FALL DETECTION USING TIME2VEC

AND TRANSFORMERS

by

Shahroz Noorani, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
December 2023

Committee Members:

Anne Hee Hiong Ngu, Chair

Vangelis Metsis

Byron Gao

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF EQUATIONS

Page

ABSTRACT

Falls in older adults can have many lasting health and financial problems if not handled properly or swiftly. The WHO estimates that, on average, 35% of older adults over the age of 60 face a fall during the year, and this number only increases with older age groups (World Health Organization, 2008). Addressing this critical concern, this research ventures into augmenting fall detection mechanisms by harnessing the capabilities of Time2Vec alongside Transformer models. Our goal is to aim for accurate detection of falls so that needed help can be rendered as soon as possible. Drawing upon established machine learning paradigms, we performed extensive experiments employing transformer encoder layers on different data types and sensors.

Time2Vec was incorporated as a vectorization layer to address the limitation of transformers in dealing with sequencing in the absence of a positional encoder. Examining various feature subsets, we established that vectorizing the x-axis of the accelerometer data using Time2Vec significantly enhances the stacked encoder model's performance, creating a robust mechanism for fall detection. This adjustment led to an optimized model capable of efficiently identifying falls, thereby holding substantial promise in mitigating the adverse impacts associated with falls in the elderly population. Our endeavors resulted in an overall improvement of 12% in F1 scores, especially in larger datasets compared to previous architectures. The findings from this research contribute to the ongoing efforts in enhancing fall detection and underscore the potential of Time2Vec and Transformer models for monitoring other health conditions when time series data are involved.

1. INTRODUCTION

Falls in elderly adults cause severe harm and financial burdens on both the patients and their families. The Centers for Disease Control (CDC) reports that falls can cause traumatic hip and brain injuries and even double the chance of falling again after the initial fall. These falls have caused $50 billion in medical costs in the US alone in 2015 (Centers for Disease Control and Prevention, 2023). Falls can happen anywhere at any time and can be detrimental if incidents are not promptly handled. This is why more research has gone into fall prevention, fall risk assessment, and fall detection over the years.

Research over the past two decades has been focused on the best ways to detect and prevent falls from happening, especially fall detection. We see this in comprehensive review papers, applications in smart devices, prevention protocols (Centers for Disease Control and Prevention, 2023), and other places. In terms of applying fall detection, we have seen results from many different methodologies, including traditional machine learning like Naïve Bayes and support vector machines (SVMs) to deep learning methods like long-short term memory models (LSTM) and convolutional neural networks (CNNs).

We have also seen many datasets used in training and testing methods, such as acceleration and gyroscope datasets to skeletal and images that capture falls and activities of daily life (ADLs). Some papers show that traditional machine learning methodologies, like SVMs, do well in a simulated environment (Liu & Cheng, 2012). In recent years, fall detection has turned to more deep learning approaches, with Smartfall (Mauldin et al., 2018) and CNN+LSTM (Ordóñez & Roggen, 2016) being examples of deep learning architectures used for detection that have improved on results of traditional machine learning.

Currently, work is being done to improve the capabilities of mobile fall applications. This

is because previous body sensors or specialized equipment methods have been noted to have spatial limitations, are obtrusive, and are expensive to acquire and maintain. With the development of phone or watch applications (RightMinder, 2017), we can use the acceleration and gyroscope sensors that are already within these devices, like the Apple Watch 9 (Apple, 2023) or Pixel Watch (Google, 2022), to provide training and testing data to create deep learning models that can generalize better, detect falls and ADLs more efficiently, and can still be used with mobile hardware despite hardware limitations on these devices. Previously, Smartfall (Mauldin et al., 2018) was developed and trained using a smartwatch to demonstrate the capabilities of LSTM methods, more specifically, Gated recurrent units (GRUs) networks. Although GRU networks are computationally heavier than SVMs or Naïve Bayes models, they perform significantly better in detecting falls on a smartwatch.

In this paper, we focus on the use of the transformer encoder model within the aspect of fall detection and how we can use the transformer to ingest both acceleration and gyroscopic data to detect falls that occur. Initially developed for natural language processing, transformers offer a transformative approach to analyzing sequential data. They leverage a self-attention mechanism for holistic data processing, capturing intricate relationships between elements without relying on their relative positions. Transformers play a pivotal role by enabling the practical analysis of time series data from accelerometers and gyroscopes. Their strength lies in modeling long-range dependencies within the data, which is crucial for detecting falls.

Given that fall detection requires analysis of time-series data, which is sequential information in nature, it is imperative to find effective methods for managing time-series data without sacrificing the temporal context. One promising solution is Time2Vec (Kazemi et al., 2019), a vectorization layer designed to enhance data readability within the primary transformer

architecture. Time2Vec achieves this by translating accelerometer or gyroscope data into vectors. What sets Time2Vec apart is its ability to enable the transformer to learn from acceleration and gyroscope sequences while remaining invariant to unit conversions. Importantly, Time2Vec excels in handling non-periodic events, making it an ideal choice for enhancing the attention mechanisms of the encoder layers.

The effectiveness of Time2Vec has been demonstrated in various domains, showcasing its versatility and potential. For instance, it has been pivotal in accurately predicting energy consumption when integrated into a hybrid architecture that combines a Transformer model with Stationary Wavelet Transform (SWT) (Saoud et al., 2022). This combination, featuring Time2Vec, achieved remarkably low Root Mean Squared Error (RMSE) values compared to traditional LSTM models and standard transformers. Such successes underscore the transformative impact of Time2Vec in enhancing the understanding of sequential data patterns, a capability of paramount importance in fall detection.

Within our proposed architecture, the integration of Time2Vec embeddings—a technique that enhances input readability and empowers transformers to extract meaningful features from sensor data efficiently is highlighted. By using transformers with Time2Vec embeddings, our research aims to advance fall detection methodologies, offering a versatile and robust solution that can revolutionize the field. The main contributions of this thesis are:

- Improving the methodologies of the transformer using Time2Vec as a vectorization machine to improve input readability for better results.
- Explore different datasets to verify the performance of the transformer with Time 2Vec embedding.

The remainder of this article is organized as follows: Section II provides related works to

provide a more thorough background on transformers, Time2Vec, and fall detection. Section III describes the proposed methodologies and provides the framework for experimentation. Section IV offers data collection, experimental parameters, and other aspects of the experiments. Section V provides the experimental results, and Sections VI and VII offer final thoughts and future directions.

2. RELATED WORKS

Fall detection has witnessed extensive exploration through numerous studies, employing various devices, methodologies, and machine learning approaches.

Smartfall (Mauldin et al., 2018) begins with collecting triaxial acceleration data, an Internet of Things (IoT) software infrastructure, a machine learning model, and a SmartFall App. The LSTM network, a key component of Smartfall's machine learning framework, has an input layer, a core LSTM layer with twenty cells, and two dense layers for output generation. This neural network is adept at processing sequential data, a fundamental requirement for accurate fall detection. A distinctive feature of Smartfall is its adoption of the sliding window approach for data processing of the streaming accelerometer data. This mechanism accumulates a specified number of data points before subjecting them to the LSTM's prediction, ensuring contextual analysis that enhances fall detection accuracy.

The Smartfall system uses a three-layer flexible software architecture that works with various commodity smartwatches (Microsoft Band, Huawei Watch, and TipWatch). Moreover, it has developed an App that can trigger alerts to carers or local emergency services when a fall is detected and the user needs help. The SmartFall system eliminates the need for expensive and cumbersome devices, such as cameras and intrusive body sensors, achieving results with just a phone and a smartwatch, which are pervasive and commonly used by older adults.

The results from Smartfall's Deep Learning model showed that it outperformed traditional machine learning models across three datasets due to its ability to learn subtle features from the raw accelerometer data, which were unavailable to Naive Bayes and Support Vector Machine models. The fall detection model showed better accuracy in predicting falls based on live wrist-worn acceleration data in offline and online/real-time experiments. While LSTM

networks are the most optimal for fall detection, transformers can be used in place to shorten training time, recognize both short- and long-term patterns, and, in theory, scale effectively to large datasets.

In “A Study of the Use of Gyroscope Measurements in Wearable Fall Detection Systems” (Casilari et al., 2020, p. 649), researchers delved into the realm of Fall Detection Systems (FDS), probing the integration of gyroscope data with Convolutional Neural Networks (CNN). Traditionally heralded for image analysis, CNNs, when tailored for 1D convolution, exhibit prowess in processing time-series data. This makes them suitable for analyzing sequences of gyroscope measurements over time, especially when dealing with multi-channel input from 3D sensors like gyroscopes and accelerometers. The central aim of their paper was to gauge the efficacy of CNNs in discerning falls from non-fall events. By harnessing data from gyroscope sensors, the study accentuated the importance of angular velocity measurements in fall detection, owing to their capability to encapsulate rapid directional changes during falls.

Employing a CNN architecture optimized for 1D convolution and capable of handling multi-channel input, the researchers trained the model using both gyroscope and accelerometer data, each providing three channels of input corresponding to the three spatial dimensions (Ordóñez & Roggen, 2016). This approach allowed for a more comprehensive analysis of motion patterns indicative of falls. The effort succeeded, with the CNN model attaining a 91.9% F1-score, showcasing its accuracy in fall detection. However, the lack of Activities of Daily Living (ADLs) in the training phase may affect the score's representativeness. This precision outstripped other traditional methodologies, thus spotlighting the promise held by CNN architectures in this domain. Moreover, the paper shed light on the inherent advantages of CNN architectures, particularly their ability to automatically learn hierarchical features from raw sensor data,

obviating the need for manual feature extraction. While CNNs are helpful in extracting features from data and can process temporal information well, they also have a bias towards processing local points in data and learning those patterns rather than understanding the global context of data.

This CNN approach showcases the fusion of modern machine learning techniques like CNN with precise sensor data for enhanced fall detection. It reverberates a broader narrative in Human Activity Recognition (HAR) and Fall Detection Systems (FDS), nudging open avenues for further exploration on the amalgamation of CNN algorithms and time-series sensor data to cultivate robust, real-time monitoring systems aimed at safeguarding the well-being of vulnerable populations.

CNNs and LSTMs both have their strengths and weaknesses. CNNs are often utilized in scenarios where feature extraction from data is challenging, real-time processing is crucial, and multi-channel data is involved. In situations such as fall detection, where the length of datasets varies and retaining information while learning from time-based data is necessary, LSTMs are the ideal choice. While LSTM is optimal for fall detection, we find in research (Ordóñez & Roggen, 2016) that combining both LSTM and CNN networks to create architecture that takes advantage of their strengths and can process acceleration data effectively.

Introduced in the reference (Ordóñez & Roggen, 2016), DeepConvLSTM presents a pioneering approach towards Human Activity Recognition (HAR), epitomizing the synergy between Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) layers. This fusion manifests a robust framework for nuanced activity recognition tasks, particularly leveraging multimodal wearable sensor data.

