ILLINOIS INSTITUTE OF TECHNOLOGY

ECASP

Vision-Based Gesture-Controlled Drone: A Comprehensive Python Package to Deploy Embedded Pose Recognition Systems

Summer 2021 Research Project – Final Report

August 11, 2021

Arthur FINDELAIR

Advisor: Dr. Jafar Saniie

Embedded Computing and Signal Processing Research Laboratory

Departement of Electrical and Computer Engineering

Illinois Institute of Technology, Chicago, Illinois, USA

Objective of the project

Develop a **vision-based gesture control** system on an autonomous drone using industry standards to facilitate **portability and extensibility**.

Source: The astounding athletic power of quadcopters, Raffaello D'Andrea, TEDGlobal, 2013

Other applications of a gesture control system

Continuous tracking

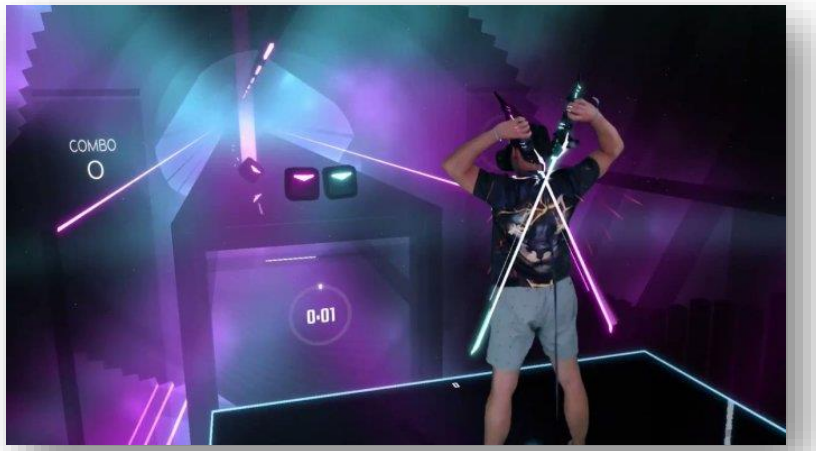
Augmented reality
VFX animations
Robotic

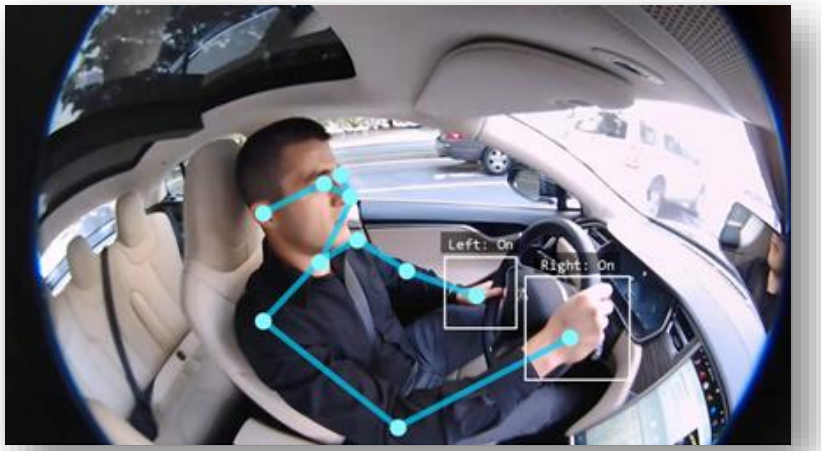
Gesture recognition

Seniors & patients monitoring
Body language interpretation
Basic HMI

Hybrid

Advanced
HMI





Source: MIT Advanced Vehicle Technology Study: Large-Scale Naturalistic Driving Study of Driver Behavior and Interaction with Automation – arXiv:1711.06976

Table of content

- Motivation & Overview of the project
- Hardware components
- Pose estimation
- Pose classification
- Deployment
- Demonstration
- Future works

Motivation of the project – Current uses of drones

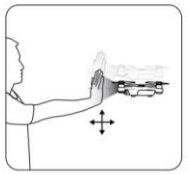
APPLIED DRONE-BASED METHODS PER INDUSTRY

Industry Vertical	Mapping & Surveying	Inspection	Photography & Filming	Localization & Detection	Other[1]	Delivery	Spraying & Seeding
Energy (Utilities)	14%	83%	0%	1%	1%	0%	0%
Construction	80%	15%	0%	0%	2%	2%	0%
Transportation and Warehousing	48%	34%	0%	6%	5%	7%	0%
Agriculture	59%	3%	0%	8%	0%	0%	30%
Mining, Quarrying, and Oil & Gas Extraction	55%	38%	0%	0%	6%	0%	2%
Public Administration	28%	5%	15%	32%	14%	0%	6%
Information	3%	6%	74%	0%	16%	0%	0%
Real Estate, Rental & Leasing, and Industrial Plants	16%	67%	17%	0%	0%	0%	0%
Arts, Entertainment, and Recreation	5%	0%	60%	0%	30%	5%	0%
Insurance	46%	10%	22%	14%	8%	0%	0%
Health Care and Social Assistance	35%	0%	2%	17%	8%	38%	0%
Professional, Scientific, and Technical Services	51%	6%	2%	15%	24%	2%	0%
Safety & Security	11%	0%	0%	69%	20%	0%	0%
Educational Services	53%	18%	0%	12%	17%	0%	0%
Waste Management & Remediation Services	34%	14%	5%	17%	30%	0%	0%

Source: 237 ways drone applications revolutionize business, Ed Alvarado, May 3, 2021

Motivation of the project

– Commercial solutions

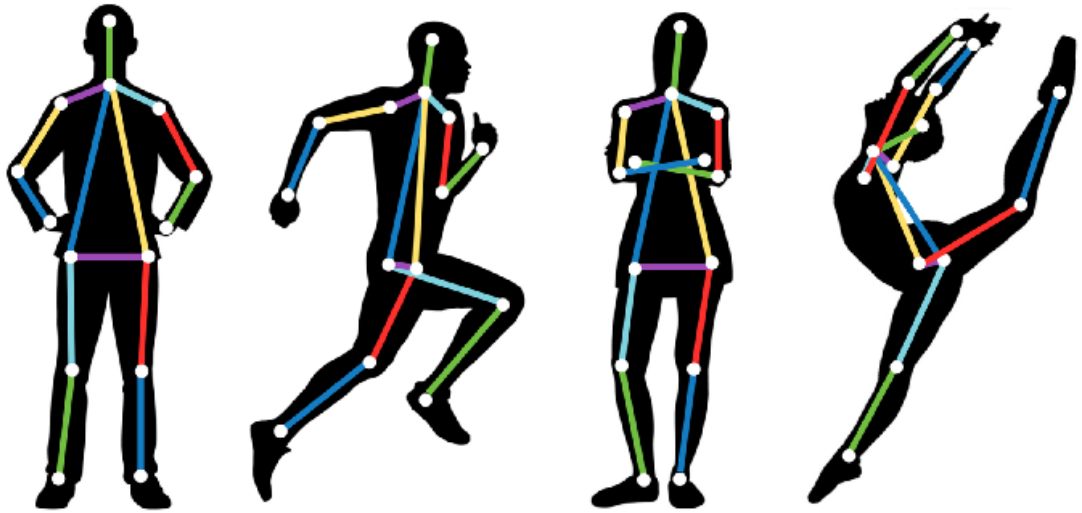
Adjusting Position		
1. Move your palm up or down slowly to control the aircraft's altitude while maintaining a constant distance between your palm and the aircraft. 2. Keep your palm at a constant distance from your body (as shown), then move your arm left or right to control the aircraft's orientation. 3. Keep your palm at a constant distance from your body, move forward or backward to fly forward or backward.	G —— Solid green	
Follow		
1. Stand in front of the aircraft then raise one of your arms and quickly wave your hand at the camera. Position your palm about 0.7 m in front of the aircraft's nose for about two seconds. 2. The front LEDs will blink green twice if the gesture is recognized successfully. The aircraft will ascend and fly backward, then hover in place 3 m from where you're standing, 2.3 m above ground. Do not move your body until the aircraft hovers in place.	G ×2 ····· Blinks green twice	
3. The front LEDs will glow solid green and the aircraft will start tracking automatically. 4. If the wave gesture fails or the aircraft loses track of you, you can activate Follow by raising both arms above your head in a Y shape, then holding for two seconds.	G —— Solid green	
Taking Selfies		
1. Make a frame with your hands within 23 ft (7 m) of the aircraft while facing toward the camera and it will begin taking a selfie. 2. Selfie gesture has been recognized successfully if the Front LEDs blink red slowly. Wait for the selfie count down for three seconds. The Front LEDs will blink red quickly, indicating the camera is about to take a selfie.	R ····· Blinks red slowly	

Spark Gesture Controls

DJI Motion Controller

Overview of the project – The pipeline

Existing tools



Added value

Overview of the project – The deployment platform

DJI Mavic Air 2

Ryze Tello

Holybro X500

Abstraction

Flexibility

Skydio X2

Parrot Anafi

Hardware – The deployment platform

Kit Contents

- Pixhawk4 Flight controller
- Power Management PM07
- Pixhawk4 GPS
- Motors – 2216 KV880
- BLHeli S ESC 20A
- Propeller 1045
- 915MHz Telemetry Radio
- Power and Radio Cables
- Battery Straps

Characteristics

- Dimensions: 410*410*300mm
- Wheelbase: 500mm
- Weight: 978g
- Max payload: 2.1kg (at 14.8V)

Additional parts

- 6000 mAh, 4 cells LiPo battery
- FrSky X8R 2.4GHz receiver
- Taranis X9D+ controller

