PromptTSS: A Unified Model for Time Series Segmentation with Multi-Granularity States

Ching Chang, Ming-Chih Lo, Wen-Chih Peng, Tien-Fu Chen
National Yang Ming Chiao Tung University
{blacksnail789521.cs10, max2306.cs12}@nycu.edu.tw, {wcpeng, tfchen}@cs.nycu.edu.tw

Abstract

Multivariate time series data, collected across various fields such as manufacturing and wearable technology, often contain states of interest at multiple levels of granularity. Effectively segmenting these states is critical for downstream tasks like predictive maintenance, anomaly detection, and performance optimization. However, existing time series segmentation (TSS) methods face two key challenges: (1) the inability to handle multiple levels of granularity within a unified model, and (2) limited adaptability to new, evolving patterns in dynamic environments. To address these challenges, we propose PromptTSS, a novel framework for time series segmentation with multi-granularity states. PromptTSS employs a unified model augmented by a prompting mechanism, leveraging label and boundary information to guide segmentation. This mechanism enables the model to capture both coarse-grained trends and fine-grained events simultaneously while adapting dynamically to unseen patterns. The framework integrates a time series encoder, prompt encoder, and state decoder, ensuring robust and flexible segmentation. Experiments on real-world datasets validate the effectiveness of PromptTSS, demonstrating its superior performance in capturing multi-granularity states and adapting to evolving time series dynamics.

1 Introduction

Multivariate time series data is collected in numerous fields and industries, including sensor readings in semiconductor manufacturing (Gaugel and Reichert 2023) and activity tracking from smartwatches (Kwapisz, Weiss, and Moore 2011). These datasets often contain informative *states* or *segments* representing the object of interest or its condition. Identifying and analyzing these informative states within the time series data is crucial as they provide valuable insights beneficial for downstream tasks like predictive maintenance, anomaly detection, and performance optimization (Hallac et al. 2017; Wang et al. 2023).

Time series data in industrial contexts often encompass states with multiple levels of granularity, from coarse to fine, each providing unique insights (as shown in the top side of Figure 1) (Reis 2019). For instance, in semiconductor manufacturing factories, states can be collected at varying levels of detail, ranging from the coarser stage-level, to the intermediate station-level, and down to the finer step-level (Yang, Bom, and Shen 2024). Coarse states (e.g., stage-level or station-level) reveal broad trends like machine performance decline, while fine-grained states (e.g., step-level) capture specific events like abrupt temperature spikes, indicating immediate maintenance needs. Effectively utilizing and integrating these different granularities of states from coarse to fine is crucial for comprehensive analysis and informed decision-making. However, segmenting multivariate time series data with multiple granularity levels of states presents two significant challenges.

The first challenge is the inability to use a unified model to handle states across multiple levels of granularity. Existing time series segmentation (TSS) models are designed to work at a single level of granularity, which significantly limits their capacity to detect both broad, long-term trends and short, localized events simultaneously (Phan et al. 2020; Li et al. 2020; Perslev et al. 2019). This single-granularity segmentation approach results in models that either overlook fine-grained details or fail to capture broader patterns, making it challenging to effectively capture the full spectrum of events and trends. For example, in a semiconductor manufacturing factory, coarse-level segmentation might miss short-lived anomalies like abrupt spikes in temperature at a specific station, while fine-grained segmentation could over-segment data, failing to capture larger patterns such as gradual declines in machine performance across multiple stages.

The second challenge is the limited adaptability to new patterns. Most segmentation models are static, meaning they are trained on specific datasets and lack the ability to adapt to evolving patterns (Gaugel and Reichert 2023; Ordóñez and Roggen 2016). In dynamic environments, such as smart manufacturing or financial markets, time series patterns can change over time, necessitating continuous adaptation (Hammami et al. 2020). Models trained on historical data often struggle to generalize well to new, unseen behaviors, leading to outdated segmentation rules. This lack of flexibility hinders the model's ability to accurately detect novel trends or anomalies, ultimately reducing the effectiveness of the analysis in environments where conditions are constantly shifting.

