Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/51/20241209

FPGA acceleration of convolutional neural networks: Current applications and future prospects

Zexiao Li

Electronics and Information Engineering, Suzhou University of Science and Technology, Suzhou, 215009, China

201010013428@stu.swmu.edu.cn

Abstract. In the contemporary era of big data, the utilization of deep learning technology has become widespread in the extraction of valuable analytical insights from the data under consideration. This technology finds extensive applications in various domains, including image information recognition, speech processing, and text language analysis. When it comes to accelerating convolutional neural networks (CNNs), the utilization of Field-Programmable Gate Arrays (FPGAs) offers distinct advantages over other hardware accelerators. Nevertheless, it is important to acknowledge that FPGA-based acceleration also comes with its own structural limitations. This article places its focus on two main aspects: firstly, it delves into the present-day application landscape and the latest developmental trajectories of convolutional neural networks. Secondly, it elucidates the inherent characteristics of FPGA implementations of CNNs. Furthermore, the article conducts a comprehensive examination of the pertinent constraints associated with FPGA-accelerated deep learning algorithms. The discussion extends beyond the present moment and ventures into the future, offering insights into the potential advancements in the field of deep learning. Importantly, it also brings into focus the prospective avenues for further research regarding the application of FPGAs in the domain of convolutional neural networks.

Keywords: Convolutional Neural Network, FPGA, Deep Learning.

1. Introduction

Deep learning technology has recently developed rapidly and has shown good performance in various application scenarios. Among them, convolutional neural networks stand out as the best developed branch in deep learning. By learning from a large amount of data to be processed, CNN can extract complex features from the data and achieve various functions. CNN was initially mainly used for handwritten digit recognition and has achieved good results in this field [1]. With the development of CNN models, their application scope continues to expand and can now be applied in fields such as voice recognition and gesture recognition [2, 3]. The commonly used convolutional neural network hardware platforms include ASIC, GPU, and FPGA. Compared to ASIC, FPGA can be developed more flexibly; Compared to GPUs, FPGAs have a higher energy efficiency ratio. However, due to its unique hardware structure, FPGA also faces difficulties in program development and time-consuming issues, which have become important constraints for the development and application of FPGA in deep learning. Therefore, how to solve these problems is a research hotspot in deep learning today.

DOI: 10.54254/2755-2721/51/20241209

This article summarizes and summarizes the current application status of convolutional neural networks and FPGA through extensive literature research, lists the latest application characteristics of convolutional neural networks, elaborates on the factors that restrict the further large-scale application of FPGA, and explains the main development direction of FPGA application in convolutional neural networks. The first chapter of this article is an introduction. This article briefly describes the current application status of convolutional neural networks and FPGA, introduces the advantages of FPGA in accelerating CNN and the problems caused by its own structure. Chapter 2 focuses on FPGA implementation of convolutional neural networks. Clarified the typical structure of CNN and the development process of FPGA, while also elaborating on the characteristics of FPGA as a deep learning accelerator. Chapter 3 is the application analysis of convolutional neural networks. By listing research results, it fully demonstrates the current application status of convolutional neural networks, and elaborates on the latest research trends and progress directions of convolutional neural network applications. Chapter 4 discusses the many limiting factors of FPGA application in convolutional neural networks. List the factors that currently limit the further application of FPGA in deep learning. Chapter 5 is the conclusion. This chapter summarizes the investigation and research work of this article, and provides some prospects for the future development of FPGA application in deep learning, including CNN.

2. FPGA Implementation of Convolutional Neural Networks

2.1. Structure of CNN

CNN is a deep learning algorithm that enhances the data processing capabilities of neural networks through convolutional and pooling layers. Figure 1 shows a typical CNN structure diagram.

