An Analysis of Operations Modification in Deep Neural Network in Hardware Perspective

B03901026 許凱傑, B03901101 楊其昇, B03901160 楊仲萱

***Abstract*—In this report, we reviewed many state-of-art very deep neural networks, including AlexNet, VGG net, Inception, ResNet and Xception. In addition to the classification accuracy, we conduct many aspects of tests on hardware resource consumption, namely, number of parameters, computation complexity and power consumption. We analyze the advantages and trade-off of these deep neural networks in the view of hardware implementation and provide some insight of hardware-friendly designs.**

I. Introduction

Since the great success of AlexNet in 2012 ImageNet challenge, it has infused a Deep Neural Network (DNN) boom on every aspects, including the field of image classification. The ultimate goal of ImageNet competition is aimed to get the highest accuracy on large scale multi-class image classification problem framework. However, they only pursue optimal classification accuracy and neglect problems encountered in practical applications such as memory, computation complexity and power consumption.

First, it is a common practice to run several trained instances of a given model over multiple-crops of the validation images. This practice is called "ensemble" technique to enhance the performance of the machine learning model, but would drastically increased the amount of required computation resources and inference time to achieve the reported accuracy. Furthermore, to achieve higher accuracy on large-scale image classification, the model architecture tends to be deeper and larger which also implies the increase of required computation resources. Fortunately, there were many interesting and creative model architectures such as ResNet module and Inception module proposed to increase the accuracy with less parameters and shorter inference time. Besides, as the great breakthrough on image classification brought by DNN, there are more and more discussions about how to customize the efficient hardware implementation on various applications given the model architecture. Thus, it is necessary to give an overall analysis including accuracy and the relevant hardware costs of the model architecture.

In this report, we aimed at comparing state-of-the-art DNN model architecture submitted for ImageNet Challenge over the pass 5 years in terms of the accuracy and hardware resources requirement. We compared and analyzed these models on multiple metrics related to resource utilization in actual deployments: accuracy, the number of parameters, the number of operations, inference time per image, and power consumption. The purpose of this report is to stress the importance of these metrics, which are essential for the practical implementation of the related customized hardware accelerator in the future. This report is organized as follows. The model architecture is reviewed at section II. Simulation results and analysis are presented in section III. Hardware-friendly designs are in section IV and conclusion is drawn in section V.

II. Deep Learning Models

A. AlexNet

AlexNet was proposed by Alex Krizhevsky et al.[1] on the 2012 ImageNet challenge. The great success of AlexNet on image classification have grabbed the public's attention on Convolutional Neural Network (CNN). They modified the existed CNN architecture, and have a "deeper" architecture in Fig. 1. Besides, several techniques used on AlexNet like dropout and ReLu activation function have been popular tricks in modern training and architecture of neural networks.

B. VGG

Seeing the success of AlexNet, people started to find the way to increase the accuracy of classifying images. One structure was proposed by Karen Simonyan and Andrew Zisserman, which is VGG network [2]. VGG is an acronym for Visual Geometry Group which is their group name. They adopted the simplest method, using 3x3 convolution layers and 2x2 max pooling layers to make the network deeper. Usually the number following VGG stands for the umber of its layer.

Fig. 2. shows the basic structure of VGG-16 and we could find out that just increasing the layers results a better performance in classification. The result at a single test scale is in Fig 3. A has 11 weight layers, A-LRN also has 11 weight layers but has local response normalisation, B has 13 weight layers, C has 16 weight layers but replace three 3x3 convolution layer with 1x1 layers, D is VGG-16 and E is VGG-19 with 19 weight layers. It's clear that the error rate decreases with the number of layers increasing.

C. Inception I - III

1) Inception I: Although the depth of the network is important, number of parameters and unexpected computational resources are the difficulty of deep neural network. In order to solve this problem, GoogLenet [3] was proposed

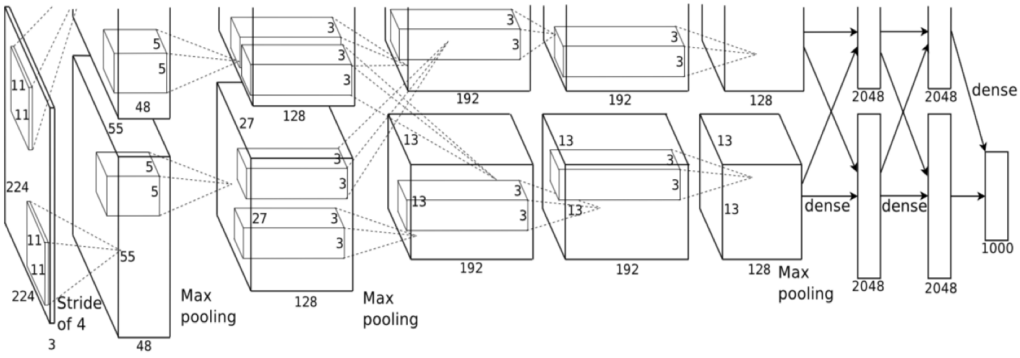

Fig. 1: AlexNet model architecture [1].

