Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

WMGIN: Windowed Multi-Granularity Interpretable Network for Hard Landing Warning

Zemin Lin
Zhuhai Xiangyi Aviation Technology Co., Ltd
Yonglian Li
Zhuhai Xiangyi Aviation Technology Co., Ltd
Wei Li
Zhuhai Xiangyi Aviation Technology Co., Ltd
Guolong Li
Zhuhai Xiangyi Aviation Technology Co., Ltd
Zonghua Ye
Zhuhai Xiangyi Aviation Technology Co., Ltd
Tian Zeng
Zhuhai Xiangyi Aviation Technology Co., Ltd
Yan Liu
yanliu@csair.com
Zhuhai Xiangyi Aviation Technology Co., Ltd
Zhifeng Wang
Zhuhai Xiangyi Aviation Technology Co., Ltd

Research Article

Keywords: QAR data, Hard landing, Interpretable AI, Deep learning, Flight safety

Posted Date: October 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10891137/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

WMGIN: Windowed Multi-Granularity Interpretable Network for Hard Landing Warning

Zemin Lin[1], Yonglian Li[2], Wei Li[3], Guolong Li[4], Zonghua Ye[1], Tian Zeng[1], Yan Liu[1*], Zhifeng Wang[2]

[1*]Information Technology Department, Zhuhai Xiangyi Aviation Technology Co., Ltd, Zhuhai, 519030, Guangdong, China.
[2]Flight Training Department, Zhuhai Xiangyi Aviation Technology Co., Ltd, Zhuhai, 519030, Guangdong, China.
[3]Management Team, Zhuhai Xiangyi Aviation Technology Co., Ltd, Zhuhai, 519030, Guangdong, China.
[4]Safety and Quality Department, Zhuhai Xiangyi Aviation Technology Co., Ltd, Zhuhai, 519030, Guangdong, China.

*Corresponding author(s). E-mail(s): yanliu@csair.com;
Contributing authors: linzm@csair.com; liyonglian@csair.com; davidlee@csair.com; liguol@csair.com; yezh@csair.com; zengtian@csair.com; wangzhif@csair.com;

Abstract

Landing is a critical phase of flight with the highest incidence of accidents. Existing studies on flight safety either rely on traditional machine learning methods with limited predictive accuracy or adopt deep learning approaches that improve accuracy but overlook interpretability. In this work, we focus on predicting hard landing events to address these issues. To this end, we propose WMGIN, a self-attention-based multi-granularity interpretable deep learning model for hard landing prediction. WMGIN processes Quick Access Recorder (QAR) data through a three-step strategy. First, WMGIN directly learns embedded representations from multi-scale raw data, avoiding information loss. Second, it employs window- and layer-wise fusion strategies: temporal information is exchanged within fixed windows to preserve intuitive interpretability, while layer-wise fusion progressively expands available information. Finally, learnable class tokens are introduced to capture the influence of different time periods across all variables, providing multi-granularity interpretability. Experiments on a QAR dataset of 44,804 A321 flights demonstrate the effectiveness of the proposed method.

Keywords: QAR data, Hard landing, Interpretable AI, Deep learning, Flight safety

1 Introduction

Flight safety remains a central concern in civil aviation. According to the 2023 Global Commercial Aviation Safety Report published by the International Air Transport Association (IATA) [1], the aviation industry continued to improve its safety performance in 2023, achieving results that surpassed every year since 2014. Nevertheless, aviation safety cannot be underestimated despite these advancements and will remain a critical issue for the industry. Aviation safety statistics indicate that the majority of fatal flight accidents occur during the landing phase, which accounts for only 1% of the total flight duration but represents as much as 31% of accidents between 2013 and 2022 [2]. This clearly indicates the crucial role of landing in ensuring flight safety. Among these incidents, a hard landing is a typical safety hazard, defined as an event in which the aircraft's vertical acceleration exceeds a specified threshold during landing. Such occurrences may not only cause structural damage to the aircraft but also pose serious risks to passenger safety. Modern aircraft are generally equipped with a Quick Access Recorder (QAR). This flight data recording device continuously records aircraft state parameters, pilot operation parameters, and environmental parameters at varying frequencies throughout the flight. Leveraging QAR data to analyze and explain the causes of hard landings is therefore of significant importance.

In recent years, QAR data have attracted increasing research attention, and numerous studies have been conducted on flight safety issues such as runway excursion [3, 4], long landing [5–7], tail strike [8–10]. About hard landing events, existing research can be broadly classified into two categories: machine learning–based approaches [11–13] and deep learning–based approaches [14–16]. However, both lines of research exhibit notable limitations. First, QAR data are inherently multi-scale and multivariate time series. Many deep learning–based studies either reduce the data by averaging high-frequency parameters on a per-second basis or fail to effectively capture complex feature interdependencies due to model constraints. These limitations often result in reduced prediction and classification accuracy. Second, most existing studies lack interpretability in their model outputs. Early work on hard landings employed correlation analysis and variance analysis [11], but such methods only revealed superficial factors, without uncovering deeper causal mechanisms. More recent studies based on machine learning and deep learning have largely emphasized predictive performance, often neglecting or underestimating the importance of interpretability [17, 18]. In summary, current research either overlooks parameter interdependencies—thereby compromising predictive accuracy—or prioritizes accuracy at the expense of model interpretability.

To address the above limitations, this study proposes **WMGIN** (**W**indowed **M**ulti-**G**ranularity **I**nterpretable **N**etwork), a novel deep learning model that achieves both high predictive performance and strong interpretability. The architecture of WMGIN consists of three main components: an embedding layer, a time window encoder (TW Encoder), and an interpretable hybrid encoder (IH Encoder) with

a global inter-dimensional mask. Specifically, the embedding layer standardizes the length of each parameter in the time series while minimizing information loss, thereby unifying the data in the feature dimension. The TW Encoder, composed of multiple TW blocks, captures temporal dependencies of time tokens at multiple granularities for each variable. Within each TW block, adjacent time tokens are merged through a fusion layer, enabling coarser-grained representations via a window-based Transformer. The IH Encoder, which contains the same number of IH blocks as TW blocks, operates on the outputs of the TW Encoder. Each IH block incorporates a learnable class token [19] and employs a global inter-dimensional mask to explicitly model the dynamic influences among different variables across time. This mechanism not only enhances the representation of interdependencies but also provides interpretability by revealing how different variables contribute to the prediction. Importantly, to ensure interpretability, the final prediction relies solely on the learnable class tokens from each layer. We evaluate WMGIN on a real-world QAR dataset comprising 44,804 A321 flight samples. The experimental results demonstrate that WMGIN achieves superior predictive performance while offering high interpretability in analyzing hard landing events.

In conclusion, the main contributions of this paper are as follows:

- **A New Deep Learning Architecture (WMGIN)**: A newly developed model named WMGIN is proposed for hard landing event recognition with interpretable explanations. The architecture is constructed by integrating IH blocks and TW blocks with global inter-dimensional masks through a multi-layer fusion design, enabling the capture of both local temporal patterns in individual variables and global dynamic correlations among variables at multiple granularities.
- **Multi-Granularity Interpretability**: A learnable class token is incorporated into each IH block, aggregating the weights of all tokens in the corresponding layer after data processing. Through the integration of class tokens across layers during prediction, the contributions of time tokens at different granularities for each feature are revealed.
- **Experimental Validation**: Comprehensive experiments are conducted on a real-world QAR dataset. The results demonstrate that WMGIN outperforms state-of-the-art baselines in hard landing prediction. Additionally, interpretability analysis performed on hard landing flights shows that the underlying factors of hard landings can be identified by the model.

The remainder of this paper is organized as follows. Section 2 reviews related work on hard landing prediction and interpretability. Section 3 presents the proposed WMGIN model. Section 4 introduces the QAR dataset and experiments. Section 5 concludes the paper.

