automation

MDPI

Article

An Adaptive Reconstruction and Branch-Interaction Network for False Data Injection Attack Detection in Smart Grids

Shuxiang Wang [1] and Xinyu Wang [2,*]

[1] Qing Gong College, North China University of Science and Technology, Tangshan 063000, China
[2] School of Electrical Engineering, Yanshan University, Qinhuangdao 066004, China
* Correspondence: wxyzmya@ysu.edu.cn

Abstract

False data injection attacks (FDIAs) pose a serious threat to smart grids because they can alter state estimates without triggering conventional bad-data detection. Although learning-based detectors reduce the need for an exact system model, many of them focus on either temporal changes or relations among measurement variables. To address this problem, this paper presents an Adaptive Reconstruction and Branch-Interaction Network (ARBI-Net). A bidirectional long short-term memory network extracts temporal features from measurement windows, while an inverted Transformer learns relations among electrical variables. Between the two branches, a variable-token reconstruction module converts the temporal representations into tokens linked to individual measurement channels. An adaptive fusion module then combines the temporal and cross-variable features for classification. Experiments are conducted on the IEEE 6-bus, 14-bus, and 118-bus systems. The ablation study shows that the complete model achieves an F1-score of 97.18%, compared with 96.98% for BiLSTM alone and 94.39% for iTransformer alone. On the IEEE 14-bus and 118-bus systems, ARBI-Net achieves F1-scores of 92.60% and 98.50%, exceeding the best learning-based baselines by 1.41 and 10.63 percentage points, respectively. It also provides a better balance between missed detections and false alarms. Further tests show that its detection performance remains more stable under changes in attack strength and measurement noise.

Keywords: smart grid; false data injection attack; BiLSTM; iTransformer; adaptive feature fusion; cyberattack detection

1. Introduction

The smart grid relies on sensing, communication, and computing systems for state estimation, power dispatch, and real-time control [1–3]. Although this integration improves grid operation, it also exposes measurement and control processes to cyberattacks. False data injection attacks (FDIAs) are particularly difficult to detect because they can modify voltage magnitudes, phase angles, and active and reactive power measurements while preserving their consistency with the state-estimation model. Liu et al. first showed that a carefully constructed attack vector can bypass conventional bad data detection (BDD) [4]. Hug and Giampapa extended the vulnerability analysis to nonlinear AC state estimation [5]. These studies established the basic mechanism of stealthy FDIAs: an attacker can change the estimated system state while keeping the residual statistic below the BDD threshold. As a result, the control center may use an incorrect state estimate without receiving an alarm.

Model-based detection methods use system equations, observers, state estimators, or residual statistics to identify abnormal measurements. Pasqualetti et al. treated attacks as

Check for updates

Academic Editor: Ahmed Abu-Siada

Received: 14 September 2026
Revised: 16 October 2026
Accepted: 6 November 2026
Published: 10 November 2026

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

unknown inputs and established conditions for attack detection and identification [6]. Their analysis provides a clear system-level interpretation, but it requires an accurate dynamic model and sufficient observability. Manandhar et al. combined Kalman filtering with a chi-square detector to detect and isolate faults and attacks [7]. This approach depends on known system dynamics and noise statistics, and simultaneous attacks on several sensors remain difficult to isolate. Li et al. developed a quickest-detection method to reduce the delay in identifying an FDIA [8]. Its performance, however, depends on the assumed pre-attack and post-attack distributions. Foroutan and Salmasi learned a Gaussian mixture model from state-estimation residuals [9], but changes in the operating point or residual distribution can affect the learned decision boundary. Drayer and Routtenberg used graph signal processing and the AC grid model to detect abnormal state patterns [10]. In addition, to overcome the limitations of threshold design, a detection method based on interval observers has been proposed [11]. Although the method uses the physical grid structure, it assumes that the topology used by the detector is available and reliable. These methods have clear physical meanings, but their dependence on system models, topology, or statistical assumptions limits their use when operating conditions change.

Data-driven methods learn attack patterns directly from measurement data and require fewer explicit assumptions about the residual distribution. He et al. developed a deep neural network for real-time FDIA detection [12]. However, a model based mainly on individual measurement states cannot fully describe how an attack develops over time. Kundu et al. proposed an attention-based autoencoder for anomaly detection [13]. The reconstruction structure reduces the need for attack labels, but weak attacks may be reconstructed together with normal measurements. Boyaci et al. used a graph neural network to capture the spatial relations defined by the grid topology [14]. This improves the use of physical connections, although the detector depends on a known graph and may require adjustment after topology changes. Li et al. introduced federated deep learning to protect local measurement data [15], but communication cost and non-identically distributed local data remain practical concerns. Mukherjee studied the detection of data-driven blind attacks using deep learning [16], while Feng et al. used a semi-supervised adversarial network to reduce the need for labeled attack samples [17]. Nasir et al. compared CNN, LSTM, attention-based BiLSTM, and hybrid models and found clear differences in accuracy, training time, and memory use [18]. These studies mainly emphasize temporal, topological, reconstruction-based, or distributed representations. They provide useful results, but the joint use of bidirectional temporal features and reconstructed variable tokens has received less attention. Recent reviews identify generalization, scalability, data quality, real-time implementation, and false alarms as continuing problems in FDIA detection [19]. These problems are more evident when the detector must handle both temporal changes and relations among many electrical variables. This study therefore considers two related questions. First, how can temporal information be retained when measurement sequences are reorganized into variable tokens? Second, how can temporal and cross-variable features be combined without assigning the same importance to every feature component?

To overcome these limitations, this paper develops an Adaptive Reconstruction and Branch-Interaction Network (ARBI-Net) for stealthy FDIA detection. ARBI-Net learns temporal changes and cross-variable relations in parallel. It retains the identity of each measurement variable and adjusts the contribution of the two feature branches for each input sample. The main contributions are as follows.

1. A dual-output BiLSTM encoder is developed to retain both the complete hidden-state sequence and a pooled temporal feature. The sequence is used for variable-token reconstruction, while the pooled feature is sent directly to the fusion module. This direct path preserves a separate temporal representation for later fusion.

2. A variable-token reconstruction module is introduced to form variable-indexed tokens from the original measurement trajectory, variable identity, and BiLSTM context. A residual iTransformer models relations among these tokens, while a vector-valued gate and a difference-enhancement term combine the temporal and cross-variable features.
3. ARBI-Net is evaluated through ablation, comparison, robustness, cross-season, and cross-data-source experiments. The evaluation also includes WLS-BDD, parameter count, training time, and inference time to examine both detection performance and computational cost.

The rest of this paper is organized as follows. Section 2 introduces power-system state estimation, conventional bad-data detection, and the construction of stealthy FDIAs. Section 3 presents the structure and main modules of ARBI-Net. Section 4 describes the datasets, evaluation metrics, and experimental results. Section 5 concludes the paper and outlines future work.

2. Problem Formulation

2.1. Conventional State Estimation and Bad-Data Detection

State estimation provides the system states needed for grid monitoring and control. The measurement vector may contain bus-voltage magnitudes, active and reactive power injections, and branch power flows. The nonlinear AC measurement model can be written as [20]

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{e} \tag{1}$$

where z is the measurement vector, x is the system-state vector, $h(\cdot)$ represents the nonlinear AC measurement function, and e is the zero-mean measurement noise with covariance matrix R.

The system estimation state $\hat{x}$ can be obtained using weighted least-squares estimation:

$$\hat{\boldsymbol{x}} = \text{argmin}[\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x})]^T \boldsymbol{R}^{-1} [\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x})] \tag{2}$$

Based on Equation (2), the measurement residual can be calculated as [20]

$$\boldsymbol{r} = \boldsymbol{z} - \boldsymbol{h}(\hat{\boldsymbol{x}}) \tag{3}$$

where r is the estimation residual. Then, the corresponding conventional bad-data detection (BDD) is given as follows

$$\begin{cases} \boldsymbol{r} \le \zeta & \boldsymbol{Normal} \\ \boldsymbol{r} > \zeta & \boldsymbol{Abnormal} \end{cases} \tag{4}$$

where ζ is the chi-square detection threshold. Based on this rule, BDD can judge an abnormal measurement when the residual statistic exceeds the threshold. Its effectiveness therefore depends on whether the attack produces a sufficiently large residual [20].

2.2. Construction of a Stealthy FDIA

It is assumed that attackers have access to some measurements and basic information about the measurement model. Then, the compromised measurements are given below:

$$\boldsymbol{z}^a = \boldsymbol{z} + \boldsymbol{a} \tag{5}$$

where z^a denotes the attacked measurement vector, and a is the attack vector.

Let c be the state-deviation vector selected by the attacker. For an attack targeting bus i, it can be written as

$$\boldsymbol{c} = \begin{bmatrix} \Delta\theta^T & \Delta\boldsymbol{V}^T \end{bmatrix}^T \tag{6}$$

where $\Delta\theta$ and ΔV denote the malicious phase-angle and voltage-magnitude deviations, respectively. To ensure consistency between the compromised measurements, the nonlinear attack vector is constructed as [4]

$$\boldsymbol{a} = \boldsymbol{h}(\hat{\boldsymbol{x}} + \boldsymbol{c}) - \boldsymbol{h}(\hat{\boldsymbol{x}}) \tag{7}$$

Substituting Equations (1) and (7) into Equation (4), one can obtain the attacked residual as

$$\boldsymbol{r}^a = \boldsymbol{z}^a - \boldsymbol{h}(\hat{\boldsymbol{x}}^a) = \boldsymbol{z} - \boldsymbol{h}(\hat{\boldsymbol{x}}) = \boldsymbol{r} \tag{8}$$

where r^a is the residual under FDIA, and $\hat{x}^a$ is the estimation state under FDIA. Although the estimated operating state is shifted by c, Equation (8) indicates that the BDD statistic remains unchanged. Consequently, the FDIA cannot be detected solely by comparing the residual statistic with a fixed threshold.