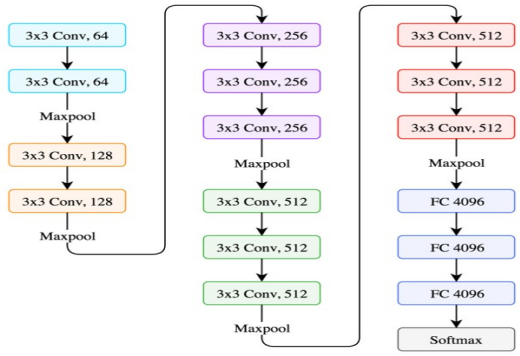

Fig. 2: VGG-16 model architecture

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

Fig. 3: ConvNet performance at a single test scale [2].

in ImageNet Large-Scale Visual Recognition Challenge 2014, having the highest top-5 accuracy. The most important concept of GoogLeNet is the inception module, causing GooLenet to be called "network in network". The final form of inception module is in Fig. 4. Basically it acts as multiple convolution filter outputs, that are processed on the same input. Every inception module does pooling at the same time, since pooling has been essential for the success of current CNNs. All the outputs are then concatenated using 1x1 convolution, allowing the model to take advantage of multi-level feature extraction from each input. The 1x1 convolutions before 3x3 and 5x5 convolutions and after 3x3 max pooling are used to reduce the dimension of the outputs. Beside being used as reductions, they also include the use of rectified non-linear activation making them dual-purpose.

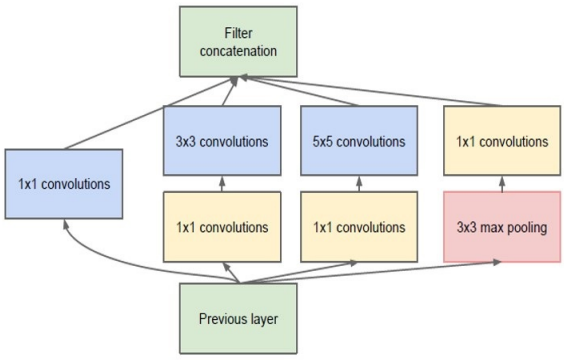

Fig. 4: Inception modle with dimensionality reduction [3].

Finally, by stacking some inception modules, we could get the architecture of GoogLeNet as Fig. 5. Total number of layers is 22. In the 2014 ImageNet challenge, GoogLeNet got the first prize, showing that the modification of network is also important for increasing the accuracy.

2) Inception II: For reducing the needs for layers to continuously adapt to the new distribution, the inception version 2 was proposed [4]. The difference between version 1 and version 2 is that inception v2 added batch normalization. Normalizing parameters of each layer helps training be faster and has better performance. Fig. 6. shows the effect of using batch normalization. It's obvious that model with inception v2 would achieve the same accuracy with fewer training steps.

3) Inception III: Since the gains of inception v1 arose from the use of dimension reduction, it comes up with inception v3 [5]. finding ways of factorizing convolutions. First, it's the factorization into smaller convolution. For example, one 5x5 convolution can be changed to two 3x3 convolutions. By doing so, parameters decrease and so does the computational cost. Fig. 7. shows how to replace 5x5 convolution with 3x3 convolution.

The other way is called spatial factorization into asymmetric convolutions. One can replace any nxn convolution by a 1xn convolution followed by a nx1 convolution and the computational cost saving will increase dramatically

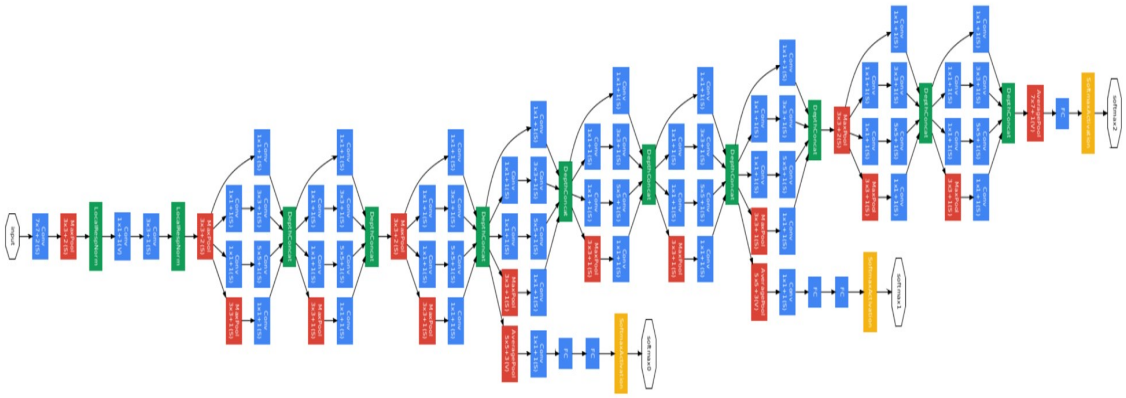

Fig. 5: The architecture of GoogLenet [3].