Hardware
– The deployment platform

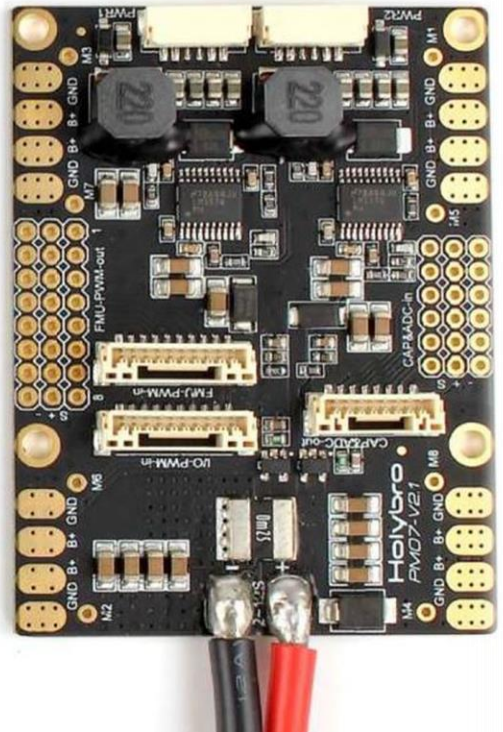
- **Main FMU processor:** STM32F765
- **IO Processor:** STM32F100
- **Main sensors:**
  ICM-20689 (IMU), BMI055 (IMU), IST8310 (Mag), MS5611 (Baro)
- **GNSS module sensors:**
  Ublox Neo-M8N (GPS/GLONASS receiver), IST8310 (Mag)

Transceiver Telemetry

GNSS module

Pixhawk 4 (FMUv5)

Power distribution board



Hardware

– The deployment platform

Nvidia Jetson Nano Development Kit B01
& ArduCam IMX477 Jetson compact camera

SOFTWARE MODULES

Depth Estimation	Object Detection	Pose Estimation	Gesture Recognition	Path Planning	Autonomous Navigation	Ecosystem Modules

JETPACK SDK

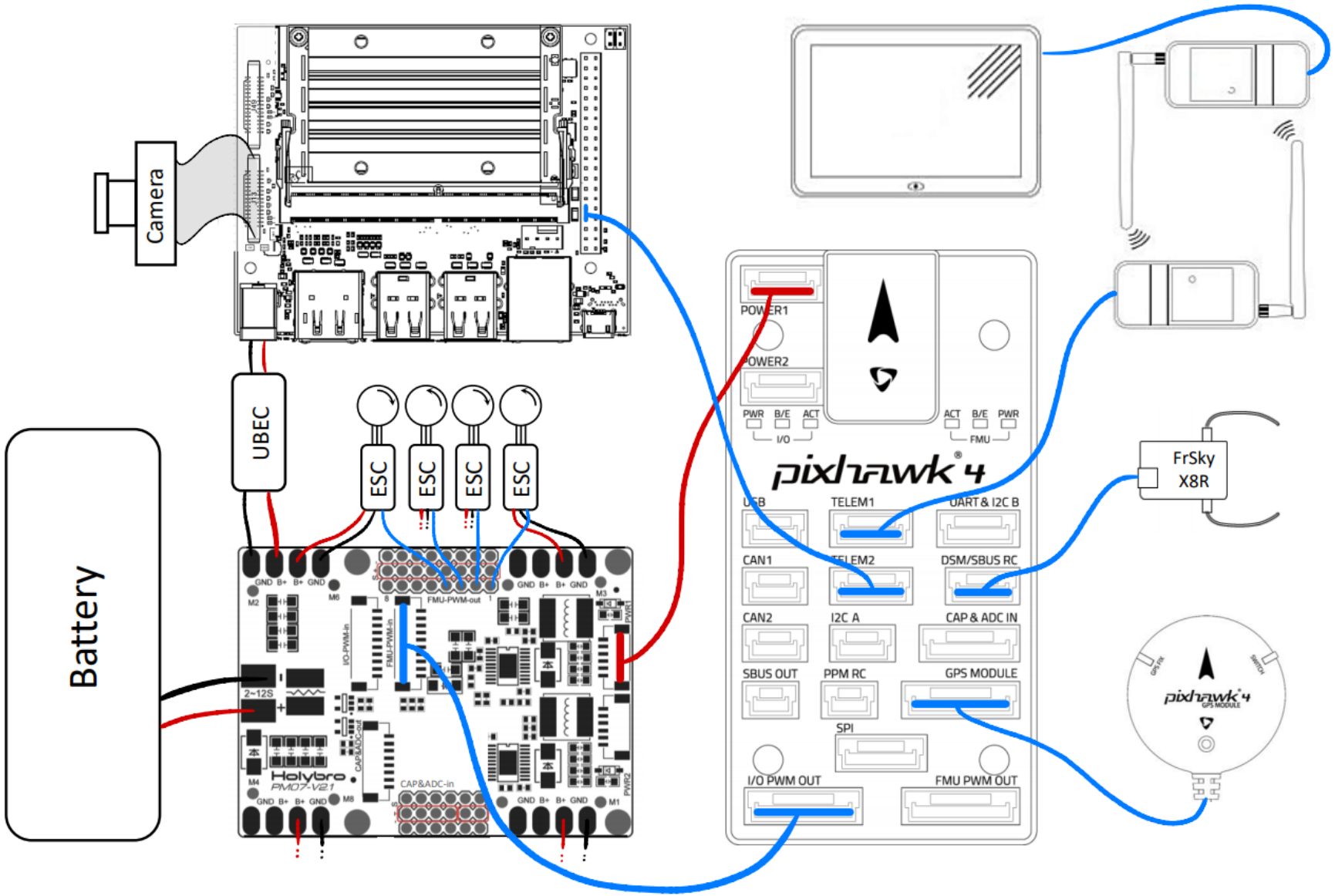
Deep Learning	Computer Vision	Accelerated Computing	Graphics	Multimedia	Sensors
TensorRT cuDNN	VisionWorks OpenCV	cuBLAS cuFFT	Vulkan OpenGL	libargus Video API	Drivers Ecosystem

CUDA | CUDA-X AI | LINUX FOR TEGRA | RTOS

JETSON COMPUTER

	Jetson Nano
AI Performance	472 GFLOPS
GPU	128-core NVIDIA Maxwell™ GPU
CPU	Quad-core ARM® Cortex®-A57 MPCore processor
Memory	4 GB 64-bit LPDDR4 25.6GB/s
Storage	16 GB eMMC 5.1
Power	5W | 10W

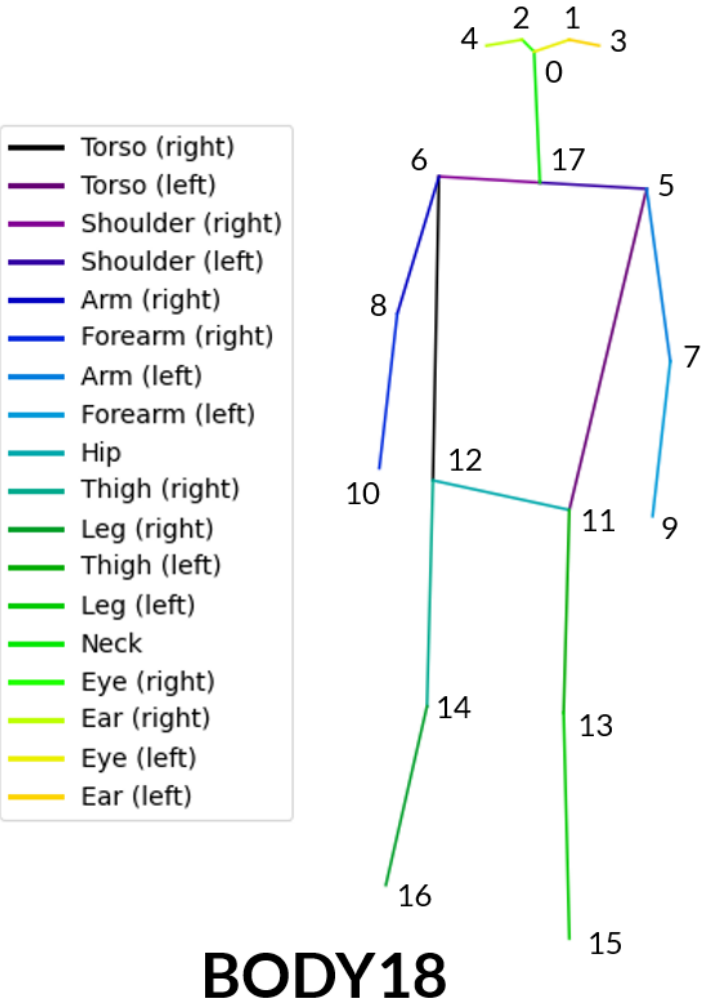
Hardware – Connections



Pose estimation – What is it?

Microsoft COCO 2020 Keypoint Detection Task

(250,000 person instances labeled)



Source: Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. CoRR, abs/1405.0312, 2014.

Pose estimation

– The Bottom-Up approach

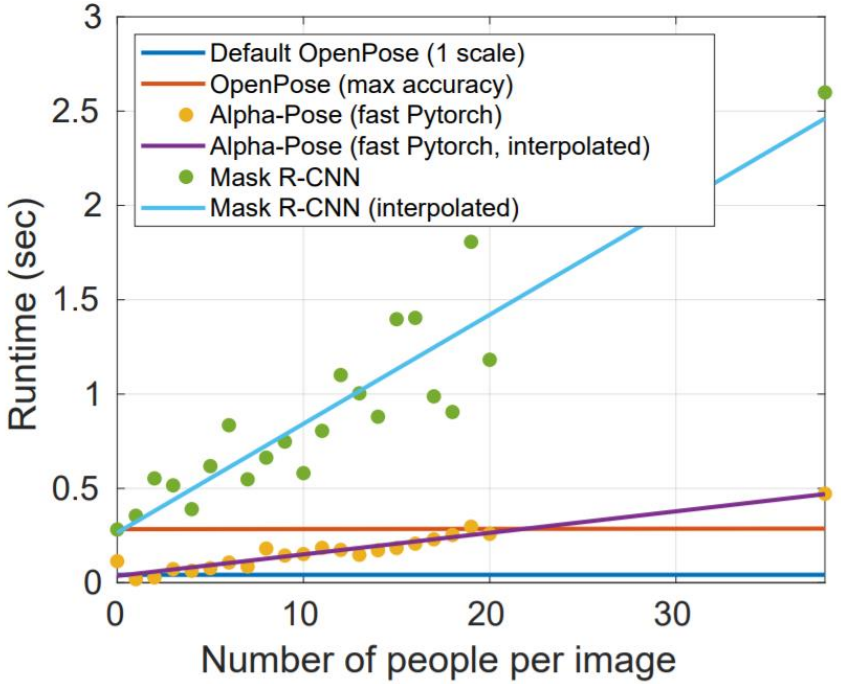
Source: Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2017. (arXiv:1611.08050)

