Edge-FPGA Deployment of DeepLabCut for Real-Time Human Pose Estimation: A Distributed INT8/FP32 Hybrid Inference Architecture

Mohamed Elmahlavy, Basim Elmashharavi, and Bahadır Çatalbaş

***Abstract*— Due to the high cost of commercial motion capture systems, their widespread utilization in research and teaching is limited. To the best of our knowledge, this paper presents the first successful deployment of the DeepLabCut ResNet-101 backbone to human pose estimation on the AMD Kria KV260 using a Xilinx DPUCZDX8G Deep Processing Unit (DPU). Since the transposed convolution prediction head of DeepLabCut is incompatible with the instruction set available on the DPU, the model is divided into two parts, with the first part running on the DPU with quantization to INT8 and executed at the edge of the network, with the transposed convolution prediction head running on a host PC in FP32, creating a hybrid inference pipeline via Gigabit Ethernet. The developed system operates at 13.1 FPS on a $250 KV260 board, with a peak joint-detection confidence of 0.92 on visible keypoints, which is significantly less expensive than compared embedded GPU platforms for this operation.**

***Index Terms*— FPGA, DeepLabCut, Deep Processing Unit (DPU), Real-time Human Pose Estimation, Distributed Inference, INT8 Quantization, Edge Computing, Model Partitioning.**

I. INTRODUCTION

Studio set-up systems like Vicon provide submillimeter accuracy but may be considered too expensive for many research and teaching laboratories. Executing complex pose estimation models purely in software on off-the-shelf CPUs introduces significant throughput bottlenecks, often precluding their use in low-latency closed-loop control [1]. Field-Programmable Gate Arrays (FPGAs) provide a reasonable compromise: re-programmable hardware acceleration at a fraction of the cost of GPU-based embedded systems.

However, it is difficult to transfer the latest pose estimation models to FPGAs. The Xilinx Deep Processing Unit (DPU) is the native neural network accelerator of the AMD FPGAs and provides support for standard convolution, pooling, batch normalization, and non-linear activation layers. However, the prediction head of DeepLabCut [2] relies on transposed convolution operations that are strictly unsupported by the DPU instruction set architecture. We are not aware of any published report of DeepLabCut deployment to an FPGA to date.

This paper has the following contributions. First, we find the incompatibility of the DPU with the direct deployment of DeepLabCut. We propose, in the second, a surgical partitioning method for such incompatibility, partitioning the model between the edge FPGA device and host CPU to realize a distributed INT8/FP32 hybrid inference pipeline. Third, we show that the system can achieve a throughput of 13.1 FPS with a hardware cost of only $250, and a joint detection confidence of 0.92 on visible keypoints. This solution is benchmarked against previously published embedded results [1] and is compared to be competitive with much more expensive solutions.

The remainder of this paper is organized as follows. Section II describes the system architecture. Section III details the FPGA deployment methodology. Section IV presents experimental results. Section V outlines ongoing and future work. Section VI concludes.

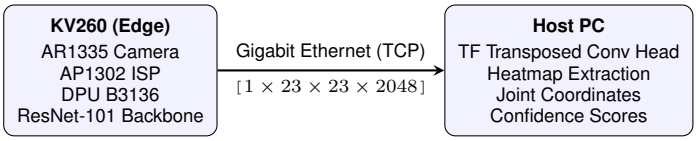

Fig. 1. Distributed inference pipeline architecture.

TABLE I
HARDWARE PLATFORM SPECIFICATIONS

Parameter	Value
Board	AMD Kria KV260
DPU Architecture	DPUCZDX8G_ISA1_B3136
DPU Clock	300 MHz
DPU Peak Throughput	1.882 TOPS
ARM CPU	Quad-core Cortex-A53 @ 1.2 GHz
DDR Memory	4 GB LPDDR4
Camera Sensor	AR1335 (MIPI CSI-2)
Host Connection	Gigabit Ethernet

II. SYSTEM ARCHITECTURE

A. Edge and Host Nodes

The system is an architecture of two computing nodes that are connected through Gigabit Ethernet, as illustrated in Fig. 1. The AMD Kria KV260 Vision AI Starter Kit (Node 1) is an edge node. The second node (PC host) processes the part of the model that cannot be processed by the AI accelerator and coordinate extraction.

