Expert Systems

WILEY

Expert Systems

ORIGINAL ARTICLE OPEN ACCESS

Continuous Time Markov Chain for Smartwatch Sensors

Iti Chaturvedi[1] | Wei Liang Seow[2] | Amber Hogarth[1] | Luca Adornetto[1] | Erik Cambria[2]

[1]College of Science and Engineering, James Cook University, Townsville, Queensland, Australia | [2]College of Computing and Data Science, Nanyang Technological University, Singapore

Correspondence: Iti Chaturvedi (iti.chaturvedi@jcu.edu.au)

Received: 22 July 2025 | **Revised:** 28 August 2025 | **Accepted:** 11 September 2025

Funding: The work was supported by the James Cook University, Townsville, Australia, CAUL Open Access.

Keywords: dual control | LSTM | Markov chain | smartwatch

ABSTRACT

Time-series forecasting is essential for predicting events in the future and for tracking objects. The conventional recurrent neural network model needs to pad the target with zeros when handling long inputs, resulting in a loss in accuracy. Recently, it was proposed to divide a time series input into patches and merge the learned weights. However, such a model is difficult to interpret. In this article, we consider a mixture of continuous and discrete Markov states to model long-range time dependencies. For example, in a vehicle, each gear level can be a discrete state and the throttle input is continuously controlled to maximise the efficiency of the engine. Data collected from the sensor is prone to noise due to component faults or external disturbances. Hence, we apply a stability constraint to select samples for training. We validate our algorithm on three datasets: (1) Apple Watch, (2) Car engine and (3) Election tweets. On all datasets, we achieve an improvement in the range of 5%–20% in the F-measure. Furthermore, the features learned are easy to explain in terms of real-world scenarios.

1 | Introduction

In recent years, smartwatches have become increasingly popular. They are capable of capturing signals from users across various activities and emotional states (Wang et al. 2020a). This continuous stream of data has opened up opportunities for applications in personal assistants and smart healthcare (Cambria et al. 2010, 2012; Grassi et al. 2011; Mehta et al. 2020; Wang et al. 2020b). Nonetheless, these devices are susceptible to sudden changes caused by component malfunctions or environmental factors (Chaturvedi, Pandelea, et al. 2024; Chaturvedi et al. 2021). Additionally, to protect user identity, intentional perturbations are introduced into the trained models, which may lead to a reduction in accuracy.

Markov chains typically compute state probabilities based on observations with a limited number of discrete states. However, such models tend to have low memory capacity due to rapidly diminishing gradients from past observations (Chaturvedi, Satapathy, et al. 2024). While discrete-time data is often collected at fixed intervals—such as every second or minute—real-world signals often change continuously, requiring parameter integration over time (Spaeh and Tsourakakis 2024). In these cases, transitions between states do not occur at regular intervals but are instead governed by an exponential random variable. In this work, we introduce a Markov chain model that incorporates both discrete and continuous states to better capture long-range dependencies. We refer to our proposed approach as the Continuous-time Markov Chain (CMC).

To quickly optimise the parameters of the Markov chain, we propose the use of a control system. In (Wang and Hunt 2023) the authors consider a control model to determine the speed of a treadmill that maintains the heart rate close to a reference line. Similarly, in (Zhang et al. 2011) a control model was used to estimate knee movement from wearable sensors. They have indicated that sensors often have high distortions in measurements and hence it is useful to design a controller that can align the

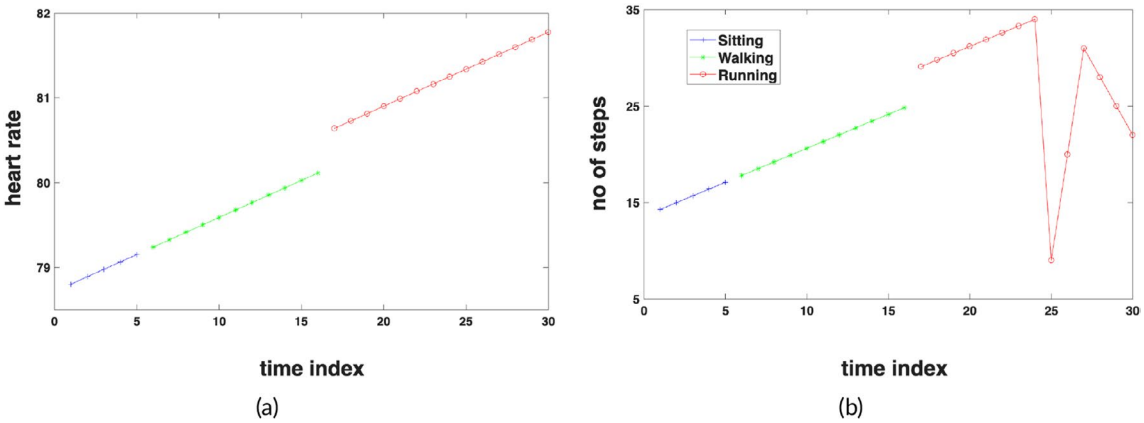

FIGURE 1 | Apple watch data collected during three different activities (a) Hear rate (b) Number of steps. Heart rate increases from sitting to running. Number of steps also increases and then reduces when the stopping. Data is collected at a fixed interval gap of a few minutes.

measured values to the desired values. In this article, we further explore the dual problem of controlling and detecting the models by discarding noisy samples.

We begin our evaluation with data collected from a smartwatch, which is used to predict whether a person is walking or running. Figure 1 shows smartwatch data recorded during three different activities: (a) sitting (blue), (b) walking (green), and (c) running (red). This data is sampled at fixed intervals of a few minutes. As illustrated, the participant's heart rate rises from 78 to 81, and the number of steps increases from 15 to 35 within a single interval. A sharp decline in steps is also observed as the participant slows down. In addition to smartwatch data, we evaluate our approach on telemetry data from a car engine across various gear levels. Finally, we analyse a two-week time series of election-related tweets to detect named entities such as individuals, political parties, and social issues.

