SF2D: Semi-supervised Federated Learning for Fall Detection using (Un)labelled Data in Edge-Cloud

Seyed Alireza Rahimi Azghadi[†,*], Hung Nguyen[†§], Irina Kondratova[§], Hélène Fournier[§], Monica Wachowicz[†◇], Francis Palma[†‡], René Richard[§†], Hung Cao[†]
[†] *Analytics Everywhere Lab, University of New Brunswick, Canada*
[‡] *SE+AI Research Lab, University of New Brunswick, Canada*
[◇] *RMIT University, Australia*
[§] *National Research Council, Fredericton, New Brunswick, Canada*

Abstract

The aging population faces increased health risks, with falls being a major concern for individuals over 65, leading to healthcare strain and distress. We propose a semi-supervised federated learning-based fall detection (SF2D) method that leverages edge devices to maintain user privacy while ensuring accurate detection. Our approach first trains an unsupervised autoencoder with federated learning, then uses its encoder to train a cloud-based classifier with benchmark datasets. Our proposed SF2D improves accuracy by 1% and recall by 4% over state-of-the-art systems, offering a practical, accurate solution for fall detection and elderly care.

Keywords: Semi-supervised Federated Learning, Fall Detection, Edge, Cloud

1. Introduction

Falling is a leading cause of mortality, reduced mobility, and cognitive decline in aging populations [1], making fall detection a critical priority. Fall Detection Systems (FDS) use sensors and algorithms to detect falls, primarily through three approaches: 1) vision-based systems, 2) environmental signal detection, and 3) wearable sensors integrated into devices like smartwatches and fitness trackers [2, 3]. Wearable sensors collect data from accelerometers, gyroscopes, and heart rate monitors, enabling portability and continuous monitoring. However, challenges in privacy, accuracy, and diversity still impact their effectiveness.

Federated learning (FL) enhances privacy in wearable fall detection systems (FDS) by processing data locally and transmitting only aggregated insights, preserving user privacy while improving model accuracy. FL-based fall detection methods, such as Fed-ELM [4], FL-FD [5], and FEEL [6], integrate techniques like Extreme Learning Machines, sensor-visual data fusion, user profile-based personalization, and few-shot learning.

However, existing FL fall detection systems rely heavily on labeled data, which is impractical and raises privacy concerns [7–10]. Labeling is intrusive and limits real-world applicability. Moreover, few methods effectively operate without labeled data while ensuring user privacy. Addressing these limitations is crucial for advancing fall detection technology.

To address these challenges, we propose **Semi-supervised Federated Learning Fall Detection (SF2D)**, which integrates semi-supervised learning with FL for fall detection. Semi-supervised FL offers a promising solution to the limitations of current FDS by leveraging both labeled and unlabeled data.

By leveraging patterns in unlabeled data with a small labeled set, semi-supervised FL enables accurate, privacy-preserving detection. Our contributions:

(1) We presented SF2D, a semi-supervised federated learning approach, to tackle the data labelling challenges with FDS.
(2) We conducted an extensive evaluation of our SF2D based on a benchmark SiSFall dataset [11]. Our approach, SF2D, outperforms other federated-based FDS [4] using different metrics.

[*] alireza.rahimi@unb.ca

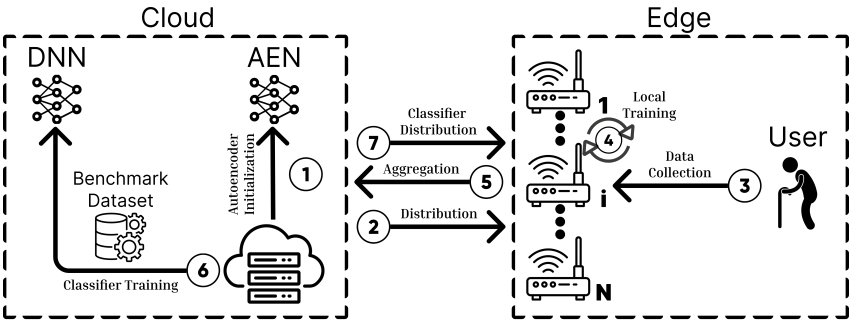

Figure 1. Our SF2D Fall Detection System Overview, showcasing the system's steps for fall detection and communication between cloud and edge

2. Related Work

Federated Learning (FL) in Fall Detection Systems (FDS):. FL has emerged as a promising solution to address the limitations of traditional FDS in wearable devices. By processing data locally, FL mitigates privacy concerns while transmitting only aggregated insights to a central server. Various studies have explored its potential in healthcare, particularly for fall detection and personalized monitoring.

