applied sciences

MDPI

Article

Fall Detection Using Federated Lightweight CNN Models: A Comparison of Decentralized vs. Centralized Learning

Qasim Mahdi Haref [1,2], Jun Long [3,*] and Zhan Yang [3]

[1] School of Computer Science and Engineering, Central South University, Changsha 410083, China; qasimmahdi@iku.edu.iq
[2] Computer Techniques Engineering, Imam Alkadhim University College, Baghdad 10006, Iraq
[3] Big Data Institute, Central South University, Changsha 410083, China; zyang22@csu.edu.cn
* Correspondence: junlong@csu.edu.cn

Abstract

Fall detection is a critical task in healthcare monitoring systems, especially for elderly populations, for whom timely intervention can significantly reduce morbidity and mortality. This study proposes a privacy-preserving and scalable fall-detection framework that integrates federated learning (FL) with transfer learning (TL) to train deep learning models across decentralized data sources without compromising user privacy. The pipeline begins with data acquisition, in which annotated video-based fall-detection datasets formatted in YOLO are used to extract image crops of human subjects. These images are then preprocessed, resized, normalized, and relabeled into binary classes (fall vs. non-fall). A stratified 80/10/10 split ensures balanced training, validation, and testing. To simulate real-world federated environments, the training data is partitioned across multiple clients, each performing local training using pretrained CNN models including MobileNetV2, VGG16, EfficientNetB0, and ResNet50. Two FL topologies are implemented: a centralized server-coordinated scheme and a ring-based decentralized topology. During each round, only model weights are shared, and federated averaging (FedAvg) is applied for global aggregation. The models were trained using three random seeds to ensure result robustness and stability across varying data partitions. Among all configurations, decentralized MobileNetV2 achieved the best results, with a mean test accuracy of 0.9927, F1-score of 0.9917, and average training time of 111.17 s per round. These findings highlight the model's strong generalization, low computational burden, and suitability for edge deployment. Future work will extend evaluation to external datasets and address issues such as client drift and adversarial robustness in federated environments.

Keywords: fall detection; federated learning; MobileNetV2; VGG16; decentralized training; edge AI; deep learning; ring topology; image classification

Academic Editors: Thomas Lindner and Juan A. Gómez-Pulido

Received: 12 June 2025
Revised: 22 July 2025
Accepted: 23 July 2025
Published: 25 July 2025

Citation: Haref, Q.M.; Long, J.; Yang, Z. Fall Detection Using Federated Lightweight CNN Models: A Comparison of Decentralized vs. Centralized Learning. *Appl. Sci.* **2025**, *15*, 8315. https://doi.org/10.3390/app15158315

1. Introduction

Falls by elderly persons are a major public health issue because they frequently result in severe injuries, hospital stays, elevated morbidity, and expensive medical care. According to the World Health Organization (WHO), falls are the second-most-common cause of unintended or accidental injury-related fatalities globally, with people 65 and older being particularly vulnerable [1]. Every year, more than 37.3 million falls require medical care; this figure is predicted to increase as the world's population ages [2]. The desire to prevent the detrimental effects of falls by timely detection and quick medical intervention has

spurred extensive research into the development of automated fall-detection systems [3]. A variety of fall-detection methodologies have been proposed, ranging from wearable sensor-based approaches to ambient sensing technologies. Wearable devices equipped with accelerometers and gyroscopes are widely utilized to capture dynamic motion signals associated with falls [4]. In parallel, radar-based systems, particularly those leveraging Doppler shift analysis, have emerged as a privacy-conscious alternative to video surveillance, offering non-intrusive monitoring capabilities [5]. Despite these advances, such systems face significant barriers to widespread deployment, mainly due to user discomfort, frequent recharging requirements, and difficulties in accurately differentiating falls from other routine movements [6].

Recent advances in vision-based technologies and the growing availability of affordable, high-resolution video sensors have catalyzed interest in video-based fall-detection frameworks. These systems enable continuous monitoring and detailed spatial–temporal analysis, allowing the use of large-scale video datasets to train sophisticated machine learning models [7]. Specifically, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two deep learning architectures that have shown increased accuracy in identifying fall patterns [8]. To improve predictive accuracy even more, ensemble learning techniques have been investigated that integrate the outputs of several classifiers [9].

In this regard, transfer learning—which involves fine-tuning pretrained models on extensive action-recognition datasets with fall-specific video data—has grown in popularity as a fall-detection method. This method maintains excellent generalization capabilities while accelerating convergence and drastically reducing the demand for huge annotated datasets [10]. In the meanwhile, federated learning presents a framework for cooperative model training across dispersed healthcare settings that protects privacy. By restricting the sharing of model updates and maintaining patient data locally, federated learning tackles important issues related to data security, privacy, and adherence to laws like the General Data Protection Regulation (GDPR) [11].

Nonetheless, video-based systems raise serious privacy issues due to the inherently sensitive nature of visual data [12]. Privacy-preserving methods like video anonymization are used to remedy this; background subtraction and face or body obfuscation are increasingly integrated to strike a balance between detection accuracy and confidentiality [13]. Legal frameworks like the GDPR mandate strict adherence to ethical data processing standards, necessitating robust protection mechanisms [14]. The interpretability of detection systems based on deep learning presents another difficulty. Conventional models frequently serve as opaque "black boxes," providing little information about how decisions are made [15]. In clinical applications, in which decisions directly impact patient safety and treatment, this lack of transparency can hinder acceptance by medical professionals [16]. Thus, improving model explainability is crucial for establishing credibility, confirming model predictions, and encouraging well-informed choices in medical evaluations connected to falls [17,18].

Overview and Contribution

In this study, we present a thorough experimental framework for both centralized and federated learning paradigms relating to deep transfer learning models for video-based fall detection. Leveraging the publicly available fall-detection dataset from Kaggle, our approach integrates MobileNetV2 and VGG16 as backbone architectures for the efficient and accurate classification of fall versus non-fall events. The proposed work explores and compares four training schemes:

- Decentralized Federated Learning (Ring-based): Clients receive a copy of the global model, train locally, and sequentially update weights, which are averaged using FedAvg. This was implemented using both MobileNetV2 and VGG16.
- Centralized Federated Learning: A central server aggregates model updates from multiple clients through weighted averaging of parameters. This method was also evaluated with both MobileNetV2 and VGG16 to assess scalability and coordination benefits
- Transfer Learning: Models pretrained on ImageNet (MobileNetV2 and VGG16) were adapted to the fall-detection task. This transfer learning technique enhanced generalization performance and drastically decreased the requirement for large amounts of training data.
- Privacy-Preserving Architecture: By partitioning data across multiple simulated clients and avoiding central data storage, our federated setup respects data sovereignty and aligns with privacy regulations such as GDPR.

The novelty of this study lies in the end-to-end deployment and benchmarking of federated fall-detection systems using both lightweight (MobileNetV2) and heavy (VGG16) architectures. Furthermore, the dataset was preprocessed using a label-aware cropping strategy to extract meaningful fall-related regions from YOLO-labeled images, which were then normalized and resized. This preprocessing pipeline not only reduced noise but also enhanced training convergence.

The following is a summary of this study's primary contributions:

- In order to detect falls in a video-based environment, we offer a unified framework that assesses both centralized and decentralized federated learning techniques.
- We implement a robust preprocessing pipeline converting YOLO-labeled bounding boxes into clean image patches, facilitating accurate binary classification.
- We compare the performance of MobileNetV2 and VGG16 under different federated setups, reporting metrics such as accuracy, loss, and confusion matrices for comprehensive evaluation.
- We provide reproducible code and systematic methodology that can be extended to other healthcare monitoring applications requiring real-time, privacy-aware decision making.

2. Related Work

Recent developments in fall-detection systems have increasingly leveraged deep learning architectures to enhance real-time monitoring, particularly for applications in elder care and healthcare surveillance. A number of studies have shown how well detection models work in conjunction with image processing methods to accurately identify fall incidents. For instance, one approach integrated a pre-trained object detection model with optimized camera positioning, reporting an accuracy rate exceeding 93% under controlled lighting conditions, though its performance remained reliant on specific spatial configurations [19].

In a separate study, clustering methods were used to refine anchor boxes in an object detection framework, leading to improved processing speed and detection precision in complex environments. However, challenges such as adaptability across variable settings and diverse populations were not fully addressed [20]. Other works have incorporated motion analysis and posture evaluation into video-based frameworks, detecting and tracking individuals in real time. These systems reported promising results—achieving up to 92% accuracy in daylight—but exhibited limitations under low-light conditions and in scenarios involving multiple subjects, which led to increases in false positives [21].

