Journal of Distributed Computing and Systems(JDCS), Vol. 4, Issue 2, 2021, Pages 46-57.

A Review on the Development of GAN Techniques in convolutional Neural Networks and its implementation on FPGA accelerator

Hossein Ali Bagheri*, Bahador MakkiAbadi, Mohammad Hossein Arastoo
Faculty of Electronics, Azad University of Ashtian, Ashtian, Iran.

Article History:

Received: 19 November 2021
Received in revised form: 25 January 2022
Accepted: 15 February 2022
Available online: 17 March 2022

Abstract

Recently, the growth of convolutional neural networks in various scientific fields can be seen dramatically. The use of various software and hardware techniques in advancing this process provides the platform for increasing research and finding different solutions to increase the efficiency and optimization of this method. One of the important techniques in the field of neural networks is the Generative Adversarial Networks (GAN) and its implementation on FPGA accelerators. In this paper, we will provide an overview of the growth process of convolutional neural networks with using GAN technique and its implementation on FPGA accelerator over two years. In this series we intend to follow some of the most primitive projects starting almost 2017 and by the end of 2018, where significant progress can be made. In this work, we review five papers, the first of which is presented in 2017 and the other four in 2018. The method of comparing these articles is characterized by four distinct perspectives: Optimal utilization of accelerator resources, application of specific techniques, analysis of generated data, and finally the speed of execution in FPGA-based systems is a requirement. Finally, we discuss the advantages and disadvantages of these designs to optimize and improve their performance.

Keywords: Generative Adversarial Network (GAN), Convolution Neural Networks, FPGA, Accelerator Resources.

I.INTRODUCTION

Much work has been done so far on the use of deep neural networks in the context of processing activities such as image, speech , etc. [1] [2] [3] [4] [5] [6] [7]. Many hardware tools have been provided to implement this optimally. Graphics processing units (GPU) are one of those tools in deep neural networks. In this case, a lot of software tools and libraries are provided for GPU. Recently, FPGA has been used as an alternative to existing hardware due to its low power consumption and lower cost and lower latency than graphics processing units [15]. There are, of course, other advantages and disadvantages that can be explained by its use in different situations. This is a clear and plausible reason for using FPGA in current accelerators. Users are facing the big challenge of using deep neural networks, and that is the need for deep neural networks to large datasets. To meet this challenge, a new class of neural networks, namely Generative Adversarial Networks, is introduced [8]. Generative Adversarial Networks automatically generate larger and richer datasets. This class of neural networks can be used in a variety of applications, including robotics [9], Autonomous Driving [10], media synthesis [11] and medical [12]. Generative Adversarial Networks are made up of two parts: the generative model that generate the data and a discriminative model that is a typical neural network model that discriminate the original and generated data. In this class, both models reinforce each other [8]. Despite the Significant growth of this class of neural network and its increasing use in various cases, the accelerators required for this growing technique have not yet seen significant progress. In the following sections, we intend to discuss various articles on the implementation of this style of neural networks on known accelerators, especially FPGAs, and explore the strengths and weaknesses of these types of schemes.

In the following sections, various articles on the implementation of this type of neural networks on known accelerators, especially FPGAs, are reviewed. Finally, constructive ideas for the growth and enhancement of this model of implementations are presented.

II.THE METHODS TO DO THE WORK

Most review articles have attempted to explore different articles in a specific context by different methods. In this regard, one of the techniques of exploring, is visiting international sites and reviewing authoritative articles on the subject in question. Existing search engines such as Google can help researchers in this field. In the same way, this paper

*Corresponding Author: bagheri.h@aiau.ac.ir

has followed a research work and has been able to discover articles on the topic at various times. The first article in this search (2017) is an article on "A Design Methodology for Efficient Implementation of Deconvolutional Neural Networks on an FPGA" [13]. This is the most elementary work of implementing GAN techniques on FPGAs.

In the next four articles, all of which have been published in 2018, we can see some improvements to the quality of implementation of this technique on the FPGA and the completion of the implementation process with Headlines " Towards Efficient Microarchitectural Design for Accelerating Unsupervised GAN based Deep Learning " [14]," B-DCGAN: Evaluation of Binarized DCGAN for FPGA"[15]," FlexiGAN: An End-to-End Solution for FPGA Acceleration of Generative Adversarial Networks"[16] and "LerGAN: A Zero-free, Low Data Movement and PIM-based GAN Architecture"[17].

The articles listed above are discussed in detail in the Findings section of this article with a shared perspective.

Lastly, the remaining problems in these projects are expressed and analyzed separately.Contributions are to be in English. Authors are encouraged to have their contribution checked for grammar. American spelling should be used. Abbreviations are allowed but should be spelt out in full when first used. Integers ten and below are to be spelt out. Italicize foreign language phrases (e.g., Latin, French).

The text is to be typeset in 10 pt Times Roman, single spaced with line spacing of 13 pt. Text area (excluding running title) is 5 inches across and 7.7 inches deep. Number each page of the manuscript lightly at the bottom with a blue pencil. Reading copies of the paper can be numbered using any legible means (typewritten or handwritten). Final pagination and insertion of running titles will be done by the publisher.

III.FINDINGS

In this paper, the findings and their analysis are examined from several perspectives. These views are discussed below in each of the five articles and finally analyzed separately. It is natural that the strengths and weaknesses of the implementation of each of the headings in each article vary, as each article focuses on a limited number of optimization parameters.

A. Optimal Utilization of Accelerator Resources

The purpose of this section is to point out the optimal utilization of the resources available in the accelerators, including FPGAs, according to the techniques presented in each article. In other words, in order to minimize the inefficient resources of existing accelerators, it is addressed in the research articles reviewed in this article.

In [13], the first GAN implementation paper on FPGA, no significant effect of an efficient approach is observed for optimal utilization of accelerator resources. In practice, however, one can see a practical implementation of the FPGA accelerator, which may not be relevant to this section of the article, and will be discussed in the following sections. There are several issues to consider when designing an FPGA-based convolution and deconvolution accelerator, which the article [13] is no exception. First, a direct translation of optimized deconvolution CPU algorithms into an FPGA, which will generally lead to inefficient implementation. Therefore, in order to achieve high performance with low execution complexity, it is necessary to adapt the deconvolution function to a hardware platform such as FPGA. In addition, recent research has shown that discriminative CNNs are robust to low bitwidth quantization [19] [20]. However, it is necessary to systematically investigate the effects of reducing such bitwidth on product quality degradation from a generative model, such as the deconvolution neural network (DCNNs) implementation on the FPGA. Therefore, it is necessary to use criteria that determine the effect of such approximations on the deconvolution neural network in order to achieve an efficient design. To address the items listed above, the following process is effective in [13]. 1) Create a deconvolution accelerator with reverse loop and stride hole skipping to efficiently implement deconvolution on an FPGA. This solution uses, indirectly, the same computational architecture presented for implementing a convolution accelerator in [21]. 2) The paper presents a three-step method for deconvolution accelerator design, which we will mention in the specific models section. Finally, the paper confirms its work via two implementations on Xilinx Zynq-7000 FPGA.

