Design and Hardware Implementation of CNN-GCN Model for Skeleton-Based Human Action Recognition

[1]AMINE MANSOURI, [2]ABDELLAH ELZAAR, [1]MAHDI MADANI, [1]TOUFIK BAKIR

[1]ImViA Laboratory
University of Burgundy
21000 Dijon
FRANCE

[2]Laboratory of R&D in Engineering Sciences
Abdelmalek Essaadi University
93000 Tetouan
MOROCCO

Abstract: Deep learning algorithms in general have shown outstanding performances on tasks involving Human Action Recognition and spatio-temporal modeling. In our ever-more interconnected world where mobile and edge devices take center stage, there is a growing need for algorithms to operate directly on embedded platforms. Field-Programmable Gate Arrays (FPGA), owing to their reprogrammable nature and low-power attributes, stand out as excellent choices for these edge computing applications. In this work, our aims are threefold. Firstly, we aim to design and develop a novel custom model that combines Convolutional Neural Networks (CNNs) and Graph Convolutional Networks (GCNs) to effectively capture spatial and temporal features from skeleton data to achieve Human Action Recognition. Secondly, we endeavor to implement this custom model onto FPGA hardware using Vitis-AI, thus exploring the potential for efficient hardware acceleration of deep learning models. Lastly, we seek to evaluate the real-time performance of the FPGA-accelerated model in comparison to CPU and GPU implementations, assessing its suitability for deployment in real-world applications requiring low-latency inference. Experimental results based on the NTU dataset demonstrate the efficiency and accuracy of our custom model compared to traditional CPU or GPU implementations.

Key-Words: Convolutional Neural Networks, Graph Convolutional Networks, Human Action Recognition, Real-time Applications, Field-Programmable Gate Arrays, Vitis-AI.

Received: April 19, 2023. Revised: April 6, 2024. Accepted: May 9, 2024. Published: June 4, 2024.

1 Introduction

In recent times, there has been a growing need for smart applications in both research and business. Artificial Intelligence (AI), especially Deep Learning (DL), is being used in various areas like healthcare, industry, education, and safety. These applications bring clever solutions that easily become part of our daily routines. Importantly, the impact of AI and machine learning is also seen in Human Action Recognition (HAR), a pivotal task in computer vision, that has witnessed a surge in demand for real-time and efficient solutions across various applications, ranging from surveillance systems to human-computer interaction. The motivations for this present work are straightforward, we need a real-time HAR application, running in resource-constrained hardware. Our hardware choice was Field-Programmable Gate Arrays (FPGA) after comparing the performances of our proposed model between FPGA with CPU and GPU implementations. The HAR model running inside the FPGA board was custom-made combining Convolutional Neural Networks (CNN) and Graph Convolutional Networks (GCN) concepts for an efficient spatio-temporal feature extraction. This proposed model was built in such a way that it won't cause issues in the compilation process while using Vitis-AI to generate the Deep Learning Processing Unit (DPU) instructions. Implementing a HAR task within an embedded system such as FPGA presents a significant challenge, particularly when aiming for real-time functionality. The prevailing strategy for minimizing hardware usage involves adopting model compression methods, including weight quantization

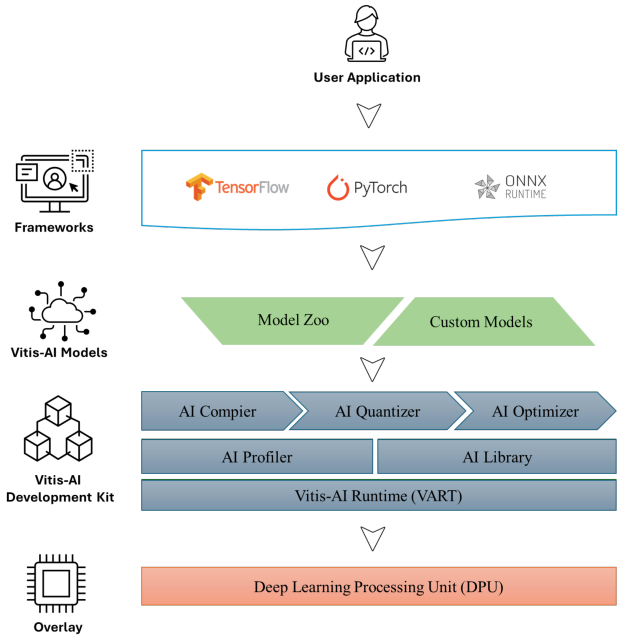

Figure 1: Vitis-AI environment structure, [1].

