A comparative analysis   
of different deep learning models   
in fall detection on SmartWatch Dataset

|  |  |
| --- | --- |
| Thi Thuy Tien Tran  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  ttran15@lakeheadu.ca | Azeem Abdul  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  aabdul13@lakeheadu.ca |
| Darcus Angeline Peter  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  dpeter3@lakeheadu.ca | Nimasha Warnakulasuriya  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  nwarna@lakeheadu.ca |

*Abstract*— Falls among the elderly are a critical issue all over the world, with 28–35% of people aged 65 and older experiencing falls every year, according to World Health Organization. With the advancement of IoT technology, fall detection systems have emerged as a promising solution for monitoring and detecting falls. This research contributes to the field by comparing state-of-the-art transformer models (TFT and BERT) with traditional deep learning models (GRU and LSTM) using SmartWatch datasets for fall detection. In data preprocessing process, as proper training requires a number of past steps to capture fall events accurately without including irrelevant data, a 40-point window size was applied to successfully capture fall dynamics and distinguish falls from other activities. There was a notable imbalance between the number of fall and non-fall instances in the generated dataset. To address this issue, non-fall data was downsampled to match the number of fall instances, ensuring a balanced training set. Regarding model training, all models were trained with 20% of the data for validation, using a learning rate of 0.0001, a batch size of 512, and 50 epochs. As a result, this study found that the TFT-LSTM model performed best with approxiamately 96% accuracy. TFT-LSTM is leading in accuracy, recall, and F1-score, making it the most effective at detecting falls.

*Index Terms*—fall detection, deep learning, BERT, LSTM, GRU, TFT, SmartWatch dataset

# INTRODUCTION

For elderly people worldwide, fall has become a prevalent issue. The World Health Organization (WHO) reports that 28-35 percent of adults 65 years of age and older fall each year [1]. 30% to 50% of residents in long-term care facilities experience falls each year, with many recurrences [2]. The incidence of falls and their negative health effects rise with age in humans [3]. Falls are a major contributing factor from minor to severe injuries and even death [2]. Approximately two-thirds of all recorded injury-related deaths in individuals over 85 were attributed to falls [4]. About

20% of falls cause fatal injuries such head trauma, hip fractures, etc. [4]. Unintentional injuries brought on by falls is among the most expensive medical procedures [5]. In the US, fall-related hospitalizations cost an average of $17,483 per hospital stay for people 65 and older in 2004 [2]. By 2040, that number is expected to rise to $240 billion [2].

Internet of Things (IoT) is referred to a network of devices that are connected to the internet [6]. Applications IoT in healthcare can improve patient care, reduce expenses, increase access to care [6]. As IoT develops rapidly, using fall detection systems to track everyday activities and identify falls in the elderly has become more promising [7]. Early intervention can potentially reduce healthcare expenses and minimize the demand on caregivers [data 3]. While fall prevention is not the exclusive objective of a fall detection system, caregivers and medical professionals can benefit greatly from the information it offers [9].

Despite the increasing research on fall detection, there is a lack of studies applying transformers to fall datasets. The key contribution of this research is to conduct a comparative analysis of state-of-the-art transformer models (TFT and BERT) versus traditional deep learning models (GRU and LSTM) to evaluate their performance in developing fall detection models using SmartWatch datasets. The code for this study is available in this Github [repository](https://github.com/ttran15/fall_detection).

The remainder structure is arranged as follows: Section II examines related works; Section III provides information about the methodology; Section IV discusses results; and Section VI summarizes the findings.

# RELATED WORKS

The development of modern technologies for the detection and prevention of falls in older people has accomplished significant advances recently [8].

There are three types of fall detection studies that are currently being conducted: wearable sensors that monitor posture, computer vision systems that use video to identify falls, and environmental sensors that detect falls through vibrations or audio [system 4].

Regarding wearable sensors, a crucial sensor for wristwatch fall detection is accelerometer, which measures changes in an object's acceleration, the rate at which velocity changes over time [11]. In order to examine the effects of sensor placement and algorithm selection, a study on fall detection systems assessed a number of sensors using four machine learning algorithms (SVM, NB, Decision Tree, and KNN) [12]. The study concluded that the most effective sensors are those that are close to the body's center of gravity, which is the waist and chest, using ANOVA analysis [12]. Using three datasets (Smartwatch, Notch, and Farseeing), the researchers tested traditional (SVM, Naive Bayes) and non-traditional (Deep Learning) algorithms for fall detection and found that Deep Learning models consistently outperformed traditional ones [1].

