Fall Detection using Graph Convolutional Neural Networks

ML Singapore Group 13

Devanshu Bisht (A0234516J), Ratnavibusena Don Shahain Manujith, Zhou Xuyan (A0258361B), John Ashwin Letchmanan (A0233403W)

Abstract

Fall detection is a critical task in healthcare, particularly for elderly populations where timely intervention can significantly reduce morbidity. We implement and evaluate four Graph Convolutional Neural Network Models (GCN), each introducing a unique design perspective. **ShahainNet** learns graph structures and enhances spatial GCNs with optical flow-based velocity vectors to capture dynamic joint motion. **DevanNet** incorporates a fused Knowledge-Informed Neural Network with traditional ST-GCN model to improve it's robustness. **JohnNet** redefines graph connectivity using a unification of the spatial and temporal aspects, allowing the model to discover far dependencies. Lastly, **XuyanNet** introduces a hybrid architecture that combines spatial and spectral graph reasoning through an adaptive fusion mechanism. All models are evaluated on the Le2i and UR fall dataset, with results showing that each architectural innovation contributes meaningfully to performance.

Background

Introduction

Falls are a critical global concern. According to the World Health Organization (WHO), falls are the second leading cause of unintentional injury deaths worldwide, accounting for approximately 684,000 deaths annually (World Health Organization 2021). As the global population ages, the prevalence and impact of falls are becoming increasingly pronounced. Several factors contribute to the rising rates of falls:

- **Physiological changes:** Ageing leads to muscle atrophy and decreased muscle strength. It also results in impaired vision, slower reflexes, and problems with balance and gait.
- **Chronic health conditions:** Conditions such as Parkinson's disease, dementia, diabetic neuropathy (affecting the feet), and heart conditions (causing dizziness) increase the risk of falls.
- **Medications:** Polypharmacy can cause side effects like dizziness, drowsiness, or postural hypotension.
- **Environmental hazards:** Poor lighting, improper footwear, lack of handrails, and clutter increase the risk of falling.

A serious issue arises when an individual falls and is unable to get back up on their own—this is referred to as the "long lie." Remaining on the floor for an extended period can lead to disastrous consequences. If the individual has suffered serious injuries, such as traumatic brain injury (TBI) leading to subdural hematomas, delayed assistance can mean the difference between life and death. Prolonged immobility can also result in hypothermia (a dangerous decrease in body temperature), dehydration (due to inability to reach fluids), rhabdomyolysis (muscle breakdown releasing harmful substances into the bloodstream, potentially leading to kidney failure), and respiratory difficulties (PhysioPedia n.d.).

This underscores the paramount importance of immediate attention to individuals who have suffered a fall. Prompt alerting of emergency services or caregivers can improve outcomes through faster treatment and minimize long-term complications.

Related Work

Early fall detection systems primarily used body-worn sensors to track movement or vision-based deep learning models to analyze video footage. However, sensor-based methods require user compliance, and models trained on raw video frames can be sensitive to background clutter and lighting variations. A major improvement came with pose estimation algorithms, which detect and track key body joints in video sequences. This enabled a more robust skeleton-based representation of posture and motion.

However, initial skeleton-based methods often relied on handcrafted features—such as joint velocities or displacement—and used simple classifiers like SVMs or 1D-CNNs to detect falls. While effective in basic scenarios, these approaches struggled to capture the complex, structured relationships in human motion. This is because the human skeleton is naturally represented as a graph, with joints as nodes and bones as edges. Treating joints as independent or forcing them into sequences can discard important relational information. To address this, Spatial-Temporal Graph Convolutional Networks (ST-GCNs) were introduced. These models improve interoperability and operate directly on skeleton graphs by applying two types of convolutions:

- In the **spatial domain**, graph convolution aggregates information from a node's neighbors to model joint connectivity. It is typically defined as: $\mathbf{H}^{(l+1)}_{\text{spatial}} = \sigma\left(\mathbf{A}_{\text{spatial}} \cdot \mathbf{H}^{(l)} \cdot \mathbf{W}^{(l)}_{\text{spatial}}\right)$, where $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times C}$ is the feature matrix at layer l, N is the number of joints (nodes), C is the number of input channels, $\mathbf{W}^{(l)}_{\text{spatial}} \in \mathbb{R}^{C \times C'}$ is a trainable weight matrix, σ is an activation function (e.g., ReLU).