The paper [14] illustrates the following challenges in the design of the GAN accelerator, which is entirely related to the analysis of the GAN training method. First, synchronization (to loss calculation) in GAN training not only requires a lot of memory to store medium data, which in turn will lead to inefficient accelerator resources, but also limit parallel optimization. Second, the abundant and variable computational phases and abnormal convolution operations in these phases further complicate the design of the accelerator. In [14], paper overcome these challenges with an algorithm and architecture co-design design approach. It first delays the synchronization operation until the batch processing is completed. Then proposes a time- multiplexed design to effectively represent the computation steps to the provided microarchitecture. Which can be called "Zero Free and Output STationary Microarchitecture" (ZFOST), which is responsible for forward data pass and backward error pass calculation and weight updating operations, which "Zero Free and Weight STationary Microarchitecture" (ZFWST) processes it [14]. By combining high-performance dataflow and new architectures, zero-operative multiplications in non-traditional convolution are omitted without losing data reuse. This is one of the most important parameters to reduce resource inefficiency or optimize resource utilization. Due to its good performance, energy efficiency, and reset capability in FPGA, this paper selects FPGA as a platform to demonstrate the design of the GAN accelerator.

The paper [15] optimizes the use of accelerator resources by using a specific technique. This method is implemented whit using binary weights and activation and whit using integer values operations in forward pass (training- time and run- time) in [15]. Binarization and using the integer-valued reduces the need for high memory capacity and the number of circuit gates, and is very effective in FPGA implementation. On the other hand, the quality of the data

generated from the model is reduced by these simplifications. However, these proposed methods are due to the use of FPGA accelerators, which may require the same amount of quality in many Works. FPGAs, with their enormous merits, as mentioned in the introduction, also have the disadvantages that researchers seek to eliminate with using proposed techniques. One of the most important problems can be said to be the resources inefficiency in accelerators. Other work has tried to use techniques such as binary, to overcome these shortcomings and other problems [27] [28] [29] [30].

In [16], an excellent method is proposed for the proper use of accelerator sources and the correct use of FPGA internal memory, due to the presence of zero- inputs in the generator segment, which is called the transposed convolution. It should be noted that good work has been done in the field of CNN accelerators on the FPGA [38] [41] as well as research on dataflow optimization techniques for CNNs [39], [42] which has fueled this optimization process. It may be important to point out that the main focus of these studies is the forward convolution operator. However, GANs, the next wave of emerging deep networks, are involved in transposed convolution. As noted earlier, while the CNN accelerator can still be used for GANs [37] [40], the insertion of zeros in transposed convolution results resource inefficiency and scarcity. Previously [36], uses Vivado HLS tools to generate an implementation of deconvolution (transposed convolution) that uses high-level code and explores the effects of bitwidth reduction. But since this article does not provide architecture, its comparison is quite unnatural. In addition, [36] is purely for the deconvolution method. Unlike previous works, the paper [16] offers a proprietary comprehensive solution from high-level descriptions to an artificial GAN accelerator that encompasses both conventional convolutions and deconvolutions, and can work for a wide range of deep neural networks, including previous unsupported GANs.

In the last article reviewed [17], the problem of optimal utilization of accelerator resources is one of the major challenges in GAN. In order to confirm the interaction between the two generator and discriminator models, a large amount of mediator data is needed to communicate frequently between the two models. Since the chip's internal memory space for storing intermediate data is quite limited, GAN training puts extra pressure on access to off-chip memory, which consumes almost twice as much power as a floating point operation [56]. Therefore, these big data movements become a bottleneck for the design of the GAN system. The paper [17] presents plans to address this challenge, which is one of the major achievements in this field. One might argue that this article has been one of the best solutions available to address this issue. The main solution to the problem in this section is the elimination of zero calculations. Paper [17] has provided methods in this area that we will discuss fully in the section "application of specific techniques".

B. Application of Specific Techniques

The second Part of this article refers to a specific technique or model for optimizing the design of the proposal in each article that uses scientific methods separately. The intelligent workflow is used in the paper [13], despite the introduction of GAN implementation on the FPGA. It should be noted that the current FPGA accelerators focus on increasing the performance of convolution neural networks (CNN) instead of deconvolution neural networks (DCNN) And from CNNs according to the "downsample" the input are used in discriminator for classification and also from DCNNs according to the "upsample" the input in generator are used to deconvolution layers for data generation [18].

In this regard, the paper offers three techniques: A) At the highest level of design, it trains DCNNs using the generative adversarial network (GAN) method [8] and uses statistical tests to analyze the quality of generative under various bandwidth accuracy, To select the most cost-effective bandwidth. B) The paper uses the roofline model proposed in [21] to explore the design space in order to find a set of high-level constraints that reaches the best tradeoff between the memory bandwidth and the accelerator's throughput. C) To optimize performance, it uses loop unrolling and pipelining, memory partitioning, and register insertion.

The paper [14] introduces two techniques presented in his design very efficiently. This proposed approach includes two novel Microarchitectures (ZFOST and ZFWST) and high-efficiency dataflow for non-traditional convolutions: S-CONV, T-CONV and W-CONV. This article notes that this microarchitectural optimization can also be widely used in the traditional CNN training procedure. Of course, one of the strengths of this technique is to eliminate zero-operative Multiplications in non-traditional convolutions without losing data reuse. This is a common technique in all advanced accelerators. This is a common technique in all advanced accelerators, which can be seen by different implementation methods and names. It should be noted that the presence of zero operands per accelerator is a factor in the inefficiency of its resources and reduces the speed of the accelerator.

The paper [15] presents specific methods for his work. DCGAN [31] is a GAN training method for image generation using CNN. [15] For optimization, this task introduces Binary-DCGAN (B-DCGAN) using integer-valued in the forward path. This technique is briefly mentioned in the previous section. Generally speaking, the B-DCGAN is a version of DCGAN that its Generator has binary weights, binary activations, and integer-valued operations. The Discriminator, too, is a vanilla network as the normal DCGAN. It should be noted that B-DCGAN uses conditional input method such as CGANs [32].

There are two techniques in [16]. The first is the MIMD-SIMD processing [16], which has a low-overhead MIMD-SIMD architecture that enables switching between these two modes of parallelism at the granularity of each individual operation. This combination is especially necessary for GANs because the accelerator must selectively jump over the irregularly inserted zeros in a deconvolution. It is important to note that the accelerator rebounds back to full SIMD mode for forward convolution. Previous work has investigated the advantages of combined MIMD-SIMD acceleration for various applications [43] [45] [44] [46], but lacks designs that combine the two models for deep neural networks, especially GANs. The second technique is Decoupled data retrieval and data processing. However, this paper argues that the main work by James Smith [47] was presented the concept of

separating data retrieval from data processing. Some other work [48], [49] follow this pattern. There are two major differences in the architecture proposed in this paper with previous work. (1) Data retrieval and data processing operations in these works are at the coarse granularity of kernels and functions. (2) They generally perform operations in a dataflow or SIMD method. This work extends the mentioned pattern to the best computational granularity for each individual accelerator operations.

