Hindawi
International Journal of Aerospace Engineering
Volume 2019, Article ID 9375437, 8 pages
https://doi.org/10.1155/2019/9375437

Research Article

A Novel Classification Method for Flutter Signals Based on the CNN and STFT

Shiqiang Duan, Hua Zheng, and Junhao Liu

School of Power and Energy, Northwestern Polytechnical University, Shaanxi, Xi'an 710072, China

Correspondence should be addressed to Hua Zheng; isea_zh@mail.nwpu.edu.cn

Received 8 November 2018; Revised 26 January 2019; Accepted 18 February 2019; Published 9 April 2019

Academic Editor: Rosario Pecora

Necessary model calculation simplifications, uncertainty in actual wind tunnel test, and data acquisition system error altogether lead to error between a set of actual experimental results and a set of theoretical design results; wind tunnel test flutter data can be utilized to feedback this error. In this study, a signal processing method was established to use the structural response signals from an aeroelastic model to classify flutter signals via deep learning algorithm. This novel flutter signal processing and classification method works by combining a convolutional neural network (CNN) with time-frequency analysis. Flutter characteristics are revealed in both time and frequency domains, which are harmonic or divergent in the time series; the flutter model energy is singular and significantly increases in the frequency view, so the features of the time-frequency diagram can be extracted from the dataset-trained CNN model. As the foundation of the subsequent deep learning algorithm, the datasets are placed into a collection of time-frequency diagrams calculated by short-time Fourier transform (STFT) and labeled with two artificial states, flutter or no flutter, depending on the source of the signal measured from a wind tunnel test on the aeroelastic model. After preprocessing, a cross-validation schedule is implemented to update (and optimize) CNN parameters though the trained dataset. The trained models were compared against test datasets to validate their reliability and robustness. Our results indicate that the accuracy rate of test datasets reaches 90%. The trained models can effectively and automatically distinguish whether or not there is flutter in the measured signals.

1. Introduction

Flutter is a destructive aeroelastic instability caused by the coupling of aerodynamic, inertial, and elastic forces of the elastic structure of an aircraft [1]. Information characteristic regarding the structural system is acquired from dynamic structure response signals to determine the characteristics of flutter in the aircraft. It is important to classify the flutter signal accurately and effectively to properly process the signals.

Flutter signals are typically processed based on system stability [2], where flutter is considered an unstable phenomenon in the structural system as an appropriate dynamic data model is established for subcritical test signals. The dynamic data modeling places model parameters under certain criteria to define "stability" and calculate the stability parameters accordingly. Finally, the modal parameters are fitted through the curve and the critical flutter velocity is extrapolated. This method has notable shortcomings. For example, although time domain method criteria (e.g., Lyapunov and Jury) have correct theoretical bases, the prediction results obtained for the same group of states in an autoregression (AR) model may differ. Various factors (e.g., low signal-to-noise ratio of measured data and nonstationary structural response process) also adversely affect the reliability of predictions based on the system stability method. Zhang et al. [1] reviewed traditional flutter signal processing methods to find that they are ineffective for certain flutter signals. By contrast, these structural response signals usually contain abundant flutter information which can be mined via deep learning algorithm. In this study, we extracted features of the structural response signal from the aeroelastic model by deep learning algorithm and then applied them to classify flutter in the wind tunnel test data. This paper proposes a novel deep learning neural network-based method that is driven by large amounts of data to process flutter signals accurately and efficiently.

In a deep learning process, the time series are usually processed by a recurrent neural network (RNN). Khan et al. [3], for example, used the RNN to forecast vehicle flow in a traffic system. Hydrological and meteorological time series [4, 5] can also be applied to predict changes in water levels and weather by RNN. Wavelet transforms (WTs) can be combined with a neural network to forecast hydrological information [6]. Xu [7] even used the RNN to estimate spacecraft orbital state trends. The signals assessed in these studies have an important common feature: the sampling rate of the time series itself is relatively low, so the data points to be extrapolated do not need to be particularly large and the RNN is well applicable. The flutter signal is problematic in two ways. First, its sampling rate is generally 128 or higher—prediction and data extrapolation take several seconds (even tens of seconds) to treat thousands of points, and the RNN cannot effectively estimate such a lengthy time series. Second, it is necessary to extract the feature from the time series and then use the RNN to predict the time series features.

There is already rigorous theoretical support for the feature extraction of flutter signals based on AR model criteria (e.g., Lyapunov and Jury) under the system stability method, but features acquired via the system stability method itself have substantial errors. The use of the RNN alone is effective for certain specific step signals. The CNN, conversely, is mainly used to process images as the time series is transformed into the time-frequency diagram by short-time Fourier transform (STFT). The CNN is an innovative approach to image-based flutter signal processing and has been widely utilized in the computer vision field in recent years. It works by multilayer convolution and pooling for the input images to extract necessary features, then classifying the image after fully connected layers. Epilepsy, for example, can be diagnosed through a CNN and intracranial and scalp electroencephalogram time-frequency diagram [8]. Time-frequency diagrams computed from STFT, constant Q transform (CQT), and WT can be used to classify time series via CNN as well [9]. Li-Hua et al. [10] and Verstraete et al. [11] used the time-frequency diagram of mechanical vibration signals and a CNN to diagnose fault in vibration signals, where deep image features corresponding to the vibration signal were successfully extracted via CNN. Time-frequency diagrams calculated from STFT of various types of radar signals were used with a CNN to discriminate radar signals in another study [12, 13]. The time-frequency diagram of ultrasonic signals of broken glass in glass containers can be used to identify glass fragments by CNN [14]. Lee et al. [15] distinguished human emotions by time-frequency diagrams of pupil size change and movement data by the CNN. In summary, it is feasible to calculate time-frequency diagrams from STFT; time series STFT can be extracted effectively and further used for CNN-based classification and prediction.