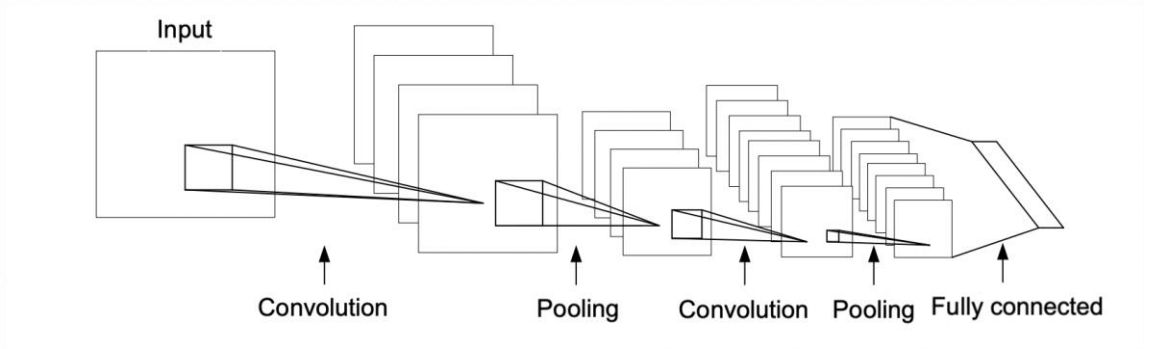

Figure 1. A typical CNN topology structure (Photo/Picture credit: Original).

The convolutional layer will filter out the characteristics of the data to be processed as input to the CNN; The pooling layer compresses the obtained feature maps; The fully connected layer connects features and obtains output results.

2.1.1. Convolutional layer

The computational complexity of CNN is mainly concentrated in the multiplication and accumulation operations of convolutional layers. Feature extraction is achieved through convolution operations between convolutional kernels and feature maps. The convolution operation in CNN is a multiplication and accumulation operation, and equation (1) defines the two-dimensional convolution operation in CNN.

$$f(x,y) = \sum_{kx=0}^{K-1} IN(x+kx, y+ky)W(kx,ky) \quad (1)$$

Among them, IN represents the input feature map of the convolutional layer, W represents the convolutional kernel with a size of K * K, and there are K * K weight data in one convolutional kernel.

DOI: 10.54254/2755-2721/51/20241209

The weight data in the convolution kernel is multiplied by the K * K data in the K * K size convolution window on the input feature map, and the sum of the results obtained is the result of a two-dimensional convolution [4].

The convolution process of a convolutional layer involves the sliding step size of the convolutional kernel and the size of the convolutional kernel. The window where the data participating in each convolution operation is located is called the convolution window. The sliding step determines how much the convolutional window moves. The convolution kernel is sequentially convolved with the convolution window according to the sliding step size to obtain the extraction result of the feature map. Figure 2 shows this process, with different coloured boxes representing different convolutional windows.

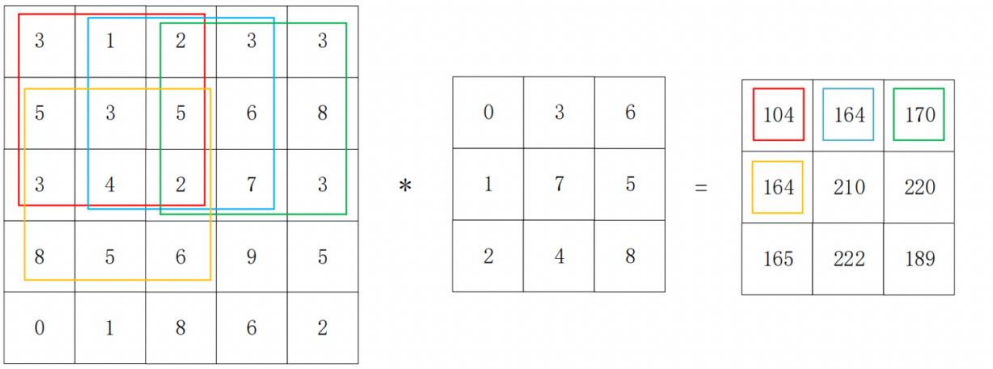

Figure 2. Sliding Window Operation and Extraction of Input Feature Maps (Photo/Picture credit: Original).

2.1.2. Pooling layer

The pooling layer samples the output of the convolutional layer through specific sampling methods, thereby compressing the scale of the feature map. The essence of it is actually sampling, pooling, and selecting a certain method to compress the input image to accelerate the computational speed of the neural network. The method mentioned here is actually a pooling algorithm, such as maximum pooling or average pooling. Figure 3 shows the schematic diagram of average pooling, with a pooling size of 2 * 2. In this pooling method, each 2 * 2 area corresponds to an output, and the average value of the numbers in the area is the output result.

