FALL DETECTION IN ELDERLY USING GAIT ANALYSIS

Richa Pandya
Student# 1009959779
richa.pandya@mail.utoronto.ca

Anya Pedersen
Student# 1007637504
anya.pedersen@mail.utoronto.ca

Krish Chhajer
Student# 1005678901
krish.chhajer@mail.utoronto.ca

Nicholas Eckhert
Student# 1010037568
nick.eckhert@mail.utoronto.ca

—-Total Pages: 9

1 INTRODUCTION

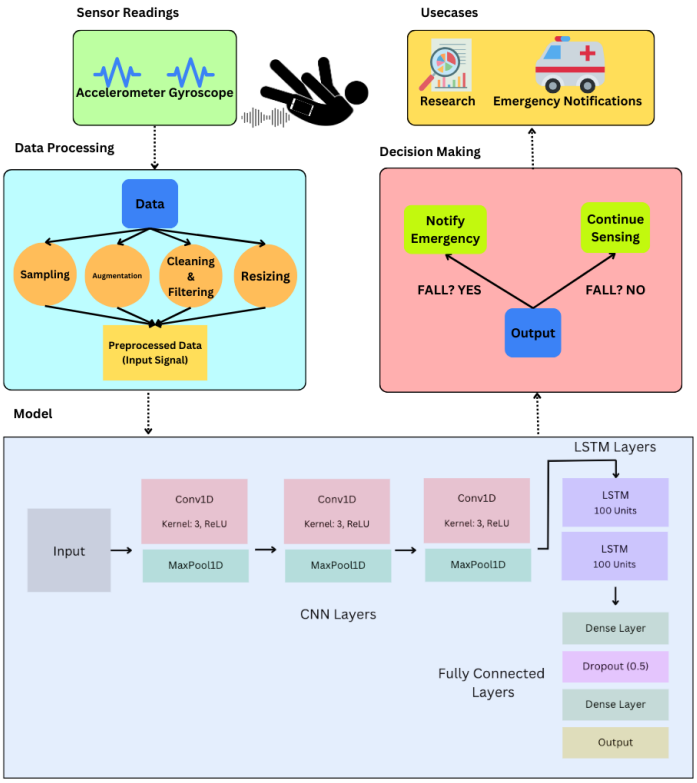


Figure 1: Project Illustration

Falls are the leading cause of injury death in adults aged 65 and over, causing about 38,000 deaths and almost 3 million hospital visits in 2021 (for Disease Control & Prevention, 2024). Through the synthesis of current deep learning techniques and gait (walking patterns) based data from gyroscopes and accelerometers (located in smartphones or other portable technologies), our team proposes a new and viable solution to predicting and preventing elderly falls. Mobile devices can make this data easily accessible- however it is often complex, containing time series data and multiple sensors.

Neural networks can easily extract and recognize patterns, making it a good tool for analyzing the complex sensor data and returning an accurate reading. Additionally, a neural network's ability to learn over time allows it to scale well with large amounts of data that other models may struggle with.

The team proposed using a hybrid neural network architecture of a CNN and LSTM that would receive input from sensor readings (off of the user's phone) and output information that can be used as data for future research as well as issue a notification to nearby emergency responders. The use of a neural network in this way should significantly alleviate the pressure on both the patient and the emergency responders and help fall victims to receive immediate and effective care (illustrated in Figure 1).

This report outlines the various details of this project and showcases results captured through implementing a baseline model and our proposed model.

2 Background and Related Work

Fall detection technology has evolved using deep learning techniques for predicting and preventing falls in the elderly. This section focuses on key advancements in the field, showcasing diverse approaches from different neural network models.

1. **Gait-based person fall prediction using deep learning approach:** Murthy et al. (2021) proposed a deep convolutional neural network (DCNN) model that analyzes gait analysis images to predict falls in persons with walking disabilities. Their DCNN model achieves 99.1% classification accuracy and a 98.64% prediction ratio, outperforming the ResNet50 and CNN methods. The model detected subtle precursors to falls with significant sensitivity

2. **Computer Vision and Machine Learning-Based Gait Pattern Recognition for Flat Fall Prediction:** Chen et al. (2022) used a motion capture system to detect spatiotemporal gait data from 7 healthy subjects walking with normal gait, pelvic obliquity gait, and knee hyperextension gait. They evaluated CNNs, SVMs, KNNs, and LSTM to classify the three gait patterns. The SVM had the highest accuracy of 94.9%.