The next Section 2 discusses the related work, followed by the theory of the algorithm in Section 3. We validate our approach on real-world datasets in Section 4. Lastly, we provide our conclusions in Section 5.

2 | Related Works

Time series forecasting is applied in a wide range of fields, from traffic prediction to human tracking (Oneto et al. 2016; Cambria, Howard, et al. 2013). Recently, several transformer-based models such as Autoformer and Fedformer (Zhou et al. 2020) have been developed. These models are capable of predicting the hourly temperature of an electrical transformer for up to 20 days, although the mean squared error (MSE) increases significantly after the first day. They effectively capture long-range dependencies in time series by employing a generative-style decoder that performs dimensionality reduction on the input. However, a notable limitation is the slow inference caused by step-by-step decoding. Additionally, zero-padding in the decoder can negatively impact accuracy. Parameter tuning requires an exhaustive grid search to find the optimal settings.

The Informer model utilises a self-attention mechanism with $O(LlogL)$ time complexity and memory usage, where L is the input sequence length. For tuning such complex models, especially in the presence of noisy data, Bayesian optimisation can be applied (Snoek et al. 2012), which assumes a Gaussian prior over the objective function. In (Kong et al. 2025) the authors divide a multivariate time series into univariate channels. Then each univariate series is segmented into patches and processed by a linear layer. They show that they can outperform Long-short-term-memory (LSTM) on different datasets. Also, they demonstrate that, as patch size increases, the prediction accuracy increases first and then decreases. Channel independence is able to avoid over-fitting problems. While they showed lower mean-square error in some time series forecasting, the result is not statistically significant. Next, it cannot model very long-term dependencies in time series. They introduced the concept of multi-head in each LSTM unit with memory mixing using a block diagonal matrix. It is not clear how such a model is more effective and interpretable than traditional regression models.

There has been an explosion in the use of wearable devices as a sedentary lifestyle has been linked to chronic cardiovascular and metabolic diseases (Chaturvedi et al. 2016). These devices have become more readily available, and medical professionals can get detailed and reliable information about their patients' daily activity. Such devices can also send reminders to consumers that they have been sedentary for long durations. Recently, machine learning is being used to identify different activities from wearable trackers. For example, neural networks have been used to extract features from real-time sensor data. (Fuller et al. 2021) combined Apple Watch and Fitbit data to predict activities such as lying, sitting, walking, and running, achieving 98% accuracy. This work highlights the reliability of LSTM-based classification even when device heterogeneity is present. Similarly, (Wang and Liu 2020) proposed a hierarchical deep LSTM model that achieved over 90% accuracy on multiple public datasets for human activity recognition. Their approach demonstrated how deeper LSTM architectures can capture both fine-grained and long-term dependencies in wearable sensor data, further supporting the effectiveness of LSTM-based models for real-time physical activity classification. Building on this, our approach integrates a stability-constrained Markov model using Simulink to enhance interpretability while maintaining strong predictive performance.

The Apple Watch dataset is understudied despite several articles on wearable devices. In (Fuller et al. 2021), the authors showed that even though the rotation forest model had the highest accuracy, the difference was only slightly different from random forests. They used a feature ranking method to understand which variable is the most important. They also conclude that machine learning can combine both device datasets for movement type classification. However, they use device type as a feature during training which can increase the dimensionality of the problem. Another limitation is that there is a lack of proper understanding of what is considered vigorous physical activity in terms of the number of steps, as it will vary across persons.

Lack of explainability has generated mistrust in the use of AI models (Cambria et al. 2023). One way of explaining a model is by using attention to identify the most critical parts of the data that influence the model's prediction (Amin et al. 2023). In the context of the car racing model, explainability is essential in making strategic decisions (Cambria et al. 2014, 2009; Cambria, Rajagopal, et al. 2013; Cambria, Schuller, et al. 2013a, 2013b). In (Villegas and García-Ortiz 2023) the authors focus on the explainability of neural networks in understanding driver performance. Here, feature selection is done to choose the most important features for prediction. They consider data splitting and regularisation to train and evaluate the model. First, they train a model on the racing dataset. Next, different explainability techniques such as attention and feature importance are applied to the trained model. Permutation importance techniques then randomly permute the values of quality and measure the impact of the model's performance. For example, the feature 'technical characteristics' has a negative importance, indicating that it reduces the accuracy of the model. However, they consider a heat map of the neural network weights to predict the attention of different data samples, which is not visibly significant. Furthermore, their parameter tuning was not well-defined and the MSE metric is a weak indicator of model quality. Lastly, they did not do parameter tuning, randomly set the number of LSTM units to 128 and used dropouts to avoid overfitting.

3 | Theory

LSTMs are unstable during training due to the large number of parameters and hence the convergence to global minima is not guaranteed. Instead, if we design a controller, we can show that with suitable initialisation of parameters, the model will converge (Luo et al. 2024). There is a constraint on the number of samples from a single individual from a smartwatch. Here, using a trial-and-error approach for tuning the model is not practical. Instead, it can learn quickly from a hidden-Markov model (HMM) with continuous time states for each activity (Swamy et al. 2022). The traditional transition matrix assumes independence between features, which is not true in the real world, hence a piece-wise integration over continuous time is more suitable for changes in activity levels (Jena et al. 2021; Li et al. 2019).

