

# 東吳演講-2021-12-29

**Title: Introduction to human action recognition**

## **Motivation**

**Q: Why we want to study the topic?**

**A:**

For fun!! ( we also can use the knowledge to control computer. )



source: <https://game.nownews.com/news/20191113/3245670/>

---

Safety ( Automatically remind hospitals that someone needs help!! )







source: <https://health.tvbs.com.tw/medical/314760>

Health ( Teaching fitness action )



source: <https://shop.lululemon.com/story/mirror-home-gym>

Difficulty:

	classification	fixing your action
video action recognition		
real time action recognition		

## Machine Learning Method

Raw Data → feature → class

- Example (BMI)  
BMI = 體重(公斤) / 身高<sup>2</sup>(公尺<sup>2</sup>)  
稍微肥胖：24 ≤ BMI < 27  
輕度肥胖：27 ≤ BMI < 30  
中度肥胖：30 ≤ BMI < 35  
重度肥胖：BMI ≥ 35
- Raw Data (某路人甲) → feature ( 算出他的 BMI ) → class (分類為哪種肥胖)



- Deep learning use a way to learn that how to find the feature.
- 

## Regression

Assume  $x_1, x_2, x_3$  are features

- We can use  $A_p = a_1x_1 + a_2x_2 + a_3x_3 + a$  to predict our target  
( if  $0 < A_p < 100$  ,  $A_p$  maybe predict your score ).

## Classification

Assume we have two classes A and B.

- $A_p = a_1x_1 + a_2x_2 + a_3x_3 + a$
- $B_p = b_1x_1 + b_2x_2 + b_3x_3 + b$
- if  $A_p > B_p$  , we predict class A.

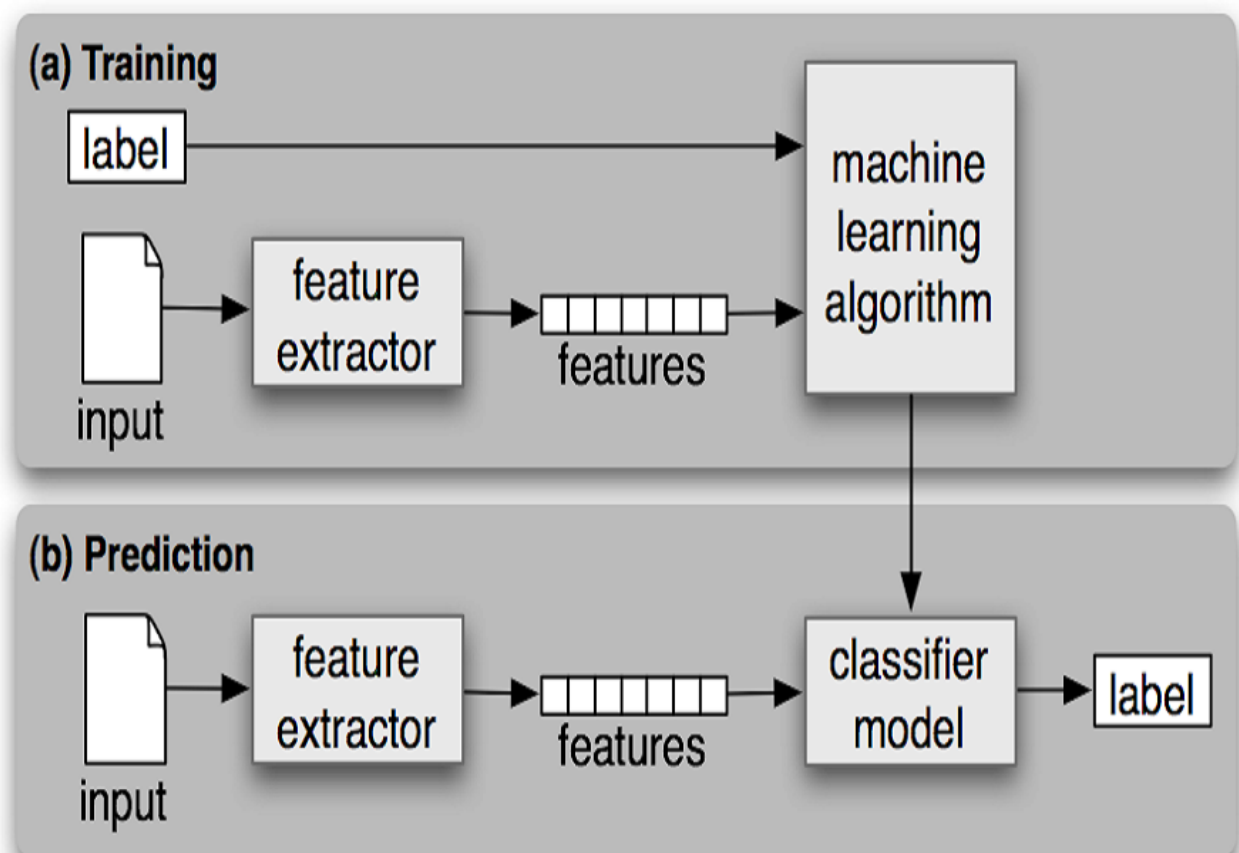
## Softmax function

$$f_i(x) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

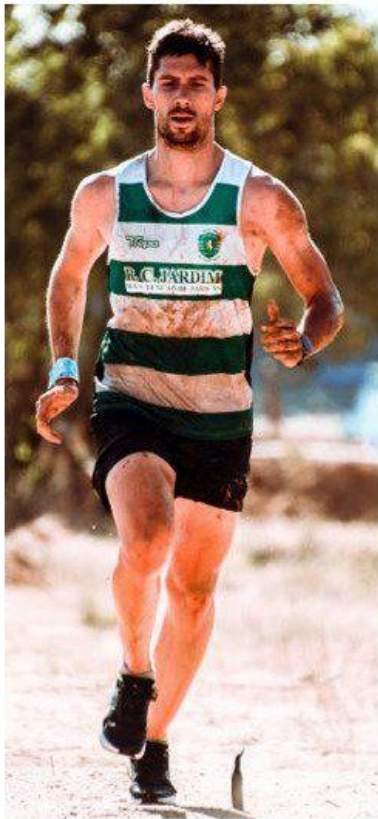
# Conclusion

- We know how to do regression and classification.

