MODIFICATION OF OPERATIONS IN CNN

FINAL PROJECT FINAL REPORT

第一組

許凱傑 B03901026
楊仲萱 B03901160
楊其昇 B03901101

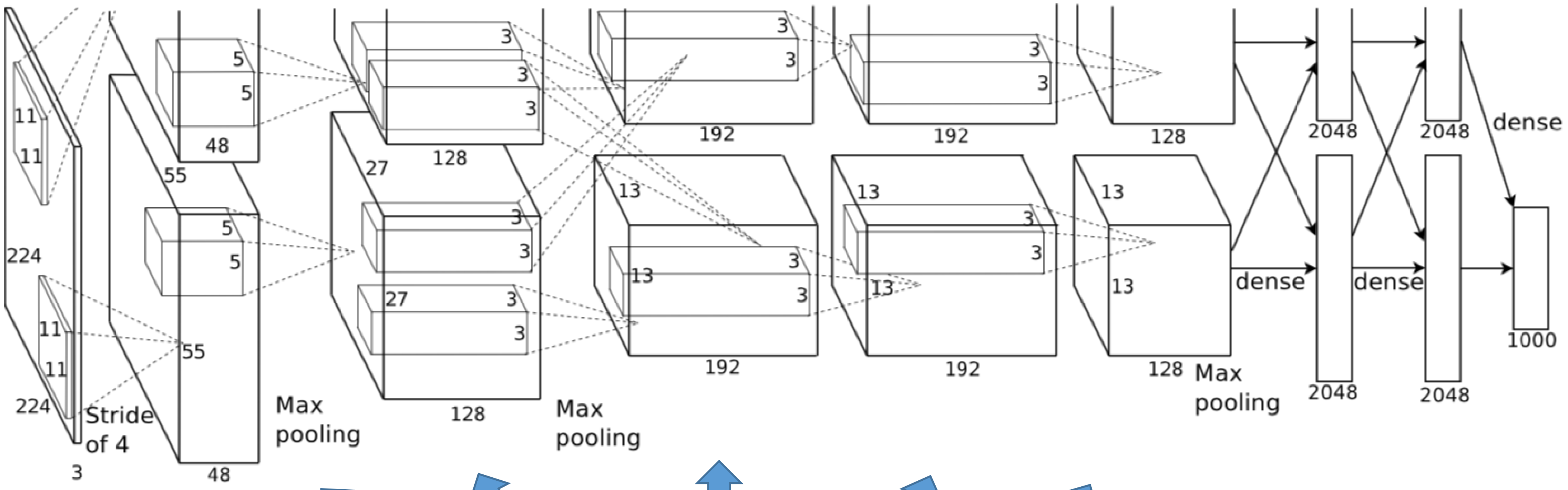
Outline

- Model Introduction
- Simulation Results
  - Experiment Setup
  - Analysis
- Hardware-friendly Design
- Conclusion
- Reference

MODEL INTRODUCTION

AlexNet Architecture [1]

