Retrieving Time-Series Differences Using Natural Language Queries

Kota Dohi, Tomoya Nishida, Harsh Purohit, Takashi Endo, Yohei Kawaguchi
Research and Development Group, Hitachi, Ltd.

***Abstract*—Effectively searching time-series data is essential for system analysis; however, traditional methods often require domain expertise to define search criteria. Recent advancements have enabled natural language-based search, but these methods struggle to handle differences between time-series data. To address this limitation, we propose a natural language query-based approach for retrieving pairs of time-series data based on differences specified in the query. Specifically, we define six key characteristics of differences, construct a corresponding dataset, and develop a contrastive learning-based model to align differences between time-series data with query texts. Experimental results demonstrate that our model achieves an overall mAP score of 0.994 in retrieving time-series pairs.**

***Index Terms*—Time series analysis, time-series retrieval, multi modality, contrastive learning**

I. Introduction

The state of any system can be represented as time-series data consisting of one or multiple channels. However, for system analysis, the most critical data often pertains to specific channels over limited time periods. Therefore, techniques for efficiently searching necessary data are essential [1], [2].

Traditional time-series data search required expertise in defining search criteria and designing similarity measures [3]–[5]. This reliance on specialized knowledge made these methods inaccessible to non-experts, limiting their usability in broader contexts. Recent studies have addressed this issue by using natural language queries for searching time-series data [6], [7]. These methods allow users to retrieve time-series data by intuitively specifying language queries that describe the shape or trend of the data, such as "The signal has an upward trend." However, a major limitation is their inability to search based on differences between multiple time-series, which are often essential in real-world applications. For instance, in industrial applications, identifying deviations from normal operating conditions often requires comparing current sensor data to historical data. Consequently, existing language query methods are inadequate in practical applications.

To address this limitation, this study proposes a natural language query-based approach for retrieving pairs of time-series data with query-specified differences. First, we define six key characteristics of differences between reference data and target data: upward trend, downward trend, spike, dropout, noise, and baseline. Based on these characteristics, we prepare pairs of time-series data and corresponding query texts, where each pair represents one of the defined characteristics. Second, we develop a contrastive learning method to align the characteristics of differences between the reference and target data with query texts. Finally, we evaluate the trained model on test data and examine its ability to retrieve pairs of reference and target data based on differences specified in natural language queries.

II. Relation to prior work

The task of retrieving specific time-series data from large datasets has been extensively studied in the field of data mining [3]–[5]. These studies typically used time-series data as the query and searched for time-series data with high similarity scores. However, defining an appropriate similarity metric requires careful consideration of the domain and use case, which often demands specialized knowledge. This reliance on expertise poses challenges for non-experts and limits the accessibility of these methods [6]. To tackle this limitation, natural language-based search methods have been explored for time-series retrieval [6], [7]. These methods allow users, including non-experts, to intuitively define search criteria by describing trends or patterns in human-interpretable terms. However, they cannot handle differences between multiple time-series. To address this, this study proposes a method for retrieving pairs of time-series data based on differences specified in the language query.

III. Problem statement

The reference data x_{ref} is time-series data that serves as the basis for comparison, while the target data x_{tgt} is the data being compared. The natural language query q describes the differences between the corresponding pairs of x_{ref} and x_{tgt}.

Given the query q, the task is to retrieve a set of corresponding pairs of $(x_{\text{ref}}, x_{\text{tgt}})$ from a set of N pairs. Formally, the objective is to find $D_q \subseteq D$, where:

$$D = \{(x^i_{\text{ref}}, x^i_{\text{tgt}}) \mid i = 1, 2, \ldots, N\},$$

$$D_q = \{(x^i_{\text{ref}}, x^i_{\text{tgt}}) \in D \mid \text{matches descriptions in } q\}.$$

To evaluate whether a pair $(x^i_{\text{ref}}, x^i_{\text{tgt}})$ matches the query q, we compute a similarity score s_i based on the query and the pair's characteristics. The similarity score is defined as:

$$s_i = f(x^i_{\text{ref}}, x^i_{\text{tgt}}, q),$$

where $f(\cdot)$ measures the extent to which the pair $(x^i_{\text{ref}}, x^i_{\text{tgt}})$ aligns with the descriptions provided in the query q.

