Pattern-based Time Series Semantic Segmentation with Gradual State Transitions

Louis Carpentier *† Len Feremans ‡ Wannes Meert * Mathias Verbeke *†

Abstract

Time series semantic segmentation is the task of extracting time intervals from the time series data that share a similar meaning within the application domain in an unsupervised manner. State-of-the-art algorithms typically treat this problem as change point detection, resulting in discrete state transitions. However, in real-world applications, states often transition gradually. This leads to a novel, more challenging variation of the traditional time series segmentation task, for which we present PaTSS, a novel, domain-agnostic algorithm to uncover those gradual state transitions. PaTSS learns a distribution over the semantic segments based on an embedding space derived from mined sequential patterns. An extensive experimental evaluation on 107 benchmark time series shows that PaTSS is capable of detecting gradual state transitions, a task current methods are unable to perform.

Keywords. Time series analysis, semantic segmentation, gradual transition, pattern-based embedding

1 Introduction

In recent years, the wide availability of low-cost high resolution sensors has led to a dramatic increase in monitoring capabilities. This trend is noticeable in various domains such as industry [14], the biomedical sector [4, 24] and human activity recognition [5, 18]. The monitored process is typically composed of multiple states, i.e., time intervals that share a similar meaning in the application domain, such as the operating regime of a machine. *Time series semantic segmentation* aims at automatically uncovering these hidden states from the data in an unsupervised manner. Learning those states enables the incorporation of contextual information in downstream time series analysis tasks such as anomaly detection or forecasting.

For example, in human activity recognition, semantic segmentation tries to recognize intervals that correspond to different activities like *sitting* or *standing* [5]. Another example is the worldwide interest in the search term 'mail' in Google during one week, shown at the top in Figure 1. There are three semantic segments:

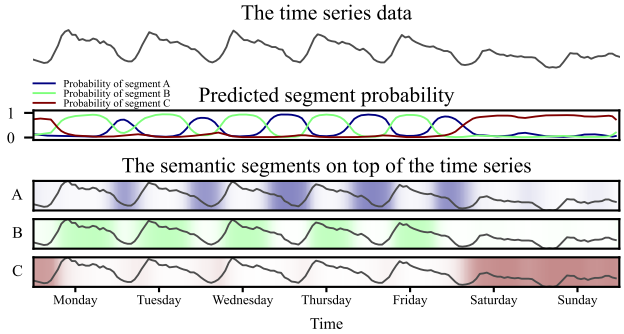


Figure 1: The worldwide interest in the search term 'mail' in Google from Monday 20/02/23 until Sunday 26/02/23. (https://trends.google.com)

(A) working hours with high interest, (B) nights during the week with brief decreased interest, and (C) weekend with longer decreased interest. While humans trivially recognize the three states, their automatic detection in an unsupervised manner is nontrivial.

State-of-the-art segmentation methods are based on change point detection and determine specific time points where a transition occurs by measuring the dissimilarity between the data before and after the time point [7, 11, 19, 22]. However, change points represent sudden state transitions, assuming *discrete* hidden states. This assumption seldomly holds in real-world applications. For human activity recognition, people gradually transition from *sitting* to *standing*. The global interest in 'mail' does not show any abrupt changes, but gradual transitions from week to weekend and from working hours to night. Ermshaus et al. [7] recently noted that such complex phenomena are "outside the scope of *current* time series segmentation methods".

We propose a novel variation of the traditional semantic segmentation problem to account for gradual transitions. We achieve this through Pattern-based Time Series Segmentation (PaTSS), our novel time series semantic segmentation method. PaTSS learns a dynamic probability distribution over the semantic segments, which is illustrated in the center of Figure 1. This enables the identification of gradual transitions, defined as intervals where the likelihood of one seman-

*KU Leuven, Belgium
†Flanders Make@KU Leuven, Belgium
‡University of Antwerp, Belgium
louis.carpentier@kuleuven.be, len.feremans@uantwerpen.be, wannes.meert@kuleuven.be, mathias.verbeke@kuleuven.be

tic segment decreases while the likelihood of another semantic segment increases. PaTSS mines frequent sequential patterns to capture the typical shapes of the time series. The sequential patterns form a new feature space in which the time series values are mapped, which is used to learn the likelihood of each semantic segment occurring per time point. In addition to capturing gradual transitions, the probability distribution inherently models reoccurring behavior, which is impossible when discrete segment boundaries are used. The main contributions of this work are as follows:

- A novel variation of time series semantic segmentation to explicitly account for gradual transitions;
- PaTSS, a novel method that computes a semantic segmentation as the likelihood of each segment at each time point, thus learning gradual transitions;
- An objective measure to quantify algorithm performance for time series segmentation with gradual state transitions. This is necessary because current evaluation metrics only quantify how well an algorithm predicts a specific time point, thus assuming discrete state transitions;
- A thorough experimental evaluation on real-world benchmark time series, as well as synthetic time series with gradual state transitions;
- An open source implementation of the methodology, including the source code, raw experimental results and synthetic data generator[1].