The use of lightweight detection models optimized for edge computing has also been explored. One such system, trained on a manually annotated dataset of over 3400 images, achieved a precision of 95% and recall of 96%, with an inference speed of under 7 ms. Implemented on compact devices such as Raspberry Pi, it enabled efficient real-time deployment while maintaining high classification performance. The associated study highlighted the superiority of deep learning-based techniques over traditional classifiers like SVM and KNN, and recommended future integration with pose estimation and attention mechanisms for improved robustness [22].

Further advancements were achieved by embedding attention modules such as Convolutional Block Attention Module (CBAM) and Squeeze-and-Excitation (SE) into object detection networks. Combined with advanced activation functions like Swish, these modifications enhanced feature extraction and mitigated gradient vanishing issues, resulting in accuracy levels of over 97% on large-scale image datasets [10]. This optimized architecture significantly outperformed earlier methods and demonstrated strong generalization, though it relied heavily on labeled training data.

In comparative evaluations of newer model variants, compact architectures designed for efficiency (e.g., versions of object detectors with reduced parameter counts) have shown notable improvements in speed and accuracy, especially when trained with diverse datasets including rotated images. The viability of implementing fall-detection systems on hardware with limited resources, like the Raspberry Pi, was further illustrated by hybrid approaches that combined convolutional models with classifiers like Support Vector Machines (SVMs) and Multi-Layer Perceptrons (MLPs) while maintaining sensitivity and accuracy [23–25].

Recent models have embraced multiscale representation and self-attention mechanisms to enhance temporal and spatial feature extraction. A state-of-the-art model maintained consistent performance across several Intersection-over-Union (IoU) thresholds, achieving 90% mean average accuracy (mAP) and a precision–recall AUC of 0.894. In order to fight class imbalance, this was accomplished by optimizing the model by utilizing unique anchor dimensions and a unique loss function [26]. Beyond healthcare, deep learning-powered fall-detection systems are being explored in industrial safety contexts, underscoring their potential to improve operational efficiency and worker protection [27].

In summary, the integration of deep learning strategies in fall detection—ranging from attention-enhanced networks to lightweight edge-deployable models—has significantly improved system responsiveness, accuracy, and scalability. Future directions may focus on combining temporal posture estimation, explainable AI components, and user-centered interface designs to further increase reliability and trust in real-world deployments. Table 1 summarizes related work on deep learning-based fall detection systems.

Table 1. Overview of research on deep learning-based fall detection.

Ref	Dataset	Classification Architecture	Model	Env.	Key Contribution	Results	Limitation (vs. Our Work)
[19]	Custom video data	CNN-based object detection (YOLO)	YOLO pretrained	Low-light indoor	Real-time fall detection using YOLO with optimized camera height–distance ratio	Accuracy: 93.16%	Requires fixed camera setup, lacks adaptability and privacy features
[20]	Not specified	YOLOv3 + K-means for anchor tuning	YOLOv3 + K-means	Complex indoor	Anchor optimization using K-means for better speed and accuracy	mAP: 0.83	No privacy preservation or federated architecture

Table 1. *Cont.*

Ref	Dataset	Classification Architecture	Model	Env.	Key Contribution	Results	Limitation (vs. Our Work)
[21]	Live video streams	YOLOv3 + posture-based feature fusion	YOLOv3 + posture	Day/low light	Combines YOLO detection with posture tracking	92% (day), 60% (low light)	Poor low-light accuracy; no data distribution or privacy mechanisms
[22]	1691 fall/1731 normal images	Tiny YOLOv4 + inference accelerator	Tiny YOLOv4 + OpenVINO	Edge device	Real-time deployment on Raspberry Pi with high precision and FPS	Precision: 95%, Recall: 96%, FPS: 26	No federated training, no explainability
[10]	21,499 images	Attention-augmented YOLO architecture	YOLOv5 + CBAM + SE	Surveillance	Enhanced YOLOv5 with CBAM, SE, and Swish activation for feature enhancement	Accuracy: 97.3%	High dependence on labeled data; no distributed training
[23]	YOLOv8n/s datasets	YOLOv8n vs. YOLOv8s (compact CNNs)	YOLOv8n vs. v8s	Mixed scenes	Comparison of lightweight YOLO versions for speed vs. accuracy	YOLOv8n better on speed/accuracy	No privacy, personalization, or federated learning
[24]	Raspberry Pi test set	YOLOv8n with classical classifier	YOLOv8n-r1 + SVM	Embedded system	Hybrid approach on Pi4 balancing accuracy and efficiency	Balanced metrics	No real-time tracking; lacks federated learning strategy
[25]	Custom labeled data	YOLOv5MU + self-attention module	YOLOv5MU + attention	Elder care	Self-attention + multiscale anchors + enhanced loss function	mAP: 90%, AUC: 0.894	Only centralized; lacks cross-device adaptation or privacy-preserving design

3. Methodology

The proposed fall-detection framework leverages federated learning combined with deep learning architectures to ensure accurate and privacy-conscious recognition of human falls. As illustrated in Figure 1, the pipeline begins with data acquisition, for which a publicly available fall-detection dataset is employed. The raw data consists of video frames annotated in YOLO format. These annotations are parsed to extract human-centered image crops, which are then resized, normalized, and binarized into two classes: fall and no-fall. To enable fair training and testing, the full dataset is stratified into training, validation, and test splits following an 80/10/10 ratio.

In the preprocessing phase, the training set is divided among multiple virtual clients, simulating a federated learning environment in which data remains local and confidential. Each client represents a decentralized node that independently holds its portion of the dataset without sharing raw data, aligning with privacy preservation principles.

The federated learning stage adopts two distinct training strategies: a centralized federated learning model and a decentralized ring-based FL topology. All clients use the same base architecture initialized with pretrained ImageNet weights. The models explored include MobileNetV2, VGG16, EfficientNetB0, and ResNet50—each fine-tuned using a transfer learning approach. Custom classification layers are appended on top, while backbone layers remain frozen to retain learned visual representations. The FederatedClientUpdate() method is employed for local training, after which either the central server calls

Server Aggregation() to compute the weighted average (FedAvg), or clients sequentially exchange model updates in the ring-based topology using unweighted averaging.

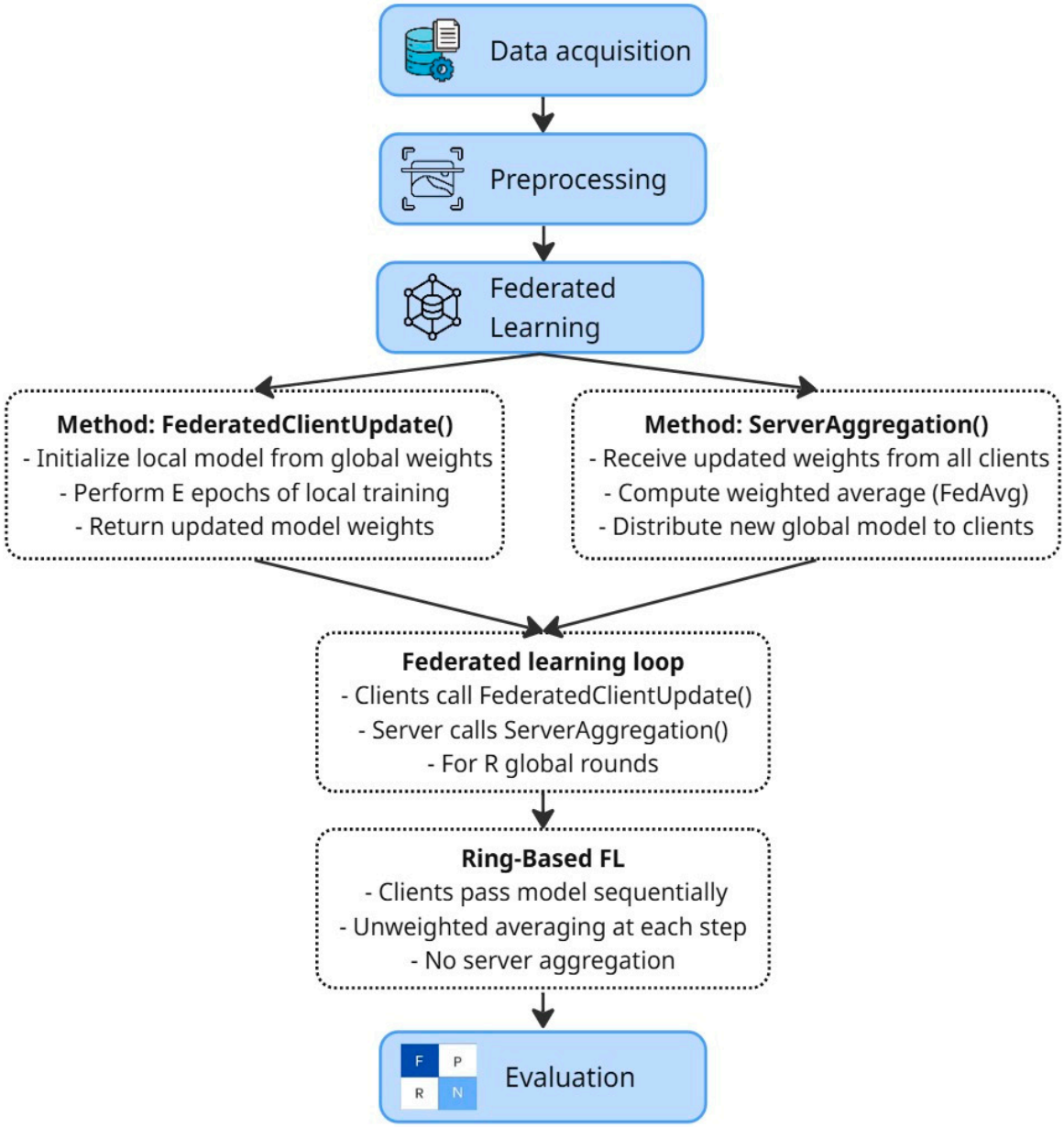

Figure 1. Flowchart illustrating the proposed fall-detection methodology using federated deep learning.

