SEED: An Effective Model for Highly-Skewed Streamflow Time Series Data Forecasting

Yanhong Li
Computer Science and Engineering
Santa Clara University
Santa Clara, CA, USA
yli20@scu.edu

Jack Xu
Santa Clara Valley Water District
Santa Clara, CA, USA
jxu@valleywater.org

David C. Anastasiu
Computer Science and Engineering
Santa Clara University
Santa Clara, CA, USA
danastasiu@scu.edu

***Abstract*—Accurate time series forecasting is crucial in various domains, but predicting highly-skewed and heavy-tailed univariate series poses challenges. We introduce the Segment-Expandable Encoder-Decoder (SEED) model, designed for such time series. SEED incorporates segment representation learning, Kullback-Leibler divergence regularization, and an importance-enhanced sampling policy. We tested our model on the 3-day ahead single-shot prediction task on four hydrologic datasets. Experimental results demonstrate SEED's effectiveness in optimizing the forecasting process (*10–30%* of root mean square error reductions over state-of-the-art methods), underlining its notable potential for practical applications in univariate, skewed, long-term time series prediction tasks.**

***Index Terms*—deep learning, representation learning, sampling policy, streamflow prediction, time series**

I. Introduction

Time series forecasting is vital in a myriad of sectors, spanning traffic management [1], meteorology [2], biology [3], financial markets [4], and water resources [5]. Yet, the task becomes hard when faced with datasets having pronounced skewness, complicating accurate long-term predictions. Taking hydrology as an example, streamflow predictions are convoluted due to multifaceted and unpredictable variables like weather patterns, geographical features, and human activities. Such intricacies heighten the challenge of obtaining accurate forecasts in this field.

Forecasting from highly-skewed [6] and heavy-tailed datasets presents a myriad of challenges. Such datasets typically offer limited training samples for extreme values, impeding models from discerning their underlying patterns. The disproportionate data distribution, with most values clustered at the lower end and few at the higher extremities, can compromise the efficacy of conventional prediction algorithms. Furthermore, the anomalous events in these datasets might adhere to distinct distributions divergent from the main data bulk, necessitating tailored techniques to address their non-Gaussian attributes. The intricacies are further compounded by the presence of long-range dependencies [7], demanding models that can track the impact of historical observations over extensive time intervals.

Current methods [7]–[10] excel in forecasting normally distributed data; however, their accuracy decreases considerably with highly-skewed time series. Our contributions include:

- The novel Segment-Expandable Encoder-Decoder (SEED) model, which is the first to integrate segment representation learning with a multi-tiered encoder-decoder framework, specifically designed for complex datasets with intricate characteristics.
- A unique regularization strategy that incorporates a Kullback-Leibler divergence regularization loss term across multiple stacked layers, thereby increasing the model's robustness against anomalous events with divergent distributions.
- An innovative importance-enhanced sampling strategy embedded within the SEED model, allowing it to skillfully identify key features and trends in datasets, leading to improved forecasting accuracy in the presence of heavily skewed time series data.
- Comprehensive experiments that demonstrate SEED's effectiveness in optimizing the forecasting process (*10–30%* of root mean square error reductions over state-of-the-art methods), underlining its notable potential for practical applications in univariate, skewed, long-term time series prediction tasks.

II. Related Work

A. Machine Learning Methods

Time series prediction has been investigated for many years. Traditional methods for accurately predicting future values in time series included the univariate Autoregressive (AR), Moving Average (MA), Simple Exponential Smoothing (SES), and Extreme Learning Machine (ELM) algorithms, and most famously the Autoregressive Integrated Moving Average (ARIMA) [11] method and its several variants. Wang et al. [12] provided a hybrid model combining Empirical Mode Decomposition (EMD), Ensemble Empirical Mode Decomposition (EEMD) and ARIMA for long-term streamflow forecasting, but they did not examine the effectiveness of their models on datasets with extreme values. Gaussian Process Regression (GPR) [13] and Quantile Regression (QR) [14] were used in some studies to not only predict but also quantify forecast uncertainty. Tree-based models, such as classification and regression trees (CARTs) and random forest (RF), have been employed due to their computational efficiency and ability to handle predictors without assuming any specific distribution. Prophet [15] is a popular time series forecasting model based on an additive model that captures nonlinear trends in the data, incorporating seasonal and holiday effects at various time scales, including annual, weekly, and daily patterns.

B. Deep Learning Methods

Recently, deep neural networks (DNNs) have shown their great advantages in various areas [16], [17]. General feed-forward deep learning models can be difficult to use with time series data since the input can have varying lengths and temporal dependencies. However, WaveNet [18] has demonstrated impressive capabilities in generating high-quality audio waveforms, which is also suitable for time series prediction tasks [19]. N-BEATS [10] is a time series prediction model with block-stacked pure fully-connected layers, which outperformed all competitors on the standard M3 [20], M4 [21], and TOURISM [22] datasets. DeepAR [23], a probabilistic forecasting model based on a Recurrent Neural Network (RNN) encoder-decoder architecture, leverages the autoregressive property of time series to generate probabilistic forecasts, allowing for uncertainty estimation.

When facing long sequence time-series forecasting (LSTF) tasks, in order to effectively capture exact long-range dependency coupling between output and input, a model must have a high prediction capacity. This means it can capture both short-term and long-term dependencies, enabling it to make accurate forecasts over extended periods of time. Recent research has demonstrated the Transformer model's ability to boost prediction power [24], [25]. However, the Transformer model suffers from a number of serious drawbacks that prohibit it from being directly applicable to LSTF, including quadratic temporal complexity, high memory utilization, and built-in limitations of the encoder-decoder design. To address these issues, alternative methods like Autoformer [26] and Reformer [27] have been proposed to improve the transformer's dependency discovery and representation ability. Informer [7] proposed a ProbSparse self-attention mechanism and a generative style decoder, while FEDFormer [8] represents time series by randomly selecting Fourier components in an attempt to improve efficiency compared to the standard Transformer.

C. Limitations of Current Approaches

While transformer-based models have shown efficacy in identifying long-range dependencies through self-attention mechanisms, forecasting across longer time horizons can have negative effects on accuracy or increased computational requirements, which restricts their usefulness [9]. In addition, earlier studies in long-term time series prediction often bypassed the nuances of heavily skewed datasets.

