applied sciences

MDPI

Article

MoE-Based Anomaly Detection of High-Altitude UAV Flight Data

Deng Chen [1], Cancan Zuo [1], Lianqing Wang [2,3,*], Dawei Wen [1], Xin Nie [1], Xiaohua Liu [1], Xiaoquan Liu [1] and Anqi Cai [4]

[1] School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430205, China; dchen@wit.edu.cn (D.C.); 22407010072@stu.wit.edu.cn (C.Z.); daweiwen_whu@163.com (D.W.); niexin@whu.edu.cn (X.N.); xiaohua.liu@wit.edu.cn (X.L.); liuxiaoquan@wit.edu.cn (X.L.)
[2] College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
[3] College of Data Intelligence, Information Support Force Engineering University, Wuhan 430010, China
[4] Institute of Electronics and Telecommunications, Peter the Great St. Petersburg Polytechnic University, St. Petersburg 195251, Russia; tsaj.a@edu.spbstu.ru
* Correspondence: lianqingw@nudt.edu.cn

Abstract

Analyzing flight data from high-altitude unmanned aerial vehicles (UAVs) is an essential approach to anomaly detection and early warning. However, flight data collected during the climb, cruise, and descent phases of high-altitude UAV missions exhibit pronounced distributional differences, making it difficult for existing anomaly detection methods to establish a unified model of normal flight behavior across all phases, thereby constraining detection accuracy. To address this, this paper proposes an anomaly detection method for high-altitude UAV flight data based on a Mixture of Experts (MoE). Specifically, a Multi-Scale Dual-Path Mixture-of-Experts encoder is constructed to extract local perturbations from UAV flight sequences; multi-scale convolution, a Transformer, and a dual-path LSTM are fused to capture temporal dependencies within the flight data; and a gating network automatically adapts to and uniformly models the data distributions across different mission phases. Furthermore, a two-stage reconstruction error is employed to construct sliding-window anomaly scores, and the median absolute deviation (MAD) is adopted to adaptively determine the anomaly decision threshold, enabling detection of flight-data anomalies across all mission phases. Experimental results on a self-constructed high-altitude UAV flight dataset with controlled anomaly injections demonstrate that the proposed method achieves a window F1 of 93.97% and an event recall of 91.77% across three independent training seeds.

Keywords: anomaly detection; flight data; high-altitude UAV; median absolute deviation; mixture of experts

Check for updates

Academic Editor: Petros Nicopolitidis

Received: 8 September 2026
Revised: 29 September 2026
Accepted: 7 October 2026
Published: 12 October 2026

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

1. Introduction

High-altitude UAVs are widely deployed in military reconnaissance, border patrol, maritime surveillance and observation under harsh environments, and are characterized by long endurance and wide operational coverage. Such missions require sustained operation under low temperature, low pressure and strong aerodynamic disturbances, imposing stringent requirements on flight safety. Once an anomaly occurs during flight, the mission may be interrupted or the vehicle may even lose control. To monitor flight state in real

time, high-altitude UAVs are typically equipped with sensor-based telemetry systems that continuously acquire flight data; these data directly reflect the physical operating states and health conditions of the on-board subsystems [1]. Analyzing flight data therefore enables fault detection and timely warning, which is essential to flight safety [2].

Here, an anomaly is a deviation from learned normal behavior; a fault is an abnormal physical or functional condition of a component; and a failure is a loss of function. The detector identifies telemetry anomalies without diagnosing their physical cause.

High-altitude UAV flight data are a typical form of multivariate time series, covering parameters from multiple subsystems such as propulsion, electrical power, thermal management and structural health, navigation and flight control, and the atmospheric environment. Such data involve complex variable dependencies and temporal dependencies. Meanwhile, a high-altitude UAV generally experiences climb, cruise, and descent phases during a mission, and the data distributions of these phases are not consistent; the variable dependencies also change with the flight state. For example, data in the cruise phase are usually relatively stationary, whereas those in the climb and descent phases exhibit stronger time-varying behavior because of power regulation, attitude transitions, and environmental disturbances. Such phase-dependent distributional differences make it difficult to model flight-data anomalies entirely on the assumption of a globally unified data distribution [3], which degrades the stability of anomaly decisions.

Existing UAV flight-data anomaly detectors include traditional methods and deep-learning-based methods. Traditional methods use thresholds or statistical rules to capture limit violations and abrupt changes, but have limited ability to describe complex variable relations and temporal variations. Deep-learning-based methods have attracted wide attention for their strong modeling capacity. Audibert et al. proposed USAD [4], which adversarially trains two autoencoders and fuses the errors of two reconstruction paths into an anomaly score. Chen et al. [5] used a Transformer conditional encoder and a variational autoencoder to predict the conditional probability distribution of flight data, scoring anomalies by how far observations deviate from the predicted distribution. Huang et al. proposed Graph-MoE [6], which fuses variable-relation representations from different levels of a graph neural network through multiple experts and flags low-likelihood samples as anomalies via density estimation. Recent work has developed along several related directions, including comparative evaluation of unsupervised deep architectures on real-world flight data [7], transfer learning under limited UAV data [8], and joint anomaly detection and localization through flight-data fusion and random channel masking [9]. Reconstruction–prediction co-learning [10] and time-sense autoencoding [11] have also been investigated for UAV flight-data anomaly detection. However, flight data distributions differ substantially across the climb, cruise and descent phases, and existing methods struggle to model the phases in a unified manner. Although the method of Huang et al. models variable relations at different levels within a single model, expert specialization follows the graph hierarchy rather than the flight phases.

Targeting the characteristics of high-altitude UAV flight data, this paper proposes an anomaly detection method based on a Mixture of Experts (MoE). The proposed method introduces expert networks and a gating mechanism to represent data patterns that vary among mission phases. For flight phases with distinct feature distributions, the gating network dynamically assigns expert weights according to the temporal features of the current input, allowing the model to combine expert representations for each window.

To extract diverse state features from multi-phase flight data, we build a Multi-Scale Dual-Path Mixture-of-Experts (MSDP-MoE) encoder. One-dimensional convolutional layers (Conv1D) first capture local short-term perturbation features; Transformer encoder layers then extract long-range dependencies; and a dual-path LSTM fuses fast-varying

dynamics with slow-varying trends. On this basis, the MoE gating network assigns weights to experts according to the latent features of the current window, enabling soft routing across flight phases. At the decision stage, a two-stage reconstruction error produces a window-level anomaly score, and a median absolute deviation (MAD) threshold is computed adaptively, realizing end-to-end anomaly detection for high-altitude UAV flight data.

To verify the detection performance, we compare the proposed method with LSTM-AE, MTGFlow, TranAD, CATCH, DAGMM, DTAAD, and a same-backbone single-expert control using the same temporal data partitions and evaluation metrics. The evaluation includes window-level and event-level detection, leave-one-flight-out testing, expert utilization, and structural ablation. These results were obtained on the controlled anomalies described in Section 4.1.2.

The main contributions are as follows: (1) to address the distribution inconsistency across flight phases, we propose an MoE-based detector whose gating network assigns expert weights from window features, achieving unified modeling of the normal patterns of the climb, cruise and descent phases with a single model and without phase labels; (2) we design the MSDP-MoE encoder, in which multi-scale convolution and a Transformer model local short-term and global long-term dependencies, a dual-path LSTM models and fuses fast/slow dynamics, and expert soft routing strengthens adaptation to phase-level distributional differences; and (3) we compute window-level anomaly scores from two-stage reconstruction errors and combine them with an MAD adaptive threshold for anomaly decisions.

2. Problem Formulation

For a formal description, assume the UAV flight data at time t is $\boldsymbol{D}_t^s$. Here, s is the dimension of the flight data, $D_t^s(i)$ represents the i-th component, i $\in$ N. High-altitude UAV flight sequence data can be expressed as:

$$\mathrm{X} = \left\langle D_{t_0}^s, D_{t_1}^s, D_{t_2}^s, \cdots D_{t_n}^s \right\rangle, \mathrm{n} \in \mathrm{N} \tag{1}$$

$\mathrm{X}(t) = D_t^s$ is the data sample at time t in the flight sequence.

A sliding-window strategy is adopted to analyze the flight sequence. A sliding window of length l is written as:

$$X^l = \langle X(t_1), X(t_2), X(t_3), \cdots X(t_l) \rangle \tag{2}$$

Its matrix is expressed as:

$$X^l = \begin{pmatrix} X(t_1) \\ X(t_2) \\ \vdots \\ X(t_l) \end{pmatrix} \tag{3}$$

The sliding window data matrix $X^l = R^{l \times s}$ is a real number matrix of $l \times s$.

The flight-data anomaly detection problem based on the sliding window can be formulated as finding the anomaly detection function f:

$$\mathrm{y} = f\left(X^l\right) \tag{4}$$

y = 0 means window data X^l is normal; y = 1 means window data X^l is anomalous.

To detect flight-data anomalies, this paper converts the problem of the anomaly detection function f into the anomaly scoring function $g : X^l \to \mathbb{R}$ to solve the problem. The higher the anomaly score $\mathrm{g}\left(X^l\right)$ of the flight data matrix in the sliding window X^l, the

greater the deviation from the normal data distribution. Given an anomaly determination threshold τ, the flight-data anomaly detection function f based on anomaly scores can be expressed as:

$$f(X^l) = \begin{cases} 1, & g\left(X^l\right) > \tau \\ 0, & g\left(X^l\right) \leq \tau \end{cases} \tag{5}$$

3. Our Methods

3.1. Overall Framework

To build a unified model for the data distributions of different high-altitude UAV mission phases, we propose an anomaly detection framework centered on a Mixture of Experts (MoE). As shown in Figure 1, each flight is partitioned before 150-point windows are constructed. Normalization parameters fitted on normal training data are applied to validation and test data, where controlled perturbations are generated separately. The MSDP-MoE encoder extracts multi-scale local perturbation and long-range dependency features from each window, and a gating network assigns weights to multiple experts according to the window features. Next, a two-stage reconstruction module uses the fused expert representation to produce the initial and secondary reconstructed window matrices, from whose reconstruction errors the anomaly score is computed. Finally, an MAD dynamic threshold decides whether the window is anomalous.

