OpenMMLab

Action Understanding in a Human-Centric View

Haodong Duan

Action recognition based on human skeletons is computationally efficient and robust to background variations or lighting changes. This talk will introduce our recent work in skeleton-based action recognition, including, 1) PoseConv3D: adapting 3D ConvNets to skeleton action recognition; 2) STGCN++: a frustratingly simple and strong GCN baseline for skeleton action recognition; 3) PYSKL: a comprehensive codebase for skeleton action recognition that supports multiple algorithms and datasets. I will also highlight the good practices for processing skeleton data, and share some thoughts on this topic and its future direction.

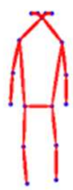
CVPR Tutorial 2022

Action Recognition

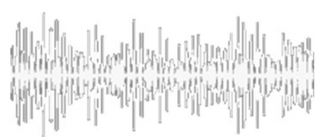
Action recognition aims at recognizing the human action in a video, usually based on various modalities: RGB (mostly used), optical flow, audio, human skeleton, etc.

RGB

Flow



Skeleton



Audio

……

Multiple modalities in a video

Skeleton-based Action Recognition

Definition: Action recognition solely based on skeleton sequence.
Extension: Eye Landmark -> Gaze; Facial Landmark -> Expression; Hand Landmark -> Gesture; …

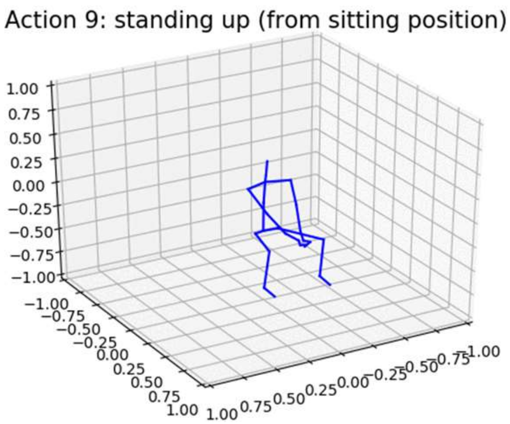
Why / When we need Skeleton-based Action Recognition?

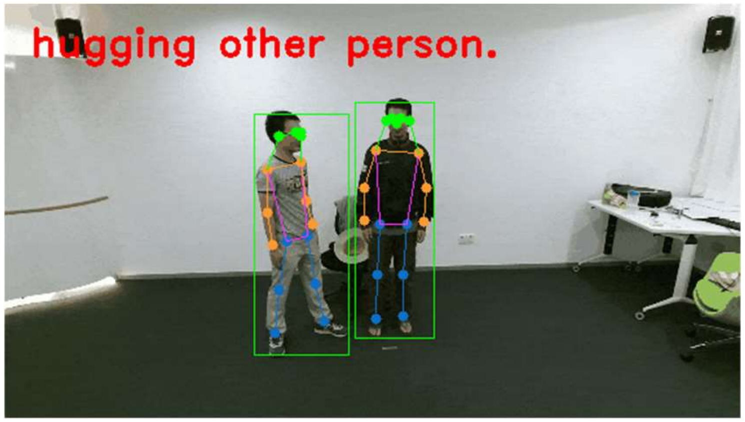
1. (Firstly) Only if it is possible to recognize the action only based on skeleton.
2. The training data (RGB) is scarce or highly biased.
3. When you need a **very light** action recognition model (skeleton models can be as light as < 1 MParams & < 1 GFLOPs).

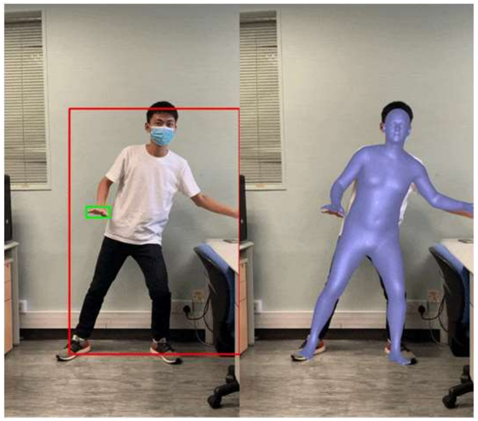
Computational Efficiency

Approach	RGB (3D-CNN)	Skeleton (3D-CNN)	Skeleton (GCN)
Backbone	SlowOnly-R50	SlowOnly-R50	ST-GCN
# Frames	8	48	100
Input Shape	3 x 8 x 224 x 224	17 x 48 x 56 x 56	2 x 100 x 17 x 3
Params	31.6M	2.0M	3.1M
FLOPs	42.2G	15.8G	3.8G

How to obtain human skeletons?





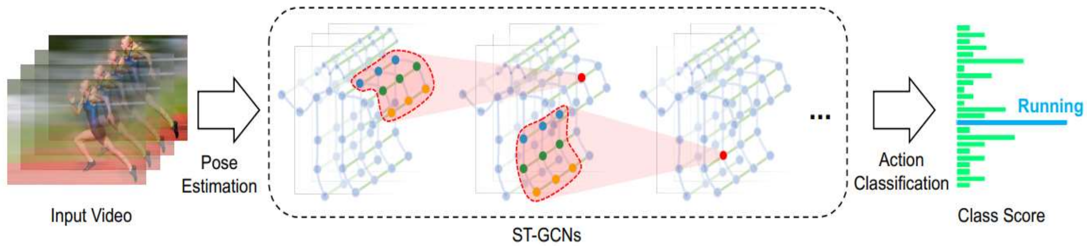


Kinect Sensor (RGBD)

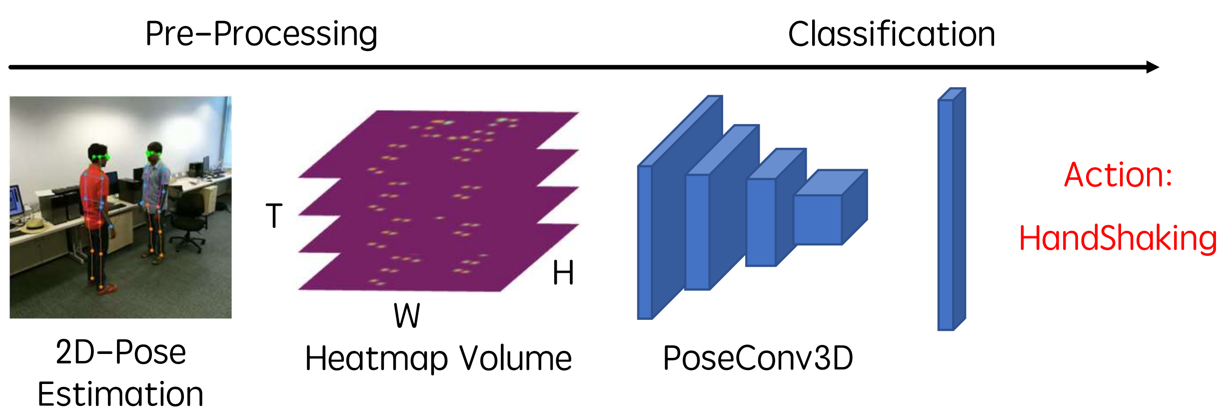
Pose Estimation (2D)

Mocap (3D)