Yu et al. [4] proposed an FL-based Extreme Learning Machine (FED-ELM) model, initially trained on younger subjects and later refined with misclassified elderly data in the SisFall dataset [11]. Wu et al. [12] introduced FedHome, leveraging a generative convolutional autoencoder to handle non-IID data while ensuring low latency and privacy protection. Ghosh et al. [6] integrated Few-shot Learning with FL, using clustering for personalized adaptation and deploying an edge-based prototype for real-world healthcare monitoring. Furthermore, Qi et al. [5] focused on multimodal data fusion, converting time-series wearable data into images and integrating them with camera inputs via a deep convolutional neural network. Together, these studies highlight FL's potential in healthcare, improving personalization, accuracy, and privacy in fall detection and remote monitoring.

Semi-Supervised Federated Learning for Privacy-Preserving Scenario: Current FDS face challenges due to their reliance on labeled user data, which is difficult to obtain in real-life situations. Labeling can be intrusive, impractical, and raises privacy concerns [7–10, 13]. Moreover, few methods operate without labeled data while preserving user privacy, making it crucial to develop more effective and privacy-aware FDS. A promising solution is semi-supervised FL, which combines FL and semi-supervised learning to utilize both labeled and unlabeled data [8, 9]. Zhao et al. [9] demonstrated how FL can integrate semi-supervised learning to improve performance while preserving privacy. Tashakori et al. [8] introduced SemiPFL, which enhances adaptability in heterogeneous environments by leveraging unlabeled data.

However, existing approaches have not specifically addressed wearable-based fall detection. Hence, our paper proposes SF2D, a semi-supervised FL approach for FDS, using a small labeled dataset on a central cloud while processing unlabeled data across user devices. Our method ensures privacy while improving accuracy, bridging the gap between effective fall detection and data security.

3. Method

In this section, we proposed SF2D, as shown in Figure 1 and Algorithm 1, with seven key steps: ① autoencoder initialization, ② distribution, ③ data collection, ④ local training, ⑤ federated averaging, ⑥ classifier training, and ⑦ classifier distribution. These steps collectively enable a semi-supervised, privacy-preserving, distributed approach to fall detection using wearable devices and edge computing.