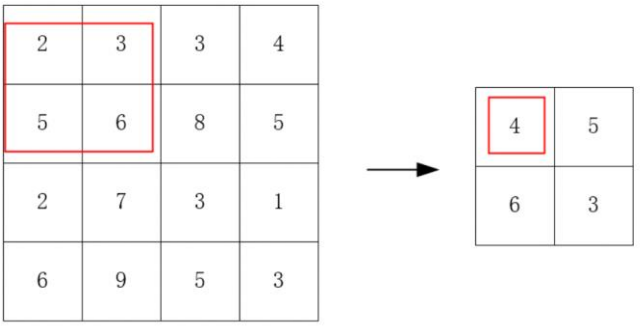

Figure 3. Average pooling (Photo/Picture credit: Original).

Figure 4 shows the schematic diagram of maximum pooling, with a pooling size of 2 * 2. The pooling result is the largest number in each 2 * 2 area.

DOI: 10.54254/2755-2721/51/20241209

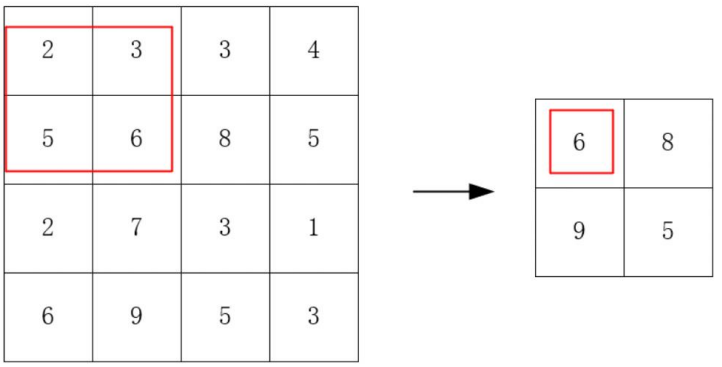

Figure 4. Maximum pooling (Photo/Picture credit: Original).

2.2. FPGA development process

The development process of FPGA includes two stages: hardware design and software design. The development process is a top-down collaborative design of software and hardware. The typical FPGA development steps are shown in Figures 5, and the main tasks of each step are as follows [4].