that shifts from floating-point 32-bit (FP32) to 8-bit integer (INT8) representations, along with network pruning. The second challenge involves crafting a DL network that aligns with hardware constraints and operates within milliseconds (ms) to achieve real-time performance. To build our proposed model, we investigated the concepts of CNNs and GCNs. CNNs have proven highly effective for the aforementioned applications in capturing spatial features, while GCNs have shown promise in modeling complex relationships within graph-structured data. In section 3 we dig into the subtleties of our proposed custom model's architecture, detailing the synergistic connection between CNNs and GCNs that aims to exploit both spatial and temporal information present in skeleton data. Our suggested approach (DL model) performs noticeably better in a variety of applications than existing CNN-based DL models or earlier Machine Learning (ML) models like Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), and Restricted Boltzmann Machines (RBMs). Furthermore, each of these methods has its shortcomings: CNNs are ineffective at capturing long-term temporal correlations, SVM is ill-suited for noisy data, RF and DT can be computationally costly, and RBMs are challenging and slow to train. For this project, we present an endeavor to fuse the strengths of CNNs and GCNs for skeleton-based human action recognition, implemented on FPGAs. Our primary hardware choice is the Xilinx Kria KV260 FPGA board. Consequently, our framework selection aligns with this board, opting for the Vitis-AI tool (see Fig. 1), an open-source and actively maintained library by Xilinx. We also discuss in this paper the nuances of the Vitis-AI tool, showcasing its role in seamlessly interfacing with FPGA hardware. Additionally, we provide insights into the training process on the NTU RGB+D 60 dataset, [2], highlighting the model's robustness and generalizability.

The contributions of the study reported here are multiple:

- To reduce latency and increase energy economy, we introduce an FPGA-based acceleration of a skeleton-based HAR (time-series) custom model, ResGCN-T, using a single SoC.
- To the best of our knowledge, this is the first time we've attempted to build a custom spatio-temporal GCN-based model (called ResGCN-T) for time-series Human Action Recognition, utilizing Xilinx Vitis AI and DPU. Also, we used the NTU large-scale dataset to train and evaluate it.
- We compared the performances between CPU, GPU, and FPGA-DPU implementations, providing a comprehensive analysis of the efficiency and speed of the ResGCN-T model and showcasing the advantages of leveraging FPGA-DPU for time-series Human Action Recognition.

These contributions collectively contribute to the ongoing efforts to advance the capabilities of embedded systems, particularly in the context of edge computing applications using FPGA technology.

2 Related work

Convolutional Neural Networks (CNNs): Previous research in the area of human action recognition has prominently featured CNNs. These CNNs have demonstrated their prowess in extracting spatial features from video frames, enabling effective action classification. Notable works, such as [3], [4], have explored the application of CNNs in this context, establishing a foundation for subsequent advancements.
Recurrent Neural Networks (RNNs) with emphasis on Long Short-Term Memory (LSTM): In addressing the temporal dynamics inherent in time-series data, RNNs, specifically LSTM and Gated Recurrent Units (GRUs) networks, have gained significant attention. The sequential nature of human actions necessitates capturing temporal dependencies, making LSTM/GRU an appealing choice. Prior studies, [5], [6], have delved into the integration of LSTM and GRU networks for improved modeling of temporal aspects in human action recognition.
Graph Convolutional Networks (GCNs): Extending beyond the conventional CNNs, GCNs have

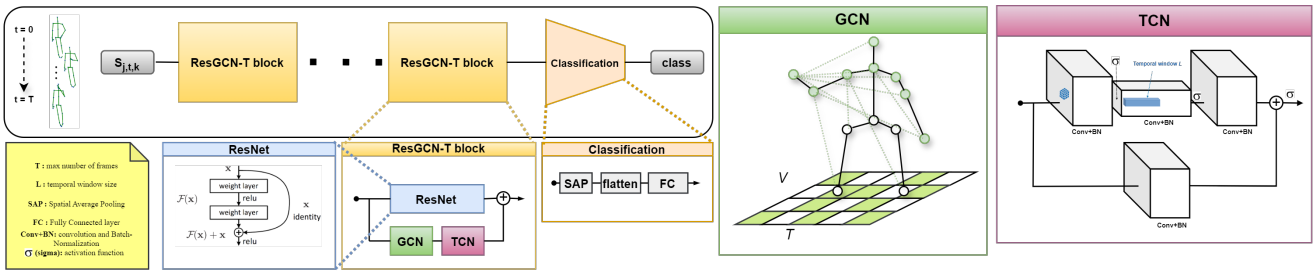

Figure 2: Framework of the proposed end-to-end Resnet {GCN+TCN}-based Network (ResGCN-T). It consists of consecutive spatio-temporal ResGCN-T blocks. In the ResNet block, we learn the dynamics representation of skeleton joints (input data $S_{j,t,k}$). We utilize the GCN module to model the dependencies of the skeleton joints. We make use of the TCN module to model the dependencies of frames.

emerged as powerful tools for modeling relationships in graph-structured data, making them particularly relevant for skeleton-based action recognition. Recent works, [7], [8], [9], have explored the application of GCNs in conjunction with traditional CNNs to capture both spatial and temporal dependencies in complex human actions, presenting a promising avenue for improved recognition accuracy.