1. Autoencoder Initialization. SF2D begins by initializing an autoencoder network **A** with random parameters $\theta_{\mathbf{A}}^0$, stored in the cloud. This provides a consistent starting point for all users, ensuring uniform training and improved model convergence.
2. Distribution. Once the network **A** is initialized, its parameters $\theta_{\mathbf{A}}^0$ are distributed to all participating edge devices. Let N users be associated with a unique edge device, and $\mathcal{E}_i$ denote the edge device associated with user i, where $i \in \{1, 2, \ldots, N\}$. Each $\mathcal{E}_i$ is tasked with training the autoencoder parameters $\theta_{\mathbf{A}}^{(i)}$ locally using user-specific data.
3. Data collection. Next, in the data collection phase, each edge device $\mathcal{E}_i$ gathers data $\mathcal{U}_i = \{x_j\}_{j=1}^{M_i}$ from its associated user's wearable device, where $x_j \in \mathbb{R}^d$ represents a data point at the j-th time step or from the j-th observation and M_i is the total number of data points collected by edge device $\mathcal{E}_i$ from the user's wearable device. The data $\mathcal{U}_i$ is used exclusively for training the autoencoder during the local training rounds and is subsequently discarded to ensure the user's privacy. Importantly, our SF2D does not associate with any user's labels y_j in this step.
4. Local training. Each edge device $\mathcal{E}_i$ proceeds to the local training phase, utilizing the collected data $\mathcal{U}_i$ and the distributed parameters from the cloud. Before training, the following preprocessing steps are applied to the raw sensor data: we resample the data to a specific frequency (e.g., 50 ms) to reduce noise and ensure uniformity. Then, we apply an exponentially weighted mean (EWM) filter for noise reduction. Finally, we convert the data into fixed-size sequences for consistent input to the autoencoder. During this training phase, the autoencoder is trained for a single epoch on the user's local data to update the parameters $\theta_{\mathbf{A}}^{(i)}$. The local update of the model is denoted by: $\theta_{\mathbf{A}}^{(i)} \leftarrow \theta_{\mathbf{A}}^{(i)} - \eta \nabla_{\theta_{\mathbf{A}}} \mathcal{L}_i(\theta_{\mathbf{A}}^{(i)}, \mathcal{U}_i)$ where η is the learning rate, and $\mathcal{L}_i$ is the loss function computed over the local dataset $\mathcal{U}_i$.
5. Federated averaging. After local training, all edge devices send their updated network parameters $\theta_{\mathbf{A}}^{(i)}$ to the cloud server. The server then performs federated averaging (FedAvg) to aggregate the received parameters and create a new global model. The aggregation step is defined as: $\theta_{\mathbf{A}}^{(\text{global})} = \frac{\sum_{i=1}^{N} M_i \theta_{\mathbf{A}}^{(i)}}{\sum_{i=1}^{N} M_i}$ where M_i is the number of data points used by edge device $\mathcal{E}_i$, which serves as a weight for the FedAvg algorithm. This iterative process of local training and parameter aggregation continues for a predefined number of epochs. Once completed, the global autoencoder can provide a meaningful representation of signals in a transformed domain. Importantly, this process respects user privacy by ensuring that raw data never leaves the edge devices.
6. Classifier training. Once the autoencoder is trained, the system progresses to the classifier **C** training phase. In this step, the encoder **E** of the autoencoder **A** is utilized as the feature extractor, serving as the first layer of a classifier. The classifier **C** is extended by adding a fully connected neural network (FCNN) to predict activity labels. **C** is trained on a labelled dataset $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^{K}$, where $x_k \in \mathbb{R}^d$ represents a data point, y_k denotes the corresponding label, and K is the total number of data points in the labelled dataset. The preprocessing steps for classifier training follow the same procedure as the autoencoder training, adding a momentum-based calculation for fall samples to help distinguish between fall and non-fall data. Specifically, the same resampling, EWM filtering, and sequence resizing are applied. A momentum calculation is used on fall data, which removes extra parts from a fall sample to create more distinct feature representations, which helps improve classification accuracy for detecting fall events. During this training, the encoder's parameters $\theta_{\mathbf{E}}$ are frozen, while only the weights of the fully connected layers $\theta_{\mathbf{C}}$ are updated. The encoder **E** serves as a feature extractor, transforming raw signals x_k into a lower-dimensional representation that captures essential characteristics. The fully connected layers then learn the mapping between these representations and activity labels, such as detecting fall events.
7. Classifier distribution. The trained classifier is distributed back to the edge devices $\mathcal{E}_i$. Each edge device integrates this classifier to process raw signals from the user's wearable

Algorithm 1: Semi-supervised Federated Learning Fall Detection (SF2D)

Input: Number of users N, Global epochs T, Labeled dataset $\mathcal{D}$
Output: Distributed classifier $\mathbf{C}$ for edge devices