2 Related Work

2.1 Hard landing prediction

2.1.1 Machine learning-based methods

With the widespread adoption of flight data recording systems, substantial research has been devoted to the study of hard landings. Wang et al. [11, 20] employed analysis of variance (ANOVA) to identify the parameters most significantly associated with hard landings and subsequently constructed a logistic regression model for hard landing prediction and risk analysis. Hu et al. [21] investigated the correlation between flight parameters and developed a support vector machine (SVM) model to predict hard landings. You et al. [12] proposed a parameter filtering method based on clustering algorithms, and further combined an autoregressive moving average model with an SVM, where the former was used for parameter prediction and the latter for determining whether a hard landing occurred. Qiao et al. [22] performed slicing and dimensionality reduction on QAR data, utilized a radial basis function neural network for prediction, and determined the structural parameters of the model through K-means clustering. In addition, other machine learning algorithms have been applied to hard landing prediction, including clustering [23], ridge regression [24], and random forests (RF) [13]. However, due to the simplicity of these models, their representational capacity and predictive performance remain unsatisfactory. Moreover, most of these methods have neglected model interpretability. Recently, Li et al. [25] proposed a K-means-based curve clustering model named CurveCluster+, which can automatically recognize and explain patterns of hard landing events.

2.1.2 Deep learning-based methods

To address the limitations of traditional machine learning methods, recent studies have shifted their focus to deep learning methods. Specifically, several studies [14–16, 26] have incorporated long short-term memory (LSTM) networks to enhance predictive performance. For instance, Kong et al. [26] employed Bayesian neural networks in conjunction with LSTM to predict the vertical landing speed of aircraft, quantifying the uncertainty in the model outputs to support risk-informed decision-making. Tong et al. [14] proposed a deep prediction model based on LSTM to predict hard landing incidents using the sensor data collected by a quick access recorder. Although LSTM-based approaches generally outperform traditional machine learning models in terms of predictive accuracy, they still fail to provide interpretability of the prediction results, which remains a critical limitation. To address the limitation of interpretability in deep learning models, Li et al. [27] defined hard landing prediction as a multi-scale time series classification problem and proposed an interpretable model named IMTCN, which combines multiple temporal convolutional networks (TCNs) with the class activation mapping (CAM) technique to provide both accurate predictions and high interpretability of flight safety. Chen et al. [28] further considered the correlations among different flight parameters, and proposed an attention-driven model named SDTAN (Scalable Deep Time-Aware Attention Network) for hard landing prediction and interpretation. Cai et al. [29] considered the unique characteristics of QAR time

series data, which exhibits limited periodicity and trends, and proposed a fine-grained time and hidden feature learning model for interpretable hard landing prediction. There are also some studies trying to address multiple flight safety events (including hard landing) with the multi-task learning framework [30].

2.2 Interpretability of models

At present, there is no unified definition of interpretability, which can generally refer to the interpretation or presentation of a model in a way that can be understood by humans. The interpretability of models can be generally divided into two categories: post-interpretation models and built-in interpretation models [31].

Post-interpretation model refers to a trained model, using a certain explanation method outside the model to explain the working mechanism of the model and the reason for the decision. Post-interpretation models can be divided into gradient-based, perturbation-based, and approximation-based models. Class Activation Mapping (CAM) [32] prediction results depend on concatenating and training the feature maps obtained after global average pooling across all channels to derive the weights for different regions of data, which provides gradient-based interpretability. Kashiparekh et al. [33] recognized key sequence segments that influence the decision-making of the classifier by obscuring part of the data, providing perturbation-based interpretability. Local Interpretable Model-agnostic Explanations (LIME) [34] is the representative of approximation-based interpretive models that train a simple interpretable linear Model to interpret the model's prediction results locally.

Built-in interpretation model is inherently interpretable, generally implemented by model with simple structures and easy to understand, such as linear model, decision tree, Naive Bayes, etc. Recently, one of the popular Built-in interpretation models is to integrate the attention mechanism [35] into the target model. The attention mechanism assigns learnable weights to the input data, which can reflect areas of interest to the model in the decision-making process. Hsieh et al. [36] applied an attention mechanism to time series data from the time dimension and the variable dimension, respectively, and interpreted according to the attention value. Lim et al. proposed TFT [37], which uses a variety of attention mechanisms in a time series prediction model and visually interprets it based on attention values. In this paper, we creatively use learnable class tokens to enable the output attention weights of the model to represent the influence on the predicted result rather than the correlation among the tokens.

3 Methodology

In this paper, we mainly aim to address the hard landing problem, i.e., predicting whether a hard landing incident will happen given the recent QAR data, which is typically represented as a multivariate time series. Specifically, a hard landing incident happens when the maximum vertical acceleration (VRTG) of the aircraft exceeds a predefined threshold after touchdown, as defined in Eq. (21). Then the problem is formally defined as follows:

Definition 1 (Hard Landing Prediction Problem) For a flight example, given the QAR multi-variate time series data X during the landing stage before touchdown, the hard landing prediction problem aims to learn a function f to prediction the hard landing label $\hat{y} = f(X; \Theta)$, where $\hat{y} = 1$ indicates hard landing and $\hat{y} = 0$ indicates normal landing, with Θ being the learnable parameters of the model.

The overall architecture of WMGIN is illustrated in Fig. 1. The left part consists of the embedding layer and a three-layer Time Window Encoder (TW Encoder), designed to process multi-scale information and extract multi-granularity local temporal relationships of individual variables within fixed windows. The right part is the three-layer Interpretable Hybrid Encoder (IH Encoder). Each of its layers takes as input the corresponding output from the TW Encoder, along with a learnable class token and a global inter-dimensional mask. The class token captures the contribution of original tokens within that layer to the prediction, while the global inter-dimensional mask explicitly models dynamic interactions among variables. Final predictions are derived from the class tokens, thereby providing interpretability to the model. Details of each module are presented in the following sections.

3.1 Embedding Layer

The dimension of the i-th variable in the input QAR data can be represented as $X_i \in \mathbb{R}^{Tf_i}$, where T denotes the span of the input data in seconds and f_i denotes the sampling frequency of the i-th variable. For each variable, the length of the time dimension is uniformly adjusted to T, meaning the sampling frequency is set to 1 Hz. While unifying the scale, the feature dimension should be expanded to a larger value to prevent information loss from high-sampling-frequency variables, defined as:

$$X'_{i,t} = \text{Concat}(X_{i,t\times f}, X_{i,t\times f+1}, \ldots, X_{i,t\times(f+1)-1}), \tag{1}$$

$$X'_i = \text{Stack}(X'_{i,1}, X'_{i,2}, \ldots, X'_{i,T}), \tag{2}$$

$$X_i^{emd} = \text{Linear}(X'_i), \tag{3}$$

where $X_{i,j}$ denotes the j-th value of the i-th variable, $X'_{i,t} \in \mathbb{R}^{f_i}$ denotes the data of the i-th variable within the t-th second, $X'_i \in \mathbb{R}^{T\times f_i}$ is obtained by stacking the data of the i-th variable per second, and $X_i^{emd} \in \mathbb{R}^{T\times d_model}$ presents the embedding result after a linear transformation. It is worth noting that variables with the same sampling frequency share a common embedding layer. After embedding, all variables are stacked along the feature dimension to form the input for the next module.

