# FPGA Accelerates Deep Residual Learning for Image Recognition

Xuelei Li$^{1,2}$, Liangkui Ding$^{1,2}$, Li Wang$^{1,2}$, Fang Cao$^{1,2}$
1. State Key Laboratory of High-end Server & Storage Technology, Beijing, China, 100085
2. Inspur Group Co., Ltd, Beijing, China, 100085
lixuelei@inspur.com, dinglk@inspur.com, wang_libj@inspur.com, caofang@inspur.com

***Abstract*—Deep learning is greatly promoting the fast development of artificial intelligence. In the image classification area, the recognition rate and inference performance become the main challenges in the real application. Recently, residual network (ResNet) has shown a competitive accuracy and nice convergence behaviors in deep neural networks. In this paper, we are devoted to accelerate this promising framework in the inference application on FPGA (Field Programmable Gate Array) using OpenCL programming language. Firstly, a supplemented deep learning accelerator is constructed to realize the residual function in ResNet. Secondly, our construction reschedules three on-chip buffers in order to store the feature data and to stream it to processor elements alternately. In addition, we also implement data parallel and pipeline execution such that the filter parameters can be synchronously processed with the image data on FPGA. Moreover, we exploit a convertor to transform any ResNet in CAFFE framework into FPGA platform. It can generate the FPGA head files using its original prototxt files through the dictionary function of Python. The experiment result analysis shows that our acceleration has a competitive performance while maintaining the high accuracy rate. Finally, we provide a solution to accelerate any construction of ResNet using OpenCL programming language on FPGA.**

***Keywords*—deep learning; residual network; FPGA, OpenCL**

## I. Introduction

Artificial intelligence has brought dramatic changes in our life through deep learning. In image classification area, deep learning has led to a series of breakthroughs [1–3] since Lecun et al. [4] successfully applied backpropagation learning to the recognition of handwritten zip code digits. As a promising method, deep convolutional neural network attracts teams of researchers to improve not just accuracy rating, but also system performance.

### *A. Related Works*

On the accuracy rating improvement side, the contributions [1–3] are impressive in the image classification competition. In 2010, Krizhevsky et al. trained a large, deep convolutional neural network (named AlexNet) in the ILSVRC-2010 (ImageNet Large-Scale Visual Recognition Challenge 2010) contest. Furthermore, they also entered a variant of this model in the ILSVRC-2012 competition. After that in 2014, Szegedy et al. proposed another deep convolutional neural network architecture (named GoogLeNet) in the ILSVRC-2014. The main hallmark of this architecture is the improved utilization of the computing resources inside the network. In addition, they increased the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions are based on the Hebbian principle and the intuition of multi-scale processing. Later in 2015, He et al. [3] presented a residual network (named ResNet) learning framework in ILSVRC-2015. It helps to ease the training of networks that are substantially deeper than those used previously. They also explicitly reformulated the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Furthermore, they provided comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth.

On the system performance (inference application) improvement side, GPUs, TPUs (Tensor Processing Units) and FPGAs (Field Programmable Gate Arrays) are the prevalent solutions in deep learning applications. More recently, FPGAs have shown promise in efficiently implementing CNNs [5–9]. In order to adapt the deep learning reference applications, FPGA plays an important role to balance power consumption and recognition efficiency. Unfortunately, most recent efforts to run CNNs on FPGAs have shown limited advantages. The vast majority of FPGA implementations of CNNs have only implemented the convolutional layers limiting the benefit of the approach since other layers may quickly become the bottleneck of the neural net [5]. Although some contributions have been achieved to implement all the layers on the FPGA [5, 7, 8], however, when compared with other publicly known implementations for GPU [10, 11], FPGA performance has fallen short significantly. Recently, Aydonat et al. [12] pointed that previous approaches on FPGAs have often been memory bound due to the limited external memory bandwidth on the FPGA device. Additionally, they also indicated that previous FPGA architectures for CNNs have not been able to take advantage of the peak operations of the device leading to low performance. In order to overcome these difficulties, Aydonat et al. [12] presented a novel architecture written in OpenCL, which they referred to as a Deep Learning Accelerator (DLA), that maximizes data reuse and minimizes external memory bandwidth. Furthermore, they showed how they can use the Winograd transform to significantly boost the performance of the FPGA. As a result, they can achieve a performance of 1020img/s, or 23img/s/W when running their DLA with the AlexNet CNN benchmark on Intel's Arria 10 device. That comes to 1382 GFLOPs and is 10x faster with 8.4x more GFLOPS and 5.8x better efficiency than the state-of-the-art on FPGAs. Additionally, 23 img/s/W is competitive against the best publicly known implementation of AlexNet on nVidia's TitanX GPU.