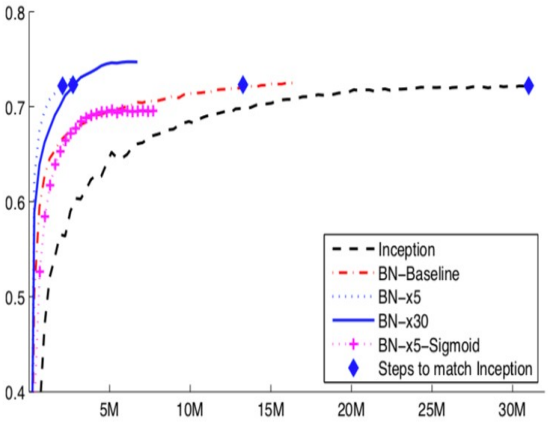

Fig. 6: Performance of inception v2 [4].

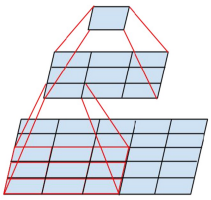

Fig. 7: Mini-network replacing the 5x5 convolutions [5].

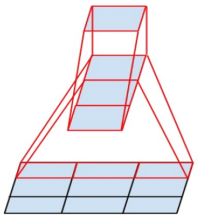

Fig. 8: Mini-network replacing the 3x3 convolutions [5].

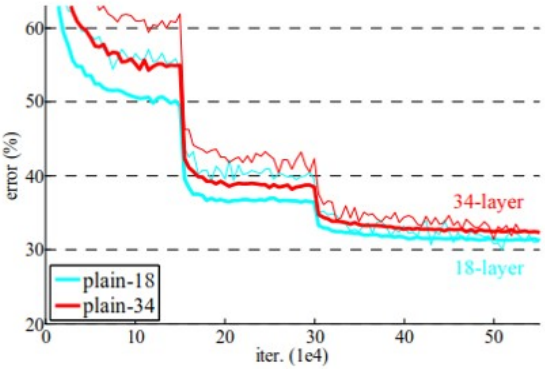

Fig. 9: Training process of plain network (without residual connection) [6].

as n grows. Fig. 8. demonstrates how to use 3x1 convolution followed by a 1x3 convolution to replace one 3x3 convolution. The above two methods are the main change which inception v3 adopted, making the module have less parameters and more hardware-friendly.

D. ResNet

Because of the success of VGG net, one may think directly adding the depth of the model results in better performance. However, Fig. 9. shows that the error increases compared to shallower model, which suggests a non-overfitting problem.

Therefore, we need to modify model structure to ensure better performance. In [6], an identity mapping structure was proposed and shown in Fig. 10. After adding identity mapping, all the model have to learn becomes a small $\mathcal{F}(x)$ instead of $\mathcal{F}(x)+x$. Fig. 11. shows the training process of model with identity mapping. It is observed ResNet-34 tenders lower error than ResNet-18 and converges faster than plain-34 (in Fig. 9.). In other words, the residual network is easier to optimize and the depth of model really improves the performance.

E. Xception

It is observed inception module in Fig. 4. first uses a 1x1 convolution to map cross-channel correlations and then separately maps the spatial correlations in every flow. We

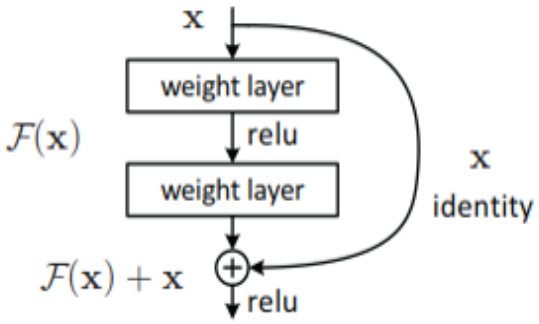

Fig. 10: Residual learning block [6].

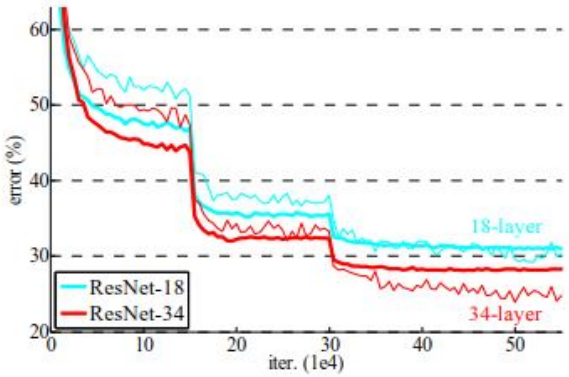

Fig. 11: Training process of plain network (with residual connection) [6].

can consider an extreme version of inception module which is shown in Fig. 12., namely, each output channel of 1x1 convolution goes to a spatial convolution.