3. **A smartphone-based online system for fall detection with alert notifications and contextual information of real-life falls:** Harari et al. (2021) developed a smartphone based online system that detects falls with 98.6% precision using accelerometer and gyroscope data. They captured and analyzed 16 unscripted falls from 9 senior adults over two years, providing insights into real life fall dynamics.

4. **Detection of Gait Abnormalities for Fall Risk Assessment Using Wrist-Worn Inertial Sensors and Deep Learning:** Kiprijanovska et al. (2020) proposed a deep learning approach to detect gait abnormalities using data collected from inertial sensors on both wrists. Their LSTM-based model analyzed the inertial data to identify abnormal gait, and achieved 98.1% accuracy.

5. **eNightTrack: Restraint-Free Depth-Camera-Based Surveillance and Alarm System for Fall Prevention Using Deep Learning Tracking:** Mao et al. (2023) introduced eNightTrack, a system that uses depth cameras and deep learning for nighttime fall prevention monitoring. Their convolutional pose machine model accurately tracks 14 key body joints from depth images and an SVM then assesses fall risk based on joint angles and velocities. Their model achieved 95.6% accurately without privacy concerns from RGB video.

6. **Fall Detection with CNN-Casual LSTM Network:** Wu et al. (2021) proposed a CNN-Casual LSTM network for fall detection using wearable sensor data from the SisFall Dataset (Sucerquia et al., 2017). Their model uses a CNN to extract spatial features from the sensor, and a Casual LSTM to capture the temporal dependencies. Their model achieved 99.79% accuracy, 100% sensitivity, and 99.73% specificity.

3 DATA PROCESSING

The model has been trained on the SisFall Dataset (Angela Sucerquia & Vargas-Bonilla, 2017), which contains accelerometer (ADXL345 and MMA8451Q) and gyroscope (ITG3200) data from a sensor attached to the subject's waistline. The data consists of 19 different types of ADL and 15 different types of falls that are sampled at 200 Hz and stored in the form of text files with the x, y and z directional components of all sensors. The procedure given below has been utilized to clean, process and extract all training data.

3.1 DATA CLEANING AND EXTRACTION FROM TEXT FILES

Gait_Analysis_Preprocessing.ipynb is a python script that reads the text files and uses string manipulation to extract sensor readings. It also converts sensor readings from bits to standard units and stores it in individual csv files (Figure 2) inside the Preprocessed Data folder in Google Drive.

	Timestamp	X_Acc_1	Y_Acc_1	Z_Acc_1	X_Gyro	Y_Gyro	Z_Gyro	X_Acc_2	Y_Acc_2	Z_Acc_2
0	0.000	-0.344883	-9.848320	-0.958008	5.126953	15.075684	1.647949	-1.149609	-9.455537	0.603545
1	0.005	-0.114961	-10.078242	-0.881367	6.042480	15.747070	2.136230	-1.053809	-9.733359	0.651445
2	0.010	-0.038320	-10.346484	-0.843047	6.958008	16.601562	2.746582	-0.900527	-9.934541	0.661025
3	0.015	0.038320	-10.614727	-0.919687	7.751465	17.456055	3.479004	-0.775986	-10.174043	0.661025
4	0.020	0.076641	-10.768008	-0.958008	8.178711	18.493652	4.272461	-0.680186	-10.336904	0.603545
...	...	...	...	...	...	...	...	...	...	...
2995	14.975	-4.521797	2.567461	-9.618398	-2.868652	0.976562	0.000000	-5.230723	2.826123	-7.922725
2996	14.980	-4.521797	2.644102	-9.465117	-2.746582	0.976562	0.000000	-5.259463	2.797383	-7.874824
2997	14.985	-4.521797	2.490820	-9.541758	-2.807617	0.976562	0.000000	-5.201982	2.797383	-7.922725
2998	14.990	-4.483477	2.605781	-9.656719	-2.868652	1.098633	0.183105	-5.259463	2.835703	-7.884404
2999	14.995	-4.291875	2.529141	-9.426797	-2.929688	1.098633	0.244141	-5.182822	2.787803	-7.884404

