Technology Strategy for Meticulous Governance of Smart Cities: Memory Bus Width Aware Pruning for Efficient Image Super-Resolution Implementation in FPGA

Journal of Physics: Conference Series **2294** (2022) 012003 doi:10.1088/1742-6596/2294/1/012003

Technology Strategy for Meticulous Governance of Smart Cities: Memory Bus Width Aware Pruning for Efficient Image Super-Resolution Implementation in FPGA

Chuo Zhang

School of Government, Beijing Normal University

zhangchuo@mail.bnu.edu.cn

Abstract. The mega-city governance is based on the aggregation, collation, development and application of multi-scene data, and efficient access to scene data is a key link to promote the meticulous governance of smart cities. Super-resolution technique is the process of upscaling and improving the details within an image. In this paper, we implement a 16-layer residual neural network (ResNet) for the efficient image super-resolution in FPGA. We discover that the memory access is the major performance bottleneck of this implementation. To reduce the memory access overhead, we design a pruning algorithm with the consideration of the memory bus width. Since the memory used in our design generates 256 bits for each access, the proposed pruning algorithm drops the kernel by aligning this bit width. That is, all kernels in one layer are ranked by its L1-norm and we drop the kernels out of the 256 bits. The experimental results show that the proposed method reduces the number of weights by 50% compared with the baseline. As a result, the inference speed can be enhanced by 3 times.

1. Introduction

Smart city construction is considered to be an important solution to solve the challenges of mega-city governance, and China has been successively carrying out digital transformation of urban governance in Beijing, Shanghai, Shenzhen, Hangzhou and other places. In this process, how to improve the efficiency of technological tools while ensuring the effectiveness of governance is a key question to test the level of governance in China's mega-cities.

In recent years, convolution neural networks (CNNs) have become the dominant approach for a variety of computer vision tasks. To achieve better results, the depth of network is getting deeper and deeper. Neural networks are typically over-parameterized, and there is significant redundancy for deep learning models[1]. The computational overhead caused by massive parameters is a huge challenge facing edge devices with limited computing resources. In order to be operated on resource-constrained hardware, CNN models need to be compressed into light-weight one by pruning out some less important parameters[2].

Liu et al.[6] proposed an effective pruning strategy, which can reduce the parameters of the original residual network[3] (ResNet) and reduce the data access overhead. Their method mainly uses the parameters of the batch normalization (BN) layer in the residual network to achieve pruning. However, in certain tasks, the improved residual network can often achieve better results than the original architecture. For example, in the image super-resolution task, Lim et al.[5] improve the performance by removing BN layers in ResNet, compared to SRResNet[4]. And proposed an enhanced version of the

residual network architecture with the simpler structure. Therefore, a pruning method for specific residual network is needed.

Compared to the massive computing-intensive and memory-intensive features from the neural network applications, Field Programmable Gate Arrays (FPGA) is a good acceleration hardware device, due to its own flexibility. Recently, many people are using FPGA to accelerate CNNs, but they have not considered the customized design of specific networks. In order to fully exploit the performance of FPGA and consider the memory access characteristics, we need to carry out a co-design approach for the model.

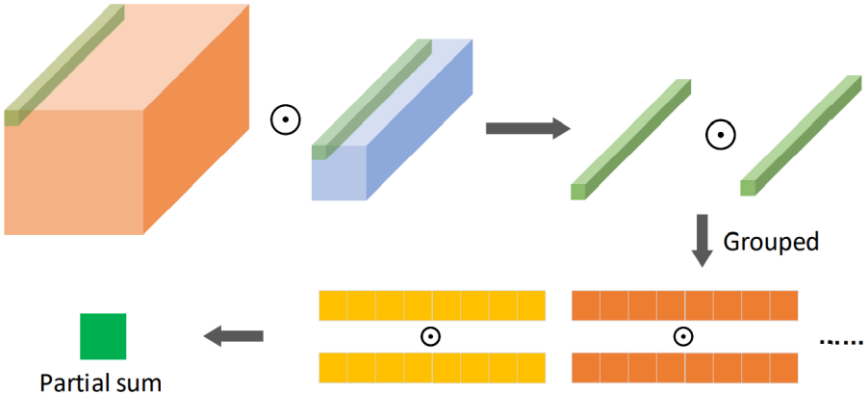


Figure 1 Data grouping in convolution operations.

