

Event Detection via Probability Density Function Regression

Clark Peng¹ and Tolga Dinger²

¹*University of California, Los Angeles, CA, USA*

²*Colgate University, Information Technology Services, Hamilton, NY, USA*

Abstract

In the domain of time series analysis, particularly in event detection tasks, current methodologies predominantly rely on segmentation-based approaches, which predict the class label for each individual timesteps and use the changepoints of these labels to detect events. However, these approaches may not effectively detect the precise onset and offset of events within the data and suffer from class imbalance problems. This study introduces a generalized regression-based approach to reframe the time-interval-defined event detection problem. Inspired by heatmap regression techniques from computer vision, our approach aims to predict probability densities at event locations rather than class labels across the entire time series. The primary aim of this approach is to improve the accuracy of event detection methods, particularly for long-duration events where identifying the onset and offset is more critical than classifying individual event states. We demonstrate that regression-based approaches outperform segmentation-based methods across various state-of-the-art baseline networks and datasets, offering a more effective solution for specific event detection tasks.

1 Introduction

1.1 Background and Problem Statement

In the past decade, neural networks have become increasingly popular for addressing multivariate time series problems, including both event detection (ED) and change-point detection (CPD). Event detection refers to identifying significant occurrences or changes that last a meaningful duration of time within a series, while change-point detection involves identifying singular points where the statistical properties of the series change suddenly. For the purposes of this study, we refer to the time-interval-based event detection task as event detection and the opposite task as change-point detection, in which events last a negligible or no duration of time. Applications of event detection are numerous, such as sleep staging, sleep detection and seizure detection, as well as bow-shock event detection [4, 14, 15]. The model architectures most recently employed for these time series problems include sequence-to-sequence (seq2seq) models based on recurrent neural networks such as gated recurrent units (GRUs), long-short-term memory units (LSTMs), and/or one-dimensional convolutional neural networks (1D CNNs), often featuring a UNet-like (hourglass) topology [14, 15, 22].

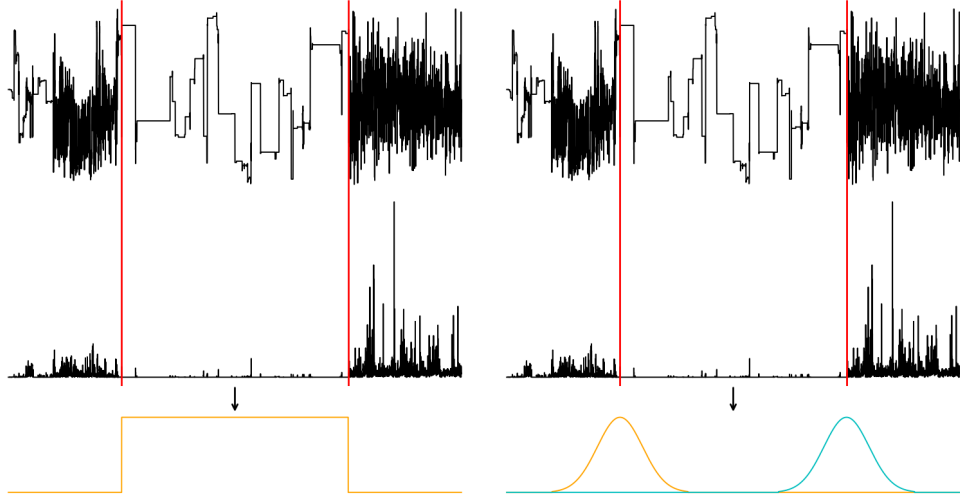
Traditionally, event detection strategies have predominantly employed segmentation-based methodologies. These approaches focus on predicting the class of each individual time step rather than the precise onset and offset of specific events [16, 18]. However, segmentation may not always be the most effective approach, particularly when the goal is to detect specific event points within the time series.

This study introduces a generalized approach to reframe the event detection problem, inspired by heatmap regression solutions for two-dimensional keypoint detection in computer vision [12, 24]. We redefine the event detection problem as a regression problem, utilizing deep learning models to predict probability density functions (pdfs) at each event’s onset and offset locations rather than class labels across the entire time series.

As it is a regression method it is also robust to class imbalance problems exhibited by segmentation techniques [2, 4]. Additionally, like the gaussian regression targets used in two-dimensional keypoint detection, our approach addresses labeling ambiguities by approximating a probability distribution instead of relying on hard ground-truth labels [12, 24]. This is particularly important in time series data, where high-resolution sampling often contrasts with coarser event annotations, or when the labels contain human error. This solution for event detection can also be generalized to supervised change-point detection, which is an easier task than event detection, since it does not consider event durations.

We have also applied this regression method in the Child Mind Institute’s (CMI) sleep detection competition on Kaggle in September 2023, utilizing it to detect sleep onset and wake events in children using wrist-worn

Figure 1: Example of Segmentation vs PDF Regression Target



accelerometer data. This approach was subsequently adopted by nearly all of the top-scoring solutions in the competition [5].

1.2 Previous Work

Recent research in event detection has introduced regression-based methodologies [4]. These methods use a function defined by the Intersection over Union (IoU) metric applied to overlapping partitions at event locations to set their regression targets. Although designed to optimize the detection of event onsets and offsets, these approaches often generate fixed target distributions focused on predicting event midpoints, assuming uniform event durations. This assumption complicates the accurate identification of event boundaries, as events typically vary in duration. Furthermore, the reliance on overlapping partitions results in redundant memory usage and restricts the model’s ability to leverage global context, limiting it to local features around specific timesteps.

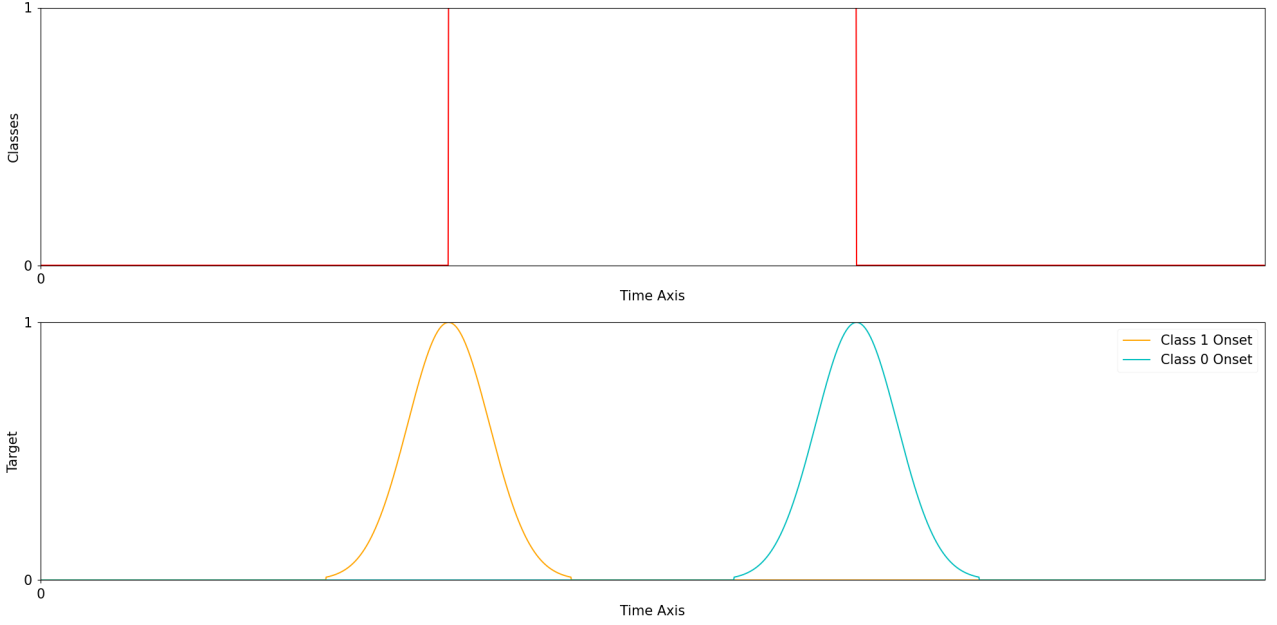
Probability density methods have been developed for sleep-detection problems by fitting Gaussian probability densities over event locations [17]. These methods involve an indirect process: first, decision trees predict the distance from each step to the event location, and then another tree predicts the event prediction’s confidence. These decision trees are only trained on local partitions of data around each timestep. These predictions are aggregated by summing Gaussians across all timesteps, fitting a probability density to each event. This approach aids in event detection by identifying the centers of these densities via peak-detection. However, the requirement for extensive data processing and tabular modeling methods limits their applicability to modern time series frameworks and models.

