A Dynamic and Collaborative Deep Inference Framework for Human Motion Analysis in Telemedicine

Michele Boldo, Damiano Carra, Davide Quaglia
Dept. of Computer Science
University of Verona
Verona, Italy
{michele.boldo,damiano.carra,davide.quaglia}@univr.it

Nicola Bombieri
Dept. of Engineering for Innovation Medicine
University of Verona
Verona, Italy
nicola.bombieri@univr.it

***Abstract*—Human pose estimation software has reached high levels of accuracy in extrapolating 3D spatial information of human keypoints from images and videos. Nevertheless, deploying such intelligent video analytic at a distance to infer kinematic data for clinical applications requires the system to satisfy, beside spatial accuracy, more stringent extra-functional constraints. These include real-time performance and robustness to the environment variability (i.e., computational workload, network bandwidth). In this paper we address these challenges by proposing a framework that implements accurate human motion analysis at a distance through collaborative and adaptive Edge-Cloud deep inference. We show how the framework adapts to edge workload variations and communication issues (e.g., delay and bandwidth variability) to preserve the global system accuracy. The paper presents the results obtained with two large datasets in which the framework accuracy and robustness are compared with a marker-based infra-red motion capture system.**

***Index Terms*—Human pose estimation, split computing, tensor quantization, run-length encoding, Edge-Cloud computing.**

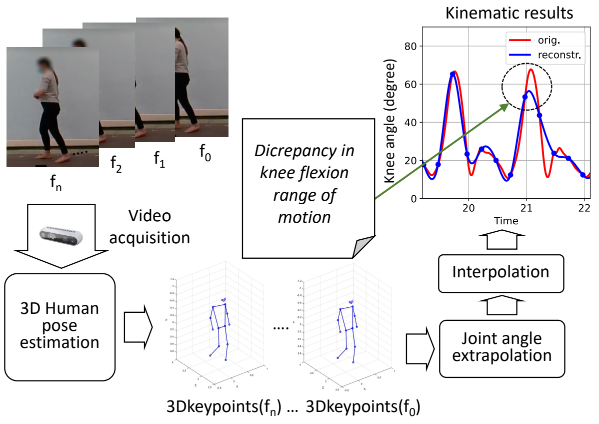

Fig. 1: The pipeline of motion analysis and impact of frame rate on the accuracy.

I. Introduction

One of the main open challenges in telemedicine is to implement *marker-less* human motion analysis at a distance through high-end RGB cameras and human pose estimation (HPE) software [1]. Delivering such a service in telemedicine requires the system to satisfy both functional and extra-functional constraints at the same time, such as high accuracy, real-time, portability, and privacy compliance. A trend solution is to offload HPE software, based on deep neural networks (DNNs), on mobile/IoT computing devices *at the edge* (*end devices* in the follows), by which the video streams (i.e., the sensitive information) are elaborated close to the camera, while only the process results are sent over the communication network (i.e., WiFi/Ethernet, Internet) [2].

To deal with the resource limitations of the edge devices, different solutions have been proposed, such as model compression (e.g., neural network pruning [3], quantization [4], and compact network design [5]), and early exiting [6]. In all these solutions, the goal is to reduce the resource consumption through a tuning of the model structure or through the discretization of the model parameters. In general, such a *model abstraction*, results in a significant accuracy loss, which prevents the software to be used in clinical applications. In the context of human motion analysis, the accuracy of the inferred kinematic information strictly depends on two factors: (i) the spatial accuracy of HPE keypoints in each single frame, and (ii) the *real-time performance* to extrapolate the keypoints from the video without losing any frame [7]. Even assuming a spatial accuracy close to 100% for each keypoint, reducing the frame rate may lead to discrepancies in the joint flexion range of motion and, as a consequence, to different clinical interpretations. For example, the plot in Fig. 1 shows the knee angle extrapolated by a 100 frame/s marker-based motion capture system (i.e., the ground truth represented by the red line in the figure) compared to the same information obtained by the OpenPose HPE system [8] limited to 4 frame/s by the available computational resources with value interpolation for the missing frames (blue line).