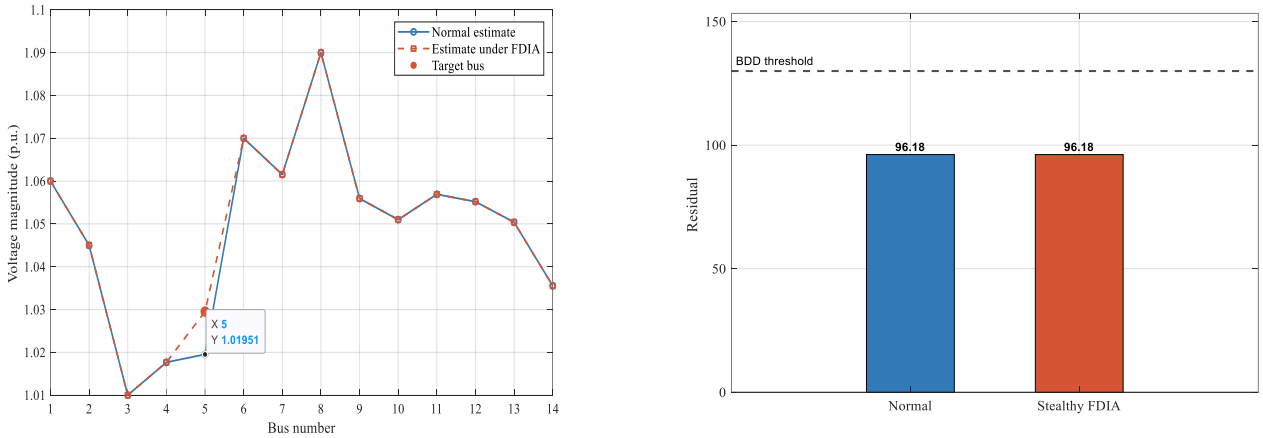
2.3. Illustrative BDD-Bypass Case

An illustrative case is constructed using the IEEE 14-bus system. The AC measurement vector contains bus-voltage magnitudes, active and reactive power injections, and active and reactive power flows measured at both ends of each branch. The voltage angle of the slack bus is excluded from the state vector. The corresponding parameters are presented in Table 1.

Table 1. Parameters of the illustrative stealthy FDIA case.

Parameters	Setting	Parameters	Setting
Number of buses	14	BDD degrees of freedom	95
Number of branches	20	Target bus	Bus 5
State dimension	27	Significance level	0.01
Measurement dimension	122		

With 95 degrees of freedom and a significance level of 0.01, the BDD threshold is set to 129.97 using the chi-square distribution. As shown in Figure 1a, the attack changes the estimated voltage magnitudes at several buses, whereas Figure 1b shows that the residual statistic remains below the BDD threshold. The simulation results indicated that the attacked measurements therefore lead to an incorrect state estimate without triggering a conventional BDD alarm.



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

Figure 1. Deceptive characteristics of the FDIA: (**a**) estimated voltage magnitudes of all buses before and after the attack; (**b**) residual statistics before and after the attack compared with the bad-data detection threshold.

3. ARBI-Net-Based Attack Detection Framework

Figure 2 shows the overall structure of ARBI-Net. The measurement sequence is first standardized and divided into fixed-length windows. A BiLSTM processes each window in both directions to extract temporal features. Since the BiLSTM output is arranged by time step, it cannot directly represent individual electrical variables. The variable-token reconstruction module therefore combines the temporal states with each measurement trajectory to form tokens for voltage, phase angle, active power, and reactive power. A residual iTransformer then applies attention across these variable tokens to learn their relations. Finally, adaptive fusion combines the BiLSTM and iTransformer features, and the classifier gives the probability of an attack.

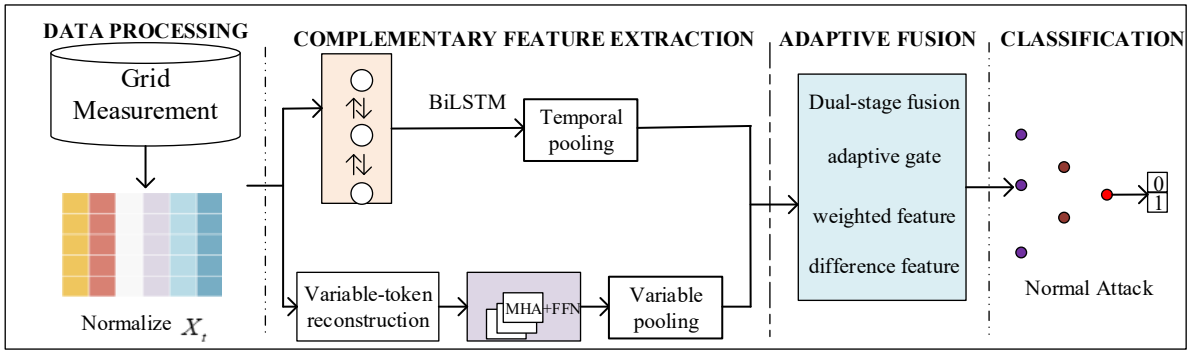


Figure 2. Architecture of the proposed ARBI-Net for FDIA detection.

3.1. Data Preprocessing

Power-system measurements have different units and numerical ranges. For example, voltage magnitude is close to one per unit, whereas active and reactive power can vary over a wider range. Direct use of these measurements may cause variables with large values to dominate model training. Therefore, each variable is standardized using statistics calculated only from the training set. Let x_t denote the measurement vector at sampling time t. The vector and its standardized components can be written as

$$\boldsymbol{x}_t = \left[x_{t,1}, x_{t,2}, \ldots, x_{t,m}\right]^T, \tilde{x}_{t,m} = \frac{x_{t,m} - \mu_m^{tr}}{\sigma_m^{tr} + \varepsilon} \tag{9}$$

where $x_{t,m}$ is the m-th component of $x_{t,m}$, m is the number of measurement variables, μ_m^{tr} and σ_m^{tr} are the training-set mean and standard deviation of the m-th variable, respectively, and ε is a small positive constant. The same training statistics are applied to the validation and test subsets. A window ending at sample t is defined as

$$\mathbf{X}_t = \left[\tilde{\boldsymbol{x}}_{t-T+1}, \tilde{\boldsymbol{x}}_{t-T+2}, \ldots, \tilde{\boldsymbol{x}}_{t-T+k}\right]^T \tag{10}$$

where T is the window length.

3.2. BiLSTM-Based Local Temporal Encoding

Grid measurements change continuously because of load variation, generator regulation, and renewable power fluctuation. An injected value may still lie within its normal range, but it can break the recent trend of the affected measurement. The BiLSTM branch is therefore used to learn changes within each observation window [21]. As shown in Figure 3, the forward and backward LSTM units process the same historical window in opposite directions. The backward unit does not use any future measurements beyond the end of the window. It only reads the available window in reverse order. At position k, the input to the BiLSTM is

$$\boldsymbol{u}_k = \tilde{\boldsymbol{x}}_{t-T+k} \tag{11}$$

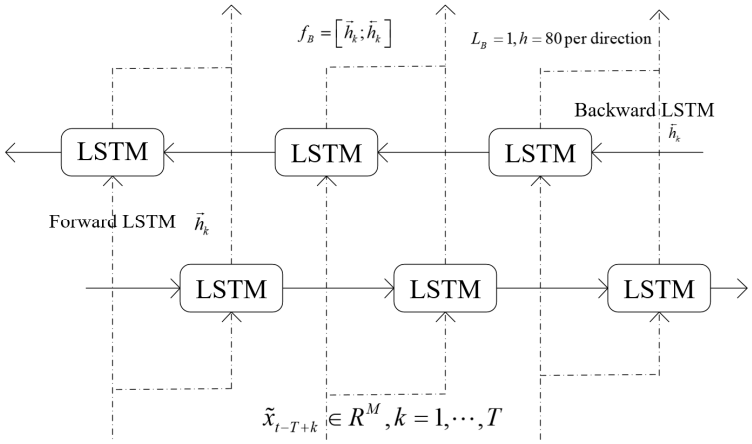


Figure 3. Structure of the BiLSTM temporal branch. The network contains one BiLSTM layer with 80 hidden units in each direction, where M and T denote the number of measurement variables and the window length, respectively.