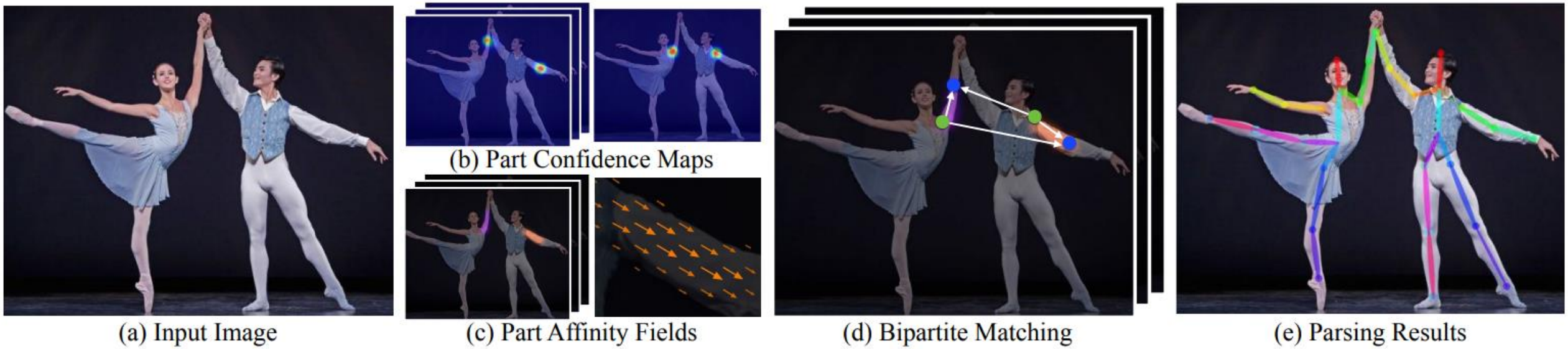
Part Confidence Maps – CMap

for body joints detection

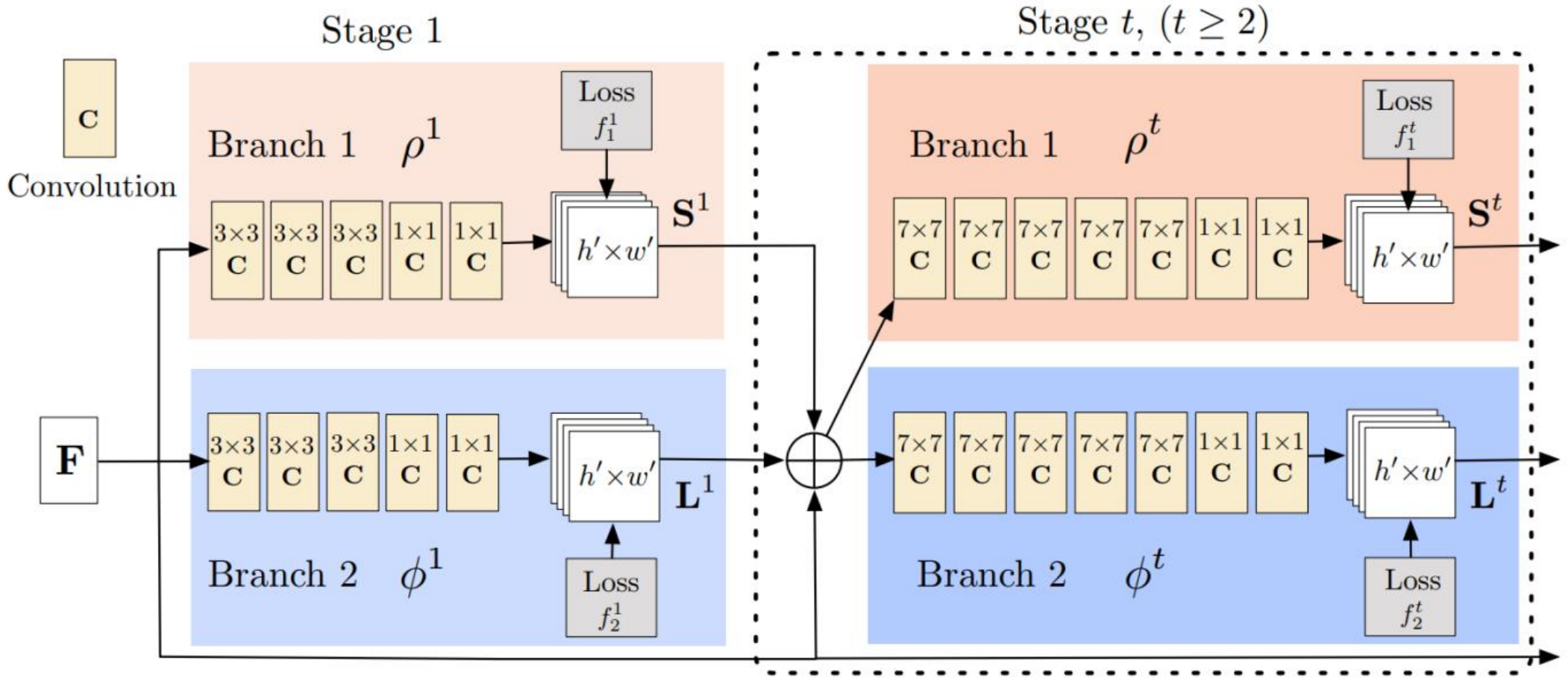
Part Affinity Fields – PAF

for body joints association



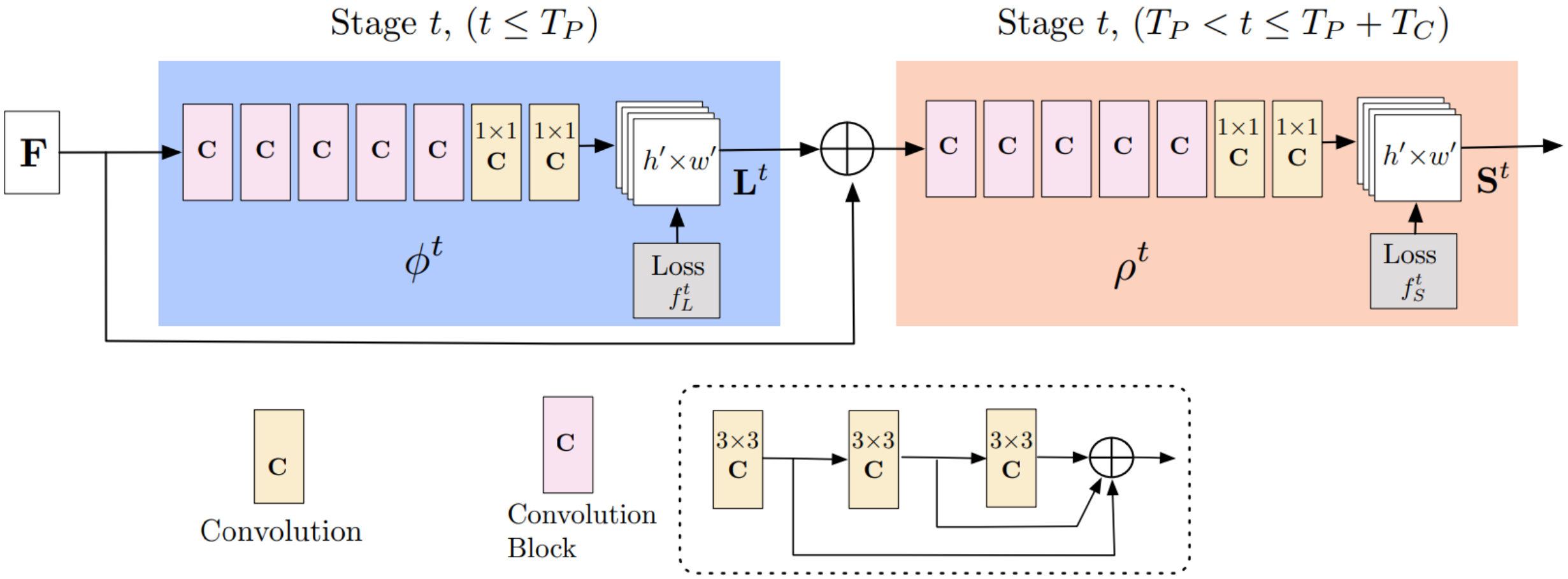


Pose estimation – Original architecture



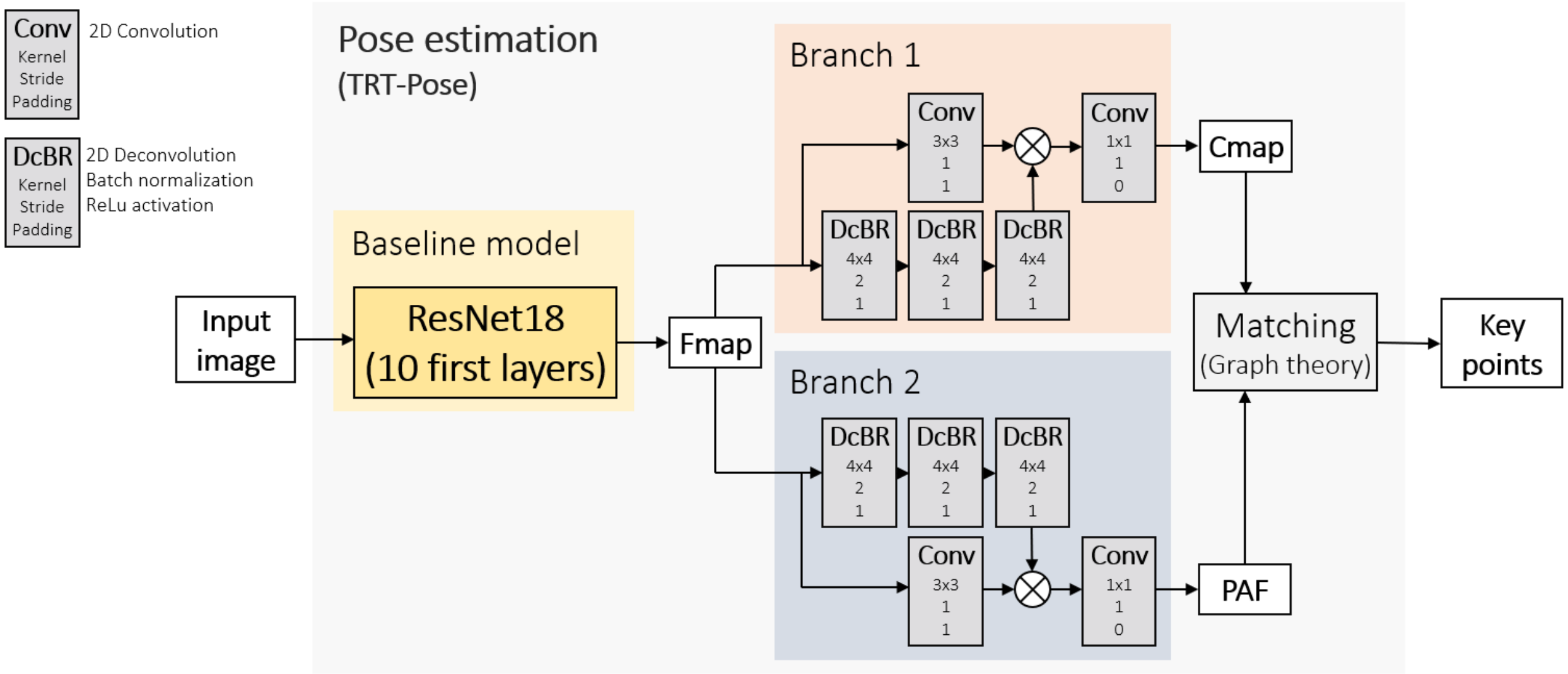
Source: Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields, 2017. (arXiv:1611.08050)

