Towards AIE-based Mutual Information for Image Registration

ID03: From FPGA to AIE
Giacomo Brunetta,
Francesco Santambrogio

June 30, 2024

Abstract

Mutual Information (MI) serves as a key similarity metric in image registration, demanding intensive computational resources due to histogram extraction and entropy computation. Despite advancements in GPU and FPGA acceleration, AI Engines remain underutilized for MI computation. This work introduces a hardware-accelerated solution leveraging the AI Engine of the ACAP Versal VCK5000 for entropy computation based on joint and marginal histograms of two digital images, focusing on parallelization across multiple tiles and SIMD operations. We propose a novel vectorized base-two logarithm currently absent in the AIE API, demonstrating superior performance over the Arm® Cortex®-A72 CPU and existing logarithm implementation of the AIE API (1.49× speedup with an error within $[-4.29 \times 10^{-6}, 5.24 \times 10^{-6}]$ for values in the range $[0, 100]$). Evaluation across three graph implementations of the AIE confirms significant speedup in entropy computation for MI, especially with parallelized kernels (9.71×). Future efforts will integrate our technique into comprehensive image registration frameworks on the ACAP Versal VCK5000. Beyond image registration, our approach suggests broader applications in domains requiring efficient MI computation, such as feature selection and cryptanalysis.

POLITECNICO
MILANO 1863

POLITECNICO MILANO 1863
NECST laboratory

1 Introduction

Image registration is an essential task in image processing used to match two or more images captured at different times, called multi-temporal analysis, from different viewpoints, called multi-view, or different sensors, called multi-modal [2]. It is widely employed in fields such as medicine, target recognition, satellite imaging, and remote sensing. In particular, image registration identifies the parameters of the geometrical transformation matrix that allows the correct overlap of the acquired images. This process typically involves a three-block structure consisting of a transformation model, an optimization method, and a similarity metric [24].

Accurate registration necessitates multiple iterations of these blocks, with the similarity metric being the most compute-intensive. Among the similarity metrics used in the literature, Mutual Information (MI) stands out as the leading one in multi-modal registration [26]. The MI computation includes the extraction of histograms and the computation of entropy which is a highly floating-point intensive task. Despite extensive research on GPU and FPGA acceleration, AI Engines have yet to be explored for the MI computation.

Our work proposes the use of the AI Engine of the ACAP Versal VCK5000 to accelerate the entropy computation given the joint and marginal histograms of two digital images. We focused on achieving parallelization on three levels:

- Evenly distributing the workload across multiple AI Engine tiles.
- Using exclusively vectorized (SIMD) floating-point operations in the kernels that compute marginal and mutual entropy.
- Employing static scheduling to explicitly parallelize the code by inserting multiple operations into a single VLIW instruction.

2 Background

This section outlines the theoretical principles behind our proposed solution aimed at speeding up MI computation, giving a high-level description of the Mutual Information and the AMD Versal VCK5000 system.

2.1 Mutual Information

A wide range of algorithms utilize Mutual Information in different fields spanning from imaging, where it is a crucial similarity metric for image registration [18], to genomics, where it is employed in phylogenetics [17], relevance networks ,[19], and in the training of Hidden Markov Models [1]. MI is a concept derived from Information Theory and related to entropy which measures the statistical dependence between two random variables, X and Y [12]. In our context, these two variables are represented by images, specifically a reference image and a floating image, and the goal is to align the latter with the former. Mathematically, MI quantifies the similarity between the joint entropy $H(X, Y)$ and the individual entropies $H(X)$ and $H(Y)$, as shown in eq. (1).

$$MI(X;Y) = H(X) + H(Y) - H(X,Y) \tag{1}$$

Therefore, an essential step is the definition and computation of the different entropies, as defined by Shannon's equation eq. (2), where $p(x)$ and $p(y)$ are the marginal probabilities, and $p(x,y)$ is the joint probability.