In [17], to solve the memory problem in GAN training, researchers proposed ReRAM-based Processing in Memory (PIM) [59] [58] [60] [57] [61], which exposes energy efficiency in reducing memory access cost compared with CPUs and GPUs. In addition, it can complete a matrix multiplier vector (MMV) operation in almost one read cycle with low energy expenditure. Since MMV operations dominate the computational patterns of GAN training, ReRAM-based PIM technologies have the potential to significantly reduce memory access costs and speed up GAN training [17]. However, GAN has two main features that are different from traditional neural networks: (1) zero-insertion during training phase; (2) complex dataflow patterns between the two models (generator and discriminator). These two features reduce the performance of the PIM-based accelerator for GAN. Firstly, zero insertion creates a heavy burden on storage. Second, I / O traffic becomes a system bottleneck because, a) the interaction between the generator and the discriminator requires more communication through I / O s in the PIM. B) There is an irregular dependence between complex dataflow of GAN. Therefore, limited I/O bandwidth stalls GAN training. [17]. To address these challenges in PIM-based GAN architecture, first a new technique is proposed, which is software-managed with "Zero Free Data Reshaping" (ZFDR) scheme To eliminate all zero operations generated by GAN. Then, a reconfigurable 3D connectivity architecture is presented, which not only fits in with the complex GAN dataflow , but also supports efficient ReRAM reads and writes, and greatly hides the I / O overhead. Putting ZFDR and reconfigurable 3D interconnection architecture together, in this paper is proposed LerGAN, a ReRAM-based 3D GAN accelerator that maps data processed by ZFDR to 3D-connected PIM. By doing so, it not only achieves higher performance than I / O, but also enables flexibly I / O connection configuration for complex dataflow in GAN training [17].

C. Analysis of Generated Data

In this section, the quality of each article has been analyzed separately and suggestions for improvement have been provided. In the paper [13], the generated data from two perspectives are examined. In this work, an analytical framework for comparing and compromising the image quality and complexity of implementation over a wide range of bitwidth is presented. Fig. 1 shows some faces and digits produced from trained DCNNs in this work.

Fig. 1. Sample MNIST and CelebA images generated by the full precision DCNN [13].

Figure 2 shows the output of DCNNs using different bitwidths for the same input. Visual evaluation of image degradation is possible, at least at very low bitwidths such as 8 bits.

Fig. 2. Images generated by different bitwidth DCNNs [13].

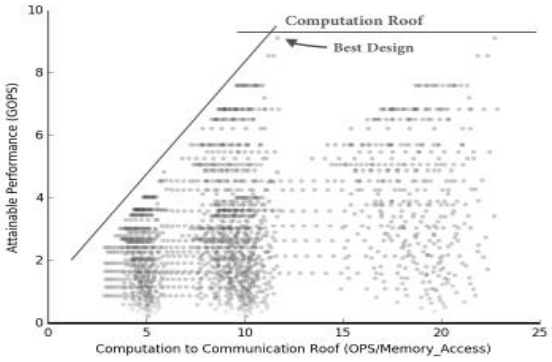

Fig. 3. Design space Exploration for a layer with input 10x2x2 and output 64x4x4[13].

The acceptable range for CelebA expressed by the desired DCNN by a graph can be seen in Figure 3. The best drawing in the diagram, top left, is introduced.

In this work, two tables are presented that analyze the performance of the hardware system. Table 1 shows the extent of utilization of the various parts of the FPGA.

TABLE I. FPGA RESOURCE UTILIZATION [13]

DSP	LUT	FF	BRAM
95%	48%	29%	48%

Table 2 “see Appendix 1 for table 2”. compares the performance of the desired DCNN with some of the existing CNN accelerators in the reference. This article claims that by implementation a ping pong buffer on the system, it can create more performance and improve it.

In [14], the proposed GAN accelerator with DCGAN implementation is shown in Figure 4. This paper utilizes Xilinx VCU118 Evaluation board that includes two sets of five 512MB DDR4 SDRAM and a Xilinx UltraScale+ XCVU9P FPGA, which consists of 2586K Logic Cells, 6840 DSP Slices, and 75.9 Mb Block RAMs.

In the article [14], in the following sections, we see phrases PE and WST. Since this type of architecture remains the same kernel weights in PEs (Processing Elements) while receiving new input neurons, it is named as WST (Weight STationary). [23], [24] exploit WST types in their work. In the synchronization mechanism, a mass of computational phases and non-traditional convolutions, lead to inefficiency when the accelerators in CNN are applied directly to GAN training. Figure 5 shows the synchronization process.

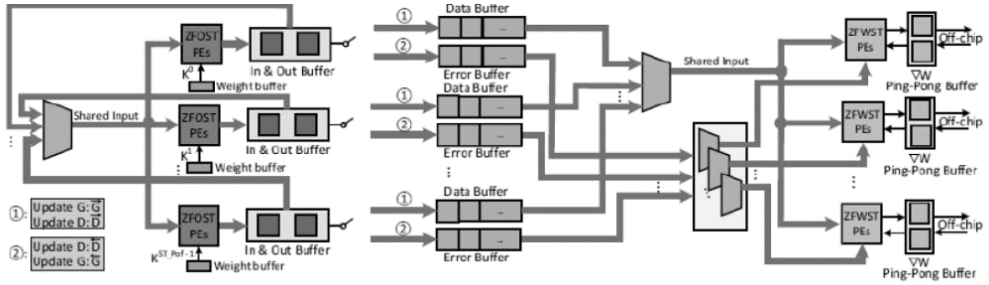

Fig. 4. GAN Accelerator Design [14]

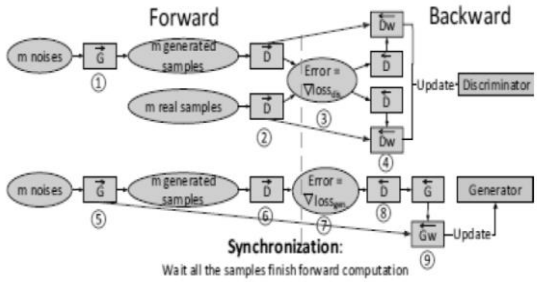

Fig. 5. GAN training computational flow [14]

Under synchronization, in Figure 6, the ZFOST architecture outperforms the synthetic architecture of this article (ZFOST-ZFWST) because synchronization Leading to an architecture (ZFOST or ZFWST) being presented in the synthetic architecture, which works at any time. With optimization, the two architectures in the synthetic design can be run in parallel, and the performance of the synthetic architecture (NLR-OST and ZFOST-ZFWST) is significantly improved, while the non- synthetic architecture performance is the same. Among the synthetic architectures, ZFOST-ZFWST is better than NLR-OST due to zero-free optimization in all computation steps. Therefore, the ZFOTS-ZFWST architecture in Figure 7 achieves the best speed (average 4.3x).