Again, the time-frequency diagram can be used for feature extraction and signal classification after input to the CNN. In this study, we computed time-frequency diagram features from the STFT of a flutter test signal as extracted by a CNN trained on a large amount of flutter data. The final model, which was determined by the dataset-trained CNN, was then used to monitor multichannel-measured signals simultaneously. We identified three distinct flutter signal characteristics based on the literature and our own analysis [16]:

(1) At the critical point of flutter, the dynamic structural response signal is harmonic or divergent in the time series as per the flutter test mechanism and signal characteristics. The flutter mode in the frequency domain tends to be singular, and the energy of the flutter signal significantly increases at this point

(2) When the measured signal is converted into a two-dimensional (2D) image by computer graphics, its shape and gray level change

(3) Theoretically, this approach covers the time and frequency domains of flutter test signals

Based on the above three signal state changes and the time-frequency diagram acquired via STFT, image feature information can be acquired through the convolutional layer of the CNN. The system stability method no longer applies, and any system-stability-based criterion cannot be input to the previous RNN. Zhao [16] proved that there are differences in flutter versus no-flutter signals in the time-frequency diagram, and Zheng and Zhang [17] introduced a three-layer neural network for flutter prediction. However, at present, there is no available reference for CNN-based image feature extraction of flutter signals or flutter boundary prediction. This paper proposes a CNN-based joint time-frequency analysis for classifying flutter signals. First, the flutter signal time series is converted to an image by time-frequency analysis to ensure that the flutter signal contains both time and frequency domain information, which differs from the traditional model parameter estimation. The multilayer CNN can then extract image features to effectively process flutter signals. The deep learning method avoids damping estimation error and directly processes the flutter structural response signal.

The method in this paper is organized as follows. We first discuss our time series preprocessing design, where the time-frequency diagram can be established via joint time-frequency analysis; as per its computational complexity, the time-frequency diagram must be processed to bring it down to an input image size that the CNN can process (32×32). Image features are then extracted by the CNN convolutional layers. Finally, a fully connected layer capable of feature fusion facilitates flutter signal classification.

2. Introduction to the Proposed Method

The proposed method can be roughly divided into three parts. First, flutter test signal $x(n)$ is subjected to zero-average preprocessing to obtain signal $y(n)$ and then the preprocessed time-frequency diagram of flutter test signal $y(n)$ is obtained by the STFT. The two groups of datasets can be established by saving the time-frequency diagram of the signal in different angles of view—either at an elevation angle of 30° and azimuth angle of 135° or at an elevation angle of 90° and azimuth angle of 0°. Finally, the two datasets are used to

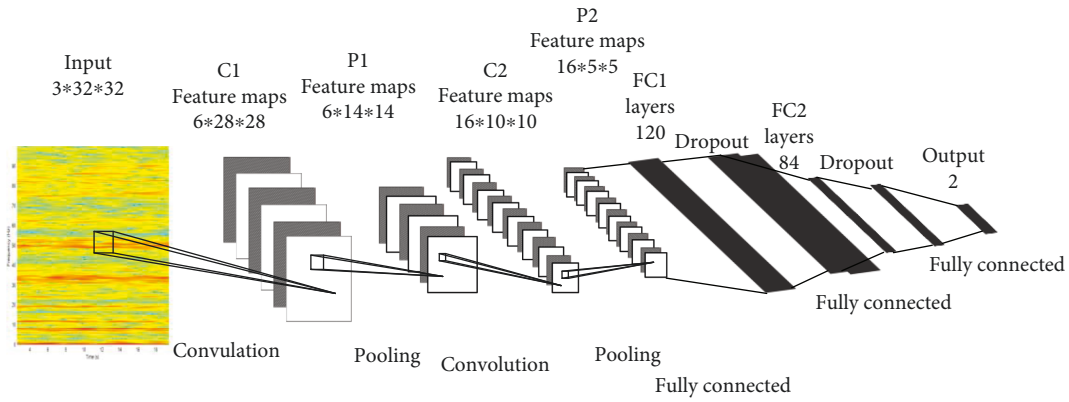

Figure 1: CNN architecture.

train two sets of CNN architectures to obtain different CNN models which classify flutter or no-flutter signals.