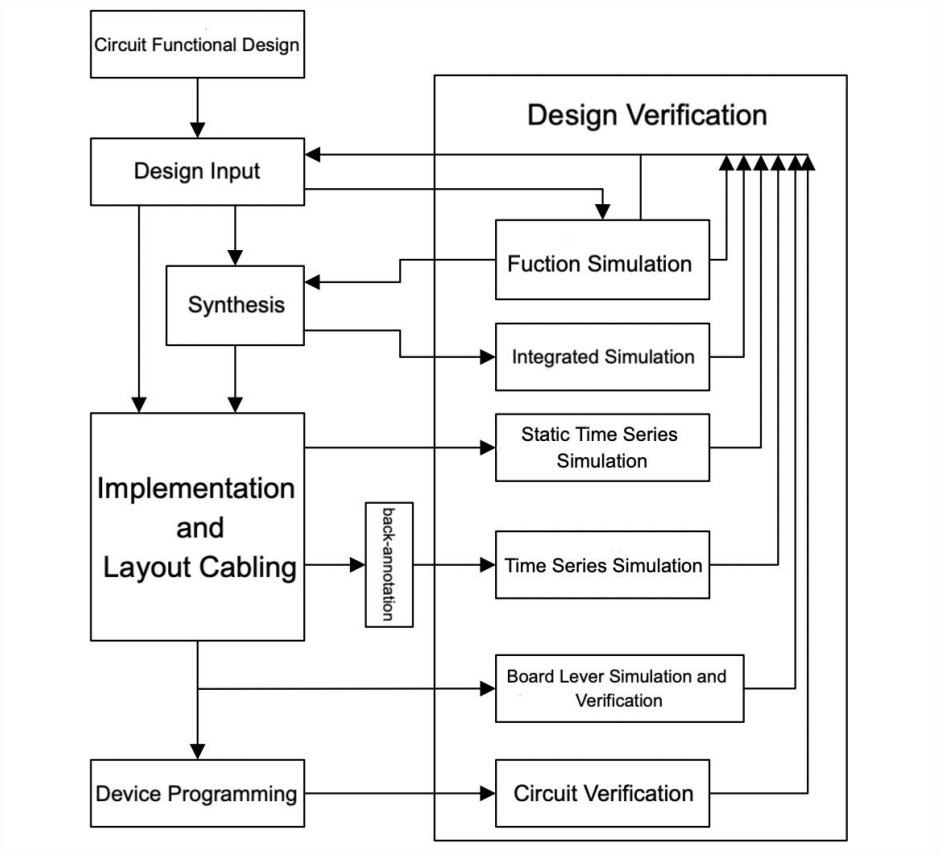

Figure 5. Typical FPGA Development Steps [4].

(1) Circuit functional design. Consider the working speed and on-chip resources of FPGA chips, and develop a design plan based on requirements and FPGA performance.

DOI: 10.54254/2755-2721/51/20241209

(2) Design input. Design input is expressed through schematic diagrams or HDL language for functional circuits. The HDL design method mainly describes the processing, control, and flow of data in logical modules and storage units.

(3) Functional simulation. Functional simulation is the logical functional verification of a designed circuit, which simply means verifying whether the code written is consistent with the target function.

(4) Logical synthesis. Comprehensive optimization is the optimization of logical connections based on expectations, that is, compiling design inputs into a standard gate level structure network table, and then generating gate circuits through FPGA wiring tools.

(5) Integrated simulation. Integrated simulation involves adding standard delay files during the simulation period to evaluate the impact of gate delay on the entire circuit.

(6) Implementation and Layout Cabling. The implementation is achieved by configuring a comprehensive generated logical network table on the FPGA through layout and wiring, and the wiring operation is to connect various components in the optimal way on the layout structure.

(7) Time series simulation and verification. Note the delay situation of layout and wiring in the design network table, and check whether the system meets the rules of establishment time, retention time, etc.

(8) Board level simulation and verification. This step is typically used in circuit design to test for signal loss and signal stability in high-speed circuits.

(9) Design and performance testing of chip functions. The design of chip functionality refers to generating a Bitstream File from the layout and wiring configuration file and burning it into the FPGA to configure the FPGA.

2.3. Characteristics of FPGA as a Deep Learning Accelerator

Unlike the fixed hardware structures of GPUs and ASICs, FPGAs have programmability, which means they can program the functions and connection relationships of their logical units through configuration tools to achieve specific functions. Therefore, the design of FPGA hardware accelerators also requires developers to have a high understanding of hardware, which is a difficult task for software developers. However, in recent research, some companies and research institutions have continuously enriched the FPGA development environment. Currently, relevant developers have become practical in using familiar programming languages such as C and C++to develop FPGA, greatly making FPGA development easier.

The computation of convolutional neural networks in the inference stage is a single data multi-instruction stream, and in practical applications, there are usually requirements such as low power consumption, high performance, and low latency. Therefore, it is more suitable for FPGA. The bottleneck of using FPGA to accelerate the performance of convolutional neural networks is mainly reflected in two aspects: computation and data transmission. Therefore, when designing FPGA acceleration, conversion model algorithms and optimizing hardware structure are usually used, by fully leveraging CNN's parallel computing capabilities, computational efficiency can be improved.

When using FPGA to accelerate convolutional neural networks, the general design approach is to conduct mathematical modeling based on the proposed design purpose, target, and the characteristics of the CNN itself, and convert it into a problem of using mathematical formulas to find the maximum or minimum value. After the mathematical modeling is completed, methods such as exhaustive and dynamic programming are used to find the relatively best solution, and then based on the parameters of the optimal solution Generate the final FPGA hardware structure based on neural network topology, etc. The design purpose is usually to improve the utilization of FPGA resources, the computational performance of accelerators, or the ability to transmit data, as well as to find relatively good neural network topology structures while optimizing resource utilization. For convolutional layers with CNN as the target, the parameters in the mathematical formula are based on specific objectives The topology structure of neural networks and optimization strategies are selected. At the same time, there are also some constraints in mathematical modeling, which are mostly converted based on the limited resource limitations of FPGA. For example, the total number of LUT and DSP resources used in the design should be smaller than the total number of LUT and DSP resources on the FPGA chip, respectively.