① **Autoencoder Initialization**
Initialize global autoencoder $\mathbf{A}$ with random parameters $\theta_{\mathbf{A}}^{0}$ in the cloud
② **Distribution**
Distribute $\theta_{\mathbf{A}}^{0}$ to all edge devices $\{\mathcal{E}_i\}_{i=1}^{N}$
③ **Data Collection**
foreach *edge device* $\mathcal{E}_i$ **do**
  Collect $\mathcal{U}_i = \{x_j\}_{j=1}^{M_i}$
  Discard data after local training
④ **Local Training**
foreach *edge device* $\mathcal{E}_i$ *in parallel* **do**
  Preprocess $\mathcal{U}_i$:
    1. Resample to 50Hz
    2. Apply EWM filter
    3. Create fixed-size sequences
  Update parameters:
  $\theta_{\mathbf{A}}^{(i)} \leftarrow \theta_{\mathbf{A}}^{(i)} - \eta \nabla_{\theta_{\mathbf{A}}} \mathcal{L}_i(\theta_{\mathbf{A}}^{(i)}, \mathcal{U}_i)$
⑤ **Federated Averaging**
Aggregate parameters in the cloud:
$\theta_{\mathbf{A}}^{(\text{global})} \leftarrow \frac{1}{\sum M_i} \sum_{i=1}^{N} M_i \theta_{\mathbf{A}}^{(i)}$
Distribute $\theta_{\mathbf{A}}^{(\text{global})}$ to all $\mathcal{E}_i$
⑥ **Classifier Training**
Freeze encoder $\mathbf{E}$ from trained $\mathbf{A}$
Add FCNN layers for classification
Preprocess $\mathcal{D}$:
  1. Apply same preprocessing as $\mathcal{U}_i$
  2. Add momentum-based fall sample calculation
Train classifier $\mathbf{C}$ on $\mathcal{D}$ to optimize:
$\arg\min_{\theta_{\mathbf{C}}} \sum_{(x_k, y_k) \in \mathcal{D}} \mathcal{L}_{\text{CE}}(\mathbf{C}(\mathbf{E}(x_k)), y_k)$
⑦ **Classifier Distribution**
Deploy trained $\mathbf{C}$ to all edge devices $\mathcal{E}_i$
return *Distributed fall detection system*

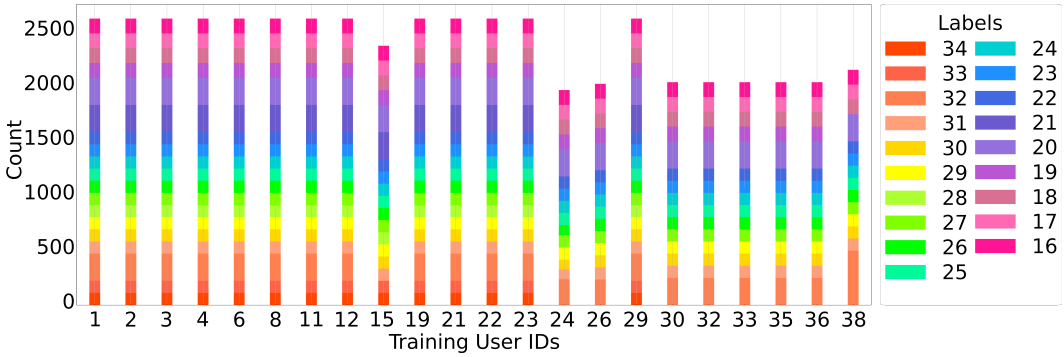

Figure 2. Real-world subset users data distribution showing different activities of daily life from different users (labels are based on the original SiSFall dataset's labels [11])

device. This enables real-time event classification and fall detection, all while preserving user privacy. Since raw data remains confined to the edge device throughout the process, the approach ensures robust privacy preservation and decentralized model operation.

4. Experiment

This section evaluates the practical effectiveness of our SF2D. We designed a real-world scenario using a benchmark dataset to highlight our method's strengths under controlled conditions. Our SF2D was assessed under two semi-supervised training approaches, i.e., Centralized and Federated, compared against Fed-ELM [4].

4.1. Dataset. For our evaluation, we employed the SiSFall dataset [11], widely recognized for FDS assessments. This dataset encompasses 38 participants, divided into young (23) and older (15) groups. It spans a broad spectrum of activities (15 falls/positive, 19 activities of daily living (ADL)/negative), enhancing data variability and representation.

To reflect real-world conditions, we divided the SiSFall dataset into two segments: a benchmark subset stored in the cloud with labels and another serving as real-world user data without any labels. This user-based division randomly assigned 40% of participants to the benchmark $\mathcal{D}$ and 60% to the real-world segment $\mathcal{U}$. The same participants were used

Method	ACC	PR	RE	F1
Fed-ELM† [4]	98.01	-	95.25	-
CL (Ours)‡	**99.33**	99.67	**99.63**	**99.65**
FL (Ours)‡	99.19	**99.69**	99.47	99.58

† Supervised ‡ Semi-supervised

Table 1. Performance metrics comparison of different methods (ACC = Accuracy, PR = Precision, RE = Recall, and F1 = F1 score).