2 Related work

2.1 Time series semantic segmentation Traditional approaches were developed for specific applications [5, 18]. In contrast, recent methods offer domain-independent solutions. AutoPlait [19], FLOSS [11], and ClaSP [7, 22] partition the time series by measuring the likelihood of each time point being a segment boundary. An alternative perspective involves clustering similar time intervals. This approach offers the advantage of detecting reoccurring patterns. Gionis and Mannila [12] present four algorithms to segment a time series into k segments originating from h sources, with $h < k$. These methods initially identify k individual segments and differ in their subsequent clustering of these k segments into h clusters.

To the best of our knowledge, all time series semantic segmentation methods aim at finding *discrete* state transitions. Ermshaus et al. [7] acknowledge this limitation and argue that gradual state transitions are outside the scope of *current* time series segmentation methods. In contrast, PaTSS learns a probability distribution over the various segments. This distribution provides a gradual state transition when the probability of one semantic segment decreases over time, while the probability of another segment increases.

2.2 Sequential patterns for time series analysis Sequential pattern mining has been proposed for various time series analysis tasks. PBAD [10] uses frequent sequential patterns to identify anomalies in mixed-type time series. PETSC [9], MrSEQL [16], and MrSQM [20] perform time series classification by constructing an embedding of the time series based on the occurrence of sequential patterns. Sequential patterns have three main advantages: (1) they are more expressive than the raw time series data as they capture the relation between consecutive measurements, (2) sequential patterns are extremely interpretable and allow for explainable predictions, and (3) sequential patterns can be mined in continuous time series and event logs. PaTSS also offers these advantages by using frequent sequential patterns.

3 Preliminaries

3.1 Terminology A time series $\mathcal{T} = (x_1, \ldots, x_n)$ is an ordered sequence of $n \in \mathbb{N}$ values $x_i \in \mathbb{R}^p$. If $p = 1$, $\mathcal{T}$ is univariate, otherwise $\mathcal{T}$ is multivariate.

A subsequence $[x_t, x_{t+1}, \ldots, x_{t+l-1}]$ of $\mathcal{T}$ equals l consecutive measurements starting at time t. A sliding window with window length l extracts all subsequences of size l from $\mathcal{T}$. Given word size w and alphabet size α, two discretization parameters, SAX [17] discretizes continuous sequences into symbolic sequences of length w using α distinct symbol. For instance, SAX transforms subsequence $[1.0, 1.3, 1.7, 1.5, 1.2, 0.9]$ into word *'abccba'* with $w = 6$ and $\alpha = 3$. Applying SAX to all subsequences of a time series results in a collection of such symbolic words.

A sequential pattern P is an ordered sequence of symbols. Given a list of discrete symbolic words, $cover(P)$ equals the subset of words that cover P. For instance, pattern *'abc'* is covered by word *'abccba'* but not by *'abbcca'*. The relative support $sup_{rel}(P)$ of P equals the percentage of words that cover P. Frequent sequential pattern mining aims at mining the patterns with the highest relative support.

3.2 Problem formulation We assume that the time series is composed of multiple distinct states, which result in (slightly) different measurements [7, 11, 22].

State-of-the-art defines a semantic segmentation as boundaries separating two states [7, 11, 22]. However, this definition is too limiting as it does not cover gradual transitions. This work removes this obligation and defines the semantic segmentation task as follows:

[1] code: `https://gitlab.kuleuven.be/u0143709/patss`
data: `https://doi.org/10.48804/G2YRDR`

DEFINITION 1. *Given a time series $\mathcal{T}$, for every timestamp t and state s, infer the probability $P(s,t)$ of state s occurring at time t.*

This work considers a probability distribution to model a semantic segmentation, as it offers the most natural way to express gradual transitions through an increasing and decreasing likelihood.

4 Pattern-based Time Series Segmentation

In this section, we describe PaTSS, our proposed semantic segmentation algorithm. PaTSS performs a semantic segmentation with gradual state transitions through the following steps (depicted in Figure 2):

1. Mine frequent sequential patterns in symbolic representations of the time series.
2. Embed the time series values using the mined frequent sequential patterns.
3. Learn the likelihood of each semantic segment occurring at each time point.

We discuss each step in detail below. Every attribute is handled independently in multivariate time series, and the individual pattern-based embeddings are concatenated in step 2.

4.1 Frequent pattern mining PaTSS first converts the time series $\mathcal{T}$ into a list of symbolic words through SAX. Similar to related work [9, 16], PaTSS extracts subsequences at multiple resolutions, employing multiple sliding windows with window sizes l, $2l$, $4l$, ..., and so on, until the window size exceeds the length of the time series or reaches a maximum size. We choose l equal to word size w, which means that for the smallest window size every symbol corresponds to a single value of the time series, but as the window size increases, more and more consecutive values are aggregated. Consequently, words corresponding to smaller window sizes approach the original data, while words corresponding to large window sizes capture the global behavior.

Frequent sequential patterns are mined in the symbolic representations, for which we leverage the mature field of frequent pattern mining with highly optimized algorithms [2]. Specifically, we use the implementation of Feremans et al. [8] to mine the top-k most frequent patterns with the highest relative support.

