UNICA UNIVERSITÀ DEGLI STUDI DI CAGLIARI

This is the Author's *accepted* manuscript version of the following contribution:

F. Manca *et al.*, "Runtime Reconfigurable FPGA Accelerator for Tactile Texture Classification Based Shallow CNN," *2025 20th International Conference on PhD Research in Microelectronics and Electronics (PRIME)*, Taormina, Italy, 2025, pp. 1-4.

© 2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

The publisher's version is available at:

http://dx.doi.org/10.1109/PRIME66228.2025.11203750

When citing, please refer to the published version.

This full text was downloaded from UNICA IRIS https://iris.unica.it/

Runtime Reconfigurable FPGA Accelerator for Tactile Texture Classification Based Shallow CNN

Federico Manca*, Riccardo Testa†, Francesco Ratto*,
Mohamad Yaacoub†, Maurizio Valle†, Luigi Raffo*, Francesca Palumbo*
*Università degli Studi di Cagliari, Cagliari, Italy
†Università di Genova, Genova, Italy

***Abstract*—Electronic skin (e-skin) represents a transformative advancement in human-machine interaction, offering tactile sensing capabilities that emulate the mechanical and physiological properties of human skin. The integration of edge AI for this type of application enables sensors to process complex data in real time, but deploying AI models on resource-constrained embedded platforms remains a challenge due to limitations in memory, energy efficiency, and computational power. In this work, we present the case study of a 1D-CNN adaptive accelerator for texture recognition implemented on an FPGA, leveraging a design flow that offers support for the design and deployment of quantized CNN accelerators with runtime reconfiguration capabilities. As a step toward a future project that combines an FPGA with a tactile acquisition interface, we extended the design flow to support 1D-CNNs and subsequently analyzed the effects of quantization on the CNN accelerator's precision, resource usage, and power consumption.**

***Index Terms*—FPGA, CNN, QONNX, textural features, texture discrimination**

I. Introduction

Electronic skin (e-skin) has emerged as a transformative technology designed to emulate the tactile mechanical and physiological properties of human skin [1]. Its primary objective is to deliver advanced sensory feedback and interpretation, thereby enabling more natural and effective interfaces between humans and machines [2].

Recent advancements in materials science have facilitated the development of highly sensitive and flexible systems. The tactile sensing capabilities of e-skin technologies are crucial for their application in robotics and prosthetics. Embedded tactile sensors can detect pressure, strain, and temperature variations, emulating the human skin's ability to perceive touch and environmental stimuli [3]. When integrated into prosthetic and rehabilitation devices, e-skin can significantly enhance the quality of life for people with disabilities by enabling more intuitive and natural interactions with their surroundings.

In addition to material innovation, considerable research has focused on developing algorithms that reconstruct various aspects of the mechanical contact on the skin surface. However, despite progress in sensor design and tactile data processing, current technological solutions still fall short of replicating the full capabilities of human touch [4]. To address this gap, the integration of AI algorithms allows these sensors to process complex data in real-time, such as object recognition [5], [6], texture recognition [7], touch modality [8], etc, thereby enhancing the functionality and responsiveness of e-skin [9]. These AI-driven approaches hold the potential to provide predictions that mimic or even occasionally surpass human expertise [9]. Nevertheless, the complexity of these algorithms presents major challenges for implementation on resource-constrained embedded systems, due to the limitation exhibited by these devices in terms of memory capacity, energy efficiency, and computational speed [10]. Field Programmable Gate Array (FPGA) have been widely adopted due to their parallel processing capabilities and reconfigurability, offering a promising platform for deploying ML models in embedded environments [11].

