ML Model for Fall Detection- Team 39

Shivam Sood, Joseph John Cherukara, Aarushi Mittal, Priyansh Khunger

Motivation

Construction workers and such at times during their work may fall from heights leading to injuries and accidents which may be treated properly if given care at early stages itself. But this does not happen most of the times since the injured cannot move and others may notice them only at a later stage. These kinds of accidents can be dealt with earlier itself if we get to implement a fall detection system for these workers and thus lead to safer environments where in case a worker does fall, he can be administered care or taken to the hospital very early itself due to the notice

Block Diagram

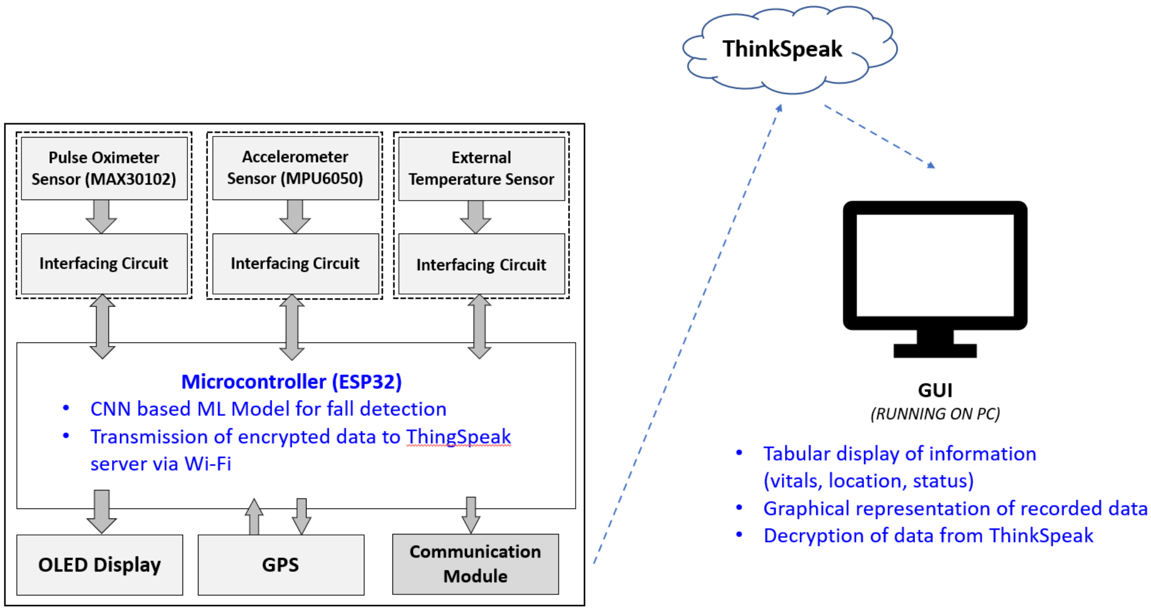


Figure 1: Block Diagram with sensor specs

Figure 2: Setup of microcontroller, sensors and battery

Methodology

For testing the setup, we drop the setup from various heights, and keep collecting the data and save it as fall files. Then we keep the setup with us while doing normal daily activities like exercise, walking, running, jumping etc. All these files are saved as ADL files. Then these files are fed to the ML model for training and testing. The ML model chosen was CNN since it showed the highest accuracy when compared to other models such as SVM, KNN, XGB etc. Another reason for choosing CNN was that it could be modified to use lesser number of convoluted layers and dense layers without affecting the accuracy of the model much so as to be lightweight enough to run on the ESP32 board.

The structure of the CNN model used is one convolutional layer of ten neurons, with relu activation, followed by one dense layer of two neurons with softmax activation. The CNN layer currently has kernel size of (50,6) and stride is 5. This gives approx 4500 parameters.