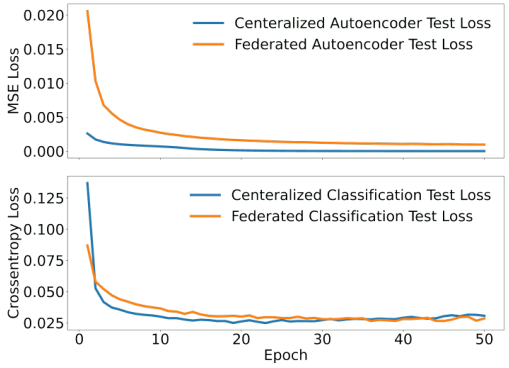

Figure 3. Autoencoder and Classifier Test Loss Over Epochs.

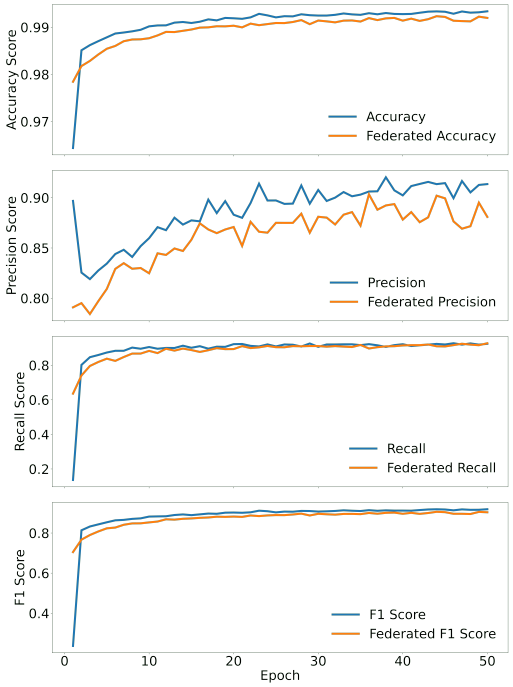

Figure 4. Test Accuracy, Precision, Recall, and F1 Score Over Epochs.

in both centralized and federated training setups. Figure 2 illustrates the data volume and activity distribution among our real-world users $\mathcal{U}$.

4.2. Network architecture. We proposed two neural network architectures:

4.2.1. Autoencoder: The autoencoder was configured with three Long Short-Term Memory (LSTM) layers with 128, 32, and 128 nodes, respectively. The model used an L2 kernel regularizer with a weight decay of 0.01 and was optimized using the Adam optimizer. Mean Squared Error loss function minimized the reconstruction error.

4.2.2. Classifier: The classifier was constructed using a three-layer dense network. The final layer consisted of two output classes, corresponding to the "Fall" and "ADL". The network was optimized using the Adam optimizer with a learning rate of 10^{-4}, and categorical cross-entropy was used as the loss function.

4.3. Training modalities. Our SF2D was trained with two approaches:

4.3.1. Centralized learning (CL): The autoencoder was trained using real-world user data $\mathcal{U}$ in the cloud for 50 epochs. Post-training, the classifier was trained using the benchmark subset containing labels $\mathcal{D}$. The classifier was assessed using real-world user data $\mathcal{U}$.

4.3.2. Federated learning (FL): The autoencoder was trained on real-world user data $\mathcal{U}$ using the Flower framework [14] across 50 communication rounds. Subsequent classifier training occurred in the cloud using the benchmark subset with labels $\mathcal{D}$. Classifier performance was evaluated using real-world user data $\mathcal{U}$.