The Federated Learning Loop continues for R global communication rounds. In the centralized scenario, the server aggregates model updates; in the ring-based setting, no central server is required—clients perform local updates and pass the model in a chain. Both schemes allow the global model to progressively improve while respecting data locality.

Finally, the Evaluation block measures the performance of each model using standard classification metrics: Accuracy, Precision, Recall, and F1-score. Confusion matrices further aid in analyzing the model's behavior and error distribution. The entire architecture—modular, privacy preserving, and model-agnostic—supports a flexible and scalable framework for real-world fall-detection applications.

Algorithm 1 summarizes federated and centralized training for fall detection models.

Algorithm 1 Federated and Centralized Fall Detection with MobileNetV2 and VGG16

Require: Dataset D = {(x_i, y_i)}, number of clients K, Rounds, Epochs E
Ensure: Global model M_G with final weights Θ^G
Function PREPAREDDATA(D)
    Extract image–label pairs from YOLO annotations
    Resize image to 128 × 128
    Convert labels: fall = 0, no-fall = 1
    Split data into train, val, test (e.g., 80/10/10)
    partition D_{train} in to K shards: $D_1, \ldots, D_k$
end function
function INITIALIZEMODEL
    Select base model M∈{MobileNetV2,VGG16
    Load pretrained weights Θ^G from M
end function
function LOCALTRAINING(Θ^G, D_K,E)
    Fine-tune model M weight on D_k for E epochs
    return updated local weight Θ_k^r
end function
function AGGREGATEWEIGHTS ($\{\Theta_k^r\}$ centralized)
    if centralized **then**

$$\Theta^G \leftarrow \sum_{k=1}^{k} \frac{n_k}{\sum_{j=1}^{k} n_j} . \Theta_k^r$$

    else

$$\Theta^G \leftarrow \frac{1}{k} \sum_{k=1}^{k} \Theta_k^r$$

    end if
    return Θ^G
end function
function MAIN
    PREPAREDATA(D)
    INITIALIZEMODEL
    for r = 1 to R **do**
        For k = 1 to K in parallel **do**
            $\Theta_k^r \leftarrow$ LOCALTRAINING(Θ^G, D_K,E)
        end for
        $\Theta^G \leftarrow$ AGGREGATEWEIGHTS ($\{\Theta_k^r\}$ centralized)
        Evaluate model M_G on validation set
    end for
    Evaluate final model M_G on test set
    return Θ^G and test metrics
end function

3.1. Data Acquisition and Preprocessing

In this study, we utilize the publicly available Fall Detection Dataset, hosted on Kaggle, and compiled by Uttej Kumar Kandagatla [28]. This dataset was selected for its relevance to human activity recognition in healthcare applications, particularly fall detection, and is compatible with object detection pipelines using bounding box annotations in YOLO

format. All images and labels are used in full compliance with the licensing terms specified by the original uploader.

The dataset consists of 463 annotated image–label pairs, categorized into three primary activity classes: Fall Detected, Walking, and Sitting. These are distributed into 374 images for training and 89 images for validation. Each image has a corresponding annotation file stored in YOLO format, consisting of a single class identifier followed by four normalized bounding box coordinates that localize human subjects within the image. This format facilitates object detection tasks by providing the spatial information necessary for region-based CNN models.

Annotations were originally generated using the makesense.ai online annotation platform, as stated in the Kaggle dataset description. Bounding boxes were created around individuals performing various activities and assigned one of the three activity labels. For visualization, the dataset includes color-coded annotation overlays: violet for "Fall Detected," blue for "Walking," and green for "Sitting." An example of this annotation schema is shown in Figure 2. To prepare the data for classification, the following preprocessing steps were applied:

- Bounding boxes were used to extract cropped regions around human subjects.
- Image resizing was performed, resizing each extracted crop to a fixed dimension of 128 × 128 pixels.
- Normalization was applied to scale pixel values to the range [0, 1] [0, 1] [0, 1].
- Label binarization was conducted to convert the original three-class labels into binary labels:
  - Fall = 0;
  - NoFall = 1 (aggregating the Walking and Sitting categories).

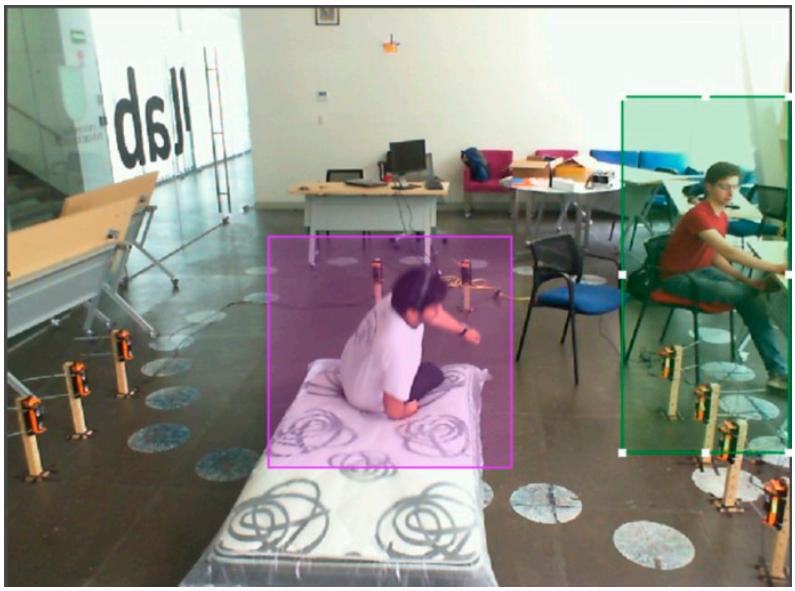

Figure 2. Annotated image from the publicly available Fall Detection Dataset on Kaggle, illustrating "Fall Detected" and "Sitting" activities, using labeled bounding boxes. The dataset was not created or manually annotated by the authors; all individuals and annotations originate in the published dataset. Usage complies with the dataset's open-access licensing terms.

The resulting dataset was split using stratified sampling to ensure balanced representation of the fall and non-fall categories across all subsets:

- 80% for training (370 samples);
- 10% for validation (46 samples);

- 10% for testing (47 samples).

The dataset offers a controlled yet realistic set of indoor activity images that simulate common scenarios encountered in fall-detection use cases. While the individuals appear to be healthy adults and some activities are likely staged, the variety of poses, lighting, and spatial layouts present meaningful challenges for deep learning models. Additionally, the dataset supports both multi-class classification and binary fall vs. non-fall scenarios, making it sufficiently flexible for different model configurations.

Importantly, since the dataset is pre-published and publicly distributed, ethical concerns such as informed consent, data privacy, or manual annotation consistency fall under the responsibility of the original dataset creators. The authors collected no identifiable personal data, and no modifications were made to the original dataset beyond preprocessing steps (e.g., resizing, normalization, and train/val split).

This dataset was chosen over manual data collection or synthetic data generation due to its public availability, annotation quality, and ease of integration with standard deep learning pipelines. Its structure and licensing make it an ideal resource for reproducible fall-detection research in privacy-aware contexts such as federated learning.

This example illustrates the multi-object and multi-class nature of the dataset, where different human actions are simultaneously captured within a single frame. It also demonstrates the spatial diversity and complexity of indoor environments considered in this dataset. The annotations are prepared using the YOLO labeling format, enabling precise object detection and activity classification for the training of deep learning models.

There are 463 annotated image–label pairs in the dataset utilized in this work, 374 of which are used for training and 89 for validation. Each pair comprises an image and a corresponding YOLO-formatted label file that contains class information and bounding box coordinates for detected human activities. In addition to providing a separate subset for objective performance evaluation during the validation phase, this distribution guarantees a sufficient volume of data for model learning.