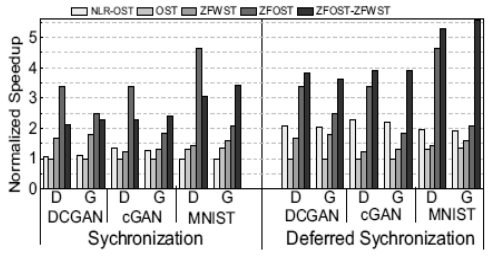

Fig. 6. Comparison of overall performance between different computing architectures [14]

In Figure 7, ZFOST-ZFWST always performs best performance with different PEs. In addition, by leveraging 512 PEs, ZFOST-ZFWST, it achieves similar performance to NLR-OST and ZFOST with 1024 PE

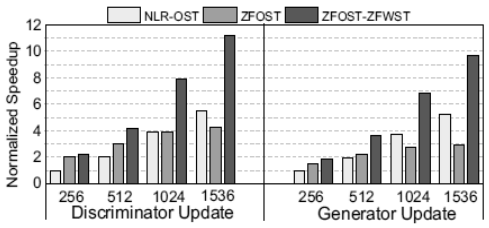

Fig.7. Performance diversity with different types of PE [14]

The evaluation results show that the proposed design achieves the best performance (average 4.3x) among the conventional deep learning accelerators. Desired Accelerator in [14] has an average of 8.3X speedup and 45.2X energy-efficiency over CPU. It also achieves an average of 5.2X energy-efficiency over Titan X and 7.1X energy-efficiency over K20 [14]. In the paper [15] they have conducted a series of experiments on Theano [33] / Lasagne [34] libraries, which demonstrate that it is possible to binarize the DCGAN model, and shows that binarization is acceptable along with maintaining output quality. According to the results of the experiments carried out in this paper, the last layer of B-Deconv-2 mainly has an initiative for output image quality. So the last layer must work in real-valued process. Other than the last layer, other layers can be fully binary [15]. More specifically, the MNIST dataset [35] is generally used for training. Python has been used for this training program using the Theano library [33] and the

Lasagne library [29] and Part of this has been done using BinaryNet code [29] and DCGAN code [31]. Runs the application on Ubuntu Linux with NVIDIA Titan-X GPU. The application code is available online at (https://github.com/hterada/b-dcgan) [15]. In the remainder of this paper, the training is performed using a stochastic gradient decent (SGD) and mini-batches of size 128 and an initial learning rate of 0.0001, which was linearly decreased down to 0 with the decay step of $0.0001/3000=3x10^{-8}$. Using visual evaluation for the quality of the output images of the generator, the quality of the training of each scenario is evaluated to the end of the process. This means that in each training iteration, the program writes the output and current image parameters of the generator to different files at the time, so these features can be checked to achieve the proper quality. For example, Figure 8 shows the output images in the middle of the training and at the training peak. As work has shown, quality is also increasing with respect to repetition of training.

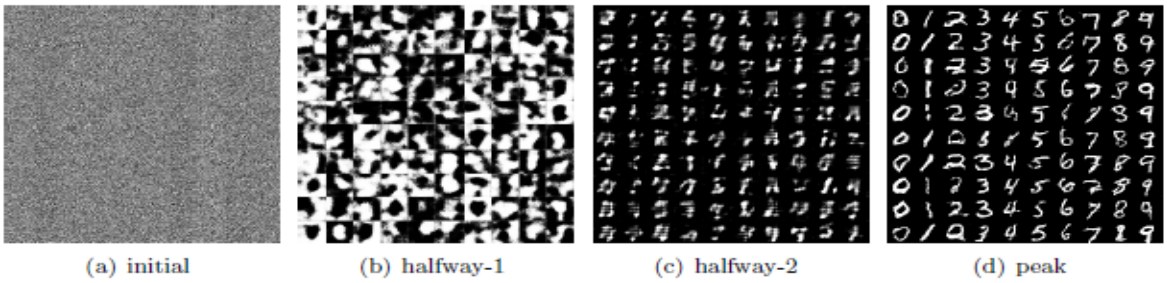

Fig. 8. Generator's output images in the middle of training

Therefore, qualitative growth and acceptable performance can be seen in this article. Of course, despite the binary technique in this work, the image quality is not very good, which is a matter of debate. Article [16] introduces the FPGA-based accelerator with the mentioned techniques in this paper as FlexiGAN-FPGA and the conventional accelerator as Conv-FPGA. Table 3 “see Appendix 1 for table 3”. summarizes the FPGA and GPU specifications.

Paper [16] uses several advanced GANs to evaluate the effectiveness of FlexiGAN, including 3D-GAN [50], ArtGAN [51], DCGAN [52], DiscoGAN [53], GP-GAN [54] and MANAN [55]. Table 4“see Appendix 1 for table 4” shows the details of the evaluated GAN models. These GAN models are used for various applications including text-to-image synthesis, high-resolution image generation, music synthesis and 3D object generation.

Table 5“see Appendix 1 for table 5” shows the amount of FPGA (XCVU13P) the resource utilization in the generated accelerator for DC-GAN using the FlexiGAN workflow [16]. The amount of resources used is limited to the amount of memory available on the chip. The number of entrances for global and local instruction buffers is 512 and 128 respectively. In the last step, the number of entries for each retrieval data FIFO and buffers between data retrieval and data processing units is fixed at 32. Under this arrangement, and due to the limited amount of on-chip memory, this paper can only map 1560 computational engines (CE) on its evaluated FPGA [16].

Figure 9 shows the accelerator speed of this article (FlexiGAN-FPGA) and the Titan X GPU Normalized to Conv-FPGA. As you can see in the Fig. 9, the FlexiGAN-FPGA delivers 2.2 x higher speedup than the Conv-FPGA. The main source of speed is FlexiGAN's ability to efficiently bypass a large percentage of zero values in TranConv layers. Compared to the Titan X, the number of compute nodes and clock frequency in FlexiGAN are 2.0x and 5.0x lower, respectively. However, because of the new pattern architecture design for inefficient operation bypass, FlexiGAN is, on average, only about 2.2x slower [16].

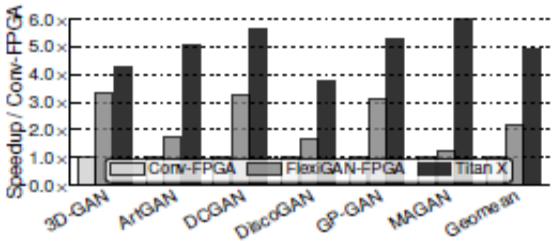

Fig. 9. Speed Boost with FlexiGAN-FPGA and GPU (Titan X) vs. Conv-FPGA [16].

As expected, the FlexiGAN-FPGA performs better than the Conv-FPGA in Performance-per-Watt with the same speedup pattern; since they are both based on an FPGA platform but the FlexiGAN-FPGA is significantly faster. However, in order to provide a better image with respect to the provided merits for FlexiGAN, Fig. 10 shows the Performance-per-Watt improvement with FlexiGAN on a high quality GPU baseline for each GAN model. The average performance-per-watt improvement for FlexiGAN is 3.5x better. FlexiGAN achieves significant energy efficiency by eliminating inefficient functions and minimizing Van Neumann's overhead of fetching and instruction decoding [16].

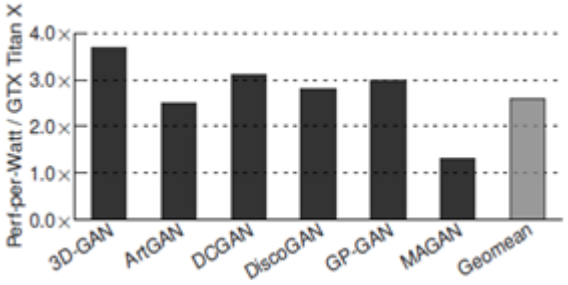

Fig. 10. Performance-per-Watt, FlexiGAN-FPGA over GPU (Titan X).