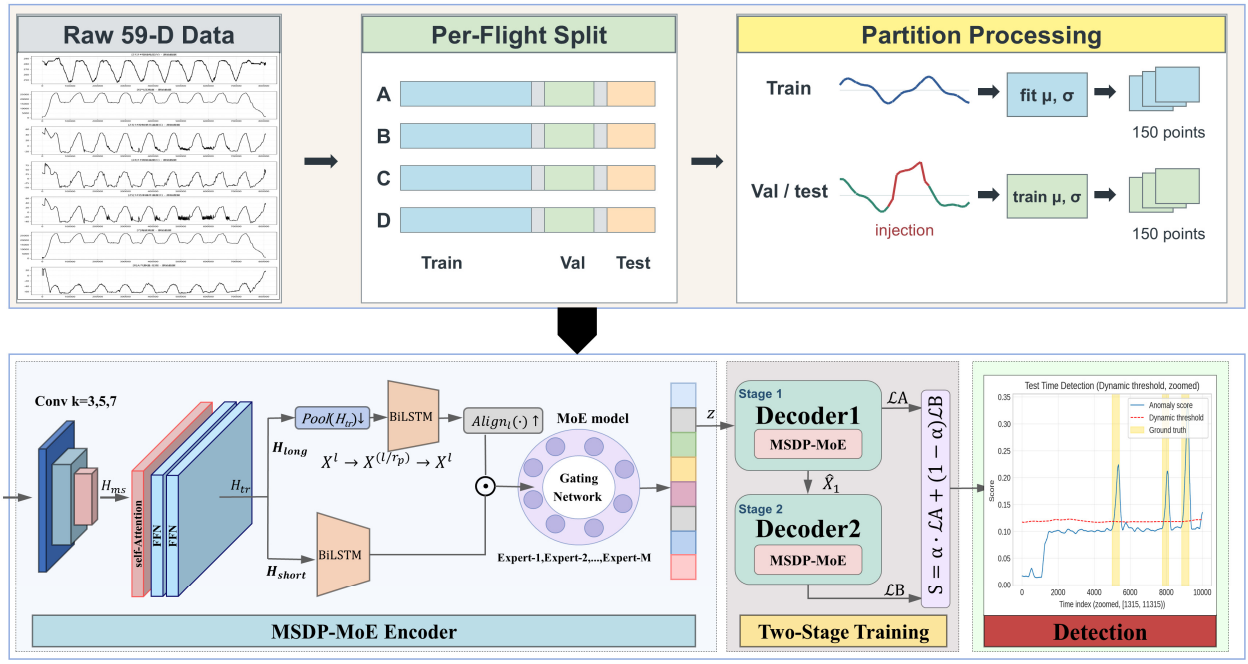


Figure 1. MoE-based flight-data anomaly detection framework. A–D denote Flights A–D. In the per-flight split, blue, green, and orange denote training, validation, and test partitions, respectively; gray bands denote excluded boundary intervals. The red segment in the validation/test trace denotes an injected anomaly.

3.2. Flight Data Preprocessing

During acquisition and transmission, flight data may contain outliers and the dimensions differ greatly in scale and value range, requiring outlier handling and standardization. Each flight is first divided into continuous training, validation, and test blocks before constructing windows, with 75 sampling points excluded on each side of a split boundary. Each 150-point window remains within one block. Channel means and population standard

deviations are fitted on normal training observations and reused for validation and test. Controlled perturbations are injected after partitioning.

This paper uses the median absolute deviation [12] to handle isolated points in flight data. Given a sliding window flight data matrix X^l, $\mu\left(X^l, j\right) = median\left(X^l(:, j)\right)$ is the median of the j-th column of matrix X^l, and the median absolute deviation of the j-th column is MAD(X^l, j):

$$MAD\left(X^l, j\right) = median\left(\left|X^l(:, j) - \mu\left(X^l, j\right)\right|\right) \tag{6}$$

For $\forall x_j \in X^l(:, j)$, if the condition $\left|x_j - \mu\left(X^l, j\right)\right| > \gamma \cdot \text{MAD}\left(X^l, j\right)$ is met, then the entry is treated as an outlier. The deviation coefficient controls the strictness of outlier identification: the larger its value, the fewer entries are flagged.

The index set for $M = X^l(:, j)$ is $I = \{1, 2, \cdots, l\}$, and the index $p \in I$ corresponds one-to-one to the sampling time in the sliding window. I_o is the index set identified as an isolated point according to the above criteria:

$$I_o = \forall p \in I, \left|M_p - \mu(M)\right| > \gamma \cdot MAD(M) \tag{7}$$

$I_v = I \setminus I_o$ is the set of indexes for valid data in M.

Given an isolated point index $o \in I_o$, the ρ-nearest valid data index set of o is NNρ(o):

$$NN_\rho(o) = \{S \subseteq I_v : |S| = \rho \wedge \forall u \in S, v \in I_v \setminus S \Rightarrow |u - o| \leq |v - o|\} \tag{8}$$

When $|I_v| < \rho$, let $NN_\rho(o) = I_v$, thus retrieving all valid data. If multiple equidistant indexes exist, the one with the smaller index is selected first to ensure uniqueness.

During data preprocessing, for any data index $p \in I$, if it is an isolated point, it is replaced with the median of the ρ-nearest valid dataset; otherwise, it remains unchanged:

$$M_p \to median\left\{M_u \mid u \in NN_\rho(p)\right\},\ p \in I_o\ M_p \to M_p,\ p \in I_v \tag{9}$$

Here, $a \to b$ represents replacing data a with data b, and ρ is used to control the number of nearest valid data points involved in replacement. For each dimensional column of each sliding window data matrix of the UAV flight sequence data, the above replacement is performed to obtain the flight data sequence after isolated-point preprocessing.

To eliminate differences between dimensions and numerical ranges in each dimension, Z-score normalization is performed on flight sequence data. $\varphi\left(X^l, j\right)$ and $\sigma\left(X^l, j\right)$ are assumed to be the mean and population standard deviation of channel j estimated only from normal training observations, rather than from the current window X^l, respectively. The data $X^l(i, j), i = 1 \cdots l, j = 1 \cdots s$ are standardized as follows:

$$X^l(i, j) = \frac{X^l(i, j) - \varphi\left(X^l, j\right)}{\sigma\left(X^l, j\right)}, \sigma\left(X^l, j\right) > 0 \tag{10}$$

3.3. MoE-Based Detection Model

To detect anomalies across different flight phases, we propose the MoE-based detection model shown in Figure 2, which consists of feature encoding, two-stage reconstruction and adaptive-threshold decision modules. For an input sliding window, the feature encoder maps it to a low-dimensional latent feature vector; the two-stage reconstruction module then generates an initial reconstruction and a secondary reconstruction, whose errors form the anomaly score; finally, the MAD dynamic threshold produces the decision.

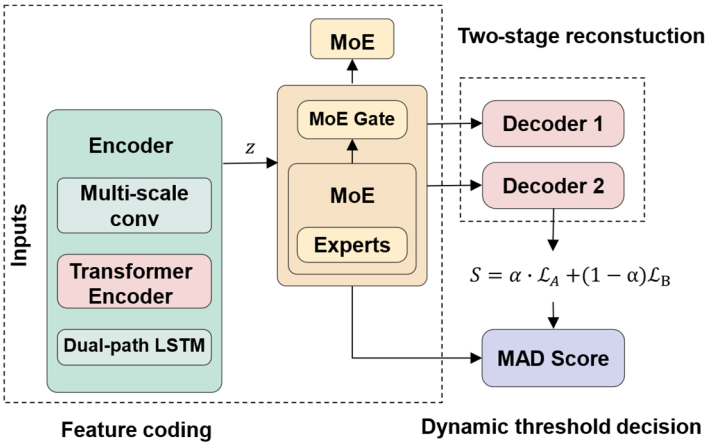


Figure 2. Architecture of the MoE-based UAV anomaly detection model.

3.3.1. Multi-Scale Temporal Feature Encoder

High-altitude UAV flight data contain both local perturbations and long-range dependencies. We therefore design the Multi-Scale Dual-Path Mixture-of-Experts (MSDP-MoE) encoder shown in Figure 3, which comprises multi-scale convolution, Transformer encoder layers, a dual-path LSTM and an MoE module. Multi-scale convolution captures local short-term variations, the Transformer layers model global long-range dependencies, and the dual-path LSTM fuses local short-term features with global trend features. The MoE module uses multiple parameter-independent experts—each a two-layer feed-forward network with the same structure but independent parameters—to learn different latent representations; a gating network then performs soft routing according to the sliding-window features so as to adapt to the distributions of the climb, cruise and descent phases.

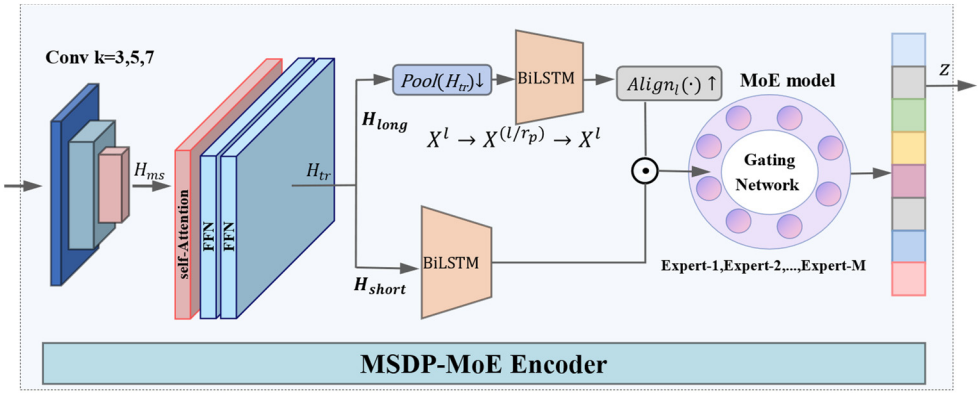


Figure 3. Structure of the MSDP-MoE encoder.

Multi-scale convolution is used to extract local disturbances over different time ranges. The convolutional kernel size set used in this paper is $\mathcal{K} = \{3,5,7\}$. For the sliding window data matrix X^l, the convolutional feature of scale $\kappa \in \mathcal{K}$ is:

$$H^{(k)} = GELU\left(Conv1D_\kappa\left(X^l\right)\right) \in \mathbb{R}^{l\times s}, \kappa \in \mathcal{K} \tag{11}$$

Here, κ represents the size of the convolutional kernel in the time dimension, $GELU(\cdot)$ is the activation function, and $H^{(k)}$ is the local feature obtained from this scale. Features of multiple scales are concatenated along the last dimension, and then linear mapping yields multi-scale representations:

$$H_{ms} = Linear_{ms}\left(\left[H^{(3)}\middle|H^{(5)}\middle|H^{(7)}\right]\right) \in \mathbb{R}^{l\times s} \tag{12}$$

where $\left[H^{(3)}, H^{(5)}, H^{(7)}\right]$ represents the concatenation of features at three scales along the last dimension, and $Linear_{\mathrm{ms}}(\cdot)$ remaps the concatenated features to s-dimension. Then, the Transformer encoder layer extracts long-term dependencies:

$$H_{tr} = TransformerEncoder(H_{ms}) \in \mathbb{R}^{l \times s} \tag{13}$$

Next, a dual-path LSTM is used to fuse local short-term features with global trend features. As shown in Figure 3, the dual-path LSTM includes both $\mathrm{H}_{\mathrm{long}}$ and $\mathrm{H}_{\mathrm{short}}$ branches. The $\mathrm{H}_{\mathrm{short}}$ branch directly processes the Transformer output sequence with length l, and does not undersample, thus preserving the variation at each sampling time; the $\mathrm{H}_{\mathrm{long}}$ branch groups adjacent sampling points by stride length r_p, averages them, compresses the sequence length to about l/r_p, and then uses bidirectional LSTM to extract slow changes. To facilitate moment-by-moment fusion of the two branches, linear interpolation is used to restore the time length of the $\mathrm{H}_{\mathrm{long}}$ branch to l:

$$\begin{aligned} H_{short} &= BiLSTM_{short}(H_{tr}) \in \mathbb{R}^{l \times d} \\ H_{long} &= Align_l\Big(BiLSTM_{long}\Big(AvgPool_{r_p}(H_{tr})\Big)\Big) \in \mathbb{R}^{l \times d} \end{aligned} \tag{14}$$

where d is the dual-path hidden feature dimension, and r_p is the downsampling stride of the $\mathrm{H}_{\mathrm{long}}$ branch; $Align_l(\cdot)$ represents linear interpolation in the time dimension, restoring the $\mathrm{H}_{\mathrm{long}}$ output of about l/r_p after downsampling to l. Then, two branches of equal length are fused elementwise through a learned gate:

$$\begin{aligned} G &= \sigma\Big(\Big[H_{short} + H_{long}\Big]W_g + b_g\Big) \in \mathbb{R}^{l \times d} \\ H_{dp} &= G \odot H_{short} + (\mathbf{1} - G) \odot H_{long} \in \mathbb{R}^{l \times d} \end{aligned} \tag{15}$$

where $\sigma(\cdot)$ is the Sigmoid function, $\odot$ represents elementwise multiplication, W_g and b_g are learnable parameters, and $\mathbf{1}$ is an all-ones matrix of the same shape as the gated matrix. Averaging the fusion features over time dimensions yields window-level feature vectors:

$$F = \frac{1}{l}\sum_{\ell=1}^{l} H_{dp}(\ell, :) \in \mathbb{R}^{d} \tag{16}$$

The MoE module consists of M parallel expert networks and one gating network. The gating network calculates the expert weights of the sliding window data matrix X^l based on the feature vector F, and normalizes the weights using the Softmax function:

$$w = Softmax(W_{gate}F + b_{gate}) = \left[w_1, w_2, \ldots, w_M\right]^{\mathrm{T}} \tag{17}$$