DOI: 10.54254/2755-2721/51/20241209

3. Application Analysis of Convolutional Neural Networks

3.1. Applications of CNN Today

With the development and progress of CNN, its performance continues to improve and its application range continues to expand. Up to now, it can be applied not only to traditional speech information recognition, image and video information recognition and processing, but also to text information extraction, language, and the transmission of semantic information that humans can access. The DBN and HMM models have achieved satisfactory results in speech information recognition. Recently, with the help of DBNs with strong discriminative ability and HMMs with sequence modeling ability, CNN has been successfully applied to speech and continuous speech recognition (LVC-SR) tasks with a large amount of language information; A deep trust network consists of many layers, which are further divided into explicit and implicit neurons. Explicit neurons are used to receive input, while implicit neurons are used to extract features. After training, these neurons can mathematically model the information extracted from speech and explore the characteristics of statistical structures. These discovered features can be applied to initialize the neural network, which can then be used to train the model to predict the state of HMM. Mohamed replaces the model in GMM-HMM with a five-layer DBN and innovatively uses the state of a single phoneme as a mathematical model. After model training, it can perform high accuracy speech recognition [5]. Nair et al. introduced a new deep network model and asked it to attempt to identify a complex three-dimensional object to evaluate its accuracy and practicality based on the recognition results. They proposed using an improved DBN of a third-order Boltzmann machine at the top level of the model and applying the model to the NORB database for three-dimensional object recognition tasks [6]. The results were shocking, and the recognition error was quite small, the results can even be close to the best results that can be obtained using the convolutional neural network-based NORB with translation invariance built-in principle, which has been recently announced.

The research on deep learning in language file processing is increasingly receiving widespread attention. The use of neural networks for mathematical modeling of human natural language and text has become relatively mature, and the development of multi-layer and deep network applications in language file processing is gradually accelerating. Reference utilizes a multi task simultaneous linear processing method based on DBN to complete computer speech recognition, input, and other problems [7]. After improvement, it can also be extended to computer language translation problems. The use of DBN and deep automatic encoders for file retrieval can display word features, which has significant advantages over widely used semantic analysis and can make literature retrieval simpler and more convenient. This method has been preliminarily applied in other fields, such as the extraction and recognition of sound and speech information in a series of fields [8].

3.2. Development Trends of Convolutional Neural Network Applications

Nowadays, the related applications of CNN are rapidly developing towards complexity and diversification. Overall, its practical application has developed rapidly, with the main changes being as follows:

(1) In recent years, the continuous deepening of research on deep learning and CNN has improved the accuracy of the results obtained in various scenarios. For example, deep learning is applied to image classification. A classic CNN model proposed by Ilya Sutskever and Geoffrey Hinton in the 2012 Image Classification Competition - AlexNet. After improving the accuracy of image classification in the ImagNet database to about 85%, new models continuously broke the record. Representative networks include VGG proposed by Oxford's Visual Geometry Group, which achieved the first performance in ILSVRC 2014 that year and successfully demonstrated that increasing the depth of the model network can improve the performance of the network model to a certain extent [9]. In 2014, Christian Szegedy proposed a new deep learning structure, GoogLeNet, which can more efficiently utilize the computing resources it has, so it can extract more features under the same computational load [10]. A new activation method - PReLU net, which solves the problem of gradient explosion and gradient disappearance [11]. Google researchers have made improved versions based on Inception, such as BN concept [12].

DOI: 10.54254/2755-2721/51/20241209

Convolutional neural networks have greatly improved the accuracy of existing datasets today, bringing urgent requirements for designing larger databases in the future.

(2) The development of real-time applications. Through literature research, it has been found that some studies have demonstrated the great potential of CNN in real-time applications. Giship and Ren et al. have made significant contributions to the development of CNN in real-time applications [13-15]. They have successively completed R-CNN, Fast R-CNN, and Fast R-CNN models in the field of 3D entity recognition based on CNN, enabling CNN to be officially applied in real-time applications. The R-CNN model utilizes convolutional neural networks to detect three-dimensional solid substances based on region proposals [16]. Fast R-CNN achieves very fast detection speed by sharing convolutional features between regions to reduce computational overhead. Faster R-CNN creatively utilizes end-to-end CNN to recognize and extract region proposals, achieving real-time object detection [17]. CNN has shown good application prospects in multiple fields that it was not good at in the past.

