FPGA-Based Acceleration of Deep Learning Networks: Techniques and Applications

Han Wang
School of Computer Science
WuHan Donghu University
WuHan, Hubei, China
2180105279@qq.com

Xiaoyue Liu
School of Computer Science
WuHan Donghu University
WuHan, Hubei, China
3250199433@qq.com

Yi Yang
School of Computer Science
WuHan Donghu University
WuHan, Hubei, China
1240452292@qq.com

Qin Liao
School of Computer Science
WuHan Donghu University
WuHan, Hubei, China
1526884703@qq.com

Shiyu Ke
School of Computer Science
WuHan Donghu University
WuHan, Hubei, China
548721736@qq.com

Yong Dong*†
WuHan Donghu University
WuHan, Hubei, China
salonmoner@qq.com

Abstract

Deep learning models have made significant progress in fields such as image recognition and natural language processing. However, the increasing complexity and size of these models pose challenges for deployment on resource - constrained devices. This paper reviews the development of FPGA acceleration for deep learning models, with a focus on Convolutional Neural Networks (CNNs) and Transformers. The discussion of FPGA acceleration for deep neural networks is divided into software optimization and hardware optimization. On the software side, it includes network pruning, parameter quantization, and network architecture design. The paper also highlights the latest research on FPGA - based hardware accelerators for these models. Furthermore, it discusses future trends in the development of advanced hardware architectures and the deepening of software - hardware co - optimization to further improve the performance and energy efficiency of deep learning models.

CCS Concepts

• **Hardware** → Very large scale integration design; On-chip resource management.

Keywords

Transformer, FPGA, CNN, Acclerator

ACM Reference Format:
Han Wang, Xiaoyue Liu, Yi Yang, Qin Liao, Shiyu Ke, and Yong Dong. 2025. FPGA-Based Acceleration of Deep Learning Networks: Techniques and Applications. In *2025 6th International Conference on Computer Information and Big Data Applications (CIBDA 2025), March 14–16, 2025, Wuhan, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746709.3746843

*Corresponding author
†School of Computer Science, WuHan Donghu University, Wuhan, 430212, China, 1526884703@qq.com

CIBDA 2025, Wuhan, China

ACM ISBN 979-8-4007-1316-3/2025/03
https://doi.org/10.1145/3746709.3746843

1 Introduction

In recent years, deep learning has been widely applied in both research and industrial fields. Meanwhile, CNN (Convolutional Neural Network), as the most representative network in the field of deep learning, has a high recognition accuracy and has been widely used in directions such as face recognition, image classification, and object tracking [1] [2] [3]. However, tasks based on CNN models usually require high computational power. Taking the AlexNet [4] network in 2012 as an example, the number of convolutional layers in this network structure is five, the number of fully connected layers is three, and the network's parameter volume has reached sixty million. It requires 1.4 GOPS (Giga Operations per Second) to process a single image in the ImageNet dataset. In 2017, Google first proposed the Transformer model based on the attention mechanism [5]. Nowadays, the Transformer model has performed outstandingly in the fields of computer vision (CV) and natural language processing (NLP), gradually surpassing convolutional models. However, the huge amount of computing tasks makes it more difficult to deploy network models in embedded devices.

In addition, the computing tasks in neural networks are more suitable for parallel computing, while general-purpose processors (Central Processing Unit) suitable for handling serial tasks are usually less efficient when dealing with such highly parallel tasks. The most widely used hardware acceleration platform at present is the graphics processing unit (Graphics Processing Unit, GPU). As a general-purpose computing platform, the GPU has a good performance in processing speed, but due to its high power consumption, it is usually not used in the inference stage. On the other hand, neural networks can be accelerated through application-specific integrated circuits (Application Specific Integrated Circuit, ASIC) and field-programmable gate arrays. Among them, ASIC can achieve performance similar to that of a GPU during inference. However, the long design cycle of ASIC makes its flexibility poor. FPGAs (Field-Programmable Gate Arrays) can build customized hardware circuits according to specific convolutional neural network models,