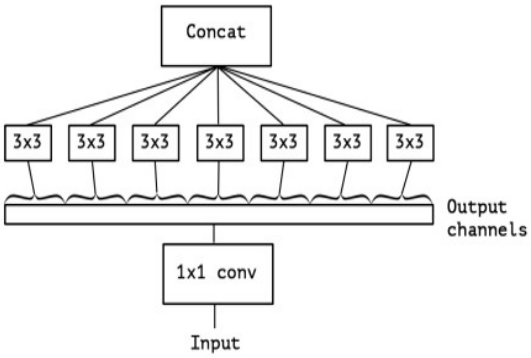

Fig. 12: Extreme version of inception module [7].

Motivated by this, the author proposed a new structure to reverse the order of 1x1 convolution and spatial convolution shown in Fig. 13 [7] . In addition to the different operational order, the Xception module doesn't have non-linear activation function such as ReLU after spatial convolution, which will deteriorate the validation accuracy.

Furthermore, the author adopted residual learning to accelerate training process. Fig. 14. shows a middle flow of Xception architecture and Fig. 15. shows the effect of residual learning.

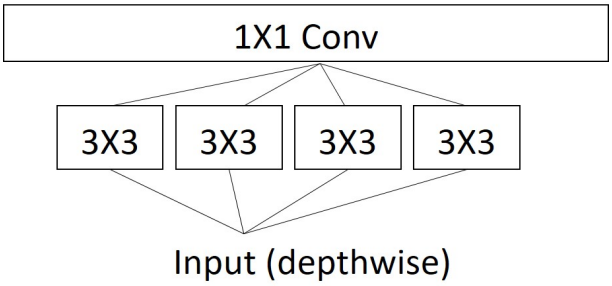

Fig. 13: Xception module.

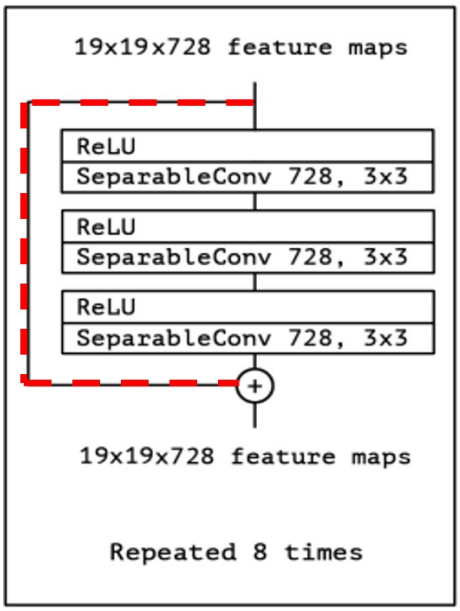

Fig. 14: Middle flow of Xception arthitecture [7].

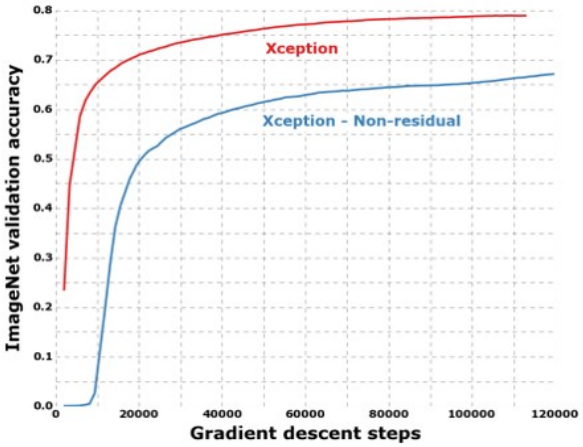

Fig. 15: Training process with and without residual connections [7].

III. Simulation Results

In this section, we report our results and comparisons. We analyzed following models: AlexNet, VGG16, VGG19, ResNet50, ResNet152, Inception v3, and Xception. Some of them are viewd as states-of-art models in ImageNet challenge. Similar to what [8] have done, we have measured accuracy, the number of parameters, the number of floating point operations (FLOPS), inference time per image, and power of each model. We will first prevent our experiment setup and then our simulation results.

A. Experiment Setup

Our testing data is the validation data of ILSVRC 2012 provided by ImageNet. There are 50000 images and 1000 classes in the database. The dimensions (width and length) of each image size range from 200 to 600. The required memory of database is about 6.8 GB.

We used Python3 and the related machine learning packages including `Tensorflow 1.4` and `Keras 2.0` to do our experiment on software. On hardware, we utilized `2 GHz Intel Core i5 CPU` with 2 cores to measure the inference time per image on each architecture mentioned above, and utilized the command provided by `Nvidia Cuda` on the terminal: `nvidia-smi -i 0 --query-gpu=index,timestamp,power.draw,clocks.sm, clocks.mem,clocks.gr --format=csv -l 1` to monitor the power consumption on `Nvidia GeForce GTX 1080 GPU` when the neural network is doning inferences.

B. Analysis

1) Accuracy: Fig. 16 and Fig. 17 showed one-crop accuracies of each model. We could find that the performance on accuracy from AlexNet to Xception model has a great improvement and that the performance of recent models like Inception v3 and Xception have slight difference between them which implies that the performance of accuracy has been gradually having less improvement.