(3) As a tool for feature extraction and pattern recognition, CNN has exceeded the traditional field of image recognition in its application. For example, the famous Alpha Go successfully utilized CNN to assess the situation of Go [18]. Kalchbrenner used CNN to extract vocabulary and sentence level information from text information, enabling CNN to be applied to natural language processing today [19]. It can be said that CNN has gradually demonstrated its application prospects in a wider range of fields.

4. The main factors affecting the application of FPGA in convolutional neural networks

4.1. Imperfect ecological environment

At present, the FPGA ecosystem is relatively closed, with the main industry chain concentrated in a few companies and no industry standards. This results in high learning and development barriers, long investment cycles, limited development tool usage, and expensive chip prices for FPGA. CNN has been widely used, so how to efficiently utilize FPGA to accelerate convolutional neural networks and make their applications more accurate and convenient is a challenge. Due to the limitations of the FPGA ecosystem, its application speed is difficult to keep up with the development speed of software algorithms, which is the biggest challenge limiting the application of FPGA in convolutional neural networks.

4.2. High programming language threshold

With the continuous development of FPGA, its complexity is also increasing. How to program hardware resources conveniently has become an important factor affecting the application of FPGA in convolutional neural network models. Hardware programming languages have increasingly high requirements for programmers in FPGA, which greatly affects the design efficiency of FPGA and limits its application in convolutional neural networks. In this situation, the High Level Synthesis of FPGA has emerged. It can directly convert C++, C and other languages into code that can run on FPGA through compilation tools. In the process of exploring HLS, Yi-Hsiang Lai et al. explored FPGA programming based on Python, and the proposed HeteroCL reconfigurable computing programming framework is very friendly to algorithm designers [20]. In addition, Stephen Ibanez et al. used high-level language "P4" to construct network algorithms and applications in the field of network packets [21]. Although the application scope of FPGA continues to expand, its programming is still mainly based on RTL development. Therefore, mapping the logical structure described in high-level languages to HLS in hardware languages will undoubtedly become a powerful tool for assisting FPGA development, and will have broad development prospects.

4.3. Complexity of data and models

With intelligence becoming a demand in various industries, the data processed by convolutional neural networks continues to increase, and models continue to become more complex. This makes convolutional neural networks based on FPGA have better hardware performance. FPGA performance

DOI: 10.54254/2755-2721/51/20241209

is often limited by memory system throughput, meaning that applications are prone to crashes when limited by memory bandwidth [22, 23]. In order for FPGA to carry big data and complex network computing, we must increase FPGA bandwidth or improve bandwidth utilization. Therefore, how to expand FPGA bandwidth or improve bandwidth utilization at low cost has become a major development trend of FPGA. In addition, artificial intelligence products often require rapid response ability, which requires FPGA to have fast computing speed. At present, there are also many hardware acceleration methods and accelerated network frameworks in the research on FPGA computing acceleration. However, with the continuous development of artificial intelligence, the demand for speed will become increasingly high. Therefore, how to accelerate the computational speed of neural networks on FPGA will still become a research hotspot in FPGA.

5. Conclusion

The article analyzes and summarizes the current application status of convolutional neural networks through extensive research literature: their main application areas include the extraction and processing of speech, images, videos, as well as text and language information, and provides numerous examples of recent research in these areas. Afterwards, the article presented three latest development trends of convolutional neural networks: improved accuracy, rapid development in real-time applications, and expansion of their application scope.

The article also lists and analyzes several significant issues that affect the application of FPGA in CNN: imperfect ecological environment, high programming language threshold, and complexity of data and models. After analysis, the application of FPGA in convolutional neural networks has broad prospects, and it should also be noted that there are still many urgent problems to be solved.