For discrete HMM, we bin each input feature into three or more levels and the probability for all levels adds up to 1. Instead, the continuous HMM can be viewed as a multiple regression over continuous inputs where the sum of weights adds up to 1. While discrete HMMs use frequencies to compute the probabilities, the continuous HMM does an integration over time, as the time spent in a particular state may vary (Hawkes and Sykes 1990). A control model is also more transparent compared to an LSTM where the features learned in the hidden neurons can only be understood by visualising the activations of input samples. It can be made detectable by enforcing constraints so that the eigenvalues of the feature weights are always negative and hence stable during gradient descent (Mescheder et al. 2018). Next, we detail our implementation of the HMM with both discrete and continuous states and the use of control theory for feature learning.

3.1 | Hidden Markov Model

A HMM is used to identify contextual dependencies between observations in a time series. For example, for smartwatch data, the observable states (O_1, O_2, etc.) are activities such as 'Sitting', 'Walking' and 'Running' that each participant performed in a single session (see Figure 2). The hidden states (s_1, s_2, etc.) represent underlying contextual patterns that influence the transitions between these entity types. For instance, how likely is the activity of walking to be followed by running.

In a Markov chain, a state i is accessible from a state j if it is possible to begin in j and arrive in i in finite number of words. The probability of transitioning from state i to state j is called

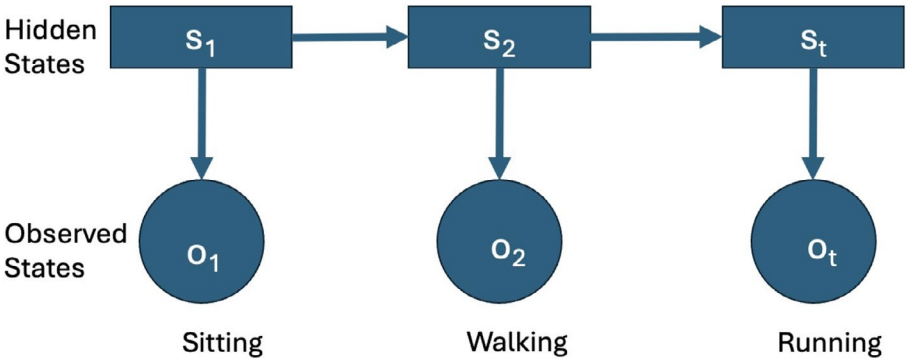

FIGURE 2 | A hidden Markov Model with three observed states O_t for 'Sitting', 'Walking' and 'Running' conditional on the hidden states s_t 'Heart rate', 'Number of Steps' and 'Intensity', where t is the time index for single participant.

the transition probability. The state transition probability is given by:

$$\lambda_{ij} = \frac{m_{ij}}{\sum_{j=1}^{3} m_{ij}} \text{ where } \sum_{i \neq j} \lambda_{ij} = 1 \tag{1}$$

where m_{ij} is the number of occurrences where the activity type O_i is followed by the activity type O_j in the smartwatch dataset.

Here, we consider three hidden dimensions, namely: (1) Heart rate measured by the smartwatch (2) The number of steps measured by the smartwatch over an interval of few minutes (3) The intensity of the activity. The three hidden dimensions are discretized into three levels. Next, we compute the emission and transition probabilities that maximize the likelihood of observed states given the hidden states for each participant.

For the continous time case we can model the system using a transition function $r(t)$ such that:

$$\begin{aligned} y(t) = C(r(t))x(t) \text{ where } \frac{dx(t)}{dt} = A(r(t))x(t) \\ \text{and } \lambda_{ij} = p(r(t+\delta) = j \mid r(t) = i) \end{aligned} \tag{2}$$

where $y(t)$ is the output of the system, δ is the variable change in time, $A(r(t))$ and $C(r(t))$ are the input and output weights.

Next, if we assume that the $r(t)$ jumps from one Markov state to another at time instants $t_0, t_1, \ldots, t_T$ then:

$$\begin{aligned} \Gamma(t_0, t_T) = \Gamma(t_0, t_1) + \Gamma(t_1, t_2) + \ldots + \Gamma(t_k, t_T) \quad \text{where} \\ \Gamma(t_k, t_{k+1}) = \exp\left[A'_{t_0}(t_0 - t_1)\right] \ldots . \exp\left[A'_{i_{k-1}}(t_{k-1} - t_k)\right] \\ \Gamma(t_k) . \exp\left[A_{i_{k-1}}(t_{k-1} - t_k)\right] \exp\left[A_{i_0}(t_1 - t_0)\right] \end{aligned} \tag{3}$$

where:

$$\Gamma(t_k) = \int_{t_k}^{t_{k-1}} \exp\left[A'_{i_k}(\tau - t_k)\right] C'_{i_k} C_{i_k} \exp\left[A'_{i_k}(\tau - t_k)\right] d\tau \tag{4}$$

then the system is piecewise constant in parameters $A(r(t))$ and $C(r(t))$ and $\Gamma(t_0, t_T)$ is the observability matrix that ensures the system is detectable.

3.2 | Stability Constraint

The HMM described by Equation (1) assumes discrete transition states from one time point to the next; however, in the real world, transition states may change continuously and are more suitably calculated using an integration over the time interval as given by Equation (4). Figure 3 illustrates a feedback control system for human activity prediction. Here, the objective is to maximise the speed of running given the input 'heart rate', 'number of steps' and 'intensity'. Here we are faced with a dual control problem of balancing two objectives: (1) controlling the system effectively for stability; (2) detecting and improving future states of the system using feedback.

