applied sciences

MDPI

Article

Bayesian Convolutional Neural Network-Based Power System Transient Stability Assessment with Uncertainty Estimation

Luqin Fan [1,2], Mengwei Yuan [3], Yu He [1,2,*], Jing Zhang [1,2], Xu Wang [3] and Xiaopeng Wang [4]

[1] School of Electrical Engineering, Guizhou University, Guiyang 550025, China; lqfan@gzu.edu.cn (L.F.); zhangjing@gzu.edu.cn (J.Z.)
[2] Guizhou Provincial Key Laboratory of Power System Intelligent Technologies, Guiyang 550025, China
[3] Duyun Power Supply Bureau, Guizhou Power Grid Company, Duyun 558000, China; mwyuan1119@163.com (M.Y.); wxttty1113@126.com (X.W.)
[4] School of Computer Science and Technology, Guizhou University, Guiyang 550025, China; wxpbird1978@gmail.com
* Correspondence: yhe7@gzu.edu.cn

Abstract

Transient stability assessment (TSA) is pivotal to power system security and stability, with online TSA providing core support for operators to make emergency control decisions. While deep learning has driven significant advancements in TSA, most existing deterministic models fail to quantify output uncertainty—a critical factor for evaluating prediction confidence and enabling operators to make informed decisions. This paper proposes a Bayesian Convolutional Neural Network (BCNN)-based TSA method that enables uncertainty estimation. By modeling the weights of network parameters as Gaussian distributions and adopting the Bayesian backpropagation algorithm for weight updates, the approach enhances TSA's robustness against PMU noise and missing data. A case study on the New England 39-bus system demonstrates the model's superior adaptability to small sample sizes and effective uncertainty estimation. Furthermore, the confidence intervals of evaluation results serve as valuable support for operators' decision-making.

Keywords: transient stability assessment; Bayesian convolutional neural network; uncertainty estimation; uncertainty threshold

1. Introduction

The transient stability of a power system refers to the ability of each synchronous generator to maintain synchronous operation and either transition to a new state or return to its original steady-state mode after a significant disturbance [1]. As modern power systems expand and the interconnections among various regional grids become more complex, these systems exhibit high-dimensional nonlinear characteristics. Consequently, an effective, accurate, and rapid TSA is essential, as it provides dispatchers with crucial stability analysis for various incidents within power systems.

Conventional methods for transient stability assessment primarily include time-domain simulation (TDS) [2], which is widely regarded as the most accurate and reliable approach. TDS solves the system's differential-algebraic equations using numerical integration techniques and assesses stability based on the disturbed trajectory of the system. However, TDS is computationally intensive and inefficient, making online implementation challenging. Alternatively, the direct method [3] employs an energy function derived

Academic Editor: Roberto Zivieri

Received: 12 October 2026
Revised: 9 November 2026
Accepted: 12 November 2026
Published: 16 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

from Lyapunov theory to analyze the system's transient stability, providing rapid conclusions regarding stability levels. However, its application is limited by the difficulty of constructing accurate energy functions for complex power systems. Moreover, this method demands significant computational effort for various load levels and fault clearing times. Consequently, conventional methods struggle to meet the online application requirements for transient stability assessment.

The widespread adoption of phasor measurement units (PMUs) [4] in power systems has resulted in the accumulation of vast amounts of power data, facilitating the development of data-driven methods. Data-driven methods for transient stability assessment mainly include traditional machine learning and deep learning. Traditional machine learning techniques can effectively map input features to transient stability states, with various methods, such as support vector machines (SVM) [5], decision trees (DT) [6], and extreme learning machines (ELM) [7]—successfully applied to transient stability assessment. However, due to the limitations of traditional machine learning's shallow network structure, its feature extraction ability is insufficient, often requiring expert experience for assistance, which introduces subjectivity into the model and weakens the generalizability of the evaluation results.

Deep learning methods automatically learn hierarchical representations from raw data rather than relying on expert-designed handcrafted features. In recent years, these methods have been extensively applied to transient stability assessment. Ref. [8] utilized Convolutional Neural Networks (CNNs) to predict transient stability and instability modes, aiding operators in devising emergency control measures. Ref. [9] introduced a cascaded CNN model based on an information entropy strategy, which allows for early termination of time-domain simulations, thereby conserving computational resources. Ref. [10] proposed a prediction framework using long short-term memory networks (LSTMs), enabling rapid identification of power system stability. Additionally, other innovative deep learning models, such as deep belief networks (DBNs) [11], gated recurrent units (GRUs) [12], and graph-based deep learning techniques [13], have also been applied to transient stability assessment. The above studies design corresponding neural network structures based on the temporal features of transient data or the spatial structural features of the power grid to explore the mapping relationship between data and transient stability states, achieving good assessment results. However, these models do not analyze the reliability of the assessment results, which limits their practical application.

Power system operators are increasingly focused on the trustworthiness of neural network output results. In Ref. [14], stability thresholds were introduced to classifier outputs to balance the trade-off between accuracy and assessment time. Similarly, Ref. [15] applied probabilistic neural networks (PNNs) to transient stability assessment and established probability distributions for insecure domains. However, both studies failed to address the uncertainty generated by the models themselves, leaving a critical gap in ensuring the reliability of the predictions.

To address uncertainty, Bayesian deep learning (BDL) can be employed, which integrates Bayesian theory with deep learning by assuming that model parameters follow specific probability distributions. BDL models provide outputs with distributional characteristics, enabling uncertainty estimation. Research has demonstrated that uncertainty estimation in BDL outputs is crucial for the model's acceptance, particularly in safety-critical domains such as medical diagnosis [16,17], natural disaster warnings [18], traffic accident detection [19], and fault risk prediction [20] in distribution systems. In the context of power system stability, Bayesian neural networks (BNNs) have been applied to stability prediction and dynamic security assessment [21,22]. However, the BNN model used in Ref. [21] consists solely of fully connected layers, limiting its ability to effectively handle

multi-input time series data. As input data scales up, the model faces the issue of parameter growth, making it challenging for BNNs to extract features from such data. Ref. [22] establishes a transient stability assessment model by combining Bayesian networks and an active learning mechanism, while also utilizing posterior probabilities to adaptively assess prediction confidence, but this model only considers model uncertainty and does not account for data uncertainty.

In this paper, we integrate Convolutional Neural Networks with Bayesian theory to develop a Bayesian Convolutional Neural Network (BCNN) for transient stability assessment. The proposed method utilizes a one-dimensional convolutional neural network (1D-CNN) to automatically extract temporal features of transient processes while modeling the uncertainty arising from PMU measurement data [P, Q, V, θ] and the model itself. Test results on the New England 39-bus system demonstrate that the BCNN model can effectively quantify uncertainty while maintaining high prediction accuracy. Moreover, the output exhibits robustness and adaptability to small datasets. The key contributions of this paper are as follows:

(1) This paper proposes a transient stability assessment framework that integrates a one-dimensional Convolutional Neural Network and Bayesian theory. The proposed method uses a Convolutional Neural Network for feature extraction while employing Bayesian theory to quantify data and model uncertainty. This incorporation of uncertainty enhances the reliability of the neural network's prediction results.
(2) This paper compares the robustness and adaptability to small datasets of Bayesian models against non-Bayesian models. Simulation results demonstrate that integrating Bayesian theory significantly enhances the robustness and adaptability of the Bayesian model when faced with small datasets.
(3) This paper analyzes the sources and influencing factors of uncertainty and employs an uncertainty threshold derived from the results of uncertainty quantification to filter out samples with high uncertainty. Operators can adjust the uncertainty threshold according to specific requirements, thereby improving prediction accuracy and making the results more instructive.

The rest of this paper is organized as follows: Section 2 describes the proposed methodology. Section 3 introduces the prediction framework design. Section 4 provides results and discussions. Section 5 summarizes the conclusions.

2. Bayesian Convolutional Neural Network

2.1. Bayesian Convolutional Neural Network Description

1D-CNN extracts features from the input data across different time windows by sliding the convolution kernel in a single direction, making it suitable for processing time-series data. In 1D-CNN, model parameters are usually set as unknown values. In contrast, Bayesian neural networks treat these model parameters as probability distributions (i.e., unknown variables) rather than fixed values [23]. This probabilistic approach captures the uncertainty in weight parameters and enables the estimation of predictive uncertainty. Figure 1 illustrates the structural differences between CNN and BCNN.

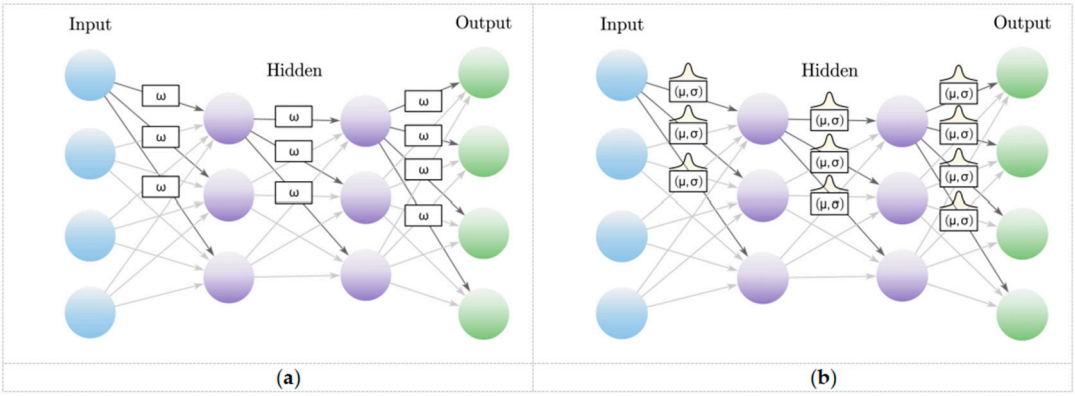


Figure 1. The comparison of the structures of CNN and BCNN: (**a**) frequentist CNN: Each filter weight has a fixed value; (**b**) Bayesian CNN: Each filter weight has a distribution.