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{2}$$

$$H(X,Y) = -\sum_{x,y \in X,Y} p(x,y) \log p(x,y) \tag{3}$$

In imaging, probabilities are derived from individual image histograms, and joint probabilities come from the joint histogram, that is a square matrix of size $N \times N$ where N is the number of gray levels in the images. Each element in the joint histogram represents the number of voxels in image X with intensity x that correspond to voxels in image Y with intensity y [12]. Using the joint histogram, it is possible to derive the individual ones by summing its rows and columns, yielding the reference and floating histograms, respectively [12]. This histogram computation phase involves integer operations since it mainly consists in counting corresponding voxels to construct the joint histogram. This step is often the most computationally intensive as its complexity scales with the image size.

For the entropy computation instead, we need to obtain the marginal and joint probabilities, which can be extracted by dividing each value of the single and joint histograms by the image dimensions in voxels. Referring to eq. (2), the last step is to multiply each probability by its logarithmic value, and, by accumulating them, we obtain the entropies. Finally, we combine all the entropy values to extract the MI, as in eq. (1). Unlike histogram computation, this phase is very floating-point intensive and it does not scale with the image size. This work specifically focuses on the entropy computation for MI, aiming to accelerate it by finding a hardware architecture that is suitable for its characteristics.

2.2 AMD Versal VCK5000

The Versal Adaptive Compute Acceleration Platform (ACAP) [9] is a fully software-programmable, heterogeneous system that combines Scalar Engines, Adaptable Engines, and Intelligent Engines. Its main purpose is to provide high parallelization with the introduction of the AI Engine, a tiled array of Very Long Instruction Word (VLIW), and Single Instruction Multiple Data (SIMD) processing elements. The core principle of VLIW architectures is Instruction Level Parallelism (ILP), where all the operations that are supposed to begin at the same time are packaged into a single VLIW instruction. In particular, the parallelization is achieved with a vector unit present in each AIE tile of the 2D array composing the AI Engine, which allows for the execution of the same operation on different data elements of the vectors. In this work, we used a Versal VCK5000 to exploit its AI Engine to compute the entropy for MI.

3 State of the art

As stated in Section 2.1, MI computation is composed of two phases, namely histogram computation and entropy computation, approached using various architectures in the literature, particularly GPUs and FPGAs.

Histogram computation is a challenging task for GPUs because the memory allocated to each thread block typically ranges from 16 to 64 KB, while the joint histogram requires 256 KB, making it too large to fit. Many solutions have been proposed to overcome this problem. Kei Ikeda et Al. [11] proposed an **approximated method** that consists of computing the MI only of the portion of the joint histogram that sits around the diagonal. Another proposal was made by Shifu Chen et Al. [5] suggesting to use **sort and count**.

In recent years many proposals of FPGA-based architectures for both 2D [7],[22],[8] and 3D [21],[4] image registration have been made. Field Programmable Gate Arrays (FPGAs) offer flexibility by enabling the creation of custom hardware infrastructures for histogram computation. However, FPGAs have downsides. While efficient at histogram computation, the subsequent step of entropy calculation from histograms is floating-point-intensive and less efficient on FPGAs than on other hardware architectures. In fact, floating point computations on FPGAs are resource-intensive, lower in performance compared to CPUs or GPUs, complex to design, power-hungry, and can introduce significant latency. Therefore, an ideal hardware architecture for image registration would integrate an FPGA for histogram computation with another component optimized for fast entropy computation.

The characteristics of the Versal system presented in 3 make it suitable for floating-point computation-intensive applications, and for this reason has been employed in the literature in domains like signal processing [23], Deep Learning [20] [16] [14], or computer simulations [3], and involved in operations like matrix multiplication [15] [16] [14], or arbitrary-precision integer multiplication [25]. To the best of our knowledge though, no research has yet attempted to employ the Versal platform for the computation of mutual information, due also to its relatively recent release.

4 Methodology

To leverage AIE for the MI computation, we focus on its parallelization on three levels:

- Evenly distributing the workload across multiple AI Engine kernels.
- Using exclusively vectorized (SIMD) floating-point operations in the kernels that compute marginal and mutual entropy.
- Employing static scheduling to explicitly parallelize the code by inserting multiple operations into a single VLIW instruction.

4.1 Vectorized Entropy Computation

In this section, we detail the implementation of AI Engine kernels leveraging vectorized floating-point operations to efficiently compute both marginal and mutual entropy given their respective histograms. First, we describe our approach to general entropy computation, then we propose a vectorized implementation of the base-two logarithm.