The employment of FPGAs enables the possibility of achieving real-time decision-making [12], [13], which can significantly enhance the quality of service provided to patients, through the integration of Edge AI [14], which offers fast, cloud-independent processing directly at the edge. By processing data locally, edge computing enhances context and location awareness, offers low latency for real-time services, and improves network efficiency by allowing smart devices to offload only relevant data [15]. Lightweight implementations of Convolutional Neural Networks (CNNs) can be employed to facilitate texture classification directly at the edge, as they have proven effective in maintaining high classification accuracy while operating within the limited computational and power budgets of embedded systems [16]. Moreover, leveraging the reconfiguration capabilities of the FPGA, runtime-adaptive CNN accelerators can be implemented, which can be reconfigured on the fly depending on specific constraints [17], [18].

We envision an embedded system that integrates an acquisition interface with a FPGA, to directly process the acquired data on the spot, providing real-time data acquisition and dynamic processing capabilities, to deliver fast feedback to the user while optionally reducing power and energy consumed. In this work, we specifically focus on the reconfiguration aspect of the FPGA's accelerator, exploring the design and implementation of a runtime-reconfigurable CNN for texture recognition. The main contributions of this paper are:

- Extended the design flow presented in [18] to support 1D-CNN and the Global Average Pooling layer.
- Implemented quantization-aware training for the 1D-CNN model, with a systematic exploration of various quantization levels to assess their impact on model accuracy.

II. Methodology

A. Dataset & Pre-processing

The dataset used in this work was collected as described in [19]. The system features a biomimetic finger equipped with a piezoelectric sensing array and embedded electronics. The sensor array comprises eight sensing units based on PVDF materials, screen-printed on a finger-shaped flexible PET substrate. The embedded electronics condition and transmit tactile signals to a host PC, sampling at 2 kSamples/s per channel to capture the full frequency bandwidth of the sensors (1 Hz–1 kHz). The finger was mounted on the z-axis of a three-axis cartesian robot to perform a series of sliding actions on eight artificial textures. These experiments were carried out at two sliding velocities. Additionally, three indentation forces were considered. In total, 600 trials were recorded, corresponding to six distinct experimental combinations, each repeated 100 times per texture for a duration of 10 seconds.

The dataset can be formalized as follows: $\mathcal{D} = \{(\mathcal{X}_i, y_i); \mathcal{X}_i \in \mathbb{R}^{N_s \times N_c}; y_i \in \{\text{Texture 1},\ldots, \text{Texture 8}\}, i = 1,\ldots,4800\}$ where $N_c = 8$ is the number of sensors and $N_s = 20000$ (10 seconds $\times$ 2000 samples per second).To reduce the input size for the machine learning algorithms described in Sect. II-B, each signal was segmented into ten windows with randomly selected starting points and a fixed window length of 300 samples. This window length was chosen based on the analysis presented in [19], which identified 300 samples as the minimum required to achieve good accuracy. The resulting dataset is defined as: $\tilde{\mathcal{D}} = \{(\tilde{\mathcal{X}}_i, \tilde{y}_i); \tilde{\mathcal{X}}_i \in \mathbb{R}^{N_s \times N_c}; y_i \in \{\text{Texture 1},\ldots, \text{Texture 8}\}, i = 1,\ldots,48000\}$ where $N_c = 8$ is the number of sensors and $N_s = 300$.

B. ML algorithm for texture classification

Fig. 1 shows the proposed shallow 1D-CNN architecture employed in this paper to achieve real-time texture recognition. The architecture was specifically developed with the goal of addressing the constraints and requirements of real-time texture classification in embedded systems, utilizing 1D layers to match the format of the input data. In this model, raw time-series sensor data are directly processed by a 1D convolutional block with filter and kernel size optimized during training. This block consists of a Conv1D layer with ReLU activation, followed by batch normalization and MaxPooling1D operations. Finally, the data are processed by Global Average Pooling and classified by the last dense layer. This shallow and streamlined design ensures low inference latency and low energy consumption, making it ideally suited for deployment on resource-constrained embedded systems. A grid search was performed during training to tune the hyperparameters. Three filter sizes (8, 16, 32) and four kernel sizes (2, 3, 4, 5) were tested. The best hyperparameters were determined by the best-performing model in test accuracy. The final convolutional layer has a kernel size of (1,5) and a filter size of 32. The model was trained on the $\tilde{\mathcal{D}}$. The networks are all implemented in Python using the Keras API with the following settings:

- Adam optimizer with learning rate $l_r = 15 \cdot 10^{-3}$

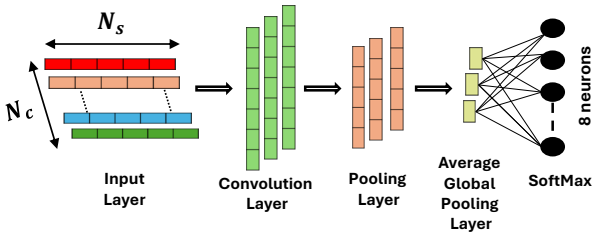

Fig. 1: Diagram of the proposed 1D-CNN.

- batch size of $b = 64$
- early stop criterion on the validation loss with a patience value of $p = 40$ to prevent over-fitting
- learning parameter reduction criterion with a patience value of $p = 5$ and a reduction factor of $f = 0.5$

C. Adopted Framework for FPGA accelerator design

Several frameworks have been developed to facilitate the deployment of Neural Network (NN) accelerators on FPGAs [20]–[24]. These tools generally lack support for runtime reconfiguration, a feature that is essential for the targeted application in this work. We leveraged and extended the work in [18] to obtain a reconfigurable CNN accelerator for texture recognition. The design flow presented in [18] begins with the QONNX[1] representation of the CNN model. This format is obtained by converting a Keras model into the QONNX intermediate representation, which is then parsed to produce the following artifacts:

- C++ templates for each layer of the CNN.
- TCL scripts to automate the Vitis HLS synthesis process.
- Network topology files (.xdf) and actor interface files (.cal).

The C++ templates are synthesized into an HDL library using Vitis HLS, guided by the provided TCL scripts. The MDC Frontend then takes one or more topology files to generate a multi-dataflow architecture which is subsequently populated by the MDC Backend with the synthesized HDL library components, ultimately producing the reconfigurable accelerator, as depicted in Fig.2.

III. Towards adaptive FPGA acceleration

A. Extension of the design flow

Handling the 1D-CNN required upgrades to the parsing script, making it possible to handle smaller QONNX tensor sizes. The initial problem was managing the converted QONNX model, as the tf2onnx function, dedicated to converting a Keras model into the QONNX format, forces the model to utilize the NCHW format. Rather than modifying the model directly, it was more practical to adapt the dataset itself by reshaping it for compatibility with the FPGA accelerator during inference. To support the target CNN, a Global Average Pooling layer was added to the existing templates. This

[1]https://github.com/fastmachinelearning/qonnx

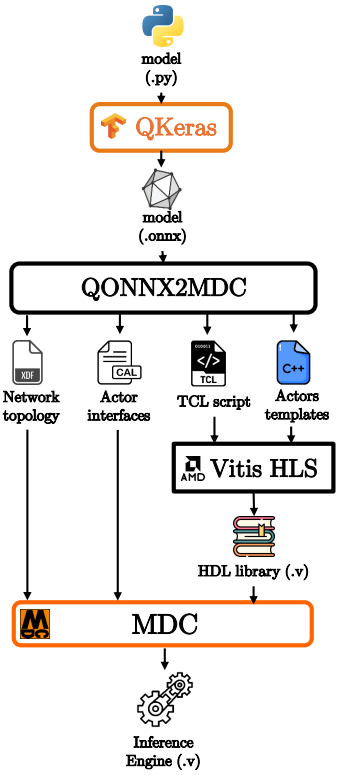

Fig. 2: Design flow pipeline.

extension, along with the complete design flow, is available in the open-source GitHub repository[2].

