A Novel AI Temporal-Spatial Analysis Approach for GNSS Error Source Recognition

Kit-Lun Tong*, Yi Ren*, Xin Shi†, Zhaohui Chen †, Xu Zhang*
*School of Computing Sciences, University of East Anglia, Norwich, United Kingdom
†CHC Tech Limited, Norwich, United Kingdom
*{k.tong, e.ren, x.zhang27}@uea.ac.uk, †{xin_shi, zoe_chen}@chcnav.com

***Abstract*—Global navigation satellite systems (GNSS) error source analysis is crucial for identifying factors that affect the accuracy of positioning, navigation, and timing services (PNT). Detecting and correcting these factors is essential for enhancing overall service accuracy. Traditional methods primarily focus on surface-level receiver output data, which may overlook underlying factors. Additionally, analyzing daily generated data is expensive and requires advanced proficiency. This research uses a novel temporal-spatial analysis approach to analyze GNSS error sources with artificial intelligence (AI) model support. We develop a noise segments dataset categorized into six types, with a particular focus on ionospheric disclosure, a deeper-level receiver data calculating PNT result. By applying clustering combined with a z-score normalization filter (ZFilter), we identify highly consistent noise segments in daily data, which aids in understanding potential causes. We then employ a multi-model deep learning approach to classify the noise segments, as opposed to relying on a single baseline model. Additionally, we experiment with semi-supervised learning through pseudo-labeling to improve classification performance. Our experiments show that our classifier achieves approximately 84% accuracy in identifying the noise segments.**

***Index Terms*—GNSS error source, PNT, Clustering, Deep learning**

I. Introduction

Global navigation satellite systems (GNSS), including the global positioning system (GPS), have been developed to offer comprehensive positioning, navigation, and timing (PNT) services with worldwide coverage. In these systems, L-band radio-frequency signals are transmitted from satellites and received by ground-based GNSS receivers. By processing these signals, the receivers calculate their distances from the observed satellites, enabling them to determine an accurate PNT solution. Nevertheless, the accuracy of PNT solutions heavily depends on the quality of the GNSS observable, which is affected by various GNSS error sources [1]. These include satellite clock and ephemeris errors, atmospheric delays, cycle slips, interference and jamming, etc. All of these errors can be expressed in units of distance, and must be detected and corrected to improve accuracy [2].

Conventional error detection methods encounter several limitations. Firstly, methods such as those described by [3] [4] typically focus on analyzing common receiver output parameters like elevation, observation data, signal-to-noise ratio (S/N), and PNT results. However, these methods are superficial compared to a more comprehensive analysis of the parameters involved in calculating PNT [5]. Next, the PNT system generates an enormous volume of data daily, making it highly challenging to extract consistent error segments. It is essential to identify these segments for diagnosing the root causes of errors and enhancing the overall accuracy of the PNT system [6] [7]. Lastly, when applying AI models for noise classification tasks, traditional works such as [3] and [8] primarily employ a single baseline model on mono-angle instances, which may be limited in their ability to capture the complexity of diverse noise types.

In this paper, we propose an artificial intelligence (AI)-based temporal-spatial approach for the automatic recognition of noise types in segments using a multi-model classification strategy. However, training these models typically requires manual labeling of noise data, which is both costly and demands significant expertise. To address this, we perform clustering to group highly similar noise segments, then apply a z-score normalization filtering (ZFilter) strategy to select the tightest cluster. This approach not only extracts segments with high consistency but also assists in building a pseudo-labeled dataset for model training.

We make the following contributions. First, rather than analyzing surface-level receiver data, we focus on ionospheric misclosure [9] [10], a deeper-level PNT parameter, to develop new GNSS error detection methods. We categorize six types of noise in our dataset, each representing different potential errors. Next, to analyze noise segments, we employ a temporal-spatial analysis approach that considers both time sequences and value distributions. Then, to identify consistent error segments within large volumes of daily data, we apply clustering, using the ZFilter strategy to pinpoint segments that closely resemble our reference records. Meanwhile, instead of relying on a single baseline model for automatic noise classification, we use multiple models to extract deep features and build a hybrid model for the classification task. Finally, to reduce the need for manually labeled data, we experiment with generating a pseudo-labeled dataset from the clustering model, aiming to enhance classification performance through semi-supervised learning.

The rest of this paper is organized as follows. Related works are surveyed in Section II. Section III introduces the noise types in our dataset and the AI temporal-spatial analysis approach. Section IV presents our evaluation results, followed by conclusions and future work in Section V.