Despite the original title of the review article entitled Implementation of GAN Techniques on the FPGA, the presented implementation in the article [17] is PIM-based. Of course, with a specific technique that seems to have no connection with the main subject of this article. But the reason for reviewing this article [17] and introducing it as one of the articles reviewed in this study is due to two reasons. First, this article is finally compared to the implemented techniques on the FPGA and its strengths and weaknesses are examined. Second, one of our goals is Implementation of the article [17] on FPGA in the future. In the [17] article, the networks with the values listed in Table 6"see Appendix 1 for table 6" with batch size 64 are fully trained and the results are shown as follows.

[17] First examines the effectiveness of the proposed ZFDR and 3D connection mechanisms in this paper. It then compares the performance and energy between LerGAN and the alternative PIM design such as PRIME. In addition, it compares LerGAN with the FPGA-based GAN accelerator and GAN running on the GPU platform. PipeLayer [42] uses H-tree for tutorials that require reading and writing extensive memory to update kernel weight. To solve the problem of inefficient inputs and outputs, a PIM with 3D connection is proposed, with the aim of efficiently matching the GAN training dataflow. It should be noted that 2D and D3 are used respectively to illustrate the design of the H-tree and 3D connection and examine configurations with varying degrees of duplication (i.e. low, medium and high). To better understand the problem of duplication, if N1 is a relatively small GAN, small GAN parallelization is used to achieve a larger GAN such as N2 (i.e. duplicating kernel weight for several times).

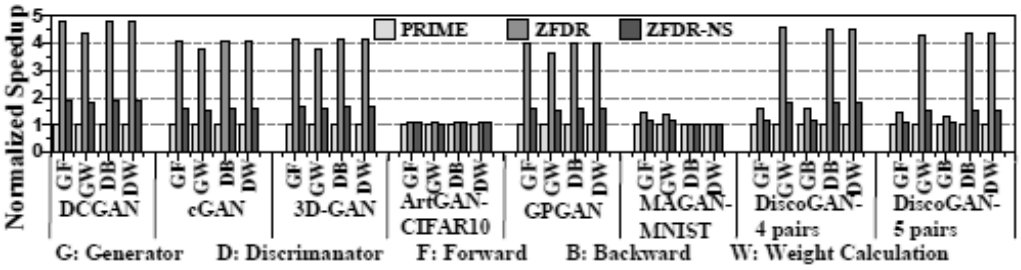

Fig. 11. Comparison of performance in phases of using ZFDR with PRIME [17]

Figure 11 shows the performance of ZFDR at different stages of GAN. ReRAM-based PIM includes RRAM arrays and peripheral circuits. It is important to note that ReRAM arrays can be configured to support MMV (called CArrays in this article). NS is used to denote the normalized space, meaning that PRIME uses the same CArray space in [17]. ZFDR achieves significant speedup on DCGAN, cGAN, 3DGAN, GPGAN, and DiscoGAN, Which indicates that there are large amounts of zero in these GANs. Note that DiscoGAN-4pair has 5 steps using ZFDR because its generator has S-CONV and T-CONV. By the way, no high speed can be observed in the MAGAN-MNIST discriminator, because its layers are fully connected. When evaluating the entire GAN training process with H-tree connectivity, the ZFDR speed is almost ignored. This is due to overhead of data transfers. Figure 12 shows the 3D connection design performance compared to the H-tree connection.

We can see that with the 3D connection design, the ZFDR speed is very evident. In addition, 3D connection with duplication (low-degree) has a much higher performance rate than ZFDR without duplication, while duplication with H-tree connection is slow.

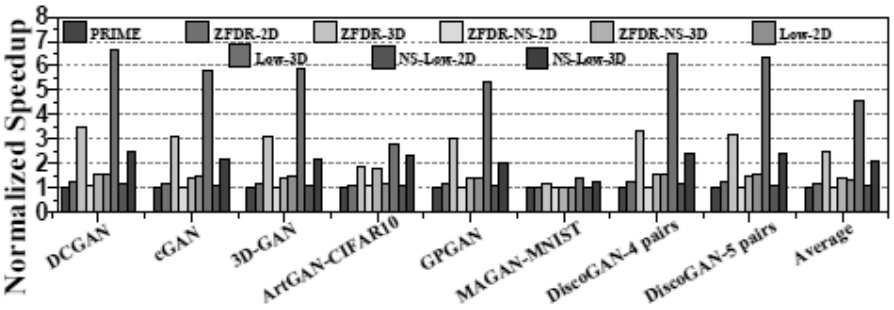

Fig. 12. Comparison of performance between 3D connection and H-tree connection with ZFDR [17]

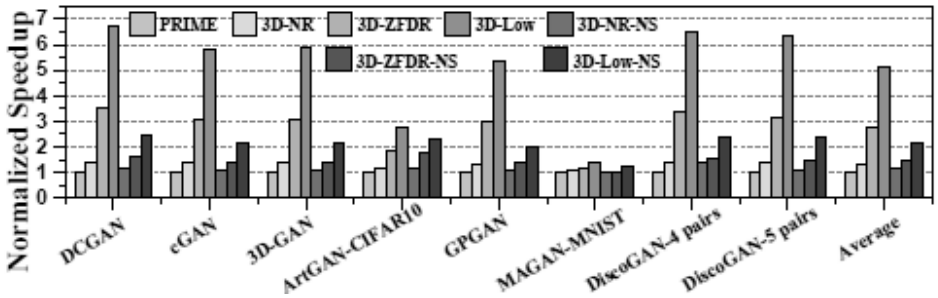

***Fig. 13**. Comparison of performance between ZFDR and Normal reshaping with 3D connectivity*

Figure 13 compares the performance between ZFDR and Normal reshaping (NR) with 3D connectivity. The results show that with 3D connectivity, ZFDR with (without) duplication reaches 5.11x (2.77x), speedup on average, While the normal reshaping shows only 1.31x the speedup, Which indicates that the design of the 3D and ZFDR connection is important for speeding up GAN implementation. LerGAN is compared to an FPGA-based GAN accelerator and GPU platform. Figures 14 and 15 show the performance and energy consumption of the above architectures, respectively. In terms of performance, the LerGAN is on average 47.2x speedup and 21.42x speedup than the FPGA-GAN and GPU, respectively.