For a given transient stability assessment dataset $\mathcal{D}$, the input data is $X=\{x_1,...,x_N\}$ and the corresponding output is $Y=\{y_1,...,y_N\}$. To estimate the parameters that generate the given $\mathcal{D}$, we can use Bayesian reasoning by assigning a prior distribution over the parameter space $p(\omega)$. Typically, the prior distribution is modeled as a Gaussian distribution. In this case, the likelihood function is defined as $p(Y|\omega,X)$, representing the probability model of the output given the parameters ω. For classification problems, the likelihood function is

$$p(y=d|x,\omega)=\frac{\exp\left(f_d^{\omega}(x)\right)}{\sum_{d'}\exp\left(f_{d'}^{\omega}(x)\right)} \tag{1}$$

where d denotes the category of the classifier output.

For the probability model of the given dataset $\mathcal{D}$, we can apply Bayes' theorem to obtain the posterior distribution under the parameter space $p(\omega)$:

$$p(\omega|X,Y)=\frac{p(Y|X,\omega)\,p(\omega)}{p(Y|X)} \tag{2}$$

The posterior distribution describes the most probable model parameters given the dataset, a process referred to as inference.

The denominator $p(Y|X)$ of Equation (2) represents the distribution of the true data, expressed as Equation (3).

$$p(Y\mid X)=\int p(Y\mid X,\omega)p(\omega)d\omega \tag{3}$$

The difficulty in obtaining the posterior distribution of the parameter space lies in calculating Equation (3), which requires us to traverse all ω. In the case of finite parameters ω, $p(Y\mid X)$ may be calculated, but in BCNN there are thousands of parameters ω, and it is impossible to obtain the posterior distribution by direct calculation. As a result, approximate methods for posterior inference are typically employed.

2.2. Variational Inference

Two common methods for approximating the posterior distribution are Markov Chain Monte Carlo (MCMC) [24] and Variational Inference (VI) [15]. MCMC generates samples from the posterior distribution and computes statistics such as mean and variance from these samples. While MCMC provides high accuracy, it is not well-suited for large-

scale models. Therefore, we use VI, which is more suitable for BCNNs, to perform approximate computation.

VI approximates the posterior distribution by introducing a variational distribution $q_\theta(\omega)$ over ω, parameterized by θ, to approximate the true posterior distribution $p(\omega|X,Y)$. The parameters θ are optimized to make $q_\theta(\omega)$ as close as possible to $p(\omega|X,Y)$. This is typically achieved by minimizing the Kullback–Leibler (KL) divergence, a measure of the similarity between the two distributions.

$$\mathrm{KL}(q_\theta(\omega) \,||\, p(\omega \mid X,Y)) = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega \mid X,Y)} d\omega \tag{4}$$

The goal of variational inference is to solve the following optimization problem:

$$q_\theta^*(\omega) = \arg\min \mathrm{KL}(q_\theta(\omega) \,||\, p(\omega \mid X,Y)) \tag{5}$$

Equation (5) represents the objective function for minimizing the KL divergence. According to Bayesian inference, the predictive distribution can be approximated as follows:

$$p(y^* \mid x^*, X, Y) \approx \int p(y^* \mid x^*, w) q_\theta^*(\omega) d\omega =: q_\theta^*(y^* \mid x^*) \tag{6}$$

By further deriving Equation (5), minimizing the KL divergence can be reformulated as maximizing the Evidence Lower Bound (ELBO), expressed as follows:

$$\begin{aligned} &\mathrm{KL}(q_\theta(\omega) \,||\, p(\omega \mid X,Y)) \\ &= \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega \mid X,Y)} d\omega \\ &= \mathbb{E}_{q_\theta(\omega)} \left[\log q_\theta(\omega) - \log p(\omega) \right. \\ &\left. -\log p(Y|X,\omega) + \log p(Y|X) \right] \end{aligned} \tag{7}$$

Since $\log p(Y|X)$ is independent of ω, we can further simplify it as follows:

$$\mathrm{KL}(q_\theta(\omega) \,||\, p(\omega \mid X,Y)) = -\mathrm{ELBO}(\omega) + \log p(Y \mid X) \tag{8}$$

Since $\log p(Y \mid X)$ is a constant that depends only on the data set $\mathcal{D}$, minimizing the KL divergence between the variational distribution $q_\theta(\omega)$ and the true posterior distribution $p(\omega|X,Y)$ can be minimized by maximizing Evidence Lower Bound (ELBO), which transforms the task of finding the posterior distribution into an optimization problem that can be efficiently solved using stochastic gradient descent using neural networks. The ELBO function is defined as follows:

$$\begin{aligned} \mathrm{ELBO}(\omega) = \mathbb{E}_{q_\theta(\omega)} \left[\log p(Y \mid X, \omega) \right] \\ -\mathrm{KL}\left(q_\theta(\omega) \,||\, p(\omega) \right) \end{aligned} \tag{9}$$

Since the loss function is usually in a minimization form, Equation (9) can be expressed as minimizing a cost function:

$$\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}\left(q_\theta(\omega) \,||\, p(\omega) \right) - \mathbb{E}_{q_\theta(\omega)} \left[\log p(\mathcal{D}|\omega) \right] \tag{10}$$

2.3. Bayesian Backpropagation Algorithm

The Bayesian backpropagation algorithm, introduced by Blundell, aims to learn the posterior distribution of neural network weights, allowing for sampling during the backpropagation process [25]. Accordingly, Equation (10) can be rearranged based on the definition of KL divergence, as follows:

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) = \mathbb{E}_{q(\omega|\theta)} \log q\left(\omega\middle|\theta\right) - \mathbb{E}_{q(\omega|\theta)} \log p\left(\omega\right) \\ -\mathbb{E}_{q(\omega|\theta)} \log p\left(\mathcal{D}\middle|\omega\right)\end{aligned} \tag{11}$$

Similar to traditional neural networks, which update parameters through backpropagation, Bayesian neural networks utilize a backpropagation equation $\theta^{'} = \theta - \alpha\nabla_{\theta}\mathcal{F}$. In this context, the three terms in Equation (11) represent expectations of the variational distribution $q_{\theta}(\omega)$. Therefore, Monte Carlo methods can be employed to sample ω from $q_{\theta}(\omega)$. The result of the Monte Carlo approximation is as follows:

$$\begin{aligned}\mathcal{F}\left(\mathcal{D},\theta\right) \approx \frac{1}{n}\sum_{i=1}^{n}\Big[\log q(\omega^{(i)} \mid \theta) \\ -\log p\left(\omega^{(i)}\right) - \log p\left(\mathcal{D} \mid \omega^{(i)}\right)\Big]\end{aligned} \tag{12}$$

where n is the number of samples, $\omega^{(i)}$ represents the ith Monte Carlo sample obtained from the variational posterior distribution $q(\omega^{(i)} \mid \theta)$. To make Equation (12) backpropagable, we use the reparameterization trick [26].

The steps of the Bayesian backpropagation algorithm are as follows:

(1) Assume that $q(\omega^{(i)} \mid \theta)$ follows a Gaussian distribution, and randomly sample a vector $\epsilon \sim \mathcal{N}(0, I)$ from the Gaussian function.

(2) Let $\begin{aligned}\omega &= \mu + \sigma \odot \epsilon \\ &= \mu + \log(1+\exp(\rho)) \odot \epsilon \\ &= \mu + \text{softplus}(\rho) \odot \epsilon\end{aligned}$.

Where $\odot$ denotes pointwise multiplication, this equation means that the weight ω can be obtained by scaling according to the standard deviation σ and shifting by a mean μ after sampling from the Gaussian distribution. Using softmax to describe the standard deviation σ can ensure that the standard deviation is always positive. The variational parameters are $\theta = (\mu, \rho)$.

(3) Let $f(\omega,\theta) = \log q(\omega|\ \theta) - \log P\left(\omega\right)P(\mathcal{D}|\ \omega)$.

(4) Calculate the gradients of μ and ρ:

$$\begin{aligned}\Delta_{\mu} &= \frac{\partial f(\omega,\theta)}{\partial \omega} + \frac{\partial f(\omega,\theta)}{\partial \mu} \\ \Delta_{\rho} &= \frac{\partial f(\omega,\theta)}{\partial \omega}\frac{\epsilon}{1+\exp(-\rho)} + \frac{\partial f(\omega,\theta)}{\partial \rho}\end{aligned} \tag{13}$$

(5) Update parameters μ and ρ.

2.4. Uncertainty Estimation

In Bayesian deep learning, two types of predictive uncertainty—aleatoric uncertainty and epistemic uncertainty—can be modeled in neural networks. Aleatoric uncertainty refers to the inherent noise in the observed data, which is independent of model errors and remains even as more data becomes available. In contrast, epistemic uncertainty arises from model inaccuracies and can be reduced with additional training data. Together, these uncertainties are referred to as predictive uncertainty (PU), which reflects the

model's confidence in its predictions. Aleatoric uncertainty is typically captured by modeling the outputs of the neural network as probability distributions. Epistemic uncertainty, on the other hand, is modeled by treating the network's parameters as random variables drawn from a prior distribution (often Gaussian) and learning their variation based on the data.