- In the **temporal domain**, 1D convolution is applied across sequential frames to capture motion patterns over time. This is typically expressed as: $\mathbf{H}^{(l+1)}_{\text{temporal}} = \text{TCN}(\mathbf{H}^{(l+1)}_{\text{spatial}})$ where $\text{TCN}(\cdot)$ is a 1D convolution layer, often followed by non-linear activation and residual connections to maintain gradient flow.

This spatial-temporal modeling framework allows ST-GCNs to learn both posture and movement patterns directly from graph-structured skeleton data, leading to improved accuracy in tasks such as fall detection.

Our Solution

While there have been proposed solutions for fall detection, they have mostly relied on contact-based sensors like gyroscopes and accelerometers, often found in wearables. In contrast, we aim for no dependence on contact-based sensors and instead solely analyze video streams that are ubiquitous today, such as CCTV surveillance footage. Under the Assistive-living Technologies initiative by HDB, two product offerings currently exist: one uses radio frequency sensors to detect falls, and another detects screams. However, these solutions may suffer from high false negatives (missed falls) when detecting slow falls, and higher false positives due to the absence of captured imagery. They rely on changes in radio waves to detect sudden jerky movements, which can also be caused by non-human objects.

Our proposed solution utilizes pose estimation as features to accurately detect falls. The model learns joint movements and is trained with a loss function specifically designed for fall detection. This approach enables real-time analysis and timely alerts to a senior's emergency contact. Since we rely on pose estimation, high-resolution video is not required. Additionally, incorporating GCNs with pose estimation allows our model to achieve greater interoperability and robustness, as GCNs naturally model relationships between joints in a way that is invariant to changes in viewpoint, scale, or slight noise. This ensures the system can generalize better across different camera setups and environments without extensive retraining.

Our Novel Solutions

ShahainNet

We first address a major limitation in that we do not take into account for the physical dynamics at play. We resolve this issue by augmenting each node with keypoint velocities calculated using Lucas Kanade optical flow (see below for an illustration of an optical flow field).

This augmentation allows the graph neural network learn from positional and kinematic cues.

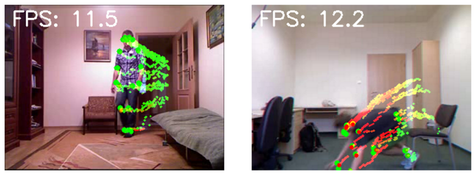

Figure 1: Optical flow fields comparing normal activity (left) and a fall event (right).

A big limitation of standard approaches is its reliance on an explicitly predefined graph structure based solely on the anatomical connections between MediaPipe keypoints. We aim to move beyond this simple topology to instead focus on graph structure learning to identify edge connections between MediaPipe keypoints that are most informative to predicting falls. We do this by extending transformers to graph neural networks wherein we construct a complete graph and learn which edges are most important to predicting a fall.

