

Computer vision tasks and corresponding NNs

Alex Net
VGG16
Res Net

U Net

Yolo
Faster RCNN

Mask RCNN

OpenPose
Keypoints RCNN

Classification



Semantic Segmentation



GRASS, CAT,
TREE, SKY

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Joint detection

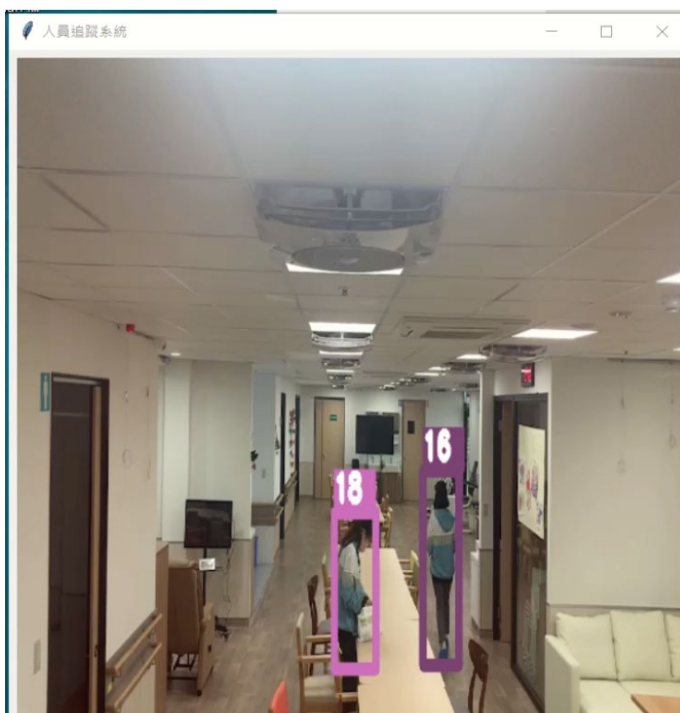


圖片來源: <https://kharshit.github.io/blog/2019/08/23/quick-intro-to-instance-segmentation>

Computer vision tasks and corresponding NNs

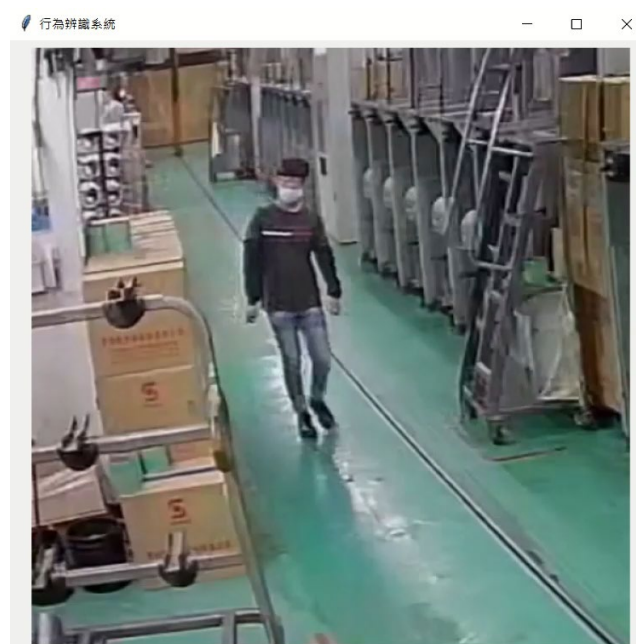
SORT, ByteTrack
DeepSORT, JDE

Object tracking



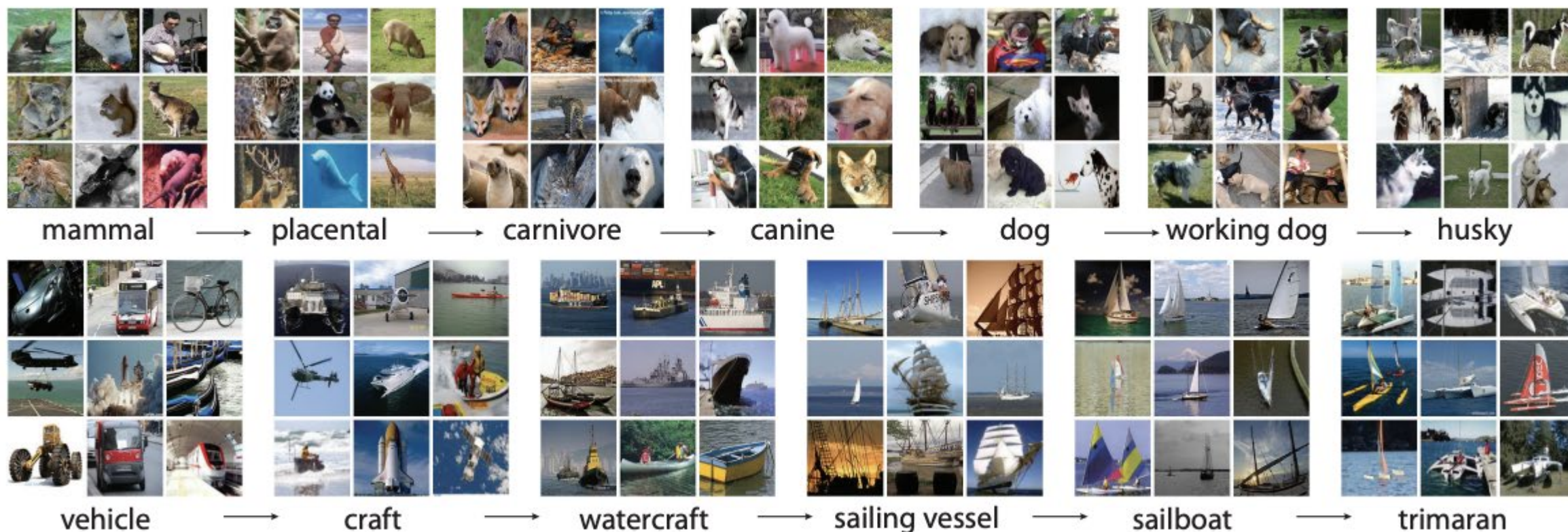
SlowFast

Action classification



21:10:22 Action = spraying 0.22, cleaning floor 0.18, garbage c
ollecting 0.16,
21:10:32 Action =

ImageNet dataset



The 1000 categories in ImageNet

1	{0: 'tench, Tinca tinca',	986	985: 'daisy',
2	1: 'goldfish, Carassius auratus',	987	986: "yellow lady's slipper, yellow lady-slipper, Cypripedium calc
3	2: 'great white shark, white shark, man-eater, man-eating shark, Carcharodon carcha	988	987: 'corn',
4	3: 'tiger shark, Galeocerdo cuvieri',	989	988: 'acorn',
5	4: 'hammerhead, hammerhead shark',	990	989: 'hip, rose hip, rosehip',
6	5: 'electric ray, crampfish, numbfish, torpedo',	991	990: 'buckeye, horse chestnut, conker',
7	6: 'stingray',	992	991: 'coral fungus',
8	7: 'cock',	993	992: 'agaric',
9	8: 'hen',	994	993: 'gyromitra',
10	9: 'ostrich, Struthio camelus',	995	994: 'stinkhorn, carrion fungus',
11	10: 'brambling, Fringilla montifringilla',	996	995: 'earthstar',
12	11: 'goldfinch, Carduelis carduelis',	997	996: 'hen-of-the-woods, hen of the woods, Polyporus frondosus, Gri
13	12: 'house finch, linnet, Carpodacus mexicanus',	998	997: 'bolete',
14	13: 'junco, snowbird',	999	998: 'ear, spike, capitulum',
15	14: 'indigo bunting, indigo finch, indigo bird, Passerina cyanea'	1000	999: 'toilet tissue, toilet paper, bathroom tissue'}