The forward LSTM updates its gates and candidate cell state as follows [22]:

$$\vec{\boldsymbol{i}}_k = \sigma\left(\vec{\mathbf{W}}_i \boldsymbol{u}_k + \vec{\mathbf{U}}_i \vec{\boldsymbol{h}}_{k-1} + \vec{\boldsymbol{b}}_i\right) \tag{12}$$

$$\vec{\boldsymbol{f}}_k = \sigma\left(\vec{\mathbf{W}}_f \boldsymbol{u}_k + \vec{\mathbf{U}}_f \vec{\boldsymbol{h}}_{k-1} + \vec{\boldsymbol{b}}_f\right) \tag{13}$$

$$\vec{\boldsymbol{o}}_k = \sigma\left(\vec{\mathbf{W}}_o \boldsymbol{u}_k + \vec{\mathbf{U}}_o \vec{\boldsymbol{h}}_{k-1} + \vec{\boldsymbol{b}}_o\right) \tag{14}$$

$$\vec{\tilde{\boldsymbol{c}}}_k = \tanh\left(\vec{\mathbf{W}}_c \boldsymbol{u}_k + \vec{\mathbf{U}}_c \vec{\boldsymbol{h}}_{k-1} + \vec{\boldsymbol{b}}_c\right) \tag{15}$$

where $\vec{\boldsymbol{i}}_k$, $\vec{\boldsymbol{f}}_k$, and $\vec{\boldsymbol{o}}_k$ denote the input, forget, and output gates, respectively. The vector $\vec{\tilde{\boldsymbol{c}}}_k$ is the candidate cell state. The matrices $\vec{\mathbf{W}}_{\{\cdot\}}$ and $\vec{\mathbf{U}}_{\{\cdot\}}$ are trainable input and recurrent weight matrices, while $\vec{\boldsymbol{b}}_{\{\cdot\}}$ denotes the corresponding bias vector.

The forward cell state and hidden state are updated as

$$\begin{cases} \vec{\boldsymbol{c}}_k = \vec{\boldsymbol{f}}_k \odot \vec{\boldsymbol{c}}_{k-1} + \vec{\boldsymbol{i}}_k \odot \vec{\tilde{\boldsymbol{c}}}_k \\ \vec{\boldsymbol{h}}_k = \vec{\boldsymbol{o}}_k \odot \tanh\left(\vec{\boldsymbol{c}}_k\right) \end{cases} \tag{16}$$

where $\odot$ denotes element-wise multiplication. The backward LSTM has the same internal structure but processes the input sequence in the reverse direction. Its hidden state is represented by

$$\overleftarrow{\boldsymbol{h}}_k = LSTM_b\left(\boldsymbol{u}_k, \overleftarrow{\boldsymbol{h}}_{k+1}\right) \tag{17}$$

where $LSTM_b(\cdot)$ denotes the backward recurrent operation. The forward and backward hidden states are concatenated to form the bidirectional representation at position k:

$$\boldsymbol{h}_k^B = \left[\vec{\boldsymbol{h}}_k; \overleftarrow{\boldsymbol{h}}_k\right] \tag{18}$$

where $[\cdot;\cdot]$ denotes vector concatenation. The output representations from all positions are collected as

$$\mathbf{H}^B = \left[\boldsymbol{h}_1^B, \boldsymbol{h}_2^B, \ldots, \boldsymbol{h}_T^B\right]^T \tag{19}$$

The complete hidden-state sequence is retained to provide temporal context for variable-token reconstruction. Meanwhile, average pooling produces a compact temporal feature for the fusion module:

$$\boldsymbol{f}_B = Dropout\left[GELU\left(\mathbf{W}_B \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{h}_k^B + \boldsymbol{b}_B\right)\right] \tag{20}$$

where $\mathbf{W}_B$ and $\boldsymbol{b}_B$ are trainable parameters. The Gaussian error linear unit, denoted by $GELU(\cdot)$, introduces nonlinear feature transformation, while dropout is applied as a regularization operation.

3.3. Variable-Token Reconstruction

The rows of $\mathbf{H}^B$ correspond to time positions, whereas the iTransformer requires tokens that represent measurement variables. The rows of the BiLSTM output correspond to time positions, whereas the subsequent iTransformer operates on variable tokens. To connect these two representations, this study introduces a variable-token reconstruction module, as shown in Figure 4. For the m-th measurement variable, its trajectory and initial embedding $\boldsymbol{e}_m^0$ are defined as

$$\begin{cases} \boldsymbol{s}_m = \left[x_{1,m}, x_{2,m}, \ldots, x_{T,m}\right] \\ \boldsymbol{e}_m^0 = \mathbf{W}_0 \boldsymbol{s}_m + \boldsymbol{b}_0 + \boldsymbol{p}_m \end{cases} \tag{21}$$

where $\boldsymbol{s}_m$ is the standardized trajectory of the m-th variable. The $\mathbf{W}_0$ and $\boldsymbol{b}_0$ are trainable projection parameters, while $\boldsymbol{p}_m$ is a trainable variable-identity embedding.

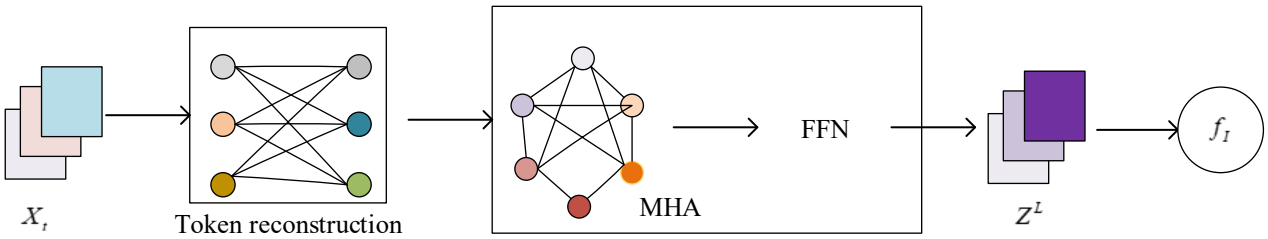


Figure 4. Structure of the variable-token reconstruction and residual iTransformer modules.

Following the scaled dot-product attention mechanism [23], the variable trajectory is used to form a query, while the BiLSTM states provide temporal keys and values:

$$\begin{cases} \boldsymbol{q}_m = \mathbf{W}_q \boldsymbol{s}_m + \boldsymbol{b}_q \\ \mathbf{K}^B = \mathbf{H}^B \mathbf{W}_k^B \\ \mathbf{V}^B = \mathbf{H}^B \mathbf{W}_v^B \\ \boldsymbol{c}_m = (\boldsymbol{a}_m)^T \mathbf{V}^B \end{cases} \tag{22}$$

where $\mathbf{W}_q$, $\mathbf{W}_k^B$, and $\mathbf{W}_v^B$ are trainable projection matrices. The vector $\boldsymbol{a}_m$ contains the attention weights assigned to the T positions, and $\boldsymbol{c}_m$ is the temporal context retrieved for the mm-th variable.

Since each variable forms its own query, the resulting attention weights may differ among voltage, phase-angle, and power measurements. The retrieved temporal context is then combined with the original variable embedding:

$$\begin{cases} \boldsymbol{e}_m = GELU\left(\mathbf{W}_r\left[\boldsymbol{e}_m^0; \boldsymbol{c}_m\right] + \boldsymbol{b}_r\right) \\ \boldsymbol{z}_m^0 = LayerNorm\left(\boldsymbol{e}_m + \boldsymbol{e}_m^0\right) \end{cases} \tag{23}$$

where $\mathbf{W}_r$, $\boldsymbol{b}_r$, and $[\cdot;\cdot]$ denote vector concatenation. Finally, the reconstructed tokens of all measurement variables are stacked to form the input of the iTransformer:

$$\mathbf{Z}^0 = \left[\boldsymbol{z}_1^0, \boldsymbol{z}_2^0, \ldots, \boldsymbol{z}_M^0\right]^T \tag{24}$$

3.4. Residual Cross-Variable Dependency Modeling

Voltage, phase-angle, and power measurements are related through the power-flow equations. A coordinated FDIA may preserve a smooth trajectory for an individual measurement while changing its relation with other variables. Following the variable-centered structure of iTransformer [24], self-attention is therefore applied across the reconstructed variable tokens. For the l-th encoder layer, the attention output is added to the input through a residual connection [25]:

$$\mathbf{U}^l = \mathbf{Z}^{l-1} + Dropout\left[MHA^l\left(\bar{\mathbf{Z}}^{l-1}\right)\right] \tag{25}$$

where $MHA^l(\cdot)$ denotes multi-head self-attention in layer l. Each head compares one variable token with all other variable tokens, which provides a mechanism for representing relations among different measurement variables. The attention output is normalized as

$$\overset{\sim}{\mathbf{U}}^l = LayerNorm\left(\mathbf{U}^l\right) \tag{26}$$

$$FFN^l\left(\overset{\sim}{\mathbf{U}}^l\right) = \mathbf{W}_2^l GELU\left(\mathbf{W}_1^l \overset{\sim}{\mathbf{U}}^l + \boldsymbol{b}_1^l\right) + \boldsymbol{b}_2^l \tag{27}$$

where $\mathbf{W}_1^l$, $\mathbf{W}_2^l$, $\boldsymbol{b}_1^l$, and $\boldsymbol{b}_2^l$ are trainable parameters. The first projection expands the feature dimension, and the second returns it to the token dimension. A second residual connection gives the output of the encoder layer:

$$\mathbf{Z}^l = \mathbf{U}^l + Dropout\left[FFN^l\left(\bar{\mathbf{U}}^l\right)\right] \tag{28}$$

After L encoder layers, the final cross-variable representation is

$$\mathbf{Z}^I = LayerNorm\left(\mathbf{Z}^L\right) \tag{29}$$

The variable tokens are averaged and projected into the fusion space:

$$\boldsymbol{f}_I = Dropout\left[GELU\left(\mathbf{W}_I \frac{1}{M}\sum_{m=1}^{M} z_m^I + \boldsymbol{b}_I\right)\right] \tag{30}$$

where $\mathbf{W}_I$ and $\boldsymbol{b}_I$ are trainable parameters, and z_m^I is the final representation of the m-th measurement variable. Consequently, $\boldsymbol{f}_I$ and the BiLSTM feature $\boldsymbol{f}_B$ have the same dimension and can be integrated by the adaptive fusion module.

