WiFi-CSI-based Fall Detection on Raspberry Pi

Sam Desai, Nishant Dash, Kristine McLaughlin
{desaisam, ndash, ksmcl}@umich.edu
University of Michigan

Project Statement

For many older adults in independent living environments, falling presents a critical risk to their health and well-being. Most current solutions for fall detection systems include wearable devices, which can be uncomfortable and/or unreliable with user involvement. Many non-wearable approaches include vision-based models, which can falter if falls occur outside the observed area and also introduce privacy concerns [1]. We propose an extension to Nakamura et al.'s work on a WiFi-based approach to fall detection, which is minimally invasive and has achieved high classification performance when Channel State Information (CSI) data is converted into spectrograms and run through a Convolutional Neural Network (CNN) [2]. Our work aims to investigate whether their techniques can produce similar positive results using limited hardware, specifically on a Raspberry Pi. We create our own dataset of spectrograms to finetune a lightweight pretrained CNN and evaluate its performance in two different environments.

Embedded Systems Component

CSI data in the original paper was collected on a laptop with an Intel WiFi chipset [2]. To recreate this experiment with limited hardware, we use a Raspberry Pi 4B (RPi) to collect CSI data and run the CNN's inference, which uses a processor much weaker than most laptops. The RPi uses the Broadcom bcm43455c0 WiFi chipset to use the Internet and collect CSI data. This chipset has only one antenna, so it collects less CSI data than the Intel chipset used in the paper, which has 4 antenna pairs [2].

In order to collect CSI data on the RPi, we used the Nexmon-CSI library [3]. This library uses Nexmon, a firmware patching tool for Broadcom chipsets (such as those on the RPi), to allow users to access CSI data that the RPi receives [4]. Nexmon-CSI allows users to specify a WiFi channel and MAC address, and will capture CSI data for all traffic received on that channel from the specified MAC address.

For the RPi to receive CSI packets, we place it in an open area around a laptop and router. The laptop pings the router at 1KHz, and Nexmon-CSI on the RPi is configured to receive CSI data from these packets. If a fall occurs between the laptop and router, the disturbance in the air will be reflected in the CSI data captured by the RPi. Through this experiment, we found that some routers were unsuitable for use with this method. Some routers had rate limiting that did not allow for pinging at 1KHz, while others used channels that were too high for the Nexmon firmware to collect CSI data from. We used an Xfinity WiFi Gateway as the router to address both of these issues.

Machine Learning Component

Spectrogram Creation
We use the same spectrogram processing techniques described in the original reference paper [2]. CSI data is collected by collecting packets in the air. When pinging the router from our laptop, many of those packets may get dropped or delayed, so we must perform interpolation to ensure the CSI data is recorded at a consistent 1KHz. After interpolation, we perform Principal Component Analysis (PCA) to get the highest variance data. Processing the whole CSI data is expensive, and falls will be reflected in the data with the highest variance [2]. We take the data that represents 90% of the variance and use that to create the spectrogram. Finally, we use the Short Time Fourier Transform with a 512ms window size and 128ms overlap. To emphasize the high variance data, we create a single

spectrogram from a weighted average based on the proportion of the variance covered. A sample spectrogram from our dataset after this processing can be seen in Figure 1, along with a sample spectrogram from the original paper.

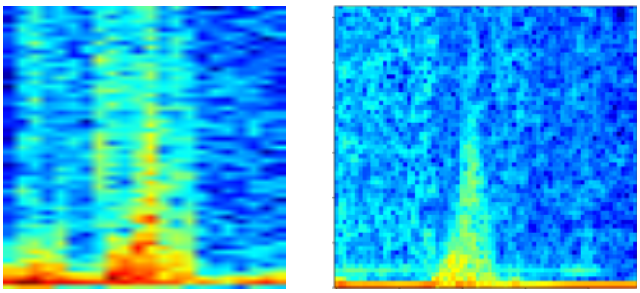

Figure 1. A sample of our spectrogram (left) compared to the original paper's sample spectrogram (right)

ML Methods

In the reference paper, the authors use ResNet34, an approximately 21 million parameter model, as the base model. We choose MobileNetV2 [5], an approximately 3 million parameter model designed for image classification on mobile and compute-constrained devices. We then implemented the two-stage training pipeline described in the reference paper [2]. In the first phase, we freeze all layers of the model except the last fully connected layer and fine tune for 16 epochs on 200 samples. Then in phase 2, we unfreeze all layers and train for 32 epochs with a cyclical learning rate. This two-stage process allows us to preserve the learnings in the pretrained model while adjusting to our dataset in a more stable, controlled manner, especially considering the small size of our finetune dataset.