In the following, we expand on these previous works by presenting our approach to applying a customizable probability density function (pdf) objective to two ED datasets, removing the need for overlapping partitions, and showing how this method can easily be applied to state-of-the-art seq2seq model architectures, which can incorporate global context, unlike previous work. In Section 3, we explain our approach, apply it to various state of the art deep learning models, and evaluate them on CMI’s sleep detection dataset. In addition, we compare them against models trained on segmentation objectives to assess which method performs the best. Section 4 summarizes our main findings, and 5 discusses limitations and applications before we conclude in Section 6.

2 Regression Methodology

Our method involves convolving a pdf over an event series to produce a series with peaks at each event location. Since we are aiming to predict both the onset and offset of events, this necessitates two output target series. Our approach eliminates the need for partitioning by directly convolving a pdf rather than calculating an IoU over partitions.

Figure 2: Example regarding a convolved gaussian pdf over a binary event detection problem. The bottom graph shows two time series, corresponding to the onset and offset of the event respectively.



After a deep learning model predicts these transformed targets, a series of simple post-processing steps are done, using simple peak finding and smoothing functions like *find_peaks* and *gaussian_filter1d* in SciPy [23] to find event locations and rank their relative prominences.

Time series segmentation with deep learning typically involves employing an encoder-decoder network that outputs a N-dimensional series with the same time dimension as the input series, with N corresponding to the class probabilities for each time step [14,22]. By simply changing the output layers of these segmentation models to have 2 dimensions and use a linear activation instead of a softmax activation on the output layer, we are able to transform these models designed for segmentation into models for pdf regression.

3 Experiments and Evaluation

3.1 Data Description

To test event detection algorithms, we use two real publicly available real-world datasets in order to measure how our technique performs in different conditions. From the human activity recognition domain, Child Mind Institute’s (CMI) Sleep Detection Dataset contains 2-dimensional accelerometer signals collected from a hand-worn wristband at a rate of 0.2 Hz from 250+ subjects [1]. The timestep when the subject begins sleeping and when the subject wakes up are considered event onsets and offsets.

The other event-detection dataset is from the brain activity domain. The CHB-MIT Scalp EEG Database contains 22-dimensional electrode signals collected at a rate of 256 Hz from only 22 subjects [21]. The signals used the International 10-20 system of EEG electrode positions and nomenclature [10]. Features were selected to utilize a bipolar montage for EEG recordings, where the voltage of one electrode is subtracted from another (e.g., FP1-F7). Only the subset of signals where seizures were detected were used.

Including these datasets with varying sampling rates, signal lengths, and dimensionality provides a robust foundation for testing our regression framework. A more comprehensive description of the datasets are available in Appendix B.

3.2 Evaluation

We will be using the Event Detection AP (EDAP) metric from the original CMI Sleep Detection competition. Event Detection AP is a soft ED metric, meaning that it can capture the degree to which a detection represents a particular event, incorporating the concept of temporal tolerance for inaccuracy in the evaluation [20].

This is different from hard metrics that only reward exact matches for specific thresholds [20]. As such, it is similar to IoU-threshold average precision metrics commonly used in object detection, but with IoU thresholds

replaced by time tolerance values [18]. The advantage of this metric is that it naturally matches with the concave probability density functions that we propose.

The metric is evaluated using the following procedure:

- **Assignment:** Predicted events are matched with ground-truth events.
- **Scoring:** Each group of predictions is scored against its corresponding group of ground-truth events via thresholded average precision (AP).
- **Reduction:** The thresholded AP scores are averaged to produce a single overall score.
- The metric thresholds for the Sleep Detection Dataset, as provided by CMI, include the following values, in minutes:

[1, 3, 5, 7.5, 10, 12.5, 15, 20, 25, 30]

- The metric thresholds for the Seizure Detection Dataset were inferred based on previous findings, which state that 50% of seizures were detected within 3 seconds, 71% within 5 seconds, and 91% within 10 seconds [21]. With this in mind, we arbitrarily define the thresholds—in seconds—to be:

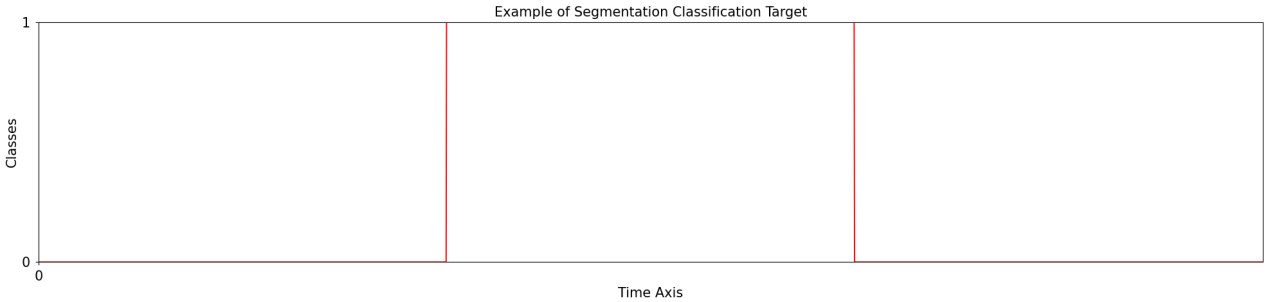
[1, 2, 5, 10, 20, 60]

3.3 Objective Distributions

Each model will be trained on the segmentation objective and 3 regression objectives created by different pdfs:

3.3.1 Segmentation

We use the traditional way to train seq2seq models for ED problems using cross entropy loss with the binary segmentation task.

