A Multi-level Collaborative Search Method Based on Hourglass Networks (Postprint)

Authors: Guirong Chen, Qiu Zhongyu, Su Tao, Chen Dihu

Date: 2022-04-07T00:00:00+00:00

Abstract

Currently, with the rapid development of artificial intelligence, excellent neural network algorithms can be efficiently deployed on FPGA accelerators by exploring the hardware design space. However, due to large parameter volumes and excessively complex operations, algorithm-hardware matching becomes difficult, resulting in low acceleration efficiency. To achieve stronger matching between algorithms and hardware, a multi-level collaborative search method is proposed, employing the SPOS search strategy with detection accuracy and latency as evaluation objectives to search for the optimal combination of neural network architecture, quantization scheme, and hardware design parameters. This method is applied to the Hourglass network, which demonstrates excellent performance in pose estimation. While obtaining detection accuracy of candidate sub-networks before and after quantization, exhaustive search is performed on hardware design parameters to obtain estimated latency, and the optimal combination with the highest score is acquired according to the objective function. To ensure the validity of the obtained data, sub-networks require retraining and re-inference after quantization to obtain detection accuracy, while hardware design parameters are acquired through simulation testing using an accelerator template designed with Spinal HDL to obtain test latency. On average, this method reduces parameters by 83.3% compared to reference [1], with accuracy decreasing by only 0.69; compared to traditional acceleration methods, it reduces parameters by 33.2% on average, with accuracy decreasing by only 0.46, reduces the total test latency of network inference by 22.1%, and reduces the test latency in Hourglass blocks by 67.8%. Overall, this collaborative search method demonstrates certain effectiveness for Hourglass network optimization and offers greater advantages over traditional acceleration design methods.

Full Text

Preamble

Vol. 39 No. 8
Application Research of Computers (ChinaXiv Partner Journal)
Accepted Paper

A Multi-level Co-exploration Method Based on Hourglass Network

Chen Guirong, Qiu Zhongyu, Su Tao†, Chen Dihu
(School of Electronics & Information Technology, Sun Yat-Sen University, Guangzhou 510006, China)

Abstract: With the rapid development of artificial intelligence, researchers can efficiently deploy excellent neural network algorithms on FPGA accelerators by exploring the hardware design space. However, due to large parameter counts and overly complex operations, algorithm-hardware matching remains difficult, resulting in suboptimal acceleration efficiency. To achieve better algorithm-hardware alignment, this paper proposes a multi-level co-exploration method that employs the SPOS search strategy with detection accuracy and latency as evaluation objectives to search for the optimal combination of neural network architecture, quantization scheme, and hardware design parameters. Applied to the hourglass network—which demonstrates superior performance in pose estimation—this method obtains the detection accuracy of candidate sub-networks before and after quantization while using exhaustive search over hardware design parameters to estimate latency, deriving the optimal combination based on the objective function. To ensure data validity, sub-networks undergo retraining and quantized re-inference to obtain detection accuracy, while hardware design parameters are evaluated through simulation testing using an accelerator template designed with Spinal HDL. On average, compared with reference [1], this method reduces parameters by 83.3% with only a 0.69 accuracy drop; compared with traditional acceleration methods, it reduces parameters by 33.2% with only a 0.46 accuracy loss, while decreasing total network inference test latency by 22.1% and hourglass block test latency by 67.8%. Overall, this co-exploration method proves effective for hourglass network optimization and offers advantages over traditional acceleration design approaches.

Keywords: neural network; FPGA; co-exploration; hourglass network; latency model

---

0 Introduction

With the rapid advancement of artificial intelligence, pose estimation has become a major research direction in computer vision, holding significant research value and application prospects across numerous domains. Stacked hourglass

networks [1] demonstrate outstanding performance in 2D pose estimation algorithms, with many current algorithms being variants based on the hourglass architecture [2,3]. However, designers primarily focused on detection accuracy during initial development, resulting in complex network models with large parameter counts that pose substantial challenges for edge computing acceleration and inference. Current edge acceleration approaches typically start with a fixed algorithmic network structure and then optimize acceleration inference based on that static architecture, lacking methods for collaborative design across algorithmic, hardware, and other dimensions. This creates a mismatch where high-accuracy algorithms struggle to be effectively deployed on hardware accelerators [4,5], leaving considerable room for improvement in computational acceleration.