B. Quantization aware training and precision exploration

An exploration was conducted on the best model found during the grid search, adopting different levels of quantization for its layers. By leveraging Quantization-Aware Training (QAT), it is possible to produce CNNs models that maintain acceptable accuracy while significantly reducing hardware resource usage during the synthesis phase. This is particularly important in embedded systems, where increased resource demands can lead to larger device footprints, higher space requirements, and greater power consumption. For the QAT procedure, which is automatically carried out on QKeras models, the dataset $\tilde{\mathcal{D}}$ was split into 70% for training, 15% for validation to select the best network configuration, and 15% for testing, maintaining class balance throughout. The quantization levels evaluated were *<16,8>*, *<8,4>*, and *<4,2>*, where each pair *<X, Y>* denotes a fixed-point format with X total bits and Y bits allocated to the integer part. The training parameters were maintained consistent with those used during the grid search for the best model hyperparameters.

C. Results analysis and discussion

After the QAT phase, the models were implemented with the Multi-Dataflow Composer (MDC)'s design flow and synthesized on Vivado 2022.2, to gather information on resource consumption. Moreover, a post-synthesis functional simulation supplied information on power consumption. These results, along with the classification accuracies of the corresponding models, are summarized in Table I. Given that the floating-point base model obtained with the grid search recorded an accuracy of 94.78%, it is noticeable how the accuracy, while worsening with smaller quantization levels, is still kept at an acceptable level. The *<16,8>* recorded indeed a loss of only 0.42%. Moreover, it was evident from the outset that an FPGA-based implementation of the CNN would significantly outperform its microcontroller-based counterpart. In [25], [26], two CNNs of similar size and complexity to the 1D-CNN proposed in this work were deployed on STM32 microcontrollers, achieving inference latencies in the millisecond range. In contrast, the FPGA accelerators developed here achieved latencies below 100 µs, as confirmed through Vivado simulations, while also supporting runtime reconfiguration. Recorded energy consumption was in the range of hundreds of µJ on the microcontroller, whereas the FPGA implementation demonstrated lower energy usage, in the order of tenths of µJ per inference. These facts lead to the decision of not implementing the same 1D-CNN on a microcontroller.

TABLE I: Resource utilization and performance of quantized accelerators.

Q	Accuracy [%]	Latency [µs]	LUT [#]	BRAM [#]	DSP [#]	FF [#]	Power [W]	Energy [µJ]
<16,8>	94.36	87	14218	11.5	86	21278	0.305	26.5
<8,4>	92.97	87	10124	8	86	13428	0.270	23.4
<4,2>	89.48	87	7886	7	70	10462	0.198	17.2
cnn_mxd	90.20	87	7456	7	86	10274	0.232	20.1
cnn_rec	90.20/89.48	87	8246	8	86	10720	0.232/0.198	20.1/17.2

To explore adaptive configurations, the base model with `<4,2>` precision was modified by increasing the Dense layer precision to `<16,8>`, creating a mixed-precision variant (`cnn_mxd`) that recovers some accuracy while sharing most layers with the base model. This variant was merged with the `<4,2>` model to form the adaptive accelerator `cnn_rec`, which significantly reduces resource usage: 7096 LUTs, 6 BRAMs, 70 DSPs, and 10016 FFs (compared to a fully parallel design, excluding potential Vivado optimizations). Using runtime adaptivity instead of multiple parallel models greatly lowers FPGA resource and area requirements, enabling smaller, more cost-effective deployments. The `cnn_rec` integrates two models while using about half the resources of a fully parallel solution.

In today's fast-paced landscape, system upgradability is key. Microcontrollers offer flexibility but suffer from high latency, while Application Specific Integrated Circuits (ASICs) are fast but rigid. FPGAs provide a balanced alternative, combining near-silicon performance with reconfigurability for seamless feature updates without hardware changes. The proposed automated toolchain further supports runtime adaptivity, speeding up the development of accelerators for new models and CNN architectures.