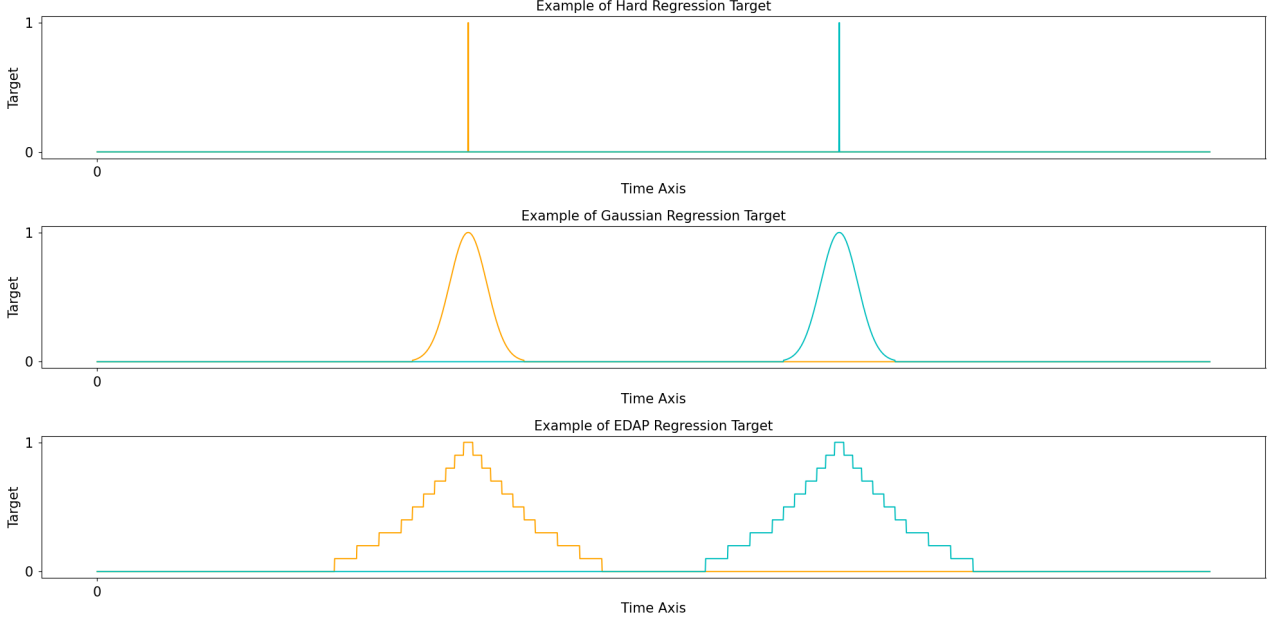


3.3.2 Regression

Three different probability density functions (pdfs) were utilized. These objectives serve to tailor the model’s performance to suit various event detection tasks. By adjusting the spread of the pdfs, we can emphasize either overall event detection accuracy or precise event localization, thereby catering to different requirements in event detection problems, as indicated by the different EDAP thresholds.

- **Hard regression**, similar to the binary classification problem with a regression objective. Defined as 1 at the event onsets and offsets and 0 everywhere else.
- **Gaussian regression**, inspired by keypoint detection methods [12,24]. Guided by metric thresholds, we arbitrarily set $\sigma = 50$ (4 minutes) on the Sleep Dataset and $\sigma = 256$ (1 second) on the Seizure Dataset.
- **EDAP Regression**, tailored to optimize the Event Detection AP metric specific to each dataset. This metric employs thresholds at intervals of 1, 3, 5, 7.5, 10, 12.5, 15, 20, 25, and 30 minutes for the Sleep Dataset and intervals of 1, 2, 5, 10, 20, and 60 seconds for the Seizure Dataset. As the event approaches the designated time thresholds, the regression target aligns with the metric, increasing accordingly.

We need to normalize each of these pdfs before using them to train the model. This ensures that the model converges consistently across various target distributions by aligning the initial expected loss from the model for each pdf. As such, we normalize the target by dividing the target sequence with the following scaling factor γ :



$$\gamma = \sqrt{\frac{1}{d} \sum_{i=0}^w y_i^2} \approx \sqrt{\frac{1}{d} \int_{-w/2}^{w/2} f(x)^2 dx} \quad (1)$$

In (2), w denotes the width of the regression target, and d is a constant that denotes the length of day in timesteps, in which we expect an event every d steps.

Thus, assuming the model initializes with normally distributed values, the model should initially output an average around zero for all steps in the time series. With this assumption in (2), applying the MSE loss function \mathcal{L} to each initial prediction yields a loss of approximately 1 using (3):

$$\hat{y} \sim \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \end{bmatrix} \quad (2)$$

$$\mathcal{L}(\hat{y}, y) = \frac{1}{d} \sum_{i=1}^d (y_i/\gamma - \hat{y}_i)^2 \approx 1 \quad (3)$$

Transforming the target in this way ensures that all regression targets scale up to generate similar loss values, thereby ensuring training stability across different pdfs.

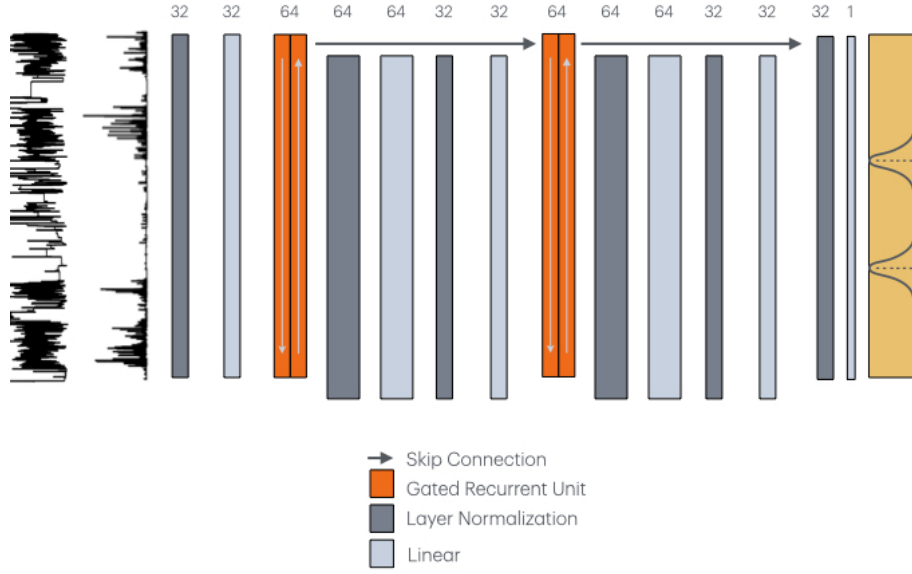
3.4 Preprocessing

Before feeding the training data into the models for deep learning, we preprocess it to reduce dimensionality and memory usage. This involves downsampling the time series by an order of magnitude, wherein aggregated features such as mean, standard deviation, maximum, and minimum are computed along each time segment for continuous features, which are then appended to the feature list. For categorical features, we take the last value in the downsampled time series.