Regarding environmental sensors, since wearable systems have a smaller sensing range and are more susceptible to interference from the environment, environmental fall detection methods—which use features like floor vibrations, sound, or infrared images—are more expensive and less accurate than wearable systems [10]. For emergency notifications, a system distinguishes falls from similar events is integrated with smart devices such as Google Home or Amazon Alexa [13]. It outperforms other methods with a 95.14% fall detection rate and a 97.22% accuracy [13]. Carla Taramasco et al. suggested classifying falls and notifying caregivers using very-low-resolution thermal sensors [14]. The data are then analyzed by three recurrent neural networks including LSTM, GRU, and Bi-LSTM [14].

Computer-vision fall detection uses probabilistic models and image processing to classify movements. While it doesn't require expensive equipment, it has drawbacks such as interference from the environment and privacy issues [10]. In contrast, skeleton recognition provides greater privacy protection and environmental resilience than human contour recognition [system 4]. Ahmad Lotfi et al. proposed method uses a multilayer perceptron neural network for classification, generates 10 features for fall detection, and extracts motion information around the body using an ellipse and bounding box [15]. With the UR Fall Detection dataset, it achieves a 99.60% detection rate and a 2.62% false alarm rate [15].

Since accelerometers are common in smart devices and falls cause a noticeable change in acceleration, this data is crucial for detecting falls [11]. Regarding models, traditional machine learning algorithms have the drawback of requiring human feature extraction and selection prior to learning, meaning that the efficacy of the algorithms depends on the researcher's ability to specify relevant features [1]. On the other hand, deep learning techniques provide a wide range of hyperparameter tuning options, which helps them overcome the limitations of traditional machine learning techniques [3]. Therefore, this study proposes a fall detection framework based on wearable-sensor fall detection dataset using deep learning techniques.

# METHODOLOGY

## Dataset

The dataset used in this research is Smart Watch dataset. SmartWatch dataset collected data from seven healthy participants who wore MS Band watches and engaged in simulated falls and activities of daily living (ADLs) like running and hand waving [1]. Four types of falls were performed onto a mattress, with ten repetitions of each direction [1]. Many sampling rates were used to collect the data, with testing determining that 31.25 Hz was the best option [1]. Real-time data labeling as "Fall" or "NotFall" was accomplished through a smartphone app; however, human labeling resulted in some errors that were later fixed by post-processing to guarantee accurate labeling of fall phases [1]. SmartWatch dataset can be accessed via this [link](https://userweb.cs.txstate.edu/~hn12/data/SmartFallDataSet/SmartWatch/).

SmartWatch dataset consists of two files, training dataset and testing dataset. Each file includes four columns: accelerometer\_x, accelerometer\_y, accelerometer\_z, and outcome. Outcome has two values "Fall" and "Not Fall". Table 1 summarize number of records in SmartWatch original training and testing dataset.

TABLE I

SmartWatch original dataset description

|  |  |  |  |
| --- | --- | --- | --- |
|  | Fall | Not Fall | Sum |
| Training | 4,550 | 29,470 | 34,020 |
| Testing | 2,275 | 14,954 | 17,229 |
| Sum | 6,825 | 44,424 | 51,249 |

Figure 1 illustrates signal values from the SmartWatch dataset, showing accelerometer measurements of daily body motion in the three axes: x, y, and z coordinates.

A graph showing a graph of a graph

Description automatically generated with medium confidence

Figure 1. Signal value extracted from SmartWatch dataset

## Techniques

Deep learning has demonstrated remarkable efficacy in applications related to computer vision, speech recognition, and natural language processing [16]. In this research, we use various kinds of deep learning models including LSTM, GRU, BERT, and TFT.

### LSTM

Long Short-term Memory (LSTM), a type of a recurrent neural network (RNN), is particularly good at learning long-term dependencies and is suitable for sequential data such as time-series [17]. In order to prevent the vanishing gradient problem, in which the gradient getting too small or large during backpropagation, they use input, forget, and output gates to controlling data flow and maintaining crucial information [modelstmgru].

The model architecture consists of a LSTM layer, dense layer, dropout layer and output layer.

The input to the model is resized into a form that resembles the training data’s dimensions. LSTM layer is set with 256 as the number of units and input shapes are provided which enables the capacity to learn pattern and dependencies in the data.

