Video Based Fall Detection using Human Poses (2021)

Ziwei Chen, Yiye Wang, Wankou Yang

발표자: 손소영

Summary

- A fall detection model which includes 3D pose estimator and a fall detection network based on human poses
- Explore the effects of factors which could contribute to the performances of fall detection including input joints, loss functions
- Achieve a high accuracy at 99.83% on NTU RGB+D dataset and real-time performance on non-GPU platform

Introduction

- Ageing of population has become a global phenomena
- Adults older than 65 years suffer the greatest number of fatal falls, which could cause serious injuries and even death

  -> increasing attention to fall detection

Two methods of fall detection

1. Wearable Sensor based

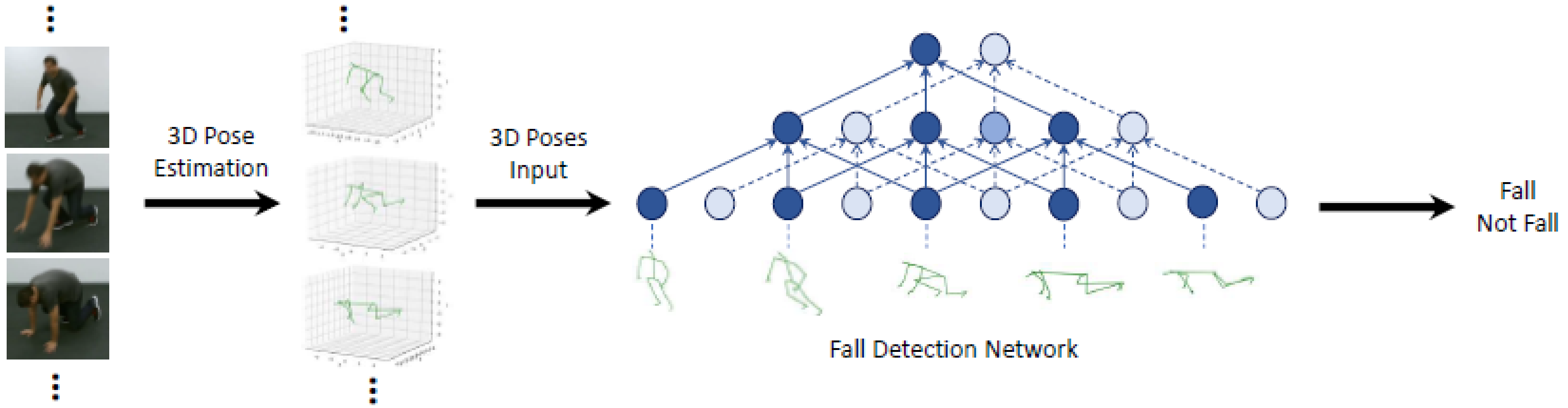
- Can detect the location change or acceleration change of human body for fall detection
- May be affected by noise and some daily activities may lead to false alarm
- Inconvenience of wearing devices

2. Vision based

- High accuracy but large computation cost

  -> hard to achieve real-time performance
- False detection may occur as a result of lighting variation, complex background, etc.