Collaborative deep inference has been introduced to overcome accuracy degradation by partitioning the DNN between multiple devices (i.e., end-devices and the cloud) [9]. In this distributed solution, the performance depends on the computational capability of the end-device (e.g., a resource constrained board) and on the condition of the communication network (e.g., frequently there is a wireless between the end-device and the edge-server/cloud). Even more challenging is the case in which both aspects are time-varying. This work addresses this challenge, by presenting an edge-cloud collaborative and adaptive framework for human motion analysis at a distance. The main contributions are:

- An analysis of the DNN partitioning technique applied to the HPE software, with particular emphasis on the the accuracy of the extrapolated kinetic results. So far, only classification has been considered in split computing.
- A framework in which the DNN partition point is tuned at run time to deal with *workload variations on the end-device* beside on the communication channel. Differently from the state of the art techniques that implement model abstraction, the proposed technique does not require re-training of the DNN, which would limit the dynamic adaptability.
- An extended analysis conducted with a standard dataset and an additional dataset of clinical trial videos with a marker-based infra-red motion capture system (Vicon) as ground truth to measure the system accuracy and robustness.

II. Background and Related work

DNN partitioning consists of executing the first N layers of the DNN in a given computing node and the remaining ones in another node. Each layer partition has its computation overhead and latency while the transmission of intermediate tensor data also consumes communication bandwidth and leads to further delay depending on data amount. The research on model partitioning mostly focused on finding the optimal partition solution by considering total latency, energy consumption, and accuracy [10], [11]. Adaptive frameworks [9], [12], [13] dynamically split DNN layers between the front-end and the back-end node according to the instantaneous communication bandwidth. Nevertheless, they do not address edge workload variations and tensor quantization. Indeed, a crucial point is the way to arrange data for their transmission on the communication channel. The main trend is to reduce data amount by simplifying the inference model [14], [15] thus reducing the accuracy. This approach requires to re-train the neural network which is time-consuming and limits the dynamic adaptability of the split. For this reason, other researchers propose to decrease tensor size by recurring to traditional compression techniques. If the compression scheme is lossless, e.g., in case of run-length encoding and entropy coding [16], [17], data reduction does not lead to accuracy loss as in case of quantization and JPEG-based techniques [18]. Regarding the effect of packet loss on accuracy, some research showed that reliable transmission protocols are not always the best option especially if they introduce further delay [19]. However, packet drop has been mainly considered as an effect of network problems and not an opportunity to reduce channel occupancy. Our approach aims at filling these gaps by providing an adaptive framework which takes into account the workload of the end-device and the available network bandwidth at runtime and dynamically chooses the partition point and the transmission strategy. In addition, it is worth noting that most of this literature focuses on DNN split for classification task, where real-time is not a constraint. In contrast we address complex inference applications, i.e., HPE systems, for which the realtime constraint is crucial for the accuracy of the kinematic data extrapolation [20].

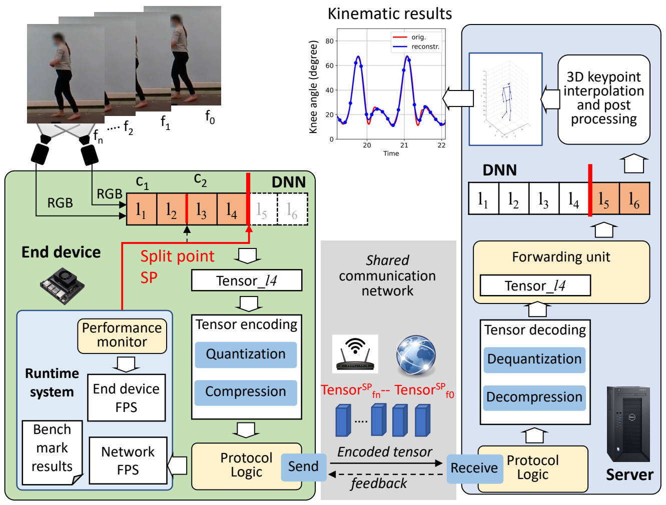

Fig. 2: Framework overview.