3.2 Time Window Encoder (TW Encoder)

In order to capture the local temporal relationship of a single variable at the time level effectively, we subtly combine the window strategy with Transformer and draw inspiration from the concept of channel-independent design. Channel-independent design refers to sequentially inputting individual variables into the backbone network, which allows the model to focus solely on learning the relation along the time dimension

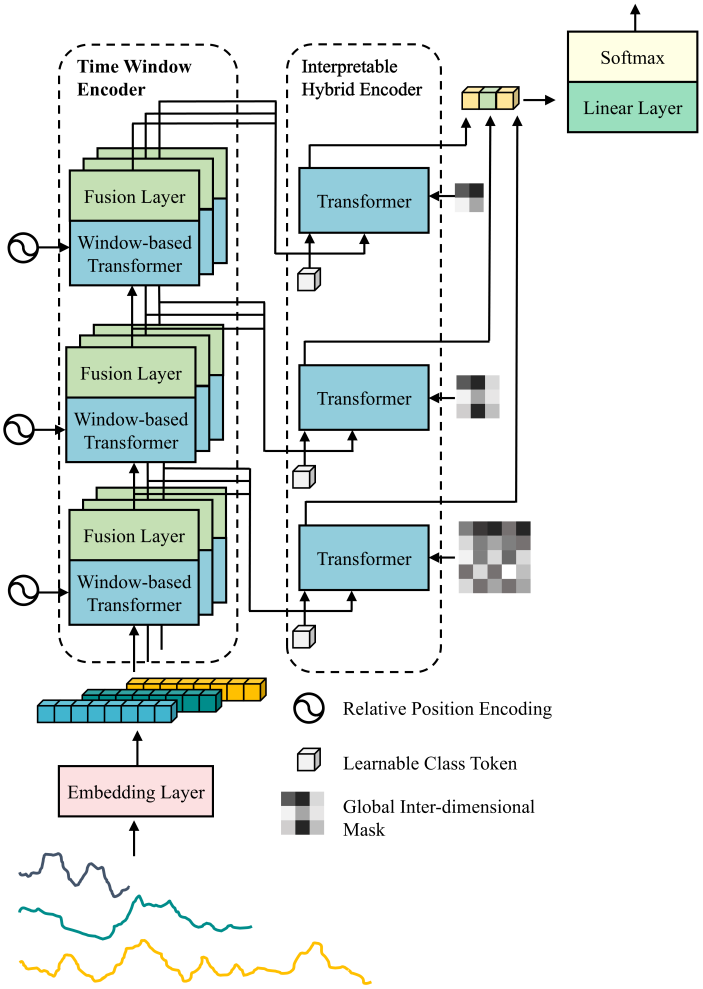


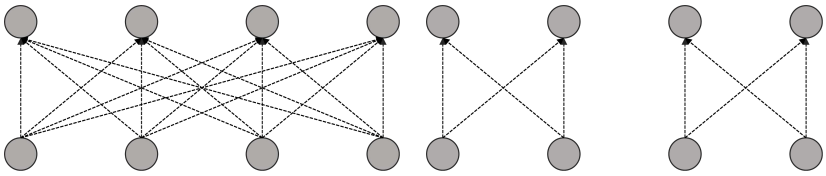
Fig. 1 The architecture of the WMGIN model, which consists of three modules: (1) Embedding Layer processes the multi-scale input data. (2) The Time Window Encoder is responsible for extracting the local temporal relationship of a single variable from fine-grained to coarse-grained. (3) Interpretable Hybrid Encoder provides the multi-granularity interpretability of the model and learns global dynamic temporal correlation among all variables.

[38]. The standard self-attention mechanism processes time series data with the entire series as input, resulting in a global receptive field, as shown in Fig.2 (a). In contrast, the window-based self-attention takes non-overlapping windows of size tokens from the original input. Fig.2 (b) illustrates the calculation pattern of the window-based self-attention with a window size of 2. Notably, the two computation methods are the

same; only the inputs differ. The formula is as follows:

$$\text{Atten}(X, W^{pos}) = \text{Softmax}\left(\frac{(XW^q)(XW^k)^T}{\sqrt{d_k}} + W^{pos}\right) \cdot (XW^v), \tag{4}$$

where $X \in \mathbb{R}^{T \times d_model}$ denotes the input time series, $W^q \in \mathbb{R}^{d_model \times d_q}$, $W^k \in \mathbb{R}^{d_model \times d_k}$ and $W^v \in \mathbb{R}^{d_model \times d_v}$ are linear transformation matrices that convert X into Q, K and V respectively. In this paper, $d_q = d_k = d_v$, and W^{pos} denotes the position encoding.



(a) The standard self-attention (b) The window-based self-attention

Fig. 2 Two types of self-attention mechanisms' different ways of processing time series data.

Obviously, after the data goes through the standard self-attention mechanism, information is propagated and aggregated among all tokens, thus integrating global information. In other words, the information barrier is broken, and the output can no longer represent the information corresponding to the original data. However, in our proposed method, information exchange is only achieved within a fixed window, thereby establishing a strong information barrier and providing a foundation for the interpretability of the model. Meanwhile, we adopt a multi-layer fusion design pattern to gradually increase the temporal information contained within a single window, which is also a prerequisite for ensuring the multi-granularity interpretability of our model. Fig.3 shows the process of each TW block, where adjacent time tokens are merged by the Fusion Layer after passing through the Window-based Transformer; as a result, the updated token contains more and coarser-grained temporal information. Based on the window-based self-attention mechanism mentioned earlier, we use to $x_{d,i}^l \in \mathbb{R}^{d_model}$ represent the input of the i-th token of the d-th variable in the l-th layer of the TW Encoder, where $d = 1, 2, \ldots, D$ and $l = 1, 2, \ldots, L$. At the same time, we use $x_{d,i}^{l,win} \in \mathbb{R}^{2d_model}$ to represent the i-th window after merging every two non-overlapping tokens as one window, and $i = 1, 2, \ldots, T/2^l$. The calculation method of TW Encoder can be defined as:

$$x_{d,i}^{l,win} = (x_{d,2i-1}^l, 2x_{d,2i}^l), \tag{5}$$

$$x_{d,i}^{l,win'} = \text{LN}\left(\text{MHA}\left(x_{d,i}^{l,win} + W^{pos}\right) + x_{d,i}^{l,win}\right), \tag{6}$$

$$x_{d,i}^{l,win''} = \text{LN}\left(\text{FNN}\left(x_{d,i}^{l,win'}\right) + x_{d,i}^{l,win'}\right), \tag{7}$$

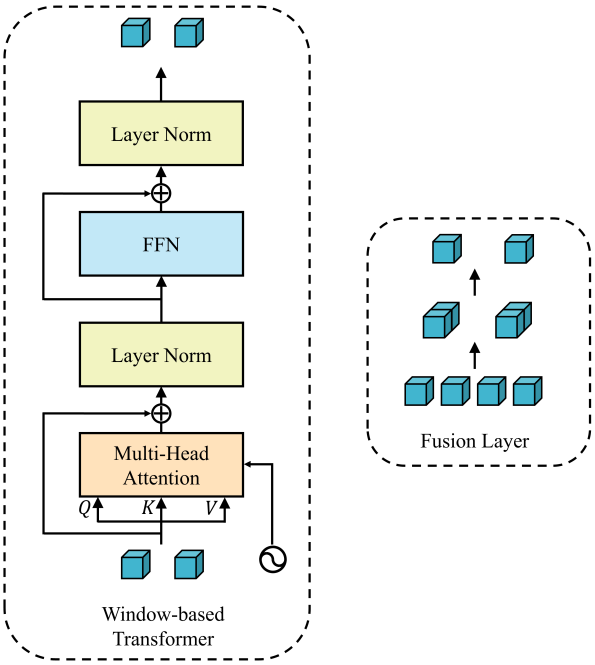


Fig. 3 The process of each TW block. The blue cube is the time token. After passing through the Window-based Transformer, adjacent tokens are merged by the Fusion Layer.

$$x_{d,i}^{l+1} = \text{Merge}\left(x_{d,2i-1}^{l,''}, x_{d,2i}^{l,''}\right) W^{merge}, \tag{8}$$

where Merge denotes the concatenation operation of the two adjacent tokens in the feature dimension, $W^{merge} \in \mathbb{R}^{2d_model \times d_model}$ is a learnable linear transformation matrix, and LN is the layer normalization operation.