The camera frames captured at the edge are NV12 format, with a resolution of 960×540, which is achieved using an ON Semiconductor AR1335 CMOS sensor and an AP1302 Image Signal Processor (ISP), with the ISP converting RAW to NV12 completely in hardware. OpenCV resizes the NV12 to 368×368 size, which it then converts to BGR for the DPU. The ResNet-101 backbone [3] is run on the DPUCZDX8G

All authors are with the Department of Control and Automation Engineering, Yildiz Technical University, Istanbul, Turkey. Emails: (mohamed.elmahlawy@std.yildiz.edu.tr, basim.elmashharavi@std.yildiz.edu.tr, bahadir.catalbas@yildiz.edu.tr)

B3136, yielding a feature tensor of size $[1 \times 23 \times 23 \times 2048]$. This is sent to the host PC through a TCP socket. The transposed convolution prediction head is realized on the PC using TensorFlow, and generates 14 heat maps of shape $[1 \times 46 \times 46 \times 14]$. These maps indicate the heat and are used for joint prediction. The split is because of the architectural limitation that is described in Section III. The hardware specifications are given in Table I.

B. Distributed Communication Protocol

There are two payloads per frame sent over the edge-to-host link: a compressed JPEG preview of the captured BGR image for visualization on the host, and the raw INT8 feature tensor generated by the DPU. For each TCP packet, it is organized in the following way:

$$\text{Packet} = \underbrace{[4\,\text{B}]}_{\text{Total Size}} + \underbrace{[4\,\text{B}]}_{\text{JPEG Size}} + \underbrace{[\text{JPEG}]}_{\sim 40\,\text{KB}} + \underbrace{[\text{INT8 Tensor}]}_{1{,}083{,}392\,\text{B}}.$$

Using TCP instead of UDP is because the loss of any single chunk is corrupting the downstream inference process – by removing any one chunk, a valid feature map becomes noise. The `TCP_NODELAY` is set on the socket to overcome the default packet batching behaviour that occurs with TCP. Small header packets are sent out immediately, without waiting for the coalescing of Nagle's algorithm.

This is approximately 14.7 MB/s per camera at the measured throughput of 13.1 FPS of NV12 video, versus 23.3 MB/s of uncompressed NV12 data for a 960×540 video size at 30 FPS. The reduction is moderate (about 37%), but is linear with the number of cameras and changes the multi-camera bandwidth picture significantly: 4 cameras of raw video are around 746 Mbps and overload a Gigabit link, but 4 edge-inferenced cameras together are still below 500 Mbps with plenty of bandwidth left for synchronization metadata and control traffic.

On the KV260, the end-to-end runtime is C++ based and consists of three concurrent threads that are linked with single-element drop queues. A capture thread fetches frames from the V4L2 input that is given by GStreamer and adds the latest BGR frame to the inference queue, replacing the previous BGR frame that hasn't been used yet. All of the *RunnerExt* instances are stored in the inference thread and passed to the next available *RunnerExt* to run. The above packet structure is then serialized in the network thread and sent to the TCP socket. The single-element drop queues prevent stalling of stale frames on the edge device when the host PC imposes backpressure, allowing rare frame drops in exchange for bounded end-to-end latency. A more detailed view of the dataflow is given in Fig. 2.

III. FPGA DEPLOYMENT METHODOLOGY

A. DeepLabCut Model and DPU Incompatibility

This work uses the DeepLabCut [2] *full_human* Model Zoo checkpoint [4], [5] which is based on a ResNet-101 [3] fully convolutional backbone. The model can be broken into two parts: a ResNet-101 backbone encoder that produces a condensed feature map, and a prediction head of transposed convolutional layers that upsample the feature map to joint heatmaps of original resolution. This work uses a pre-trained *full_human* Model Zoo model [4], [5] which estimates 14 anatomical keypoints.

The Xilinx DPUCZDX8G [6] is an accelerator that only supports the standard Conv2D, pooling, batch normalization, and activation operations. The DeepLabCut prediction head uses transposed convolution `Conv2DBackpropInput` which the DPU instruction set does not support. It was observed that the model does not work when using the Vitis AI toolchain [7] as the compilation aborted late in the mapping stage, with a `KeyError: 'shape'` exception being thrown at the transposed convolution nodes.

B. Surgical Graph Partitioning

This incompatibility is addressed by cutting the transposed convolution nodes from the compiled model, surgically. The graph is divided right after the last activation node in the ResNet-101 backbone, `resnet_v1_101/block4/unit_3/bottleneck_v1/Relu`. The backbone produces a compressed feature tensor of shape $[1 \times 23 \times 23 \times 2048]$ at this point. All following nodes of the transposed-convolution prediction head were pruned from the edge deployment and extracted using TensorFlow's `graph_util.extract_sub_graph` function. The location of the cut is determined by the model topology, i.e. the last activation shared by the convolutional encoder and the transposed-convolution prediction head that are not supported by the training, and is thus the most natural and likely the most lossless split point. The other model is the ResNet-101 backbone, compiled to the DPU [6]. Feed_dict injection is done at the stroke point on the original frozen graph to run the cut nodes of the prediction head on the host PC. INT8 quantization on FPGA, FP32 transposed convolution on PC results in an inference pipeline. The Vitis AI compiler [7] was able to break the 806-node backbone into 4 subgraphs, one of which is trivial to implement on CPU and the other 3 on the DPU.