At the outset, a layer of one-dimensional convolution scours through the sensor data. This

initial phase is instrumental in filtering anomalies or outliers while concurrently encapsulating crucial temporal information. The essence of this convolutional layer lies in its ability to automate the feature extraction process, reducing the dependency on manual, heuristic feature engineering, a common practice in traditional HAR approaches.

Following the CNN layer, the processed data is passed into an LSTM network. LSTMs analyze the temporal dynamics of the filtered sensor data, enabling precise activity recognition. The LSTM network predicts the user's activity, highlighting the subtle nuances that differentiate one activity from another.

The orchestrated combination of CNNs and LSTMs within the DeepConvLSTM framework amplifies the model's prowess in distinguishing between closely related activities. Upon evaluation, the DeepConvLSTM framework exhibited a remarkable performance, outperforming competing deep non-recurrent networks on a public activity recognition challenge dataset by an average of 4% and transcending some of the previously reported results by up to 9%. This notable performance underscores the model’s efficacy in leveraging spatial and temporal data for accurate activity recognition.

The novel architecture of DeepConvLSTM not only enhances the granularity of activity recognition but also broadens the horizons of HAR, making it more accessible. It obviates the need for specialized expertise in feature engineering, thereby paving the way for more intuitive, real-time activity recognition systems. The success of DeepConvLSTM, as delineated in (Ordóñez & Roggen, 2016), heralds a significant stride in harnessing deep learning for real-time, accurate, and robust HAR, especially in the realm of wearable technologies.

In the narrative of neural network architectures for Human Activity Recognition (HAR), the work of (Katrompas et al., 2022) presents another pivotal exploration. Their study unveils a

harmonized meld of Long Short-Term Memory (LSTM) networks and self-attention mechanisms tailored for discerning human activities over time. The architecture entails an LSTM network trailed by a self-attention layer, a confluence aiming to dissect temporal sequences inherent in human activity data. The self-attention mechanism is pivotal in assigning differential significance to various segments of the data sequences based on their temporal relationships, thereby amplifying segments that resonate strongly with the memory states of the LSTM layers.

The ability to preserve the order in time-series data, especially in the context of HAR, is essential. The self-attention mechanism honors this temporal order, attending to different segments of the sequence based on their temporal relationships. This adherence to temporal order enhances the LSTM's prowess in capturing long-term dependencies, a crucial facet for accurate human activity recognition.

A. Katrompas et al. also used a transformer model with stacked encoders, a method revered for its adeptness in handling sequential data. The evaluation uses datasets like MobiAct and Carbon Monoxide, wherein acceleration values serve as unique sequential identifications, defining the sequential order of the data. This setup orchestrates an evaluative framework, enabling a comparative performance analysis of the LSTM-self-attention ensemble against the transformer model in HAR.

Katrompas performed a comparative analysis between LSTM with self-attention and transformer mode, and the results articulated by the study highlight that the LSTM with self-attention architecture accentuates performance in human activity recognition tasks, particularly in fall detection and other classification tasks. This revelation underscores the indispensable role of melding recurrence (via LSTM) with self-attention in navigating the spatial and temporal intricacies embedded in human activity data, thereby potentiating enhanced classification

accuracy.

In our research, we are inspired to adopt the transformer model as the baseline for dissecting the effects of Time2Vec. The comparative insights harvested from A. Katrompas et al.'s endeavor serve as a valuable lens, illuminating the potential merits and challenges of diverse architectures in human activity recognition. This enriched narrative bolsters our comprehension and nurtures a more nuanced exploration of the architectural dynamics between different neural network architectures and time-series data, thereby fostering a more profound understanding of human activity recognition.

In the domain of real-time health monitoring, the work of (Jiang et al., 2022) stands as a notable contribution, particularly in the context of forecasting exercise-induced fatigue using wearable sensors. Their framework hinges on a Transformer model, embodying the generator, which adheres to the conventional encoder-decoder Transformer architecture. A pivotal aspect of this architecture is the utilization of Time2Vec for positional encoding within the encoder. Time2Vec morphs time into a vector representation by employing sine and linear activation functions, generating linear and periodic time features. These features are fed as additional inputs into the encoder, enriching the temporal comprehension of the model.

The generator is structured with three identical encoder layers, each comprising a spatiotemporal self-attention sub-layer and a fully connected feed-forward sub-layer, interspersed with residual connections and normalization layers to ensure stable and effective learning. This design is instrumental in capturing and scrutinizing the temporal dynamics inherent in wearable sensor data, laying a solid foundation for precise real-time forecasting of exercise-induced fatigue.

Furthermore, an auxiliary critic network is integrated into the framework, serving as an

evaluative mechanism for the forecasts generated by the transformer. This critic network quantifies the loss between the genuine sensor signals and the signals propagated by the transformer, thereby providing a measure of accuracy and reliability for the forecasted fatigue levels. This evaluative facet is essential for ensuring the robustness and credibility of the fatigue forecasting model.

The findings from Jiang et al.'s study underscore the performance of their model in real-time fatigue recognition, showcasing an advancement over the state-of-the-art. The fusion of Time2Vec with the Transformer architecture, coupled with the evaluative prowess of the auxiliary critic network, manifests as a promising avenue for real-time fatigue forecasting. This endeavor illuminates the potential of melding advanced encoding schemes with transformer architectures for bolstering time-series analysis, particularly in real-time health monitoring and forecasting scenarios, thus contributing significantly to the broader discourse on enhancing health monitoring systems through advanced machine learning methodologies.

(Zhang et al., 2023) pioneered a framework to adeptly manage the irregularities in multimodal Electronic Health Records (EHRs) from Intensive Care Units (ICUs). Their work spotlighted the complexities of irregular time series and clinical note sequences. A key component of their approach is utilizing a discretized multi-time attention (mTAND) module alongside Time2Vec.

Time2Vec is employed to transform each value in a continuous list of time points into a vector, yielding a series of time embeddings. This method captures periodic and non-periodic time behaviors through sine and linear functions, providing a nuanced understanding of time progression within the data.

The mTAND module operates with multiple instances of Time2Vec to produce

interpolation embeddings. This is orchestrated through a time attention mechanism, akin to multi-head attention, simultaneously embedding all time points into different dimensional hidden spaces. This mechanism captures various characteristics of different time points concerning the overall time information in different time subspaces.

Their methods consistently outperformed state-of-the-art baselines across two medical prediction tasks. They reported relative improvements of 6.5%, 3.6%, and 4.3% in F1 scores for time series, clinical notes, and multimodal fusion scenarios, respectively. These findings underscore the effectiveness of their methods and the critical importance of addressing irregularities in multimodal EHRs for advancing medical predictions.

In the study by (Wu et al., 2020), the researchers aimed to leverage Transformer-based models for forecasting the Influenza-Like Illness (ILI) rate using historical ILI data from the CDC. They employed a sliding window methodology to create supervised learning samples, facilitating their models' effective training. The performance of the models was evaluated using Root-mean-square errors (RMSE) to compare the actual and forecasted ILI rates.

A significant aspect of the study was the comparative analysis between the Transformer model and several benchmark models, including the traditional ARIMA, LSTM, and Seq2Seq models. The ARIMA model, known for its efficiency in time series forecasting; the LSTM model, recognized for handling sequence prediction challenges; and the Seq2Seq model, a deep learning paradigm with an encoder-decoder architecture, were explored to provide a comprehensive understanding of the Transformer model's performance in this domain.

The results revealed that the Transformer model outperformed the benchmark models in forecasting the ILI rate, showcasing its potential as a robust model for time series forecasting in the medical domain. This finding underscores the model's capacity to handle sequential data

effectively, providing insights into its application beyond natural language processing tasks into areas like healthcare analytics.

The literature we have reviewed reveals a broadly favorable trend in employing Time2Vec for embedding time as a vector. Utilizing this algorithm enhances the readability and scale-invariance of features, thereby bolstering the effectiveness of human activity recognition. Nonetheless, there are certain drawbacks to be noted, which were not evident in the experiments utilizing Time2Vec. It is an algorithm that demands expertise for effective implementation and entails a higher computational resource requirement than a standard transformer.

Building upon these findings, the subsequent sections will delve into the utilization of Time2Vec within the Human Activity Recognition (HAR) context, discussing the models and methodologies employed and proposing the integration of the transformer with Time2Vec to encode acceleration values, thus expanding the horizon of time series analysis and forecasting.

3. METHODOLOGIES

3.1. ARCHITECTURE

In recent years, the application of transformer models has revolutionized various fields, including natural language processing, image processing, language translation, and computer vision. Within fall detection, transformers have emerged as a powerful tool for analyzing fall patterns using diverse sensor data and enabling innovative approaches to address this critical challenge. The core of a transformer model is its fundamental components, which include self-attention mechanisms and feedforward layers. These components form the building blocks of the encoder and decoder, each playing a crucial role in processing and extracting meaningful information from the input data.

The self-attention mechanism is the cornerstone of a transformer's ability to capture intricate relationships within data. Its basic form consists of attention heads that operate in parallel, allowing the model to focus on different parts of the dataset simultaneously. Each attention head computes attention scores by first performing matrix multiplication between query (Q) and key (K) vectors. The product of the query and key is then scaled using the square root of the key's dimensionality (d_k), and attention scores for query-key pairs are obtained by applying a SoftMax function. These attention scores are subsequently multiplied by the corresponding value (V) vectors, as seen in Equation 1. This process, executed for each head, enables the model to capture diverse dependencies within the data. Attention heads' outputs are then concatenated and linearized, as shown in Equation 3, to create a comprehensive representation of the data. This multi-headed self-attention mechanism allows transformers to capture local and global patterns, facilitating the modeling of intricate relationships within the input data.

$$Attention(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad [1]$$

Following the multi-head self-attention layers, the transformer employs an add and normalize layer in which the output from the multi-head self-attention sub-layer is added elementwise to the original input of the attention layer to preserve information. Once adding occurs, we use layer normalization on the sum(x) by first computing the mean (μ) and standard deviation (σ), learning the scale (γ) and shift (β) parameters from data, adding a very small number to avoid dividing by zero (ϵ), and using these in Equation 2.

$$y = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} * \gamma + \beta \quad [2]$$

Feed Forward layers that follow add and normalize consist of a linear transformation followed by a Rectified Linear Unit (ReLU) activation and another linear transformation. The purpose of this feedforward layer is to enable the model to learn complex non-linear relationships within the data and capture information about the relative positioning of different data points. This is particularly relevant in fall detection, where understanding data's temporal and spatial aspects is crucial.

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_x)W^O \quad [3]$$
$$where\ head_x = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

Where the projections are matrices $W_i^Q \in \mathbb{R}^{dmodel*d_k}, W_i^K \in \mathbb{R}^{dmodel*d_k}, W_i^V \in \mathbb{R}^{dmodel*d_v}$

Combining the multi-headed self-attention and position-wise feedforward layers forms the encoder's core in a transformer model. The encoder is responsible for processing and encoding the input data, creating rich representations that capture essential features and dependencies.