4.1.1 Entropy Computation

Algorithm 1 Entropy

```
1 procedure entropy(float histo[])
2     acc = null_vector(8);
3     for (int i = 0; i < size(histo)/8; i++){
4         // load 8 values from histogram
5         x = histo[i*8: (i+1)*8];
6         acc += log2(x)*x;
7     }
8     return sum(acc); // reduce vec to float
9 end procedure
```

To compute both the mutual and the marginal entropy it is sufficient to apply eq. (2) to the marginal and the mutual histogram respectively. To do so, as shown in the snippet above, an accumulator vector is created and it is initialized as a null vector. At each iteration, eight values of the histogram are loaded into the vector $\vec{x}$. Then, the base-two logarithm of such vector is computed and the accumulator is updated by summing the product of $\vec{x}$ by its logarithm. To maximize the throughput, the kernel must use only vectorized operations but $\log_2$ lacks a vectorized implementation in the AIE API. For this reason, we propose a vectorized $\log_2$ implementation for the Versal AI Engine.

4.1.2 Vectorized log2

Algorithm 2 Log2 of single floating-point

```
1 procedure log2(float x)
2     exp = x >> 23 // Separate the exponent
3     exp = mantissa - 127 // Remove bias
4     // overwrite the exponent with 128
5     m = x & 0x007fffff; // m = mantissa
6     m = m | 0x3f800000; // m = 1+mantissa
7
8     return exp + log_approx(m);
9 end procedure
```

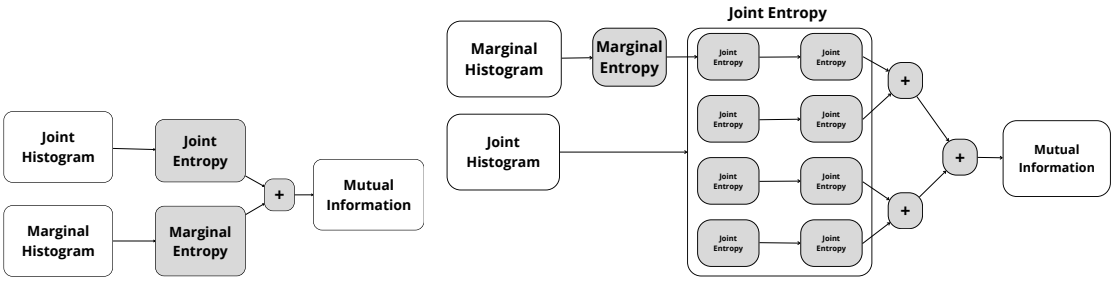
To understand the implementation of the vectorized base-two logarithm, we recall the IEEE 754 floating point notation and its base-two logarithm:

$$\text{value} = (-1)^{\text{sign}} \times (1 + \text{mantissa}) \times 2^{\text{exponent}-\text{bias}} \tag{4}$$

$$\log_2(x) = \text{exponent}(x) - \text{bias} + \log_2(1 + \text{mantissa}(x)) \tag{5}$$

As shown in eq. 5, the computation of the $\log_2$ of any positive real number can be reduced to the extraction of exponent and mantissa and the computation of $\log_2(1 + \text{mantissa})$ [6]. Since $\text{mantissa} \in [0, 1)$, $\log_2(1 + \text{mantissa})$ can be easily approximated in its domain $[1, 2)$. Many implementations exist [10] and they typically differ in how $\log_2\_\text{approx}(1 + \text{mantissa})$ is computed. By adopting a polynomial approximation, we can compute log_2 of a vector of 8 floating points using SIMD operations only. After analyzing the precision of different solutions, we conclude that a polynomial of grade 6 offers the best trade-off between error and performance.

4.2 System Design



(a) 3-kernel implementation (b) 12-kernel implementation

Figure 1: The two employed AI Engine graphs

To save resources that could be used for the steps of Image Registration, we employed the Programmable Logic (PL) component of the system for data movement only, applying burst optimization of AXI-master to read the histograms from memory and provide them to AIE. On the other hand, we leveraged AIE to compute the entropy of the joint histogram and the two entropies of the marginal histograms for the MI computation, as shown in eq. (1). We performed this computation with three different dispositions of AIE kernels, called graphs.