3000 rows × 10 columns

Figure 2: Sample Fall CSV

3.2 WINDOW SAMPLING

A 15 second data sample has been window sampled into 14 windows such that the window sizes are 2 seconds (400 samples @ 200 Hz) with a 1 second overlap (200 samples @ 200 Hz). This was selected carefully as previous work suggests that windows can capture temporal dependencies easily. However, as different activities have different lengths, varying techniques were applied prior to window sampling based on the type of activity as follows: (For proof of reasoning, check Data_visualization_v1.ipynb and for code check Window Sampling and Feature Extraction + CNN Layers.ipynb)

- **Fall Activities:** Were not processed. 15 seconds selected as the baseline for all samples. If the fall sample had one less reading, the last reading was copied at the end to keep the sample size consistent.
- **ADLs (1-4):** Continuous activities such as Jogging and Walking of time lengths of 100 seconds are separated into five 15 second samples and processed separately.
- **ADLs (5-6):** These activities are 25 seconds in length and capture movement going up and down the stairs. Since the separation between the two movements occurs at the half mark for most samples, the data has been split and processed such that the first 15 seconds capture one movement and the last 15 seconds capture the other movement.
- **ADLs (7-19):** Other ADL activities of 12 seconds in length have important movements captured within the first 10 seconds, with no activity recorded in the last few seconds. Thus, the length of these samples is increased to 15 seconds using forward fill.

3.3 Feature Extraction

The team utilized window sampling to down sample our data. However, to ensure minimal loss of information, 56 graph, time-domain and frequency-domain features were extracted for both accelerometer and gyroscope data in all three directions for each window. The features (**Mean**, **Median**, **Max**, **Min**, **Standard Deviation**, **Inter-Quartile Range**, **Jerk**, **Skewness**, **Kurtosis** , **Energy**, **Covariance**) were referenced from the UCI HAR Dataset (D. Anguita, 2013). Only readings from the ADXL345 accelerometer were used as the model is only expected to work with one set of accelerometer readings.

3.4 Processing and Merging

The last step involves processing all files in the "Preprocessed Folder" through the window sampling above and saving it in the "Processed Folder" at drive. Each new file (sample in Figure 3) now contains 14 windows and 56 features representing a trial data sample. The data is merged into a 3D numpy array while the labels (1 for Fall and 0 for ADL) were converted into a numpy array. This resulted in 5408 total samples (Figure 4) with 1768 Fall samples and 3610 ADL samples.

	Window	X_Acc_Mean	X_Acc_Max	X_Acc_Std	X_Acc_Min	X_Acc_IQR	X_Acc_Skew	X_Acc_Kurtosis	X_Acc_Energy	X_Acc_Jerk	...	Z_Acc_Mean.1	Z_Gyro_Max
0	1	-0.011209	3.333867	1.180099	-3.640430	1.072969	-0.629072	1.373224	1.389279	-0.210762	...	2.385254	76.538086
1	2	-0.016095	2.912344	1.311070	-4.521797	1.504072	-0.796797	1.086280	1.714866	-2.126777	...	0.439453	76.538086
2	3	0.005940	3.832031	1.455470	-4.521797	1.283730	-0.396738	1.146642	2.113131	0.191602	...	-1.145477	76.782227
3	4	0.027878	5.479805	1.808183	-4.866680	1.159189	-0.184764	1.208545	3.262128	1.571133	...	0.426636	81.848145
4	5	-0.124828	5.479805	1.705863	-4.866680	1.187930	-0.206821	1.827534	2.918276	0.076641	...	-0.218506	81.848145
5	6	0.035542	10.691367	2.354291	-8.660391	1.772314	1.261404	5.582205	5.530092	-5.633086	...	0.365601	80.444336
6	7	-3.565322	44.374922	8.112413	-42.803789	6.706055	-0.532750	9.346877	78.358236	-2.912344	...	12.981110	551.940918
7	8	-6.369602	44.374922	6.893272	-42.803789	1.005908	0.122249	19.432383	87.970238	3.391348	...	12.766876	551.940918
8	9	-5.147280	-4.636758	0.249330	-5.824688	0.421523	-0.808778	-0.436576	26.556503	0.479004	...	0.683594	3.662109
9	10	-4.929333	-4.560117	0.085077	-5.173242	0.114961	0.038899	1.295277	24.305548	0.153281	...	0.239258	2.319336
10	11	-4.820600	-4.483477	0.110448	-5.096602	0.153281	0.417441	-0.014434	23.250348	0.134121	...	0.081177	3.112793
11	12	-4.645284	-4.330195	0.135750	-5.019961	0.229922	-0.093164	-0.870819	21.597046	0.134121	...	0.210876	3.112793
12	13	-4.521893	-4.176914	0.083158	-4.790039	0.076641	-0.153532	0.986279	20.454411	0.057480	...	0.058594	1.098633