MHA is the multi-head attention mechanism, which can be calculated as:

$$head_h = \text{Atten}\left(X_h, W_h^{pos}\right), \tag{9}$$

$$\text{MHA}\left(X, W^{pos}\right) = \text{Concat}\left(head_1, head_2, \ldots, head_h\right) W^o, \tag{10}$$

where h denotes the h-th head, $X_h \in \mathbb{R}^{2\times \frac{d_model}{h}}$ denotes the input of the h-th head, and $W^o \in \mathbb{R}^{hd_v \times d_model}$ is a matrix, where d_v is the number of feature dimension of the V after the linear transformation for input of each head. The role of W_o is to project the concatenated features of multiple heads back to d_model. In this paper, $W^{pos} \in \mathbb{R}^{h\times 2\times 2}$ is the relative position encoding (RPE) matrix, where h is the number of heads in the multi-head attention mechanism, and $W_h^{pos} \in \mathbb{R}^{2\times 2}$ denotes the RPE of the h-th head. The self-attention mechanism is always based on a window of fixed size 2, so the relative positions of the tokens within the window remain constant. In this paper, RPE is assigned only within the window. Since a multi-head attention mechanism is

used, h learnable 2×2 matrices are needed to represent position encodings, where each head's RPE captures the relative distance relationship between two adjacent tokens. The relative distance between two adjacent tokens can be one of three values: -1, 0, or 1. Thus, the relative position encoding for the h-th head of each window is:

$$W_h^{pos} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_1 \end{bmatrix}, \tag{11}$$

where a_1 to a_3 are learnable parameters. The diagonal position represents the relative distance of a token to itself, while the other two values correspond to relative distances of -1 and 1, respectively.

FFN consists of two linear layers and a ReLU activation function. The first linear layer expands the feature dimension to $4 \times d_model$ and then passes through the ReLU activation function. The second linear layer changes the number of feature dimensions back to d_model, because the subsequent residual connection requires the feature dimension to remain consistent before and after the transformation. It can be calculated as:

$$X = \text{Linear}\left(\text{ReLU}(\text{Linear}(X))\right). \tag{12}$$

3.3 Interpretable Hybrid Encoder (IH Encoder)

For QAR data, the correlation between different variables over time periods varies with the distance of the time interval. For example, when a pilot adjusts control parameters, the corresponding controlled parameters change over a certain period; however, as the time interval increases, their correlation with prior actions weakens — a result of the aircraft's design. QAR data reflects real-time conditions, so correlations among other parameters follow a similar pattern: parameters far apart in time typically do not influence each other. Therefore, we propose a novel global inter-dimensional mask.

Since the input of the first layer of the IH Encoder has already undergone fusion, the number of tokens for a single variable in the time dimension is T=16. Set the current time period be the zero point. Time periods before the zero point are negative, decreasing by 1 for each preceding period. Time periods after the zero point are positive, increasing by 1 for each subsequent period, and vice versa. For two time periods, if one is defined as position 0, the relative position of the other time period is its position relative to the zero point. In the case of T=16, there are 31 relative positions for two different variables, with a range of [-15,15]. We use a deformed Gaussian function as the reference curve, from which 31 function values are taken at equal intervals to serve as the weight values for the global inter-dimensional mask. The formula for the reference curve is:

$$y = e^{-2x^2} + 0.001, \tag{13}$$

where the exponent -2 adjusts the curve's steepness, allowing values near the center to decrease more rapidly. The added 0.001 prevents the weights on both ends from becoming too small, avoiding gradient vanishing. The value results are shown in Fig.4. The 31 values form a vector $Mask^{\text{init}}$. Based on it, we construct time-aware masks $Mask^{\text{diff}} \in \mathbb{R}^{(T/2^{l-1})\times(T/2^{l-1})}$, enabling the generation of global cross-dimensional masks for the IH Encoder (see Algorithm 1). In Algorithm 1, $Mask^l \in \mathbb{R}^{(DT/2^{l-1})\times(DT/2^{l-1})}$ (with

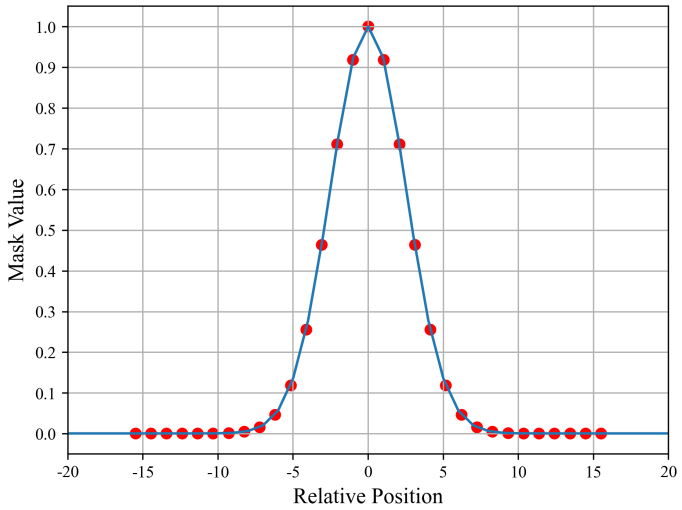


Fig. 4 Deformed Gaussian Function. The blue curve is the reference curve, and the red dots are the mask values of different relative positions.

D variables) is the inter-dimensional mask for layer l. For any two variables, we set one as the base with its first time step as reference (zero point). The mask values of the other variable decrease from max to min as the time distance increases — **assigning higher weights to closer time steps**. The weight is 1 for the same time step. To match reduced token counts in deeper layers, each layer's mask is downsampled via `AvgPool` (kernel and stride $= 2^{l-1}$).

After obtaining the masks used in each layer, the self-attention mechanism calculation method in the IH Encoder will change, as follows:

$$\text{Atten} = \text{softmax}\left(\text{sigmoid}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot W^{pos}\right) \cdot V. \tag{14}$$

Applying the sigmoid function to attention scores ensures positivity, preventing small negative values from being amplified during mask multiplication. Multiplying the attention scores by the mask before applying the Softmax allows for the constraint of the mask: smaller mask values significantly reduce the attention of these positions, thereby reducing their impact on subsequent model predictions, while larger masks have a smaller impact.

The IH Encoder is designed to learn global dynamic temporal correlation among all variables. By incorporating a global inter-dimensional mask and multi-head attention mechanism, the encoder can effectively handle data, capturing the dynamic dependencies and temporal features between different variables. Besides, the encoder utilizes multiple learnable class tokens to provide multi-granularity interpretability for the model.

Technologically, using Z^l to represent the input of the l-th layer in the IH Encoder, where $l = 1, 2, \ldots, L$. The computation process for each layer of the IH Encoder can

Algorithm 1 Construction of global inter-dimensional mask for each layer in the IH Encoder.

Input: T, D, L, $Mask^{init}$
Output: $Mask^{l}$
Define $Mask^{diff} \in \mathbb{R}^{T \times T}$
For $start \leftarrow T$ to 1 **do**
  $Mask^{diff}[T+1-start] \leftarrow Mask^{init}[start : start+T]$
end for
Define $Mask^{1} \in \mathbb{R}^{(D \times T) \times (D \times T)}$
For $i \leftarrow 1$ to D **do**
  For $j \leftarrow 1$ to D **do**
    if $i == j$ **then**
      $Mask^{1}_{(i \times T:(i+1) \times T, j \times T:(j+1) \times T)} \leftarrow 1$
    else
      $Mask^{1}_{(i \times T:(i+1) \times T, j \times T:(j+1) \times T)} \leftarrow Mask^{diff}$
  end for
end for
For $l \leftarrow 2$ to L **do**
  $Mask^{l} \leftarrow \text{AvgPool}(Mask^{1})$
end for
return $Mask^{1}, Mask^{2}, \ldots, Mask^{L}$

be defined as:

$$Z^{l} = \text{Concat}(cls^{l}, x_1^{l}, x_2^{l}, \ldots, x_D^{l}), \tag{15}$$

$$Z^{l'} = \text{LN}(\text{MHA}(Z^{l}, Mask^{l'}) + Z^{l}), \tag{16}$$

$$Z^{l''} = \text{LN}(\text{FFN}(Z^{l'}) + Z^{l'}) = \text{Concat}(cls^{l''}, x_1^{l''}, x_2^{l''}, \ldots, x_D^{l''}), \tag{17}$$