We incorporate a constraint $rdur$ on the relative duration of a pattern P to determine if a word covers it [9]. More precisely, a word covers P if the word includes a sequence of length at most $\lfloor |P| \cdot rdur \rfloor$ with the symbols of P in order. This accommodates a variable number of gaps depending on the length of P. For instance, a pattern of length 5 with $rdur = 1.2$ has a maximal allowed duration of $\lfloor 5 \cdot 1.2 \rfloor = 6$. If the pattern

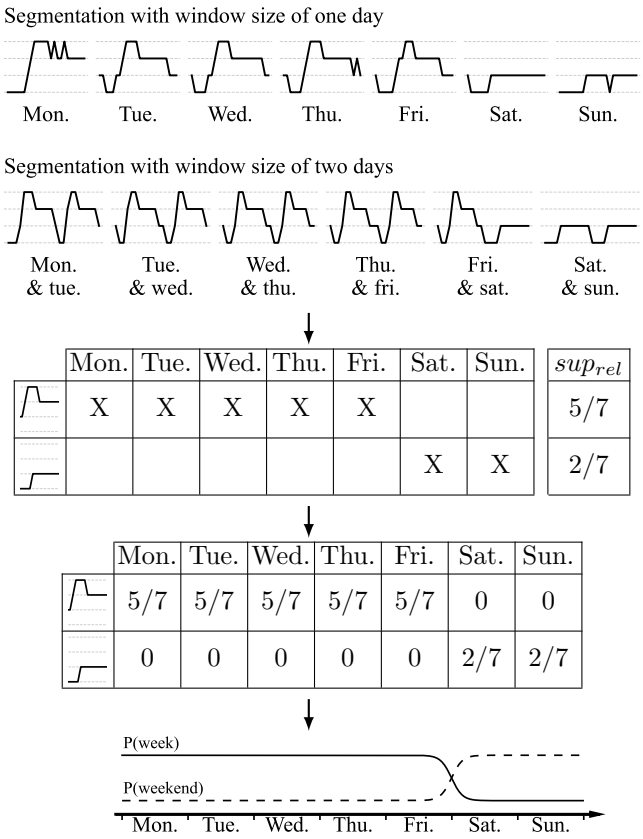


Figure 2: The different steps in PaTSS to identify semantic segments with gradual state transitions in the time series from Figure 1.

consists of 20 symbols, the maximal duration equals 24.

Two co-occurring patterns P_1 and P_2 do not provide much additional information. We utilize the Jaccard index $J(P_1, P_2)$ to quantify the amount of overlap between P_1 and P_2, which is computed as [3]:

$$J(P_1, P_2) = \frac{|cover(P_1) \cap cover(P_2)|}{|cover(P_1) \cup cover(P_2)|} \tag{4.1}$$

If $J(P_1, P_2)$ exceeds a predefined threshold λ, we remove the pattern with the lowest relative support.

4.2 Pattern-based embedding Every measurement $x_t \in \mathcal{T}$ is embedded into a new space, in which each dimension corresponds to a frequent sequential pattern. We compute embedding E_t of x_t according to Equation 4.2, with $cover(t)$ denoting the set of subsequences that cover time t. We multiply with $sup_{rel}(P)$ to focus on frequent patterns.

$$E_t(P) = \frac{|cover(P) \cap cover(t)|}{|cover(t)|} \cdot sup_{rel}(P) \tag{4.2}$$

Embedding E_t is a high-dimensional pattern-based feature vector. In semantic segmentation, interesting

patterns cover only one or a limited number of semantic segments. Thus, the values $E_t(P)$ of an interesting pattern P show a lot of variability. Therefore, we reduce the dimension of this embedding by selecting the d patterns with the highest variance. This approach prunes patterns with low frequency ($E_t(P)$ consistently approaches 0) as well as patterns with very high frequency ($E_t(P)$ consistently approaches 1). While the latter patterns are valuable for frequent pattern mining, they can not distinguish distinct semantic segments.

4.3 Semantic Segmentation. PaTSS learns the probability $\hat{P}(s,t)$[2] of semantic segment $s \in \mathcal{S}$ at time t, with $\mathcal{S}$ the set of all (possibly reoccuring) semantic states. The evolution of $\hat{P}$ over time is depicted in the center of Figure 1. This probability distribution models gradual state transitions as time intervals where probability $\hat{P}(s_1,t)$ decreases and $\hat{P}(s_2,t)$ increases for semantic segments s_1 and s_2. Furthermore, based on the distribution $\hat{P}$ it is possible to identify reoccurring segments, where the probability of a segment starts high, decreases over time as another semantic segment becomes more probable, and eventually increases again in likelihood.

We propose logistic regression [13] with supervision from k-means clustering to learn $\hat{P}$. The coefficients of the logistic regression model are learned by minimizing the negative log likelihood loss using a one-versus-all scheme. Elastic net is used as regularization [26], which integrates both L1 and L2 regularization.

PaTSS automatically determines the number of segments in an unsupervised manner using the silhouette method [21]. It constructs multiple segmentations with varying numbers of semantic segments and subsequently selects the segmentation with the highest average silhouette score.

5 Experimental validation

This section describes the experiments performed to evaluate PaTSS. We evaluate both on gradual and discrete transitions because state-of-the-art semantic segmentation algorithms only focus on discrete state transitions. First, we quantitatively evaluate PaTSS and its competitors, after which we qualitatively illustrate PaTSS' performance. The source code, benchmarks, experimental setups, raw results, visualizations, and notebooks used for processing the data are publicly available to guarantee the reproducibility of the results.