To address these challenges, we propose PromptTSS, a unified model for time series segmentation with multi-granularity states, utilizing a novel prompting mechanism guided by a small portion of label and boundary information. To tackle the first challenge, the prompting mechanism allows the model to select the appropriate granularity, all while utilizing a single unified model. This approach enhances the model's ability to capture both coarse-level trends and fine-level events simultaneously. For the second challenge, the prompting mechanism enables the model to dynamically adapt to unseen patterns by allowing users to provide prompts during inference. This capability ensures that the model can continuously incorporate ground truth information, guiding its predictions even in the presence of new or evolving behaviors. In PromptTSS, two types of ground truth information are used as prompts (as shown in the bottom side of Figure 1): label information, which provides contextual state annotations, and boundary information, which indicates the transition points between different states. The framework employs a time series encoder, a prompt encoder, and a state decoder. The time series encoder and prompt encoder encode time series data and prompts, while the state decoder predicts the corresponding states based on the encoded representations.

In summary, the paper's main contributions are as follows:

- **Unified Multi-Granularity Segmentation**: We propose PromptTSS, the first unified model for segmenting multivariate time series data with multiple granularity levels of states, addressing the challenges of capturing both broad trends and fine-grained events.
- **Dynamic Adaptability with Prompts**: We introduce a novel prompting mechanism that incorporates ground truth information (label and boundary), enabling the model to dynamically adapt to unseen patterns and evolving behaviors in dynamic environments.

2 Problem Formulation

Given a complete and evenly-sampled multivariate time series and its corresponding sequence of discrete states, we use a sliding data window of length T to extract sequential samples. Each sliding window captures a sequence of the time series data, denoted as $\mathbf{x} = (x_1, \ldots, x_T) \in \mathbb{R}^{T \times C}$, where x_t is a C-dimensional vector representing the features at time step t. The corresponding sequence of states is denoted as $\mathbf{s} = (s_1, \ldots, s_T) \in \mathbb{Z}^T$, where each s_t belongs to one of K discrete states.

In addition to the time series data, auxiliary prompt information is introduced to guide the segmentation process. The prompts consist of label prompts $\mathbf{p}_l = (p_{l,1}, \ldots, p_{l,T}) \in \mathbb{R}^{T \times D_l}$, which provide guidance on the correctness of specific states for each timestamp, and boundary prompts $\mathbf{p}_b = (p_{b,1}, \ldots, p_{b,T}) \in \mathbb{R}^{T \times D_b}$, which encode information about state transitions. Here, D_l and D_b represent the dimensions of the label and boundary prompt features, respectively. The objective is to use the input data $\mathbf{x}$ along with the prompts $\mathbf{p}_l$ and $\mathbf{p}_b$ to predict the corresponding sequence of discrete states $\mathbf{s}$.

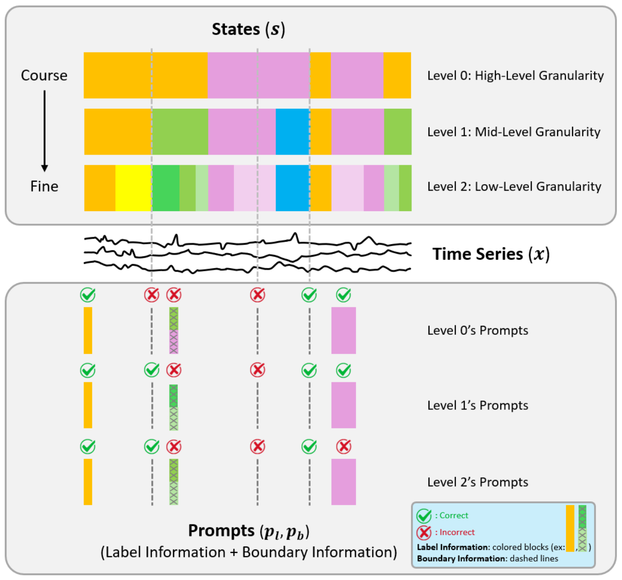
States (s)
Course
Fine
Level 0: High-Level Granularity
Level 1: Mid-Level Granularity
Level 2: Low-Level Granularity
Time Series (x)
Level 0's Prompts
Level 1's Prompts
Level 2's Prompts
Prompts (p_l, p_b)
(Label Information + Boundary Information)
✓ : Correct
✗ : Incorrect
Label Information: colored blocks (ex:)
Boundary Information: dashed lines

Figure 1: **Multiple-Granularity States and Prompts**. The top side shows the different levels of granularity in time series data, ranging from coarse to fine. The bottom side shows the two types of ground truth information used as prompts in PromptTSS, including label and boundary information. The green check indicates correct information, while the red cross indicates incorrect information.