Video based Fall Detection using HPE

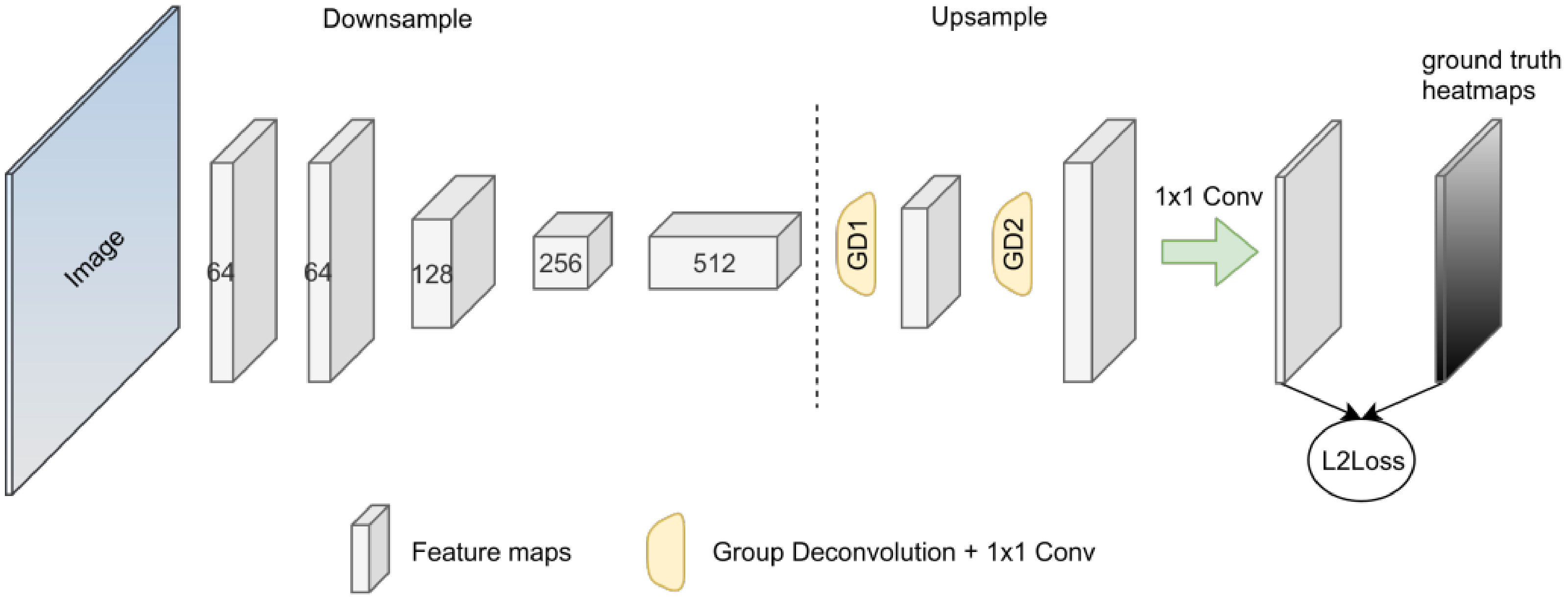


1. Pose Estimator: get 3D poses
2. Fall Detection Network: classify whether fall or not

Human Pose Estimator (HPE)

- Do not rely on any specific pose estimator
- Follow the widely-used pipeline in 3D HPE
  -> predicts 2D human poses and lifts 2D poses to 3D poses
- Adapt off-the-shelf Lightweight Pose Network(LPN) for 2D HPE
- Then lift 2D human poses to 3D poses using Lifting Network

Lightweight Pose Network (LPN)



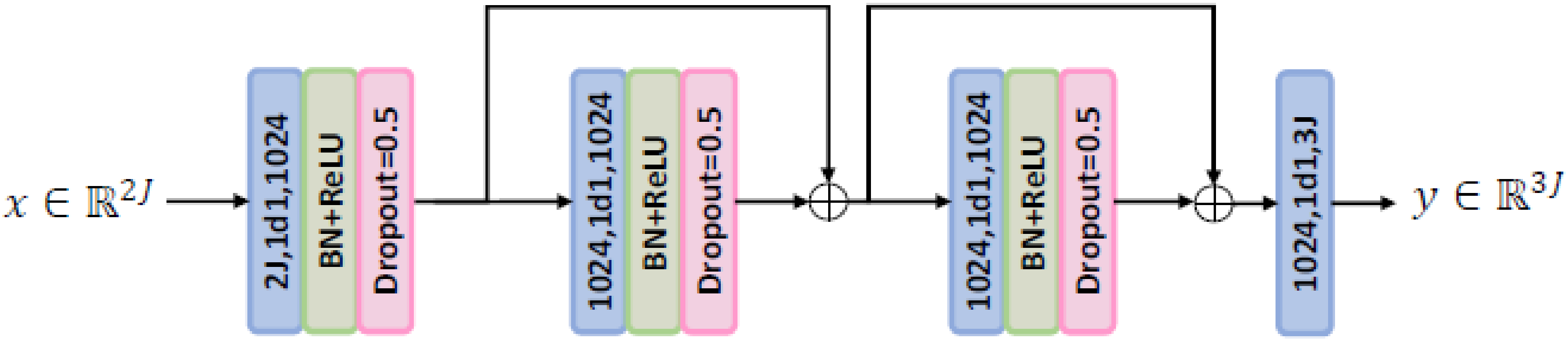
- Pretrained on MS COCO dataset and fine-tuned on NTU RGB+D dataset
- Low complexity and adequate accuracy