FPGA Implementation: As the demand for real-time and power-efficient solutions escalates, researchers have turned their attention to implementing deep learning models on FPGAs. Leveraging the reprogrammability and parallel processing capabilities of FPGAs, recent studies, [10], [11], [12], [13], [14], have investigated the deployment of different types of DL models for human action recognition or other related tasks accelerating inference on edge devices. A multimodal DCNN network called SensorNet was introduced by [13], to process multimodal time series signals as input for classification. The Artix-7 FPGA was used to execute the model and evaluate a number of applications, including HAR and stress detection. At 100 MHz, the HAR application consumed 175 mW of power while achieving 98 % accuracy. Furthermore, [14], constructed a model comparable to SensorNet in a different study that concentrated on HAR. However, by applying an LSTM instead of CNN, taking numerous sensor modalities as input. The accuracy of their architecture decreased to 87.17% with the PAMAP2 dataset, [15], although achieving a power consumption of 82 mW at 160 MHz. To handle accelerometer and gyroscope data for HAR, [10], used the DeepSense network, [16], a time-series multimodal DL architecture that combines GRU cells with CNNs. They also suggested a co-design method for improved performance in the Xilinx Vitis-AI development environment. When compared to the initial baseline, the study improves latency and energy usage by up to 2.5 and 5.2 times, respectively.

When compared to our proposed CNN-GCN-based method for action recognition inference time, [10], achieved a latency of 24 milliseconds during inference testing, whereas our model demonstrates significantly lower inference times, as shown in Table 3 (ranging from 6.8 to 19.4 milliseconds). Furthermore, our proposed model operates entirely on the DPU, unlike theirs, which relies on both DPU and CPU due to Vitis-AI limitations regarding RNN implementation. This difference can be viewed as another improvement. In comparing the accuracy performances on the NTU dataset, [2], our model outperforms the approach proposed by [17]. Their model, which utilizes a one-dimensional convolutional neural network (1D-CNN) implemented on SMIC CMOS technology hardware, achieves accuracy rates of 80.7% and 87.8% in the Cross-Subject and Cross-View benchmarks respectively. In contrast, our model demonstrates significantly better performance across the same dataset, as evidenced in Table 2. These results highlight the effectiveness of our proposed CNN-GCN method capturing spatio-temporal features to achieve the skeleton-based HAR tasks.

This section provides a comprehensive overview of relevant works, from the foundational CNNs to temporal modeling with RNNs, the synergy of CNNs and GCNs, and the recent advancements in FPGA-based implementations for human action recognition. The subsequent sections of this article build upon these foundations, exploring novel approaches for enhanced performance and efficiency in skeleton-based action recognition.

3 model explained

We present the **Res**Net {**GCN**+**T**CN}-based Network (ResGCN-T) showed in Fig. 2, a deep learning architecture that uses a series of Spatial-Temporal modules in conjunction with relevant information to improve the model's understanding of human body dynamics. Furthermore, this model includes a succession of spatio-temporal blocks (ResGCN-T blocks) in the primary stream and ends with a classification block. By focusing on specific regions of interest, the ResGCN-

T blocks can extract meaningful spatio-temporal information with effectiveness. The features that result after each ResGCN-T block's fine-grained processing are sent to the classification block as the last stage stream, where they help to produce action classification probabilities. End-to-end training is applied to the complete network to optimize the classification score and improve performance.

Given the input set represented as $S = S_{j,t,k}$, where the indices are defined as follows: The joint index is denoted by *j*, the frame index by *t*, and the coordinate axis by *k* (x, y, and z for 3D coordinates). Frame index *t* falls in the range $1, ..., T$, where the maximum number of frames in a series is constituted by *T*. In this instance, we have chosen $T = 25$ as a balance between the accuracy metric and the data volume of the input. The value of *T* is established through experimentation. The dynamics of each joint position *k* of type *j* at time *t* can be characterized. Dynamics is the three-dimensional coordinate system (x, y, and z) of a joint.

3.1 CNN-ResNet module

[18], in their research dealt with an important question in DL architectures. A degradation issue has been revealed when deeper networks are able to begin converging: accuracy increases with network depth until it becomes saturated, at which point it rapidly declines. Surprisingly, overfitting is not the cause of this degradation, and as shown in [19], and extensively confirmed by their experiments in [18], adding additional layers to a suitably deep model results in increased training error. It is evident from the training accuracy degradation that not all systems are as simple to optimize.

A deep residual learning framework called ResNet was introduced as a solution to the degradation issue. The authors specifically allowed each of the several stacked layers to fit a residual mapping rather than assuming that each one would fit a specified underlying mapping (see the module ResNet in Fig. 2). Formally, they let the stacked nonlinear layers match another mapping of $F(x) = H(x) - x$, identifying the intended underlying mapping as $H(x)$. They recast the original mapping into $F(x) + x$. They postulated that compared to the original, unreferenced mapping, it is simpler to optimize the residual mapping.

3.2 GCN module with pre-defined Adjacency matrix

A skeleton sequence can be shown in a variety of ways. In continuous frames, it can be represented by the joints' or nodes' 2D or 3D coordinates. The GCN block employed in this article draws inspiration from the ST-GCN model, [7], it builds structural and hierarchical forms conveying more precise skeleton data using a spatial-temporal graph. Put otherwise, a spatial-temporal graph designated $G = (V, E)$ $\{V : joints/nodes, E : edges\}$ will be constructed by the V (V: Nodes) intra-body joints that create a skeleton spanning T frames (duration) and conceive the inter-frame connection.