3.5. Adaptive Dual-Stage Feature Fusion

The BiLSTM and iTransformer branches provide temporal and cross-variable representations, respectively. Their relative importance may change with the input sample. A learnable gate is therefore introduced to adjust the contribution of each branch at the feature-component level.

The two features are first normalized and concatenated:

$$\boldsymbol{f}_g = \left[LayerNorm(\boldsymbol{f}_B); \left[LayerNorm(\boldsymbol{f}_I);\right]\right] \tag{31}$$

The feature gate is then calculated as

$$\boldsymbol{a} = \sigma\left[\mathbf{W}_{g2} GELU\left(\mathbf{W}_{g1}\boldsymbol{f}_g + \boldsymbol{b}_{g1}\right) + \boldsymbol{b}_{g2}\right] \tag{32}$$

where $\mathbf{W}_{g1}$, $\mathbf{W}_{g2}$, $\boldsymbol{b}_{g1}$, and $\boldsymbol{b}_{g2}$ are trainable parameters. The sigmoid function constrains each element of $\boldsymbol{a}$ to the interval $(0, 1)$. The weighted feature is calculated as

$$\boldsymbol{f}_A = \boldsymbol{a} \odot \bar{\boldsymbol{f}}_B + (1 - \boldsymbol{a}) \odot \bar{\boldsymbol{f}}_I \tag{33}$$

The gate produces feature-wise weights for the two branches. In addition, the absolute difference between the two representations is included as an interaction term. The weighted and interaction features are then concatenated and projected:

$$\boldsymbol{f}_D = \left|\bar{\boldsymbol{f}}_B - \bar{\boldsymbol{f}}_I\right| \tag{34}$$

where $|\cdot|$ denotes element-wise absolute value. The weighted and interaction features are then concatenated and projected:

$$\boldsymbol{f} = Dropout\left[GELU\left(\mathbf{W}_F\left(\boldsymbol{f}_A; \boldsymbol{f}_D\right) + \boldsymbol{b}_F\right)\right] \tag{35}$$

where $\mathbf{W}_F$ and $\boldsymbol{b}_F$ are trainable parameters. The resulting vector f contains the final features used for attack classification.

3.6. Attack Classification and Training Objective

The fused feature is mapped to an attack probability through a sigmoid classifier:

$$\hat{\boldsymbol{y}} = \sigma(o) = \frac{1}{1 + \exp(-\boldsymbol{w}_c^T \boldsymbol{f} - \boldsymbol{b}_c)} \tag{36}$$

where $\boldsymbol{w}_c^T$ and $\boldsymbol{b}_c$ are trainable classification parameters, $\hat{\boldsymbol{y}} \in (0, 1)$ represents the predicted attack probability.

The numbers of normal and attacked samples may not be equal. In this case, standard binary cross-entropy can make the model favor the larger class. ARBI-Net is therefore trained using weighted binary cross-entropy:

$$L_{loss} = -\frac{1}{B}\sum_{i=1}^{B}[w_1 y_i \log(\hat{y}_i + \varepsilon) + w_0(1 - y_i)\log(1 - \hat{y}_i + \varepsilon)] \tag{37}$$

where $\hat{y}_i$ is the predicted attack probability, and ε is a small positive constant used to ensure numerical stability. y_i and $\hat{y}_i$ are the true label and predicted probability of sample I, respectively; w_0 and w_1 are class weights.

3.7. Algorithm Procedure

Algorithm 1 summarizes the training and detection process. The measurement sequence is divided in chronological order before sliding windows are created. This prevents overlapping windows from different subsets from sharing the same measurements. Attack construction is not included in the algorithm because it belongs to dataset generation rather than model training or detection.

Algorithm 1: ARBI-Net ATTACK DETECTION ALGORITHM

Require: Labeled measurement sequence $\mathbf{X}_t$, window length T
Output: Trained ARBI-Net and predicted attack labels

1. Divide the measurement sequence into training, validation, and test sets in chronological order.
2. Calculate the mean and standard deviation of each variable from the training set.
3. Standardize all three subsets using the training-set statistics in Equation (9).
4. Construct measurement windows using Equation (10).
5. **For** each training mini-batch:
6. Use the BiLSTM to obtain $\mathbf{H}^B$ and $\boldsymbol{f}_B$ from Equations (11)–(20).
7. Reconstruct the variable tokens $\mathbf{Z}^0$ using Equations (21)–(24).
8. Use the residual iTransformer to obtain $\boldsymbol{f}_I$ from Equations (25)–(30).
9. Combine $\boldsymbol{f}_A$ and $\boldsymbol{f}_D$ using Equations (32)–(35).
10. Calculate the attack probability using Equation (36).
11. Calculate the weighted loss using Equation (37), and update the network parameters.
12. **End For**
13. Evaluate the model on the validation set after each epoch and apply early stopping.
14. Select the decision threshold from the validation results.
15. Use the best model and selected threshold to classify the test samples.

4. Results

This section evaluates ARBI-Net through five groups of experiments: ablation, cross-season and cross-source validation, comparison with existing methods, robustness analysis, and computational efficiency analysis.

4.1. Dataset Construction

The datasets were generated using MATPOWER and the IEEE 6-bus, 14-bus, and 118-bus systems. Their main parameters are listed in Table 2. A balanced steady-state AC power-flow model was used. The network topology and branch parameters remained fixed, while the loads were represented as constant active and reactive power demands. The active and reactive loads at each bus were scaled by the same factor to retain the original power factor.

Table 2. Main parameters of the IEEE 6-bus, 14-bus, and 118-bus systems.

System	Buses	Branches	Equivalent Generators	Load Buses	Base MVA
IEEE 6-bus	6	11	3	3	100
IEEE 14-bus	14	20	5	11	100
IEEE 118-bus	118	186	54	99	100

The generators were modeled as equivalent power injections using the parameters provided in the test cases. The test systems do not specify whether a generator is a photovoltaic, wind, hydropower, concentrated solar power, or thermal unit. Therefore, plant-level dynamics and generation technologies were not modeled separately. Here, a PV bus refers to a bus with specified active power and voltage magnitude, rather than a photovoltaic plant. This setting is suitable because the study focuses on measurement-level FDIA detection under steady-state conditions. Historical NYISO load data were normalized and mapped to the rated loads of the test systems [26]. An AC power flow was solved at each measurement time, and only converged operating points were retained. The measurement vector included voltage magnitudes, phase angles, active and reactive power injections, and branch power flows. All valid measurement periods were eligible for attack generation. Using a fixed random seed of 2022, 50% of these periods were selected uniformly across the complete load profile. The selection was not based on load magnitude or time of day, so low-, medium-, and high-load conditions were all included. Attacks were then generated using Equations (5)–(7). The selected periods and attack settings were fixed for all detection methods. Finally, the labeled sequence was divided chronologically into training, validation, and test sets before sliding windows were constructed.

4.2. Evaluation Metrics

Accuracy, precision, recall, and F1-score are used to evaluate detection performance. Accuracy measures the proportion of correctly classified samples. Precision indicates how many attack alarms are correct, while recall measures how many attacks are successfully detected. The F1 considers both precision and recall and is useful when missed attacks and false alarms are both important. The above metrics are calculated as [27]

$$\begin{cases} \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \\ \text{Precision} = \frac{TP}{TP+FP} \\ \text{Recall} = \frac{TP}{TP+FN} \\ \text{F1} = \frac{2TP}{2TP+FP+FN} \end{cases} \tag{38}$$

where TP and TN denote the numbers of correctly detected attack and normal samples, respectively. FP is the number of normal samples incorrectly identified as attacks, and FN is the number of missed attack samples.

4.3. Ablation Study

The ablation study is conducted to determine whether the performance improvement comes from the proposed modules rather than from a simple increase in model size. Two single-branch variants are constructed to examine the contributions of temporal and cross-variable feature learning: BiLSTM only and iTransformer only. The BiLSTM model uses 48 hidden units and a dropout rate of 0.18. Its learning rate is 6×10^{-4}, and the maximum number of training epochs is 26. The iTransformer uses an embedding dimension of 64, four attention heads, and a dropout rate of 0.15. Its learning rate is 4×10^{-4}, and it is trained for up to 30 epochs. For ARBI-Net, the number of BiLSTM hidden units is set to 80, while the token and fusion dimensions are both set to 128. Four attention heads are used. The dropout rate and learning rate are set to 0.05 and 2.5×10^{-4}, respectively. The maximum number of training epochs is 42. The corresponding simulation results are presented in Table 3 and Figure 5.

Table 3. Ablation results on the IEEE 6-bus system.