In Section 2.3, we assumed that $q(\omega \mid \theta) \sim \mathcal{N}(\omega \mid \mu, \rho)$. In classification problems, the predictive distribution is usually assumed to be a categorical distribution, so Equation (6) can be rewritten as follows:

$$p_{\mathcal{D}}(y^* \mid x^*) = \int \mathrm{Cat}(y^* \mid f_{\omega}(x^*)) \mathcal{N}(\omega \mid \mu, \rho) d\omega \tag{14}$$

Since there is no correlation between the categorical and Gaussian distributions, the expected value of this distribution cannot be directly computed. However, an unbiased estimate of this expected value can be obtained through Monte Carlo sampling from the variational distribution $q(\omega \mid \theta)$.

$$\mathbb{E}q[p_{\mathcal{D}}(y^* \mid x^*)] \approx \frac{1}{T}\sum_{t=1}^{T} p_{\omega_t}(y^* \mid x^*) \tag{15}$$

where T represents the number of the sample. References [27,28] define uncertainty as the variance of the predictive distribution and propose an efficient method for calculating this uncertainty.

$$\begin{aligned} &\mathrm{Var}_q\left(p(y^* \mid x^*)\right) \\ &= \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathrm{diag}(\hat{p}_t) - \hat{p}_t \hat{p}_t^{\,T}}_{Aleatoric\ Uncertainty} + \underbrace{\frac{1}{T}\sum_{t=1}^{T} (\hat{p}_t - p)(\hat{p}_t - p)^T}_{Epistemic\ Uncertainty} \end{aligned} \tag{16}$$

where $p = \frac{1}{T}\sum_{t=1}^{T} \hat{p}_t$ and $\hat{p}_t = \mathrm{soft\,max}\left(f_{w_t}(x^*)\right)$.

By separating predictive uncertainty into aleatoric and epistemic uncertainty, more insight can be gained into the stability of transient prediction results. This allows for the identification of whether samples with high uncertainty are due to low data quality (high aleatoric uncertainty) or poor model performance (high epistemic uncertainty).

3. Transient Stability Assessment Scheme

3.1. Model Description

3.1.1. Model Input

The deep learning model autonomously extracts and learns features from measurement data, eliminating the need for manual selection of secondary features and enabling an end-to-end approach for transient stability online assessment. This automated feature extraction ensures the completeness of the input information and removes the influence of subjective factors that may arise from manually selected features, thus enhancing the model's performance.

The main criteria for selecting features in this paper are: 1. The input data must be electrical quantities directly measured by PMUs; 2. The selected electrical quantities must effectively capture the system's dynamic behavior following a disturbance. The available features include generator angle, speed, generator terminal voltage, bus voltage phasor, transmission line active/reactive power on transmission lines. As noted in Ref. [29], the bus voltage phasor typically responds more quickly to faults than variables such as generator angle and speed.

Consequently, this paper selects bus voltage magnitude, angle, and transmission line active/reactive power flow as the input features, represented as follows:

$$\mathbf{X} = \left[\mathbf{V}_1, \cdots, \mathbf{V}_m, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_m, \mathbf{P}_1, \cdots, \mathbf{P}_n, \mathbf{Q}_1, \cdots, \mathbf{Q}_n\right] \tag{17}$$

where $\left(\mathbf{V}_m, \boldsymbol{\theta}_m, \mathbf{P}_n, \mathbf{Q}_n\right) \in R^T$, m is the bus line number, n is the transmission line number, and T is the number of sampling points. Therefore, the corresponding input feature $\mathbf{X}$ is a two-dimensional matrix of the form $\mathbf{R}^{T\times 2(m+n)}$. To better reflect the transient process after a fault, the sampling period should include three stages as much as possible: steady state before the fault, fault, and fault clearance. The maximum fault duration set in this simulation is 0.42 s. Therefore, we set the sampling interval of each sample to [tf −5T, tf +45T] and $T = 20$ ms. The test system used in this paper is the New England 10-machine 39 bus system, consisting of 39 bus lines and 34 transmission lines. From Equation (17), it follows that the input feature $\mathbf{X}$ is a 50 × 146 dimensional matrix.

3.1.2. Model Output

The output of the transient stability assessment model is a sample label, as the transient stability assessment is formulated as a binary classification problem. The transient stability index (TSI) [30] is commonly used to evaluate the angle stability of the system following a fault. TSI is expressed as follows:

$$\mathrm{TSI} = \frac{360° - \left|\Delta\delta_{\max}\right|}{360° + \left|\Delta\delta_{\max}\right|} \tag{18}$$

where $\Delta\delta_{\max}$ represents the maximum relative angle between two generators during the simulation time. If TSI > 0, the sample is considered stable and the label is set to 1. If TSI ≤ 0, the sample is deemed unstable and the label is set to 0.

3.1.3. Data Preprocessing

Since the input features include bus voltage amplitude, voltage phase angle, active power and reactive power of the transmission line, all of which have different dimensions, data normalization is necessary. This paper employs Min–Max normalization, which performs a linear transformation on the original data, mapping the normalized values to the range [0, 1]. For a sequence $\mathbf{X}$ to be normalized, we have

$$x' = \frac{x_i - \min\limits_{1\le j\le n}\{x_j\}}{\max\limits_{1\le j\le n}\{x_j\} - \min\limits_{1\le j\le n}\{x_j\}} \tag{19}$$

3.2. The Flow Chart of TSA

The proposed scheme comprises three stages: data generation, offline training, and online TSA. The transient stability assessment process based on the Bayesian Convolutional Neural Network is shown in Figure 2.

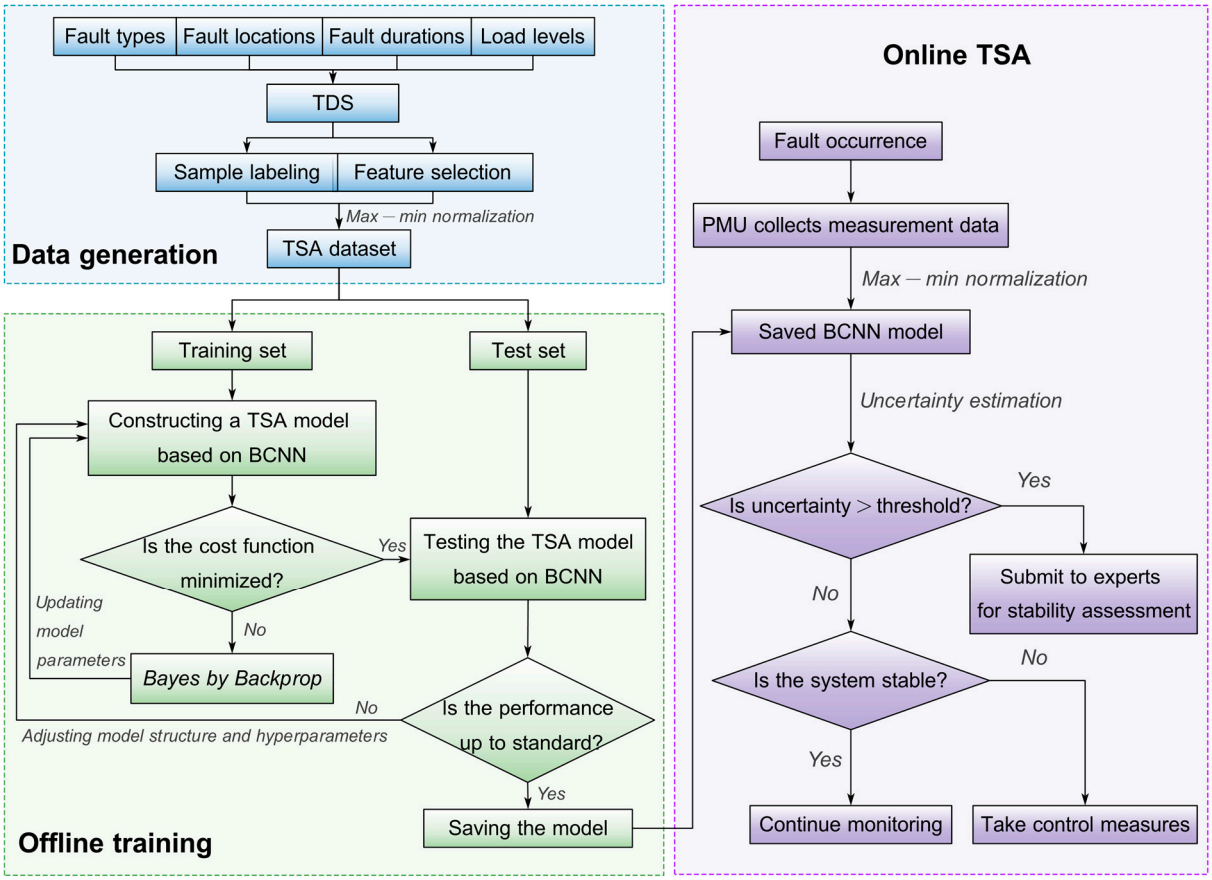


Figure 2. Transient stability assessment flow chart based on BCNN.

3.2.1. Data Generation

Considering various fault types, locations, and durations, time-domain simulations are performed under different load levels to capture the time-series data of transient processes. The TSI is used as the criterion to determine whether the system is stable or unstable, and each dataset is labeled accordingly. Then, the selected features are normalized to remove the influence of dimensions and form the dataset required for training the model.

3.2.2. Offline Training

During the offline training phase, BCNN is used to build a transient stability assessment model. This model learns the nonlinear mapping between input features and transient stability states based on training set samples, enabling feature extraction, classification prediction, and uncertainty estimation. Unlike traditional neural networks, BCNN employs a Bayesian backpropagation algorithm to update model parameters with the objective of minimizing the cost function through learning from the training data. After training, the model's generalization ability is evaluated using a test set to determine if its prediction accuracy meets the required standards. If the accuracy is insufficient, the model's hyperparameters and network structure are adjusted, and the training process is repeated until the desired performance is achieved. Once the model meets the set criteria, it is saved for deployment in the online application phase.