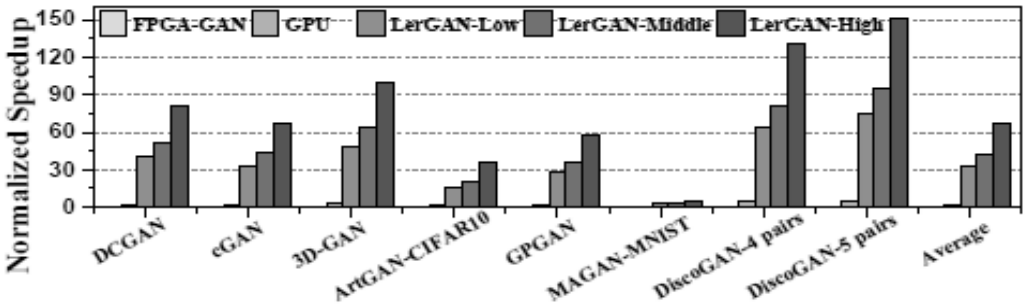

Fig. 14. Comparison of performance between FPGA-based GAN accelerator, GPU platform and LerGAN [17]

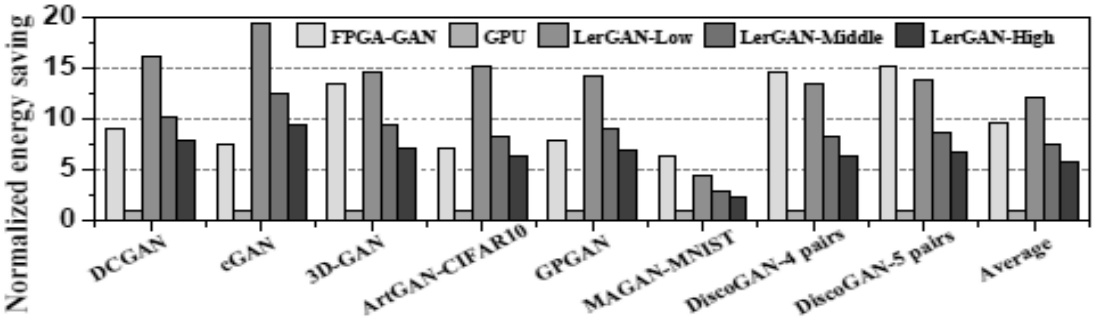

Fig. 15. Energy saving comparison between FPGA-based GAN accelerator, GPU platform and LerGAN [17]

In addition, DiscoGAN offers more speedup than others. This is the case for two reasons. (1) DiscoGAN has more T-CONVs, meaning more zeros. As a result, the LerGAN with ZFDR design shows a higher performance. (2) The size of DiscoGAN is larger and leading to more off-chip memory access for FPGA and GPUs, Which results in better performance of PIM-based LerGAN. So it can be said that small-sized GANs, such as MAGAN-MNIST and lacking of T-CONVs, cause slower speeds. To save energy, LerGAN-low saves more energy than FPGA-based GAN accelerator for small-sized GANs but with conventional T-CONVs (the left five GANs in Figure 15). However, for small-sized GANs and less T-CONVs (MAGAN-MNIST) and large-size GANs (DiscoGANs), LerGAN shows less energy saving than the FPGA-GAN accelerator. The reason is that LerGAN consumes more energy when updating networks, for this reason, saving on energy cannot eliminate the extra cost of energy. On average, LerGAN consumes 1.04x energy consumption than the FPGA-GAN accelerator. This is an advantage for FPGA-GAN. In addition, as shown in Figures 14 and 15, although more duplication (e.g. LerGAN-high) results in more speedup, it also results in more energy consumption [17]. In addition, LerGAN achieves 9.75x, 7.68x energy saving on average over GPU platform, PRIME, and has 1.04x more energy consumption over the FPGA-based GAN accelerator. With this description, in terms of energy saving, FPGA still works better than the LerGAN method.

D. The Speed of Execution in FPGA-based System

One of the most practical requirements of any implementation plan is the required speed in training and testing. In this case, it is necessary to carefully examine each of presented proposals. Lastly, it should be considered whether this accelerator speed is actually effective.

[13] Due to the start of this type of implementation on the FPGA accelerator and attention to good performance and acceptable output, no attention has been paid to the speed of learning and testing. Obviously, at the beginning of this type of project, the principles of generating and implementing and receiving appropriate outputs from the system should be addressed.

In the paper [14], the process of increasing the speed and performance of the system is slightly better in the training and learning process. Of course, this 4.3x speedup over the previous accelerators can be promising, but not enough. Overall, this has not been a major success in terms of speed on the FPGA. However, the increase of system speed in the CPU and NVIDIA GPUs has been significant.

There has also been no attempt to improve the speed of the training and testing process in the article [15]. This can be a negative factor in this article. Consideration to the speed in the accelerators is one of the important motivations for using FPGAs because of its listed capabilities.

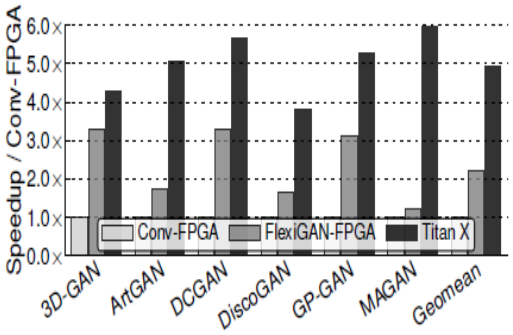

Fig. 16. Speed with FlexiGAN-FPGA and GPU (Titan X) vs. Conv-FPGA [16]

In paper [16], Fig. 16 shows the accelerator speed in this paper (FlexiGAN-FPGA) and the GPU Titan X against the normalized Conv-FPGA. On average, the FlexiGAN-FPGA delivers 2.2x speedup over Conv-FPGA. The main source of speed is FlexiGAN's ability to efficiently bypass a large percentage of zero values in Tran-Conv layers. Compared to the Titan X, the number of compute nodes and clock frequency in FlexiGAN are 2.0x and 5.0x lower, respectively. However, due to the modern design pattern architecture for inefficient operation bypass, FlexiGAN, on average, is only about 2.2x slowdown [16]. The paper [16] shows good performance in terms of speed.

In the paper [17], Experiments show that LerGAN attains 47.2x, 21.42x and 7.46x speedup over FPGA-based GAN accelerator, GPU and PRIME platforms, respectively. The influence of this paper on optimizing energy utilization and especially speed is appreciable. In this way, we were able to get closer to the accelerator desired speed. However, it should be noted that optimum speed is close to real time. But in any case, this process will greatly help us to reach the optimum speed.

IV. CONCLUSION

By examining these five articles from four different perspectives, significant gains can be made that Each of these research Achievements helps researchers to develop neural networks, especially GANs. What was ultimately highlighted in the paper [13] is the beginning of the route and the initial rollout to enhance the implementation of the GAN technique on the FPGA. You will not see any unique innovations in this article. Rather, what is presented is an acceptable quality implementation technique that highlights the application of FPGA. The paper [14] is an introduction to FPGA accelerator acceleration as well as a way to prevent inefficiency and optimal utilization of resources in accelerators as well as reduce chip memory utilization by Presentation a time-multiplexed design template and implementation of them. In [15], a model with binary weights and activation functions using integer-valued operations is presented. The discussed accelerator in this article is FPGA-based (Xilinx Zynq). It is important to note that, since binary mode and using integer-valued operation reduces utilizable memory capacity and the number of circuit gates, it is very efficient for implementation on FPGA. On the other hand, the quality of generated data

from the model is reduced by these simplification techniques. This in turn lowers the quality of the work itself. After these three articles, the next two articles are seriously proposing plans to speed up and optimize utilization of resources and energy saving. In this article, [16] also employs two models of MIMD and SIMD implementation for optimal utilization of resource. [17] Proposes a zero-free Data Reshaping scheme called LerGAN for ReRAM-based PIM, which eliminates zero-related computation. This technique results in a 2.47x speedup over the FPGA-based GAN accelerator. But instead it has 1.04x more energy utilization over the FPGA-based GAN accelerator. In the last two articles there is no serious discussion of the qualitative growth in neural network data. Speedup of learning and testing in accelerators, especially on FPGA- based accelerators, Because of the properties of parallelization, is of paramount importance. The speedup of the convolutional accelerator in the future and approaching the real time will be one of the most important factors in the researches direction. To achieve this ideal speed and quality, researchers can take advantage of FPGA-based accelerators, taking into account GAN properties. GAN properties must be considered from two perspectives:

1- Promoting computational theory and applying new techniques to achieve faster convergence in GAN systems.
2- Finding hardware solutions and Implementation of them on desirable accelerators including FPGAs and utilizing the available space within these accelerators such as parallelization opportunities, pipelining and etc.