The first one (Figure 1a) consists of three kernels: one computes the marginal entropy from two input histograms, another computes the joint entropy from the joint histogram, and the third calculates mutual information by subtracting the marginal entropy from the joint entropy. The first two kernels operate in parallel, while the third kernel waits for their outputs and then returns its result to the PL.

Since most computation is performed by the kernel computing the entropy of the joint histogram, to speed it up, the workload should be divided more equally among different AIE kernels. For this reason, we arranged these kernels in a

systolic array, namely a matrix with R rows and C columns. Each AIE kernel computes the entropy of different portions of the histogram passed by the PL kernel in different streams, one for each AIE kernel. The first column receives the input from the PL and feeds its result to the following column. Kernels of the remaining columns receive input from the previous column, sum it with their result, and pass it along. The outputs of the last column are the inputs of a tree-like reduce-structure with R-1 kernels, that eventually produce the MI final value.

The second graph (Figure 1b) implements the systolic array with 8 kernels (4 rows, 2 columns) for the computation of the joint entropy, 1 kernel for the marginal entropy, and 3 kernels for the reduce-structure to produce the MI value.

5 Results

The following Section details the performance results of the vectorized implementation of the base-two logarithm and the three graph implementations of the AI Engine.

We utilized both an ACAP Versal VCK5000 on-premises and a unit of the HACC cluster at ETH Zürich, running Vitis version 2022.2. The software was developed in C++.

5.1 Logarithm acceleration with AIE

Since we employed a new vectorized implementation of the 2-base logarithm, we applied a test bench to analyze its stand-alone performance. We created an array containing $2^{17} \approx 131$k floating point numbers between 0 and 100 and we computed the logarithm of those floating points with the CPU (using math.h log2f) and the AI Engine (using both our implementation and the one provided by the API). Since the logarithm supplied by the API only works for integers, we represented the floating points as a fraction of two integers.

Method	Latency	Throughput $10^6 \log/s$	Speedup
CPU (reference)	1.78 ms	73.636	+0%
AIE (utils)	20.6 ms	6.363	-91.37%
AIE (vectorized)	1.21 ms	108.324	+49.46%

Table 1: Performance comparison for logarithm computation

As shown in Table 1, a single AIE tile offers a speedup of 1.49× compared to the Arm® Cortex®-A72 CPU. The error introduced by the polynomial approximation is within $[-4.29 \times 10^{-6}, 5.24 \times 10^{-6}]$ for $x \in [0, 100]$. In contrast, the AIE API Utils logarithm is significantly slower and less accurate than the CPU since it introduces errors within $[0, 0.99]$ and is almost 10 times slower.

5.2 MI acceleration with AI Engine

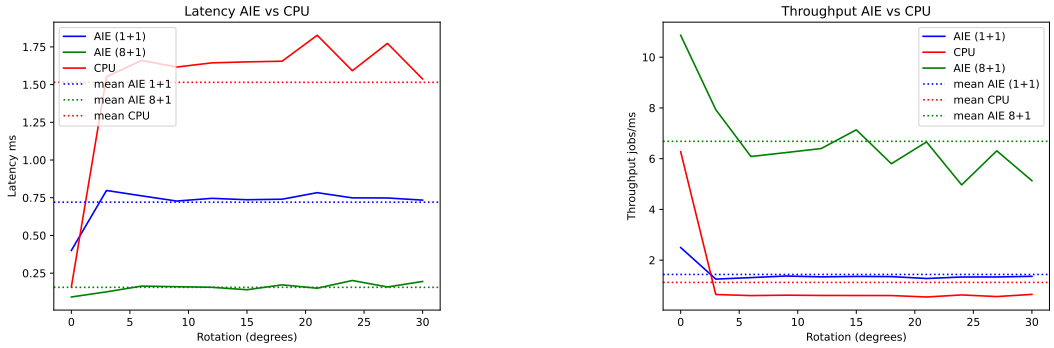
To assess our system's performance, we designed a test bench that simulates the progressive alignment of two medical images. To generate the image couples we used a high-resolution X-ray image as the reference image and generated rotations ranging from 0° to 30° with a 3° increment for the floating images. The test-bench consists of computing the Mutual Information from the marginal and joint histograms of the 11 image pairs.