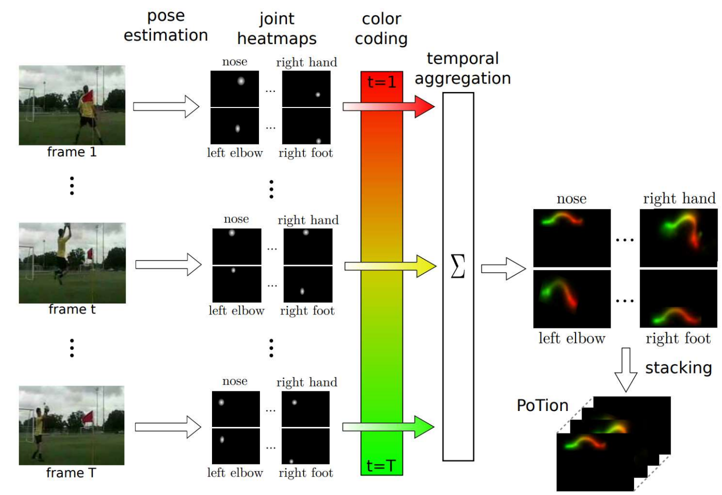
The Solutions



Arch: GCN; Input: Coordinates

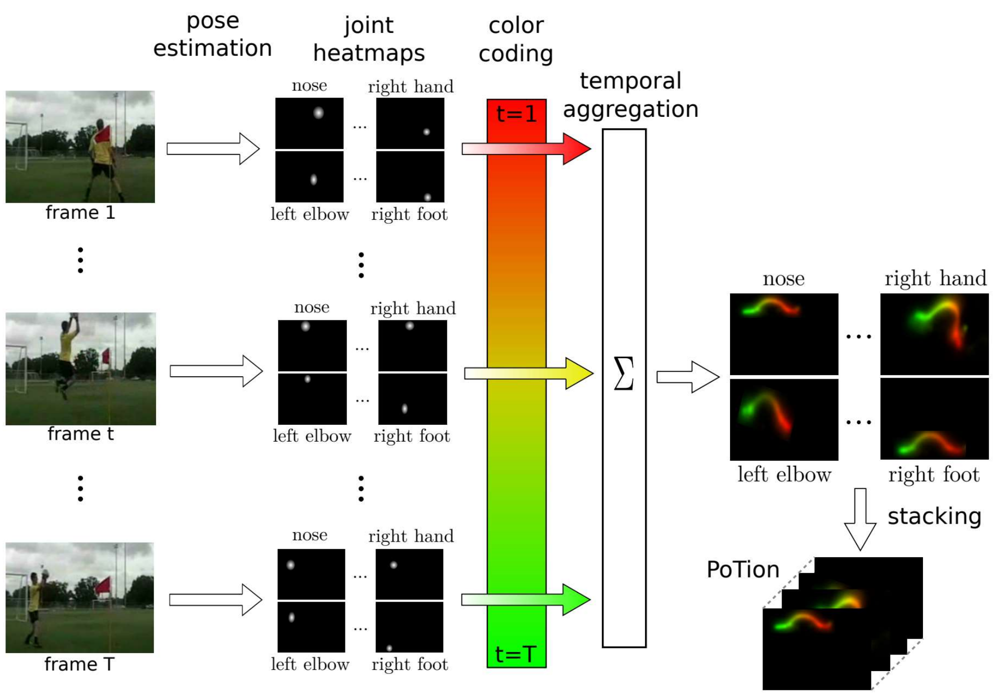


Arch: 3D-CNN; Input: Heatmap Volumes

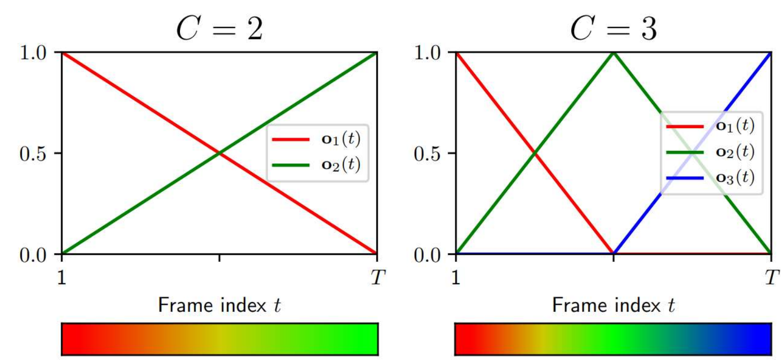


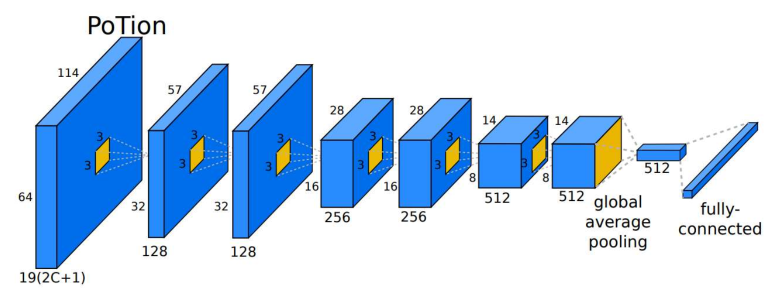
Architecture: 2D-CNN;
Input: Pseudo Image

2D-CNN approach (PoTion [1])



Information lost during color coding.



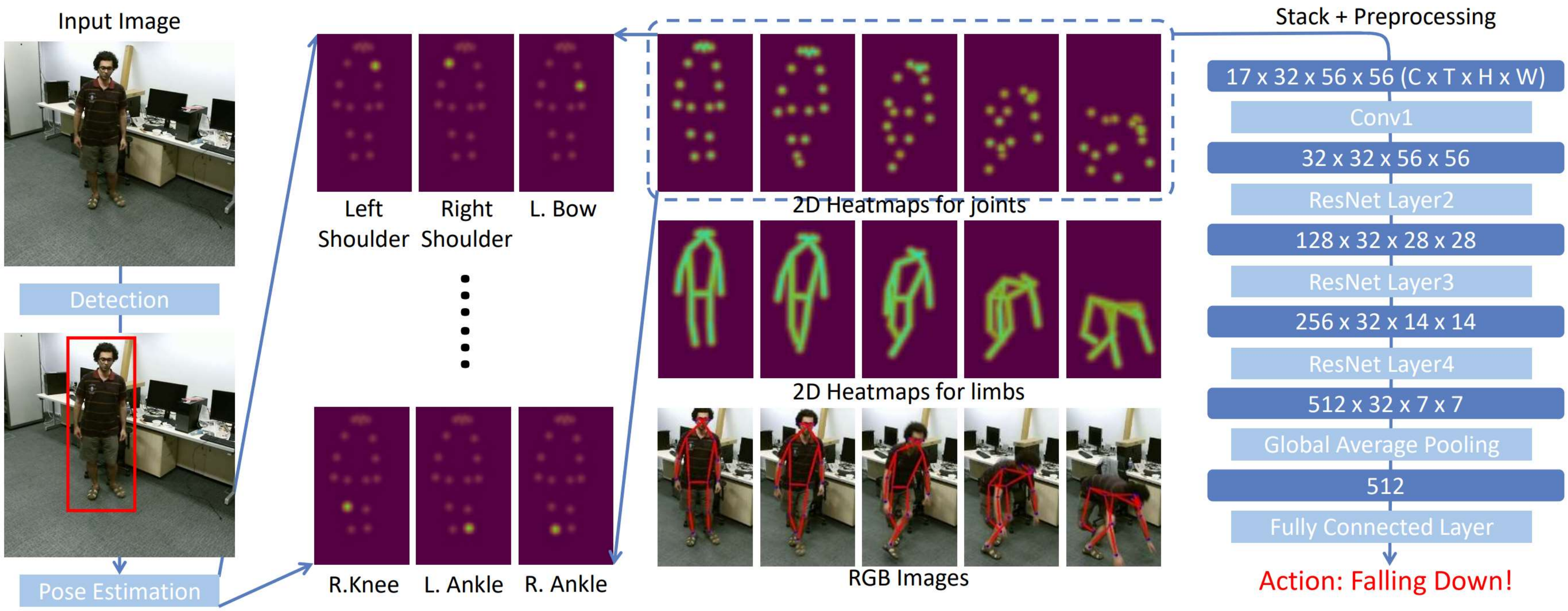


The adopted 2D-CNN architecture.