In the domain of 2D pose estimation, numerous excellent algorithms exist [1,6-9]. Using PCKh@0.5 as the accuracy metric, the hourglass network in reference [1] achieves 90.9% detection accuracy but contains 25.1M parameters. SimpleBaselines [7] reaches 91.5% accuracy but has a massive parameter count of 68.6M despite its simple and effective structure. HRNet [8] achieves 92.3% accuracy with 28.5M parameters, all represented as floating-point numbers. These outstanding algorithms present enormous challenges for edge acceleration implementation, both in terms of overall network realization and parameter volume, making hardware matching difficult.

In recent years, research teams have made significant contributions to improving neural network algorithm performance, compression quantization, and hardware design. First, to enhance neural network performance, many researchers have adopted Neural Architecture Search (NAS) [10–13] as an advanced method to escape the tedious work of manual parameter tuning. Reference [11] proposed NASNet, designing a novel search space and using reinforcement learning optimization to improve search performance. Reference [12] introduced ENAS, which enhanced NAS efficiency through weight sharing, reducing GPU computation time by over 1000×. Reference [14] proposed the One-Shot method, achieving rapid overall search speed by training a supernet as an auxiliary model. Second, to reduce neural network parameters and accelerate inference, considerable progress has been made in quantization [15]. Reference [16] binarized network weights, constraining computations primarily between +1 and -1, thereby reducing network size and computational load. Reference [17] quantized floating-point operations during inference to integer operations, ultimately quantizing weights and activations to 8 bits. Reference [18] employed mixed-precision quantization to find appropriate bit widths for energy reduction. To enable faster neural network inference, numerous aspects in accelerator hardware design warrant attention, such as Processing Element (PE) count, parallelism, and dataflow reuse patterns, which designers have extensively explored. Reference [19] utilized the Roofline model for design space exploration of CNN accelerator dataflow techniques, investigating performance across different dataflow reuse patterns. Reference [20] explored PE utilization and parallelism, achieving substantial performance improvements. Reference [21] proposed an accelerator ar-

chitecture enabling inter-layer pipelining of convolutional layers, implementing parallel computation by unfolding across three dimensions: output feature map count, width, and height. However, these efforts all approach optimization from a single perspective—such as algorithmic accuracy, hardware parameter count, or hardware design methodology—without collaborative consideration for joint optimization to obtain optimal performance.

Research on multi-objective co-exploration and hardware-aware search [4,5,18,22–30] has grown increasingly prevalent, with designers considering not only algorithmic accuracy but also hardware performance metrics such as power consumption, latency, area, and resource utilization. Reference [4] considered adaptability between layers and dataflow reuse patterns to select the most suitable dataflow mode for efficient network deployment. Reference [24] employed reinforcement learning for search, with a search space encompassing algorithmic architecture and mixed-precision quantization, establishing latency models and energy constraints to achieve reductions in both energy and latency. Reference [29] defined a novel hardware search space to generate optimal neural network models with latency guarantees on target FPGA platforms. Reference [30] proposed a device-circuit-architecture co-exploration framework including device types, circuit topologies, and neural network hyperparameters for performance optimization. However, these works are primarily based on image classification, lacking research on pose estimation and hourglass networks.

In summary, excellent algorithms in pose estimation lack hardware adaptability; network optimization, compression, and hardware acceleration efforts primarily consider single-performance optimization; and multi-objective co-exploration methods lack research on pose estimation. Neural architecture search methods can automatically explore the neural network design optimization space, allowing designers to further modify network structures for higher precision while incorporating hardware-side objectives as optimization targets. These methods effectively solve the mismatch problem between high-accuracy algorithms and hardware, enabling high-speed inference with minimal precision loss through simpler approaches.

This paper proposes a multi-level co-exploration method for hourglass networks in pose estimation that balances detection accuracy and latency, thoroughly exploring optimization spaces in algorithm and hardware design. The main contributions include: (1) A co-optimization method for hourglass networks covering algorithm, compression quantization, and hardware design, achieving significant performance improvements with low accuracy loss; (2) Establishment of a latency model that closely approximates actual accelerator operation to guide the co-exploration process; (3) Design of an accelerator template using Spinal HDL to verify the accuracy of the latency model through simulation testing of obtained dataflow and parallelism parameters.