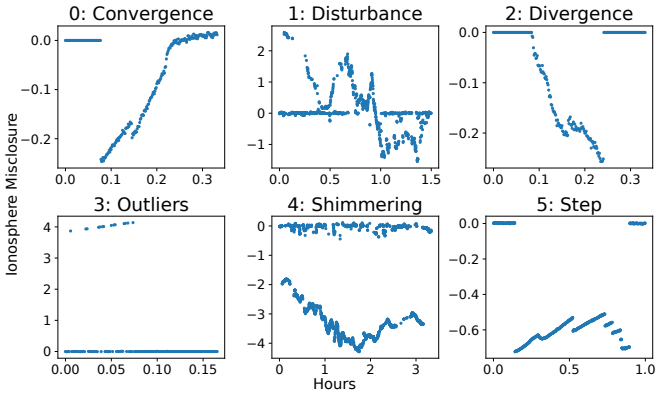

Fig. 1. The noise types in the GNSS error source dataset.

II. Related Works

Traditional research (e.g., [3], [4], [8], [11]) commonly considers surface-level data output by the receiver. The authors in [3] utilized elevation, S/N, and user speed as features in their machine-learning models to characterize multipath error distributions, [4] employed S/N for jamming detection, [8] analyzed signal strength and pseudorange residue for multipath detection, and [11] perform signal analysis for radio frequency interference (RFI). However, the surface-level data often provide an incomplete view compared to deeper-level data like [12], which employs the cross ambiguity function to detect GNSS spoofing. Here, we consider the ionosphere misclosure, which is a deeper-level parameter used to calculate the PNT result.

Previous studies (e.g., [3], [8], [11], [12]) primarily focus on single baseline models for AI-based classification tasks in GNSS error analysis. [3] employed a neural network to classify multipath noise, while [8] utilized a support vector machine. [11] classified RFI using a convolutional neural network (CNN). Similarly, [12] used a CNN to identify spoofing signals and a Gaussian mixture model to cluster and summarize the number of signals. In this research, we employ a multi-model classification approach to categorize noise segments based on different potential causes. Additionally, we experiment with clustering models using a ZFilter strategy to identify highly consistent noise segments and generate a pseudo-labeled dataset.

III. Noise Type and Methodology

In this research, we extract noise segments from the ionosphere misclosure, which refers to the discrepancies between the estimations and observations of the regional ionosphere. The noise segments are derived from data collected over four days by 53 ground stations and approximately 90 satellite units, including GPS, Galileo, and Beidou systems. We summarize six types of noise that frequently occur, as illustrated in Fig. 1. The scale and magnitude of the noise segments vary, with each type caused by different physical factors. Below, we present some potential examples:

- Convergence occurs as a result of model recalculation when tracking is lost or when clock or ephemeris errors are detected.

TABLE I
Class Sizes in the GNSS Error Source Dataset

	Labeled D^L						Unlabeled D^U	Total
Type (Y)	**0**	**1**	**2**	**3**	**4**	**5**		
Size	140	540	504	526	569	515	2114	**4906**

- Disturbance caused by interference from multiple systems or jamming by other equipment.
- Divergence arises from a mismatch between the error model and the observation.
- Outliers may result from incorrect carrier-phase ambiguity fixes.
- Shimmering can occur as a consequence of repeated carrier-phase ambiguity fixes, especially in the presence of atmospheric delays.
- Steps caused by cycle slips due to signal delays and distortions from ionospheric irregularities.

Denote our dataset $D = \{D_i\} = \{(X_i, Y_i)\}$ and the index set $I = \{i | i \in [0, |D|)\}$. The dataset D contains the noise segments $X = \{X_i\}$ and their corresponding labels $Y = \{Y_i\}$. Each segment is a sequence $X_i = (X_i[j] | j \in [0, |X_i|))$ consisting of ordered real-valued observations that may have different lengths, with $X_i[j] \in (-\infty, +\infty)$. We manually label a subset of the segments by their indices $I^L \subset I$ to create a labeled dataset $D^L = \{D_i | i \in I^L\}$, where the labels $Y_i \in [0, p)$ if $i \in I^L$, here $p = 6$ as shown in Fig. 1. The remaining indices $I^U = I \backslash I^L$ form an unlabeled dataset $D^U = \{D_i | i \in I^U\}$, with labels $Y_i = -1$ if $i \in I^U$. Table I lists the sizes of each class.

Fig. 2 illustrates our approach, which comprises three key components: a temporal pipeline for processing 1D sequential data, a spatial pipeline for handling 2D binary images, and a main pipeline that integrates both temporal and spatial information. Each pipeline follows four processing stages: Stage 1 (S1): preprocessing noise segments of varying sizes to standardize them into uniform input dimensions for the models; Stage 2 (S2): constructing a referral distance matrix (RDM) to extract similarity features between segments; Stage 3 (S3): clustering segments to identify consistent noise patterns and generate a pseudo-dataset with minimal manual labeling; Stage 4 (S4): training a classifier to identify different noise types. In the following sections, we will detail each of these stages.