Furthermore, we could dispel a myth of blindly "deeper" model architecture from both Fig.16 and Fig.17. It is obvious that VGG16 & VGG19, and ResNet50 & ResNet152 have similar model architecture but the number of layers, and that the deeper versions of both model architectures only have very limited improvements on accuracy which are far less than the improvements resulted from the modification of model architecture.

2) The number of parameters: To have better performance on accuracy, it is inevitable to have more parameters to enhance the learning capacity of the model. However, it would result in the high cost of large amount of memory to record the numerous parameters Thus, it is an critical issue to have the less parameters while still maintaining the high accuracy on image classification. Fig.18. shows that ResNet, Inception v3 and Xception have less parameters than traditional deep CNN such as AlexNet or VGG and even have better performance on image classification. It is verified that the special modifications like Inception module and Residual module indeed have higher efficiency of parameter utilization.

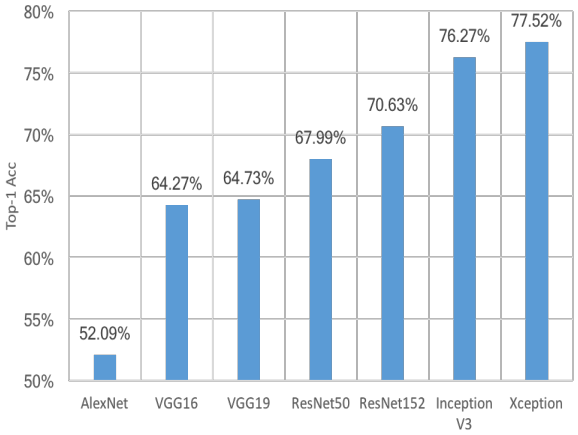

Fig. 16: Top-1 accuracy of each model.

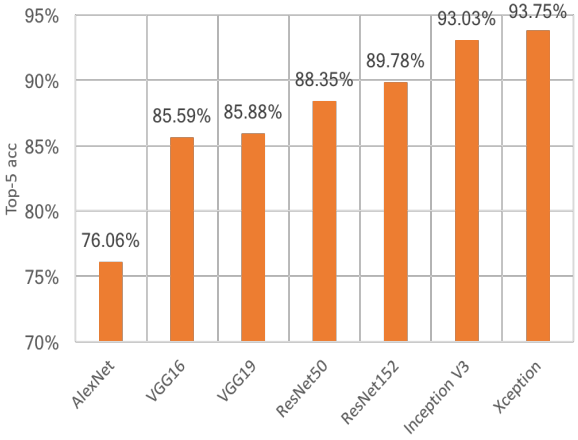

Fig. 17: Top-5 accuracy of each model.

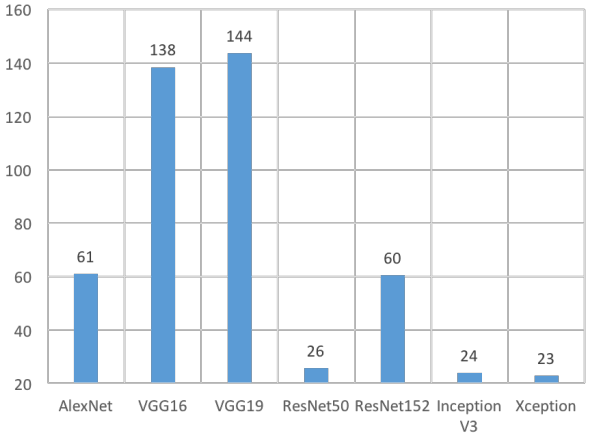

Fig. 18: The total number of parameters on each architecture.

3) The number of operations: Similarly, in order to have better performance on image classification, it is unavoidable to have higher number of operations. In our experiment, we define the number of operations as the total number of addition and multiplication conducted when doing an inference of one image. Operations count is essential for establishing a rough estimate of inference time and the size of hardware circuit in case of customized implementation of neural network accelerator. Fig.19 gives the result similar to that in Fig.18. Again, it shows that special modifications of model architecture have great benefits including less usage of parameters, less utilization of operations and higher performance on accuracy.

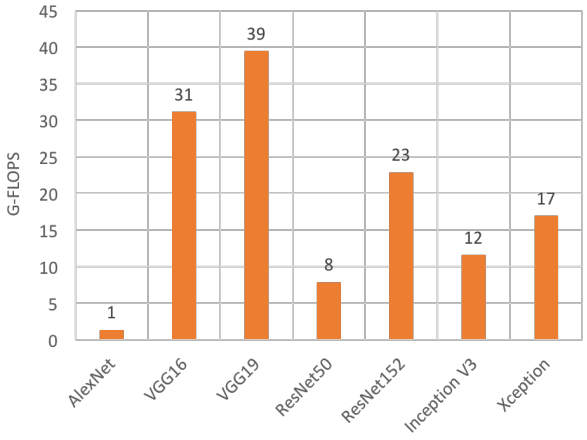

Fig. 19: The total number of operations among each model.