In contrast to 2D or 3D convolutions, a graph-based convolution (GCN) requires a few steps in between to be implemented. An equivalent implementation of GCNs, as described in [20], is obtained with the ResGCN module. An adjacency matrix *A* relating to self-connections of the nodes/joints and an identity matrix *I* representing the connections of body joints in the same frame. To establish the propagation rule for information across the graph, a crucial step in conceptualizing GCNs, in [20], suggested the following formula in Equation 1.

$$f_{out} = \sigma(\hat{D}^{-\frac{1}{2}} \cdot \hat{A} \cdot \hat{D}^{-\frac{1}{2}} \cdot f_{in} \cdot W) \qquad (1)$$

Here, the input and output feature maps are denoted by f_{in} and f_{out}, respectively. The adjacency matrix with self-connections of the undirected graph $\hat{D} = \sum(\hat{A}_{ij})$ is $\hat{A} = A + I$. In this case, the weight matrix with trainable parameters is denoted as *W*. The *ReLU* non-linear activation function is σ. Under the spatial-temporal scenario, the input feature map is actually represented by a tensor of dimensions *(C, T, V)*. A typical 2D convolution $conv(f_{in}) = f_{in} \cdot W$ is used to perform the graph convolution, and the tensors that result are multiplied to obtain the normalized adjacency matrix $\hat{D}^{-\frac{1}{2}} \cdot \hat{A} \cdot \hat{D}^{-\frac{1}{2}}$.

In a notable modification to our program, we introduced a strategic alteration to the Equation 1, specifically replacing matrix multiplication, commonly known as outer-product multiplication, with an element-wise product. The new term is $\hat{D}^{-\frac{1}{2}} \cdot \hat{A} \cdot \hat{D}^{-\frac{1}{2}} \odot f_{in} \cdot W$, with $\odot$: denotes element-wise matrix multiplication. This adjustment was necessitated by the manner of outer-product matrix multiplication performed within the FPGA DPU.

3.3 Temporal module TCN

In the context of skeleton-based action recognition, the Temporal Convolutional Network (TCN) module serves to model the temporal evolution of human actions over time. It is responsible for analyzing the sequence of skeletal poses captured at consecutive time steps and extracting relevant temporal features. We used a temporal module inspired by the methods presented in [18], and designed in [21], to recognize activities of different durations efficiently.
To decrease the number of feature channels in the convolution computation throughout a reduction rate *r*,

Figure 3: Vitis-AI workflow.

in [18], the authors proposed an elegant block structure called Bottleneck, which includes two convolutional layers with a kernel 1×1 before and after the regular convolution layer, respectively. In this study, we achieve a considerably quicker implementation of model training and inference by utilizing the Bottleneck structure within the spatial and temporal modules.

Assume that the temporal window size L is 9, the channel reduction rate r is 4, and the input and output channels are both 128. The Bottleneck block has approximately 8.5 times fewer parameters than the basic block, with just $128 \times 32 + 32 \times 32 \times 9 + 32 \times 128 = 17,408$, compared to $128 \times 128 \times 9 = 147,456$ in the basic block.

4 Xilinx Vitis AI Development Environment

For AMD boards, PCs, laptops, workstations, and some Alveo data center accelerator cards, the Vitis AI software offers a complete AI inference-building solution. A vast array of AI models, deep learning processing unit (DPU) cores that have been tuned, tools, libraries, and sample designs for AI at the edge and data centers are all included. It fully utilizes AI acceleration on AMD SoCs and RyzenTM PCs, thanks to its great efficiency and user-friendly design (see Fig. 1.

Currently, two primary methods are used to target FPGA devices: (1) Using HLS (High-Level Synthesis) language, which is C/C++-based language. The researcher has to create manually the architecture of the Neural Network (NN). (2) The second approach involves the automatic generation of hardware through software integration with machine learning frameworks such as TensorFlow and PyTorch. The second approach is becoming more frequently used and becomes the standard for increased design productivity as NN models get bigger and more complicated. On the other hand, since hardware optimizations and parameters have a significant impact on latency and energy efficiency, designers must carefully set them up.

For this paper, we opt for the second approach by harvesting the Xilinx Vitis-AI tool and we use PyTorch to create our model. Both software and hardware deployment procedures will be covered by Vitis-AI. That is, optimization, quantization, and compilation in the following manners (see Fig. 3): **1) Vitis-AI-Optimizer:** With little influence on accuracy, Vitis-AI-Optimizer will compress the model by cutting its complexity by five to fifty times. **2) Vitis-AI-Quantizer:** By calibrating and converting our floating-point 32-bit FP32 model into a fixed-point 8-bit INT8 model, Vitis-AI-Quantizer provides speed and computational efficiency. The generated model will seek less memory usage/bandwidth. **3) Vitis-AI-Compiler:** The model will be mapped by Vitis-AI-Compiler to a set of DPU-understandable instructions and data flow. The compiler identifies which operations are supported by the DPU core and assigns unsupported operations to the CPU.

