An Anatomization of FPGA-Based Neural Networks

Anvit Negi, Devansh Saxena, Kunal, and Kriti Suneja

Abstract Ongoing advancements in the improvement of multilayer convolutional neural organizations have brought about upgrades in the precision of important recognition jobs, for example, huge category picture classification and cutting-edge automated recognition of speech. Custom hardware accelerators are crucial in improving their performance, given the large computational demands of Convolution Neural Networks (CNN). The Field-Programmable Gate Arrays (FPGAs) reconfigurability, computational abilities, and high energy efficacy makes it a propitious CNN hardware acceleration tool. CNN have demonstrated their value in picture identification and recognition applications; nonetheless, they require high CPU use and memory transmission capacity tasks that cause general CPUs to neglect to accomplish wanted execution levels. Consequently, to increase the throughput of CNNs, hardware accelerators using Application-Specific Integrated Circuits (ASICs), FPGAs, and Graphic Processing Units (GPUs) have been employed to improve CNN performance. To bring out their synonymity and dissimilarity, we group the works into many groups. Thus, it is anticipated that this review will lead to the upcoming development of successful hardware accelerators and be beneficial to researchers in deep learning.

Keywords FPGA · ASIC · Deep learning · Neural net

1 Introduction

In the past decade, artificial intelligence (AI) and machine learning (ML) instruments have gained considerable prominence due to advancements in computational structure which consists of area, power, and effectiveness. A range of programs have started using AI algorithms to improve overall efficiency than conventional methods. These applications include image processing [1], for example, face identification,

A. Negi · D. Saxena (✉) · Kunal · K. Suneja
Department of Electronics and Communication Engineering, Delhi Technological University, New Delhi, India

K. Suneja
e-mail: kritisuneja@dtu.ac.in

P. Nayak et al. (eds.), *IoT and Analytics for Sensor Networks*, Lecture Notes in Networks and Systems 244, https://doi.org/10.1007/978-981-16-2919-8_45

banking and statistical surveying [2], mechanical arms in the robotized producing area [3], medical services applications [4], database administration and management [5], and face checking and investigation security applications [6].

The explosive development of big data over the past decade has inspired revolutionary approaches to obtain data from various sensors such as photos and voice samples. In reality, these Convolution Neural Networks (CNNs) [7] now are conceived as the standard method among the proposed methods by delivering "human-like" accuracy in various computer vision-related applications, such as classification [8], detection, segmentation [9], and speech recognition [10].

As CNNs need up to 38 GOP/s to identify a single frame [11], this output is obtained at the expense of a high computational price. Hence, to accelerate their execution, dedicated hardware is required. The most frequently used platform for implementing CNNs is Graphics Processing Units (GPUs), as they provide the highest performance with respect to computational throughput, hitting 11 TFLOP/s [12].

Nonetheless, FPGA systems are considered to be better energy efficient in terms of power consumption (vs GPUs). Thereby, several FPGA-based CNN accelerators were suggested, targeting both data centers for High Performance Computing (HPC) [13] and embedded applications [14].

Although GPU implementations have shown exceptional computational efficiency, for two reasons, CNN acceleration is heading momentarily towards FPGAs. First, recent advances in FPGA technology have brought about FPGA performance with a recorded performance of 9.2 TFLOP/s for the latter in striking distance to GPUs. Second, recent CNN production patterns are increasing the sparsity of CNNs and using extremely compact types of data.

The remaining paper is structured in the following way: Literature Review, summarizing the use of AI algorithms in the past decade. Explanation of CNN and challenges of FPGA-based implementation. A detailed analysis of FPGA, GPU, and ASIC-based implementation of AI networks which compares between the power, area performance, and efficiency parameters. Finally, we look into the optimizations available and scope of future research.