Model	F1%
BiLSTM only	96.98
iTransformer only	94.39
Proposed model	97.18

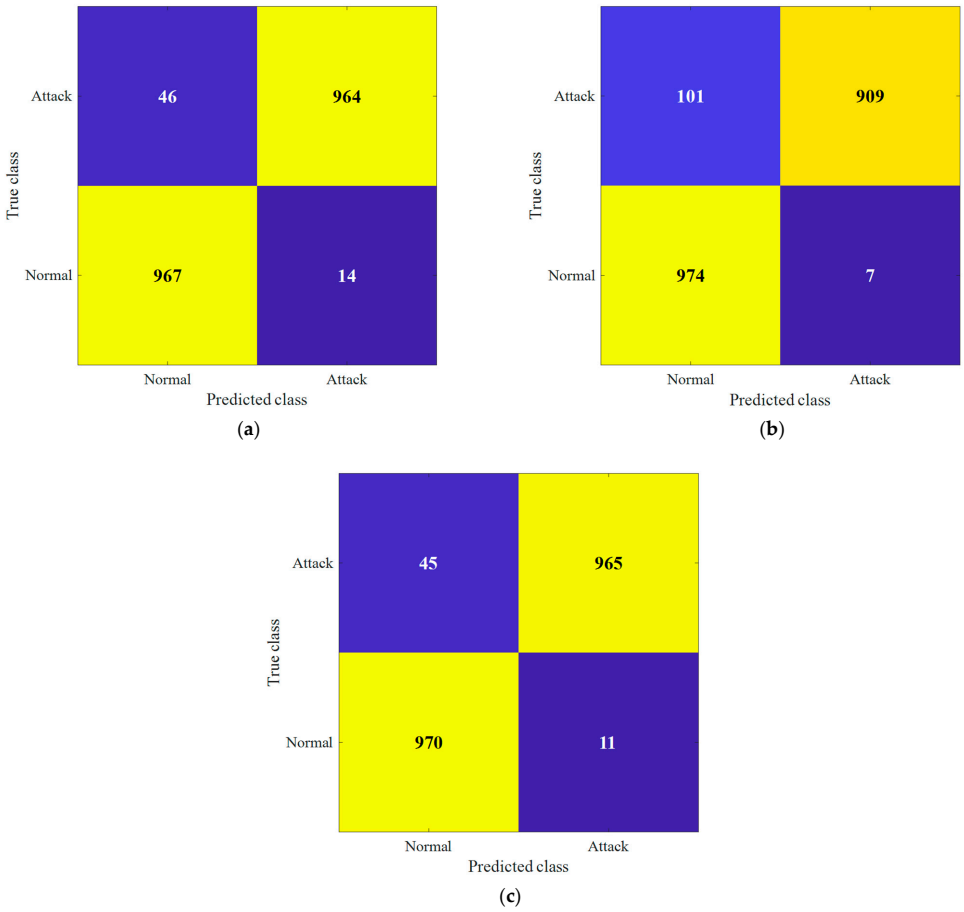


Figure 5. Confusion matrices of ARBI-Net on the IEEE 6-bus under different models: (**a**) BiLSTM only; (**b**) iTransformer only; (**c**) Proposed model.

Table 3 shows that the complete ARBI-Net achieves the highest F1-score of 97.18%. The BiLSTM-only and iTransformer-only models obtain F1-scores of 96.98% and 94.39%, respectively. The larger decrease for the iTransformer-only model indicates that temporal information makes a stronger contribution in this experiment. The complete model nevertheless gives the highest F1-score, showing that the two branches provide complementary information under the tested setting. Figure 5 provides a more detailed view of the three detection results. The BiLSTM-only model misses 46 attacks and produces 14 false alarms. The iTransformer-only model produces only seven false alarms, but it misses 101 attacks. This result suggests that cross-variable features help reduce false alarms but are less effective when used alone to identify temporal changes. ARBI-Net misses 45 attacks and produces 11 false alarms. It therefore retains the high recall of the temporal branch while using cross-variable information to control false alarms.

4.4. Cross-Season and Cross-Source Validation

The original experiments used load profiles from the New York Independent System Operator (NYISO). Although the training and test samples were separated, a random split may place similar seasonal patterns in both sets. Compared with four existing network structures, Convolutional Neural Network–Long Short-Term Memory (CNN-LSTM), Temporal Convolutional Network (TCN), Gated Recurrent Unit with Attention (GRU-Attention), and Convolutional Neural Network–Transformer (CNN-Transformer), two additional experiments were therefore conducted on the IEEE 6-bus system. The first examines seasonal changes within the NYISO data, while the second evaluates transfer to an independent load-data source. The same measurement variables, window length, attack settings, and class ratio were used for all methods.

4.4.1. Cross-Season Validation

The NYISO data were divided into four seasons according to their timestamps. A leave-one-season-out protocol was used. In each run, one complete season was used only for testing, while data from the other three seasons were divided into training and validation sets, as shown in Table 4. The held-out season was not involved in model training, threshold selection, or hyperparameter selection. This setting reduces temporal leakage and provides a stricter test than a random data split. The corresponding test results are obtained, as shown in Table 5.

Table 4. Data partitions used in the cross-season experiment.

Held-Out Season	Training Windows	Validation Windows
Spring	5125	1224
Summer	5125	1224
Autumn	5144	1229
Winter	5192	1234

Table 5. F1 in the leave-one-season-out experiment.

Model	Spring	Summer	Autumn	Winter
CNN-LSTM	90.56	90.35	89.08	90.47
TCN	91.38	87.58	90.33	95.19
GRU-Attention	91.95	91.38	88.19	94.82
CNN-Transformer	93.21	88.71	87.77	89.62
Proposed model	99.42	97.86	99.59	99.90

As shown in Table 5, ARBI-Net gives the highest F1-score in all four tests. Its F1-score ranges from 97.86% to 99.90%, with an average of 99.19%. In comparison, the highest

average F1-score among the other methods is 91.58%. The proposed method therefore provides an average improvement of 7.61 percentage points. The lowest result occurs when summer is used as the test season. Summer loads differ more clearly from the remaining seasons, but ARBI-Net still exceeds the best comparison method by 6.48 percentage points. The above results show that the detector remains stable when the seasonal load pattern is absent from the training set.

4.4.2. Cross-Source Validation

To further examine data-source dependence, the models were trained using the NYISO-based dataset and tested using load profiles obtained from the Electric Reliability Council of Texas (ERCOT) Hourly Load Data Archives. The dataset contains hourly load records from 1 January to 31 December 2024. The loads of all ERCOT weather zones were summed to obtain the system-level load profile. Unlike the NYISO data used for model development, the ERCOT data were used only for external testing and were not involved in model training, validation, threshold selection, or hyperparameter tuning. Because the two data sources have different load ranges, each profile was mapped to the operating range of the IEEE 6-bus system. AC power flow was then calculated in MATPOWER to obtain the corresponding voltage and power measurements. Only the load-data source was changed; the network parameters, measurement types, attack-generation method, window length, and class ratio were kept fixed. The training, validation, and external test sets contained 6862, 1637, and 8731 windows, respectively. The corresponding test results are obtained, as shown in Table 6.

Table 6. Results of cross-data-source validation on the IEEE 6-bus system.

Models	Accuracy %	Precision %	Recall %	F1 %
CNN-LSTM	87.75	84.15	93.04	88.37
TCN	82.40	74.47	98.65	84.87
GRU-Attention	94.74	93.33	96.38	94.83
CNN-Transformer	84.69	78.24	96.16	86.28
Proposed model	98.27	97.18	99.43	98.29

As shown in Table 6, ARBI-Net achieves the best result for all four metrics. Its F1-score is 98.29%, which is 3.46 percentage points higher than that of GRU-Attention, the best comparison method. TCN has a high recall of 98.65%, but its precision is only 74.47%, indicating a large number of false alarms. In contrast, ARBI-Net maintains both high precision and high recall. The above test results demonstrate that ARBI-Net can transfer from the NYISO profiles to a different load-data source without retraining.

4.5. Comparison with Existing Detection Methods

Compared with CNN-LSTM, TCN, GRU-Attention, CNN-Transformer, and weighted least squares-based bad data detection (WLS-BDD), the detection performance is analyzed. The above model settings are hyperparameters rather than standard values or power-system assumptions. They were selected through preliminary experiments on the validation set. The test set was not used for model selection, parameter tuning, or decision-threshold selection. Each model was tuned separately because recurrent and attention-based networks do not necessarily perform well under the same settings. However, the same data partitions, validation criterion, random seed, training budget, and early stopping rule were used for all models. Validation F1-score was used as the main selection criterion. When two configurations produced similar results, the smaller model was selected. The candidate ranges and final settings are given in Table 7. The BiLSTM model used 48 hidden units, a dropout rate of 0.18, and a learning rate of 0.0006. The iTransformer used an embedding dimension of 64, four attention heads, a dropout rate of 0.15, and a learning rate of 0.0001.

ARBI-Net used 80 BiLSTM hidden units and token and fusion dimensions of 128. Its dropout rate and learning rate were 0.05 and 0.00025, respectively. Four attention heads were used because this setting gave stable validation performance without the additional cost of eight heads. All models were trained for at most 50 epochs with an early-stopping patience of six epochs. The BiLSTM, iTransformer, and ARBI-Net models stopped after 26, 30, and 42 epochs, respectively. The model checkpoint with the best validation F1-score was retained for final testing. The corresponding test results are obtained, as shown in Table 8 and Figures 6 and 7.

Table 7. Hyperparameter ranges and selected settings.

Hyperparameter	Candidate Values	BiLSTM	iTransformer	ARBI-Net
BiLSTM hidden units	32, 48, 64, 80	48	-	80
Embedding/token dimension	32, 64, 128	-	64	128
Fusion dimension	32, 64, 128	-	-	128
Attention heads	2, 4, 8	-	4	4
Dropout rate	0.05–0.30	0.18	0.15	0.05
Learning rate	0.0001∼0.001	0.0006	0.0001	0.00025
Maximum epochs	50	50	50	50
Actual stopping epoch	Determined by early stopping	26	30	42

Table 8. Detection performance on the IEEE 14-bus and 118-bus systems.