In contrast, our SEED model is tailored for such datasets, integrating a multi-tiered encoder-decoder framework that captures both global and local trends efficiently. SEED presents a streamlined and more accurate approach to forecasting skewed time series.

III. Preliminary

A. Problem Statement

We take on a challenging univariate time series forecasting problem, considering that the majority of data represent normal values which significantly contribute to the overall prediction performance, while the data set contains much fewer extreme values that deviate from the average and cause the distribution of the data to become skewed towards one side. The problem can be described as,

$$[x_1, x_2, \dots, x_T] \in \mathbb{R}^T \rightarrow [x_{T+1}, \dots, x_{T+H}] \in \mathbb{R}^H,$$

i.e., we are predicting the vector of length-H containing H future values, given a length-T vector of observed series historical values up to the present. The $x_1, \dots, x_T$ values are the input to our problem, while $x_{T+1}, \dots, x_{T+H}$ form the output.

B. Data Descriptions

In our research, we utilized a hydrologic dataset capturing streamflow from four California streams: Ross, Saratoga, UpperPen, and SFC. Given California's lack of rainfall during summer, our forecasting focus was on the months from September to May, deliberately excluding the summer period. Data for training and validation was drawn from January 1988 to August 2021. Our objective was to accurately project the streamflow for the subsequent year (September 2021 to May 2022), with predictions made every four hours. Each prediction estimated the upcoming 3 days based on the preceding 15 days of data. The performance metrics employed were Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Since the sensors measure the streamflow and precipitation every 15 minutes, we are attempting a lengthy forecasting horizon ($h = 288$), which is unquestionably an LSTF task based on the most recent research [7]–[9]. An example visualization of our data is provided in Fig. 1. To analyze our data set, we computed several statistical values of our input time series, which provide valuable insights into the shape and distribution of the data.

TABLE I: Statistical descriptions of our stream data.

Statistic / Stream	Ross	Saratoga	UpperPen	SFC
mean	2.91	5.77	6.66	20.25
max	1440.00	2210.00	830.00	7200.00
min	0.00	0.00	0.00	0.00
median	0.17	1.00	3.20	1.20
variance	597.22	711.09	452.90	12108.14
skewness	19.84	19.50	13.42	18.05
kurtosis	523.16	697.78	262.18	555.18

Table I shows the computed statistics of our input time series, including min, max, mean, median, variance, skewness,

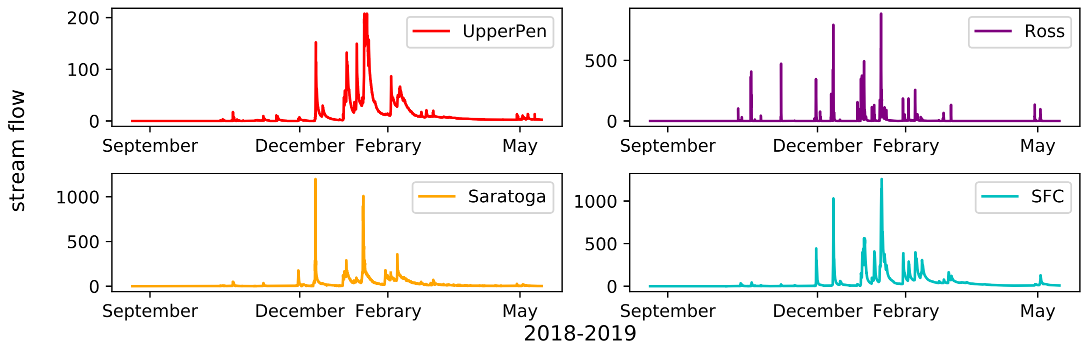


Fig. 1: The four stream datasets named Ross, Saratoga, UpperPen, and SFC. In California, prediction of streamflow and regulating timing of water to be left in stream is essential to the persistence of native species and health of the freshwater ecosystems, i.e., to support fish, wildlife, and habitat maintenance and creation. Under natural conditions, these streamsflow at stable low flows in the summer and begin to swell and surge during the winter months. All four of these streams have experienced high peaks from September 2018 to May 2019, which is representative of a general hydrologic year (September to May of the following year).

and kurtosis. The presence of high skewness and kurtosis values suggests that our data exhibit significant asymmetry and departure from a symmetric bell-shaped curve of a Normal distribution. Specifically, the positive skewness values indicate that the distribution is skewed to the right, resulting in a longer tail on the right side. This implies that there are more extreme values or outliers on the higher end of the distribution.

C. *Piecewise Linear Representation of Time Series*

Time series databases are becoming increasingly popular, and several high level representations have been proposed, such as Fourier Transforms [28], Wavelets [29], Symbolic Mappings [30] and Piecewise Linear Representation (PLR) [31].

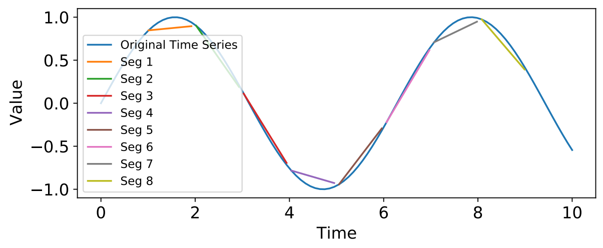


Fig. 2: Segment representation example. By dividing the time series into multiple segments and fitting a linear regression model to each segment, PLR captures the changing patterns and trends in the data more effectively compared to a single linear regression model.

PLR has been used in data mining applications for fast similarity search [32], novel distance measures [33], concurrent analysis of text and time series, vehicle speed estimation [34], and change point detection [35]. PLR simplifies the representation of time series, making their analysis more efficient while preserving key characteristics. In essence, PLR splits a series into several segments such that the maximum error of each segment does not exceed a threshold [36]. Fig. 2 shows an example PLR representation of a curve using 8 segments. However, the PLR algorithm mainly describes the linear relationship of the multi-segment representation and is often used as a preprocessing step to reduce both the space and computational cost of storing and transmitting time series.

In our research, inspired by PLR, we proposed a segment-expandable encoder-decoder architecture which aims to predict segment mean values layer by layer in an expanding way, with a goal of accurate future predictions for heavily skewed long-term time series.