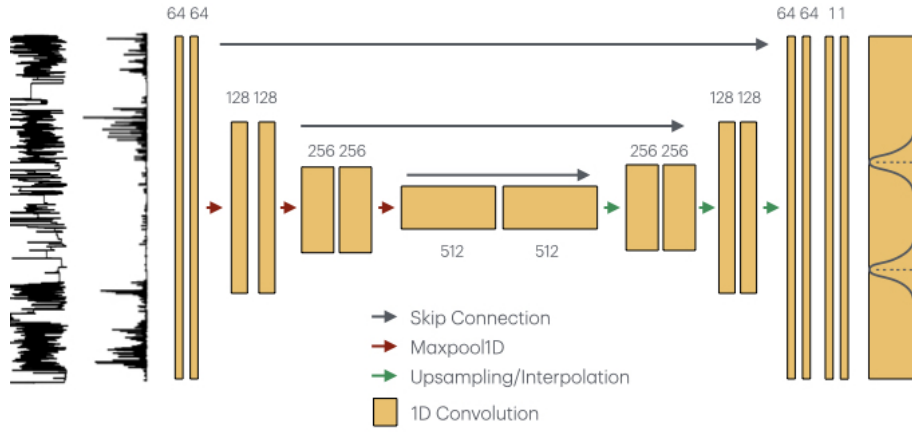
3.5 Modeling and Training Parameters

We employed three custom-designed models alongside one state-of-the-art architecture. Our custom models include a bidirectional GRU model and a simplified fully convolutional encoder-decoder network, inspired by the U-Net architecture seen in U-Time [14, 19]. Additionally, we adapted the U-Time model by incorporating an attention layer at the bottleneck to enhance feature representation and capture long-range dependencies [14].

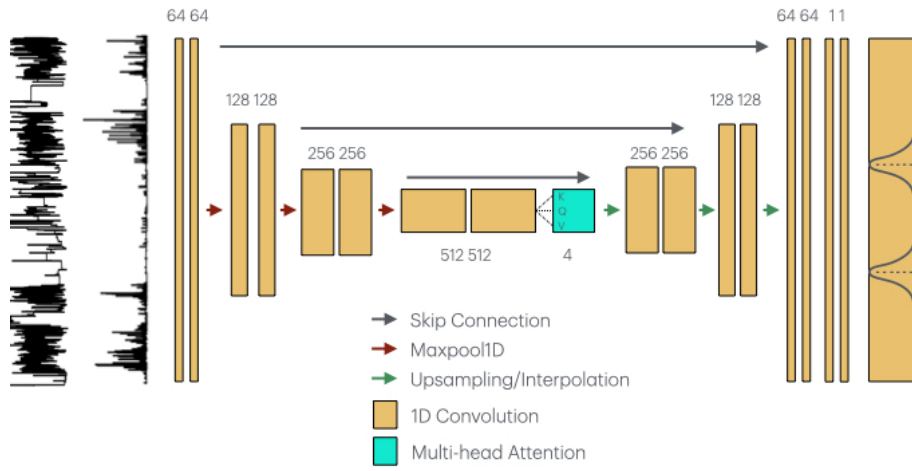
For comparison, we incorporated one state-of-the-art model, PrecTime, which combines convolutional and recurrent layers to leverage both spatial and temporal features for precise event detection [6].



(a) Bidirectional GRU



(b) UNet-1D



(c) UNet-1D with Attention

Figure 3: Custom-designed Model Architectures, a Bidirectional RNN model, a 1D UNet Model and a 1D UNet model with an Attention layer [14]

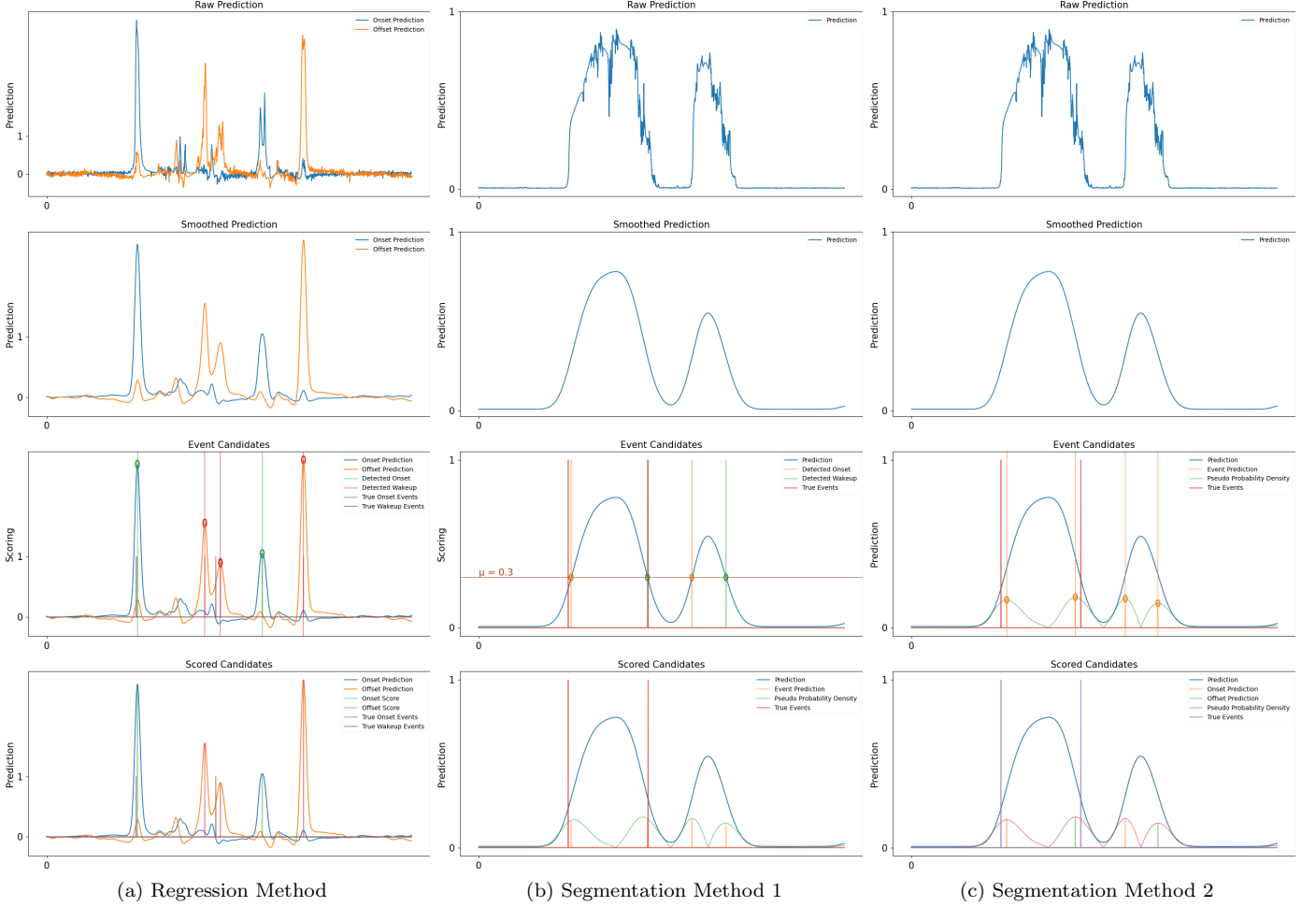


Figure 4: All post-processing procedures step-by-step. Including both forms of segmentation processing and the regression post-processing results

These diverse model architectures will help us demonstrate that our simplified regression method can be applied to various model architectures.