3 Method

Figure 2 presents the overall architecture of the PromptTSS framework. The methodology is described through three core modules in PromptTSS. First, the time series encoder encodes the time series data into time series embeddings (Section 3.1). Next, the prompt encoder processes the prompts into prompt embeddings (Section 3.2). Finally, the state decoder predicts the corresponding states based on the embeddings from both the time series encoder and the prompt encoder (Section 3.3).

3.1 Time Series Encoder

In this work, the Transformer encoder is chosen as the architecture for the time series encoder (f_x). Given time series data $\mathbf{x}$, the goal is to generate time series embeddings $\mathbf{z}_x$.

Due to the often lengthy nature of time series data, directly feeding it into a Transformer encoder can be computationally expensive. To address this issue, *patching* (Nie et al. 2023) is employed, where adjacent time steps are aggregated into a single patch-based token to reduce the computational burden. Patching has been shown to be effective not only in the time series domain but also in other fields such as computer vision (Kirillov et al. 2023) and audio processing (Gong, Chung, and Glass 2021).

The process begins with patching the time series:

$$\mathbf{x}_{\text{patched}} = \text{Patching}(\mathbf{x}), \tag{1}$$

where $\text{Patching}(\cdot)$ aggregates adjacent time steps into the patched time series $\mathbf{x}_{\text{patched}} \in \mathbb{R}^{T_{\text{patched}} \times (C \cdot P)}$. Here, T_{patched}

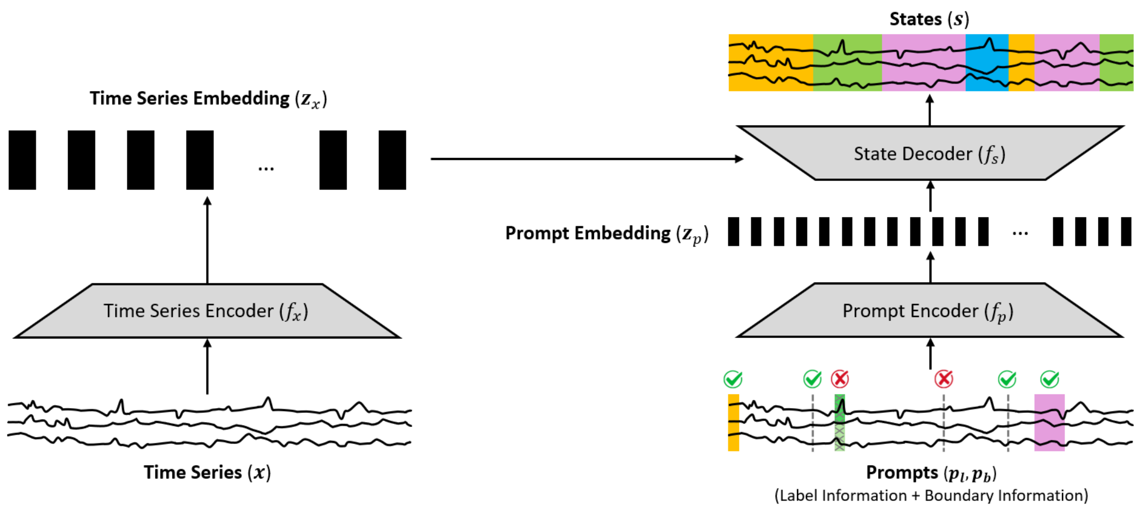

Figure 2: **PromptTSS framework**. Consists of three modules: a time series encoder that generates time series embeddings, a prompt encoder that produces prompt embeddings, and a state decoder that predicts states based on both embeddings.

denotes the number of patch tokens, and P is the patch length. The Transformer encoder f_x is then applied to the patched time series:

$$\mathbf{z}_x = f_x(\mathbf{x}_{\text{patched}}), \tag{2}$$

where $\mathbf{z}_x \in \mathbb{R}^{T_{\text{patched}} \times D}$ represents the time series embeddings, with D being the embedding dimension.