A. Preprocessing (S1)

Since the range and length of each noise segment X_i can vary, it is necessary to normalize the range and standardize the length of the segments to ensure uniform contribution from each segment and maintain consistent characteristics across them. Given a sequence s of any size and length, we apply the min-max normalization to obtain $\widetilde{s}$:

$$\widetilde{s} = \frac{s - min(s)}{max(s) - min(s)} \tag{1}$$

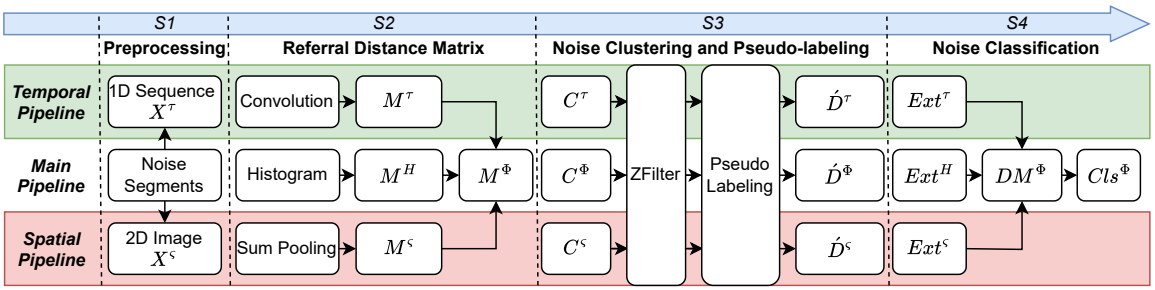

Fig. 2. Process pipeline of the temporal-spatial approach.

Furthermore, a uniform function UF is defined to standardize it into a length l:

$$UF(s,l) = \begin{cases} EP(\widetilde{s},l), & \text{if } |s| < l. \\ LS(\widetilde{s},l), & \text{otherwise.} \end{cases} \tag{2}$$

When $|s|$ is shorter than l, edge padding (EP) will be applied to extend the sequence. Otherwise, linear spacing (LS) will be applied to down-sample the sequence. Next, we divide the process into temporal, spatial, and main pipelines:

1) Temporal Pipeline: To standardize the noise segments of varying sizes and values, we apply (2) to uniform X_i into sequential data $X_i^\tau = UF(X_i, 128)$. Therefore, $X_i^\tau = \{X_i^\tau[j]\}^{1\times 128}$ where $X_i^\tau[j]|_{j\in[0,128)} \in [0,1]$.

2) Spatial Pipeline: We further transform X_i^τ into a 2D space to generate a binary image $X_i^\varsigma = \{X_i^\varsigma[h,w]\}^{128\times 128}$ where with height and width indices $h \in [0,128), w \in [0,128)$. Each pixel value $X_i^\varsigma[h,w] \in \{0,1\}$, enabling the extraction of distributional information.:

$$X_i^\varsigma[h,w] = \begin{cases} 1, & \text{if } h = \lfloor X_i^\tau[w] \times 127 \rfloor. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

3) Main Pipeline: By applying (1) to normalize each X_i into $\widetilde{X}_i$, a normalized segment set $\widetilde{X} = \{\widetilde{X}_i\}$ is obtained.

After preprocessing is complete, the min-max normalized segments $\widetilde{X}$, the temporal segments X^τ, and the spatial images X^ς are ready for the next stage.

B. Referral Distance Matrix (S2)

A global distance matrix compares the distances between segment pairs as a similarity feature during feature extraction. However, calculating all distances becomes inefficient when the dataset is large. We randomly select a subset of segments from labeled dataset as references $D^R \subseteq D^L$. The RDM $M|_{|D|\times|D^R|}$ can then be computed:

$$M(X, Dist)[i,r] = Dist(X_i, X_r^R) \tag{4}$$

where X_r^R is a reference segment, $r \in [0, |D^R|)$, and $Dist \in \{Eucl, DDTW\}$ is the metric used to compare distances. Depending on the pipelines, we apply either Euclidean distance ($Eucl$) or derivative dynamic time warping ($DDTW$) [13].