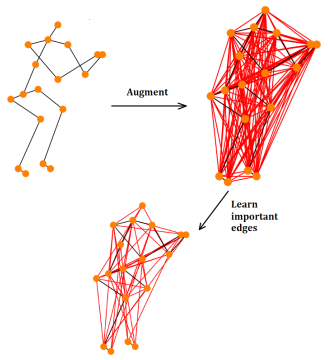

Figure 2: Graph structure learning

For a single frame f_i, we are given the set of 33 MediaPipe keypoints, X^i. Augmenting each node with the Lucas Kanade velocities in 3 dimensions, we have that each $x \in X^i$ is a vector in $\mathbb{R}^7$ described by its x, y, z coordinates and the spatial velocities v_x, v_y, v_z. Next, we form a complete graph with edges $X^i \times X^i$. This effectively connects each skeletal keypoint to every other keypoint. Defining the standard query and key weight matrices, W^θ_Q and W^θ_K, we project each keypoint into higher dimensional query and key spaces obtaining the new representations $Q = W^\theta_Q \cdot X^i$ and $K = W^\theta_K \cdot X^i$. Next we let each keypoint attend to every other keypoint by calculating the outer product $Q \cdot K^\top$. We immediately sparsify this attention map in order that we may know which edges are most informative in predicting falls. A simple way to perform sparsification would be to use Top-K selection, but results in a non-differentiable step preventing gradients from flowing back in order to learn W^θ_Q and W^θ_K. And this hard TopK criterion also prevents exploration since we are not sampling from this categorical distribution (which in itself is also non-differentiable). Instead we opt for Straight Through Gumbel Softmax that makes sampling from a categorical distribution (for each

keypoint, have to choose other keypoints that will be neighbors) differentiable. Specifically, we treat the attention score $s_{i,j}$ calculated for each edge $e_{i,j}$ as the unnormalized log-probability for the edge existing. Assuming that 0 defines the unnormalized log probobility of a non-existent edge, we create a logit vector $\boldsymbol{\ell}^{(i,j)} = [\ell^{(i,j)}_{\text{no\_edge}}, \ell^{(i,j)}_{\text{edge}}] = [0, s_{ij}]$ that represent unnormalized probabilities for the edge existence or not. Then to mimic noise, we sample from $g_0, g_1 \sim Gumbel(0,1)$ and create noisy logits which are then scaled by the temperature τ (when τ decreases, we get a sharper and discretized distribution). Now we are left with scaled and noisy logits $\boldsymbol{\ell}^{(i,j)}_{scaled,noisy} = [\frac{0+g_0}{\tau}, \frac{s_{ij}+g_1}{\tau}]$. Afterwards, with `hard=True` using `F.gumbel_softmax` we perform ST Gumbel Softmax which effectively finds the maximum out of the *no edge* and *edge* category. Crucially, because of the properties of the Gumbel destribution, taking the maximum *after* perturbing the logits with Gumbel noise is equivalent to sampling from the categorical distribution defined by the original logits. Hence, now we have determined the existence or non-existence of the edge based on the attentions scores via $idx = \text{argmax}([\frac{0+g_0}{\tau}, \frac{s_{ij}+g_1}{\tau}])$ and $c^{i,j} = \text{one\_hot}(\texttt{idx}, \texttt{num\_classes=2})$. For example one can have $[0,1]$ for *edge* and $[1,0]$ for *no_edge*. Even though taking the argmax is not differentiable, internally the Gumbel Softmax function calculates the "soft" probabilities before the "hard" argmax: $y^{i,j} = \text{softmax}(\frac{0+g_0}{\tau}, \frac{s_{ij}+g_1}{\tau})$ which is a vector that represents $[p_{no\_edge}, p_{edge}]$. This $y^{i,j}$ indeed is differentiable with respect to the logits $[0, s_{ij}]$. During the backward pass, the ST estimator uses the gradients arrived at the hard discrete output $c^{i,j}$ and passes it back by calculating the local gradient with respect to $y^{i,j}$. Even if the gradient obtained is an approximation (biased), this is often effective in practice. Effectively, we have now sparsified the complete graph to only contain edges that are most informative to predicting falls.
Next, instead of the standard approach of message passing (label propagation) where, given a node x_i (keypoint in our case),one would simply aggregate features of all 1-hop neighbors via $x_i' = \sigma\left(\sum_{j\in\mathcal{N}(x_i)} W^\theta * x_j\right)$, where the indices of the keypoints in the neighborhood of x_i are denoted by $N(x_i)$ and W^θ is the learnable shared weight matrix for this graph convolution layer. Instead we opt for a different approach where we aim to learn how important node x_k's features are important to updating node x_l's features using masked attention, where x_k and x_l are neighbors in our learned graph. This importance is called the attention coefficient, and is calculated for every edge in the graph via $\beta_{ij} = a(x_i^{'}, x_j^{'}) = \sigma(W_a^\phi(W^\theta x_i || W^\theta x_j))$. Here, W_a^ϕ the learnable weight for a neural network that is responsible for calculating the attention score from the concaternated new feature vectors $W^\theta x_i$ and $W^\theta x_j$. But because these attention coefficients for each edge can vary by a large magnitude over the entire graph, we normalize to get the normalized attention coefficient $\alpha_{i,j}$ for each edge $e_{i,j}$ via