3.2.3. Online TSA

The PMU measurement data is normalized and input into the trained BCNN model. The model provides predictions on whether the system is stable or unstable and calculates the predictive uncertainty using Equation (16). If the predictive uncertainty exceeds a pre-

defined threshold, the evaluation result is deemed unreliable, and human experts are required to assess the transient stability state manually. If the predictive uncertainty falls below the threshold, the judgment is considered reliable. In such cases, the operator can implement appropriate control measures based on the model's prediction to minimize potential losses caused by faults.

3.3. Performance Metrics

In binary classification problems, the performance of the transient stability evaluation model can be assessed using metrics derived from the confusion matrix (as shown in Table 1). The key metrics include accuracy, precision, recall, and F1-score, which provide a comprehensive evaluation of the model's classification capabilities:

Table 1. Confusion matrix of the transient stability assessment.

Prediction	Stable	Unstable
Predicted as stable	T_s	F_s
Predicted as unstable	F_{us}	T_{us}

Where F_{us} refers to a false alarm, meaning that a stable sample is judged as an unstable sample. F_s refers to a false dismissal, which means that an unstable sample is judged as a stable sample.

Since transient stability evaluation problems prioritize a minority of samples—specifically, unstable samples—the precision and recall have been modified.

1. Accuracy: refers to the proportion of correct results among the total number of samples in a classification result. It is a statistical quantity for all samples, but its evaluation result for imbalanced samples is affected by the high proportion of samples.

$$\text{Accuracy} = \frac{T_s + T_{us}}{T_s + F_{us} + T_{us} + F_s} \times 100\% \tag{20}$$

2. Precision: the evaluation accuracy metric for predicting unstable samples. The higher its value, the fewer cases where stable samples are judged as unstable samples.

$$\text{Precision} = \frac{T_{us}}{T_{us} + F_{us}} \times 100\% \tag{21}$$

3. Recall: the evaluation accuracy metric for actual unstable samples. The higher its value, the fewer cases where unstable samples are judged as stable samples.

$$\text{Recall} = \frac{T_{us}}{T_{us} + F_s} \times 100\% \tag{22}$$

4. F1-score: harmonic mean of precision and recall rate and is used to evaluate the comprehensive performance of a model.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

4. Results of the New England 39 Bus Case Study

4.1. Test System

The New England 39 bus system [31] is widely used in the field of transient stability analysis. It represents a 345 kV power system in the state of New England, USA. This system comprises 10 generators, 39 nodes, 12 transformers, and 34 transmission lines. The base power for the system is set at 100 MVA, with a base voltage of 345 kV.

The transient stability assessment dataset was generated using the PSD-BPA [32] batch simulation tool. The fault settings for the simulation are as follows: changes in operating modes are reflected by load levels of 80%, 100%, and 120%. Since three-phase faults are the most severe type of fault, they are typically selected for time-domain simulations. Permanent three-phase faults are simulated on each of the 34 transmission lines (excluding transformers) at positions of 10%, 30%, 50%, 70%, and 90% along the lines. The fault clearing time varies from 0.1 s to 0.42 s in increments of 0.02 s, resulting in a total of 17 different fault clearing times. The simulation duration is set to 10 s, with a sampling frequency of 50 Hz. In total, 8670 time-domain simulation samples were generated, consisting of 7159 stable samples and 1511 unstable samples. The samples were divided into training and test sets at a ratio of 4:1. The same data partition was maintained throughout all experiments for model training and evaluation.

4.2. Experiment Specifications

The BCNN model training utilizes the TensorFlow Probability library (v0.13.0), which is based on the TensorFlow framework for probabilistic programming. The library offers functions that integrate probability methods with deep networks and also provides variational inference modules that are suitable for building and solving BCNN.

The proposed BCNN model comprises three 1D convolutional layers with the following configurations: the first layer features 16 filters with a kernel size of 11, the second layer has 32 filters with a kernel size of 7, and the third layer includes 64 filters with a kernel size of 1. To reduce the number of network parameters, the first two convolutional layers are followed by max pooling layers. At the bottom of the architecture, there is a fully connected layer containing 64 neurons, leading to an output layer that produces a two-dimensional vector. The initial prior distribution of the network parameters in the BCNN is modeled as a normal distribution $\mathcal{N}(0,1)$. Preliminary experiments were conducted to compare different hyperparameter settings, and the final settings were determined based on the resulting model performance. Additional hyperparameter settings are detailed in Table 2.

The network architecture and hyperparameter settings of the CNN model align with those of the BCNN. The number of neurons in the hidden layers of the MLP and BayesMLP models is set to [128, 64, 2], respectively. Both the optimizer and activation function are consistent with those of the BCNN model.

Table 2. Hyperparameters of the BCNN model.

Variable	Value
Batch size	128
Learning rate	0.01
Epochs	100
Optimizer	Adamax
Activation function	Softplus

4.3. BCNN Results and Comparison with Other Models

4.3.1. Performance Comparison of Models

To assess the effectiveness of the proposed BCNN model, the MLP, CNN, and BayesMLP models—incorporating Bayesian principles—were selected as comparison benchmarks. Since the parameters of the Bayesian neural network model follow a posterior distribution rather than taking on specific values, the model's output becomes a random variable for the same input data. Consequently, to evaluate the predictive perfor-

mance of the Bayesian neural networks, this study employs the Monte Carlo method, generating 100 independent predictions for each input data point. The mean of these predicted results is then compared to those of the non-Bayesian models. The performance metrics for this evaluation are summarized in Table 3.

Table 3. Performance comparison of non-Bayesian model and Bayesian model.

Model	Precision	Recall	F_1-Score	Accuracy	Training Time/s	Prediction Time/s
MLP	96.07%	94.52%	0.953	98.33%	37.553	0.248
CNN	96.40%	97.36%	0.969	98.90%	82.1643	0.279
BayesMLP	91.96%	91.21%	0.916	97.09%	89.666	0.421
BCNN	92.00%	94.02%	0.93	97.52%	131.95	0.768

The performance evaluation metrics of the Bayesian model after the Monte Carlo sampling are shown in Figure 3.

The evaluation results presented in Table 3 indicate a decline in the accuracy, precision, recall and F1 score metrics for the neural network models. Specifically, the BCNN model exhibits reductions of 1.37%, 4.40%, 3.34%, and 0.039 in accuracy, precision, recall, and F1 score, respectively, when compared to the CNN model. Additionally, both training and prediction times have increased. Several factors may contribute to the observed decrease in model prediction accuracy and the increase in computational time:

(1) The BCNN model not only outputs predicted results but also estimates the uncertainty associated with these outputs. This necessitates using the same number of weight parameters to achieve two distinct targets, potentially leading to a reduction in overall model accuracy;
(2) The BCNN model requires estimating the distribution of weight parameters, whereas non-Bayesian neural networks focus solely on point estimations. This distinction results in higher computational complexity and increased memory consumption for the BCNN, thereby extending both training and prediction times.
(3) The use of approximate inference methods to solve for the posterior distribution in the BCNN model may introduce substantial bias, affecting prediction accuracy.
(4) The selection of prior distribution has a certain impact on the performance of the model.

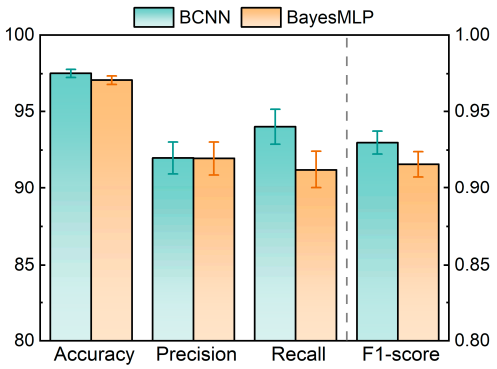


Figure 3. Predictive performance evaluation metrics of Bayesian models. The bar chart represents the average value of each prediction metric, and the error bar represents $\pm$ 1 standard deviation σ.

The bar chart in Figure 3 illustrates the predicted results and standard deviations for both the BCNN and BayesMLP models. The accuracy metric exhibits a small standard deviation, indicating that the predicted results are more tightly clustered around the mean. In contrast, the standard deviations for the other metrics are notably larger, reflecting signifi-

cant fluctuations in the model's predictions for imbalanced samples. This variability suggests that while the accuracy remains relatively stable, the model's performance on other metrics may be more susceptible to the inherent challenges posed by imbalanced data.

4.3.2. Robustness Analysis

To verify the robustness of the Bayesian model, this paper conducts robustness analyses using noise simulation and PMU missing data simulation, both of which are performed on the same training set to maintain a consistent experimental benchmark.

1. Noise simulation

The PMU signal collected in the actual power system operation process must have noise. To be close to the real situation when online applications are tested, a certain amount of Gaussian white noise is superimposed as the measurement value of PMU in the test set, and the signal-to-noise ratio (SNR) is used to measure the intensity of adding noise:

$$\mathrm{SNR}\left(\mathrm{dB}\right) = 10\lg\left(\frac{P_{signal}}{P_{noise}}\right) \tag{24}$$

where P_{signal} is signal power and P_{noise} is noise power. A higher SNR indicates a lower proportion of noise relative to the original signal. In this paper, noise with SNRs of 10 dB, 20 dB, 30 dB, 40 dB, and 50 dB is added to the test set to simulate varying noise levels. The performance of different models, including MLP, CNN, BayesMLP, and BCNN, under these noise conditions is evaluated and presented in Figure 4.