2.1. Flutter Signal Preprocessing. This step serves to normalize flutter test signal $x(n)$ to achieve signal preprocessing (equation (1)). The main purpose is to reduce the error in measured flutter signals which emerges during the artificial acquisition process.

$$y(n) = \frac{x(n) - \text{mean}x(n)}{\text{std}x(n)}, \tag{1}$$

where $\text{mean}(x(n))$ is the mean of $x(n)$ and $\text{std}(x(n))$ is the standard deviation of $x(n)$.

2.2. Short-Time Fourier Transform. The STFT is a mathematical transformation associated with a Fourier transform to determine the frequency and phase of a local region sine wave of a time-varying signal. It is defined as follows:

$$\text{STFT}_y(t,f) = \int_{-\infty}^{+\infty} [y(n)g^*(n-t)]e^{-j2\pi f n}dn, \tag{2}$$

where $y(n)$ is the output from equation (1) and $g(n)$ is the window function.

The time-frequency diagram of $y(n)$ is obtained by performing STFT on the signal output by equation (1) followed by edge clipping and downsampling to ensure a pixel size that the CNN can process.

2.3. Convolutional Neural Networks. The improved CNN based on the LeNet-5 CNN is used here to classify flutter signals. The Adam optimizer is utilized to update weights and bias in the entire network. Wu and Gu [18] proved that a dropout layer improves the neural network and can effectively prevent overfitting, so we added a dropout layer behind the two fully connected layers behind the network. Partial improvement of the basic LeNet-5 by the above method is effective for classifying flutter signals (Figure 1).

The network structure contains a 32×32 input 3-channel RGB image, convolution layer C1 with filter size of 5×5, and filter stride of 1. The C1 convolutional layer output has 6 channels. C1 has $6 \times 28 \times 28$ feature maps after the convolutional operation; $6 \times 14 \times 14$ feature maps are obtained through the 2×2 maximum pooling layer P1. The filters in convolution layer C2 are 5×5 in size and a have stride of 1. The inputs of C2 are 6 channels and the outputs of C2 are 16 channels. The previous feature maps transform the new feature maps (size $16 \times 10 \times 10$) through the convolution layer C2 then enter the maximum pooling layer P2 (2×2). The feature maps $(16 \times 5 \times 5)$ are obtained followed by three fully connected layers 400×120, 120×84, and 84×2 in size. Finally, two sets of results reveal whether the test signal contains flutter or not.

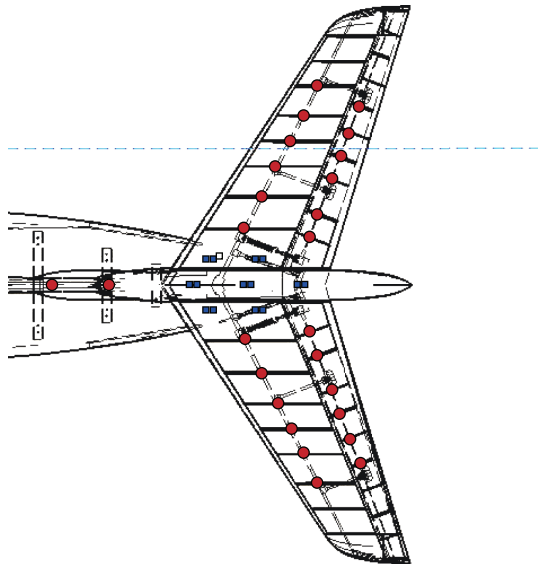

Figure 2: Sensor positions in the aeroelastic model.

3. Dataset Production

3.1. Data Source. The training, test, and validation datasets used in this study were derived from aeroelastic model wind tunnel flutter test data. The sensor layout of the aeroelastic model is as shown in Figure 2: seven strain measurements [19] were taken from locations marked by blue rectangles (each including bending and torsional data), and 26 acceleration sensors were placed in locations marked by red dots. Three strain measuring points were placed at the root of

the vertical tail and another four points at the left and right horizontal tails. Acceleration sensors were primarily placed on the rear fuselage, empennage, and elevator. Two acceleration sensors were arranged on the rear fuselage. The remaining acceleration sensors were divided into four groups arranged across the left horizontal tail, right horizontal tail, left elevator, and right elevator spanwise.

Original wind tunnel test data were recorded in full digital form on PXIe series data acquisition cards (American Virtual Instrument Company). Two sets of PXIe-4331 strain acquisition cards collected strain signals. Strain signals in one set of acquisition cards were sampled in eight channels, leaving six channels for the other set of cards; the strain signals have a total of 14 channels as analog inputs; acceleration signals were sampled by two sets of PXIe-4499 acceleration acquisition cards with 16 channels in the first set and 10 channels in the second set of cards making a total of 26 acceleration signals as analog inputs. Real-time acquisition of all experimental data was performed on an acquisition software driver in the LabVIEW environment. We set the basic sampling frequency of the wind tunnel flutter test to 200 Hz as per the relevant processing requirements and other parameter settings. The acceleration sensor is a 352C03 model (American PCB Company), and the strain sensor is a KFH prewire strain gauge (Omega).

There are 27 different statuses of aeroelastic models in the flutter wind tunnel test and a 180-steep wind (tunnel) speed. The excitation can be seen as atmospheric turbulence with various wind speeds. The deep learning training dataset was collected from a physical wind tunnel test on an aeroelastic model; the 22-second structural response signal with steady windiness at each wind speed step in the flutter test was selected as deep learning data. The time-frequency diagram obtained by the STFT of the aforementioned measured flutter signal was saved as a three-channel RGB image to compose the CNN dataset.