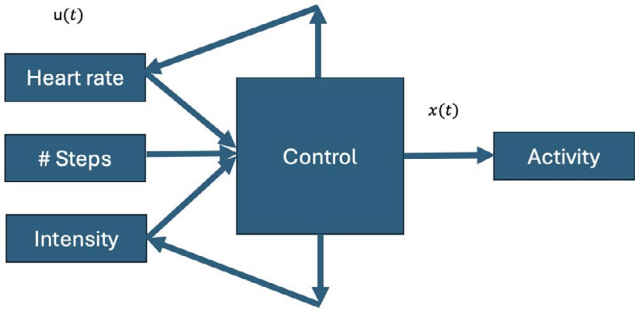

FIGURE 3 | A feedback control unit for predicting human activity from three inputs: (a) Heart rate (b) Number of steps (c) Intensity. The control unit will try to maximise the efficiency of running.

A system can be controlled by providing external inputs $u(t)$ that enable the system to complete tasks, where t is the discrete time index, such as seconds or minutes. Detectability is about estimating the internal states of the system $x(t)$ over time. The hidden states are related to the input signal by $u(t) = -Lx(t)$. The system is designed to remain stable over time, hence the error between the observed output $x(t)$ and the desired output $u(t)$ approaches zero where $\{L\}_{n \times n}$ is the learned weight matrix for the n inputs in $u(t)$. With different control inputs, it becomes difficult for the system to minimise errors. This is referred to as the dual control problem. In this work, we aim to manage this problem by considering changes in weights continuously over time, instead of discretely, by using a piecewise detectability function where each sub-interval of time is modelled by a separate function and added together.

We can optimise both the controllability and detectability constraints using the following two inequalities (Ji and Chizeck 1990):

$$\begin{aligned} 0 \geq \left(A_i - B_i L_i - \frac{1}{2}\lambda_i I\right)' M_i + M_i \left(A_i - B_i L_i - \frac{1}{2}\lambda_i I\right) + \sum_{i \neq j} \lambda_{ij} M_j \\ 1 \geq \| \exp\left(A'_i - L'_i M'_i M_i\right) \| \end{aligned} \tag{5}$$

where A are the weights for hidden states $x(t)$, B are weights for inputs $u(t)$, I is the identity matrix, $\| Z \|$ is the highest eigenvalue of the matrix $Z'Z$ and $\{M\}_{n \times n}$ is a positive semidefinite matrix to be determined.

For smooth convergence of a continuous time Markov chain to global minima of error, we find that the following inequality must hold for each time sample:

$$x'(t)x(t) \geq x'(0)Mx(0) \tag{6}$$

where $x(0)$ is the first training sample. For each training sample, we select a subset that obeys the inequality given by Equation (6). Next, we train the hidden states of CMC using this subset of samples and use the output features for classification. We can use this inequality to select a subset of training samples to determine the weights of the system.

4 | Experiments

In this section, we evaluate CMC (available on GitHub[1]) on real-world datasets. We have used the same training procedures on all three datasets to allow for benchmarking across them. We first evaluate CMC on a short-time series smartwatch dataset that has a high level of noise due to movement during an activity. Next, we use it to predict the speed of a car during racing. This is a long time series and the sensors are reliable due to a controlled racing environment. Lastly, we apply it to a stream of Tweets during an election campaign. Here, the aim was to model numerous entities that collectively influence sentiments and determine the election results.

4.1 | Parameter Tuning

To determine the optimal parameters for the LSTM and Informer baselines, a trial-and-error approach on a validation set is used on the watch dataset. Figure 4 shows the F-measure plotted against different parameter settings, namely (a) number of previous time steps, (b) number of neurons, (c) batch size and (d) size of training data.

In Figure 4a, the effect of the number of previous time steps in a sequence was considered. The F-measure increases with the number of time steps and then reduces. This can be explained by the fact that only around 30 samples were collected by the smartwatch for a single individual. In Figure 4b, the effect of the number of hidden neurons was studied. The F-measure increased slightly when the number of neurons was increased from 8 to 16. However, a further increase to 24 reduced the F-measure due to over-fitting of the model.

In Figure 4c, the effect of increasing the batch size from 1 to 16 is considered. F-measure decreases rapidly with increasing batch size. However, the speed of training also increases. Hence, a batch size of 8 was found to be optimal. In Figure 4d as expected, we find that the validation F-measure decreases with the percentage of training samples used. From the available samples following a heuristic approach, 70% are used for training, 10% for validation and 20% are used for testing.

4.2 | Watch

This dataset contains 6264 samples from 46 participants with 26 women and 20 men. They wear three types of devices, including GENEActiv, Apple Watch and Fitbit. Participants completed 40 min of total treadmill time and 25 min of sitting or lying time for a total of 65 min protocol. We consider a subset of 2769 samples from three activities, namely: sitting, walking and running.

Next, we use a Simulink model with three inputs, namely: (a) the number of steps, (b) the heart rate and (c) the Karvoven intensity. We selected these features from the smartwatch dataset as they were the most significant and the remaining variables were highly correlated to them. Like an LSTM, the Simulink feedback control model applies a controller to maximise the accuracy of predicting the activity of a person for a given input.

For an input combination where the hidden state of the model is not stable, the efficiency will be low. Hence, we can use a Markov model to represent three hidden states of activities and apply a constraint to select a subset of 1278 samples for which the prediction efficiency in the training data is high. The three states of the Markov chain are analogous to different activity levels.

