MDPI

Article

Experimental Study of LSTM and Transformer Models for Fall Detection on Smartwatches

Syed Tousiful Haque, Minakshi Debnath, Awatif Yasmin, Tarek Mahmud and Anne Hee Ngu *

Authors' address: Syed Tousiful Haque, bgu9@txstate.edu; Minakshi Debnath, stg60@txstate.edu; Awatif Yasmin, nuc4@txstate.edu; Tarek Mahmud, tarek_mahmud@txstate.edu; Anne Hee Ngu, angu@txstate.edu, Department of Computer Science, Texas State University, San Marcos, Texas, USA.

* Corresponding Author.

Abstract: Falls are the second leading cause of unintentional injury deaths worldwide. While numerous wearable fall detection devices incorporating AI models have been developed, none of them are used successfully in a fall detection application running on commodity-based smartwatches in real time. The system misses some falls and generates an annoying amount of false positives for practical use. We have investigated and experimented with an LSTM model for fall detection on a smartwatch. Even though the LSTM model has high accuracy during offline testing, the good performance of offline LSTM models cannot be translated to the equivalence of real-time performance. Transformers, on the other hand, can learn long-sequence data and patterns intrinsic to the data due to their self-attention mechanism. This paper compares three variants of LSTM and two variants of Transformer models for learning fall patterns. We trained all models using fall and activity data from three datasets, and the real-time testing of the model was performed using the SmartFall App. Our findings showed that in the offline training, the CNN-LSTM model was better than the Transformer model for all the datasets. However, the Transformer is a preferable choice for deployment in real-time fall detection applications.

Keywords: Fall detection, Deep learning, LSTM, Transformers, Wearables

1. Introduction

Falls are the second leading cause of unintentional injury deaths worldwide. Adults older than 60, suffer the greatest number of fatal falls [1]. The resultant inactivity caused by a fall in older adults often leads to social isolation and increased illnesses associated with inactivity including infections and deep vein thrombosis. Consequently, a large variety of wearable devices which incorporate fall detection systems have been developed [2–5].

One of the main sensors used in fall detection on a smartwatch is an accelerometer, which measures the acceleration of an object. Acceleration is the change in velocity over time, and velocity represents the rate at which an object changes its position. Acceleration data is commonly used in fall detection because a distinct change in acceleration happens when a fall occurs. The clustered spikes in Fig. 1 show a unique pattern in the acceleration data during the time when the fall occurs, which means that falls can be identified in acceleration data by that pattern. Previously, we have developed a watch-based SmartFall App using a basic Long Short-Term Memory neural network (LSTM), an recurrent neural network (RNN) with feedback connections, to detect falls based on the above pattern [6,7]. We have deployed this SmartFall system on a commodity-based Huawei smartwatch which has been trialed by nine senior participants [8]. Despite the system being welcomed by the participants in our trials, it still has several limitations, for example, a sudden hand or wrist movement from some ADLs (Activity of Daily Living) can interfere with the recognition of fall patterns which resulted in too many false positives for practical use.

Our existing smartwatch-based fall detection system based on the LSTM model is currently under-performing in real-world testing. As pointed out by [9], there are a few

Citation: Haque, S.; Debnath, M.; Yasmin, A.; Mahmud, T.; Ngu, A. Experimental Study of LSTM and Transformer Models for Fall Detection on Smartwatches. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

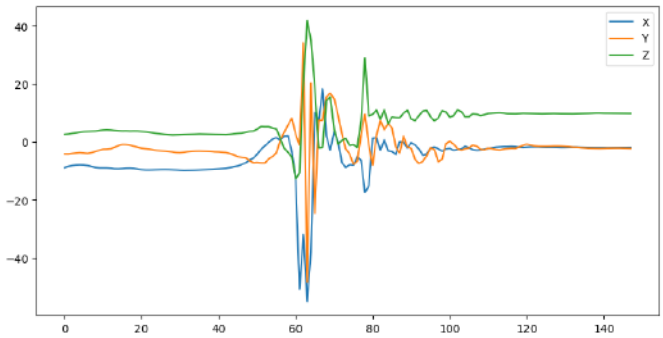

Figure 1. Acceleration from a fall.

inherent challenges in applying deep learning to wearable devices such as smartwatches. For example, the sensed data could be noisy due to lower precision sensor readings. The way data are assigned for training may be limiting as well. For example, the LSTM model is not efficient in processing a long sequence of streaming data. Therefore, input data must be segmented into windows. If the chosen window size is too small, important signals might fall outside the range; having the window size too large risks having to process useless input data that is not relevant to a fall (i.e. background noises). We want to investigate whether an alternative model called the Transformer model [10] can perform better than the LSTM fall detection model. One of the primary merits exhibited by the Transformer architecture in contrast to LSTM is its intrinsic ability to concurrently process data, a characteristic that effectively mitigates the challenges associated with the processing of long sequences. This feature significantly enhances the model's capacity to handle long input sequences efficiently. Moreover, the self-attention head in the transformer can selectively focus on different parts of sequential input data, and capture context information that is important for understanding the meaning of the whole sequence which could outperform the LSTM model.

We compared three variants of LSTM models with two variants of Transformer models across three fall datasets. The best LSTM and Transformer model trained offline using the SmartFallMM dataset (data collected in our lab with a smartwatch as one of the sensors) is then converted to TensorFlowLite model and deployed on SmartFall App. We are the first to conduct a real-time evaluation of a TensorflowLite fall detection model using a real-world SmartFall application. The main contributions of this paper are:

- Evaluate the offline trained best variant of Transformer and LSTM models in real-time on a fall detection application (SmartFall) running on a commodity-based smartwatch running on WearOS.
- Demonstrate that though CNN-LSTM performs better with F1-score of 86.7% in offline evaluation, the basic Transformer that has only 82.6% offline F1-score can maintain better performance in real-time testing making it a more suitable and dependable model for real-world deployment.
- Demonstrated that both variants of Transformers and LSTMs performed better as the size of the dataset increases.
- Demonstrated that the Transformer can maintain better recall and generate fewer false positives in real-time testing. This can be ascribed to its sophisticated capacity or self-attention to recognize and comprehend the underlying patterns in the data. .

The remainder of this paper is organized as follows. Related works to our research are described in Section 2. In Section 3, we discuss the architecture of the variants of LSTM and Transformer models. In Section 4, the methodology for the experiments is discussed. The computation model is presented in Section 5 while the experimental results are shown and compared in Section 6. In Section 7, we present our conclusion. Finally, in Section 8, we discuss future work.

2. Related Work

The early works in fall detection technologies were concentrated on specially built hardware that a person could wear or install in a specific facility [11]. The fall detection devices in general try to detect a change in body orientation from upright to lying that occurs immediately after a large negative acceleration to signal a fall. Those early wearable devices are not well-accepted by older adults because of their obtrusiveness and limited mobility. However, modern smartphones and smartwatches now contain more sensors than ever before. Data from those devices can be collected more easily and more accurately with the increase in the computing power of those devices. Smartphones are also widespread and widely used daily by people of all ages. There has thus been a dramatic increase in the research on wearable-based fall detection and prevention in the last few years. This is highlighted in the survey paper [12]. The smartphone-based fall detection solutions in general collect accelerometer, gyroscope, and magnetometer data for fall detection. Among the collected sensor data, the accelerometer is the most widely used. The collected sensor data were analyzed using two broad types of algorithms. The first is the threshold-based algorithm which is less complex and requires less computation power. The second is the machine learning-based fall detection solutions. The dynamic of the fall of different people cannot be captured in any rule-based or threshold-based systems. For example, falls from older adults are generally considered as "soft falls" verses "hard falls" when a person falls from a bike. Moreover, threshold-based algorithms cannot be generalized to the variability of falls and ADLs from people of different heights and weights.