1.1 Problem Definition

Designers focus on network model accuracy, making the goal of various NAS efforts to find a more precise network structure. However, hardware designers prioritize accelerator performance metrics such as latency and power consumption, easily leading to algorithm-hardware mismatches. The multi-level co-exploration method aims to optimize hourglass network inference latency while minimizing accuracy loss.

Accuracy acquisition requires candidate models to infer on the validation set. All accuracy values in this paper are specific MPII dataset metric values PCKh@0.5. The Percentage of Correct Keypoints (PCK) evaluates human joint localization accuracy, typically denoted as PCK@k. For keypoint i, when the distance d_i between the predicted and ground-truth positions is less than a certain proportion k of the head length L_{head}, it is called PCKh@k, calculated as follows:

$$\text{PCKh@k} = \frac{\sum_i \mathbb{1}(d_i < k \cdot L_{\text{head}})}{\sum_i 1}$$

When k equals 0.5, it is expressed as PCKh@0.5.

Network inference latency is obtained through mathematical modeling for prediction, which can be described using Equation (2) below. The latency model will be introduced in detail later.

$$T_{\text{total}} = T_{\text{comp}} + T_{\text{off-chip}} + T_{\text{on-chip}}$$

Total latency includes computation latency, off-chip memory access latency, and on-chip data movement latency.

To better evaluate the searched network models and accelerator design parameters, this paper proposes an objective function that also guides the entire co-exploration process, as shown in Equation (3):

$$\text{Score} = \alpha \cdot \frac{\text{Acc}_{\text{pre}}}{\text{Acc}_{\text{baseline}}} + \beta \cdot \frac{\text{Acc}_{\text{post}}}{\text{Acc}_{\text{baseline}}} - \gamma \cdot \frac{T_{\text{pred}}}{T_{\text{baseline}}}$$

where Acc_{pre} and Acc_{post} represent the accuracy before and after quantization, respectively, primarily ensuring good accuracy in both cases. Here, accuracy and latency undergo normalization to guarantee consistent value ranges. α, β, and γ are weighting factors obtained through multiple parameter tuning iterations; this paper adopts values of 0.4, 0.4, and 0.2, respectively.

1.2 Method Overview

The multi-level co-exploration method framework primarily comprises supernet training, co-exploration, and hardware deployment, as shown in Figure 1 [Figure 1: see original paper]. SPOS [15] (Single Path One-Shot) separates supernet training from search, with the search process employing genetic algorithms for multiple iterations. Input constraints for the co-exploration method mainly include hourglass block models at the algorithmic end, intermediate compression quantization methods, and accelerator design parameters for dataflow reuse patterns and parallelism, where parallelism is reflected in DSP count from FPGA synthesis results. The search output is the optimal combination of network model, quantization method, accelerator dataflow, and parallelism parameters evaluated by the objective function. The network architecture requires retraining, quantization, and inference to ensure structural validity and accuracy. Obtained hardware design parameters adjust the accelerator template to generate new RTL (Register Transfer Level) code for subsequent synthesis and implementation, ultimately achieving high-speed inference on FPGA.

In traditional neural network-to-hardware acceleration deployment, algorithm designers first determine the network structure, after which hardware designers complete accelerator design based on this fixed structure, leading to suboptimal solutions. The proposed method treats algorithm, quantization, and hardware design as a unified search space, enabling joint optimization.

The entire search process begins by encoding hourglass blocks in the supernet to facilitate genetic algorithm operations such as crossover and mutation. After supernet training and quantization, the search proceeds. During the search phase, candidate sub-networks are randomly sampled based on hourglass block encoding to form the initial population. The pre-quantization candidate sub-networks infer on the validation set, while sub-networks using different quantization schemes also undergo inference to obtain pre- and post-quantization accuracy for objective function evaluation. The genetic algorithm selects optimal sub-networks from candidates as parents for the next iteration. Simultaneously, hardware design employs exhaustive search, returning the lowest latency estimated by the latency model to the objective function along with corresponding hardware design parameters. By combining pre-/post-quantization accuracy and estimated latency, the objective function yields scores to ensure optimal sub-network structure, quantization method, and hardware design parameters.