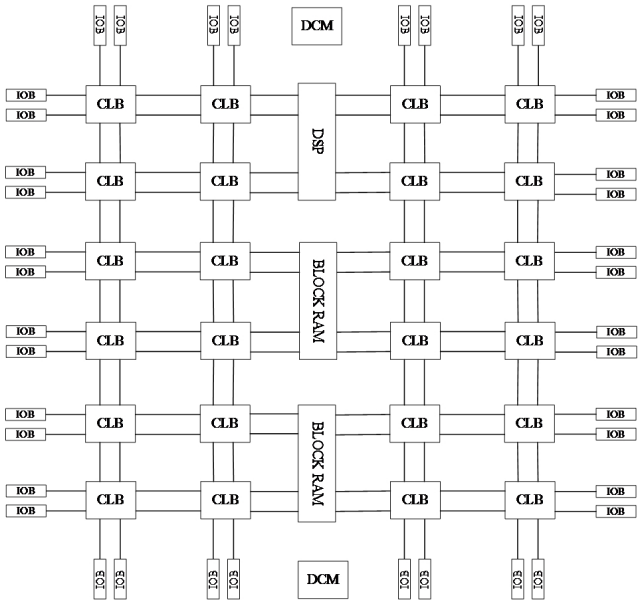

Figure 1: The basic structure of FPGA

and provide the computing power needed by the network through this method.

2 Background

2.1 FPGA

The difference between FPGAs and general-purpose processors is that FPGAs are instruction-free. CPUs and GPUs are based on the von Neumann architecture, and in the von Neumann architecture, the execution unit requires the participation of instruction memory, decoder, arithmetic unit, and branch jump handling logic when processing instructions. FPGAs do not need to use instructions when processing computing tasks because the functions of each logic block in FPGAs are determined when the bitstream is burned. CPUs and GPUs require access arbitration due to the characteristic of shared memory, and each execution unit has its own private cache, which needs to ensure cache consistency. The registers and BRAM on FPGAs have already determined the logic and do not need arbitration. In addition, the advantage of FPGAs lies in their real-time performance. For example, in Microsoft's Bing search engine, users need to get the search results as soon as possible, so it is necessary to ensure that the delay of each link is low. The main reason why FPGAs can have lower latency than GPUs is that FPGAs have a deeper pipeline and can ensure data parallelism. GPUs can only ensure data parallelism, and their pipeline depth is limited by bandwidth. For typical single-instruction multiple-data stream tasks like neural networks, calculations can be performed faster on FPGAs.

The typical FPGA structure mainly consists of three parts: the first part is the logic block (LB) used for logic circuit functions, the second part is the module responsible for signal input and output between the FPGA and the external environment (IOB), and the third part is the programmable wiring module (CB) used to connect the LB and IOB modules. In commercial scenarios, to improve the computing performance of FPGAs, other modules are usually added to the FPGA, such as DSP for calculation, BRAM for data storage, and PLL and DLL for clock generation. After adjacent modules are connected, they can form a CLB, and the CLB is finally arranged in an array form to form an island-type structure. The figure shows the basic structure of the FPGA. Figure 1. shows the basic architecture of FPGA.

2.2 Deep Nerual Networks

2.2.1 CNN.. The LeNet-5 model [6], one of the earliest proposed convolutional neural networks, was used for MNIST handwritten digit recognition and achieved good results. Figure 2. shows the structure of a typical convolutional neural network, which from left to right includes convolutional layers, pooling layers, activation layers, normalization layers, and fully connected layers. First, multiple convolutional kernels are used to extract features from the input feature maps of the previous layer. The size of the feature maps after feature extraction is usually large, so a downsampling layer is added after convolution to prevent the curse of dimensionality.