The Support Vector Machine (SVM) learning algorithm has been used for fall detection by scholars in the early days in [13]. These scholars used a trunk-mounted tri-axial sensor (a specialized hardware) to collect and sense data. They were able to achieve 99.14% accuracy with four features using only high-pass and low-pass accelerometer data. They used a 0.1-second sliding window to record minimum and maximum directional acceleration in that period for a feature.

Other work in fall detection has focused on using multiple sensors attached to the subject. For instance, sensors can be placed on the lapel, trunk, ankle, pocket, and wrist. These systems typically show marvelous results of 100% accuracy but lack convenience, and portability, and are too computationally intense for a smartphone due to more data needing to be collected and processed in real-time.

There has been a lot of research work on using Recurrent Neural Networks (RNN) or Convolutional Neural Network (CNN) to detect falls in recent years: in [14], the authors describe an RNN architecture in which an accelerometer signal is fed into two Long Short-Term Memory (LSTM) layers, and the output of these layers is passed through two feed-forward neural networks. The second of these networks produces a probability that a fall has occurred. The model is trained and evaluated on the URFD dataset [15], which contains accelerometer data taken from a sensor placed on the pelvis, and produces a 95.71% accuracy. The authors also describe a method to obtain additional training data by performing random rotations on the acceleration signal; training a model with this additional data gives an increased accuracy of 98.57%.

The authors in [16] also proposed an RNN to detect falls using accelerometer data only. The core of their neural network architecture consists of a fully connected layer, which processes the raw data, followed by two LSTM layers, and ends with another fully connected layer. The normalization and dropout layers are introduced in their architecture for generalization. The authors train and test their model with the SisFall dataset [17], which contains accelerometer data sampled at 200 Hz collected from a sensor attached to the belt buckle. In order to deal with a large imbalance in training data, of which ADL's form the vast majority, the authors define a weighted-cross entropy loss function, based on the frequency of each class in the data set that they use to train their model. In the end, their model attains a 97.16% accuracy on falls and a 94.14% accuracy on ADL's. However, it is unrealistic to expect a commodity-based wearable device to sample at 200 Hz in real

time because of battery constraints. Moreover, none of these RNN-based models have been tested or trialed in real world.

Our earlier work [18] compared traditional machine learning (SVM, Naive Bayes) techniques with deep learning (DL) model, in particular, the Recurrent Neural Network (RNN), for fall detection using only acceleration data captured through a commodity-based Microsoft Band wrist-worn watch, and concluded that DL model has better fall detection performance. The best model we obtained using RNN on our SmartFall dataset [19] collected with a Microsoft MSBAND watch has an overall accuracy of 86% in the offline test. Because of its placement on the wrist, a smartwatch will naturally show more fluctuation in its measurements than a sensor placed on the pelvis or belt buckle. In addition, data sensed by a commodity watch is noisier than specialized equipment. The computational power of a smartwatch is constrained and thus we adopted a basic LSTM model with only two dense layers. Our LSTM model is one of the few models that have been trialed in the real world [8]. We have also benchmarked a 1D-CNN against stacked/ensemble LSTM models in [6] and found the stacked LSTM to perform better. This demonstrated that despite being constrained with a smaller dataset, a deep-learning-based approach can learn the fall patterns better than traditional machine learning techniques.

In [20], a hybrid of CNN and LSTM architecture called ConvLSTM is proposed. ConvLSTM can predict falls with a recall of 96% and a precision of 98.69% using SisFall dataset with 5-fold cross-validation. The CNN layer acts as a feature extractor and provides an abstract representation of the input sensor data in feature maps. In other words, the CNN layer captures the spatial relationship in the data while the LSTM layer captures the long-term temporal relationship. They demonstrated that ConvLSTM outperformed models that are solely CNN or LSTM. In [21], a DeepConvLSTM was presented for recognizing two families of human activities, the periodic motion activities such as walking, bicycling and gestures such as drinking water from a cup on Skoda and OPPORTUNITY datasets. The DeepConvLSTM achieved the F1-score of 93% on the OPPORTUNITY dataset. The resulting model has large parameters and data is sensed from multiple locations of the body which makes it impractical for real-world use. DeepConvLSTM has also shown that for sequential data a combination of recurrent LSTM units and Convolutional Neural Networks can outperform CNN models.

Transformer has recently achieved a state-of-the-art in natural language processing (NLP) tasks. The popular ChatGPT tool uses Transformer blocks in its architecture. Transformer when used in natural language processing has been shown to outperform older approaches for sentence classification, which made use of conventional RNNs/LSTMs [22]. Transformer can efficiently process long sequences, it supports parallel computing with fast computation. Transformers have gained popularity for fall detection recently. For example, in [23], they demonstrated that using the accelerometer signals of a waist-worn Inertial Measurement Unit (IMU) can gain a noticeable gain in accuracy of the model of 95.7% using the SisFall dataset. In the Edge Impluse Studio Project [24], a Fall Detection model trained with the Transformer model using the SisFall dataset was deployed on a resource-constrained Arduinio microcontroller with an ADXL345 accelerometer sensor. The model accuracy for fall is around 88%. None of the trained Transformer models have been tested in the real world.

In the Patch-Transformer Network [25] for fall detection, the network includes a convolution layer, a transformer encoding layer, a global average pooling layer, and a linear classification layer. The convolution layer is used to extract local features. Global features of falls are learned through the multi-head self-attention mechanism in the transformer encoding layer. The final classification is provided by the linear layer. The accuracy result obtained using the SisFall dataset for training is 99.86% with a detection time of 0.004 seconds. They demonstrated that an accurate transformer model with a low number of training parameters and model complexity can be obtained with three attention heads and a maximum of six encoding layers. They demonstrated that there is an advantage in adding a CNN layer to a transformer for better detection in fall patterns.

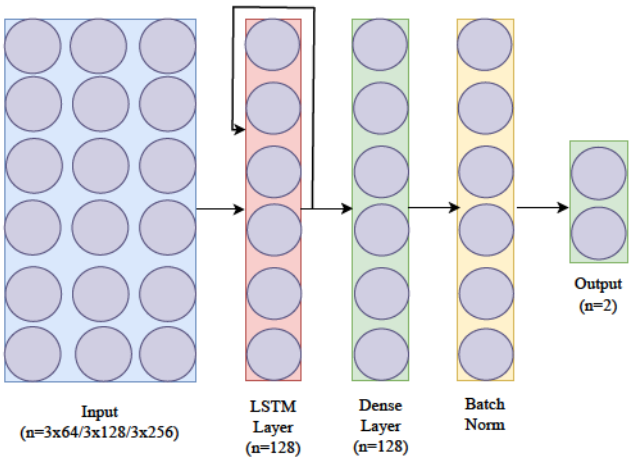

Figure 2. Basic LSTM Architecture.