IEEE System	Models	Accuracy %	Precision %	Recall %	F1 %
14-bus	CNN-LSTM	89.25	86.35	92.86	89.48
	TCN	89.90	90.37	88.99	89.67
	GRU-Attention	91.31	91.05	91.33	91.19
	CNN-Transformer	90.80	91.56	89.60	90.57
	WLS-BDD	50.92	55.26	2.14	4.12
	Proposed model	92.81	94.01	91.23	92.60
118-bus	CNN-LSTM	86.89	89.32	83.19	86.15
	TCN	88.29	89.31	86.47	87.87
	GRU-Attention	81.51	79.28	84.32	81.72
	CNN-Transformer	87.09	85.70	88.42	87.04
	WLS-BDD	50.42	47.46	10.55	17.26
	Proposed model	98.54	99.37	97.64	98.50

Table 7 compares ARBI-Net with four learning-based methods and weighted least squares-based bad data detection (WLS-BDD). On the IEEE 14-bus system, ARBI-Net achieves an accuracy of 92.81%, a precision of 94.01%, a recall of 91.23%, and an F1-score of 92.60%. Its F1-score is 1.41 percentage points higher than that of GRU-Attention, the best learning-based baseline. CNN-LSTM gives a higher recall of 92.86%, but its precision is only 86.35%. This means that CNN-LSTM detects more attack samples at the cost of more false alarms. ARBI-Net gives a better balance between precision and recall. WLS-BDD performs poorly, with a recall of 2.14% and an F1-score of 4.12%, because most stealthy attacks do not cause the residual to exceed the BDD threshold. The performance gap becomes larger on the IEEE 118-bus system. ARBI-Net achieves an accuracy of 98.54%, a precision of 99.37%, a recall of 97.64%, and an F1-score of 98.50%. The best learning-based baseline is TCN, with an F1-score of 87.87%. ARBI-Net therefore improves the F1-score by 10.63 percentage points. WLS-BDD reaches an F1-score of only 17.26%. Although this value is higher than its result on the IEEE 14-bus system, it still misses most attack samples. The above results show that residual-based detection has limited ability to identify attacks designed to remain consistent with the state-estimation model. In contrast, ARBI-Net maintains a high F1-score on the tested IEEE 118-bus dataset, which contains more measurement variables than the IEEE 14-bus dataset.

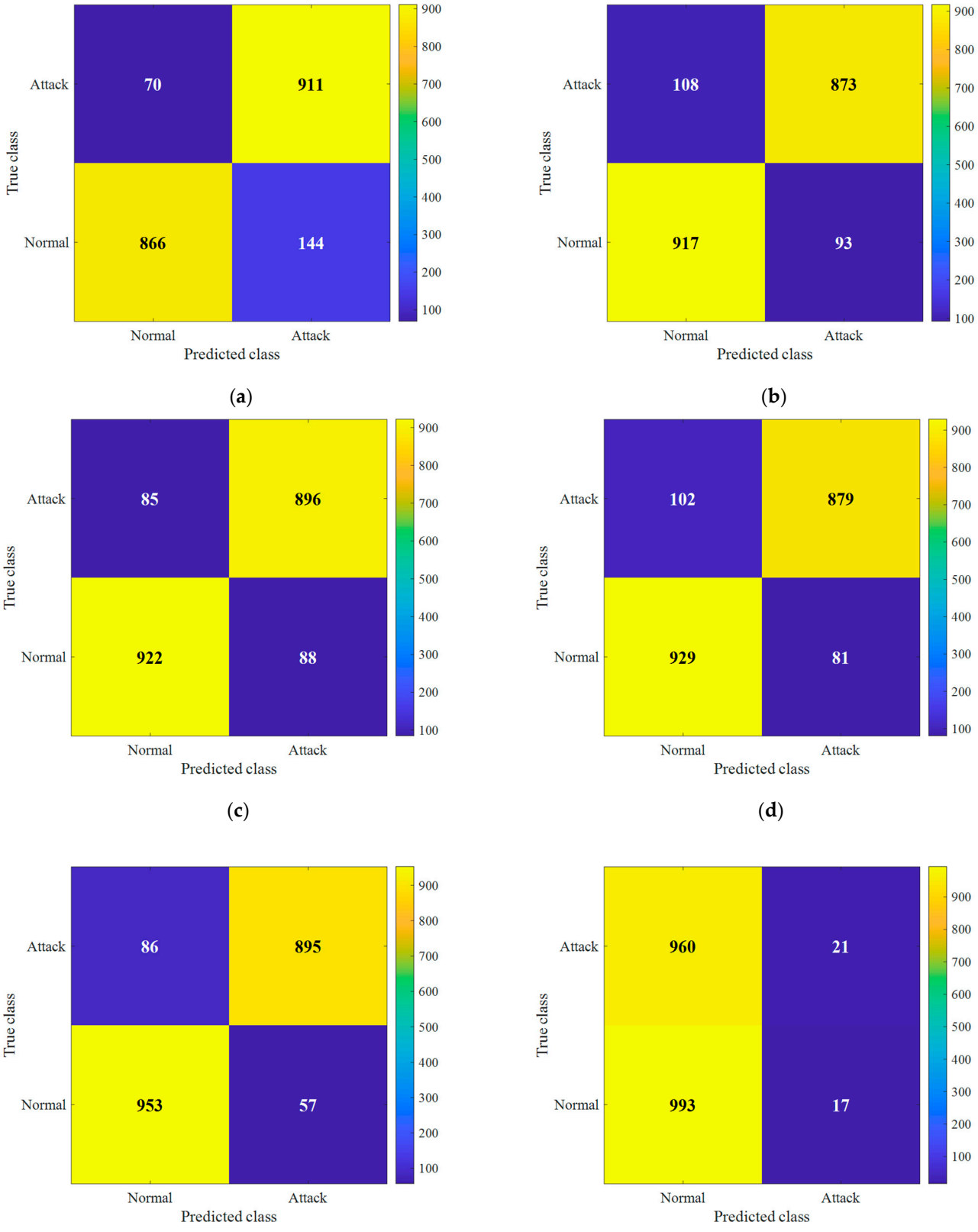


Figure 6. Confusion matrices of different detection methods on the IEEE 14-bus under different models: (**a**) CNN-LSTM; (**b**) TCN; (**c**) GRU-Attention; (**d**) CNN-Transformer; (**e**) Proposed model; (**f**) WLS-BDD.

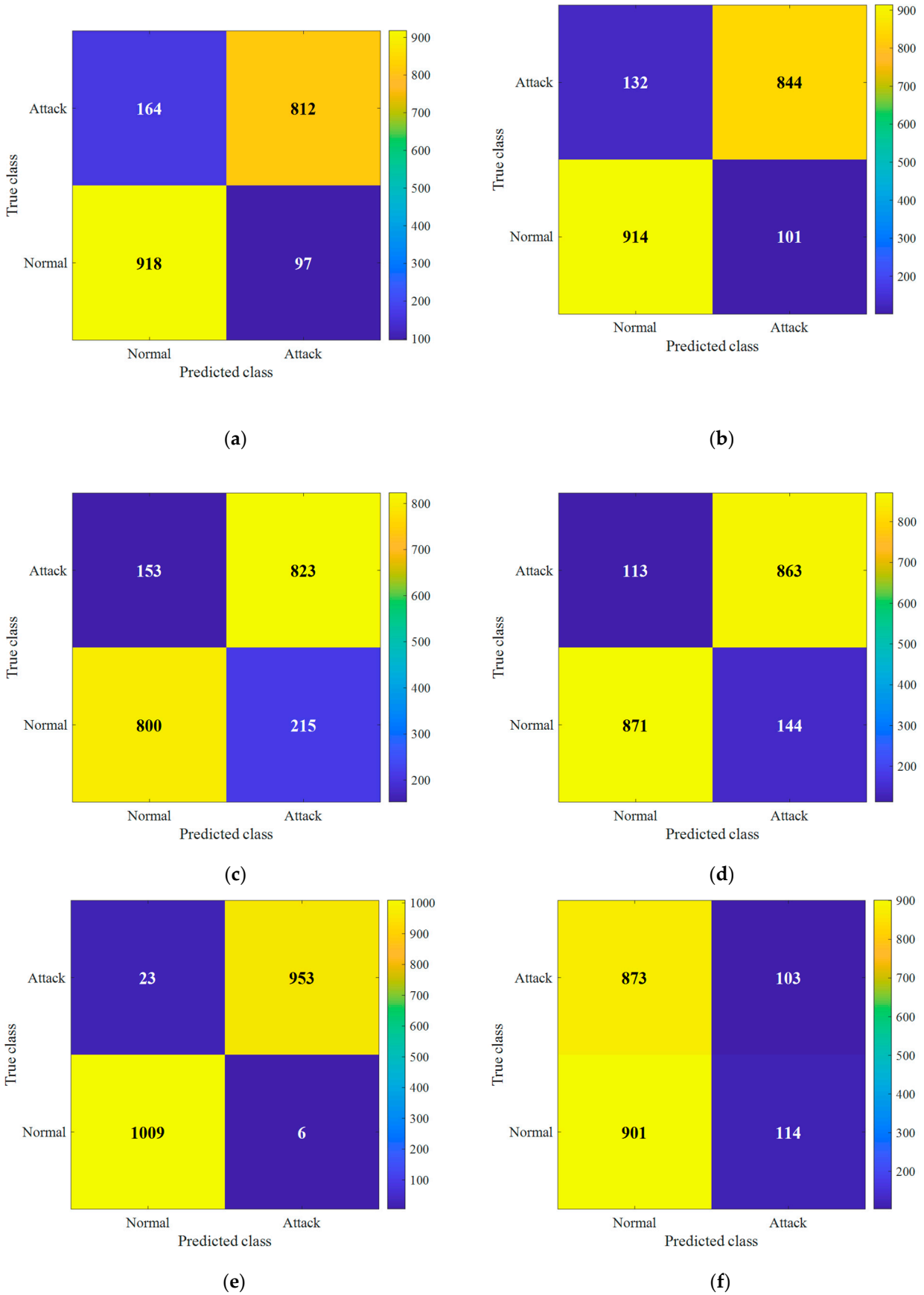


Figure 7. Confusion matrices of different detection methods on the IEEE 118-bus under different models: (**a**) CNN-LSTM; (**b**) TCN; (**c**) GRU-Attention; (**d**) CNN-Transformer; (**e**) Proposed model; (**f**) WLS-BDD.