Putting it Together

Our full fall-detection pipeline uses the RPi and Nexmon-CSI to capture data into a suite of .pcap files. Then, using the methods described above, the .pcap files containing CSI data are converted into spectrograms. These spectrograms are used to finetune MobileNet v2 to produce a final CNN. We can then capture .pcap files of CSI data in other testing environments and use the final, finetuned CNN to predict fall and non-fall events. We also implemented a live detection method to stream a 10-second window of CSI data, apply the signal processing as above to create a spectrogram, and output predictions from the CNN live.

Implementation and Evaluation

Dataset

Our final dataset consists of spectrograms from 100 falls and 150 non-falls performed by the three of us. The non-falls included the following actions: sit down, stand up, walking, lay down, pick something up from the ground, stationary sitting, upper body movement, singular jump, and multiple jumps. These categories reflect the non-fall actions described in the original paper [2]. Figure 2 depicts our training environment (Environment 1), with the pinging laptop, Raspberry Pi, and router placed in a triangular formation. All actions, falls and non-falls, were performed in various locations within the fall area in the center.

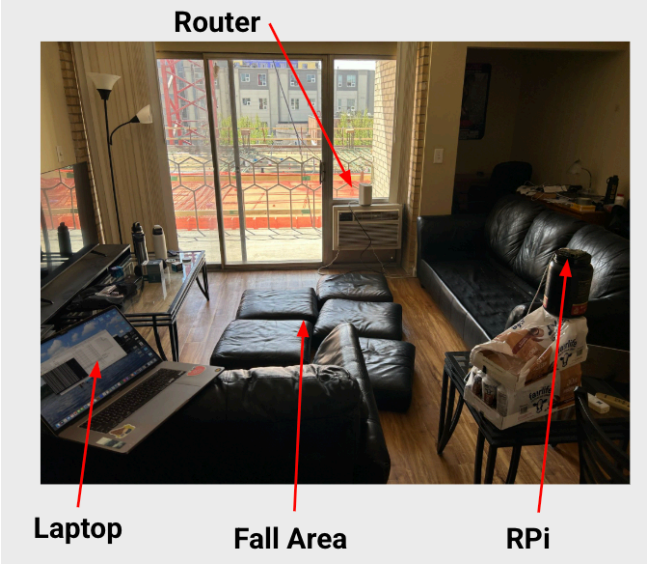

Figure 2. Setup of Environment 1 where training data pcap files were collected.

Evaluation

We tested our model in two different environments, using the held-out test set from our training room (Environment 1) and a new test set from a previously unseen room (Environment 2) to assess our model's ability to generalize, as that was an advantage outlined by the original paper. To create this new test set, we mimicked the same triangular placement with the action area in the center, as depicted by Figure 3. This test set was smaller, containing 15 falls and 20 non-falls. The non-fall actions included sit down, stand up, walking, lay down, and stationary sitting.

The results for precision, recall, F1, and accuracy for both environments can be found in Table 1. Confusion matrices outlining the distribution of false/true positives and false/true negatives in Environment 1 and 2 can be found in Tables 2 and 3, respectively. For Environment 2, we also measured the average time of our pipeline from spectrogram creation to final prediction, which was 6.94 seconds.

	Environment 1	**Environment 2**
Precision	0.67	0.39
Recall	0.70	0.60
F1 Score	0.68	0.47
Accuracy	0.74	0.43

Table 1. Evaluation metrics for both environments. Environment 1 is the same environment as the training data, and Environment 2 is the new environment to assess generalizability.

	Predicted Non-Fall	Predicted Fall
Non-Fall	23	7
Fall	6	14

Table 2. Confusion matrix for the held-out test set, which includes spectrograms from Environment 1

	Predicted Non-Fall	Predicted Fall
Non-Fall	6	14
Fall	6	9

Table 3. Confusion matrix for the generalizability test set, which includes spectrograms from Environment 2

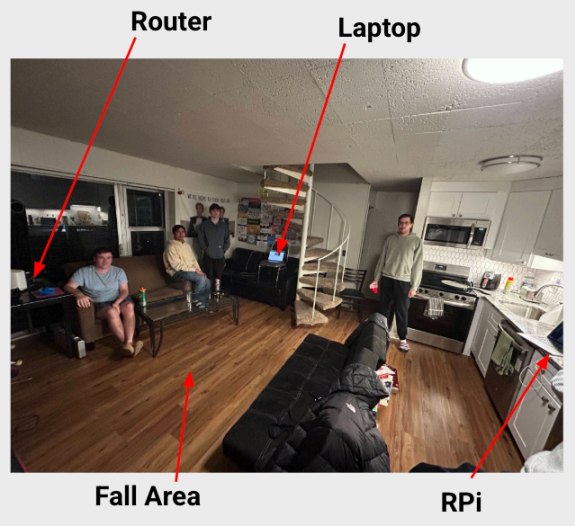

Figure 3. Setup of Environment 2, where the generalizability test set was created.