### B. *Problems and Contributions*

As similar with Aydonat et al.'s [12] contributions to the acceleration on AlexNet, the acceleration of residual network should be implemented to achieve the state-of-the-art accuracy and performance. However, the differences between AlexNet and ResNet lead to a bad situation when reconstructing ResNet with DLA. In particular, the computation of residual function is a new component which should be defined as a novel DLA primitive for FPGA acceleration. The main obstacles are data dependency and buffer stream that affect performance. In addition, the norm layer, fully-connected layer and data transfer also impede the performance of our construction. These operations should be reconstructed in FPGA platform. As a consequence, there exit a serious of challenges to achieve the acceleration of ResNet on FPGA using OpenCL.

In order to address the above challenges, we realize the acceleration of ResNet on FPGA according to Aydonat et al.'s DLA architecture. The detailed contributions are provided as follows:

1) The residual function is constructed as a novel DLA primitive to supplement ResNet acceleration on FPFA platform.
2) Stream buffers are rescheduled to solve the problems of data parallel and pipeline.
3) A convertor is exploited to transform any form of ResNet in CAFFE framework into FPGA acceleration environment through Python.

### C. *Organizations*

The rest of this paper is organized as follows. Section II introduces the related preliminaries in our implementation. Our design methods are described in Section III. Section IV analyze the performance of our implement. Finally, Section V concludes this paper.

## II. Preliminary

This section introduces the basic concepts about residual network and FPGA heterogeneous computing platform.

### A. *Residual Network*

According to He et al.'s contributions [3], deep residual learning is introduced to address the degradation problem. Formally, denoting the desired underlying mapping as $H(x)$, let the stacked nonlinear layers fit another mapping of $F(x) := H(x) - x$. As shown in Fig. 1, the original mapping is recast into $F(x) + x$, i.e.,

$$H(x) := F(x) + x.$$

Note that, this equation is assumed that the dimensions of $x$ and $F(x)$ are the same. In addition, Fig. 1 also defines a

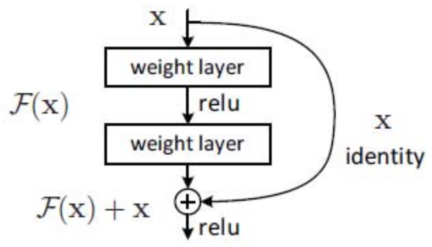

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 1. Residual Learning Building Block.

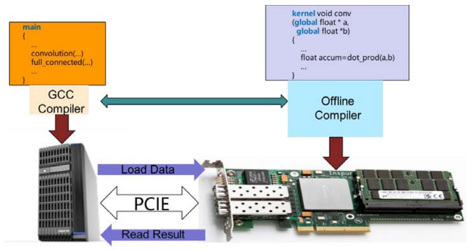
building block of identity mapping by shortcuts to learn every few stacked layers. Formally, a building block is defined as:

$$y = F(x, \{W_i\}) + x,$$

where $x$ and $y$ are the input and output vectors of the layers considered. The dimensions of $x$ and $F$ must be the same. If this is not the case, a linear projection $W_s$ should be performed by the shortcut connections to match the dimensions:

$$y = F(x, \{W_i\}) + W_s x.$$

### B. *Heterogeneous Computing*



```
main
{
   ...
   convolution(...)
   full_connected(...)
   ...
}
```

```
kernel void conv
(global float * a,
  global float *b)
{
   ...
   float accum=dot_prod(a,b)
   ...
}
```

Fig. 2. FPGA Heterogeneous Computing Platform.

FPGA heterogeneous computing platform consists of host and device as shown in Fig. 2. Our platform allows users to program FPGAs with OpenCL that is an open parallel programming language. Such platform uses a master-slave model. The master host is used to control all memory transfers. A user is required to write a host program which calls a predefined OpenCL API to control the accelerator device. The slave device is used to execute all the kernel programs in order to accelerate the data processing. The user is required to write OpenCL kernel functions that are compiled to the accelerator. When the program starts, the master host transfers data to slave device. After receiving the execution command with the pre-defined data, the slave device begins to process the data according to the compiled kernel program. The master host will read the results from the device DDR if the slave device finishes the program. The data are transferred through PCIE between master host and slave device.