Since supernet training uses weight sharing, sub-network accuracy has room for improvement. Therefore, after obtaining the target network model, its structure undergoes retraining, quantization, and validation to achieve better accuracy. Acquired hardware design parameters adjust accelerator template settings to obtain new RTL code for subsequent synthesis and implementation processes.

2.1 Algorithm Level

The original hourglass block structure is shown in Figure 2 [Figure 2: see original paper], where each small block is a residual block. Residual block operations are relatively complex to implement in hardware, and each branch pathway in the hourglass block requires additional buffering for feature fusion, leaving optimization space in both branch feature fusion and feature extraction operations.

For branch feature fusion, this paper' s search space offers three choices: (1) No feature fusion, which removes bypass branches and discards previous features; (2) Direct fusion without convolution, a shortcut approach without convolution overhead; (3) Original convolutional feature fusion, as shown in Figure 3 [Figure 3: see original paper]. Convolution-free feature fusion significantly reduces buffering, computation time, and additional weight parameters compared to convolutional feature fusion. No feature fusion reduces buffering time most substantially. In terms of operator block search space, the original residual block requires considerable computation and buffering time; this paper attempts to replace it with depthwise separable convolution blocks to reduce these overheads.

To facilitate genetic encoding of excellent hourglass block structures, the computation order of hourglass blocks in the accelerator is encoded. Fusion methods are encoded as 0, 1, 2, and operator blocks as 0, 1. For example, original convolutional feature fusion corresponds to 2, and the original residual operation block encodes as 0, making the original structure [2,0,2,0,2,0,2,0,0,0,0,0,0]—a total of 13 positions. Figure 3 represents [2,0,1,0,0,0,2,0,0,0,0,0,0].

2.2 Quantization Level

In neural network compression, methods include quantization, pruning, sparsification, and distillation, with quantization being the most common. Since each layer of a neural network has different data distributions and each quantization method exhibits certain differences, applying different quantization methods to different structures may yield varying results. To explore the best quantization approach, this paper' s exploration space includes fixed-point and integer types.

For 8-bit fixed-point quantization, this paper uses Equation (4) to explore integer bit widths of 1, 2, and 3 bits, where m represents signed two' s complement, b represents fractional bit width, and N represents total quantization bit width.

$$\mathrm{Fix}N(m) = \frac{m}{2^b}, \quad b = N - 1 - m$$

For example, Fix8(2) indicates 2 integer bits, 1 sign bit, and remaining fractional bits, as shown in Figure 4 [Figure 4: see original paper].

The alternative approach explores 8-bit integer quantization, primarily expressed in Equations (5) and (6):

$$q = \text{round}\left(\frac{f}{S}\right) + Z$$

$$S = \frac{\max(|f|)}{2^{N-1} - 1}$$

where f is the floating-point number, S represents the scaling factor, q is the quantized integer, and Z represents the zero-point.

2.3 Hardware Design Level

CNN accelerator design requires reasonable dataflow reuse patterns and parallelism based on FPGA resource constraints, as both factors affect accelerator performance. Inappropriate dataflow may cause additional latency overhead, while excessive parallelism may exhaust resources and insufficient parallelism wastes resources.

1) Data Reuse Patterns

Dataflow refers to the computation order of convolutions, primarily categorized as Output Reuse (OR), Input Reuse (IR), and Weight Reuse (WR) [20]. Different dataflows exhibit different computation sequences in accelerators, each with distinct execution times. Therefore, accelerator design must determine optimal dataflow based on FPGA resource constraints. The OR pseudocode is shown in Algorithm 1, with notation explained in Table 1 .