V.ACKNOWLEDGMENT

Special thanks to Dr. Atefeh Khodaei for her contribution in classifying the contents of this article.

REFERENCES

[1]. A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in Acoustics, speech and signal processing (icassp), 2013 IEEE international conference on. IEEE, 2013, pp. 6645–6649.

[2]. G. Hinton, L. Deng, D. Yu et al., “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” IEEE Signal processing magazine, vol. 29,no. 6, pp. 82–97, 2012.

[3]. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, 2012, pp. 1097–1105.

[4]. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” nature, vol. 521, no. 7553, p. 436, 2015.C. M. Wang, J. N. Reddy and K. H. Lee, New set of buckling parameters, in *Shear Deformable Beams*, ed. T. Rex (Elsevier, Oxford, 2000), pp. 201–213.

[5]. J. Ngiam, A. Khosla, M. Kim et al, “Multimodal deep learning,” in Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 689–696.

[6]. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv: 1409.1556, 2014.

[7]. I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in International conference on machine learning, 2013, pp. 1139–1147..

[8]. I. Goodfellow et al., “Generative Adversarial Nets,” in NIPS, 2014.

[9]. J. Ho and S. Ermon, “Generative Adversarial Imitation Learning,” in NIPS, 2016.

[10]. A. Ghosh et al., “SAD-GAN: Synthetic Autonomous Driving using Generative Adversarial Networks,” in arXiv, 2016.

[11]. H.-W. Dong, W.-Y. Hsiao, L.-C. Yang et al , “MuseGAN: Symbolic-domain Music Generation and Accompaniment with Multi-track Sequential Generative Adversarial Networks,” in arXiv, 2017.

[12]. D. Nie et al., “Medical Image Synthesis with Context-aware Generative Adversarial Networks,” in International Conference on Medical Image Computing and Computer-Assisted Intervention, 2017.

[13]. Xinyu Zhang, Srinjoy Das,Ojash Neopane et al., “A Design Methodology for Efficient Implementation of Deconvolutional Neural Networks on an FPGA” in arXiv:1705.02583v1 [cs.LG] 7 May 2017

[14]. Ming Cong Song,Jiaqi Zhang., “Towards Efficient Microarchitectural Design for Accelerating Unsupervised GAN based Deep Learning”in IEEE International Symposium on High Performance Computer Architecture, 2018

[15]. Hideo Terada, and Hayaru Shouno., “B-DCGAN: Evaluation of Binarized DCGAN for FPGA” in arXiv: 1803.10930v1 [cs.CV] 29 Mar 2018

[16]. Amir Yazdanbakhsh, Michael Brzozowski, Behnam Khaleghi, et al “FlexiGAN: An End-to-End Solution for FPGA Acceleration of Generative Adversarial Networks” in Appears in the Proceedings of the 26th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2018

[17]. Haiyu Mao, Mingcong Song, Tao Li “LerGAN: A Zero-free, Low Data Movement and PIM-based GAN Architecture” in 51st Annual IEEE/ACM International Symposium on Microarchitecture, 2018

[18]. M. D. Zeiler, D. Krishnan, G. W. Taylor et al ,“Deconvolutional networks,” in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 2528–2535.

[19]. J. Wu, C. Leng, Y. Wang, et al, “Quantized convolutional neural networks for mobile devices,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016,pp. 4820–4828.

[20]. G. Dundar and K. Rose, “The effects of quantization on multilayer neural networks,” IEEE ransactions on Neural Networks, vol. 6, no. 6,pp. 1446–1451, 1995.

[21]. C. Zhang, P. Li, G. Sun, et al “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2015, pp. 161–170.

[22]. M. Peemen, A. A. Setio, B. Mesman, et al ,“Memory-centric accelerator design for convolutional neural networks, “in Computer Design (ICCD), 2013 IEEE 31st International Conference On. IEEE, 2013, pp. 13–19.

[23]. Clément Farabet, Berin Martini, Benoit Corda, et al. NeuFlow: A Runtime Reconfigurable Dataflow Processor for Vision. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2011.

[24]. Cyril Poulet and Yann Lecun. An FPGA-Based Stream Processor for Embedded Real-Time Vision with Convolutional Networks. In IEEE 12th International Conference on Computer Vision Workshops, 2009.

[25]. Zidong Du, Robert Fasthuber, Tianshi Chen, et al. ShiDianNao: Shifting Vision Processing Closer to the Sensor. In Proceedings of the 42Nd Annual International Symposium on Computer Architecture (ISCA), 2015.

[26]. Yufei Ma, Yu Cao, Sarma Vrudhula, et al. Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural

Networks. In ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2017.

[27]. Kamel *Abdelouahab*, Maxime Pelcat, Fran_cois Berry, et al. Accelerating CNN inference on FPGAs: A Survey. Research report, Universit_e Clermont Auvergne; Institut Pascal, Clermont Ferrand; IETR/INSA Rennes, January 2018.

[28]. Y. Cheng, D. Wang, P. Zhou, et al. A Survey of Model Compression and Acceleration for Deep Neural Networks. ArXiv e-prints, October 2017.

[29]. M. Courbariaux, I. Hubara, D. Soudry, et al. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. ArXiv e-prints, February 2016.

[30]. Y. Umuroglu, N. J. Fraser, G. Gambardella, et al. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. ArXiv e-prints, December 2016.

[31]. A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep convolutional Generative Adversarial Networks. ArXiv e-prints, November 2015.

[32]. M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. ArXiv e-prints, November 2014.

[33]. Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, et al : A Python framework for fast computation of mathematical expressions. ArXiv e-prints, abs/1605.02688, May 2016.

[34]. Sander Dieleman, Jan Schlter, Colin Ra_el, et al. Lasagne: First release. August 2015 URL http://dx.doi.org/ 10.5281/zenodo.27878.

[35]. Yann Lecun, Lon Bottou, Yoshua Bengio, et al. Gradient-based learning applied to Document recognition. In Proceedings of the IEEE, pages 2278{2324, 1998.

[36]. X. Zhang et al., "A Design Methodology for Efficient Implementation of Deconvolutional Neural Networks on an FPGA," in arXiv, 2017.

[37]. Y.-H. Chen et al., "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in ISCA, 2016.

[38]. H. Sharma et al., "From High-Level Deep Neural Models to FPGAs," in MICRO, 2016.

[39]. W. Lu et al., "Flex Flow: A Flexible Dataflow Accelerator Architecture for Convolutional Neural Networks," in HPCA, 2017.

[40]. A. Parashar et al., "SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks," in ISCA, 2017.

[41]. T. Moreau et al., "SNNAP: Approximate Computing on Programmable SoCs via Neural Acceleration," in HPCA, 2015.