Pose estimation – OpenPose architecture



Sources: Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2018. (arXiv:1812.08008)

Pose estimation – TensorRT-Pose architecture



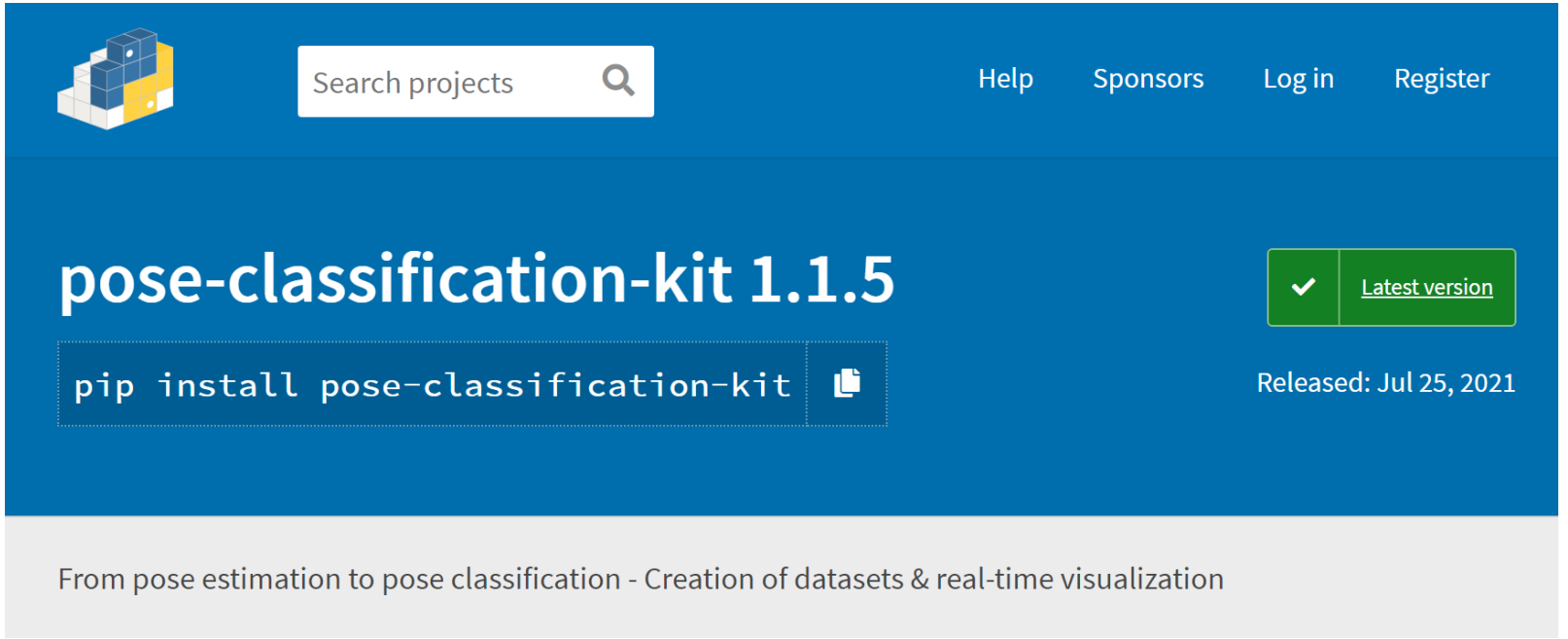
Pose estimation – TensorRT-Pose source code

```python
class UpsampleCBR(torch.nn.Sequential):
    def __init__(self, input_channels, output_channels):
        layers = []
        for i in range(3):
            if i == 0:
                inch = input_channels
            else:
                inch = output_channels
            layers += [
                torch.nn.ConvTranspose2d(inch, output_channels,
                                         kernel_size=4, stride=2, padding=1),
                torch.nn.BatchNorm2d(output_channels),
                torch.nn.ReLU()
            ]
        super(UpsampleCBR, self).__init__(*layers)

class CmapPafHeadAttention(torch.nn.Module):
    def __init__(self, input_channels, cmap_channels, paf_channels, upsample_channels=256):
        super(CmapPafHeadAttention, self).__init__()
        self.cmap_up = UpsampleCBR(input_channels, upsample_channels)
        self.paf_up = UpsampleCBR(input_channels, upsample_channels)
        self.cmap_att = torch.nn.Conv2d(upsample_channels, upsample_channels,
                                        kernel_size=3, stride=1, padding=1)
        self.paf_att = torch.nn.Conv2d(upsample_channels, upsample_channels,
                                       kernel_size=3, stride=1, padding=1)

        self.cmap_conv = torch.nn.Conv2d(upsample_channels, cmap_channels,
                                         kernel_size=1, stride=1, padding=0)
        self.paf_conv = torch.nn.Conv2d(upsample_channels, paf_channels,
                                        kernel_size=1, stride=1, padding=0)

    def forward(self, x):
        xc = self.cmap_up(x)
        ac = torch.sigmoid(self.cmap_att(xc))

        xp = self.paf_up(x)
        ap = torch.tanh(self.paf_att(xp))

        return self.cmap_conv(xc * ac), self.paf_conv(xp * ap)
```

```python
class ResNetBackbone(torch.nn.Module):

    def __init__(self, resnet):
        super(ResNetBackbone, self).__init__()
        self.resnet = resnet

    def forward(self, x):
        x = self.resnet.conv1(x)
        x = self.resnet.bn1(x)
        x = self.resnet.relu(x)
        x = self.resnet.maxpool(x)
        x = self.resnet.layer1(x) # /4
        x = self.resnet.layer2(x) # /8
        x = self.resnet.layer3(x) # /16
        x = self.resnet.layer4(x) # /32
        return x

def resnet18_baseline_att(cmap_channels, paf_channels, upsample_channels=256, pretrained=True):
    resnet = torchvision.models.resnet18(pretrained=pretrained)
    return _resnet_pose_att(cmap_channels, paf_channels, upsample_channels, resnet, 512)
```