Figure 3: Fall Sample after Window Sampling and Feature Extraction

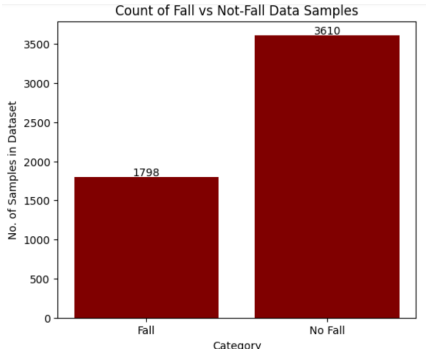


Figure 4: Final Number of Processed Samples

4 Architecture

The fall detection system deploys a hybrid CNN-LSTM model (Figure 5) that leverages the strength of both convolutional neural networks and long short-term memory networks, and is designed to efficiently process and analyze complex temporal sensor data.

After thorough data preprocessing (segmenting raw inertial sensor data into fixed-length sequences and applying normalization and noise reduction techniques), the model accepts a 3D tensor with dimensions (sample number, time step, and feature type). This data first gets passed through three convolutional layers that each increase in complexity to detect increasingly intricate features: the

first layer has 64 filters, the second layer has 128 filters, and the final layer has 256 filters. Each layer is followed by a ReLU activation function and max pooling (pool size of 2). The output from the CNN then gets passed into two LSTM layers - each with 100 units. After this, the output is sent to a dense layer with 100 units and ReLU activation, a dropout layer with rate of 0.5, and the output layer is 1 unit with a sigmoid activation function to produce the accurate output (for code check training_v2.ipynb