3.2. Stereoscopic View of Dataset. The known measured signals were classified into flutter and no-flutter categories (Figures 3(a) and 4(a)) as an artificial dataset. The time-frequency diagram of the signal was obtained by the method described in Section 2 (Figures 3(b) and 4(b)). In the frequency domain, the flutter mode tends to be singular and the energy markedly increases when flutter occurs (as discussed in Section 1). The dotted line in the time-frequency diagram (Figure 4) indicates the flutter phenomenon; the time-frequency diagram was saved at an elevation angle of 30° and azimuth angle of 135°. Label "1" indicates with flutter and "0" indicates with no flutter. There were 1396 flutter signals and 780 no-flutter signals in the training dataset, and 241 flutter signals and 190 no-flutter signals in the test dataset. The validation dataset generated 172 flutter signals and 100 no-flutter signals (Table 1).

The abscissa in Figures 3(a) and 4(a) represents time and the ordinate represents flutter signal amplitude; the x-axis represents time, the y-axis represents frequency, and the z-axis represents the amplitude of the image computed from the STFT of the flutter signal. Portions marked in red have higher energy than portions marked in blue.

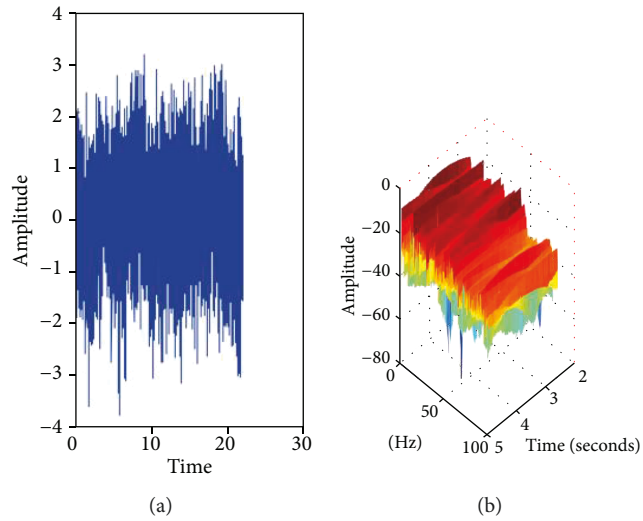

Figure 3: Signal time series without flutter (a) and STFT (b).

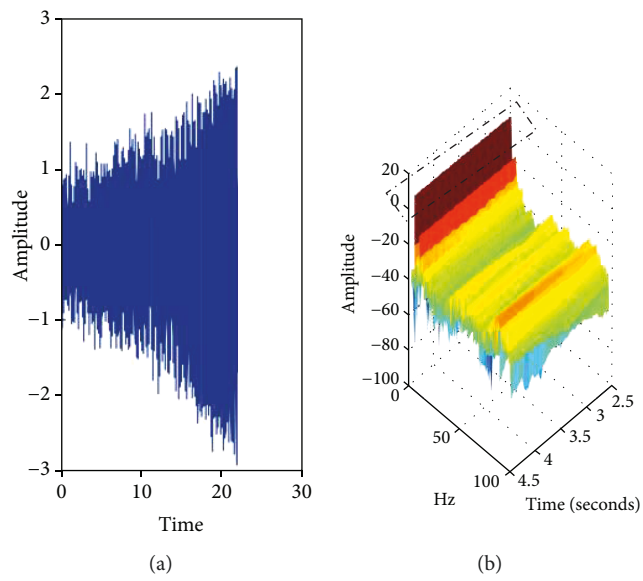

Figure 4: Signal time series (a) and STFT (b) when flutter occurs.

Table 1: 3D dataset statistics.

	Flutter	No flutter
Training dataset	1193	780
Test dataset	241	190
Validation dataset	172	100

3.3. Top View of Dataset. The measured signals were manually classified into no-flutter and flutter states and then subjected to the STFT to obtain image data as shown in Figures 5 and 6. The flutter state is marked by a dotted line in the time-frequency diagram. The time-frequency diagrams were saved at an elevation angle of 90° and an azimuth of 0°. Training datasets included 1573 flutter signals and 1527 no-flutter signals. Test datasets included 215 flutter signals and 200 no-flutter signals. The validation dataset included 139 images with flutter and 140 images without (Table 2).