Deployment Campaign

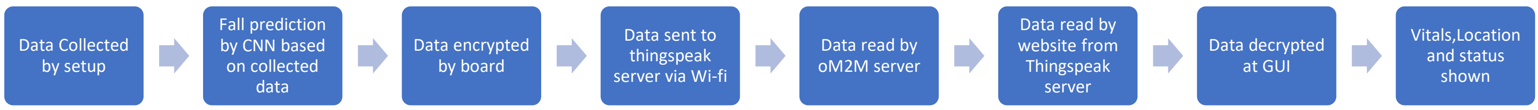
The setup will be installed in a 3d printed box containing the power source, communication model and complete setup. This box will have a trigger button and a touch sensor. The vitals are collected when the user keeps holding the sensor. The trigger can be used to send/refresh the fall status that is being transmitted to the central system. This box is worn at the waist along with the belt. Thus all the workers using this system will carry the box at the waist level at all times while working and all the workers data can be seen on the central GUI. When a worker falls from a height the status of the worker changes in the GUI and a notification appears which has the user details, location and fall time.

Data Validation

The dataset we collected consisted of 30 Fall data points and 9000 ADL data points. Thus the sensitivity of the model was less than 0.5. To rectify this we used SMOTE [Synthetic Minority Oversampling Technique] for fixing the class imbalance by increasing the number of fall data points. This is used along with random under sampling to reduce the number of ADL data points. Thus the new dataset has 1000 Fall data points and 2000 ADL data points. The sensitivity of the model now was 1.0. This was for the training dataset.

We will check the output of the model with the files we have collected and see if it predicted the fall status accurately or not. With this we found the accuracy to be 99.7% in predicting fall.

Data Flow Pipeline



Communication to ThingSpeak and oM2M

The data that is being collected by the setup, including the vitals, location and fall status, is being stored in the sd card and also sent to the ThingSpeak server via Wi-Fi. This data is then read by the GUI from the ThingSpeak server. It is also read by the om2m server from the ThingSpeak server.

Development of dashboard

For the dashboard/GUI we used various technologies:

- Firebase real-time database - for storing the user data and collected data in a database
- Cryptojs - for decryption of AES
- Chartjs - for graphs
- Fontawesome - for icons
- jQuery - for general stuff

With these our GUI can register a new user which will collect the user info such as name, age, designation and their ThingSpeak url. [We have made the system such that each user/setup-node will be linked to one ThingSpeak channel with the respective fields for vitals, location and fall status. Thus the url is the Read API for that particular user's ThingSpeak channel. Once a new user is registered, his/her vitals, location and status can be seen on the GUI. On hovering near the user-id, we can see the registered info of the user. The data is refreshed every 15 sec. When a fall is detected, the status of that user changes to fall (either ADL/Sam-floor fall or regular fall from a height) and a notification with timestamp pops up. This notification contains the user id, location, fall status and time. On clicking the location we can see the Google Maps location of the user. The status of the user will not change to ADL(default) unless and until the user himself/herself presses the trigger manually to reset the status. This is to keep a system to show that the worker is alright and not in need of medical help at the moment. But if the status does not change even after a while, then that means the worker is unconscious or not physically able to change the status which means they require medical help at the earliest.

Use of IoT security tools in your project

For security, we used the AES encryption library. Before sending the collected data and predicted fall status to ThingSpeak server, we encrypt the data on the ESP32 board itself using the AES library. Thus the data that we see on the ThingSpeak server and oM2M server are the encrypted data values. The decryption only takes place at the GUI/website. We have included the decryption command for the AES library in the javascript of the GUI.