3.6 Postprocessing

Figure 4 illustrates examples of various post-processing pipelines applied to raw regression and segmentation model outputs.

3.6.1 Regression Postprocessing

Our post processing pipeline for regression follows three simple steps with adjustable hyperparameters applied on a raw regression prediction:

- **Gaussian-smoothing step:** Smooth all predictions along the time series with a gaussian filter, with hyperparameter σ denoting the standard deviation of the gaussian kernel.
- **Finding Events:** Apply a simple peak-finding function, for example, *scipy.find_peaks*, to find candidate peaks in the predictions. This function is similar to the peak detection algorithm detailed in [17].
- **Scoring Each Event:** The EDAP metric requires each prediction to have a score assigned to it in order to rank the predictions and determine the optimal thresholds. Thus, we assign the peak probability of the probability density of an event to that event.

3.6.2 Segmentation Postprocessing

In order to compare our regression methods against segmentation models, we also use a separate post processing step for segmentation models to determine events. This involves setting a threshold that denotes a state change

Algorithm 1 Regression Post-Processing Procedure

Inputs time series $\{\hat{Y}_{on}(t)\}_{t=0}^T$ and $\{\hat{Y}_{off}(t)\}_{t=0}^T$, threshold μ , smooth coef σ
 $T = \lfloor L/D \rfloor$ - sequence length

1: **procedure** FINDEVENTS(\hat{Y}, μ, σ)
2: Initialize S an empty list ▷ onsets
3: Initialize E an empty list ▷ offsets
4: Initialize $\{\hat{Y}'_{on}(t)\}_{t=0}^T \leftarrow \{\text{gaussian_smooth}(\hat{Y}_{on}(t), \sigma)\}_{t=0}^T$
5: Initialize $\{\hat{Y}'_{off}(t)\}_{t=0}^T \leftarrow \{\text{gaussian_smooth}(\hat{Y}_{off}(t), \sigma)\}_{t=0}^T$ ▷ perform gaussian smoothing
6: $S' \leftarrow \text{peak_detect}(\{\hat{Y}'_{on}(t)\}_{t=0}^T)$
7: $E' \leftarrow \text{peak_detect}(\{\hat{Y}'_{off}(t)\}_{t=0}^T)$
8: **for** $t \leftarrow S'$ **do**
9: $S.\text{append}(t, \hat{Y}_{on}(t))$
10: **for** $t \leftarrow E'$ **do**
11: $E.\text{append}(t, \hat{Y}_{off}(t))$

Outputs S, E event locations with relative probability scores

and then finding time steps where the probability for the event exceeds the threshold. We locate events by locating where the probabilities for a class exceed the given threshold. However, probabilities can go up for more than one timestep when an event is detected. It is possible that for one single event, the probabilities go up and down many times, even though they are usually expected to increase continuously afterwards. In other words, multiple changes in the output of the model can represent one single event. As such, the following steps are used to reduce the likelihood of this happening:

- Gaussian-smoothing step: Smooth all probability predictions along the time series with a gaussian filter, with the hyperparameter σ denoting the standard deviation of the gaussian.
- Finding Events: Use a threshold μ to denote class thresholds. Events occur where the event class's predicted probabilities exceed μ .
- Scoring Each Event:

$$H(t) = \begin{cases} 0, & t \leq -\alpha, \\ -1/\alpha, & -\alpha \leq t \leq 0, \\ 1/\alpha, & 0 \leq t \leq \alpha, \\ 0, & \alpha \leq t \end{cases} \quad (4)$$

$$I(t) = (\hat{Y} * H)(t) \quad (5)$$

The EDAP metric requires each prediction to have a score assigned to it in order to rank the predictions and determine the optimal thresholds. This is addressed by convolving the probability predictions along the time dimension using the following piecewise sliding window function in Equation (4) to obtain a pseudo-probability function $I(t)$ mimicking to the probability function obtained using pdf regression, with α denoting the window size:

A pseudo probability density distribution is generated by the procedure depicted in equation 5. The absolute value of this distribution is maximized at points where class probabilities change suddenly and remain changed. Consequently, the probability density generated by this function allow us to score each candidate event, giving more weight to events with more dramatic changes in class probabilities.

The above segmentation post-processing procedure will be referred to as Method 1.

Alternatively, events can be directly detected performing a regression-based peak-finding procedure on the pseudo probability function generated by I . This peak-based event detection method will be referred to as Method 2. Both methods were tested and compared.

4 Results

4.1 Experimental Setup

4.1.1 Training Parameters

The models were implemented using the Pytorch v2.0.0 framework [3]. The training parameters are shown in Table 1 and 2. The network training was performed using the Adam optimizer and a cosine learning rate

Algorithm 2 Traditional Segmentation Post-Processing Procedure (Method 1)

Inputs time series $\{\hat{Y}(t)\}_{t=0}^T$, threshold μ , smooth coef σ
 $T = \lfloor L/D \rfloor$ - sequence length

- 1: **procedure** FINDEVENTS(\hat{Y}, μ, σ)
- 2: Initialize S an empty list ▷ onsets
- 3: Initialize E an empty list ▷ offsets
- 4: Initialize $\{\hat{Y}'(t)\}_{t=0}^T \leftarrow \{\text{gaussian_smooth}(\hat{Y}(t), \sigma)\}_{t=0}^T$ ▷ perform gaussian smoothing
- 5: Initialize $\{I(t)\}_{t=0}^T \leftarrow \{(\hat{Y}' * H)(t)\}_{t=0}^T$ ▷ get pseudo probability density
- 6: **for** $t \leftarrow 1 \dots T$ **do**
- 7: **if** $\hat{Y}'[t-1] < \mu$ **and** $\hat{Y}'[t] > \mu$ **then**
- 8: $S.\text{append}(t, |I[t]|)$
- 9: **else if** $\hat{Y}'[t-1] > \mu$ **and** $\hat{Y}'[t] < \mu$ **then**
- 10: $E.\text{append}(t, |I[t]|)$

Outputs S, E event locations with relative probability scores

Algorithm 3 Alternative Segmentation Post-Processing Procedure (Method 2)

Inputs time series $\{\hat{Y}(t)\}_{t=0}^T$, threshold μ , smooth coef σ
 $T = \lfloor L/D \rfloor$ - sequence length

- 1: **procedure** FINDEVENTS(\hat{Y}, μ, σ)
- 2: Initialize S an empty list ▷ onsets
- 3: Initialize E an empty list ▷ offsets
- 4: Initialize $\{\hat{Y}'(t)\}_{t=0}^T \leftarrow \{\text{gaussian_smooth}(\hat{Y}(t), \sigma)\}_{t=0}^T$ ▷ perform gaussian smoothing
- 5: Initialize $\{I(t)\}_{t=0}^T \leftarrow \{(\hat{Y}' * H)(t)\}_{t=0}^T$ ▷ get pseudo probability density
- 6: $S' \leftarrow \text{peak_detect}(\{\hat{I}'(t)\}_{t=0}^T)$
- 7: $E' \leftarrow \text{peak_detect}(\{-\hat{I}'(t)\}_{t=0}^T)$
- 8: **for** $t \leftarrow S'$ **do**
- 9: $S.\text{append}(t, |I[t]|)$
- 10: **for** $t \leftarrow E'$ **do**
- 11: $E.\text{append}(t, |I[t]|)$

Outputs S, E event locations with relative probability scores

Table 1: Sleep Detection Training Parameters

Parameter	Value
Day Length d	$24 \times 60 \times 12 = 17280$
Sequence Length L	$d \times 7 = 120960$
Min Event Interval α	$12 * 30 = 360$
Batch Size M	10
Down-sampling Factor D	10
Learning Rate	0.001
Gradient Clipping Norm λ	0.1

Table 2: Seizure Detection Training Parameters

Parameter	Value
"Day" Length d	$60 \times 60 \times 256 = 921600$
Sequence Length L	$d = 921600$
Min Event Interval α	25600
Batch Size M	8
Down-sampling Factor D	64
Learning Rate	0.001
Gradient Clipping Norm λ	0.1

scheduler with no restarts [9, 11].