<https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a>

Coco dataset and 80 categories included in Coco

person	fire hydrant	elephant	skis	wine glass	broccoli	dining table	toaster
bicycle	stop sign	bear	snowboard	cup	carrot	toilet	sink
car	parking meter	zebra	sports ball	fork	hot dog	tv	refrigerator
motorcycle	bench	giraffe	kite	knife	pizza	laptop	book
airplane	bird	backpack	baseball bat	spoon	donut	mouse	clock
bus	cat	umbrella	baseball glove	bowl	cake	remote	vase
train	dog	handbag	skateboard	banana	chair	keyboard	scissors
truck	horse	tie	surfboard	apple	couch	cell phone	teddy bear
boat	sheep	suitcase	tennis racket	sandwich	potted plant	microwave	hair drier
traffic light	cow	frisbee	bottle	orange	bed	oven	toothbrush

<https://opencv.org/introduction-to-the-coco-dataset/>

Kinects-400 data set



(a) headbanging



(b) stretching leg



(e) robot dancing



(c) shaking hands



(d) tickling



(g) riding a bike



(e) robot dancing



(f) salsa dancing








(i) playing violin

<https://www.deepmind.com/open-source/kinetics>

400 categories in Kinects-400

1. abseiling (1146)
2. air drumming (1132)
3. answering questions (478)
4. applauding (411)
5. applying cream (478)
6. archery (1147)
7. arm wrestling (1123)
8. arranging flowers (583)
9. assembling computer (542)
10. auctioning (478)
11. baby waking up (611)
12. baking cookies (927)
13. balloon blowing (826)
14. bandaging (569)
386. watering plants (680)
387. waxing back (537)
388. waxing chest (760)
389. waxing eyebrows (720)
390. waxing legs (948)
391. weaving basket (743)
392. welding (759)
393. whistling (416)
394. windsurfing (1114)
395. wrapping present (861)
396. wrestling (488)
397. writing (735)
398. yawning (398)
399. yoga (1140)
400. zumba (1093)

Pre-trained NN for CV tasks

Torchvision Detectron2	  Detectron2	Alex Net, VGG16, Res Net Faster RCNN, Keypoints RCNN
PyTorch Hub		SlowFast
GitHub	 GitHub	SORT, DeepSORT, JDE, ByteTrack
MicroSoft CV recipes		https://github.com/microsoft/computervision-recipes

Our own CV tasks

	Fine-tuned pre-trained NN	CV task
HW3	ImageNet pre-trained VGG19	
HW4	COCO pre-trained FasterRCNN	
HW5	Kinects-400 pre-trained SlowFast	

Video classification

SlowFast (1).ipynb

Load pre-trained NN

```
import torch
model = torch.hub.load('facebookresearch/pytorchvideo', 'slowfast_r50', pretrained=True)
```


Kinects 400 labels

Download the id to label mapping for the Kinetics 400 dataset on which the torch hub models were trained. This will be used to get the category label names from the predicted class ids.

```
json_url = "https://dl.fbaipublicfiles.com/pyslowfast/dataset/class_names/kinetics_classnames.json"
json_filename = "kinetics_classnames.json"
try: urllib.urlopen().retrieve(json_url, json_filename)
except: urllib.request.urlretrieve(json_url, json_filename)
```

Kinects 400 labels

```
{
  "\sharpening knives\": 290, "\eating ice cream\": 115, "\cutting nails\": 81, "\changing wheel\": 53, "\bench pressing\": 19, "deadlifting": 88, "\eating carrots\": 111, "marching": 192, "\throwing discus\": 358, "\playing flute\": 231, "\cooking on campfire\": 72, "\breeding or breadcrumbing\": 33, "\playing badminton\": 218, "\ripping paper\": 276, "\playing saxophone\": 244, "\milking cow\": 197, "\juggling balls\": 169, "\flying kite\": 130, "capoeira": 43, "\making jewelry\": 187, "drinking": 100, "\playing cymbals\": 228, "\cleaning gutters\": 61, "\hurling (sport)\": 161, "\playing organ\": 239, "\tossing coin\": 361, "wrestling": 395, "\driving car\": 103, "headbutting": 150, "\gymnastics tumbling\": 147, "\making bed\": 186, "abseiling": 0, "\holding snake\": 155, "\rock climbing\": 278, "\cooking egg\": 71, "\long jump\": 182, "\bee keeping\": 17, "\trimming or shaving beard\": 365, "\cleaning shoes\": 63, "\dancing gangnam style\": 86, "\catching or throwing softball\": 50, "\ice skating\": 164, "jogging": 168, "\eating spaghetti\": 116, "bobsledding": 28, "\assembling computer\": 8, "\playing cricket\": 227, "\playing monopoly\": 238, "\golf putting\": 143, "\making pizza\": 188, "\javelin throw\": 166, "\peeling potatoes\": 211, "clapping": 57, "\brushing hair\": 36, "\flipping pancake\": 129,
```

Input format

```
side_size = 256
mean = [0.45, 0.45, 0.45]
std = [0.225, 0.225, 0.225]
crop_size = 256
num_frames = 32
sampling_rate = 2
frames_per_second = fps
slowfast_alpha = 4
num_clips = 10
num_crops = 3
```

```
# The duration of the input clip is also specific to the model.
clip_duration = (num_frames * sampling_rate)/frames_per_second
```

```
start_sec = 0
end_sec = start_sec + clip_duration
```


Human activity recognition

Introduction to Video Classification and Human Activity Recognition



Taha Anwar (BleedAI.com)