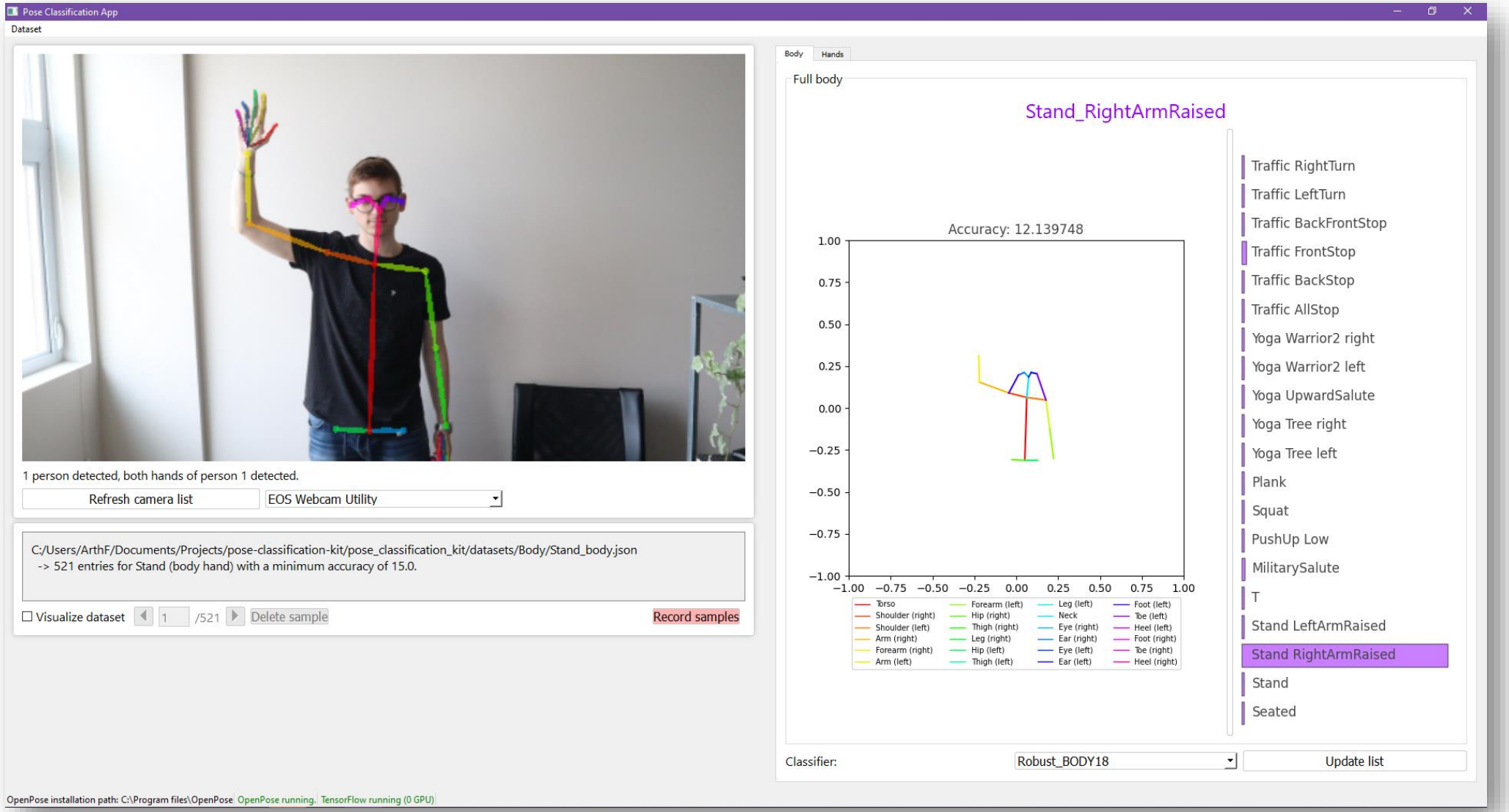
Pose classification

– Pose-Classification-Kit (PCK) package

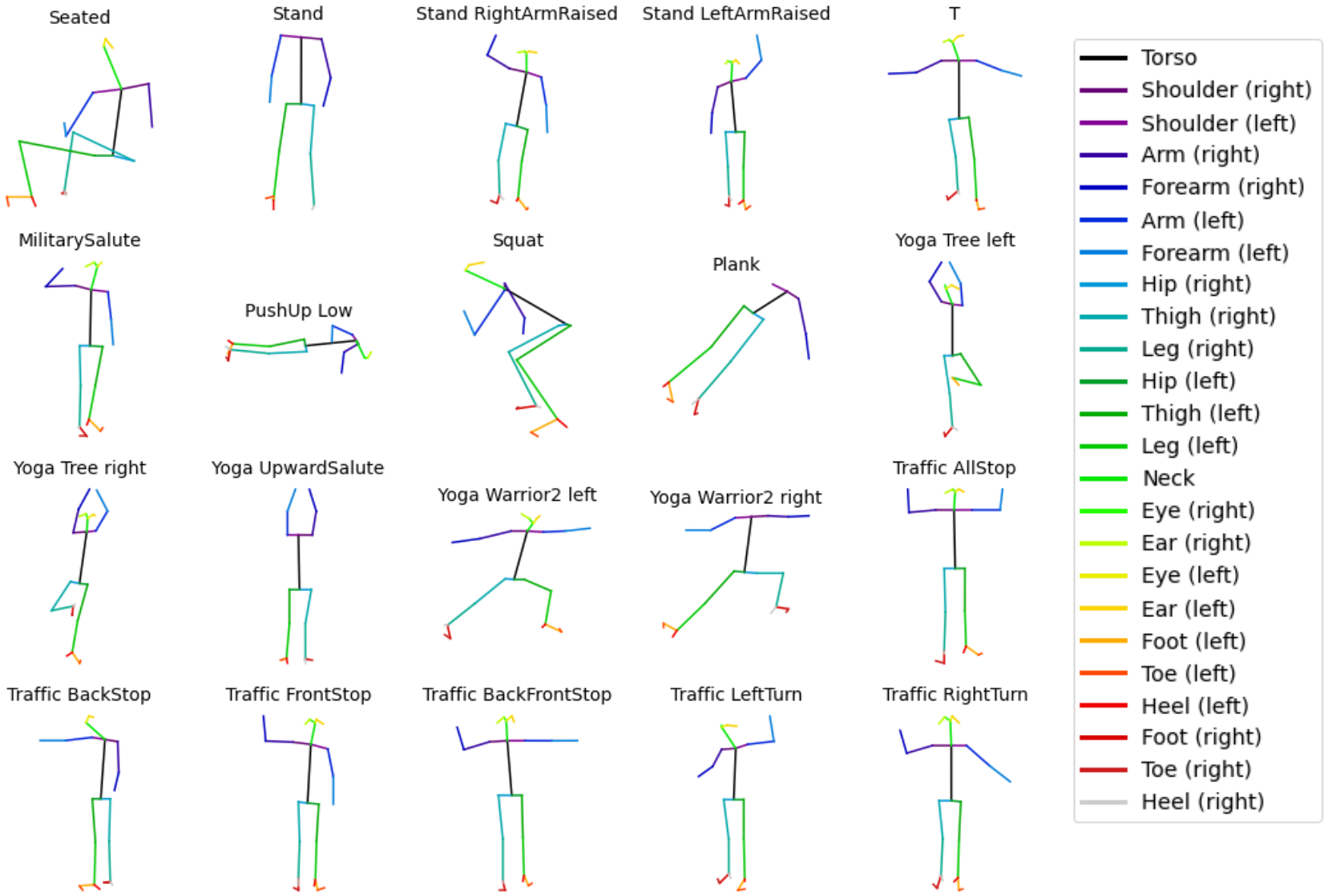


- Dataset creation tool
- Two datasets for body and hand gesture classification
- Several pre-trained models
- Utility functions for data augmentation and video analysis
- Example Jupyter Notebooks to ease DL model creation

Pose classification – PCK Application



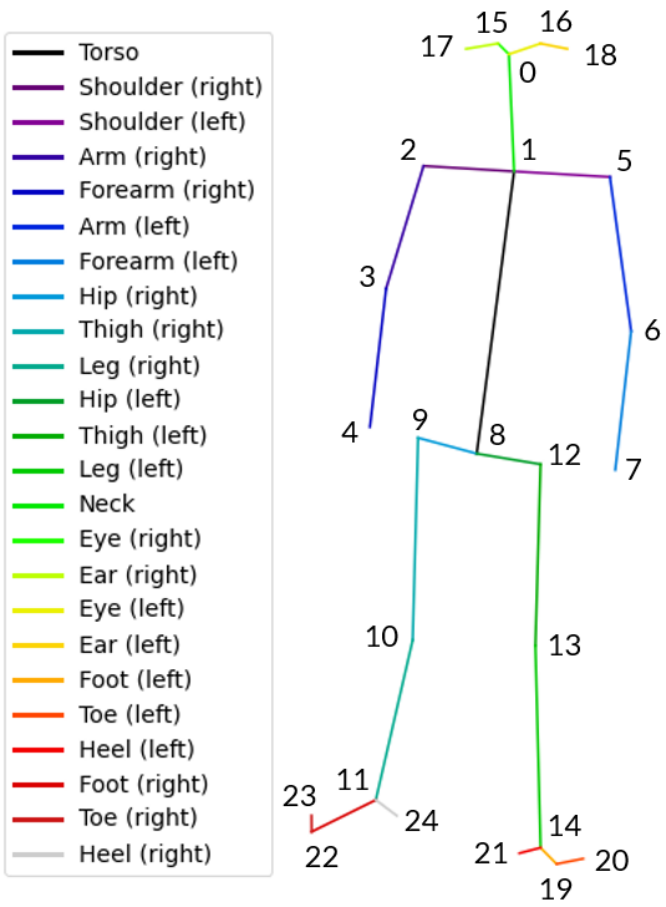
Pose classification – Body dataset



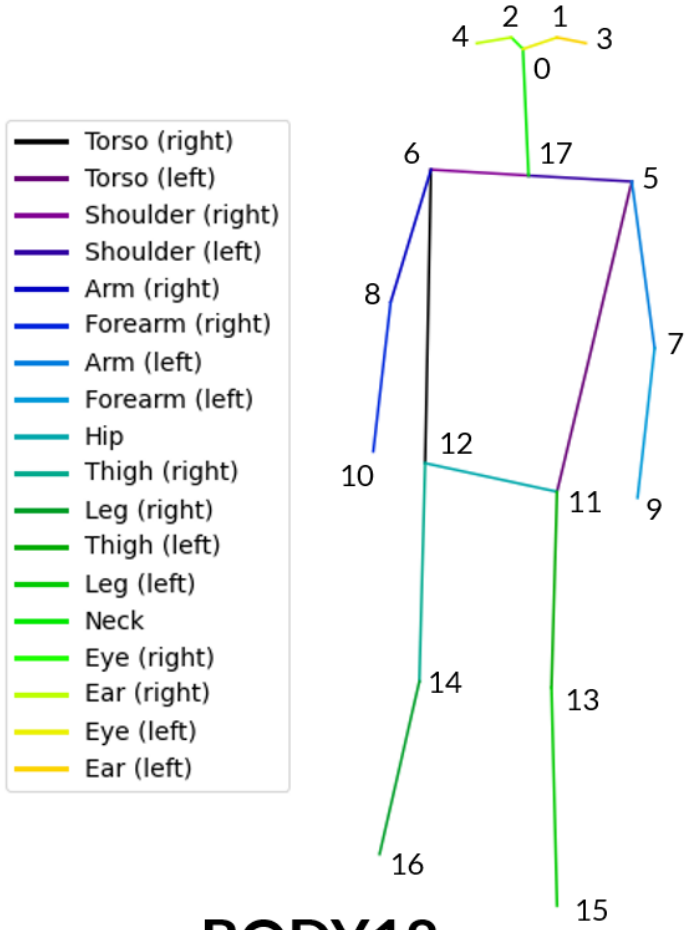
Number of samples per class

Yoga_Warrior2_left	521
Yoga_Warrior2_right	515
Yoga_Tree_left	580
Yoga_Tree_right	502
Traffic_AllStop	571
Traffic_BackStop	560
Traffic_FrontStop	528
Traffic_BackFrontStop	516
Traffic_LeftTurn	555
Traffic_RightTurn	521
Seated	606
Stand	521
Stand_RightArmRaised	503
Stand_LeftArmRaised	534
MilitarySalute	524
T	503
PushUp_Low	508
Squat	552
Plank	535
Yoga_UpwardSalute	526

Pose classification – Body models & Normalization



BODY25



BODY18

Consider a sample $\boldsymbol{x}$ of format BODY$\{I\}$, $\quad \forall i \in [\![0, I-1]\!]$:

$$f(\overrightarrow{x_i}) = \frac{\left(\overrightarrow{x_i} - \overrightarrow{\langle \boldsymbol{x} \rangle}\right)}{\max_{j \in \Delta} \delta_j \cdot \gamma_j}$$

Where δ_j is the length and γ_j the scaling factor of limb j

$$\Delta = \left\{ \begin{matrix} Torso, Neck, Shoulders, \\ Thigh\ (right), Thigh\ (left), \\ Leg\ (right), Leg\ (left) \end{matrix} \right\}$$

Keypoints normalization

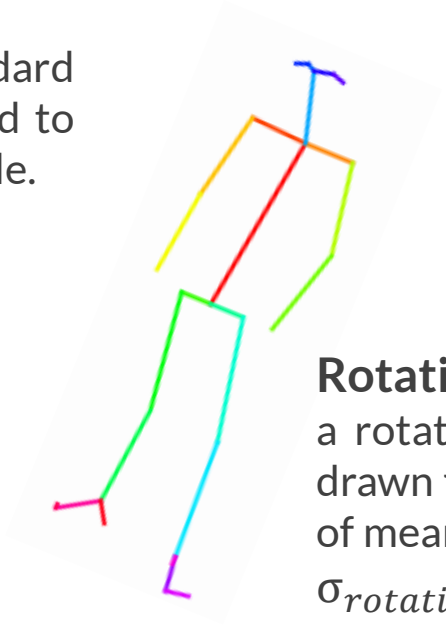
Pose classification – Dataset augmentation

Remove keypoints
Pre-defined or a random list of keypoints are removed (coordinates set to 0) from the sample.

