Supplementary Material C: Evaluation Details and Implementation Parameters

DETAILED EVALUATION METRICS

In addition to conventional metrics (Precision, Recall, F1-score, Accuracy), we employ several interpretability metrics to assess the model's ability to identify actionable stall precursors: Cosine Similarity (CS), Perturbation Curves, and Area Over the Perturbation Curve (AOPC).

Perturbation Curves and AOPC

Below we provide more details on the AOPC metric, which are based on the process proposed by (Early et al. 2022). To evaluate without time-wise precursor labels, we use a modified perturbation analysis by replacing precursor flight data with normal data. The concept is that, given a correct precursor-score order in flight data, iteratively substituting the most important precursor data with interpolated preceding data should cause the model prediction to decline. Conversely, an incorrect order will result in a slower decrease in prediction. Formally, when evaluating the interpretations generated by classifier F_θ for a flight series $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L\}$, we first re-order the time index according to the precursor scores $\boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_L\}$, where $\boldsymbol{o}_1$ corresponds to the timestamp with the highest precursor probability. The perturbation metric is then calculated by:

$$AOPC(\mathbf{X}, \mathbf{O}) = \sum_{j=1}^{L-1} F_\theta(\mathbf{X}) - F_\theta(MoRF(\mathbf{X}, \mathbf{O}, j)), \tag{S1}$$

$$MoRF(\mathbf{X}, \mathbf{O}, j) = \{x_i \quad if \quad i \notin \mathbf{O}_{1:j} \quad else \quad \tilde{x}_i | i \in \{1, 2, ..., L\}\}. \tag{S2}$$

Where $\boldsymbol{O}_{1:j}$ represents the first j elements in $\boldsymbol{O}$, and $\tilde{x}_i$ is the new data obtained by interpolating from the data prior to i. $MoRF$ is utilized to reconstruct flight data using a Most Relevant First ordering. In the equation above, the perturbation curve is obtained by substituting precursory data with normal data until only one data remains. Specifically, we focus on the perturbation curves of positive samples and calculate the area between the perturbation curve and the zero-axis as the AOPC.

Cosine Similarity

We use cosine similarity to evaluate the model-predicted precursors. In the task of precursor identification, it is essential not only to focus on the accuracy of sequence prediction but also to examine whether the trends of the predicted precursors align with those of the true labels in the time series. Cosine similarity is adept at capturing this kind of similarity. In our experiments, there are moments with high precursor scores, while the rest of the time remains relatively stable. The sensitivity of cosine similarity to sparse data enables it to effectively capture changes at these critical moments. Given the predicted label vector $\hat{\boldsymbol{Y}} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_L\}$and the ground truth $\boldsymbol{Y} = \{y_1, y_2, ..., y_L\}$, the CS is acquired through

$$CS = \frac{\boldsymbol{Y} \cdot \hat{\boldsymbol{Y}}}{\|\boldsymbol{Y}\| \left\|\hat{\boldsymbol{Y}}\right\|} = \frac{\sum_{i=1}^{L} y_i \hat{y}_i}{\sqrt{\sum_{i=1}^{L} y_i^2} \sqrt{\sum_{i=1}^{L} \hat{y}_i^2}}, \tag{S3}$$

where, $\|\cdot\|$ is the norm, $Y \cdot \hat{Y}$ stands for the dot production of Y and $\hat{Y}$.

Intelligent agent

We constructed an intelligent agent using the Proximal Policy Optimization (PPO) algorithm to control the aircraft back to a normal stable state from (Zhang et al. 2025). The detailed progress is summarized below:

PPO Algorithm Architecture: The PPO algorithm takes aircraft states (e.g., speed, altitude) as input. Its network comprises an input layer, two hidden layers (each with 128 neurons and ReLU activation), and an LSTM layer for temporal processing. The action network outputs control actions, while the value network estimates state values.

Training Process: Training data includes state, action, and reward sequences. The Generalized Advantage Estimation (GAE) method calculates the advantage function. The policy loss uses a clipped surrogate objective to ensure stable policy updates. The value network minimizes mean squared error (MSE) loss. Both networks are updated via the Adam optimizer with a learning rate of 0.0003.

TRAINING CONVERGENCE ANALYSIS

Figure S1 illustrates the changes in the loss function and accuracy on the query set during the meta-training process for all benchmark models in Group 1.