The decoder typically begins with a masked multi-headed attention layer as its first sub-layer, allowing it to attend to different parts of the input sequence while preventing information

leakage from future data points. A residual connection from the encoder to the second multi-head attention layer enables the decoder to leverage the encoded features and attention mechanisms to generate meaningful outputs. The decoder then concludes with a position-wise feedforward layer to produce the final predictions or classifications. The entire architecture is shown in Figure 1.

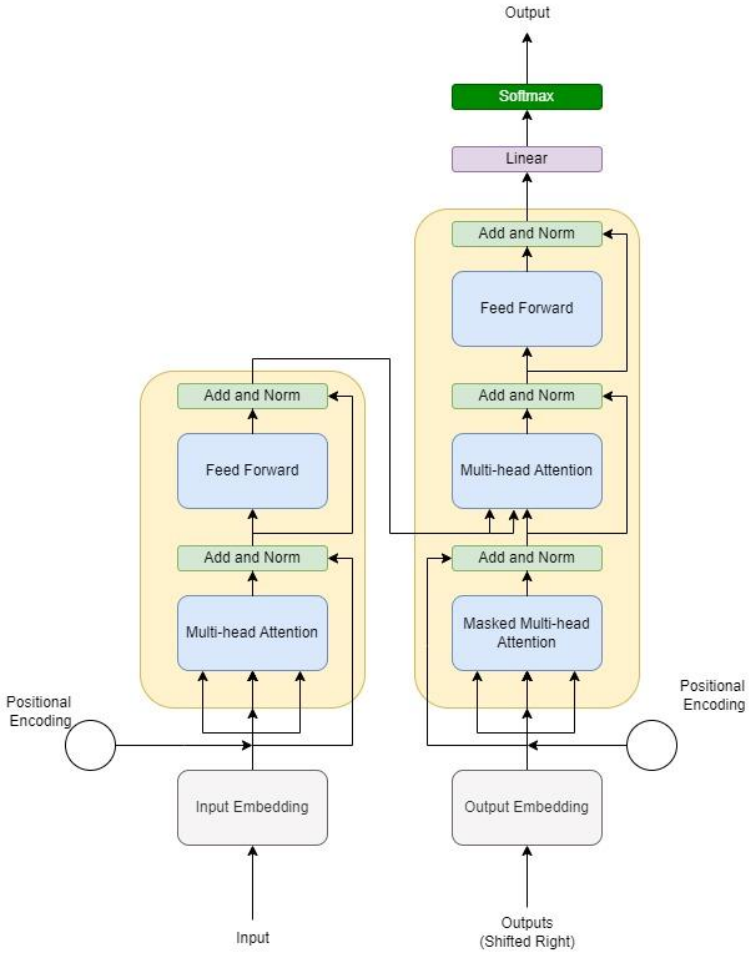

Figure 1. Transformer Architecture (Vaswani et al., 2017)

Our experiments only use stacked encoders to classify if a particular data point or group of data points is a “fall” or “non-fall” activity. This is because encoders are used to encode data and pass it to a decoder, where it is re-sequenced and then can be used to predict a possible outcome after the sequence. However, encoders can still provide probabilities for a classification when a SoftMax or Linear layer is applied directly to the concatenation after stacked encoders. Since stacked encoders are still classified as using a transformer model, we use the terms

interchangeably.

The focus of our research is to see how much Time2Vec affected the performance of a standard transformer while using it for fall detection. Time2Vec is a vectorization layer that uses periodic and linear functions to embed time-related features such as UNIX timestamps and DateTime data. Once a periodic function (ℱ) is chosen, we use two learnable parameters (ω_i , φ_i) to capture information from the time-related features over the vector of size k + 1 using Equation 4.

$$t2v(\tau)[i] = \begin{cases} \omega_i\tau + \varphi_i\tau, & if\ i = 0 \\ \mathcal{F}(\omega_i\tau + \varphi_i\tau), & if\ 1 \leq i \leq k \end{cases} \quad [4]$$

We use Time2Vec to produce more readable and interpretable vectors by the transformer, which uses sequential information to support its attention mechanism. It is also invariant to scaling to different units, and because of the trainable parameters, it can correct embeddings when more loss occurs due to the wrong embedding results. Using time2Vec vectorization layers, we were able to avoid feature engineering acceleration and gyroscope features that would otherwise create a heavier pre-processing stage. While acceleration and gyroscope data do not necessarily need to be embedded, these two types of sensors would not be enough to establish a robust model that can be generalized to any device.

The vectors generated by Time2Vec are subsequently fed into the stacked encoders for examination and analysis. As illustrated by Figure 2, it is evident that the datapoint marked as a fall (Label 1) is more pronounced compared to the vector denoted as an ADL (Lable 0). This

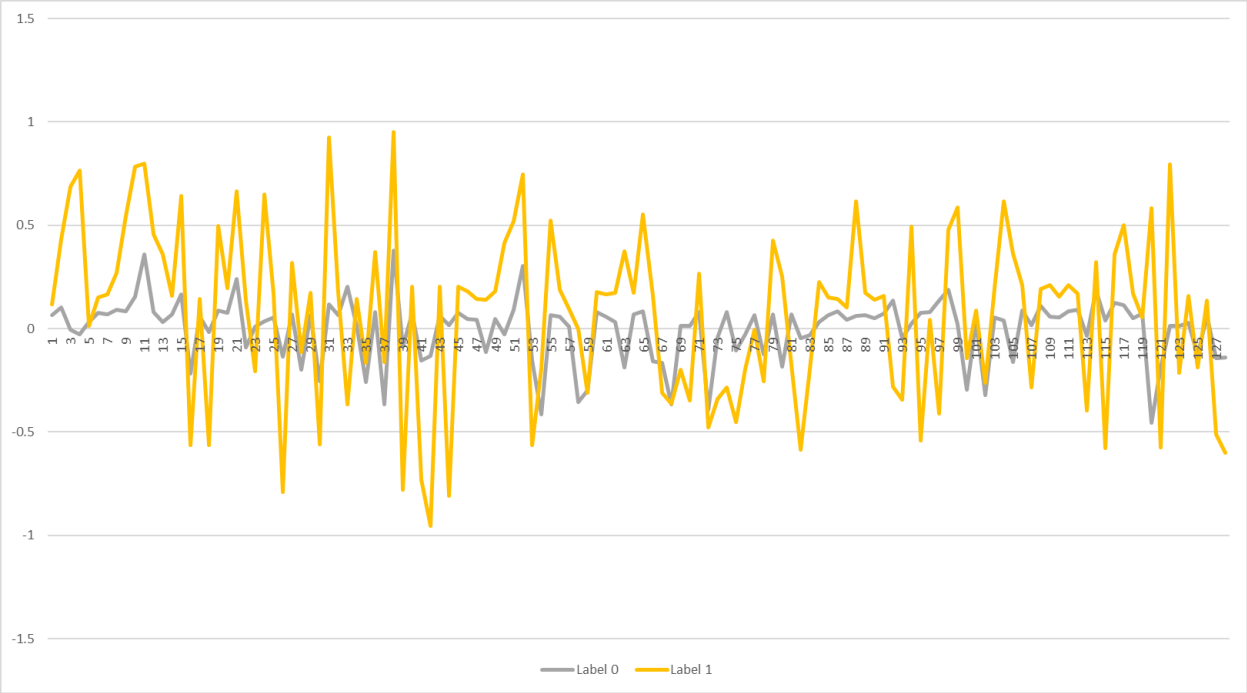

Figure 2. Vectorization Output of Fall and ADL Data Points using Time2Vec Layer

assists the stacked encoder model in better capturing and assimilating the data.

To ensure compatibility with the stacked encoders, we performed data trimming by removing surplus data points from both the beginning and end of the datasets. This trimming process ensured that the dataset's length became divisible by the chosen window size. The architecture adopted for this research capitalizes on the Time2Vec technique, employed to embed x-axis acceleration values effectively. Notably, Time2Vec can also be extended to embed y and z-axis acceleration values if the application warrants it. These resulting embeddings serve as the additional input to our transformer module.

At the core of our architecture lies the transformer module, which comprises several encoder layers in series. Each encoder layer processes the input data individually, with the final output of the encoders being concatenated together. This stacked mechanism empowers the model to capture increasingly intricate temporal patterns within the data. For the experiments, we used a varying number of encoders depending on the number of features, data points, and other

factors.

The final output of the transformer module, having concatenated the output of the stacked encoders, is then passed through a linear layer followed by a sigmoid activation layer. This architectural choice is deliberate, as it generates probabilities requisite for fall event classification. The sigmoid activation function ensures that the output values fall within the range [0, 1], with higher values indicating a greater likelihood of a fall event. The architecture is shown in Figure 3.

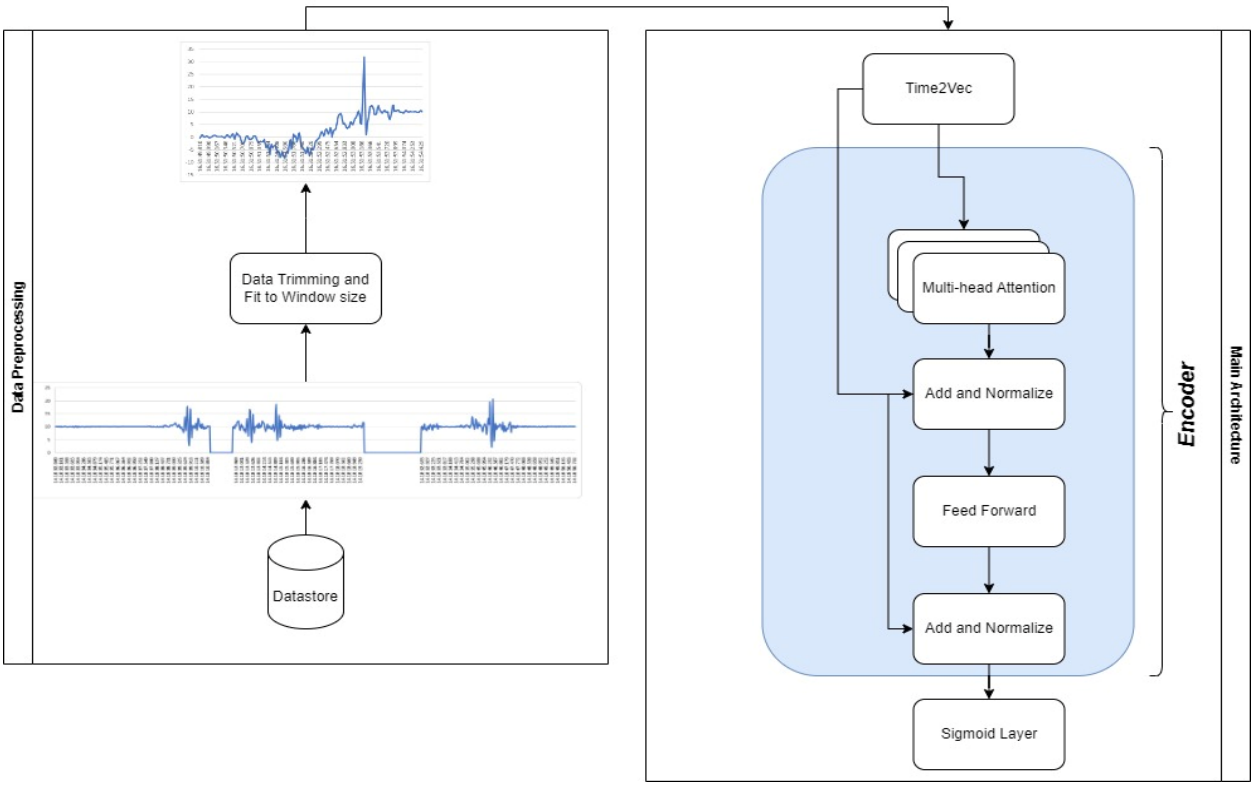

Figure 3. Architecture Diagram

In summary, our architecture, shown in Figure 4, seamlessly integrates Time2Vec embeddings, a stacked encoder module, and a sigmoid-based classification layer. This approach seamlessly merges transformer capabilities with specific adaptations for the nuances of fall detection tasks, resulting in a versatile and practical architecture. Furthermore, while our