3.2 Prompt Encoder

A novel prompting mechanism is introduced, enabling users to specify a desired level of granularity during inference or guide the model in adapting to unseen patterns. The prompt encoder (f_p) is designed to transform the provided prompts into embeddings that effectively guide the model. In PromptTSS, two types of prompts are utilized: label prompts ($\mathbf{p}_l$) and boundary prompts ($\mathbf{p}_b$), each capable of incorporating both correct and incorrect information.

Label Prompts For label prompts, the sequence of prompts is denoted as $\mathbf{p}_l = (p_{l,1}, \dots, p_{l,T})$. The correct label provides the single correct state for a given timestamp, guiding the model toward the target state. The incorrect labels, in contrast, provide multiple incorrect states for a timestamp, helping the model avoid specific incorrect states. Notably, for label prompts, both the correct label and incorrect labels can be provided at the same time. However, each timestamp can be assigned only one correct label, while multiple incorrect labels may be provided to guide the model.

To encode label prompts, the correct label and incorrect labels for each timestamp are first collected to form a single vector $\mathbf{p}_{l,t} \in \{0, 1\}^{2K}$, where K represents the total number of possible states. Specifically, the first K-dimensional space represents the correct label using a one-hot encoding (only one element can be 1, while all others are 0), while the second K-dimensional space represents the incorrect labels using a multi-hot encoding (multiple elements can be 1 to indicate the incorrect states). The vector $\mathbf{p}_{l,t}$ is then passed through a linear layer Linear_l to transform its dimension into the embedding space of size D:

$$\mathbf{e}_{l,t} = \text{Linear}_l(\mathbf{p}_{l,t}), \tag{3}$$

where $\mathbf{e}_{l,t} \in \mathbb{R}^D$ denotes the embedding of the label prompt for timestamp t.

Boundary Prompts For boundary prompts, the sequence of prompts is denoted as $\mathbf{p}_b = (p_{b,1}, \dots, p_{b,T})$, where T is the total number of timestamps. The correct boundary indicates that a state transition should occur at a specific timestamp, while the incorrect boundary signals that a state transition should not occur, thereby maintaining the current state. Unlike label prompts, either the correct boundary or the incorrect boundary can be provided for a given timestamp, but not both.

To encode boundary prompts, the binary value $p_{b,t} \in \{0, 1\}$ is used to represent the boundary information at each timestamp, where $p_{b,t} = 1$ indicates the correct boundary (a state transition should occur) and $p_{b,t} = 0$ indicates the incorrect boundary (no state transition should occur). This value is mapped into the embedding space of size D:

$$\mathbf{e}_{b,t} = E_b(p_{b,t}), \tag{4}$$

where $E_b(\cdot)$ is a trainable lookup table, and $\mathbf{e}_{b,t} \in \mathbb{R}^D$ denotes the embedding of the boundary prompt for timestamp t.

Combining Label and Boundary Prompt Embeddings To fully incorporate the information from both label prompts and boundary prompts, the embeddings $\mathbf{e}_{l,t}$ and $\mathbf{e}_{b,t}$ for each timestamp t are first adjusted by setting their values to zero when the corresponding prompts are not provided. The adjusted embeddings are then simply added together to form a combined vector:

$$\mathbf{z}_{p,t} = \mathbf{e}_{l,t} + \mathbf{e}_{b,t}. \tag{5}$$

For timestamps where no prompts (neither label nor boundary) are provided, both embeddings are set to 0, and therefore their sum $\mathbf{z}_{p,t}$ is also 0. This approach ensures the model appropriately handles missing prompts while maintaining simplicity in combining the embeddings.