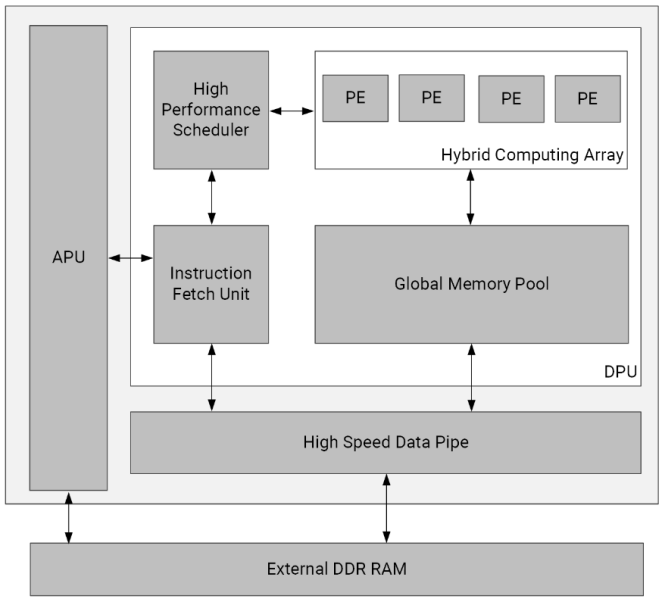

Figure 4: Top-Level overview of the DPUCZDX8G. APU - Application Processing Unit. PE - Processing Engine. DPU - Deep Learning Processing Unit, [1].

4.1 DPU

The Vitis AI solution's Deep Learning Processing Units (DPUs) are an essential part (see Fig. 4. A soft accelerator aimed at deep learning inference is called a DPU. It could also refer to several possible accelerator systems that support various network topologies. A DPU can be built as a collection of micro-coded functions that are implemented on the FPGA AI Engine, or it can consist of components that are available in the FPGA programmable logic fabric, such as DSPs (Digital Signal Processors), LUTs (Look-Up Tables), Flip-Flops, BlockRAM, and UltraRAM. The following graphic shows an example of the **DPU**CZDX8G, which targets Zynq Ultrascale+ MPSoC device (i.e., the board Kria-KV260 utilized for this paper). The user can deploy and process several CNNs and streams simultaneously using a sin-

Table 1: Evaluating performance across our proposed network configurations for identifying the best approach, in terms of accuracy (%), using the NTU 60 dataset on CS benchmark.

State	Model name	Model architecture			Approach	Hardware	Parameters (M)	Accuracy (CS %)
		ResNet	GCN	TCN				
1.1	(GCN+TCN)	-	✓	✓	Only	Local	**0.8**	81.74
1.2.1	Resnet12-Bo	✓	-	-	Only	Local	3.4	84.26
1.2.2	Resnet18	✓	-	-	Only	Local	9.8	84.78
2.1	Resnet8-Ba + (GCN + TCN)	✓	✓	✓	Serial	Cloud	5.3	83.93
2.2	Resnet12-Bo + (GCN + TCN)	✓	✓	✓	Serial	Cloud	4.3	83.99
2.3.1	Resnet18 + (GCN + TCN)	✓	✓	✓	Serial	Cloud	10.8	83.52
2.3.2	Resnet18 + (GCN + TCN)	✓	✓	✓	Serial	Local	10.8	83.59
2.4	Resnet50 + (GCN + TCN)	✓	✓	✓	Serial	Local	13.2	83.98
2.5	Resnet152 + (GCN + TCN)	✓	✓	✓	Serial	Cloud	32.8	84.32
3.1.1	Resnet12-Bo // (GCN + TCN)	✓	✓	✓	Parallel	Cloud	4.2	84.47
3.1.2	Resnet12-Bo // (GCN + TCN)	✓	✓	✓	Parallel	Local	4.2	84.58
3.2	Resnet18 // (GCN + TCN)	✓	✓	✓	Parallel	Local	10.6	85.12
3.3	Resnet50 // (GCN + TCN)	✓	✓	✓	Parallel	Local	13.0	85.37
3.4	Resnet152 // (GCN + TCN)	✓	✓	✓	Parallel	Cloud	32.7	**85.52**

gle DPU core or instance in a design. The size of the DPU can be adjusted to meet the needs of the user. Processing is dependent on the DPU's ability to support the number of streams and network combinations with enough parallelism. A device can instantiate more than one DPU instance. The NN model is compiled with Vitis-AI-Compiler to an instruction-specific set inside an executable *.Xmodel* file. The DPU executes the *.Xmodel* to deploy the NN model within the FPGA card. The fundamental chores of scheduling numerous streams, different networks, or even multiple DPU instances if necessary, are taken care of by the Vitis AI Runtime. The DPU may contain several engines, each tailored to handle a particular task. For example, a dedicated PE Processing Engine accelerates the Conv2d layers, while a separate process handles depthwise convolutions.

5 Experiments

5.1 Model results

We conducted a series of experiments utilizing the NTU RGB+D 60 dataset to assess the performance of our proposed network configurations and validate the effectiveness of our approach. The Table 1 presents the results of these experiments, showcasing the accuracy (%) achieved by various network configurations on the CS benchmark. Each configuration is labeled with a state and model name, and its corresponding model architecture, approach, hardware deployment, number of parameters (in millions), and accuracy on the CS benchmark are provided. Our aim was to identify the most effective approach for real-time Human Action Recognition, considering different combinations of ResNet, GCN, and TCN architectures. The results highlight the significant impact of architecture choices and deployment strategies on the accuracy of our HAR model, providing valuable insights for optimizing performance in resource-constrained environments.