3 Fully connected layer



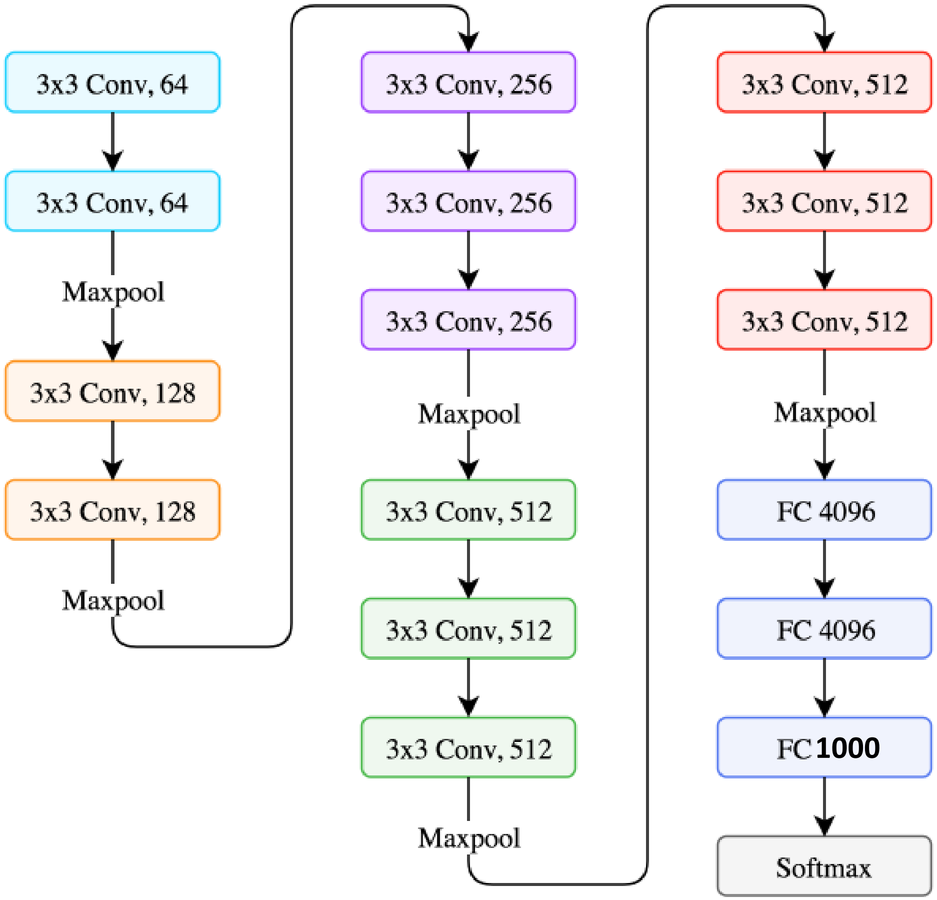
5 Convolutional layer

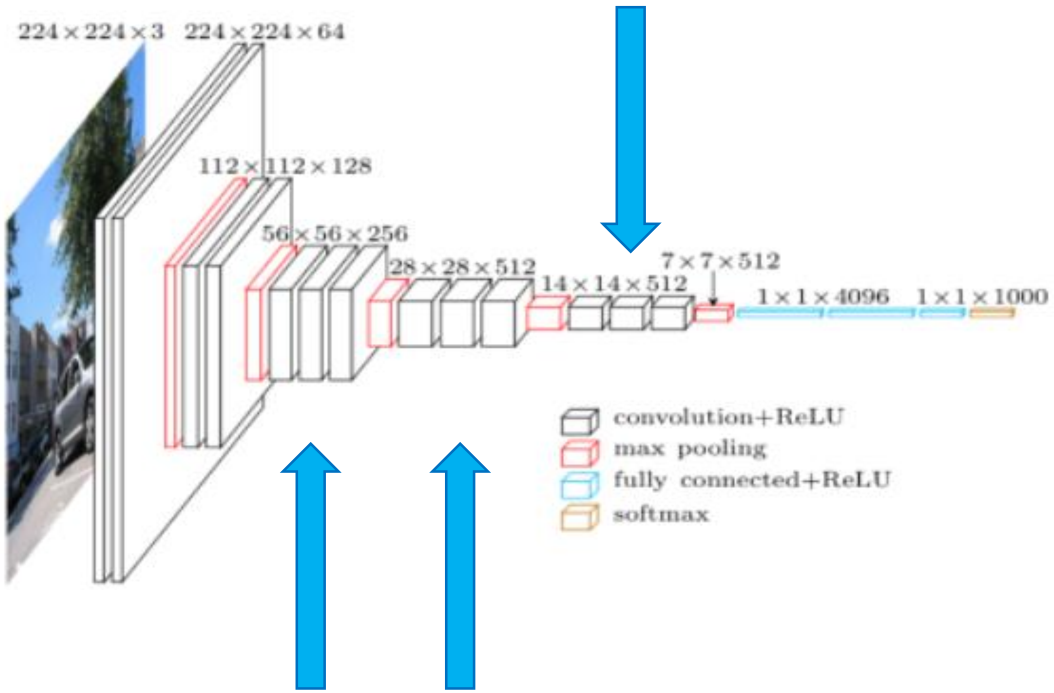
Total:
60 million parameters
650000 neurons

VGG NETWORK [2]

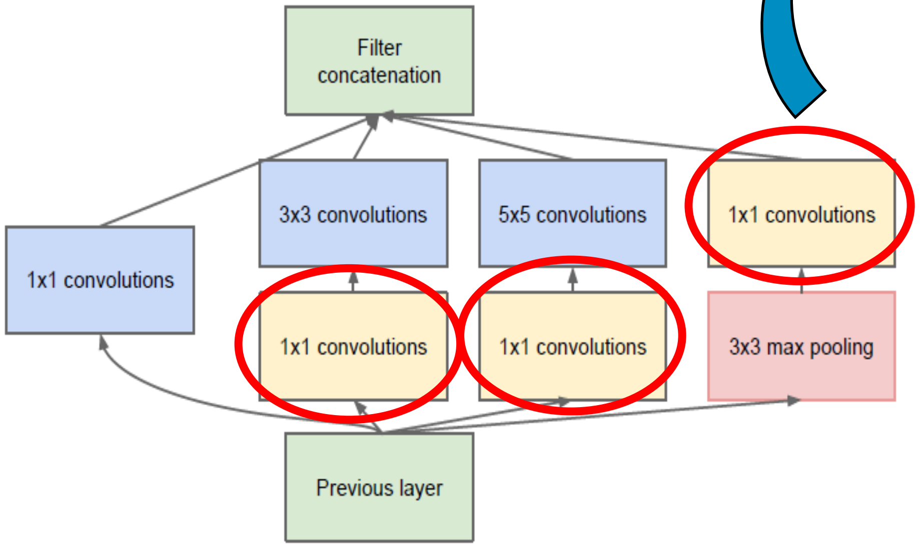
- Simonyan and Zisserman, 2014
- Simplicity
- Small 3 x 3 filters