where $x_d^{l} \in \mathbb{R}^{(T/2^{l-1}) \times d_model}$ is all output tokens of d-th variable at the l-th layer of the TW Encoder, and $cls^{l} \in \mathbb{R}^{1 \times d_model}$ denotes the learnable class token of the l-th layer. After MHA, the model learns the weights of all tokens in the layer to the class token and only uses the output of the class tokens in subsequent predictions. The reason for adding a learnable class token is that the attention weights learned by the multi-head attention mechanism represent the similarity between different variables across different time periods, which does not directly reflect their impact on the prediction results. However, after adding the learnable class token, cls^{l} has no semantic similarity with the other tokens. Therefore, from a computational perspective, the attention weights of all other tokens on cls^{l} measure the impact of other tokens on the prediction results. The $Mask^{l'}$ is modified based on the $Mask^{l}$, because of the addition of the cls^{l}. In the $Mask^{l'}$, for the parts related to cls^{l}, both the mask from other tokens to cls^{l} and the mask from cls^{l} to other tokens are set to 1.

3.4 Prediction

Concatenate the first element, which is the learnable class token of the output from each layer of the IH Encoder, to obtain a vector $x \in \mathbb{R}^{Ld_model}$. The class tokens are

used because they capture the global and multi-granularity information from both the TW Encoder and the IH Encoder. This vector is then passed through a linear layer to produce output $\hat{y}$. The above process can be defined as:

$$x = \text{Concat}(cls^{1''}, cls^{2''}, \ldots, cls^{L''}), \tag{18}$$

$$\hat{y} = \text{Linear}(x), \tag{19}$$

The binary cross-entropy function is used as the loss function for training the model as follows:

$$\mathcal{L}(y, \hat{y}) = -ylog\hat{y} - (1-y)log(1-\hat{y}), \tag{20}$$

, where y represents the true label of the sample, while $\hat{y}$ is the probability predicted by the model for the sample being a positive case.

4 Experimental Evaluation

4.1 Datasets

The dataset used in this study comprises 44,804 A321 flights that took off or landed at a domestic airport in China. The QAR data for each flight are stored in a CSV file, where each row corresponds to one second and each column to a flight parameter. In total, 145 parameters are recorded with varying sampling frequencies, among which 19 are utilized in this paper, as summarized in Table 1.

Table 1 Description of the QAR parameters.

No.	Parameter	Description	Sampling Rate
1	ALT_STD	Standard altitude corrected	1 Hz
2	IAS	Indicated airspeed	1 Hz
3	GS	Ground speed	1 Hz
4	IVV	Vertical speed	1 Hz
5	N11	Engine 1 speed ratio	1 Hz
6	N12	Engine 2 speed ratio	1 Hz
7	TLA1	Throttle lever 1 position	1 Hz
8	TLA2	Throttle lever 2 position	1 Hz
9	GW	Aircraft gross weight	1 Hz
10	WIN_CRS	Cross wind	1 Hz
11	WIN_ALG	Longitudinal wind	1 Hz
12	ROLL	Roll angle	2 Hz
13	RUDD	Rudder position	2 Hz
14	RADIO_LH	Left radio height	4 Hz
15	RADIO_RH	Right radio height	4 Hz
16	PITCH	Pitch angle	4 Hz
17	VRTG	Vertical acceleration	8 Hz
18	PITCH_CMD	Pitch control	8 Hz
19	ROLL_CMD	Roll control	8 Hz

In this paper, the maximum vertical acceleration (VRTG) within 2 seconds after the landing time of the aircraft is used to judge whether a hard landing occurs,

formulated as:

$$\max\left(\mathrm{VRTG}_{\mathrm{landing}}\right) \geq \theta. \tag{21}$$

We set the threshold θ to 1.5 g. The 44,804 flights are categorized into 44,115 normal landings and 689 hard landings, where the latter are treated as positive samples and the former as negative samples. A stratified sampling strategy is employed to divide the data into training, validation, and test sets with a ratio of 6:2:2. To address class imbalance, positive samples in the training set are upsampled to match the number of negative samples. After upsampling, the number of positive samples in the training set is increased form 413 to 26,469, equaling the number of negative samples. The original data distribution is preserved in the validation and test sets to evaluate model performance under real-world conditions. The resulting sample sizes for the training, validation, and test sets are 52,938, 8,961, and 8,961, respectively, with balanced classes in the training set. Each of the validation and test sets contains 138 hard landings and 8,823 normal landings. In addition, three temporal segments are extracted for early warning and interpretability: from 36s to 4s before landing (Type III), from 34s to 2s before landing (Type II), and from 32s to touchdown (Type I). These three data types are referred to as I, II, and III, respectively, in the remainder of the paper.

4.2 Benchmark Methods

The following baselines are used for comparison.

- **RF (Random Forest) [13]**: Sun et al. qualitatively analyzed the factors influencing hard landings using QAR data, based on which they developed a random forest classifier to predict hard landing events. Owing to the strong generalization capability of RF, their model achieved a recall of 85.50%.
- **LSTM [39]**: Long Short-Term Memory (LSTM) networks are widely used for hard landing prediction owing to their inherent strength in modeling time-series data. Zhang et al. [39] formulated hard landing detection as a classification task and employed a three-layer LSTM model to predict hard landing events.
- **MLSTM-FCN [40]:** The MLSTM-FCN model combines LSTM with the Fully Convolutional Network (FCN) [41], and incorporates a squeeze-and-excitation block. This block explicitly models inter-channel dependencies to enhance the model's classification performance on multivariate time-series data.
- **Transformer [42]**: Zerveas et al. proposed a transformer-based framework for multivariate time-series representation learning. The method employs a channel-mixing strategy, in which all parameters at each time step are combined and projected into a shared vector space. Along the temporal dimension, it processes individual time points rather than intervals. The resulting representations can be directly used for downstream classification tasks.
- **FormerTime [43]:** FormerTime adopts a hierarchical network architecture, where data fusion is performed at each layer to learn multi-scale representations. The resulting feature representations are then utilized for classification. In addition, FormerTime introduces a novel strategy for generating contextual positional encodings.

- **PatchTST [38]:** PatchTST employs a channel-independent strategy that inputs time series data into the model as patches, achieving strong performance in time-series forecasting tasks. Originally designed for long-term prediction, PatchTST can be adapted for hard landing prediction by replacing its final prediction head with a binary classification head.

To ensure consistent input lengths, parameters with sampling frequencies higher than 1 Hz are averaged per second. For a fair comparison across methods, model hyperparameters are fine-tuned using cross-validation. Furthermore, to comprehensively assess model generalization, all experiments are repeated five times with different random seeds, and the reported results represent the average performance.

Due to the imbalanced distribution of samples in the validation and test sets, accuracy is no longer an appropriate metric for evaluating model performance. Instead, we adopt precision, recall, F1-score, and PR-AUC (Area Under the Precision–Recall Curve) as evaluation metrics. Precision measures the proportion of correctly identified hard landing samples among all predicted positives, reflecting the reliability of positive predictions. Recall quantifies the proportion of actual positive samples correctly identified by the model, indicating its capability to uncover hard landing events. The F1-score, as the harmonic mean of precision and recall, provides a balanced assessment of both. PR-AUC summarizes the overall trade-off between precision and recall across different thresholds, offering a more robust evaluation under class imbalance.

4.3 Experimental setup

WMGIN standardizes all QAR parameters with varying sampling frequencies to a unified temporal length of $T = 32$, with the feature dimension set to $d_model = 128$. The model consists of $L = 3$, and each multi-head self-attention module includes 4 heads. The attention window size in the TW Encoder and the data fusion window size in the fusion layer are both set to 2.