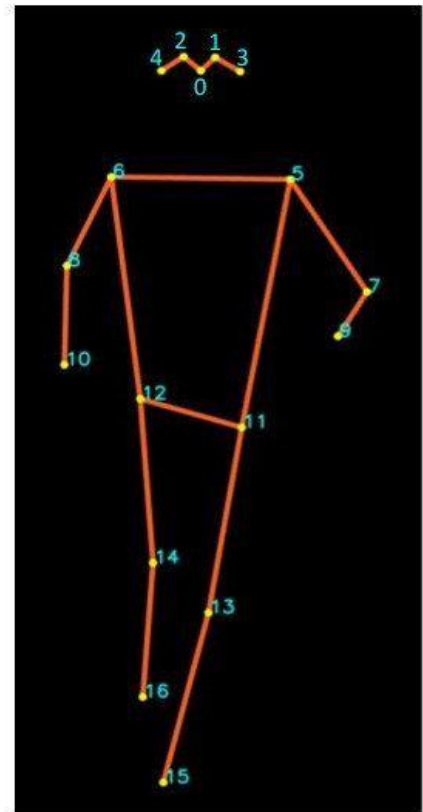
## Normal Classify Model.



# Human Skeleton



Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



source : <https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/>

---

---

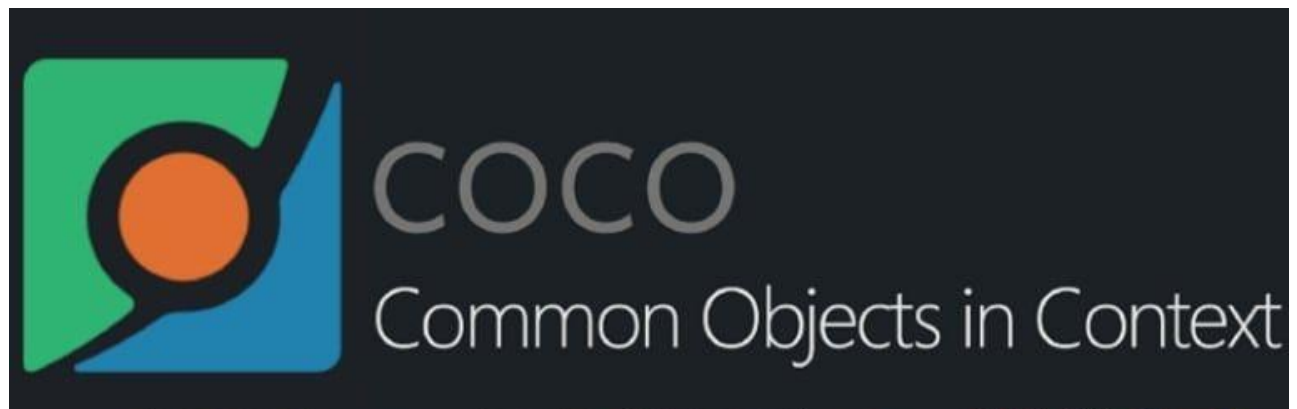
## Introduce DataSets

- [MPII](#) ( max planck institut informatik ) Human Pose Models



*MPII dataset*

- 
- 
- [coco](#) ( Common Objects in Context )