As one of the most typical network models in CNNs, AlexNet [4] was proposed by Alex et al. in 2012. For the first time, the ReLU activation function was used in this network, replacing the previous sigmoid and tanh activation functions, which accelerated the training process. The model consists of five convolutional layers and three fully connected layers. The VGG [7] network model was proposed by Visual et al. in 2014. Its characteristic is the use of multiple consecutive small convolutional kernels (3x3) instead of a large convolutional kernel, which increased the depth of the network and improved the network's ability to extract image features. ResNet [8] is a convolutional neural network model proposed by He et al. in 2015. By introducing residual connections, the features of the previous layer are directly passed to the subsequent layers, solving the problem of gradient disappearance and allowing the depth of the network model to be greater.

2.2.2 Transformer. The Transformer model has brought about a significant transformation in the field of natural language processing (NLP) thanks to its distinctive self-attention mechanism. Unlike traditional recurrent neural networks (RNNs) that process sequences step by step, the Transformer architecture handles all input elements in parallel, making it highly efficient for modern computing platforms. The self-attention mechanism enables the model to assess the importance of different parts of the input sequence, allowing for more effective modeling of long-range dependencies. This is especially useful for tasks like machine translation, where understanding the context of words in a sentence is crucial for accurate translation. Figure 3. shows the architecure of Transformer.

In recent years, models based on the Transformer architecture have gained significant traction in various fields. For instance, the BERT model [9], built upon the Transformer, leverages vast amounts of text data for pre-training, enabling it to learn rich language representations. BERT is composed of multiple layers of Transformer encoders, including an embedding layer (comprising token, position, and segment embeddings), Transformer encoder layers (featuring multi-head self-attention, feed-forward neural networks, residual connections, and LayerNorm), and a pooling

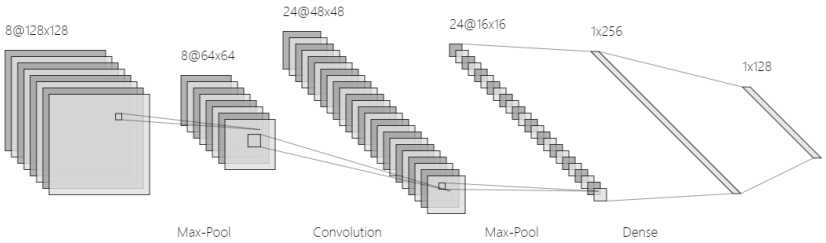

Figure 2: The architecure of Lenet-5 [6]

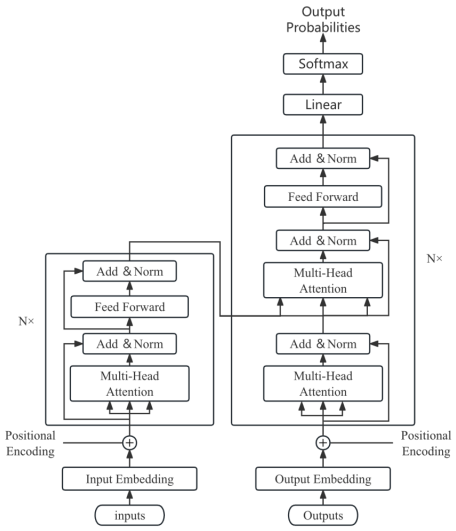

Figure 3: The architecure of Transfomer [5]

layer. This allows it to capture bidirectional context information in the text, generating high-quality text representations. In the field of image processing, the Vision Transformer (ViT) [10] has also shown impressive performance. It divides images into fixed-size patches, flattens these patches, and embeds them into a sequence as input for the Transformer. ViT consists of multiple layers of Transformer encoders, including an image patch embedding layer, Transformer encoder layers, and a classification head. In the image patch embedding layer, the patches are flattened and position embeddings are added before being fed into the Transformer encoder for feature extraction. Additionally, there are models in the GPT series [11] [12] [13].

3 FPGA acceleration method

3.1 Software Optimization

The increase in network model accuracy is typically achieved by increasing the number of layers in the network. This approach leads to an increase in the number of network parameters. The vast number of parameters makes it difficult for the network to run effectively on mobile devices with limited storage and computing resources.