Here, $w_m \geq 0$ and $\sum_{m=1}^{M} w_m = 1$. $m = 1, 2, \ldots, M$ is the expert index, and w_m represents the weight of the m-th expert for the current window data matrix. $W_{\mathrm{gate}} \in \mathbb{R}^{M \times d}$ and $b_{\mathrm{gate}} \in \mathbb{R}^{M}$ are learnable parameters of the gated network.

Let the latent feature dimension be k, let $f_m(\cdot)$ represent the m-th expert network, and let z_m be the latent feature output by the m-th expert. Then:

$$z_m = f_m(F) \in \mathbb{R}^{k} \tag{18}$$

Each expert applies a non-linear mapping to the window feature F, producing different candidate latent representations. The model fuses the expert outputs according to the gating weights, obtaining the multi-expert fused feature z:

$$z = \sum_{m=1}^{M} w_m z_m \in \mathbb{R}^{k} \tag{19}$$

The MoE mechanism dynamically adjusts expert contributions according to each input window. Three experts gave the highest mean validation endpoint F1 among the one-to-five-expert configurations in Table A2, while retaining a compact model with multiple latent representations. Section 4.5 evaluates this choice against same-backbone and parameter-matched single-expert controls.

ForouzeshNejad and Gegov [13] include named domain concepts in their diagnostic model. MSDP-MoE instead learns expert representations through reconstruction; the gate weights their contributions for each input window. No expert is assigned a climb, cruise, or descent label during training. Section 4.4 reports how the experts are used. Testing whether an expert consistently responds to a particular flight phase would require comparing its gate weights with independently labeled phase intervals.

3.3.2. Anomaly Scoring via Two-Stage Reconstruction

To calculate the anomaly score of the sliding window data matrix, this paper adopts a two-stage progressive reconstruction mechanism. Let the MSDP-MoE encoder be $\mathcal{E}(\cdot)$, and the encoding of the sliding window data matrix be X^l is $\mathrm{z} = \mathcal{E}\left(X^l\right)$. Decoder $\mathcal{D}_1(\cdot)$ takes $\mathcal{E}\left(X^l\right)$ as input and generates the initial reconstruction $\hat{X}^{(1)}$. The initial reconstruction $\mathcal{E}(\cdot)$ is re-encoded to obtain $\mathcal{E}(\hat{X}^{(1)})$, which is passed to decoder $\mathcal{D}_2(\cdot)$ to generate the secondary reconstruction $\hat{X}^{(2)}$:

$$\begin{aligned} \hat{X}^{(1)} &= \mathcal{D}_1(z) \in \mathbb{R}^{l \times s} \\ \hat{X}^{(2)} &= \mathcal{D}_2\left(\mathcal{E}\left(\hat{X}^{(1)}\right)\right) \in \mathbb{R}^{l \times s} \end{aligned} \tag{20}$$

Each reconstruction error is averaged over all elements of a window with l sampling points and s channels:

$$e^{(i)} = \frac{1}{ls}\left\| X^l - \hat{X}^{(i)} \right\|_F^2, i \in \{1,2\} \tag{21}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. For the evaluated window shape, Equation (21) averages over all 150×59 residual elements. Based on the two elementwise reconstruction MSEs, the anomaly score of the sliding-window matrix X^l is computed as follows:

$$g\left(X^l\right) = \beta e^{(1)} + (1-\beta)e^{(2)}, \beta \in [0,1] \tag{22}$$

β is the reconstruction-error factor that balances the contributions of the initial and secondary reconstruction errors. For normal windows, both reconstructions are usually close to the input; when window features deviate from the normal patterns learned during training, both reconstruction errors and the weighted anomaly score increase.

3.3.3. MAD-Based Adaptive Thresholding

To determine abnormality in UAV flight data based on anomaly scores, an abnormality threshold τ must be calculated. Considering that anomaly scores may fluctuate significantly at different flight stages, this paper uses MAD to describe the dispersion degree of abnormal scores [12] and calculate adaptive thresholds.

Suppose the sliding window step size is 1 and the window length is l. The flight data matrix for the i-th sliding window is X_i^l, with an anomaly score $\mathrm{g}\left(X_i^l\right)$. The set of w-backward window anomaly scores, $\overleftarrow{\mathrm{G}}\left(X_i^l, w\right)$ contains the prior scores defined below. Only prior scores are included: $\mathrm{i} - \mathrm{w} \leq \mathrm{q} < \mathrm{i}$. The current score is tested before it is added

to the history. Threshold initialization must use previously available reference scores, not future test observations.

$$\overleftarrow{G}\left(X_i^l, w\right) = \left\{ g\left(X_q^l\right) \;\middle|\; q \in N_+ \wedge q \in [i-w, i) \right\} \tag{23}$$

The threshold history is initialized from normal training-reference scores. Before the rolling history reaches its configured length, the median and MAD use the available reference and preceding online scores. Each new score is evaluated first and appended afterward. A small positive floor prevents the reference dispersion from vanishing.

The adaptive anomaly threshold for the i-th window is computed as follows:

$$\tau(i) = \text{median}\left(\overleftarrow{\text{G}}\left(X_i^l, w\right)\right) + \alpha \cdot MAD\left(\overleftarrow{\text{G}}\left(X_i^l, w\right)\right) \tag{24}$$

Here, $\alpha > 0$ is the threshold amplification factor, used to control the increase in the threshold relative to the dispersion degree of anomaly scores in the backward window. Given the i-th sliding window, whose window data matrix is X_i^l, it has:

$$\begin{cases} \text{If } \text{g}\left(X_i^l\right) > \tau(i), \text{ window } i \text{ is anomalous} \\ \text{If } \text{g}\left(X_i^l\right) \leq \tau(i), \text{ window } i \text{ is normal} \end{cases} \tag{25}$$

4. Experiments and Results

4.1. Dataset

4.1.1. Data Source and Composition

Experiments use preprocessed data from four real flights of a certain type of high-altitude UAV. The data record of one flight is defined as a sortie, and the row of records output at one timestamp is defined as a sampling point. Each sampling point consists of 59 continuous numerical channels grouped into five data categories, as detailed in Table 1. Anomaly injections, sample construction and model evaluation are all performed on the preprocessed data. These sampling points are nested within four sorties and are not independent flight-level observations.

Table 1. Classification of UAV flight data channels.

No.	Category	Data Channels
1	Propulsion system	Motor speed, commanded speed, No. 1–4 motor bus voltage, No. 1–4 motor bus current, No. 1–4 motor resolved torque, No. 1–4 motor D/Q-axis current, No. 1–4 motor rotor position
2	Electrical system	No. 1–8 single-cell battery voltage, power-supply system state
3	Thermal management and structural health	No. 1–4 motor control unit temperature, No. 1–4 motor winding temperature, No. 1–4 motor power-device temperature
4	Navigation and flight control	Indicated airspeed, true airspeed, ground speed
5	Atmospheric and environmental parameters	Barometric altitude, GPS altitude, atmospheric temperature

The four sorties are denoted Flight A, Flight B, Flight C and Flight D, with 84,819, 78,752, 20,693 and 806,629 sampling points, respectively. To describe the flight composition, flight-phase labels are generated. Following the altitude- and vertical-velocity-based phase identification approach of Sun et al. [14], a 31-point median filter is first applied to the altitude sequence; the altitude rate between adjacent samples is then computed and smoothed with a 61-point moving average. Samples with a smoothed rate above 0.30 m/s

are labeled as climb, those below −0.30 m/s as descent, and the remainder as cruise. Phase labels are not used as model input; the phase proportions of each sortie are given in Table 2.

Table 2. Sampling-point composition and flight-phase proportions for each sortie.

Sortie	Sampling Points	Climb (%)	Cruise (%)	Descent (%)
Flight A	84,819	38.18	28.83	32.99
Flight B	78,752	37.92	22.01	40.06
Flight C	20,693	5.95	89.83	4.22
Flight D	806,629	23.06	56.43	20.50
Total	990,893	25.18	52.03	22.79

The median interval between consecutive recorded timestamps in the retained continuous segments is approximately one second. Detection delays and the observed flight durations used for hourly false-alarm rates are calculated directly from these timestamps.

4.1.2. Anomaly Injection and Labeling

The number of naturally labeled faults in the flight records is limited. Therefore, following the anomaly construction methods in [15,16], controlled anomalies are injected into copies of normal validation and test sequences while the training sequences remain unchanged. UAV specialists helped select the affected channels and discuss plausible magnitudes and durations. The four injection types are step changes, which simulate sustained offsets; spikes, which simulate short abrupt changes; drift, which simulates gradually increasing offsets; and correlation changes, which disrupt relationships among selected variables. The injected intervals contain 121–416 sampling points and affect one to four variables, including bus voltage and current, torque, D/Q-axis currents, temperature, altitude, and atmospheric temperature. At the approximately one-second median sampling interval, these intervals correspond to roughly two to seven minutes. Figure 4 illustrates three of the four types; Section 4.6 reports detection results for all four.

The perturbations were based on abnormal telemetry behaviors encountered in UAV operation and were reviewed with UAV specialists for affected channels, magnitudes and durations. In their operational judgment, the four anomaly types correspond to recurring operational behaviors: a step represents a sustained sensor offset, a spike a brief transient excursion, a drift a gradual sensor bias or calibration change, and a correlation-structure change an altered relationship among channels. Together, these patterns cover most of the recurring signal-level anomaly behaviors relevant to the flights considered here. This is a qualitative assessment of observable patterns, not a measured frequency of fault types. Controlled injections cannot reproduce every interaction, flight condition or naturally occurring failure; detection performance on independently annotated operational faults remains to be established.