COCO Dataset

nk7260ynpa 20190406

*Coco dataset*

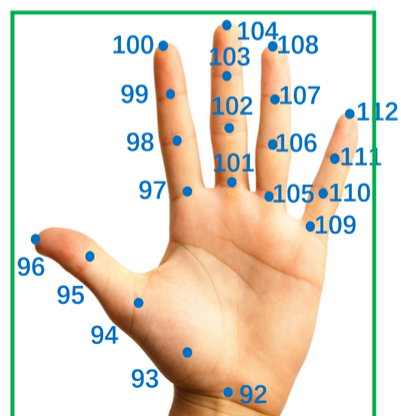
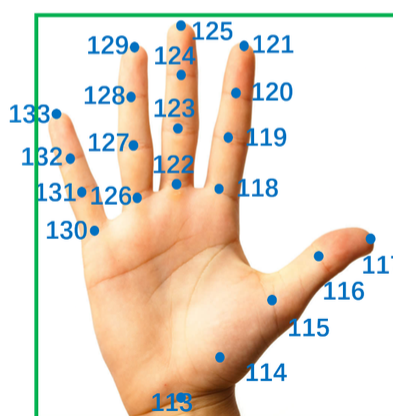
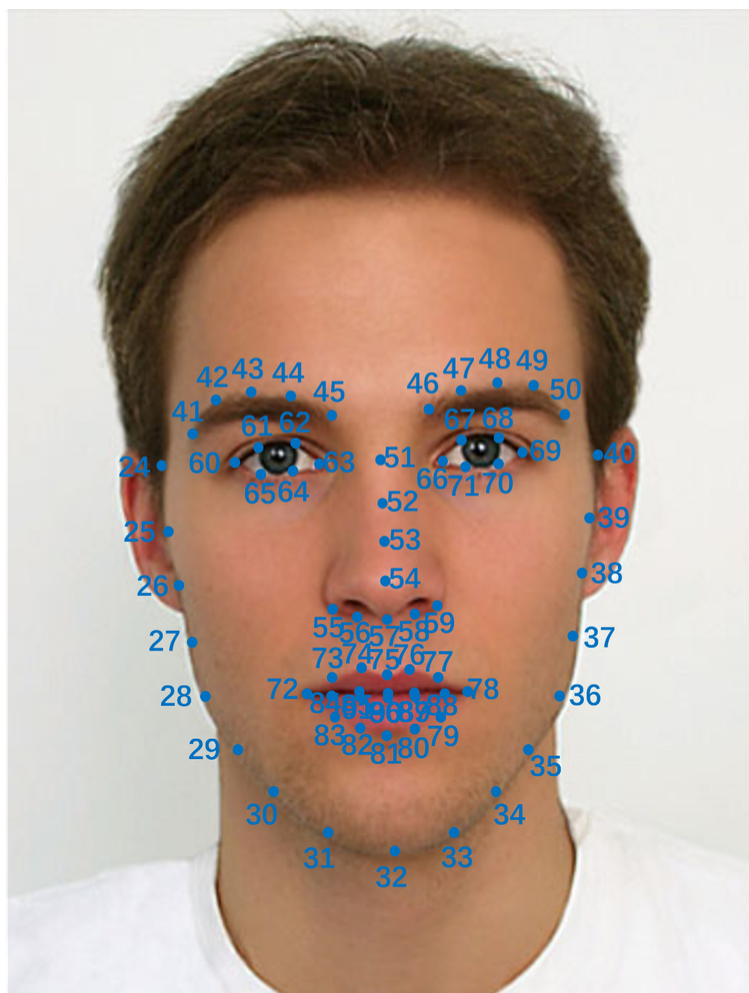
- 
- [COCO-WholeBody](#)

COCO-WholeBody = [COCO 2017](#) ++



133 keypoints (17 for body, 6 for feet, 68 for face and 42 for hands)