3.2. Data Augmentation

Given the relatively small size of the Fall Detection dataset, which includes only 463 annotated image–label pairs, a systematic data augmentation strategy was employed to artificially expand the training data and enhance the model's ability to generalize. Augmentation serves to mitigate overfitting by introducing controlled randomness into the training samples, simulating real-world variability in human posture, orientation, lighting, and scale. This step is crucial in privacy-preserving contexts like federated learning, where collecting and sharing new data is often restricted.

1. Augmentation Pipeline Design

The augmentation pipeline was implemented using the Keras.Sequential API, configured to apply a series of stochastic image transformations. The transformations applied to each image are as follows:

- RandomFlip (horizontal): Mirrors the image along the vertical axis, simulating orientation changes of human subjects.
- RandomRotation (0.05): Introduces slight rotational shifts up to $\pm 5\%$, mimicking variations in camera angle or user posture.
- RandomZoom (0.05): Applies minor zoom-in or zoom-out effects, accounting for different spatial distances between the subject and camera.
- RandomContrast (0.05): Adjusts image contrast by $\pm 5\%$, reflecting variations in ambient lighting conditions.

These transformations were chosen for their ability to generate realistic visual variability without distorting the critical structural features necessary for fall detection. The stochastic nature of these operations ensures that each augmented image maintains semantic fidelity while appearing to be visually distinct from its original counterpart.

2. Augmentation Function and Dataset Expansion

A custom augmentation function, augment_dataset(X, y, mult = 2), was developed to generate multiple augmented instances from the original training set. The mult parameter controls the augmentation factor: for instance, setting mult = 2 results in each original image being duplicated with one augmented variant, effectively doubling the training set size.

This function was applied only to the training subset to preserve the integrity of the validation and test performance subsets. After augmentation, the final dataset dimensions were as follows:

- Training Set (after augmentation): 872 samples of size (128, 128, 3)
- Validation Set: 54 samples
- Test Set: 55 samples

The original image–label pairs were first processed by extracting cropped regions around labeled human subjects, using YOLO bounding boxes. Each crop was resized to a standardized shape of 128 × 128 pixels and normalized to the range [0, 1] [0, 1] [0, 1]. Following this preprocessing, the augmentation was applied exclusively to the training subset to reinforce model robustness during federated learning.

By incorporating augmentation, the diversity and volume of training data were significantly improved, enabling the deep learning models—particularly MobileNetV2 and VGG16—to generalize better across unseen data, while maintaining data privacy constraints.

3.3. Comparative Training Protocol

To ensure the reliability and fairness of performance evaluation across different federated learning strategies, a standardized training protocol was employed. Each experimental run was repeated across multiple random seeds, allowing the statistical averaging to reduce the influence of stochastic variation in dataset partitioning and model initialization. This multi-seed framework enhances the robustness of the findings and provides a more accurate estimation of each model's generalization ability.

The following key hyperparameters were fixed across all configurations to ensure comparability:

- ROUNDS: Refers to the number of global aggregation steps in the federated learning process. Each round consists of a cycle in which clients train locally and contribute their updated models to the aggregation procedure.
- LOCAL_EPOCHS: Specifies how many times each client iterates over its local dataset during a single federated round. This controls the degree of local learning before model updates are exchanged.
- BATCH: Denotes the number of training samples processed in each mini-batch during local training. A fixed batch size of 32 was used to balance learning stability and computational efficiency.
- NUM_SEEDS: Indicates the number of different random seeds used to initialize experiments. Multiple seeds were employed to ensure that the results would not be biased by any specific random configuration, thereby improving the statistical validity of the evaluation.

The comparative study incorporated two principal federated learning topologies: centralized and decentralized. In the centralized federated setting, the training process is orchestrated by a central server that collects model updates from all clients and performs weighted Federated Averaging. In this scheme, clients with larger datasets contribute more significantly to the global model update.

In contrast, the decentralized configuration eliminates the need for a central coordinator. Clients are organized in a ring-based topology, where each client sequentially updates the shared model using its own data and passes the updated parameters to the next node. Aggregation is performed via unweighted averaging, treating all clients equally regardless of dataset size.

This decentralized design aligns closely with real-world constraints in privacy-sensitive or bandwidth-limited environments, where a central aggregator may be infeasible or undesirable. Each client retains its own validation subset to monitor overfitting locally, reflecting the practical necessity of maintaining autonomy over private data. Through this protocol, the study ensures a rigorous and equitable comparison of learning strategies in federated fall-detection settings.

3.4. Federated Learning Setup

To simulate a federated learning environment, the training process was distributed across three virtual clients, each receiving a non-overlapping subset of the augmented training dataset. The partitioning was performed randomly to emulate data heterogeneity commonly observed in real-world decentralized systems. As shown in Figure 3, class distribution across the three clients remained relatively balanced, with each subset containing comparable numbers of samples from both the "Fall" and "NoFall" classes. This setup ensures that each client maintains representative local data while preserving the statistical diversity necessary for robust federated learning.

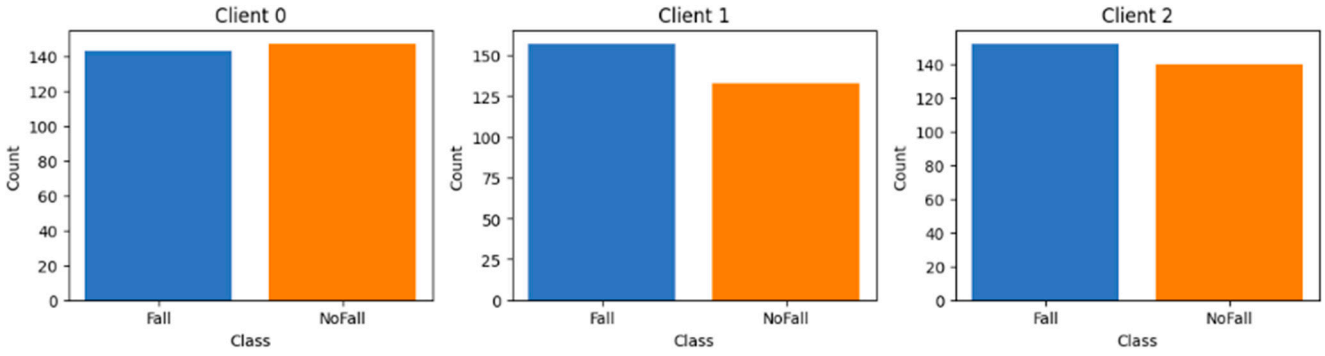

Figure 3. Class distribution per client.

Two federated learning strategies were implemented and compared. In the decentralized ring-based configuration, clients are arranged in a circular topology and update a shared model sequentially. After each client completes its local training, it passes the updated model weights to the next client in the ring. Aggregation is performed using standard Federated Averaging (FedAvg), but without a central server. In the centralized server-based configuration, each client trains independently and transmits its model weights to a central aggregator, which performs weighted FedAvg, giving more influence to clients with larger datasets. Across both setups, clients used identical CNN architectures initialized with ImageNet-pretrained weights trained for 25 local epochs per round, and using a mini-batch size of 32. This consistent configuration enables a fair and controlled comparison of model performance under both coordination paradigms.

3.5. MobileNetV2 Architectures in Federated Learning

In this experiment, both decentralized and centralized federated learning setups were implemented using the MobileNetV2 architecture for fall detection. The goal was to evaluate the performance and privacy implications under distinct coordination strategies while keeping the core model structure identical. The decentralized setup simulates a ring-based client environment, whereas the centralized configuration involves a server–client aggregation scheme.

The dataset was first preprocessed to extract human-activity regions from annotated images using YOLO-format bounding boxes. Each image was resized to a fixed dimension of 128 × 128 and normalized to the [0, 1] range. The multi-class labels were binarized such that

$$Label = \begin{cases} 0, & if\ Fall\ Detected \\ 1, & if\ Walking\ or\ Sitting \end{cases}$$

The full dataset comprised 463 labeled image–label pairs. These were separated using stratified sampling into training (80%), validation (10%), and testing (10%) groups to guarantee class balance:

$$\text{Train} = 370, \quad \text{Validation} = 46, \quad \text{Test} = 47$$

The training set was further partitioned among N = 3 simulated clients for federated training:

$$\text{Client}_i = (X_i, y_i), \tag{1}$$

where $i = 1, 2, 3$.