5.1 Data sets Extensive research in time series semantic segmentation has resulted in many benchmark datasets containing time series with labeled segment boundaries. We use all 32 time series from the UCR Time Series Semantic Segmentation Archive (UTSA) [11] and all 75 time series from the Time Series Segmentation Benchmark (TSSB) [7, 22]. These data sets are only used to assess PaTSS' capabilities for change point detection (Section 5.6) because only discrete segment boundaries are labeled. Further, no objective measure exists, independent of any segmentation algorithm, to transform the labeled segment boundaries into labels for gradual state transitions.

To assess the performance for gradual state transitions, we developed a publicly available synthetic data generator. The generator starts by defining a set of states $\mathcal{S}$, with each state $s \in \mathcal{S}$ consisting of a sequence of repetitions of some random shape. Next, the generator randomly computes several (discrete) segment boundaries, and each segment is associated with a state. The time series has reoccurring behavior if one state is associated with more than one segment. A sigmoid function interpolates between two consecutive segments to guarantee smooth transitions. The generator computes the ground truth distribution $P(s,t)$ over the different states $\mathcal{S}$. When no transition is occurring at time t, $P(s,t) = 1$ for the active state $s \in \mathcal{S}$ and $P(s',t) = 0$ for all other states $s' \in \mathcal{S} \setminus \{s\}$. When there is a transition, the probability of the first segment decreases over time while the probability of the second segment increases, according to the interpolating sigmoid. A synthetic time series is shown in Figure 7. We generated 1000 synthetic time series with various properties (e.g., length, number of states, number of segments, ...), of which we use 500 for tuning PaTSS (Sections 5.3 and 5.4) and the remaining 500 for validation (Section 5.5) to avoid overfitting.

5.2 Benchmark methods and evaluation metrics For *discrete* state transitions, we compared PaTSS with three state-of-the-art semantic segmentation methods: ClaSP [7, 22], FLOSS [11] and AutoPlait [19]. Additionally, three change point detection methods were used for comparison: BOCD [1], BinSeg [23], and Window-L_2 [25]. The scoring function proposed in [11] is used to assess algorithm performance for semantic segmentation with discrete state transitions. Formally, for ground truth change points b and predicted change points $\hat{b}$, the loss is computed as follows:

$$loss = \frac{1}{n \cdot |\hat{b}|} \cdot \sum_{\hat{p} \in \hat{b}} \min_{p \in b} |p - \hat{p}| \tag{5.3}$$

Intuitively, this loss measures the average distance of a predicted change point $\hat{b}$ to the nearest true change point b. To compute this loss for PaTSS, we convert the learned distribution $\hat{P}$ to a sequence of discrete segment

[2] Distribution $\hat{P}$ is an estimate of the true probability P.

labels, in which each time t is assigned to the segment s with maximum likelihood $\hat{P}(s,t)$. There is a segment boundary at time t if the segment label at time t differs from the label at time $t-1$.

For *gradual* state transitions, we compare PaTSS with ClaSP and FLOSS, the only algorithms that significantly outperform PaTSS on discrete state transitions (see Figure 6). To evaluate the methods on gradual state transitions, we propose a score based on the mean absolute error. For a gradual transition with ground truth probability $P(s,t)$ that transitions from state s_1 to s_2 and spans time interval T, and estimated probability $\hat{P}(s,t)$ the loss is computed as follows:

$$loss = \frac{1}{|T|} \cdot \sum_{t \in T} |P(s_1,t) - \hat{P}(s_1,t)| + |P(s_2,t) - \hat{P}(s_2,t)| \tag{5.4}$$

Figure 3 visually illustrates this loss. Intuitively, we compute the normalized distance between ground truth distribution P and predicted distribution $\hat{P}$. The loss is averaged over all transitions.

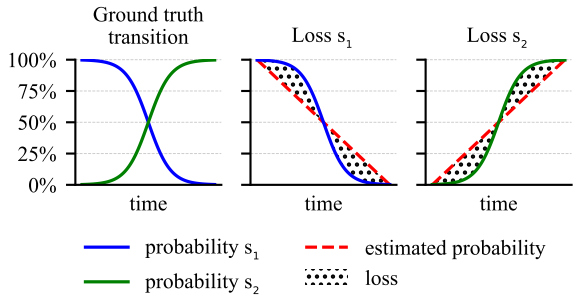


Figure 3: Visual explanation of the loss in Equation 5.4 for a gradual transition from state s_1 to state s_2.

The runtime of PaTSS ranges from a few seconds to a minute, depending on the time series, comparable to state-of-the-art time series segmentation techniques ClaSP and FLOSS [7, 11].