In terms of model training, the batch size is set to 128. The Adam optimizer is used with an initial learning rate of 0.0001, and the entire training process consists of 100 iterations. During training, the learning rate undergoes step decay, reducing to half of its original value every 20 iterations. To prevent overfitting, the dropout rate is set to 0.5, and the L2 regularization weight is set to 0.0005.

4.4 Results and Discussion

4.4.1 Overall Results

Table 2 presents the performance of all models on the three data types, with the best results highlighted in bold. As shown, WMGIN achieves the highest performance in nearly all cases, except for the precision on Type I data and the recall on Type II data. Although WMGIN restricts the computation of attention weights to a local window in the TW Encoder—thus capturing primarily local dependencies—the IH Encoder leverages the weighted representations of different parameters and time steps for prediction, enabling the model to capture global information and compensate for the TW

Encoder's limitation. Overall, Transformer-based models demonstrate the best performance, followed by LSTM and MLSTM-FCN, while the random forest exhibits a clear performance gap. This trend reflects the relationship between model complexity and representation capacity: when overfitting is not a concern, models with greater parameterization tend to better capture nonlinear relationships, resulting in improved detection and prediction of hard landings. Among the four Transformer-based models, WMGIN achieves the best predictive accuracy, with PatchTST performing slightly lower—confirming the effectiveness of WMGIN's multi-layer fusion design. The performance of Transformer and FormerTime varies across data types, each showing distinct strengths. Although FormerTime enhances Transformer by adopting a hierarchical structure and performing layer-wise data fusion to learn multi-scale features, its overall performance remains lower than that of WMGIN.

Table 2 Classification results of models on three types of data.

Data Type	Model	Precision	Recall	F1	PR-AUC
I	RF	32.04	90.36	47.70	47.14
	LSTM	39.76	91.53	54.35	52.78
	MLSTM-FCN	37.92	93.83	54.06	54.88
	Transformer	35.81	94.02	52.87	60.02
	FormerTime	38.76	95.37	55.10	62.38
	PatchTST	**39.02**	95.51	55.68	70.18
	WMGIN	38.93	**96.25**	**55.73**	**71.53**
II	RF	27.43	87.81	41.49	42.38
	LSTM	32.21	90.34	48.16	47.06
	MLSTM-FCN	32.50	91.25	48.29	49.57
	Transformer	33.68	91.63	49.20	58.90
	FormerTime	32.91	89.67	48.95	57.82
	PatchTST	34.02	**92.98**	50.32	68.85
	WMGIN	**35.97**	92.74	**51.78**	**69.77**
III	RF	23.16	85.91	36.78	40.45
	LSTM	31.59	86.58	46.70	45.86
	MLSTM-FCN	29.54	87.29	43.94	44.28
	Transformer	31.28	87.14	46.15	56.50
	FormerTime	28.45	88.04	43.07	55.91
	PatchTST	31.85	88.78	47.05	64.32
	WMGIN	**32.67**	**91.33**	**48.53**	**67.54**

Notably, in the context of a hard landing warning, a normal landing misclassified as a hard landing merely indicates a potential risk. In contrast, misclassifying a hard landing as normal may delay pilot reactions and lead to actual hard landing events. Therefore, for this task, recall is a more critical metric than precision. Additionally, due to the pronounced class imbalance in the test sets, both precision and F1-score remain relatively low across all experiments.

4.4.2 Ablation Experiment

To assess the contribution of different components in WMGIN, a series of ablation experiments is conducted, and the results are summarized in Table 3. Specifically:

- **WMGIN (EMB–):** The Embedding Layer is removed. The variables are averaged per second, their feature dimensions are constructed, and they are linearly transformed before being fed into the TW Encoder.
- **WMGIN (FUS–):** The Fusion Layer in each block of the TW Encoder is removed, and the receptive field of the IH Encoder is adjusted accordingly.
- **WMGIN (RPE–):** The relative position encoding (RPE) module is removed.
- **WMGIN (MSK–):** The global inter-dimensional mask (MSK) is removed.

Table 3 The ablation experiments of WMGIN on three types data

Data Type	Model	Precision	Recall	F1	PR-AUC
I	WMGIN (EMB-)	38.17	94.08	54.58	65.02
	WMGIN (FUS-)	37.78	93.53	52.52	63.87
	WMGIN (RPE-)	37.45	94.03	53.88	69.72
	WMGIN (MSK-)	36.27	95.01	54.82	70.24
	WMGIN	**38.93**	**96.25**	**55.73**	**71.53**
II	WMGIN (EMB-)	**36.08**	92.09	51.49	63.70
	WMGIN (FUS-)	33.82	89.41	48.98	60.48
	WMGIN (RPE-)	34.24	91.58	49.69	67.12
	WMGIN (MSK-)	34.42	91.13	49.96	67.98
	WMGIN	35.97	**92.74**	**51.78**	**69.77**
III	WMGIN (EMB-)	32.53	89.60	46.82	60.95
	WMGIN (FUS-)	29.84	87.16	44.27	58.52
	WMGIN (RPE-)	30.57	89.98	45.63	64.80
	WMGIN (MSK-)	31.49	89.45	46.52	64.48
	WMGIN	**32.67**	**91.33**	**48.53**	**67.54**

As shown in Table 3, removing either the Fusion Layer or the Embedding Layer leads to a decline in model performance. The degradation observed in WMGIN (FUS–) is more pronounced than in WMGIN (EMB–), indicating that the Fusion Layer contributes more significantly to model effectiveness. The Fusion Layer enables WMGIN to capture multi-granularity local temporal dependencies, while the Embedding Layer effectively processes multi-scale input data and prevents information loss. Furthermore, the removal of the RPE module results in a noticeable performance drop, highlighting the crucial role of positional encoding in helping the model capture temporal order information. When the global inter-dimensional mask is omitted in the IH Encoder, performance decreases substantially. This is likely because the product of the time window size and the number of variables, corresponding to the number of tokens fed into the IH Encoder, is relatively large. Without an external mask, it becomes difficult for the model to directly learn relationships across different variables and time steps, thereby requiring more data for adequate fitting.

4.4.3 Model Interpretability Analysis

To interpret hard landing events, we extract and visualize the attention weights of the learnable class tokens under Data Type I. The attention weight dimension of the l-th layer is $\mathbb{R}^{\mathrm{HDT}/2^l}$, where $H = 4$ denotes the number of attention heads. The attention weights from all heads within each layer are jointly analyzed to explain the model's hard landing predictions. In the visualization, brighter regions indicate higher attention importance, whereas darker regions correspond to lower importance.

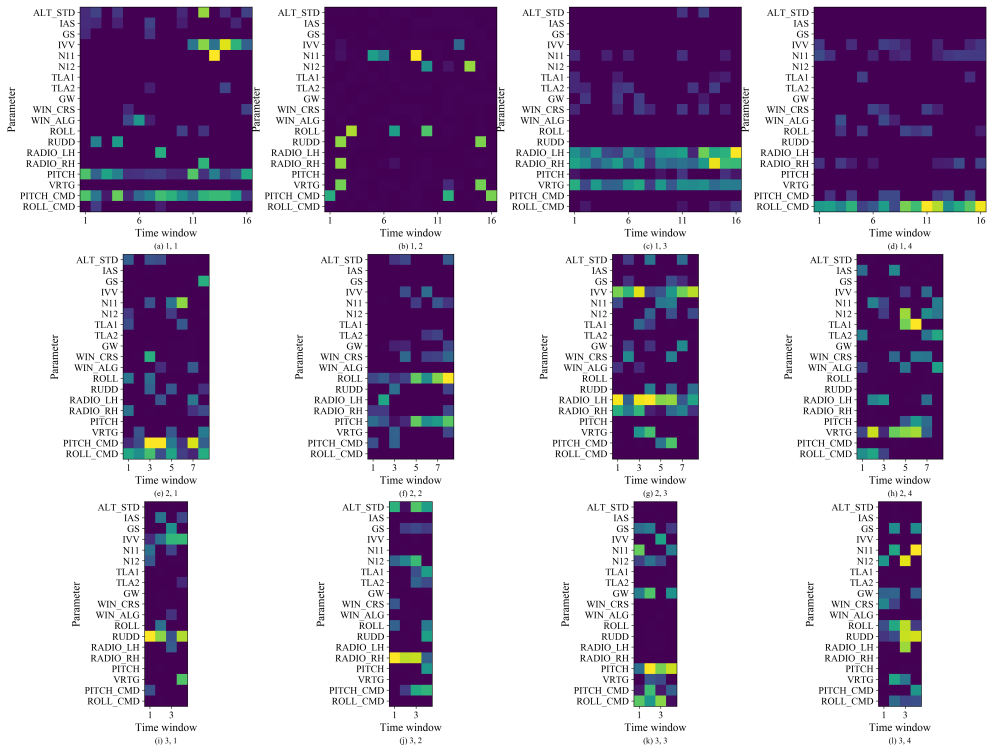


Fig. 5 The attention weights obtained from a hard landing flight.