Scaling
A random scaling factor drawn from a normal distribution of mean 0 and standard deviation σ_{scale} is applied to all sample coordinates.

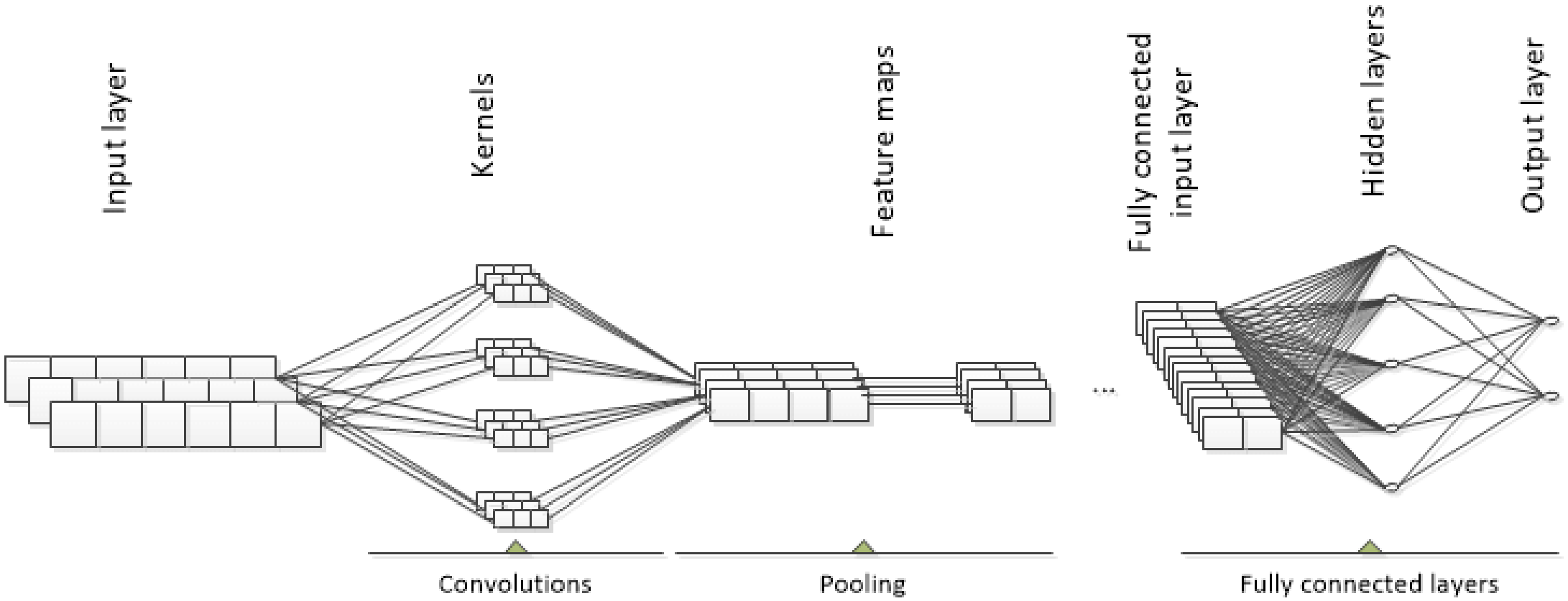
Noise
Gaussian noise of standard deviation σ_{noise} is added to coordinates of the sample.

Augmentation Ratio	σ_{scale}	$\sigma_{rotation}$	σ_{noise}	Remove keypoints
10%	0.08	0.0	0.0	None
10%	0.0	10.0	0.0	None
15%	0.0	0.0	0.03	Legs
15%	0.0	0.0	0.03	Legs & Hip
20%	0.0	0.0	0.03	2 random



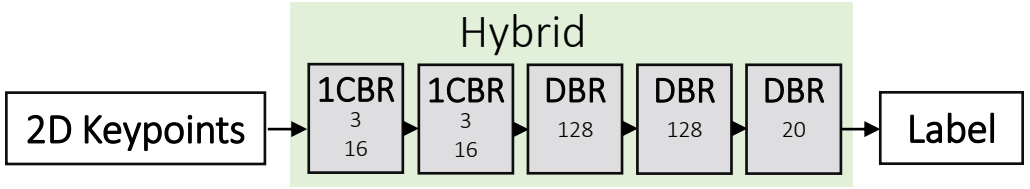
Rotation
a rotation of an angle randomly drawn from a normal distribution of mean 0 and standard deviation $\sigma_{rotation}$ is applied to the sample.

Pose classification – One dimensional convolution



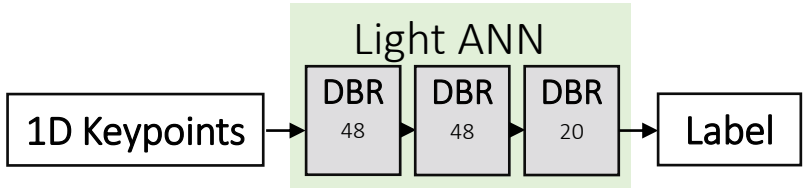
Source: Kolar, Davor & Lisjak, Dragutin & Pająk, Michał & Gudlin, Mihael. (2021). Intelligent Fault Diagnosis of Rotary Machinery by Convolutional Neural Network with Automatic Hyper-Parameters Tuning Using Bayesian Optimization. Sensors. 21. 2411. 10.3390/s21072411.