Network pruning [14]is an effective method for reducing model size, inspired by biological principles where the number of neurons in a mammal's brain decreases after reaching a peak Network pruning can be divided into parameter pruning and neuron pruning. Parameter pruning involves setting weights in the network to zero.

Parameter quantization primarily optimizes the storage of weights and includes two common approaches: reducing weight precision and weight clustering. Reducing precision involves decreasing the bit width of weights, such as from 64 bits to 32 bits or 16 bits. For example, Wang et al. [15] demonstrated that quantizing 32-bit floating-point weights to 8 bits can reduce the overall model size and the time required for computation.

Network architecture design involves modifying parts of the network structure without reducing model accuracy to achieve model compression.

3.2 Hardware Optimization

Han et al. [16] designed the ESE accelerator for LSTM networks, which can avoid idle processing units when pruning LSTM networks. They also added a zero - skip gating structure to further improve efficiency. Mukkara et al. [17] proposed an FPGA - based SCNN hardware architecture that can efficiently handle pruned convolutional neural networks during computation. This is mainly achieved by skipping weights with a size of 0 generated by pruning algorithms and nonlinear activation functions. Compared with accelerators that do not use this structure, the performance is improved by 2.7 times. Struhanik et al. [18] proposed a coarse - grained reconfigurable architecture based on the FPGA platform, called CoNNA, in 2018. By dynamically configuring the network, it can perform convolution operations efficiently during computation. Compared with Eyeriss under the same computing resources and working frequency, it achieved a 14.1 - times improvement. Kung et al. [19] proposed a new network encapsulation method in 2019, which allows sparse convolutional neural networks to be efficiently implemented on hardware using a systolic array. Under the same working frequency and dataset, the overall performance is three times better than that of the TrueNorth [20] with the same architecture.

There have been relatively few optimization studies targeting the Transformer model. Lu et al. [21] conducted specialized hardware acceleration for the multi - head attention module and feed -

Table 1: FPGA Accelerators

Year	Framework	Technology	Speedup	Energy Efficiency	Baseline
2020	FTRANS [22]	FPGA	27x-81x	8.8x	CPU/GPU (RTX5000)
2021	NPE [23]	FPGA	35x	4x-6x	CPU/GPU (RTX5000)
2023	FlexRun [24]	FPGA	2.7x	—	GPU (V100)
2020	MNNFast [25]	FPGA	7020	5.4x-6.5x	CPU (Xeon 24-core)
2022	SoftMax [26]	GPU	2.5x	—	GPU (A100)

forward layer module of the Transformer model. They proposed a large matrix partitioning method, enabling all matrix multiplication operations to be completed on the same systolic array to share hardware resources. However, this research was based on the original Transformer model. The large number of parameters makes it extremely power - consuming to read data from off - chip and perform calculations on - chip, which is not an ideal solution for deployment on edge devices.

FTRANS is another FPGA - based Transformer acceleration framework proposed by Li et al. [22], aiming to accelerate large - scale language representations based on the Transformer. FTRANS significantly improves speed and energy efficiency. Table 1 show the performance newest accelerators.

4 Future Trends

In the future, models may reduce their dependence on GPUs, thereby increasing the demand for other hardware and enhancing the flexibility of model deployment. Meanwhile, the size of models may further increase, and specific hardware accelerators will be designed for specific models to meet the particular needs of different scenarios.

Moreover, FPGAs are not limited to standalone deployment; they can also be combined with CPUs and GPUs for collaborative acceleration, achieving higher resource allocation and efficiency improvements. This, in turn, enhances the speed of model inference and training and improves the energy efficiency ratio of the overall hardware system. For example, FPGA cluster architectures and cloud computing technologies can be employed to realize rapid data processing and adapt to its dynamic flow characteristics.