2. THE PROPOSED METHOD

2.1. Model compression

To extract more feature information, Ledig et al.[4] uses original ResNet with a batch normalization (BN) layer. Each residual block contains two convolution layers, and each convolution layer is followed by a BN layer. By deepening the number of network layers, SRResNet has achieved good results in image super-resolution task. Through lots of experiments and analysis, Lim et al.[5] remove the BN layers from residual block. Since BN layers get rid of range flexibility from networks by normalizing the features, they decide to remove them. Meanwhile, GPU memory usage is also sufficiently reduced since the BN layers consume the same amount of memory as the preceding convolution layers. Due to the excellent performance of EDSR, we implement it on FPGA.

Generally, the residual network usually contains a batch normalization layer, the current pruning method for the ResNet mainly uses the characteristics of the BN layer. The BN layer contains two parameters, which are γ (scale factor) and β (bias factor). The method[6] applies an L1 regularization on the scaling factors to achieve the sparse effect, using the characteristics of parameters in the BN layer. Pushing the value of the BN scale factor close to zero through L1 regularization can identify insignificant channels (or neurons), because each scale factor corresponds to a specific convolution channel (or neuron in a fully-connected layer). It is easy to implement pruning without introducing any changes to the existing CNN architecture.

However, there have no BN layers in EDSR network structure. In each ResNet block, we establish key-value pairs for channel scaling factors in convolution layer. Then use L1-Norm regularize all the parameters of the index key corresponding to the filters, and the regularized value is used as the value of the corresponding key-value pair. Therefore, we can set a threshold as the pruning ratio, and remove the channels smaller than the threshold according to the index value of the filters num. Since the ResNet has a skip-connection branch, the proposed method only operates on the convolution layer, and all ResNet block share the same pruning threshold to ensure the correct dimensions of feature maps.

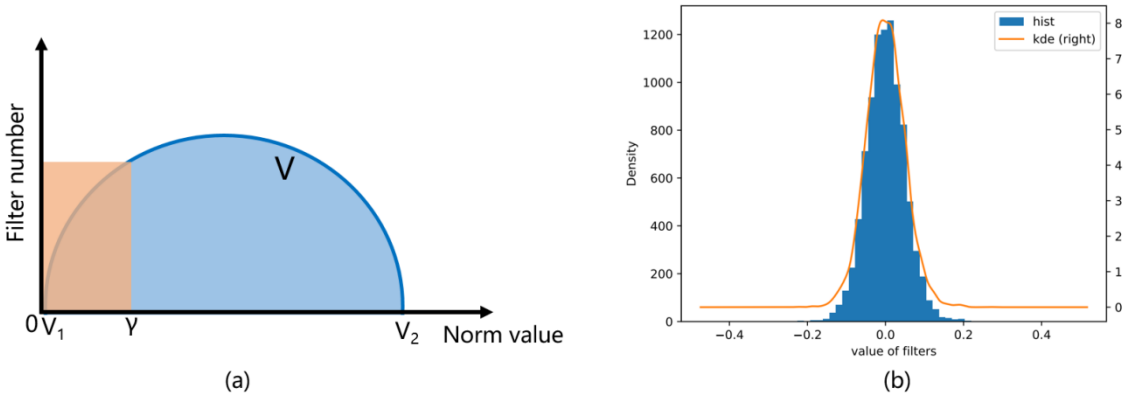


Figure 2 Norm evaluation index condition and true distribution of filter parameters.

In general, existing methods consider that the smaller the norm of the filter (Norm), the closer the corresponding feature map is to 0. Therefore, the following judgment is made: the contribution of filters with small norm values to the network is small, so these filters can be removed without seriously affecting the performance of the network. Therefore, we use the norm of the filter (such as the L1 norm) to rank the importance, and remove the filter with small norm value.

However, the above-mentioned norm evaluation index actually contains two implicit conditions: one is that the norm standard deviation is large enough; the other is that the minimum norm value is close to 0. In Figure 2(a), the blue curve represents the ideal norm distribution of the filters (i.e. parameters) in the network, where v_1 and v_2 are the minimum and maximum values of the norm distribution, respectively. To choose an appropriate threshold γ (shaded in red) as the filter's importance indicator, two requirements should be satisfied, namely, a large norm standard deviation and a small norm minimum value. However, the real filter norm distribution may not meet the above conditions. Figure 2(b) is the histogram (Hist) of the parameters in the first layer of convolution in the first residual block in the original model and Kernel Density Estimation (KDE) curve. The Hist and KDE of the parameters can well represent the distribution of the data. As can be seen from Figure 2(b), the range of parameter values is concentrated near the 0 point, so when using the norm-based pruning method, since the filters are concentrated near the points with smaller values, the norm is also smaller. Many filters have similar importance (similar norm values), and it is impossible to tell which filter contributes less to the network.