$$\alpha_{i,j} = \text{softmax}_j(e_{ij}) = \frac{\exp(\beta_{ij})}{\sum_{k\in\mathcal{N}(x_i)} \exp(\beta_{ik})}$$
$$= \frac{\exp(\sigma(W_a^\phi(W^\theta x_i || W^\theta x_j)))}{\sum_{k\in\mathcal{N}(x_i)} \exp(\sigma(W_a^\phi(W^\theta x_i || W^\theta x_k)))}$$

These normalized attention scores α_{ij} that are used to weight the importance of each neigbour keypoint in updating the state of the current keypoint. Now, finally we have that $x_i' = \sigma(\sum_{j\in\mathcal{N}(x_i)} \alpha_{ij} W^\theta * x_j)$. After four iterations of graph attention convolution along with `BatchNorm`, we pool the graph into a single representation using `GlobalAttention`. `GlobalAttention` weighs node features using learned importance scores. This allows us to focus on the most salient nodes in a graph. Thus if we have f frames, we will have f graph representations. And these f graph representations will be treated as the context that is input to a standard Transformer backbone.

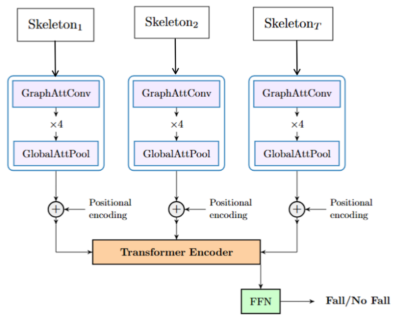

Figure 3: Architecture of ShahainNet

DevanNet

While most existing models are driven purely by data-driven approaches, we propose a novelty in the form of fusing human-based reasoning and domain knowledge into our models. Such approaches, falling under the umbrella of Informed Machine Learning or Knowledge-Informed Neural Networks (KINNs), seek to combine the pattern-recognition strengths of neural networks with explicit prior knowledge derived from the problem domain (Karpatne et al. 2017). To complement the data-driven features learned by the ST-GCN, we explicitly calculate two distinct types of continuous biomechanical features based on domain knowledge about fall events. These features aim to capture specific kinematic signatures characteristic of falls.

(a) Head-to-Ground Relative Proximity

A primary indicator of a fall is the resulting posture where the person is lying on the ground. Consequently, during and after a fall, the head is typically positioned significantly closer to the floor plane than during normal activities like standing or walking. To measure this robustly, we use the vertical positions of the feet as a dynamic proxy for the floor level immediately beneath the person. This relative measurement provides invariance to the person's absolute position in the camera frame and, critically, is largely agnostic to the camera's viewing angle or height, as it reflects the internal configuration of the body relative to its base of support.

We calculate this feature, $f^{(1)}$, for each batch element b and time step t. Let $p^{(y)}_{b,t,\mathcal{I}_{\text{joint}}}$ denote the y-coordinate (vertical position) of the joint specified by index $\mathcal{I}_{\text{joint}}$. The feature is computed as the difference between the head's y-coordinate and the minimum (lowest) y-coordinate of the two feet:

$$f^{(1)}_{b,t} = p^{(y)}_{b,t,\mathcal{I}_{\text{head}}} - \min\left(p^{(y)}_{b,t,\mathcal{I}_{\text{lfoot}}}, p^{(y)}_{b,t,\mathcal{I}_{\text{rfoot}}}\right)$$

Using the minimum foot coordinate provides robustness if one foot is lifted off the ground. A smaller (or negative, depending on coordinate system origin) value of $f^{(1)}$ indicates that the head is closer to the estimated floor plane defined by the feet.