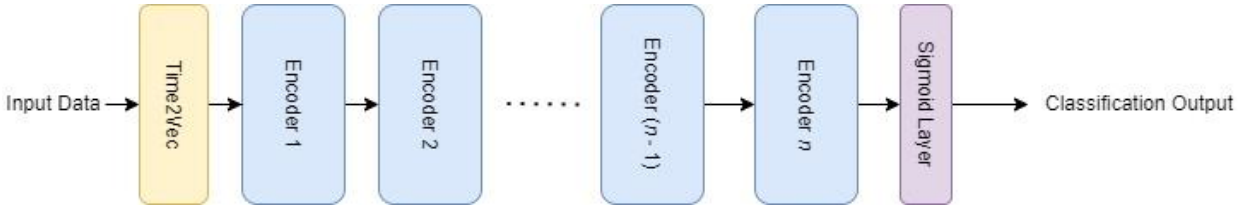

Figure 4. Simplified Model Showing Stacked Encoders

architecture bears similarities to that of (Katrompas et al., 2022), we have innovatively incorporated Time2Vec layers and expanded its capacity to process gyroscope data, further enhancing its suitability for fall detection applications.

3.2. EVALUATION

To comprehensively evaluate the performance of our model and facilitate comparisons with existing research, we employed a set of five essential metrics along with a shuffled 60/20/20 validation technique. This technique utilizes 60% of the available dataset for training, 20% of the dataset for training episode validation, and 20% for final testing. These metrics offer a well-rounded perspective on architecture's effectiveness in fall detection.

A. BINARY CROSS ENTROPY

Binary Cross Entropy is a fundamental metric for determining the amount of loss. They provide a quick and straightforward means of comparing our model's performance with baseline models. However, they alone may not offer a nuanced view of the model's capabilities. We describe BCE as the sum of the predicted label's (yi) probability p(yi) of being true positive or true negative vs the total number of predicted labels. We refer to Equation 5 to calculate BCE loss.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log\big(p(y_i)\big) + (1 - y_i) \cdot log(1 - p(y_i)) \quad [5]$$

B. ACCURACY

Accuracy is described as the total number of positive cases that are labeled as positive cases by the architecture, known as true positives (TP), and the negative cases that are labeled as

true negatives (TN) by the architecture divided by the total number of predictions which also include false positives and false negatives. The mathematical equation is shown in Equation 6.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad [6]$$

C. PRECISION AND RECALL

Precision and Recall are crucial metrics for understanding the model's classification behavior. They shed light on the classifications being made and, more importantly, which classifications are correctly identified. These metrics are particularly valuable in fall detection scenarios where distinguishing between false alarms and genuine fall events is essential. Precision, as described in Equation 7, is the true positive rate at which the model correctly identifies positive cases out of all the predicted positive cases. In other words, a higher precision means fewer false alarms caused by the model. Recall, described by Equation 8, measures the rate at which positive cases are correctly identified out of all actual positive cases. This rate is crucial because it shows the number of true positive falls missed by the model, which can cause a fall to go unnoticed.

$$Precision\ (P) = \frac{TP}{TP+FP} \quad [7]$$

$$Recall\ (R) = \frac{TP}{TP+FN} \quad [8]$$

D. F1 SCORE

Recognizing the inherent class imbalance in fall detection datasets, we utilized the F1 Score, shown in Equation 9, to evaluate the binary model independently. The F1 Score balances Precision and Recall, making it especially relevant for imbalanced datasets. It provides a comprehensive measure of the model's performance, accounting for false positives and false negatives. We describe the F1 Score (F1) as the harmonic mean between the precision and recall scores.

$$F1 = 2 \times \frac{(P \times R)}{(P+R)} \quad [9]$$

By incorporating this diverse set of metrics, we ensured a thorough assessment of our architecture's capabilities. Collectively, these metrics offer insights into the model's classification accuracy, ability to identify fall events correctly, and robustness in handling imbalanced data distributions. This multifaceted evaluation approach allows us to gauge architecture's effectiveness comprehensively and make meaningful comparisons with other studies in the field.

4. EXPERIMENTS

4.1. DATA COLLECTION

To facilitate proper experimentation, we selected datasets from various sources and mediums based on several criteria. These included the types of sensors and sensor placement, available features, data volume, the variety of activities of daily living (ADLs) and falls, and the number of available trials on ADLs and falls.

One dataset we utilized was the MobiAct Second Release (Vavoulas et al., 2016), consisting of 3,200 trials collected using a Samsung S3 device with an LSM330DLC module capturing acceleration and gyroscope data. The device was placed in the subject's pocket, allowing for the simulation of falls using a smartphone's sensors positioned near the waistline. This dataset contains 12 unique ADLs and four different fall types, gathered from 66 subjects ranging from 20 to 50 years old with varying heights, weights, and genders. Due to the substantial size of the MobiAct dataset, we opted to work with a subset comprising four falls and four ADLs, which contained 767 total fall trials and 1095 total ADL trials. We list the subsets used and their abbreviations in Table 1 below.

Table 1. MobiAct Subsets Used (Vavoulas et al., 2016)

Abbreviation	Activity Type	Subset Description
JUM	ADL	Jumping
JOG	ADL	Jogging
SDL	Fall	Sideways Fall while bending knees
STU	ADL	Going Up the Stairs
STN	ADL	Going Down the Stairs
BSC	Fall	Falling backwards
FOL	Fall	Falling forwards, using hands to dampen fall
FKL	Fall	Falling forwards, first impact on knees

SisFall (Sucerquia et al., 2017) was another dataset under consideration, even though it is smaller than MobiAct. SisFall collected data from 38 participants, including 15 individuals over the age of 60. It comprises 19 ADLs and 15 fall types, with over 4,000 independent trials. The data was recorded from various sensors connected to a microcontroller, focusing on acceleration features from the MMA8451Q sensor and gyroscope features from the ITG3200 sensor.

For experiments with smaller datasets, we turned to UniMiB SHAR (Micucci et al., 2017). This dataset provided an opportunity to explore the impact of the Time2Vec layer on a transformer with a smaller dataset. The dataset featured nine different ADLs and eight unique fall types, and it was recorded using a Samsung Galaxy Nexus i9250 equipped with a Bosh BMA220 triaxial acceleration sensor. The dataset's participants are male and female, ranging from 19 to 75 years of age, are 1.49 meters to 1.83 meters, and weigh between 42 kilograms and 102 kilograms.

To check the performance of the Transformer + Time2Vec model on a smartwatch, we also ran experiments using data collected using a Huawei Watch 2 watch sampled at 32 Hz. Participants ranged from 21 to 35 years of age and weighed between 100 and 150 pounds. The dataset contains five unique falls – front, back, right, left, and a rotating fall. The participants were also requested to perform six unique ADLs. The data was trimmed to eliminate unusable or resting data points before and after each activity trial.

Another watch dataset collected using MSBAND watch is also used for our experiments. The dataset contains four unique ADLs and four different types of falls. The subjects ranged from 21 to 55 years of age and were between 150 and 200 centimeters tall. Each participant was asked to wear the watch on their left hand while performing the ADLs and the falls. Data was collected at a sampling frequency of 31.25 Hz. Table 2 summarizes the data set details.

Table 2. Dataset Statistics

Dataset Name	**Num. of ADLS type**	**Num. of Fall type**	**Num. of Subjects**	**Sampling Frequency**	**Types of Devices**	**Device Position**	**Environment**
MobiAct v2	4	4	66	100Hz	Smartphone	Left/Right Pants Pocket	Lab
SisFall	19	15	38	200Hz	Smartphone	Center Waist	Lab
UniMiB SHAR	9	8	30	50 Hz	Smartphone	Left/Right Pants Pocket	Lab
MSBand (Smartfall)	4	4	7	32 Hz	Smartwatch – MSBand	Left Wrist	Lab
Huawei (Smartfall)	6	5	12	32 Hz	Smartwatch – Huawei Watch 2	Left wrist	Lab

4.2. DATA PREPROCESSING

Data must be split into training, testing, and validation sequences for all experiments. The transformer can only read a single .csv file, so most datasets must be combined into a single sheet. To randomize and combine the dataset into one .csv file, we used a process that read all the .csv files from a directory into a list of data frames, shuffled the list, and concatenated the data frames together. Once combined, we used 60% training, 20% validation, and 20% testing split after randomizing the dataset. This approach offered a good balance between training the transformer, using unseen data to test the transformer, and correcting and evaluating the model at the end of each epoch. Figure 5 shows the process of combining and splitting the dataset into appropriate sets for the model. Once the data was divided into training test and validation subsets, we had to trim from the beginning and end of the set, shown in Figure 6, so that the transformer was trained based on stagnant windows of a set size. Importantly, no features were extracted or created during preprocessing since fall detection requires real-time processing of raw data and optimizing detection.