Abrupt-change anomalies are used to simulate sensor jumps, spikes, or short-term control disturbances. Let $\mathcal{B}$ be the interval corresponding to the anomaly injection, $\mathcal{C}$ be the set of injected dimensions, and $\mathcal{P} \subseteq \mathcal{B}$ be the set of actual applied mutations. After injection, the data for the j-th dimension at time t is:

$$\widetilde{D}_t(j) = D_t(j) + \xi\delta_j \cdot 1_{\{t\in\mathcal{P}\}}$$

Here, ξ is the global disturbance intensity coefficient; δ_j is the disturbance amplitude corresponding to the j-th dimension, and $\mathbf{1}_{\{t\in\mathcal{P}\}}$ is the indicator function: when $t \in \mathcal{P}$ is set, it is taken as 1; otherwise, it is 0. When $\mathcal{P} = \mathcal{B}$, interval steps occur; when $\mathcal{P}$ contains only a few discrete sampling points, a spike is formed. Figure 4a shows an example of a step perturbation.

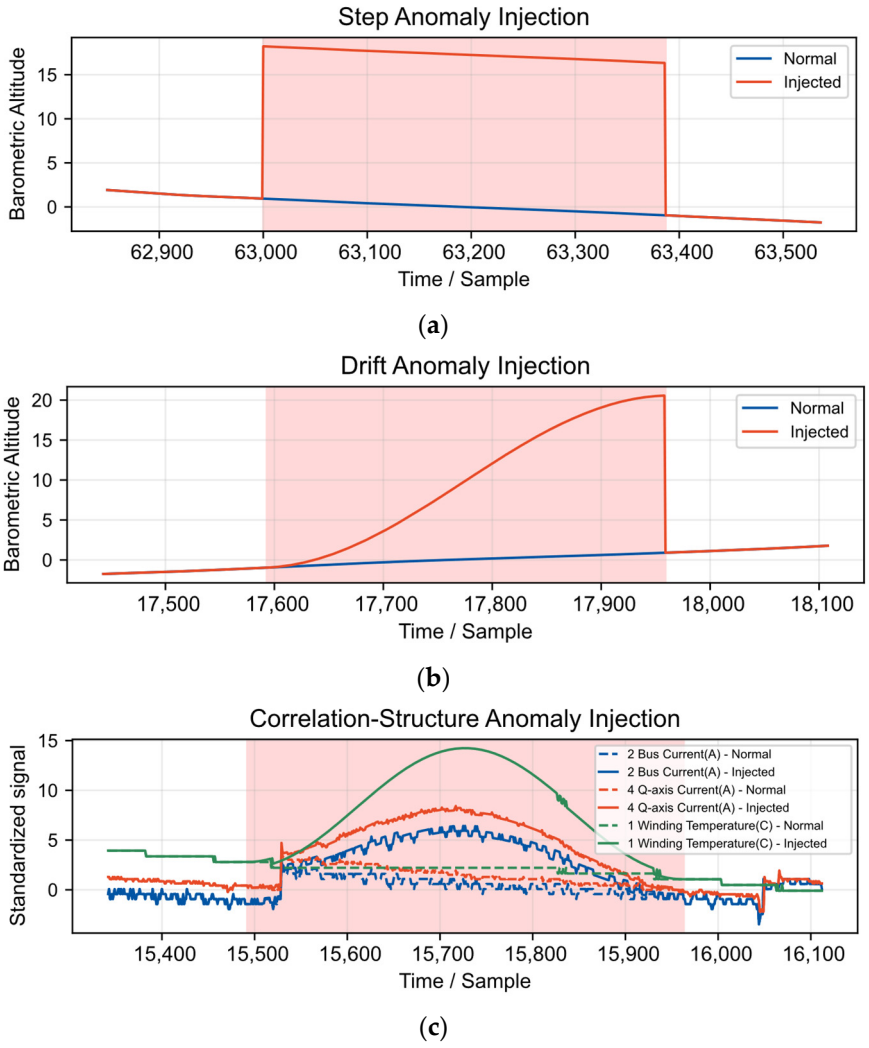


Figure 4. Examples of injected anomalies: (**a**) step perturbation; (**b**) drift perturbation; (**c**) correlation-structure perturbation. Red shading marks injection intervals. In (**a**,**b**), blue/orange denote normal/injected signals. In (**c**), colors identify channels 2, 4, and 1; dashed/solid lines denote normal/injected signals, standardized and vertically offset.

Drift-type anomalies are used to simulate sensor aging, calibration offset, or accumulation of environmental interference. Within interval $\mathcal{B}$, it grows linearly from 0 to $\xi\delta_j$, with the start and end indices of interval $\mathcal{B}$ set as t_s, t_e, respectively. For j-th dimensional data, the value at time t after injection is:

$$\widetilde{D}_t(j) = D_t(j) + \xi\delta_j \frac{t - t_s}{t_e - t_s}, t \in \mathcal{B}, j \in \mathcal{C}.$$

Figure 4b shows an example of a drift anomaly. Compared with step anomalies, this type evolves continuously over the injection interval and tests the model's ability to detect gradual deviations.

Correlation-structure anomalies act on multiple dimensions simultaneously to simulate the breaking of the original variable coupling relationships. Let $\boldsymbol{d}_{t,\mathcal{C}}$ be the normal data subvector corresponding to the dimension set $\mathcal{C}$ at time t. The subvector after injecting the anomaly is denoted as $\widetilde{\boldsymbol{d}}_{t,\mathcal{C}}$. The injection process is:

$$\widetilde{\boldsymbol{d}}_{t,\mathcal{C}} = \boldsymbol{A}_{\mathcal{C}}\boldsymbol{d}_{t,\mathcal{C}} + \boldsymbol{\delta}_{k,\mathcal{C}}, A_{\mathcal{C}} \neq I, t \in \mathcal{B}.$$

Here, A_C is the non-identity transformation matrix acting on the affected dimension, $\delta_{k,C}$ is the additional perturbation vector, and I is the identity matrix. This operation changes the joint change relationships between affected dimensions without requiring each dimension to experience separate overruns. Figure 4c provides a corresponding example.

In the test set, 162 injections changed at least one telemetry value. We treat these as 162 anomaly events when calculating event recall, while the endpoint label count in Table A11 counts anomalous newest-sample decisions. Validation and test injections are generated separately after the time-series split, so an injected event cannot cross between the two sets.

We move each 150-point window forward by one sample in training and evaluation. Adjacent windows therefore share 149 samples, and their total count is much larger than the number of independent time periods. During monitoring, the most recent window gives one decision for each new sample; for training, windows are drawn from the overlapping set using the balanced sampling procedure in Appendix E. Table A10 also lists how many nonoverlapping windows would fit in the retained data as a guide to scale, not as an estimate of statistical independence. A window receives a positive label if it contains any injected sample. By contrast, the endpoint label records whether the newest sample is anomalous. Appendix B tests stricter window labels and less frequent output evaluation; Section 4.4 reports held-out-flight results, and Table A5 reports event counts and delays.

Each flight is first divided into continuous training, validation, and test time blocks. We leave 75 samples unused on both sides of every boundary, then construct 150-sample windows entirely within each block. Means and standard deviations are calculated from normal training data only and then applied to the other blocks. Validation and test anomalies are injected separately within their own blocks, so the same injected event cannot appear in two sets. Model and threshold settings are chosen on validation data before the test set is evaluated. Table A10 gives the numbers of samples and windows retained after the boundaries and continuity checks; Section 4.3 presents the detection results.

Step, drift, spike, and correlation perturbations are stratified by amplitude and duration using fixed layout seeds. Both nominal injection labels and labels based on actual numerical changes are retained. Attempts producing no change on constant backgrounds remain in the task record; effective-event recall uses the events with numerical changes.

4.2. Experimental Setup

The models were implemented in Python 3.10.18 and PyTorch 2.5.1 (CUDA 12.1) and evaluated on an NVIDIA A100 GPU server. The blocked comparison applies the same raw-time partitions, perturbation tasks, labels, and metrics to all methods in the comparative evaluation. The one-expert control changes the number of experts in the MSDP-MoE code. For the six published baselines, available author-released code was used; where no such release was available, the implementation followed the published method. TranAD uses its released model implementation with a 150-point input adapter, training-fitted min–max scaling, and its native second-output endpoint score. CATCH uses the public benchmark implementation with its frequency-domain training and scaling. Each model is trained with seeds 42, 2026, and 3407; the checkpoint with the lowest normal-validation loss is retained for each run. Score thresholds are chosen on validation data using the same candidate MAD histories and multipliers, then held fixed for testing. The reconstruction weight applies only to the two-error MSDP-MoE score. Model-specific optimizer settings are retained rather than forcing different architectures to use the same loss. Window F1 uses the label of the full window; endpoint F1 is sample-level F1 at the newest sample in each window. Event recall counts events with an alarm on a changed sample, and normal alarm percentage is the proportion of normal endpoints flagged.

Training uses Adam, mean squared error loss, and a cosine-annealing learning-rate schedule. In Stage 1, the encoder and first decoder (D1) are trained together to reconstruct normal input windows. We retain the checkpoint with the lowest validation reconstruction loss. In Stage 2, that encoder and D1 checkpoint are frozen, and only the second decoder (D2) is trained to reconstruct the same windows. Stage 1 runs for at most 100 epochs and Stage 2 for at most 50. At test time, both reconstruction errors contribute to the anomaly score. We repeat both stages independently with seeds 42, 2026, and 3407, retaining a checkpoint, log, sampled windows, and reference residuals for each seed. The expert count, temporal pooling factor, reconstruction weight, MAD history, and threshold multiplier are selected by validation grid search over the candidates in Table A9 and fixed before testing. Table 3 gives the selected settings; Table A2 shows one-factor sensitivity around those settings.

Table 3. Model architecture and settings selected from the validation search.

Parameter	Symbol	Value	Description and Setting Rationale
Flight-data dimension	s	59	Continuous telemetry channels retained in the model input.
Dual-path LSTM output dimension	d	176	Two bidirectional paths, each with 88 hidden units per direction.
MoE latent dimension	k	64	Latent representation produced by each expert.
Long-path pooling stride	r_p	4	Selected from pooling factors 2, 4, and 8 in Table A2.
Number of experts	M	3	Highest validation endpoint F1 among one to five experts in Table A2.
First-reconstruction score weight	β	0.5	Selected on the validation tasks from 0.1, 0.5, and 0.9.
MAD history	w	5000	Selected on validation from 1000, 5000, and 10,000 prior scores.
MAD threshold multiplier	α	4.4478	Selected on validation from the candidate grid in Table A9.

The structural study includes the full three-expert model, variants without multi-scale convolution or the Transformer, a single-path encoder, two single-expert controls, and a matched variant without the Stage 2 decoder. Each configuration follows its specified reconstruction protocol; all configurations except the no-Stage-2 variant train both reconstruction stages. Task metrics are aggregated within each seed before calculating the three-seed mean and sample standard deviation. The main comparison pools window predictions within each seed. The structural and sensitivity analyses average task-level metrics within each seed, giving evaluation tasks equal weight; these task averages need not equal the pooled scores in Table 4.

Table 4. Detection performance under the common blocked protocol (mean ± sample SD).