Lifting Network

$$f^* = \min_f \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x_i) - y_i)$$

- Takes 2D human poses as input and lifts them to 3D poses
- The goal is to find a function $f^* = \mathbb{R}^{2J} \rightarrow \mathbb{R}^{3J}$ that minimizes the prediction error of N poses over a dataset

Lifting Network



- Consists of two residual blocks
- 2D coordinates of joints are concatenated

  -> 1d conv can be adopt to reduce parameters and complexity
- The core idea of lifting network is to predict depth information of key points effectively and efficiently

Lifting Network

$$\mathcal{L}(\hat{y_i}, y_i) = ||\hat{y_i} - y_i||_2^2$$

- Care relative location between key points, not the absolute location

-> before a 2D pose is lifted to 3D, normalized by centering to its root joint and scaled by dividing it by its Frobenius norm

- Prediction error as the squared difference between the prediction and the ground-truth pose

Frobenius Norm

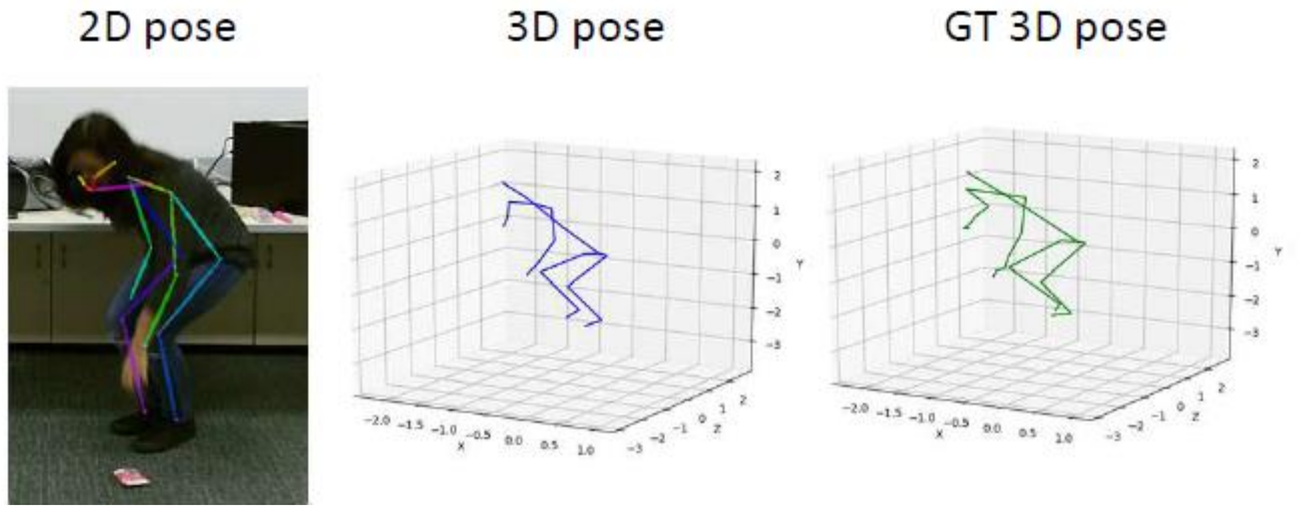
$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

- L2 norm of matrix
- Defined as the square root of the sum of the absolute squares of its elements

Fall Detection Network

- Fully convolutional architecture with residual connections that takes 3D poses $X \in \mathbb{R}^{T \times 3J}$ as input (T: # frame)
- Predict whether there is a fall behavior
- In convolutional networks, the path of gradient between output and input has a fixed length
- T was set to 300 to recognize falls in such a long video sequence
- Dilated convolutions are applied to model long-term dependencies while maintaining efficiency