[1] Choutas et al., Potion: Pose motion representation for action recognition, CVPR 2018

PoseConv3D [1]

A 3D-CNN based solution.

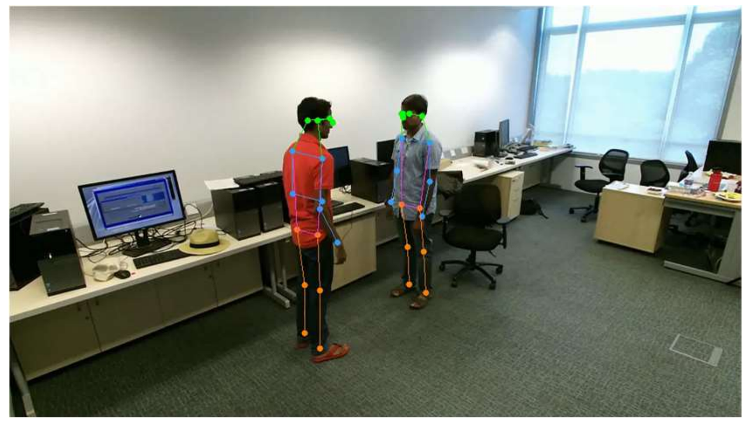


[1] Duan et al., Revisiting skeleton-based action recognition, CVPR 2018

PoseC3D Pipeline

1. Pose Extraction

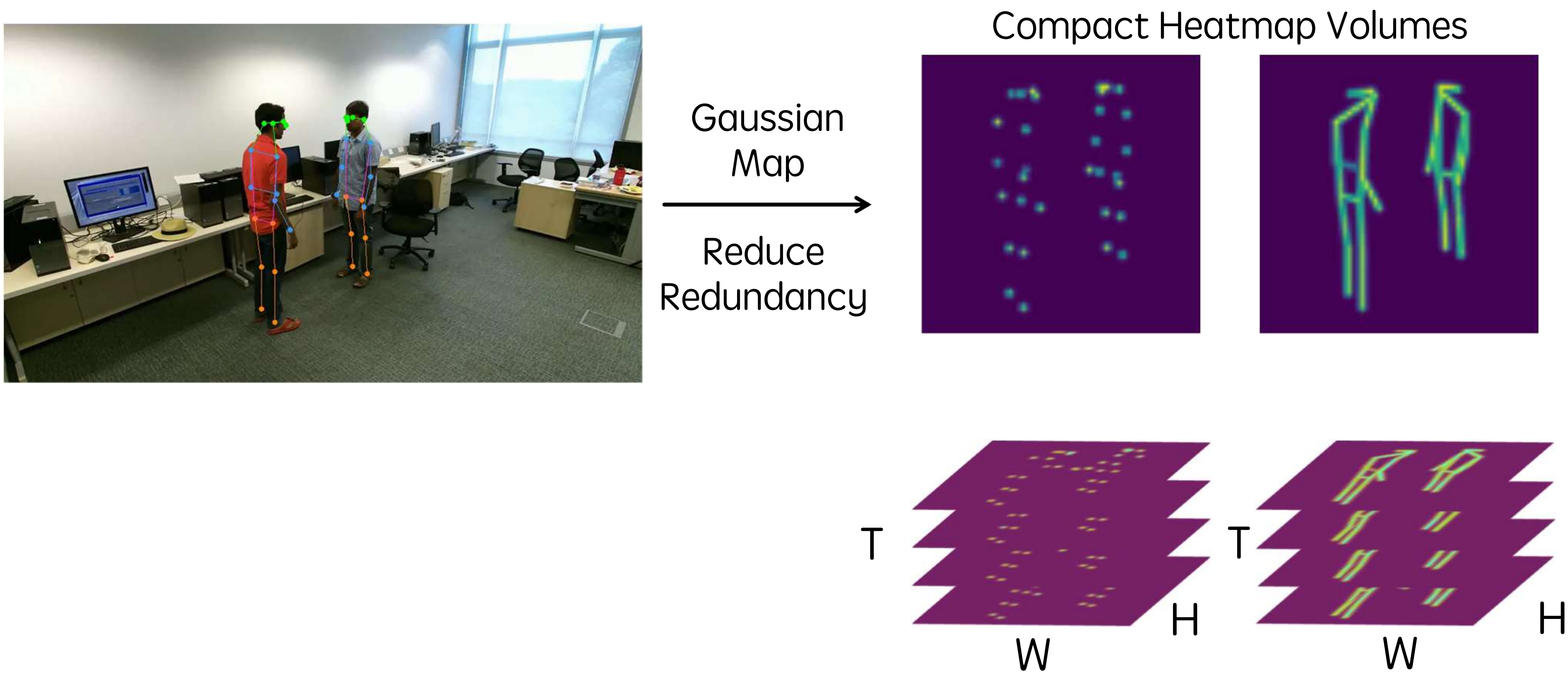
2D-Pose Estimation →



Person 1	Person 2
Left-shoulder (x_{11}, y_{11}, c_{11})	Left-shoulder (x_{21}, y_{21}, c_{21})
Right-shoulder (x_{12}, y_{12}, c_{12})	Right-shoulder (x_{22}, y_{22}, c_{22})
……	……
Right-ankle (x_{1k}, y_{1k}, c_{1k})	Right-ankle (x_{2k}, y_{2k}, c_{2k})