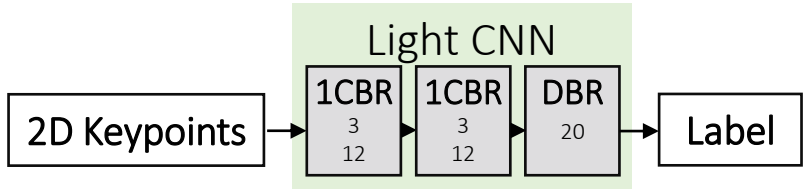
Pose classification
– Models training

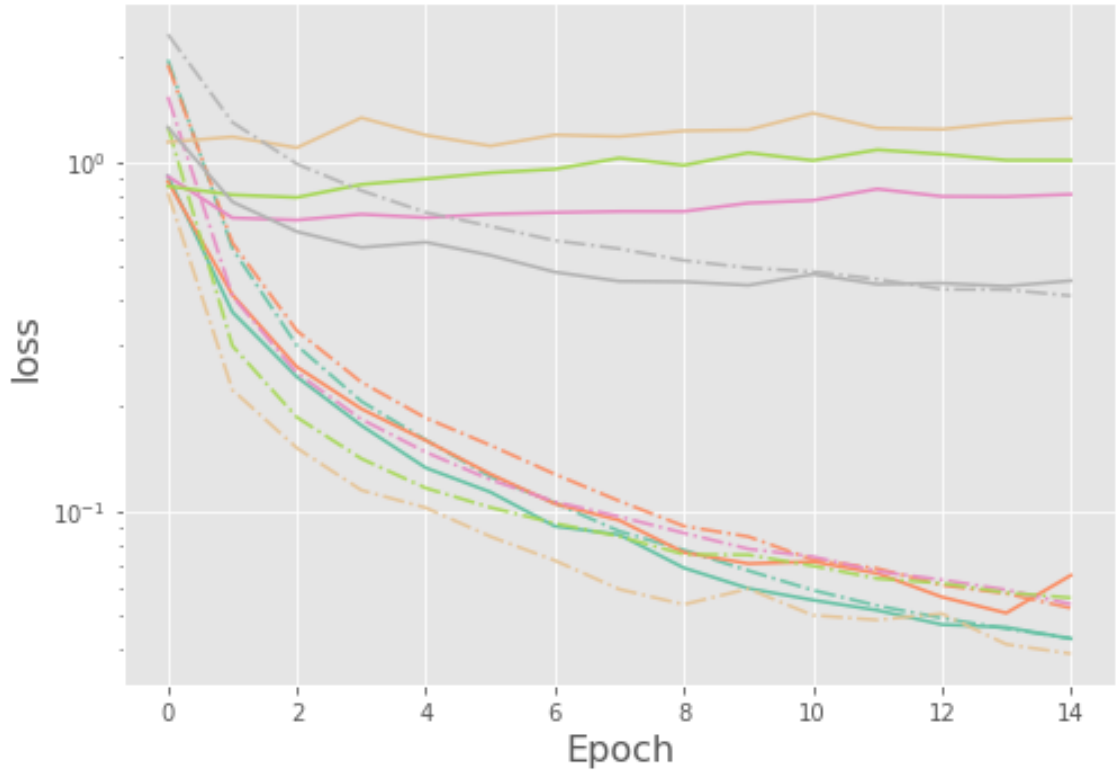


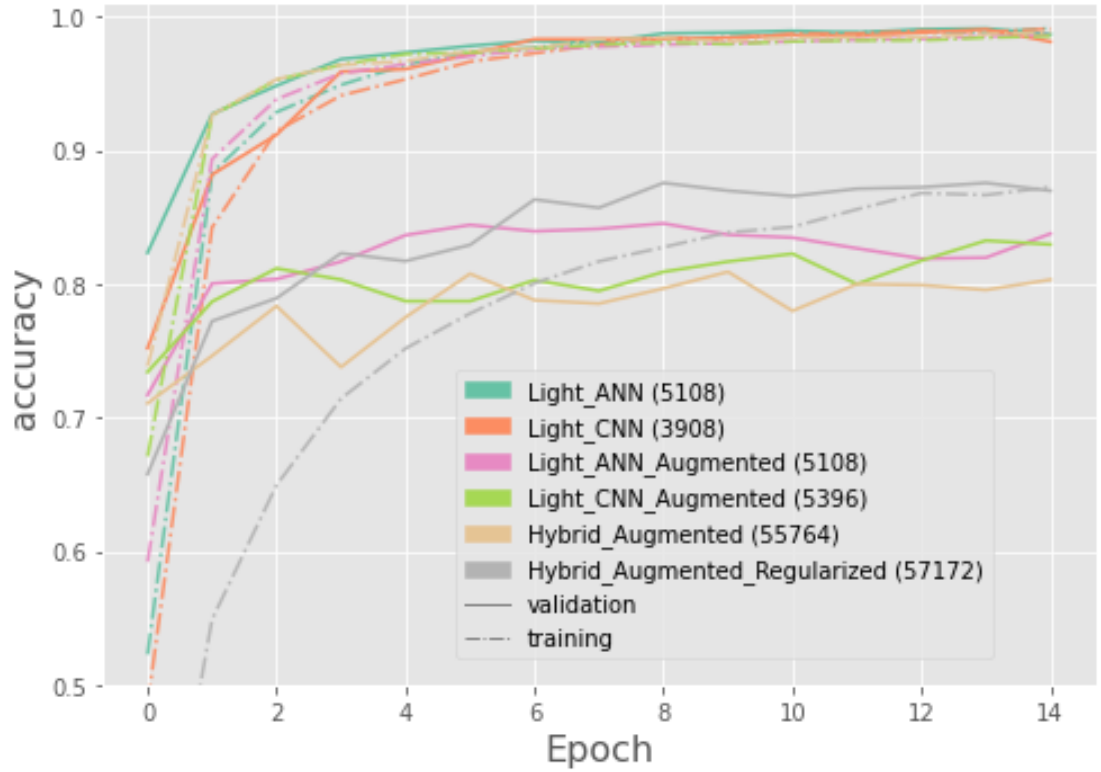
1CBR (Kernel, # Filters): 1D Convolution, Batch normalization, Drop-out, ReLu activation