There is no systematic study that compares fall detection models trained using variants of LSTM and Transformer. In [26], the authors compared LSTM with self-attention to Transformer and concluded that LSTM with self-attention gave better model accuracy than Transformer across four public time series datasets (only one of the datasets is related to fall). However, the trained model is not evaluated in real-time on any wearable device. It is reported in [27] that only 7.1% of fall detection projects performed real-world testing on their models. In particular, we are interested in evaluating whether there is a distinct advantage of using a **Transformer** over **LSTM** in obtaining better model accuracy and acceptable **real-time performance of the model** on a wearable device. For fall detection technologies to be adopted by older adults, the trained offline model must be tested and verified in real world.

3. Architecture of Models

In the literature, there is no consensus on what is the best LSTM or Transformer architecture for fall detection using only accelerometer data from a wristwatch. There is also no universal standard fall dataset that can be used for comparison. We choose to use three variants of the LSTM model. The first one is the basic LSTM model which has been deployed by us successfully in the SmartFall App running on a Huawei watch, the second is a hybrid CNN and LSTM model that has shown good performance by several scholars, and the last one is LSTM with self-attention. The latter has been compared with the Transformer model in [26] and demonstrated better performance over the Transformer. For the Transformer, we adopted a vanilla Transformer used in our SmartFall App and also a ConFormer from [28].

3.1. LSTM

The basic LSTM architecture we used contains an input layer, an LSTM layer, a dense layer, a batch normalization, and an output layer. The input layer contains 3 nodes for the raw data; the accelerometer x,y,z vectors are provided with a variable input shape of (W, 3, 64) where W denotes window size, and 3 signifies the x,y,z values of accelerometer data and 64 is the batch size used. It then feeds through a recurrent LSTM layer and a fully connected dense layer. The output is a sigmoid layer that outputs a predicted probability that a fall has occurred.

The Binary Cross Entropy (BCE) with ADAM optimizer is used. RNNs are traditionally trained with backpropagation through time (BPTT), so it is necessary to specify how many steps n in the past the network should be trained on. This defines the size of the window W of data points that must be fed to the model at each data sample selection. Figure 2 shows the basic LSTM that we have used in our study.

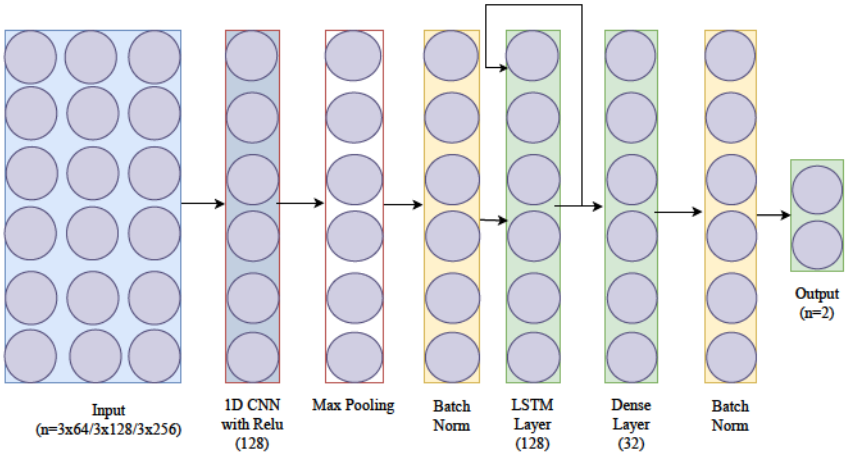

Figure 3. CNN-LSTM Architecture.

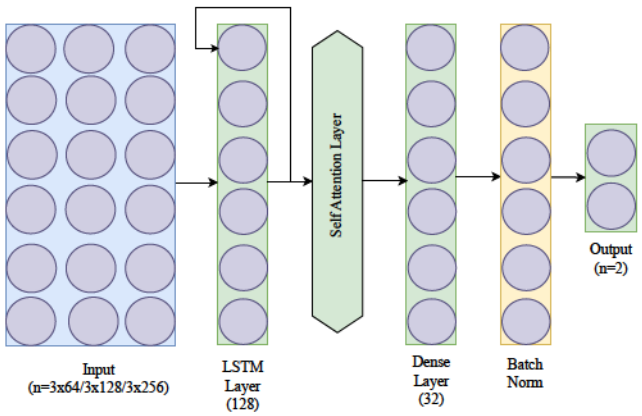

Figure 4. LSTM with self-attention Architecture.

The CNN-LSTM we used integrates two fundamental neural network components: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. It starts with a 1D convolutional layer with a ReLU activation function, followed by max-pooling to capture local patterns in the input sequence. This combination is repeated for increased feature extraction. A subsequent LSTM layer is introduced to capture long-term dependencies in the data. Dense layers with a ReLU activation follow. Batch normalization is applied throughout the model to enhance training stability. The final output layer employs a sigmoid activation function for binary classification. The model is compiled using BCE and an Adam optimizer. Figure 3 shows the CNN-LSTM that we have used in our study.

We implemented another variant of the LSTM model called LSTM w/self-attention, a self-attention layer is incorporated into the basic LSTM implementation to enhance its ability to process time series data. Self-attention is a mechanism that allows the model to weigh the importance of different parts of the input data. It helps the model to focus on the important time points of the input sequence for making predictions. Here, the output of the LSTM layer is passed to a self-attention layer. This means that the sequential output of the LSTM is further processed by the attention mechanism. This addition enriches the model's capacity to identify crucial patterns in data while maintaining its lightweight nature for wearable devices. The model, still trained using backpropagation through time (BPTT), benefits from this advanced feature, offering a more nuanced understanding of temporal dynamics without compromising efficiency. Figure 4 presents the architecture of LSTM with self-attention that we have used.

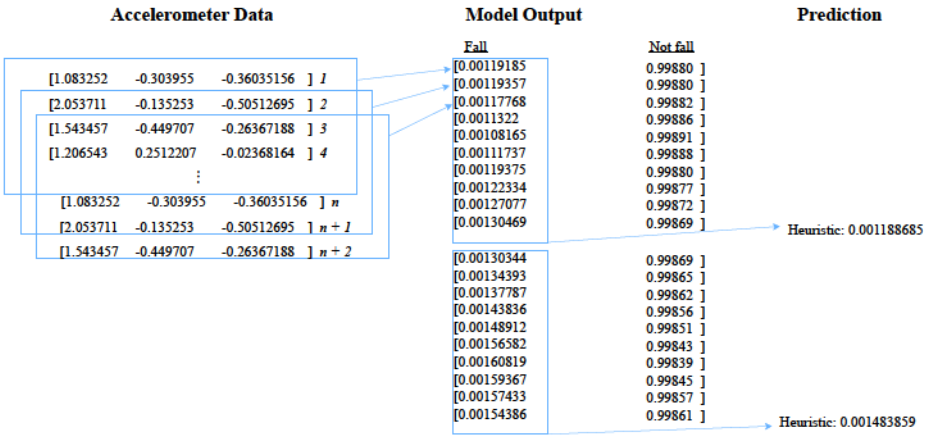

Figure 5. Prediction scheme for LSTM model

The fall prediction using all three variants of LSTM-based architecture is made on a sliding window of data that is W data points in length. Each prediction outputs a probability of fall between 0 and 1. Fig. 5 depicts the prediction schematic. The model prediction begins once the number of sensor data points acquired is equal to the number of the configured window size W. Every model prediction thereafter will only require additional data points which is the step size S. That means the consecutive windows have a $W - S$ time step overlap. However, before producing a final prediction, we generate a heuristic value based on the probabilities produced by the average of 20 consecutive model predictions. This value is derived from experimentation with values ranging from 1 to 20 with 5 increments. In essence, we compute the average value of 20 consecutive probabilities and compare this with a pre-defined threshold value. During the training and validation of the model, a threshold of 0.5 yields the optimal results and is used in the SmartFall App. If the average probability exceeds this threshold, then it is considered a fall prediction. This helps to prevent isolated positive model predictions from triggering a false positive.