Data visualization and analysis

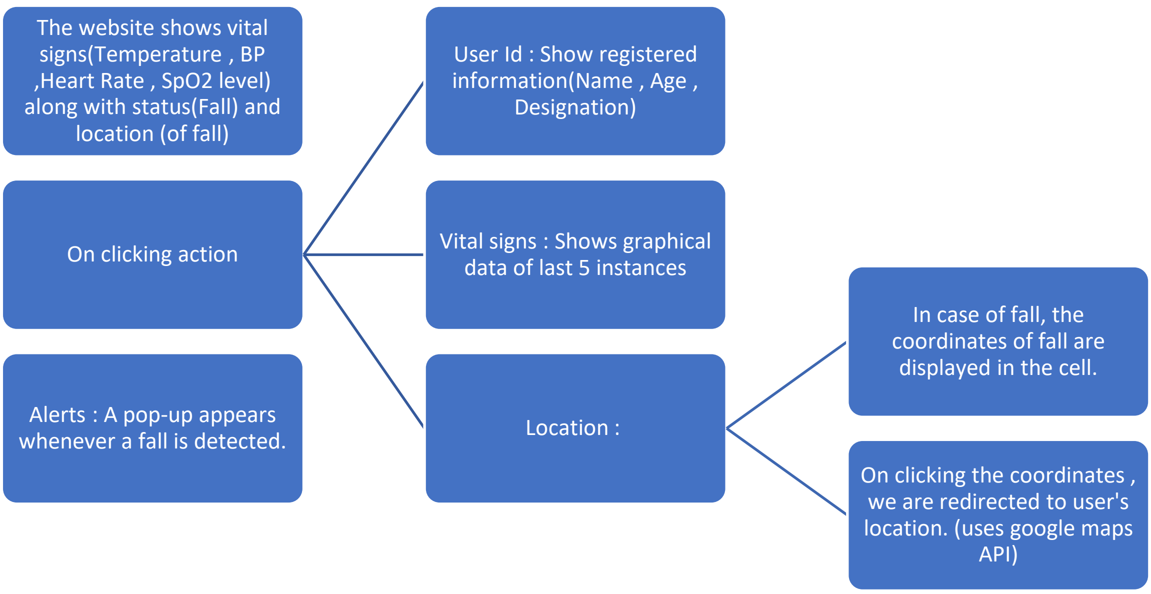


Figure 3: Working of GUI

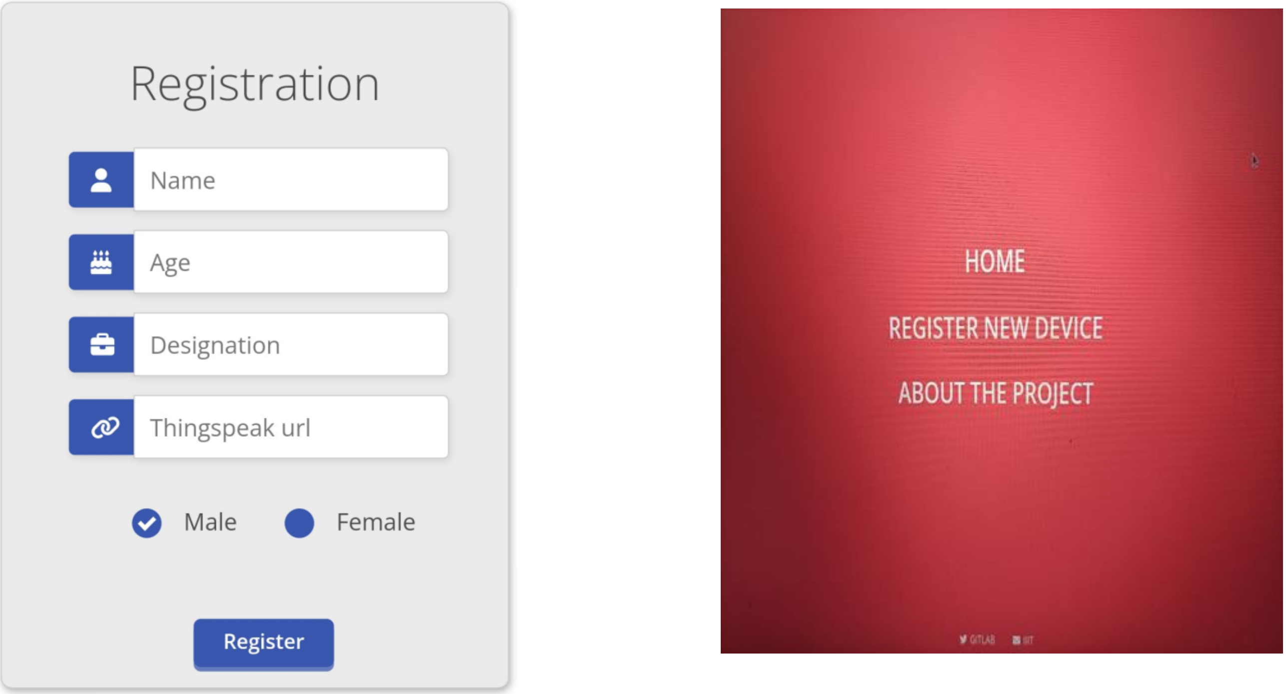