VGG Architecture [2]

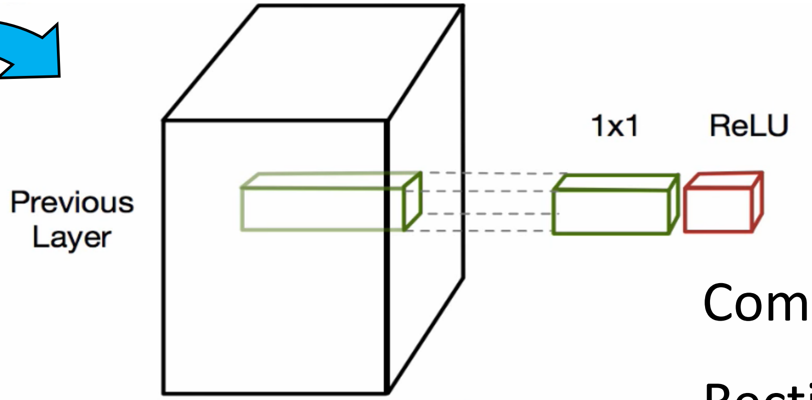




Inception v1 - Module [3]

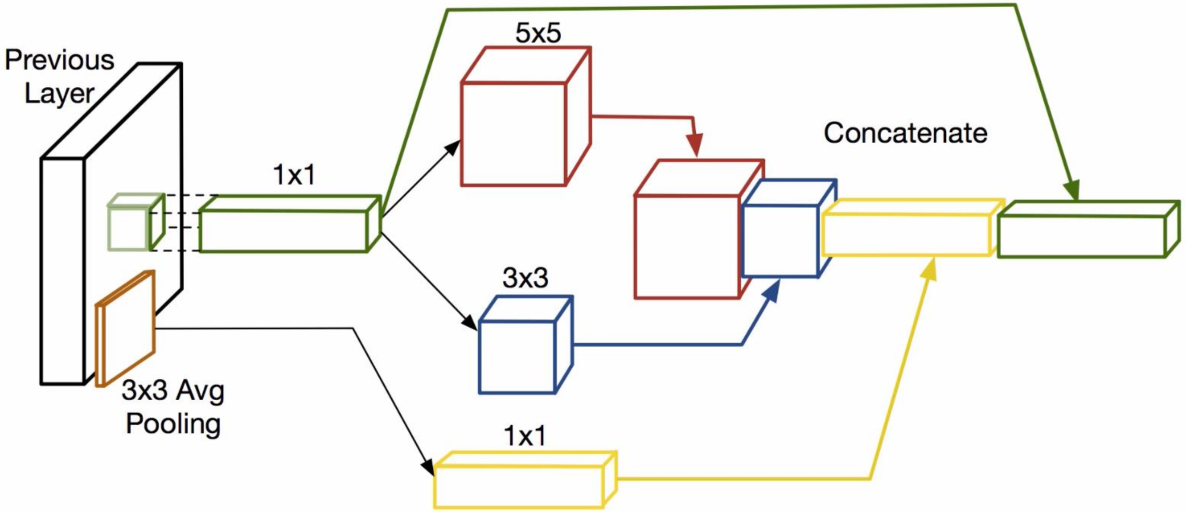


(b) Inception module with dimensionality reduction

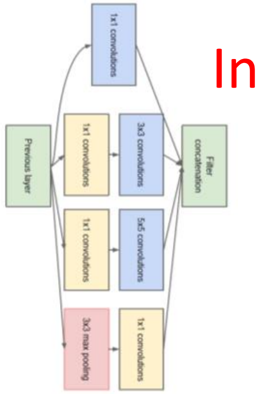


Compute reductions