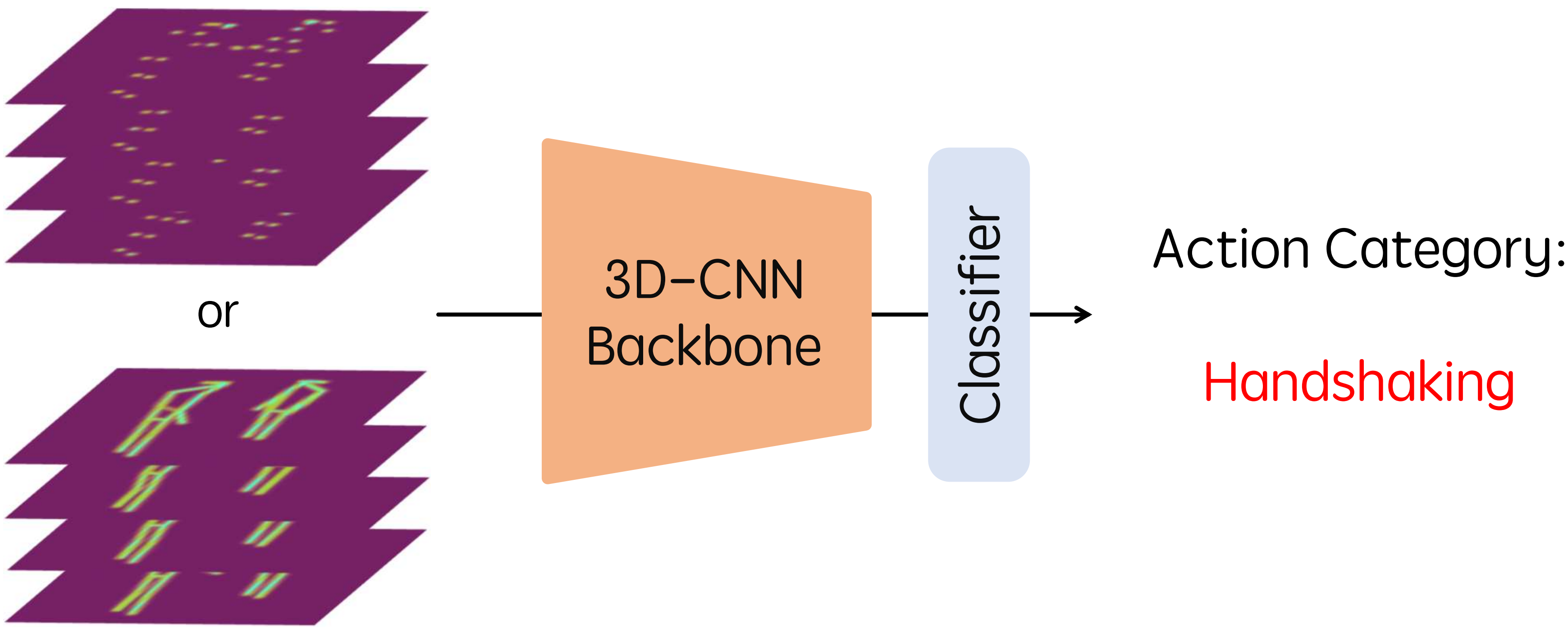
PoseC3D Pipeline

2. Generating Compact Heatmap Volume



PoseC3D Pipeline

3. Action Recognition with 3D-CNN



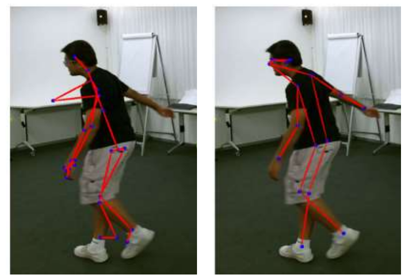
Pose Extraction

We adopt a two-stage pose estimator (HRNet [1]) for pose extraction.

Takeaways:

1. Estimated 2D skeletons are of **superior quality**, compared to 3D skeletons estimated or collected by sensors.

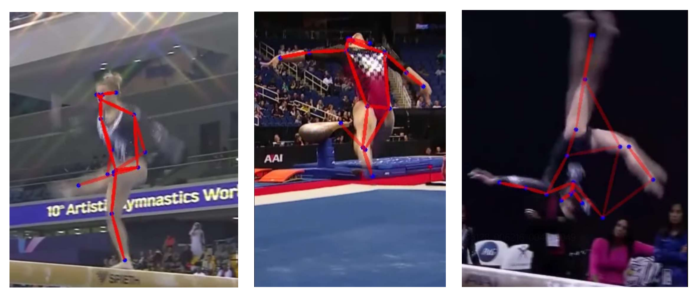
3D 2D



Pose Annotations	NTU-60
3D [Kinect Sensor]	87.0
2D [HRNet]	92.0
2D [MobileNet]	89.0

2D skeleton v.s. 3D skeleton (MS-G3D)

2. Skeleton action recognition does not need perfect pose estimation results, as long as action patterns can be revealed.



Inaccurate pose estimation

Mean Top-1: 94.1%

GYM Accuracy (99 classes)

[1] Sun et al., Deep high-resolution representation learning for human pose estimation, CVPR 2019

Pose Storing

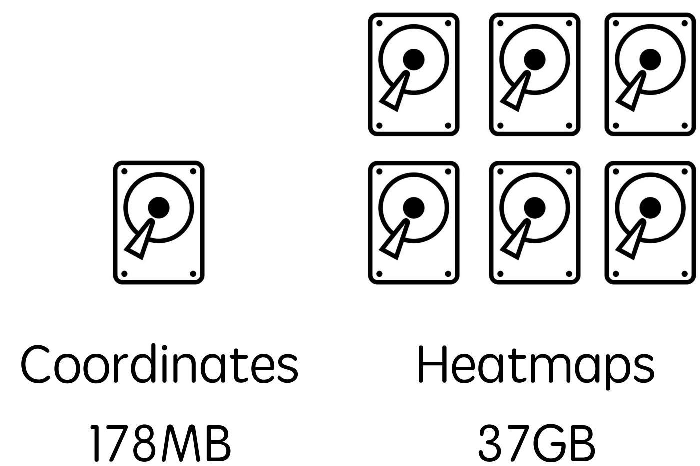
The extracted skeletons can be saved as heatmaps / coordinates.

Heatmaps take much more storage but the improvement is limited.

	Mean-Top1
Coordinate [LQ]	90.7
Coordinate [HQ]	93.2
Heatmap [LQ]	92.7
Heatmap [HQ]	93.6

Coordinates *vs.* Heatmaps.

The degradation in performance is moderate if a high-quality pose estimator is used.



Coordinate -> Pseudo Heatmap

1. Each joint -> A gaussian map with size H x W
2. A skeleton with K joints -> A pseudo heatmap with K channels (K x H x W)
3. Stacking heatmaps in temporal -> A 3D heatmap volume (K x T x H x W)

L-shoulder (x_1, y_1, c_1)

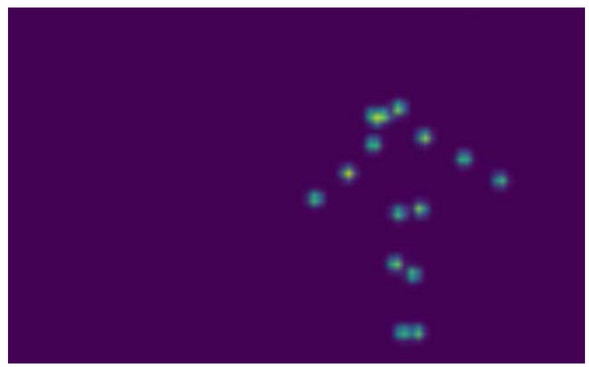
R-shoulder (x_2, y_2, c_2)

L-bow (x_3, y_3, c_3)

……

R-ankle (x_k, y_k, c_k)

Gen Gaussians



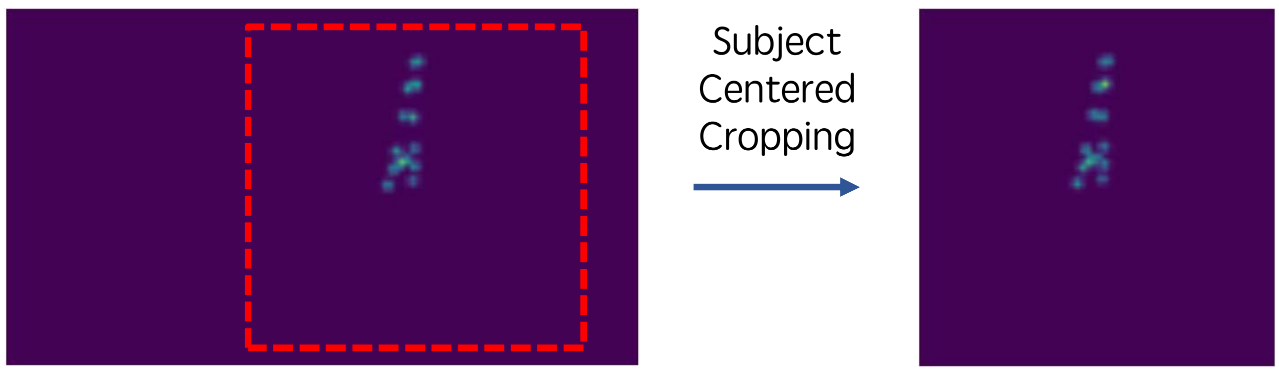
The heatmap has K channels.