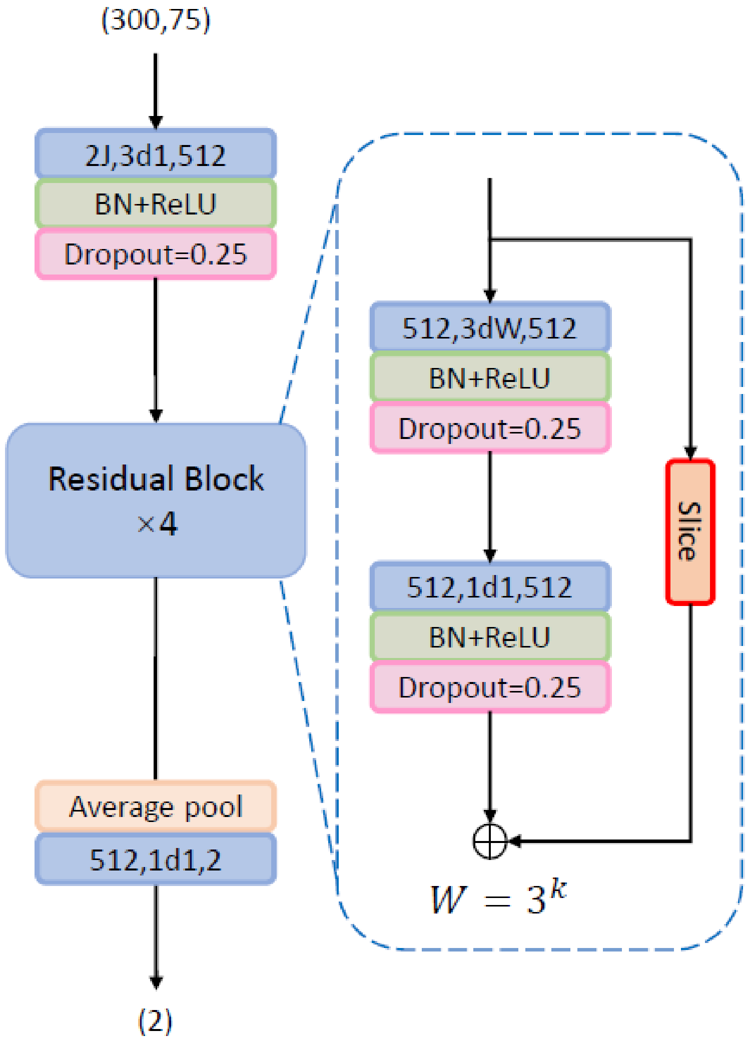
Fall Detection Network



- To obtain normalized 3D poses, 3D poses are rotated by paralleling the bone between

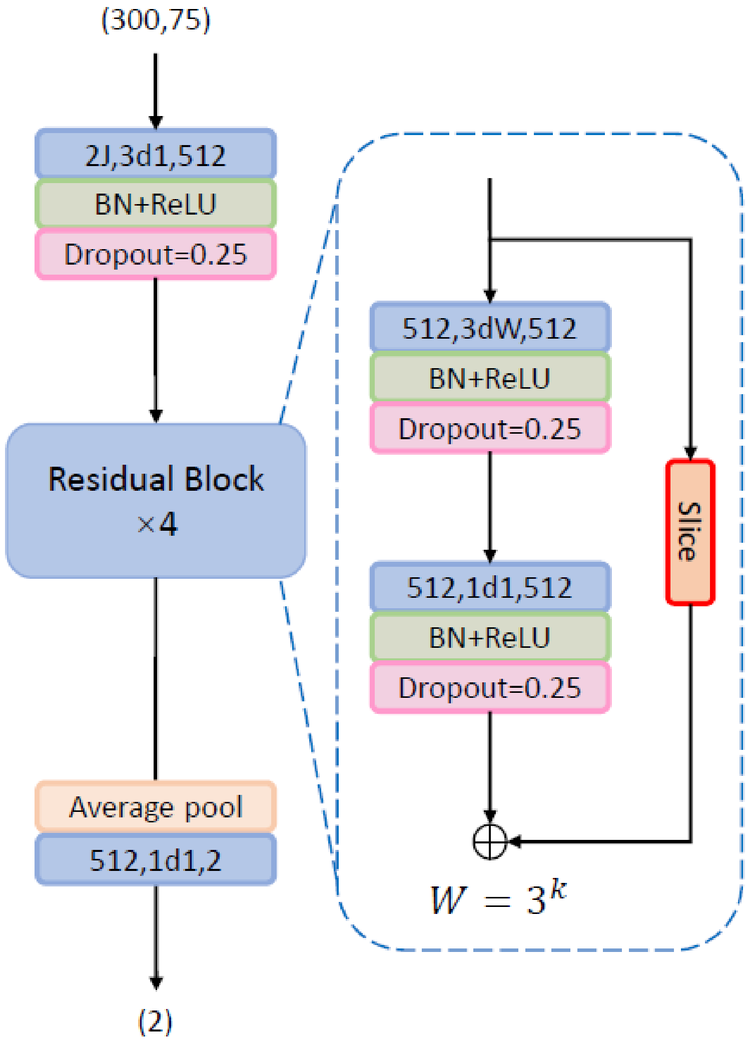
1. hip and spine to the z axis
2. left shoulder and right shoulder to the x axis