[2]https://github.com/mdc-suite/qonnx2mdc

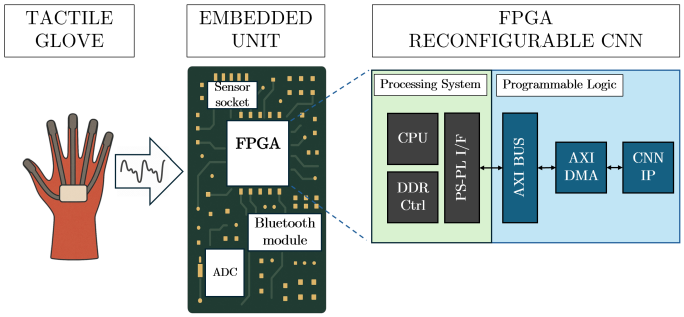

Fig. 3: Envisioned system for Texture Classification on the edge.

IV. Conclusion & Future Work

In this work, we explore the feasibility of enhancing solutions for data acquisition, conditioning, and transmission, like the one proposed in [27], through FPGA-based edge acceleration. A 1D-CNN for Texture Recognition tasks was selected as the baseline model and quantized with different bit precisions. The model demonstrated high resilience to quantization, maintaining performance even at 4-bit precision. To achieve runtime adaptivity, two versions of the model with different quantized precisions were merged into a single, adaptive accelerator. The reconfigurable accelerator enabled runtime adaptivity while significantly reducing resource consumption, using nearly half the resources compared to a theoretical fully parallelized implementation. This capability is particularly crucial for edge applications, where stringent constraints on area and resource efficiency are paramount.

Building on this work, we envision a fully embedded, on-edge system for real-time texture classification (Fig. 3). In this setup, sensors in a tactile glove transmit signals via Bluetooth to an embedded unit. After signal conditioning and digitization through the ADC, data is processed by an onboard FPGA for texture classification, potentially driving e-skin interactions. The FPGA's computational power also enables possibilities beyond inference. Future work will explore on-edge refinement training of NNs, allowing system upgrades and personalized feedback tailored to individual users [28].

References

[1] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing—from humans to humanoids," *IEEE transactions on robotics*, vol. 26, no. 1, pp. 1–20, 2009.

[2] R. Yin, D. Wang, S. Zhao, Z. Lou, and G. Shen, "Wearable sensors-enabled human–machine interaction systems: from design to application," *Advanced Functional Materials*, vol. 31, no. 11, p. 2008936, 2021.

[3] C. C. Vu, S. J. Kim, and J. Kim, "Flexible wearable sensors-an update in view of touch-sensing," *Science and Technology of Advanced Materials*, vol. 22, no. 1, pp. 26–36, 2021.

[4] L. Seminara, S. Dosen, F. Mastrogiovanni, M. Bianchi, S. Watt, P. Beckerle, T. Nanayakkara, K. Drewing, A. Moscatelli, R. L. Klatzky, *et al.*, "A hierarchical sensorimotor control framework for human-in-the-loop robotic hands," *Science Robotics*, vol. 8, no. 78, p. eadd5434, 2023.

[5] S. Sundaram, P. Kellnhofer, Y. Li, J.-Y. Zhu, A. Torralba, and W. Matusik, "Learning the signatures of the human grasp using a scalable tactile glove," *Nature*, vol. 569, no. 7758, pp. 698–702, 2019.

[6] M. Yaacoub, A. Ibrahim, F. Khansa, L. Hammadi, and C. Gianoglio, "Wearable multisensory glove for shape, size, and stiffness recognition based on off-the-shelf components," *IEEE Sensors Letters*, vol. 9, no. 4, pp. 1–4, 2025.

[7] Y. Yan, Z. Hu, Y. Shen, and J. Pan, "Surface texture recognition by deep learning-enhanced tactile sensing," *Advanced Intelligent Systems*, vol. 4, no. 1, p. 2100076, 2022.

[8] A. Ibrahim and M. Valle, "Real-time embedded machine learning for tensorial tactile data processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 11, pp. 3897–3906, 2018.

[9] C. Xu, S. A. Solomon, and W. Gao, "Artificial intelligence-powered electronic skin," *Nature machine intelligence*, vol. 5, no. 12, pp. 1344–1355, 2023.