We merge them into one single channel for visualization.

Generating a pseudo heatmap.

Generating Compact Heatmap Volume

Reduce Spatial Redundancy: Subject Centered Cropping

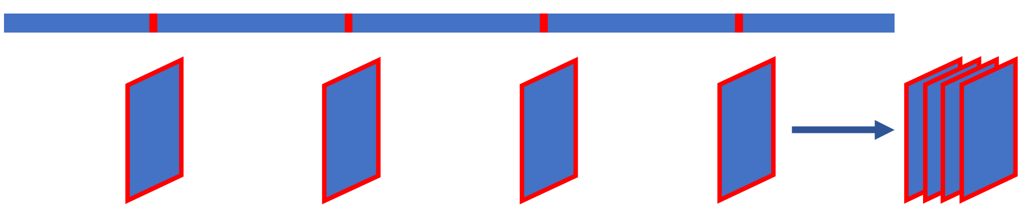


92.2% Top-1 (on NTU-60)

93.2% Top-1 (on NTU-60)

Reduce Temporal Redundancy : Uniform Sampling (smaller)

Uniform Sampling (real image)



Top-1
on NTU-60

PoseConv3D: The Architecture

Input:

1. Small Spatial Size (56 *vs.* 224)

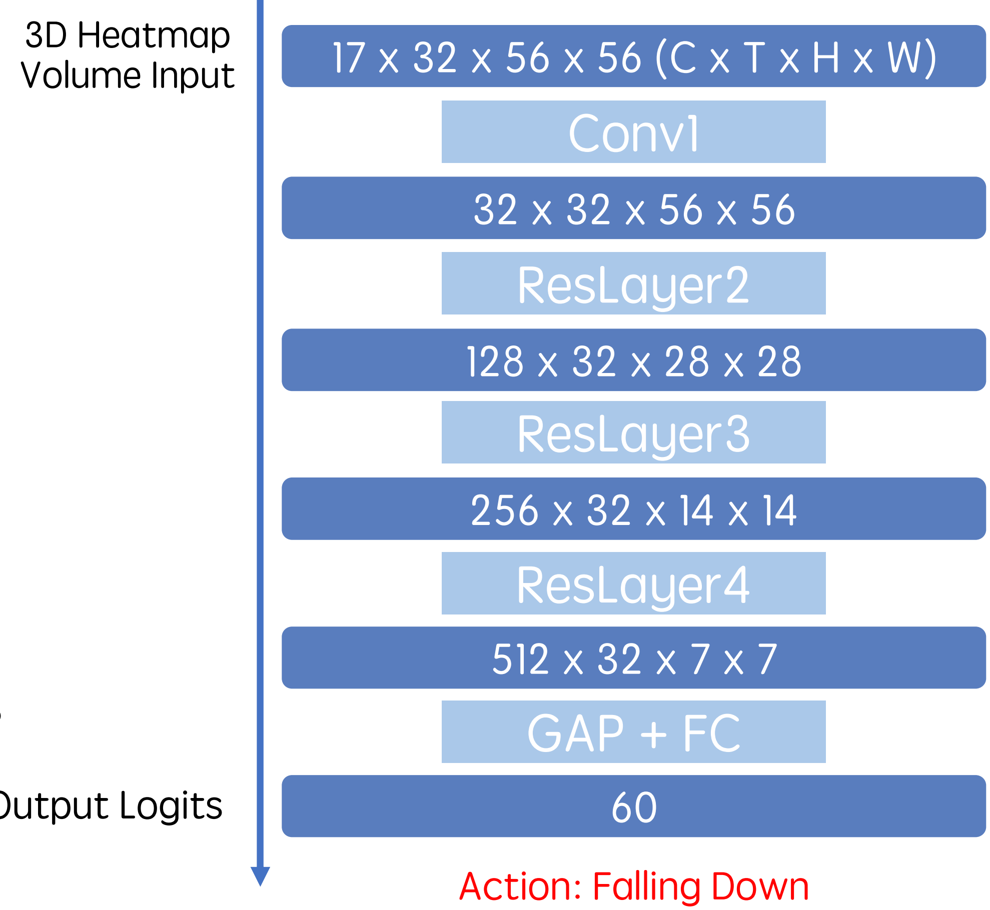
Model:

1. Small Channel Width (32 *vs.* 64)
2. Shallower (1 less stage)

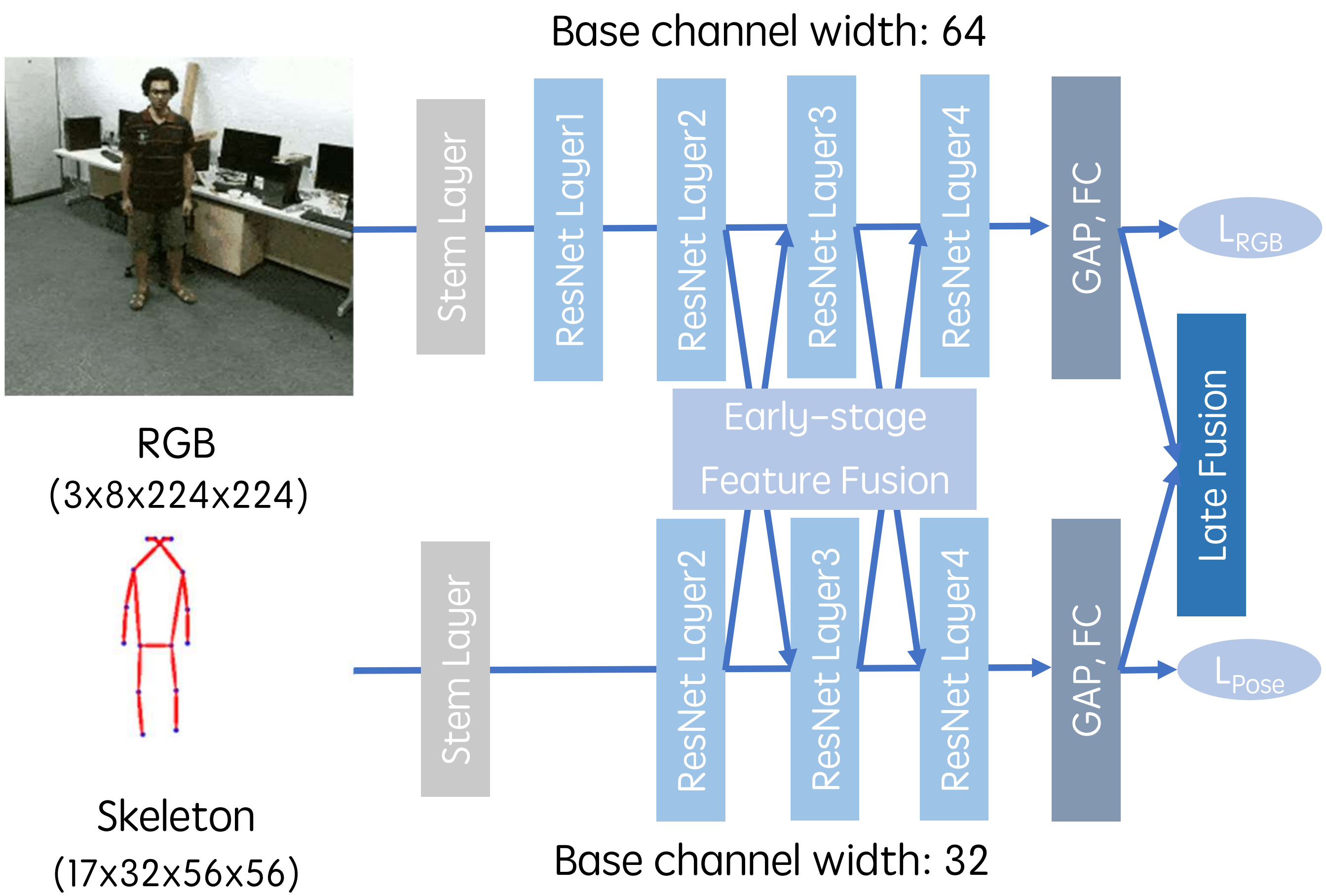
Processing a 32-frame clip

Pose: 10 GFLOPs << RGB: 157 GFLOPs

Adapting SlowOnly in PoseConv3D



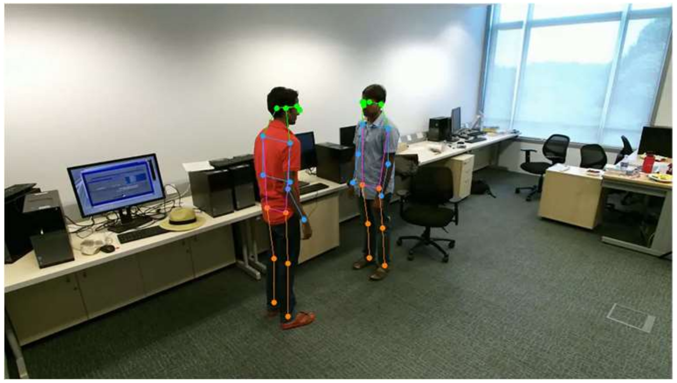
RGBPose-Conv3D



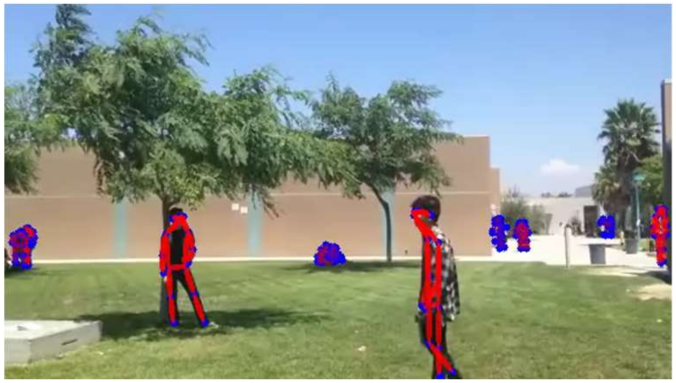
	1-clip	10-clip
Late-Fusion	92.6	93.4
RGB->Pose	93.0	93.7
Pose->RGB	93.4	93.8
Bi-directional	93.6	94.1

Bi-directional lateral connections outperform uni-directional ones.

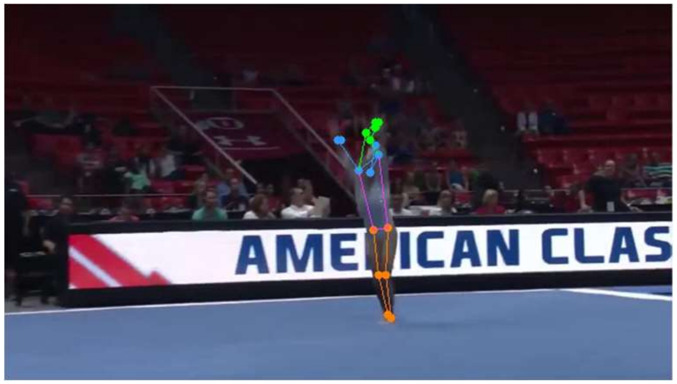
Experiments



NTURGB+D / NTURGB+D 120



Kinetics400 / UCF101 / HMDB51



FineGYM

Volleyball

Strong Recognition Performance

	GCN (MS-G3D [1])			3D-CNN (PoseSlowOnly)		
Dataset	Acc	Params	FLOPs	Acc	Params	FLOPs
FineGYM	92.0	2.8M	24.7G	92.4	2.0M	15.9G
NTU60 Xsub	91.9	2.8M	16.7G	93.1		
NTU120 Xsub	84.8	2.8M	16.7G	85.1		
Kinetics-400	44.9	2.8M	17.5G	44.8		

[1] Liu et al., Disentangling and unifying graph convolutions for skeleton-based action recognition, CVPR 2020

Other advantages to GCN

Robustness

Drop prob	0	1/8	1/4	1/2	1
GCN	92.0	91.0	90.2	86.5	77.7
GCN (robust train)	90.9	91.0	91.0	91.0	90.6
3D-CNN	**92.4**	**92.4**	**92.3**	**92.1**	**91.5**

Randomly drop 1 joint in each frame with prob p

Scalability

	GCN	3D-CNN
Params	2.8M	**0.52M**
FLOPs	7.2G	**1.6G**
Top-1	89.2	**91.3**

Scaling 3D-CNN requires no extra costs

Generalization

GCN Test/Train	MobileNet	HRNet	3D-CNN Test/Train	MobileNet	HRNet
MobileNet	89.0	79.3	MobileNet	**90.7**	**86.5**
HRNet	87.9	92.0	HRNet	**91.6**	**93.2**

Train & Test with poses from different sources

Interoperability

	RGB	Pose	LateFusion	RGBPose-Conv3D
FineGYM	87.2	91.0	92.6	**93.6**
NTU-60	94.1	92.8	93.5	**96.2**

Action Recognition with multiple modalities (1-clip test)

Comparison with SOTA

Method	NTU60-XSub	NTU60-XView	NTU120-XSub	NTU120-XSet	Kinetics	FineGYM
ST-GCN [63]	81.5	88.3	70.7	73.2	30.7	25.2*
AS-GCN [29]	86.8	94.2	78.3	79.8	34.8	-
RA-GCN [47]	87.3	93.6	81.1	82.7	-	-
AGCN [44]	88.5	95.1	-	-	36.1	-
DGNN [43]	89.9	96.1	-	-	36.9	-
FGCN [64]	90.2	96.3	85.4	87.4	-	-
Shift-GCN [9]	90.7	96.5	85.9	87.6	-	-
DSTA-Net [45]	91.5	96.4	86.6	89.0	-	-
MS-G3D [35]	91.5	96.2	86.9	88.4	38.0	-
MS-G3D ++	92.2	96.6	**87.2**	89.0	45.1	92.6
PoseConv3D (J)	**93.7**	**96.6**	86.0	**89.6**	**46.0**	**93.2**
PoseConv3D ($J + L$)	**94.1**	**97.1**	86.9	**90.3**	**47.7**	**94.3**

Results of skeleton-based action recognition.

Takeaways

Advantages

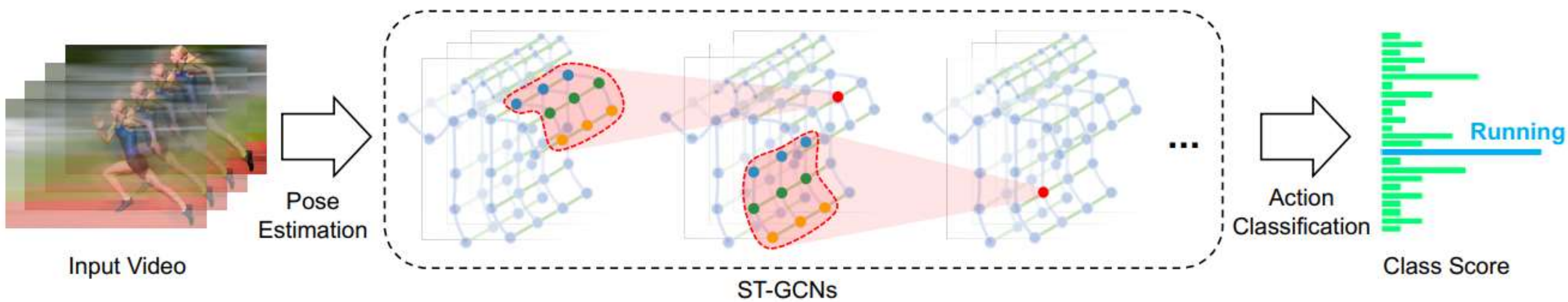
1. 2D skeletons: better quality -> improved recognition accuracy.
2. 3D-CNNs are of good spatio-temporal modeling capability.
3. 3D-CNN has unique pros in robustness, scalability, interoperability.