After the LSTM layer, we added a dense layer with 128 neurons and passed it through a ReLU activation function. This layer is benefical in bringing non linearity to the model and thus,such non linear patterns from the data can be learned by the model.

Again,regularization technique called dropout layer was used with a dropout rate of 0.5 .This will effectively drop out half the input units during training thus improving the model’s regularity and its performance.

The last layer is a dense layer with one neuron and a sigmoid activation,which is used for binary classification and it outputs a probability value that ranges from zero to one concering the event of a fall.

The adam optimizer which is efifcent and it has the ability to adjust the learning rate is used to compile the model.Having a fixed learning rate of 0.001.Binary cross-entropy loss is used to train this model as it falls under the binary classification problems.

However in our experiment,50 epochs and a batch size of 32 were set.The early stopping callback is used to prevent overfitting and automatically stop training if the model’s performance on the validationset dos not increase with time during training.

### GRU

Similar to LSTM, Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network, but it has a simpler structure with no output gate [17]. Instead, it uses update and reset gates to choose which data should be passed to the output [17]. In this paper, a fall detection system is designed using the GRU architecture, which is quite fitting for sequential data since it manages to capture temporal dependencies very well.

The input to the model is then reshaped according to the dimensions of the training data, representing sequence length and number of features respectively. We configured the GRU layer with 256 units.

Then, we applied a fully connected Dense layer of 128 units and applied the ReLU activation function after the GRU layer. This layer helps in the addition of nonlinearity into the model, therefore allowing the model to pick complicated patterns from the data. To prevent overfitting, a dropout layer has been used with a dropout rate of 0.5. This will randomly set half of the input units to zero during training, helping the model to be more robust and generalizable.

The last layer is a dense layer with one unit and sigmoid activation which is very appropriate for binary classification, as it returns a probability value between zero and one regarding the occurrence of a fall. We are using the Adam optimizer, which is efficient and has an adaptive learning rate. We set a pre-defined learning rate with a value of 0.0001. Also we used the binary cross-entropy loss function and optimized this model for binary classification. As an evaluation metric of model performance, we chose accuracy.

The number of epochs is 50, with a batch size of 32. We used 20% for the validation split to check how well our model worked on data it had not seen before. The early stopping callback will stop training if the model fails to perform better in the validation set in order to prevent overfitting and save time during training.

### BERT

BERT (Bidirectional Encoder Representations from Transformers) is a cutting-edge natural language processing method released by Google in 2018 [18]. The transformer architecture it uses has 12 encoder layers, 8 attention heads, 768 hidden units, 512 hidden units, and 110 million parameters in total [18].

It begins by importing essential libraries such as PyTorch and the Transformers library from Hugging Face. The data is loaded from CSV files into pandas DataFrames, which contain text data and corresponding labels indicating the occurrence of a fall.

Next, a custom dataset class, FallDetectionDataset, is created to handle the dataset's text and label fields. This class leverages a tokenizer to preprocess the text data, converting it into a format that BERT can process. The tokenizer ensures that the text sequences are appropriately padded or truncated to a specified maximum length.

The DistilBERT model is then initialized for sequence classification tasks. This involves setting up the model to output two classes (fall and not fall). The model's architecture is specifically designed to handle classification tasks by leveraging the powerful contextual embeddings provided by BERT.

The training phase involves setting up an optimizer to adjust the model's parameters. A training loop iterates over the dataset in batches, passing the tokenized inputs through the model to obtain predictions. The loss between the predicted labels and actual labels is calculated, and backpropagation is used to update the model's weights.

Evaluation of the model is conducted on a test set to measure its performance. Various metrics such as accuracy, precision, recall, and F1 score are calculated to provide a comprehensive assessment of the model's effectiveness. These metrics help in understanding how well the model can distinguish between fall and non-fall events.

### TFT

Temporal Fusion Transformer (TFT) is a deep neural network based on attention to multi-horizon forecasting [19]. To integrate data from multiple time steps, TFT uses multi-head attention, LSTMs for local time-dependent processing, and Gated Residual Network blocks for effective information flow [19].In this paper, we use two different TFT architectures in our fall detection system to utilize the benefits of both GRU and LSTM units in extracting temporal dependencies needed for an accurate prediction.

The input shape for both models here is (403\*3), which signifies sequence length and the number of features, respectively. The first model has a GRU layer as its backbone, while the second model uses an LSTM layer, both with 256 units. Conversely, GRUs and LSTMs are RNNs that deal well with the vanishing gradient problem and thus are fitting for processing and predicting sequential data. These recurrent layers output sequences, which are then fed into a Multi-Head Attention layer with four heads. Moreover, such an attention mechanism enhances the capability of the model to focus on different parts of the sequence for improving learning.