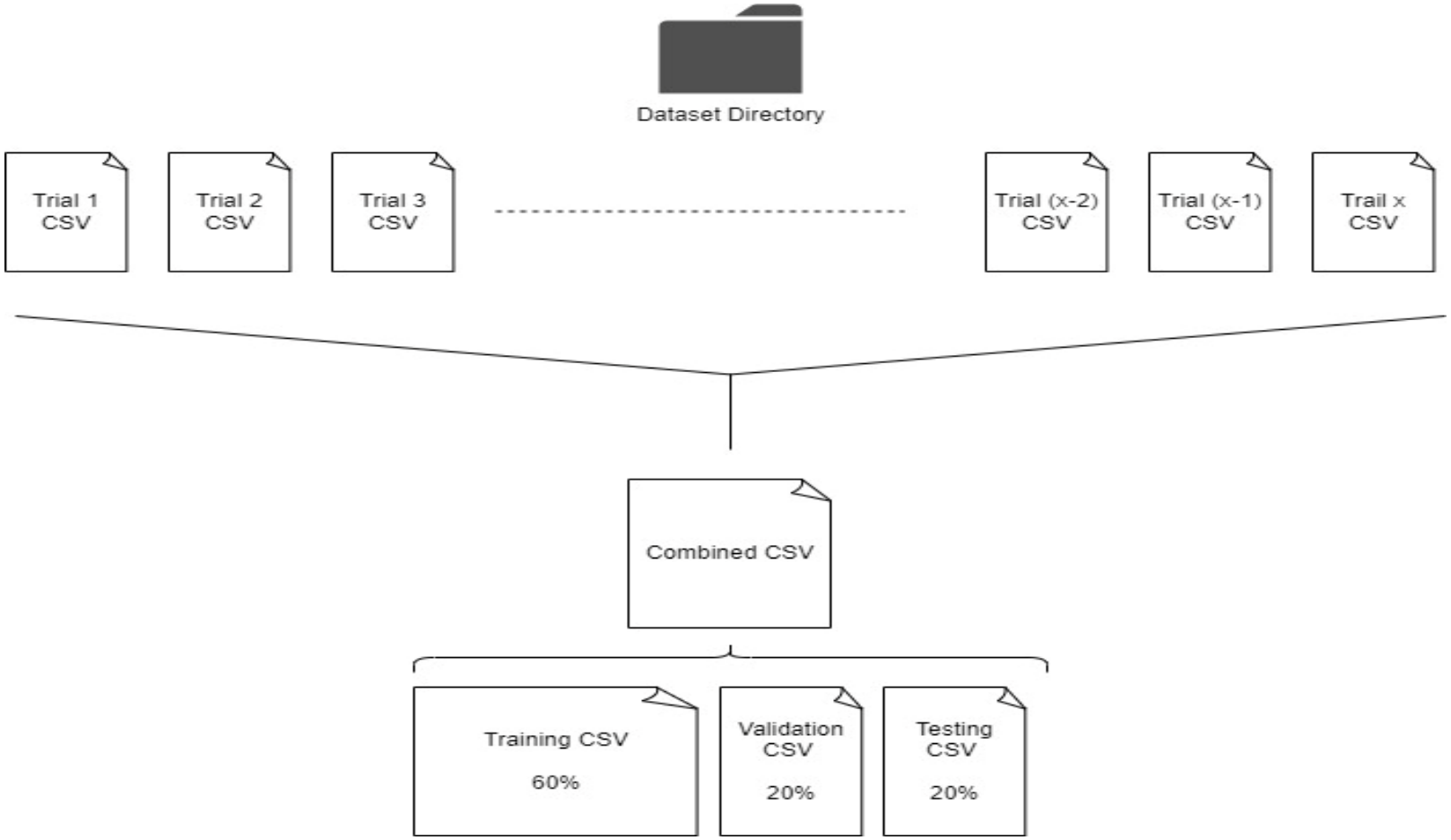

Figure 5. Data preprocessing from directory to training, validation, and testing sets

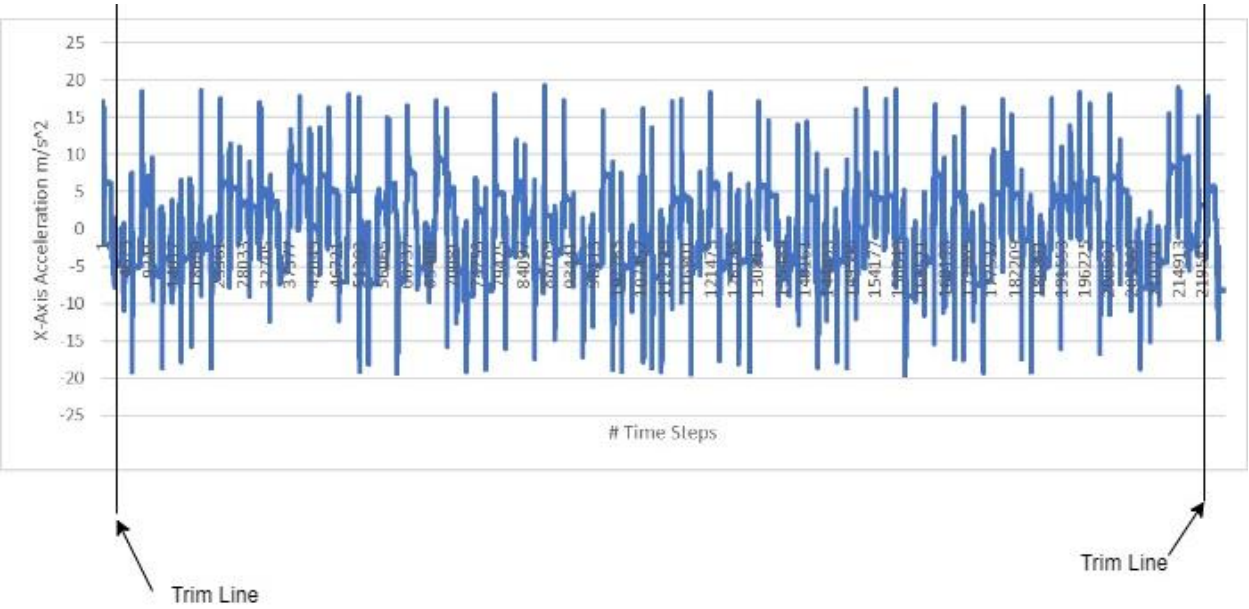

Figure 6. Data Trimming Process

4.3. HYPOTHESIS

Our primary objective is to use Time2Vec as a vectorizing input layer to encoders in series to show that vectorizing and expanding on the given acceleration data can improve detection accuracy. Time2Vec allows the encoders to evaluate sequence-dependent features and provides a way to let the model reliably embed features that are necessary to capture sequences within the data for us. By embedding axis acceleration values, we can expand the values into vectors, which the encoder will read using attention to extract meaningful information.

Our secondary objective is to explore different datasets to understand how different sets of features and data volumes impact the stacked encoders' performance. This involves considering the inclusion of gyroscope data collected from different body regions for improved motion understanding.

4.4. EXPERIMENTATION APPROACH

Experiment 1 – Baseline versus Time2Vec: To validate the effectiveness of Time2Vec, we used a baseline encoder model and tested that model using a MobiAct subset of four ADLs

and four falls. The baseline and Time2Vec models were tested with the same hyperparameters, except for Time2Vec dimensions and vectorized features, across eight data subsets. We tested each model twice with different number of encoders on eight data subsets. The hyperparameters are listed in Table 4, and their definitions are listed in Table 3.

Table 3. Parameter Definitions

Parameter	Definition
SEQLENGTH	Continuous window length that would be fed as input. For example, if the window size is 35, we will provide 35x3 acceleration values for 35 classifications representing a single sample of data.
NUM_ATTN_LAYERS	Number of stacked encoders within the module. The first encoder's output is input to the second encoder, and so on.
TIME_2_VECT_OUT_DIM	Time2Vec Output dimension. This hyperparameter will allow us to expand a scalar value to a vector by the specified amount. For example, given a single x-axis value and using 16 as the value of the parameter, it will expand that single value to 1x32
VECTOR_FEAT	The feature to expand. Each feature has an index depending on the data. For most datasets, x-axis values are on the 0 index, the y-axis on the one index, and the z-axis on the two index.
NUM_HEADS	Number of attention heads within each encoder. This will split the data from each window into the parameters specified number of chunks processed per pass. For example, if the parameter equals eight, an encoder can simultaneously process eight chunks.
HEAD_SIZE	Number of rows to be processed per head.
DROPOUT	Percentage of attention scores to drop out during the current training epoch. Attention scores are calculated based on the query key and value matrix multiplication. Some of these scores will then be randomly set to zero in the following training episode to help generalize the architecture.

Table 4. Experiment 1 Parameters

Parameters	Parameters for Baseline	Parameters for Time2Vec
SEQLENGTH	100	100
NUM_ATTN_LAYERS	2 (MobiAct testing with two encoders)	2 (MobiAct testing with two encoders)
	3 (MobiAct testing with three encoders)	3 (MobiAct testing with three encoders)
TIME_2_VECT_OUT_DIM	(N/A)	64
VECTOR_FEAT	(N/A)	0
NUM_HEADS	4	4
HEAD_SIZE	128	128
DROPOUT	.05	.05

Experiment 2 – Axis Vectorizations: As the x-axis values can be effectively vectorized using Time2Vec, we can extend our investigation to include the vectorization of the y and z-axes or even explore combinations of all three axes. This approach will assist us in identifying the most suitable axis for vectorization. Leveraging the fall subsets from MobiAct v2, we can conduct experiments involving the Time2Vec input layer and fine-tune it to enhance the transformer's performance. We use the parameters in Table 5 to run the experiment.

Table 5. Experiment 2 Training Parameters using MobiAct v2

Parameters	x-axis	y-axis	z-axis	x, y	x, z	y, z	x, y, z
SEQLENGTH	100	100	100	100	100	100	100
NUM_ATTN_LAYERS	3	3	3	3	3	3	3
TIME_2_VECT_OUT_DIM	64	64	64	64	64	64	64
VECTOR_FEAT	0	1	2	0,1	1,2	0,2	0,1,2
NUM_HEADS	4	4	4	4	4	4	4
DROPOUT	.25	.25	.25	.25	.25	.25	.25

As mentioned before, within the VECTOR_FEAT parameter definition, we can use this parameter to change which axis within the acceleration data is vectorized by the Time2Vec layer.

Experiment 3 – Acceleration Data versus Acceleration Plus Gyroscope Data: Further experimentation involved using acceleration and gyroscope features together, with the number of encoder layers maintained at three. Using the MobiAct v2 data set, we preprocessed the data with the gyroscope data included and used the parameters in Table 6 to test.

Table 6. Experiment 3 Training Parameters using MobiAct v2

Parameters for MobiAct v2	Acceleration Only	Acceleration + Gyroscope
SEQLENGTH	100	100
NUM_ATTN_LAYERS	3	3
TIME_2_VECT_OUT_DIM	64	64
VECTOR_FEAT	0	0
NUM_HEADS	4	4
DROPOUT	.25	.25

Experiment 4 – Model Performance using Other Datasets: To validate and corroborate the findings obtained in MobiAct v2, it became necessary to conduct testing on additional datasets. Utilizing larger datasets such as SisFall (Sucerquia et al., 2017) allows us to assess whether the model encounters challenges when dealing with more evenly distributed datasets or when handling diverse types of falls not present in MobiAct. Since SisFall represents falls and ADL events differently and boasts a substantial dataset size, replicating the experiments from MobiAct v2 using identical parameters is expected to yield valuable insights. We test using parameters comparable to those used in MobiAct, as listed in Table 7.

Table 7. Training Parameters for SisFall Testing

Parameters for SisFall	Baseline	Time2Vec
SEQLENGTH	100	100
NUM_ATTN_LAYERS	3	3
TIME_2_VECT_OUT_DIM	N/A	64
VECTOR_FEAT	N/A	0
NUM_HEADS	4	4
DROPOUT	.25	.25

Utilizing the UniMiB SHAR dataset (Micucci et al., 2017), which comprises a limited number of data points related to smartphone activities, we can gain insights into how Time2Vec performs under conditions with insufficient data to train and test transformer models. However, despite its limited size, we can still leverage this dataset to understand how Time2Vec behaves in such scenarios. The parameters used for UniMiB testing are in Table 8.