4) Inference time: Fig.20. reports the inference time per image on each architecture. Fig.21. demonstrates the high-positive correlation between the inference time and the number of operations. As we mentioned above, the number of operations and the number of parameters are critical to the cost of the hardware implementation of the customized neural network accelerator. Fig. 21. shows that AlexNet has the least inference time due to the least operations count among other model, and the recent model architectures like ResNet50, Inception v3 and Xception have less number of parameters and the acceptable inference time and operations count. To have better evaluation among all architectures, Fig. 22. shows the relationship among accuracy, inference time per image and the number of parameters. When considering the trade-off between the performance (accuracy and inference time) and the cost (the number of parameters and operations count) of each model, we could find that ResNet50, Inception V3, and Xception are the most competitive models among others.

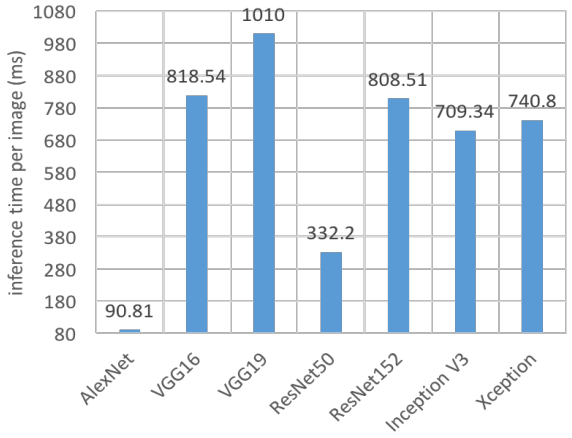

Fig. 20: Inference time on each architecture.

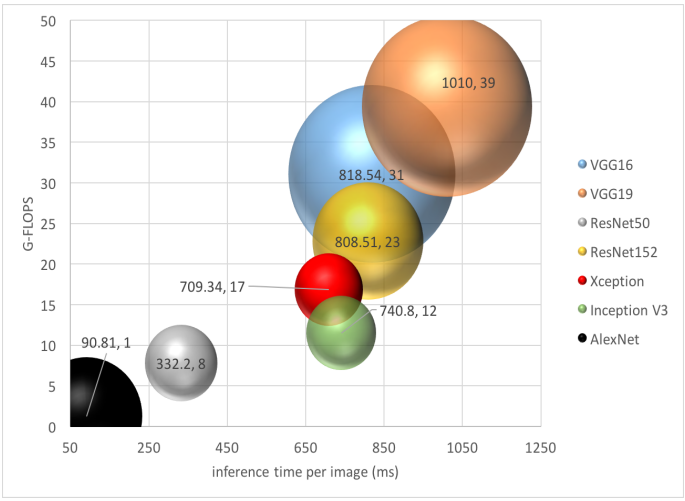

Fig. 21: Inference time per image v.s. Operations $\propto$ the number of parameters

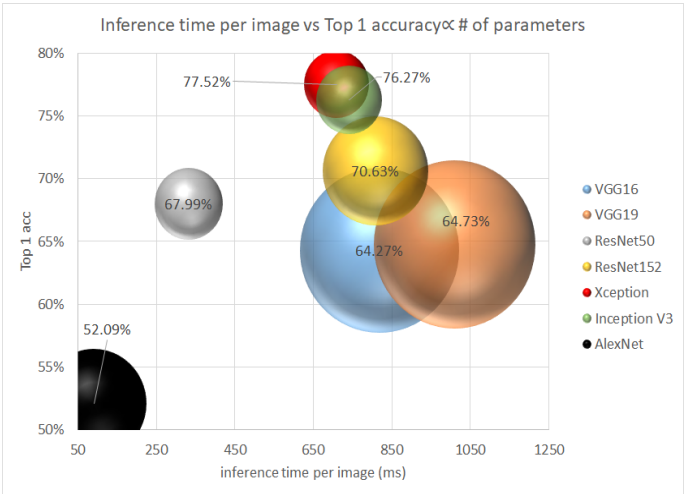

Fig. 22: Inference time per image v.s. Top 1 accuracy $\propto$ the number of parameters

5) Power: Fig. 23. reports the power measurements. The power consumption of `Nvidia Geforce GTX 1080`

GPU in idle state is about 17W. Fig.24. shows that power consumption is positive correlated to the number of operations and the number of parameters. Besides, we could also find that Xception and ResNet50 outperformed other architectures with respect to power consumption and the number of parameters.