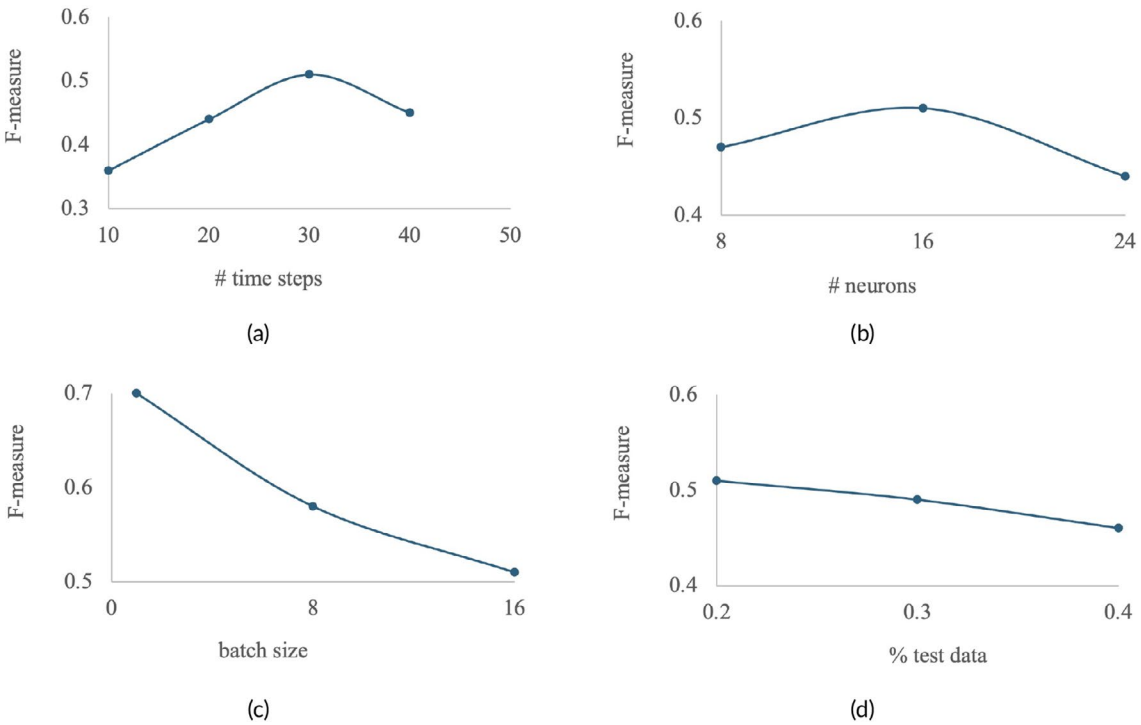

FIGURE 4 | Effect of parameters in LSTM on F-measure (a) Number of previous time steps (b) Number of neurons (c) Batch size (d) Percentage of test samples.

Hence, we can determine the transition probabilities when the state changes from one activity to another.

Finally, we simulate the MATLAB model again to determine the controlled hidden states and use them to predict the label using an LSTM. Table 1 compares the F-measure of the proposed algorithm CMC with different baselines. The highest value in each column is shown in bold. The last column is the weighted sum normalised by the number of samples in each class in the testing dataset. Our method can outperform Informer by almost 40% in the F-measure. Informer performed the worst in 'running' class. LSTM has 22% lower F-measure than CMC. It does well in 'running' motion, however it performs poorly in detecting the 'walking' class. We can conclude that our method is effective in classifying sensor measurements from smartwatches.

To further compare the stability of different algorithms, we consider the mean and standard deviation of each activity class using 10-fold cross-validation. Figure 5 compares the F-measure of different algorithms across different activities. We can see that 'running' activity performs the best in most models. LSTM and NN show a very large variance for the sitting activity, suggesting poor convergence due to noisy samples. CMC has the highest F-measure and also has low variance on all activity types.

Lastly, we visualise the features learned at the hidden neurons of the LSTM for different activities in the watch dataset. Figure 6 compares the activations for the original dataset and those for CMC. The time step index shows the activity at that time where 's' denotes sitting, 'w' denotes walking and 'r' denotes running. We can see that the activations dont change much for the original dataset. However, after selecting samples using stability constraint and running the control model, significantly different activations are observed for different classes. For example, neuron 2 is activated during walking and neuron 3 is highly activated during running.

TABLE 1 | Comparison of baselines on watch dataset. CMC outperforms baseline LSTM by over 22% in *F*-measure.

Method	Sitting	Walking	Running	Total
NN	0.5	0.42	0.42	0.45
k-NN	0.73	0.74	0.77	0.74
Informer	0.71	0.30	0.14	0.53
LSTM	0.7	0.65	0.74	0.7
CMC	**0.9**	**0.91**	**0.94**	**0.92**

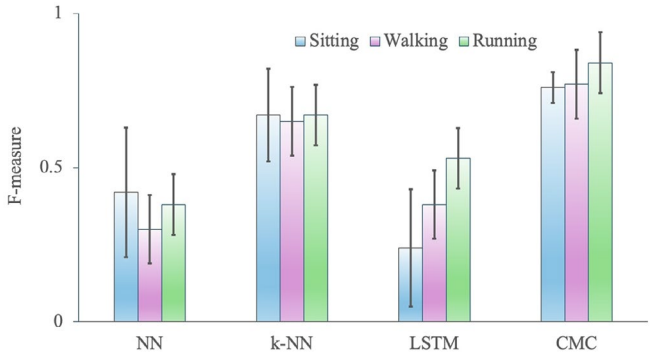

FIGURE 5 | This graph shows the 10-fold cross-validation F-measure and variance for each activity class in the watch dataset. LSTM and NN show very large variance for the sitting activity.

4.3 | Car

The car dataset was downloaded using the FastF1 library (Grover 2022) which is an open-source python library for accessing F1 telemetry data such as speed, throttle etc. of each car during a race. This dataset contains historical telemetry data for each race, such as the speed of a driver, his position and any accidents on each lap. For this article, we randomly selected several different racetracks and drivers and combined them, resulting in a time series of 4597 observations where each race has around 300 time points. This data was used as input to the spark engine control model. We consider three features, namely: (1) the throttle, (2) revolutions per min and (3) break as input. The model was trained to predict the gear levels, which were discretised into three values where 1 represents the lowest speed and 3 is the highest speed.