III. Methodology

Fig. 2 shows the overview of the proposed framework, which partitions the DNN model between the end-device and a server connected by a communication network. The end-device processes the video frames of the RGB sensors by using the first inference layers up to the split point. The end-device also implements a runtime system, which dynamically sets the split point according to the available resources on the end-device and on the communication network (see Section III-A). The runtime system selects the split point among a set of *candidate layers*, whose impact on the performance system is extrapolated statically through a preliminary benchmarking phase (Section III-B). The tensor at the split point undergoes 4-bit sample quantization and run-length encoding before being sent over the channel together with additional information on the current split layer (Section III-C). The server node decodes the tensor and forwards the result to the next DNN layer to complete the inference process. It also interpolates the two sets of 2D keypoints to generate the set of 3D keypoints. It applies spatial and temporal filtering to eliminate outliers and performs a cubic spline interpolation to generate spatial and angular kinematic data.

A. Runtime system for two-level adaptability

Considering that the application is a pipeline of end-device computation, tensor transmission and server computation, the global working frequency is limited by the slowest stage (assuming no computational constraints for the server). Since the resources of the end-device and of the transmission network are shared among several processes and data flows, respectively, their availability can vary during the execution of the healthcare application. This fact can increase the latency of the corresponding stage of the pipeline thus compromising the global working frequency. Algorithm 1 shows the pseudocode

Algorithm 1 Run-time manager

```
Require: $current\_layer, E, D, F, B_{current}, EndFPS_{current}$
 1: $max\_data\_size \leftarrow \frac{B_{current}}{F}$
 2: $slow\_down \leftarrow \frac{E(current\_layer)}{EndFPS_{current}}$
 3: if $\frac{B_{current}}{data\_size_{current\_layer}} < F$ then
 4:     $new\_layer \leftarrow \arg\max_i(D(i) < max\_data\_size)$
 5:     if $EndFPS_{new\_layer} \cdot slow\_down < F$ then
 6:         jump to line 4 excluding the current $new\_layer$ from $D$
 7:     else
 8:         $new\_layer$ is the new splitting layer
 9:     end if
10: else if $EndFPS_{current} < F$ then
11:     $new\_layer \leftarrow \arg\max_i(E(i) < F \cdot slow\_down)$
12:     if $data\_size_{new\_layer} < max\_data\_size$ then
13:         jump to line 11 excluding the current $new\_layer$ from $E$
14:     else
15:         $new\_layer$ is the new splitting layer
16:     end if
17: end if
```

of the proposed runtime system that dynamically changes the split point to compensate such variations.

The procedure is called whenever the total system frequency, defined as $min(EndFPS_{current}, \frac{B_{current}}{data\_size})$, where $EndFPS_{current}$ is the current edge device working frequency, $B_{current}$ is the current network bandwidth and $data\_size$ is the size of the compressed tensor at the current splitting layer, is less than the target frequency F for our application. An ad-hoc UDP-based protocol (*Protocol logic* in Fig. 2) is used to transmit the tensor and estimate $B_{current}$ by measuring the time between the transmission of the first UDP packet and the reception of the ack of the last one.

Given, as inputs, the identifier of the current splitting layer $current\_layer$, the two ordered lists E and D generated by the benchmark (see Section III-B), the target working frequency F, the current network bandwidth $B_{current}$, and the current working frequency of the edge $EndFPS_{current}$, the system first computes the maximum tensor size max_data_size that allows the system to keep pace with the current available bandwidth (Line 1). Assuming that an increase of the end device workload impacts proportionally on the inference step and tensor encoding of each layer, the algorithm extrapolates a $slow\_down$ factor from the optimal FPS (i.e., with no interferences) of the current layer collected in the benchmarking phase and the current frequency $EndFPS_{current}$ (Line 2).

Then, the runtime manager identifies the issue that slows down the system between the network bandwidth reduction and the workload increase on the end device (Lines 3-10). In the first case, it identifies the new split layer from D through $argmax$, which retrieves the right-most layer from D among all candidates that solve the issue (Lines 3-4). Since the selected layer may involve a different workload in the end device and, in turn, may lead the system performance slower than F, the manager checks if the new_layer performance satisfies F before switching (Line 5). If not, it selects a new candidate from the list. The algorithm selects the new_layer similarly by checking the workload increase in the end device. If there is no candidate layer that satisfies the constraint, the runtime manager selects, as the new_layer, the candidate that provides the best tradeoff between network and end-device performance.