We ran a test with the Arm® Cortex®-A72 CPU and the two AI Engine configurations shown in Section 1

	Latency		**Throughput**	
	Mean (ms)	**Variance**	**Mean (jobs/ms)**	**Variance**
CPU	1.515	0.103	1.123	0.337
AIE (3 kernels)	0.720	0.029	1.436	1.542
AIE (12 kernels)	0.156	0.436	6.684	1.629

Table 2: Average Latency and Throughput Measurements

Table 2 shows how the CPU and the two AI Engine implementations perform on average in the 11 test cases. Both the AI Engine implementations outperform the Arm® Cortex®-A72 CPU: the 3 kernel implementation achieves an average speedup of 1.56× and the 12 kernel implementation achieves an average speedup of 9.71×.



(a) Latency comparison

(b) Throughput comparison

Figure 2: Performance comparison of CPU and AI Engine

Figure 2 provides a detailed comparison of the performance of the three implementations in the test cases. The AI Engine implementations generally achieve near-average speedup for images rotated between 3° and 30°. However, the 0° rotation case (two identical images) is an outlier. This edge case results in a maximally sparse joint histogram (a diagonal matrix), favoring the CPU, which outperforms the 3-kernel implementation. The 12-kernel implementation still surpasses the CPU in performance but with a reduced speedup.

6 Discussion

Our experimentation with the ACAP Versal VCK5000 demonstrated that our vectorized implementation of the base-two logarithm exhibits superior performance and accuracy compared to both the Arm® Cortex®-A72 CPU and the logarithm function provided by the AIE API. By optimizing floating-point operations using vectorization, we achieved superior performance, with a speedup of 1.49×, and maintained precision within a smaller error margin, namely $[-4.29 \times 10^{-6}, 5.24 \times 10^{-6}]$ for values in the range $[0, 100]$, than the one of the AIE API logarithms. This is also because the current AIE API logarithm works only with integers, further confirming that an implementation working with floating points was needed.

Furthermore, our analysis revealed that the AI Engine, thanks to its parallel architecture and the vector units present in its tiles, is considerably faster than the Arm® Cortex®-A72 CPU in the computation of Mutual Information. In particular, by increasing the number of AIE kernels, the performance further increases, from an average speedup of 1.56× to 9.71×, pointing out that distributing the workload across more kernels improves the throughput and reduces the latency.

7 Conclusions and Future Work

In this work, we presented a hardware-based solution to speed up the entropy computation for the Mutual Information with the AI Engines of the ACAP Versal VCK5000. In particular, we proposed a vectorized implementation of the base-two logarithm, which is currently missing in the AIE API, demonstrating it outperforms the Arm® Cortex®-A72 CPU and the current implementation of the logarithm provided by AIE API. We also evaluated the performance of two different graph implementations of the AIE, demonstrating that AIE outperforms the Arm® Cortex®-A72 CPU in the computation of entropies for MI and that by parallelizing the entropy computation across several AIE kernels the speedup is even more evident.

The vectorized implementation of the base-two logarithm can be used also in other applications due to its high reusability. As future works, we will integrate our technique to compute Mutual Information (MI) within an image registration framework, such as those described in [7] or [21], to execute all stages of image registration on the AMD ACAP Versal VCK5000. While this study concentrated on Mutual Information for image registration, future research could also investigate accelerating the computation of Mutual Information in other domains, such as feature selection [13] or cryptanalysis.

References

[1] Lalit R. Bahl et al. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1986, Tokyo, Japan, April 7-11, 1986*. IEEE, 1986, pp. 49–52. DOI: 10.1109/ICASSP.1986.1169179. URL: https://doi.org/10.1109/ICASSP.1986.1169179.

[2] Lisa Gottesfeld Brown. "A Survey of Image Registration Techniques". In: *ACM Comput. Surv.* 24.4 (1992), pp. 325–376. DOI: 10.1145/146370.146374. URL: https://doi.org/10.1145/146370.146374.