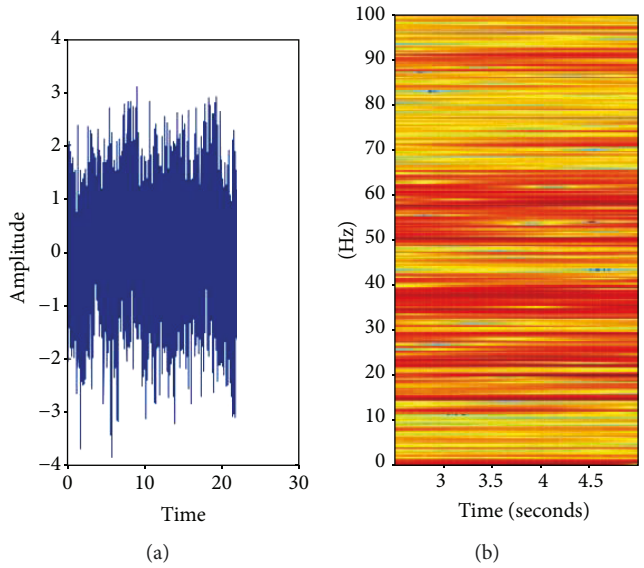

Figure 5: Signal time series (a) and STFT (b) when no flutter occurs.

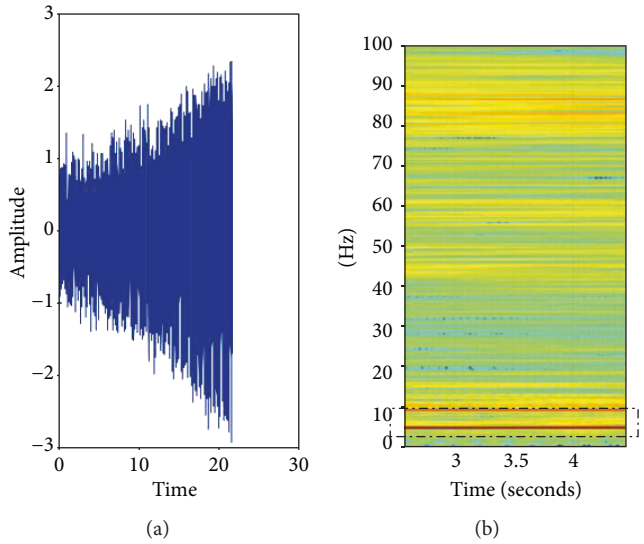

Figure 6: Signal time series (a) and STFT (b) when flutter occurs.

Table 2: Top view dataset statistics.

	Flutter	No flutter
Training dataset	1573	1527
Test dataset	215	200
Validation dataset	139	140

The abscissa in Figures 5 and 6 represents time and the ordinate represents the amplitude of the flutter signal (Figures 5(a) and 6(a)) or frequency (Figures 5(b) and 6(b)). Red and blue colors are again used to indicate energy.

4. Data Training and Test Results

4.1. Training Model Evaluation Criteria. The results obtained by training under the PyTorch deep learning framework indicate that the model can effectively separate flutter from no-flutter signals in the time-frequency diagrams. We assessed the training process under two criteria: training loss and accuracy rate.

4.1.1. Training Loss. The cross-entropy loss function is as follows:

$$\text{Loss}\left(y_{\text{pred}}, y_{\text{label}}\right) = -\frac{1}{n}\sum_{\text{class}}\left[y_{\text{pred}} \ln y_{\text{label}} + \left(1 - y_{\text{pred}}\right) \ln\left(1 - y_{\text{label}}\right)\right], \tag{3}$$

where class indicates the class to which the label needs to be divided; in our case, class = [0, 1]. y_{label} represents labeled results of the time-frequency diagram of flutter signals, y_{pred} represents the output of the network structure, and n is the dimension of training data.

4.1.2. Accuracy Rate. In the latter stage of the training process, the accuracy rates of training datasets are calculated after each epoch; flutter versus no-flutter signals are distinguished in the time-frequency diagram under the current weights and bias of the CNN. When the loss function is minimal, the weights and biases can be expended as the values of image feature extraction in the CNN. Here, the models constituted by the CNN architecture and its parameters were saved for reliability verification and the accuracy rate was calculated as follows:

$$\text{acc} = \frac{\text{sum}\left(y_{\text{pred}} == y_{\text{label}}\right)}{N}, \tag{4}$$

where acc indicates the accuracy rate, y_{pred} is the output of the flutter signal of the neural network, y_{label} is the result of the labeled flutter signal, and the sum function calculates the same number between y_{pred} of flutter results output by the neural network and y_{label} of the flutter signals; N is the dimension of y_{pred}.

4.2. Training Process and Test Results. The trained CNN model was expressed through the tough loss curve and accuracy rate curve based on test datasets and validation datasets. The loss curve is depicted here according to the network result and labeled result of the time-frequency diagram of the flutter test signal (equation (3)) on all three datasets (training, test, and validation)—the only difference is that the test and validation datasets do not participate in the training. The accuracy rates of the three datasets for each iteration (equation (4)) were calculated after each training weight, and bias was updated and finally plotted as a curve. Figures 7 and 8 show the training loss and accuracy rate, respectively of datasets in the stereoscopic time-frequency diagram (Figures 3(b) and 4(b)). Figures 9 and 10 show the training loss and accuracy rate of datasets for the top view time-frequency diagram (Figures 5(b) and 6(b)).