## III. FPGA Accelerates ResNet

In this section we described our solution for accelerating residual networks using OpenCL on FPGA.

## A. Design Goals

The design goals of our contribution are described as follows:

- Construct an improved DLA architecture (with a new residual learning components) to accelerate ResNet on FPGA.
- Achieve a high-performance method to adapt inference application of ResNet.
- Provide a convenient solution to convert any form of ResNet under CAFFE framework into FPGA acceleration environment.

## B. Overall Architecture

Our DLA architecture is an improvement of Aydonat et al.'s [12] contributions. There exits an additional primitive to handle with the Eltwise module which corresponds to the residual learning function. The Eltwise is implemented by executing code shown in Fig. 3.

```
data_value_t cached_input = READ_FROM_INPUT_CACHE(cache_write_addr, wvec, c);
float cached_input1 = data_value_to_float(cached_input);
float cache_write_data1 = data_value_to_float(cache_write_data);
cache_write_data1 +=cached_input1;
cache_write_data = float_to_data_value(cache_write_data1);
```

Fig. 3. Eltwise.

In addition, the differences between ResNet and AlexNet leads to several changes that are effect the execution of the kernel function. Fortunately, our improvement keep a high performance and remain the accuracy rating.

## C. Head Files

In this research work, we need a transformation that can achieve the parameter conversion between CAFFE deep learning configuration and OpenCL accelerating environment. This transformation needs lots of work to convert the parameters of CAFFE model into C++ and OpenCL type. Fortunately, as shown in Fig. 4, we invent a tool that achieves auto-convert the CAFFE prototxt into C++ head files. These template files consists the parameters that decide the network scale. This tool is based upon the dictionary function of Python. It generates the template files from CAFFE prototxt to C++ and OpenCL head file. Therefore, we can use FPGA and OpenCL platform to accelerate any form of residual network implemented in CAFFE framework.

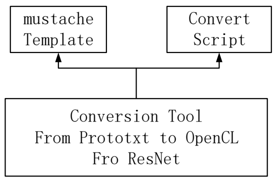


Fig. 4. Conversion Tool.

TABLE I
INSPUR F10A ACCELERATION BOARD FEATURES

| Feature | Parameter |
|---|---|
| Board Format | ○ Half-Length, low profile x8 PCIe form factor |
| Host I/F | ○ PCI Express®Gen3 x8<br>○ 2 SFP+10GbE |
| FPGA Device | ○ Arria 10 GX115,10AX115H3F34E2SG |
| On Board Memory | ○ Dual 8GB DDR4@2133MT/S SODIMM<br>○ Device: Micron MTA8ATF1G64HZ-2G3B1 |
| On Board Flash | ○ 1Gbit of Flash memory for booting FPGA<br>○ QSPI Flash Device: MT25QU01GBBB8E12-0SIT |
| On Board CPLD | ○ Device: 10M02SCU169C8G<br>○ Temperature and Fan monitoring<br>○ SMBUS bus access<br>○ Clock Configuration |

## D. Stream Buffers

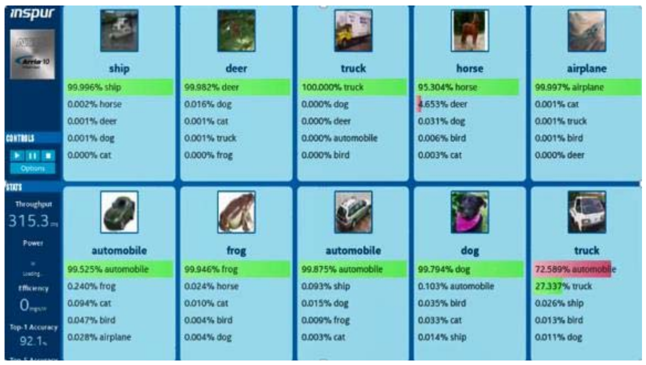
On-chip buffers are expensive hardware resources. According to Aydonat et al.'s [12] stream buffer design, we also constructed three buffers to read and write in a stream manner. The buffer size is computed as the maximize space. In addition, we change the filter load manner in order to adapt the data process which should be pipelined and paralleled. The details are processed as follows in Fig. 5.

```
const int para_fpga_size = NUM_CONVOLUTIONS * MAX_BIAS_SIZE*4;
para = (data_value_t*)acl_aligned_malloc( sizeof(data_value_t) * para_fpga_size );
copy_from_para_raw();
float para_float;
for(int i=0; i<NUM_CONVOLUTIONS * MAX_BIAS_SIZE*4;i+=4){
    para_float=data_value_to_float(para[i]);
}
```

Fig. 5. Filter Load.