Table 1 Explanation of Some Notations

Symbol	Meaning
Ic, Oc, Oh, Ow	Input channels, output channels, height, width
Tic, Toc, Tox, Toy, Kh, Kw	Tiling for input/output channels, height/width, kernel dimensions
Pic, Poc, Pox, Poy	Parallelism for input/output channels, output feature map dimensions
x, y, m, n, etc.	Loop variables

Algorithm 1: Output Dataflow Reuse (OR)

Input: IFM[Ic][iw][ih], Weight[Ic][Oc][Kw][Kh]
Output: OFM[Oc][Oh][Ow]

```
for(y=0; y<Oh; y+=Toh) {
  for(x=0; x<Ow; x+=Tox) {
    for(m=0; m<Oc; m+=Toc) {
```

```
      for(n=0; n<Ic; n+=Tic) {
        // Computation kernel
      }
    }
  }
}
```

2) Parallelism

Parallelism refers to how many computations can be performed simultaneously during convolution, representing the number of multiplications executed concurrently. Parallel computation can be unfolded across different dimensions, such as convolution kernels, input/output channels, and feature map dimensions. Under the three dataflow patterns, Pic and Poc correspond to parallel computation of input and output channels, while Pox and Poy represent parallel computation of convolution windows (i.e., feature points in output feature map height and width directions). Different parallelism combinations also exhibit varying computation times.

As shown in Figure 5 [Figure 5: see original paper], input channel parallelism means that during output feature map computation, convolution kernels and corresponding input feature map channels are multiplied and accumulated to obtain one output feature point. This process is implemented on FPGA using multiple multipliers for parallel computation; thus, higher parallelism may reduce computation latency. Parallelism in other dimensions follows convolution computation principles and will not be elaborated here.

2.4 Multi-level Co-exploration Space

Based on the above analysis, this paper summarizes the multi-level co-exploration space in Table 2 below.

Table 2 Search Space of Multi-level Co-exploration

Algorithm-side Search Space	Quantization Method Search Space	Hardware Design Search Space
Residual convolution block	Dataflow	Total parallelism
Depthwise separable convolution block	No feature fusion	Branch fusion operation
Direct feature fusion	Convolution feature fusion	Fix8(1), Fix8(2), Fix8(3)

---

3 Latency Model

To accurately evaluate co-exploration results, this paper establishes a latency model that closely approximates actual accelerator operation, as shown in Equation (2) above. The latency model comprehensively considers time costs for computation, on-chip, and off-chip data access, providing greater accuracy and completeness than simple estimations based on network model parameters and computation volume. In modeling computation latency, this paper addresses the issue where actual computational performance falls below peak performance due to PE utilization. In modeling memory access latency, it fully considers repeated data access patterns under different dataflows. This model is suitable for latency estimation of common CNN accelerators with varying resource constraints, dataflows, and parallelism.

3.1 Computation Delay

Computation delay represents the time required for entire network computation, with convolution accounting for over 90% of network computation. Computation delay can be expressed as computation volume divided by the product of total parallelism and utilization, as shown in Equation (7), where standard convolution computation volume is given by Equation (8). PE utilization can be expressed as Equation (9).

$$T_{\text{comp}} = \frac{\text{Computation Volume}}{\text{Parallelism} \times \text{Utilization}}$$

$$\text{Computation Volume} = \sum_{\text{layer}} (Kh \times Kw \times Ic \times Oc \times Oh \times Ow)$$

$$\text{Utilization} = \frac{\text{Actual Active PEs}}{\text{Total PEs}}$$

3.2 On-chip Memory Access Delay

On-chip memory access delay refers to data movement time between on-chip buffers and computation arrays. Since the accelerator design in this paper employs a pipeline architecture, on-chip data movement time is hidden within network computation, and therefore this component is omitted from the total latency.

3.3 Off-chip Memory Access Delay

Off-chip memory access delay represents the time for reading and writing data from DDR to on-chip buffers, specifically calculated as the total data volume under a certain dataflow (M_{dtflw}) multiplied by bit width (DW) divided by bus bandwidth (BW), as shown in Equation (10). Due to limited on-chip capacity in FPGAs, single-layer data may exceed on-chip buffer size, requiring segmentation

into smaller blocks. Under different dataflows, one data type is reused first while other data block types require repeated reading or writing, resulting in varying memory access volumes for each dataflow.

$$T_{\text{off-chip}} = \frac{M_{\text{dtflw}} \times DW}{BW}$$

4 Accelerator Design

To verify the accuracy of the latency model and the effectiveness of the entire multi-level co-design approach, this paper employs the novel hardware description language Spinal HDL to design a CNN accelerator template. Similar to Verilog, Spinal HDL adopts a hardware-oriented mindset, unlike high-level synthesis which focuses more on algorithmic aspects while neglecting hardware design details. Since hardware-side search incorporates numerous hardware parameters, the CNN accelerator template in this paper achieves parameterization for different bit widths and parallelism levels, enabling the entire template to test different hardware parameter combinations through various test scripts. Test latency acquisition first obtains configuration instructions for the corresponding sub-network structure to complete parameter configuration for each layer, then samples waveforms of the control module state machine' s read/write processes to obtain cycle counts.