1) Temporal Pipeline: We use X^τ as the input. We apply convolution with a kernel $\nu = \{1\}|_{1\times 5}$ to smooth the sequence X_i^τ and further uniform it to $\widehat{X_i^\tau}|_{1\times 32}$ by (2). Therefore, $\widehat{X_i^\tau} = UF((X_i^\tau * \nu)_{|X_i^\tau|}, 32)$. After that, we compute the temporal RDM $M^\tau = M(\widehat{X^\tau}, DDTW)$ via (4), using $DDTW$ distance.

2) Spatial Pipeline: We use X^ς as the input. We apply sum-pooling (SP) [14] with a kernel of 16×16 to illustrate the distribution of the binary image X_i^ς, then flatten into $\widehat{X_i^\varsigma}|_{1\times 64}$, which $\widehat{X_i^\varsigma} = Flatten\left(\frac{SP(X_i^\varsigma)}{128}\right)$. After that, we compute the spatial RDM $M^\varsigma = M(\widehat{X^\varsigma}, Eucl)$ via (4), using Euclidean distance.

3) Main Pipeline: We directly extract a 10-bin histogram from $\widetilde{X}$ and compute the difference in value distribution using Euclidean distance.

$$M^H = M\left(\left\{\frac{Histogram(\widetilde{X_i})}{|\widetilde{X_i}|}\right\}, Eucl\right). \tag{5}$$

This is then concatenated with M^τ and M^ς to obtain the hybrid RDM $M^\Phi = [M^H, M^\varsigma, M^\tau]$.

The temporal RDM M^τ, the spatial M^ς, the histogram RDM M^H, and the hybrid RDM M^Φ are used as inputs for the remaining stage.

C. Noise Clustering and Pseudo-labeling (S3)

It is necessary to identify consistent noise to ensure accuracy and precision in PNT. However, the daily generation of large amounts of unlabeled data complicates the analysis process. By giving a small set of manually labeled examples, this data can be compared using the clustering approach to identify similar noise patterns. Additionally, pseudo-labels could be assigned to the unlabeled data, which would facilitate further training through semi-supervised learning.

Clustering is performed to group similar segments into clusters $C = \{C_k | k \in [0, |C|)\}$ along with the corresponding index set $I_k \subset I$. Here, $D_i \in C_k$ if a segment X_i belongs to the k-th cluster, which implies that $i \in I_k$ as well. Note that each segment belongs to only one cluster but some segments may not fit into any cluster (i.e. $k < 0$) and are excluded from consideration. Moreover, the number of clusters $|C|$ should be sufficiently large to ensure that the segments within each cluster are as similar as possible. M^τ, M^ς, and M^Φ are the inputs to the clustering algorithm, generating the

temporal cluster C^τ, spatial cluster C^ς, and hybrid cluster C^Φ, respectively.

To select the clusters with higher consistency segments, we apply ZFilter to identify more confident clusters. Firstly, the average intra-cluster distance δ_k is calculated to assess the tightness within C_k using the corresponding RDM features $M[i]$:

$$\delta_k = \frac{\sum_{i \in I_k} Eucl(M[i], \overline{M[I_k]})}{|I_k|} \tag{6}$$

where $|I_k|$ and $\overline{M[I_k]}$ denote the size and centroid of C_k, respectively. Following this, a z-normalized confidence score $Z_k \in [0,1]$ is computed based on δ_k for each C_k, along with the overall mean μ and standard deviation σ. The score $Z_k = CDF\left(\frac{\delta_k - \mu}{\sigma}\right)$ is normalized using the cumulative distribution function. A smaller Z_k indicates that the segments within the cluster C_k are more similar. Therefore, we can define a threshold $Z' \in [0,1]$ to filter the clusters.

A pseudo-labeled dataset $\acute{D} \subset D^U$, can also be generated from unlabeled dataset D^U using the ZFilter strategy applied to the labeled dataset D^L, thereby increasing the sample size during classification model training. For each cluster C_k, we compute the label score $LS_k|_{1\times 6}$, which represents the weightings for each noise type:

$$LS_k = \begin{cases} \sum_{i \in (I_k \cap I^L)} \ddot{Y}_i, & \text{if } Z_k \leq Z' \\ \{0\}|_{1\times 6}, & \text{otherwise.} \end{cases} \tag{7}$$

where $\ddot{Y}_i$ is the one-hot encoded Y_i. Finally, we construct the pseudo-labeled dataset as $\acute{D} = \bigcup_{k \in [0,|C|)} \{D_i | i \in (I_k \cap I^U)\}$ by ensuring that $Z_k \leq Z'$ and $\sum LS_k > 0$.