3.2. *Transformer*

We implemented the transformer architecture described in [10] that has an encoder-decoder structure. We chose the encoder part for fall detection since it has been shown in [29] that the transformer can achieve high accuracy for human activity recognition. This architecture as shown in Fig. 6 has four encoder layers and four attention-heads in each layer to balance the model size and performance metrics when deployed on power and computation-constrained wearable devices. The original transformer also has a decoder part that generates a new sequence in machine translation. We don't need any decoder layers for fall detection as we don't need to re-construct any new sequences. The attention mechanism is essential in the transformer model, with multiple attention heads seeking different relevance definitions or correlations [29]. Multi-head attention(MHA) involves mapping queries and key-value pairs to an output. In a single attention head, scaled dot products of queries(Q) and keys(K) are calculated, followed by softmax normalization, which yields weights that are multiplied by values (V), as shown in equation 1. The output from multiple attention heads is then concatenated and projected to get the final output as shown in equation 2. The Multi-Head attention is followed by Layer Normalization and a skip connection with input before the Multi-Attention.

Layer normalization is a technique used in the Transformer architecture to normalize the activation of each layer independently, stabilizing training by addressing issues like vanishing gradients and internal co-variate shift. It improves gradient flow, accelerates convergence, and reduces sensitivity to hyperparameters. This contributes to the model's effective training and enhanced performance. The last layer of the Transformer encoder is the feed-forward network/module (FFN). It independently processes each position in the

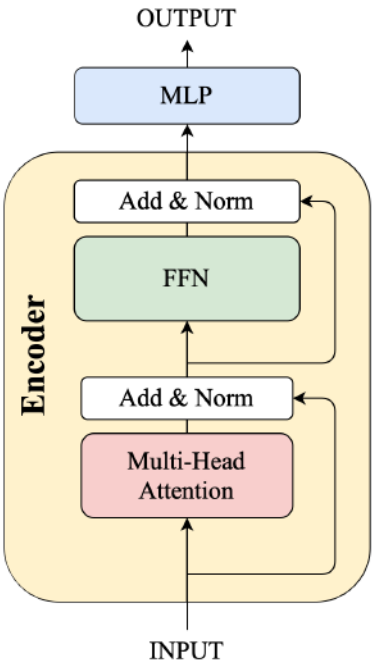

Figure 6. Transformer Architecture

input sequence using equation 3. This step enhances the model's ability to capture complex relationships within the data. The output of the last encoder layer is fed into an MLP layer consisting of three linear layers with 8, 16, and 1 neurons, respectively, to classify falls and ADLs. The sigmoid function was used as an activation function of the output layer to yield a probability between 0 and 1.

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{1}$$

$$\text{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\cdot\mathbf{W}^O \tag{2}$$

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{3}$$

The ConvTransformer as shown in Fig. 7 was inspired by [28]. The encoder layer of it has a sandwich-like structure where two half-feed forward Network(FFN) sandwich between the Multi-Headed Self-Attention module and the Convolution module. The sandwich-like architecture was inspired by Macaron-Net[30]. Starting from scratch, the convolution module combines a ReLU activation with a pointwise convolution. Then, there is only one 1-dimensional depthwise convolution layer used with swish activation. To make training deep models easier, batch normalization is added right after the depthwise convolution process. The sandwich-like structure and convolution module differentiate ConFormer and basic Transformer. We refer to this architecture as ConvTransformer in this paper.

4. Methodology

4.1. Dataset

We evaluated variants of LSTM and Transformer on three datasets: UniMib [31], K-fall [32], and SmartFallMM (https://anonymous.4open.science/r/smartfallmm-4588, data collected in our lab). Fall is a rare event and it is very labor-intensive to collect large amounts of fall data. The largest public dataset we found that has the practical sampling rate for wearable devices, types of falls, and ADLs is the K-fall dataset.

UniMib is a human activity recognition dataset of acceleration data collected from a smartphone. The data was collected from 30 subjects with ages ranging from 18 to 60 years.

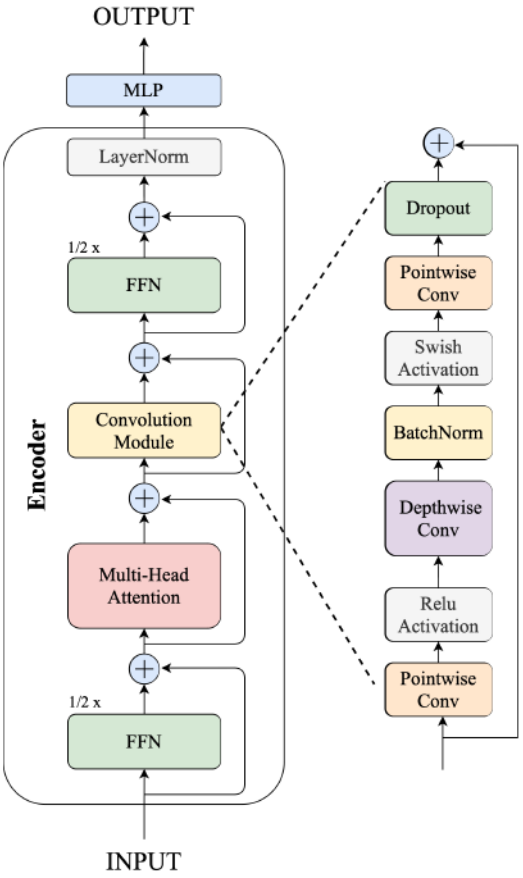

Figure 7. ConvTransformer Architecture

The participants performed 9 types of activities of daily life (ADLs) and 8 types of falls. The data was collected at a sampling rate of 50Hz. The participant puts phones in the left and right pockets of trousers and hand clapping is used to signal the beginning and end of a fall. We processed this dataset and have all the nine types of ADLs labeled as "NonFall". We only retained the common five fall types that we used in our SmartFallMM fall dataset for ease of comparison. The final number of falls is 710 and ADLs is 486.

The K-fall dataset is built for pre-fall, fall, and post-fall detection. 21 types of activities of daily life and 15 types of simulated falls were performed by 32 young and healthy participants. A nine-axis inertial sensor was attached to the participant's lower back to collect the accelerometer, gyroscopes, and magnetometer data. We did not filter out any type of falls so that a reasonably large dataset can be used. In total, K-fall has 5075 motion files including 2729 ADLs and 2346 fall.