D. *Kullback-Leibler Divergence Regularization*

Multiple-loss functions have been introduced as a heterogeneous multitask learning method [37]–[40]. We propose using multiple loss functions to guide our training objective. This encourages the model to learn rich representations capable of capturing the diverse variations and hidden relationships present in heavily skewed long-term time series.

In our model, Kullback-Leibler (KL) divergence is employed as a regularization loss term to ensure that the predicted segment distributions align with the ground truth segment distributions. The KL divergence is defined as

$$\mathrm{KL}(p \parallel q) = \sum p(x) \log \left(\frac{p(x)}{q(x)} \right),$$

where p and q are two distributions of the predicted data and the original data. The equation measures the average loss of information when approximating the p distribution with the q distribution. Essentially, it gauges the dissimilarity between the two distributions, with a lower value indicating a closer match.

IV. Methods

A. *Overview of the SEED Framework*

Our proposed Segment-Expandable Encoder Decoder (SEED) model is constructed as a variant of the encoder-

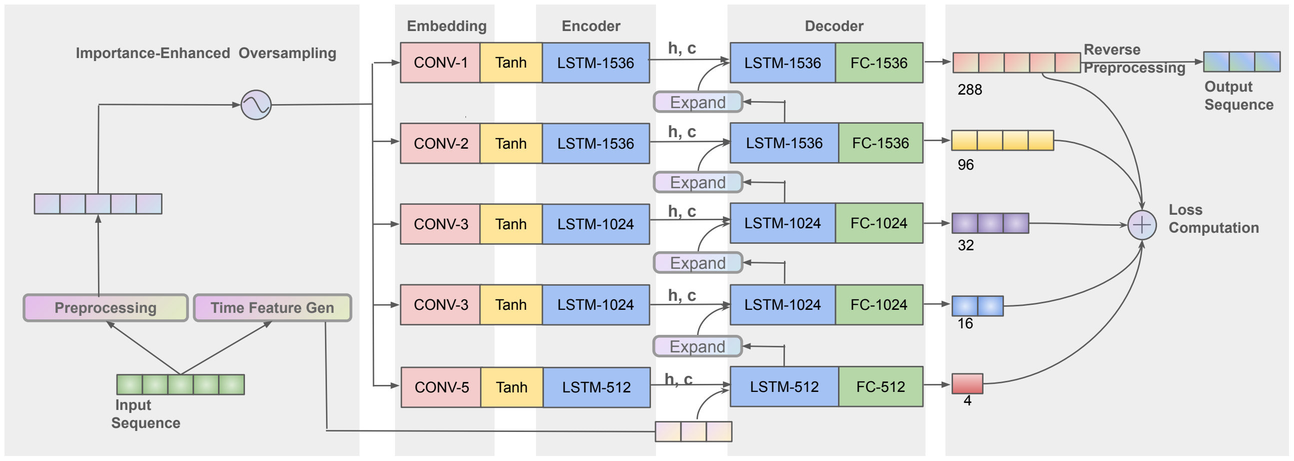


Fig. 3: The SEED architecture comprises three core components: embedding, encoder, and decoder. Initially, the input sequence undergoes preprocessing and sampling, with time features generated using sine and cosine transformations. Specifically, each month-day date is encoded into a feature pair using trigonometric or cyclical encoding, capturing the 365-day periodicity within a range of -1 to 1. Each layer's output contributes to the loss as a regularization factor. The output sequence length escalates from lower to upper layers, allowing varied scale information capture. The top layer's output is then refined to produce the final predicted sequence.

decoder architecture built with long short-term memory (LSTM) networks. It contains three main components, including embedding, encoder, and decoder. What sets SEED apart are the specially designed multiple levels in the decoder. Each layer is responsible for capturing specific aspects of the input sequence, representing the mean value distribution of predicted subsequences of varying lengths. This allows the model to handle different levels of complexity and capture temporal dependencies across various time scales. To be specific, each convolutional embedding block, denoted as CONV in Fig. 3, creates an input for the respective level in the encoder, then each layer of the encoder generates a hidden representation individually and feeds it into its corresponding decoder layer.

In this varied style of the encoder-decoder architecture, each encoder layer operates independently to encode the input and generates a unique hidden state and cell state. These states serve as the initial values for the corresponding layers in the decoder. Consequently, each layer is assigned a distinct task, as they represent the mean value distribution of different lengths of subsegments in the predicted sequence.

For example, as shown in Fig. 3, the first layer focuses on learning the mean value of four segments, resulting in an output length of 4. On the other hand, the last layer is responsible for the entire sequence and generates an output with a length of 288 in our scenario. It is important to note that lower layers handle simpler tasks compared to higher layers. As a result, they can be constructed with fewer trainable weights. As shown in Fig. 3, for example, the hidden layers have a size of 512 in the first encoder-decoder layer, while in the fifth encoder-decoder layer they have a size of 512×3. This allows for the allocation of more computational capacity to the higher layers, which are expected to predict longer sequences with greater fluctuations.

Our architecture offers a flexible framework specifically for tackling long-term time series forecasting tasks. By distributing the prediction responsibilities across multiple layers, each layer can specialize in capturing different characteristics of the data. This hierarchical approach optimizes the use of computational resources and enhances the model's ability to handle complex patterns and variations in long-term sequences.

B. Convolutional Embedding Layers

We utilize convolutional embedding layers with different kernel sizes and a subsequent $tanh$ activation function as a preprocessing step before feeding the data into the LSTM encoder. By using different kernel sizes, we can extract features at different spatial scales. Smaller kernel sizes are effective at capturing local patterns and fine-grained details, while larger kernel sizes capture broader patterns and global context. This allows each level of the LSTM encoder to focus on different aspects of the input sequence. Additionally, we use larger kernel sizes without padding in the convolutional embedding layers for the lower-level inputs. This choice serves two purposes. First, it helps to decrease the input size for the lower-level LSTM layers, shortening the LSTM computation path. This approach helps to mitigate the issues of exploding and vanishing gradients, which can arise when propagating gradients through a long sequence. By shortening the computation path, we minimize the impact of these gradient-related problems and facilitate more stable and efficient training of the LSTM model. Second, it allows us to allocate more computational resources to the higher-level LSTM layers, which are responsible for capturing longer-term dependencies and global patterns in the sequence.