Future works

1. Extend to 3D skeleton.
2. More explorations on the architecture design.

GCN-based approaches

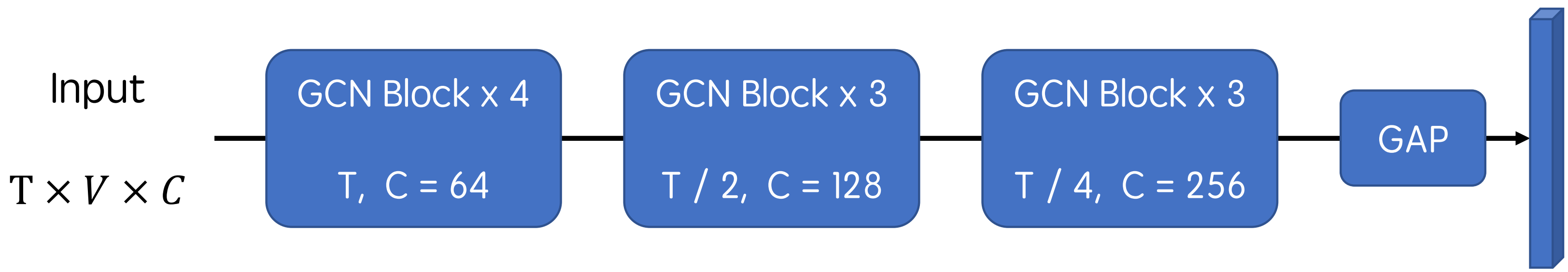
ST-GCN:



KeyNotes:

1. GCN take coordinate sequences as inputs (shape $\mathrm{T} \times V \times C$)
2. For multiple persons, GCN extracts features in parallel and average them.
3. A GCN recognizer is a stack of multiple GCN Blocks (like Bottleneck -> ResNet)

ST-GCN Arch



The forward fn of a GCN Block

```python
def forward(self, x, A=None):
    x = self.tcn(self.gcn(x, A)) + self.residual(x)
    return self.relu(x)
```

GCN Block = GCN Layer + TCN Layer

GCN Layer: Inter-Joint Feature Fusion with coeff matrix A (A.shape == (K, V, V))

TCN Layer: Temporal modeling with 1D convolutions (kernel 9)

ST-GCN Arch

TCN Layer:

```
class unit_tcn(nn.Module):
    def __init__(self,
                 in_channels,
                 out_channels,
                 kernel_size=9,
                 stride=1):
        super(unit_tcn, self).__init__()
        pad = (kernel_size - 1) // 2
        self.conv = nn.Conv2d(
            in_channels,
            out_channels,
            kernel_size=(kernel_size, 1),
            padding=(pad, 0),
            stride=(stride, 1))
        self.bn = nn.BatchNorm2d(out_channels)

    def forward(self, x):
        x = self.bn(self.conv(x))
        return x
```

A GCN Layer:

```
class unit_gcn(nn.Module):

    def __init__(self,
                 in_channels,
                 out_channels,
                 s_kernel=3):
        super().__init__()

        self.s_kernel = s_kernel
        self.conv = nn.Conv2d(
            in_channels,
            out_channels * s_kernel,
            kernel_size=1)

    def forward(self, x, A):
        # The shape of A is (s_kernel, V, V)
        assert A.size(0) == self.s_kernel
        x = self.conv(x)

        n, kc, t, v = x.size()
        x = x.view(n, self.s_kernel, kc // self.s_kernel, t, v)
        x = torch.einsum('nkctv,kvw->nctw', (x, A))
        return x.contiguous()
```

ST-GCN++: Better TCN

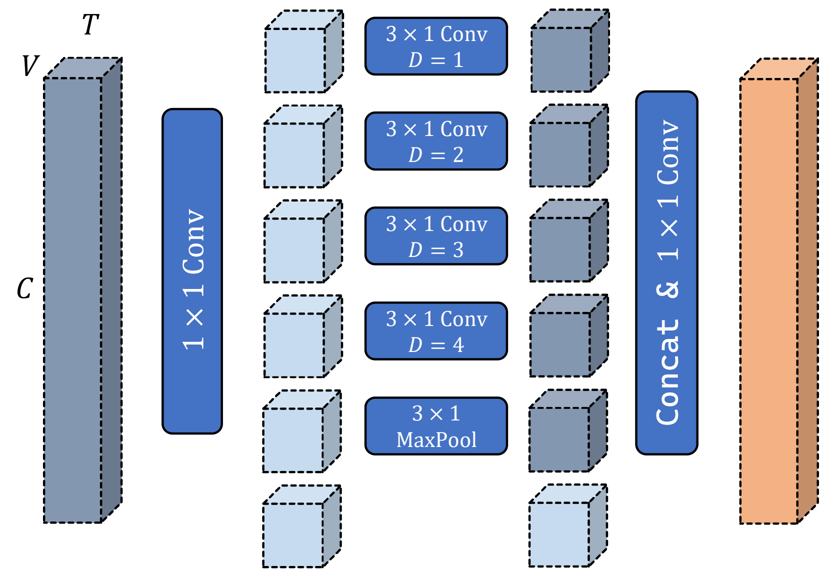
TCN (Old Version)

```python
class unit_tcn(nn.Module):
    def __init__(self,
                 in_channels,
                 out_channels,
                 kernel_size=9,
                 stride=1):
        super(unit_tcn, self).__init__()
        pad = (kernel_size - 1) // 2
        self.conv = nn.Conv2d(
            in_channels,
            out_channels,
            kernel_size=(kernel_size, 1),
            padding=(pad, 0),
            stride=(stride, 1))
        self.bn = nn.BatchNorm2d(out_channels)

    def forward(self, x):
        x = self.bn(self.conv(x))
        return x
```

A single 1D conv (kernel 9)

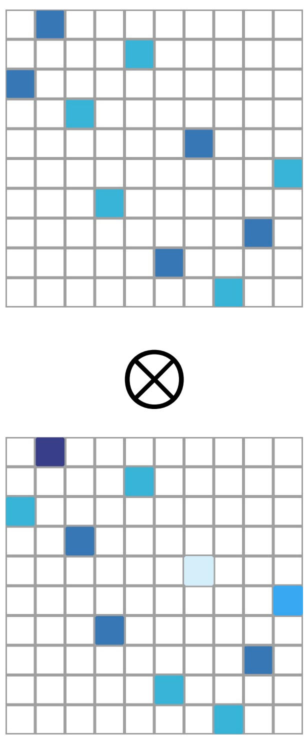
TCN (New Version)



Multiple branches with different D

ST-GCN++: Better GCN

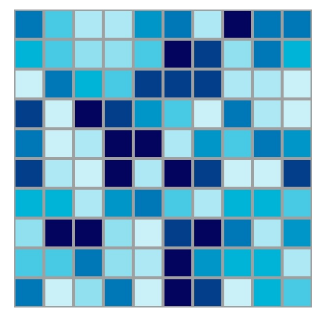
GCN (Old Version)