Figure 4: Registration Page and Menu of GUI

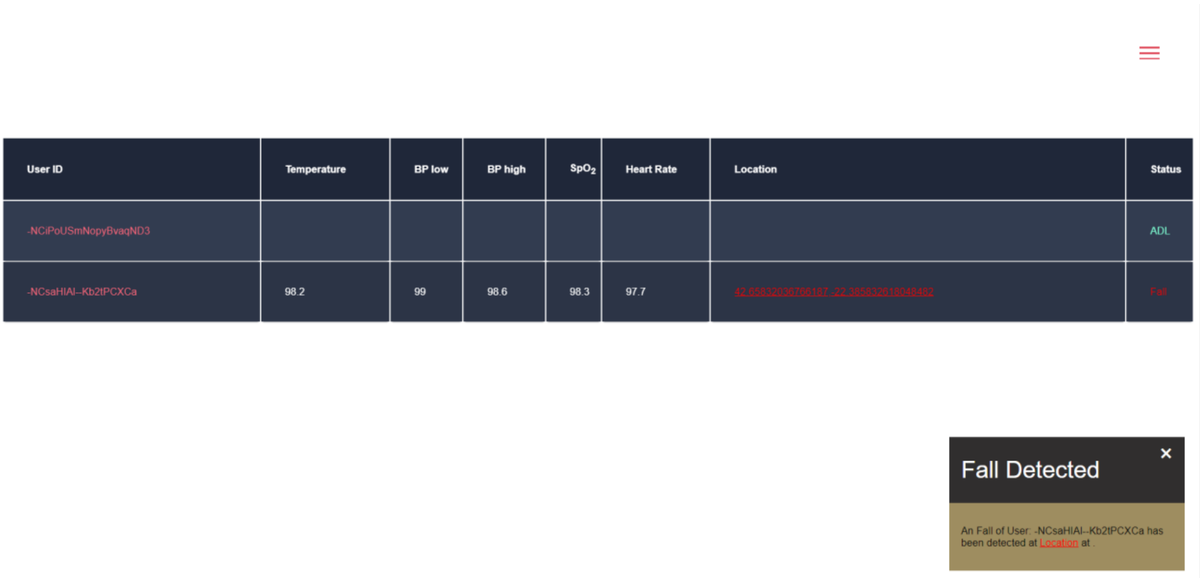


Figure 5: GUI Home Page with users and their details with fall notification of one user