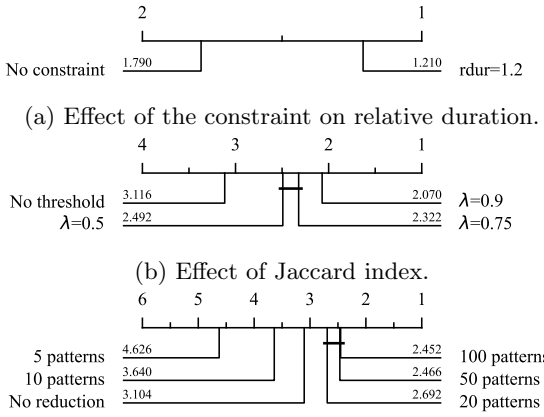
5.3 Sensitivity analysis We analyse the impact of three key parameters of PaTSS: word size w, alphabet size α and the top-k patterns to mine, using the 500 synthetic time series generated for tuning. Our experimentation covers a range of values: $w \in \{8, 16, 32\}$, $\alpha \in \{3, 5, 7, 15\}$ and $k \in \{25, 100, 500, 10\,000\}$.

Typically, a pattern consists of 4 or 5 symbols, covering around 30% of the subsequence when $w = 16$. When the word size is increased, the patterns become less representative as they cover only a small portion of the subsequence. Conversely, a smaller word size is overly restrictive, requiring an almost exact match. Therefore, we fix the word size to $w = 16$.

For larger values of α, the discretized time series closely resembles the original time series, but it also increases the pattern space. This leads to an average loss of 0.453 for $\alpha = 7$ and 0.510 for $\alpha = 15$. The performance improves for smaller values of α, with an average loss of 0.411 for $\alpha = 3$ and 0.419 for $\alpha = 5$. The optimal choice depends on the specific time series characteristics, however. We fix $\alpha = 3$ for the experiments in Sections 5.5 and 5.6, because of its smaller loss, but we adapt it for the qualitative evaluation in Section 5.7.

PaTSS mines the top-k patterns, but only a few patterns remain after pruning. Increasing k beyond 100 has no impact because the same patterns persist after pruning, yielding identical embeddings. However, some of the remaining patterns are not among the top 25 most frequent patterns. Consequently, for $k = 25$, the embedding differs, and the segmentation as well. The difference is, however, minimal with an average loss of 0.524 for $k = 25$ and 0.535 for $k \geq 100$. We propose $k = 25$ as it does not yield improved performance to mine more patterns, and a smaller k allows for stronger pruning by the pattern mining algorithms.

5.4 Ablation study PaTSS incorporates several components to calculate the semantic segmentation. Their significance is illustrated through critical difference diagrams [6] in Figure 4. Below we discuss each component in more detail.



(a) Effect of the constraint on relative duration.

(b) Effect of Jaccard index.

(c) Effect of reducing the dimension of the embedding.

Figure 4: Ablation study on 500 synthetic time series.

One key element involves introducing a constraint on the relative duration of a pattern, which allows a variable number of gaps depending on the pattern size [9]. We have chosen a value of $rdur = 1.2$ to accommodate for some gaps without being overly permissive. This constraint effectively mitigates approximation errors in

the SAX discretization, leading to significant improvements, as illustrated in Figure 4a.

Utilizing the Jaccard index to measure the similarity between two patterns is also critical. This metric helps to eliminate patterns that cover the same segments, yielding significant improvements as shown in Figure 4b. Without using the Jaccard index ($\lambda = 1$), many patterns correspond to only one semantic segment, resulting in an average loss of 0.478. The amount of tolerated overlap decreases as λ decreases. For $\lambda = 0.9$, the segmentation performance increases, with an average loss of 0.386. However, the performance decreases if λ becomes too small because it is less likely that two different patterns reinforce the same semantic segment, leading to an average loss of 0.411 for $\lambda = 0.5$ and 0.396 for $\lambda = 0.75$. Therefore, we select $\lambda = 0.9$ because it offers the best trade-off.

After creating the pattern-based embedding, its dimension is reduced by selecting the top-d patterns with the highest variance, yielding significant improvements as demonstrated in Figure 4c. While PaTSS effectively handles a large dimensional embedding and remains robust against various choices of d, it encounters difficulties in distinguishing the semantic segments if d is too small. Setting $d = 20$ strikes a balance between maintaining enough dimensions and allowing straightforward interpretation.

5.5 Performance for gradual state transitions

We first discuss the results of PaTSS, ClaSP, and FLOSS on a test set consisting of 500 synthetic time series with *gradual* state transitions, because this is our focus. The proposed evaluation metric requires a probability distribution to evaluate a method. Therefore, the segment boundaries of ClaSP and FLOSS must be converted to a distribution. For predicted segment boundaries $\hat{b}$, we construct $|\hat{b}| + 1$ semantic segments. Each segment is assigned to a time interval demarcated by two consecutive boundaries. Similar to the approach adopted to introduce gradual transitions in the synthetic data generator, an interpolating sigmoid is imposed to arrive at the transition probabilities. The length of this interpolating sigmoid is chosen to minimize the average loss over all test instances, thus reducing the transformation penalty on the evaluation score for ClaSP and FLOSS. Further, we examine the effect of providing the ground truth number of segments and optimal window size to both ClaSP and FLOSS, which we mark with a GT subscript. We also compare ClaSP without providing ground truth labels, as it can learn these from the data. Gharghabi et al. [11] discusses methods to extend FLOSS to learn these ground truth labels and their robustness, but these are not part of the reference implementation in `STUMPY` [15]. We do, however, implement a temporal constraint for FLOSS, as described in [11], to cope with reoccurring behavior. This is not necessary for ClaSP, which automatically learns an ensemble of multiple random constraints [7].