We conduct 4-fold cross-validation on the CMI sleep detection dataset and the CHB-MIT seizure detection dataset. The models were all trained for 20 epochs, evaluating the network after every epoch, and the one that yielded the best overall score on the validation set was retained for evaluation. The outputs of the cross-validation folds were pooled and considered as a whole for computing the final performance of the model.

4.1.2 Evaluation Parameters

Our post-processing step for pdf regression, as shown in Section 3.6.1, involves Gaussian-smoothing using a smoothing parameter σ . Similarly, our process for segmentation uses the tunable smoothing parameter σ and an additional tunable event threshold μ .

$$\begin{aligned} \mu &\in \{0, \frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \dots, 1\} \\ \sigma &\in \{None, 1, 10, 100, 1000\} \end{aligned} \tag{6}$$

In our experiments, we compared our model's evaluation by deploying default hyperparameters $\mu = 0.5$ and no gaussian smoothing step. The optimal values of these hyperparameters are estimated using grid search with the parameter grid shown in Figure 6 to maximize the EDAP metric.

4.2 Experimental Results

Table 3: Performance obtained by pdf regression on the baseline models with default and tuned hyperparameters, compared with segmentation objectives on the CMI dataset. We mark our proposed methods in italics, the highest performing pdf for each model type in bold.

		Model	Model Objectives				
			Segmentation		Regression		
			Method 1	Method 2	Hard	Gaussian	EDAP
Hyperparameters	Tuned	GRU	0.477	0.508	0.621	0.610	0.572
		UNet	0.470	0.481	0.548	0.566	0.574
		PrecTime [6]	0.410	0.318	0.480	0.483	0.475
		UNet+Attention	0.309	0.324	0.521	0.552	0.542
	Default	GRU	0.302	0.347	0.586	0.589	0.569
		UNet	0.436	0.425	0.522	0.544	0.540
		PrecTime [6]	0.269	0.258	0.374	0.467	0.473
		UNet+Attention	0.263	0.248	0.494	0.530	0.499

Table 4: Performance obtained by pdf regression on the baseline models with default and tuned hyperparameters, compared with segmentation objectives on the CHB-MIT Scalp EEG Database. We mark our proposed methods in italics, the highest performing pdf for each model type in bold.

		Model	Model Objectives				
			Segmentation		Regression		
			Method 1	Method 2	Hard	Gaussian	Custom
Hyperparameters	Tuned	GRU	0.082	0.060	0.159	0.075	0.082
		UNet	0.221	0.123	0.099	0.192	0.214
		PrecTime [6]	0.077	0.061	0.134	0.138	0.093
		UNet+Attention	0.076	0.000	0.015	0.005	0.014
	Default	GRU	0.025	0.006	0.137	0.069	0.075
		UNet	0.109	0.018	0.095	0.188	0.208
		PrecTime [6]	0.037	0.008	0.116	0.124	0.092
		UNet+Attention	0.011	0.000	0.012	0.005	0.012

Table 5: Performance of the best regression and segmentation models on the CMI database with each threshold outlined by the EDAP metric. The highest-scores for each threshold are marked in bold.

Model Architecture	Objective		Tolerance Thresholds									
	Type	Method	12	36	60	90	120	150	180	240	300	360
RNN	Regression	Hard	0.04	0.29	0.52	0.66	0.72	0.75	0.77	0.80	0.81	0.83
RNN	Segmentation	Method 2	0.02	0.19	0.34	0.47	0.56	0.62	0.66	0.71	0.74	0.76

Table 6: Performance of the best regression and segmentation models on the CHB-MIT Scalp EEG Database with each threshold outlined by the EDAP metric. The highest-scores for each threshold are marked in bold.

Model Architecture	Objective		Tolerance Thresholds					
	Type	Method	256	512	1280	2560	5120	15360
UNet	Regression	Custom	0.02	0.05	0.13	0.19	0.31	0.58
UNet	Segmentation	Method 1	0.01	0.02	0.16	0.28	0.39	0.47

Regression vs Segmentation Models trained on pdf regression consistently achieved comparable or improved performance than the models trained on segmentation tasks (Table 3 and 4). The top-performing regression models also match or exceed the precision of segmentation models across all time tolerance thresholds, demonstrating that pdf regression is superior to the segmentation approaches on these two datasets (Table 5 and 6)

Impact of peak-finding approaches There is a significant difference between framing the event detection challenge as a peak detection task versus a segmentation task. This is evident in the superior performance of regression-based methods compared to segmentation methods on the CMI dataset. The best regression approach achieved a tuned EDAP of 0.621, while the best segmentation method achieved 0.508 (Table 3). Even within segmentation models, employing peak-finding techniques instead of solely identifying class change locations can be beneficial. Segmentation Method 2, which uses a sliding window function to identify peaks from class probabilities, outperformed Method 1 in 75% of optimized models on the CMI Dataset, achieving a maximum EDAP of 0.508 compared to 0.477 for Method 1. However, Method 1 outperformed Method 2 on the CHB-MIT

Dataset, achieving a maximum EDAP of 0.221 compared to 0.123 (Table 4). This suggests that peak detection can enhance or degrade performance depending on the model’s underlying confidence. It is clear, though, that peak-finding benefits regression-based models.

5 Discussion

We have demonstrated that the proposed pdf regression framework outperforms segmentation-based approaches in the given event detection problems. Additionally, our method can be applied to any seq2seq segmentation neural-network model by changing only the output layer, meaning that existing segmentation models can easily be converted to support our objectives.

Limitations We may find more effective methods for extracting events from segmentation probabilities, potentially altering these experiments’ results. This limitation could be tackled in future research. Moreover, our selection of datasets and evaluation metrics is limited, warranting further exploration to validate the suitability of pdf regression for various event detection problems. This could involve exploring different evaluation metrics or datasets.

Cross-Method Ensembling Our method supports the straightforward ensembling of predictions across multiple probability density functions and model architectures by blending output probability distributions. Additionally, the segmentation post-processing method Method 1 converts segmentation outputs into pseudo probability distributions that can be blended with regression outputs. This ensemble approach could combine the strengths of both segmentation and regression methods, leading to more precise event detection.

Online detection Our method could potentially apply to real-time event detection tasks using live data streams [2]. This could involve using a unidirectional GRU for regression on our target variable and utilizing its predictions to detect events in real time.