C. Post-Training Quantization

The model is quantized from FP32 to INT8 using the Vitis AI 2.5 `vai_q_tensorflow` quantizer [7] and deployed on the DPU. Each tensor is given a fixed-point parameter that represents the number of fractional bits, so that the floating-point value is obtained by

$$x_{\text{float}} = x_{\text{int8}} \times 2^{-\text{fix\_point}}. \tag{1}$$

The relevant fixed-point parameters at the FPGA-host boundary are summarized in Table II. For the input tensor, fix_point $= -1$, giving a preprocessing scale of 0.5. Raw uint8 pixel values are scaled to INT8 as

$$x_{\text{in}} = \lfloor x_{\text{raw}} \times 2^{\text{fix\_point\_in}} \rfloor = \lfloor x_{\text{raw}} \times 0.5 \rfloor. \tag{2}$$

For the backbone output tensor, fix_point $= 3$, giving a dequantization scale of 0.125. The host PC recovers the FP32 feature tensor as

$$x_{\text{float}} = x_{\text{int8}} \times 2^{-3} = x_{\text{int8}} \times 0.125. \tag{3}$$

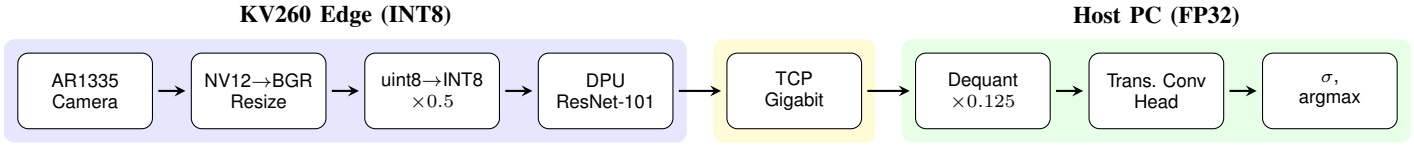

Fig. 2. Detailed dataflow through the INT8/FP32 pipeline. The ResNet-101 backbone runs INT8 on the DPU at the edge; the transposed convolution prediction head and post-processing run FP32 on the host.

TABLE II
QUANTIZATION PARAMETERS AT THE FPGA-HOST BOUNDARY

Tensor	fix_point	Scale
Input (`sub_inserted_fix_0`)	-1	0.5
Backbone output (`...Relu/aquant`)	3	0.125

One major finding in the calibration process [8]: using normalized pixel values, in the range $[0, 1]$, rather than raw uint8 values, in the range $[0, 255]$ would result in a maximum joint confidence of below 0.02 which effectively disables joint detection. For this reason, the DeepLabCut frozen graph has an internal mean subtraction node that requires raw uint8 values as input: if calibrated with normalized values, this node's activation distribution was $256\times$ smaller than the activation distribution for the runtime values, resulting in incorrect fix-point assignments across the graph. This quantizer should have the same input distribution as the runtime that is used.

D. Joint Coordinate Extraction

The dequantized backbone tensor is inserted at the cut point of the original TensorFlow graph and the 50 cut nodes are executed to generate joint heatmaps with a shape of $[1 \times 46 \times 46 \times 14]$. Each of the 14 channels corresponds to a skeletal point determined by the training annotations *full_human* used for the *full_human* dataset. For the lower body, these are the ankle, knee, hip1 and hip2 points, while for the upper body they are the wrist1, elbow1, shoulder1, shoulder2, elbow2 and wrist2 points, and for the head they are the chin and forehead points. Per joint confidence scores C_j are calculated as

$$C_j = \max_{x,y} \sigma(H_j(x, y)), \quad \sigma(z) = \frac{1}{1 + e^{-z}}, \qquad (4)$$

where H_j is the heatmap for joints. The pixel coordinates for the joint are computed as the spatial argmax of the sigmoid transformed heatmap, and then scaled to show the display resolution. Detects that have $C_j < 0.3$ are removed.