C. Decoder Architecture

In the decoder LSTM stack, each layer operates as a level in a hierarchy to capture different distributions of the mean values of segment sequences. The number of layers is chosen to correspond to specific segment lengths, which are determined based on the desired granularity of the predictions.

For instance, as shown in Fig. 3, if we aim to generate a 288-length output sequence, we can use 5 layers in the decoder stack. Each layer is responsible for capturing the distribution of a specific segment length, such as 4, 16, 32, 96, and 288. These segment lengths represent the number of mean values obtained by dividing the output sequence into corresponding equal-size segments. In the first level, the output length is 4, which is meant to predict the mean values of 4 segments, each of which contains $288/4 = 72$ points in the forecasted series. Then, the 4 outputs are expanded to 16, which are injected into the second level, which will be responsible for predicting 16 mean values of 16 segments, each containing $288/16 = 18$ points. Expansion is done by repeating each value in the vector of means four times, i.e., a vector of size 4 containing values $\langle a, b, c, d\rangle$ becomes the vector $\langle a, a, a, a, b, b, b, b, , c, c, c, c, d, d, d, d\rangle$. In the last level, the length of the output is the same as the predicted sequence, meaning that the means of the segment values are the values themselves, since each segment contains $288/288 = 1$ point.

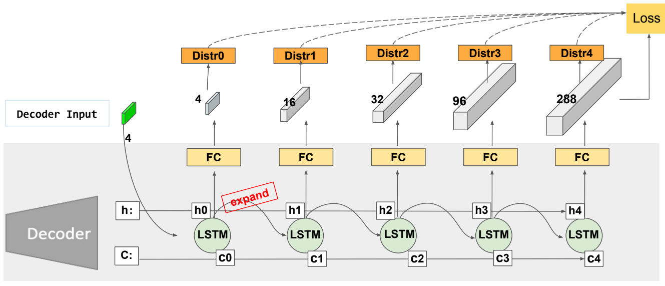


Fig. 4: Decoder architecture of SEED: To enhance the performance of each layer, the output of the prior layer is expanded by duplicating the values in the high dimensional hidden space, thereby providing more context and information for the following layers. Fully connected layers will generate one dimensional outputs, followed by softmax activations to predict the distribution of the means. This strategy of expanding the input helps improve the model's ability to capture complex patterns and long dependencies in the data.

This hierarchical approach helps handle extreme values effectively. By predicting the mean value of different length subsegments, extreme values are represented and spread across multiple levels in the hierarchy, leading to higher mean values in the segments containing them. This enables the model to capture both fine-grained and coarse-grained patterns in the data. As shown in Fig. 4, to better regulate each layer's function, we incorporate a fully connected layer after the LSTM. This layer generates a one-dimensional output of the same length and helps the model learn the distribution of segment mean values in the predicted sequence. This addition enhances the model's ability to capture segment-specific characteristics and improves overall performance.

D. Multiple-Objective Loss Functions

Our model utilizes the KL divergence loss as a way to ensure that the predicted segment distributions align with the ground truth segment distributions. By minimizing the Kullback-Leibler divergence between the two distributions, the model is encouraged to iteratively improve predictions of the segment distributions during the training process. This enables the model to capture the desired characteristics of the ground truth distribution, guiding the convergence of the ultimate output towards the ground truth.

The Kullback-Leibler divergence loss acts as a regularization term that encourages the model to match the target distribution while balancing the sequence generation loss. For example, in the 5-level setting described in Fig. 3, the regularization loss term for the ith layer can be described as,

$$\mathcal{L}_i = \mathrm{KL}\left(\mathrm{softmax}(p\_m_i), \mathrm{softmax}(g\_m_i)\right),$$

where p_m_i is the output of the FC layer in Fig. 4, which represent the predicted segment mean values in the ith layer, while g_m_i is the vector of computed ground truth mean values for the segments in the ith layer. Applying the $softmax$ function turns both vectors of mean values into distributions, which then allows us to compute the KL divergence between the two distributions. Then, the overall loss is composed as,

$$\mathcal{L} = RMSE(\hat{y}, y) + \lambda \times \left(\sum_{i=1}^{k} \mathcal{L}_i\right),$$

where λ is a multiplier applied on those regularization items. The value of λ decreases with each epoch and is set to $\lambda = 100 \times \max\left(-1 \times e^{epoch/45} + 2, 0.0\right)$ in our experiments.

E. Importance-Enhanced Oversampling Policy

We use a creative oversampling policy to capture important sequences in our dataset. As shown in algorithm 1, our method randomly samples sequences and checks if their maximum values in the inference section of the series exceed a threshold T. Those that do are deemed important sequences. We adjust these sequences by moving the maximum value to the middle of the inference section and perform multiple iterations of sampling from the beginning with a specified step size S. This effectively captures the characteristics of important samples. For normal, or less significant sequences, we randomly select them without special attention. This oversampling policy improves the representation of important sequences while maintaining diversity in the training set. The threshold T and the step size S both can be adjusted as a parameter based on the data set distribution.

V. Experiments

To provide a more comprehensive evaluation of our proposed method, we applied SEED to four distinct streamflow

```
Input  : Dataset with training and inference sequences
Output: Oversampled training set
Procedure Oversampling;
while training set size is not satisfied do
    Randomly sample a sequence including training and
      inference sections;
    if maximum value in inference section $> T$ then
        Mark sequence as important;
        Move maximum value to the middle of the inference
          section of the sequence;
        foreach index $I$ in the sequence with step size $S$ do
            Sample starting at $I$;
            Add sampled sequence to oversampled training
              set;
        end
    end
    else
        Add sequence to oversampled training set;
    end
end
```

Algorithm 1: Importance-Enhanced Oversampling Policy

datasets[1]. We compared the performance of SEED against 7 state-of-the-art models. To gain a better understanding of the effectiveness of the segment layers setting, the role of the regularization terms, and the impact of the oversampling policy, we conducted a series of additional experiments and ablation studies. The results of these experiments led to the following findings:

- SEED outperforms all the baseline models in the 3-day ahead prediction task for heavily skewed long-term hydrologic time series.
- Segment expansion in a hierarchical way is essential in capturing more accurate hidden patterns and improving the robustness of the model to predict abnormal data.
- The importance-enhanced sampling policy can effectively improve the performance of SEED.
- The added level-wise regularization leads to prediction improvements.