Method	Window F1	Endpoint F1	Event Recall	Normal Alarm (%)
LSTM-AE	0.8549 ± 0.0071	0.8517 ± 0.0074	0.8189 ± 0.0094	3.88 ± 0.24
MTGFlow	0.8612 ± 0.0064	0.8577 ± 0.0067	0.8066 ± 0.0128	4.18 ± 0.27
TranAD	0.8906 ± 0.0039	0.8874 ± 0.0044	0.8621 ± 0.0094	2.92 ± 0.12
CATCH	0.9195 ± 0.0035	0.9167 ± 0.0039	0.9053 ± 0.0036	3.08 ± 0.11
DAGMM	0.8684 ± 0.0058	0.8648 ± 0.0061	0.8395 ± 0.0123	3.71 ± 0.22
DTAAD	0.8405 ± 0.0069	0.8370 ± 0.0072	0.8045 ± 0.0143	4.57 ± 0.29
Single expert	0.9108 ± 0.0042	0.9061 ± 0.0046	0.8930 ± 0.0094	3.35 ± 0.20
MSDP-MoE	0.9397 ± 0.0046	0.9364 ± 0.0048	0.9177 ± 0.0036	2.47 ± 0.16

For window F1, a window is positive if at least one of its 150 samples was changed by an injection. Endpoint F1 instead evaluates the decision for the newest sample in each window. An event is detected if an alarm occurs at a sample that actually changed during its injection interval. Table A5 gives the number of detected and missed events and the time from the start of each event to its first qualifying alarm. For false alarms, Table A6 reports the percentage of normal samples assigned an alarm and the number of new false-alarm episodes per hour of recorded normal flight. The first measure describes how often normal samples are flagged; the second counts how often a new false alarm begins. Both hourly rates and detection delays use the recorded timestamps.

When a window spans two flight phases, we label it by its last sample, the sample at which the detector makes its decision. The motion groups used to balance normal training examples are a separate sampling rule and do not change this phase label. The group uses altitude change over the preceding 60 observations divided by their timestamp difference, with thresholds of +0.3 and −0.3 m/s. Intermediate slopes form a level-motion group, and insufficient or invalid history is retained as unknown. This endpoint rule assigns transition windows consistently; Appendix E records the data and implementation settings.

4.3. Comparative Experiments

We compare MSDP-MoE with LSTM-AE [17], MTGFlow [18], TranAD [19], CATCH [20], DAGMM [21], DTAAD [22], and a same-backbone single-expert control. The methods represent recurrent reconstruction, graph/flow modeling, Transformer-based reconstruction, frequency-domain channel modelling, mixture-density modeling, and temporal convolution with attention. The comparison follows the temporal partitions, injection tasks, label definitions, and validation-selection procedure in Sections 4.1 and 4.2.

Table 4 and Figure 5 report the metrics defined in Section 4.2 as means and sample standard deviations across training seeds 42, 2026, and 3407. Because neighboring windows overlap, window F1 is interpreted alongside event recall and the held-out-flight results in Table 5. The standard deviations describe variation across training runs.

Table 5. Leave-one-flight-out evaluation (three-seed mean ± sample SD).

Held-Out Flight	Window F1	ROC-AUC	Normal Alarm (%)
A	0.9021 ± 0.0084	0.9491 ± 0.0062	3.18 ± 0.31
B	0.8954 ± 0.0091	0.9443 ± 0.0070	3.57 ± 0.36
C	0.9108 ± 0.0076	0.9548 ± 0.0055	2.84 ± 0.27
D	0.8837 ± 0.0102	0.9382 ± 0.0078	4.12 ± 0.41
Equal-flight macro	0.8980 ± 0.0049	0.9466 ± 0.0038	3.43 ± 0.19

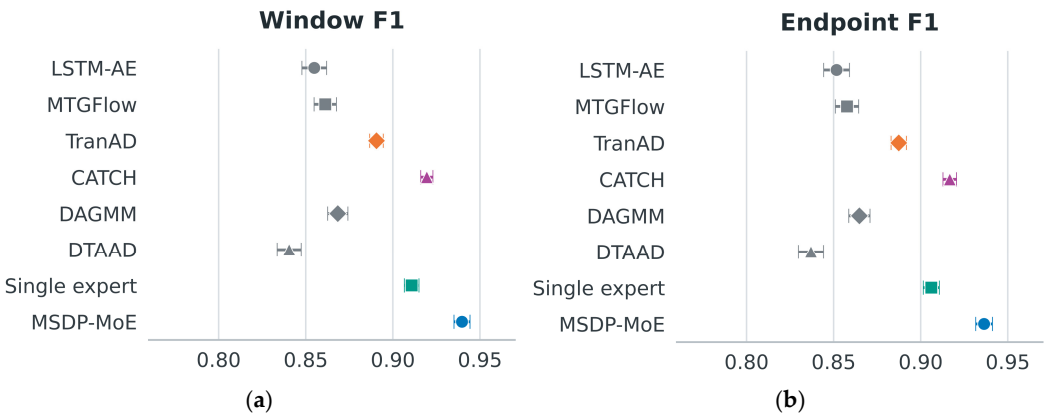


Figure 5. *Cont.*

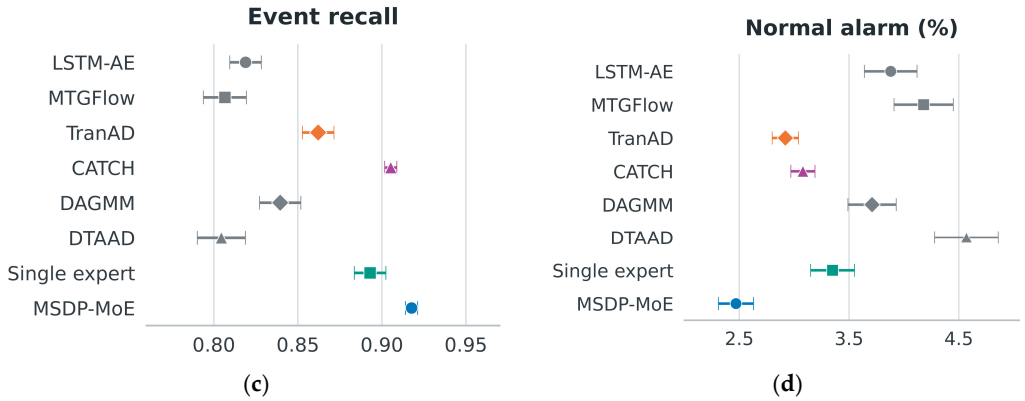


Figure 5. Detection metrics for the eight compared methods: (**a**) window F1; (**b**) endpoint F1; (**c**) event recall; (**d**) normal alarm percentage. Colors and marker shapes distinguish the methods identified by the corresponding vertical-axis labels. Marker positions indicate means, and error bars indicate sample standard deviations over three training seeds.

MSDP-MoE achieves mean window F1 of 0.9397, endpoint F1 of 0.9364, and event recall of 0.9177, the highest values in Table 4. Its normal alarm percentage is 2.47%, compared with 3.35% for the single-expert control, 2.92% for TranAD, and 3.08% for CATCH. Figure 5 displays the same eight-method comparison with between-seed variability.

MSDP-MoE detects 148–149 of the 162 effective test events in each run. Table A5 reports detected and missed events together with first-alarm delay, while Table A6 reports normal alarms by flight. Tables A3 and A4 examine window-label occupancy and output subsampling. These results complement the aggregate comparison without treating overlapping windows as independent flights.

4.4. Flight-Level Evaluation and Expert Routing

Table 5 tests each flight in turn after training and selecting the threshold on the other three flights. The test flight contributes no data to training, normalization, or threshold selection. We then average the four flight scores equally; this prevents the much longer Flight D from determining most of the summary score. Table A6 separately gives normal alarm statistics for each flight in the within-flight evaluation.

Window F1 ranges from 0.8837 to 0.9108 across the held-out flights, with an equal-flight macro value of 0.8980. The reduction from the within-flight result in Table 4 indicates the additional variation introduced by testing on a flight excluded from training.

Flight D contributes approximately 81% of the sampling points, so pooled within-flight metrics are strongly influenced by that sortie. The equal-flight macro average in Table 5 gives the four held-out flights the same weight, but each fold is trained on only three flights, with uneven coverage of mission conditions. Thus, the leave-one-flight-out experiment tests transfer among these four sorties; it does not establish performance on other UAV platforms, mission profiles or operating environments. More independent flights and naturally labeled faults are needed for broader validation.

Table 6 summarizes 364,134 normal-window evaluations across the three seeds. For each window, the expert with the largest gate weight receives one hard assignment. The resulting shares are 35.23%, 34.02%, and 30.75%, which sum to 100%. Even the least-used expert in any run receives 4.92% of its windows, above the 1% utilization check. Normalized gate entropy ranges from 0.8173 to 0.8483; values near 1 mean that weights are spread across experts. Thus, all three experts participate, often with shared weights. Expert 1 in one training run need not have the same role as Expert 1 in another, so the aggregate shares alone do not establish stable flight-phase roles. Appendix C gives each seed's average weights and the metric definitions. These diagnostics indicate broad expert

participation, but because the windows are pooled across all flight phases, they do not show whether routing changes systematically with phase. We did not compare gate-weight distributions conditioned on climb, cruise and descent or align expert identities across seeds. Accordingly, the current results do not establish phase-specific expert roles; testing this association would require phase-stratified gate weights and traces around phase transitions.

Table 6. Expert utilization on normal windows; columns retain the raw expert indices.

Seed	Expert 1 (%)	Expert 2 (%)	Expert 3 (%)	Normalized Entropy
Aggregate	35.23	34.02	30.75	0.8303
42	41.77	37.86	20.37	0.8252
2026	10.17	22.86	66.97	0.8173
3407	53.76	41.33	4.92	0.8483

4.5. Structural Ablation and Threshold Analysis

To analyze the core designs, Table 7 compares the full model with variants without multi-scale convolution, without the Transformer, with a single temporal path, and with one expert. The two single-expert controls use the same backbone or a parameter-matched configuration. Each configuration is trained independently for the three seeds using the partition and validation-selection rules in Section 4.2. We also include a matched variant without the Stage 2 decoder; its encoder and first decoder are trained and scored under the same protocol.

Table 7. Structural ablation using task-averaged metrics (mean ± sample SD across three training seeds).

Configuration	F1-Score	ROC-AUC	PR-AUC
Full three-expert model	0.9410 ± 0.0074	0.9704 ± 0.0018	0.9502 ± 0.0059
Without multi-scale convolution	0.9212 ± 0.0036	0.9630 ± 0.0026	0.9284 ± 0.0022
Without Transformer	0.9097 ± 0.0009	0.9580 ± 0.0035	0.9266 ± 0.0037
Single-path encoder	0.9009 ± 0.0026	0.9588 ± 0.0015	0.9230 ± 0.0045
Same-backbone single expert	0.9059 ± 0.0003	0.9629 ± 0.0019	0.9281 ± 0.0021
Parameter-matched single expert	0.9111 ± 0.0035	0.9628 ± 0.0008	0.9269 ± 0.0050
Without Stage 2 decoder	0.9238 ± 0.0051	0.9651 ± 0.0025	0.9390 ± 0.0034

The full three-expert configuration has the highest mean F1 in this structural comparison (0.9410). Removing multi-scale convolution, the Transformer, or the second temporal path gives 0.9212, 0.9097, and 0.9009, respectively. The same-backbone and parameter-matched single-expert variants give 0.9059 and 0.9111, while omitting the Stage 2 decoder gives 0.9238. Figure 6 shows the F1 differences and their between-seed variability. The no-Stage-2 row uses only the first reconstruction; the other listed configurations use the two-stage score. The parameter-matched single-expert control has comparable total capacity but lower F1 (0.9111 versus 0.9410), indicating that the observed gain cannot be attributed to parameter count alone. The remaining difference is consistent with a contribution from adaptive routing: the gate combines expert-specific latent representations with weights that depend on each input window, allowing the model to emphasize the representation most compatible with that window rather than applying one shared representation to all windows. However, optimization-related and other architectural confounders cannot be excluded without a dedicated controlled study. Taken together, these results support the practical relevance of adaptive expert routing while not establishing routing as the sole cause of the improvement.