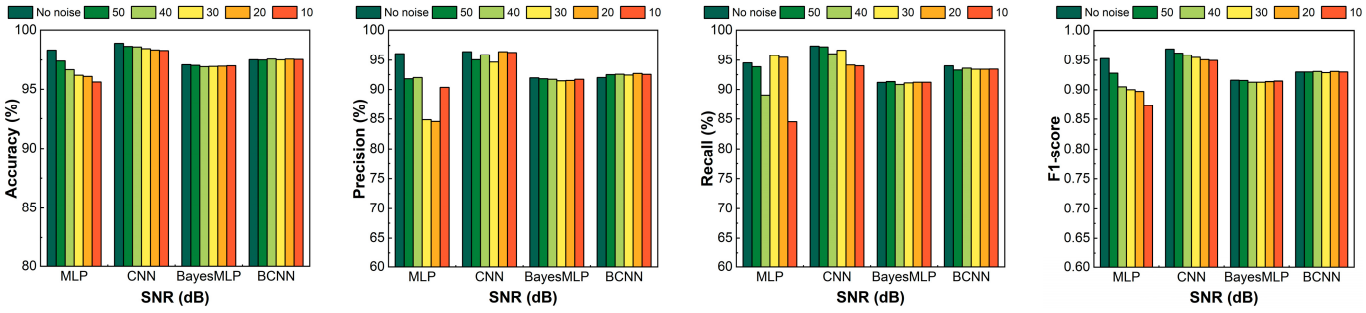


Figure 4. Comparison of performance metrics of Bayesian and non-Bayesian models with different noise levels. (**a**) Accuracy; (**b**) Precision; (**c**) Recall; (**d**) F_1-score.

By comparing the changes in evaluation metrics between Bayesian and non-Bayesian models under different noise levels, it is evident that the performance of non-Bayesian models significantly degrades as the noise level increases. In contrast, the Bayesian models demonstrate relatively small fluctuations in their evaluation metrics across varying noise levels. For example, when comparing the scenarios of no noise and SNR = 10 dB, the CNN model's accuracy, precision, recall, and F1 score decreased by 0.6%, 0.12%, 3.34%, and 0.02, respectively. Meanwhile, the BCNN model's respective metrics changed by −0.02%, −0.55%, 0.55%, and 0, showing minimal impact. Similarly, the BayesMLP model exhibited the same trend, indicating that Bayesian deep learning (BDL) models have strong robustness to noisy data.

2. Data missing simulation

In real power system operations, measurement data may become abnormal or be lost due to equipment faults and external interference. To simulate such scenarios, a certain proportion of feature values in each time interval of the test set is randomly selected and

set to zero to simulate missing measurements. Specifically, the missing data ratios are set to 5%, 10%, and 15%, respectively. The evaluation results of different models, including MLP, CNN, BayesMLP, and BCNN, under these varying data missing ratios are displayed in Figure 5.

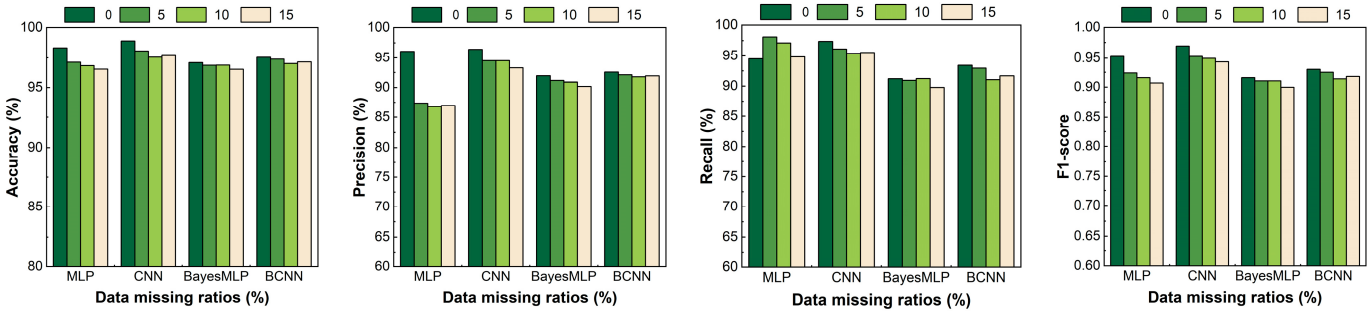


Figure 5. Comparison of performance metrics between Bayesian and non-Bayesian models with different proportions of missing data. (**a**) Accuracy; (**b**) Precision; (**c**) Recall; (**d**) F_1-score.

By comparing the metrics in Figure 5, we observe that as the data missing ratio increases, the evaluation metrics of non-Bayesian models significantly deteriorate, while the Bayesian models exhibit much smaller changes in their metrics. For instance, when comparing the performance of models with no data loss to those with 15% data loss, the CNN model's accuracy, precision, recall, and F1 score decreased by 1.14%, 3.11%, 1.80%, and 0.025, respectively. In contrast, the BCNN model's corresponding metrics only decreased by 0.40%, 0.62%, 1.73%, and 0.012, respectively. The BayesMLP model followed a similar trend, further demonstrating that Bayesian neural network models have strong robustness in the presence of missing data.

The robust performance of the BCNN models under varying conditions illustrates their strong resistance to uncertainty factors, such as data noise and random data loss. This resilience can be attributed to the inherent nature of Bayesian models, which characterize network parameters as random variables following a specific distribution. By incorporating this uncertainty into the modeling process, Bayesian approaches enhance the model's robustness against noisy and abnormal data, allowing it to maintain performance even in challenging scenarios.

4.3.3. Small Sample Analysis

To evaluate the ability of Bayesian neural network models to maintain good performance with limited data, the original dataset was retrained using 90%, 70%, and 50% of the available samples. The performance of the MLP, CNN, BayesMLP, and BCNN models under these varying sample sizes is illustrated in Figure 6.

As shown in Figure 6, the performance of non-Bayesian models significantly deteriorates as the number of training samples decreases. For instance, the CNN model's accuracy, precision, recall, and F1 score decreased by 2.90%, 13.75%, 4.54%, and 0.095, respectively. In contrast, the BCNN model exhibited much smaller declines in its metrics, with decreases of only 0.55%, 1.97%, 0.85%, and 0.015, respectively. These results indicate that Bayesian neural network models can effectively fit the training data and maintain high prediction accuracy even when trained with fewer samples.

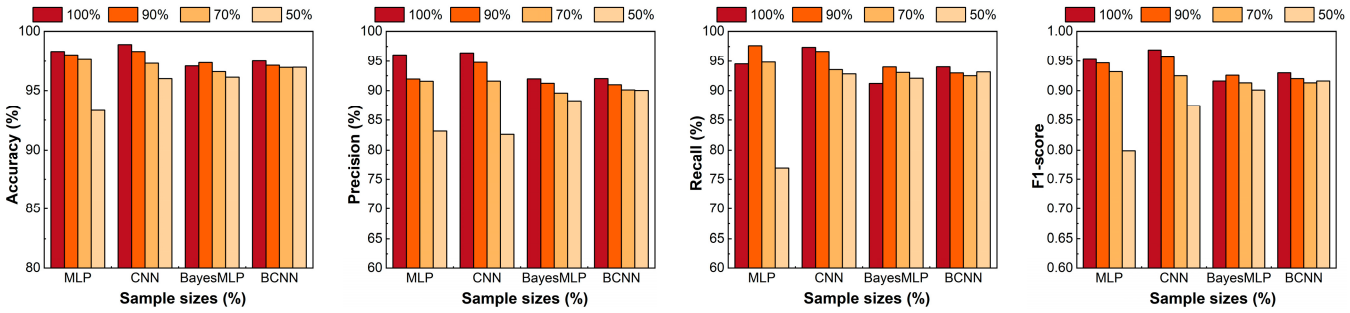


Figure 6. Comparison of performance metrics of Bayesian and non-Bayesian models with different sample sizes. (**a**) Accuracy; (**b**) Precision; (**c**) Recall; (**d**) F_1-score.

4.4. Uncertainty Estimation Discussion

The BCNN model is capable of providing uncertainty estimates for its prediction results by employing Monte Carlo sampling on the probability distribution of network weights. To quantify the uncertainties associated with the test set samples, both aleatoric and epistemic uncertainties are calculated according to Equation (16). The predictive uncertainty is then obtained by summing these two types of uncertainty, as follows:

$$\begin{aligned}&\text{Predictive Uncertainty} = \\ &\text{Aleatoric Uncertainty} + \text{Epistemic Uncertainty}\end{aligned} \tag{25}$$

To investigate the correlation between uncertainty prediction results, sample features, and prediction outcomes, this paper conducts a comparative analysis of the uncertainty performance across different sample types and prediction results. This analysis aims to verify the effectiveness of uncertainty estimates in guiding decision-making processes.

4.4.1. The Distribution of Uncertainty Estimation

This paper conducted a statistical analysis of the three types of classification results (false alarm, false dismissal, and correct) and their uncertainty in the test set. The analysis results are shown in Table 4 and Figure 7. The uncertainty score in the table is calculated based on the average uncertainty of each type of sample.

Table 4. The uncertainty comparison of different classification results.

Classification Results	Aleatoric Uncertainty	Epistemic Uncertainty	Predictive Uncertainty
False alarm	0.1626	0.0194	0.1820
False dismissal	0.1682	0.0276	0.1958
Correct	0.0123	0.0012	0.0134

Table 4 and Figure 7 provide a comparison of uncertainty among false alarm samples, false dismissal samples, and correct judgment samples. Figure 7 employs kernel smoothing to illustrate the distribution of uncertainty associated with different classification outcomes. Observing the data in the chart, several conclusions can be drawn: the aleatoric uncertainty, epistemic uncertainty, and predictive uncertainty for correct samples are significantly lower than those of false alarm samples. This finding suggests that uncertainty metrics can effectively reflect the confidence level of predictions, indicating that BCNN models perform better on samples with higher confidence levels.