5 Conclusion

This paper comprehensively reviews the optimization of deep learning models on FPGAs, focusing primarily on two types of models: CNNs and Transformers. It begins by introducing the challenges brought about by the development of deep neural networks, making it difficult to deploy network models on resource-constrained devices. Subsequently, it discusses relevant optimization techniques from two aspects: software optimization and hardware optimization. It provides a detailed description of network compression techniques. Hardware optimization mainly focuses on hardware architecture design for specific network models. Finally, it discusses the future direction of FPGA accelerators.

References

[1] Yuan W, Lv Z, Schmidt T, *et al.* STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 13144-13152.

[2] Kim I, Baek W, Kim S. Spatially attentive output layer for image classification[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 9533-9542.

[3] Yuan W, Lv Z, Schmidt T, *et al.* STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 13144-13152.

[4] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.

[5] Vaswani A. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017.

[6] LeCun Y, Bottou L, Bengio Y, *et al.* Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[7] Simonyan K. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[8] He K, Zhang X, Ren S, *et al.* Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[9] Devlin, J., Chang, M. - W., Lee, K., & Toutanova, K. (2018). BERT: Pre - training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[10] Dosovitskiy, Alexander, *et al.* "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[11] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre - training. OpenAI Blog, 5(1), 1 - 1.

[12] Radford, A., Wu, J., Child, R., Chen, D., Altucher, A., Sutskever, I., ... & Amodei, D. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 6(3), 1 - 1.

[13] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few - shot learners. In Advances in neural information processing systems (Vol. 33, pp. 1877 - 1901).

[14] Cortes C, Jackel L D, Solla S, *et al.* Learning curves: Asymptotic values and rate of convergence[J]. Advances in neural information processing systems, 1993, 6.

[15] Wang N, Choi J, Brand D, *et al.* Training deep neural networks with 8-bit floating point numbers[J]. Advances in neural information processing systems, 2018, 31.

[16] Han S, Kang J, Mao H, *et al.* Ese: Efficient speech recognition engine with sparse lstm on fpga[C]//Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2017: 75-84.

[17] Parashar A, Rhu M, Mukkara A, *et al.* SCNN: An accelerator for compressed-sparse convolutional neural networks[J]. ACM SIGARCH computer architecture news, 2017, 45(2): 27-40.

[18] Struharik R, Vukobratović B, Erdeljan A, *et al.* CoNNA–compressed CNN hardware accelerator[C]//2018 21st Euromicro Conference on Digital System Design (DSD). IEEE, 2018: 365-372.

[19] Kung H T, McDanel B, Zhang S Q. Packing sparse convolutional neural networks for efficient systolic array implementations: Column combining under joint optimization[C]//Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. 2019: 821-834.

[20] Akopyan F, Sawada J, Cassidy A, *et al.* Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip[J]. IEEE transactions on computer-aided design of integrated circuits and systems, 2015, 34(10): 1537-1557.

[21] Lu S, Wang M, Liang S, *et al.* Hardware accelerator for multi-head attention and positionwise feed-forward in the transformer[A]. In: 2020 IEEE 33rd International System-onChip Conference (SOCC)[C]. Piscataway: IEEE, 2020: 84–89.

[22] Li B, Pandey S, Fang H, *et al.* Ftrans: energy-efficient acceleration of transformers using fpga[C]//Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design. 2020: 175-180.

[23] Khan H, Khan A, Khan Z, *et al.* NPE: An FPGA-based overlay processor for natural language processing[J]. arXiv preprint arXiv:2104.06535, 2021.

[24] Hur S, Na S, Kwon D, *et al.* A fast and flexible FPGA-based accelerator for natural language processing neural networks[J]. ACM Transactions on Architecture and Code Optimization, 2023, 20(1): 1-24.
[25] Jang H, Kim J, Jo J E, *et al.* Mnnfast: A fast and scalable system architecture for memory-augmented neural networks[C]//Proceedings of the 46th International Symposium on Computer Architecture. 2019: 250-263.
[26] Choi J, Li H, Kim B, *et al.* Accelerating transformer networks through recomposing softmax layers[C]//2022 IEEE International Symposium on Workload Characterization (IISWC). IEEE, 2022: 92-103.