## IV. PERFORMANCE ANALYSIS

This section analyzes the performance of our improvement. The experiment is executed on Inspur NF5540M3 host server with E5-2407@2.2GHz CPU, 128GB DDR3 memory and CentOS 7.1 operation system. In addition, the kernel program is performed on Inspur F10A FPGA acceleration board. The acceleration board features are described in Table I.



```
ship
99.996% ship
0.002% horse
0.001% deer
0.001% dog
0.000% cat

deer
99.982% deer
0.016% dog
0.001% cat
0.001% truck
0.000% frog

truck
100.000% truck
0.000% dog
0.000% deer
0.000% automobile
0.000% bird

horse
95.304% horse
4.653% deer
0.031% dog
0.006% bird
0.003% cat

airplane
99.997% airplane
0.001% cat
0.001% truck
0.001% bird
0.000% deer

automobile
99.525% automobile
0.240% frog
0.094% cat
0.047% bird
0.028% airplane

frog
99.946% frog
0.024% horse
0.010% cat
0.004% bird
0.004% dog

automobile
99.875% automobile
0.093% ship
0.015% dog
0.009% frog
0.003% cat

dog
99.794% dog
0.103% automobile
0.035% bird
0.033% cat
0.014% ship

truck
72.589% automobile
27.337% truck
0.026% ship
0.013% bird
0.011% dog
```

Fig. 6. FPGA Accelerate Residual Network.

Our verification is tested on CIFAR-10 data set that every picture is 32*32 pixels. The experiment is tested on the

9-layers residual network that defined $bach\_size = 10$, $k\_vector = 24$. The test results are shown in Fig. 6. The result of our experiment is shown as: throughput 315.3fps, Top-1 accuracy 92.3% and Top-5 accuracy 99.6%.

The results presents that our solution is effective and has a competitive performance while maintaining the same accuracy rate.

## V. Conclusion

In this paper, we provide a solution to accelerate residual network for the inference application of image recognition. It helps to construct a heterogeneous computing platform using OpenCL programming language on FPGA to achieve the accelerating of deep learning. In addition, the performance analysis shows that our solution is suitable for the inference application of image recognition. Moreover, the performance can be further improved in our future research. Finally, we also focus on studying the training applications using FPGA in deep learning area.

## References

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. International Conference on Neural Information Processing Systems (Vol.25, pp.1097-1105). Curran Associates Inc.

[2] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Vol. 07-12-June-2015, pp. 1C9). IEEE Computer Society. https://doi.org/10.1109/CVPR.2015.7298594

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

[4] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

[5] Zhang, C., Fang, Z., Zhou, P., Pan, P., & Cong, J. (2016, November). Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. In Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on (pp. 1-8). IEEE.

[6] Peemen, M., Setio, A. A. A., Mesman, B., & Corporaal, H. (2013). Memory-centric accelerator design for Convolutional Neural Networks. IEEE, International Conference on Computer Design (pp.13-19). IEEE.

[7] Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., & Zhou, E., et al. (2016). Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. Acm/sigda International Symposium on Field-Programmable Gate Arrays (pp.26-35). ACM.

[8] Suda, N., Chandra, V., Dasika, G., Mohanty, A., Ma, Y., Vrudhula, S., Seo J.-s., & Cao, Y. (2016, February). Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 16-25). ACM.

[9] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. (2015, February). Optimizing fpga-based accelerator design for deep convolutional neural networks. In Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 161-170). ACM.

[10] S. Chintala. convnet-benchmarks, 2016.

[11] nVidia. GPU-Based Deep Learning Inference: A Performance and Power Analysis, November 2015.

[12] Aydonat, U., O'Connell, S., Capalija, D., Ling, A. C., & Chiu, G. R. (2017). An OpenCL (TM) Deep Learning Accelerator on Arria 10. FPGA'17, February 22-24, 2017, Monterey, CA, USA.