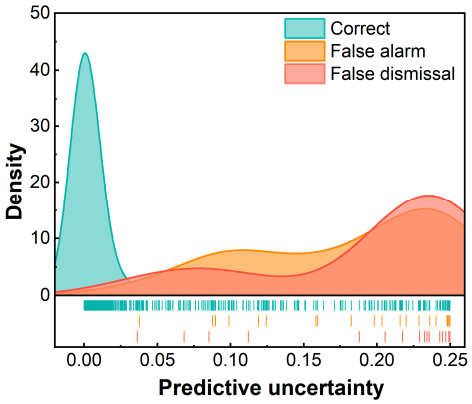


Figure 7. The uncertainty distribution of different classification results.

4.4.2. Exploring the Influence of Uncertainty Estimation

1. The impact of sample imbalance on uncertainty

The dataset used for transient stability assessment exhibits the characteristic of sample imbalance, with the number of unstable samples being significantly lower than that of stable samples. This paper analyzes the uncertainty prediction results for both unstable and stable samples, which are presented in Table 5 and illustrated in Figure 8.

Table 5. Comparison of uncertainty for different types of samples.

Sample Types		Aleatoric Uncertainty	Epistemic Uncertainty	Predictive Uncertainty
Actual label	Stable	0.0087	0.0008	0.0095
label	Unstable	0.0496	0.0055	0.0551
Predicted label	Stable	0.0081	0.0009	0.0089
label	Unstable	0.0515	0.0054	0.0568
Total Sample		0.0158	0.0017	0.0175

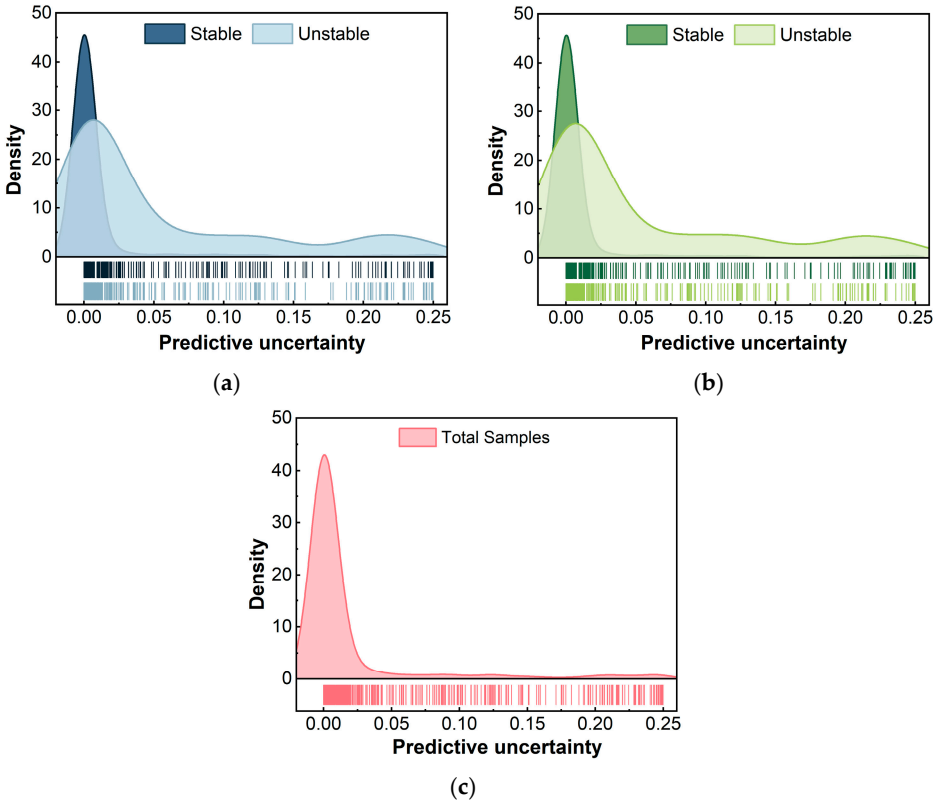


Figure 8. The uncertainty distribution of different types of samples. (**a**) Actual label; (**b**) Predicted label; (**c**) Total sample.

Table 5 presents the uncertainty scores for three different types of uncertainty across various sample categories. Analysis of the data in the table reveals a significant difference in uncertainty between unstable and stable samples. Specifically, the aleatoric uncertainty, epistemic uncertainty, and predictive uncertainty for unstable samples in the training set are substantially higher than those for stable samples. Notably, the uncertainty associated with unstable samples is approximately five times greater than that of stable samples.

Figure 8 illustrates the distribution characteristics of different sample types based on actual and predicted labels. The figure reveals clear differences in the uncertainty distribution between stable and unstable samples. In contrast, unstable samples show higher uncertainty, with a larger proportion of samples exhibiting high uncertainty and a tendency toward a multi-peak distribution. Notably, two smaller peaks emerge on the high-uncertainty side in both the (a) and (b) graphs of Figure 8, highlighting this trend. The overall distribution of uncertainty in the sample is similar to that of stable categories (i.e., most sample categories in the training set).

These phenomena illustrate that sample imbalance has a significant impact on the prediction results of uncertainty. When the Bayesian neural network model encounters a small number of samples, it tends to exhibit lower confidence levels.

2. The impact of different data noise intensities on uncertainty.

To investigate the impact of data noise on uncertainty, noise levels ranging from 50 dB to 10 dB were incrementally added to the test set with a step size of 10 dB. The uncertainty for each noise intensity was calculated using the Monte Carlo method, as shown in Table 6.

Table 6 reveals that as the noise intensity increases, uncertainties also increase. This indicates that the noise in the test set creates a divergence between the test set data and the training set data, affecting the generalization and prediction ability of the model for test set data.

Table 6. Comparison of uncertainty for different data noise.

Noise Intensity	Aleatoric Uncertainty	Epistemic Uncertainty	Predictive Uncertainty
No noise	0.0158	0.0017	0.0175
50 dB	0.0167	0.0020	0.0186
40 dB	0.0171	0.0022	0.0193
30 dB	0.0168	0.0020	0.0188
20 dB	0.0170	0.0022	0.0192
10 dB	0.0171	0.0022	0.0193

3. The impact of different data missing ratios on uncertainty.

To explore the impact of data missing ratios on uncertainty, 5%, 10%, and 15% of data were randomly set to zero in each time section of the test sample. The uncertainties under different data missing ratios were then compared, as shown in Table 7.

Table 7. Comparison of uncertainty for different data missing ratios.

Data Missing Ratios	Aleatoric Uncertainty	Epistemic Uncertainty	Predictive Uncertainty
0%	0.0158	0.0017	0.0175
5%	0.0181	0.0024	0.0204
10%	0.0201	0.0024	0.0225
15%	0.0205	0.0027	0.0232

The results show that as the data missing ratio increases, aleatoric uncertainty, epistemic uncertainty, and predictive uncertainty also rise. This suggests that missing data disrupts the consistency between the test set and the training set data distributions, resulting in higher uncertainty in the test set predictions.

4. The impact of sample size on uncertainty.

To examine the effect of sample size on uncertainty, the training set sample sizes were reduced to 90%, 70%, and 50% of the original dataset. The resulting changes in aleatoric uncertainty, epistemic uncertainty, and predictive uncertainty were observed, with the results summarized in Table 8.

Table 8. Comparison of uncertainty for different sample sizes.

Sample Size	Aleatoric Uncertainty	Epistemic Uncertainty	Predictive Uncertainty
100%	0.0158	0.0017	0.0175
90%	0.0173	0.0018	0.0191
70%	0.0176	0.0020	0.0196
50%	0.0221	0.0021	0.0242

The data in Table 8 illustrates that the number of training samples significantly affects the prediction results in terms of uncertainty, revealing a negative correlation between sample size and uncertainty. This finding aligns with Gal's research conclusion, which suggests that increasing the number of samples helps the dataset better approximate the true data distribution, thereby reducing uncertainty [33].

4.4.3. Using Uncertainty Estimation to Improve Prediction Results

Sections 4.4.1 and 4.4.2 establish the correlation between uncertainty and model prediction performance. This section focuses on leveraging uncertainty to enhance model performance. In transient stability evaluation, high uncertainty in the Bayesian neural network model's predictions signals low confidence, increasing the likelihood of false alarms or false dismissals. Given the critical nature of application scenarios, and to prevent severe consequences from these errors, this paper proposes using an uncertainty threshold as a decision-making criterion. When the uncertainty of a predicted result exceeds the predefined threshold, the result is deemed unreliable and requires further assessment by human experts.

To investigate the effects of different uncertainty thresholds, this paper employs Min–Max normalization to standardize the predictive uncertainty of all samples in the test set. Figure 9 illustrates the distribution of uncertainty for both correctly predicted and incorrectly predicted samples.

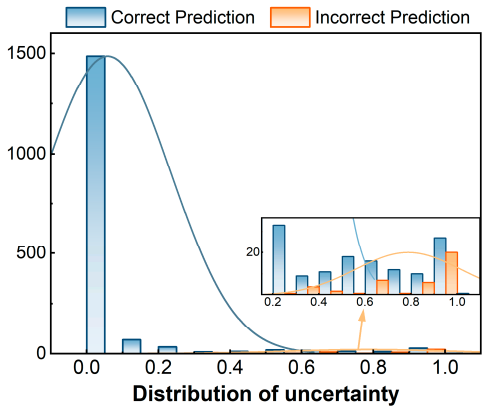


Figure 9. Distribution of uncertainty after Min-Max normalization for correctly predicted samples and incorrectly predicted samples.