B. Benchmarking and candidate identification

To find the *candidate split layers*, we selected a representative set of video streams from the standard H3.6M dataset [21]. The benchmark phase runs the HPE application over such a dataset on the end-device. As a result, each split point at layer i-th is characterized in terms of average inference time t_i, average tensor quantization and compression time t_i^{enc}, and average size of the compressed 4-bit tensor $data\_size_i$. An additional task incrementally scans the DNN layers to retrieve, for each layer i, the memory footprint ($MemFootprint_i$) required to store and run the partial inference up to layer i. The algorithm for the candidate identification takes as input the final number of candidates and the target working frequency of the system F (poses per second) and consists of two steps. First, all split points that do not satisfy the following equations are discarded:

$$MemFootprint_i \leq EndDeviceMem \quad (1)$$

$$EndFPS_i = \frac{1}{(\sum_{j=0}^{i} t_j) + t_i^{enc}} \leq F \quad (2)$$

$$NetFPS_i = \frac{B}{data\_size_i} \leq F \quad (3)$$

The first equation represents the architectural constraint on the end-device, and limits the selection to the layers that can be hosted in the available memory of the board. The second and third equations derive from the consideration that the application is a pipeline of end-device computation ($EndFPS_i$), tensor transmission ($NetFPS_i$) and server computation. As a consequence, the global working frequency is limited by the slowest stage (assuming no computational constraints for the server). Tensor transmission time depends on the instantaneous network bandwidth B and the size ($data\_size_i$) of the compressed tensor.

The second step implements a clustering phase, by which the remaining layers are grouped, in order from the left to right, into n equal sets. For each set s_j, the algorithm identifies a candidate split layer as the layer that represents the local maximum performance:

$$FPS_j = max[min(EndFPS_i, NetFPS_i)], \quad l_i \in s_j$$

This clustering approach allows the system to execute the maximum part of the inference phase in the end device while satisfying the performance constraint. The framework is independent of this choice and other voting solutions, which are part of our future work, can be adopted. The final result of the benchmarking phase is two ordered lists E and D, which contain the $EndFPS$ and $data\_size$, respectively, of the selected candidates.

C. Tensor encoding

To limit the amount of data sent across the communication channel, the end-device encodes the tensor information through *quantization* and *compression* to reduce the tensor size while guaranteeing a negligible impact on the overall system accuracy. We implement quantization directly on a single element of the output tensor. Both DNN weights and activation functions do not change and, thus, no retraining is required.

We adopt a scheme that uniformly quantizes the interval between the minimum and the maximum value [22]. Given n bits for the quantization, the quantized coefficients are integers that represent the identifiers of the 2^n possible intervals. To correctly rebuild the tensor coefficients at the destination, the minimum and the maximum values (32 bits each) are included in the message together with the quantized coefficients. An irreversible quantization error is generated but, as confirmed by our experimental results, it has a negligible effect on the application accuracy.

With quantization, the smallest coefficients are mapped to the value of zero. Furthermore, the rectifier linear unit (ReLu) activation function, which is largely adopted in HPE DNNs for pooling the convolutional layer results, transforms all the negative numbers into zeros. The combinations of these two steps produces a large amount of zeros in the resulting multidimensional tensor[1], which can be exploited by a run-length encoding compression scheme that works on a single pass on the input with small complexity.

