DEEP LEARNING FOR GESTURE RECOGNITION IN SPORTIVE TRAINING OPERATION PERFORMED BY STANDALONE SPECIALIZED NEURAL NETWORK VISION-BASED SYSTEMS

Bernardo Lanza[(1)], Cristina Nuzzi[(1)], Simone Pasinetti[(1)], Matteo Lancini [(1)]
[(1)] Dip. di Ingegneria Meccanica e Industriale, Università di Brescia, Via Branze, 38- 25123 Brescia
mail autore di riferimento: bernardo.lanza@unibs.it

1. INTRODUCTION

Thanks to our collaboration with AB-Horizon, we are developing a smart mirrored AR *cobot* to assist people during training. Our aim is to develop an intelligent vision system for action recognition in fitness training. Human movements during physical exercise are various and heterogeneous so traditional mechanical measurement systems are ill-suited to assess human motion in such a wide variety of movements [1]. A subjective approach is limited and not suitable for an extensive commercial production. However new techniques as Deep Learning (DL) for RGB images let us perform model-based estimation to measure quantities intrinsically subjective [2]. DL visual network reads data from the image plane and generates a skeletal model in this 2D representation. New DL architectures and frameworks are constantly released, their oriented approach to hardware optimization supports and inspires embedded device application [3][4]. In this paper we show our first result, the operative strategy and the calibration test we intend to perform to validate our pose estimation project.

This project is focused on developing an embedded device capable of qualitative bio-mechanical analysis. A key point of this application is to establish a trade-off between hardware cost and software computational consumption: for this reason we benchmarked different solutions that could be used for this task. Different MPU suitable for inference engine (CPU, GPU, VPU) were tested with well-known DNN (Deep neural network) to assess their performance. Our specific objective leads to a highly customized and specialized software development: general on the motion to acquire and specific on the hardware constrains. We will confront various network manager such the *TensorFlow* framework and different optimization technique as *Transfer Learning* and *Fine tuning* for our sequential convolutional network. After choosing the right hardware, customized training operation for our model will be mandatory to accomplish accuracy requirements.

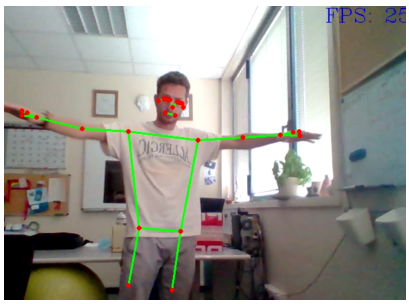

Figure 1 – Real time pose acquisition with CPU backend.

2. METHODOLOGY

We propose a vision-based acquisition system composed by two wide-angle RGB cameras. Each camera system will perform independent inference operation based on a neural network specialized in a specific body-part.

Our aim is to:

- Identify and test an embedded machine capable of running inference operation from a DNN.
- Develop and customize a software for neural networks handling and cameras control.
- Estimate the position of human body's joints
- Validate these measurements via ground truth calibration.

Fitness' experts require the pose and movement of the fundamental segments of our body to assess training, so it is mandatory to correctly estimate the vectors and the body-frame that describe human motion.

From the geometrical point of view, we want to firstly validate our system with a ground-truth analysis using as reference a marker-based infrared multicamera mocap system (BTS - Italy) [5]. Calibration information such the uncertainty associated to the roto-translation vectors of each body frame will allow further correction. From our multi-camera system, we will detect multi-view joint

position and fed it into a post-processing sensor-fusion algorithm tuned from the ground-truth calibration.
Furthermore, understanding the performance of our software means to evaluate its real-time processing capability. The camera's framerate serves as an upper limit for our algorithm processing rate, but speed is mainly dependent on the processing power available, and the algorithms used.
A first benchmark we propose is based on confrontation between different inference engine supporting compatible neural networks. For instance, offline DL algorithms need significant amounts of computing resources (high level GPU's) and they are not embedded-optimized but, with no real-time restriction, they are capable of handling much more data and produce better-accuracy estimations resulting in a much longer processing time. In addition, it is possible to integrate depth information in this class of algorithms [6].
Therefore, a post-processing analysis of our embedded pose estimation system can explain how real-time application are strongly affected by the latency phenomenon.
Offline pose estimator will generate the same class of data as our software and a complete analysis in different condition will let us understand the weakness of RT application. We will correct it with model tuning operation based on the fundamental movement of the fitness training.
We will strive to support different specialized neural networks and our proposed solution is to design a multi-core independent system with embedded cameras. Each sub-system will work separately but with parallel threads and a synchronization routine. We suggest the implementation of a Raspberry PI cluster. Such a system can support multiple CPU-based inference engine networks. To gain more computing power we manage to integrate VPU (Myriad) backend for neural networks, upgrading the cluster hardware with INTEL Movidius NCS 2. Our intent is to develop is a total customized and embedded optimized model for DL. It will be computationally efficient with pre and post processing algorithm like ROI detection approach [7] and pose prediction. Transfer learning will also allow re-training of the network with a customized dataset loaded with fitness movements recorded and specific pose images.