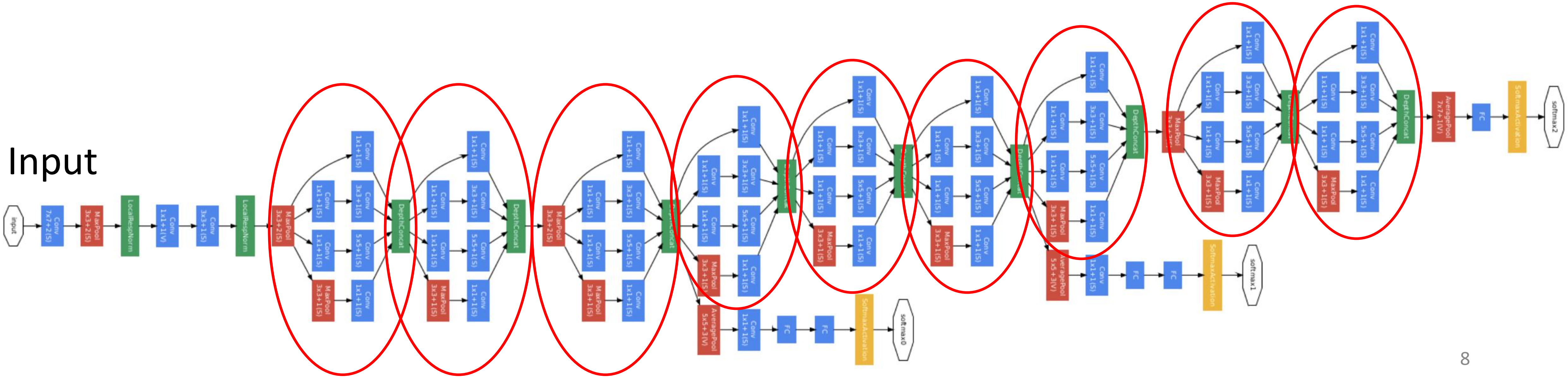
Rectified linear activation



Inception v1 - Architecture [3]



Inception module



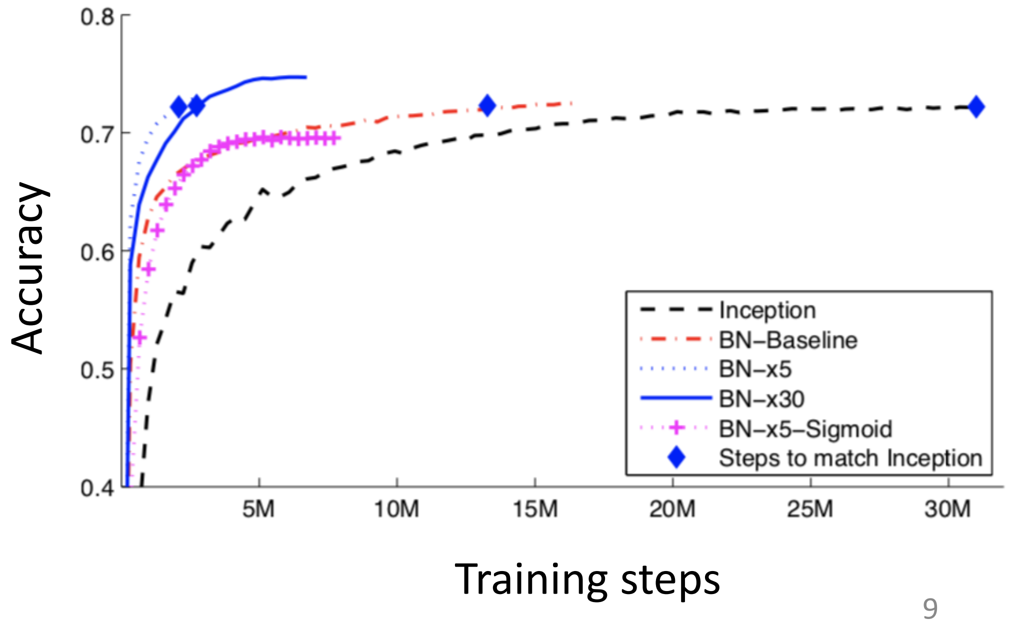
Input

Inception v2 [4]

- Reduce the needs for layers to continuously adapt to the new distribution:

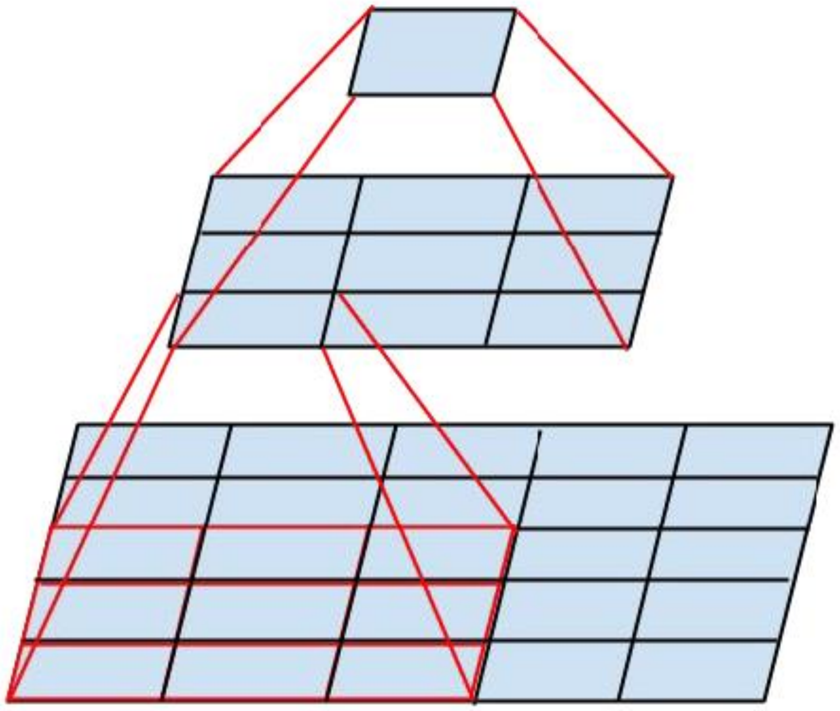
Batch Normalization

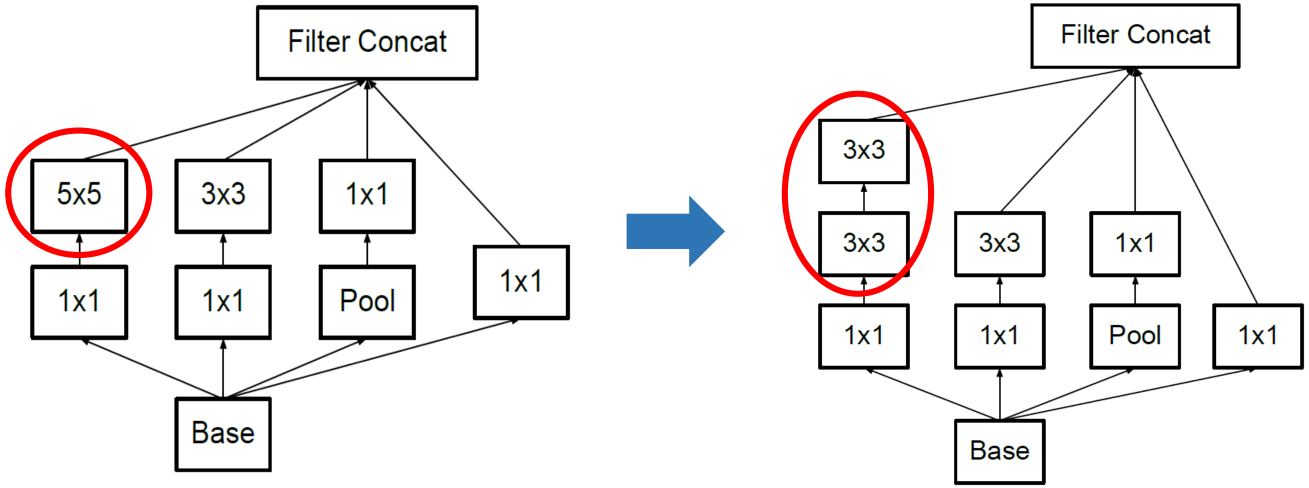
- Train faster and have better performance



Inception v3 [5]

- Factorization into smaller convolutions

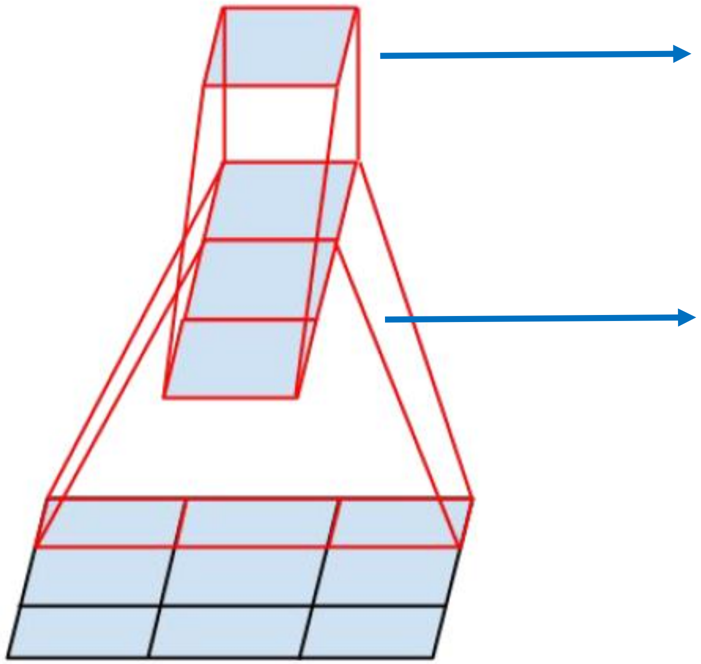


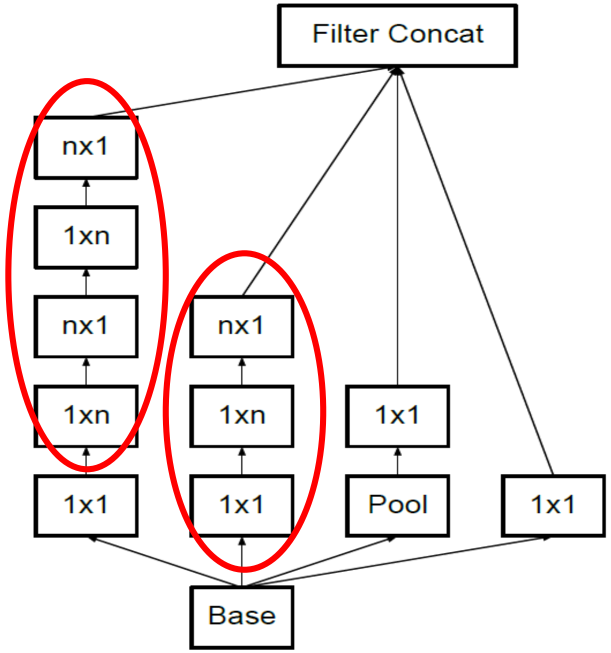


Less parameters with the same input size and output depth

Inception v3 [5]