The pairs in D are ranked in descending order of their similarity scores s_i. The retrieved set D_q consists of the top k pairs, formally defined as:

$$D_q = \{(x^{i_j}_{\text{ref}}, x^{i_j}_{\text{tgt}}) \mid j = 1, 2, \ldots, k\},$$

where $i_1, i_2, \ldots, i_k$ are the indices corresponding to the top k similarity scores $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$.

IV. Preparation of Dataset

A. Key Characteristics of Differences between Time-Series

There are numerous ways to describe the differences between two time-series. In this study, however, we focus on six key characteristics of differences: upward trend, downward trend, spike, dropout, noise, and baseline. These characteristics are selected based on their importance in capturing changes commonly observed in real-world time-series datasets.

Upward / Downward trend: Refers to the extent to which the data consistently increases or decreases over time. This represents the overall strength of the trend across all time points.

Spike / Dropout: Refers to the magnitude of short-term fluctuations. Variations in spikes or dropouts may reflect the severity of anomalies, sudden events, or transient disturbances within a system.

Noise: Refers to the level of random fluctuations in time-series data. Increased noise often indicates higher measurement uncertainty, sensor malfunctions, or unstable environmental conditions.

Baseline: Refers to the mean value of time-series data. Changes in baseline may result from sensor calibration errors, variations in operational conditions, or persistent deviations from expected values.

In this study, the reference data and target data pairs are assumed to differ in only one of these characteristics.

B. Generation of Synthetic Time-Series Data Pairs

No publicly available dataset contains pairs of time-series data with differences in the characteristics defined in IV-A. Therefore, we synthetically generate reference-target data pairs from real-world time-series data.

We first normalize the data using min-max scaling to ensure that all values fall within the range $[0, 1]$. Min-max scaling standardizes the data by uniformly adjusting its range, facilitating stable and efficient model training.

After normalization, we randomly select one of the six characteristics and apply the corresponding perturbation to the data.[1] The process of adding perturbation for each characteristic can be described as follows:

Upward / Downward trend: A linear trend is added to the data. Given a trend magnitude α and original time-series data x_{orig}, we define:

$$x_{\text{ref/target}}(t) = \begin{cases} x_{\text{orig}}(t) + \alpha t, & \text{(Upward trend)} \\ x_{\text{orig}}(t) - \alpha t. & \text{(Downward trend)} \end{cases}$$

To ensure that the perturbation does not result in a simple magnitude relationship between the reference and target data, we apply min-max scaling after adding the perturbation.

[1]If the upward trend is selected but the slope of the data is negative, the characteristic is switched to a downward trend. Similarly, if the downward trend is selected but the slope is positive, the characteristic is switched to an upward trend.

TABLE I
Perturbation Parameters Used for Reference-Target Pair Generation

Parameter name	Smaller perturbation level	Larger perturbation level
Trend magnitude α	$0.0 - 0.5$	$0.5 - 1.0$
Spike magnitude β	$0.0 - 0.1$	$0.1 - 0.5$
Noise level γ	$0.0 - 0.05$	$0.05 - 0.1$
Baseline shift θ	$0.0 - 0.1$	$0.1 - 0.5$

Spike / Dropout: A spike (sudden increase) or dropout (sudden decrease) is introduced at a randomly selected time t_s. Given a magnitude β, we define:

$$x_{\text{ref/target}}(t) = \begin{cases} x_{\text{orig}}(t) + \beta\delta(t - t_s), & \text{(Spike)} \\ x_{\text{orig}}(t) - \beta\delta(t - t_s), & \text{(Dropout)} \end{cases}$$

where $\delta(t - t_s)$ is the Dirac delta function at t_s.

Noise: Random Gaussian noise with the noise level γ is added to the target data:

$$x_{\text{ref/target}}(t) = x_{\text{orig}}(t) + \gamma v_{\text{noise}}(t),$$

$$v_{\text{noise}}(t) \sim \mathcal{N}(0, 1).$$

Baseline: A constant baseline shift factor θ is added to the entire time series:

$$x_{\text{ref/target}}(t) = x_{\text{orig}}(t) + \theta.$$

To introduce various relative differences between the reference and target data, we prepare two levels of perturbation: a larger and a smaller perturbation level. The parameters $(\alpha, \beta, \gamma, \theta)$ for these perturbations are randomly sampled within the ranges specified in Table I. These perturbations are applied to the same time-series data to generate two time-series, one of which is randomly assigned as the reference data and the other as the target data.