Pre-defined Sparse Coeff Matrix

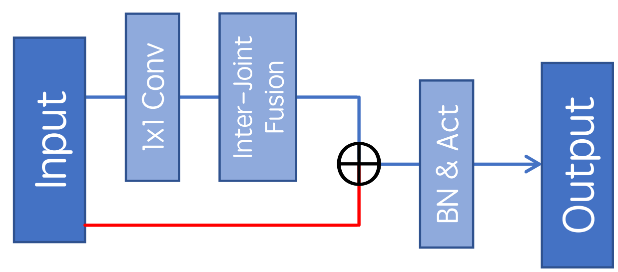
Learnable Edge Weights

GCN (New Version)



Learnable Coeff Matrix

Add Residual Connections



Other Good Practices

ST-GCN

Data Pre-Processing

- Data BN Only
- ZeroPad to 300 frames

HyperParam Setting

- MultiStep Scheduler
- Small Weight Decay (1e-4)

ST-GCN ++

Data Pre-Processing

- Data BN +
  - 1st frame center at (0, 0, 0)
  - 1st frame spine // z-axis
- UniformSample to get 100 frames

HyperParam Setting

- CosineAnnealing Scheduler
- Large Weight Decay (5e-4 or 1e-3)

Strong Performance (Ranking @ PapersWithCode)

Model	Annotation	Setting	NTU60 XSub	NTU60 Xview	NTU120 Xsub	NTU120 Xset
STGCN	3D	Vanilla	86.6 [#46]	93.2 [#47]	–	–
STGCN++	3D	PYSKL	92.6 [#3]	97.4 [#3]	88.6 [#3]	90.8 [#1]
STGCN	2D	Vanilla	90.1 [#23]	95.1 [#29]	–	–
STGCN++	2D	PYSKL	93.2 [#2]	98.5 [#1]	86.4 [#13]	90.3 [#2]
AAGCN	3D	–	90.0 [#24]	96.2 [#17]	–	–
MS-G3D	3D	–	91.5 [#12]	96.2 [#17]	86.9 [#10]	88.4 [#12]
CTRGCN	3D	–	92.4 [#4]	96.8 [#5]	88.9 [#1]	90.6 [#1]
PoseC3D	2D	–	94.1 [#1]	97.1 [#3]	86.9 [#10]	90.3 [#2]

ST-GCN++ is a simple & strong baseline

, not a complicated so-called SOTA model

Used

✓ Good practices for data pre-processing

✓ Strong spatial & temporal augmentations

✓ Simple improvement in structure design

✓ Well-tuned hyper-param settings

Not Used

❌ Attention schemes

❌ Sample-dependent coefficient matrices

❌ Other novel designs or training schemes

Codes are available in PYSKL

PoseConv3D Paper

STGCN++ Report

PYSKL Code

PYSKL: A Skeleton Action Recognition Toolbox

- Algorithms of strong recognition performance with good practices & extremely simple design
- Large model zoo: 6 algorithms and 9 benchmarks
- Distributed training and testing with DDP (much faster than DP, used in other repos)
- Ready-to-go pickle annotations files for users
- Visualization of 2D / 3D skeletons
- Tools for building skeleton annotation files with your custom video dataset

Code

Report

What's Next?

- The performance on traditional benchmarks is nearly saturated 🤔

Several Numbers (Top 1):

NTURGB+D (60 classes): 94.1% (XSub), 97.4% (XView)

NTURGB+D 120 (120 classes): 88.9% (XSub), 90.8% (XSet)

Kinetics 400 (400 classes): 49.1% (Due to low quality poses)

What to do next?

- For broader applications: data efficiency
- For deployment: computational efficiency

Data Efficiency

- In current skeleton action recognition benchmarks (like NTU), each action category has hundreds of training samples.

With fewer training samples?

1. Pretraining
    - Massive Web Videos -> Automatically generated 2D poses -> Self-supervised pretraining
2. Adaptation

Computational Efficiency

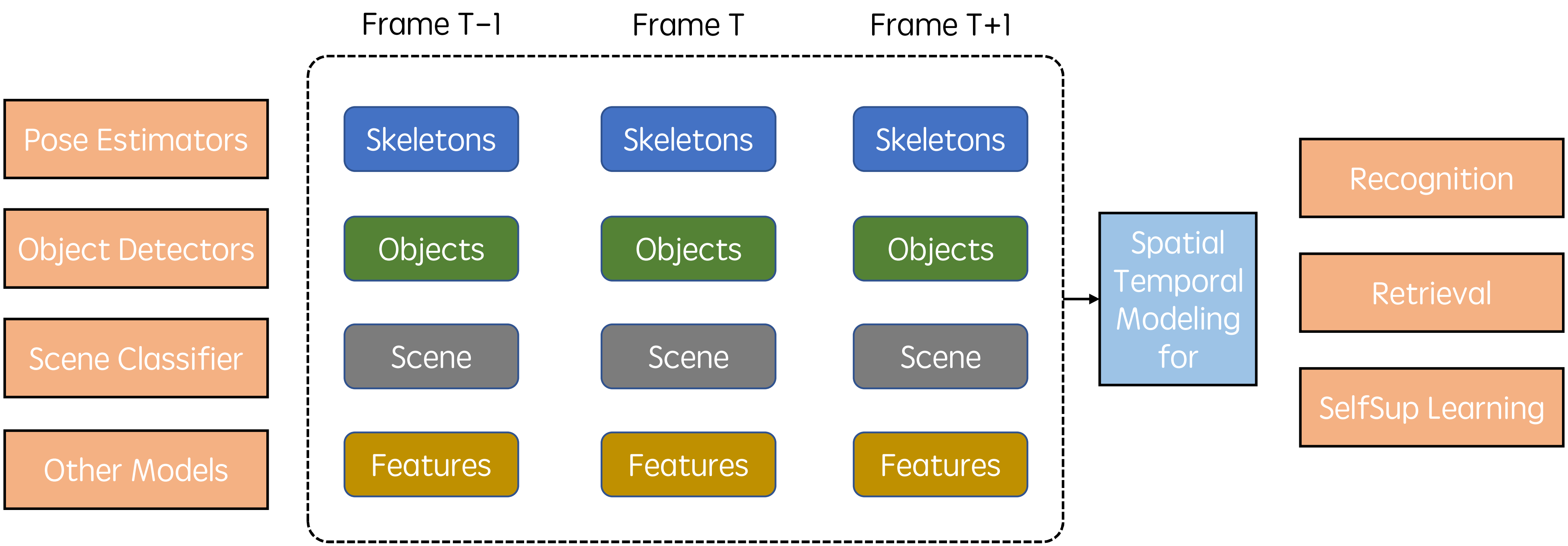
Accelerate the three components (can be realtime)

Detection: YOLO v5 (100+ FPS GPU)
Pose: Fast Implementations (60+ FPS CPU)
Action: STGCN++ already fast enough (>80+ sample/s per GPU)
Write a pipeline to combine them.

Skeleton + X: The Goal and Challenge

- Motivation
  - Some Actions can not be recognized solely based on skeleton
- Goal
  - Utilize other cues in videos (object, scene, *e.g.*) while keeping the good properties of skeleton, *i.e.*, lightweight, robust.
  - Direct multi-stream fusion ≈ RGB-based action recognition, which does not have those good properties

Modeling mid-level features



Thanks for your attention!

Email: dhd.efz@gmail.com, Poster: Jun 21 afternoon 40b