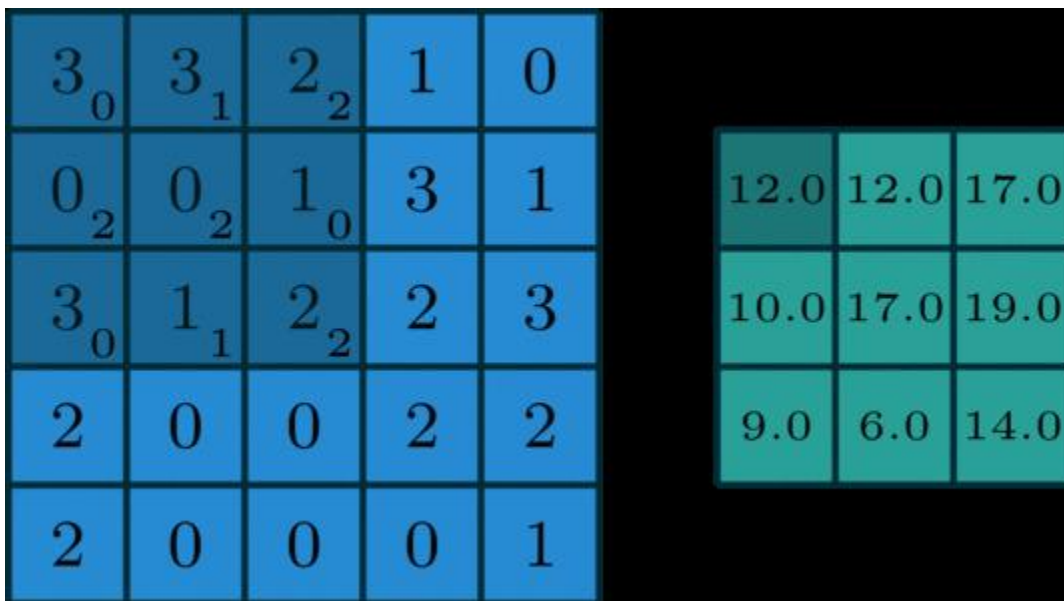




# How to find the datasets which you in need.

- BiFrost
  - Google
  - Kaggle
- 

## Convolutional Neural Networks (CNN's)



source: <https://blog.csdn.net/zhangphil/article/details/103067872>

1	0	1
0	1	0
1	0	1

*Example X*

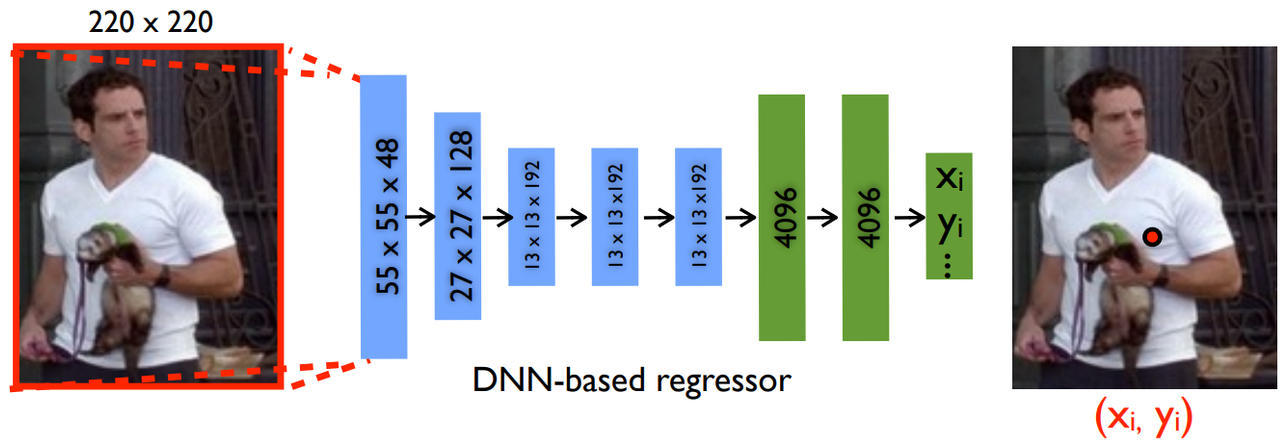
---

---

**A Problem : Finding Human Skeleton is not a classification problem.**

Regression or Heatmap?

**Regression**



*DeepPose CVPR 2014*

---

---

Heatmap ( now, a common method is heatmap )



*Hourglass Networks ECCV 2016*

### Note of Conference:

- ICCV ( International Conference on Computer Vision )
  - CVPR ( International Conference on Computer )
  - ECCV ( European Conference on Computer Vision )
- 
-

Another Problem: if we have more than one people



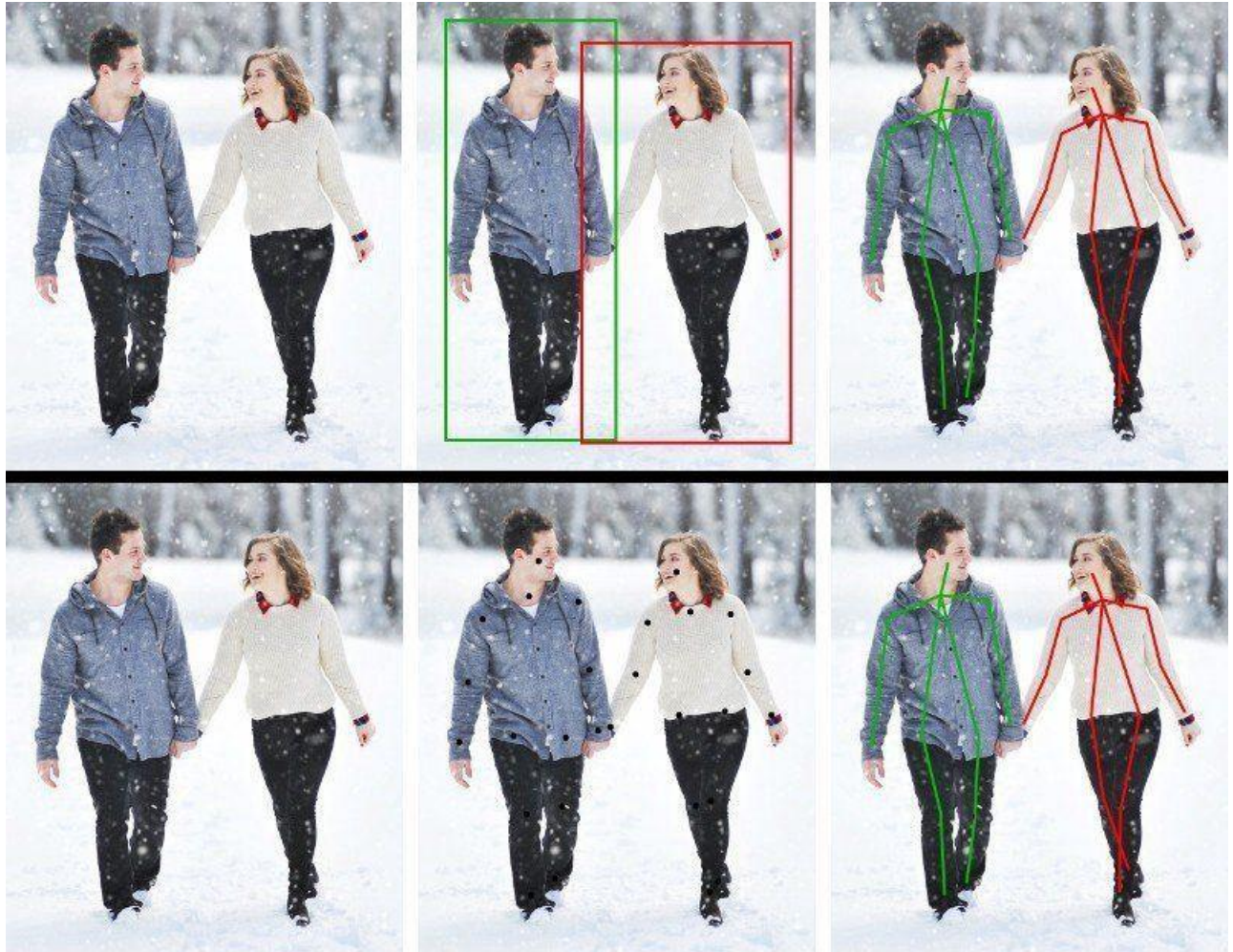
*Multi Person*

source: <https://www.youtube.com/watch?v=6DW2fD4WoMY>

---

---

Top-down and Bottom-up



source: <https://beyondminds.ai/blog/an-overview-of-human-pose-estimation-with-deep-learning/>