3.3 State Decoder

The goal of the state decoder (f_s) is to map the time series embeddings ($\mathbf{z}_x$) and prompt embeddings ($\mathbf{z}_p$) to the output states ($\mathbf{s}$). To efficiently leverage the information in these embeddings, we employ a two-way Transformer decoder, inspired by SAM (Kirillov et al. 2023) and other Transformer-based segmentation models (Carion et al. 2020; Cheng, Schwing, and Kirillov 2021). The term "two-way" indicates that, unlike traditional Transformer decoders that utilize a single direction of cross-attention, this design incorporates bidirectional cross-attention between the prompt embeddings and the time series embeddings.

In the state decoder, the core unit is the two-way Transformer decoder block. Each block consists of four steps: (1) self-attention is applied to the prompt embeddings ($\mathbf{z}_p$), enabling interaction among prompts; (2) cross-attention is performed, with the prompt embeddings ($\mathbf{z}_p$) acting as queries and the time series embeddings ($\mathbf{z}_x$) serving as keys and values, allowing the prompts to attend to the time series information; (3) a bottleneck MLP refines the prompt embeddings ($\mathbf{z}_p$) by updating their representation; and (4) cross-attention is applied in the opposite direction, where the time series embeddings ($\mathbf{z}_x$) act as queries and the prompt embeddings ($\mathbf{z}_p$) serve as keys and values, incorporating prompt information into the time series embeddings. This two-way interaction ensures that the decoder effectively integrates the time series and prompt embeddings, enabling accurate prediction of the output states $\mathbf{s}$.

3.4 Training

Since PromptTSS is designed as an interactive system where users provide prompts to control the desired output states, the training process must mimic this interactive behavior (Xu et al. 2016; Mahadevan, Voigtlaender, and Leibe 2018). The training is divided into two main stages: pretraining the time series encoder and iterative training for the full interactive system.

Pre-training the Time Series Encoder Before training the interactive system, the time series encoder (f_x) is pretrained to capture robust representations of time series data. In this stage, prompts are not used, as the focus is solely on enhancing the encoding capability of the time series encoder. Inspired by PatchTST (Nie et al. 2023), we adopt Masked Language Modeling (MLM) as the self-supervised pretext task. Specifically, patched time series are masked, and the model is trained to predict the masked values. Given the patched time series $\mathbf{x}_{\text{patched}} \in \mathbb{R}^{T_{\text{patched}} \times (C \cdot P)}$, random masking is applied to create the masked patched time series $\mathbf{x}^{\text{masked}}_{\text{patched}}$. The time series encoder f_x is then applied to the masked patched time series to produce the embeddings:

$$\mathbf{z}_x^{\text{masked}} = f_x(\mathbf{x}^{\text{masked}}_{\text{patched}}), \tag{6}$$

where $\mathbf{z}_x^{\text{masked}} \in \mathbb{R}^{T_{\text{patched}} \times D}$ represents the embeddings of the masked patched time series. A linear projection head $\text{Linear}_{\text{MLM}}$ maps these embeddings back to the original space to reconstruct the masked patches:

$$\hat{\mathbf{x}}^{\text{masked}} = \text{Linear}_{\text{MLM}}(\mathbf{z}_x^{\text{masked}}), \tag{7}$$

where $\hat{\mathbf{x}}^{\text{masked}} \in \mathbb{R}^{T_{\text{patched}} \times (C \cdot P)}$ represents the predicted values for the masked patches. The model is optimized using Mean Squared Error (MSE) between the predicted masked patches and their ground-truth counterparts:

$$\mathcal{L}_{\text{MLM}} = \frac{1}{|\mathcal{M}|} \sum_{t \in \mathcal{M}} \left\| \hat{\mathbf{x}}_t^{\text{masked}} - \mathbf{x}_t^{\text{masked}} \right\|^2, \tag{8}$$

where $\mathcal{M}$ denotes the set of masked patch indices.

Iterative Training for Interactive Behavior After pretraining the time series encoder, all three core modules of PromptTSS (time series encoder f_x, prompt encoder f_p, and state decoder f_s) are trained together iteratively to mimic the interactive system behavior. This stage is designed to help the model effectively utilize user-provided prompts during inference.