Next, we used an existing Simulink model for a spark engine where we replaced the spark angle with the rpm. The throttle and the break were used as such. Like an LSTM, the spark engine model applies a controller to maximise the efficiency of the engine and predict the speed of the vehicle for a given input.

For an input combination where the hidden state of the model is not stable, the efficiency will be low. Hence, we can use a Markov model to represent three different gear levels and apply a constraint to select a subset of 1175 samples for which the prediction efficiency in the training data is high. The three states of the Markov chain are analogous to different gear levels in a spark engine. Hence, we can determine the transition probabilities when the state changes from one gear to the next.

Finally, we simulate the MATLAB model again to determine the controlled hidden states and use them to predict the labels using an LSTM. Table 2 compares the F-measure of the proposed algorithm CMC with different baselines. The highest value in each column is shown in bold. Our method can outperform Informer and LSTM by almost 11% in the F-measure. Informer performed well on this dataset compared to the other two datasets. It could be because all the drivers had taken similar paths. We have almost 30% improvement at the speed of 'Gear 2'. This is expected as we used the Simulink model for a spark engine that is optimised for car data.

4.4 | Elections

This dataset was created on Twitter using hashtags relevant to the 2025 Queensland elections. We crawled tweets daily for two weeks until the elections and for one week after the event. We

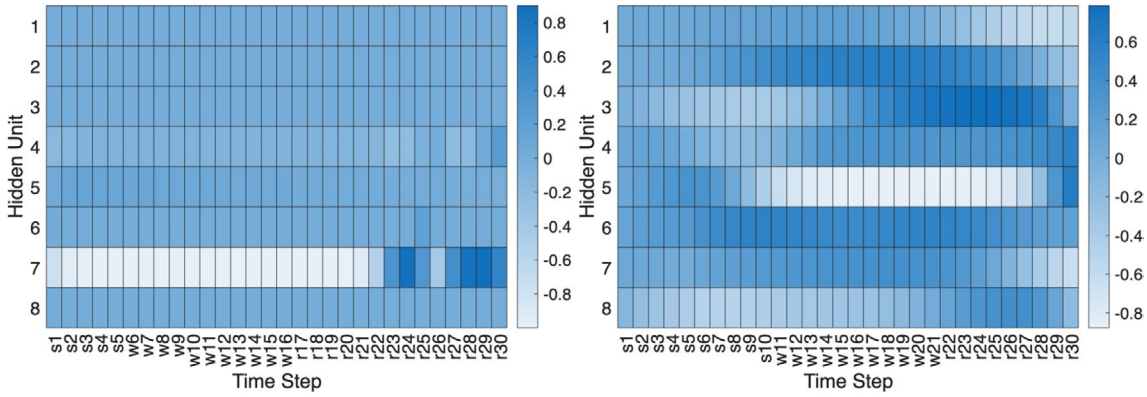

FIGURE 6 | Visualisation of features learned at eight hidden neurons in an LSTM for sitting (s), walking (w) and running (r) (a) Original dataset (b) CMC Features. CMC features are significantly different across activities.

TABLE 2 | Comparison of baselines on car dataset. CMC outperforms baseline Informer by over 12% in *F*-measure.

Method	Gear 1	Gear 2	Gear 3	Total
NN	0.69	0.21	0.62	0.55
k-NN	0.79	0.1	0.42	0.43
Informer (Zhou et al. 2020)	0.36	0.55	0.71	0.65
LSTM	0.11	0.65	0.72	0.65
CMC	**0.38**	**0.86**	**0.61**	**0.76**

used ChatGPT to label a subset of 2767 tweets for named entities such as 'Person' or 'Political Party'. For each tweet we computed the sentiment score using a previously trained model. We found 4830 entities across the three classes. Entities without Glove word vectors were discarded, resulting in 2783 entities.

Glove contains word embeddings of length 300 for 6 billion words in the English language. These are generated from co-occurrence data in social media articles. However, to determine the context of a word, we use the cosine angle between two consecutive words in an election tweet. For example, when the entity 'Labor' is followed by the word 'women' in a tweet. Then we can determine that the social issue 'women' is being targeted by the political party 'Labor' (Zhou et al. 2022).

We removed tweets for which a sentiment score could not be determined and balanced the dataset across the three class types, resulting in 1965 named entities. Next, we can determine the magnitude of each named entity from the word vector. This will represent its relative position in the vector space. Lastly, to capture the positive or negative emotional content in the tweet, we compute the sentiment score using a pretrained sentence classifier. In this work we only consider three entity types, namely: 'person', 'political party' and 'social issue'.

Next, we used a Simulink model with three inputs namely (a) cosine of word vector, (b) the magnitude of the word vector and (c) the sentiment score. These features were selected as explained above and discussed in our recent review article on named

TABLE 3 | Comparison of baselines on election dataset. CMC outperforms baseline LSTM by over 7% in F-measure.

Method	Person	Political party	Social issue	Total
BERT (Devlin et al. 2019)	0.69	0.83	0.46	0.66
GLINER (Zaratiana et al. 2024)	0.73	0.91	0.57	0.74
RAG-NER (Zhenwei et al. 2024)	0.69	0.85	0.21	0.58
Agent RAG-NER (Yao et al. 2023)	0.47	0.70	0.19	0.47
Informer (Zhou et al. 2020)	0.78	0.6	0	0.65
LSTM	0.83	0.42	0.97	0.77
CMC	**0.8**	**0.92**	**0.81**	**0.85**

enitity recognition (Dhelim et al. 2022). Like an LSTM, the Simulink feedback control model applies a controller to maximise the accuracy of predicting the entity class of a word based on previous words in the tweet or discussion.