MARCH 8, 2021 — LEAVE A COMMENT

Deep Learning

Keras

Tensorflow

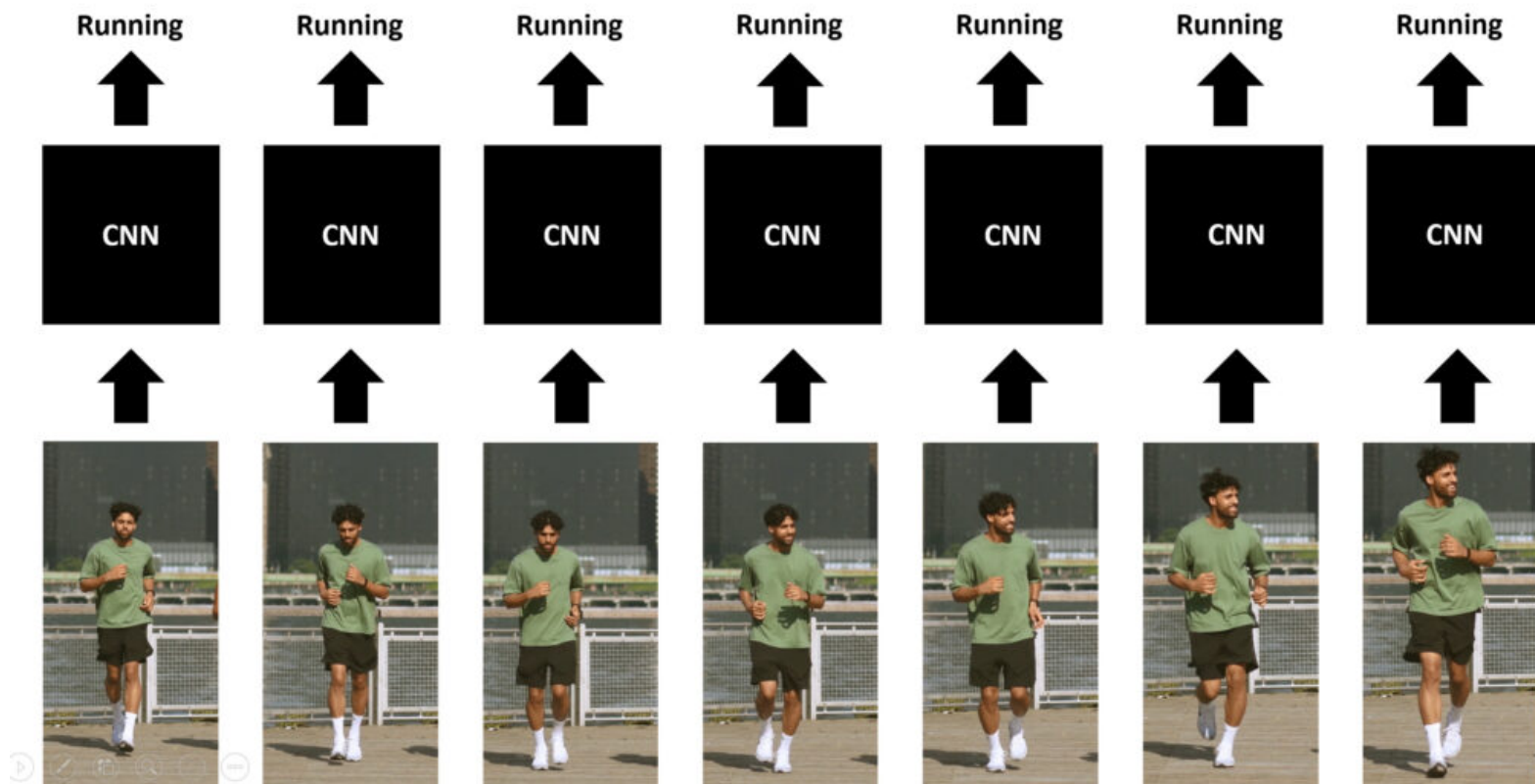
Theory

Video Analysis

<https://learnopencv.com/introduction-to-video-classification-and-human-activity-recognition/>

Single frame CNN

Videos generally contain a lot of frames, and we do not need to run a classification model on each frame, but only a few of them that are spread out throughout the entire video.



Can image classification be used to predict action?



Image classification model can predict each individual frame of the video when NN can use environmental context to predict actions.



Using environmental context instead of the actual action sequence to predict actions could cause problems and lead to over fitting.