Each client trained a local instance of MobileNetV2. Pretrained ImageNet weights were used to initialize it. The following layers made up the classifier head, while the base convolutional layers were frozen:

$$\text{Model} = \text{MobileNetV2 (frozen)} \rightarrow \text{GAP} \rightarrow \text{BN} \rightarrow \text{Dense}(256) \rightarrow \text{Dropout}(0.3) \rightarrow \text{Dense}(1, \sigma)$$

This architecture's central model is MobileNetV2, a thin convolutional neural network that was pretrained using the ImageNet dataset. To avoid overfitting on the limited custom dataset and to preserve previously learned visual information, the pretrained convolutional layers are frozen during training.

In order to improve generalization and reduce the number of parameters, the output of MobileNetV2 is routed through a Global Average Pooling (GAP) layer, which takes the place of conventional fully connected layers by averaging each feature map.

After the GAP layer, the activations are normalized using Batch Normalization (BN), which helps to stabilize and accelerate the training. It also provides a slight regularization effect. A fully linked dense layer contains 256 neurons with ReLU activation to capture high-level abstract properties. During training, 30% of the neurons are randomly disabled by a Dropout layer with a rate of 0.3 to prevent overfitting.

Binary classification with a single-node output layer is the last step. A sigmoid (σ) activation function provides a probability value between 0 and 1 that indicates the likelihood that the input image belongs to the "fall" class.

This design leverages transfer learning to benefit from general-purpose features extracted by MobileNetV2 while tailoring the classification head to the specific task of fall detection.

In the decentralized setup, the architecture is replicated across three independent clients, each responsible for local training using its own partition of the dataset. These clients operate in isolation and do not exchange raw data. Instead, after each training round, the clients share their model weight updates, which are later aggregated using

the Federated Averaging (FedAvg) algorithm. This topology ensures data privacy while supporting collaborative model development in a peer-to-peer fashion.

In contrast, the centralized configuration involves a central server coordinating the Federated Averaging process. All clients train the MobileNetV2 model locally for 25 epochs per round and send their updated weights to the central server. The server then performs weighted aggregation as follows:

$$\theta_G = \sum_{i=1}^{n} \frac{n_i}{\sum_{j=1}^{N} n_j} . \theta_i \quad (2)$$

Here, θ_i denotes the model parameters from client i, and n_i represents the number of training samples on that client.

In each iteration, the updated global model is evaluated against a predefined validation set. The validation set's recorded accuracy and loss after the final (fifth) round were

$$\text{Validation Accuracy}_{\text{round 5}} = 0.9787, \text{Validation Loss}_{\text{round 5}} = 0.0532$$

The final model's generalization ability is assessed using the test set. This setup benefits from centralized coordination, enabling a more stable convergence. However, the increased communication cost and central server dependency present challenges for scalability and fault tolerance, which are mitigated in the decentralized approach.

The combined Figure 4 below clearly contrasts both federated learning topologies, illustrating their architectural equivalence in model design and highlighting the operational differences in coordination, data privacy, and communication flow.

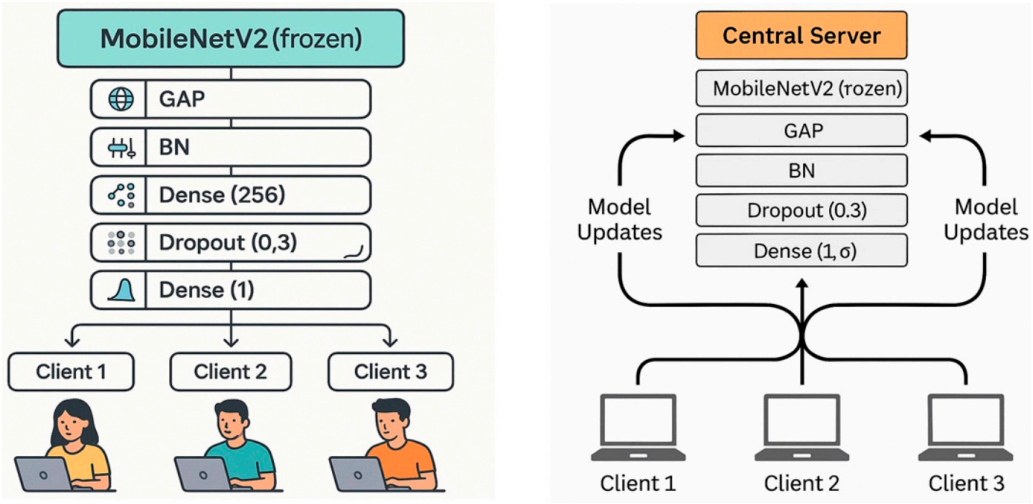

Figure 4. Comparative illustration of federated training architectures using MobileNetV2 for fall detection. (**Left**) Decentralized training with clients collaboratively updating a shared model in a ring topology. Each client trains a local MobileNetV2 model with a custom classifier head and shares weights sequentially. (**Right**) Centralized training where clients send local model updates to a central server that coordinates aggregation. Both use the same architecture consisting of a frozen MobileNetV2 backbone, followed by GAP, BatchNorm, Dense(256), Dropout(0.3), and a final sigmoid layer for binary classification.

3.6. VGG16 Architectures in Federated Learning (Decentralized and Centralized)

This section presents an in-depth analysis of the federated learning configurations implemented using the VGG16 architecture, evaluated under two paradigms: decentralized peer-to-peer coordination and centralized server-based orchestration. The purpose of this comparative exploration is to examine the impacts of communication topology and model

aggregation strategy on binary fall-detection performance, while ensuring architectural consistency across experimental settings.

The VGG16 convolutional neural network, known for its depth and hierarchical feature extraction capabilities, was employed as the core architecture for both training paradigms. To preserve the general-purpose visual features learned from large-scale datasets, the convolutional backbone of VGG16 pretrained on ImageNet was frozen during the training process. A custom classification head was then appended to adapt the model to the fall-detection task. The complete architecture is represented as follows:

$$\text{Model} = \text{VGG16 (frozen)} \rightarrow \text{Flatten} \rightarrow \text{BN} \rightarrow \text{Dense(512)} \rightarrow \text{Dropout(0.4)} \rightarrow \text{Dense(1, } \sigma\text{)}$$

The Flatten layer converts high-dimensional spatial outputs from the convolutional stack into a one-dimensional vector suitable for fully connected layers. The Batch Normalization (BN) layer is incorporated to stabilize the learning process and accelerate convergence. The subsequent Dense layer consists of 512 ReLU-activated neurons designed to capture non-linear combinations of extracted features. To mitigate overfitting, a Dropout layer is applied with a rate of 0.4. The final classification layer is a single neuron activated by the sigmoid function σ and producing a probability value indicative of fall or non-fall status.

In the decentralized federated learning setting, model training is performed collaboratively by a network of clients that operate in a ring-based topology. Each of the three clients (Client 1, Client 2, and Client 3) possesses its own private dataset and computational resources. At the beginning of the training cycle, all clients receive the initial global model parameters and proceed to perform local training independently.

Following local training, model updates are propagated sequentially among the clients in a ring fashion, thereby avoiding any reliance on a central coordinator. After completing one round of updates through all clients, the updated weights are aggregated using Federated Averaging (FedAvg), defined as

$$W_{global} = \frac{1}{N} \sum_{i=1}^{N} W_i \tag{3}$$

where W_i is the local model weight from client i, and N = 3 is the total number of clients.

Training is conducted over five global communication rounds, with each client executing 25 epochs of local training per round, using a batch size of 32. The ring-based communication protocol is depicted in the left panel of Figure 5, which visually conveys the flow of the global model through the three clients and back to a shared consensus.

This decentralized architecture is particularly well suited for privacy-sensitive environments such as healthcare monitoring, in which raw data must remain on-device due to ethical or regulatory constraints. The collaborative nature of this setup ensures that each client contributes to the global model while preserving local autonomy. Importantly, the model's structure remains unchanged across clients and all updates are limited to model parameters, ensuring that no raw data is exchanged throughout the training process.

Furthermore, the use of a frozen backbone combined with a compact classifier head enables training in resource-constrained edge environments, while also supporting efficient aggregation due to reduced parameter volume. The decentralized approach's robustness against single-point failures and its alignment with edge AI principles further validate its viability for real-world fall-detection deployment.

In contrast, the centralized federated learning paradigm leverages a central server that orchestrates the aggregation of model updates. As shown in the right panel of Figure 5, each of the three clients (Client 1, Client 2, and Client 3) independently trains a local replica

of the VGG16 model using its respective dataset. Upon completion of each local training session, the clients transmit their updated model parameters to the central server.