In recent years, CNN has been widely used in many fields that are closely related to human development and life. However, due to its inherent need for high computational complexity chips and the requirements for chip power consumption and speed in applications, the use of hardware accelerators to accelerate deep learning is an inevitable trend. Currently, the design of FPGA hardware accelerators for deep learning is in the ascendant, and research work in the future is expected, Mainly focused on the following aspects: (1) A more comprehensive development environment for FPGA. The information library, programming language, and development language environment for FPGA accelerated convolutional neural networks should be more comprehensive. At the same time, various comprehensive development tools should provide software developers with a more convenient and easy to learn development experience, allowing them to focus on algorithm exploration instead of learning the corresponding programming language. (2) A more optimized FPGA communication mechanism. Currently, the bandwidth issue is one of the urgent issues that FPGA accelerated convolutional neural networks need to solve. In the future, in addition to using more direct methods such as reducing the model's demand for large bandwidth to solve the bandwidth problem, researchers can also use methods such as reasonably utilizing computational time to mask communication delays to solve this problem, Implementing a more optimized FPGA communication mechanism. (3) Improving FPGA cloud services. The combination of FPGA and cloud computing brings new opportunities for FPGA to accelerate the development of deep learning. At present, FPGA cloud services are in their early stages, and there are still many imperfections and valuable research topics, such as how to virtualize FPGA hardware resources and how to achieve an efficient and energy-efficient multi machine multi FPGA acceleration architecture. (4) Further optimization of deep learning algorithms using FPGA accelerated compression. Currently, many people in the academic community are studying compressed convolutional neural network models. The compression algorithm can effectively reduce the overhead required by the model to a large extent, resulting in an overall performance improvement. In the future, using FPGA to accelerate compressed deep learning algorithms will also be a research hotspot.

References

[1] Hongtao Lu, Qinchuan Zhang. A Review of the Application Research of Deep Convolutional Neural Networks in Computer Vision. data acquisition and processing, 2016, 31(01): 1-17.

DOI: 10.54254/2755-2721/51/20241209

[2] Chuxiong Qin, Lianhai Zhang. A Convolutional Neural Network Acoustic Modeling Method for Integrating Multi stream Features in Low Resource Speech Recognition. computer application, 2016, 36(09): 2609-2615.
[3] Juan Cai, Jianyong Cai, Xiaodong Liao, et al. Preliminary Study on Gesture Recognition Based on Convolutional Neural Networks. Computer System Applications, 2015, 24(04): 113-117.
[4] Dahl G E, Yu D, Deng L, et al. Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2011: 4688-4691.
[5] Mohamed A, Sainath T N, Dahl G, et al. Deep belief networks using discriminative features for phone recognition. 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2011: 5060-5063.
[6] Nair V, Hinton G E. 3D object recognition with deep belief nets. Advances in neural information processing systems, 2009, 22.
[7] Deselaers T, Hasan S, Bender O, et al. A deep learning approach to machine transliteration. Proceedings of the Fourth Workshop on Statistical Machine Translation. 2009: 233-241.
[8] Deng L, Seltzer M L, Yu D, et al. Binary coding of speech spectrograms using a deep auto-encoder. Eleventh annual conference of the international speech communication association. 2010.
[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
[10] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
[11] He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE international conference on computer vision. 2015: 1026-1034.
[12] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. International conference on machine learning. pmlr, 2015: 448-456.
[13] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.
[14] Girshick R. Fast r-cnn. Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
[15] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015, 28.
[16] Uijlings J R R, Van De Sande K E A, Gevers T, et al. Selective search for object recognition. International journal of computer vision, 2013, 104: 154-171.
[17] Hou X, Zhang L. Saliency detection: A spectral residual approach. 2007 IEEE Conference on computer vision and pattern recognition. Ieee, 2007: 1-8.
[18] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. nature, 2016, 529(7587): 484-489.
[19] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188, 2014.
[20] Lai Y H, Chi Y, Hu Y, et al. HeteroCL: A multi-paradigm programming infrastructure for software-defined reconfigurable computing. Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2019: 242-251.
[21] Ibanez S, Brebner G, McKeown N, et al. The p4-> netfpga workflow for line-rate packet processing. Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2019: 1-9.
[22] Asiatici M, Ienne P. Dynaburst: Dynamically assemblying dram bursts over a multitude of random accesses. 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2019: 254-262.

DOI: 10.54254/2755-2721/51/20241209

[23] Peltenburg J, Van Straten J, Wijtemans L, et al. Fletcher: A framework to efficiently integrate FPGA accelerators with apache arrow. 2019 29th International Conference on Field Programmable Logic and Applications (FPL). IEEE, 2019: 270-277.