Human activity recognition

You need a series of image frames to classify human activity correctly



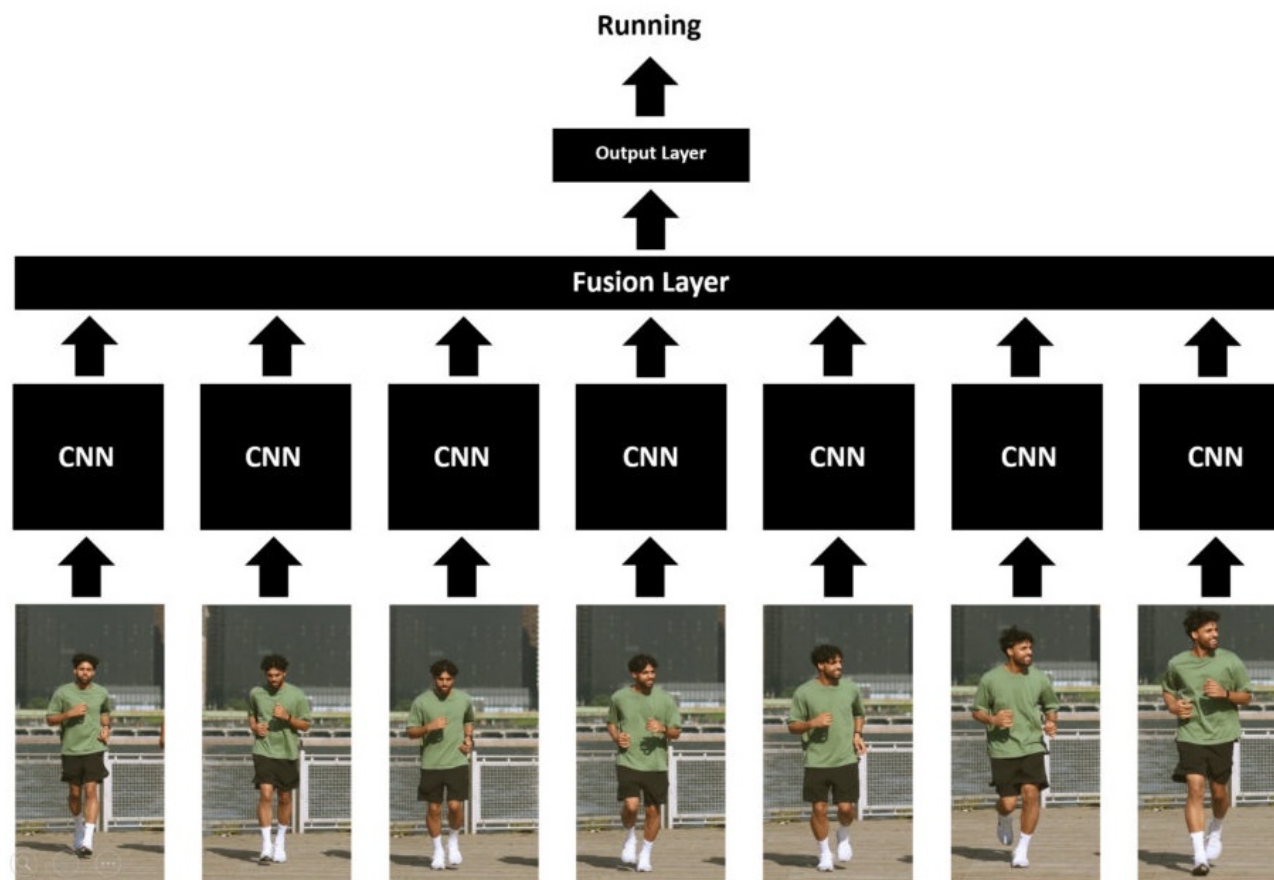
Human activity recognition

You need a series of image frames to classify human activity correctly



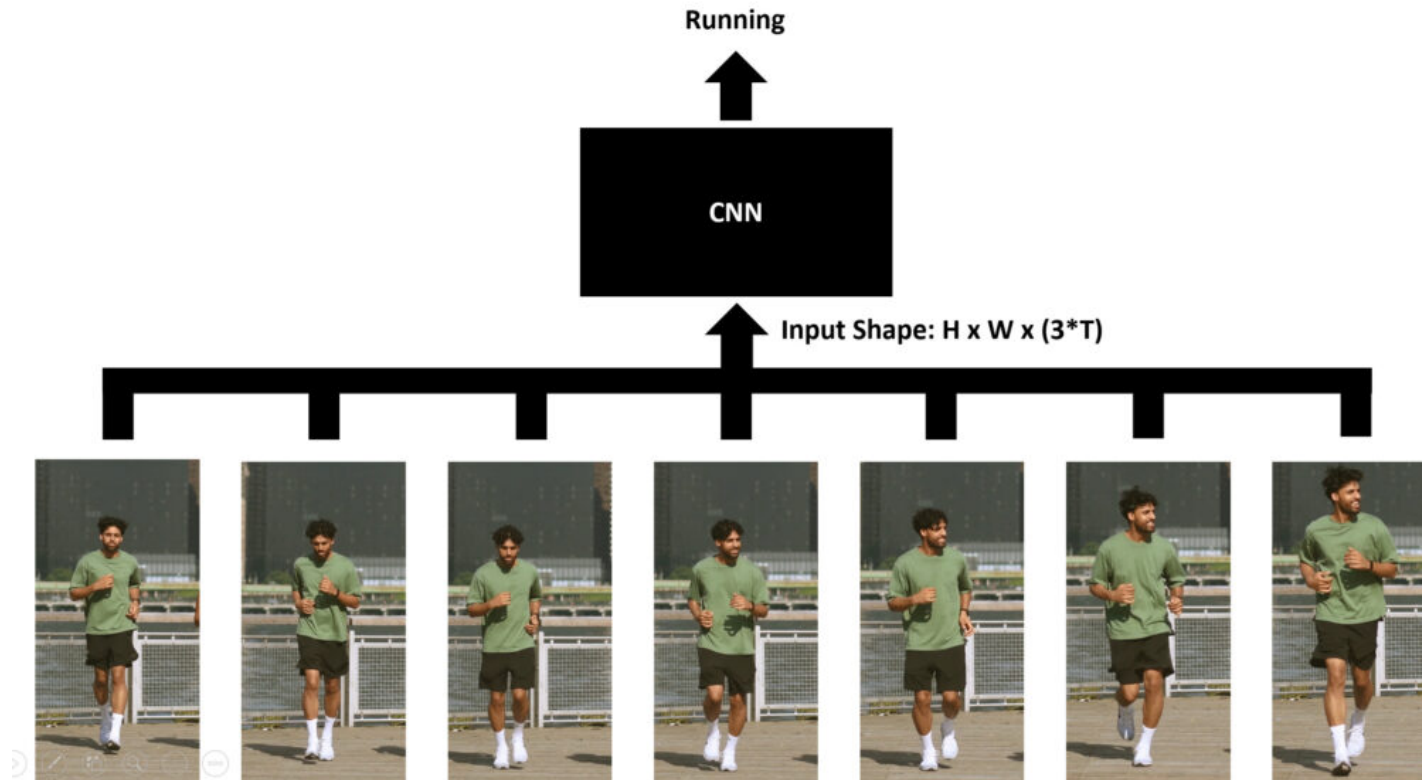
Late fusion

This approach enables the model to learn spatial as well as temporal information about the appearance and movement of the objects in a scene. The Fusion layer is normally implemented using the max pooling, average pooling or flattening technique.