Depending on the characteristic of difference and whether the target data is subjected to a larger or smaller perturbation, there are 12 possible relationships between the reference and target data. Each reference-target pair is assigned a label y_s from 1 to 12 to represent its relationship.

C. Generation of Query Texts

We generate query texts for reference-target data pairs using the following template:

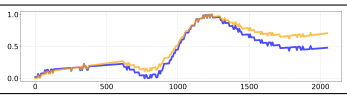
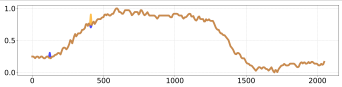
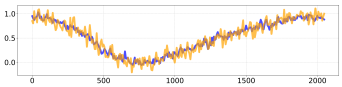
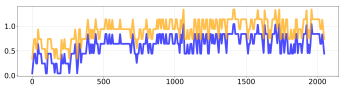
> *The target data has* {*direction*} {*characteristic of difference*} *than the reference data.*

Here, {*direction*} is either *larger* or *smaller*, indicating whether the target data is subjected to a larger or smaller perturbation. {*characteristic of difference*} corresponds to one of the six predefined characteristics.

Once the template is instantiated with specific values, we rephrase the resulting texts using the Azure OpenAI API with the GPT-4o model [8]. This process enhances the naturalness and diversity of the query texts while preserving their core semantic meaning.

TABLE II

EXAMPLES OF QUERIES AND THEIR ASSOCIATED REFERENCE-TARGET DATA PAIRS. IN THE PLOTS, THE ORANGE LINE REPRESENTS THE TARGET DATA, AND THE BLUE LINE REPRESENTS THE REFERENCE DATA

Query text	Reference-target data pair
The target data exhibits a stronger trend of increase than the reference data.	
The target dataset contains a significantly higher surge compared to the reference dataset.	
The target dataset contains a greater degree of distortion relative to the reference dataset.	
The baseline present in the target dataset is noticeably higher than that of the reference dataset.	

Each query text is assigned a label y_t according to the relationship between the reference and target data it describes, similar to the labeling method for reference-target data pairs. These labels are then used to generate the training dataset, consisting of reference-target data pairs and their corresponding query texts. Table II shows examples of query texts and their associated reference-target data pairs.

V. MODEL FOR ALIGNING TIME-SERIES DATA AND QUERY TEXTS

Fig. 1 illustrates the overview of our model for aligning time-series data with query texts. The model encodes reference-target data pairs and query texts into a shared representation space and leverages contrastive learning for alignment. The framework comprises three key components: signal and text encoders, a method for merging signal embeddings, and supervised contrastive learning across time-series and text data.

A. Encoding Pairs of Time-Series Data and the Query Texts

First, the reference data x_{ref} and target data x_{tgt} are encoded into z_{ref} and z_{tgt} using a common signal encoder. Optionally, cross-attention can be applied between z_{ref} and z_{tgt} to effectively model the relative correlations between the reference and target data. The z_{ref} and z_{tgt} are then merged and passed into the projection head, which adjusts its dimensionality to align with that of the query texts. We prepare two methods for merging z_{ref} and z_{tgt}: taking their difference (*diff*) and concatenating them along the embedding dimension (*concat*).

The query texts are also encoded using a text encoder, and the embedding is passed into the projection head.

B. Supervised Contrastive Learning with Relationship Labels

Let us denote the signal and text embeddings obtained from a batch of N samples by

$$Z_{\text{signal}} = \left[z_{s1},\, z_{s2},\, \ldots,\, z_{sN}\right] \quad \in \mathbb{R}^{N\times d},$$
$$Z_{\text{text}} = \left[z_{t1},\, z_{t2},\, \ldots,\, z_{tN}\right] \quad \in \mathbb{R}^{N\times d}.$$

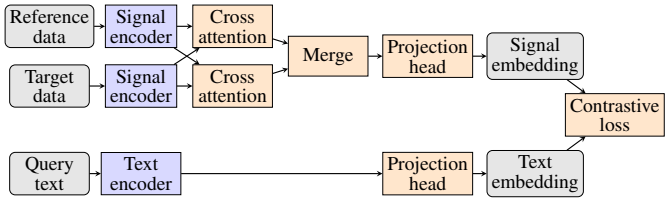


Fig. 1. Overview of the model architecture for aligning time-series data with query texts.