◆ Spatial Factorization into Asymmetric Convolutions

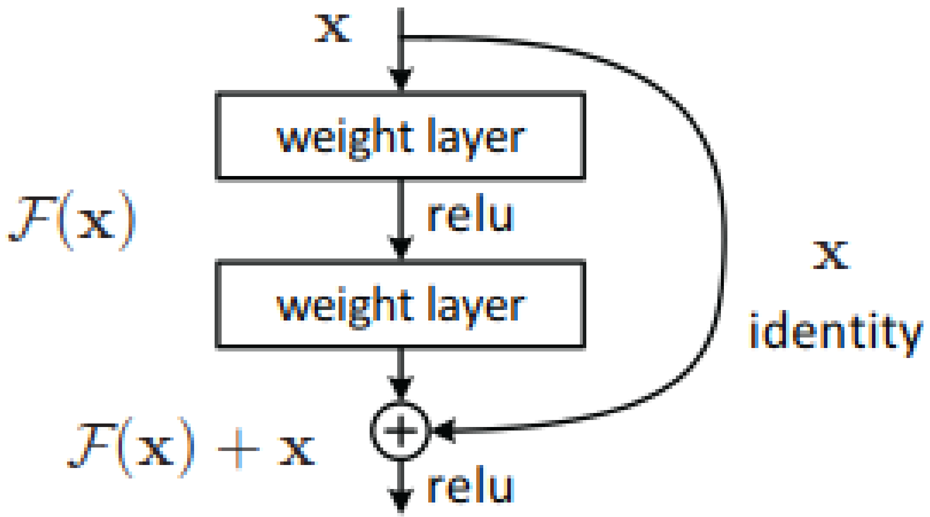




Cost saving increases dramatically as n grows

ResNet Module [6]

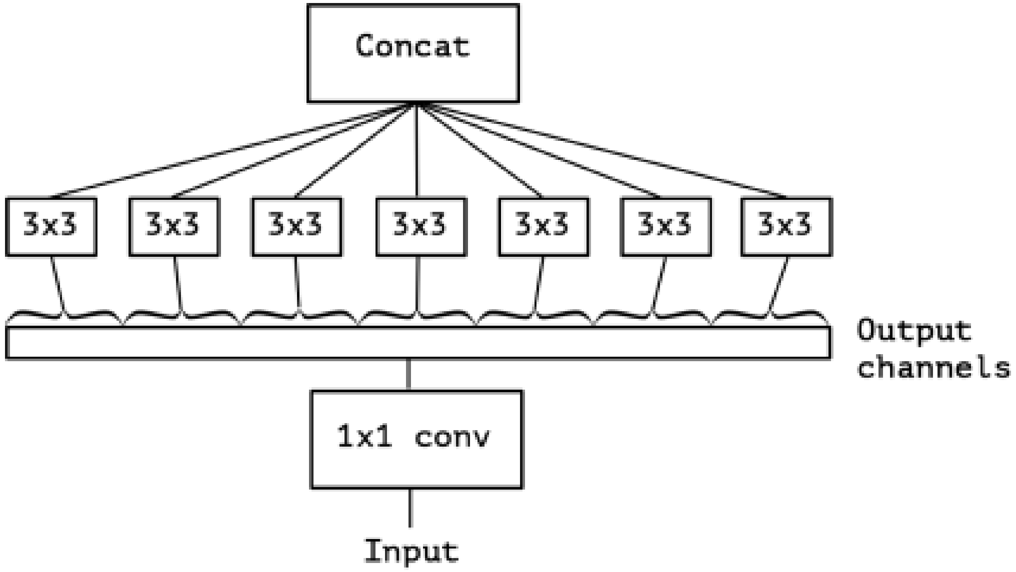
- A intuitive implementation of deeper model with at least same performance with shallower counterpart is identical mapping of adding layers.
- A residual learning framework was proposed : Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly **let these layers fit a residual mapping (Hypothesis of this paper).**