Figure 7 shows a loss curve of 2500 iterations with a learning rate of 0.000015 for the training process. The dropout layer

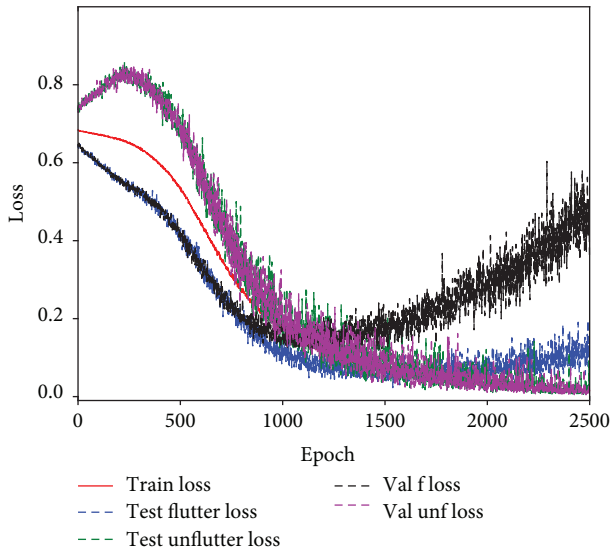

Figure 7: Training loss curve.

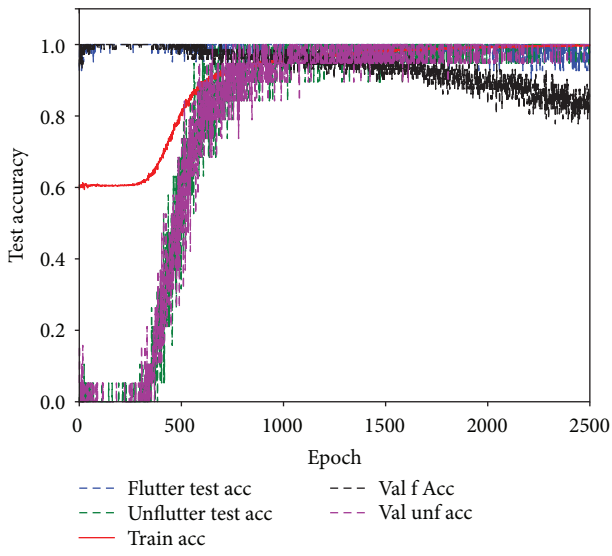

Figure 8: Accuracy rate curve.

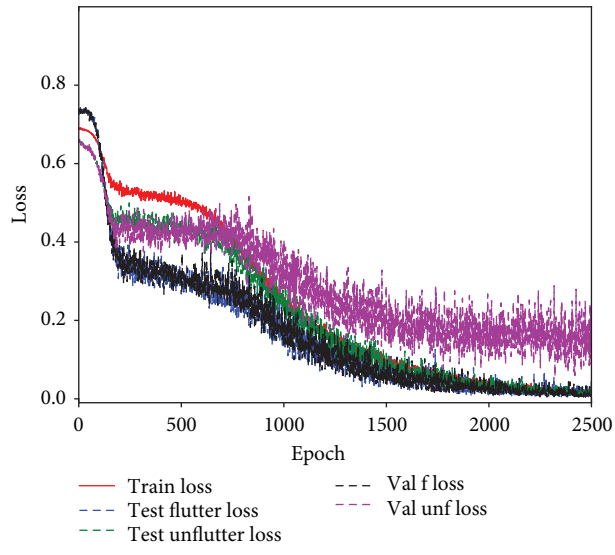

Figure 9: Training loss curve.

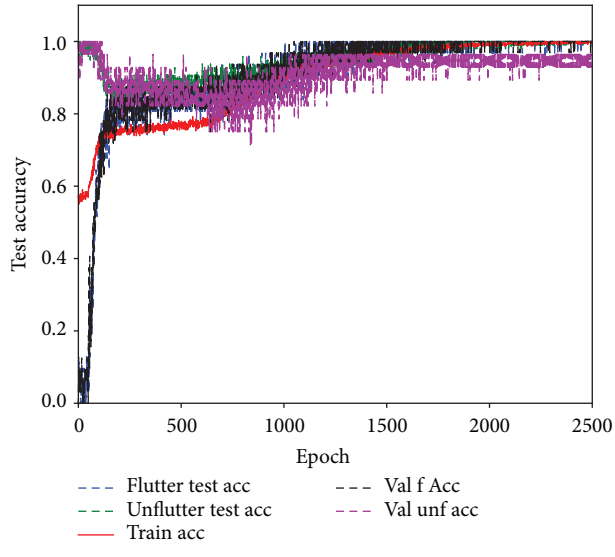

Figure 10: Accuracy rate curve.

was set to 0.5 after the fully connected layer was set to prevent overfitting. Figure 8 shows the accuracy rate curve for 2500 iterations with a learning rate of 0.000015. Figure 9 shows the training loss curve when the learning rate is 0.0004 and the number of iterations is 2500. The dropout layer was set to 0.5 after the fully connected layer to prevent overfitting. Figure 10 shows the accuracy rate curve when the learning rate is 0.00004 and the number of iterations is 2500. We found that the accuracy rate of test and validation datasets can reach almost 90% while that of the training dataset reaches 100%.

In Figures 7 and 9, the abscissa represents training iterations and the ordinate is training loss (equation (3). In Figures 8 and 10, the abscissa represents training iterations and the ordinate represents the accuracy rate (equation (4)).