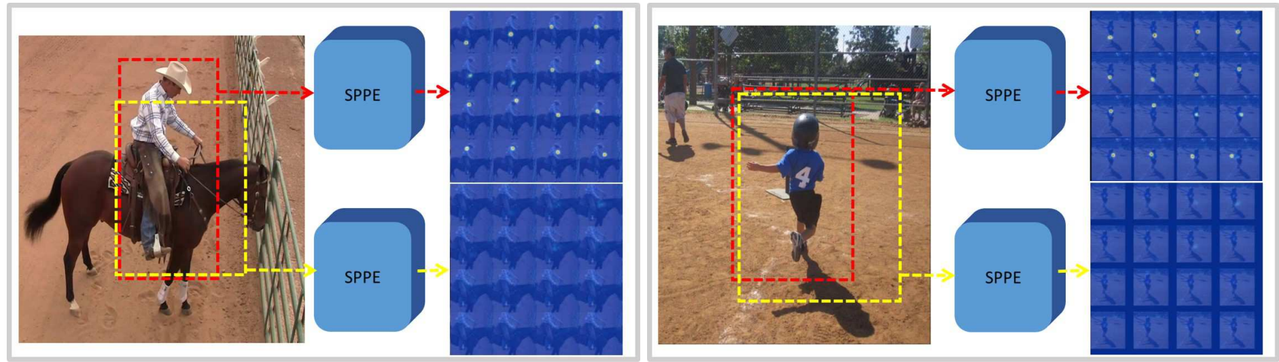
---

---

## AlphaPose (Top-down)

Stacked Hourglass Networks for Human Pose Estimation





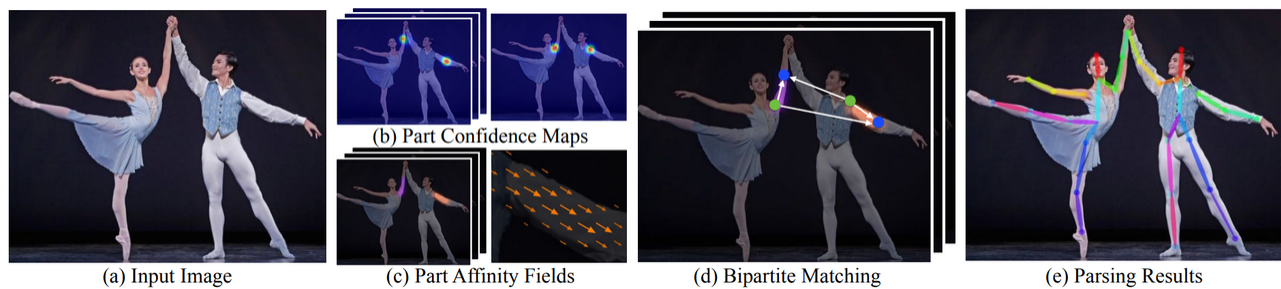
RMPE ICCV 2017, <https://arxiv.org/pdf/1612.00137.pdf>

Regional Multi-person Pose Estimation:

- The target bounding box is important.

## OpenPose (Bottom-up)

Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, CVPR 2017



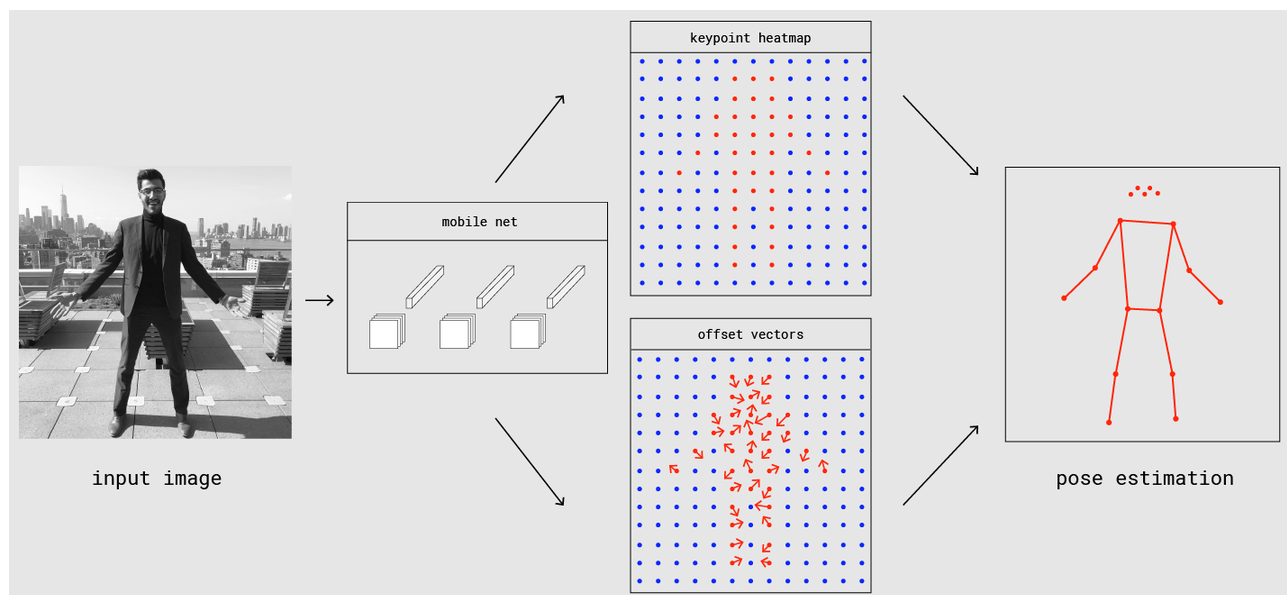
OpenPose CVPR 2017, <https://arxiv.org/pdf/1812.08008.pdf>