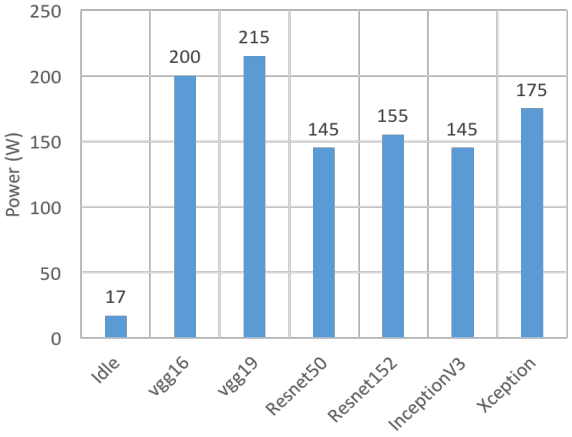

Fig. 23: Power consumption of each model

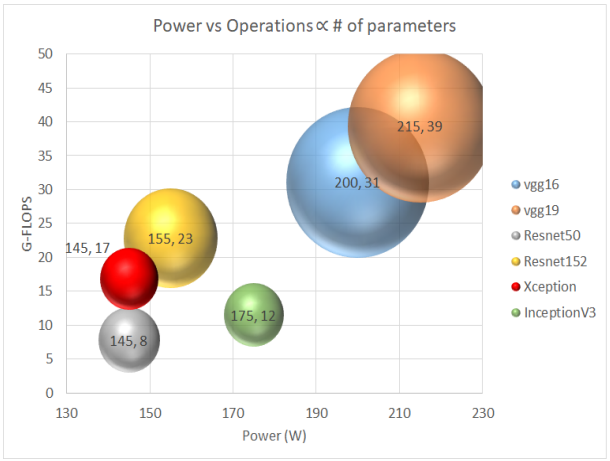

Fig. 24: Power v.s. Inference time per image $\propto$ the number of parameters

All the simulation results are summarized in Table. I. which suggests Xception and Inception have the best classification accuracy and ResNet50 has the best hardware resource consumption.

IV. Hardware-Friendly Design

Since there are a lot of matrix-matrix and matrix-vector operations in CNNs, these operations dominate the computational cost of inference. Therefore, systolic array is reused in CNN, making the calculation more efficient. Fig. 25. shows the concept of systolic array. We want to calculate matrix A multiplied with matrix B, so we let the parameter get into the small module sequentially. Every small module just do the same thing

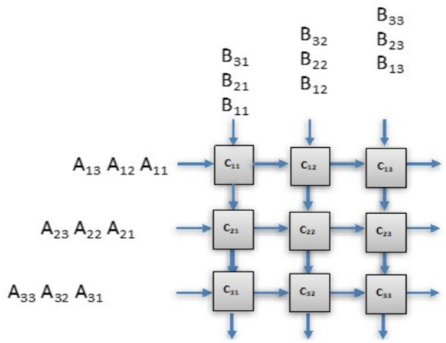

Fig. 25: Basic concept of systolic array

which is multiplication, saving the answer in it and pass the parameter to the next module.

Quantization is also one method to reduce hardware resources and also the computation time. Mainly, they just change the floating point design to the fixed-point design. Not only the design will be simpler but also the amount of memory that has to be used decreases. Accuracy will be affected, but recent research has shown that by retraining the network the decrease of accuracy can be made acceptably small.

Pruning is to remove the weight and neuron values which are close to zero or by one own decision rule to determine which value can be pruned. This can avoid unnecessary computing and reduce the number of the parameters.

Memory access is a critical problem in DNNs. Fig. 26. shows a multiply-and-accumulate (MAC) operation. It is observed for each MAC operation, there will be 4 memory access. Consider AlexNet having about 724M MACs and it will have about 2.9G times memory access, which is a non-tolerable hardware cost.

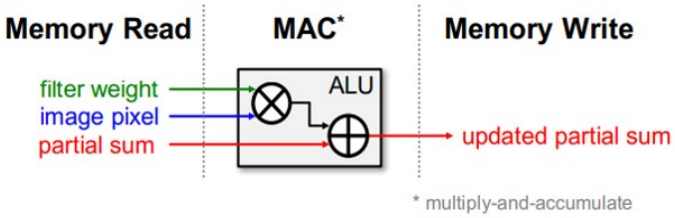

Fig. 26: Memory access of a MAC operation [10].

The author in [10] suggests three kinds of data reuse, which is shown in Fig. 27.

- Convolutional Reuse : Unlike fully-connected layer where each edge between two nodes are different, convolutional layer using same filter to form sliding window. Therefore, about E^2 parameter can be reduced, where E is the size of the output feature map.
- Image Reuse : After we load an input feature map, we will use every filter to conduct convolution. In other words, we can reduce the memory access of input feature map by M, where M is the number of filters.

Model	Top-1 Acc	Top-5 Acc	Number of parameters (M)	Power (W)	GFLOPs	Inference Time (ms)
AlexNet	52.09%	76.06%	60.96	- - -	1.27	90.81
VGG16	64.27%	85.59%	138.36	200	31.06	818.54
VGG19	64.73%	85.88%	143.67	215	39.40	1010
ResNet50	67.99%	88.35%	25.64	145	7.80	332.2
ResNet152	70.63%	89.78%	60.50	155	22.83	808.51
Inception III	76.27%	93.75%	23.85	175	11.51	740.8
Xception	77.52%	93.03%	22.91	145	16.87	709.34

TABLE I: Summary of simulation results. The best results are shown in red.

