number of times a particular example is included in the bootstrap sample, and then readjust the target. For the bootstrap model, the following equation is valid for the solution:

$$\nabla_\theta J_D(\hat{\theta}; w) = L w_0 + \nabla_\theta R(\hat{\theta}) = C \tag{8}$$

where L is a matrix in which column i is the loss gradient of the ith sample $\nabla_\theta \ell_i(\hat{\theta})$, and $w_0 \in \mathbb{R}^n$ is a vector.

Assuming that $\nabla_\theta J_D(\hat{\theta}; w)$ is smooth on θ (the loss and regularizer are twice continuously differentiable), we can use the implicit function theorem to derive the existence of a local function such that $\phi(w_0) = \hat{\theta}$. An implicit function F expressing the relationship between w and θ can be obtained from Eqs. (6) and (8):

$$F = L w_0 - \sum_{i=1}^{n} \nabla_\theta \ell_i(\hat{\theta}) \tag{9}$$

This suggests that there exists a mapping from the sample weights to the learned model parameters, and it is possible to directly sample the new weights and map them to the parameters, thus obtaining the parameters corresponding to the new weights. Our approach approximates this function using a first-order Taylor expansion that:

$$\begin{aligned} \theta^* &\approx \hat{\theta} + \frac{\partial \phi}{\partial w}|_{w_0}(w^* - w_0) \\ &= \hat{\theta} + \frac{\frac{\partial F}{\partial w_0}}{\frac{\partial F}{\partial \phi}}(w^* - w_0) \\ &= \hat{\theta} + \frac{L}{-\nabla^2_\theta J_D(\theta)}(w^* - w_0) \\ &= \hat{\theta} - H^{-1} L (w^* - w_0) \end{aligned} \tag{10}$$

where the Hessian matrix $H = \nabla^2_\theta J_D(\hat{\theta})$.

After obtaining the mapping relation from the sample weights to the learned model parameters, we only need to formulate new weights w^* to sample the parameters of the key nodes in the neural network through the mapping relation to obtain θ^*. To approximate the sampling distribution of the model predictions at the sample points, we compute the predictions for the set of models with key node parameters $\{\theta_i^*\}_{i=1}^b$. The algorithm for key node resampling uncertainty estimation is shown in algorithmic 1:

Algorithm 1 KN-RUE

Input: training set D, trained model f, the ratio of key nodes r, number of training data m, number of network extensions b, test data x_{test}, damping term λ.

Output: network output result Y

Generate the adversarial set D^a according to Eq. (1)
Feed the D^a and the D into the network f
Obtain A^a and A^c according to Eq. (2)
Choose the top r nodes of contribution as key nodes and remember the key node parameter as θ
for $i = 0, 1, \ldots, n-1$ **do**
    $L(:, i) \leftarrow \nabla_\theta \ell_i(\theta)$
end for
$H \leftarrow \nabla^2_\theta J_D(\hat{\theta})$
$\tilde{H} \leftarrow H + \lambda * \mathrm{EYE}(d)$
$A \leftarrow \tilde{H}^{-1} L$
$w_0 \leftarrow \mathrm{ONES}(n)$
$p_0 \leftarrow w_0 / n$
for $i = 0, 1, \ldots, b-1$ **do**
    $w_i \sim \mathrm{Multinomial}(n, p_0)$
    $\theta_i^* \leftarrow \theta - A(w_i - w_0)$
    $Y(i) = f_{\theta_i^*}(x_{test})$
end for
return Y

III. Experiments

A. Experimental Setting

In order to validate the effectiveness of our method, we are conducting simulation validation work based on the publicly available benchmark classification datasets CIFAR-10 [17] and CIFAR-100 [17] as well as the commonly used VGG16 [18] and ResNet18 [19] networks.

CIFAR-10 is a dataset widely used for image recognition and computer vision tasks and consists of 60,000 color images of 32x32 pixels with 10 categories. These categories are plane, car, bird, cat, deer, dog, frog, horse, boat and truck.CIFAR-100 is a dataset for image classification tasks, similar to CIFAR-10 but more challenging.CIFAR-100 contains a total of 100 categories with 600 images per category.

We first trained the VGG16 and ResNet18 networks on the CIFAR-10 dataset with 40 training epochs and a learning rate of 0.01, respectively, and used cosine decay for the learning rate to decrease to 0. The VGG16 and ResNet18 networks with good performance are obtained after training. The accuracy of VGG16 is 99.77% in the CIFAR-10 train set and 92.51% in the test set; the accuracy of ResNet18 is 99.86% in the CIFAR-10 train set and 93.54% in the test set.