A. Experimental Settings

We utilized four hydrologic datasets, Ross, Saratoga, UpperPen, and SFC, which represent different stream locations. Our objective was to forecast streamflow for a year, excluding the summer months, from September 2021 to May 2022. The training and validation datasets were randomly selected from time series data covering the period from January 1988 to August 2021.

During the inference phase, we predicted streamflow using rolling predictions at intervals of 4 hours. Each prediction, however, inferred 288 data points, i.e., the predicted streamflow over the next 3 days at 15 minute intervals. To make these predictions, we utilized the previous 15 days of data, equivalent to 1440 time steps. Prior to model training, all time series underwent pre-processing steps including a logarithmic transformation, $x_i = \log(1 + x_i)\ \forall i$, and standardization (subtracting the mean and dividing by the standard deviation). In order to obtain the final inference predictions, we performed post-processing by reversing the standardization and logarithmic transformations.

In our models, we utilized a 5-layer SEED architecture, where each layer consisted of a 1-layer bidirectional LSTM. After experimenting with different LSTM layer widths, we determined that using 512 nodes per layer yielded the best performance. In the embedding stage, we employed five CNN layers, each generating 384 channels. The kernel sizes for these layers were set to 5, 3, 3, 2, and 1, respectively, from the bottom to the top layer. We applied a stride equal to the kernel size, with no padding, and a subsequent $tanh$ activation function.

B. Baseline Methods

We compared our proposed method, SEED, against a wide array of state-of-the-art time series and hydrologic prediction methods, introduced earlier in the related work section, including,

- FEDFormer [8], which combines Transformer with the seasonal-trend decomposition methods, has been shown to be more efficient than the standard Transformer, yielding improved results for long-term series forecasting;
- Informer [7], a transformer-style model for long-term time series prediction with a prob-sparse self-attention mechanism;
- NEC+ [41] a group of LSTM-based models that holds the best performance for *hydrologic* time series prediction in the presence of extreme events;
- NLinear [9], an effective linear model with one order difference preprocessing for long-term time series;
- DLinear [9], a trend decomposed linear model for long-term time series prediction;
- N-BEATS [10], a state-of-the-art time series prediction method that outperformed all competitors on the standard M3 [20], M4 [21] and TOURISM [22] datasets; and
- EnDecoder, the common encoder-decoder model built with LSTM layers, which is a popular architecture for sequence-to-sequence tasks such as machine translation, text summarization, and speech recognition.

C. Main Results

Table II shows the test RMSE and MAPE performance for the models that achieved the best performance on our out-of-sample test set. To comprehensively evaluate the performance of all the methods, we group the RMSE and MAPE metrics in three different categories. By analyzing the results from multiple perspectives, we can gain more insight into the strengths and weaknesses of each method. To be specific, for the RMSE value, "All" represents the average RMSE of all test samples compared with the ground truth. The "High" column represents the average RMSE value for those samples containing at least one value larger than the mean value of the whole time series, while the "Low" column computes the

[1]Data and code for the project can be found at https://github.com/davidanastasiu/seed.

TABLE II: Effectiveness comparisons with state-of-the-art methods. The best value of each metric is bolded. The second best value of each metric is underlined.

Methods	**Metric**	**Ross**			**Saratoga**			**Upperpen**			**SFC**		
		All	High	Low	All	High	Low	All	High	Low	All	High	Low
FEDformer	RMSE	6.49	30.82	2.51	6.85	11.95	4.59	2.38	17.68	1.07	24.15	94.28	6.68
	MAPE	2.49	5.27	2.04	2.26	1.50	2.60	1.02	2.37	0.90	2.81	1.70	3.09
Informer	RMSE	9.14	31.00	5.56	4.89	13.64	1.01	5.33	16.26	4.40	19.00	85.40	2.46
	MAPE	5.45	5.80	5.39	0.73	1.40	0.43	4.21	2.70	4.34	0.54	0.71	0.49
NLinear	RMSE	5.84	32.12	1.54	4.98	14.61	0.70	1.74	15.07	0.61	18.43	83.31	2.26
	MAPE	1.62	4.89	1.09	0.75	1.74	0.31	0.57	1.69	0.47	0.87	1.03	0.83
DLinear	RMSE	6.90	30.96	2.97	4.06	7.63	2.48	3.25	14.01	2.33	23.64	79.76	9.65
	MAPE	2.79	4.03	2.58	1.31	0.85	1.51	2.04	1.68	2.07	4.02	1.04	4.76
NEC+	RMSE	9.33	38.34	4.58	1.95	5.55	0.35	1.94	13.92	0.92	16.39	76.63	1.38
	MAPE	4.53	8.33	3.91	0.21	0.30	0.17	0.80	0.84	0.80	0.55	0.61	0.54
NBeats	RMSE	5.16	30.09	1.08	3.60	9.44	1.01	1.23	13.20	0.21	31.47	95.33	15.55
	MAPE	1.25	3.17	0.94	0.70	1.21	0.47	0.25	0.78	0.20	3.24	0.88	3.83
EnDecoder	RMSE	5.58	30.81	1.45	1.93	5.69	0.26	2.95	16.33	1.80	17.46	79.04	2.11
	MAPE	1.62	3.72	1.28	0.16	0.29	0.11	1.81	2.34	1.76	0.84	0.96	0.80
SEED	RMSE	**4.23**	**29.74**	**0.05**	**1.67**	**5.14**	**0.12**	**1.07**	**12.83**	**0.07**	**14.44**	**70.04**	**0.59**
	MAPE	**0.11**	**0.53**	**0.04**	**0.09**	**0.19**	**0.05**	**0.10**	**0.57**	**0.06**	**0.20**	**0.42**	**0.14**

RMSE for those test samples where all the ground truth values were lower than the mean value. We note that most of the extreme events exist in samples included in the "High" RMSE calculation, and the samples included in the "Low" RMSE calculation pertain to primarily normal events. The best results are highlighted in bold.