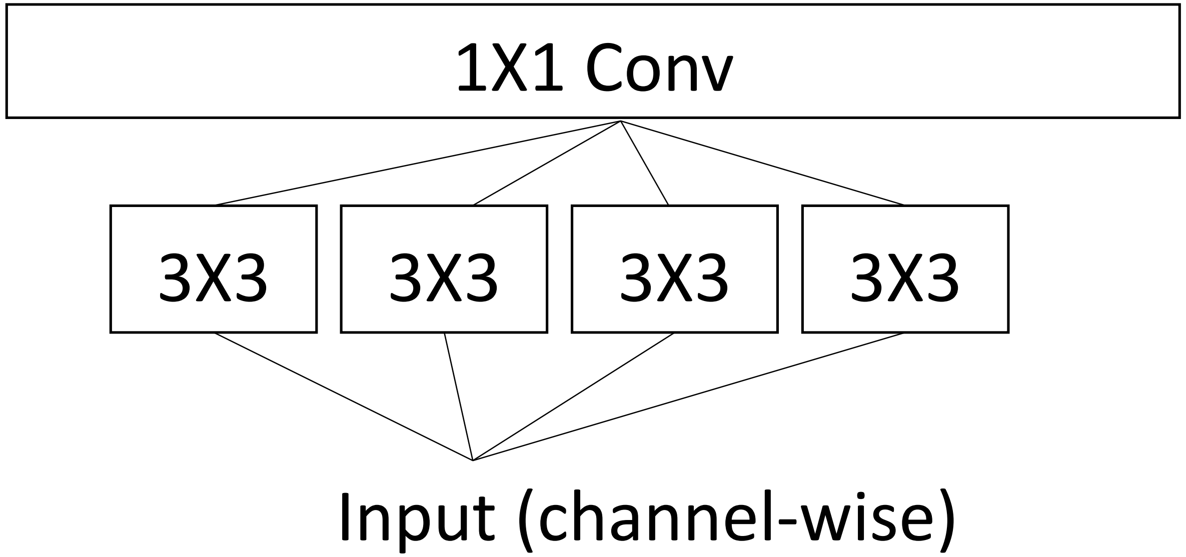


[6] Fig. 2. residual learning block

Xception v.s. Inception

[7] Fig. 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.



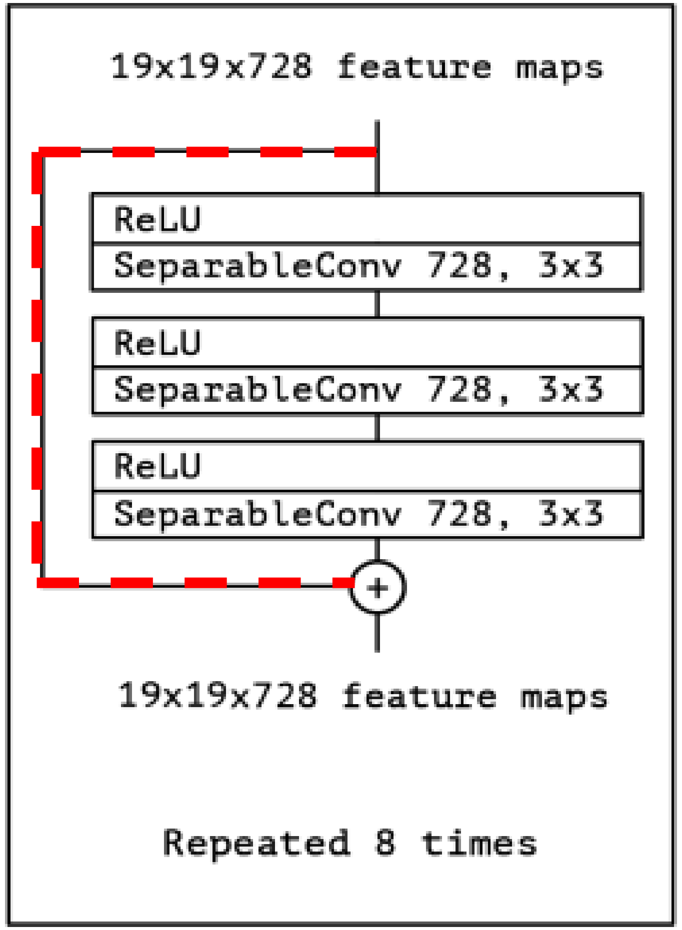


Difference :