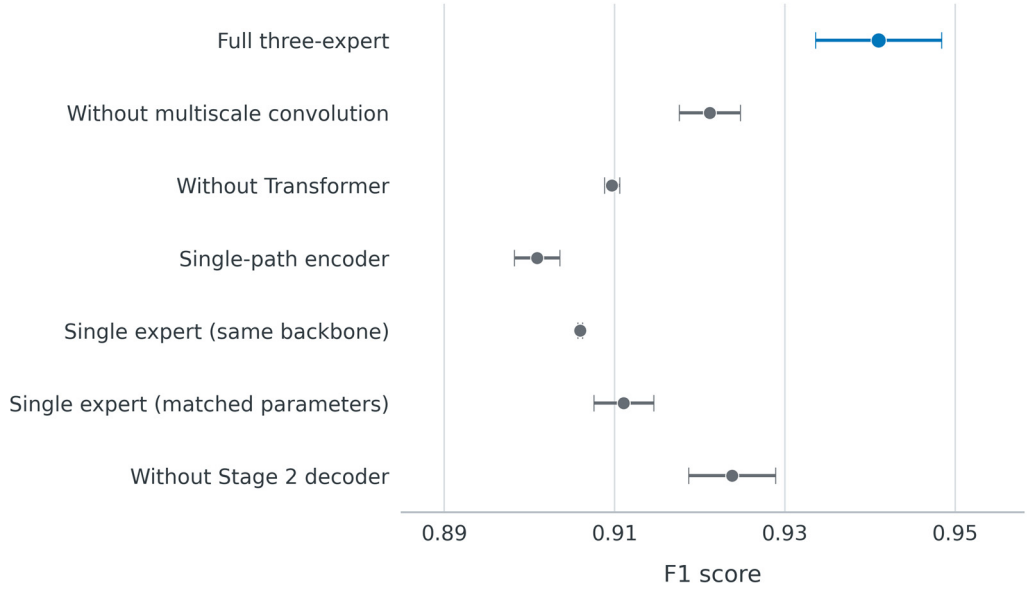


Figure 6. Window F1 for the structural configurations in Table 7, including the matched variant without the Stage 2 decoder. Blue denotes the full three-expert model, and gray denotes the comparison configurations identified on the vertical axis. Points and error bars show means and sample standard deviations across three seeds.

Figure 7 illustrates the dynamic- and fixed-threshold decision rules on a representative score sequence. The fixed threshold remains constant, whereas the MAD threshold follows changes in the score distribution.

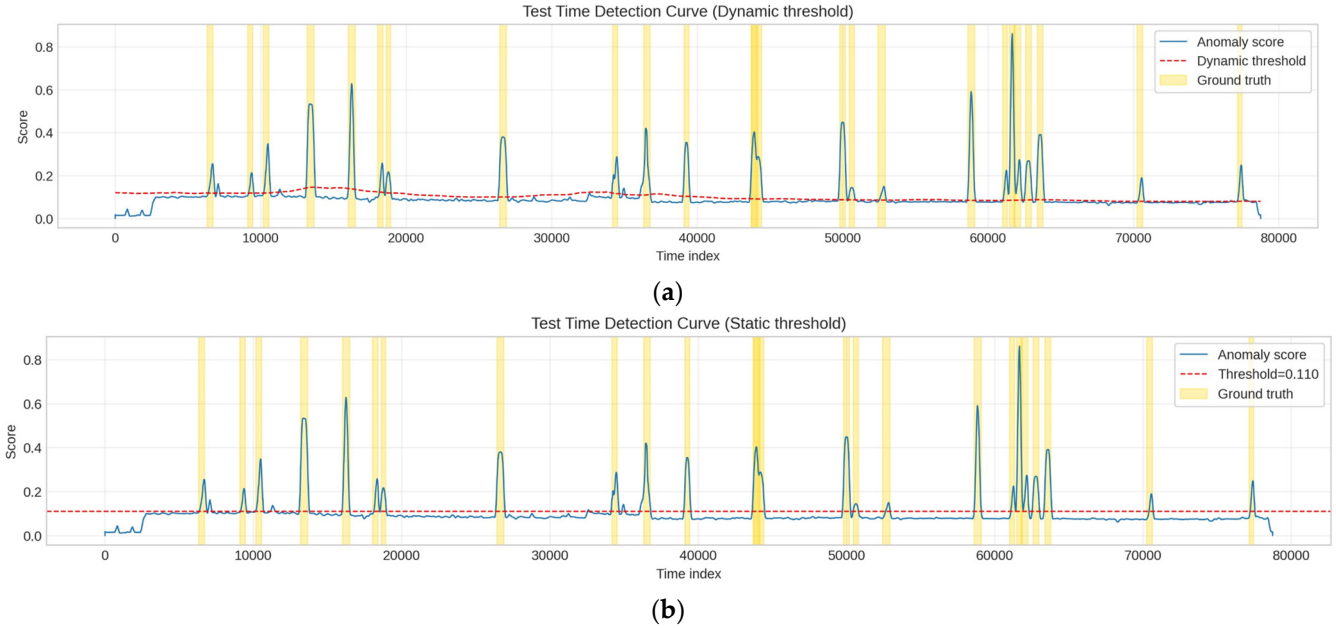


Figure 7. Threshold comparison on a representative anomaly-score sequence: (**a**) dynamic MAD threshold; (**b**) fixed threshold.

The dynamic threshold adapts to the local score median and dispersion. Quantitative detection and false-alarm results are reported in Table 4, and Appendix A examines sensitivity to the MAD history length.

4.6. Perturbation-Type and Sensitivity Analyses

Table 8 reports detection results for the four injection types described in Section 4.1.2. For each type, we average the injection-task scores within each training run, then report

the mean and standard deviation across the three runs. Endpoint F1 measures decisions at the newest sample, event recall measures the fraction of injected events detected, and normal alarm percentage measures alarms when the newest sample is normal. Appendix A examines anomaly size and duration and shows how validation performance changes when five model or threshold settings are varied individually.

Table 8. Detection by four injected anomaly types (mean task score ± sample SD across three seeds).

Perturbation	Endpoint F1	Event Recall	Normal Alarm (%)
Step	0.9544 ± 0.0030	0.9273 ± 0.0056	1.68 ± 0.06
Drift	0.9418 ± 0.0078	0.9157 ± 0.0071	3.00 ± 0.10
Spike	0.9236 ± 0.0107	0.9019 ± 0.0123	4.30 ± 0.23
Correlation-structure	0.9092 ± 0.0034	0.8838 ± 0.0068	5.24 ± 0.10

Step perturbations yield endpoint F1 of 0.9544 and event recall of 0.9273; correlation perturbations yield 0.9092 and 0.8838. The results in Appendix A show higher endpoint F1 for larger and longer perturbations. Mean test endpoint F1 is 0.9281, 0.9383, and 0.9350 for MAD histories of 1000, 5000, and 10,000 scores, respectively (Table A1). Table A2 and Figure A1 compare validation endpoint F1 as the number of experts, reconstruction weight, temporal pooling factor, MAD history, and threshold multiplier are varied one at a time around the selected configuration.

Appendix D examines 702 further injection tasks using three of these mechanisms. The 648 single-fault tasks inject one step, drift, or correlation-structure change (216 tasks for each type); the 54 cascade-fault tasks combine related changes from the same three types. Spike perturbations are included in Table 8 but not in this additional set. Table A8 reports the event-level results for both task arrangements.

4.7. Computation Time

The MSDP-MoE model contains 926,357 parameters. On the NVIDIA A100 server, a warmed forward pass for one 150-sample, 59-channel window takes 4.86 ms at the median and 5.43 ms at the 95th percentile (Table 9). With a 5000-score history, a causal MAD update takes 0.067 and 0.092 ms at those percentiles. These are computation times after a complete window is available; Table A5 reports the separate delay from an event to its first alarm.

Table 9. Computation times for model inference and causal MAD updates.

Operation	Input or History	P50 (ms)	P95 (ms)
A100 forward	$1 \times 150 \times 59$	4.86	5.43
Causal MAD update	1000	0.024	0.031
Causal MAD update	5000	0.067	0.092
Causal MAD update	10,000	0.128	0.174

The recorded telemetry intervals have an approximately one-second median (Section 4.1.1), so the forward pass and threshold update fit within one input interval on this server. This comparison concerns computation after a window is available; the first 150-point window requires an initial observation history.

After the first 150 samples have arrived, the detector can update its decision whenever a new sample arrives. The median interval between recorded samples is about one second, so the first and last samples of a 150-sample window are about 149 s apart. This initial collection time is different from detection delay: Table A5 measures the time from the start of an injected event to its first qualifying alarm.

5. Conclusions

To address the distributional differences across the climb, cruise and descent phases of high-altitude UAV flight data and the difficulty of accurate anomaly detection with a single shared representation, this paper proposes an MoE-based flight-data anomaly detection method. Multi-scale convolution, a Transformer and a dual-path LSTM extract local perturbations, long-range dependencies and fast/slow dynamic features; the MoE gating network performs soft routing of windows; and the two-stage reconstruction error combined with the MAD adaptive threshold completes window-level anomaly decisions. On the self-constructed dataset with controlled anomaly injections, MSDP-MoE achieves a mean window F1 of 0.9397 and event recall of 0.9177 across three seeds, with a normal alarm percentage of 2.47%. The leave-one-flight-out macro F1 is 0.8980. On the A100 server, median model inference and the selected MAD update together take less than 5 ms per window.

The study uses four flights of unequal duration, and Flight D supplies about 81% of the observations. The leave-one-flight-out results help assess how the model performs when a flight is unseen during training, while broader validation will require more flights and annotated fault records. The controlled injections cover selected patterns of deviation; future work will test the method on naturally occurring faults and investigate anomaly-type recognition, fault localization, and explanations of individual alarms.