Our proposed model, SEED, demonstrated superior performance over all baselines across all four benchmark datasets. In comparison to the three second-best models (NEC+, NBeats, and EnDecoder), SEED achieved an average, relative RMSE reduction of 31.44%, 34.68%, and 29.67% across the datasets. Specifically, compared to NEC+, SEED's reductions ranged from modest (11.90% and 14.36%) to significant (54.66% and 44.85%) for the SFC, Saratoga, Ross, and UpperPen sensors, respectively. Similarly, SEED outperformed NBeats with reductions ranging from modest (18.02% and 13.01%) to substantial (53.61% and 54.11%). Compared to EnDecoder, SEED consistently delivered better RMSE performance across all datasets, with reductions of 24.19%, 13.47%, 63.72%, and 17.30% for Ross, UpperPen, Saratoga, and SFC, respectively.

When we compare results from the "High" and "Low" categories, we can see that NLinear performs better on the low group of time series, while DLinear performs better on the high group. FEDFormer and Informer do not show a clear advantage in either group. NBeats performs poorly, particularly in the high group, due to its performance on the Saratoga and SFC datasets. NEC+ and EnDecoder demonstrate benefits across both groups. SEED exhibits the best performance of both groups on all four datasets with varying levels of complexity.

Fig. 5 shows the results of SEED compared against the second best baselines. Top-right, middle-left, and bottom-right show three intricate stream data patterns. In these instances, SEED outperforms the runner-up approach by a significant margin. We did not include the other models' poor performance in each figure as it would impede visualizing the performance of the top models. For the Ross and UpperPen sensors, the NBeats model performs well, while for the Saratoga sensor, the EnDecoder model is the second best. In the case of the SFC sensor, the NEC+ model shows promising results. To illustrate the differences, we provide inference examples in comparison with these models. The examples illustrate that SEED is better able to capture complex patterns and provides more reliable predictions for streamflow forecasting tasks.

VI. Ablation Studies

To answer the question of the effects of our sampling policy, regularization, and segment expanding strategies, we conducted a group of experiments on the UpperPen and Saratoga datasets. For each of these models, we used 10,000 samples in the training set.

A. *Effect of the Importance-Enhanced Oversampling Policy*

In algorithm 1, the threshold T is used to classify the importance of a sampled sequence, while the step size S should be

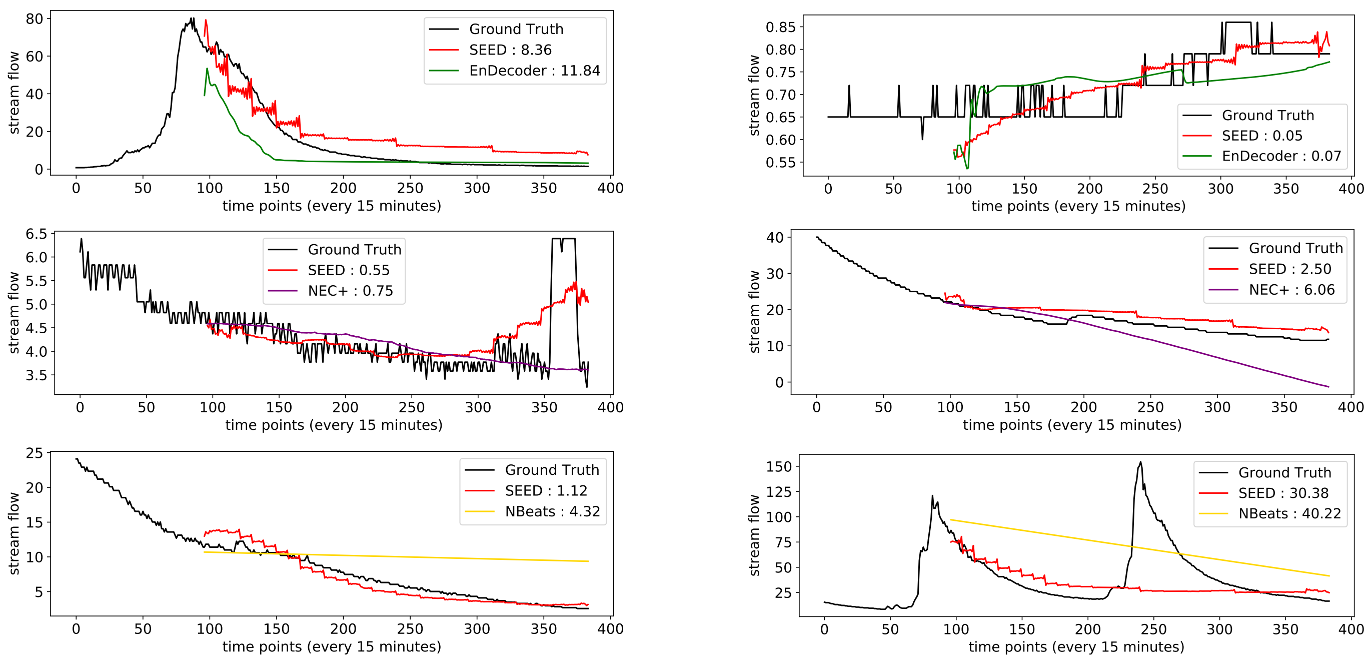
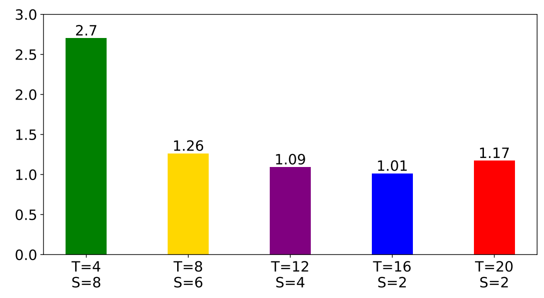


Fig. 5: Example comparisons with the second best baselines; the EnDecoder model can capture the general trend of the ground truth, but the segment expanding mechanism of SEED enables it to more accurately predict the streamflow values, both during normal conditions and in the presence of rare events. On the other hand, the NEC+ and NBeats models sometimes succeed in following the trend in predictions, but other times they deviate from the ground truth. Best viewed in color.

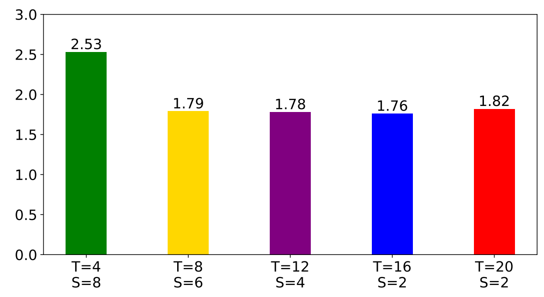
adjusted according to the frequency of peak periods in the data. Considering the heavy skewness and high kurtosis in the data, which indicate a departure from a regular distribution, it is not suitable to determine the threshold and step size solely based on data observation. Therefore, we recommend employing a grid search approach to determine the optimal values for these parameters.