Table 8. Training Parameters for UniMiB

Parameters for UniMiB	Baseline	Time2Vec
SEQLENGTH	100	100
NUM_ATTN_LAYERS	3	3
TIME_2_VECT_OUT_DIM	N/A	64
VECTOR_FEAT	N/A	0
NUM_HEADS	4	4
DROPOUT	.05	.05

Experiment 5 – Model Performance using Smartwatch Data: During smartwatch training, we reduced the amount of DROPOUT to .05 since both Smartfall datasets were smaller than the other datasets acquired. All the hyperparameters for the acceleration testing are in Table 9.

Table 9. Training Parameters for SmartFall Datasets

Parameters for Smartwatch Sets	Baseline (Huawei)	Time2Vec (Huawei)	Baseline (MSBand)	Time2Vec (MSBand)
SEQLENGTH	100	100	100	100
NUM_ATTN_LAYERS	2	2	3	3
TIME_2_VECT_OUT_DIM	N/A	64	N/A	64
VECTOR_FEAT	N/A	0	N/A	0
NUM_HEADS	4	4	4	4
DROPOUT	.05	.05	.05	.05

Various other considerations, including using 1D convolutional layers, activation functions, optimizers, and feedforward dimensions, remained consistent across baseline and Time2Vec models. The final output layers followed the same structure in both architectures, consisting of a single Dense layer with a sigmoid activation function.

The architectures were built using TensorFlow (v2.12) as the neural network open-source library and Python (v3.11) as the scripting language. After building the architecture, we trained the transformer for 100 training episodes or epochs, but the script was allowed to stop early if the model did not see any improvement after 20 epochs. Once testing was done, we compared the results and analyzed them to conclude.

5. RESULTS

In this section, we present the results of all the experiments conducted to demonstrate the value of Time2Vec and the use of gyroscope data. We use MobiAct to illustrate the model's capabilities on a large dataset collected using a mobile phone. On the other end of the spectrum, we use Smartfall's datasets to demonstrate the model's capabilities on smaller-scale datasets from smartwatches.

5.1. BASELINE VERSUS TIME2VEC – ACCELERATION ONLY

In this initial experiment, we aimed to evaluate the comparative performance of the baseline model and the Time2Vec model using the MobiAct v2 datasets. It is essential to note that higher values for accuracy, precision, F1 Score, and Recall indicate superior performance, while lower loss values are considered desirable. The results of running MobiAct v2 with the baseline and Time2Vec models are summarized in Table 10. Table 10 reveals significant improvements in loss and recall metrics for the Time2Vec model compared to the baseline model. Notably, loss has decreased by approximately 10%, and recall has increased by over 15%.

Table 10. All MobiAct v2 Subsets – Time2Vec versus Baseline using two encoders

	Baseline Model (2 Encoders)					**Time2Vect (2 Encoders)**				
	Binary Loss	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Binary Loss**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
JOG	0.034	**0.988**	0.992	**0.996**	**0.994**	**0.029**	0.988	**0.994**	0.993	0.994
JUM	0.022	0.991	0.997	0.993	0.995	**0.015**	**0.993**	**0.998**	**0.995**	**0.996**
FOL	0.329	0.839	0.690	0.377	0.488	**0.169**	**0.932**	**0.825**	**0.842**	**0.834**
FKL	0.256	0.895	0.911	0.610	0.730	**0.133**	**0.952**	**0.906**	**0.888**	**0.897**
BSC	0.383	0.838	**0.970**	0.470	0.633	**0.187**	**0.926**	0.940	**0.803**	**0.866**
STU	0.213	0.929	0.962	**0.947**	0.954	**0.169**	**0.939**	**0.981**	0.941	**0.960**
STN	0.306	0.848	0.976	0.808	0.884	**0.243**	**0.883**	**0.990**	**0.846**	**0.912**
SDL	0.357	0.868	0.801	0.624	0.701	**0.169**	**0.945**	**0.897**	**0.877**	**0.887**
Average	0.238	0.899	0.912	0.728	0.797	**0.139**	**0.945**	**0.941**	**0.898**	**0.918**

Table 11. All MobiAct v2 Subsets – Time2Vec versus Baseline using three encoders

	Baseline Model (3 Encoders)					**Time2Vect (3 Encoders)**				
	Binary Loss	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Binary Loss**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
JOG	0.045	0.981	0.982	0.999	0.991	**0.026**	**0.990**	**0.996**	**0.994**	**0.995**
JUM	0.022	0.991	0.996	**0.995**	0.995	**0.013**	**0.994**	**0.999**	0.995	**0.997**
FOL	0.388	**0.836**	0.781	0.268	0.399	**0.162**	0.834	**0.893**	**0.813**	**0.851**
FKL	0.232	0.902	0.903	0.649	0.755	**0.141**	**0.948**	**0.948**	**0.821**	**0.880**
BSC	0.333	0.869	0.929	0.604	0.732	**0.196**	**0.922**	**0.944**	**0.785**	**0.857**
STU	0.195	0.925	0.950	0.954	0.952	**0.156**	**0.942**	**0.965**	**0.961**	**0.963**
STN	0.345	0.833	0.963	0.798	0.872	**0.161**	**0.930**	**0.982**	**0.920**	**0.950**
SDL	0.364	0.859	**0.894**	0.488	0.631	**0.172**	**0.942**	0.892	**0.872**	**0.882**
Average	0.240	0.899	0.925	0.719	0.791	**0.128**	**0.938**	**0.952**	**0.895**	**0.922**

However, it is worth mentioning that transitioning from a 2-encoder-layer model to one with three encoder layers had a minor impact on performance metrics, as seen in Table 11. Although recall demonstrated significant improvement, precision increased only by approximately 3%. This balance suggests that a 2-encoder-layer model performed optimally while achieving high precision, which is especially crucial in handling imbalanced datasets. Figures 7 and 8 show the loss trends over epoch training episodes for the baseline and Time2Vec models using two encoder layers.

It is important to note that our research focuses on fall detection and explores the recognition of Activities of Daily Living (ADLs). The results of including ADLs in our overall analysis demonstrate that Time2Vec does not hinder overall performance. Moreover, the model remains robust and unaffected when handling a combination of ADLs and falls according to average F1 Scores in Tables 9 and 10.

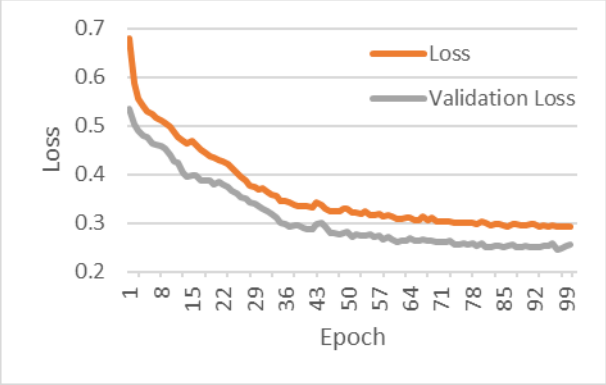

Figure 7. Loss over Epochs for Baseline

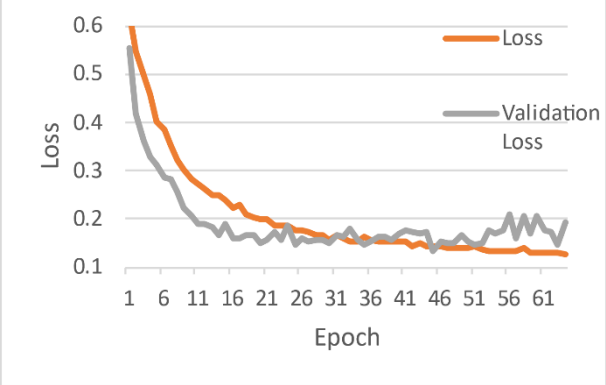

Figure 8. Loss over Epochs for Time2Vec

The outcomes observed in the MobiAct experiments demonstrate that Time2Vec contributes to enhanced fall detection performance and an overall improvement in model performance. Nevertheless, the evaluations conducted on the SisFall and UniMiB datasets highlight the model's distinct capabilities in diverse contexts.

5.2. AXIS VECTORIZATION

Time2Vec's approach to converting scalar values into vector representations through periodic wave transformations holds promise for fall detection applications. This vectorization process can be applied to any of the three-axis acceleration values (and gyroscope values if they are part of the features). This significantly enhances the model's ability to interpret input data.

Depending on the specific problem, we have the flexibility to vectorize the x-axis, y-axis, z-axis, or even combinations thereof. This allows us to gain a more comprehensive understanding of the dataset. We employed the VECTOR_FEAT index to vectorize a specific axis in our experiments. We conducted tests to determine the optimal single axis for vectorization or assess the potential benefits of combining multiple axes. Table 12 represents the overall testing average using MobiAct v2 Falls subsets.

Table 12. Results for Axis Vectorization on MobiAct v2 Fall only Subsets

MobiAct v2 Fall Subsets	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Time2Vec – x-axis	**.184**	**.935**	.938	**.829**	**.880**
Time2Vec – y-axis	.195	.928	.946	.795	.864
Time2Vec – z-axis	.321	.863	.951	.551	.698
Time2Vec – x-axis, y-axis	.206	.916	.967	.734	.834
Time2Vec – x-axis, z-axis	.264	.903	.921	.727	.813
Time2Vec – y-axis, z-axis	.192	.930	.974	.777	.865
Time2Vec – x, y and z	.218	.919	**.977**	.724	.832

The findings indicate that prioritizing the vectorization of the x-axis yields favorable results in terms of the F1 Score when considered within the model. Nevertheless, it is noteworthy that combining all three axes results in the highest precision while also leading to a slightly lower recall. For a better visualization of the results, we plotted the result using Figure 9.