[3] Nick Brown. "Exploring the Versal AI Engines for Accelerating Stencil-based Atmospheric Advection Simulation". In: *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA 2023, Monterey, CA, USA, February 12-14, 2023*. Ed. by Paolo Ienne and Zhiru Zhang. ACM, 2023, pp. 91–97. DOI: 10.1145/3543622.3573047. URL: https://doi.org/10.1145/3543622.3573047.

[4] Carlos R Castro-Pareja and Raj Shekhar. "Hardware acceleration of mutual information-based 3D image registration". In: *Journal of Imaging Science and Technology* 49.2 (2005), pp. 105–113.

[5] Shifu Chen et al. "CUDA-based acceleration and algorithm refinement for volume image registration". In: *2009 International Conference on Future BioMedical Information Engineering (FBIE)*. IEEE. 2009, pp. 544–547.

[6] M. Combet, H. Van Zonneveld, and L. Verbeek. "Computation of the Base Two Logarithm of Binary Numbers". In: *IEEE Trans. Electron. Comput.* 14.6 (1965), pp. 863–867. DOI: 10.1109/PGEC.1965.264080. URL: https://doi.org/10.1109/PGEC.1965.264080.

[7] Davide Conficconi et al. "A Framework for Customizable FPGA-based Image Registration Accelerators". In: *FPGA '21: The 2021 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Virtual Event, USA, February 28 - March 2, 2021*. Ed. by Lesley Shannon and Michael Adler. ACM, 2021, pp. 251–261. DOI: 10.1145/3431920.3439291. URL: https://doi.org/10.1145/3431920.3439291.

[8] Omkar Dandekar and Raj Shekhar. "FPGA-Accelerated Deformable Image Registration for Improved Target-Delineation During CT-Guided Interventions". In: *IEEE Trans. Biomed. Circuits Syst.* 1.2 (2007), pp. 116–127. DOI: 10.1109/TBCAS.2007.909023. URL: https://doi.org/10.1109/TBCAS.2007.909023.

[9] Brian Gaide et al. "Xilinx Adaptive Compute Acceleration Platform: Versal^{TM} Architecture". In: *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2019, Seaside, CA, USA, February 24-26, 2019*. Ed. by Kia Bazargan and Stephen Neuendorffer. ACM, 2019, pp. 84–93. DOI: 10.1145/3289602.3293906. URL: https://doi.org/10.1145/3289602.3293906.

[10] Paweł Gepner et al. “Innovative, simple and efficient algorithms for logarithms with adjustable accuracy”. In: (2023).

[11] Kei Ikeda, Fumihiko Ino, and Kenichi Hagihara. “Efficient Acceleration of Mutual Information Computation for Nonrigid Registration Using CUDA”. In: *IEEE J. Biomed. Health Informatics* 18.3 (2014), pp. 956–968. DOI: 10.1109/JBHI.2014.2310745. URL: https://doi.org/10.1109/JBHI.2014.2310745.

[12] John Kieffer. “Elements of Information Theory (Thomas M. Cover and Joy A. Thomas)”. In: *SIAM Rev.* 36.3 (1994), pp. 509–511. DOI: 10.1137/1036124. URL: https://doi.org/10.1137/1036124.

[13] Nick Kyparissas, Gavin Brown, and Mikel Luján. “FINESSD: Near-Storage Feature Selection with Mutual Information for Resource-Limited FPGAs”. In: *32nd IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2024*. 2024.

[14] Jie Lei, José Flich, and Enrique S. Quintana-Ortí. “Toward Matrix Multiplication for Deep Learning Inference on the Xilinx Versal”. In: *31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2023, Naples, Italy, March 1-3, 2023*. Ed. by Raffaele Montella, Javier García Blas, and Daniele D’Agostino. IEEE, 2023, pp. 227–234. DOI: 10.1109/PDP59025.2023.00043. URL: https://doi.org/10.1109/PDP59025.2023.00043.

[15] Jie Lei and Enrique S. Quintana-Ortí. “Mapping Parallel Matrix Multiplication in GotoBLAS2 to the AMD Versal ACAP for Deep Learning”. In: *CoRR* abs/2404.15043 (2024). DOI: 10.48550/ARXIV.2404.15043. arXiv: 2404.15043. URL: https://doi.org/10.48550/arXiv.2404.15043.