Here, d denotes the embedding dimensionality. Each z_{si} corresponds to the embedding of the i-th reference-target data pair, while each z_{ti} corresponds to the embedding of the i-th query text.

Both the signal and text embeddings are first normalized using L2 normalization. Then, we compute a logit matrix $M \in \mathbb{R}^{N\times N}$ whose entries measure the similarity between text and signal embeddings. Specifically,

$$M_{ij} \;=\; (z_{ti} \cdot z_{sj})/\tau,$$

where $\tau > 0$ is a temperature parameter that controls the scale of the logits, and $z_{ti} \cdot z_{sj}$ denotes the dot product of the i-th text embedding and the j-th signal embedding.

Based on the labels for each reference-target data pair (y_{si}) and each query text (y_{ti}), we construct a target matrix $G \in \mathbb{R}^{N\times N}$ such that

$$G_{i,j} \;=\; \begin{cases} 1 & \text{if } y_{ti} = y_{sj}, \\ 0 & \text{otherwise.} \end{cases}$$

Viewed row-by-row, this matrix captures which reference-target data pairs match which query texts. We then normalize each row by dividing all elements in that row by their collective sum, which ensures that each row has values adding up to 1. Hence, each row can be interpreted as a probability distribution over signal embeddings.

Let $\arg\max\big(G_{i,:}\big)$ denote the index of the largest entry in the i-th row of G. We define two cross-entropy losses. The first treats each row of M as a predictive distribution over signal embeddings for a given text embedding:

$$L_{\text{texts}} \;=\; \text{CE}\Big(M,\; \arg\max(G)\Big),$$

while the second treats each column (i.e., $M^\top$) as a predictive distribution over text embeddings for a given signal embedding:

$$L_{\text{signals}} \;=\; \text{CE}\Big(M^\top,\; \arg\max(G)\Big).$$

Here, $\text{CE}(\cdot,\cdot)$ denotes the standard cross-entropy loss function, computed along rows of the logit matrix (or its transpose).

Finally, we average these two losses to obtain the overall contrastive objective:

$$L_{\text{contrastive}} \;=\; \frac{L_{\text{texts}} + L_{\text{signals}}}{2}.$$

TABLE III
MAP SCORES FOR EACH RELATIONSHIP BETWEEN THE REFERENCE AND TARGET DATA. "TOTAL" DENOTES THE OVERALL MAP SCORE

Method			Characteristic of difference												Total
Signal encoder	Merge method	Cross attention	Upward		Downward		Spike		Dropout		Noise		Baseline		
			larger	smaller	larger	smaller	larger	smaller	larger	smaller	larger	smaller	larger	smaller	
1D-CNN	concat		0.770	0.610	0.562	0.659	0.687	0.697	0.522	0.497	**1.000**	0.995	**1.000**	0.969	0.747
1D-CNN	diff		0.448	0.494	0.637	0.443	0.912	0.969	0.983	0.956	**1.000**	**1.000**	**1.000**	0.969	0.818
1D-CNN	concat	✓	**1.000**	0.991	**1.000**	**1.000**	0.613	0.491	0.471	0.516	0.987	0.956	**1.000**	**1.000**	0.835
1D-CNN	diff	✓	0.865	0.872	0.860	0.794	0.634	0.321	0.358	0.456	0.966	0.911	**1.000**	**1.000**	0.753
Informer	concat		**1.000**	**1.000**	**1.000**	**1.000**	0.815	0.768	0.847	0.745	**1.000**	**1.000**	0.993	**1.000**	0.931
Informer	diff		**1.000**	**1.000**	**1.000**	**1.000**	**0.975**	**0.974**	**1.000**	**0.980**	**1.000**	**1.000**	**1.000**	**1.000**	**0.994**
Informer	concat	✓	**1.000**	**1.000**	**1.000**	**1.000**	0.969	0.956	**1.000**	0.962	**1.000**	**1.000**	0.993	**1.000**	0.990
Informer	diff	✓	**1.000**	0.968	**1.000**	**1.000**	0.973	0.961	0.994	0.988	**1.000**	**1.000**	**1.000**	**1.000**	0.990