Label smoothing will also be applied to adjust the weighting of the pseudo-labels, helping to prevent the model from becoming overly confident in the predictions:

$$\ddot{Y}_i = \begin{cases} \ddot{Y}_i, & \text{if } i \in I^L \\ (1-\alpha) * \frac{LS_k}{\sum LS_k} + \frac{\alpha}{6} & \text{if } D_i \in \acute{D}, i \in I_k \\ \{0\}|_{1\times 6}, & \text{otherwise.} \end{cases} \tag{8}$$

where α is a hyperparameter that determines the amount of smoothing.

Afterward, we can generate the temporal pseudo-labeled dataset $\acute{D}^\tau$, the spatial pseudo-labeled dataset $\acute{D}^\varsigma$, and the hybrid pseudo-labeled dataset $\acute{D}^\Phi$, using C^τ, C^ς, and C^Φ, respectively.

D. Noise Classification (S4)

Traditionally, the baseline model outputs the classification result based on individual instances. In a multi-model approach, the deep features learned by the baseline model are extracted and concatenated to form a new instance, which is then used to train a hybrid model.

To classify noise segments, we experimented with various deep-learning models. Fig. 3 illustrates the architectures of our baseline models, including the multilayer perceptron (MLP) in Fig. 3a, long short-term memory (LSTM) Fig. 3b, and CNN in Fig. 3c, designed to process RDM, sequences, and binary images, respectively. While each model received different input types and had distinct hidden layer configurations, they all shared a common output structure: a rectified linear unit (ReLU) activated dense layer serving as a feature extractor, outputting a list of deep features denoted as $Ext|_{1\times 64}$, followed by a final softmax-activated dense layer that produces the classification result with probabilities denoted as $Cls|_{1\times 6}$, for each instance input.

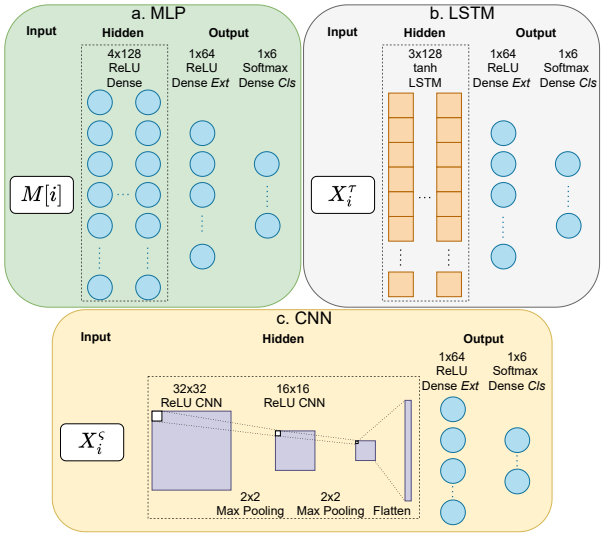

Fig. 3. The baseline classification models.

Each pipeline utilizes different baseline models. In the temporal pipeline, we input M^τ into the MLP or X^τ into the LSTM, the procedures are referred to as 'TMLP' or 'TLSTM', respectively. In the spatial pipeline, we input M^ς into the MLP or X^ς into the CNN, referred to as 'SMLP' or 'SCNN', respectively. In the main pipeline, we input M^H into the MLP, named 'HMLP'.

To perform temporal-spatial classification with multi-models, we use the deep features output from the baseline models: Ext^H from HMLP, Ext^τ from TMLP or TLSTM, and Ext^ς from SMLP or SCNN. These features are combined to generate a hybrid deep RDM, denoted as $DM^\Phi|_{|D|\times|D^R|}$:

$$\begin{aligned} DM^\Phi &= M(X^\Phi, Eucl), \\ \text{where } X^\Phi &= [Ext^H, Ext^\tau, Ext^\varsigma] \end{aligned} \tag{9}$$

Finally, we input DM^Φ into the MLP model to train a hybrid classifier, denoted as Cls^Φ as shown in Fig. 2. Combining the models in Fig. 3, four Cls^Φ are trained: 'TMLP_SMLP', 'TLSTM_SMLP', 'TLSTM_SCNN', and 'TMLP_SCNN'.

IV. Evaluation

The experiment runs on an Ubuntu 18.04 server with an R9-5950x CPU, 32GB RAM, and RTX3090 GPU. It uses Python 3.10 and Tensorflow 2.17. The RDM reference sizes are set to 60. All models in Fig. 3 employ categorical focal cross-entropy loss. The models are trained for 100 epochs with restored best weights based on loss.