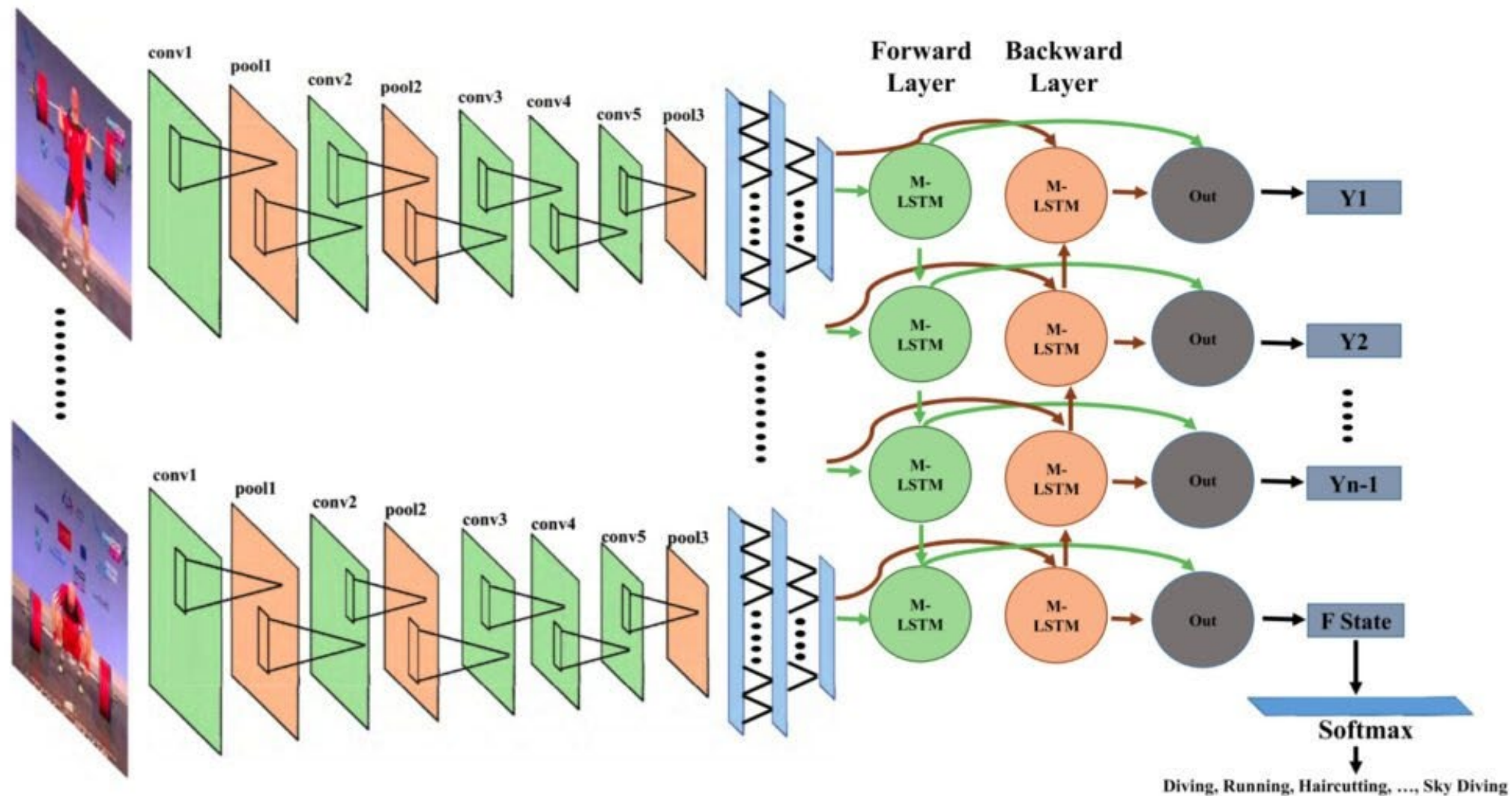
Early fusion

The temporal dimension and the channel (RGB) dimension of the video are fused at the start before passing it to the model which allows the first layer to operate over frames and learn to identify local pixel motions between adjacent frames.