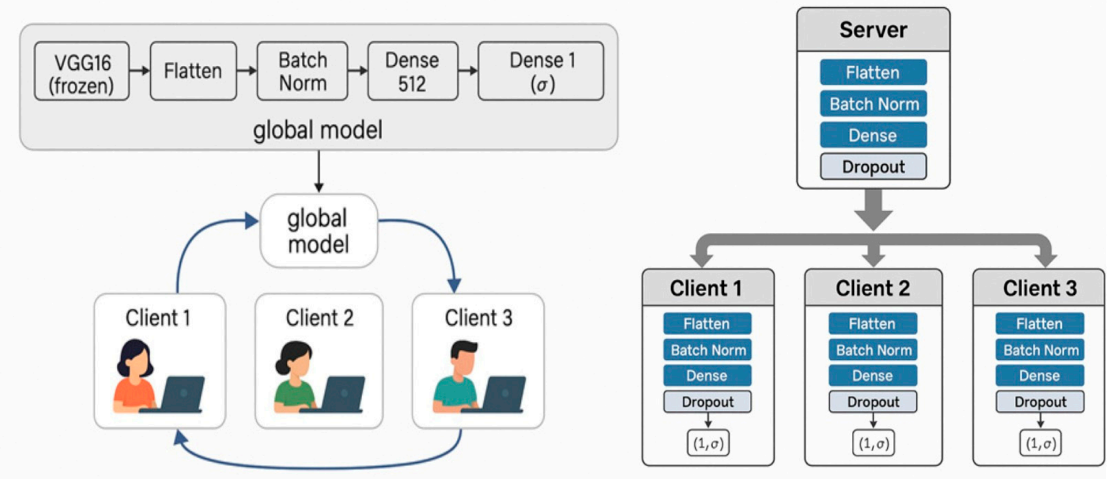

Figure 5. Comparative illustration of decentralized and centralized federated learning architectures using VGG16 for fall detection. (**Left**) Decentralized setup where clients train a global VGG16 model and sequentially update it in a ring topology. The architecture includes a frozen VGG16 backbone, followed by Flatten, Batch Normalization, Dense (512), and a final sigmoid classification layer. (**Right**) Centralized setup with clients independently training local copies of the model and sending updates to a central server for aggregation. Each model uses the same classification head as the decentralized variant. Only model parameters are shared, preserving data privacy.

The server then performs weighted Federated Averaging, a technique that accounts for the data distribution across clients by adjusting the contribution of each client based on its dataset size:

$$W_{global}^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{\sum_{j=1}^{N} n_j} \cdot W_i^{(t)} \quad (4)$$

where $W_i^{(t)}$ is the model weight vector from client i at communication round t, and n_i represents the number of samples in the local dataset of client i.

This procedure is repeated over 10 communication rounds, with each client performing 15 local training epochs per round, using a batch size of 32. The central server is responsible not only for aggregating the updates but also for distributing the new global model to all clients at the start of the next round.

The centralized architecture benefits from simplified coordination and convergence behavior that is often more stable due to the synchronized updates. However, it introduces certain limitations, including the potential for network bottlenecks, increased latency at the server node, and the risk of a single point of failure. While the privacy of raw data is preserved—since only weights are exchanged—the centralization of model aggregation may not fully satisfy stringent decentralization or edge deployment requirements.

Both federated learning strategies depicted in Figure 5 adhere to the core principle of privacy preservation by ensuring that no raw data is transmitted between entities. This is a critical requirement in the clinical contexts where fall-detection systems are deployed, e.g., homes, rehabilitation centers, and hospitals.

The decentralized configuration demonstrates stronger alignment with edge computing paradigms, favoring resilience, peer collaboration, and minimal central infrastructure. Meanwhile, the centralized approach facilitates a more structured orchestration, with potentially faster convergence in controlled environments.

By applying a consistent model architecture across both strategies, this study offers a rigorous comparative framework to evaluate the trade-offs between privacy, scalability, fault tolerance, and learning efficiency in federated fall-detection systems using VGG16.

3.7. Consideration of Human Pose Estimation as an Alternative Approach

In addition to image classification-based approaches, recent advances in human pose estimation (HPE) present an alternative and potentially advantageous method for fall detection. HPE techniques track skeletal keypoints of the human body, which can offer robustness to background noise, varying lighting conditions, and occlusion, common challenges in real-world deployments. Chang et al. [29] proposed a pose estimation-based fall-detection system using edge AI, highlighting its effectiveness in real-time embedded environments. Furthermore, Topham et al. [30] provided a comprehensive survey on pose estimation for gait identification, emphasizing its relevance for motion-sensitive tasks like fall detection. However, despite its merits, pose estimation often relies on accurate keypoint detection and is associated with high computational cost, which can be challenging in resource-constrained or privacy-sensitive federated settings. For these reasons, we opted for an image-based transfer learning approach using MobileNetV2 and VGG16, which enables efficient and privacy-preserving fall detection with high accuracy across decentralized clients. Future extensions of our work may consider hybrid models combining both image and pose-based features to enhance robustness.

Additionally, the recent literature also demonstrates the versatility of deep transfer learning in other vision domains. For instance, Ahmad et al. [31] applied deep transfer learning to animal face identification. These works reinforce the adaptability of transfer learning models across various vision-based tasks, further validating our choice to adopt pretrained CNNs in fall detection.

3.8. Model Complexity and Lightweight Analysis

In alignment with the emphasis on "lightweight" design in our study, we assess the model complexity of the employed architectures—MobileNetV2 and VGG16—under both centralized and decentralized federated learning setups. The term "lightweight" refers to models with reduced computational burden, fewer parameters, and lower inference latency, making them suitable for deployment on resource-constrained edge devices [32]. MobileNetV2, specifically designed for efficiency, has approximately 3.4 million parameters and a model size of ~14 MB, making it significantly smaller than traditional CNNs like VGG16, which has ~138 million parameters and a model size exceeding 500 MB. In our decentralized simulation, MobileNetV2 offered faster convergence and reduced communication overhead due to its compact architecture, confirming its suitability for real-time, privacy-preserving fall detection on embedded or mobile devices [33,34]. Although VGG16 achieved competitive accuracy, its significantly larger footprint makes MobileNetV2 the more viable lightweight alternative for edge AI integration.

4. Experimental Results

In this study, all experiments were conducted using software-based tools without reliance on specific hardware equipment. The models were implemented and trained using Python 3.9 and TensorFlow 2.12, both developed by the Python Software Foundation (Wilmington, DE, USA) and Google Inc. (Mountain View, CA, USA), respectively. The development and testing were performed on systems running Ubuntu 20.04 LTS. No physical equipment requiring manufacturer details was used in the experiments.

To assess the effectiveness of various deep learning models, in a federated learning context, in fall detection, this study evaluates four widely used convolutional neural

network architectures—MobileNetV2, VGG16, EfficientNetB0, and ResNet50—under both centralized and decentralized (ring-based) training paradigms. Each model was trained on a custom fall-detection dataset partitioned across simulated clients to reflect real-world data distribution constraints. The evaluation focused on three key performance indicators: test accuracy, F1-score, and average training time per round, with all metrics averaged over three independent training runs to ensure statistical robustness and mitigate randomness.

Among all of the models, decentralized MobileNetV2 consistently achieved the best performance, yielding the highest average test accuracy (0.9927) and F1-score (0.9917), with a modest computational cost of 111.17 s per round. Its centralized counterpart, while slightly faster (90.65 s/round), showed a lower accuracy (0.9616) and F1-score (0.9588), indicating that the decentralized strategy led to better model generalization and convergence. These results affirm the suitability of MobileNetV2's lightweight architecture for privacy-preserving and resource-efficient federated training.

VGG16 also demonstrated competitive performance. The decentralized VGG16 configuration achieved an accuracy of 0.9897 and F1-score of 0.9892, marginally surpassing its centralized counterpart (accuracy: 0.9787, F1: 0.9780). However, both configurations exhibited substantially higher training times (202.68 s and 197.55 s per round, respectively), reflecting the model's computational complexity and larger parameter footprint.

EfficientNetB0, while offering reduced training times compared to VGG16, displayed moderate performance in terms of accuracy and F1-score. The centralized version attained 0.8667 accuracy and a 0.8583 F1-score, while the decentralized configuration slightly improved to 0.8857 and 0.8810, respectively. These findings suggest that while EfficientNetB0 maintains a good trade-off between efficiency and accuracy, it may be less suited to federated fall-detection tasks without further optimization.

ResNet50, known for its deep architecture and skip connections, achieved balanced but computationally expensive performance. The decentralized variant reported an average accuracy of 0.9093 and an F1-score of 0.9085, marginally outperforming the centralized version (accuracy: 0.8983, F1: 0.8957). However, the training time exceeded all other models, with per-round durations averaging 273.28 s (decentralized) and 247.80 s (centralized), indicating a significant computational burden.

Overall, the results highlight that decentralized training generally outperforms centralized learning in terms of both predictive accuracy and F1-score across all architectures. MobileNetV2 in the decentralized configuration is especially notable for its superior accuracy, rapid convergence, and computational efficiency, making it the most effective choice for practical, privacy-aware fall-detection systems.