The Dense layer in both models is implemented to be 256 units with a sigmoid activation to allow the models to control the amount of information that goes through and retain only important features. Then, a Dense layer with 128 units and ReLU activation is added in order to introduce non-linearity that will let it learn complex patterns, followed by a dropout layer of 0.5 to avoid overfitting.

It should be noted that both models use only a single unit with the sigmoid activation function for their Dense layers in the final output layer, making them appropriate for binary classification, since it would output values that correspond to probability values between zero and one related to the likelihood of a fall. For both models, the compilation has been done using the Adam optimizer due to its efficiency and adaptive learning rate. This model has used an initial learning rate of 0.0001. Binary cross-entropy is used as the loss function for the optimization process of the models in binary classification, and the accuracy evaluation metric is used to measure their performance.

The number of epochs will be set up to 50 with a batch size of 512. Here, the validation split will be 20%. No part of the data will be used for training; it will all be saved to test model performance. An early stopping callback is applied to stop the training process whenever the models' performance on the validation set does not improve further, thus avoiding overfitting and reducing train time.

## Experience Designs

Fall is an activity that happens over a period of time, so it is important to figure out how many steps in the past the network should be trained on [1]. The network won't fully capture the scope of the activity if it is trained on a small number of previous steps [1]. On the other hand, the network might consider irrelevant historical accelerometer data if we train it over an excessive number of steps [1].

A 40-point window size was used to process the training and testing dataset to create data instances [20]. This is sufficient time to record the features of a fall and exclude certain actions that could be considered falls [1]. Additionally, we labeled the data instances that were generated by assigning a fall label to each instance that had 25 or more fall label signal points within the window size [20]. If not, the data instance was classified as non-fall [20]. The generated training and testing datasets are displayed in Tables II.

TABLE II

SmartWatch generated dataset description

|  |  |  |  |
| --- | --- | --- | --- |
|  | Fall | Not Fall | Sum |
| Training | 2,912 | 31,068 | 33,980 |
| Testing | 1,456 | 15,733 | 17,189 |
| Sum | 4,368 | 46,801 | 51,169 |

As shown in Table II, the datasets used in the study were highly imbalanced, with significantly more non-fall instances compared to fall instances. Unbalanced class distribution could lead to poor classification results because the deep learning method is skewed or biased toward the dominant class rather than the minority class [20]. To address this, the non-fall data was downsampled to match the number of fall instances in both generated training and testing dataset. As a result, we randomly selected 2,912 non-fall instances out of 31,068 in generated training dataset, and 1,456 non-fall instances of of 15,733 in generated testing dataset. The final training and testing dataset are illustrated in Table III.

TABLE III

SmartWatch final dataset description

|  |  |  |  |
| --- | --- | --- | --- |
|  | Fall | Not Fall | Sum |
| Training | 2,912 | 2,912 | 5,824 |
| Testing | 1,456 | 1,456 | 2,912 |
| Sum | 4,368 | 4,368 | 8,736 |

The generated dataset was used to train and evaluate the performance of four different models: LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), BERT (Bidirectional Encoder Representations from Transformers), and TFT (Temporal Fusion Transformer). In the process of training, 20% of the training data was used for validation, and the model's parameters were initialized at random. 50 epochs of training were conducted with a batch size of 512 and a learning rate of 0.0001. In order to avoid overfitting, training was stopped early when the model's performance on the validation set stopped improving. The models' performance is then assessed using the testing dataset.

Model parameters are represented in Table IV.

TABLE IV

Model Parameters

|  |  |
| --- | --- |
|  | Value |
| learning rate | 0.0001 |
| batch size | 512 |
| validation split ratio | 0.2 |
| number of epochs | 50 |
| Early stopping | monitor='val\_loss', patience=1, restore\_best\_weights=True |

## Environment Setup

Python and Jupyter Notebook is used for implementation in this study. The specific environment setup includes:

* Platform: Tensorflow
* Libraries: numpy, pandas, tqdm, collections, time, pickle, matplotlib, random, sklearn

## Evaluation Metrics

Accuracy, precision, recall, and roc are used to evaluate the performance of the models.

Accuracy is the percentage of correctly classified data [21].