The first investigation done using the information in Table 1 is about proving the efficiency of the hierarchical architecture chosen for our final model. Useful to know that we exploited three approaches while building the networks. The first approach is using only ResNet architecture or GCN+TCN modules (denoted *Only* in Table 1). The second one is using ResNet layers in serial to the GCN and TCN modules (denoted *Serial* in Table 1). The third approach is the same as the second approach but the ResNet layers are in parallel to the GCN and TCN modules (denoted *Parallel* in Table 1).

Here, the network "Resnet12-Bo + (GCN + TCN)" denotes a hierarchical formation where the baseline is a ResNet Block followed by the modules GCN and TCN. This baseline will be repeated as shown in Fig. 2. The symbols "//" and "+", denote respectively "parallel" and "serial" combinations between the blocks/modules. Note that the modules GCN and TCN are always serial in this research, which is why they are always inside parenthesis "(GCN+TCN)". We tested several ResNet architectures under the name ResnetX-Y. Here "X" is the number of layers and takes the value (8, 12, 18, 50, or 152), "Y" is the type of layer used, and it can be either Basic-layer "Ba" or Bottleneck-layer "Bo". We notice the subsequent observations taken from the Table 1:
a: from the state 1.1, using only the GCN and TCN modules to construct the network. It gives us an accuracy of 81.74% which is relatively a similar score as the ST-GCN, [7], since the GCN implementation for this paper was inspired by this ST-GCN model. There is a 0.24% improvement compared to the orig-

inal ST-GCN model with fewer learning parameters, this improvement of accuracy and less weight is due to the addition of the TCN module, we used 3 consecutive modules only compared with 10 consecutive st-gcn blocks in [7].

b: From comparing the states 1.2.1 and 1.2.2, or from states 3.1.1 to 3.4, we notice that the more we add learnable parameters (building a denser network) the more accuracy we get. Thus, better performance. For example, if comparing state 3.1.1 with 4.2 million parameters to state 3.4 reaching 32.7 million parameters, the accuracy on the CS benchmark is 84.47% and 85.52%, respectively.

c: If we compare the types of approaches used to train the networks, we observe that the *Parallel* networks perform better than the *Serial* networks. Concretely, comparing the states 2.2 and 3.1.2, there is a difference of 0.57% accuracy in favor of *Parallel* state 3.1.2. A logical reasoning for this improvement resides under the fact that by processing input data through multiple parallel pathways (in our case, ResNet layers in parallel to GCN+TCN modules), the model can capture a richer set of features from the input data. Each parallel pathway may specialize in extracting different types of features, leading to a more diverse and comprehensive feature representation. In addition, parallel convolution blocks can act as a form of regularization by introducing additional pathways for feature extraction. This can help prevent overfitting by promoting generalization and reducing the reliance on individual convolutional pathways.

d: comparing the approach *Only* with *Parallel* showcases the fact that our contribution implementing the concepts GCNs in addition to TCNs improves the overall performance of the ResGCN-T baseline (see Table 2).

e: We used two types of GPUs to train the networks, one is on a *Local* machine and the other one is on the *Cloud* throughout the CCUB [1] (the university cluster, utilizing a GPU Tesla V100s card). The study shows that training the networks on our local machine (GPU NVIDIA RTX 3070) it gives slightly better performance. If we compare the same architecture used in states 3.1.1 and 3.1.2 but the first is trained on the *cloud* and the other on our *Local* GPU, we find a 0.11% difference in accuracy. This is not much, but, in some cases, it can make a difference. Our findings demonstrate that the increase in performance can be attributed to the utilization of recent GPU models on our *Local* machine compared to those deployed on the CCUB *Cloud* infrastructure. However, The CCUB *Cloud* GPUs have more significant RAM capabilities, which allowed us to train the state 3.4 model with its 32 million parameters.

[1]Calculations were performed using HPC resources from DNUM CCUB (Centre de Calcul de l'Université de Bourgogne)

Table 2: In terms of accuracy (%), the performance of the presented approach ResGCN-T was compared with SOTA methods on the Cross-Subject (CS) and Cross-View (CV) benchmarks of the NTU 60 dataset.

Method	CS	CV
HBRNN-L, [22].	59.1	64.0
Part-Aware LSTM, [2].	62.9	70.3
ST-LSTM + Trust Gate, [23].	69.2	77.7
STA-LSTM, [24].	73.4	81.2
GCA-LSTM, [25].	74.4	82.8
Clips+CNN+MTLN, [26].	79.6	84.8
VA-LSTM, [27].	79.4	87.6
ElAtt-GRU, [28].	80.7	88.4
ST-GCN, [7].	81.5	88.3
DPRL+GCNN, [29].	83.5	89.8
SR-TSL, [30].	84.8	**92.4**
HCN, [31].	**86.5**	91.1
ResGCN-T-12-Bo	84.5	91.5
ResGCN-T-18	85.1	91.2
ResGCN-T-50	85.3	**91.6**
ResGCN-T-152	**85.5**	**91.6**

5.2 Comparison with the State-Of-The-Art methods

Using the NTU 60 dataset, we provide a comprehensive examination of existing State-Of-The-Art (SOTA) methods with our suggested model/baseline, ResGCN-T, in the following section. You may find the evaluation results in Table 2. Notably, to attain better performance, our strongest baseline, called "Resnet152 // (GCN + TCN)", combines temporal convolution-based and graph-based concepts.