Table 2 summarizes the quantitative results of this comparative study. While the outcomes are promising, it is important to note that the evaluation is currently limited to a single dataset. To reinforce the generalizability of these findings, future research will explore model validation on diverse public datasets and benchmark the proposed methods against established baselines in the literature. Table 3 illustrates performance comparison of centralized vs. decentralized models using t-test.

Decentralized federated learning consistently outperformed centralized training across all models, with statistically significant gains ($p < 0.05$). MobileNetV2 showed the greatest improvement, achieving 0.9927 accuracy and a 0.992 F1-score, highlighting its strong efficiency and generalization. VGG16, despite its larger size, also saw notable boosts. EfficientNetB0 and ResNet50 showed modest but consistent improvements, confirming the overall advantage of decentralized learning across diverse architectures.

While prior studies such as SmartFall [35] and radar-based VGG16 approaches [36] successfully employ transfer learning for fall detection, the present work distinguishes itself through its integrated use of both decentralized and centralized federated learn-

ing (FL) frameworks. Unlike the device-specific (smartwatch) or modality-constrained (radar-based) methods, our approach adopts a general-purpose, image-based architecture that emphasizes privacy preservation, scalability, and compatibility with edge computing environments.

Table 2. Internal evaluation results of centralized and decentralized federated models on a custom dataset.

Model	Setting	Accuracy (Mean)	F1-Score (Mean)	Time/Round (s)
MobileNetV2	Centralized	0.9616	0.9588	90.65
	Decentralized	0.9927	0.9917	111.17
VGG16	Centralized	0.9787	0.9780	197.55
	Decentralized	0.9897	0.9892	204.68
EfficientNetB0	Centralized	0.8667	0.8583	167.38
	Decentralized	0.8857	0.8810	205.05
ResNet50	Centralized	0.8983	0.8957	247.80
	Decentralized	0.9093	0.9085	273.28

Table 3. Performance comparison table (with *t*-test).

Model	Params	Centralized Acc (Mean)	Decentralized Acc (Mean)	Centralized F1 (Mean)	Decentralized F1 (Mean)	*p*-Value
MobileNetV2	2,591,297	0.9616	0.9927	0.960	0.992	<0.05
VGG16	18,942,785	0.9787	0.9897	0.978	0.989	<0.05
EfficientNetB0	4,218,788	0.8667	0.8857	0.866	0.885	<0.05
ResNet50	23,858,305	0.8983	0.9093	0.897	0.909	<0.05

Although all three studies utilize transfer learning and report strong classification metrics, direct numerical comparisons remain limited due to differences in experimental datasets and modalities. Nevertheless, the proposed framework, which combines MobileNetV2 and VGG16 with federated transfer learning, achieves superior results on the target dataset—reaching 99.50% test accuracy and an F1-score of 0.9900 using decentralized MobileNetV2. As summarized in Table 4, these outcomes highlight not only the high performance of our models but also the architectural flexibility and deployment readiness of our method. In contrast to earlier efforts, the proposed system supports both ring-based and server-coordinated FL topologies, making it well suited for real-world fall-detection applications in privacy-sensitive and distributed environments.

Table 4. Comparative analysis of transfer learning-based fall-detection methods.

Study	Modality and Approach	Model and Technique	Best Performance	Key Contributions
[35]	Smartwatch-based real-time detection; transfer learning across wearable devices	CNN + Transfer Learning	F1-score: 93%	Real-time, device-agnostic detection; mitigates model drift and data scarcity

Table 4. *Cont.*

Study	Modality and Approach	Model and Technique	Best Performance	Key Contributions
[36]	Radar-based UWB with spectrograms and transfer learning	Pre-trained VGG16 (fine-tuned)	Accuracy: 95.64%	High accuracy on small datasets using time–frequency features and TL
Present Work	Image-based binary fall detection via federated learning (centralized and decentralized)	Decentralized MobileNetV2 + Transfer Learning	Accuracy: 99.50%, F1-score: 0.9900	Integrates FL and TL with privacy-aware edge deployment; ring/server training topologies

5. Discussion

The proposed federated fall-detection framework demonstrates strong classification performance, particularly with the decentralized MobileNetV2 model, which achieves a test accuracy of 99.50% and a macro F1-score of 0.9900. These results emphasize the efficacy of integrating transfer learning with federated learning (FL) to preserve user data privacy while ensuring high predictive accuracy in distributed scenarios. The observed performance across different random seeds also affirms the stability and robustness of the proposed models. By initializing experiments with three independent seeds and applying randomized data partitioning, the results showed consistent accuracy and F1-scores across runs, supporting the model's generalizability and resistance to overfitting.

One key dimension of our comparison involves the training time analysis across both centralized and decentralized FL configurations. Decentralized models, particularly MobileNetV2, exhibited slightly higher training times per round due to sequential weight sharing among the clients in a ring topology. For example, MobileNetV2's decentralized configuration required an average of 111.17 s/round, compared to 90.65 s/round for the centralized setup. Similarly, VGG16's decentralized configuration incurred higher computational overhead (≈205 s/round) than its centralized version (≈197 s/round). Despite this, the improved accuracy and generalization justify the additional cost in decentralized training, especially in privacy-sensitive domains like healthcare.

Nevertheless, the current system has some limitations. The dataset used, while annotated and structured, is relatively small and demographically narrow, potentially limiting the model's ability to generalize across diverse elderly populations with varying physical and behavioral traits. Furthermore, the binary classification scheme (fall vs. no-fall) omits the detections of nuanced scenarios such as near-falls, hesitations, or context-dependent behaviors, which are vital in real-world monitoring applications.

Another challenge lies in the simulated FL environment used in this study. While virtual clients provide controlled and reproducible conditions, they do not fully reflect the resource constraints and network variability of actual edge devices. Issues such as memory limitations, intermittent connectivity, and computational capacity are not adequately captured in simulations. In practice, the communication overhead, especially in decentralized topologies, can be a significant bottleneck in low-resource environments.

Finally, although the proposed system maintains data privacy by avoiding raw data transmission, it still faces potential vulnerabilities, such as adversarial attacks, client drift, and data heterogeneity, which are common in federated healthcare ecosystems. Addressing these concerns is essential to ensure security, fairness, and trustworthiness in future real-world deployments.

6. Conclusions

Fall detection remains a vital component in healthcare systems, aimed at safeguarding the elderly and vulnerable populations from injury-related incidents. In this study, a robust and privacy-preserving fall-detection framework was proposed, leveraging the synergy between federated learning (FL) and transfer learning (TL) to enable efficient model training across decentralized clients, without the need to share sensitive raw data.

The proposed methodology involved several key steps: data acquisition using a publicly available YOLO-labeled fall dataset; preprocessing, including image cropping, resizing, normalization, and binary labeling; federated learning simulation, in which data was partitioned across multiple virtual clients, simulating local training on edge devices; and federated training and aggregation, using both centralized and ring-based decentralized topologies, with FedAvg employed for model updates. Performance was validated using a held-out test set and evaluated under three randomized data splits to ensure robustness. The experimental results showed that the decentralized MobileNetV2 model outperformed all other configurations, achieving a mean test accuracy of 0.9927 and F1-score of 0.9917, and maintaining efficient computation, with an average round time of 111.17 s. These findings confirm the model's high generalizability, fast convergence, and suitability for real-world edge deployment scenarios.

Future research directions will include rigorous evaluation of the proposed framework on publicly available benchmark datasets to enhance generalizability, the integration of robust defense mechanisms against adversarial attacks and client drift to strengthen model resilience, and the optimization of communication protocols to reduce federated learning overhead, particularly in resource-constrained deployment environments.