SmartFallMM is a multimodal dataset focused on falls and ADLs data, gathered from 16 student participants (11 male and 5 female) with a median age of 23 and 26 older participants (12 male, 14 female) with a median age of 65.5. Our multimodal data collection is approved by IRB 7846 at Texas State University and is the first multi-modal dataset that has both older and younger adults' data. We only asked student participants to perform falls on an air mattress. All participants must sign consent forms before their data can be collected. Two modalities are collected using four types of devices. We collected skeleton data using Azure Kinect cameras, time-series data such as accelerometer, and gyroscope using three inertial sensors (i.e. Meta Sensor, Huawei Smartwatch, and Nexus phone). Meta Sensor was developed by MBIENTLAB in San Francisco (mbientlab.com). The participant wears the Huawei watch on the left wrist and puts the Nexus smartphone on the right hip inside a harness. A wrist meta sensor is placed on the right wrist of the participant and the hip meta sensor is placed on the left hip of the participant (clips on the belt). This setup

Table 1. Summary of the datasets used

	SmartFalllMM	UniMib	K-fall
#ADLs	560	710	2729
#Fall	400	486	2346
# Subjects	16	30	32
Sampling Frequency	32Hz	50Hz	100Hz
Type of devices	Kinect cameras, phone, watch, meta sensors	phone	9 axis IMU
Device placement	Left and right wrists and hips	Left and right pockets of pants	Lower back

allows us to collect data from four important joints of the human body when a person moves. Figure 8 shows the positions and the type of sensors used for the data collection.

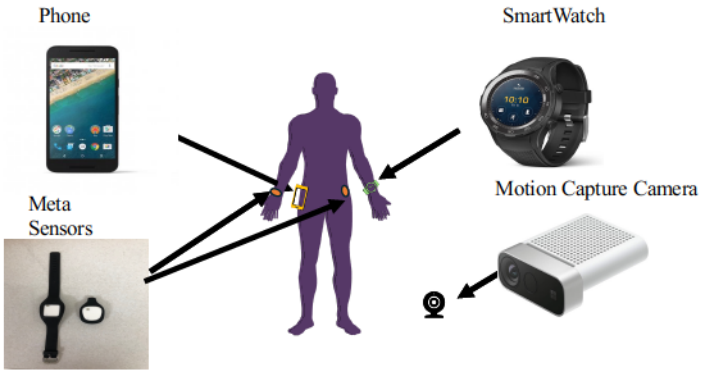

Figure 8. Sensors placement on the participants

For the training and comparative purposes of LSTM and Transformer models for fall detection, we used only the accelerometer data captured at 32Hz from the Huawei watch worn on the left wrist of the 16 student participants because it has both fall and ADL data. We do not and cannot collect fall data from older adults. Five types of falls and eight types of ADL activities are collected. The subset of the SmartFallMM dataset we used has a total of 560 ADLs and 400 Falls. Table 1 gives a summary of the three datasets used for our experiments.

4.2. Input Data Processing

We used the sliding window technique to divide the input data into a series of overlapping windows. The Algorithm 1 shows the technique used for the sliding window. The window size W in the equation below indicates how many data points the model sees at each prediction and the *step* parameter dictates how far the window moves down during each iteration of sample selection. We utilized a step size of 10 which allows ten new data points to be added to every new window created. We have experimented with different step sizes ranging from one to 50 and found that using a step size of 10, captures more meaningful information when defining training instances. We experimented with different window sizes of 64, 128, and 256 in our study.

5. The Computational Model

5.1. Hyperparameters

We train the models using TensorFlow on Dell Precision 7820 Tower with 256 GB RAM and one GeForce GTX 1080 GPU. We try to keep similar hyper-parameters for both variants of models as much as possible. However, due to the salient differences in their

Algorithm 1 Sliding Window Algorithm

Input: Input dataframe df, window size W, step size $step$
Output: Output dataset of all windows $windows$
Initialize an empty array $windows$
for $i \leftarrow 0 :$ step : length of activity **do**
    Set $currentWindow$ as the subarray of df from index i to $i + W - 1$
    Append $currentWindow$ to $windows$
end for

Table 2. Hyper Parameters

Names	Models	
	Transformer*	LSTM*
Learning Rate	.001	.001
Epochs	100	100
Batch Size	64	64
Optimizer	Adam	Adam
Loss Function	BCE	BCE
Encoder	4	-
MHA Heads	4	-
MHA Dim	128	-
MLP Dimension	16	128

architecture, some differences are unavoidable. The best hyperparameters used for the variants of Transformer and LSTM are listed in Table 2.

5.2. Training and Evaluation

As our dataset is not very large, we employed a Leave-One-Out strategy, wherein the entire set of activities associated with a particular individual is reserved exclusively for testing. This strategic maneuver ensures that the model remains entirely unexposed to any data emanating from a given participant during the training phase. By adopting such an approach, we effectively mitigate the risk of data leakage, thereby having a more reliable and robust assessment of model generalization on unseen data.

5.3. Evaluation Metric

To evaluate and compare the performance of Transformer and LSTM, we look at F1 score, Precision, Recall and Accuracy. These metrics are defined as:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\_Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

True Positive(TP) occurs when the model correctly predicts a positive instance, such as accurately detecting a fall. True Negative (TN) is when the model correctly predicts a negative instance(ADL for our case). False Positive is the case when the model erroneously predicts a negative data sample as positive and False Negative is the opposite case. While Precision assesses the accuracy of positive prediction, Recall quantifies the correct identification of

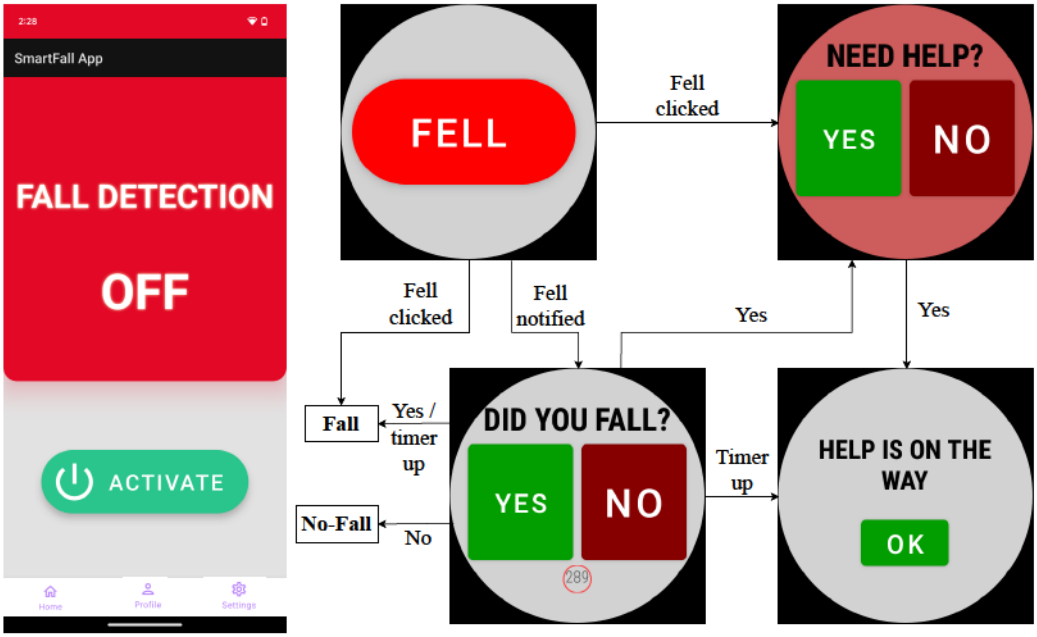

Figure 9. User Interface of SmartFall App for Real-time Evaluation

real positives. F1-Score is a harmonic mean of precision and recall to provide a balanced measure of model performance.