2 Literature Review

Lately, the application of AI algorithm as a replacement of conventional algorithms have become popular. With the advent of ML and AI, there is a worry to basically address computational cost and power-burning through AI calculations. This leads to the necessity for particular equipment with high capacity to perform AI estimations with large scope issues [15]. The aim of all the research is to achieve better and more capable handling of AI algorithmic calculations. Numerous researches have been conducted over the span of the latest decade discussing hardware and programming advancements and execution strategies in this field [1, 16–21].

Table 1 A comprehensive compendium of survey papers

References	Publication year	Description and scope
[16]	2010	This research addresses all the main architecture methods and models for the 1990–2010 "hardware neural network." They exemplify numerous HNN implementations in a variety of "ANN models" like CNN, neural bursts, and so on. Digital, analog, composite, neuromorphic, "FPGA," or optical implementations are used in these HNN
[19]	2013	The scope of the research work was if developers were designing customized versions or were using pre-tailored free platforms for the development. The "artificial neural networks (ANN)" software framework table was also included in the work
[1]	2017	The paper provides a brief analysis of past work on the main approaches of FPGA-based neural inference networks
[21]	2018	This survey paper gives insight about the Moore's law in conjunction with the increasing use of CPUs and GPUs. Major applications of AI are also discussed
[22]	2020	This paper summarizes AI neural network's hardware implementation, with emphasis on all three-hardware equipment: ASIC, GPU, and FPGA. A comprehensive flatting strategy has been used to pick the research articles to address issues of science

Here in Table 1, we discuss corresponding survey articles published roughly between 2009 and 2019 on the implementation of AI algorithms, more precisely neural nets for hardware. Few studies have covered hardware implementations of artificial neural networks [16, 19, 20]. While other studies have centered on FPGA-based deep learning neural network accelerators [1, 17, 18]. In [19], the FPGA implementation of coevolutionary neural networks was the subject of the authors (CNNs). The survey addressed GPU implementations in [21].

This examination distinguishes and addresses various papers. These papers were evaluated as per the hardware used. The bulk of the documents include FPGA-based implementations. With its high adaptability and solidness, the FPGA is viewed as a promising option for the use of these calculations. Likewise, recent FPGA design improvements have brought about making it more accessible because of which profound learning research has procured significant consideration [23].

One of the powerful "application-specific integrated circuit" (ASICs) is also used for AI algorithm implementation. ASICs are personalized chips that are conceived for a particular purpose. They save high-power and are speed-efficient, consume less silicon area, making them ideal solutions for accelerating AI algorithms [24]. To accelerate AI algorithms, graphical processing units (GPUs) are also used to pace up algorithms to hundreds of times the initial speed [25]. GPUs are made to carry out scalar and parallel intensive computing [26]. Unlike multicore CPUs, when accessing DRAM memories, GPUs try not to depend on shrouded latencies utilizing large cache memories [27]. Such highlights have made GPUs increasingly more useful for AI acceleration.

Using small single-board processors, some AI algorithms are applied. For example, raspberry pi. As a result of their little size and low force prerequisites, single-board PCs are viewed as a decision for AI usage.

3 Overview of CNN

One of the classic profound learning networks is the deep convolutional neural network. They are widely used in these areas for the continued development of deep learning technologies, machine vision, and language recognition [28]. Earlier studies have indicated that the calculation of the cutting-edge CNNs is overwhelmed by the convolutional layers [29, 30] formation.

It contains numerous falling layers, pooled layers, and fully-connected complex layers. The convolutionary neural network distinguishes the image as the input, and obtains the outcome across several "convolutional layers, pooling layers and associated layers."

3.1 Model of Convolutional Layer

The input f^{in} and convolution kernel composed of weights w_{ij}, comprises the convolution layer. The sampled function is set by balancing results to get the output $f^{out.}$ [30].

$$f_i^{out} = \sum_{i=1}^{n_{in}} f_i^{in} * w_{i,j} + b_i,\ 1 \le i \le n_{out} \tag{1}$$

3.2 Model of Pooling Layer

The pooling layer typically uses the maximum scan or core scan to minimize the size of the input matrix. The activity will viably lessen the following layer's data processing ability while forestalling the loss of characteristic information.