In order to verify the effectiveness of the model extended by key node resampling, we selected MC dropout [20] as

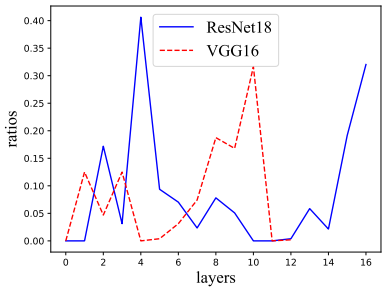

Fig. 2. Distribution of the ratio of key nodes in each layer of the ResNet18 and VGG16 networks.

TABLE I
THE ACCURACY AND AUC OF THE FUSION RESULTS OF THE EXTENDED MODELS BY MC DROPOUT AND KN-RUE.

	MC Dropout [20]		KN-RUE	
	VGG16	ResNet18	VGG16	ResNet18
Accuracy	0.4140	0.6430	0.8921	0.9316
AUC	0.9739	0.9767	0.9950	0.9959

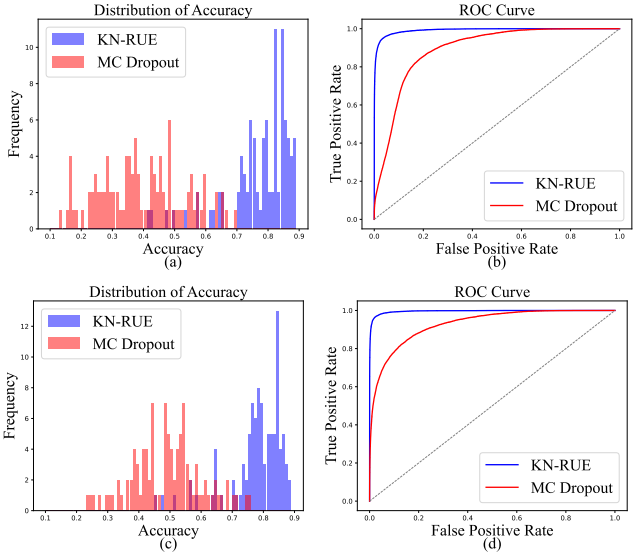

Fig. 3. The distribution of accuracy of the extended model by KN-RUE and MC Dropout algorithm as well as the fusion result ROC curves. (a) and (b) show the experimental results on the VGG16 network; (c) and (d) show the experimental results on the ResNet18 network.

the comparison algorithm for network extension. MC dropout achieves the extension of a given network by repeating the dropout operation on the network several times during the testing process so that the outputs of the input samples can be obtained from the extended network with different structures with the extended network. In the experimental process for the key node resampling and MC dropout, both extended 100 networks. The evaluation metrics are each extended model's accuracy and the combined results obtained by the probability-averaging decision fusion algorithm.

In order to verify that the key node resampling uncertainty estimation algorithm has good uncertainty estimation capability, we further analyze the performance of the extended models on the CIFAR-10 train set, CIFAR-10 test set, and CIFAR-100 test set. The standard deviation of the confidence level of all extended models on the combined outcome categories obtained from the probability-averaged decision fusion algorithm is used as the uncertainty estimate of the prediction results.

B. Experimental Results

Firstly, we perform node contribution analysis on VGG16 and Resnet18 networks and consider the nodes with the top 10% contribution as key nodes. Figure 2 shows the distribution of key points in VGG16 and Resnet18 respectively. From the figure, it can be seen that the key nodes in both VGG16 and Resnet18 show an aggregated distribution in the front part and the back part of the network, which may be due to the fact that the front network nodes are mainly responsible for extracting the useful features from the images and the back network nodes are mainly responsible for mapping the abstract feature information into the prediction results, so these two parts have a greater contribution to the network, compared to the intermediate nodes, which are mainly responsible for feature transformation, have a smaller contribution.

We further compare KN-RUE with the MC Dropout algorithm. In KN-RUE, we resample the parameters of the selected nodes with the top 10% contribution to get 100 extended models. In MC Dropout, we randomly select 10% of the nodes from the network for Dropout, thus obtaining 100 extended models with different structures. Then we test the accuracy of these 200 extended models and the accuracy and AUC value of the fusion decision results through the extended models respectively. Figure 3 shows the accuracy of the extended models obtained by the KN-RUE and MC Dropout methods and the AUC curves of the fusion results. From the figure, it can be seen that the distribution of the accuracy of the extended model obtained by KN-RUE is concentrated in the higher part, indicating that resampling preserves the performance of the network, while the distribution of the accuracy of the MC Dropout extended model is concentrated in the lower part, indicating that Dropout destroys the original performance of the network to a certain extent, and the same can be reflected from the AUC curve that the KN-RUE fusion results are superior compared to MC Dropout. Table I presents the accuracy and AUC values of the fusion results of KN-RUE and MC Dropout. From the table, it can be seen that the KN-RUE fusion results are superior to the MC Dropout results on both VGG16 and ResNet18, and the KN-RUE fusion results are very close to the original network performance.