VI. EXPERIMENTS

A. Dataset

We used the TACO dataset [9] for generating pairs of reference data and target data. The TACO dataset consists of 1.2 million sensor time-series. First, we linearly interpolated each time-series so that all time-series have 2,048 points. Each time-series was then normalized using min-max scaling. Next, we extracted time-series one by one and created pairs of reference data, target data, and their corresponding labels, as described in IV-B. In total, we generated 100,000 pairs for the training dataset, 2,000 pairs for the validation dataset, and 400 pairs for the test dataset.

The query texts were generated following the procedure described in IV-C. For each of the 12 relationships between reference and target data, we created 1,000 unique query texts. These query texts were then randomly divided into 900 for the training dataset and 100 for the test. The label was also assigned for each query text. For each pair of reference-target data and its label, we randomly selected one of the query text with the same label and assigned it to the pair.

B. Experimental conditions

For the signal encoder, we employed 1D-CNN [10], [11] and Informer [12]. For the text encoder we used BART-Large-XSum [13], [14] [2], T5-base [15][3], and RoBERTa-large [16][4]. The cross attention module was implemented using a multi-head attention mechanism with eight attention heads. The projection head is a multi-layer perceptron consisting of two linear layers with a GELU activation, dropout, a residual connection, and layer normalization. During training, we set the batch size to 512. The learning rates were set as follows: 1×10^{-3} for the projection head, 1×10^{-5} for the cross attention module and signal encoder, and 1×10^{-4} for the text encoder. We trained the model for 100 epochs and saved the version that achieved the minimum validation loss. The temperature parameter for contrastive learning was set to 1.0.

[2]https://huggingface.co/facebook/bart-large-xsum
[3]https://huggingface.co/google-t5/t5-base
[4]https://huggingface.co/FacebookAI/roberta-large

C. Evaluation

To evaluate the retrieval performance of the model we described in V, we used the mAP score [17]. First, for each relationship between the reference and target data, we converted 100 test query texts into text embeddings. We also converted 400 test pairs of reference and target data into signal embeddings. Then, for each query's text embedding, we computed the cosine similarities with the signal embeddings. Using the obtained similarities and the labels assigned to each reference-target data pair and query text, we calculated the mAP score for each of the twelve relationships. We also calculated the overall mAP score across all relationships.

D. Results

We first conducted experiments with eight different configurations of signal encoders, merge methods, and cross-attention. In these experiments, the Bart-Large-XSum was used as the text encoder. During training, the signal encoder, cross-attention modules, and projection heads were trained from scratch, and the text encoder's parameters were unfrozen for fine-tuning. Table III presents the mAP scores for each type of relationships between reference and target data. The highest overall mAP score of 0.994 was achieved with the Informer, the *diff* merge method, and without cross-attention. Other configurations using the Informer with cross-attention also achieved comparable performance. The Informer consistently outperformed the 1D-CNN across most configurations, which can be attributed to its stronger ability to model long-term dependencies in sequences. Unlike characteristics such as *noise* or *baseline*, differences in *upward / downward trend* cannot be captured solely through local relationships. This limitation likely contributed to the lower performance of the standalone 1D-CNN. However, the application of cross-attention to the 1D-CNN significantly improved its performance on *upward / downward trend*, highlighting the importance of mechanisms that account for global dependencies. The *diff* merge method excelled in retrieving *spike* and *dropout*, particularly when used without cross-attention. This may be attributed to the fact that computing the differences between the embeddings of the reference and target data effectively emphasizes the unique

TABLE IV
OVERALL MAP SCORES WITH DIFFERENT TEXT ENCODERS, FROZEN OR UNFROZEN TEXT ENCODER PARAMETERS, AND SUPERVISED OR SELF-SUPERVISED CONTRASTIVE LEARNING

Text encoder	Frozen / self-supervised	Unfrozen / self-supervised	Frozen / supervised	Unfrozen / supervised (Ours)
BART-Large-XSum	0.208	0.226	0.986	**0.994**
T5-base	0.193	0.210	0.990	**0.992**
RoBERTa-large	0.214	0.208	0.989	**0.994**