3.3 Full Connection Layer

The layer changes over the input to straight space, hence, changing over the input to a direct space. Output is received.

$$f^{out} = \sum_{j=1}^{n} f_j^{in} * w_{i,j} + b \tag{2}$$

3.4 Activation Layer

A nonlinear change of the input is presented by the activation function excitation, and the output after each layer is regularly handled. Some common functions include nonlinear (Relu), trigonometric function (Tanh), shock response (Sigmod), etc.

4 Challenges in FPGA

4.1 Tradeoff Between RTL and HLS Approaches

Via high-level abstractions, the HLS-based design approach enables fast growth. This demonstrates the conventional plan utilizing the OpenCL-based methodology [31]. Key features including partitioning, automatic pipelining of the CNN model, etc., can also be supported but for HW performance, however, the resulting design cannot be optimized.

4.2 Necessity of Diligent Design

Considering the particular characteristics of both FPGAs and Convolutional Neural Net, the optimization of throughput demands careful design. In general, two architectures for the same use of resources may have significantly different performances [30],and thus, the use of resources may not be a reliable overall performance measure. There are vastly different criteria for separate CNN layers. In addition, the use of FPGA relies heavily on the burst duration of memory access [32], so the access pattern of CNN memory must be cautiously configured. Furthermore, the compute-throughput ought to be adjusted with the memory or the memory can end up being the bottleneck [19].

4.3 FPGAs Over GPUs/ASICs/CPUs

Although the computer software environments are already mature for developing convolutional Neural Net designs for CPUs or GPUs, those for FPGAs are even now nascent. Furthermore, despite the HLS software, it can take several months for an experienced HW designer to implement the CNN model on FPGAs [33]. Therefore, in general, modeling efforts for GPUs is quite less when compared to FPGAs. Accounting for rapid changes growth related to neural nets, it may not be feasible to re-architect accelerators based on FPGA with every upcoming neural net algorithm. Therefore, the most recent NN algorithm cannot efficiently model an FPGA-based architecture. In addition, a Field-Programmable Gate Array (FPGA) modeling demands greater resource overhead than an ASIC design because of reconfiguration of logic blocks and switches. These factors give these custom boards a competitive disadvantage over other HW acceleration computing platforms for NNs.

4.4 A Comparative Study Between FPGA, ASIC, and GPU

Here, we bring the execution technique, GPUs/FPGAs/ASICs, in relation to regular merit figures. A distinction between the three methods is seen in Table 2. Since the ASIC specification is unaltered post-production, it becomes unsuitable for application where subsequent to installation, the model needs to be revised. ASIC is better used where low power and area are the goal. In comparison to ASICs, FPGAs are available for post-implementation upgrades. It is worth mentioning that even though the ultimate objective is ASIC execution, FPGAs are also used for prototyping and validation [22].

GPUs provide a solution at the software level, while FPGAs and ASICs have a solution at the hardware level. FPGAs and ASICs thus have greater stability compared to GPUs during the deployment process. As a result, the implementation of the GPU

Table 2 A comparison between the aforesaid

	FPGA (Hardware)	ASIC (Hardware)	GPU (Software)
Skills required	Verilog	Verilog	High level language
Implementation level	Medium	High	Medium
Versatility	High	High	Low
Post implementation change	High	Very low	High
Overall expenditure	Medium	High	Low
Area consumption	High	Low	High
Performance	Medium	High	Low/Medium
Power consumption	Medium	Low	High

is constrained by the current underlying hardware. Thus, in some situations, FPGAs can be quicker than GPUs.