For an input combination where the hidden state of the model is not stable, the efficiency will be low. Hence, we can use a Markov model to represent three hidden states of named entities and apply a constraint to select a subset of 1196 named entities for which the prediction efficiency in the training data is high. The three states of the Markov chain are analogous to different types of named entities. Hence, we can determine the transition probabilities when the state changes from one named entity to another.

Finally, we simulate the MATLAB model again to determine the controlled hidden states and use them to predict the labels using a LSTM. Table 3 compares the F-measure of the proposed algorithm CMC with different baselines. The highest value in each column is shown in bold. Our method can outperform the pretrained BERT model and Informer by almost 20% in F-measure.

We have almost 40% improvement in the subclass 'Social Issue'. This could be because we are able to determine the context of a 'social issue' with respect to a 'person' or 'political party' by using a Markov chain of named entities in a sentence. LSTM has a 10% lower F-measure than CMC, it does well on 'social issue' entities. However, it performs poorly on entities from the 'political party' class.

5 | Conclusion

In this paper, we explored a model with both discrete and continuous Markov states to capture long-term dependencies in time-series data. To model the discrete states we use a hidden Markov model. The continuous dependence between states is modelled by a piecewise exponential random variable. Baseline models are heavily dependent on the use of attention between consecutive observations and need a lot of parameter optimisation. Instead, here we consider the dual problem of controlling and detecting the system using a spark engine control model. A subset of training samples is selected using a stability constraint.

We consider the problem of determining human activity from wearable sensors and outperform the baselines by over 10% in the F-measure. We also apply the method to classify a time-series of election tweets collected from social media. Lastly, we can ensure that the system is detectable and hence easy to interpret. Predicting activity from sensors presents challenges, particularly due to movement disturbances that may cause the device to lose contact. In the future, we plan to enhance the system by incorporating body posture information captured through a mobile camera. We also plan to explore the use of Bayesian Optimisation to initialise the parameters of the control model.

Acknowledgments

Open access publishing facilitated by James Cook University, as part of the Wiley - James Cook University agreement via the Council of Australian University Librarians.

Data Availability Statement

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Endnotes

[1]http://github.com/ichaturvedi/continous-time-markov.

References

Amin, M., E. Cambria, and B. Schuller. 2023. "Will Affective Computing Emerge From Foundation Models and General AI? A First Evaluation on ChatGPT." *IEEE Intelligent Systems* 38, no. 2: 15–23.

Cambria, E., T. Benson, C. Eckl, and A. Hussain. 2012. "Sentic PROMs: Application of Sentic Computing to the Development of a Novel Unified Framework for Measuring Health-Care Quality." *Expert Systems With Applications* 39, no. 12: 10533–10543.

Cambria, E., N. Howard, J. Hsu, and A. Hussain. 2013. *Sentic Blending: Scalable Multimodal Fusion for Continuous Interpretation of Semantics and Sentics*, 108–117. IEEE SSCI.

Cambria, E., A. Hussain, T. Durrani, C. Havasi, C. Eckl, and J. Munro. 2010. *Sentic Computing for Patient Centered Applications*, 1279–1282. IEEE ICSP.

Cambria, E., A. Hussain, C. Havasi, and C. Eckl. 2009. "Common Sense Computing: From the Society of Mind to Digital Intuition and Beyond." *Biometric ID Management and Multimodal Communication* 5707: 252–259.

Cambria, E., L. Malandri, F. Mercorio, M. Mezzanzanica, and N. Nobani. 2023. "A Survey on XAI and Natural Language Explanations." *Information Processing & Management* 60: 103111.

Cambria, E., D. Rajagopal, D. Olsher, and D. Das. 2013. *Big Social Data Analysis*, edited by R. Akerkar, 401–414. Big Data Computing.

Cambria, E., B. Schuller, B. Liu, H. Wang, and C. Havasi. 2013a. "Knowledge-Based Approaches to Concept-Level Sentiment Analysis." *IEEE Intelligent Systems* 28, no. 2: 12–14.

Cambria, E., B. Schuller, B. Liu, H. Wang, and C. Havasi. 2013b. "Statistical Approaches to Concept-Level Sentiment Analysis." *IEEE Intelligent Systems* 28, no. 3: 6–9.

Cambria, E., H. Wang, and B. White. 2014. "Guest Editorial: Big Social Data Analysis." *Knowledge-Based Systems* 69: 1–2.

Chaturvedi, I., Y. S. Ong, I. Tsang, R. Welsch, and E. Cambria. 2016. "Learning Word Dependencies in Text by Means of a Deep Recurrent Belief Network." *Knowledge-Based Systems* 108: 144–154.

Chaturvedi, I., V. Pandelea, E. Cambria, R. E. Welsch, and B. Datta. 2024. "Barrier Function to Skin Elasticity in Talking Head." *Cognitive Computation* 16, no. 6: 3405–3416.

Chaturvedi, I., R. Satapathy, C. Lynch, and E. Cambria. 2024. "Predicting Word Vectors for Microtext." *Expert Systems* 41, no. 8: e13589.

Chaturvedi, I., K. Thapa, S. Cavallari, E. Cambria, and R. E. Welsch. 2021. "Predicting Video Engagement Using Heterogeneous DeepWalk." *Neurocomputing* 465: 228–237.

Devlin, J., M. W. Chang, K. Lee, and K. Toutanova. 2019. *BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding*, 4171–4186. HLT-NAACL.