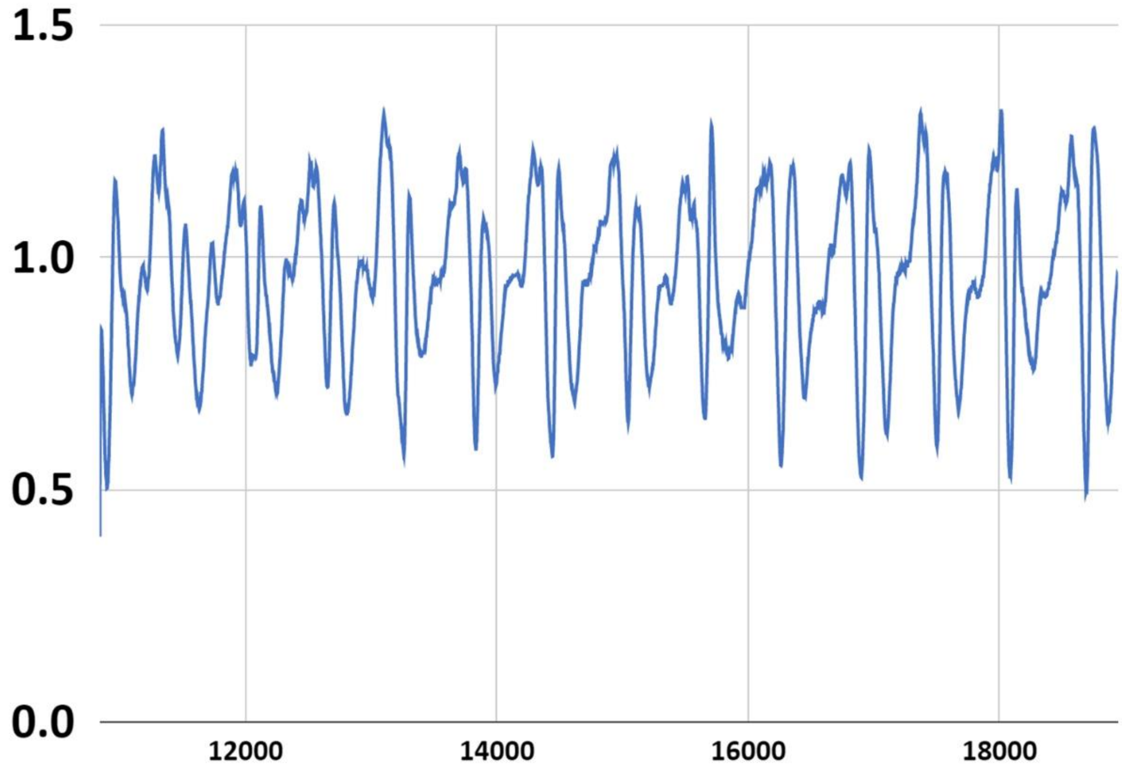


Figure 6: MAG vs Time for walking

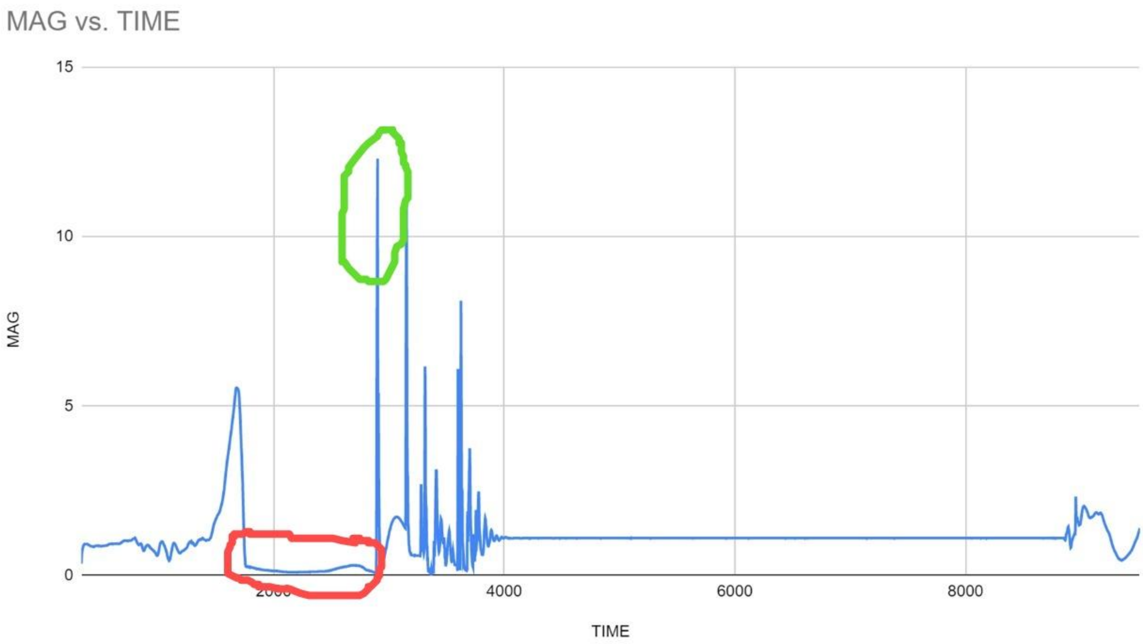


Figure 7: MAG vs Time for freefall (red part indicated freefall and peak is after impact)

- From the fall status, type of fall, and vitals we can predict whether the user is unconscious or not
- By collecting the location and fall type we can also infer what all are the accident prone areas from where a person is likely to fall from a height and thus ensure more security measures are taken in those locations

Final results and conclusion

The end product of this project was a working system which has the hardware setup to collect the vitals, location accelerometer and gyroscope data, use that data to predict whether the user has fallen or not using a CNN model and then view that output on a central GUI by reading the values from the ThingSpeak server to which the setup send the data to. Thus the GUI will help the authorities to know when and where a person has fallen and if he/she requires medical help.

- The fall prediction CNN model outputs the probability values of it being a fall or an ADL
- From these probability values we predict if the user has fallen or not
- This prediction is then sent to ThingSpeak, and then shown on the GUI as the status of user

We hope that this system can be deployed in real world industries and is helpful for the workers to receive help immediately if they face a fall.

Final code base (can attach links to git repo)

https://gitlab.com/shivam-sood/esw-39/ - For the complete codes

http://shivam-sood.github.io/ - GUI