Fig. 5 illustrates the three layers of attention weights for a representative hard landing flight. Each layer contains four subfigures corresponding to the four attention heads. This flight experienced a maximum vertical acceleration (VRTG) of 1.79g, occurring 1s after touchdown, and WMGIN correctly classified it as a hard landing with a probability of 0.92. Since the input to the first layer of the IH Encoder has already passed through a fusion layer in the TW Encoder, each time window in the first layer's class token representation corresponds to 2s of data. As shown in Fig. 5(a), the first head of the first layer assigns the highest attention to IVV from 12s before touchdown to landing, and to PITCH and PITCH_CMD throughout the entire 32-second sequence. The second head, as presented in Fig. 5(b), shows no clear distribution

pattern. The third and fourth heads, as shown in Fig. 5(c)–(d), focus strongly on RADIO_LH, RADIO_SH, VRTG, and ROLL_CMD, indicating their significant influence on hard landing events. These results suggest that different heads capture distinct data patterns, enabling the model to attend to diverse aspects of the flight. Among the influential parameters, IVV, RADIO_LH, and RADIO_SH are closely related to the descent process, while PITCH_CMD, ROLL_CMD, and their controlled variables (PITCH, ROLL) reflect pilot operations that directly affect aircraft attitude. Given that hard landing severity is directly related to VRTG, the model's focused attention on this parameter further confirms its effectiveness in identifying critical indicators of hard landings.

In the second layer, although the attention patterns learned by each head differ from those in the first layer, the model still primarily focuses on parameters such as IVV, RADIO_LH, RADIO_SH, PITCH, ROLL, PITCH_CMD, ROLL_CMD, and VRTG. These parameters are thus particularly critical for hard landing prediction. Moreover, across all heads in the second layer, the model assigns higher attention weights to certain time windows, indicating that different time periods contribute unequally to the prediction outcome. In the third layer, the model also attends to specific parameters such as RUDD, ALT_STD, N11, and N12, suggesting that, at a coarser granularity, these parameters exert a notable influence on hard landing prediction. Building upon the findings from the previous two layers, this layer focuses more on abnormal descent patterns and pilot control behaviors.

To further investigate the impact of these two factors on hard landings, Fig. 6 presents the original values of eight key parameters, i.e., IVV, VRTG, RADIO_LH, RADIO_RH, PITCH, PITCH_CMD, ROLL, and ROLL_CMD. In the figure, the red line represents the hard landing flight, the blue line denotes the mean of all flights, and the shaded regions with varying transparency indicate percentile ranges of the overall parameter distributions. The horizontal axis corresponds to the time period from 31s to 0s before touchdown, while the vertical axis represents parameter values. Note that VRTG in Fig. 6(b) does not reach 1.79g, as the maximum value occurs one second after touchdown, outside the observation window. As shown in Fig. 6, IVV, RADIO_LH, and RADIO_RH remain consistently high throughout the descent, confirming the abnormal descent pattern observed in Fig. 5. Other parameters, including VRTG, PITCH_CMD, and ROLL_CMD, exhibit either large deviations from the mean or pronounced fluctuations, resulting in substantial variations in PITCH and ROLL.

According to domain knowledge and standard flight operation guidelines, an aircraft typically passes over the runway threshold at an altitude of approximately 50ft. As shown in Fig. 6(c), the analyzed aircraft took about 5s to descend from 50ft to touchdown, significantly shorter than the group average of 9s. Normally, once the aircraft descends below 50ft, the pilot should focus on the aircraft's altitude, pull back on the control stick at around 30ft to increase the pitch angle, which generates additional lift and reduces the vertical velocity, thereby achieving a smooth landing, as observed in the group behavior. However, in this case, the pilot deviated from standard procedure and frequently adjusted the pitch control during descent, causing large fluctuations in the pitch angle. Consequently, the vertical velocity remained unstable and failed to return to the normal range. Although the pilot attempted to increase

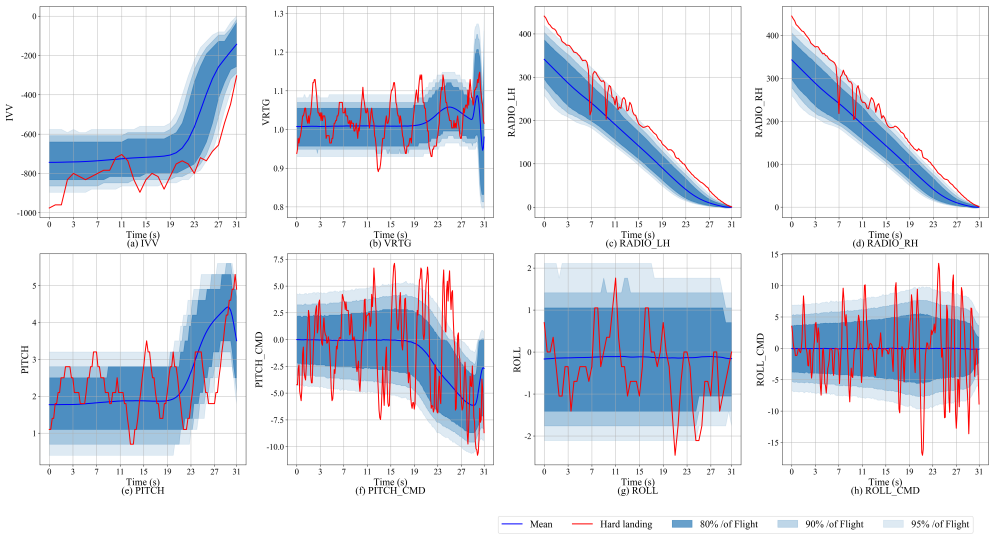


Fig. 6 Visualization of eight flight parameters for a hard landing flight.

the pitch angle shortly before touchdown, the vertical velocity was too high to be sufficiently reduced, ultimately resulting in a hard landing. In summary, the primary cause of this event was the pilot's failure to increase the pitch angle in time to mitigate vertical velocity. To prevent such risks, pilots should closely monitor aircraft state parameters during the landing phase and apply timely, smooth control inputs to ensure a stable touchdown. Additionally, the pronounced fluctuations in ROLL_CMD, ROLL, and VRTG indicate instability in aircraft attitude and dynamics, further contributing to the hard landing outcome. Overall, the visualization results demonstrate that WMGIN effectively identifies key parameters and their corresponding critical time windows related to hard landing events, providing strong interpretability and valuable insights into flight safety analysis.

5 Conclusion

The interpretability of hard landing prediction results is crucial for understanding and preventing landing safety events. To enhance both predictive performance and interpretability, this paper proposes a novel model, WMGIN, based on the self-attention mechanism, which provides high prediction accuracy and multi-granularity interpretability. Experimental results on the QAR dataset verify the model's effectiveness in both predicting and explaining hard landing events. A limitation of this study is that it focuses solely on a single type of event. In future work, we plan to extend WMGIN as a backbone network for multi-classification tasks involving various flight safety events, such as tail strikes, runway overruns, and long landings. By exploring the intrinsic relationships among these events, we aim to further enhance the model's applicability to real-world aviation safety management.