Discussion

Results

Overall, our model performed decently well when tested in Environment 1, the environment in which it was trained. We were able to successfully achieve a higher recall over precision, which was our desired goal as false negatives (not detecting a fall when a fall occurred) are much more critical of an error in this application. Additionally, we reached a 74% accuracy as seen in Table 1, which is lower than the 92% accuracy achieved by the reference paper with a larger base model. However, the time it takes to process a 10-second window of data was 6.94 seconds, which allows for real-time processing and rapid detection of falls. Our model did not perform as well in Environment 2, with only a 43% accuracy and 60% recall, as we can see in Table 1. We discuss the accuracy of both environments in the Limitations section.

Limitations
A significant limitation of our pipeline was its accuracy when moved to Environment 2. This may have been because of noise in our testing and training CSI data. Both environments had multiple people and furniture in the room when data was collected, whose small movements could cause reflections that disturb the CSI data. Removing these environmental factors could have yielded better results through cleaner training and testing data. Alternatively, generalizability could be improved by training the model on data from multiple environments. Using a WiFi chipset with more than one antenna, such as the Intel chipset used in the original paper, could also make our CSI data collection more resistant to environmental factors. Finally, our WiFi chipset used Automatic Gain Control (AGC) to dynamically adjust the gain of the antenna based on the strength of the incoming signal. AGC's dynamic gain has the potential to cause significant disruptions to our CSI data. Software-based compensation for AGC could have been used to compensate for this effect.

Other limitations of our pipeline include its feasibility. It requires a sender (laptop), receiver (router), and monitor (RPi) placed in a specific configuration to collect data, which may not be feasible for implementing this method in a variety of settings. Also, Nexmon-CSI disables WiFi on the RPi to collect CSI data. If our CSI-based fall detection method were to be implemented into an actual product, it would require two WiFi chipsets, with one dedicated for collecting CSI data while the other connects to the Internet.

Future Work
To enable better CSI data collection, more research should be done testing this method on hardware outside the RPi. For example, the ESP32 offers similarly limited processing capabilities while also allowing for the connection of multiple external antennas to receive cleaner CSI data.

Another extension of this paper would be implementing the pipeline on commodity edge devices. Amazon's Alexa already collects CSI data and can use the internet. This pipeline could be implemented on hardware like Alexa, using CNN inference performed at the edge or in the cloud, to determine its effectiveness in commodity devices in multiple environments. And, more research could be done into how the pipeline could be connected to emergency contacts or services to send for help in case of a fall.

Individual Contributions

Sam - Setting up the RPi software and configuring Nexmon-CSI, writing code to generate spectrograms, collecting data in Environment 1 and Environment 2, testing the CNN inference on the RPi, adding live processing
Nishant - Collecting data in Environment 1, writing code for CNN training and prediction, training on GPU
Kristine - Collecting data in Environment 1, writing code for CNN training

Project Materials

Source Code
The source code and dataset for this experiment can be accessed through this link. Documentation on setup, data collection, training, and running predictions can be found in the README.

Video Demo
The video demo is of the live processing pipeline for a fall. The video can be accessed through this link.

Presentation Slides
Final presentation slides can be accessed through this link.

References

[1] Amrit Pal Kaur, Ejay Nsugbe, Amy Drahota, Matthew Oldfield, Iman Mohagheghian, Radu A. Sporea, State-of-the-art fall detection techniques with emphasis on floor-based systems—A review, Biomedical Engineering Advances, Volume 9, 2025, 100179, ISSN 2667-0992, https://doi.org/10.1016/j.bea.2025.100179.

[2] T. Nakamura, M. Bouazizi, K. Yamamoto and T. Ohtsuki, "Wi-Fi-CSI-based Fall Detection by Spectrogram Analysis with CNN," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, Taipei, Taiwan, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322323.

[3] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free Your CSI: A Channel State Information Extraction Platform for Modern Wi-Fi Chipsets," in Proc. 13th Int. Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH '19), Oct. 2019, pp. 21–28. doi: 10.1145/3349623.3355477.

[4] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: The C-based Firmware Patching Framework," 2017. [Online]. Available: https://nexmon.org

[5] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ‘Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation’, CoRR, vol. abs/1801.04381, 2018.