- Part Confidence Maps ( Heatmap )
- Part Affinity Fields

## Top-down vs Bottom-up

- The speed

## PoseNet ( Single Person )



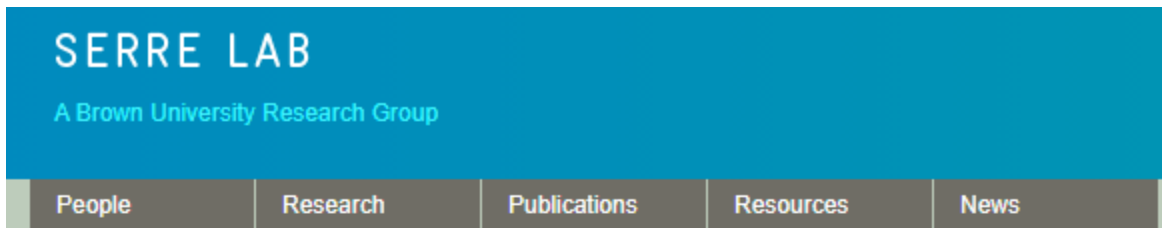
- Heatmap
- Offset

## Human Actions

### Introduce DataSets



- [HMDB51](#)



[< back to Resources](#)

## HMDB: a large human motion database

[Evaluation](#)

[Download](#)

[Illustration of the 51 Actions](#)

[Introduction](#)

[Citation](#)

[Dataset](#)

[Other action recognition benchmark](#)

[About the page](#)

[Evaluation](#)

Current benchmarks provided by [actionrecognition.net](#):

- [AVA](#)



AVA is a project that provides audiovisual annotations of video for improving our understanding of human activity. Each of the video clips has been exhaustively annotated by human annotators, and together they represent a rich variety of scenes, recording conditions, and expressions of human activity.

We provide the following annotations.

### [AVA-Kinetics Dataset](#)

AVA-Kinetics, our latest release, is a crossover between the AVA Actions and [Kinetics](#) datasets. In order to provide localized action labels on a wider variety of visual scenes, we've provided AVA action labels on videos from Kinetics-700, nearly doubling the number of total annotations, and increasing the number of unique videos by over 500x. We hope this will expand the generalizability of localized action models, and open the door to new approaches in multi-task learning.

AVA-Kinetics is described in detail in [the arXiv paper](#).

AVA-Kinetics is now available for [download](#). It was the basis of a [challenge](#) in partnership with the [ActivityNet workshop](#) at CVPR 2020.

### [AVA Actions Dataset](#)

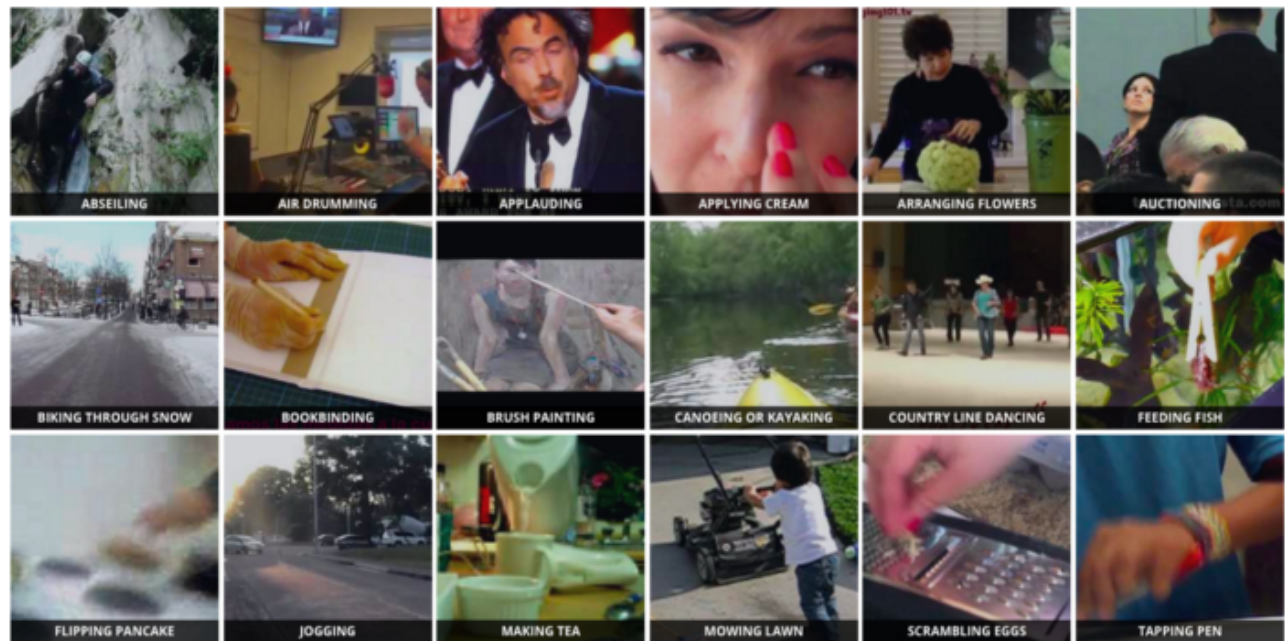
The AVA dataset densely annotates 80 atomic visual actions in 430 15-minute movie clips, where actions are localized in space and time, resulting in 1.62M action labels with multiple labels per human occurring frequently. A detailed description of our contributions with this dataset can be found in our accompanying [CVPR '18 paper](#).

AVA v2.2 is now available for [download](#). It was the basis of a [challenge](#) in partnership with the [ActivityNet workshop](#) at CVPR 2019.