The critical difference diagram for segmentations with gradual state transitions is shown in Figure 5, illustrating that PaTSS significantly outperforms the benchmark methods. The results are detailed in Table 1, which shows a summary of the number of wins, losses, and ties of PaTSS compared to the competitor approaches. PaTSS computes a better segmentation than ClasP and FLOSS on 442 and 470 instances (out of 500) respectively.

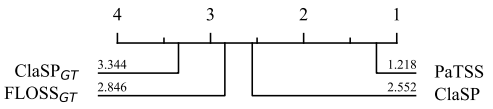


Figure 5: Ranking of PaTSS, ClaSP and FLOSS for semantic segmentation on 500 synthetic time series with gradual state transitions.

	ClaSP	ClaSP$_{GT}$	FLOSS$_{GT}$
PaTSS	442 / 58 / 0	479 / 21 / 0	470 / 30 / 0

Table 1: Summary of the wins/losses/ties of PaTSS compared to ClaSP and FLOSS on 500 synthetic time series with gradual transitions. For each individual time series, PaTSS wins if the loss in Equation 5.4 is smaller than the loss of the competitor.

Table 2 summarizes the number of transition areas (i.e., intervals where two semantic segments have non-zero ground truth probability) in which a segment boundary was identified by the different methods. The first row shows the true positive rate (TPR), i.e., the transitions that were correctly identified. The second row equals the false positive rate (FPR), i.e., the segment boundaries that do not correspond to a transition. We see that PaTSS has a much larger TPR and identifies almost all transition areas, while the competitors seem to struggle to identify the gradual transitions. This does come at a slightly increased FPR, however.

5.6 Performance for discrete state transitions

Although PaTSS aims at identifying gradual state transitions, we want to assess its performance on discrete transitions with methods that were developed for this task. Figure 6 shows the critical difference diagram of this comparison. For PaTSS, there is uncertainty at the transitions, because the focus is on identifying grad-

	PaTSS	ClaSP	ClaSP$_{GT}$	FLOSS$_{GT}$
TPR	**4 202**	2 958	2 237	2 870
	(97.11%)	(68.36%)	(51.7%)	(66.33%)
FPR	2 230	1 811	1 627	**1 543**
	(51.54%)	(41.85%)	(37.6%)	**(35.66%)**

Table 2: The number of transition areas in which a segment boundary was detected out of 4 327 transitions (first row) and the number of segment boundaries that did not correspond to a transition (second row) in 500 synthetic time series with gradual transitions.

ual transitions. Therefore, PaTSS has trouble identifying the exact location of a segment boundary. Nevertheless, for 81, 66% the transitions, PaTSS identifies a segment boundary within a margin of 1% of the total time series length from b. ClaSP and FLOSS, on the other hand, are better at detecting the exact segment boundary, resulting in a significantly higher rank than PaTSS and the other competitors. Nevertheless, PaTSS is competitive with existing change point detection algorithms Windows-L_2 and BOCD, and even significantly outperforms BinSeg-L_2. Additionally, PaTSS significantly outperforms AutoPlait.

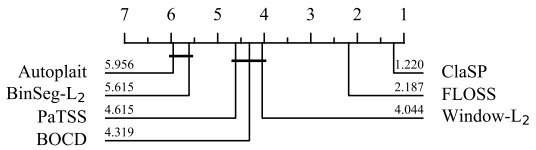


Figure 6: Ranking of PaTSS and its competitors for semantic segmentation on the UTSA [11] and TSSB [22, 7] datasets with discrete state transitions.

	ClaSP	FLOSS	Window-L_2
PaTSS	0 / 91 / 0	4 / 87 / 0	35 / 56 / 0

	BCOD	BinSeg-L_2	AutoPlait
PaTSS	41 / 50 / 0	69 / 22 / 0	68 / 23 / 0

Table 3: Summary of the wins/losses/ties of PaTSS compared to its competitors on the UTSA [11] and TSSB [22, 7] datasets with discrete state transitions. For each individual time series, PaTSS wins if the loss in Equation 5.3 is smaller than the loss of the competitor.

5.7 Qualitative evaluation Next to the quantitative evaluation, we qualitatively illustrate the semantic segmentations to provide more insights into the behavior and performance of PaTSS, beyond the numerical

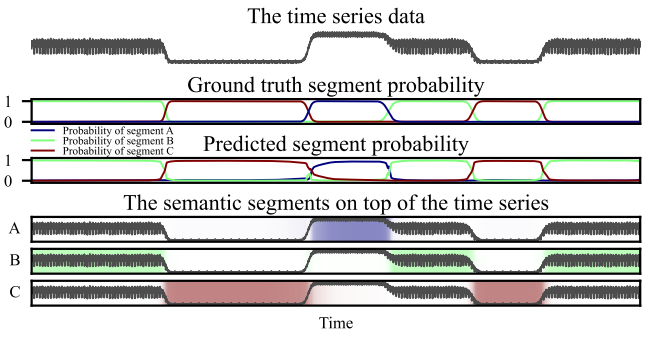


Figure 7: Semantic segmentation of a synthetic time series.

analysis described above. Figures 7, 8 and 9 illustrate the semantic segmentations of various time series. The time series data is shown at the top. In Figure 7, the second row illustrates the ground truth probabilities $P(s,t)$ over time. This ground truth is not available for the other time series, as these do not originate from the synthetic data generator and only discrete transition labels are available. Next, the predicted probabilities $\hat{P}(s,t)$ are plotted. Finally, the identified semantic segments are shown on top of the time series data. PaTSS uses tuned parameters as described above, and alphabet size $\alpha = 5$ for the time series in Figures 9 and 8.