At the beginning of each training iteration, no prompts are provided, meaning all prompts are represented as the specialized `NA token`. This ensures the model learns to predict the output states $\mathbf{s} = (s_1, \dots, s_T)$ based solely on the time series embeddings $\mathbf{z}_x$, without relying on any ground-truth information from prompts. The training process then proceeds iteratively. During each iteration, a random subset of prompts is sampled and added to the previously sampled prompts to simulate varying levels of user guidance. Specifically, the number of newly sampled prompts for each iteration is chosen uniformly from a range $(n_{\text{min}}, n_{\text{max}})$. The label prompts ($\mathbf{p}_l$) and boundary prompts ($\mathbf{p}_b$) are uniformly sampled to ensure balanced exposure during training.

Formally, for each iteration r, a subset of prompts $\mathbf{p}_l^{(r)}$ and $\mathbf{p}_b^{(r)}$ is sampled, and the prompt embeddings $\mathbf{z}_p$ are updated accordingly. The model is then optimized using cross-entropy between the ground-truth states s_t and the predicted states $\hat{s}_t$:

$$\mathcal{L}_{\text{iter}} = \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}_s(s_t, \hat{s}_t). \tag{9}$$

In our experiments, we use 8 training iterations, as this was found to be sufficient for real-world use cases. Additional

Table 1: Statistical overview of the 4 datasets for time-series segmentation.

Datasets	# Features	# Timesteps	# Time-Series	# States	State Duration
Pump V35	9	27,770 ∼40,810	40	41 ∼43	1 ∼3,290
Pump V36	9	26,185 ∼38,027	40	41 ∼43	1 ∼1,960
Pump V38	9	20,365 ∼30,300	40	42 ∼43	1 ∼1,820
USC-HAD	6	25,356 ∼56,251	70	12	600 ∼13,500
Pump V35 (2x Coarser)				22	1 ∼3,570
Pump V36 (2x Coarser)				21 ∼22	1 ∼2,085
Pump V38 (2x Coarser)				22	1 ∼3,305
USC-HAD (2x Coarser)				6	1,700 ∼19,100
Pump V35 (4x Coarser)				11	1 ∼3,855
Pump V36 (4x Coarser)				11	1 ∼4,874
Pump V38 (4x Coarser)				11	1 ∼4,154

Table 2: Segmentation performance of PromptTSS vs. the baseline (without prompts) on multi-granularity datasets.

Dataset	Pump V35 (1x + 2x + 4x, Coarser)			Pump V36 (1x + 2x + 4x, Coarser)			Pump V38 (1x + 2x + 4x, Coarser)			USC-HAD (1x + 2x, Coarser)		
Metric / Model	ACC	MF1	ARI	ACC	MF1	ARI	ACC	MF1	ARI	ACC	MF1	ARI
PromptTSS	**0.912**	**0.795**	**0.846**	**0.909**	**0.832**	**0.839**	**0.894**	**0.78**	**0.814**	**0.952**	**0.928**	**0.914**
PromptTSS (w/o prompting)	0.25	0.154	0.126	0.271	0.14	0.149	0.247	0.152	0.119	0.478	0.361	0.321

Table 3: Segmentation performance of PromptTSS vs. the baseline (without prompts) on single-granularity datasets.

Dataset	Pump V35			Pump V36			Pump V38			USC-HAD		
Metric / Model	ACC	MF1	ARI	ACC	MF1	ARI	ACC	MF1	ARI	ACC	MF1	ARI
PromptTSS	**0.923**	**0.738**	**0.889**	**0.898**	**0.729**	**0.853**	**0.889**	**0.71**	**0.836**	**0.981**	**0.978**	**0.965**
PromptTSS (w/o prompting)	0.668	0.517	0.616	0.734	0.574	0.636	0.725	0.587	0.64	0.771	0.734	0.633

iterations were tested but did not yield significant performance gains. This iterative approach allows the model to gradually learn how to effectively incorporate varying levels of user-provided prompts, enabling fine-grained control over state granularity and improving the model's adaptability to unseen patterns during inference.