[16] Jie Lei et al. “GEMM-Like Convolution for Deep Learning Inference on the Xilinx Versal”. In: *High Performance Computing - ISC High Performance 2023 International Workshops, Hamburg, Germany, May 21-25, 2023, Revised Selected Papers*. Ed. by Amanda Bienz et al. Vol. 13999. Lecture Notes in Computer Science. Springer, 2023, pp. 593–604. DOI: 10.1007/978-3-031-40843-4_44. URL: https://doi.org/10.1007/978-3-031-40843-4%5C_44.

[17] Flavio Lichtenstein, Fernando Antoneli, and Marcelo R. S. Briones. “MIA: Mutual Information Analyzer, a graphic user interface program that calculates entropy, vertical and horizontal mutual information of molecular sequence sets”. In: *BMC Bioinform.* 16 (2015), 409:1–409:19. DOI: 10.1186/S12859-015-0837-0. URL: https://doi.org/10.1186/s12859-015-0837-0.

[18] Frederik Maes et al. “Multimodality Image Registration by Maximization of Mutual Information”. In: *IEEE Trans. Medical Imaging* 16.2 (1997), pp. 187–198. DOI: 10.1109/42.563664. URL: https://doi.org/10.1109/42.563664.

[19] Ann E. Nicholson and Nathalie Jitnah. "Using Mutual Information to Determine Relevance in Bayesian Networks". In: *PRICAI'98, Topics in Artificial Intelligence, 5th Pacific Rim International Conference on Artificial Intelligence, Singapore, November 22-27, 1998, Proceedings*. Ed. by Hing-Yan Lee and Hiroshi Motoda. Vol. 1531. Lecture Notes in Computer Science. Springer, 1998, pp. 399–410. DOI: 10.1007/BFB0095287. URL: https://doi.org/10.1007/BFb0095287.

[20] Tiago David Sousa Ramos. "Implementation of Convolutional Neural Networks on a Versal Device". In: (2023).

[21] Giuseppe Sorrentino et al. "Hephaestus: Codesigning and Automating 3D Image Registration on Reconfigurable Architectures". In: *ACM Trans. Embed. Comput. Syst.* 22.5s (2023), 134:1–134:24. DOI: 10.1145/3607928. URL: https://doi.org/10.1145/3607928.

[22] Ioannis Stratakos et al. "Hardware Acceleration of Image Registration Algorithm on FPGA-based Systems on Chip". In: *Proceedings of the International Conference on Omni-Layer Intelligent Systems, COINS 2019, Crete, Greece, May 5-7, 2019*. Ed. by Farshad Firouzi et al. ACM, 2019, pp. 92–97. DOI: 10.1145/3312614.3312636. URL: https://doi.org/10.1145/3312614.3312636.

[23] Jason Timpe et al. "Application of AMD Versal™ Adaptive SoC to Radar Space Time Adaptive Processing in Space". In: *2023 European Data Handling & Data Processing Conference (EDHPC)*. IEEE. 2023, pp. 1–5.

[24] Petra A Van den Elsen, E-JD Pol, and Max A Viergever. "Medical image matching-a review with classification". In: *IEEE Engineering in Medicine and Biology Magazine* 12.1 (1993), pp. 26–39.

[25] Zhuoping Yang et al. "AIM: Accelerating Arbitrary-Precision Integer Multiplication on Heterogeneous Reconfigurable Computing Platform Versal ACAP". In: *IEEE/ACM International Conference on Computer Aided Design, ICCAD 2023, San Francisco, CA, USA, October 28 - Nov. 2, 2023*. IEEE, 2023, pp. 1–9. DOI: 10.1109/ICCAD57390.2023.10323754. URL: https://doi.org/10.1109/ICCAD57390.2023.10323754.

[26] Barbara Zitová and Jan Flusser. "Image registration methods: a survey". In: *Image Vis. Comput.* 21.11 (2003), pp. 977–1000. DOI: 10.1016/S0262-8856(03)00137-9. URL: https://doi.org/10.1016/S0262-8856(03)00137-9.