5 A Compendium of Numerous Hardware Implementations

Implementation details of neural nets on various FPGA boards are shown in Table 3. Loop optimizations were first investigated in [30] to extract an FPGA-based CNN accelerator. This design was modeled utilizing HLS tools, therefore, it relies on the arithmetic of 32 floating points. Works in [34, 35] pursue the same unrolling scheme. In addition, [35] design has 16 bits of fixed-point arithmetic and RTL design, which results in an increase of performance by approximately two times. In recent works [36], the same unrolling and tiling scheme is used where authors report an improvement of x13.4 over their original works [30]. Consequently, unrolling and tiling loops can be proficient as it were for devices with large computational capabilities (i.e., DSP). This is often illustrated in works of Rahman et al. [34] which improves speed by 1.2 times over [30].

Conversely, all modeling factors looking for ideal loop unroll are fully explored in the works of Ma et al. [37, 40, 41]. More specifically, researchers show that using unroll function for single input arithmetic, the input FMs and weights are optimally reused [38, 39, 42–46].

6 Conclusion and Future Work

In this paper, a comprehensive study related to development and deployment of FPGA in CNN accelerators has been provided. To highlight their similarities and distinctions, we categorized the works based on many parameters. We outlined the influential avenues for study and summarized the key themes of numerous works. Usage of more than one FPGA is important for computational acceleration of massive and extensive Neural Net models, considering the limited hardware resources available on an FPGA board. This allows the architecture to be partitioned across several FPGAs. Although a software for automatic partitioning will not be readily accessible, the manual approach to partitioning is vulnerable to error and unscalable. In addition, random splitting will increase the complexity of interconnections in a manner that the I/O pin count cannot complete the requirement of number of connections.

Lately, researchers have also reconnoitered models of the spike neural network (SNN) that model the biological neuron more closely [47]. This study concentrates on (ANN) which is an acronym for artificial neural network. SNNs and ANNs vary considerably. As a result, their training rules and network architectural structure

Table 3 Specifications of various hardware implementations

	Neural net	FPGA board	Frequency (MHz)	Performance (GOPs)	Consumed power (W)	LUT (K)	DSP	Memory (MB)
[30]	AlexNet-C	Virtex7 VX485T	100	62	19	186	2240	19
[14]	VGG16SVD-F	Zynq Z7045	150	137	10	183	780	18
[31]	AlexNet-C	Stratix5 GSD8	120	187	34	138	635	18
	AlexNet-F			72		272	752	30
	VGG16-F			118		524	1963	52
[34]	AlexNet-C	Virtex7 VX485T	100	75		28	2695	20
[36]	AlexNet-F	Virtex7 VX690T	150	826	126		14,400	
	VGG16-F			1280	160		21,600	
[35]	NIN-F	Stratix5 GXA7	100	114	20	224	256	47
	AlexNet-F			134	19	242	256	31
[40]	AlexNet-F	Virtex7 VX690T	156	566	30	274	2144	35
[41]	AlexNet-C	Virtex7 VX690T	100	62		273	2401	20
[37]	VGG16-F	Arria10 GX1150	150	645	50	322	1518	38
[42]	AlexNet-F	Arria10 GT1150	240	360		700	1290	47
	VGG-F		222	460		708	1340	49
	VGG-F		232	117		626	1500	33
[43]	AlexNet-C	Cyclone5 SEM	100	12		22	28	1
		Virtex7 VX485T	100	445			2800	
[39]	NiN	Stratix5 GXA7	150	283		453	256	30
	VGG16-F			352		424	256	44
	ResNet-50			251		347	256	39
	NiN	Arria10 GX1150	200	588		320	1518	31
	VGG16-F			720		263	1518	45
	ResNet-50			619		437	1518	39
[38]	AlexNet-F	Virtex7 VX690T	100	446	25	207	2872	37
	VGG16SVD-F			473	26	224	2950	47

vary drastically. Large-scale SNN modeling onto SOC/FPGA boards will offer an exhilarating and remunerating challenge for IT designers later on [45].