E. Edge-Host Workload Distribution

The proposed architecture is not a fully edge-bound design; the transposed convolution prediction head is deliberately executed on the host PC. This is not a workaround for the partitioning scheme but rather a deliberate architectural decision. The ResNet-101 backbone has about 42 GFLOPs per inference and about 44.5 million parameters, while the prediction head is a thin set of transposed convolutions on a $23 \times 23 \times 2048$ tensor that incurs about 2% of the total computational cost. If the majority of FLOPs are placed on the FPGA, the benefit of hardware acceleration can be achieved with almost full recovery of the available hardware acceleration benefit and without making the deployment difficult.

The PC-side computation, which consists mainly of dequantization, transposed convolutions, and argmax over heatmaps, is light enough to be emulated on the host PC if ported back to the KV260's application processor, the Cortex-A53. A more complete solution (and future direction) is to replace the transposed convolution layers with a series of nearest neighbor resize and subsequent Conv2D layers, both of which are supported by the DPU. This would enable the entire model to be implemented on the FPGA and complete the system to be edge-deployed.

F. Implementation and Software Stack

It is realized based on Vitis AI (version 2.5) for FPGA compilation and quantization. The DeepLabCut model is the public full-human checkpoint from the Model Zoo, and was kept in a frozen-graph format from TensorFlow 1.15 for the entire process of the partitioning step. The capture from the edge (capture) is done with a GStreamer pipeline that reads the data from the MIPI CSI-2 sensor, and OpenCV is responsible for converting the NV12 data to BGR and resizing it before passing it to the DPU input. The host-side transposed convolution prediction head and the sigmoid activation and argmax extraction run on a standard x86 Linux workstation, which does not need a GPU to run the host.

IV. EXPERIMENTAL RESULTS

A. Inference Throughput and Memory Bottleneck

The measured throughput of 13.1 FPS is significantly lower than the theoretical compute-bound limit of 44.4 FPS, which is computed based on the DPU's TOPS peak of 1.882 TOPS and the backbone's GFLOPs per inference of $\sim$42. This discrepancy suggests that the system is bandwidth-limited rather than compute-bound. The ResNet-101 backbone has 44.5 million INT8 parameters, while the DPUCZDX8G's 'Low' on-chip memory configuration [6] makes it extremely necessary to repeatedly access the DDR4 to retrieve weights and intermediate activations.

Single-runner throughput measured at 8.4 FPS, significantly less than the compute ceiling and with repeated weight reloads coming in at a major cost from DDR4. To counteract that, the *RunnerExt* instances are spawned in 3 different C++ threads to take advantage of the DPU's

pipelined memory architecture. This approach takes advantage of concurrent inferences to distribute the access cost of the memory transactions across multiple inferences, boosting throughput to 13.1 FPS. The per-runner speedup is sublinear, a 13.1/8.4 = $1.56\times$ improvement in throughput at 3 runners parallelism, which is consistent with the memory-bandwidth-limited interpretation: 3 runners share the same number of DDR4 channels to perform run parallel, but the number of channels does not scale proportionally to the number of runners. The results measured for the different configurations are summarised in Table III.

TABLE III
INFERENCE THROUGHPUT ACROSS PIPELINE CONFIGURATIONS

Configuration	FPS	Notes
1 runner, Python	8.4	Single-threaded, UDP
1 runner, Python + heatmap	6.4	Postprocessing overhead
3 runners, Python TCP	13.1	Multi-threaded, TCP
3 runners, benchmark only	13.2	DPU-only

B. Quantization Accuracy and Joint Confidence

Post-training INT8 quantization can introduce accuracy degradation [8], particularly in dense heatmap estimation. We present the results of the distributed pipeline using INT8/FP32 quantization for visible keypoints on the correct side for the correct calibration setting, as described in Section III, which is consistently > 0.92 for visible keypoints, far exceeding the rejection threshold of 0.3. Normalizing pixel values to $[0, 1]$ reduces the maximum confidence below 0.02, effectively disabling joint detection, whereas using the raw $[0, 255]$ range recovers the expected behavior. We should clarify that confidence should not be used as a replacement for an accurate benchmark: it is, however, a good sign that the quantization step is not silently destroying the learned heatmap structure.

C. Comparison with Published Embedded Benchmarks

Table IV lists the performance of the proposed system with respect to the results obtained by the embedded platform published by Kane et al. [1]. They use the light-weight ResNet-50 backbone to obtain their benchmarks. With the heavier ResNet-101 model, the $250 KV260 runs at 13.1 frames per second, over twice the frames per second for the TX2 running the ResNet-50 model, and competitive with Xavier, which runs for about a third of the price.