(b) Torso Vertical Velocity

Falls, particularly uncontrolled ones, often involve a rapid downward acceleration and velocity of the body's core, distinct from controlled movements like sitting or intentionally lying down. We specifically focus on the vertical component of the torso's motion and monitor it's vertical velocity to provide a dynamic signal that helps differentiate a fall from simply being in a static prone position.

We approximate the instantaneous vertical velocity of the torso, $f^{(2)}$, using the backward difference of the y-coordinate of a central torso joint (e.g., the spine joint, $\mathcal{I}_{\text{spine}}$) between consecutive frames:

$$f^{(2)}_{b,t} = p^{(y)}_{b,t,\mathcal{I}_{\text{spine}}} - p^{(y)}_{b,t-1,\mathcal{I}_{\text{spine}}} \quad \text{for } t > 1$$

A large negative value of $f^{(2)}$ indicates a significant downward velocity of the torso during that time interval, characteristic of the descending phase of a fall.

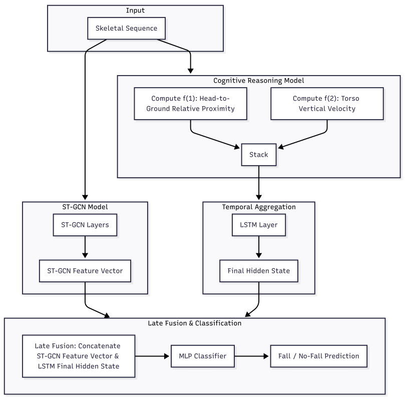
Input
Skeletal Sequence
Cognitive Reasoning Model
Compute f(1): Head-to-Ground Relative Proximity
Compute f(2): Torso Vertical Velocity
Stack
ST-GCN Model
ST-GCN Layers
ST-GCN Feature Vector
Temporal Aggregation
LSTM Layer
Final Hidden State
Late Fusion & Classification
Late Fusion: Concatenate ST-GCN Feature Vector & LSTM Final Hidden State
MLP Classifier
Fall / No-Fall Prediction

Figure 4: Architecture of DevanNet

Temporal Aggregation via LSTM for Fusion

The cognitive features ($\mathbf{F} \in \mathbb{R}^{B\times T\times D_{\text{feat}}}$) offer frame-by-frame biomechanical snapshots, implemented within the Cognitive Reasoning Module. In contrast, the ST-GCN outputs a single feature vector summarizing the input sequence. To fuse these two information streams, we employ a Long Short-Term Memory (LSTM) network to aggregate the cognitive features into a fixed-size representation that preserves temporal dynamics. Simple temporal averaging is avoided, as it disregards sequential order and may dilute short-duration fall indicators.

The LSTM processes the sequence of cognitive features ($\mathbf{f}_{b,1}, \mathbf{f}_{b,2}, \ldots, \mathbf{f}_{b,T}$ where $\mathbf{f}_{b,t} \in \mathbb{R}^{D_{\text{feat}}}$) recurrently, updating its hidden ($\mathbf{h}_t$) and cell ($\mathbf{c}_t$) states at each time step. We utilize the final hidden state ($\mathbf{h}_T$) as a compact representation of the sequence.

This aggregated cognitive vector is concatenated with the ST-GCN feature vector and passed to the fusion MLP for classification, enabling the model to leverage both complex learned patterns and structured, rule-based temporal cues for fall detection.