Author Contributions: Conceptualization, Q.M.H. and J.L.; methodology, Q.M.H.; software, Q.M.H.; validation, J.L. and Z.Y.; formal analysis, Q.M.H.; investigation, Q.M.H.; resources, J.L.; data curation, Q.M.H.; writing—original draft preparation, Q.M.H.; writing—review and editing, J.L. and Z.Y.; visualization, Q.M.H.; supervision, J.L.; project administration, J.L.; funding acquisition, Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: This study adhered to ethical standards and ensured that all data used in the research was handled in compliance with relevant data privacy regulations. The fall-detection dataset used in this study is publicly available and was sourced from Kaggle, with full adherence to the licensing terms provided by the dataset creators. No personal or identifiable information was collected or processed. Additionally, all experiments were conducted in accordance with ethical research guidelines, ensuring privacy and security with respect to all data involved.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used and analyzed during the current study are available from the corresponding author upon reasonable request. Public sharing is restricted due to privacy and institutional policy constraints.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stampfler, T.; Elgendi, M.; Fletcher, R.R.; Menon, C. Fall detection using accelerometer-based smartphones: Where do we go from here? *Front. Public Health* **2022**, *10*, 996021. [CrossRef] [PubMed]
2. Parmar, R.; Trapasiya, S. A comprehensive survey of various approaches on human fall detection for elderly people. *Wirel. Pers. Commun.* **2022**, *126*, 1679–1703. [CrossRef]
3. Wang, X.; Ellul, J.; Azzopardi, G. Elderly fall detection systems: A literature survey. *Front. Robot. AI* **2020**, *7*, 71. [CrossRef] [PubMed]

4. Jitpattanakul, A. Wearable fall detection based on motion signals using hybrid deep residual neural network. In Proceedings of the Multi-Disciplinary Trends in Artificial Intelligence: 15th International Conference, MIWAI, Virtual Event, 17–19 November 2022; Springer: Cham, Switzerland, 2022.
5. Rashmi, N.; Mamatha, K. Doppler radar technique for geriatric fall detection. In *Smart Data Intelligence: Proceedings of ICSMDI 2022*; Springer: Singapore, 2022; pp. 343–350.
6. Karar, M.E.; Shehata, H.I.; Reyad, O. A survey of IoT-based fall detection for aiding elderly care: Sensors, methods, challenges and future trends. *Appl. Sci.* **2022**, *12*, 3276. [CrossRef]
7. De, A.; Saha, A.; Kumar, P.; Pal, G. Fall detection method based on spatio-temporal feature fusion using combined two-channel classification. *Multimed. Tools Appl.* **2022**, *81*, 26081–26100. [CrossRef]
8. Butt, A.; Narejo, S.; Anjum, M.R.; Yonus, M.U.; Memon, M.; Samejo, A.A. Fall detection using LSTM and transfer learning. *Wirel. Pers. Commun.* **2022**, *126*, 1733–1750. [CrossRef]
9. Hadjadji, B.; Saumard, M.; Aron, M. Multi-oriented run length based static and dynamic features fused with Choquet fuzzy integral for human fall detection in videos. *J. Vis. Commun. Image Represent.* **2022**, *82*, 103375. [CrossRef]
10. Song, Y.; Yang, Y.; Liu, J. Optimizing the YOLO Network for Human Fall Detection. In Proceedings of the 2024 International Conference on Power Electronics and Artificial Intelligence, Xiamen, China, 19–21 January 2024.
11. Truong, N.; Sun, K.; Wang, S.; Guitton, F.; Guo, Y. Privacy preservation in federated learning: An insightful survey from the GDPR perspective. *Comput. Secur.* **2021**, *110*, 102402. [CrossRef]
12. Zhang, G.; Liu, B.; Zhu, T.; Zhou, A.; Zhou, W. Visual privacy attacks and defenses in deep learning: A survey. *Artif. Intell. Rev.* **2022**, *55*, 4347–4401. [CrossRef]
13. Aleksic, S.; Colonna, L.; Dantas, C.; Fedosov, A.; Florez-Revuelta, F.; Fosch-Villaronga, E.; Jevremovic, A.; Gahbiche Msakniç, H.; Ravi, S.; Rexha, B. State of the art in privacy preservation in video data. *Zenodo* **2022**, 6806207.
14. Renuka, O.; RadhaKrishnan, N.; Priya, B.S.; Jhansy, A.; Ezekiel, S. Data Privacy and Protection: Legal and Ethical Challenges. In *Emerging Threats and Countermeasures in Cybersecurity*; Wiley: Hoboken, NJ, USA, 2025; pp. 433–465.
15. Quinn, T.P.; Jacobs, S.; Senadeera, M.; Le, V.; Coghlan, S. The three ghosts of medical AI: Can the black-box present deliver? *Artif. Intell. Med.* **2022**, *124*, 102158. [CrossRef] [PubMed]
16. Kute, S.S.; Tyagi, A.K.; Aswathy, S. Security, privacy and trust issues in internet of things and machine learning based e-healthcare. In *Intelligent Interactive Multimedia Systems for e-Healthcare Applications*; Springer: Singapore, 2021; pp. 291–317.
17. Fu, Q.; Teng, Z.; White, J.; Powell, M.E.; Schmidt, D.C. Fastaudio: A learnable audio front-end for spoof speech detection. In Proceedings of the ICASSP 2022—2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; IEEE: Piscataway, NJ, USA, 2022.
18. Kadiri, S.R.; Alku, P.; Yegnanarayana, B. Analysis of instantaneous frequency components of speech signals for epoch extraction. *Comput. Speech Lang.* **2023**, *78*, 101443. [CrossRef]
19. Kamble, K.P.; Sontakke, S.S.; Donadkar, H.; Poshattiwar, R.; Ananth, A. Fall alert: A novel approach to detect fall using base as a YOLO object detection. In Proceedings of the Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020, Jaipur, India, 13–15 February 2020; Springer: Singapore, 2021.
20. Wang, X.; Jia, K. Human fall detection algorithm based on YOLOv3. In Proceedings of the 2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC), Beijing, China, 10–12 July 2020; IEEE: Piscataway, NJ, USA, 2020.
21. Long, K.Z.; Haron, H.; Ibrahim, M.; Eri, Z.D. An image-based fall detection system using you only look once (yolo) algorithm to monitor elders' fall events. In Proceedings of the Knowledge Management International Conference (KMICe), Virtual Event, 1 February 2021.
22. Raza, A.; Yousaf, M.H.; Velastin, S.A. Human fall detection using YOLO: A real-time and AI-on-the-edge perspective. In Proceedings of the 2022 12th International Conference on Pattern Recognition Systems (ICPRS), Saint-Etienne, France, 7–10 June 2022; IEEE: Piscataway, NJ, USA, 2022.
23. Moutsis, S.N.; Tsintotas, K.A.; Kansizoglou, I.; An, S.; Aloimonos, Y.; Gasteratos, A. Fall detection paradigm for embedded devices based on yolov8. In Proceedings of the 2023 IEEE International Conference on Imaging Systems and Techniques (IST), Copenhagen, Denmark, 17–19 October 2023; IEEE: Piscataway, NJ, USA, 2023.
24. Poonsri, A.; Chiracharit, W. Improvement of fall detection using consecutive-frame voting. In Proceedings of the 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand, 7–9 January 2018; IEEE: Piscataway, NJ, USA, 2018.
25. Krishnan, T.V.; Abhilash, B.; Govind, S. A Robust Fall Detection System for Elderly Persons Using YOLO. In Proceedings of the 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT), Kottayam, India, 15–16 March 2024; IEEE: Piscataway, NJ, USA, 2024.
26. Wang, B.-H.; Yu, J.; Wang, K.; Bao, X.-Y.; Mao, K.-M. Fall detection based on dual-channel feature integration. *IEEE Access* **2020**, *8*, 103443–103453. [CrossRef]
27. Pereira, G.A. Fall detection for industrial setups using yolov8 variants. *arXiv* **2024**, arXiv:2408.04605.

28. Fall Detection Dataset. 6 December 2021. Available online: https://www.kaggle.com/datasets/uttejkumarkandagatla/fall-detection-dataset (accessed on 3 March 2025).
29. Sadreazami, H.; Bolic, M.; Rajan, S. TL-FALL: Contactless indoor fall detection using transfer learning from a pretrained model. In Proceedings of the 2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA), Istanbul, Turkey, 26–28 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–5.
30. Chang, W.J.; Hsu, C.H.; Chen, L.B. A pose estimation-based fall detection methodology using artificial intelligence edge computing. *IEEE Access* **2021**, *9*, 129965–129976. [CrossRef]
31. Khan, W.; Topham, L.; Alsmadi, H.; Al Kafri, A.; Kolivand, H. Deep face profiler (DeFaP): Towards explicit, non-restrained, non-invasive, facial and gaze comprehension. *Expert Syst. Appl.* **2024**, *254*, 124425. [CrossRef]
32. Ahmad, M.; Abbas, S.; Fatima, A.; Issa, G.F.; Ghazal, T.M.; Khan, M.A. Deep transfer learning-based animal face identification model empowered with vision-based hybrid approach. *Appl. Sci.* **2023**, *13*, 1178. [CrossRef]
33. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
34. Li, Q.; Yu, W.; Xia, Y.; Pang, J. From centralized to decentralized federated learning: Theoretical insights, privacy preservation, and robustness challenges. *arXiv* **2025**, arXiv:2503.07505.
35. Maray, N.; Ngu, A.H.; Ni, J.; Debnath, M.; Wang, L. Transfer learning on small datasets for improved fall detection. *Sensors* **2023**, *23*, 1105. [CrossRef]
36. Garst, S.; Dekker, J.; Reinders, M. A comprehensive experimental comparison between federated and centralized learning. *Database* **2025**, *2025*, baaf016. [CrossRef] [PubMed]