4 Experiments

Datasets We use four real-world datasets to validate the effectiveness of PromptTSS. The statistics of these datasets are summarized in Table 1. Three of these datasets are derived from industrial manufacturing operations, while the remaining one focuses on human activity recognition. **Pump** (Gaugel and Reichert 2023) contains multivariate time series data from End-of-Line (EoL) testing of hydraulic pumps. It consists of three subsets corresponding to different pump types: V35, V36, and V38. Each subset includes sensor readings from testing cycles, with samples annotated by operational states. **USC-HAD** (Zhang and Sawchuk 2012) contains data from human activities, such as standing, walking, and sitting, recorded using a 3-axis accelerometer and a 3-axis gyrometer attached to the front of the hip.

To create datasets with multiple levels of state granularity, we systematically merge adjacent states in the original fine-grained states. For the Pump V35, Pump V36, and Pump V38 datasets, we generate both 2$\times$ and 4$\times$ coarser versions by progressively grouping consecutive states. For the USC-HAD dataset, we only create a 2$\times$ coarser version because the original dataset contains only 12 states, and further merging would result in an excessively small number of states, limiting meaningful segmentation analysis.

Metrics To evaluate the segmentation performance of PromptTSS, we use three metrics: Accuracy (ACC), Macro F1-score (MF1), and Adjusted Rand Index (ARI). These metrics provide a comprehensive assessment of the model's ability to correctly predict states, balance performance across classes, and capture clustering consistency.

4.1 Effectiveness of Prompting

To evaluate the impact of the prompting mechanism in PromptTSS, we conduct two experiments comparing the full PromptTSS model (with prompts) to a baseline version without prompts. The baseline model still has three core components—a time series encoder, a prompt encoder, and a state decoder—but no ground truth information is provided as prompts. This comparison highlights how incorporating label and boundary prompts affects segmentation performance. We evaluate both models on datasets from two distinct scenarios: one involving multi-granularity states, where datasets contain both coarse-grained trends and fine-grained events, and the other involving single-granularity states, where datasets are limited to a single level of granularity. These scenarios provide a comprehensive assessment of the models' abilities to handle complex, multi-scale segmentation tasks and simpler, more uniform tasks.

Multi-Granularity States In Table 2, we evaluate datasets containing multiple granularities of states, such as coarse-grained trends and fine-grained events. These datasets challenge the model to balance broad, long-term patterns with localized, short-term transitions. The Pump dataset (V35, V36, and V38) includes three levels of granularity: the original (1x), 2x coarser, and 4x coarser. For the USC-HAD dataset, there are two levels of granularity: the original (1x) and a 2x coarser level. In this setting, PromptTSS consistently outperforms the baseline, leveraging its prompting mechanism to adaptively guide segmentation at the desired granularity. Label prompts help the model focus on target states, while boundary prompts enable better localization of state transitions. In contrast, the baseline struggles to capture both granularities simultaneously, often missing fine-grained details or over-segmenting coarse patterns. These results highlight the effectiveness of prompting in handling complex multi-granularity scenarios.

Single-Granularity States In Table 3, we evaluate datasets with single granularity states, where the segmentation task involves only the original level of granularity without considering coarser or finer variations. This setup simplifies the task by requiring the model to focus on consistent, uniform state segmentation, avoiding the complexity introduced by multiple granularity levels. For this evaluation, we use the Pump dataset (V35, V36, and V38) and the USC-HAD dataset, focusing solely on their original granularity (1x). In this scenario, PromptTSS still outperforms the baseline, although the margin of improvement is smaller compared to multi-granularity experiments. By leveraging label and boundary prompts, the model refines its predictions and maintains high segmentation accuracy, particularly in challenging cases with subtle transitions or noise. Label prompts guide the model toward the correct states, while boundary prompts ensure precise localization of state transitions.