XuyanNet

Falls typically begin with localized instability (e.g., a knee buckling) and escalate into full-body collapse involving coordinated movement across multiple joints. Capturing this progression requires both fine-grained spatial detail and global structural awareness. Our model addresses this by combining:

- A **spatial adjacency matrix** based on hop distance and anatomical structure, and
- A **spectral adjacency matrix** derived from the normalized Laplacian, which encodes the global topology of the pose graph.

While ST-GCNs model local joint relationships well, they fall short in capturing long-range dependencies critical for actions like falls. For instance, during a fall, distant joints such as the head and feet often exhibit synchronized movement (Huang et al. 2024). To address this, we propose a **spectral–spatial hybrid GCN** that integrates both local and global perspectives. The spatial stream preserves short-range anatomical structure, while the spectral stream leverages the graph Laplacian to model global dependencies. A learnable fusion weight $\boldsymbol{\alpha} \in [0,1]$ dynamically balances both pathways during training.

(a) Spectral Graph Convolution

Spectral graph convolution operates in the frequency domain using the graph Laplacian. The normalized graph Laplacian $\mathbf{L}$ is defined as: $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is the degree matrix, $\mathbf{A}$ is the adjacency matrix. Take the absolute value of $\mathbf{L}$ and row-normalize it: $\mathbf{A}_{\text{spectral}} = \text{RowNorm}(|\mathbf{L}|)$ Then the spectral convolution becomes: $\mathbf{H}^{(l+1)}_{\text{spectral}} = \sigma\left(\mathbf{A}_{\text{spectral}} \cdot \mathbf{H}^{(l)} \cdot \mathbf{W}^{(l)}_{\text{spectral}}\right)$, where $\mathbf{W}^{(l)}_{\text{spectral}}$ is another trainable weight matrix.

(b) Hybrid Spectral-Spatial Convolution with Adaptive Fusion

We design a hybrid graph convolutional block with two parallel streams:

- A **spatial stream**, using standard graph convolution with multi-hop adjacency matrices $\{\mathbf{A}^{(k)}\}$,

- A **spectral stream**, using Chebyshev polynomial approximation $T_k(\mathbf{L})$ of the normalized Laplacian $\mathbf{L}$.

Each stream outputs:

$$\mathbf{H}^{(l+1)}_{\text{spatial}} = \sigma\left(\sum_{k=0}^{K} \mathbf{A}^{(k)} \cdot \mathbf{H}^{(l)} \cdot \mathbf{W}_k^{(l)}\right)$$

$$\mathbf{H}^{(l+1)}_{\text{spectral}} = \sum_{k=0}^{K-1} T_k(\mathbf{L}) \cdot \mathbf{H}^{(l)} \cdot \mathbf{\Theta}_k$$

The two outputs are fused via:

$$\mathbf{H}^{(l+1)} = \mathrm{TCN}\left(\alpha \cdot \mathbf{H}^{(l+1)}_{\text{spatial}} + (1-\alpha) \cdot \mathbf{H}^{(l+1)}_{\text{spectral}}\right) + \mathrm{Residual}(\mathbf{H}^{(l)}),$$

where $\mathrm{TCN}(\cdot)$ denotes a temporal convolutional layer, and $\mathrm{Residual}(\cdot)$ is a skip connection applied for stability.

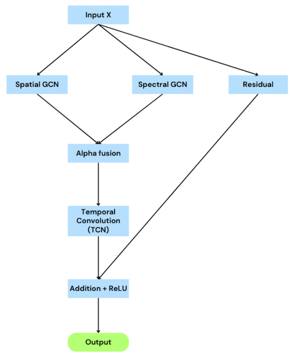

Figure 5: Illustration of hybrid-st-gcn structure

This hybrid formulation enables our solution to adaptively learn how much to rely on global (spectral) versus local (spatial) structure at each layer. Combined with temporal modeling, this allows the network to effectively capture both the onset and escalation of falls—ranging from local perturbations to whole-body collapse.