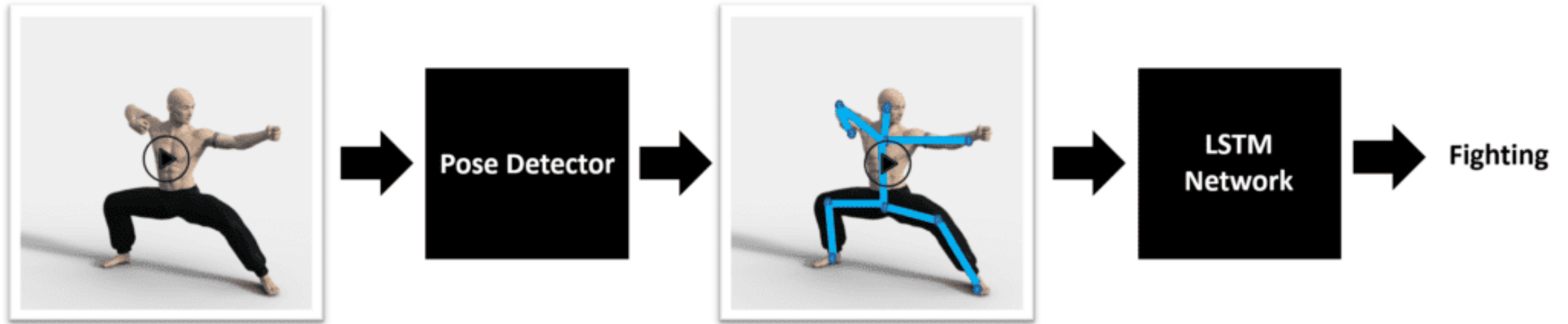


CNN + LSTM

22

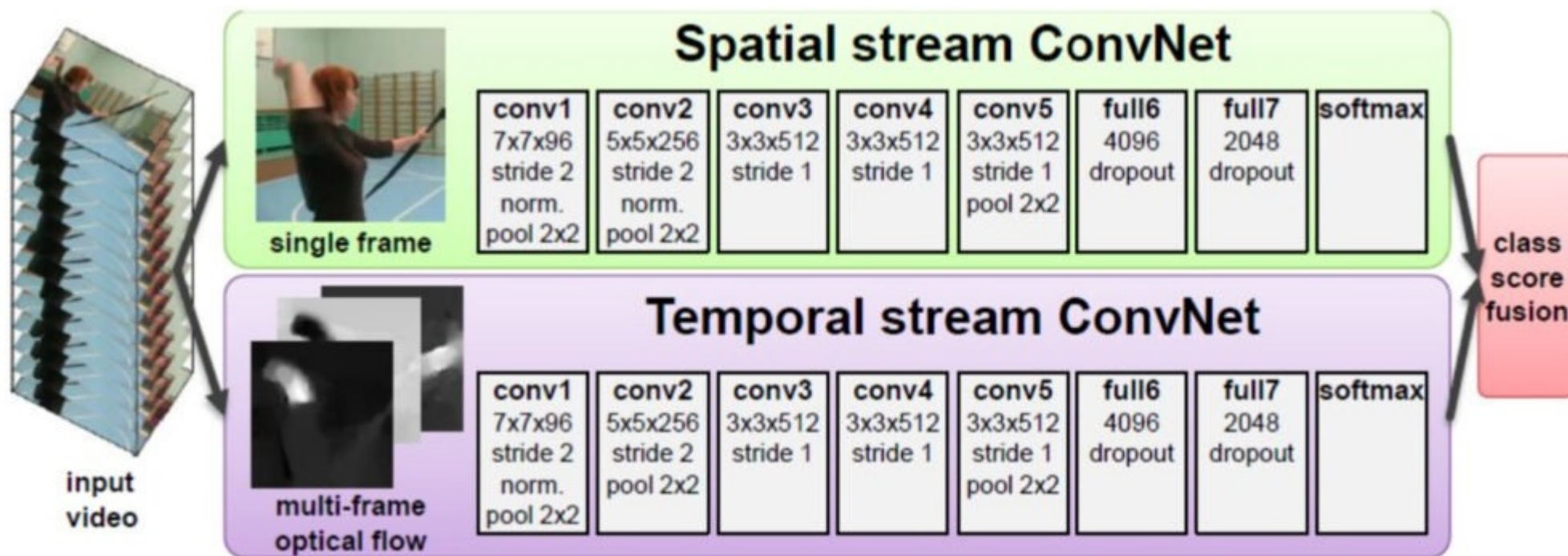


Pose detection + LSTM



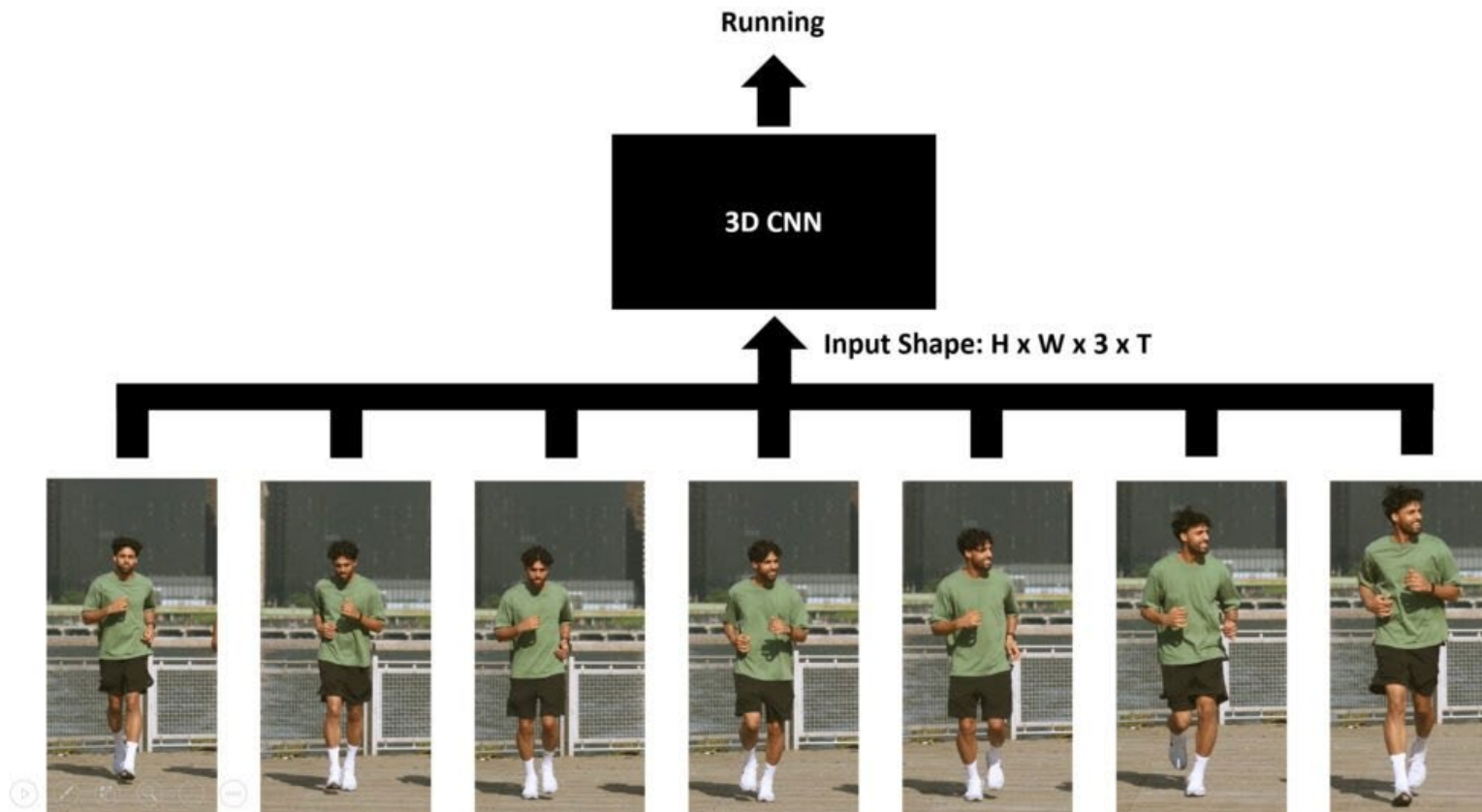
Optical flow + CNN

Optical flow is the pattern of visible motion of objects, edges and helps calculate the motion vector of every pixel in a video frame. The stream on the bottom called the Temporal stream takes every adjacent frame's optical flows after merging them using the early fusion technique and then using the motion information to make a prediction.



3D CNN slow fusion

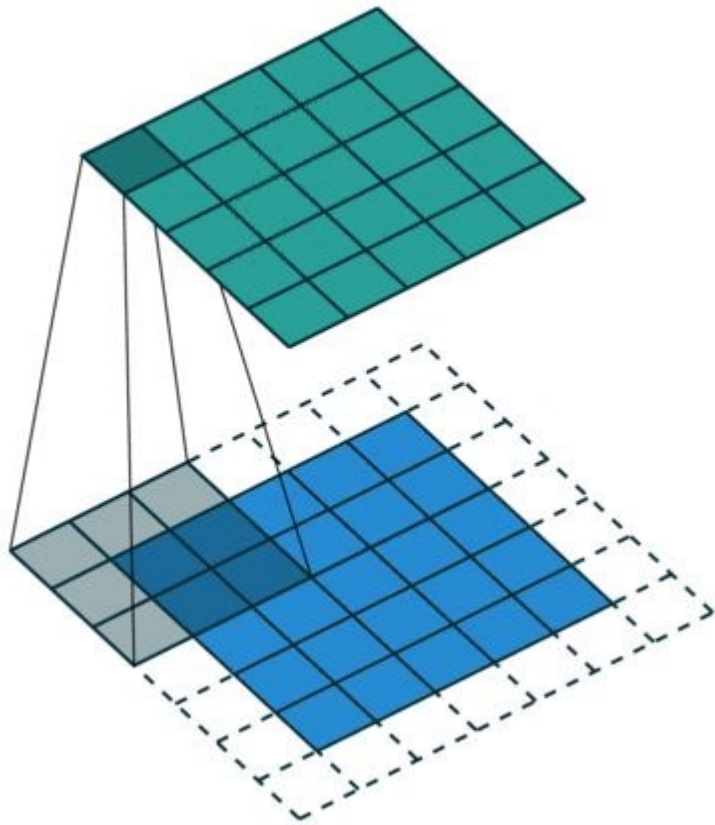
This approach uses a 3D convolution network that allows you to process temporal information and spatial by using a 3 Dimensional CNN. This method is also called the Slow Fusion approach. Unlike Early and Late fusion, this method fuses the temporal and spatial information slowly at each CNN layer throughout the entire network.