References

1. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*.
2. Pau, L. F. (1991). Artificial intelligence and financial services. *IEEE Transactions on Knowledge and Data Engineering*.
3. Yao, X., Zhou, J., Zhang, J., & Boer, C. R. (2017). From intelligent manufacturing to smart manufacturing for industry 4.0 driven by next generation artificial intelligence and further on. In *Proceedings—2017 5th International Conference on Enterprise Systems ES*.
4. Bishnoi, L., & Narayan Singh, S. (2018). Artificial intelligence techniques used in medical sciences: A review. In *Proceedings of 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*.
5. Parker, D. S. (1989). Integrating AI and DBMS through stream processing.
6. Fraley, J. B., & Cannady, J. (2017). The promise of machine learning in cybersecurity. In *Conference of Proceedings—IEEE SOUTHEASTCON*.
7. Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*.
8. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*.
9. Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
10. Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Laurent, C., Bengio, Y., et al. (2016). Towards end-to-end speech recognition with deep convolutional neural networks. In *Proceedings of Annual Conference of the International Speech Communication Association, INTERSPEECH*.
11. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings*.
12. Nurvitadhi, E., Venkatesh, G., Sim, J., Marr, D., Huang, R., Ong, J. G. H., et al. (2017). Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In *FPGA 2017—Proceedings 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*.
13. Ovtcharov, K., Ruwase, O., Kim, J., Fowers, J., Strauss, K., & Chung, E. S. (2015). Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Research Whitepaper*.
14. Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., et al. (2016). Going deeper with embedded FPGA platform for convolutional neural network. In *FPGA 2016—Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*.
15. Rigos, S., Mariatos, V., & Voros. N. (2012). A hardware acceleration unit for face detection. In *2012 Mediterranean Conference on Embedded Computing*.
16. Misra, J., & Saha. I. (2010). Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*.
17. Baji, T. (2018). Evolution of the GPU device widely used in AI and massive parallel processing. In *2018 IEEE Electron Devices Technology and Manufacturing Conference EDTM 2018—Proceedings*.
18. Shawahna, A., Sait, S. M., & El-Maleh, A. (2019). FPGA-based accelerators of deep learning networks for learning and classification: A review.
19. Mittal, S. (2020). A survey of FPGA-based accelerators for convolutional neural networks. *Neural Computing & Applications*.