More PDFs Segmentation-based event-detection tasks often necessitate complex loss functions to optimize different task-specific objectives, such as early event detection, in which events should be detected as soon as they happen [16]. Our method could potentially simplify this process significantly. By adjusting a pdf function to be asymmetrical and left-skewed, prioritizing pre-event detection over post-event becomes feasible. This approach could potentially yield comparable results to customized loss functions, without the need for the fine-tuning and design efforts needed to design custom loss functions.

Adaptive PDFs: In diffusion models, noise levels start high and are gradually reduced throughout the training process, which allows the model to generate higher-quality samples as it converges [8]. This gradual reduction of noise helps the model progressively refine its outputs. Similarly, in event detection, a comparable approach can be used where the variance of the probability distributions is systematically decreased over time. This reduction in variance enables the model to focus on finer temporal details as it converges. This method was effectively utilized in the CMI Sleep Detection Competition, where decaying target PDFs with reduced variability were employed to enhance the model’s temporal accuracy as training progressed [5]. Future work could investigate how scheduling affects model performance.

Problem Types We’ve only tested our method on a subset of event detection involving long term time-interval based events, but it’s evident that our method should also be applicable to change-point detection. For CPD, this would involve treating each event point as only an onset and ignoring offsets, requiring only one output dimension instead of two, and performing pdf regression on this objective.

For preliminary experimental data on the point-based event detection case, the reader is referred to the additional materials presented in Appendix A.

6 Conclusion

This study presents a generalized regression-based approach to event detection, providing an alternative to traditional segmentation-based methods and existing regression-based approaches. We introduce techniques to normalize regression targets across unique pdfs and convert both regression and segmentation predictions into precise event locations. Specifically, we employ peak detection for regression predictions and various strategies for segmentation outputs, including peak identification and traditional event-point determination using thresholds.

This approach to event detection surpasses traditional segmentation methods in accuracy and effectiveness. By leveraging regression techniques, we provide a versatile and powerful tool for identifying events in time series data. Unlike existing regression approaches, our method supports the integration of state-of-the-art neural network models, enhancing its robustness and applicability.

While this study primarily focuses on event detection tasks such as sleep detection, our method shows promise for other applications, including change-point detection tasks like fraud event detection, as evidenced by preliminary experiments in appendix A. This methodology not only enhances the precision of event detection but also opens new avenues for the application of deep learning models to complex time series problems, with potential for broad applicability across many fields.

References

- [1] Lindsay M. Alexander, Jasmine Escalera, Lei Ai, Charissa Andreotti, Karina Febre, Alexander Mangone, Natan Vega-Potler, Nicolas Langer, Alexis Alexander, Meagan Kovacs, Shannon Litke, Bridget O’Hagan, Jennifer Andersen, Batya Bronstein, Anastasia Bui, Marijayne Bushey, Henry Butler, Victoria Castagna, Nicolas Camacho, Elisha Chan, Danielle Citera, Jon Clucas, Samantha Cohen, Sarah Dufek, Megan Eaves, Brian Fradera, Judith Gardner, Natalie Grant-Villegas, Gabriella Green, Camille Gregory, Emily Hart, Shana Harris, Megan Horton, Danielle Kahn, Katherine Kabotyanski, Bernard Karmel, Simon P. Kelly, Kayla Kleinman, Bonhwang Koo, Eliza Kramer, Elizabeth Lennon, Catherine Lord, Ginny Mantello, Amy Margolis, Kathleen R. Merikangas, Judith Milham, Giuseppe Minniti, Rebecca Neuhaus, Alexandra Levine, Yael Osman, Lucas C. Parra, Ken R. Pugh, Amy Racanello, Anita Restrepo, Tian Saltzman, Batya Septimus, Russell Tobe, Rachel Waltz, Anna Williams, Anna Yeo, Francisco X. Castellanos, Arno Klein, Tomas Paus, Bennett L. Leventhal, R. Cameron Craddock, Harold S. Koplewicz, and Michael P. Milham. An open resource for transdiagnostic research in pediatric mental health and learning disorders. *Scientific Data*, 4(1):170181, Dec 2017.
- [2] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, May 2017.
- [3] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhersch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS ’24)*. ACM, April 2024.
- [4] Menouar Azib, Benjamin Renard, Philippe Garnier, Vincent Génot, and Nicolas André. Event detection in time series: Universal deep learning approach, 2023.
- [5] Nathalia Esper, Maggie Demkin, Ryan Hoolbrok, Yuki Kotani, Larissa Hunt, Andrew Leroux, Vincent van Hees, Vadim Zipunnikov, Kathleen Merikangas, Michael Milham, Alexandre Franco, and Gregory. Kiar. Child mind institute - detect sleep states, 2023.
- [6] Stefan Gaugel and Manfred Reichert. Preptime: A deep learning architecture for precise time series segmentation in industrial manufacturing operations. *Engineering Applications of Artificial Intelligence*, 122:106078, 2023.
- [7] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23), June 2000.
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.
- [9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [10] G H Klem, H O Lüders, H H Jasper, and C Elger. The ten-twenty electrode system of the international federation. the international federation of clinical neurophysiology. *Electroencephalogr. Clin. Neurophysiol. Suppl.*, 52:3–6, 1999.
- [11] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

- [12] Zhengxiong Luo, Zhicheng Wang, Yan Huang, Liang Wang, Tieniu Tan, and Erjin Zhou. Rethinking the heatmap regression for bottom-up human pose estimation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13259–13268, 2021.
- [13] Jairo H. Migueles, Alex V. Rowlands, Florian Huber, Séverine Sabia, and Vincent T. van Hees. Ggir: A research community-driven open source r package for generating physical activity and sleep outcomes from multi-day raw accelerometer data. *Journal for the Measurement of Physical Behaviour*, 2(3):188 – 196, 2019.
- [14] Mathias Perslev, Michael Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4415–4426. Curran Associates, Inc., 2019.
- [15] Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y. Chén, and Maarten De Vos. Seqsleepnet: End-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):400–410, 2019.
- [16] Huy Phan, Philipp Koch, Ian McLoughlin, and Alfred Mertins. Enabling early audio event detection with neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 141–145, 2018.
- [17] Huy Phan, Marco Maaß, Radoslaw Mazur, and Alfred Mertins. Random regression forests for acoustic event detection and classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):20–31, 2015.
- [18] Ashwin Ram, Sundar Sripada V. S., Shuvam Keshari, and Zizhe Jiang. Annotating sleep states in children from wrist-worn accelerometer data using machine learning, 2023.
- [19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
- [20] Rebecca Salles, Janio Lima, Rafaelli Coutinho, Esther Pacitti, Florent Masegla, Reza Akbarinia, Chao Chen, Jonathan Garibaldi, Fabio Porto, and Eduardo Ogasawara. Softed: Metrics for soft evaluation of time series event detection, 2023.
- [21] Ali Shoeb and John Guttag. Application of machine learning to epileptic seizure detection. *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 975–982, 08 2010.
- [22] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):1998–2008, 2017.
- [23] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, Yoshiki Vázquez-Baeza, and SciPy 1.0 Contributors. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, Mar 2020.
- [24] Baosheng Yu and Dacheng Tao. Heatmap regression via randomized rounding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8276–8289, 2022.