After normalization, it can be seen that the uncertainty of correctly predicted samples is mainly distributed in the interval (0, 0.2), while the uncertainty of incorrectly predicted samples is mainly between 0.4 and 1. To improve the accuracy and practicality of transient stability evaluation, it is crucial to select a suitable uncertainty threshold to screen out most of the false alarm samples and improve prediction accuracy. Figure 10 describes the variations in various indicators when different uncertainty thresholds are applied.

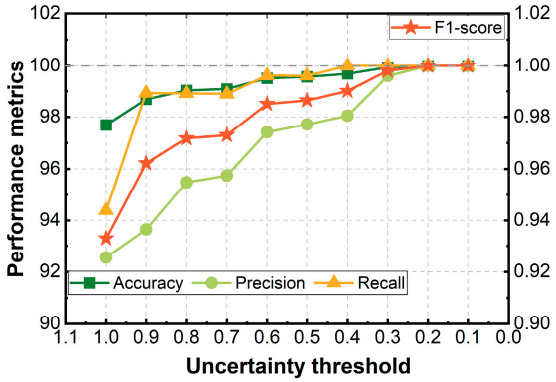
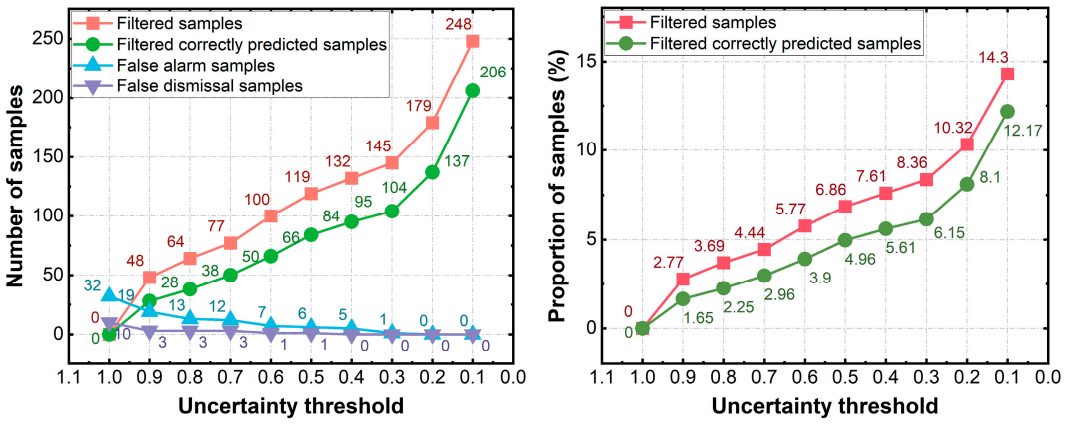


Figure 10. Performance metrics under different uncertainty thresholds.

As illustrated in Figure 10, a lower uncertainty threshold correlates with improved prediction performance of the model. When the threshold is set to 0.2, all samples are predicted correctly, and all evaluation metrics achieve a perfect score of 100%. However, selecting an excessively low uncertainty threshold may also exclude a significant number of correct samples, thereby increasing the burden on human experts. Figure 11 presents the number and proportion of samples filtered out at varying uncertainty thresholds.



(**a**) (**b**)

Figure 11. (**a**) The number of samples filtered out by different uncertainty thresholds; (**b**) The proportion of samples filtered out by different uncertainty thresholds.

Figure 11 reveals a slight distinction in the lowest uncertainty levels between false dismissal samples and false alarm samples, with false dismissal samples exhibiting marginally higher uncertainty. When the uncertainty threshold is set to 0.4, false dismissal samples are eliminated, leaving only 5 false alarm samples. Under these conditions, all unstable samples are accurately predicted, achieving a recall rate of 100%. The accuracy, precision, and F1-score at this threshold stand at 99.69%, 98.05%, and 0.991, respectively. While the indicators have improved, 7.61% of the total samples and 5.61% of the correct samples were filtered out. In contrast, setting the uncertainty threshold to 0.2 results in

perfect scores (100%) across all evaluation metrics; however, this choice leads to the exclusion of 10.32% of the total samples and 8.10% of the correct samples. The 10.32% of filtered samples necessitate further stability evaluation through time-domain simulations, thereby increasing computational costs. Therefore, the selection of an appropriate uncertainty threshold should consider the trade-off between prediction accuracy, automatic assessment coverage, and the workload and computational costs associated with further evaluation. The threshold of 0.2 is thus regarded as a representative choice based on the current test-set analysis rather than a universally optimal threshold.

To establish a reasonable uncertainty threshold, Monte Carlo sampling was conducted on the BCNN model to assess the variability in uncertainty outputs for the same input data. This analysis focused on the minimum uncertainty levels associated with false alarm samples and false dismissal samples. The results of this investigation are illustrated in Figure 12 and detailed in Table 9.

The data presented in Figure 12 and Table 9 show that the uncertainty of false alarm samples is generally slightly lower than that of false dismissal samples. The actual choice of threshold depends on the needs of the operator. If the primary goal is to ensure the accuracy of the model outputs, a suitable threshold could be established at the lower bound of the 95% confidence interval of the mean for false alarm samples, which is 0.10. Setting the threshold at this level would necessitate further simulation and expert evaluation for approximately 15% of the samples. Conversely, if the focus is on minimizing the number of samples requiring re-evaluation by experts while still ensuring the accuracy of unstable sample predictions, the threshold could be set at the lower bound of the 95% confidence interval for false dismissal samples, which is 0.48. In this scenario, around 8% of the samples would still require expert judgment for stability. Operators can therefore adjust the thresholds flexibly according to their operational requirements.

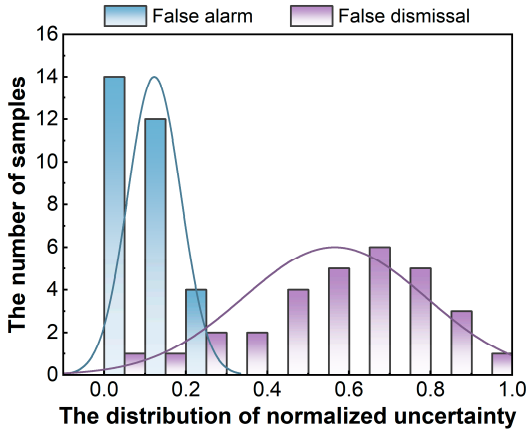


Figure 12. The minimum uncertainty distribution of false alarm samples and false dismissal samples.

Table 9. The minimum uncertainty of false alarm samples and false dismissal samples.

Sample Type	Mean	Standard Deviation	The Lower Bound of the 95% Confidence Interval	The Upper Bound of the 95% Confidence Interval
false alarm	0.12	0.06	0.10	0.15
false dismissal	0.57	0.22	0.48	0.65

5. Conclusions

This paper proposed a BCNN-based method for transient stability assessment of power systems, effectively addressing uncertainty in prediction results by applying Bayesian principles to CNN parameters. Results from the New England 39 bus case study demonstrated improved robustness of the proposed model when faced with noisy data,

missing data, and small sample datasets. Furthermore, the model's ability to estimate predictive uncertainty allowed for a statistical analysis that revealed significantly higher uncertainty in false alarm and false dismissal samples compared to correctly predicted samples. This validates the utility of an uncertainty threshold in effectively filtering out unreliable predictions. The research also identified various factors influencing uncertainty prediction results. It was found that unstable samples exhibit higher uncertainty than stable ones. Predictive uncertainty increases with noise intensity and missing data rate, whereas larger training datasets lead to lower predictive uncertainty. The analysis of uncertainty thresholds showed a clear trade-off: while lower thresholds enhance model performance, they also increase the number of samples requiring additional verification. Operators are therefore advised to adjust uncertainty thresholds based on specific operational needs.

Future developments of the proposed method could focus on several key areas: (1) Although the BCNN model quantifies uncertainty effectively, there is still potential for improving prediction accuracy. Future research could involve optimizing the Bayesian neural network's structure to enhance its expressiveness and generalization capabilities. Additionally, exploring the integration of other deep learning models with Bayesian theory might further enhance prediction performance. (2) In this study, the standard Gaussian distribution was exclusively used as the prior distribution for Bayesian modeling of network parameters, which may constrain the model's expressiveness and generalization ability. Future work could investigate the use of alternative prior distributions, such as the Laplace distribution, to assess their impact on model accuracy and prediction speed.

Author Contributions: Conceptualization, L.F. and Y.H.; methodology, L.F., M.Y. and X.W. (Xu Wang) and Y.H.; validation, L.F., M.Y. and Y.H.; formal analysis, L.F., Y.H. and X.W. (Xu Wang); investigation, L.F., M.Y. and J.Z.; resources, L.F., Y.H. and X.W. (Xiaopeng Wang); data curation, L.F., M.Y. and X.W. (Xu Wang); writing—original draft preparation, L.F.; writing—review and editing, L.F., M.Y., Y.H., J.Z., X.W. (Xu Wang) and X.W. (Xiaopeng Wang); visualization, L.F. and M.Y.; supervision, Y.H.; project administration, L.F., Y.H. and J.Z.; funding acquisition, L.F., Y.H. and J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Youth Guidance Project of Guizhou Provincial Basic Research (grant number QN[2025]079), Science and Technology Foundation of Guizhou Province (grant number[2024] General018), Science and Technology Foundation of Guizhou Province (grant number [2025] General021) and Science and Technology Foundation of Guizhou Province (grant number LH[2025] Major014).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: Authors Mengwei Yuan and Xu Wang were employed by the company Guizhou Power Grid Company. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TSA	Transient stability assessment
BCNN	Bayesian Convolutional Neural Network
TDS	Time-domain simulation