Figure 6 shows the confusion matrices for the IEEE 14-bus system. ARBI-Net correctly classifies 895 attack samples and 953 normal samples, with 86 missed attacks and 57 false alarms. CNN-LSTM misses fewer attacks, but it produces 144 false alarms. The other learning-based methods also show a weaker balance between these two errors. WLS-BDD correctly detects only 21 attack samples and misses 960, confirming that the residual threshold is ineffective against most stealthy attacks in this dataset. Figure 7 presents the confusion matrices for the IEEE 118-bus system. ARBI-Net correctly identifies 953 attack samples and 1009 normal samples. It misses only 23 attacks and produces six false alarms. By comparison, CNN-LSTM, TCN, GRU-Attention, and CNN-Transformer miss 164, 132, 153, and 113 attacks, respectively. WLS-BDD misses 873 attacks and correctly detects only 103. Under the tested conditions, the larger performance difference on the IEEE 118-bus system is consistent with the use of both temporal and cross-variable representations.

4.6. Robustness Analysis

In this section, we further test the robustness of the proposed detection model. The robustness of ARBI-Net is tested against escalating attack strengths and noise levels in comparison with CNN-LSTM, TCN, GRU-Attention, and CNN-Transformer. The corresponding results are obtained, as shown in Figures 8 and 9.

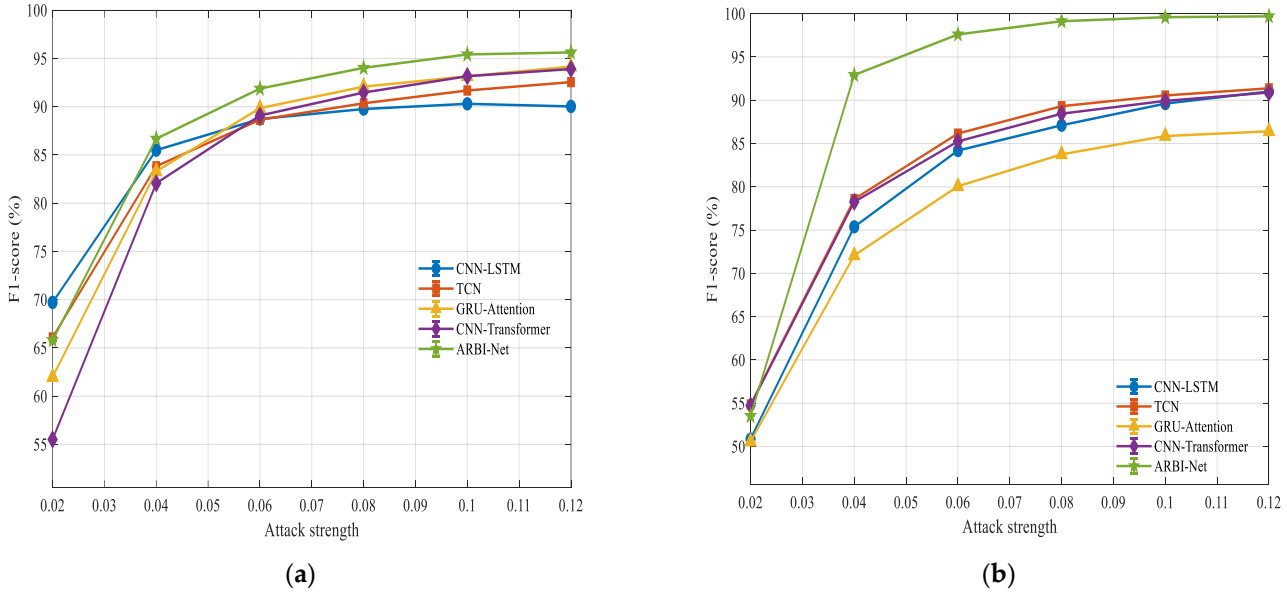


Figure 8. F1-scores of different methods under different attack strengths: (**a**) IEEE 14-bus; (**b**) IEEE 118-bus.

Figure 8a shows the results on the IEEE 14-bus system. At an attack strength of 0.02, all methods give relatively low F1-scores because the attack changes are close to normal load variations. ARBI-Net is not the best at this level, but its performance improves quickly as the attack becomes stronger. Its F1-score increases from about 66% at 0.02 to about 95% at 0.12. From 0.04 onward, ARBI-Net gives the highest result. Under the tested settings, ARBI-Net performs better once the attack-induced changes become more distinguishable. Figure 8b presents the results on the IEEE 118-bus system. The difference between ARBI-Net and the comparison methods is more evident on this larger system. At an attack strength of 0.04, ARBI-Net reaches an F1-score of about 93%, while the other methods remain below 80%. Its F1-score further increases to about 98% at 0.06 and remains close to 100% at higher attack strengths. The result suggests that joint temporal and cross-variable learning is especially useful when the measurement dimension increases.

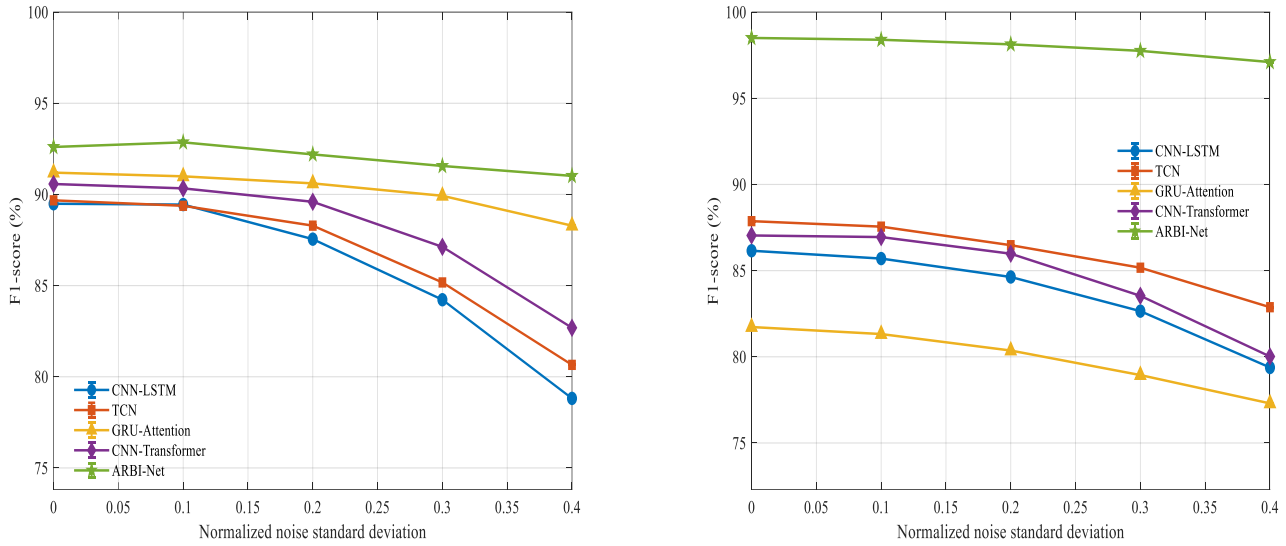


Figure 9. F1-scores of different methods under measurement-noise levels: (**a**) IEEE 14-bus; (**b**) IEEE 118-bus.

Figure 9a shows the effect of measurement noise on the IEEE 14-bus system. The performance of the four comparison methods decreases steadily as the noise level increases. At a normalized noise standard deviation of 0.40, their F1-scores fall to about 78–83%. In contrast, the F1-score of ARBI-Net decreases only slightly, from about 99% to 97%. The model therefore remains stable even when the measurements contain strong random disturbances. Figure 9b gives the results for the IEEE 118-bus system. ARBI-Net maintains an F1-score of about 91% at the highest noise level. CNN-LSTM and TCN fall to about 79% and 81%, respectively, while CNN-Transformer decreases to about 83%. GRU-Attention is the strongest baseline under high noise, but its F1-score is still around 2.5 percentage points lower than that of ARBI-Net at 0.40. These results show that the complete ARBI-Net is less sensitive to measurement noise than the comparison methods under the tested conditions.