3. VALIDATION

We based our core functionalities on the Mediapipe pipeline. This framework combines the performance of the Tflite [11] DL pose model [3] with a graph-based-calculator written in Cpp. It is capable of synchronously predict and inference the pose data from a video streaming. The two-step detector-tracker ML pipeline combines powerful prediction and tracking algorithm with inference on a Tflite pose model for body recognition. The prediction calculator is based on a region of interest (ROI) detection Tflite model, tiny and lite. The detector calculator generates a ROI according to the result of the neural face detector (TFlite DL model, it runs at a speed of 200–1000+ FPS [9]). However, if some high confident pose data are present in the previous frame (pose presence detected and tracked) the heavy pose landmark TFlite model will work only on a Tracking-extracted area (Landmark from the previous frame);
In this way the detector is invoked only as needed.

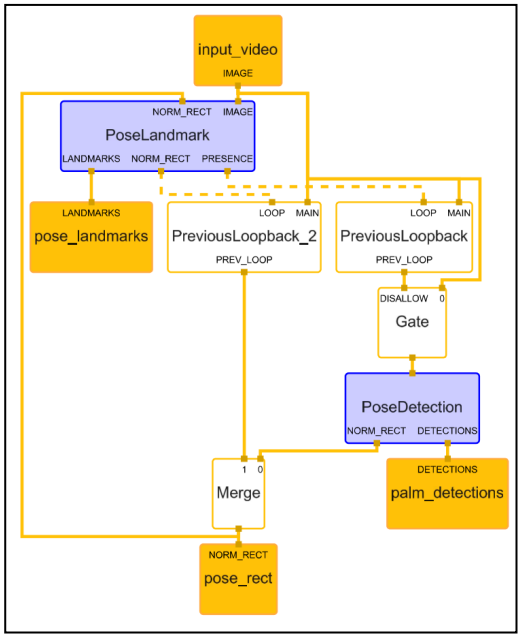

Figure 2 – Mediapipe Pipeline block diagram: in blue the detector and inference calculator [10].

Mixtures of different processing technique allow saving computation resources; latency phenomena are under control of supervisor calculators that tune and combine tracking and inference as needed (our aim is an accuracy and speed tradeoff).
This framework also offers GPU-optimized libraries to exploit the power of dedicated video card running an inference engine [12].
Mediapipe is built as a modular block diagram, therefore customized model can be easily implemented and incorporated in the framework.
We developed a customized Blazepose model (Tflite), specifically tuned for sportive action recognition and based on the original MP Pose model [3]. We customize the model to simplify and specialize its capacity. Tensor Flow (TF) framework manage the transfer learning operation, a technique to repurpose the identity of a neural network [13]. The TFLite model converted back in TF standard format is splitted up to his fundamental layer. Then the last class layer is retrained as needed. Afterwards the whole model is reassembled and easily compressed in the TFLite format. For the retraining operation we assemble a whole new dataset, much smaller respect to standard trainer dataset but effective for our tuning purpose.

Performance is the critical aspect in real time application. Market investigation let us define an embedded PC capable to run and support Neural networks and inference process. We focused on the ARM-based device because of their combination of performance,