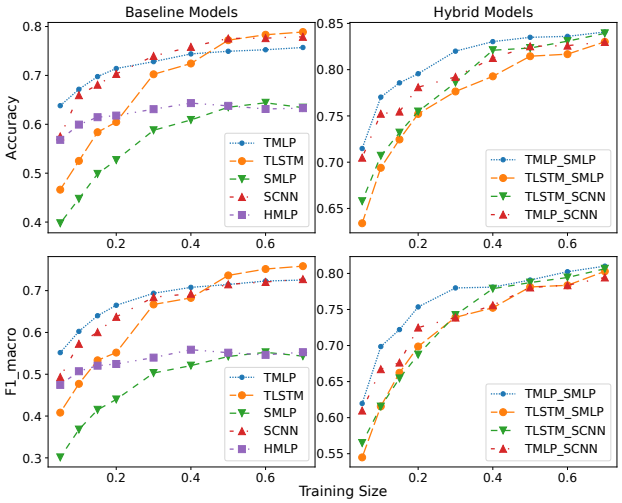

Fig. 4. The comparison of accuracy and F1-macro score among baseline and hybrid models using different training sizes.

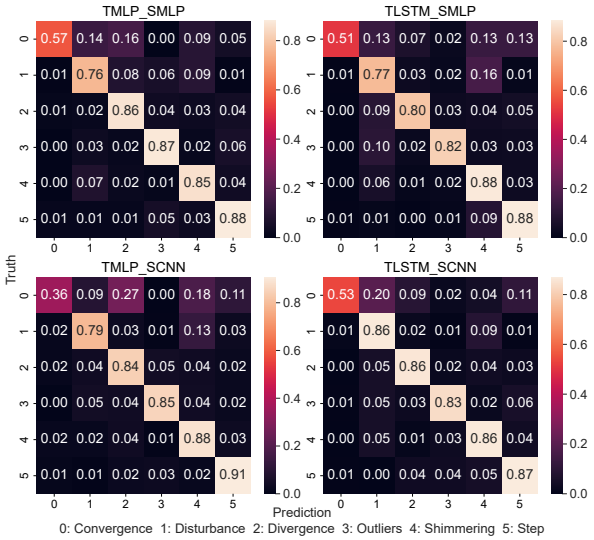

Fig. 5. Normalized confusion matrices of the hybrid models.

In the following section, we first evaluate the noise classifiers, then experiment with noise clustering, and finally test the classifiers using the pseudo-labeled dataset.

A. Evaluation of Baseline and Hybrid Noise Classification

Fig. 4 presents the average results from 10 trials based on classification accuracy and F1-macro score using only the labeled dataset. To evaluate the minimal manual labeling condition, the training size ranges from 0.05 to 0.7 of the labeled dataset, with the rest as the test set. In the baseline models, TMLP outperforms the others when the training size is below 0.3 for both metrics. The end-to-end methods, including TLSTM and SCNN, demonstrate better performance when the training size exceeds 0.5, with over 70% accuracy. Among the hybrid models, the RDM-based model 'TMLP_SMLP' leads the others in both metrics, achieving approximately 84% accuracy and 80% F1-macro score when the training size increases to 0.7. Meanwhile, the differences of the models diminish as the training size increases. Compared to the baseline models, hybrid models generally outperform by at least 5% in both accuracy and F1-macro, demonstrating that multi-model approaches can enhance classification performance.

Fig. 5 shows the normalized confusion matrices comparing the true labels with the predicted targets of the hybrid models when the training size is 0.7. According to the results, shimmering and step are generally easy for the models to detect. On the other hand, the predictions for convergence segments are less accurate due to dataset imbalance. Introducing data augmentation or Balanced Batch Sampling may help address this issue. Overall, 'TLSTM_SCNN' demonstrates more balanced predictions across the classes than the other models.

B. Evaluation of Noise Clustering

Table II presents the results of the clustering experiments by adjusting Z'. The training size is set to 0.2, and the testing data is mixed with the unlabeled data to generate the pseudo-labeling dataset. The evaluation focuses on two

TABLE II
THE COMPARISON OF ACCURACY AND DATA INCREMENT OF CLUSTERING MODELS.

Z'	*Model*	**Accuracy**			**Increment**		
		HDBSCAN	*Hierarchical*	*KMeans*	*HDBSCAN*	*Hierarchical*	*KMeans*
0.1	C^{τ}	0.958	0.974	**1**	0.378	**0.526**	0.172
	C^{ς}	**1**	0.992	**1**	0.498	**0.535**	0.401
	C^{Φ}	**1**	0.979	**1**	0.397	**0.543**	0.259
0.3	C^{τ}	**0.967**	0.95	0.916	0.523	**0.926**	0.535
	C^{ς}	0.973	0.95	**0.984**	0.689	**1.055**	0.695
	C^{Φ}	0.971	0.935	**1**	0.512	**0.965**	0.584
0.5	C^{τ}	**0.943**	0.88	0.897	0.664	**1.474**	1.229
	C^{ς}	**0.944**	0.867	0.894	0.87	**1.659**	1.263
	C^{Φ}	0.973	0.915	**0.986**	0.645	**1.523**	0.896
0.7	C^{τ}	**0.917**	0.86	0.856	0.833	2.025	**2.105**
	C^{ς}	**0.892**	0.822	0.851	1.132	**2.279**	2.145
	C^{Φ}	**0.942**	0.858	0.929	0.815	**2.462**	1.792
0.9	C^{τ}	**0.889**	0.799	0.795	1.084	2.707	**3.087**
	C^{ς}	**0.84**	0.78	0.765	1.611	3.32	**3.354**
	C^{Φ}	**0.903**	0.828	0.855	1.184	**3.279**	3.145