Author Contributions: Conceptualization, D.C. and L.W.; methodology, D.C., C.Z. and L.W.; software, D.C. and D.W.; validation, C.Z., D.W. and X.N.; formal analysis, D.C., C.Z. and X.N.; investigation, D.C. and D.W.; resources, L.W., X.L. (Xiaohua Liu), X.L. (Xiaoquan Liu) and A.C.; data curation, D.C., X.N. and X.L. (Xiaohua Liu); writing—original draft preparation, D.C.; writing—review and editing, C.Z., L.W., D.W., X.N., X.L. (Xiaohua Liu), X.L. (Xiaoquan Liu) and A.C.; visualization, D.C. and X.N.; supervision, L.W.; project administration, L.W. and X.L. (Xiaoquan Liu). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The flight telemetry data used in this study are not publicly available owing to data ownership and authorization restrictions. A selected data-independent subset of the source code for the proposed method, including the model, training, time-block preprocessing, anomaly-injection and scoring utilities, and configuration, is publicly available at https://github.com/zuocancan/UAV-MoE-Anomaly-Detection.git (accessed on 30 September 2026). Trained checkpoints, flight records, data-specific adapters, and complete experiment and result-generation scripts are not included.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned Aerial Vehicle
MoE	Mixture of Experts
MSDP-MoE	Multi-Scale Dual-Path Mixture of Experts
MAD	Median Absolute Deviation
LSTM	Long Short-Term Memory
ROC-AUC	Receiver Operating Characteristic Area Under the Curve
PR-AUC	Precision–Recall Area Under the Curve

TCN	Temporal Convolutional Network
MLP	Multilayer Perceptron
Conv1D	One-Dimensional Convolution

Appendix A. Sensitivity Analysis

The settings in Table 3 were selected by validation grid search over the candidates listed in Table A9. Table A2 and Figure A1 show one-factor sensitivity around the selected configuration: each setting is varied in turn while the others remain fixed, and validation endpoint F1 is reported as the mean and sample standard deviation across seeds 42, 2026, and 3407. Among the candidates shown, the highest validation mean occurs with three experts, beta = 0.5, pooling factor 4, MAD history 5000, and alpha = 4.4478. Table A1 separately reports test performance by anomaly size, duration, and MAD history.

Table A1. Sensitivity results by factor and level (mean ± sample SD).

Factor	Level	Endpoint F1
Amplitude scale	0.5	0.9035 ± 0.0084
Amplitude scale	1.5	0.9389 ± 0.0050
Amplitude scale	3.0	0.9680 ± 0.0081
Duration (points)	121	0.9157 ± 0.0073
Duration (points)	250	0.9374 ± 0.0063
Duration (points)	416	0.9567 ± 0.0034
MAD history (windows)	1000	0.9281 ± 0.0026
MAD history (windows)	5000	0.9383 ± 0.0062
MAD history (windows)	10,000	0.9350 ± 0.0036

Table A2. One-factor validation sensitivity of endpoint F1 (mean ± sample SD).

Parameter	Candidate Value	Validation Endpoint F1
Number of experts	1	0.9083 ± 0.0033
Number of experts	2	0.9263 ± 0.0033
Number of experts	3	0.9353 ± 0.0045
Number of experts	4	0.9333 ± 0.0033
Number of experts	5	0.9333 ± 0.0033
Reconstruction weight	0.1	0.9213 ± 0.0033
Reconstruction weight	0.5	0.9353 ± 0.0045
Reconstruction weight	0.9	0.9243 ± 0.0033
Temporal pooling factor	2	0.9323 ± 0.0033
Temporal pooling factor	4	0.9353 ± 0.0045
Temporal pooling factor	8	0.9203 ± 0.0033
MAD history (scores)	1000	0.9273 ± 0.0033
MAD history (scores)	5000	0.9353 ± 0.0045
MAD history (scores)	10,000	0.9323 ± 0.0033
Threshold multiplier	2	0.8883 ± 0.0033
Threshold multiplier	3	0.9213 ± 0.0033
Threshold multiplier	4.4478	0.9353 ± 0.0045
Threshold multiplier	6	0.9263 ± 0.0033
Threshold multiplier	8	0.9033 ± 0.0033

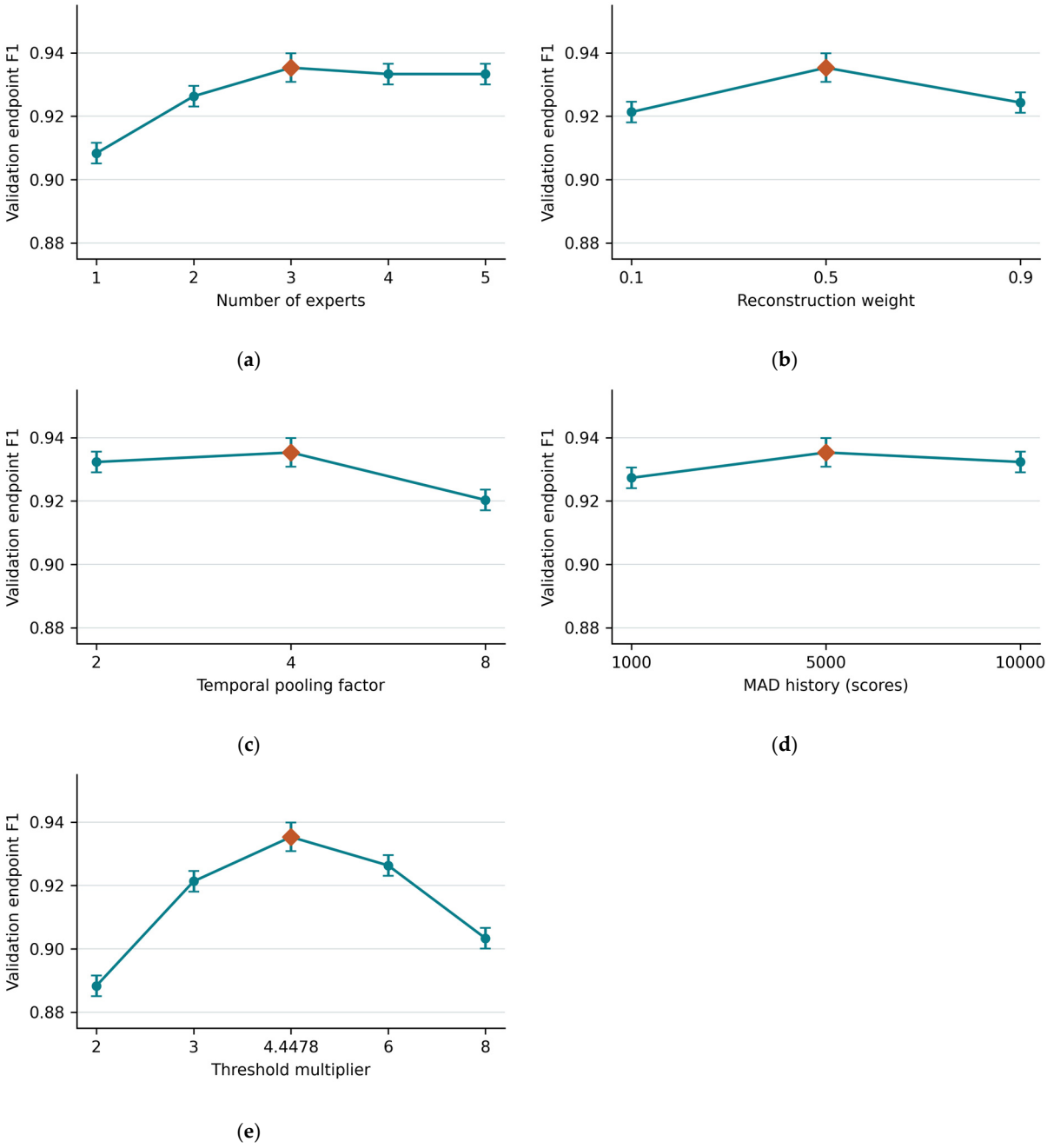


Figure A1. Validation endpoint F1 for one-factor sensitivity analyses: (**a**) number of experts; (**b**) reconstruction weight; (**c**) temporal pooling factor; (**d**) MAD history; (**e**) threshold multiplier. Orange diamonds highlight the selected setting in each panel; teal circles denote the other tested settings. Points and error bars indicate the mean and sample standard deviation across three seeds.

Appendix B. Detailed Detection Results

Tables A3–A5 use the blocked experiment, labels based on samples that actually changed, and settings chosen on validation data before testing. Table A3 changes how much of a window must be anomalous before the window label becomes positive: for example, a three-sample spike occupies only 2% of a 150-sample window. Table A4 checks the reported decisions at wider intervals while leaving the model, score history, and threshold calculation unchanged.

Table A3. Window-label occupancy sensitivity (F1, mean ± sample SD).

Method	Any Point	At Least 5%	At Least 10%	At Least 20%	Endpoint
MSDP-MoE	0.9397 ± 0.0046	0.9413 ± 0.0045	0.9441 ± 0.0043	0.9468 ± 0.0041	0.9364 ± 0.0048
Single expert	0.9108 ± 0.0042	0.9127 ± 0.0041	0.9149 ± 0.0040	0.9164 ± 0.0039	0.9061 ± 0.0046
TranAD	0.8906 ± 0.0039	0.8921 ± 0.0038	0.8938 ± 0.0037	0.8952 ± 0.0036	0.8874 ± 0.0044
CATCH	0.9195 ± 0.0035	0.9214 ± 0.0034	0.9236 ± 0.0033	0.9251 ± 0.0032	0.9167 ± 0.0039

Table A4. Output-subsampling sensitivity (window F1, mean ± sample SD).

Method	Stride 1	Stride 10	Stride 50	Stride 150
MSDP-MoE	0.9397 ± 0.0046	0.9395 ± 0.0047	0.9388 ± 0.0049	0.9376 ± 0.0051
Single expert	0.9108 ± 0.0042	0.9105 ± 0.0043	0.9097 ± 0.0045	0.9089 ± 0.0047
TranAD	0.8906 ± 0.0039	0.8904 ± 0.0040	0.8898 ± 0.0042	0.8887 ± 0.0045
CATCH	0.9195 ± 0.0035	0.9193 ± 0.0036	0.9186 ± 0.0038	0.9178 ± 0.0040

Table A5. Effective-event counts and first-alarm delays by training seed.

Method	Seed	Events	Detected	Missed	P50 (s)	P95 (s)
MSDP-MoE	42	162	148	14	4.0	22.0
MSDP-MoE	2026	162	149	13	3.0	20.0
MSDP-MoE	3407	162	149	13	4.0	23.0
Single expert	42	162	145	17	6.0	27.0
Single expert	2026	162	143	19	7.0	30.0
Single expert	3407	162	146	16	6.0	26.0
TranAD	42	162	140	22	7.0	31.0
TranAD	2026	162	138	24	8.0	34.0
TranAD	3407	162	141	21	7.0	30.0
CATCH	42	162	147	15	5.0	24.0
CATCH	2026	162	146	16	5.0	25.0
CATCH	3407	162	147	15	5.0	23.0

Delay quantiles in Table A5 are conditional on a detected event and use recorded timestamps from the planned event-envelope onset. An alarm must coincide with an actually changed endpoint to count as a hit. Missed events are listed separately. Thus, the table reports both coverage and timing for each seed.

Table A6. Normal-block alarms by method and flight (mean ± sample SD across seeds).