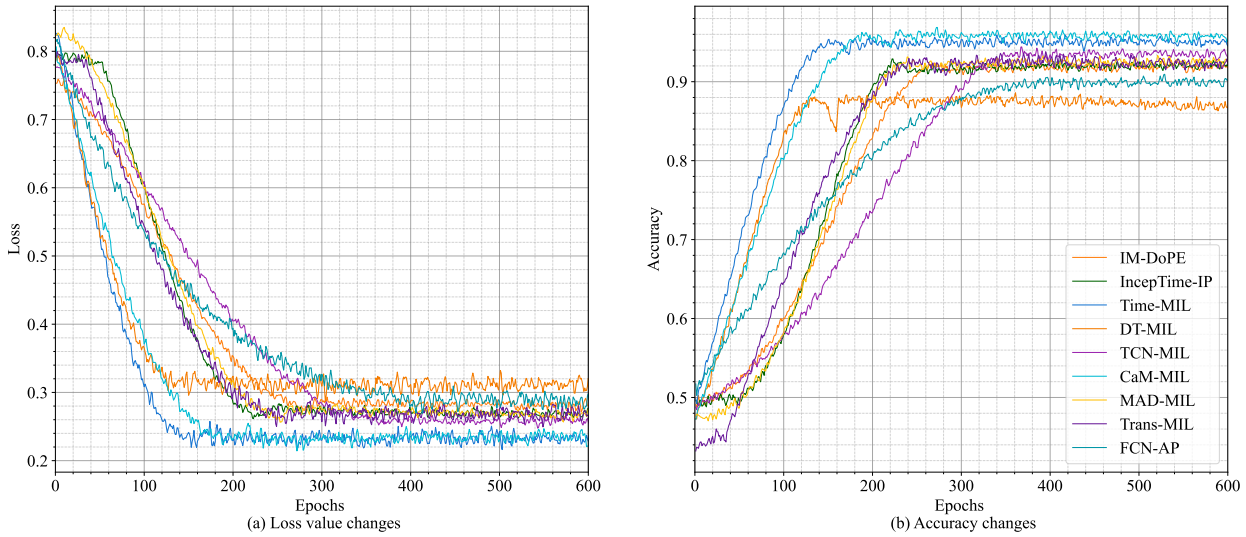

FIG. S1: The model performance on the testing set of Group 1 during training. The left side displays the loss value changes, and the right side shows the accuracy changes.

As observed, all models converge rapidly after an initial period of decline (around 300 epochs), demonstrating the stability of the model training process.

DETAILED HYPERPARAMETER SETTINGS

Table S1 summarizes the key hyperparameters used for training the CaM-MIL and the baseline models during the meta-learning and fine-tuning phases.

The meta-learning hyperparameters were specifically tuned for the RML process. During the fine-tune phase on $\mathcal{D}_{fine}$, the support set S and query set Q were constructed using a

TABLE S1: Key Hyperparameters for CaM-MIL and Baseline Models

Parameter	**Value/Setting**
Optimizer	Adam
Epochs	1000
Batch Size	16
Inner Loop Learning Rate (α_{IL})	0.0001
Outer Loop Learning Rate (β_{OL})	0.002
RML Training Support Set Size (K)	20
RML Inner Loop Update Steps	5
Data Split ($\mathcal{D}_{fine}$ Fine-tuning)	70 % Support / 30 % Query
Window Size (for DCaM-MIL/KD)	15 s

70%/30% random split of the target aircraft data, with the model updated 5 times on the support set.

COMPUTATION CONSUMPTION

The following tables present the detailed computational overhead for the model training process and the low-latency requirements for online monitoring.

Table S2 summarizes the memory and time consumption for one epoch of training for all models. These metrics are crucial for evaluating the feasibility of training different meta-learning architectures in resource-constrained environments. Table S3 reports the memory

TABLE S2: Training process model latency and memory usage (per epoch, batch size = 128)

Model name	Memory consumption (MB)	Time consumption (s)
DT-MIL	6.547	0.4351
TCN-MIL	9.082	1.4707
IM-DoPE	15.062	4.1227
Trans-MIL	9.655	3.1974
FCN-AP	10.711	2.4544
CaM-MIL	8.954	6.6281
IncepTime-IP	9.130	0.9248
Time-MIL	9.653	3.3692
MAD-MIL	8.386	1.1953
DCaM-MIL	8.349	0.7651

usage and time delay during the forward propagation of a single window of data. This scenario directly reflects the computational cost for real-time online monitoring, highlighting the superior efficiency of the DCaM-MIL model (0.059 MB memory and 3.142 ms delay), which is essential for actionable cockpit warnings.

TABLE S3: Online monitoring model latency and memory usage (per windowed data).

Model name	Memory consumption (MB)	Time delay (ms)
DT-MIL	0.291	15.356
TCN-MIL	0.574	34.931
IM-DoPE	6.542	108.736
Trans-MIL	1.345	20.541
FCN-AP	1.211	14.934
CaM-MIL	2.567	28.950
IncepTime-IP	1.090	16.946
Time-MIL	1.173	11.997
MAD-MIL	2.424	13.992
DCaM-MIL	0.059	3.142

REFERENCES

Early, J., Evers, C., and Ramchurn, S. D. (2022). "Model agnostic interpretability for multiple instance learning." *International Conference on Learning Representations (ICLR 2022)*, OpenReview, <https://openreview.net/forum?id=K3wXH8jM3k>.

Zhang, Y., Gao, Z., Gu, H., and Xiang, Z. (2025). "Optimization of wind shear escape training for civil aviation pilot trainees via reinforcement learning and sequential pattern mining." *Journal of Physics: Conference Series*, 2963(1), 012010.