In order to solve the above problems, combined with the idea of Geometric Median (GM) pruning[7], we propose a novel filter pruning method, the pruning method based on geometric centre and norm correction (L1-FPGM). Different from the previous method of pruning based on the filter norm value alone, L1-FPGM selects the filter with the maximum replaceability and corrects it by the L1 norm value to avoid the same distance from the geometric centre. Good pruning effect. Specifically, we first compute the geometric median of all filters within the same convolutional layer. The geometric median is an estimate of the centre of a point in Euclidean space (Euclidean space).

Next, the geometric median is introduced: Given a set of n points, $a(1), \ldots, a(n)$, where $a(i) \in \mathbb{R}^d$, the geometric median x^* is such that This set of data has the smallest sum of Euclidean distances, where $x^* \in \mathbb{R}^d$. The geometric median method is expressed as:

$$x^* \in \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} f(x) \tag{1}$$

where

$$f(x) = \sum_{i \in [1,n]} \left\| x - a^{(i)} \right\|_2 \tag{2}$$

We think that the value of the filter is also a point in the Euclidean space, so the "centre" of the filter can be obtained by calculating GM, which is their common property. If a filter is close to the geometric centre, it can be considered that the information of this filter is coincident with other filters, or even redundant, and removing this filter will not have a large negative impact on the network. That is, its function can be replaced by other filters. Assuming that the number of layers of the neural network is L, we denote the number of input channels and the number of output channels of the i_{th} convolutional layer by N_i and N_{i+1}, respectively. $F_{i,j}$ denotes the jth filter of the ith layer convolution. Therefore, the filter $F_{i,j}$ is $\mathbb{R}^{N_i \times K \times K}$, where K is the size of the filter. We use the geometric center of the filter, that is, x^{GM}, to represent the inner layer of the i-th layer in the network

Common information for all filters. x^{GM} is the geometric centre of the filter in Euclidean space and can be expressed as Equation (3).

$$x^{GM} = \underset{x \in \mathbb{R}^{N_i \times K \times K}}{\operatorname{argmin}} \sum_{j' \in [1, N_{i+1}]} \left\| x - F_{i,j'} \right\|_2 \tag{3}$$

After getting x^{GM}, we can find F_{i,j^*} by Equation (4). Since the geometric median is a classical robust estimator of data centrality in Euclidean space, the chosen filter F_{i,j^*} contains feature information shared by other filters besides itself, as well as its own characteristic information ($\left\| F_{i,j'} \right\|_1$). Therefore, removing filters F_{i,j^*} has little negative impact on network performance.

$$F_{i,j^*} = \underset{F_{i,j'}}{\operatorname{argmin}} \left(\left\| F_{i,j'} - x^{GM} \right\|_2 + \left\| F_{i,j'} \right\|_1 \right) \tag{4}$$

2.2. Algorithm acceleration

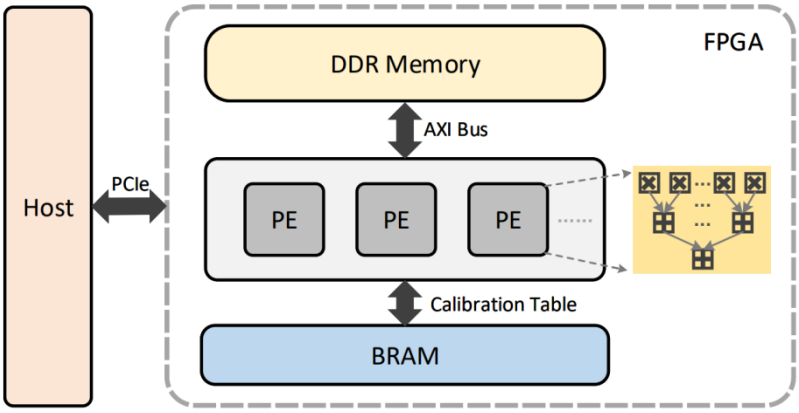


Figure 3 Architecture of our accelerator.

FPGA processing of floating point numbers consumes a lot of computing resources, so we have to quantify the data. We use the INT8 quantization method mentioned in Nvidia TensorRT to quantize the weight parameter of FP32 to INT8, and get the corresponding calibration table. At the cost of little loss of data precision, we have achieved a 4x increase in memory access efficiency by reducing the bit width. As shown in Figure 3, the architecture of our design transfers the image data from the host side to the FPGA through the PCIe interface and saves it on the on-board DDR storage.

Then process elements (PE) accesses the data through the AXI bus, and writes the intermediate result back to memory, until all data processing is completed.

The bus width of on-board memory bus width is 256 bits, and contains 2.1Mb block RAM (BRAM).

We store weight parameters and the calibration table in BRAM. Due to the limitation of memory bus width, we prune the network from 64 filters batch to 32. The data grouping in the convolution operation is shown in Figure 1.