5.4. Model Evaluation Method

The experiment explores two forms of model evaluation. The traditional offline machine learning model evaluation and the real-time evaluation with the SmartFall App which is a fall detection application developed in our lab. In the offline model evaluation phase, a model is trained on a training dataset, tested on a validation set to get the best hyper-parameters, and used the best-validated model on the test data.

We selected only the two best Transformer and LSTM models trained on the SmartFallMM dataset for real-time evaluation. The model trained with a window size of 128 has the highest F1 score for both the Transformer and LSTM and was chosen for real-time evaluation. The models trained with UniMib and K-fall cannot be used for real-time testing by the SmartFall App [33] because the data were not collected using a smartwatch at a wrist position.

Fig. 9 shows the phone and watch's UI of the SmartFall application. After pairing the phone and the watch, as soon as the user presses "activate" on the phone's UI, the sensor on the watch will sense the accelerometer data from the watch continuously. Model predictions (i.e., predictions produced by the neural architecture) begin once the number of sensor data points acquired is equal to the defined window size.

We recruited three student participants for the real-time evaluation of LSTM and Transformer models at our lab. Our real-time evaluation of the model is also covered under the IRB 7846 at Texas State University. We asked each student participant to sign a consent form. The consent form gives the student participant the information on why this research study is being done and describes what they will need to do to participate as well as any known risks, inconveniences, or discomforts that they may have while participating. Each student participant is asked to wear the watch on the left wrist and perform the five types of falls on a 12-inch high queen-size air mattress. The same participant is also asked to perform eight prescribed lists of ADLs.

Table 3. Offline Evaluation with three datasets

Dataset	Window Size	Metrics	Transformers		LSTM		
			Transformers	ConvTransformers	LSTM	CNN-LSTM	LSTM w/self-Attn
SmartFallMM	64	Precision	84.9	64.1	64.8	74.7	71.3
		Recall	65.8	72.3	77.2	89.4	82.1
		F1-Score	74.0	67.9	71.2	81.3	76.3
	128	Precision	88.9	74.4	76.5	83.2	76.5
		Recall	77.0	83.1	83.2	92.5	82.9
		F1-Score	**82.6**	78.5	79.7	**87.6**	79.6
	256	Precision	78.3	72.9	64.3	83.5	75.2
		Recall	84.0	89.0	76.8	79.2	79.6
		F1-Score	81.1	80.8	70.1	81.3	77.3
UniMib	64	Precision	79.1	78.4	84.1	94.4	85.0
		Recall	75.5	80.5	89.1	93.6	89.3
		F1-Score	77.2	79.4	86.1	**93.7**	87.1
	128	Precision	84.8	85.6	71.3	83.1	85.8
		Recall	77.2	87.7	88.4	91.2	93.5
		F1-Score	80.8	**86.6**	79.1	87.5	89.4
	256	Precision	70.1	79.0	61.3	85.4	81.1
		Recall	83.9	85.0	81.2	96.1	86.1
		F1-Score	76.4	81.9	70.2	90.3	83.6
K-fall	64	Precision	82.6	72.5	82.4	83.5	78.8
		Recall	84.9	86.3	78.7	81.5	83.9
		F1-Score	83.8	78.8	80.1	82.2	80.1
	128	Precision	84.8	81.1	89.1	85.4	81.0
		Recall	86.8	89.0	85.6	89.1	89.2
		F1-Score	85.8	84.9	87.3	87.4	84.3
	256	Precision	88.5	93.4	95.3	93.1	91.5
		Recall	93.5	95.7	89.6	96.3	93.1
		F1-Score	90.9	**94.0**	92.2	**95.5**	92.3

6. Results

6.1. Results of Traditional Offline Model Evaluation

Table 3 showed the performance of five different machine learning models across three datasets: SmartFallMM, UniMib, and K-fall, with varying window sizes of 64, 128, and 256. The performance metrics used for comparison were precision, recall, and F1-score.

As shown in Table 3, on the SmartFallMM dataset, the models demonstrated varied performance across different window sizes. The basic Transformer model exhibited an increase of 8.6% in F1-score with the window size increasing from 64 to 128. The basic Transformer model has the highest F1-score of 82.6% with a window size of 128.

The basic LSTM showed a relatively lower F1-score across all window sizes, indicating that simple recurrent structures might not be sufficient for the complexity of the task. The CNN-LSTM performed at higher F1-scores. The best-performing variant of the LSTM model is CNN-LSTM with a score of 87.6% at the window size of 128. A 7.2% improvement in F1-score for the LSTM with attention for windows size of 256, compared to the basic LSTM, indicates that self-attention effectively captures relationships between different timesteps.

Results on the UniMib dataset were less consistent. The basic Transformer model improved by 3.6% going from a window size of 64 to 128. The ConvTransformer showed its highest F1-score of 86.6% at the mid-window size of 128, suggesting a balance between input sequence length and the influence of spatial model capability. The LSTM models, particularly the CNN-LSTM and LSTM with self-attention, again demonstrated strong performance, with the CNN-LSTM performing at the highest F1-score (93.7%) with the

Table 4. Realtime evaluation with SmartFallMM

	Metrics	Transformer	CNN-LSTM
Participant 1	Precision	65.7	53.1
	Recall	100	100
	F1-score	78.7	69.2
	Accuracy	80.0	66.1
Participant 2	Precision	73.5	55.5
	Recall	100	100
	F1-score	84.3	70.9
	Accuracy	86.1	69.2
Participant 3	Precision	52.0	59.4
	Recall	100	88.0
	F1-score	68.4	70.6
	Accuracy	64.6	76.9
Average	Precision	63.7	56.0
	Recall	100	96.0
	F1-score	77.1	70.2
	Accuracy	76.9	70.7

smallest window size of 64. This might suggest that when the dataset is small, the influence of the convolutional layer is stronger.

For the K-fall dataset, the trend shifted with the ConvTransformer achieving the highest F1-score of 94.0% at the largest window size of 256, surpassing the basic Transformer model by 4% at that window size. Moreover, the recall and precision are both above the 90%. A similar trend is observed in all the variants of LSTM models. The better performance of models with a larger window size in K-fall can be attributed to the higher sampling rate of K-fall data. K-fall has a sampling rate of 100Hz which is double that of UniMib with 50Hz and SmartFallMM with 32Hz. As the K-fall had more data points per second, a bigger window size can accommodate the whole duration of the fall sequence better.

K-fall is the largest dataset we used and our experimental results thus indicate that there is a definite increase in F1-score by training with a bigger dataset across all the variants of transformer and LSTM. The model trained with the bigger dataset, also presented the most balanced and consistently high precision and recall. For the offline test, the CNN-LSTM which can learn both the spatial and temporal nature of the data performs better than both the basic Transformer and ConvTransformer. We believe this might be because the transformers don't have any inductive bias for temporal information and ConvTransformer only has bias for spatial information.