The training loss, test loss, and accuracy rate were computed by the CNN architecture from the previous training iteration (Table 3). Dataset 1 is a time-frequency diagram with azimuth and elevation angles of 135° and 30°, respectively. The data in the table is the corresponding training dataset and test dataset loss and their accuracy rates with a learning rate of 0.000015 in the 2500th iteration. Dataset 2 is a top view time-frequency diagram; the table lists the training dataset and test dataset loss and their accuracy rates for the learning rate of 0.00004 in the 2500th iteration.

4.3. Comparison of the Symmetrical Structure. The CNN works by convolution and pooling operations to obtain image features which are then classified in a fully connected layer. The time-frequency diagram of the structural response signal from the flutter test can be processed by the CNN to determine whether or not flutter has occurred. The method proposed here was designed to exploit features from the time-frequency diagram to classify flutter phenomena, so it

Table 3: Model validation.

	Dataset 1	Dataset 2
Training set loss	0.018562	0.016996
Flutter test set loss	0.008938	0.003387
No-flutter test set loss	0.005237	0.091222
Training set accuracy rate	100%	100%
Flutter test set accuracy rate	97.51%	98.60%
No-flutter test set accuracy rate	90.52%	95.5%

Dataset 1 is a 3D stereoscopic time-frequency diagram dataset. Dataset 2 is a top view time-frequency diagram dataset.

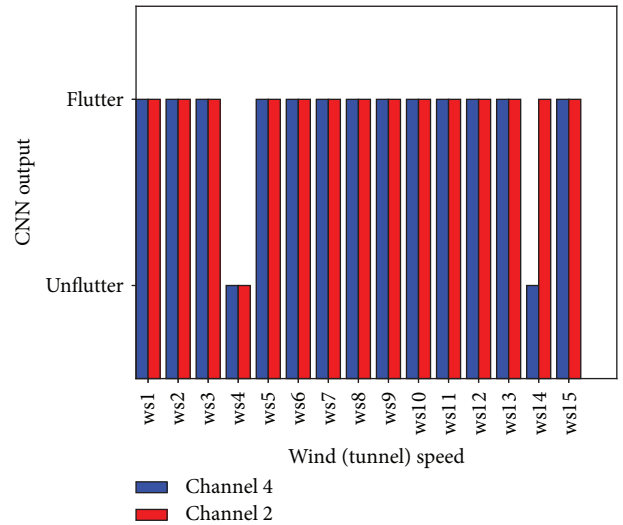

Figure 11: Symmetrical structures in flutter test dataset.

is unreasonable to make a direct comparison between the traditional method and deep learning-based method.

Existing flutter signal processing techniques are based on criterial estimations such as frequency and damping. The flutter boundary is predicted by the downward criterial trend, but technically, due to the simplification of the model calculation, uncertainty in the actual wind tunnel test, and acquisition system error, there is an error between the actual experimental results and theoretical design results. This is also why a flutter wind tunnel test should be performed in the engineering design. To further verify the reliability of our method, we selected structural response signals of symmetrical positions from the aeroelastic model in the test dataset and then used the top view time-frequency diagrams of those signals (Section 3) to validate the trained model under different wind speeds. Fifteen sets of flutter signals from the horizontal tails were compared as shown below. In Figures 10 and 11, the abscissa represents wind (tunnel) speed and the ordinate indicates the output of the trained CNN. Channel 4 is a flutter signal sampled from the strain sensor in the right horizontal tail root, and Channel 2 is a flutter signal sampled from the strain sensor in the left horizontal tail root.

Structural response signal CNN outputs from the right horizontal tail test point under flutter conditions are marked in blue and those from the left point are marked in red in

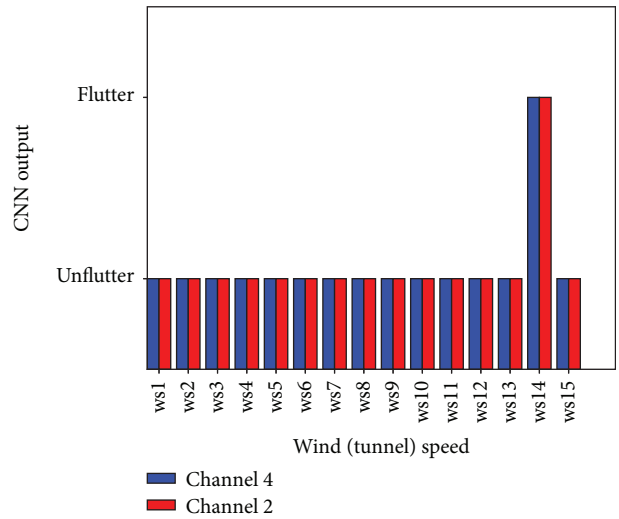

Figure 12: Symmetrical structures in no-flutter test dataset.

Figure 11; similar demarcations are made for the right and left test points under no-flutter conditions in Figure 12. The results from both sides of the symmetrical model are in accordance with the set results across all the wind speeds that we tested. These results suggest that the proposed deep learning-based technique allows for effective feature extraction of structural response signals.