- UCF101



- Kinetics

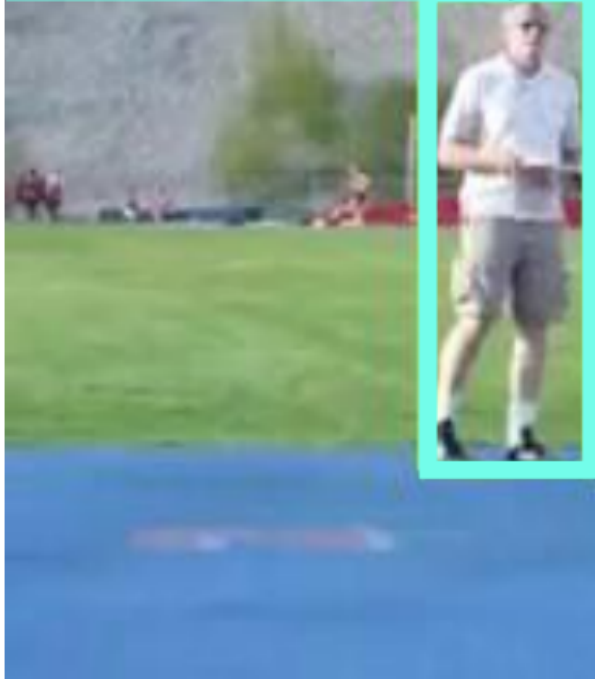


## AVA-Kinetics Datasets ( [link](#) ) = AVA Actions + Kinetics Datasets

AVA (Atomic Visual Actions)



**carry/hold (an object);  
stand;  
talk to (e.g., self, a person,  
a group);  
watch (a person)**



**jump/leap;  
touch (an object)**



**high jump**

source: AVA paper ( <https://arxiv.org/pdf/2005.00214.pdf> )

## AVA Json (ava.json) 80 classes

```
{  
  "bend/bow (at the waist)": 0,  
  "crawl": 1,  
  "crouch/kneel": 2,  
  "dance": 3,  
  "fall down": 4,  
  "get up": 5,  
  "jump/leap": 6,
```

"lie/sleep": 7,  
"martial art": 8,  
"run/jog": 9,  
"sit": 10,  
"stand": 11,  
"swim": 12,  
"walk": 13,  
"answer phone": 14,  
"brush teeth": 15,  
"carry/hold (an object)": 16,  
"catch (an object)": 17,  
"chop": 18,  
"climb (e.g., a mountain)": 19,  
"clink glass": 20,  
"close (e.g., a door, a box)": 21,  
"cook": 22,  
"cut": 23,  
"dig": 24,  
"dress/put on clothing": 25,  
"drink": 26,  
"drive (e.g., a car, a truck)": 27,  
"eat": 28,  
"enter": 29,  
"exit": 30,  
"extract": 31,  
"fishing": 32,  
"hit (an object)": 33,  
"kick (an object)": 34,  
"lift/pick up": 35,  
"listen (e.g., to music)": 36,

"open (e.g., a window, a car door)": 37,  
"paint": 38,  
"play board game": 39,  
"play musical instrument": 40,  
"play with pets": 41,  
"point to (an object)": 42,  
"press": 43,  
"pull (an object)": 44,  
"push (an object)": 45,  
"put down": 46,  
"read": 47,  
"ride (e.g., a bike, a car, a horse)": 48,  
"row boat": 49,  
"sail boat": 50,  
"shoot": 51,  
"shovel": 52,  
"smoke": 53,  
"stir": 54,  
"take a photo": 55,  
"text on/look at a cellphone": 56,  
"throw": 57,  
"touch (an object)": 58,  
"turn (e.g., a screwdriver)": 59,  
"watch (e.g., TV)": 60,  
"work on a computer": 61,  
"write": 62,  
"fight/hit (a person)": 63,  
"give/serve (an object) to (a person)": 64,  
"grab (a person)": 65,  
"hand clap": 66,

```
"hand shake": 67,  
"hand wave": 68,  
"hug (a person)": 69,  
"kick (a person)": 70,  
"kiss (a person)": 71,  
"lift (a person)": 72,  
"listen to (a person)": 73,  
"play with kids": 74,  
"push (another person)": 75,  
"sing to (e.g., self, a person, a group)": 76,  
"take (an object) from (a person)": 77,  
"talk to (e.g., self, a person, a group)": 78,  
"watch (a person)": 79  
}
```

---

---

## SlowFast ( 2019 Facebook AI Research )

### Motivation:

- [Visual System](#)
- M cells:  
with large center-surround receptive fields that are sensitive to depth, indifferent to color, and rapidly adapt to a stimulus.



- P cells:  
with smaller center-surround receptive fields that are sensitive to color and shape.
- 

## Dynamic Vision Sensor (DVS)

A normal camera send picture. But the DVS sends an event whenever the illumination of a pixel changes.