6.2. Results of Real-time Evaluation

For the real-time evaluation of the models, we choose the best-performing offline model of transformer and LSTM trained using SmartFallMM dataset with a window size of 128 as shown in Table 3. We use the terms Transformer and basic Transformer synonymously here for ease of reference. The performance of each model for three young adults is shown in Table 4. Each participants performed each activity five times. The outcome is recorded as "Yes" if a true positive is detected. The Transformer model consistently outperformed the CNN-LSTM model for all three participants. For the Participants 1 and 2, the Transformer outperforms CNN-LSTM across all evaluation metrics. Although, for Participant 3, the CNN-LSTM performs better in terms of F1-score by 2.2% , it misses some falls. In addition, the average F1-score (77.1%) of the Transformer in real-world test is closer to the F1-score (82.6%) of the Transformer model trained offline. However, that is not the case for the CNN-LSTM model which has an offline F1-score of 87.6% but only achieved a 70.2% F1-score in a real-world test. This indicates that only the Transformer model can transfer the offline model result to the real world.

Table 5. Real Time Evaluation Using SmartFall App.

		Transformer					CNN-LSTM				
		Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
Participant 1	Front Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Back Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Left Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Right Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Rotate Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Participant 2	Front Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Back Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Left Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Right Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Rotate Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Participant 3	Front Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Back Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Left Fall	Yes	Yes	Yes	Yes	Yes	Yes	**No**	**No**	Yes	Yes
	Right Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**No**	Yes	Yes
	Rotate Fall	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

To analyze the model's performance in more detail, we examined the individual fall activities' detection outcomes, as displayed in Table 5. By examining the table, it becomes evident that certain falls are readily identifiable, as both models exhibit comparable accuracy in these cases. The main difference between the Transformer and LSTM is imminent in case of detecting the Left and Right falls. For instance, both models detected Front, Back, and Rotate falls. For all 3 participants, the Transformer does not miss any of the falls. Whereas, CNN-LSTM missed one Left fall and two Right falls for Participant 3.

The three participants also performed eight ADL activities with a repetition of five of each activity which amounts to 40 activities in total for each participant. For the Transformer, the number of false positives for Participant 1 is 13/40, Participant 2 is 9/40 and Participant 3 is 23/40. For the CNN-LSTM model, the false positives for Participant 1 is 11/40, Participant 2 is 20/40 and Participant 3 is 15/40. This again indicates that the Transformer can classify ADLs better than CNN-LSTM which is reflected in the lower precision scores of CNN-LSTM across the three participants.

The low F1-score obtained when evaluating the CNN-LSTM model in real-time scenarios indicates that the model struggles to effectively apply its learned capabilities to the real world. Though a CNN-LSTM model with a high 87.6% F1-score was deployed on the SmartFall App for real-time evaluation, none of the three participants achieved F1-score close to that. The highest achieved F1-score is 70.9% by Participant 1. In contrast, a Transformer model can maintain almost the same high level of F1-score in real-time circumstances. This suggests that the Transformer model inherits the capability of successfully putting into practice what it has learned and adapted to the dynamic nature of the real-time environment, demonstrating its robustness and reliability for deploying to edge devices.

7. Conclusions

We assessed the performance of five deep learning models' performance on three fall datasets and evaluated the real-time performance of the best model trained with SmartFallMM data from the variants of LSTM and the Transformer using our SmartFall App. The real-time evaluation confirm that the Transformer outperforms LSTM in dynamic real-world scenarios. To the best of our knowledge, we are the first to perform comparative studies involving real-time testing of deep learning-based fall detection models on a smartwatch. The only other real-world test of a fall detection App on a smartwatch is with a model using a threshold-based algorithm [34] that achieved a recall of 77%. This means a threshold algorithm missed 23% of falls in the real-world test while a deep learning-based model has almost 100% recall based on our real-world evaluation. The real-world test is performed using Huawei watch, but our system is not restricted to the Huawei brand of

watch. We have used TicWatch in other real world evaluation studies. Any smartwatch that runs WearOS can be used.

Our findings showed that in the offline training, the CNN-LSTM model was better than the basic Transformer model for all the datasets and all the windows in F1 scores. This is due to the joint spatial and temporal inductive bias on the CNN-LSTM. In Transformers, due to the lack of inductive biases, it needs a lot of data to gain optimal performance. This is observable in the K-fall dataset, as Transformer achieved an F1-score of 94.0% for 256 window size. The transformer might need an even larger dataset to outperform the CNN-LSTM.

For the real-world test, CNN-LSTM produces a higher number of false positives than Transformers showing that Transformers can learn complex patterns better due to exploring the relationships in data via the attention heads. The drop in F1-score is 5.5% for Transformer in the real-time test which is much smaller than CNN-LSTM's 17.4% which makes the Transformer a more reliable model.

When running the trained model on an edge device like an Android phone, there is a need to convert the trained TensorFlow model to a TensorFlow Lite version. As TensorFlow Lite is a compressed version of the original model, there will be a reduction in model accuracy. This is confirmed by the fact that the F1 scores from all three participants were lower in the real-time test across both types of models.

Based on our experiments comparing the Transformer and LSTM, our findings suggest that the Transformer may be a preferable choice for deployment in real-time fall detection applications. This comparison could offer valuable insights to future researchers aiming to transition their offline deep learning models into real-world testing. In scenarios like fall detection, it's crucial to consider the feasibility of implementing such models in practical applications, making this a significant consideration. This study also demonstrated that models trained from data collected from a wrist position performed worse than data collected from the hips and lower back. This indicates the challenges of using only the wrist data for fall detection.

The metric we used for comparison is restricted to model accuracy such as precision, recall, and F1-Score. The power consumption and inference time are not within the scope of this study. Interested readers can refer to our recent paper "An Empirical Study on AI-Powered Edge Computing Architectures for Real-Time IoT Applications" in [33] regarding the impact of software architecture on the power consumption, inference latency, and model accuracy.

8. Future Work

Although the transformer model outperforms CNN-LSTM in terms of our assessment scores in the real-world test, the studied model has been trained so far on only a small-scale dataset and there are still far too many false positives. To the best of our knowledge, a larger dataset than SmartfallMM collected using a smartwatch as the sensor is not publicly available at the moment. Since Transformers are known to perform well with bigger datasets, one of our immediate future goals is to increase the size of our dataset by leveraging techniques such as data augmentation, generative models, or extracting from videos of people falling to augment our dataset [35]

Our real-world test is conducted with three participants. To further validate our conclusion that the Transformer is preferable for real-world deployment, we will recruit 10 more participants to validate the rate of true positives and false positives of the model this year. Our longer-term goal is to leverage SmartFallMM, the multimodal dataset we collected for multi-modal learning and to create a fall detection model that can leverage this dataset consisting of skeleton and accelerometer data during training and can make inferences only using accelerometer data from the wrist via the knowledge distillation method.

Acknowledgment

We thank the National Science Foundation for funding the research under the NSF-SCH (21223749). We also thank the summer 2023 REU students Jamee Labberaton and Vasilisa Ignatova for conducting an initial study on this subject and for labeling the SmallFallMM dataset.