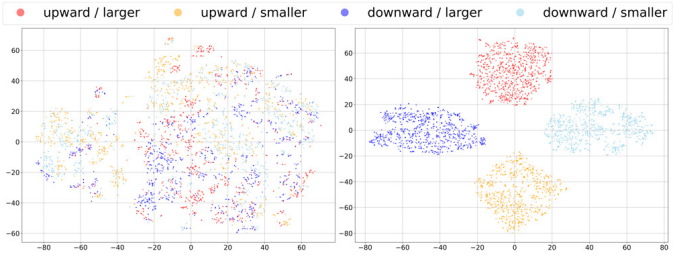


Fig. 2. t-SNE plots of query text embeddings. **Left:** Embeddings generated by the pre-trained BART-Large-XSum model. **Right:** Embeddings generated by the fine-tuned BART-Large-XSum model.

features of the differences. The cross-attention improved the overall performance for the *concat* merge method. However, its impact on other configurations was less consistent.

We also conducted experiments with various text encoders to compare their performance. In addition, we explored configurations combining self-supervised contrastive learning with frozen text encoder parameters. This was motivated by our previous works on contrastive learning between time-series data and text [7], [9], where the text encoder parameters were frozen during training, and the self-supervised contrastive learning framework proposed in [18] was employed.

Table IV summarizes the overall mAP score for each configuration. In these experiments, the signal encoder was set to Informer, the merge method was *diff*, and cross-attention was not employed. The term "supervised" refers to the supervised contrastive learning framework described in V-B, while "self-supervised" denotes self-supervised contrastive learning, which does not utilize labels. Across all text encoders, the "unfrozen / supervised" configuration consistently achieved the highest performance. Conversely, the performance of self-supervised configurations was generally lower. Fig. 2 presents t-SNE plots of 4,000 query text embeddings representing four different relationships. The left-hand side plot, which depicts text embeddings from the pre-trained BART-Large-XSum model, fails to distinguish embeddings corresponding to different relationships. In contrast, the right-hand side plot, generated using BART-Large-XSum fine-tuned with the "unfrozen / supervised" configuration, successfully separates embeddings for all relationships. These plots illustrate that pre-trained text encoders are not well-optimized for capturing nuanced differences in relationships, which led to the failure of self-supervised contrastive learning.

VII. CONCLUSION

We proposed a natural language query-based method for retrieving pairs of time-series data with specified differences. We first created pairs of time-series and query texts based on six key characteristics of differences. We then developed contrastive learning methods for aligning differences in time-series data with query texts. Evaluation with test data showed overall mAP score of 0.994, confirming accurate retrieval across all characteristics of differences.

REFERENCES

[1] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[2] C. Kleist, "Time series data mining methods: A review," Master's Thesis, Humboldt-Universität zu Berlin, School of Business and Economics, 2015.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *SIGMOD Rec.*, vol. 23, no. 2, p. 419–429, 1994.

[4] E. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[5] T. Nakamura, K. Taki, H. Nomiya, K. Seki, and K. Uehara, "A shape-based similarity measure for time series data with ensemble learning," *Pattern Analysis and Applications*, vol. 16, no. 4, pp. 535–548, 2013.

[6] S. Imani, S. Alaee, and E. Keogh, "Putting the human in the time series analytics loop," in *Proc. World Wide Web Conference (WWW)*, 2019, p. 635–644.

[7] A. Ito, K. Dohi, and Y. Kawaguchi, "CLaSP: Learning concepts for time-series signals from natural language supervision," *arXiv:2411.08397*, 2024.

[8] OpenAI, J. Achiam *et al.*, "GPT-4 technical report," *arXiv:2303.08774*, 2023.

[9] K. Dohi, A. Ito, H. Purohit, T. Nishida, T. Endo, and Y. Kawaguchi, "Domain-independent automatic generation of descriptive texts for time-series data," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, to appear.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1578–1585.

[12] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[13] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 7871–7880.

[14] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 1797–1807.

[15] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized bert pretraining approach," *arXiv:1907.11692*, 2019.

[17] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *arXiv:2103.00020*, 2021.