20. Guo, K., Zeng, S., Yu, J., Wang, Y., & Yang, H. (2017). [DL] A survey of FPGA-based neural network inference accelerator.
21. Blaiech, A. G., Ben Khalifa, K., Valderrama, C., Fernandes, M. A. C., & Bedoui, M. H. (2019). A survey and taxonomy of FPGA-based deep learning accelerators. *The Journal of Systems Architecture.*
22. Talib, M. A., Majzoub, S., Nasir, Q., & Jamal, D. (2020) A systematic literature review on hardware implementation of artificial intelligence algorithms. *The Journal of Supercomputing.*
23. Schneider, S., Taylor, G. W., Linquist, S., & Kremer, S. C. (2019). Past, present and future approaches using computer vision for animal re-identification from camera trap data. *Methods in Ecology and Evolution.*
24. Faraone, J., Gambardella, G., Fraser, N., Blott, M., Leong. P., & Boland, D. (2018). Customizing low-precision deep neural networks for FPGAs. In *Proceedings—2018 International Conference on Field Programmable Logic and Applications FPL.*
25. Cheng, K. T., & Wang, Y. C. (2011). Using mobile GPU for general-purpose computing a case study of face recognition on smartphones. In *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test VLSI-DAT 2011.*
26. Ouerhani, Y., Jridi, M., & AlFalou, A. (2010). Fast face recognition approach using a graphical processing unit "GPU." In *2010 IEEE International Conference on Imaging Systems and Techniques IST 2010—Proceedings.*
27. Li, E., Wang, B., Yang, L., Peng, Y. T., Du, Y., Zhang, Y., et al. (2012). GPU and CPU cooperative acceleration for face detection on modern processors. In *Proceedings—IEEE International Conference on Multimedia and Expo.*
28. Lu, L., Liang, Y., Xiao, Q., & Yan, S. (2017). Evaluating fast algorithms for convolutional neural networks on FPGAs. In *Proceeding—IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines FCCM 2017.*
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*
30. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *FPGA 2015—2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.*
31. Suda, N., Chandra, V., Dasika, G., Mohanty, A., Ma, Y., Vrudhula, S., et al. (2016). Throughput-optimized openCL-based FPGA accelerator for large-scale convolutional neural networks. In *FPGA 2016—Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.*
32. Zhang, C., Fang, Z., Zhou, P., Pan, P., & Cong, J. (2016). Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. In *IEEE/ACM International Conference on Computer-Aided Design Digital Technical Paper ICCAD.*
33. Guan, Y., Liang, H., Xu, N., Wang, W., Shi, S., Chen, X., et al. (2017). FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. In *Proceedings—IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines FCCM 2017.*
34. Rahman, A., Lee, J., & Choi, K. (2016). Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array. In *Proceedings of 2016 Design, Automation & Test in Europe Conference & Exhibition DATE 2016.*
35. Ma, Y., Suda, N., Cao, Y., Seo, J. S., & Vrudhula, S. (2016). Scalable and modularized RTL compilation of Convolutional Neural Networks onto FPGA. In *FPL 2016—26th International Conference on Field-Programmable Logic and Applications.*
36. Zhang, C., Wu, D., Sun, J., Sun, G., Luo, G., & Cong. J. (2016). Energy-efficient CNN implementation on a deeply pipelined FPGA cluster. In *Proceedings of International Symposium on Low Power Electronics and Design.*
37. Ma, Y., Cao, Y., Vrudhula, S., & Seo, J. S. (2017). Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks. In *FPGA 2017—Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays.*

38. Liu, Z., Dou, Y., Jiang, J., Xu, J., Li, S., Zhou, Y., et al. (2017). Throughput-optimized FPGA accelerator for deep convolutional neural networks. *ACM Transactions on Reconfigurable Technology and Systems.*
39. Ma, Y., Cao, Y., Vrudhula, S., & Seo, J. S. An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks. In *2017 27th International Conference on Field-Programmable Logic and Applications FPL.*
40. Li, H., Fan, X., Jiao, L., Cao, W., Zhou. X., & Wang. L. (2016). A high performance FPGA-based accelerator for large-scale convolutional neural networks. In *FPL 2016—26th International Conference on Field-Programmable Logic and Applications.*
41. Alwani, M., Chen, H., Ferdman, M., & Milder, P. (2016). Fused-layer CNN accelerators. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).*
42. Wei, X., Yu, C. H., Zhang, P., Chen, Y., Wang, Y., Hu, H., et al. (2017). Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In *Proceedings of the 54th Annual Design Automation Conference 2017*.
43. Motamedi, M., Gysel, P., & Ghiasi, S. (2017). PLACID: A platform for FPGA-based accelerator creation for DCNNs. *ACM Transactions on Multimedia Computing, Communications, and Applications.*
44. Ma, Y., Kim, M., Cao, Y., Vrudhula, S., & Seo, J. S. (2017). End-to-end scalable FPGA accelerator for deep residual networks. In *Proceedings—IEEE International Symposium on Circuits and Systems.*
45. Maguire, L. P., McGinnity, T. M., Glackin, B., Ghani, A., Belatreche, A., & Harkin, J. (2007). Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing.*
46. Negi, A., Saxena, D., & Suneja, K. (2020). High level synthesis of chaos based text encryption using modified Hill Cipher algorithm (pp. 3–7).
47. Thapa, S., Adhikari, S., Naseem, U., Singh, P., Bharathy, G., & Prasad, M. (2020). Detecting Alzheimer's disease by exploiting linguistic information from Nepali transcript. *Communication in Computer and Information Science.*