References

1. Falls, World Health Organization howpublished=www.who.int/news-room/fact-sheets/detail/fall, note = Accessed: 13 June 2023.
2. Tacconi, C.; Mellone, S.; Chiari, L. Smartphone-based applications for investigating falls and mobility. In Proceedings of the 2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops, May 2011, pp. 258–261. https://doi.org/10.4108/icst.pervasivehealth.2011.246060.
3. Chen, L.; Li, R.; Zhang, H.; Tian, L.; Chen, N. Intelligent fall detection method based on accelerometer data from a wrist-worn smart watch. *Measurement* **2019**, *140*, 215 – 226. https://doi.org/https://doi.org/10.1016/j.measurement.2019.03.079.
4. Medical Life Alert Systems. http://www.lifealert.com.
5. Mobilehelp Smart. https://www.mobilehelp.com/pages/smart. Accessed: 2019-11-18.
6. Mauldin, T.R.; Ngu, A.H.; Metsis, V.; Canby, M.E. Ensemble Deep Learning on Wearables Using Small Datasets. *ACM Trans. Comput. Healthcare* **2021**, *2*. https://doi.org/10.1145/3428666.
7. Mauldin, T.R.; Canby, M.E.; Metsis, V.; Ngu, A.H.; Rivera, C.C. SmartFall: A Smartwatch-Based Fall Detection System Using Deep Learning. *Sensors* **2018**, *18*.
8. Ngu, A.H.; Metsis, V.; Coyne, S.; Srinivas, P.; Mahmud, T.; Chee, K.H. Personalized Watch-Based Fall Detection Using a Collaborative Edge-Cloud Framework. *International Journal of Neural Systems* **2022**, *32*, 2250048, [https://doi.org/10.1142/S0129065722500484]. PMID: 35972790, https://doi.org/10.1142/S0129065722500484.
9. Guan, Y.; Plötz, T. Ensembles of Deep LSTM Learners for Activity Recognition Using Wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2017**, *1*, 11:1–11:28. https://doi.org/10.1145/3090076.
10. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems; Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R., Eds. Curran Associates, Inc., 2017, Vol. 30.
11. Nahian, M.J.A.; Raju, M.H.; Tasnim, Z.; Mahmud, M.; Ahad, M.A.R.; Kaiser, M.S. Contactless fall detection for the elderly. *Contactless Human Activity Analysis* **2021**, pp. 203–235.
12. Habib, M.A.; Mohktar, M.S.; Kamaruzzaman, S.B.; Lim, K.S.; Pin, T.M.; Ibrahim, F. Smartphone-based solutions for fall detection and prevention: challenges and open issues. *Sensors* **2014**, *14*, 7181–7208.
13. Liu, S.H.; Cheng, W.C. Fall Detection with the Support Vector Machine during Scripted and Continuous Unscripted Activities. *Sensors* **2012**, *12*, 12301. https://doi.org/10.3390/s120912301.
14. Theodoridis, T.; Solachidis, V.; Vretos, N.; Daras, P. Human fall detection from acceleration measurements using a Recurrent Neural Network. In *Precision Medicine Powered by pHealth and Connected Health*; Springer, 2018; pp. 145–149.
15. Kwolek, B.; Kepski, M. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer methods and programs in biomedicine* **2014**, *117*, 489–501.
16. Musci, M.; De Martini, D.; Blago, N.; Facchinetti, T.; Piastra, M. Online Fall Detection using Recurrent Neural Networks. *arXiv preprint arXiv:1804.04976* **2018**.
17. Sucerquia, A.; López, J.D.; Vargas-Bonilla, J.F. SisFall: A fall and movement dataset. *Sensors* **2017**, *17*, 198.
18. Khojasteh, S.B.; Villar, J.R.; Chira, C.; González, V.M.; de la Cal, E. Improving Fall Detection Using an On-Wrist Wearable Accelerometer. *Sensors (14248220)* **2018**, *18*, N.PAG.
19. Fall data collected using Microsoft Band Smartwatch. http://www.cs.txstate.edu/~hn12/data/SmartFallDataSet. Accessed: 2019-04-18.
20. Yu, X.; Qiu, H.; Xiong, S. A Novel Hybrid Deep Neural Network to Predict Pre-impact Fall for Older People Based on Wearable Inertial Sensors. *Frontiers in Bioengineering and Biotechnology* **2020**, *8*. https://doi.org/10.3389/fbioe.2020.00063.
21. Ordóñez, F.J.; Roggen, D. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* **2016**, *16*. https://doi.org/10.3390/s16010115.
22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the NIPS, 2017.
23. Mohammed Sharook, K.; Premkumar, A.; Aishwaryaa, R.; Amrutha, J.M.; Deepthi, L.R. Fall Detection Using Transformer Model. In Proceedings of the ICT Infrastructure and Computing; Tuba, M.; Akashe, S.; Joshi, A., Eds., Singapore, 2023; pp. 29–37.
24. Fall Detection Using A Transformer. https://www.edgeimpulse.com/blog/save-yourself-a-trip-fall-detect-using-a-transformer-model, note= Accessed: 2023-08-14.
25. Wang, S.; Wu, J. Patch-Transformer Network: A Wearable-Sensor-Based Fall Detection Method. *Sensors* **2023**, *23*. https://doi.org/10.3390/s23146360.

26. Katrompas, A.; Ntakouris, T.; Metsis, V. Recurrence and self-attention vs the transformer for time-series classification: a comparative study. In Proceedings of the International Conference on Artificial Intelligence in Medicine. Springer, 2022, pp. 99–109.
27. Chaudhuri, S.; Thompson, H.; Demiris, G. Fall Detection Devices and their use with Older Adults: A Systematic Review. *Journal of Geriatric Physival Therapy* **2014**, *37*, 178–196.
28. Gulati, A.; Qin, J.; Chiu, C.C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y.; et al. Conformer: Convolution-augmented Transformer for Speech Recognition **2020**.
29. Maray, N.; Ngu, A.H.; Ni, J.; Debnath, M.; Wang, L. Transfer Learning on Small Datasets for Improved Fall Detection. *Sensors* **2023**, *23*. https://doi.org/10.3390/s23031105.
30. Lu, Y.; Li, Z.; He, D.; Sun, Z.; Dong, B.; Qin, T.; Wang, L.; Liu, T.Y. Understanding and improving transformer from a multi-particle dynamic system point of view. *arXiv preprint arXiv:1906.02762* **2019**.
31. Micucci, D.; Mobilio, M.; Napoletano, P. UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones. *Applied Sciences* **2017**, *7*. https://doi.org/10.3390/app7101101.
32. Yu, X.; Jang, J.; Xiong, S. A large-scale open motion dataset (KFall) and benchmark algorithms for detecting pre-impact fall of the elderly using wearable inertial sensors. *Frontiers in Aging Neuroscience* **2021**, *13*, 692865.
33. Yasmin, A.; Mahmud, T.; Debnath, M.; Ngu, A. An Empirical Study on AI-Powered Edge Computing Architectures for Real-Time IoT Applications. In Proceedings of the IEEE Computers, Software, and Applications Conference (COMPSAC 2024), 2023.
34. Brew, B.; Faux, S.G.; Blanchard, E. Effectiveness of a Smartwatch App in Detecting Induced Falls: Observational Study. *JMIR Form Res* **2022**, *6*, e30121. https://doi.org/10.2196/30121.
35. DATABRARY - VIDEOS OF FALLS IN LONG TERM CARE. https:https://www.sfu.ca/ipml/research/data-sharing.html.