References

[1] Association), I.I.A.T.: 2023 iata annual safety report executive summary. https://www.iata.org/en/publications/safety-report/executive-summary/ (2024)

[2] Airplanes, B.C.: Statistical summary of commercial jet airplane accidents worldwide operations 1959-2022. https://www.boeing.com/safety (2024)

[3] Lv, H., Yu, J., Zhu, T.: A novel method of overrun risk measurement and assessment using large scale qar data. In: 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), pp. 213–220 (2018). IEEE

[4] Li, X., Zhang, L., Shang, J., Li, X., Qian, Y., Zheng, L.: A runway overrun risk assessment model for civil aircraft based on quick access recorder data. Applied Sciences **13**(17), 9828 (2023)

[5] Wang, L., Wu, C., Sun, R.: Pilot operating characteristics analysis of long landing based on flight qar data. In: Engineering Psychology and Cognitive Ergonomics. Applications and Services: 10th International Conference, EPCE 2013, Held as Part of HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part II 10, pp. 157–166 (2013). Springer

[6] Kang, Z., Shang, J., Feng, Y., Zheng, L., Wang, Q., Sun, H., Qiang, B., Liu, Z.: A deep sequence-to-sequence method for accurate long landing prediction based on flight data. IET Intelligent Transport Systems **15**(8), 1028–1042 (2021)

[7] Yang, X., Ren, J., Li, J., Zhang, H., Yang, J.: Data-driven long-landing event detection and interpretability analysis in civil aviation. IEEE Access **10**, 64257–64269 (2022)

[8] Chen, Y., Han, Z., Gu, J.: Analyzing the causes of tail strike event during takeoff with stepwise regression. In: 2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT), pp. 418–424 (2019). IEEE

[9] Zhang, N., Wang, L., An, J., Qi, X.: Risk analysis of human factors in aircraft tail strike during landing: A study based on qar data. In: International Conference on Human-Computer Interaction, pp. 178–187 (2023). Springer

[10] Wang, X., Yu, L., Zhao, X.: Discover the tail strike risk during take-off of an airline based on qar big data. In: 2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI), pp. 553–556 (2020). IEEE

[11] Wang, L., Ren, Y., Wu, C.: Effects of flare operation on landing safety: A study based on anova of real flight data. Safety science **102**, 14–25 (2018)

[12] You, Y., Qin, K., Yu, Y.: Heavy landing detection and prediction based on qar data. In: 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT, pp. 1066–1071 (2020). IEEE

[13] Sun, R., Li, C.: A risk prediction model of hard landing based on random forest algorithm. In: E3S Web of Conferences, vol. 257, p. 02080 (2021). EDP Sciences

[14] Tong, C., Yin, X., Li, J., Zhu, T., Lv, R., Sun, L., Rodrigues, J.J.: An innovative deep architecture for aircraft hard landing prediction based on time-series sensor data. Applied Soft Computing **73**, 344–349 (2018)

[15] Lu, Y., Zhu, T.: Pre-training of autoregressive model for aircraft hard landing prediction based on qar data. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1613–1617 (2019). IEEE

[16] Jeong, S.H., Park, E.G., Cho, J.Y., Kim, J.H.: Development of automatic hard landing detection model using autoencoder. International Journal of Aeronautical and Space Sciences **24**(3), 778–791 (2023)

[17] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[18] Chen, R., Jin, J.: Detection of influential features of hard landing based on qar data. In: 2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT), pp. 526–529 (2021). IEEE

[19] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (long and Short Papers), pp. 4171–4186 (2019)

[20] Wang, L., Wu, C., Sun, R., Cui, Z.: An analysis of hard landing incidents based on flight qar data. In: Engineering Psychology and Cognitive Ergonomics: 11th International Conference, EPCE 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings 11, pp. 398–406 (2014). Springer

[21] Hu, C., Zhou, S.-H., Xie, Y., Chang, W.-B.: The study on hard landing prediction model with optimized parameter svm method. In: 2016 35th Chinese Control Conference (CCC), pp. 4283–4287 (2016). IEEE

[22] Qiao, X., Chang, W., Zhou, S., Lu, X.: A prediction model of hard landing based on rbf neural network with k-means clustering algorithm. In: 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 462–465 (2016). IEEE

[23] Qian, S., Zhou, S., Chang, W.: An improved aircraft hard landing prediction model based on panel data clustering. In: 2017 29th Chinese Control And Decision Conference (CCDC), pp. 438–443 (2017). IEEE

[24] Tian, X., Liu, K., Wang, X., Xu, Y.: Research on prediction of hard landing for aircraft based on ridge regression. In: 2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT), pp. 653–658 (2023). IEEE

[25] Li, X., Shang, J., Zheng, L., Wang, Q., Sun, H., Qi, L.: Curvecluster+: Curve clustering for hard landing pattern recognition and risk evaluation based on flight data. IEEE Transactions on Intelligent Transportation Systems **23**(8), 12811–12821 (2022)

[26] Kong, Y., Zhang, X., Mahadevan, S.: Bayesian deep learning for aircraft hard landing safety assessment. IEEE transactions on intelligent transportation systems **23**(10), 17062–17076 (2022)

[27] Li, X., Shang, J., Zheng, L., Wang, Q., Liu, D., Liu, X., Li, F., Cao, W., Sun, H.: Imtcn: An interpretable flight safety analysis and prediction model based on multi-scale temporal convolutional networks. IEEE transactions on intelligent transportation systems **25**(1), 289–302 (2024)

[28] Chen, H., Shang, J., Zheng, L., Li, X., Liu, X., Sun, H., Zhao, X., Yu, L.: Sdtan: Scalable deep time-aware attention network for interpretable hard landing prediction. IEEE Transactions on Intelligent Transportation Systems **24**(9), 10211–10223 (2023)

[29] Cai, J., Shang, J., Li, X., Li, C., Zheng, L.: Fine-grained time and hidden feature learning for interpretable hard landing prediction based on qar data. IEEE Transactions on Intelligent Transportation Systems (2025)

[30] Song, L., Li, X., Liu, H., Wu, L., Sun, H., Zheng, L., Shang, J.: Mdgnn: Multiple flight safety incidents prediction model based on dynamic graph neural networks. IEEE Transactions on Intelligent Transportation Systems (2025)

[31] Zhao, Z., Shi, Y., Wu, S., Yang, F., Song, W., Liu, N.: Interpretation of time-series deep models: A survey. arXiv preprint arXiv:2305.14582 (2023)

[32] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929 (2016)

[33] Kashiparekh, K., Narwariya, J., Malhotra, P., Vig, L., Shroff, G.: Convtimenet: A pre-trained deep convolutional neural network for time series classification. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2019). IEEE

[34] Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)

[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[36] Hsieh, T.-Y., Wang, S., Sun, Y., Honavar, V.: Explainable multivariate time series classification: a deep neural network which learns to attend to important variables as well as time intervals. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 607–615 (2021)

[37] Lim, B., Arık, S.Ö., Loeff, N., Pfister, T.: Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting **37**(4), 1748–1764 (2021)

[38] Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. In: Proc. 11th Int. Conf. Learn. Represent. (ICLR), pp. 1–24 (2023)

[39] Zhang, H., Zhu, T.: Aircraft hard landing prediction using lstm neural network. In: Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control, pp. 1–5 (2018)

[40] Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate lstm-fcns for time series classification. Neural networks **116**, 237–245 (2019)

[41] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 3431–3440 (2015)

[42] Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., Eickhoff, C.: A transformer-based framework for multivariate time series representation learning. In: Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min., pp. 2114–2124 (2021)

[43] Cheng, M., Liu, Q., Liu, Z., Li, Z., Luo, Y., Chen, E.: Formertime: Hierarchical multi-scale representations for multivariate time series classification. In: Proc. ACM Web Conf., pp. 1437–1445 (2023)