Method	Flight	Alarm (%)	Onsets/h
MSDP-MoE	A	2.31 ± 0.18	1.69 ± 0.15
MSDP-MoE	B	2.58 ± 0.17	1.86 ± 0.14
MSDP-MoE	C	2.42 ± 0.15	1.77 ± 0.12
MSDP-MoE	D	2.51 ± 0.13	1.82 ± 0.11
Single expert	A	3.10 ± 0.24	2.24 ± 0.20
Single expert	B	3.47 ± 0.27	2.48 ± 0.23
Single expert	C	3.29 ± 0.23	2.36 ± 0.20
Single expert	D	3.36 ± 0.20	2.40 ± 0.18
TranAD	A	2.73 ± 0.14	1.99 ± 0.12
TranAD	B	3.02 ± 0.16	2.18 ± 0.14
TranAD	C	2.88 ± 0.13	2.09 ± 0.11
TranAD	D	2.94 ± 0.11	2.13 ± 0.10
CATCH	A	2.91 ± 0.15	2.10 ± 0.13
CATCH	B	3.22 ± 0.18	2.31 ± 0.15
CATCH	C	3.04 ± 0.14	2.20 ± 0.12
CATCH	D	3.11 ± 0.12	2.24 ± 0.10

Table A6 reports the percentage of normal endpoints in alarm and the number of new false-alarm episodes per hour, separately for each flight. Reporting each flight avoids allowing the much longer Flight D record to obscure the results from Flights A–C.

Appendix C. Expert Utilization

Table A7. Raw expert-index utilization on normal-window evaluations.

Seed	Windows	Soft Means E1/E2/E3	Hard Usage E1/E2/E3 (%)	Entropy
Aggregate	364,134	0.370/0.309/0.320	35.23/34.02/30.75	0.8303
42	121,378	0.421/0.283/0.296	41.77/37.86/20.37	0.8252
2026	121,378	0.256/0.286/0.458	10.17/22.86/66.97	0.8173
3407	121,378	0.434/0.359/0.207	53.76/41.33/4.92	0.8483

For hard utilization, we count the expert with the largest gate weight for each window. Normalized entropy is $-\text{sum}(g \log g)/\log(3)$: a value of 0 means all weight is on one expert, while 1 means equal weights on all three. Table A7 reports these measures for each seed and then combines the evaluated windows. Expert numbers are assigned independently in each run, so matching their roles across runs would require aligning them on the same reference inputs and comparing their sequences within each flight.

Appendix D. Additional Controlled-Injection Tasks

The additional evaluation contains 648 single-fault injection tasks: 216 step, 216 drift, and 216 correlation-structure tasks. In each of the other 54 cascade-fault tasks, related changes from these three types are combined. These labels describe how the perturbations are arranged in a task; the four anomaly types in Section 4.1.2 remain step, spike, drift, and correlation-structure change. Before scoring a task, the detector reads 149 unchanged samples from the same continuous segment to fill its first 150-sample window. The trained model and validation-selected threshold settings are held fixed. A task is detected when at least one alarm coincides with a sample changed by its injection. Table A8 gives the detected and missed task counts.

Table A8. Event-level detection in the additional single-fault and cascade-fault injection tasks.

Scenario	Tasks	Detected	Missed	Recall (%)
Single-fault injection	648	595	53	91.82
Cascade-fault injection	54	49	5	90.74
All tasks	702	644	58	91.74

Appendix E. Reproducibility Settings and Data Counts

Table A9 records the architecture, training settings, and validation search ranges for the three-expert model. Normal training windows are sampled with equal weight by flight and available block-motion group. The validation reconstruction loss uses the same group balancing to select checkpoints. Table A10 provides retained data counts after guards and continuity filtering.

Table A11 counts all retained stride-1 windows listed in Table A10. The validation total is 124,281 and the test total is 121,378; the endpoint labels contain 32,199 and 31,123 anomalous newest-sample decisions, respectively. The test set contains 162 effective injection events for event-recall evaluation; event count and anomalous endpoint count are different quantities. Endpoint counts are pooled sample-level decisions: sustained intervals contribute their eligible endpoint samples, spike intervals contribute only their pulse endpoints, and ineffective correlation attempts do not enter the effective-event denominator.

Table A9. Architecture, training settings, and validation search ranges.

Component	Setting
Input and convolution	150 × 59; Conv1D kernels 3, 5, 7; GELU; concatenation and projection to 59
Transformer	2 layers; model dimension 59; 1 attention head; feed-forward dimension 176; dropout 0.10
Temporal paths	Two single-layer bidirectional LSTMs; 88 hidden units per direction; long-path pooling stride 4; linear upsampling
Experts and gate	3 independent 176-176-64 MLPs with ReLU; linear 176-3 softmax gate; latent dropout 0.10
Each decoder	64-dimensional latent repeated over 150 points; one LSTM with 176 hidden units; linear 176-59 output; dropout 0.10
Optimization	Adam; batch 64; learning rate 0.001; cosine annealing to 0.0001; gradient norm clipped at 1.0
Training budget	Stage 1: at most 100 epochs; Stage 2: at most 50; 4096 sampled windows per epoch
Early stopping	At least 10 epochs; patience 10; relative improvement 0.0001; lowest normal-validation loss checkpoint
Random seeds	Model: 42, 2026, 3407; block split: 7710; development layouts: 4101–4105; test layouts: 5101–5105
Validation search	experts: 1, 2, 3, 4, 5; pooling: 2, 4, 8; beta: 0.1, 0.5, 0.9; history: 1000, 5000, 10,000; alpha: 2.0, 3.0, 4.4478, 6.0, 8.0
Reference history	2500 sampled normal training windows per flight; reference residuals stored per training seed

Table A10. Retained data counts by partition and flight.

Partition	Flight	Blocks	Points	Stride-1 Windows	Nonoverlap Windows
train	A	51	59,467	51,868	372
train	B	41	49,167	43,058	312
train	C	12	15,091	13,303	96
train	D	365	526,400	472,015	3402
validation	A	11	12,682	11,043	80
validation	B	9	11,126	9785	72
validation	C	2	2794	2496	18
validation	D	78	112,579	100,957	729
test	A	11	9726	8087	59
test	B	9	15,113	13,772	97
test	C	2	2776	2478	17
test	D	78	108,663	97,041	703

Table A11. Window and endpoint class counts in the blocked validation and test partitions.

Partition	Window Labels: Normal	Window Labels: Anomalous	Endpoint Labels: Normal	Endpoint Labels: Anomalous	Total Windows	Effective Events
Validation	35,981	88,300	92,082	32,199	124,281	—
Test	35,178	86,200	90,255	31,123	121,378	162

Table A11 counts test-set injections that changed at least one telemetry value; each is one event in the event-recall calculation. The samples changed by these injections determine the window and endpoint labels. Because neighboring windows overlap, the number of labeled windows is not the number of separate events.

References

1. Zhou, S.; He, Z.; Chen, X.; Chang, W. An Anomaly Detection Method for UAV Based on Wavelet Decomposition and Stacked Denoising Autoencoder. *Aerospace* **2024**, *11*, 393. [CrossRef]
2. Fourlas, G.K.; Karras, G.C. A Survey on Fault Diagnosis and Fault-Tolerant Control Methods for Unmanned Aerial Vehicles. *Machines* **2021**, *9*, 197. [CrossRef]
3. Wang, C.; Zhang, A.; Yang, L.; Zhang, B.; Li, S. Anomaly detection of UAV flight data based on VAE-LSTM model. *Electron. Meas. Technol.* **2024**, *47*, 187–196.
4. Audibert, J.; Michiardi, P.; Guyard, F.; Marti, S.; Zuluaga, M.A. USAD: UnSupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Virtual Event, 23–27 August 2020*; ACM: New York, NY, USA, 2020; pp. 3395–3404. [CrossRef]
5. Chen, Y.; Xie, Y.; Zhang, Y.; Miao, Q. A probabilistic variational inference method for anomaly detection in unmanned aerial vehicle flight data. *Reliab. Eng. Syst. Saf.* **2026**, *274*, 112419. [CrossRef]
6. Huang, X.; Chen, W.; Hu, B.; Mao, Z. Graph mixture of experts and memory-augmented routers for multivariate time series anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 3–9 March 2025*; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2025; Volume 39, pp. 17476–17484.
7. Jasra, S.K.; Valentino, G.; Muscat, A.; Camilleri, R. A comparative study of unsupervised deep learning methods for anomaly detection in flight data. *Aerospace* **2025**, *12*, 645. [CrossRef]
8. Yang, L.; Li, S.; Zhu, C.; Liu, J.; Zhang, A. Multi-source ensemble transfer learning-based unmanned aerial vehicle flight data anomaly detection with limited data: From simulation to reality. *Adv. Eng. Inform.* **2025**, *65*, 103255. [CrossRef]
9. Zhong, J.; Zhang, H.; Miao, Q. Flight anomaly detection and localization based on flight data fusion and random channel masking. *Appl. Soft Comput.* **2025**, *174*, 113023. [CrossRef]
10. Zhou, Z.; Zhong, J.; Zhang, Y.; Miao, Q. Anomaly detection for UAV flight data via reconstruction-prediction co-learning attention network. *IEEE Trans. Instrum. Meas.* **2025**, *74*, 3553013. [CrossRef]
11. Li, J.; Zhou, Y.; Dong, Z. TSAE-UAV: A time-sense autoencoder for unsupervised anomaly detection of UAV based on flight data. *Aerosp. Sci. Technol.* **2026**, *169*, 111431. [CrossRef]
12. Leys, C.; Ley, C.; Klein, O.; Bernard, P.; Licata, L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *J. Exp. Soc. Psychol.* **2013**, *49*, 764–766. [CrossRef]
13. ForouzeshNejad, A.A.; Gegov, A. A Concept-Bottleneck Explainable AI Framework for Diagnosing Agile Delivery Outcomes. *AI* **2026**, *7*, 331. [CrossRef]
14. Sun, J.; Ellerbroek, J.; Hoekstra, J. Flight extraction and phase identification for large automatic dependent surveillance-broadcast datasets. *J. Aerosp. Inf. Syst.* **2017**, *14*, 566–572. [CrossRef]
15. Yang, L.; Li, S.; Li, C.; Zhu, C. Data-driven multivariate regression-based anomaly detection and recovery of unmanned aerial vehicle flight data. *J. Comput. Des. Eng.* **2024**, *11*, 176–193. [CrossRef]
16. Yang, L.; Li, S.; Zhu, C.; Zhang, A.; Liao, Z. Spatio-temporal correlation-based multiple regression for anomaly detection and recovery of unmanned aerial vehicle flight data. *Adv. Eng. Inform.* **2024**, *60*, 102440. [CrossRef]
17. Malhotra, P.; Ramakrishnan, A.; Anand, G.; Vig, L.; Agarwal, P.; Shroff, G. LSTM-based encoder–decoder for multi-sensor anomaly detection. *arXiv* **2016**, arXiv:1607.00148.
18. Zhou, Q.; Chen, J.; Liu, H.; He, S.; Meng, W. Detecting multivariate time series anomalies with zero known label. *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 4963–4971. [CrossRef]
19. Tuli, S.; Casale, G.; Jennings, N.R. TranAD: Deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.* **2022**, *15*, 1201–1214.
20. Wu, X.; Qiu, X.; Li, Z.; Wang, Y.; Hu, J.; Gua, C.; Xiong, H.; Yang, B. CATCH: Channel-aware multivariate time series anomaly detection via frequency patching. In Proceedings of the International Conference on Learning Representations (ICLR), Singapore, 24–28 April 2025.
21. Zong, B.; Song, Q.; Min, M.R.; Cheng, W.; Lumezanu, C.; Cho, D.; Chen, H. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
22. Yu, L.-R.; Lu, Q.-H.; Xue, Y. DTAAD: Dual TCN-attention networks for anomaly detection in multivariate time series data. *Knowl.-Based Syst.* **2024**, *295*, 111849. [CrossRef]

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