metrics: the accuracy of the labeled testing data and the overall increase in the number of generated pseudo-labels. We test three clustering models: HDBSCAN, hierarchical clustering, and KMeans. Both hierarchical clustering and KMeans are configured to cluster the segments into 1000 classes, which is close to the number of clusters generated by HDBSCAN.

According to Table II, as Z' increases, the overall number of pseudo-labels increases, but accuracy decreases. Considering the clustering models, HDBSCAN is more accurate, while hierarchical clustering generates more pseudo-labels. From the perspective of the pipelines, C^{Φ} is relatively more accurate, whereas C^{ς} generates more pseudo-labels. Overall, the noise segments extracted using the clustering method with ZFilter exhibit greater consistency than those obtained through classification.

C. Noise Classification Experiment Using Pseudo-Labeling

Fig. 6 displays the experimental results on noise classification using hybrid models with pseudo-labeling datasets

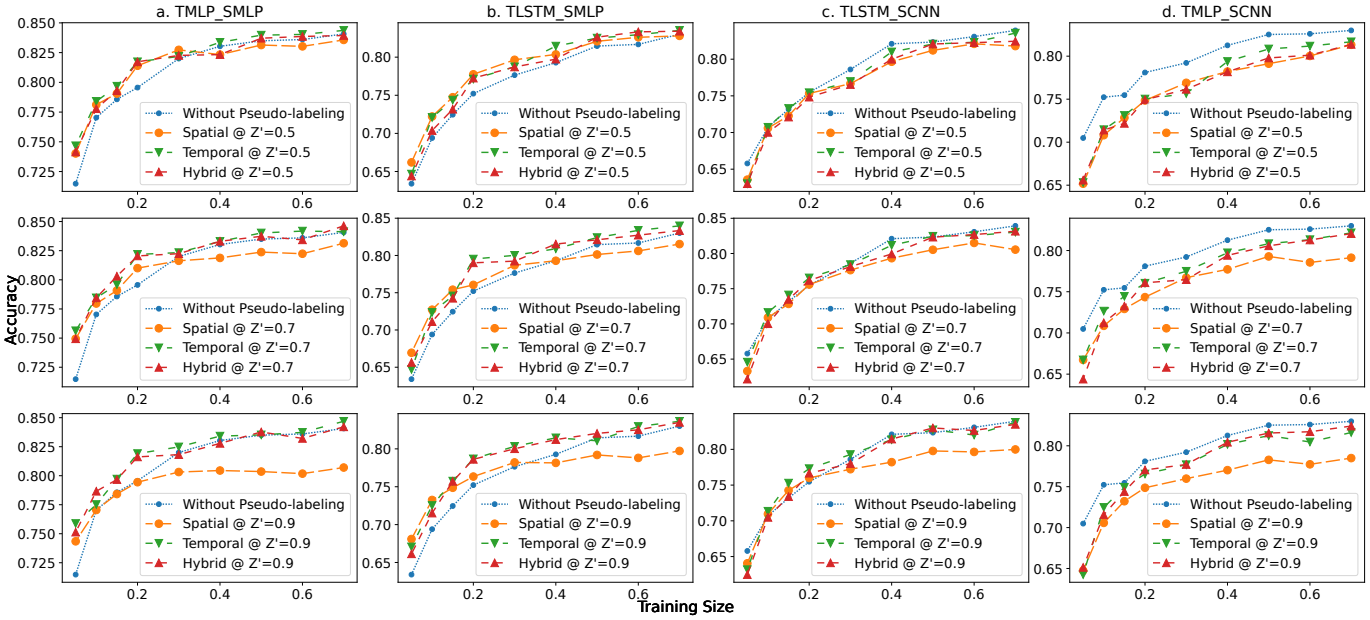

Fig. 6. Experimental results on noise classification with pseudo-labeling.