After accessing 256-bit data from the memory, it is divided into 32 groups (filters number), each group 8 bits (quantized weight parameters). Then we use PE to calculate the multiplication and accumulation of 32 input data and 32 corresponding weight parameters. The internal structure of the PE is shown in Figure 3.

3. EXPERIMENTAL RESULTS

For evaluations, we used DIV2K dataset to train the model. We compare the performance of the test model in terms of Peak signal-to-noise ratio (PSNR), which is one of the most popular reconstruction quality metrics for image super-resolution. For image super-resolution, PSNR is defined by the maximum pixel value and the mean square error (MSE) between the images. The proposed pruning technique is implemented by PyTorch. After pruning, we retrained the model for fine-tuning. The comparison of experimental results is shown in Figure 4.

Figure 4 Comparison of PSNR of reconstructed images with different algorithms.

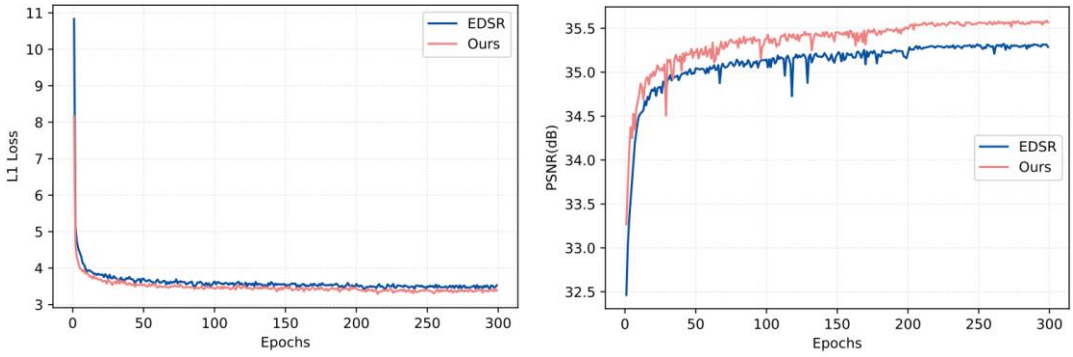
Then the pruned model is fine-tuned. The fine-tuning process is the same as the original model training stage, and 300 rounds of training are also carried out. The final experimental results are shown in Table 1.

Table 1. Image super-resolution performance comparison.

Dataset	Scale	Bicubic	VDSR[5]	A+	EDSR[5]	Ours
DIV2K	×2	31.01	33.66	32.89	35.12	**35.58**

After the pruning is completed, our model does not reduce the convergence speed. And we get a lower L1 Loss than the original model. In terms of image quality, the PSNR of the model after pruning reaches 35.58dB, which is better than the original model (35.12dB). At the same time, the model

parameters were reduced to half of the original model. The inference speed was reduced from 5.1s to 1.7s.



(a) (b)

Figure 5 Comparison results of PNSR and Loss.

4. CONCLUSION

In this work, we propose a novel design for a specific ResNet in image super resolution field, and implement a 16-layer residual neural network (ResNet) for the efficient image super-resolution in FPGA. We design a pruning method with the consideration of the memory bus width to reduce the memory access overhead. Since the memory used in our design generates 256 bits for each access, the proposed pruning algorithm drops the kernel by aligning this bit width. The experimental results show that the proposed method reduces the number of weights by 50% compared with the baseline. As a result, the inference speed can be enhanced by 3 times. One interesting and unexpected discovery in our experiment is that PSNR is also improved by 1.3% compared with the original network.

References

[1] Denil M, Shakibi B, Dinh L, Ranzato M, and Freitas N. 2013. Predicting Parameters in Deep Learning. In Advances in Neural Information Processing Systems 26, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., pp. 2148–2156.

[2] Han S, Pool J, Tran J, and Dally W J. 2015. Learning both Weights and Connections for Efficient Neural Network. In Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. pp. 1135–1143.

[3] He K, Zhang X, Ren S, and Sun J. 2016. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778.

[4] Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, and Shi W. 2017. Photo-Realistic Single Image Super Resolution Using a Generative Adversarial Network. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 105–114.

[5] Lim B, Son S, Kim H, Nah S, and Lee K. 2017. Enhanced Deep Residual Networks for Single Image Super-Resolution. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

[6] Liu Z, Li J, Shen Z, Huang G, Yan S, and Zhang C. 2017. Learning Efficient Convolutional Networks through Network Slimming. In 2017 IEEE International Conference on Computer Vision (ICCV). pp. 2755–2763

[7] He Y, Liu P, Wang Z, Hu Z and Yang Y. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4340-4349.