JohnNet

Distinguishing true falls from similar actions, such as stumbling or sitting quickly, requires capturing both fine-grained joint movements and overall body dynamics over time. Leveraging a hierarchical structure (Hu et al. 2019) allows the model to learn multi-level skeletal representations and long-range temporal dependencies, going beyond the local spatio-temporal patterns captured. In this structure, each joint node is connected not only to its adjacent body parts in the same frame but also to its exact joint in the next time step.

Stacking GCN Layers

With more GCN layer, the receptive field expands, allowing the model to reach node that are further away temporally or spatially. Intuition would be to capture transitions across multiple frames specifically the acceleration of the nodes collapse when falling.

Pooling Layers

This layering enables the model to progressively extract and, more importantly, condense both the temporal and spatial features of the skeletal dataset. The intuition is to "coarsen" the unified graph by removing potential noise from redundant joints. The pooling operation uses a learnable or predefined pooling matrix $P^{(l)} \in \mathbb{R}^{N_{l+1} \times N_l}$, which contains binary values (0s and 1s) to indicate which nodes to retain. At each layer l, the pooled node features are computed as $H^{(l+1)} = P^{(l)} H^{(l)}$, and the new adjacency matrix is updated as $A^{(l+1)} = P^{(l)} A^{(l)} (P^{(l)})^\top$. Here, $H^{(l)}$ represents the node features and $A^{(l)}$ the adjacency matrix at layer l.

Multi Feature Combination

To ensure the model captures both general body motion and critical fall indicators, we concatenate the global mean and max pooling layers. The global mean pooling captures overall motion patterns and trends, allowing the model to understand the broader temporal context. In contrast, the global max pooling highlights critical and extreme movements, such as sudden shifts in the x and y directions, which are crucial for detecting significant fall events. This combined representation is expressed as $H_{\text{multi}} = \mathrm{Concat}(\mathrm{MaxPool}, \mathrm{MeanPool})$, enabling the model to aggregate fine-to-coarse features and better recognize the multi-phase nature of falls.

Residual Connections and Batch Normalisation

Adding more GCN layers can lead to oversmoothing, where node representations become too similar and lose discriminative power. To mitigate this, we introduce residual connections after pooling layers to preserve key features and maintain the uniqueness of each node. This is achieved by combining the transformed features with the original ones using the formulation $H^{(l+1)} = \sigma\left(\hat{A} H^{(l)} W^{(l)}\right) + H^{(l)}$, where $\hat{A}$ is the normalized adjacency matrix, $H^{(l)}$ represents node features at layer l, and $W^{(l)}$ is the learnable weight matrix.

Another method in dealing with over-smoothing due to multiple GCN layers would be normalizing the node features.

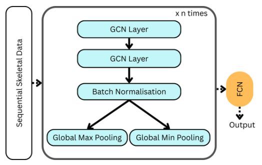

Figure 6: JohnNet Architecture

Experiment

Datasets used are the publicly available Le2i (Charfi et al. n.d.) and UR Fall (Kwolek and Kepski 2014) datasets, both focused on fall detection in video. Each dataset was split into training and validation sets, with videos preprocessed

Model	Le2i Dataset				UR Fall Dataset			
	Train Acc	**Val Loss**	**Val Acc**	**Val F1 Score**	**Train Acc**	**Val Loss**	**Val Acc**	**Val F1 Score**
BASELINE-ST-GCN	100%	0.0346	97.37%	0.9740	90%	0.6127	58.33%	0.44298
ShahainNet	100%	0.590	89.47%	0.933	**100%**	0.1226	91.67%	**0.9231**
DevanNet	100%	**0.0009**	**100%**	**1.0**	85%	0.3656	91.67%	0.9148
JohnNet	100%	0.0190	**100%**	**1.0**	**100%**	0.2070	91.67%	0.9172
XuyanNet	100%	0.0867	**100%**	0.97	83.33%	**0.0906**	**100%**	0.83

Table 1: Performance Metrics for Different Models on Le2I and UR Fall Datasets

into individual frames and annotated with fall intervals. Le2i provides 152 training and 38 validation videos, while UR Fall includes 60 for training and 12 for validation.