[10] P. Gastaldo, E. Ragusa, S. Dosen, and F. Palmieri, "Special issue on integration of machine learning and edge computing for next generation of smart wearable systems," 2024.

[11] M. Papadonikolakis and C.-S. Bouganis, "Novel cascade fpga accelerator for support vector machines classification," *IEEE Trans. on neural networks and learning systems*, vol. 23, no. 7, pp. 1040–1052, 2012.

[12] C. Rubattu, A. Ledda, F. Ratto, C. Jugade, D. Goswami, and F. Palumbo, "Integrating fpga-based acceleration in industrial motion control system," *IEEE Open Journal of the Industrial Electronics Society*, 2025.

[13] C. Rubattu, F. Palumbo, S. S. Bhattacharyya, and M. Pelcat, "Pathtracer: Understanding response time of signal processing applications on heterogeneous mpsocs," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 6, no. 4, pp. 1–30, 2022.

[14] R. Singh *et al.*, "Edge AI: A survey," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023.

[15] C. Sau, T. Fanni, C. Rubattu, L. Fanni, L. Raffo, and F. Palumbo, "Feasibility study and porting of the damped least square algorithm on fpga," *Ieee Access*, vol. 8, pp. 175483–175500, 2020.

[16] K. Sun, X. Wang, X. Miao, and Q. Zhao, "A review of AI edge devices and lightweight CNN and LLM deployment," vol. 614, p. 128791.

[17] F. Ratto, Á. P. Máinez, C. Sau, P. Meloni, G. Deriu, S. Delucchi, M. Massa, L. Raffo, and F. Palumbo, "An automated design flow for adaptive neural network hardware accelerators," *Journal of Signal Processing Systems*, vol. 95, no. 9, pp. 1091–1113, 2023.

[18] F. Manca *et al.*, "ONNX-To-Hardware Design Flow for Adaptive Neural-Network Inference on FPGAs," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, (Cham), Springer, 2025.

[19] H. A. H. Ali, Y. Abbass, C. Gianoglio, A. Ibrahim, C. Oh, and M. Valle, "Neuromorphic tactile sensing system for textural features classification," *IEEE Sensors Journal*, 2024.

[20] T. Aarrestad *et al.*, "Fast convolutional neural networks on FPGAs with hls4ml," Apr. 2021. arXiv:2101.05108.

[21] A. Ghaffari *et al.*, "Cnn2gate: An implementation of convolutional neural networks inference on fpgas with automated design space exploration," *Electronics*, vol. 9, p. 2200, 2020.

[22] K. Guo *et al.*, "Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, pp. 35–47, Jan. 2018.

[23] Y. Umuroglu *et al.*, "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in *Proceedings of the 2017 ACM/SIGDA*, pp. 65–74, Feb. 2017. arXiv:1612.07119 [cs].

[24] S. I. Venieris *et al.*, "fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs," in *2016 IEEE 24th (FCCM)*, (Washington, DC, USA), IEEE, May 2016.

[25] Y. Amin, C. Gianoglio, and M. Valle, "Embedded real-time objects' hardness classification for robotic grippers," *Future Generation Computer Systems*, vol. 148, pp. 211–224, Nov. 2023.

[26] M. Yaacoub, H. Al Haj Ali, C. Gianoglio, M. Valle, and A. Ibrahim, "Multisensory wearable glove for object recognition based embedded machine learning," in *2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4. ISSN: 2995-0589.

[27] M. Saleh, Y. Abbass, A. Ibrahim, and M. Valle, "Experimental assessment of the interface electronic system for pvdf-based piezoelectric tactile sensors," *Sensors*, vol. 19, no. 20, 2019.

[28] M. Tsukada, M. Kondo, and H. Matsutani, "A Neural Network-Based On-Device Learning Anomaly Detector for Edge Devices," *IEEE Transactions on Computers*, vol. 69, pp. 1027–1044, July 2020.