The complete accelerator structure comprises several components: The control module contains a state machine that controls computation, read, and write processes, with different controllers designed to support various dataflows in the search space. The memory access module serves as a bridge between on-chip and off-chip memory, writing on-chip data back to DDR and fetching DDR data to on-chip buffers. On-chip buffer modules cache data and weights, divided into two data buffers and one weight buffer. The reordering module rearranges cached data before delivery to the computation array. The computation array employs an inter-layer reuse pattern, with each layer reusing the array containing three-dimensional parallelism (input channel, convolution window, and output channel) to match hardware-side parallelism settings, capable of performing convolution, pooling, and matrix addition operations, as shown in Figure 6 [Figure 6: see original paper].

5.1 Experimental Setup

The experimental environment configuration is as follows: Ubuntu 16.04 LTS 64-bit system, PyTorch 1.7.1 deep learning framework, NVIDIA GTX 1080ti GPU. This paper uses the MPII dataset containing 25K images, employing PCKh@0.5 as the evaluation metric. The hourglass block stack number is 1, using RMS optimizer with a learning rate of 1e-3. Supernet training runs for 220 epochs,

with genetic algorithm search iterating 20 times. The FPGA platform is Xilinx Zynq series 7z045 as the resource constraint. Considering that excessive resource utilization causes routing difficulties, DSP limit is set to 800, total on-chip buffer size to 1.312MB (two data buffers at 512KB each, weight buffer at 288KB).

5.2 Experimental Results

a) Traditional Acceleration Method

The traditional acceleration method is a hardware design approach based on fixed algorithmic structures. Therefore, this paper conducts design searches for dataflow, parallelism parameters, and quantization methods targeting the original hourglass network structure.

Figure 7 [Figure 7: see original paper] shows the relationship between parallelism and latency, revealing that under the same dataflow, total parallelism of 1536 yields the lowest latency, not at the maximum parallelism of 1600. This demonstrates the significance of search—higher parallelism does not necessarily mean lower latency, as it must align with specific convolution channel counts. Figure 8 [Figure 8: see original paper] illustrates the relationship between dataflow and latency, showing that for hourglass networks, Output Reuse (OR) achieves lower latency under the same parallelism.

Table 3 presents keypoint detection results for different quantization methods, showing detection accuracy for each keypoint. Various quantization methods reduce detection accuracy for different joints in the original hourglass network, with Int8 showing the minimal loss of only 0.2. Since ankles are more easily occluded, their detection performance is worst, with the largest accuracy drop after quantization.

Table 3 Key Point Estimation Results of Different Quantization Methods

Method	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Average
Fix8(1)	95.4	93.5	···	···	···	···	···	···
Fix8(2)	93.6	90.1	···	···	···	···	···	···
Fix8(3)	···	···	···	···	···	···	···	···

The top five search results for the original hourglass network structure are shown in Table 4 , including pre-/post-quantization accuracy, parameter count, dataflow, and parallelism. Predicted latency refers to values obtained from the established latency model, while test latency refers to simulation results from the accelerator template, primarily used to verify latency model correctness. Through multiple repeated tests, the average error between predicted and test

latency is only 4.02%, sufficiently demonstrating the latency model' s effectiveness.

Table 4 shows that in traditional acceleration method search experiments, the optimal performance configuration uses parallelism combination [Pox, Poy, Pic, Poc] = [8,3,8,8], Output Reuse (OR) dataflow, Int8 quantization, achieving lowest latency with minimal accuracy loss.