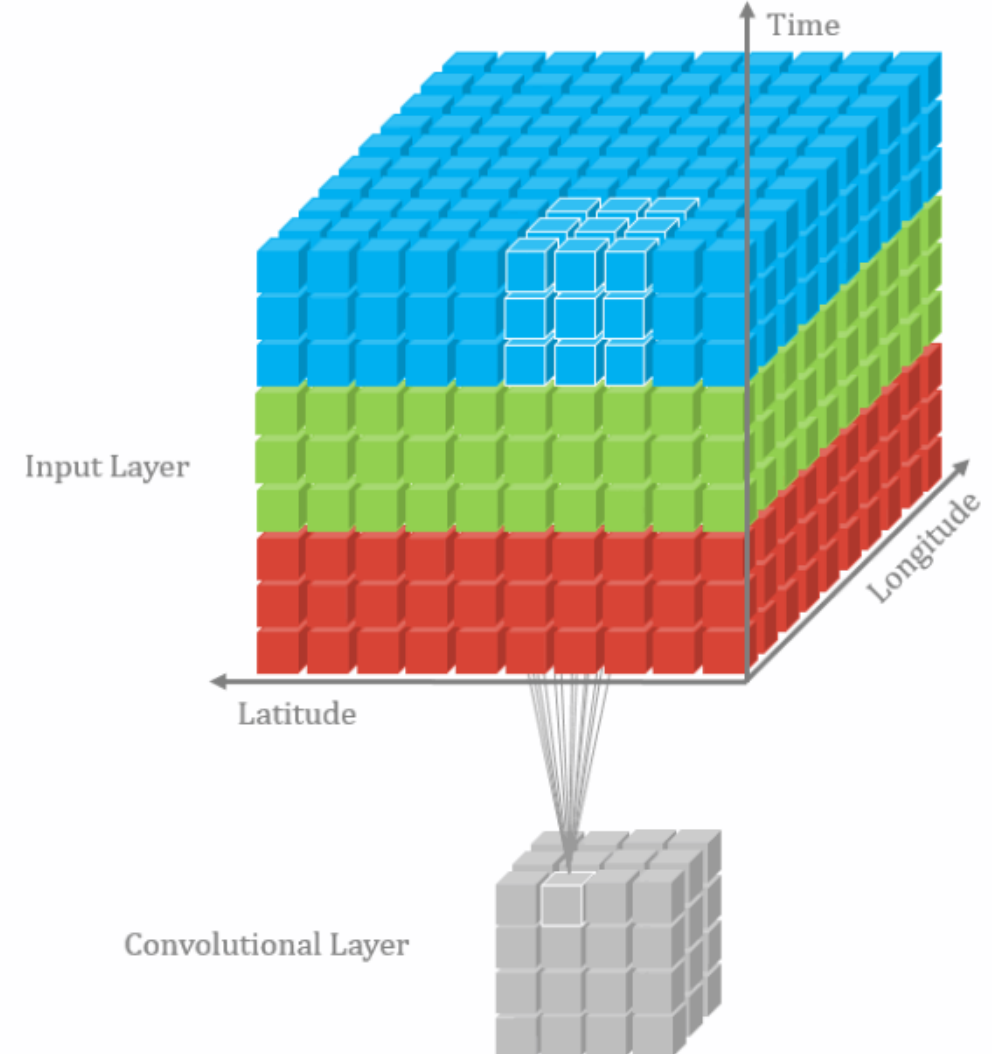
3D convolution

26

2D convolution

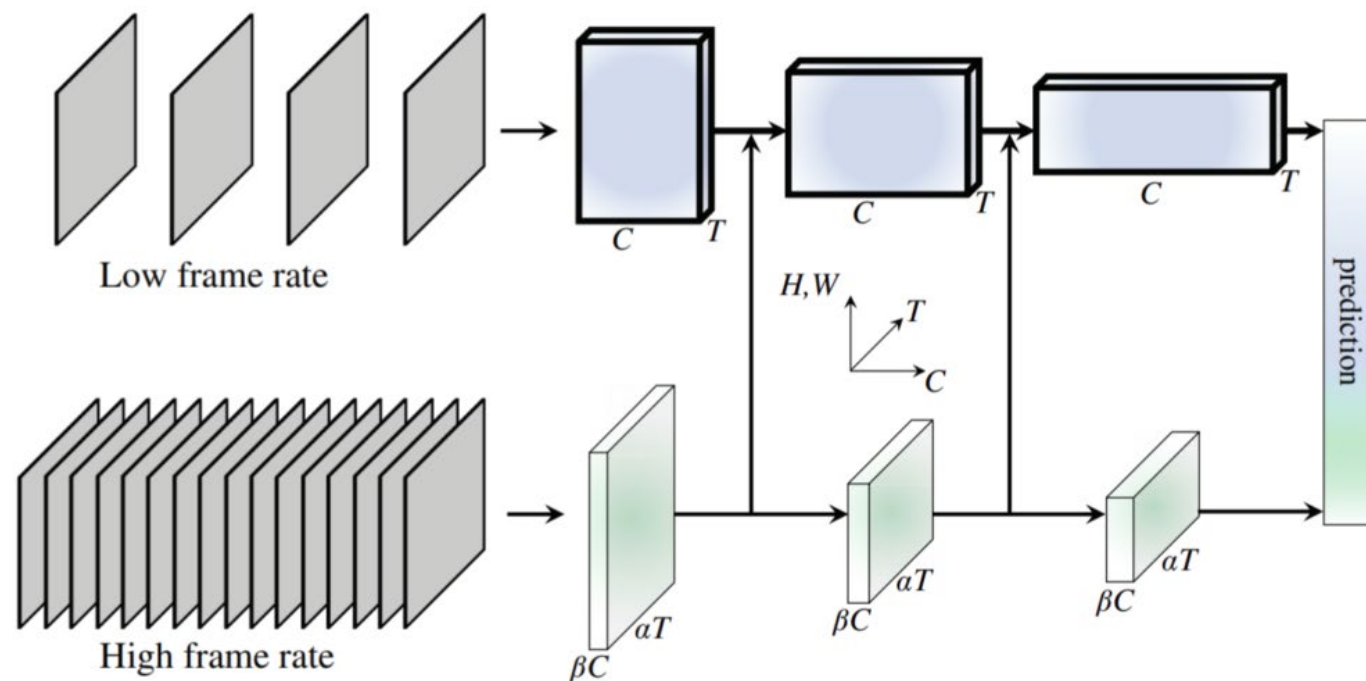


3D convolution



SlowFast

The stream on top, called the slow branch, operates on a low temporal frame rate video and has a lot of channels at every layer for detailed processing for each frame. On the other hand, the stream on the bottom, also known as the fast branch, has low channels and operates on a high temporal frame rate version of the same video.

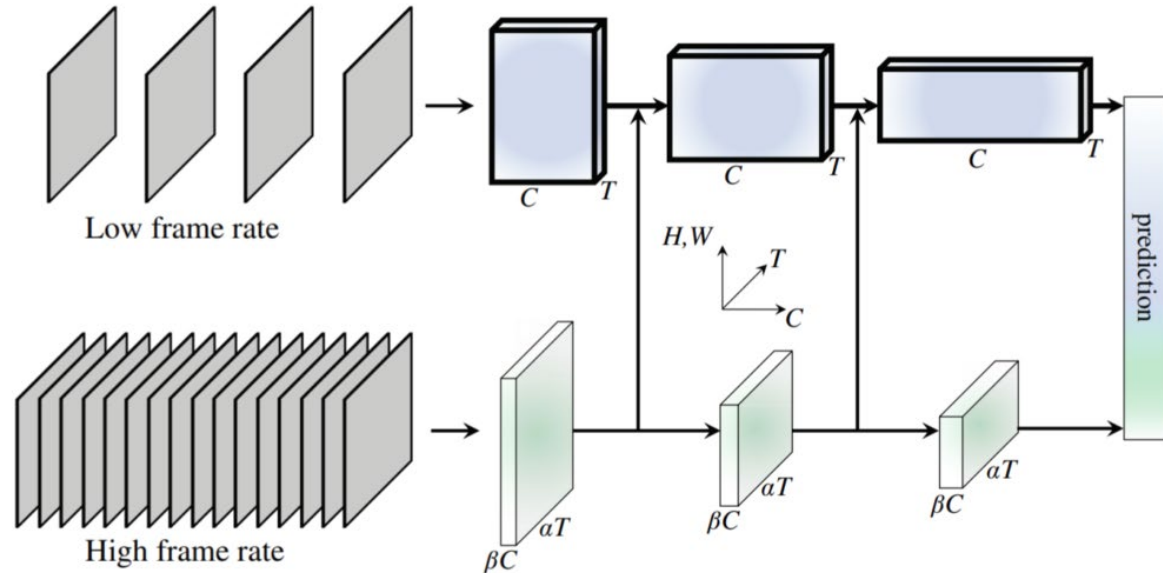


Select input clips

```
# The duration of the input clip is also specific to the model.  
clip_duration = (num_frames * sampling_rate)/frames_per_second
```

```
# Select the duration of the clip to load by specifying the start and end duration  
# The start_sec should correspond to where the action occurs in the video  
start_sec = 0  
end_sec = start_sec + clip_duration  
print(end_sec)
```