A synthetic time series is shown in Figure 7, for which we have labeled *gradual* transitions. It illustrates the two major advantages of PaTSS. First, the predicted distribution is similar to the ground truth distribution, which includes gradual transitions. This shows PaTSS' capabilities for identifying semantic segments with gradual state transitions. Methods based on change points may compute segment boundaries in the transition areas, but do not properly model what is truly happening in the data. Second, PaTSS correctly identifies the reoccurring behavior. State-of-the-art semantic segmentation methods typically have no built-in mechanism to detect this, and can only identify reoccurring behavior through an additional postprocessing step.

Figure 8 illustrates the semantic segmentation of a multivariate time series [19]. The time series corresponds to the "chicken dance", a sequence of 4 different moves, which is repeated twice, for a total of 8 equal-length segments. While PaTSS correctly identifies the 4 different moves, the segmentation does not perfectly match the 8 (discrete) ground truth segments. For example, after semantic segment C, segment D briefly reoccurs. However, we argue that immediately after segment C, the measurements more closely relate to segment D than to segment A, possibly caused by a similar movement during the gradual transition.

The time series in Figure 9 measures the arterial

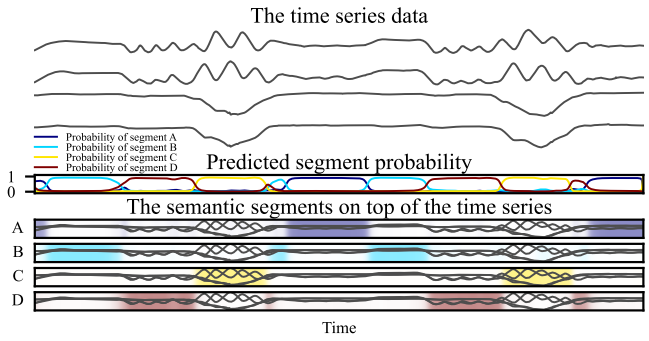


Figure 8: Semantic segmentation of a multivariate time series [19].

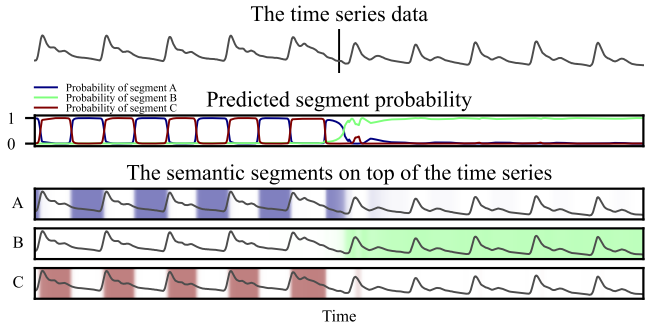


Figure 9: Semantic segmentation of the `TiltABP` time series [11], zoomed in on the unique change point. If we provide PaTSS with the ground truth number of segments (2), it correctly identifies the periods before and after the rotation.

blood pressure of a subject lying on a tilt table with foot support. Subsequently, the tilt table is rotated to the stand-up position, which results in a changing blood pressure. This time series was used by Ermshaus et al. [7] to illustrate a limitation of current time series segmentation methods: gradual state transitions. Figure 9 zooms in at the time the table was rotated (indicated by a vertical bar in the time series). PaTSS automatically identifies three segments: two alternating segments before the rotation, and one after. Around the rotation time, the likelihood of segment A starts decreasing due to the gradually increasing likelihood of segment B. Even more, immediately after the rotation point, both segments A and C have a slightly increased likelihood, conform the alternating pattern.

6 Conclusion

This paper proposed a novel variation of the time series semantic segmentation task to capture gradual transitions. Additionally, we presented PaTSS, a novel time series semantic segmentation algorithm that solves this variant by learning a probability distribution. It leverages frequent sequential pattern mining because of its high efficiency and interpretability. Quantitative experiments showed that PaTSS is highly effective when learning gradual transitions, unlike state-of-the-art solutions [7, 11, 22] that focus on discrete transitions. Qualitative experiments on real-world use cases demonstrated PaTSS' ability to uncover interesting segments. In future work, the objective is to leverage PaTSS for downstream time series analysis tasks, including anomaly detection and forecasting, in which semantic segmentation can be employed to account for contextual information. Further, the aim is to adapt PaTSS to a streaming setting with concept drift.

Acknowledgements

This research is supported by Flanders Innovation & Entrepreneurship (VLAIO) through the AI-ICON project CONSCIOUS (HBC.2020.2795), the Flemish government under the Flanders AI Research Program, and by Internal Funds KU Leuven (STG/21/057).

References

[1] Ryan Prescott Adams and David JC MacKay. "Bayesian online changepoint detection". In: *arXiv preprint arXiv:0710.3742* (2007). DOI: 10.48550/arXiv.0710.3742.