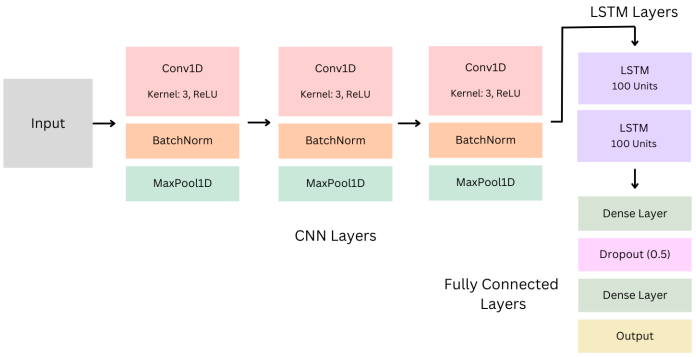


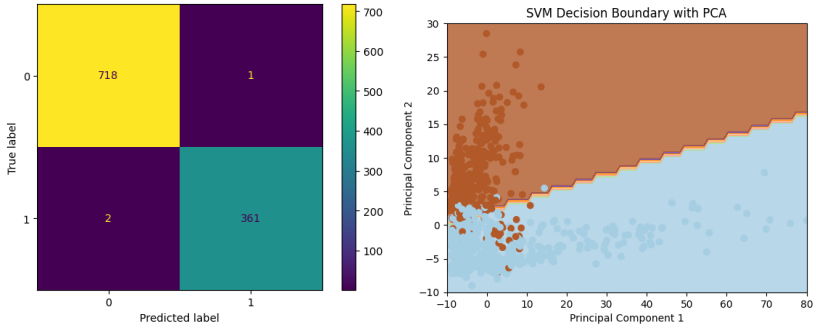
Figure 5: CNN LSTM Model Architecture

5 Baseline Model

To establish a baseline to compare our model, the team utilized a Support Vector Machine (SVM) with a linear kernel (implemented in SVM.ipynb) to classify our data. The SVM finds a hyper plane that maximizes the margin between the two nearest points of each label (in binary classification), fits the data, and then creates a decision boundary to classify based on the chosen hyperplane.

Since the processed data was three-dimensional and SVMs only accept two-dimensional data, the team decided to convert the time series windows and features into columns for each labeled event. This resulted in a final 5408 x 771 dataframe. The team then removed the numbering for each sample, shuffled the data to remove any potential biases, and normalized it using the Standard Scaler from the Sci-Kit Learn (SKLearn) library.

Using SKLearn's split_test_train function, the team divided the data into 80% training data and 20-percent testing. For the SVM model we chose to use SKLearn's Support Vector Classifier, and fit the model to the training data. The model was able to receive an overall testing accuracy of 99.72-percent (0.28-percent error) on the remaining data by correctly classifying 718 out of 720 ADLs and 361 out of 362 falls (Figure 6).



(a) Confusion Matrix (b) Loss

Figure 6: SVM Results

6 Quantitative and Qualitative Results

Accuracy, Sensitivity and Specificity have been chosen for evaluating our model. Accuracy gives a measure of the overall correctness of the model (percentage of all samples classified correctly) and is necessary as the team aims to have the model classify all samples correctly. Sensitivity is the percentage of Positive (Fall) Samples classified correctly. As Fall Detection is an extremely sensitive topic and having undetected Falls (False Negatives) causing loss of human lives, this metric will measure how accurately our model can detect Falls. Specificity is a measure of the percentage of Negative (ADL) Samples classified correctly. Since False classification of ADLs as Falls (False Positives) will disrupt user experience, this metric will measure our model's capacity in accurately classifying ADLs.

The SisFall Dataset has been utilized for training the model. The model has been trained on a batch size of 64, for 30 epochs with a learning rate of 0.001 using the Adam optimizer. The SisFall Dataset was split into a 60-20-20 ratio for training, validation and test respectively. The model achieved a 99.2% training, 99.1% validation and 99.2% test accuracy, highlighting the model's effectiveness in classification between Fall and ADLs on SisFall Dataset. The model also achieves a Sensitivity of 98.3% and Specificity of 99.6% across the entire SisFall Dataset (for code check Quantitative Results.ipynb).

Due to a high quantitative performance of this model on the SisFall dataset (Angela Sucerquia & Vargas-Bonilla, 2017), drawing accurate qualitative results is difficult. The final model was selected by assessing the model performance on data collected on phones by the team, and qualitative results were drawn on the MobiFall dataset (BMI) instead. See sections 7.2.1 for more details.

7 Evaluate Model on New Data

Given that most alternative models achieved high accuracies of over 98% on the SisFall Dataset (Angela Sucerquia & Vargas-Bonilla, 2017), the team explored two sources of new data - collecting raw data from the Sensor Logger ((Choi)) app on iPhone to determine the best model before testing on the MobiFall dataset (BMI).

7.1 Data from Sensor Logger App

The app was set to record accelerometer and gyroscope readings at 10 Hz and to forward fill any missing data. The team placed the phone in each of the subject's pockets and then monitored as the subject re-enacted each ADL and fall.

Preprocessing included aligning the axes of the data as per the SisFall Dataset and the gyroscope readings were converted from radians to degrees. Window Sampling and Feature Extraction as described above were applied to the data and the samples were either shortened or forward filled to achieve 14 windows per sample. In total, the team recorded 45 ADLs and 52 falls.