- Filter Reuse : If we consider to do a batch of N input feature maps, we can reuse filter weights to do convolution of all N input feature maps. Therefore, we can reduce memory access to filter weights by N.
- Partial Sum Reuse : After every convolution, we need to update partial sum and needs a memory read and a memory write. If we store partial sums nearby, we can reduce a lot of memory access.

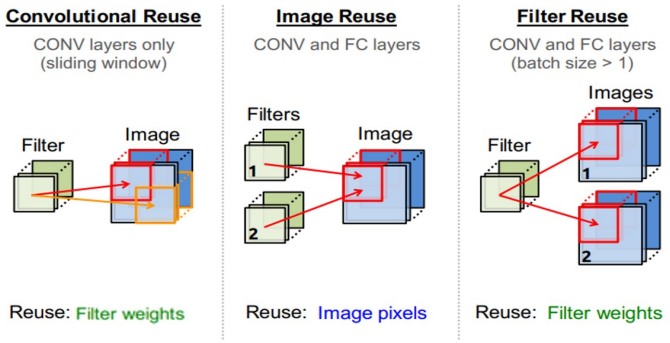

Fig. 27: Three kinds of data reuse [10].

V. Conclusion

Table. I. shows ResNet, Inception v3 and Xception are among the best models in the trade-off of classification accuracy and hardware resources consumption. In order to know the trade-off more specifically, we depict Fig. 28. and Fig. 29. If the accuracy of about 70% is acceptable, ResNet50 will be the optimal choice. However, if accuracy is strictly-required, Xception and Inception v3 are both reasonable choices with Xception slightly performs better in power consumption and inference time. Fig. 4. and Fig. 13. show the basic module of Inception v3 and Xception, respectively. Since the module of Xception is simpler, the control signal, pipeline stage and special ALU is easier to design, suggesting Xception is easier to implement.

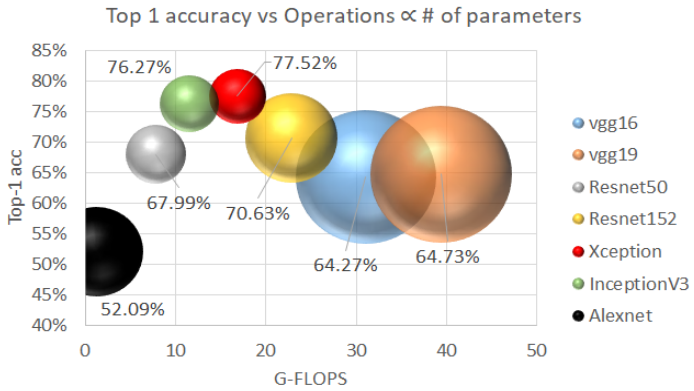

Fig. 28: Top-1 accuracy v.s. Operations ∝ the number of parameters.

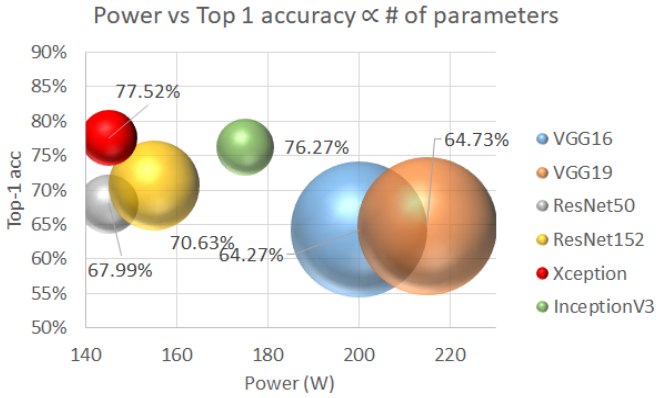

Fig. 29: Top-1 accuracy v.s. Power ∝ the number of parameters.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", 2012

[2] K. Simonyan, and A. Zisserman, "Very Deep Convolution Networks For Large-Scale Image Recognition," 2015

[3] C. Szegedy, W. Liu, Y. Jia et al, "Going deeper with convolutions," IEEE conf. Computer Vision and Pattern Recognition, 2015

[4] S. Loffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", 2015

[5] S. Ioffe, V. Vanhoucke, C. Szegedy et al., "Rethinking the Inception Architecture for Computer Vision," 2015

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2015

[7] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017

[8] A. Canziani, E. Culurciello and A. Paszke, "An Analysis of Deep Neural Network Models For Practical Applications," 2017

[9] Y. J. Lin and T. S. Chang, "Data and Hardware Efficient Design for Convolutional Neural Network," IEEE Trans. Circuits and Systems I: Regular Papers, 2017, pp. 1-10

[10] Y. H. Chen, J. Emer and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016, pp. 367-379

[11] K. Kiningham, M. Graczyk and A. Ramkumar, "Design and Analysis of a Hardware CNN Accelerator"