IV. Experimental Results

Settings. We evaluated the proposed framework using a platform composed by an NVIDIA Jetson Nano (4-cores CPU, 128-cores GPU, 4 GB RAM) as end-device, and a server Desktop with Intel i5 7400 CPU, 2xNvidia RTX 2070 GPUs (SLI), 16GB DDR4 RAM, and Ubuntu 18.04 LTS OS. The end-device is connected to a Ubiquiti UAP-AC-M access point with a 50 Mbps WiFi connection. Without loss of generality, we assume that the server is directly accessible through the WiFi link. We implemented the HPE applications with OpenPose [8] and $BODY\_19$ DNN (see Fig. 3) trained with $COCO$ and $MPII$ datasets. We assessed the framework accuracy and robustness using two different datasets. The first is the standard Human3.6M dataset [21], which includes high-resolution videos (1,000x1,000 pixels) from 17 scenarios with 11 actors. The second is a collection of videos taken in a clinical laboratory on a group of five healthy adults using ZED 2 RGB sensors (848x480 pixels) (15 minutes of videos in total). For such a dataset, the ground truth is provided by a 8-camera infrared motion capture system (MoCap) VICON[2] MX 13. For the MoCap, the kinematic data were collected at 100 Hz and the reflective markers (14 mm in diameter) were placed over the following bony landmarks bilaterally: cheekbones, medial and lateral epicondyle of the humerus, acromion, ulnar and radial styloid processes, hand dorsum, greater trochanter, medial and lateral epicondyle of the femur, medial and lateral malleolus ankle, heel, 5th metatarsal of the foot, tip of the big toe.

Results on single frames. We evaluated the impact of the tensor quantization and compression proposed in Sec. III-C on the HPE accuracy, i.e., the difference between the inferred keypoints and the ground truth on 25,200 frames. Table I reports

TABLE I: Impact of the tensor quantization and compression on the HPE spatial accuracy (average with the H3.6M dataset).

Q Level	MAE:AVG (px)	MAE:STD (px)	RMSE:AVG	RMSE:STD	Corr (ρ)
16 bit	0.000	0.010	0.010	0.011	0.999
8 bit	0.068	0.428	0.436	0.483	0.999
6 bit	0.256	0.713	0.769	0.704	0.999
4 bit	0.993	1.325	1.694	1.146	0.998

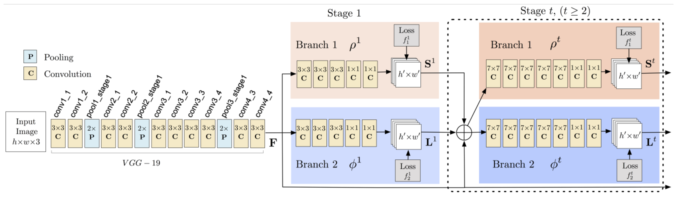

Fig. 3: The adopted $BODY\_19$ model and the corresponding layers for the candidate selection.

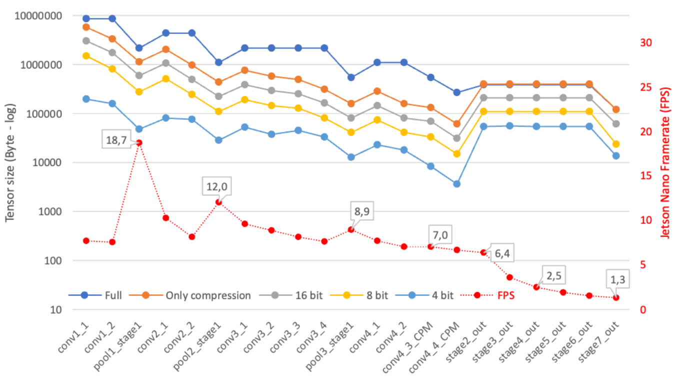

Fig. 4: Tensor size with different levels of quantization and compression (upper side lines and left-most ordinate axis). Inference throughput (FPS) on the Nvidia Jetson end-device (dotted line at the bottom and right-most ordinate axis).

the results of the mean average error (MAE), averaged on all keypoints, and the corresponding standard deviation, the root mean square error (RMSE), and Pearson correlation between the inferred keypoints without and with the quantization and compression. For the sake of space, the table reports only the results of the most meaningful layers (i.e., VGG-19 of Fig. 3). The results show that the impact of the tensor quantization on the spatial accuracy of all keypoints is negligible. The impact on the Pearson correlation, which is a key value for the accuracy of kinematic data, is also negligible. Even with a strong quantization (i.e., 4 bit), the average discrepancy w.r.t. the ground truth is less than one pixel in 1,000x1,000 pixels input frames. The adopted run-length encoding scheme is lossless and therefore does not impact on the accuracy.