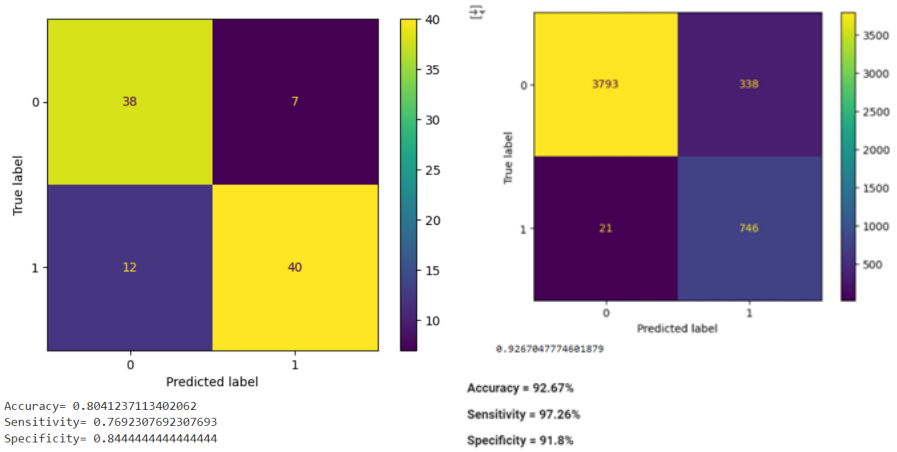
The proposed model performed most effectively, achieving an accuracy of 80.5%. Sensitivity of Phone Data stood at 77% whereas Specificity was 84.4% across the entire collected data (Figure 7).

7.2 MobiFall and MobiAct Datasets

The app was set to record accelerometer and gyroscope readings at 10 Hz and to forward fill any missing data. The team placed the phone in each of the subject's pockets and then monitored as the subject re-enacted each ADL and fall.

Preprocessing included aligning the axes of the data as per the SisFall Dataset and the gyroscope readings were converted from radians to degrees. Window Sampling and Feature Extraction as described above were applied to the data and the samples were either shortened or forward filled to achieve 14 windows per sample. In total, the team recorded 45 ADLs and 52 falls.

The proposed model performed most effectively, achieving an accuracy of 80.5%. Sensitivity of Phone Data stood at 77% whereas Specificity was 84.4% across the entire collected data (Figure 7).



(a) Phone Collected Data (b) MobiAct and MobiFall

Figure 7: Quantitative Results

Like SisFall, the samples in the MobiFall dataset are further divided into different types of non-fall actions and different types of falls. To assess where and how our model fails we evaluated the accuracy for each subject and each activity performed (Figure 8).

This difference from the mean accuracy was assessed using the Kruskal-Wallis Test. This test is applicable given error was non-normal and variance differed for each group, yet each accuracy measurement was obtained from a unique sample so independence is assumed. A significant difference was seen in the mean accuracies for each group (p value = 2.2e-16). The primary source for Type I errors is the action of getting into cars and sitting (encoded as CSI and SIT for MobiAct).

No significant difference was found (p value = 0.9994) or the mean accuracy for each subject group. This indicates that it is the movement patterns that the model is using to detect falls, and this is recognizable across many different subjects (Check Statistical tests for code).

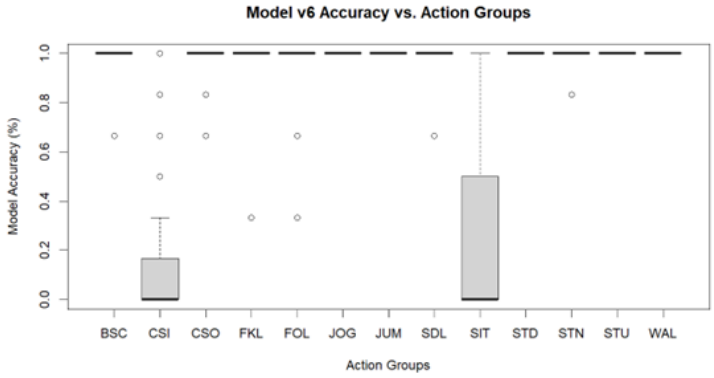


Figure 8: MobiAct and MobiFall Qualitative Results

8 Real Life Implementation

To bridge the gap between theoretical model development and practical application, two avenues for real life implementation were explored: a smartphone app and a Raspberry Pi-based wearable device. These prototypes aim to show the feasibility of deploying the model in real life scenarios.