The significant impact of using the spatial-temporal ResGCN-T architecture with GCNs and TCNs is seen in Table 2. Notably, prominent CNN-based and RNN-based techniques are represented by the "Clips+CNN+MTLN" and "ElAtt-GRU" approaches, [26], [28], respectively. By contrast, ResGCN-T outperforms them, with accuracy margins of 5.9% and 4.8% for the CS benchmark, respectively.

We compare our proposed method to other GCN-based approaches such us ST-GCN, [7], which was the first model that introduced the concept of GCNs into HAR and laid the foundation for subsequent prominent graph-based works. Compared to ST-GCN we observe an improvement of 4% and 3.3% in respectively CS and CV benchmarks, in favor of the ResGCN-T model. DPRL+GCNN architecture, [29], combining a deep progressive reinforcement learning method empowered with GCNs for an effective skeleton-based action recognition, this is another model utilizing GCNs and our proposed model shows better performances with a difference of 2% and 1.8% on respectively CS and CV benchmarks.

Table 3: Inference Time Comparison for CPU, GPU, and DPU on the NTU CS Benchmark evaluation set.

State	Running all Test set			Running one sample		
	CPU time (s)	GPU time (s)	DPU time (s)	CPU time (s)	GPU time (s)	DPU time (s)
3.1.2	24 318	51	113	1.4	3.0×10^{-3}	6.8×10^{-3}
3.2	59 299	89	257	3.5	5.3×10^{-3}	15.5×10^{-3}
3.3	100 891	129	321	6.1	7.8×10^{-3}	19.4×10^{-3}
3.4	128 134	184	-	7.7	11.1×10^{-3}	-
Power consumption	-	GPU$\leq$ 220	DPU$\leq$ 36			

Moreover, compared with the model HCN, [31], that exploits the concept of Co-occurrence Feature Learning with CNN, this model performs better than ours on the CS benchmark with a difference of 1%, but the percentage lost on CS is slightly regained on the CV benchmark with a difference of 0.5% to our favor.

As an explanation for the superiority of our proposed scheme compared with the mentioned SOTA models, we believe that CNNs are effective in capturing spatial features from input data, while RNNs are good at modeling temporal dependencies over sequential data. However, for tasks like skeleton-based action recognition, where both spatial and temporal information are crucial, the combination of CNN and GCN can leverage the strengths of both architectures to capture both spatial and temporal dependencies simultaneously. GCNs are specifically designed to operate on graph-structured data, such as skeleton data. By treating the skeleton data as a graph and applying GCN layers, the model can effectively capture the relational information between joints, which is essential for understanding human actions. Another perspective is that CNNs are known for their ability to learn hierarchical features through convolutional layers. By integrating GCN layers into the architecture, the model can learn hierarchical representations not only in the spatial domain (through CNNs) but also in the relational structure of the data (through GCNs), leading to more informative and discriminative features. On the other hand, RNNs can suffer from vanishing or exploding gradient problems, especially with long sequences, which can lead to overfitting or poor generalization. By incorporating CNNs and GCNs, which are less prone to such issues, the model can mitigate overfitting and improve generalization performance.

5.3 Inference time (processing speed)

To assess the inference time of our proposed model through different hardware components (CPU, GPU, and FPGA's DPU), we conducted the following experiment: We tested different states (3.1.2, 3.2, 3.3, and 3.4) using the NTU CS test set with 16487 samples. We run the different networks (states) on CPU, GPU, and then DPU to measure the time (in seconds) spent predicting the test set, we report the results in Table 3. In general, lower inference times indicate faster model speeds, meaning the model can process data more quickly and efficiently. From Table 3, it's evident that the GPU generally offers faster inference times compared to both the CPU and DPU across different states, suggesting that it may be the preferred choice for inference tasks in this context. However, it's essential to consider other factors such as cost, power consumption, and hardware availability when choosing between CPU, GPU, and DPU for inference tasks. The explanations below will clarify more this point of view. If we compare only from a numerical perspective, to process the test set samples/data on the CPU under state 3.1.2, it takes 24328 s (6.7 hours) in total, or 1.4 s latency to process one sample (one action). In contrast, the DPU takes 6.8 ms, and the GPU takes 3 ms. However, we believe that the best approach is to use the FPGA's DPU. The reasons behind this proposition depend on the fact that:
1) The data is still processed in a few milliseconds even if the application is 2x faster with the GPU compared with the DPU, which means 6.8 ms latency is enough to provide decent real-time performance. 2) The power consumption of the FPGA board (Kria KV260) is just 12 V (volts) with a maximum of 3 A (amps) of current, which gives us a P=UxI=12x3=36 W (watts) as the max theoretical value, in practical it is only sipping around 7 W. However, our GPU model has a power consumption that can increase up to 220 W.
To summarize, the FPGA consumes less power and provides decent inference time (speed) compared with the GPU.