DBR (Units): Dense layer, Batch normalization, Drop-out, ReLu activation





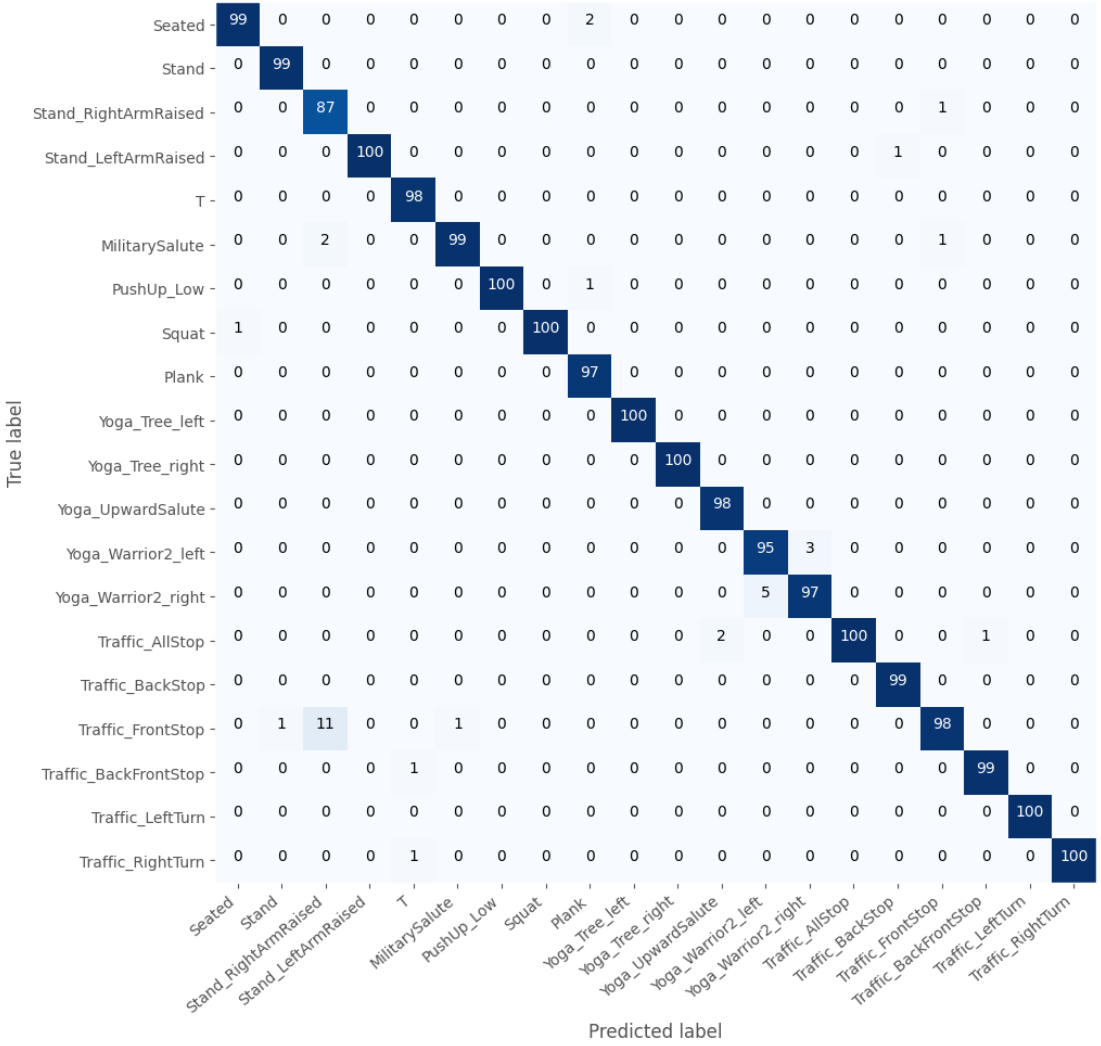




Pose classification – Light CNN results

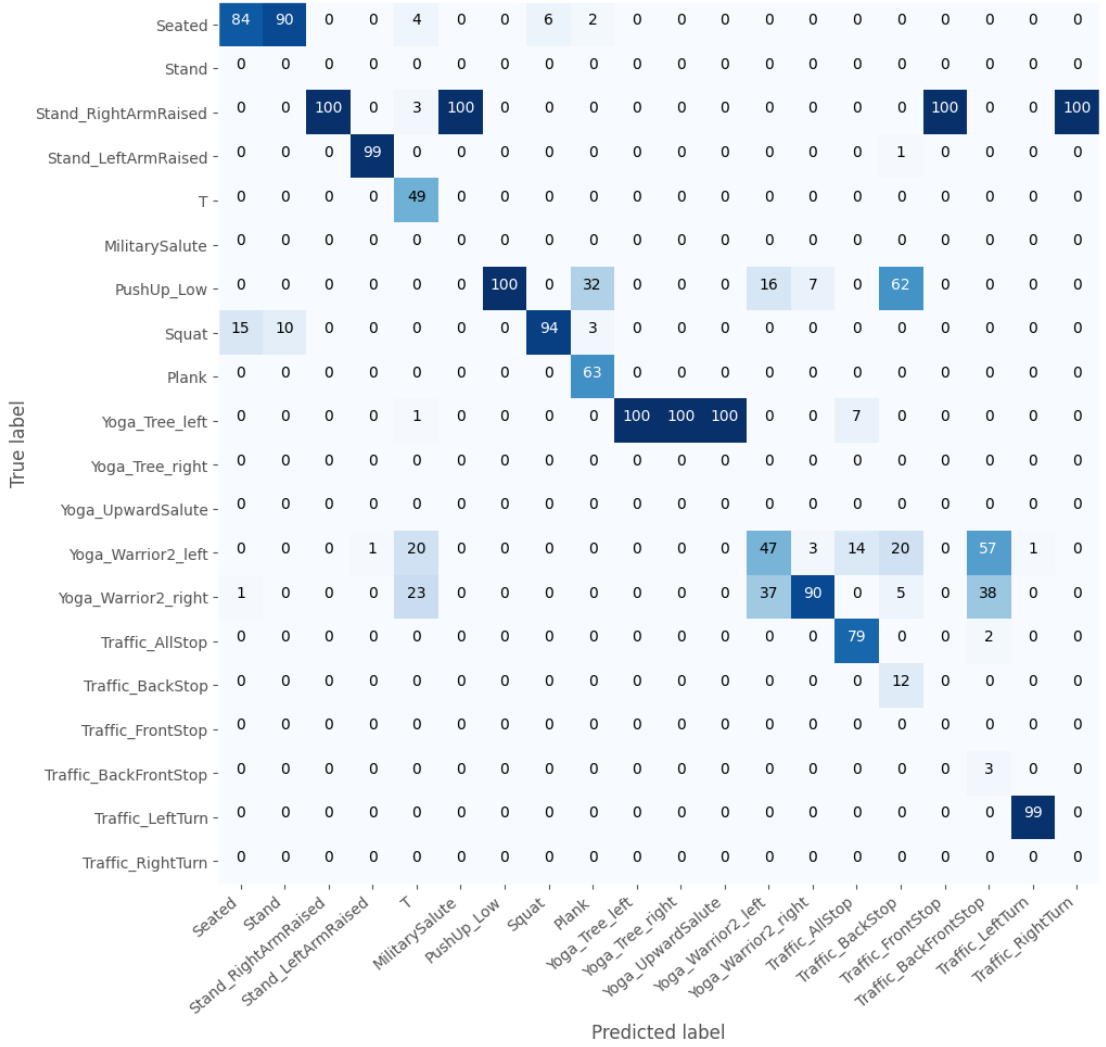
Original dataset

Testing accuracy: 98.25%



Augmented dataset

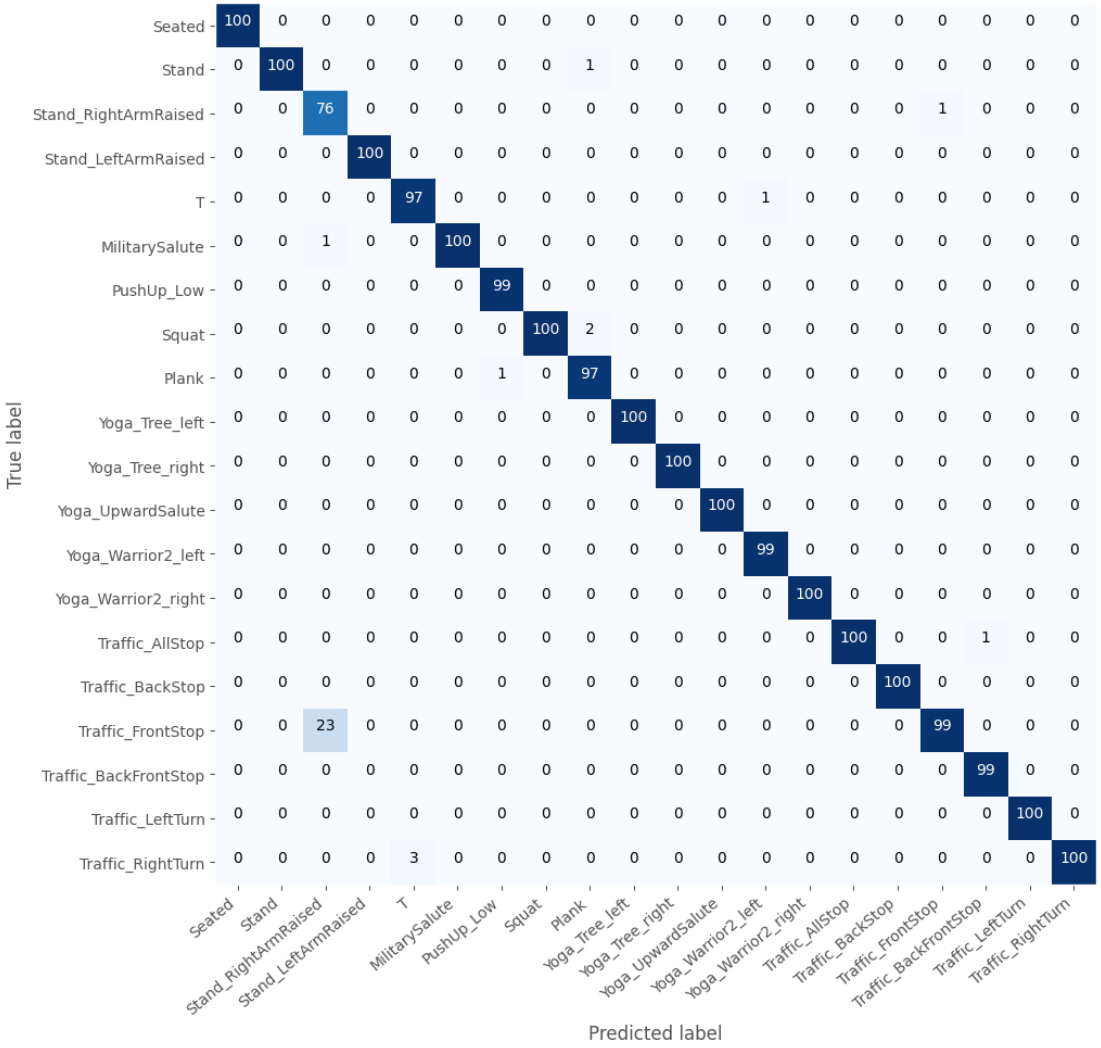
Testing accuracy: 50.95%



Pose classification – Hybrid model results

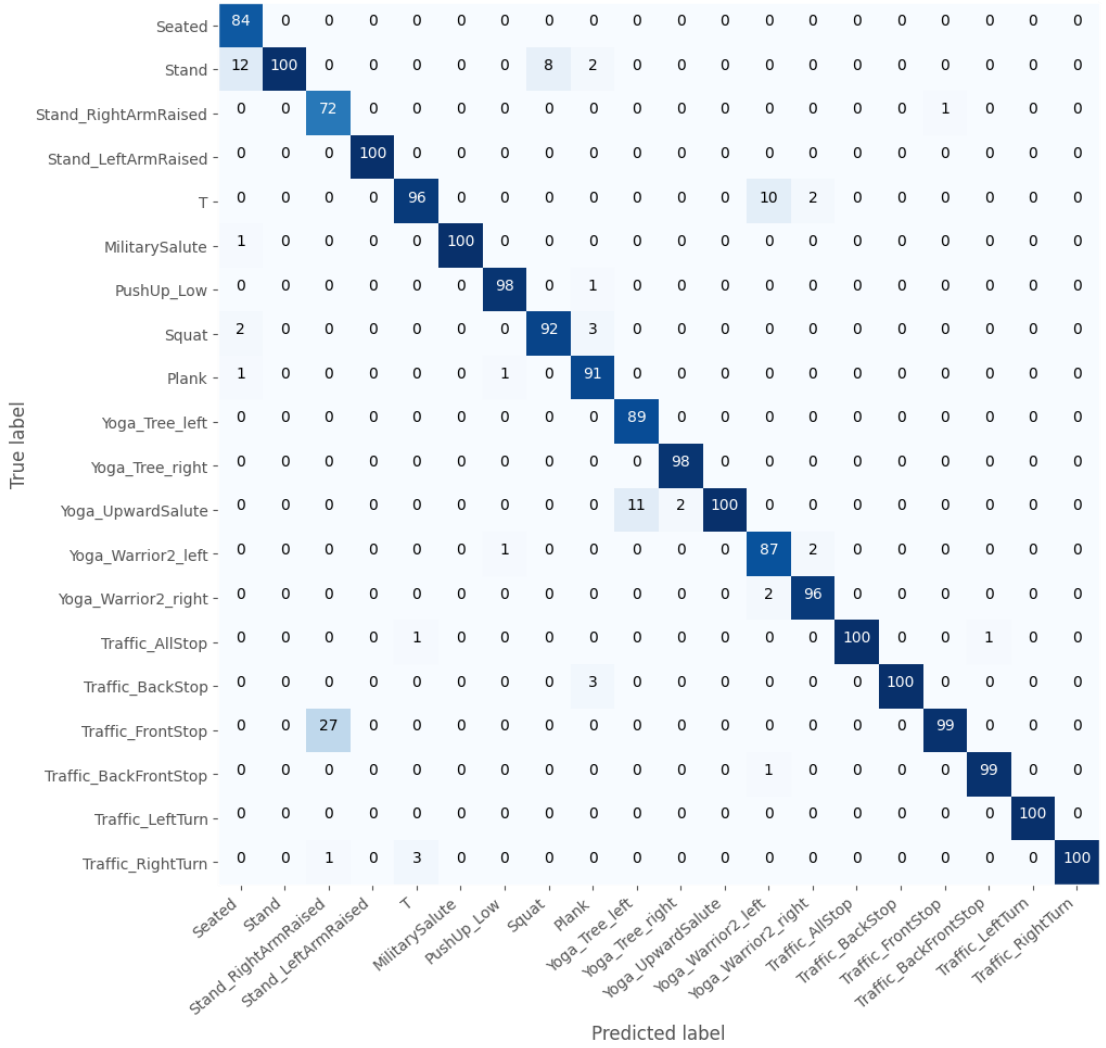
Original dataset

Testing accuracy: 98.30%



Augmented dataset

Testing accuracy: 95.05%



Deployment – Software stack

CSI Camera

Flight Controller

MAVLINK
MICRO AIR VEHICLE COMMUNICATION PROTOCOL

NVENC Chip
Video capture

gstreamer

```
nvarguscamerasrc ! video/x-raw(memory:NVMM),
↪  width=(int)398, height=(int)224,
↪  format=(string)NV12, framerate=(fraction)60/1 !
↪  nvvidconv top=0 bottom=224 left=87 right=311
↪  flip-method=0 ! video/x-raw, width=(int)224,
↪  height=(int)224, format=(string)BGRx !
↪  videoconvert ! video/x-raw, format=(string)BGR !
↪  appsink
```

CPU
Orders management

DRONEKIT

Data transfer
& threads synchronization

TensorRT

GPU

NVIDIA CUDA

Jetson Nano

Deployment – Results

Pipeline inference time [ms] after optimization and compilation:

Original: 453		Data length FP32	Data length FP16
Size	1GB	89	68
	33MB	86	67

Pose estimation model

Original: 184		Data length FP32	Data length FP16
Size	1GB	6	6
	33MB	5	5

Pose classification model

Fail-safe systems:

- Output buffer to filter-out misclassified labels.
- Check the status of the drone to ensure the feasibility of actions.
- Block the transmission of orders with a switch on the radio controller and allow manual order override.
- Ignore critical commands on the serial port of the flight controller.

Orders manager: Gesture controls currently supported

Pose	Action
T	Arm the drone if it is disarmed and landed; Disarm the drone if it is armed and landed
Traffic_AllStop	Take-off at an altitude of 2.5m if the drone is armed and landed; Land if the drone is in flight
Traffic_RightTurn	Move 2m to the right if the drone is armed
Traffic_LeftTurn	Move 2m to the left if the drone is armed
Yoga_UpwardSalute	Return to Launch (RTL)

Demonstration

Future works

Technical improvements:

- **Concatenate both DL models** (estimation & classification) of the processing pipeline as a single model using a common framework.
- Deploy the pipeline on a **more efficient processing platform** such as Coral devices.
- **Increase input image resolution** to reach better accuracy and optimize multi-video feed simultaneous analysis.

Additional features:

- Augment the order manager with **new gestures** to broaden control possibilities.
- Create a new dynamic dataset to train recurrent neural network and allow **dynamic gestures detection**.
- **Add user tracking** to improve robustness and usefulness.
- **Add continuous controls** to the order manager to allow precise control.