TABLE IV
COMPARISON WITH PUBLISHED EDGE PLATFORM BENCHMARKS [1]

Platform	Price	Model	FPS	Src
KV260 (this work)	$250	ResNet-101	**13.1**	Meas.
NVIDIA Xavier	$699	ResNet-50	19	[1]
NVIDIA TX2	$400	ResNet-50	6	[1]
Intel Xeon CPU	—	ResNet-50	4	[1]

V. ONGOING AND FUTURE WORK

In this current implementation, we test the end-to-end single-camera inference with the FPGA. This section provides an overview of the current and future research to be carried out on a multi-camera 3D motion capture system.

Multi-Camera 3D Triangulation. The first planned pipeline extension is a multi-camera, multi-KV260 version. There are several KV260 boards, located around the capture area, to send the 2D joint coordinates to the host PC through parallel TCP streams. Camera intrinsic and extrinsic parameters will be obtained by using ChArUco board calibration. In real time, the PC will employ weighted least-squares triangulation to map the 2D joint coordinate streams into real-world 3D coordinates.

Temporal Synchronization. Another problem in a distributed edge network is misaligned video frames between cameras (as a result of different capture times) which can produce ghosting artifacts in the triangulated 3D skeleton. The TCP packet header will include a hardware-level timestamp, with the host PC synchronizing the coordinates of received coordinate frames in accordance with the hardware-level timestamp before triangulation.

State Estimation and Jitter Filtering. Noisy 2D detections can be triangulated into 3D to get a jittery skeleton. On the host PC, the coordinate fusion loop will be implemented with a Kalman filter, which will smooth the joint trajectories by using motion history and generate stable 3D coordinates suitable for downstream control applications like robotic teleoperation.

Throughput Optimization via Dynamic Cropping. Dynamic cropping will be added to reduce the memory-bandwidth limitation to 13.1 FPS. Once the subject has been detected, the following DPU inputs will focus on a smaller area of interest around the subject and decrease the amount of DDR4 traffic per frame, which may increase effective throughput.

VI. CONCLUSIONS

To the best of our knowledge, this paper presented the first successful FPGA deployment of the DeepLabCut ResNet-101 backbone for human pose estimation. The central technical contribution is a surgical graph partitioning strategy that resolves a fundamental incompatibility between DeepLabCut's transposed convolution prediction head and the Xilinx DPUCZDX8G B3136 instruction set architecture. By splitting the model across the KV260 edge device and a host PC, a distributed INT8/FP32 hybrid inference pipeline is established that places 98% of the model's FLOPs on the FPGA while preserving the original heatmap structure.

With the more robust ResNet-101 backbone, the system runs at 13.1 FPS on a $250 AMD Kria KV260 board, more than double the performance of the baseline implemented by the $400 Jetson TX2 with the lighter ResNet-50 backbone, and at a much lower cost than the published embedded GPU baselines. As demonstrated in Section IV, the INT8 quantization step does not destroy the learned heatmap structure with a peak joint detection confidence of 0.92 for the visible

keypoints. Continuing efforts focus on multi-camera 3D triangulation, timestamped synchronization, Kalman-filter state estimation and dynamic input cropping to exceed the current DDR4 memory bandwidth limit.

CODE AVAILABILITY

The distributed INT8/FP32 hybrid inference pipeline described in this paper is open-source and publicly available, including source code, hardware configuration files, and deployment scripts. The entire repository is available on GitHub at: https://github.com/YTU-Mocap-Team/ZyncPose.

ACKNOWLEDGMENT

This work was supported in part by the AMD University Program.

REFERENCES

[1] G. A. Kane, G. Lopes, J. L. Saunders, A. Mathis, and M. W. Mathis, "Real-time, low-latency closed-loop feedback using markerless posture tracking," *eLife*, vol. 9, p. e61909, 2020.

[2] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, vol. 21, no. 9, pp. 1281–1289, 2018.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] The DeepLabCut Team, "DeepLabCut model zoo," [Online]. Available: http://modelzoo.deeplabcut.org, 2023, accessed: May 25, 2026.

[5] S. Ye, A. Filippova, J. Lauer, S. Schneider, M. Vidal, T. Qiu, A. Mathis, and M. W. Mathis, "SuperAnimal pretrained pose estimation models for behavioral analysis," *Nature Communications*, vol. 15, no. 1, p. 5165, 2024.

[6] *DPUCZDX8G Product Guide (PG338)*, AMD Inc., San Jose, CA, USA, 2022, [Online]. Available: https://docs.amd.com/r/en-US/pg338-dpu.

[7] *Vitis AI User Guide (UG1414) v2.5*, AMD Inc., San Jose, CA, USA, 2022, [Online]. Available: https://docs.amd.com/r/2.5-English/ug1414-vitis-ai.

[8] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," arXiv preprint arXiv:2106.08295, 2021. [Online]. Available: https://arxiv.org/abs/2106.08295