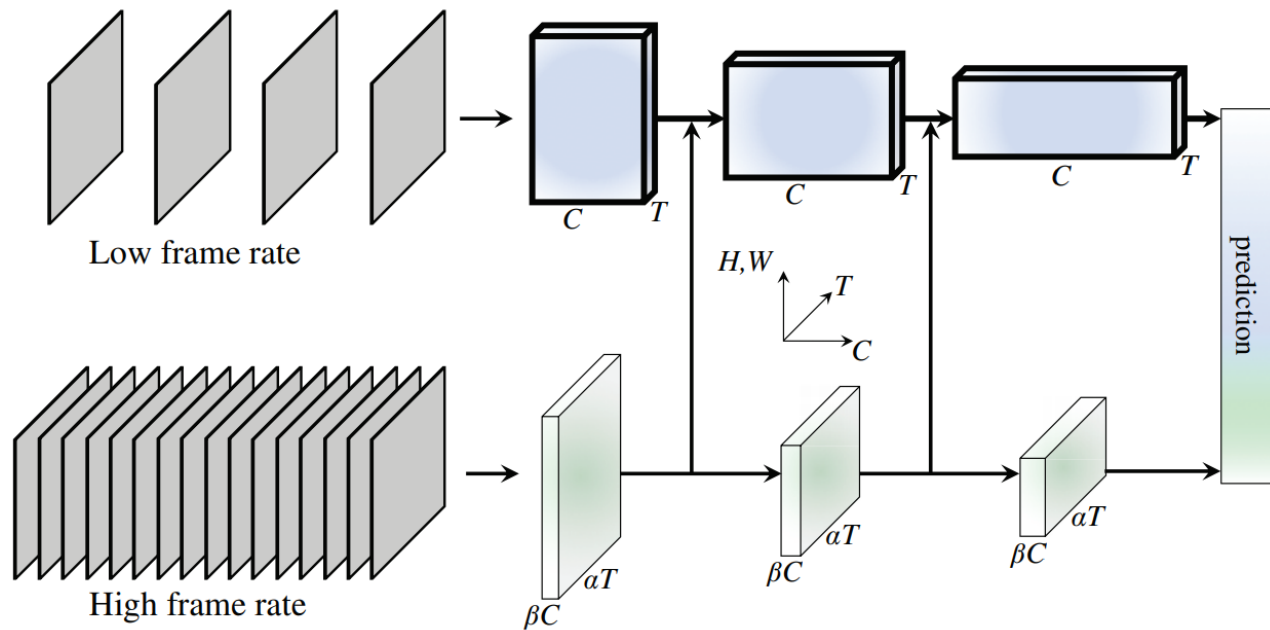
### Advantage

- speed
- small volume

### Dataset

- [DHP19](#)
  - Watching DHP19 video.
  - [Event-based Vision Sensor \(EVS\)](#)
- 

## The Big Picture



SlowFast ICCV 2019, <https://arxiv.org/pdf/1812.03982.pdf>

The ideal:

If the video has  $T * t$  frames

- Slow pathway : to capture spatial semantics  
Slow choose one frame for each  $t$  frames.  
So it contain  $T$  frames.
- Fast pathway : to capture motion feature  
Fast choose one frame for each  $t/s$  frames. ( $s > 1$ )  
So it contain  $T * s$  frames.

Assume  $T=10, t=3, s=3$

Total frames :  $30 = 10 * 3 = T * t$

Slow frames :  $10 = 30 / 3 = (T * t) / t$

Fast frames :  $30 = 30 / 1 = (T * t) / (t/s)$

stage	<i>Slow</i> pathway	<i>Fast</i> pathway	output sizes $T \times S^2$
raw clip	-	-	$64 \times 224^2$
data layer	stride 16, $1^2$	stride <b>2</b> , $1^2$	<i>Slow</i> : $4 \times 224^2$ <i>Fast</i> : <b>32</b> $\times 224^2$
conv <sub>1</sub>	$1 \times 7^2$ , 64 stride 1, $2^2$	$\underline{5 \times 7^2}$ , <b>8</b> stride 1, $2^2$	<i>Slow</i> : $4 \times 112^2$ <i>Fast</i> : <b>32</b> $\times 112^2$
pool <sub>1</sub>	$1 \times 3^2$ max stride 1, $2^2$	$1 \times 3^2$ max stride 1, $2^2$	<i>Slow</i> : $4 \times 56^2$ <i>Fast</i> : <b>32</b> $\times 56^2$
res <sub>2</sub>	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \underline{3 \times 1^2}, \textcolor{brown}{8} \\ \underline{1 \times 3^2}, \textcolor{brown}{8} \\ 1 \times 1^2, \textcolor{brown}{32} \end{bmatrix} \times 3$	<i>Slow</i> : $4 \times 56^2$ <i>Fast</i> : <b>32</b> $\times 56^2$
res <sub>3</sub>	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \underline{3 \times 1^2}, \textcolor{brown}{16} \\ \underline{1 \times 3^2}, \textcolor{brown}{16} \\ 1 \times 1^2, \textcolor{brown}{64} \end{bmatrix} \times 4$	<i>Slow</i> : $4 \times 28^2$ <i>Fast</i> : <b>32</b> $\times 28^2$
res <sub>4</sub>	$\begin{bmatrix} \underline{3 \times 1^2}, 256 \\ \underline{1 \times 3^2}, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \underline{3 \times 1^2}, \textcolor{brown}{32} \\ \underline{1 \times 3^2}, \textcolor{brown}{32} \\ 1 \times 1^2, \textcolor{brown}{128} \end{bmatrix} \times 6$	<i>Slow</i> : $4 \times 14^2$ <i>Fast</i> : <b>32</b> $\times 14^2$
res <sub>5</sub>	$\begin{bmatrix} \underline{3 \times 1^2}, 512 \\ \underline{1 \times 3^2}, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \underline{3 \times 1^2}, \textcolor{brown}{64} \\ \underline{1 \times 3^2}, \textcolor{brown}{64} \\ 1 \times 1^2, \textcolor{brown}{256} \end{bmatrix} \times 3$	<i>Slow</i> : $4 \times 7^2$ <i>Fast</i> : <b>32</b> $\times 7^2$
global average pool, concat, fc			# classes

The Convolution size formula:

- X: the input size ( if your image shape is 10X10X3, then X=10)
- K: the kernel size
- P: the padding size
- S: the stride

The output size is  $[(X - K + 2P)/S] + 1$

The fc is fully connected layers.