Dhelim, S., N. Aung, M. Amine Bouras, H. Ning, and E. Cambria. 2022. "A Survey on Personality-Aware Recommendation Systems." *Artificial Intelligence Review* 55: 2409–2454.

Fuller, D., J. Anaraki, B. Simango, et al. 2021. "Predicting Lying, Sitting, Walking and Running Using Apple Watch and Fitbit Data." *BMJ Open Sport & Exercise Medicine* 7, no. 4: e001004.

Grassi, M., E. Cambria, A. Hussain, and F. Piazza. 2011. "Sentic Web: A New Paradigm for Managing Social Media Affective Information." *Cognitive Computation* 3, no. 3: 480–489.

Grover, R. 2022. "Analysing the Importance of Qualifying in Formula 1 Using the Fastf1 Library in Python." *International Journal of Advanced Research* 10, no. 8: 1138–1150.

Hawkes, A. G., and A. M. Sykes. 1990. "Equilibrium Distribution of Finite-State Markov Processes." *IEEE Transactions on Reliability* 39, no. 5: 592–595.

Jena, R., C. Liu, and K. Sycara. 2021. "Augmenting GAIL With BC for Sample Efficient Imitation Learning." In *Proceedings of the 2020 Conference on Robot Learning (CoRL 2020)*, vol. 155, 80–90. PMLR.

Ji, Y., and H. J. Chizeck. 1990. "Controllability, Stabilizability, and Continuous-Time Markovian Jump Linear Quadratic Control." *IEEE Transactions on Automatic Control* 35, no. 7: 777–788.

Kong, Y., Z. Wang, Y. Nie, et al. 2025. "Unlocking the Power of LSTM for Long Term Time Series Forecasting." *Proceedings of the AAAI Conference on Artificial Intelligence* 39, no. 4: 11968–11976.

Li, Y., Q. Pan, S. Wang, H. Peng, T. Yang, and E. Cambria. 2019. "Disentangled Variational Auto-Encoder for Semi-Supervised Learning." *Information Sciences* 482: 73–85.

Luo, T., T. Pearce, H. Chen, J. Chen, and J. Zhu. 2024. "C-GAIL: Stabilizing Generative Adversarial Imitation Learning With Control Theory." *Advances in Neural Information Processing Systems* 38: 29464–29488.

Mehta, Y., S. Fatehi, A. Kazemeini, C. Stachl, E. Cambria, and S. Eetemadi. 2020. *Bottom-Up and Top-Down: Predicting Personality With Psycholinguistic and Language Model Features*, 1184–1189. ICDM.

Mescheder, L., A. Geiger, and S. Nowozin. 2018. "Which Training Methods for GANs do Actually Converge?" In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 3481–3490. PMLR.

Oneto, L., F. Bisio, E. Cambria, and D. Anguita. 2016. "Statistical Learning Theory and ELM for Big Social Data Analysis." *IEEE Computational Intelligence Magazine* 11, no. 3: 45–55.

Snoek, J., H. Larochelle, and R. P. Adams. 2012. "Bayesian Optimization With Gaussian Processes." In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)*, 2960–2968. Curran Associates, Inc.

Spaeh, F. C., and C. Tsourakakis. 2024. "Learning Mixtures of Continuous-Time Markov Chains." In *The Web Conference 2024*.

Swamy, G., N. Rajaraman, M. Peng, et al. 2022. "Minimax Optimal Online Imitation Learning via Replay Estimation." *Advances in Neural Information Processing Systems (NeurIPS 2022)* 35: 26313–26326.

Villegas, W., and J. García-Ortiz. 2023. "An Approach Based on Recurrent Neural Networks and Interactive Visualization to Improve Explainability in AI Systems." *Big Data and Cognitive Computing* 07, no. 7: 136.

Wang, H., and K. J. Hunt. 2023. "Feedback Control of Heart Rate During Treadmill Exercise Based on a Two-Phase Response Model." *PLoS One* 18, no. 10: e0292310.

Wang, L., and R. Liu. 2020. "Human Activity Recognition Based on Wearable Sensor Using Hierarchical Deep LSTM Networks." *Circuits, Systems, and Signal Processing* 39: 837–856.

Wang, Z., S. Ho, and E. Cambria. 2020a. "A Review of Emotion Sensing: Categorization Models and Algorithms." *Multimedia Tools and Applications* 79, no. 41: 35553–35582.

Wang, Z., S. B. Ho, and E. Cambria. 2020b. "Multi-Level Fine-Scaled Sentiment Sensing With Ambivalence Handling." *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 28, no. 4: 683–697.

Yao, S., J. Zhao, D. Yu, et al. 2023. "React: Synergizing Reasoning and Acting in Language Models." In *ICLR 2023*.

Zaratiana, U., N. Tomeh, P. Holat, and T. Charnois. 2024. *GLiNER: Generalist Model for Named Entity Recognition Using Bidirectional Transformer*, 5364–5376. HLT-NAACL.

Zhang, Z., C. W. Wong, and J. Wu. 2011. "Wearable Sensors for 3D Upper Limb Motion Modeling and Ubiquitous Estimation." *Journal of Control Theory and Applications* 9, no. 1: 10–17.

Zhenwei, D., C. Luo, Z. Li, et al. 2024. "RA-NER: Retrieval Augmented NER for Knowledge Intensive Named Entity Recognition," The Second Tiny Papers Track at ICLR 2024.

Zhou, H., S. Zhang, J. Peng, et al. 2020. "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting." *Proceedings of the AAAI Conference on Artificial Intelligence* 35, no. 12: 11106–11115.

Zhou, R., X. Li, R. He, et al. 2022. *MELM: Data Augmentation With Masked Entity Language Modeling for Low-Resource NER*, 2251–2262. ACL.