8.1 Smartphone App Implementation

The prototype Android app for fall detection (check the code in this repository) utilizes the phone's accelerometer and gyroscope and records sensor data at 100 Hz for 15 seconds. Afterwards, basic processing is applied to the data in order to convert it to a form acceptable for our model. The app then can pass the recorded data into a model for fall/no fall classification. However, due to time constraints, the team was unable to fully enact our model within the app. Instead, as a placeholder, we utilized a basic classifier based upon achieving acceleration and rotation thresholds. After each 15 second recording, the app displays its prediction and begins recording the next window. If a fall is detected, the loop is terminated and the app displays the fall prediction indefinitely.

This prototype gives the team a blueprint for a real world implementation of our model on a widely accessible platform. By enacting our actual model and implementing an alert system upon detection of a fall, the app would become fully capable of serving as a tool to help fall victims receive effective and immediate care.

8.2 Raspberry Pi Implementation

While a phone-based implementation could be successful, phone placement variability could introduce excessive noise and cause the model to produce inaccurate results, which led the team to explore a physical wearable solution.

This involves adapting the model for deployment on a Raspberry Pi 3B, and using an MPU6050 sensor (6 axis gyroscope and accelerometer). The Raspberry Pi and MPU6050 sensor communicate via the I2C protocol. The PyTorch, Adafruit CircuitPython MPU6050 and Twilio libraries were installed to handle data processing and alert notifications.

This implementation involves a python script that samples gyroscope and accelerometer data from the MPU6050 at 200 Hz, maintaining a rolling window of the 14 most recent readings. For each window, the same features as the model are extracted, and the preprocessed data is fed through the CNN-LSTM neural network to determine if a fall has occurred. If the system detects a fall, it triggers an alert using the Twilio API to send an SMS notification to a designated phone number. While full testing was constrained by time and hardware constraints, the necessary code and architecture has been developed and can be accessed in the Github.

Both implementations showcase two distinct approaches to deploying this model in real-world scenarios and show the potential for a fall detection system to become incorporated into the lifestyles of individuals.

9 Discussion

Developing this model revealed important insights and challenges in applying deep learning to real world problems. This section examines the model performance, key learnings from the process, and assesses complexity.

10 Model Performance

The hybrid CNN-LSTM architecture demonstrated strong performance on the SisFall dataset - but it faced challenges with out-of-distribution data, especially with real world falls that often occur in unpredictable ways (slips, trips, health-related falls) that may not have strong correlations with the individuals' gait patterns, could lead to missed detections in actual emergencies. The model could also falsely classify non-fall events that resemble falls (quickly sitting down or getting in/ out of cars) due to the datasets' limited range of "normal" activities. Despite this, the model maintained accuracy across different subjects and showed that it is able to effectively filter individual gait patterns (See section 8.2.2).

10.1 Insights from Model Development

During training, the team tested several alternate models with different hyperparameters (batch size, learning rate, epochs) and layers (Reduced Feature Depth, Batch Normalization Layers, number of CNN/ LSTM layers.

Most models had an accuracy of 98%+ on the test split of the SisFall dataset, which made it difficult to identify the best model configuration. Therefore, the team collected raw data on the Sensor Logger App and an Iphone and results from this data were used to select the final model which was then tested on unseen MobiFall and MobiAct Datasets (see section 8.2.2).

Initially there were BatchNorm layers in all the models which achieved a high accuracy on the SisFall test set, however this made the mode extremely susceptible to changes in batch sizes and the composition of the batch inputs. Even utilizing MinMaxScaling in several ways did not achieve good results on Phone Collected Data. The final proposed model includes no form of normalization anywhere in the architecture of data processing and achieves the best set of results.