based on HDBSCAN. The α in (8) is set to 0.2. We compare Z' values of 0.5, 0.7, and 0.9. As shown in the results, pseudo-labeling enhances the performance of TMLP_SMLP and TLSTM_SMLP by approximately 3% when the training size is low. Notably, when the training size is less than 0.2, the accuracy of TMLP_SMLP increases from around 78% to over 82% when $Z' = 0.5$. Generally, the performance of $\acute{D}^{\tau}$ and $\acute{D}^{\Phi}$ outperforms $\acute{D}^{\varsigma}$. When comparing the case where $Z' = 0.9$ with the others, it generates more pseudo-labels, but this results in reduced performance due to the inclusion of more inaccurate data. The results indicate that ZFilter effectively selects the most similar noise segments, enhancing the model's performance when the labeled dataset is small. To further improve performance, incorporating additional unlabeled data can help generate more pseudo-labels.

V. Conclusions

In conclusion, we developed an innovative AI approach utilizing temporal-spatial features for GNSS error source analysis. Our noise segments dataset is based on regional ionospheric misclosure, which is derived from deep-level receiver data rather than the traditional surface-level data. To handle the large volume of daily data, we applied clustering along with the ZFilter to extract consistent noise segments, which also creates a pseudo-labeled dataset to improve performance by around 3% in low-training-data scenarios. Our hybrid classification model achieved an accuracy of 84% in identifying noise types within segments, outperforming the common baseline models by at least 5%.

There are several potential future research directions. More deep-level parameters, such as orbit clock update residuals and tropospheric misclosure, can be considered to achieve a deeper characterization of error sources, including multipath interference, tropospheric delays, and receiver clock errors. Error forecasting can be performed by considering additional factors like ionospheric activity and tropospheric conditions. Consistent noise data can also be used to validate and enhance existing GNSS error models.

References

[1] A. Leick, L. Rapoport, and D. Tatarnikov, *GPS satellite surveying*, 4th ed. Wiley, 2015.

[2] P. Teunissen and O. Montenbruck, *Springer Handbook of Global Navigation Satellite Systems*. Springer, 2017.

[3] H. No and C. Milner, "Machine learning based overbound modeling of multipath error for safety critical urban environment," in *Proc. 34th. ION GNSS+ 2021*, 10 2021.

[4] S. Jada, M. Psiaki, S. Landerkin, S. Langel, A. Scholz, and M. Joerger, "Evaluation of PNT situational awareness algorithms and methods," in *Proc. 34th. ION GNSS+*, 2021, pp. 816–833.

[5] W. Stock, R. T. Schwarz, C. A. Hofmann, and A. Knopp, "Survey on opportunistic PNT with signals from LEO communication satellites," *IEEE Commun. Surv. Tutor.*, pp. 1–1, 2024.

[6] J. Zidan, O. Alluhaibi, E. I. Adegoke, E. Kampert, M. D. Higgins, and C. R. Ford, "3D mapping methods and consistency checks to exclude GNSS multipath/NLOS effects," in *Proc. UCET*, 2020, pp. 1–4.

[7] R. Sun, L. Fu, Q. Cheng, K.-W. Chiang, and W. Chen, "Resilient pseudorange error prediction and correction for GNSS positioning in urban areas," *IEEE Internet Things J.*, vol. 10, pp. 9979–9988, 2023.

[8] L.-T. Hsu, "GNSS multipath detection using a machine learning approach," in *Proc. 20th ITSC*, 2017, pp. 1–6.

[9] S. Schaer, G. Beutler, L. Mervart, M. Rothacher, and U. Wild, "Global and regional ionosphere models using the GPS double difference phase observable," in *Proc. IGS Workshop*, 1995, pp. 77–92.

[10] Z. Nie, P. Zhou, F. Liu, Z. Wang, and Y. Gao, "Evaluation of orbit, clock and ionospheric corrections from five currently available SBAS L1 services: Methodology and analysis," *Remote Sens.*, vol. 11, no. 4, 2019.

[11] A. Elango, S. Ujan, and L. Ruotsalainen, "Disruptive GNSS signal detection and classification at different power levels using advanced deep-learning approach," *Proc. ICL-GNSS*, pp. 1–7, 2022.

[12] P. Borhani-Darian, H. Li, P. Wu, and P. Closas, "Detecting GNSS spoofing using deep learning," *EURASIP J. Adv. in Sig. Pr.*, vol. 2024, no. 1, 1 2024.

[13] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. SDM*, 2001.

[14] N. Akhtar and U. Ragavendran, "Interpretation of intelligence in CNN-pooling processes: a methodological survey," *Neural Comput. and Appl.*, vol. 32, no. 3, pp. 879–898, 7 2019.