4.4. Result. Table 1 highlights the performance metrics of different methods evaluated in our study. Among these, the CL method achieved the highest overall performance, serving as the baseline with accuracy, recall, and F1 scores of 99.33%, 99.63%, and 99.65%, respectively. The FL method was followed closely, delivering comparable results and outperforming CL in precision with a score of 99.69%. Both our CL and FL significantly outperformed Fed-ELM [4] across all metrics, showcasing the effectiveness of our novel techniques. Notably, the performance gaps between CL and FL were minimal, underlining the robustness of both approaches, even in FL settings. These results demonstrate that our proposed methods

not only surpass the state-of-the-art in predictive performance but also offer substantial practical advantages. Our FL approach works in a more practical, real-world scenario, which assumes that there is no label available for the end-user's data. By achieving near-baseline performance while preserving data privacy and utilizing semi-supervised training, our methods exemplify a strategic balance between performance and real-world applicability, setting a new standard for predictive tasks in sensitive or distributed data environments.

Figure 3 shows that in both training setups, our networks converged at some points. Similarly, Figure 4 presents the performance metrics for Centralized and Federated training. The results indicate that Federated training achieves performance comparable to Centralized training while maintaining user privacy, highlighting its practical advantage.

5. Conclusion and Future work

Our study introduces SF2D, a semi-supervised, privacy-preserving fall detection method using wearable devices and FL. Our approach leverages edge devices and an unsupervised autoencoder, ensuring both privacy and accuracy. Future work will focus on data quality monitoring, drift detection, and model updates within FL. Further evaluation of benchmark datasets, real-world testing, and alternative preprocessing techniques, such as resampling and filtering, will provide deeper insights into improving predictive performance.

Acknowledgement

This study is supported by the UNB-FCS Startup Fund (22–23 START UP/H CAO). The equipment used in the experiments was supported by CFI Project Number 39473. The Cloud resource is supported by The Digital Research Alliance of Canada.

References

[1] C. Trevisan et al. "The association between injurious falls and older adults' cognitive function: the role of depressive mood and physical performance". In: *The Journals of Gerontology: Series A* 76.9 (2021), pp. 1699–1706.

[2] A. Ramachandran and A. Karuppiah. "A survey on recent advances in wearable fall detection systems". In: *BioMed research international* 2020.1 (2020), p. 2167160.

[3] D. Mohan et al. "Artificial Intelligence and IoT in Elderly Fall Prevention: A Review". In: *IEEE Sensors Journal* (2024).

[4] Z. Yu et al. "An Elderly Fall Detection Method Based on Federated Learning and Extreme Learning Machine (Fed-ELM)". In: *IEEE Access* 10 (2022), pp. 130816–130824.

[5] P. Qi et al. "FL-FD: Federated learning-based fall detection with multimodal data fusion". In: *Information Fusion* 99 (Nov. 1, 2023), p. 101890. ISSN: 1566-2535.

[6] S. Ghosh and S. K. Ghosh. "FEEL: FEderated LEarning Framework for ELderly Healthcare Using Edge-IoMT". In: *IEEE Transactions on Computational Social Systems* 10.4 (Aug. 2023), pp. 1800–1809. ISSN: 2329-924X.

[7] E. Diao et al. "Semifl: Semi-supervised federated learning for unlabeled clients with alternate training". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17871–17884.

[8] A. Tashakori et al. "SemiPFL: Personalized semi-supervised federated learning framework for edge intelligence". In: *IEEE Internet of Things Journal* 10.10 (2023), pp. 9161–9176.

[9] Y. Zhao et al. "Semi-supervised Federated Learning for Activity Recognition". In: *ACM Trans. Intell. Syst. Technol* 1.1 (2021).

[10] H. Xiao et al. "Towards Privacy-Supporting Fall Detection via Deep Unsupervised RGB2Depth Adaptation". In: *IEEE Sensors Journal* (2023).

[11] A. Sucerquia et al. "SisFall: A fall and movement dataset". In: *Sensors* 17.1 (2017), p. 198.

[12] Q. Wu et al. "FedHome: Cloud-Edge Based Personalized Federated Learning for In-Home Health Monitoring". In: *IEEE Transactions on Mobile Computing* 21.8 (2022).

[13] S. Ji et al. "SiFall: Practical online fall detection with RF sensing". In: *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. 2022, pp. 563–577.

[14] D. J. Beutel et al. "Flower: A Friendly Federated Learning Research Framework". In: *arXiv preprint arXiv:2007.14390* (2020).