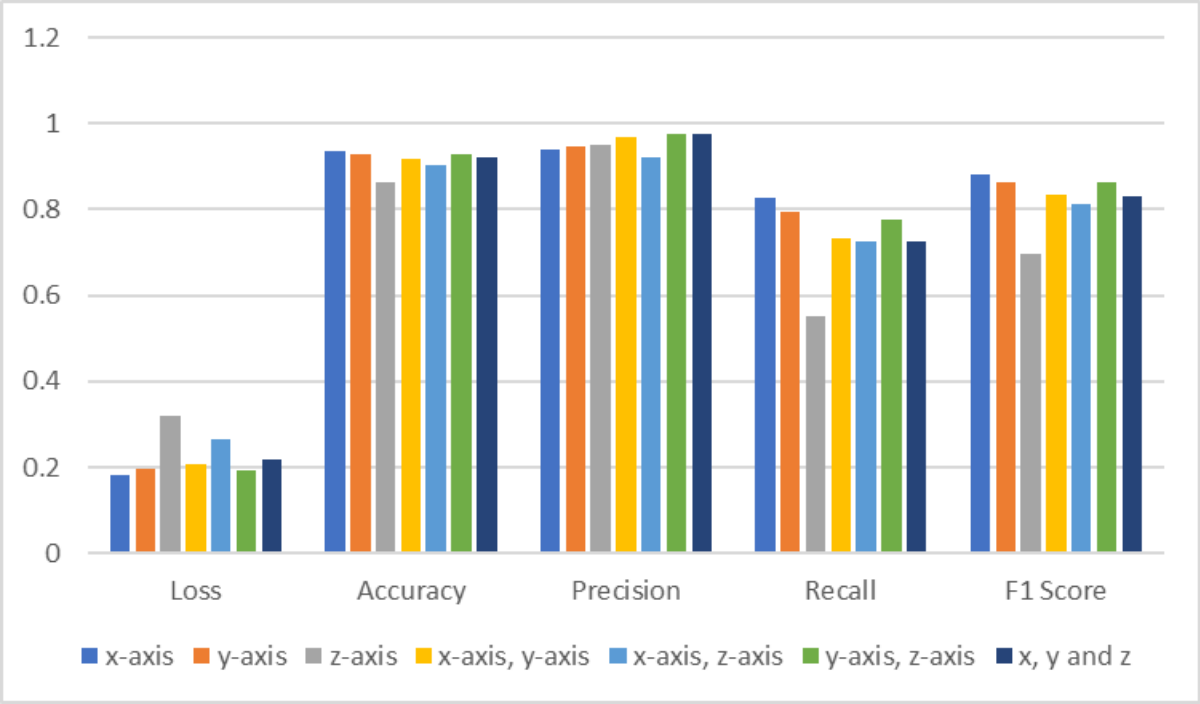

Figure 9. Summary of Vectorization Results using MobiAct v2

We have focused on vectorizing the x-axis for the subsequent experiments to ensure a well-balanced input data representation. For reference, Figures 10 through 13 depict the training results across epochs. These visual representations illustrate precision, recall, accuracy, and loss, providing a clear basis for direct comparisons between the vectorizations of different axes.

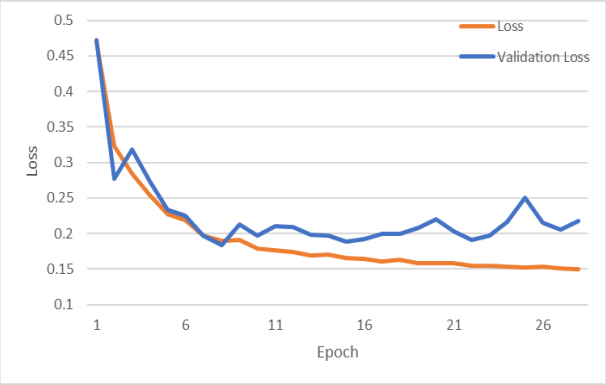

Figure 10. Loss over Epochs for X-Axis Vectorization

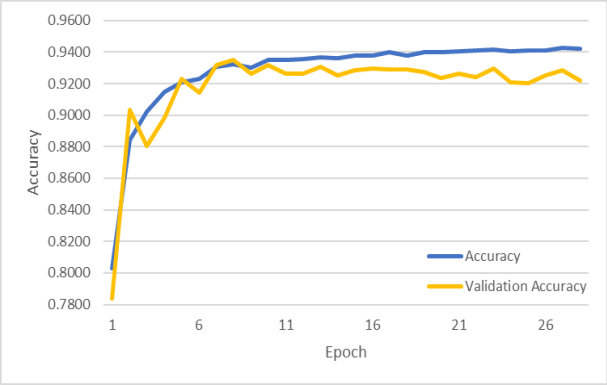

Figure 11. Accuracy over Epochs for X-Axis Vectorization

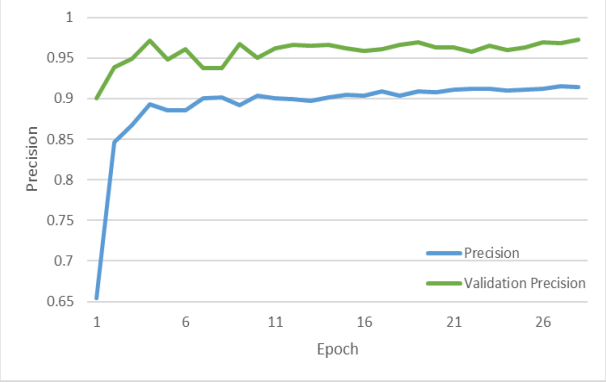

Figure 12. Precision over Epochs for X-Axis Vectorization

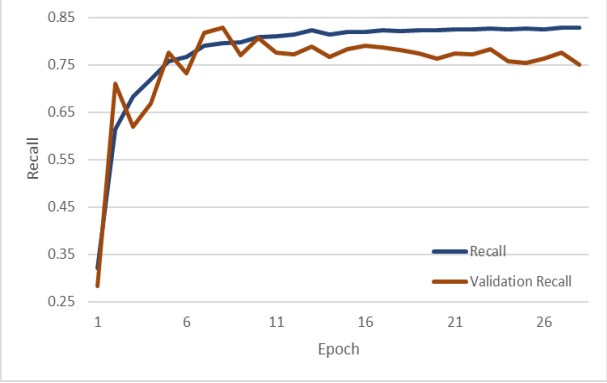

Figure 13. Recall over Epochs for X-Axis Vectorization

5.3. ACCELERATION VERSUS ACCELERATION PLUS GYROSCOPE

In this experiment, we sought to compare two subsets of the MobiAct dataset: one with acceleration data alone (previously used for baseline comparison) and another with the addition of gyroscope features. Performance metrics for these two subsets are presented in Table 13.

Table 13. Acceleration versus Acceleration plus Gyroscope using MobiAct Fall Subsets only

Average Fall Performance	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Time2Vec Acc. Only	.168	.911	**.919**	.823	.867
Time2Vec Acc + Gyro	**.153**	**.941**	.914	**.830**	**.870**

The results highlight that while adding gyroscope data led to a marginal performance enhancement, it simultaneously increased the model's complexity. This complexity is primarily due to the inclusion of three gyroscope features, which results in additional matrix

multiplications and neuronal connections. This prompts us to consider the trade-off between performance, generality, and efficiency. The data suggests that acceleration features, when subject to fine-tuning, often deliver strong results, emphasizing the potential of these features.

5.4. MODEL PERFORMANCE USING OTHER DATASETS

SisFall(Sucerquia et al., 2017), being a larger dataset and featuring a more balanced distribution compared to MobiAct v2 (Vavoulas et al., 2016), provides valuable insights into the model's adaptability within an atypical fall detection dataset. Fall detection datasets exhibit an imbalance due to the infrequent occurrence of falls in real-world scenarios.

On the other hand, UniMiB, characterized by its limited dataset size, offers a unique opportunity to assess the transformer's capacity to learn effectively when confronted with limited data. This analysis sheds light on the model's performance under data-scarce conditions. The results for both experiments are shown in Tables 14 and 15:

Table 14. Results for SisFall Testing

SisFall Average Performance	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Baseline – 3 Encoders	.504	.800	.768	**.556**	.645
Time2Vec – 3 Encoders	**.457**	**.811**	**.818**	.542	**.652**

Table 15. Results for UniMiB SHAR Testing

UniMiB Average Performance	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Baseline – 3 Encoders	.571	.701	.710	**.821**	.762
Time2Vec – 3 Encoders	**.501**	**.787**	**.836**	.787	**.811**

In our evaluation of both datasets, when comparing the results with Time2Vec to the baseline results without Time2Vec, we observe a modest but notable improvement in the F1-Score. This improvement is evident in both the SisFall dataset and the UniMiB dataset.

The enhancement in the F1-Score for both SisFall and UniMiB datasets primarily stems from increased precision measurements. This signifies that the model's ability to identify true positive cases or falls correctly has been notably strengthened with the incorporation of

Time2Vec.

5.5. MODEL PERFORMANCE USING SMARTWATCH DATA

The final experiment involved training and testing using the Huawei and MSBAND datasets from Smartfall, using the baseline and Time2Vec models with two encoder layers. The results of this experiment are summarized in Table 16.

Table 16. Overall Performance Time2Vec versus Baseline using Huawei Dataset

Smartfall Huawei	Loss	Accuracy	Precision	Recall	F1 Score
Baseline – 2 Encoders	.599	.667	**.868**	.406	.553
Time2Vec – 2 Encoders	**.568**	**.731**	.832	**.590**	**.690**

As observed earlier, the Time2Vec model led to significant improvements in loss and recall. Additionally, the model's accuracy increased by approximately 6%. While the overall performance in this experiment was not optimal, it is evident in both Tables 16 and 17 that Time2Vec equipped the model with a better understanding of its limited data, demonstrating its value as an input layer, especially in scenarios involving smaller datasets. Due to the limited data presented to the models, we can observe some overfitting in Figures 14 through 17, especially in the Time2Vec models.

Table 17. Overall Performance using MSBand

Smartfall MSBand	Loss	Accuracy	Precision	Recall	F1 Score
Baseline – 3 Encoders	.300	.872	**.705**	.185	.197
Time2Vec – 3 Encoders	**.281**	**.896**	.664	**.472**	**.365**

Figure 14. Loss over Epochs for Baseline testing using Huawei

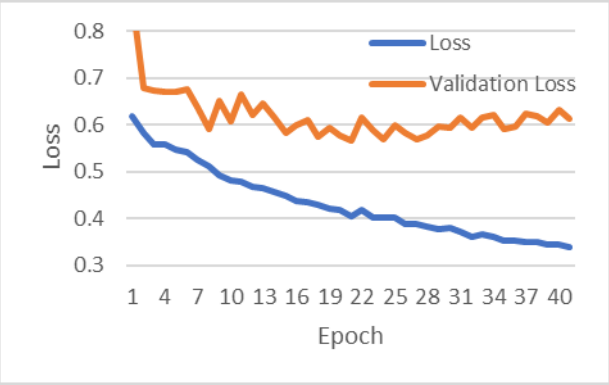

Figure 15. Loss over Epochs for Time2Vec testing using Huawei

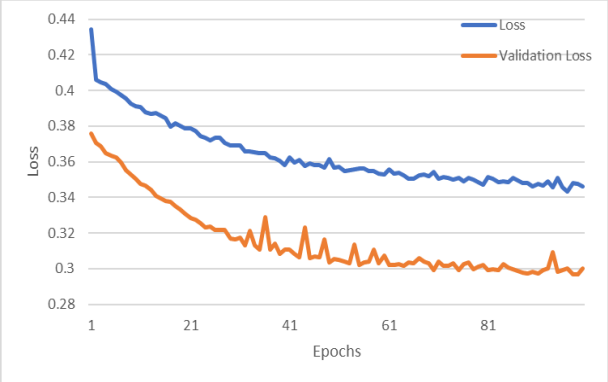

Figure 16. Loss over Epochs for Baseline using MSBand

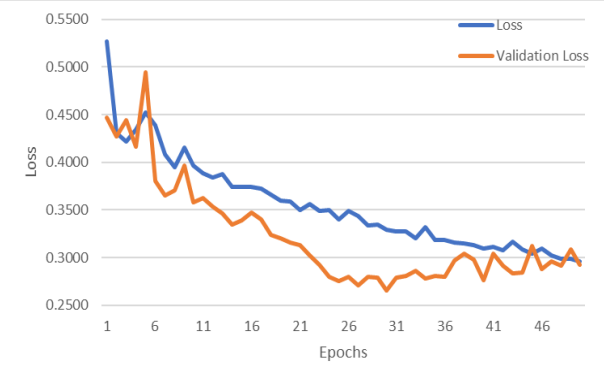

Figure 17. Loss over Epochs for Time2Vec Using MSBand

The trends observed in the Huawei dataset are like the ones in the MSBand dataset, which, being considerably smaller, is not well-suited for transformer models. Despite achieving notable enhancements in recall and loss reduction, it is worth noting that over half of the predictions made in this context were erroneously categorized as ADL cases. We summarize the smartwatch results in Figure 18.