Fig. 4 and 5 show the results of the micro-benchmarking (Sec. III-B). As expected, the pooling layers reduce the tensor size and, as a consequence, they are good candidates as split points. In the $VGG-19$ subnetwork, the more the input data undergoes the inference process (i.e., from the left to the right) the smaller the tensor size is. The size starts increasing after the VGG model and remains constant across the sequential stages. This is due to the fact that the considered DNN (and many other state-of-the-art DNN adopted in the HPE

[1]In our experimental analysis, we measured from 60% to 80% of coefficients equal to zero in all datasets.

[2]https://www.vicon.com

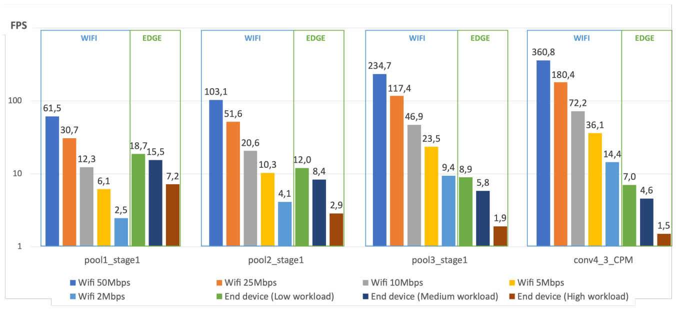

Fig. 5: Result of the benchmarking and candidate identification procedure: system performance (in FPS) with different workloads in the end-device and Wifi available bandwidth.

platforms) is based on a residual model, and the output tensor (e.g., F in the example of Fig. 4) is concatenated to the other tensors (S and L) after every stage. The high number of zeros after the ReLu activation function leads to an efficient compression. Fig. 4 (curve at the bottom, which refers to the y-axis on the right) shows also the performance (FPS) obtained by the end-device, as it processes more and more layers: this provides an indication of the candidate layers that can be used as split points given a target frame rate.

Results on a sequence of frames. Table II reports, for each identified candidate split layer (and for some additional layers for completeness), the system performance (in FPS), with and without compression/quantization, and the corresponding accuracy with no interference on the end-device workload or WiFi bandwidth (i.e., 50 Mbps available). Starting from the 3D spatial position of the keypoints provided by the HPE framework and by the MoCap, the human joint angles are extrapolated geometrically (e.g., ankle, knee, and hip keypoints for the knee angle) according to [23]. For the sake of space, we report the results obtained for the right knee angle, as (i) it is one of the most representative and clinically meaningful in the context of walking, and (ii) its extrapolation relies on one of the most sensitive 3D keypoints (i.e., the ankle keypoint). The magnitude and polarity of the angle were described such that when the knee was fully extended, it was described as 0 degrees flexion, and when the shank moved to a posterior direction relative to the thigh, the knee joint angle was said to be in flexion (knee angle > 0). Two vectors are generated from the inferred keypoints to get the orientation of the knee articulation. The hip keypoint is then defined in the knee reference system. In this way, the angle corresponding to the sagittal plane is derived through standard trigonometric formulas. Without tensor quantization and compression (i.e., state-of-the-art techniques of collaborative deep inference [9]), the $stage2\_out$ split layer provides the best performance (6.77 FPS), and it has 7.29°maximum error in the inferred right knee angle. The performance and the corresponding accuracy sensibly decrease with the other split layers. With compression and quantization, the $stage2\_out$ layer provides similar results, while the left-most layers allow for better performance and higher accuracy (down to $4.84°$). In general, we find that the system keeps the maximum inferred angle error within reasonable limits with computational performance ≥ 7 FPS. Fig. 5 shows the overall system performance with the proposed 4-bit quantization and compression when the runtime system switches the split point among the candidate layers as a consequence of interference on the end-device workload and Wifi bandwidth. We characterize the end-device workload with an increasing number of additional applications of intelligent video analytic with $\sim 30\%$ (*medium*) and $\sim 50\%$ (*high*) of additional GPU workload. As for the network resources, we use the Linux traffic control tools tc6 and tc-netem7 to create a bottleneck and reduce the available bandwidth considering some representative values of 25, 10, 5 and 2 Mbps. Starting from $conv4\_3\_CPM$ as initial split layer (i.e., with no interference on the end-device and WiFi) and by adopting a 4-bit encoding, the runtime system addresses congestion on the end-device and on the WiFi by moving the split point on the left-most or right-most layers, respectively. As summarized by the results in Fig. 5, we find that, in general, the proposed framework achieves the minimum computational performance to keep the maximum inferred angle error within reasonable limits (i.e., 7 FPS) in all conditions except for high workloads on the end-device combined to WiFi bandwidth lower than 2 Mbps. Thanks to the low complexity of the runtime manager algorithm, the switching latency is negligible (2-3 ms).