Observations Across both experiments, the results clearly show that prompts substantially improve segmentation performance, especially in datasets with multi-granularity states where the task demands balancing coarse-grained trends and fine-grained events. The prompting mechanism enables PromptTSS to adapt dynamically to varying levels of granularity, making it particularly effective in handling the complexities of multi-granularity scenarios. In single-granularity tasks, where the segmentation is more straightforward, prompts still provide noticeable benefits by enhancing accuracy in noisy or ambiguous regions. These findings highlight the critical role of prompts in improving both the precision and adaptability of PromptTSS across diverse segmentation challenges.

5 Conclusion

In this paper, we introduce PromptTSS, a novel framework for time series segmentation that tackles two key challenges: capturing states across multiple levels of granularity and adapting to evolving patterns in dynamic environments. PromptTSS integrates a unified model with a prompting mechanism that dynamically incorporates label and boundary information to guide segmentation. Its three core components—a time series encoder, a prompt encoder, and a state decoder—work together to accurately segment coarse-grained trends and fine-grained events. Experiments demonstrate its effectiveness in handling multi-granularity states and adapting to unseen patterns, offering a flexible solution for complex time series data.

Future work will focus on completing the remaining experiments to answer two critical research questions: Can PromptTSS handle multi-granularity states using a unified model, and can it adapt to unseen patterns using prompts? These evaluations will ensure the framework's robustness and establish it as a state-of-the-art solution for time series segmentation.

References

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In *European conference on computer vision*, 213–229. Springer.

Cheng, B.; Schwing, A.; and Kirillov, A. 2021. Per-pixel classification is not all you need for semantic segmentation. *Advances in neural information processing systems*, 34: 17864–17875.

Gaugel, S.; and Reichert, M. 2023. PrecTime: A deep learning architecture for precise time series segmentation in industrial manufacturing operations. *Engineering Applications of Artificial Intelligence*, 122: 106078.

Gong, Y.; Chung, Y.-A.; and Glass, J. 2021. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*.

Hallac, D.; Vare, S.; Boyd, S.; and Leskovec, J. 2017. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 215–223.

Hammami, Z.; Sayed-Mouchaweh, M.; Mouelhi, W.; and Ben Said, L. 2020. Neural networks for online learning of non-stationary data streams: a review and application for smart grids flexibility improvement. *Artificial Intelligence Review*, 53(8): 6111–6154.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4015–4026.

Kwapisz, J. R.; Weiss, G. M.; and Moore, S. A. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2): 74–82.

Li, S.; Farha, Y. A.; Liu, Y.; Cheng, M.-M.; and Gall, J. 2020. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 45(6): 6647–6658.

Mahadevan, S.; Voigtlaender, P.; and Leibe, B. 2018. Iteratively trained interactive segmentation. *arXiv preprint arXiv:1805.04398*.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*. OpenReview.net.

Ordóñez, F. J.; and Roggen, D. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1): 115.

Perslev, M.; Jensen, M.; Darkner, S.; Jennum, P. J.; and Igel, C. 2019. U-time: A fully convolutional network for time series segmentation applied to sleep staging. *Advances in Neural Information Processing Systems*, 32.

Phan, H.; Chén, O. Y.; Koch, P.; Lu, Z.; McLoughlin, I.; Mertins, A.; and De Vos, M. 2020. Towards more accurate automatic sleep staging via deep transfer learning. *IEEE Transactions on Biomedical Engineering*, 68(6): 1787–1798.

Reis, M. S. 2019. Multiscale and multi-granularity process analytics: A review. *Processes*, 7(2): 61.

Wang, C.; Wu, K.; Zhou, T.; and Cai, Z. 2023. Time2state: An unsupervised framework for inferring the latent states in time series data. *Proceedings of the ACM on Management of Data*, 1(1): 1–18.

Xu, N.; Price, B.; Cohen, S.; Yang, J.; and Huang, T. S. 2016. Deep interactive object selection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 373–381.

Yang, Y.; Bom, S.; and Shen, X. 2024. A hierarchical ensemble causal structure learning approach for wafer manufacturing. *Journal of Intelligent Manufacturing*, 35(6): 2961–2978.

Zhang, M.; and Sawchuk, A. A. 2012. USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, 1036–1043.