10.2 Project Complexity

This project presented significant challenges and pushed the team's understanding of deep learning beyond APS360 course material. The complexity arose not only due to the model's architecture design, but also from the intricacies of real-world data processes and the nuances of fall detection as a focus area.

When deploying the model on raw data, the team encountered data inconsistency across sources, the impact of normalization techniques on model generalization, hardware integration, and the challenges of preparing a model for real world deployment. These experiences provided valuable insights into the gap between theoretical model development and practical application in the real world.

11 Ethical Considerations

While the proposed fall detection system offers significant potential in enhancing elderly care, it is crucial to critically examine the ethical implications and limitations. Concerns in this system include:

1. **Privacy concerns:** continuous monitoring of individuals' movements raises data anonymity issues
2. **Data bias:** datasets primarily feature younger adults (20-47 years), potentially misrepresenting elderly gait patterns and leading to algorithmic bias
3. **Lack of diversity:** datasets often focus on participants from European and North American regions, overlooking gait variations that are correlated with lifestyle, terrain, and disabilities.
4. **Practical constraints:** this model assumes consistent placement of the sensor (smartphone, wearable, etc) which may not accurately reflect the variation that occurs in real world usage.

While this fall detection system shows promise, its deployment demands careful ethical consideration and navigation. Addressing these concerns requires diversifying datasets, refining models for real world variability and conducting extensive testing with a diverse set of individuals.

12 Link to Github

- **GitHub repository:** `https://github.com/rpandya5/gaitanalysis`

REFERENCES

The mobifall and mobiact datasets — bmi. URL https://bmi.hmu.gr/the-mobifall-and-mobiact-datasets-2/.

José David López Angela Sucerquia and Jesús Francisco Vargas-Bonilla. Sisfall: A fall and movement dataset. Technical report, Universidad de Antiquia, 2017.

B. Chen et al. Computer vision and machine learning-based gait pattern recognition for flat fall prediction. *Sensors*, 22(20):7960, Oct 2022. doi: 10.3390/s22207960.

Kelvin Choi. Android sensor logger app. URL https://www.tszheichoi.com/sensorlogger.

L. Oneto X. Parra Jorge Luis Reyes-Ortiz D. Anguita, A. Ghio. Human activity recognition using smartphones. Technical report, UC Irvine Machine Learning Repository, 2013.

Centers for Disease Control and Prevention. Older adult falls, 2024. URL https://www.cdc.gov/falls/about/index.html#:~:text=Key%20points,injury%20death%20for%20that%20group.&text=In%202021%2C%20emergency%20departments%20recorded,visits%20for%20older%20adult%20falls.

Y. Harari et al. A smartphone-based online system for fall detection with alert notifications and contextual information of real-life falls. *Journal of NeuroEngineering and Rehabilitation*, 18(1), Aug 2021. doi: 10.1186/s12984-021-00918-z.

I. Kiprijanovska, H. Gjoreski, and M. Gams. Detection of gait abnormalities for fall risk assessment using wrist-worn inertial sensors and deep learning. *Sensors*, 20(18):5373, Sep 2020. doi: 10.3390/s20185373.

Y.-J. Mao, A. Y.-C. Tam, Q. T.-K. Shea, Y.-P. Zheng, and J. C.-W. Cheung. Enighttrack: Restraint-free depth-camera-based surveillance and alarm system for fall prevention using deep learning tracking. *Algorithms*, 16(10):477, Oct 2023. doi: 10.3390/a16100477.

A. Sampath Dakshina Murthy, T. Karthikeyan, and R. Vinoth Kanna. Gait-based person fall prediction using deep learning approach. *Soft Computing*, 26(23):12933–12941, Sep 2021. doi: 10.1007/s00500-021-06125-1.

A. Sucerquia, J. López, and J. Vargas-Bonilla. Sisfall: A fall and movement dataset. *Sensors*, 17 (12):198, Jan 2017. doi: 10.3390/s17010198.

J. Wu, J. Wang, A. Zhan, and C. Wu. Fall detection with cnn-casual lstm network. *Information*, 12 (10):403, 2021.