6 Conclusion

In this work, we detailed the development of a custom Deep Learning model leveraging Convolutional Neural Network (CNN) and Graph Convolutional Network (GCN) concepts, specifically tailored for implementation on Field-Programmable Gate Arrays (FPGAs) using the Vitis-AI development tool. The FPGA-DPU latency performances have demonstrated

remarkable efficiency, surpassing CPU capabilities by a significant margin of 202 times, and exhibiting latency levels comparable to GPU processing, with differences ranging from just 3 to 12 milliseconds. Notably, while achieving impressive performance metrics, FPGAs exhibit minimal power consumption, consuming less than 36 watts, in contrast to the considerable power requirements of GPUs (around 220 watts). Our findings not only contribute to the expanding field of FPGA-accelerated deep learning but also provide a valuable perspective on the feasibility of deploying custom models for real-time skeleton-based human action recognition.

References:

[1] Xilinx, "Dpuczdx8g for zynq ultrascale+ mpsocs product guide (pg338)," *https://docs.xilinx.com/r/en-US/pg338-dpu*, 2023.

[2] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1010–1019, 2016.

[3] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive neural networks for high performance skeleton-based human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1963–1978, 2019.

[4] A. Banerjee, P. K. Singh, and R. Sarkar, "Fuzzy integral-based cnn classifier fusion for 3d skeleton action recognition," *IEEE transactions on circuits and systems for video technology*, vol. 31, no. 6, pp. 2206–2216, 2020.

[5] S. Li, W. Li, C. Cook, and Y. Gao, "Deep independently recurrent neural network (indrnn)," *arXiv preprint arXiv:1910.06251*, 2019.

[6] C. Li, C. Xie, B. Zhang, J. Han, X. Zhen, and J. Chen, "Memory attention networks for skeleton-based action recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4800–4814, 2021.

[7] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[8] Y.-F. Song, Z. Zhang, C. Shan, and L. Wang, "Constructing stronger and faster baselines for skeleton-based action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[9] A. Mansouri, T. Bakir, and S. Femmam, "Human action recognition with skeleton and infrared fusion model," *Journal of Image and Graphics*, vol. 11, no. 4, pp. 309–320, 2023.

[10] M. T. Ajili and Y. Hara-Azumi, "Multimodal neural network acceleration on a hybrid cpu-fpga architecture: A case study," *IEEE Access*, vol. 10, pp. 9603–9617, 2022.

[11] N. Attaran, A. Puranik, J. Brooks, and T. Mohsenin, "Embedded low-power processor for personalized stress detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 12, pp. 2032–2036, 2018.

[12] A. N. Mazumder, H. Ren, H.-A. Rashid, M. Hosseini, V. Chandrareddy, H. Homayoun, and T. Mohsenin, "Automatic detection of respiratory symptoms using a low-power multi-input cnn processor," *IEEE Design & Test*, vol. 39, no. 3, pp. 82–90, 2021.

[13] A. Jafari, A. Ganesan, C. S. K. Thalisetty, V. Sivasubramanian, T. Oates, and T. Mohsenin, "Sensornet: A scalable and low-power deep convolutional neural network for multimodal data classification," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 274–287, 2018.

[14] A. N. Mazumder, H.-A. Rashid, and T. Mohsenin, "An energy-efficient low power lstm processor for human activity monitoring," in *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, pp. 54–59, IEEE, 2020.

[15] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th international symposium on wearable computers*, pp. 108–109, IEEE, 2012.

[16] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th international conference on world wide web*, pp. 351–360, 2017.

[17] B. Zhang, J. Han, Z. Huang, J. Yang, and X. Zeng, "A real-time and hardware-efficient processor for skeleton-based action recognition with lightweight convolutional neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 12, pp. 2052–2056, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer*

vision and pattern recognition, pp. 770–778, 2016.

[19] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] Y.-F. Song, Z. Zhang, C. Shan, and L. Wang, "Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition," in *proceedings of the 28th ACM international conference on multimedia*, pp. 1625–1633, 2020.

[22] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110–1118, 2015.

[23] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *European conference on computer vision*, pp. 816–833, Springer, 2016.

[24] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[25] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, "Global context-aware attention lstm networks for 3d action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1647–1656, 2017.

[26] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3d action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3288–3297, 2017.

[27] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *Proceedings of the IEEE international conference on computer vision*, pp. 2117–2126, 2017.

[28] P. Zhang, J. Xue, C. Lan, W. Zeng, Z. Gao, and N. Zheng, "Adding attentiveness to the neurons in recurrent neural networks," in *proceedings of the European conference on computer vision (ECCV)*, pp. 135–151, 2018.

[29] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5323–5332, 2018.

[30] C. Si, Y. Jing, W. Wang, L. Wang, and T. Tan, "Skeleton-based action recognition with spatial reasoning and temporal stack learning," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 103–118, 2018.

[31] C. Li, Q. Zhong, D. Xie, and S. Pu, "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation," *arXiv preprint arXiv:1804.06055*, 2018.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.