Fall Detection Network



- Input: concatenation of 3D coordinates (x, y, z) of J joints for each frame
- Convolution with kernel size 3
- C output channels
- N residual blocks
- Each block includes two 1d conv:
    1. Dilated and dilation factor $W = 3^N$
    2. Kernel size 1
- Batch normalization, reLU, dropout are used after every convolution except last one

Fall Detection Network



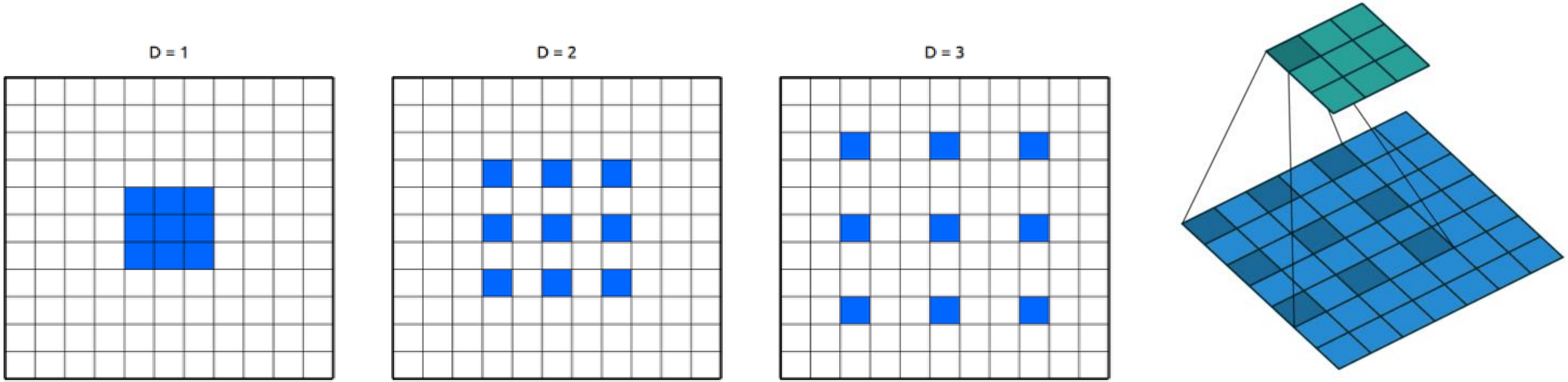
- Use unpadded convolutions

  -> the output size of each block is different
- Average pooling to fuse features and change the dimension for the final convolution

Parameters

- Length of video sequence T=300
- N=4
- Output channels C=512
- Dropout rate p=0.25

Dilated Convolution



- Inflate the kernel by inserting holes between the kernel elements
- Increase receptive field with low computational cost
- Only blue part has weight, and the others are filled with 0

Why Slice?