Table 4 Search Results of the Original Structure of Hourglass Network

Rank	Original Accuracy	Quantized Accuracy	Quantization	Parameters (MB)	Reuse Mode	Parallelism	DSPs	Predicted Cycles	Test Cycles
1	···	···	Int8	···	OR	[8,3,8,8]	···	1.250$×10^{7}$	1.284$×10^{7}$
2	···	···	···	···	···	[8,4,8,6]	···	1.256$×10^{7}$	1.270$×10^{7}$
···	···	···	···	···	···	···	···	···	···

b) Multi-level Co-exploration Results

After reconstructing the supernet for hourglass modules and employing multi-level co-exploration search, the top five results are shown in Table 5 , including network structure encoding, pre-/post-quantization accuracy, parameter count, and test latency. The optimal hourglass block structure is [0,1,1,0,1,1,2,0,1,0,0,1,0], achieving 84.10% post-quantization accuracy with only 2.23MB parameters. Since quantization, dataflow, and parallelism searches use the same exhaustive method as traditional approaches, these parameter searches are essentially consistent. On average, the multi-level search method achieves 84.29% original accuracy and 84.23% post-quantization accuracy with minimal loss, using only 2.27MB parameters. Test latency comparisons are discussed later.

Table 5 Search Results of Multi-level Co-exploration

Network Structure	Original Accuracy	Quantized Accuracy	Quantization	Reuse Mode	Parallelism	Parameters (MB)	Whole Network Cycles	Hourglass Block Cycles
[0,1,1,0,1,1,2,0,1,0,0,1,0]	84.29	84.10	Int8	OR	[8,3,8,8]	2.23	1.009$×10^{7}$	7.58$×10^{6}$
···	···	···	···	···	···	···	···	···

Table 6 compares traditional method optimal/average results (from Table 4) with co-exploration optimal/average results (from Table 5). On average, compared with reference [1] using a single stack block, this method reduces parameters by 83.3% with only 0.69 accuracy drop; compared with traditional acceleration methods, it reduces parameters by 33.3% with only 0.46 accuracy loss,

decreases total network inference test latency by 22.1%, and reduces hourglass block test latency by 67.8%. Compared with other pose estimation algorithms, this work' s parameter count is 0.83% of reference [7], 2% of reference [8], and 0.9% of reference [9]. While accuracy may lag behind some state-of-the-art algorithms, this approach offers advantages in parameter count and network complexity, better adapting to FPGA acceleration.

c) Experimental Comparison

Since this work optimizes based on hourglass networks, the most direct performance comparison targets the original structure—specifically, results from reference [1] with a single stack block and traditional acceleration methods for the original structure, as shown in Table 6.

In summary, the proposed co-exploration method demonstrates favorable performance in reducing runtime latency with minimal accuracy loss. Overall, multi-level co-exploration achieves better acceleration effects than traditional design processes.

---

6 Conclusion

This paper addresses hourglass networks for pose estimation by integrating hourglass block structure, quantization methods, and hardware design, using runtime latency and accuracy as objectives. It establishes a latency model closely approximating actual FPGA accelerator operation and proposes a multi-level co-exploration method, validated through a configurable accelerator template. Results show this co-design approach outperforms traditional methods in both parameter count and latency with minimal accuracy degradation, effectively addressing algorithm-hardware mismatch in hourglass networks.

Future improvements include considering different hourglass block counts, convolution channel numbers, etc., in the search space to obtain higher-accuracy networks while maintaining lower latency.

References

[1] Newell A, Yang K, Deng J. Stacked hourglass networks for human pose estimation [C]// Proceedings of the 14th European Conference on Computer Vision, Springer, 2016: 483-499.

[2] Zhou Yan, Liu Ziqin, Zeng Fanzhi, et al. Overview of 2D human pose estimation based on deep learning [J]. Computer Science and Exploration, 2021, 15(4): 641-657.

[3] Liu Yong, Li Jie, Zhang Jianlin, et al. Research progress of 2D human pose estimation based on deep learning [J]. Computer Engineering, 2021, 47(3): 1-16.

[4] Li Chuxi, Fan Xiaoya, Zhang Shengbing, et al. Hardware-Aware NAS Framework with Layer Adaptive Scheduling on Embedded System [C]// ASPDAC' 21: 26th Asia and South Pacific Design Automation Conference. 2021.