(1)

where FN stands for false negative rate, FP for false positive rate, TN for true negative rate, and TP for true positive rate.

When a binary classification precision score is 100%, all items identified as belonging to the positive class actually do belong to the positive class [21].

(2)

Recall or sensitivity estimates the likelihood of accurately identifying positive cases from those who genuinely have the target condition [21].

(3)

F1-score is the harmonic mean of recall and precision [3].

(4)

where stands for the weighting coefficient

AUC stands for the degree or measure of separability, and ROC is a probability curve [13]. They provide an indication of the model's degree of class distinction [13]. An AUC that is close to 1 indicates that a model is good in terms of separability [reference 1]. The AUC-ROC curve is used to visualize the effectivenes of a model’s classification performance [13].

# RESULTS

Table V illustrates the performance of four models based

on different evaluation metrics.

TABLE V

Model performance comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Evaluation Metrics | LSTM | GRU | TFT-LSTM | TFT-GRU | BERT |
| Accuracy | 0.9107 | 0.9293 | 0.9550 | 0.9354 | 0.6730 |
| Precision | 0.9579 | 0.9503 | 0.9396 | 0.9232 | 0.6795 |
| Recall | 0.8592 | 0.9059 | 0.9725 | 0.9499 | 0.6730 |
| F1-score | 0.9059 | 0.9276 | 0.9558 | 0.9364 | 0.6702 |
| ROC\_AUC | 0.9107 | 0.9293 | 0.9550 | 0.9354 | 0.7349 |

Accuracy of all models is outstanding, which is more than 0.9. GRU and LSTM outperform the others in terms of precision, demonstrating superior efficacy in reducing false positives. With the highest recall, TFT-LSTM is able to record the majority of positive instances.

With its high precision and ROC\_AUC, LSTM model exhibits good overall performance, suggesting its efficacy in classifying and finding positive situations. However, its recall is lower than that of other models, which implies it could miss some positive instances.

GRU model provides higher accuracy together with a well-balanced recall and precision. It performs better overall in categorization than LSTM, outperforming it in terms of accuracy and recall.

With the best accuracy, recall, and F1-score, TFT-LSTM outperforms the others in terms of overall classification performance as well as positive case detection.

The TFT-GRU model exhibits great accuracy and recall, performing similarly to the TFT-LSTM. Despite having the lowest precision of all the models, it still has a high F1-score. This suggests that TFT-GRU achieves a balance between reducing false positives and finding positive cases.

# CONCLUSION

In summary, through this research, we have developed sophisticated deep learning models to evaluate the SmartWatch dataset, improving classification performance of fall detection. Furthermore, we conducted a methodical comparison of LSTM, GRU, BERT, TFT-LSTM, and TFT-GRU to assess each deep learning method's efficiency.

In the future, we would like to apply training set resampling techniques to address the imbalanced classes issue. Instead of depending just on one random downsampling technique, we will run multiple downsampling iterations, then calcualte the average result. This method aims to mitigate bias when selecting dataset for training, leading to increase the reliability of model.

# REFERENCES

[1] T. Mauldin, M. Canby, V. Metsis, A. Ngu, and C. Rivera, “SmartFall: A Smartwatch-Based Fall Detection System Using Deep Learning,” *Sensors*, vol. 18, no. 10, p. 3363, Oct. 2018, doi: https://doi.org/10.3390/s18103363.

[2] M. Musci, D. De Martini, N. Blago, T. Facchinetti, and M. Piastra, “Online Fall Detection Using Recurrent Neural Networks,” *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1276–1289, Jul. 2021, doi: <https://doi.org/10.1109/tetc.2020.3027454>.

[3] M. Farsi, “Application of ensemble RNN deep neural network to the fall detection through IoT environment,” *Alexandria Engineering Journal*, Jul. 2020, doi: <https://doi.org/10.1016/j.aej.2020.06.056>.

[4] Md. M. Islam et al., “Deep Learning Based Systems Developed for Fall Detection: A Review,” *IEEE Access*, vol. 8, pp. 166117–166137, 2020, doi: https://doi.org/10.1109/access.2020.3021943.

[5] Z. Jiang et al., "Fall Detection Systems for Internet of Medical Things Based on Wearable Sensors: A Review," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2024.3421336.

[6] R. Kaur and R. Sharma, “Wearable sensors and datasets for evaluating systems predicting falls and activities of daily living: recent advances and methodology,” *Multimedia Tools and Applications*, Jun. 2024, doi: https://doi.org/10.1007/s11042-024-19504-1.