To evaluate the impact of this policy, we increased the threshold T while simultaneously decreasing the step size S, aiming to capture more information from the more important sampled sequences. We initially trained the models without any regularization loss components, i.e., $\mathcal{L} = RMSE(\hat{y}, y)$. Fig. 6 shows the RMSE of the inference on the UpperPen and Saratoga test sets. For both datasets, we observed that increasing the threshold T and decreasing the step size S had a positive impact on the results. However, there was a point of diminishing returns, where further increasing the threshold T did not lead to a significant decrease in RMSE. This suggests that there is an optimal threshold T value beyond which the policy's effectiveness plateaus.

It is worth noting that the effect of this policy was more pronounced on the UpperPen dataset compared to the Saratoga sensor dataset. This is likely due to the difference in variance between the two datasets. The UpperPen dataset has a smaller variance, indicating that it contains a narrower range of values. As a result, the policy's ability to capture important samples and extract meaningful patterns is more evident in the UpperPen dataset.



(a) RMSE of UpperPen



(b) RMSE of Saratoga

Fig. 6: Sampling policy effects of different thresholds T and step size S settings on the UpperPen and Saratoga datasets; five groups of experiments were conducted, with the threshold $T \in [4, 8, 12, 16, 20]$ and the step size $S \in [8, 6, 4, 2, 2]$ respectively.

B. Effect of the KL Regularization Terms

In this part of our study, we aimed to test the importance of our layer-based regularization strategy on the performance of our model by reintroducing regularization components back to the experiments in Section VI-A. The aim was to demonstrate the benefits of regularization techniques in our context and assess their impact on the model's results. Specifically, we continued conducting experiments on the UpperPen and Saratoga sensor datasets with $T = 4$ and $S = 8$. After reintroducing the regularization components, we observed improvements in the RMSE values for both datasets. In the case of UpperPen, the RMSE decreased from 2.70 to 1.18, indicating a notable enhancement. Similarly, for the Saratoga dataset, the performance improved from $RMSE = 2.53$ to 2.21.

Fig. 7 visually depicts the impact of the Kullback-Leibler divergence regularization loss terms on the inference results for the two sensors. The green line is the prediction without regularization items and the red one is the result after including regularization in the model training. Results indicate that the regularization term plays a significant role in guiding the model to adhere to the segment trends layer by layer.

C. Effect of Segment Expanding

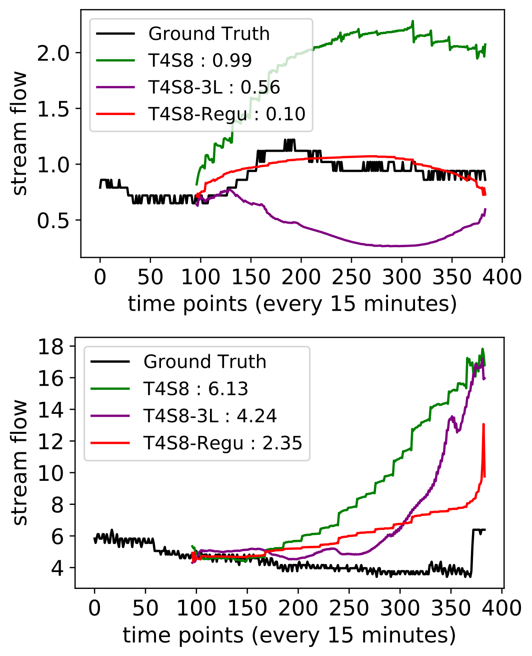
In order to examine the impact of our segment expanding strategy, we conducted experiments by reducing the number of expanding layers from 5 to 3, while using a threshold $T = 4$, step size $S = 8$, and training the models with our added regularization loss for each layer. The segment lengths for the 3 layers were set to 36, 144, and 288, respectively.

The inference examples of the 3 layer models are represented by the purple line in Fig. 7. For the UpperPen dataset the RMSE increased from 1.18 to 1.38 and for the Saratoga dataset the RMSE increased from 2.21 to 2.38 after reducing the number of expansion layers. According to these findings, the SEED model's prediction accuracy may suffer if the number of layers is reduced, as seen by the rise in RMSE. This shows that the segment expanding process is essential for identifying significant dependencies and patterns in the data, and that reducing the number of expansion layers may lead to the loss of vital information on some datasets.

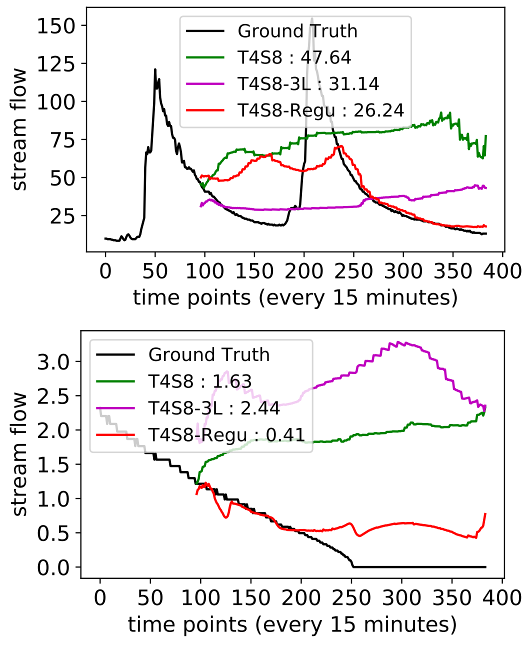
VII. Conclusion

In our study, we introduced SEED, a unique segment-expandable encoder-decoder model, crafted to address single-shot prediction challenges in skewed and heavy-tailed univariate long-term time series datasets. Utilizing segment representation learning and a hierarchical Kullback-Leibler divergence regularization loss, SEED markedly elevates the accuracy of streamflow prediction. Its tiered structure ensures efficient learning of expandable representations, and the incorporated loss bolsters the model's resilience.