- Output of convolution layer:
  - Input: (N, C_{in}, L_{in}) or (C_{in}, L_{in})
  - Output: (N, C_{out}, L_{out}) or (C_{out}, L_{out}), where

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor$$

- Output shape is changed when the tensor passes dilation convolution layer

\- Slice is used to make the shape identical

Dataset

NTU RGB+D Action Recognition Dataset

- Contains 60 action classes and 56,880 video samples including falling
- Videos with three different horizontal angles: -45°, 0°, +45°
- Contains RGB videos, depth map sequences, 3D skeletal data, infrared videos for each sample
- The resolution of RGB frames: 1920 x 1080, speed: 30 FPS
- 3D skeletal data are composed of 3D coordinates of 25 joints
- Used only falling samples

Training Details - HPE

- off-the-shelf LPN to predict 2D poses from each video frame
  - LPN is pretrained on COCO dataset and fine-tuned on NTU RGB+D dataset
  - When fine-tuning LPN, joint heatmaps were generated according to annotations as the output target which could avoid directly learning the mapping from images to coordinates
  - 2D joint coordinates could be obtained by calculating the center of mass of heatmaps
- 2D poses were normalized and scaled before lifting to 3D
- Adam optimizer and MSE loss
- 60 epochs
- Initial learning rate of 1e-4 and exponential decay at 20^{th} and 40^{th} epoch

Training Details – Fall Detection Network

- 3D poses were normalized before being sent to the network
- All samples were expanded to 300 frames by padding null frames with previous ones
- Adam optimizer and Cross Entropy Loss
- Initial learning rate to 1e-4 with exponential decay
- Nvidia GTX 1660 GPU for 20 epochs

Results - HPE

B spi	Head	L elb	L wri	R elb	R wri	R ank
99.69	98.02	98.45	97.91	94.22	90.82	71.06

- Metric: JDR (Joint Detection Rate) – the percentage of successfully detected joints
- A joint is regarded as successfully detected if the distance between the estimation and ground-truth is smaller than a threshold
- Set the threshold to be half of the distance between neck and head

Results - HPE

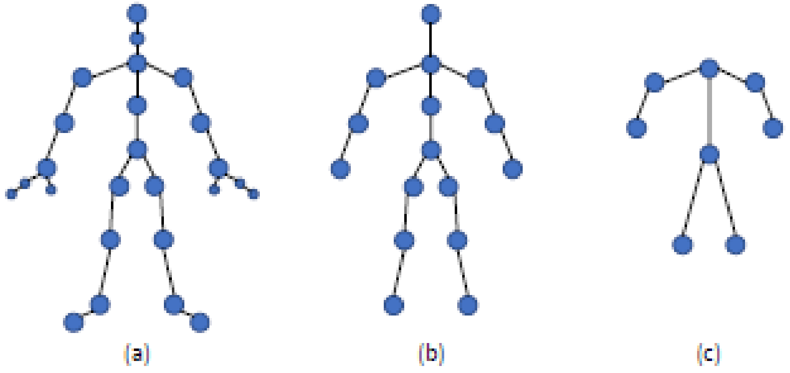


Figure 5. Skeleton information of different inputs. (a) All 25 joints. (b) Selected 16 joints. (c) Selected 8 joints.

Joints	mJDR
25 joints	86.02
16 joints	94.07
8 joints	94.52

- Mean JDR of three aggregations of joints (25, 16, 8)

Results – Fall Detection

Methods	Input	Feature	Network	Accuracy
Xu et al. [45]	RGB	Pose	2D conv	91.70%
Anahita et al. [35]	Depth	Pose	LSTM	96.12%
Han et al. [39]	Depth	Pose	1D conv	99.20%
Ours	RGB	Pose	1D conv	**99.83%**

Method	Accuracy	Precision	Recall
8 joints-CEL	99.72%	97.15%	89.74%
8 joints-WCEL	99.29%	98.70%	74.32%
16 joints-CEL	**99.83%**	97.47%	**94.25%**
16 joints-WCEL	99.50%	**98.73%**	80.67%
25 joints-CEL	99.77%	97.79%	91.35%
25 joints-WCEL	99.66%	97.47%	87.11%

- WCEL: $\alpha = \frac{59}{60}, \beta = \frac{1}{60}$

Results – Fall Detection

Part	Params	FLOPs	non-GPU	GPU
LPN	2.7M	1.0G	20 FPS	74 FPS
LN	2.2M	0.28G	560 FPS	1450 FPS
FDN	4.2M	0.9G	260 FPS	590 FPS
Whole	9.1M	2.18G	18 FPS	63 FPS

- The inference speed of lifting network and fall detection network is very fast that only takes a few milliseconds
- LPN is the one mainly limits the inference speed, but it can be changed to another efficient pose estimator

Thank you