[7] X. Wu, Y. Zheng, C.-H. Chu, L. Cheng, and J. Kim, “Applying deep learning technology for automatic fall detection using mobile sensors,” *Biomedical Signal Processing and Control*, vol. 72, p. 103355, Feb. 2022, doi: https://doi.org/10.1016/j.bspc.2021.103355.

[8] [8]O. Reyad, H. Shehata, and M. Karar, “Developed Fall Detection of Elderly Patients in Internet of Healthcare Things,” *Computers, Materials & Continua*, vol. 76, no. 2, pp. 1689–1700, 2023, doi: https://doi.org/10.32604/cmc.2023.039084.

[9] T. Theodoridis, V. Solachidis, N. Vretos, and P. Daras, “Human fall detection from acceleration measurements using a recurrent neural network,” *Precision Medicine Powered by pHealth and Connected Health*, pp. 145–149, Nov. 2017. doi:10.1007/978-981-10-7419-6\_25.

[10] C.-B. Lin, Z. Dong, W.-K. Kuan, and Y.-F. Huang, “A Framework for Fall Detection Based on OpenPose Skeleton and LSTM/GRU Models,” *Applied Sciences*, vol. 11, no. 1, p. 329, Dec. 2020, doi: https://doi.org/10.3390/app11010329.

[11] Nader Maray, Anne, J. Ni, M. Debnath, and L. Wang, “Transfer Learning on Small Datasets for Improved Fall Detection,” *Sensors*, vol. 23, no. 3, pp. 1105–1105, Jan. 2023, doi: https://doi.org/10.3390/s23031105.

[12] J. Santoyo-Ramón, E. Casilari, and J. Cano-García, “Analysis of a Smartphone-Based Architecture with Multiple Mobility Sensors for Fall Detection with Supervised Learning,” *Sensors*, vol. 18, no. 4, p. 1155, Apr. 2018, doi: https://doi.org/10.3390/s18041155.

[13] J. Clemente, F. Li, M. Valero, and W. Song, “Smart Seismic Sensing for Indoor Fall Detection, Location, and Notification,” *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 524–532, Feb. 2020, doi: https://doi.org/10.1109/jbhi.2019.2907498.

[14] C. Taramasco et al., “A Novel Monitoring System for Fall Detection in Older People,” *IEEE Access*, vol. 6, pp. 43563–43574, 2018, doi: https://doi.org/10.1109/access.2018.2861331.

[15] A. Lotfi, S. Albawendi, H. Powell, K. Appiah, and C. Langensiepen, “Supporting Independent Living for Older Adults; Employing a Visual Based Fall Detection Through Analysing the Motion and Shape of the Human Body,” *IEEE Access*, vol. 6, pp. 70272–70282, 2018, doi: https://doi.org/10.1109/access.2018.2881237.

[16] T. R. Mauldin, Anne, Vangelis Metsis, M. E. Canby, and Jelena Tesic, “Experimentation and Analysis of Ensemble Deep Learning in IoT Applications,” *The Internet of Things*, vol. 5, no. 1, pp. 133–149, Jan. 2019.

[17] P. T. Yamak, L. Yujian, and P. K. Gadosey, “A comparison between Arima, LSTM, and GRU for time series forecasting,” *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, Dec. 2019. doi:10.1145/3377713.3377722.

[18] H. Ramirez, S. A. Velastin, S. Cuellar, E. Fabregas, and G. Farias, “Bert for activity recognition using sequences of skeleton features and data augmentation with gan,” *Sensors*, vol. 23, no. 3, p. 1400, Jan. 2023. doi:10.3390/s23031400.

[19] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, “Temporal Fusion Transformers for interpretable multi-horizon time series forecasting,” *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021, doi: https://doi.org/10.1016/j.ijforecast.2021.03.012.

[20] M. S. Al-Rakhami et al., “FallDeF5: A Fall Detection Framework Using 5G-Based Deep Gated Recurrent Unit Networks,” *IEEE Access*, vol. 9, pp. 94299–94308, 2021, doi: https://doi.org/10.1109/access.2021.3091838.

‌[21] G. Santos, P. Endo, K. Monteiro, E. Rocha, I. Silva, and T. Lynn, “Accelerometer-Based Human Fall Detection Using Convolutional Neural Networks,” *Sensors*, vol. 19, no. 7, p. 1644, Apr. 2019, doi: https://doi.org/10.3390/s19071644.

‌