Previous work showed limited accuracy of markerless motion capture gait analysis compared to IMU or MoCap methods and recommended caution in using these systems in clinical movement analysis [1], [24]. In partial agreement with these considerations, our analysis shows some discrepancies in knee flexion's range of motion estimations between the proposed framework and MoCap. As reported in Table II, the differences between the two systems range between 4.84° and 5.97° within the entire gait cycle. Therefore, a crucial question about the accuracy of the framework analysis concerns the acceptability of this measured error. In general, the literature suggests that a measurement error in joints range of motion estimation greater than 5-7° could be large enough to lead to misleading clinical consideration [25]. In contrast, errors lower than 5° are considered acceptable. A recent reliability study on MoCap gait analysis found relatively large repeated measure errors and suggested a minimal detectable change (MDC) of 6.5° in knee flexion and 6.4° in knee total ROM [26]. The correlation between joints angles assessed during repeated gait analysis with a MoCap system in their study was surprisingly in line with our results comparing data from MoCap and the HPE system ranging around 0.99° in different gait phases. Although some discrepancies in the knee flexion angle estimation between the proposed adaptive HPE framework and the MoCap system could occur, it is conceivable that this should not lead to different clinical interpretations.

V. Conclusion

This paper addressed the challenge of applying human pose estimation software through collaborative deep inference in edge-cloud systems. It focused on the functional and extra-functional constraints that such a system should satisfy to be applied in telemedicine. It presented framework that adapts to edge workload variations or communication issues (e.g., delay and bandwidth variability), which otherwise may compromise the global system accuracy and, as a consequence, could lead to different clinical interpretations. The paper presented

TABLE II: System performance with Jetson Nano and Wifi 50 Mbps in terms of FPS for each candidate layer: Inference on end-device (End FPS), Wifi with no compression/quantization (WiFi full), WiFi with 4-bit quantization+compression, the corresponding MAE, Maximum error, and Pearson correlation on the right knee angle degree.

Candidate	**Edge inf. (FPS)**	**Edge inf. + quant/ compr (FPS)**	**Wifi full (FPS)**	**Wifi 4 bit (FPS)**	**MAE Full (deg)**	**MAX err Full (deg)**	**Corr. (ρ) Full**	**MAE 4 bit (deg)**	**MAX err 4 bit (deg)**	**Corr. (ρ) 4 bit**
pool1_stage1	46.73	18.70	1.37	61.48	15.14	61.33	0.255	0.65	5.28	0.99
pool2_stage1	15.21	11.99	2.74	103.13	9.43	37.40	0.71	0.69	5.01	0.99
pool3_stage1	9.70	8.93	5.49	234.71	1.68	10.39	0.99	0.82	4.84	0.99
conv4_3_CPM	7.47	7.03	5.49	360.78	1.68	10.39	0.99	1.10	5.97	0.99
stage2_out	6.77	6.36	7.60	54.33	1.27	7.29	0.99	1.27	7.29	0.99
stage4_out	2.54	2.48	7.60	55.34	11.26	41.19	0.62	11.26	41.19	0.62
stage7_out	1.31	1.30	24.67	220.65	16.04	61.95	0.10	16.04	61.95	0.10

the results obtained with two large datasets in which the framework accuracy and robustness are compared with a marker-based infra-red motion capture system.

Acknowledgements

The work has been partially supported by Fondazione Cariverona with the grant "Ricerca & Sviluppo". The study was carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, $ECS\_00000043$). This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] E. D'Antonio, J. Taborri, E. Palermo, S. Rossi, and F. Patanè, "A markerless system for gait analysis based on OpenPose library," in *Proc. of IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2020, pp. 1–6.