[5] Zhen Dong, Gao Yizhao, Huang Qijing, et al. HAO: Hardware-aware neural Architecture Optimization for Efficient Inference [J]. 2021.

[6] Wei S E, Ramakrishna V, Kanade T, et al. Convolutional Pose Machines [J]. IEEE, 2016.

[7] Xiao Bin, Wu Haiping, Wei Yichen. Simple Baselines for Human Pose Estimation and Tracking [J]. arXiv e-prints, 2018.

[8] Sun Ke, Xiao Bin, Liu Dong, et al. Deep High-Resolution Representation Learning for Human Pose Estimation [J]. arXiv e-prints, 2019.

[9] Zhang Feng, Zhu Xiatian, Dai Hanbin, et al. Distribution-Aware Coordinate Representation for Human Pose Estimation [J]. 2019.

[10] Baker B, Gupta O, Naik N, et al. Designing Neural Network Architectures using Reinforcement Learning [J]. 2016.

[11] Zoph B, Vasudevan V, Shlens J, et al. Learning Transferable Architectures for Scalable Image Recognition [J]. 2017.

[12] Pham H, Guan M Y, Zoph B, et al. Efficient Neural Architecture Search via Parameter Sharing [J]. 2018.

[13] Cai Han, Yang Jiacheng, Han Song, et al. Path-Level Network Transformation for Efficient Architecture Search. In ICML, 2018.

[14] Brock A, Lim T, Ritchie J M, et al. SMASH: One-Shot Model Architecture Search through HyperNetworks [J]. 2017.

[15] Guo Zichao, Zhang Xiangyu, Mu Haoyuan, et al. Single Path One-Shot Neural Architecture Search with Uniform Sampling [J]. 2019.

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

[17] Jacob B, Kligys S, Chen B, et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference [J]. 2017.

[18] Wang Kuan, Liu Zhijian, Lin Yujun, et al. HAQ: Hardware-Aware Automated Quantization With Mixed Precision [C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2019.

[19] Chan P, Park S, Park C S. Roofline-Model-Based Design Space Exploration for Dataflow Techniques of CNN Accelerators [J]. IEEE Access, 2020, 8: 172509-172523.

[20] Chen Y H, Krishna T, Emer J S, et al. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks [J]. IEEE Journal of Solid-State Circuits, 2017, 52(1): 127-138.

[21] Chen Y H, Yang T J, Emer J, et al. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices [J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 2, pp. 292-308, June 2019.

[22] Ma Yufei, Cao Yu, Vrudhula S, et al. Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 7, pp. 1354-1367, July 2018.

[23] Sharma H, Park J, Suda N, et al. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks [J].

[24] Gong Chengyue, Jiang Zixuan, Wang Dilin, et al. Mixed Precision Neural Architecture Search for Energy Efficient Deep Learning [C]// ICCAD 2019. pp. 1-7.

[25] Jiang Weiwen, Li Yang, Sha H M, et al. Hardware/Software Co-Exploration of Neural Architectures [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, PP(99): 1-1.

[26] Luo Xiangzhong, Liu Di, Huai Shuo, et al. HSCoNAS: Hardware-Software Co-Design of Efficient DNNs via Neural Architecture Search [J]. In DATE, 2021.

[27] Jiang Yuhang, Wang Xin, Zhu Wenwu. Hardware-Aware Transformable Architecture Search with Efficient Search Space [C]// 2020 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2020.

[28] Qiu Zhongyu, Li Jianli, Liang Dongbao, et al. A CNN/FPGA Co-Exploration Method Based on Hourglass Network. [C]// 2021 IEEE 4th International Conference on Electronics Technology (ICET), 2021, pp.

[29] Jiang Weiwen, Zhang Xinyi, Sha H M, et al. Accuracy vs. Efficiency: Achieving Both through FPGA-Implementation Aware Neural Architecture Search. [C]// 2019 56th ACM/IEEE Design Automation Conference (DAC), 2019, pp. 1-6.

[30] Jiang Weiwen, Lou Qiuwen, Yan Zheyu, et al. Device-Circuit-Architecture Co-Exploration for Computing-in-Memory Neural Accelerators [J]. IEEE Transactions on Computers, 2020, PP(99): 1-1.

Note: Figure translations are in progress. See original paper for figures.