Table 7: Bowshock Experimental Parameters

Parameter	Value
Day Length d	$24 \times 60 \times 15 = 21600$
Sequence Length L	$d = 21600$
Min Event Interval α	1
Batch Size M	10
Down-sampling Factor D	15
Learning Rate	0.001
Gradient Clipping Norm λ	0.1

Table 9: Fraud Experimental Parameters

Parameter	Value
Day Length d	$1 \times 60 \times 60 = 3600$
Sequence Length L	900
Min Event Interval α	1
Batch Size M	5
Down-sampling Factor D	1
Learning Rate	0.001
Gradient Clipping Norm λ	0.1

Table 8: Bowshock Results

Regression Method	Precision	F1
Universal Deep Learning Approach [4]	0.95	0.95
Ours	0.992	0.951

Table 10: Fraud Results

Regression Method	Precision	F1
Universal Deep Learning Approach [4]	0.98	0.85
Ours	0.783	0.809

A Additional Experiments

We additionally evaluated our approach on two CPD datasets provided by [4]: the bow shock event detection dataset and fraud-detection dataset. Table 7 and 9 lists the experimental training parameters for each dataset respectively, while Table 8 and Table 10 presents the validation results after training with 5 fold cross validation.

It is worth mentioning that the validation scheme used in our experiments are not the same used in [4]. Thus, results may not be entirely comparable.

B Detailed Data Descriptions

B.1 CMI Sleep Detection Dataset

We conducted experiments on Child Mind Institute’s (CMI) sleep detection dataset. The data is provided by CMI along with the Healthy Brain Network, a landmark mental health study based in New York City [1].

The data contains over 250 publicly available recordings of wrist-worn accelerometer data, collected using wearable accelerometers on children spanning multiple days, where each time step corresponds to the accelerometer’s reading over 5 seconds. The two types of events are onset and wakeup, which correspond to the onset of sleep and the onset of awakesness, respectively.

This dataset additionally contains the following specifications about sleep periods:

- Each sleep period must be at least 30 minutes long.
- A single sleep period can be interrupted by activity, provided that these interruptions do not exceed 30 consecutive minutes.
- Sleep windows are only detected if the watch is worn for the entire duration of the monitoring period (details provided below).
- Only the longest sleep window during the night is recorded.
- If no valid sleep window is found, no sleep onset or wake-up event is recorded for that night.
- Sleep events do not need to cross the daytime, and there is no strict limit on the number of sleep events per day. However, only one sleep window should be assigned per night.
- The number of nights recorded for a series is approximately equal to the number of 24-hour periods in that series.

There are two input variables that are used to train the models:

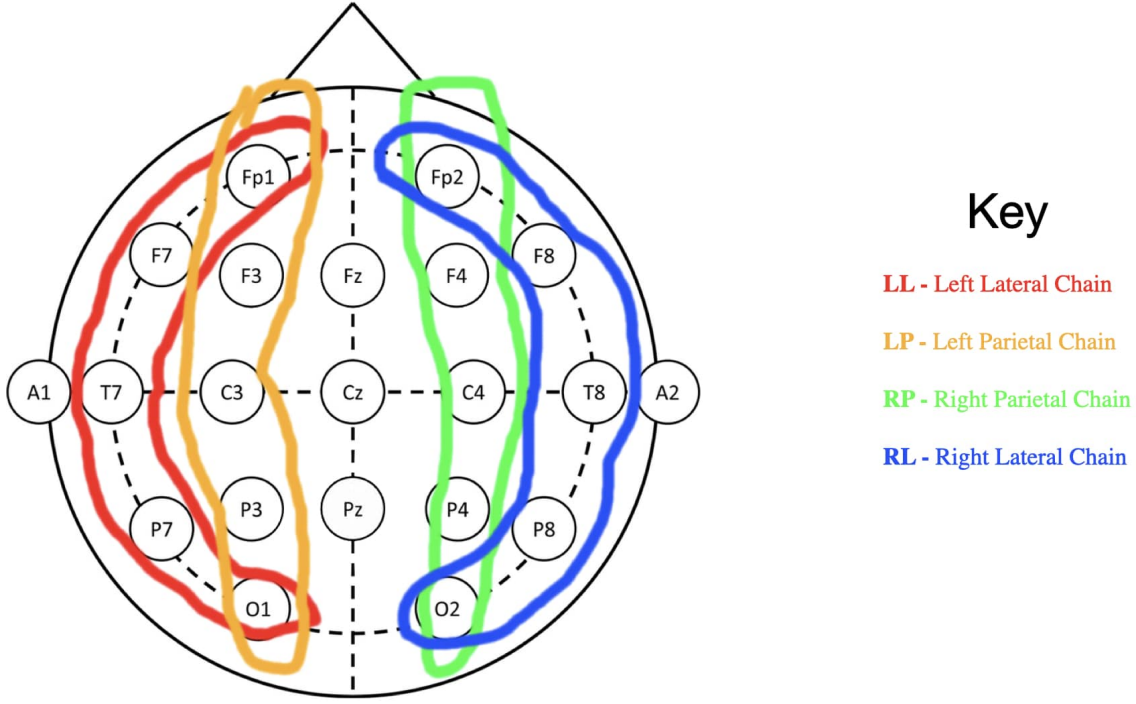


Figure 5: Chains used to caculate timeseries features using a bipolar montage

- *anglez* - a metric derived from individual accelerometer components and is commonly used in sleep detection. It measures the angle of the arm relative to the vertical axis of the body [13].
- *enmo* (Euclidean Norm Minus One) - the Euclidean norm of all accelerometer signals, with negative values rounded to zero. *enmo* is one of several commonly computed features representing the acceleration detected by the accelerometer [13].

B.2 CHB-MIT Scalp EEG Seizure Detection Database

We conducted our experiments on a subset of the CHB-MIT Scalp EEG Database [7]. The original data provided by the Children’s Hospital Boston and consists of EEG recordings from pediatric subjects with intractable seizures [21].

The entire database includes recordings from 22 subjects with a total of 23 cases. Subjects in this dataset were given anti-seizure medication. Each case originally contained between 9 and 42 continuous waveforms, collected from a single subject. These recordings have been segmented into one-hour waveforms, though some cases include longer segments (up to four hours). We only included the continuous waveforms with seizure events.

The EEG signals were sampled at 256 samples per second with 16-bit resolution. All signals used the International 10-20 system of EEG electrode positions and nomenclature [10]. Features were selected to include the same 22 EEG signals, utilizing a bipolar montage for EEG recordings, where the voltage of one electrode is subtracted from another (e.g., FP1-F7). This method reduces common noise and improves the signal-to-noise ratio, enhancing the detection of localized brain activity by focusing on the potential difference between adjacent electrodes.

Electrode were placed according to the International 10-20 system. Figure 5 highlights the bipolar electrode chains used in this study; each chain, for example, the Left Temporal Chain, subtracts neighboring measurements to produce signals like FP1-F7, F7-T7, and others. These pairs are critical for obtaining high-quality EEG signals by focusing on the differential voltages between adjacent electrodes.