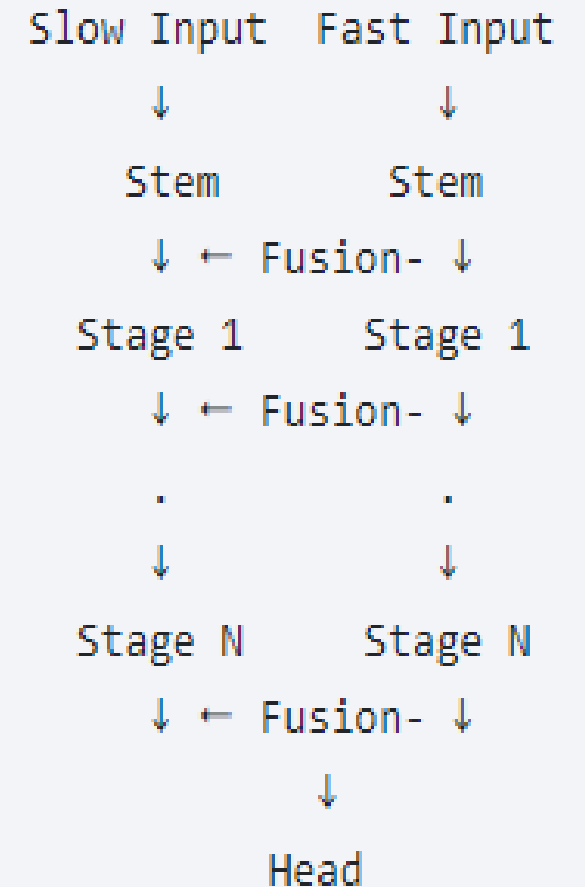
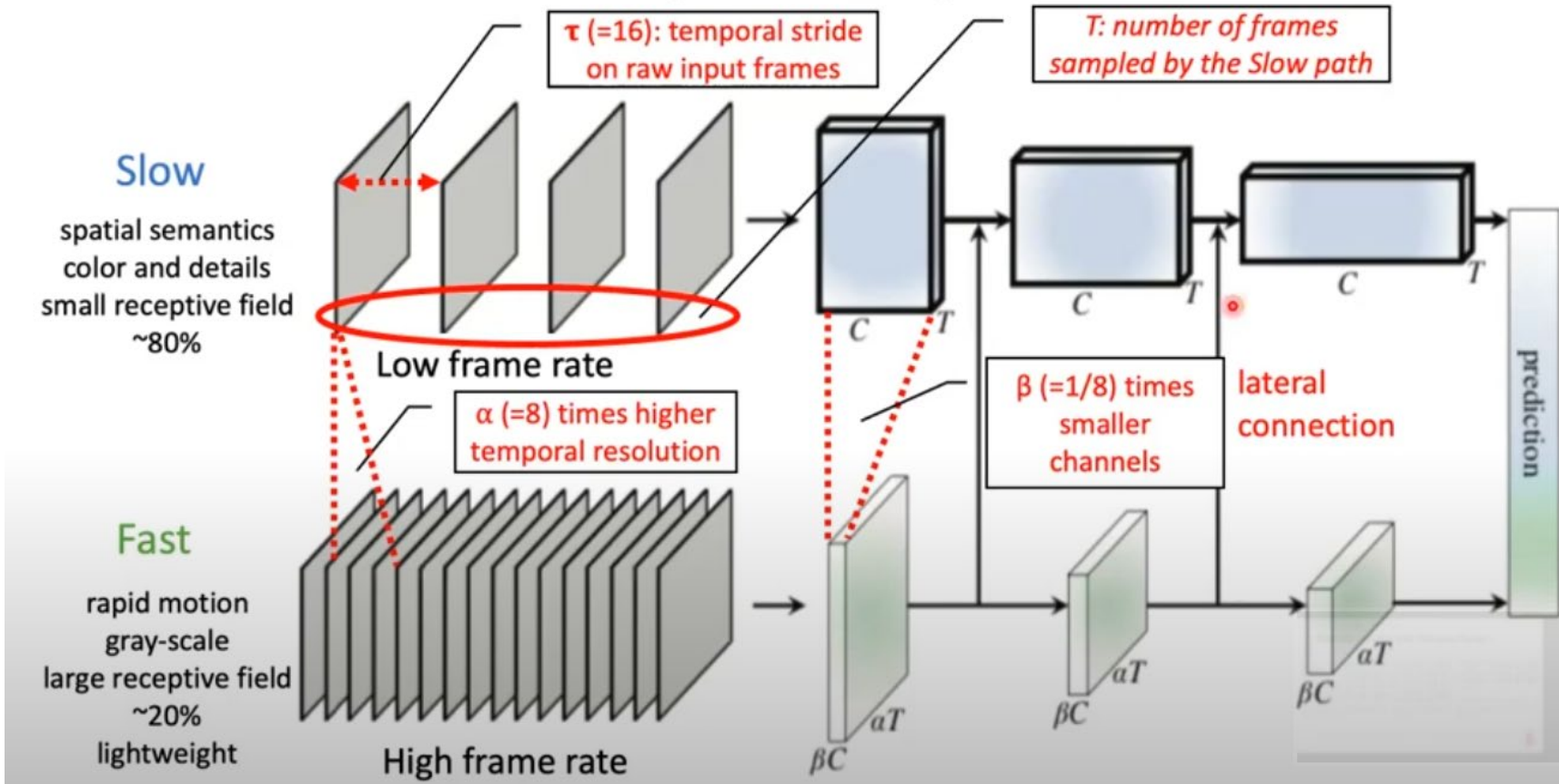
SlowFast – Two pathways



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

SlowFast – Two pathways

Basic Idea: Two pathways



<https://pytorchvideo.readthedocs.io/en/latest/api/models/slowfast.html>

<https://arxiv.org/pdf/1812.03982.pdf>

<https://www.youtube.com/watch?v=nOdnHhco39E>

Input transformation

```
class PackPathway(torch.nn.Module):
    """
    Transform for converting video frames as a list of tensors.
    """
    def __init__(self):
        super().__init__()

    def forward(self, frames: torch.Tensor):
        fast_pathway = frames
        # Perform temporal sampling from the fast pathway.
        slow_pathway = torch.index_select(
            frames,
            1,
            torch.linspace(
                0, frames.shape[1] - 1, frames.shape[1] // slowfast_alpha
            ).long(),
        )
        frame_list = [slow_pathway, fast_pathway]
        return frame_list
```

Input transformation

```
transform = ApplyTransformToKey(  
    key="video",  
    transform=Compose(  
        [  
            UniformTemporalSubsample(num_frames),  
            Lambda(lambda x: x/255.0),  
            NormalizeVideo(mean, std),  
            ShortSideScale(  
                size=side_size  
            ),  
            CenterCropVideo(crop_size),  
            PackPathway()  
        ]  
    ),  
)
```