[42]. L. Song et al., "Pipe Layer: A Pipelined ReRAM-based Accelerator for Deep Learning," in HPCA, 2017.

[59]. S. Li, C. Xu, Q. Zou, et al, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*. IEEE, 2016, pp. 1–6.

[60]. A. Shafiee, A. Nag, N. Muralimanohar, et al , "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[61]. L. Song, X. Qian, H. Li, et al, "Pipelayer: A pipelined rerambased accelerator for deep learning," in *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 541–552.

[62]. A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv: 1511.06434*, 2015.

[63]. D. Pathak, P. Krahenbuhl, J. Donahue, et al, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[43]. H. J. Siegel et al., "PASM: A Partition able SIMD/MIMD System for Image Processing and Pattern Recognition," IEEE TC, no. 12, pp. 934–947, 1981.

[44]. A. Nieto et al., "PRECISION: A reconfigurable SIMD/MIMD coprocessor for Computer Vision Systems-on-Chip," IEEE TC, vol. 65, no. 8, pp. 2548–2561, 2016.

[45]. H. M. Waidyasooriya et al., "FPGA Implementation of Heterogeneous Multicore Platform with SIMD/MIMD Custom Accelerators," in ISCAS, 2012.

[46]. X. Wang and S. G. Ziavras, "Performance-energy Tradeoffs for Matrix Multiplication on FPGA-based Mixed-mode Chip Multiprocessors," in ISQED, 2007.

[47]. J. E. Smith, "Decoupled Access/Execute Computer Architectures," in ACM SIGARCH Computer Architecture News, 1982.

[48]. T. Nowatzki et al., "Stream-Dataflow Acceleration," in ISCA, 2017.

[49]. K. Wang and C. Lin, "Decoupled Affine Computation for SIMT GPUs," in ISCA, 2017.

[50]. J. Wu et al., "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," in NIPS, 2016.

[51]. W. R. Tan et al., "ArtGAN: Artwork Synthesis with Conditional Categorial GANs," in arXiv, 2017.

[52]. A. Radford et al., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in arXiv, 2015.

[53]. T. Kim et al., "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks," in arXiv, 2017.

[54]. H. Wu et al., "GP-GAN: Towards Realistic High-Resolution Image Blending," in arXiv, 2017.

[55]. R. Wang et al., "MAGAN: Margin Adaptation for Generative Adversarial Networks," in arXiv, 2017.

[56]. A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim, "Nda: Near-dram acceleration architecture leveraging commodity dram devices and standard memory modules," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 283–295.

[57]. M. N. Bojnordi and E. Ipek, "Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1–13.

[58]. P. Chi, S. Li, C. Xu, et al, "Prime: a novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 27– 39.

[64]. J. Wu, C. Zhang, T. Xue, et al, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.

[65]. W. R. Tan, C. S. Chan, H. Aguirre, et al, "Artgan: artwork synthesis with conditional categorial gans," *arXiv preprint arXiv: 1702.03410*, 2017.

[66]. R. Wang, A. Cully, H. J. Chang, et al "Magan: Margin adaptation for generative adversarial networks," *arXiv preprint arXiv: 1704.03817*, 2017.

[67]. H. Wu, S. Zheng, J. Zhang, et al , "Gp-gan: Towards realistic high-resolution image blending," *arXiv preprint arXiv: 1703.07195*, 2017.

[68]. T. Kim, M. Cha, H. Kim, et al , "Learning to discover crossdomain relations with generative adversarial networks," *arXiv preprint arXiv: 1703.05192*, 2017.

VI.APPENDICES(TABLES)

TABLE 2. COMPARISON TO PREVIOUS IMPLEMENTATIONS [13]

	Chip	precision	#DSP	Freq	GOPS	GOPS/DSP
[22]	VLX240T	Fixed	768	150M	17	0.022
[21]	VX485T	Float	2800	100M	61.62	0.022
[13]	7Z020	12Fixed	220	100M	2.6	0.012

Table 3. FPGA (XCVU13P) and GPU (Titan X) Specifications [16]

	FPGA (xvcu13p)		CPU (GTX Titan X)
Logic Cells(K)	747	Cores	3072
Flip-Flops(K)	3456	Frequency(GHz)	1
LUT(K)	1728	Memory(GB)	12
Total BRAM(mb)	94.5	Memory Clock(GHz)	6.6
UltraRAM(Mb)	360	Technology(nm)	28
DSP Slices	12288	Platform	CUDA V8.044

Table 4. Evaluated GANs, their release years, their usage, and the number of transposed convolution (TranConv and convolution (Conv)) layers [16]

Name	Year	Description	#Conv	#TranConv
3D-GAN	2016	Generates 3D objects	5	4
Art GAN	2017	Generates complex artworks	6	5
DCGAN	2015	Generates bedroom images	4	4
Disco GAN	2017	Discovers the cross- domain relation between pair of images	10	4
GP-GAN	2017	Generates high-resolution realistic images	5	4
MAGAN	2017	Proposes a novel training procedue for GANs	6	12

Table 5. The resource utilization of Target FPGA (XCVU13P) for DCGAN generated with FlexiGAN Workflow [16]

	LUTs(K)	BRAMs(mb)	UltaRAMs(mb)	DSP Slices
Used	1325	88	280	1560
Available	1728	94.5	360	12,288
Utilization	77%	93%	78%	13%

Table 6. Topology of GAN values (f: fully-connected c: convolution t: transposed convolution k: kernel s: stride) [17]

Name	Generator	Item Size	Discriminator
DCGAN[62]	100f-(1024t-512t-256t-128t)(5k2s)-t3	64×64	(3c-128c-256c-512c-1024c)(5k2s)-f1
cGAN[63]	100f-(256t-128t-64t)(4k2s)-t3	64×64	(3c-64c-128c-256c)(4k2s)-f1
3D-GAN[64]	100f-(512t -256t-128t)(4k2s)-t3	64×64×64	(1c-64c-128c-256c-512c)(4k2s)-f1
ArtGAN-CIFAR-10 [65]	100f-1024t4k1s-512t4k2s-256t4k2s-128t4k2s-128t3k1s-t3	32×32	3c4k2s-128c3k1s-(128c-256c-512c-1024c)(4k2s)-f11
GPGAN[67]	100f-(512t -256t-128t-64t)(4k2s)-t3	64×64	(3c-64c-128c-256c-512c)(4k2s)-f1
MAGAN-MINIST[66]	50f-128t7k1s-64t4k2s-t1	28×28	784f-256f-256f-784f-f11
DiscoGAN-4pairs[68]	(3c-128c-256c-512t-256t-128t- 64t)(4k2s)-t3	64×64	(3c-64c-128c-256c-512c)(4k2s)-f1
DiscoGAN-5pairs[68]	(3c-128c-256c-512c))(4k2s)- 100f-(512t -256t-128t-64t)(4k2s)-t3	64×64	(3c-64c-128c-256c-512c)(4k2s)-f1

How to cite: H.A.Bagheri, B.MakkiaAbadi, M.H.Arastoo. A Review on the Development of GAN Techniques in convolutional Neural Networks and its implementation on FPGA accelerator, Journal of Distributed Computing and Systems(JDCS), Vol 4, Issue 2, Pages 46-57, 2021.