To illustrate the uncertainty estimation capability of KN-RUE, we take the CIFAR-10 training set, the CIFAR-10 test set, and the CIFAR-100 test set, which has a different distribution from the CIFAR-10 dataset, as test samples, and compute the results of the uncertainty estimation of the KN-RUE algorithm for the prediction results of the samples in these datasets, respectively. Figure 4 illustrates the uncertainty estimation test results on the VGG16 and ResNet18 networks.

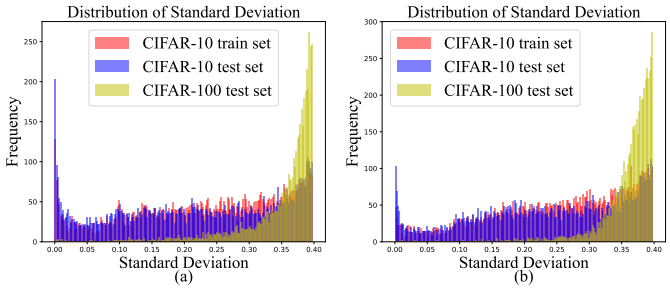

Fig. 4. Distribution of uncertainty (standard deviation of extended model prediction confidence) for KN-RUE on different samples of the dataset. (a) shows the experimental results on the VGG16 network; (b) shows the experimental results on the ResNet18 network.

From the figure, it can be seen that the uncertainty distribution of the samples in the CIFAR-10 train set and the test set are consistent and most of the data have low uncertainty. This is due to the fact that the samples in the CIFAR-10 test set and the training set obey the same distribution, and after training, the samples can be accurately classified for the samples under this distribution. Whereas the CIFAR-100 test set uncertainty distribution is gathered at a higher position, this is because the samples in the CIFAR-100 dataset have an unknown distribution pattern for the network, so the network is not able to make an accurate judgment, thus giving a high uncertainty. The experimental results of uncertainty distribution on different datasets are in line with our expectation of the network in facing out-of-distribution data, thus KN-RUE has the excellent uncertainty estimation ability.

IV. Conclusions

In this paper, the resampling uncertainty estimation (RUE) algorithm is improved to address the need for uncertainty estimation of large-scale neural networks, and the key node resampling uncertainty estimation (KN-RUE) is proposed to achieve uncertainty estimation of prediction results for any given large-scale neural network. In this method, firstly, the neural network is evaluated for node importance, and the key nodes in the network are identified by analyzing the differences between the adversarial samples and the clean samples in the feature space and locating the main nodes affected by the adversarial samples. Secondly, by resampling the parameters of the key nodes, the model expansion is achieved under the premise of guaranteeing the model performance as much as possible, and the uncertainty estimation for the prediction results is completed. Finally, experimental evaluations on commonly used datasets (CIFAR-10, CIFAR-100) and models (VGG16, ResNet18) are conducted to further validate the effectiveness of KN-RUE.

References

[1] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou, and J. Lu, "Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 21 729–21 740.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] Z. Chen, D. Agarwal, K. Aggarwal, W. Safta, M. M. Balan, and K. Brown, "Masked image modeling advances 3d medical image analysis," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 1970–1980.

[4] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 369–385.

[5] M. Seeger, "Gaussian processes for machine learning," *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

[6] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[7] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *stat*, vol. 1050, p. 3, 2018.

[8] R. Novak, L. Xiao, J. Lee, Y. Bahri, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Bayesian deep convolutional networks with many channels are gaussian processes," *arXiv preprint arXiv:1810.05148*, 2018.

[9] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison, "Deep convolutional networks as shallow gaussian processes," in *International Conference on Learning Representations*, 2019, pp. 1–16.

[10] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[11] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[12] V. Nemani, L. Biggio, X. Huan, Z. Hu, O. Fink, A. Tran, Y. Wang, X. Zhang, and C. Hu, "Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial," *Mechanical Systems and Signal Processing*, vol. 205, p. 110796, 2023.

[13] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[14] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[15] P. Schulam and S. Saria, "Can you trust this prediction? auditing pointwise reliability after learning," in *The 22nd international conference on artificial intelligence and statistics*. PMLR, 2019, pp. 1022–1031.

[16] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[17] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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