In the future, we aim to test SEED's versatility across various domains beyond just hydrologic predictions and explore replacements for the LSTM base layer, seeking further performance optimization. These endeavors aim to expand and amplify SEED's applicability in varied scenarios.



(a) Segment expanding effects on the Saratoga dataset



(b) Segment expanding effects on the UpperPen dataset

Fig. 7: Effect of regularization and segment expanding strategies. The green line with the label 'T4S8' is the result of inference using the model trained with $T = 4$ and $S = 8$ in Fig. 6. The red line labeled 'T4S8-Regu' represents the model with our layer-based regularization strategy added to the original 'T4S8' model. The purple line labeled 'T4S8-3L' represents the model with the number of layers reduced from 5 to 3, based on the 'T4S8-Regu' model.

References

[1] S. Hua, M. Kapoor, and D. C. Anastasiu, "Vehicle tracking and speed estimation from traffic videos," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, ser. CVPRW'18, vol. 1. : IEEE, July 2018, pp. 153–1537.

[2] P. Hewage, M. Trovati, E. Pereira, and A. Behera, "Deep learning-based effective fine-grained weather forecasting model," *Pattern Analysis and Applications*, vol. 24, no. 1, pp. 343–366, 2021.

[3] B. Bose, T. Downey, A. K. Ramasubramanian, and D. C. Anastasiu, "Identification of distinct characteristics of antibiofilm peptides and prospection of diverse sources for efficacious sequences," *Frontiers in Microbiology*, vol. 12, 2022.

[4] S. Mohan, S. Mullapudi, S. Sammeta, P. Vijayvergia, and D. C. Anastasiu, "Stock price prediction using news sentiment analysis," in *2019 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, ser. BDS 2019. : IEEE, April 2019, pp. 205–208.

[5] Z. Zhang, H. Qin, L. Yao, Y. Liu, Z. Jiang, Z. Feng, S. Ouyang, S. Pei, and J. Zhou, "Downstream water level prediction of reservoir based on convolutional neural network and long short-term memory network," *Journal of Water Resources Planning and Management*, vol. 147, no. 9, p. 04021060, sep 2021. [Online]. Available: https://doi.org/10.1061\%2F\%28asce\%29wr.1943-5452.0001432

[6] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*. Springer, 2003, pp. 107–119.

[7] 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 *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 11 106–11 115.

[8] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International Conference on Machine Learning*. PMLR, 2022, pp. 27 268–27 286.

[9] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" *arXiv preprint arXiv:2205.13504*, 2022.

[10] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[11] G. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. : Holden-Day, 1976.

[12] Z.-Y. Wang, J. Qiu, and F.-F. Li, "Hybrid models combining emd/eemd and arima for long-term streamflow forecasting," *Water*, vol. 10, no. 7, 2018. [Online]. Available: https://www.mdpi.com/2073-4441/10/7/853

[13] J. Han, X.-P. Zhang, and F. Wang, "Gaussian process regression stochastic volatility model for financial time series," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 6, pp. 1015–1028, 2016.

[14] Q. Huang, H. Zhang, J. Chen, and M. He, "Quantile regression models and their applications: A review," *Journal of Biometrics & Biostatistics*, vol. 8, no. 3, pp. 1–6, 2017.

[15] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018. [Online]. Available: https://doi.org/10.1080/00031305.2017.1380080

[16] D. C. Anastasiu, J. Gaul, M. Vazhaeparambil, M. Gaba, and P. Sharma, "Efficient city-wide multi-class multi-movement vehicle counting: A survey," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 3, pp. 235–250, Dec 2020.

[17] Y. Pei, Y. Liu, N. Ling, L. Liu, and Y. Ren, "Class-specific neural network for video compressed sensing," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[19] F. Dorado Rueda, J. Durán Suárez, and A. del Real Torres, "Short-term load forecasting using encoder-decoder wavenet: Application to the french grid," *Energies*, vol. 14, no. 9, p. 2524, 2021.

[20] S. Makridakis and M. Hibon, "The m3-competition: results, conclusions and implications," *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, 2000, the M3- Competition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207000000571

[21] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802–808, 2018.

[22] G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu, "The tourism forecasting competition," *International Journal of Forecasting*, vol. 27, no. 3, pp. 822–844, 2011.

[23] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[24] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[25] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," *arXiv preprint arXiv:2211.14730*, 2022.

[26] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.

[27] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[28] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, pp. 263–286, 2001.

[29] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*. IEEE, 1999, pp. 126–133.

[30] R. A. K.-l. Lin and H. S. S. K. Shim, "Fast similarity search in the presence of noise, scaling, and translation in time-series databases," in *Proceeding of the 21th International Conference on Very Large Data Bases*. Citeseer, 1995, pp. 490–501.

[31] L. Chua and R. Ying, "Canonical piecewise-linear analysis," *IEEE Transactions on Circuits and Systems*, vol. 30, no. 3, pp. 125–140, 1983.

[32] H. Wu, B. Salzberg, and D. Zhang, "Online event-driven subsequence matching over financial data streams," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, pp. 23–34.

[33] F. Yajing and G. Yujian, "A novel approach based on neural networks and support vector machine for stock price pattern discovery," in *2017 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2017, pp. 259–263.

[34] S. Hua, M. Kapoor, and D. C. Anastasiu, "Vehicle tracking and speed estimation from traffic videos," in *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, ser. CVPRW'18, vol. 1, July 2018, pp. 153–1537.

[35] Q. Tao, L. Li, X. Huang, X. Xi, S. Wang, and J. A. Suykens, "Piecewise linear neural networks and deep learning," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 42, 2022.

[36] Y. Hu, P. Guan, P. Zhan, Y. Ding, and X. Li, "A novel segmentation and representation approach for streaming time series," *IEEE Access*, vol. 7, pp. 184 423–184 437, 2018.

[37] S. S. Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 2015, pp. 643–650.

[38] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[39] H. Ma, Z. Zhang, W. Li, and S. Lu, "Unsupervised human activity representation learning with multi-task deep clustering," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–25, 2021.

[40] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1930–1939.

[41] Y. Li, J. Xu, and D. C. Anastasiu, "An extreme-adaptive time series prediction model based on probability-enhanced lstm neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, pp. 8684–8691, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26045