[2] Charu C Aggarwal et al. *Data mining: the textbook*. Vol. 1. Springer, 2015. DOI: 10.1007/978-3-319-07821-2.

[3] Charu C Aggarwal and Jiawei Han. *Frequent Pattern Mining*. Springer, 2014. DOI: 10.1007/978-3-319-07821-2.

[4] Stuart JH Biddle et al. "Physical activity and sedentary behaviours in youth: issues and controversies". In: *The journal of the Royal Society for the Promotion of Health* 124.1 (2004), pp. 29–33. DOI: 10.1177/146642400312400110.

[5] Faicel Chamroukhi et al. "Joint segmentation of multivariate time series with hidden process regression for human activity recognition". In: *Neurocomputing* 120 (2013), pp. 633–644. DOI: 10.1016/j.neucom.2013.04.003.

[6] Janez Demšar. "Statistical comparisons of classifiers over multiple data sets". In: *The Journal of Machine learning research* 7 (2006), pp. 1–30. DOI: 10.5555/1248547.1248548.

[7] Arik Ermshaus, Patrick Schäfer, and Ulf Leser. "ClaSP: parameter-free time series segmentation". In: *Data Mining and Knowledge Discovery* (2023), pp. 1–39. DOI: 10.1007/s10618-023-00923-x.

[8] Len Feremans, Boris Cule, and Bart Goethals. "Efficient pattern-based anomaly detection in a network of multivariate devices". In: *arXiv preprint arXiv:488.1590* (2023). DOI: 10.48550/arXiv.2305.05538.

[9] Len Feremans, Boris Cule, and Bart Goethals. "PETSC: pattern-based embedding for time series classification". In: *Data Mining and Knowledge Discovery* 36.3 (2022), pp. 1015–1061. DOI: 10.1007/s10618-022-00822-7.

[10] Len Feremans et al. "Pattern-based anomaly detection in mixed-type time series". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*. Springer. 2020, pp. 240–256. DOI: 10.1007/978-3-030-46150-8_15.

[11] Shaghayegh Gharghabi et al. "Domain agnostic online semantic segmentation for multi-dimensional time series". In: *Data mining and knowledge discovery* 33 (2019), pp. 96–130. DOI: 10.1007/s10618-018-0589-3.

[12] Aristides Gionis and Heikki Mannila. "Finding recurrent sources in sequences". In: *Proceedings of the seventh annual international conference on Research in computational molecular biology*. 2003, pp. 123–130. DOI: 10.1145/640075.640091.

[13] David G Kleinbaum, Mitchel Klein, and Erica Rihl Pryor. *Logistic regression: a self-learning text*. Vol. 94. Springer, 2002. DOI: 10.1007/978-1-4757-4108-7.

[14] Heiner Lasi et al. "Industry 4.0". In: *Business & information systems engineering* 6 (2014), pp. 239–242. DOI: 10.1007/s12599-014-0334-4.

[15] Sean M. Law. "STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining". In: *The Journal of Open Source Software* 4.39 (2019), p. 1504. DOI: 10.21105/joss.01504.

[16] Thach Le Nguyen et al. "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations". In: *Data mining and knowledge discovery* 33 (2019), pp. 1183–1222. DOI: 10.1007/s10618-019-00633-3.

[17] Jessica Lin et al. "A symbolic representation of time series, with implications for streaming algorithms". In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. 2003, pp. 2–11. DOI: 10.1145/882082.882086.

[18] Jonathan Feng-Shun Lin, Michelle Karg, and Dana Kulić. "Movement Primitive Segmentation for Human Motion Modeling: A Framework for Analysis". In: *IEEE Transactions on Human-Machine Systems* 46.3 (2016), pp. 325–339. DOI: 10.1109/THMS.2015.2493536.

[19] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. "Autoplait: Automatic mining of co-evolving time sequences". In: *Proceedings of the 2014 ACM SIGMOD international conference on management of data*. 2014, pp. 193–204. DOI: 10.1145/2588555.2588556.

[20] Thach Le Nguyen and Georgiana Ifrim. "Fast time series classification with random symbolic subsequences". In: *International Workshop on Advanced Analytics and Learning on Temporal Data*. Springer. 2022, pp. 50–65. DOI: 10.1007/978-3-031-24378-3_4.

[21] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65. DOI: 10.1016/0377-0427(87)90125-7.

[22] Patrick Schäfer, Arik Ermshaus, and Ulf Leser. "Clasp-time series segmentation". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 1578–1587. DOI: 10.1145/3459637.3482240.

[23] Andrew Jhon Scott and M Knott. "A cluster analysis method for grouping means in the analysis of variance". In: *Biometrics* (1974), pp. 507–512. DOI: 10.2307/2529204.

[24] Mark S Tremblay et al. "Sedentary behavior research network (SBRN)–terminology consensus project process and outcome". In: *International journal of behavioral nutrition and physical activity* 14 (2017), pp. 1–17. DOI: 10.1186/s12966-017-0525-8.

[25] Charles Truong, Laurent Oudre, and Nicolas Vayatis. "Selective review of offline change point detection methods". In: *Signal Processing* 167 (2020), p. 107299. DOI: 10.1016/j.sigpro.2019.107299.

[26] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00527.x.