4.7. Computational Efficiency Analysis

The computational cost of each method was measured on the IEEE 14-bus and IEEE 118-bus datasets. The reported measures include the number of trainable parameters, training time, total inference time, and average inference time per sample. All models used the same training and test partitions. Inference time was measured after three warm-up runs. Each test was then repeated ten times, and the median result was used. The corresponding results are obtained, as shown in Table 9.

Table 9 shows that WLS-BDD has the lowest computational cost because it does not require neural-network training. However, its F1-scores are only 4.12% and 17.26% on the IEEE 14-bus and IEEE 118-bus systems, respectively. Its low cost is therefore accompanied by poor detection of stealthy attacks. Among the learning-based methods, ARBI-Net requires more parameters and a longer inference time. On the IEEE 14-bus system, its average inference time is 0.0741 ms per sample. This value increases to 0.2988 ms on the IEEE 118-bus system. This increase is consistent with the larger number of measurement variables processed by the variable-wise attention module. Even so, the model can process about 13,494 samples per second on the IEEE 14-bus system and 3347 samples per second on the IEEE 118-bus system. Both results indicate that inference can be completed within one millisecond for each sample. The increase in cost is accompanied by a clear improvement in detection performance. ARBI-Net obtains F1-scores of 92.60% and 98.50% on the two

systems. On the IEEE 118-bus system, its F1-score is 10.63 percentage points higher than that of the best learning-based comparison method. These results show that ARBI-Net trades a moderate increase in computation for stronger attack detection. The sub-millisecond inference time indicates the potential for online detection on the tested hardware. However, deployment on resource-limited devices requires further evaluation.

Table 9. Computational efficiency on the IEEE 14-bus and IEEE 118-bus systems.

IEEE System	Models	Parameters	Training Time (s)	Total Inference Time (s)	Inference Time (ms/Sample)
14-bus	CNN-LSTM	1354	24.365	0.073533	0.036933
	TCN	2706	94.583	0.039212	0.019695
	GRU-Attention	4274	29.264	0.066215	0.033257
	CNN-Transformer	1170	27.588	0.057686	0.028973
	WLS-BDD	0	-	0.001647	0.000827
	Proposed model	67,906	51.062	0.147550	0.074107
118-bus	CNN-LSTM	7338	15.992	0.128790	0.064685
	TCN	14,674	55.354	0.125840	0.063203
	GRU-Attention	22,226	55.149	0.158320	0.079517
	CNN-Transformer	7154	49.361	0.159500	0.080108
	WLS-BDD	0	-	0.012852	0.006455
	Proposed model	115,780	134.030	0.594830	0.298760

5. Conclusions

This study developed ARBI-Net for detecting stealthy FDIAs. The BiLSTM and iTransformer branches provide temporal and cross-variable representations, while variable-token reconstruction connects the two forms of features before adaptive fusion. In the ablation study, the complete model achieved an F1-score of 97.18%, compared with 96.98% for BiLSTM alone and 94.39% for iTransformer alone. ARBI-Net obtained F1-scores of 92.60% and 98.50% on the IEEE 14-bus and 118-bus systems, exceeding the best learning-based baselines by 1.41 and 10.63 percentage points, respectively. The robustness experiments also showed a smaller performance loss under changing attack strengths and measurement-noise levels.

ARBI-Net requires more parameters and a longer inference time than the comparison networks, although its measured inference time remains below 1 ms per sample on the tested hardware. Future work will investigate lightweight token projection, parameter sharing, knowledge distillation, and structured pruning, with the aim of reducing this computational cost. The present results are limited to IEEE benchmark systems and semi-synthetic measurements generated through AC power-flow simulations. Validation using field SCADA and PMU measurements is still required.

Author Contributions: Conceptualization, S.W. and X.W.; methodology, S.W.; software, S.W.; validation, X.W.; formal analysis, X.W.; investigation, X.W.; resources, X.W.; data curation, X.W.; writing—original draft preparation, S.W.; writing—review and editing, X.W.; visualization, S.W.; supervision, S.W.; project administration, X.W.; funding acquisition, S.W. 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 original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: During the preparation of this manuscript, the authors used ChatGPT 5.6 (OpenAI) to improve the English language, grammar, and readability of the text. The authors reviewed and edited all AI-assisted content and take full responsibility for the final manuscript.

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

References

1. Farhangi, H. The path of the smart grid. *IEEE Power Energy Mag.* **2010**, *8*, 18–28. [CrossRef]
2. Fang, X.; Misra, S.; Xue, G.; Yang, D. Smart grid—The new and improved power grid: A survey. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 944–980. [CrossRef]
3. Gungor, V.C.; Sahin, D.; Kocak, T.; Ergüt, S.; Buccella, C.; Cecati, C.; Hancke, G.P. Smart grid technologies: Communication technologies and standards. *IEEE Trans. Ind. Inform.* **2011**, *7*, 529–539. [CrossRef]
4. Liu, Y.; Ning, P.; Reiter, M.K. False data injection attacks against state estimation in electric power grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2009; pp. 21–32.
5. Hug, G.; Giampapa, J.A. Vulnerability assessment of AC state estimation with respect to false data injection cyber-attacks. *IEEE Trans. Smart Grid* **2012**, *3*, 1362–1370. [CrossRef]
6. Pasqualetti, F.; Dörfler, F.; Bullo, F. Attack detection and identification in cyber-physical systems. *IEEE Trans. Autom. Control* **2013**, *58*, 2715–2729. [CrossRef]
7. Manandhar, K.; Cao, X.; Hu, F.; Liu, Y. Detection of faults and attacks including false data injection attack in smart grid using Kalman filter. *IEEE Trans. Control Netw. Syst.* **2014**, *1*, 370–379. [CrossRef]
8. Li, S.; Yılmaz, Y.; Wang, X. Quickest detection of false data injection attack in wide-area smart grids. *IEEE Trans. Smart Grid* **2015**, *6*, 2725–2735. [CrossRef]
9. Foroutan, S.A.; Salmasi, F.R. Detection of false data injection attacks against state estimation in smart grids based on a mixture Gaussian distribution learning method. *IET Cyber-Phys. Syst. Theory Appl.* **2017**, *2*, 161–171. [CrossRef]
10. Drayer, E.; Routtenberg, T. Detection of false data injection attacks in smart grids based on graph signal processing. *IEEE Syst. J.* **2020**, *14*, 1886–1896. [CrossRef]
11. Wang, X.; Luo, X.; Zhang, M.; Jiang, Z.; Guan, X. Detection and isolation of false data injection attacks in smart grid via unknown input interval observer. *IEEE Internet Things J.* **2020**, *7*, 3214–3229. [CrossRef]
12. He, Y.; Mendis, G.J.; Wei, J. Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism. *IEEE Trans. Smart Grid* **2017**, *8*, 2505–2516. [CrossRef]
13. Kundu, A.; Sahu, A.; Serpedin, E.; Davis, K. A3D: Attention-based auto-encoder anomaly detector for false data injection attacks. *Electr. Power Syst. Res.* **2020**, *189*, 106795. [CrossRef]
14. Boyaci, O.; Umunnakwe, A.; Sahu, A.; Narimani, M.R.; Ismail, M.; Davis, K.; Serpedin, E. Graph neural networks based detection of stealth false data injection attacks in smart grids. *IEEE Syst. J.* **2022**, *16*, 2946–2957. [CrossRef]
15. Li, Y.; Wei, X.; Li, Y.; Dong, Z.; Shahidehpour, M. Detection of false data injection attacks in smart grid: A secure federated deep learning approach. *IEEE Trans. Smart Grid* **2022**, *13*, 4862–4872. [CrossRef]
16. Mukherjee, D. Detection of data-driven blind cyber-attacks on smart grid: A deep learning approach. *Sustain. Cities Soc.* **2023**, *92*, 104475. [CrossRef]
17. Feng, H.; Han, Y.; Li, K.; Si, F.; Zhao, Q. Locational detection of the false data injection attacks via semi-supervised multi-label adversarial network. *Int. J. Electr. Power Energy Syst.* **2024**, *155*, 109682. [CrossRef]
18. Nasir, Q.; Abu Talib, M.; Arshad, M.A.; Ishak, T.; Berrim, R.; Alsaid, B.; Badway, Y.; Abu Waraga, O. Comparison of deep learning algorithms for site detection of false data injection attacks in smart grids. *Energy Inform.* **2024**, *7*, 71. [CrossRef]
19. Irfan, M.; Sadighian, A.; Tanveer, A.; Al-Naimi, S.J.; Oligeri, G. A survey on detection and localisation of false data injection attacks in smart grids. *IET Cyber-Phys. Syst. Theory Appl.* **2024**, *9*, 313–333. [CrossRef]
20. Abur, A.; Gómez Expósito, A. *Power System State Estimation: Theory and Implementation*; Marcel Dekker: New York, NY, USA, 2004.
21. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef] [PubMed]
22. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
24. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. ITransformer: Inverted Transformers are effective for time series forecasting. In *Proceedings of the International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024*; Curran Associates, Inc.: Red Hook, NY, USA, 2024; pp. 1–25.
25. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 770–778.

26. New York Independent System Operator. Load Data. Available online: https://www.nyiso.com/load-data (accessed on 15 October 2024).
27. Wang, X.; Zhu, H.; Luo, X.; Guan, X. Data-Driven-Based Detection and Localization Framework Against False Data Injection Attacks in DC Microgrids. *IEEE Internet Things J.* **2025**, *12*, 36079–36093. [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.