5. Conclusion

In this study, we applied a time-frequency diagram of the flutter signal and a CNN to classify flutter versus no-flutter test signals. The test dataset and validation dataset show accuracy of up to 90% based on the trained CNN model. We tested a stereoscopic view of the dataset to validate the proposed method under flutter conditions; we did find, however, that the loss function value in the dataset increases if the number of iterations continues to increase beyond the hyperparameters that we set. That being said, the top view dataset is more robust than the stereoscopic view. We compared different angles of view and adjusted the hyperparameters to secure optimal classification effects on the test and validation datasets.

Our results altogether demonstrate the feasibility of the proposed CNN for the classification of flutter signals. The method described in this paper also represents a novel data processing algorithm for actual wind tunnel test data. Further research is yet warranted to analyze flutter speeds through the deep learning algorithm.

Data Availability

The partial test data of the flutter signal used to support the findings of this study are included within the supplementary information files (available here).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant no. 11302175), the Aeronautical Science Foundation of China (20151353017), and the Chinese Universities Scientific Fund (3102017HQZZ013)

Supplementary Materials

The supplementary material file is a partial part of the experimental data of a flutter test data used to produce the dataset in Section 3 to verify our proposed method. More details can be found from the data_description.txt in the supplementary material zip file. *(Supplementary Materials)*

References

[1] W. W. Zhang, H. S. Huashou, and H. Xiao, "Review and prospect of flutter boundary prediction methods for flight flutter testing," *Acta Aerodynamica Sinica*, vol. 36, no. 5, pp. 1367–1384, 2015.

[2] H. Zheng, J. Liu, and S. Duan, "Flutter test data processing based on improved Hilbert-Huang transform," *Mathematical Problems in Engineering*, vol. 2018, Article ID 3496870, 8 pages, 2018.

[3] M. D. Z. Khan, S. M. Khan, M. Chowdhury, and K. Dey, "Development and evaluation of recurrent neural network based models for hourly traffic volume and AADT prediction," 2018, https://arxiv.org/abs/1808.10511.

[4] Y. Yi-Yue, F. U. Qian, and W. Ding-Sheng, "A Prediction Model for Time Series Based on Deep Recurrent Neural Network," *Computer Technology and Development*, 2017.

[5] Y. Han, *Research on Weather Forecasting Based on Deep Learning*, Harbin Institute of Technology, 2017.

[6] Y. L. Zhu, S. J. Li, Q. S. Fan, and D. S. Wan, "Wavelet-neural network model based complex hydrological time series prediction," *Shandong Daxue Xuebao(GongxueBan)*, vol. 41, no. 4, pp. 119–124, 2011.

[7] Y. Xu, *Research on Prediction Algorithm of Machine Learning Methods for Spacecraft In Orbit Mutation Status*, University of Electronic Science and Technology of China, 2017.

[8] N. D. Truong, A. D. Nguyen, L. Kuhlmann et al., "Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram," *Neural Networks*, vol. 105, pp. 104–111, 2018.

[9] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," 2017, https://arxiv.org/abs/1706.07156.

[10] L.-H. Wang, X.-P. Zhao, J.-X. Wu, Y.-Y. Xie, and Y.-H. Zhang, "Motor fault diagnosis based on short-time Fourier transform and convolutional neural network," *Chinese Journal of Mechanical Engineering*, vol. 30, no. 6, pp. 1357–1368, 2017.

[11] D. Verstraete, A. Ferrada, E. L. Droguett, V. Meruane, and M. Modarres, "Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings," *Shock and Vibration*, vol. 2017, Article ID 5067651, 17 pages, 2017.

[12] X. Wang, G. Huang, Z. Zhou, and J. Gao, "Radar emitter recognition based on the short time fourier transform and convolutional neural networks," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5, Shanghai, China, October 2017.

[13] M. Zhang, M. Diao, L. Gao, and L. Liu, "Neural networks for radar waveform recognition," *Symmetry*, vol. 9, no. 5, p. 75, 2017.

[14] B. Zhao, P. Yang, O. A. Basir, and G. S. Mittal, "Ultrasound based glass fragments detection in glass containers filled with beverages using neural networks and short time Fourier transform," *Food Research International*, vol. 39, no. 6, pp. 686–695, 2006.

[15] H.-J. Lee and S.-G. Lee, "Arousal-valence recognition using CNN with STFT feature-combined image," *Electronics Letters*, vol. 54, no. 3, pp. 134–136, 2018.

[16] Z. Shanhong, *Method of Time Frequency Syntony for Flutter Boundary Prediction*, Northwestern Polytechnical University, 2002.

[17] H. Zheng and J. Zhang, "A stability criterion method based on neural network and its application on flutter boundary prediction," *Engineering Intelligent Systems*, vol. 24, no. 3-4, pp. 99–104, 2016.

[18] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *Neural Information Processing*, S. Arik, T. Huang, W. Lai, and Q. Liu, Eds., vol. 9489 of Lecture Notes in Computer Science, pp. 46–54, Springer, Cham, 2015.

[19] D. E. Raveh, M. Iovnovich, and T. Nahom, "Wind-tunnel study of the ARMA flutter prediction method," in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Kissimmee, Florida, January 2018.