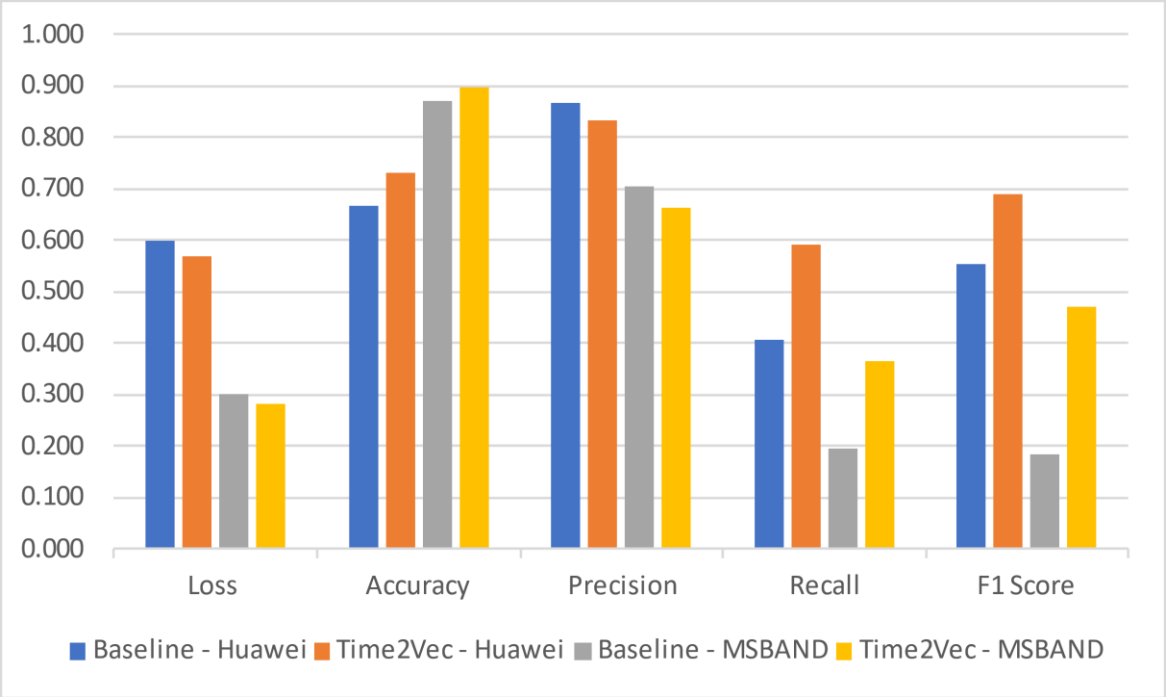

Figure 18. Summary of all Smartwatch Testing

6. CONCLUSION

The exploration of Time2Vec within this study under the context of fall detection has led to valuable insights. The study's primary objective was to explore Time2Vec's efficacy using stacked encoder models and comparing models trained with Tim2Vec to see how it improves the stacked encoder model's performance. The first step to gaining insight into this objective was to tune the model correctly to take full advantage of the available data while training the model.

Once we experimented with various combinations of axes as input for Time2Vec, we could derive the suitable Time2Vec model and the baseline model and compare the two. Settling on the x-axis for vectorization, based on the F1 scores, we were able to correctly predict more positive fall cases, achieve a low loss result, and show more than a 10% increase in recall over the previous fall detection transformer models.

Tests were conducted across a spectrum of datasets and different sets of features to deduce reliable outcomes. For instance, the SisFall dataset, comparable in size to MobiAct v2, was used to verify the impact of Time2Vec on fall detection accuracy. We see a marginal improvement using Time2Vec within the same models, about 1% to 5% improvement in all the metrics. Furthermore, UniMiB, a smaller dataset, also reflected improvement compared to the baseline encoder model of approximately 5% according to tested F1 scores. Depending on the situation, we can see that both precision and recall are affected by applying Time2Vec, but overall, we see an improvement in F1 Scores and a minimization of loss.

We anticipated that the model would be more effective when classifying falls within the MobiAct v2 subsets by adding gyroscope features to the dataset for the model to learn from angular motion. During testing, results indicated an increase in training times and an increase in performance. However, the insights also revealed that over-complicating the model is not

beneficial. Adding the gyroscope data improved results by 5% or less across most metrics while observing a decrease in recall. The additional features caused the model training times to increase by 5 seconds for each epoch episode. For reference, a model can be trained for anywhere between 20 to 100 epochs, so we added a maximum of 500 seconds by adding more features to analyze. We concluded that adding more features would not merit the extra computational complexity.

We have already seen results from a multitude of smartphone datasets mentioned above. However, ultimately, the model should run on a smartwatch. The datasets from Huawei and MSBand smartwatches are used to train watch models. These two datasets are much smaller than MobiActV2 and SisFall.

The limited data restricts the models' capabilities and shows the weakness of transformers: the quantity of data required to train these models appropriately. Using the datasets representing smartwatch devices, we tested the baseline against the Time2Vec model and gained a considerable advantage in recall measurements. Conclusively speaking, Time2Vec has also improved the performance of a watch-based mode.

The conclusion from these findings is the affirmation of Time2Vec's role in fall detection when synergized with stacked encoder transformers. The algorithm has significantly elevated the interpretability and enabled transformer models to accentuate crucial events like falls. However, it is imperative to acknowledge that the magnitude of improvements exhibited a variance across different mediums and datasets.

7. FUTURE WORKS

While we explored Time2Vec while classifying falls, we believe that further experiments and research can be done. One of the aspects mentioned in the Time2Vec literature is that the algorithm is model agnostic, which can enhance the performance of any architecture when applied correctly (Kazemi et al., 2019). The research was conducted primarily on transformer encoders in series because these models take advantage of abundant available data and the generally unique approach to learning self-attention. However, different architectures can take advantage of smaller datasets and window sizes because they perform analysis locally rather than simultaneously looking at the entire window. Time2Vec allows any model to read focus points and discern between fall and ADL events while still allowing them to analyze the data in much smaller window sizes, making the model more efficient.

The central aspect of the research was to study the use of Time2Vec with the transformer model to improve how the transformer reads the input data. However, modifying the transformer to use either a sliding window or a hybrid form of attention mechanisms like local analysis plus a sliding window (Beltagy et al., 2020) can be studied to determine the best architecture. This will help the transformer create relationships between windows to better analyze the input. We can vectorize continuous data frames instead of individual timesteps to study Time2Vec further. This advantage will allow the transformer to see a broader picture of the data.

Our last direction is the use of the model on smartwatch data. We have certainly improved the performance of the transformer trained on smartwatch datasets. However, the scarcity of this type of data is a disadvantage to the transformer model. The intention in the future is to collect more data of this type to get a model capable of effectively classifying a fall and to test the generated model using the SmartFall App for real-world tests.

REFERENCES

1. Centers for Disease Control and Prevention. (2023). Facts about falls. Retrieved from https://www.cdc.gov/falls/facts.html.

2. World Health Organization. (2008). WHO global report on falls prevention in older age. Retrieved from https://www.who.int/publications/i/item/9789241563536.

3. Liu, S. H., & Cheng, W. C. (2012). Fall detection with the support vector machine during scripted and continuous unscripted activities. *Sensors*, *12*(9), 12301-12316.

4. Apple. (2023). *Apple Watch Series 9*. Retrieved from https://www.apple.com/apple-watch-series-9/.

5. Google. (2022). *Google Pixel Watch 2: Turn insights into improvements*. Retrieved from https://store.google.com/product/google_pixel_watch?hl=en-US&pli=1.

6. RightMinder. (2017). *RightMinder is an Android Wear app for seniors*. Retrieved from https://mhealthspot.com/2017/03/rightminder-android-wear-app-seniors/.

7. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *ArXiv*. Retrieved from https://ar5iv.org/abs/1706.03762.

8. Kazemi, S. M., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., ... & Brubaker, M. (2019). Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321*.

9. Saoud, L. S., Al-Marzouqi, H., & Hussein, R. (2022). Household energy consumption prediction using the stationary wavelet transform and transformers. *IEEE Access*, *10*, 5171-5183.

10. Mauldin, T. R., Canby, M. E., Metsis, V., Ngu, A. H., & Rivera, C. C. (2018). SmartFall: A smartwatch-based fall detection system using deep learning. *Sensors*, *18*(10), 3363.

11. Jiang, Y., Malliaras, P., Chen, B., & Kulić, D. (2022). Real-time forecasting of exercise-induced fatigue from wearable sensors. *Computers in Biology and Medicine*, *148*, 105905.

12. Zhang, X., Li, S., Chen, Z., Yan, X., & Petzold, L. R. (2023, July). Improving medical predictions by irregular multimodal electronic health records modeling. In *International Conference on Machine Learning* (pp. 41300-41313). PMLR.

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

14. Casilari, E., Álvarez-Marco, M., & García-Lagos, F. (2020). A Study of the use of gyroscope measurements in wearable fall detection systems. *Symmetry*, *12*(4), 649.

15. Wu, N., Green, B., Ben, X., & O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*.

16. Katrompas, A., Ntakouris, T., & Metsis, V. (2022, June). Recurrence and self-attention vs the transformer for time-series classification: a comparative study. In *International Conference on Artificial Intelligence in Medicine* (pp. 99-109). Cham: Springer International Publishing.

17. Vavoulas, G., Chatzaki, C., Malliotakis, T., Pediaditis, M., & Tsiknakis, M. (2016, April). The mobiact dataset: Recognition of activities of daily living using smartphones. In *International Conference on Information and Communication Technologies for Ageing Well and e-Health* (Vol. 2, pp. 143-151). SciTePress.

18. Sucerquia, A., López, J. D., & Vargas-Bonilla, J. F. (2017). SisFall: A fall and movement dataset. *Sensors*, *17*(1), 198.

19. Micucci, D., Mobilio, M., & Napoletano, P. (2017). Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, *7*(10), 1101.

20. Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

21. Bucur, A. M., Cosma, A., Rosso, P., & Dinu, L. P. (2023, March). It's Just a Matter of Time: Detecting Depression with Time-Enriched Multimodal Transformers. In *European Conference on Information Retrieval* (pp. 200-215). Cham: Springer Nature Switzerland.