PMUs	Phasor measurement units
SVM	Support vector machines
DT	Decision trees
ELM	Extreme learning machines
CNN	Convolutional Neural Network
LSTM	Long short-term memory networks
DBN	Deep belief networks
GRU	Gated recurrent units
PNN	Probabilistic neural networks
BDL	Bayesian deep learning
BNN	Bayesian neural networks
MCMC	Markov Chain Monte Carlo
VI	Variational Inference
KL	Kullback–Leibler
ELBO	Evidence Lower Bound
PU	Predictive uncertainty
TSI	Transient stability index

References

1. Annakkage, U.D. Basics of Transient Stability Assessment. In *Use of Voltage Stability Assessment and Transient Stability Assessment Tools in Grid Operations*; Nuthalapati, S., Ed.; Springer International Publishing: Cham, Switzerland, 2021; pp. 79–97.
2. Prabhashankar, K.; Janischewsyj, W. Digital Simulation of Multimachine Power Systems for Stability Studies. *IEEE Trans. Power Appar. Syst.* **1968**, *PAS-87*, 73–80. https://doi.org/10.1109/TPAS.1968.292258.
3. Xue, Y.; Van Cutsem, T.; Ribbens-Pavella, M. A simple direct method for fast transient stability assessment of large power systems. *IEEE Trans. Power Syst.* **1988**, *3*, 400–412. https://doi.org/10.1109/59.192890.
4. Li, M.; Pal, A.; Phadke, A.G.; Thorp, J.S. Transient stability prediction based on apparent impedance trajectory recorded by PMUs. *Int. J. Electr. Power Energy Syst.* **2014**, *54*, 498–504. https://doi.org/10.1016/j.ijepes.2013.07.023.
5. Hu, W.; Lu, Z.; Wu, S.; Zhang, W.; Dong, Y.; Yu, R.; Liu, B. Real-time transient stability assessment in power system based on improved SVM. *J. Mod. Power Syst. Clean Energy* **2019**, *7*, 26–37. https://doi.org/10.1007/s40565-018-0453-x.
6. Lashgari, M.; Shahrtash, S.M. Fast online decision tree-based scheme for predicting transient and short-term voltage stability status and determining driving force of instability. *Int. J. Electr. Power Energy Syst.* **2022**, *137*, 107738. https://doi.org/10.1016/j.ijepes.2021.107738.
7. Li, Y.; Yang, Z. Application of EOS-ELM With Binary Jaya-Based Feature Selection to Real-Time Transient Stability Assessment Using PMU Data. *IEEE Access* **2017**, *5*, 23092–23101. https://doi.org/10.1109/ACCESS.2017.2765626.
8. Shi, Z.; Yao, W.; Zeng, L.; Wen, J.; Fang, J.; Ai, X.; Wen, J. Convolutional neural network-based power system transient stability assessment and instability mode prediction. *Appl. Energy* **2020**, *263*, 114586. https://doi.org/10.1016/j.apenergy.2020.114586.
9. Yan, R.; Wang, Z.; Yuan, Y.; Geng, G.; Jiang, Q. Information entropy based prioritization strategy for data-driven transient stability batch assessment. *CSEE J. Power Energy Syst.* **2021**, *7*, 443–455. https://doi.org/10.17775/CSEEJPES.2020.04500.
10. Xie, Z.; Zhang, D.; Han, X.; Hu, W. Research on Transient Stability Assessment Method of Power System Based on Improved Long Short Term Memory Network. *Power Syst. Technol.* **2024**, *48*, 998–1007. (In Chinese)
11. Wu, S.; Zheng, L.; Hu, W.; Yu, R.; Liu, B. Improved Deep Belief Network and ModelInterpretation Method for Power SystemTransient Stability Assessment. *J. Mod. Power Syst. Clean Energy* **2020**, *8*, 27–37. https://doi.org/10.35833/MPCE.2019.000058.
12. Zhan, X.; Song, H.; Na, R.; Yun, C. A hybrid transfer learning method for transient stability prediction considering sample imbalance. *Appl. Energy* **2023**, *333*, 120573.
13. Huang, J.; Guan, L.; Su, Y.; Yao, H.; Guo, M.; Zhong, Z. A topology adaptive high-speed transient stability assessment scheme based on multi-graph attention network with residual structure. *Int. J. Electr. Power Energy Syst.* **2021**, *130*, 106948. https://doi.org/10.1016/j.ijepes.2021.106948.
14. Yu, J.J.Q.; Hill, D.J.; Lam, A.Y.S.; Gu, J.; Li, V.O.K. Intelligent Time-Adaptive Transient Stability Assessment System. *IEEE Trans. Power Syst.* **2018**, *33*, 1049–1058. https://doi.org/10.1109/TPWRS.2017.2707501.
15. Gonzalez, J.; Papadopoulos, P.N.; Milanović, J.V.; Peskir, G.; Moriarty, J. Risk-Constrained Minimization of Combined Event Detection and Decision Time for Online Transient Stability Assessment. *IEEE Trans. Smart Grid* **2021**, *12*, 4564–4572. https://doi.org/10.1109/TSG.2021.3086236.

16. Nair, T.; Precup, D.; Arnold, D.L.; Arbel, T. Exploring uncertainty measures in deep networks for Multiple sclerosis lesion detection and segmentation. *Med. Image Anal.* **2020**, *59*, 101557. https://doi.org/10.1016/j.media.2019.101557.
17. Singh, R.K.; Gorantla, R.; Allada, S.G.R.; Narra, P. SkiNet: A deep learning framework for skin lesion diagnosis with uncertainty estimation and explainability. *PLoS ONE* **2022**, *17*, e0276836. https://doi.org/10.1371/journal.pone.0276836.
18. Bueno, A.; Benítez, C.; De Angelis, S.; Díaz Moreno, A.; Ibáñez, J.M. Volcano-Seismic Transfer Learning and Uncertainty Quantification With Bayesian Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 892–902. https://doi.org/10.1109/TGRS.2019.2941494.
19. Liu, G.; Jin, H.; Li, J.; Hu, X.; Li, J. A Bayesian deep learning method for freeway incident detection with uncertainty quantification. *Accid. Anal. Prev.* **2022**, *176*, 106796. https://doi.org/10.1016/j.aap.2022.106796.
20. Du, Y.; Liu, Y.; Wang, X.; Fang, J.; Sheng, G.; Jiang, X. Predicting Weather-Related Failure Risk in Distribution Systems Using Bayesian Neural Network. *IEEE Trans. Smart Grid* **2021**, *12*, 350–360. https://doi.org/10.1109/TSG.2020.3019263.
21. Liu, T.; Liu, Y.; Liu, J.; Wang, L.; Xu, L.; Qiu, G.; Gao, H. A Bayesian Learning Based Scheme for Online Dynamic Security Assessment and Preventive Control. *IEEE Trans. Power Syst.* **2020**, *35*, 4088–4099. https://doi.org/10.1109/TPWRS.2020.2983477.
22. Wang, K.; Chen, Z.; Wei, W.; Sun, X.; Mei, S.; Xu, Y.; Zhu, T.; Liu, J. Power System Transient Stability Assessment Based on Deep Bayesian Active Learning. In Proceedings of the 2022 IEEE/IAS Industrial and Commercial Power System Asia (I&CPS Asia), Shanghai, China, 8–11 July 2022; pp. 1692–1696. https://doi.org/10.1109/ICPSAsia55496.2022.9949775.
23. Shridhar, K.; Laumann, F.; Liwicki, M. A Comprehensive Guide to Bayesian Convolutional Neural Network with Variational Inference. *arXiv* **2019**, arXiv:1901.02731.
24. Bardenet, R.; Doucet, A.; Holmes, C. On Markov chain Monte Carlo methods for tall data. *J. Mach. Learn. Res.* **2017**, *18*, 1–43.
25. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight Uncertainty in Neural Networks. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1613–1622.
26. Kingma, D.P.; Salimans, T.; Welling, M. Variational Dropout and the Local Reparameterization Trick. In Proceedings of the Advances in Neural Information Processing Systems 28, Montreal, QC, Canada, 7–12 December 2015; Volume 28, pp. 2575–2583.
27. Kwon, Y.; Won, J.-H.; Kim, B.J.; Paik, M.C. Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation. *Comput. Stat. Data Anal.* **2020**, *142*, 106816. https://doi.org/10.1016/j.csda.2019.106816.
28. Shridhar, K.; Laumann, F.; Liwicki, M. Uncertainty Estimations by Softplus normalization in Bayesian Convolutional Neural Networks with Variational Inference. *arXiv* **2019**, arXiv:1908.00598.
29. Rajapakse, A.D.; Gomez, F.; Nanayakkara, K.; Crossley, P.A.; Terzija, V.V. Rotor Angle Instability Prediction Using Post-Disturbance Voltage Trajectories. *IEEE Trans. Power Syst.* **2010**, *25*, 947–956. https://doi.org/10.1109/TPWRS.2009.2036265.
30. Gupta, A.; Gurrala, G.; Sastry, P.S. An Online Power System Stability Monitoring System Using Convolutional Neural Networks. *IEEE Trans. Power Syst.* **2019**, *34*, 864–872. https://doi.org/10.1109/TPWRS.2018.2872505.
31. Athay, T.; Podmore, R.; Virmani, S. A Practical Method for the Direct Analysis of Transient Stability. *IEEE Trans. Power Appar. Syst.* **1979**, *PAS-98*, 573–584. https://doi.org/10.1109/TPAS.1979.319407.
32. Jing, L.; Juan, L.; Jianyu, H.; Hu, Y.; Xiaoshi, L. The Design of a Auxiliary System for the PSD-BPA Power Simulation Software. In Proceedings of the 2021 IEEE 4th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), Shenyang, China, 19–21 November 2021; pp. 638–641.
33. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning*; PMLR: New York, NY, USA, 2016; Volume 48, pp. 1050–1059.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.