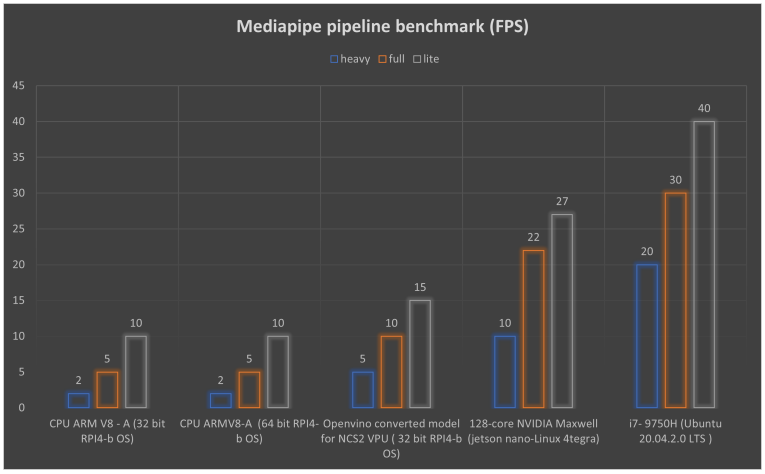

Figure 3 – Benchmark results on ARM CPU's. Mediapipe pipeline was tested on Raspberry PI 4b and NVIDIA Jetson nano (64bit ARM CPU's). Performance analysis focus on the frame rate processing speed.

compactness, and cheapness. To evaluate every embedded system candidate, we tested the neural network with the python package developed for ARM CPU ([14]). We run The Mediapipe pipeline on Raspberry Pi 4b with the powerful ARMv8 CPU but current state of art of the MP embedded-optimized library does not support 64-bit ARM CPU (from MP framework is seen as an ARMv7 32-bit CPU) so we are not able to exploit the power of this innovative MPU (micro-processor-unit).
For these compact devices, real time Deep learning application are quite well supported but performances are not stable, and the architectures of neural network can considerably affect latency and accuracy.
A first approach was to convert the TFlite model to Openvino format and run the inference computation on the Neural Compute Stick v2 VPU. We achieved speed and accuracy but not as expected. Both the detection and the landmark models were converted to the Openvino format and rearranged into the MP pipeline. In fact, the inference computed on such these models can only be performed by the Movidius Myriad core (Openvino doesn't support ARM CPU [15]).
At this point, the ARM strategy seems to be deprecated and not functional to our scope, anyway these CPU can manage the environment for our project and support the GPU backend strategy. We're now focusing on a NVIDIA based device with an aarch64 architecture.
GPU based embedded device, clearly more expansive and a bit more power consumptive, will be our identified solution. The NVIDIA Jetson Nano is compact and quite powerful to run the Tflite full

model at 22 FPS, enough for our real-time application. The tiny and compact GPU of this board is the perfect compromise we are searching for. Finally, we tested the MP pipeline on the Intel i7-9750H CPU for reference data.

REFERENCES

[1] C. S. Chan and H. Liu, "Fuzzy Qualitative Human Motion Analysis," in *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 851-862, Aug. 2009, doi: 10.1109/TFUZZ.2009.2016553.

[2] Zhe Cao, Student Member, IEEE, Gines Hidalgo, Student Member, IEEE, Tomas Simon, Shih-En Wei, and Yaser Sheikh, "Trends in OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", 2019.

[3] Valentin Bazarevsky and Ivan Grishchenko, Research Engineers, Google Research,"On-device, Real-time Body Pose Tracking with MediaPipe BlazePose", 2020.

[4] OpenVINO™ Toolkit Overview https://docs.openvinotoolkit.org/latest/index.html#index

[5] Matt Topley, James G. Richards University of Delaware, Newark, DE 19716, USA. "A comparison of currently available optoelectronic motion capture systems", 2020.

[6] Mao Ye, Xianwang Wang, Ruigang Yang, Liu Ren, Marc Pollefeys, University of Kentucky, HP Labs, Palo Alto, Bosch Research, ETH Zurich. "Accurate 3D Pose Estimation From a Single Depth Image", 2011.

[7] https://google.github.io/mediapipe/solutions/pose.html https://github.com/google/mediapipe/blob/master/mediapipe/modules/face_landmark/face_landmark_landmarks_to_roi.pbtxt

[8] Jie Lu Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, Guangquan Zhang "Transfer learning using computational intelligence: A survey", 2015.

[9] BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs Valentin Bazarevsky Yury Kartynnik Andrey Vakunov Karthik Raveendran Matthias Grundmann Google Research 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA

[10] https://viz.mediapipe.dev/demo/pose_tracking

[11] https://www.tensorflow.org/lite

[12] https://google.github.io/mediapipe/framework_concepts/gpu.html

[13] Sarkar, Dipanjan, Raghav Bali, and Tamoghna Ghosh. *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*. Packt Publishing Ltd, 2018.

[14] https://pypi.org/project/mediapipe-rpi4/

[15] https://docs.openvinotoolkit.org/2021.3/openvino_docs_IE_DG_supported_plugins_Supported_Devices.html