[2] M. Sassi and M. Abid, "Security and privacy protection in the e-health system: Remote monitoring of covid-19 patients as a use case," *Smart Innovation, Systems and Technologies*, vol. 237, pp. 829–843, 2022.

[3] B. Li, B. Wu, J. Su, and G. Wang, "EagleEye: fast sub-net evaluation for efficient neural network pruning," *LNCS*, vol. 12347, pp. 639–654, 2020.

[4] P. Wang, X. He, G. Li, T. Zhao, and J. Cheng, "Sparsity-inducing binarized neural networks," in *Conference on Artificial Intelligence*, 2020, pp. 12 192–12 199.

[5] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, Q. Le, and H. Adam, "Searching for mobilenetv3," in *IEEE ICCV*, 2019, pp. 1314–1324.

[6] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. of International Conference on Pattern Recognition*, vol. 0, 2016, p. 2464 – 2469.

[7] E. Martini, M. Boldo, S. Aldegheri, N. Valè, M. Filippetti, N. Smania, M. Bertucco, A. Picelli, and N. Bombieri, "Enabling gait analysis in the telemedicine practice through portable and accurate 3d human pose estimation," *Computer Methods and Programs in Biomedicine*, vol. 225, p. 107016, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260722003984

[8] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. of IEEE CVPR*, 2017, pp. 1302–1310.

[9] L. Zhang, L. Chen, and J. Xu, "Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning," in *Web Conference*. ACM, 2021, p. 3111–3123.

[10] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-Efficient Offloading for DNN-Based Smart IoT Systems in Cloud-Edge Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 683–697, 2022.

[11] M. Odema *et al.*, "LENS: Layer distribution enabled neural architecture search in edge-cloud hierarchies," in *ACM/IEEE DAC*, 2021, pp. 403–408.

[12] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive DNN surgery for inference acceleration on the edge," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE, Apr. 2019. [Online]. Available: https://doi.org/10.1109/infocom.2019.8737614

[13] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud," 2020. [Online]. Available: http://arxiv.org/abs/2008.06402

[14] M. Sbai, M. R. U. Saputra, N. Trigoni, and A. Markham, "Cut, distil and encode (cde): Split cloud-edge deep inference," in *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2021, pp. 1–9.

[15] V. Goyal, V. Bertacco, and R. Das, "Myml: User-driven machine learning," in *Proc. ACM/IEEE Design Automation Conference*, 2021, pp. 145–150.

[16] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "JALAD: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS)*, dec 2018, pp. 671–678.

[17] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," in *IEEE AVSS*, 2018, pp. 1–6.

[18] A. E. Eshratifar, A. Esmaili, and M. Pedram, "Bottlenet: A deep learning architecture for intelligent mobile cloud computing services," in *IEEE/ACM Int. Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[19] J. Liu and Q. Zhang, "To improve service reliability for ai-powered time-critical services using imperfect transmission in MEC: An experimental study," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9357–9371, 2020.

[20] N. Saini *et al.*, "AirPose: Multi-view fusion network for aerial 3D human pose and shape estimation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4805–4812, 2022.

[21] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Trans. on Pattern An. and Machine Intell.*, vol. 36, no. 7, pp. 1325–1339, jul 2014.

[22] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of the IEEE CVPR*, 2018, pp. 2704–2713.

[23] D. A. Winter, *Biomechanics and motor control of human movement*. John Wiley & Sons, 2009.

[24] A. Pfister *et al.*, "Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis," *J. of Med. Engin. and Tech.*, vol. 38, no. 5, pp. 274–280, 2014.

[25] J. L. McGinley, R. Baker, R. Wolfe, and M. E. Morris, "The reliability of three-dimensional kinematic gait measurements: A systematic review," *Gait & Posture*, vol. 29, no. 3, pp. 360–369, Apr. 2009.

[26] M. Geiger, A. Supiot, D. Pradon, M.-C. Do, R. Zory, and N. Roche, "Minimal detectable change of kinematic and spatiotemporal parameters in patients with chronic stroke across three sessions of gait analysis," *Human Movement Science*, vol. 64, pp. 101–107, Apr. 2019.