Results & Discussion

JohnNet demonstrated strong performance across both datasets. Although residual connections were introduced to mitigate over-smoothing, they failed to prune irrelevant temporal edges effectively, due to excessive noise caused by the unification of both temporal and spatial edges. Removing this and lowering the GCN layers improved the model significantly. Overall, the HGCN shows promising results in fall detection and can be considered on par with the ST-GCN.

ShahainNet demonstrated clear improvements over the baseline ST-GCN by incorporating kinematic dynamics into play. Moving beyond a simple skeletal topology it employed graph structure learning to adaptively identify and focus on the joint relationships that provide the most evidence for fall prediction. This rich spatio-temporal representation was processed over time by a transformer backbone allowing it to model complex dependencies over time.

DevanNet showcased improvements with the added human-cognition model over the baseline ST-GCN. The hybrid model outperforms the baseline ST-GCN by combining complex learned patterns with explicit domain knowledge. While ST-GCN captures intricate spatio-temporal cues, it may miss rare or geometrically precise events. The Cognitive Reasoning Module compensates by encoding high-impact biomechanical indicators, enhancing robustness and generalization through a fusion of data-driven and rule-based features.

XuyanNet showed strong performance across both fall and non-fall cases. Combining both the spatial path and the spectral path attributed to this models' success. The learnable fusion weight provided adaptability in different types of motion, from small stumbles to full-body falls. This combination of local and global feature extraction, along with adaptive fusion, enhances the model's robustness and accuracy, making it well-suited for real-world fall detection scenarios.

Conclusion

In summary, we contribute four GCN models to solve the task of fall detection. Our results showcase that XuyanNet has the best validation loss. Such models offers a powerful complementary offering under HDB's Assistive-Living Technologies umbrella. By leveraging existing low-resolution CCTV feeds rather than wearables or radio-wave sensors, it overcomes the blind spots of slow-fall detection and non-human motion false alarms inherent in current motion-sensing and scream-detection products. Furthermore the lightweight, real-time GCN message passing meets real-time streaming constraints and allows seniors' emergency contact to be alerted immediately.

Team Members & Roles

Shahain Manujith developed `ShahainNet` and performed data preparation on the `UR-Fall dataset`. **John Ashwin** developed `JohnNet` and provided annotations for `LE2I dataset` **Devanshu Bisht** developed `DevanNet`, ST-GCN, and did feature engineering for `LE2I dataset` **Zhou Xuyan** developed `XuyanNet`, hybrid-spectral-ST-GCN, and provided annotations for `LE2I dataset`.

References

Charfi, I.; Mitran, J.; Dubois, J.; Atri, M.; Tourki, R.; and Tuyen, L. D. V. N. n.d. Optimised spatio-temporal descriptors for real-time fall detection. https://www.kaggle.com/datasets/tuyenldvn/falldataset-imvia. Accessed: 2025-04-19.

Hu, F.; Zhu, Y.; Wu, S.; Wang, L.; and Tan, T. 2019. Hierarchical Graph Convolutional Networks for Semi-supervised Node Classification. ArXiv:1902.06667, arXiv:1902.06667.

Huang, T.-Y.; Lin, H.-Y.; Lin, S.-H.; Lee, Y.-C.; Lin, Y.-H.; and Huang, Y.-H. 2024. GaitFall: Vision-Based Fall Detection and Classification for the Elderly Using Gait Analysis. *Electronics*, 13(18): 3737.

Karpatne, A.; Atluri, G.; Faghmous, J. H.; Steinbach, M.; Banerjee, A.; Ganguly, A.; Shekhar, S.; Samatova, N.; and Kumar, V. 2017. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10): 2318–2331.

Kwolek, B.; and Kepski, M. 2014. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine*, 117: 489–501.

PhysioPedia. n.d. Long Lie. Accessed: 2025-04-19.

World Health Organization. 2021. Falls. Accessed: 2025-04-19.