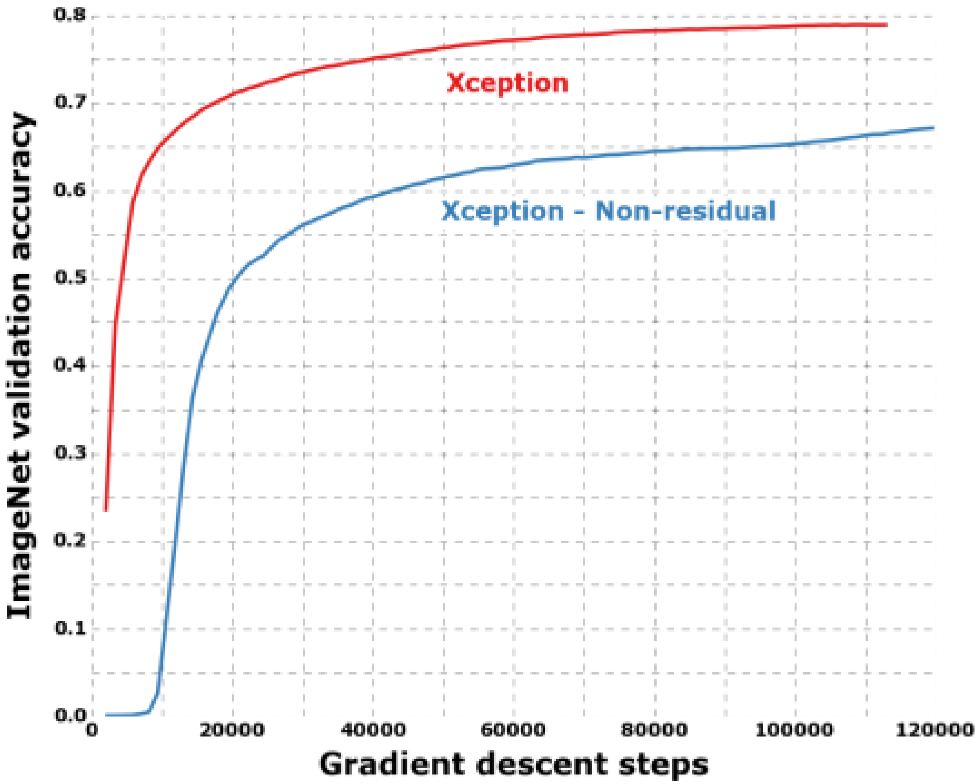
(1) order : 1X1-spatial ; spatial-1X1

(2) Non-linearity

Xception v.s. ResNet [7]



[7] Fig. 5. The Xception architecture – middle flow



[7] Fig. 9. Training profile with and without residual connections.

SIMULATION RESULTS

Testing Data:

- ILVSRC 2012 validation data
- 50,000 images, 1000 classes

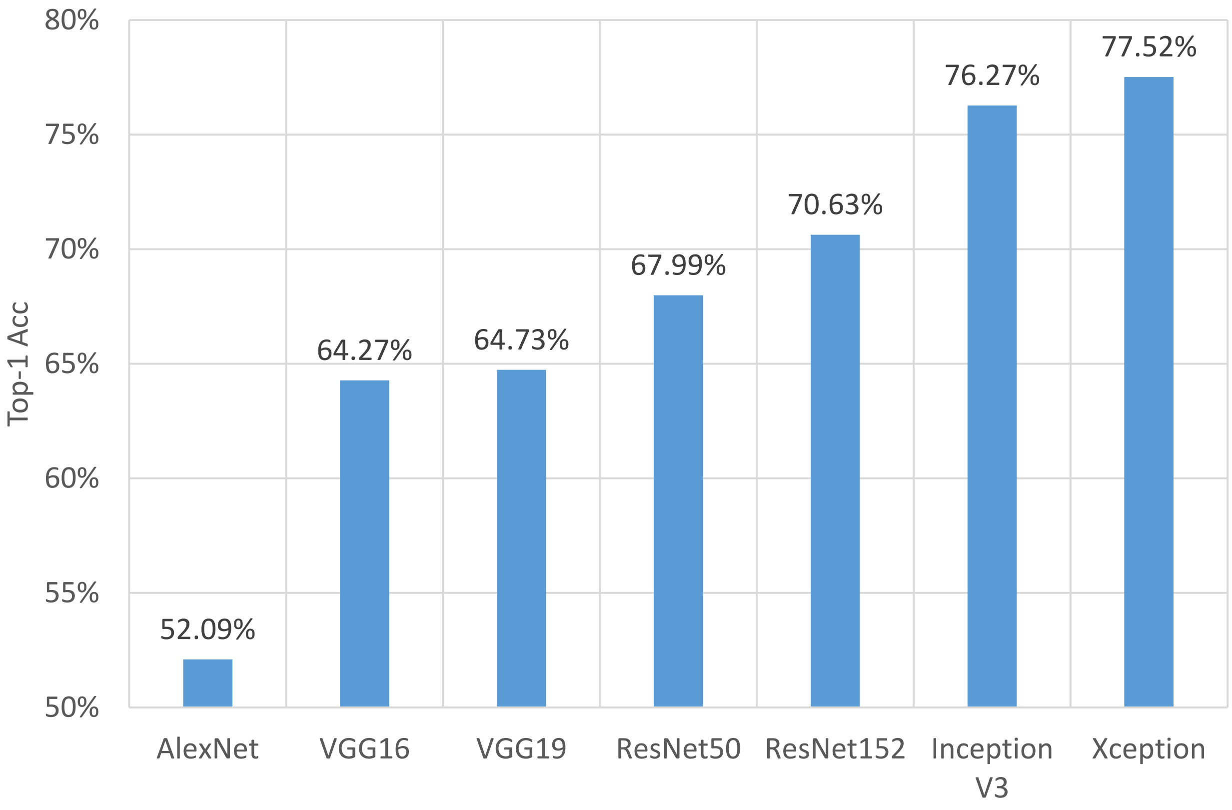
Experiment Setup

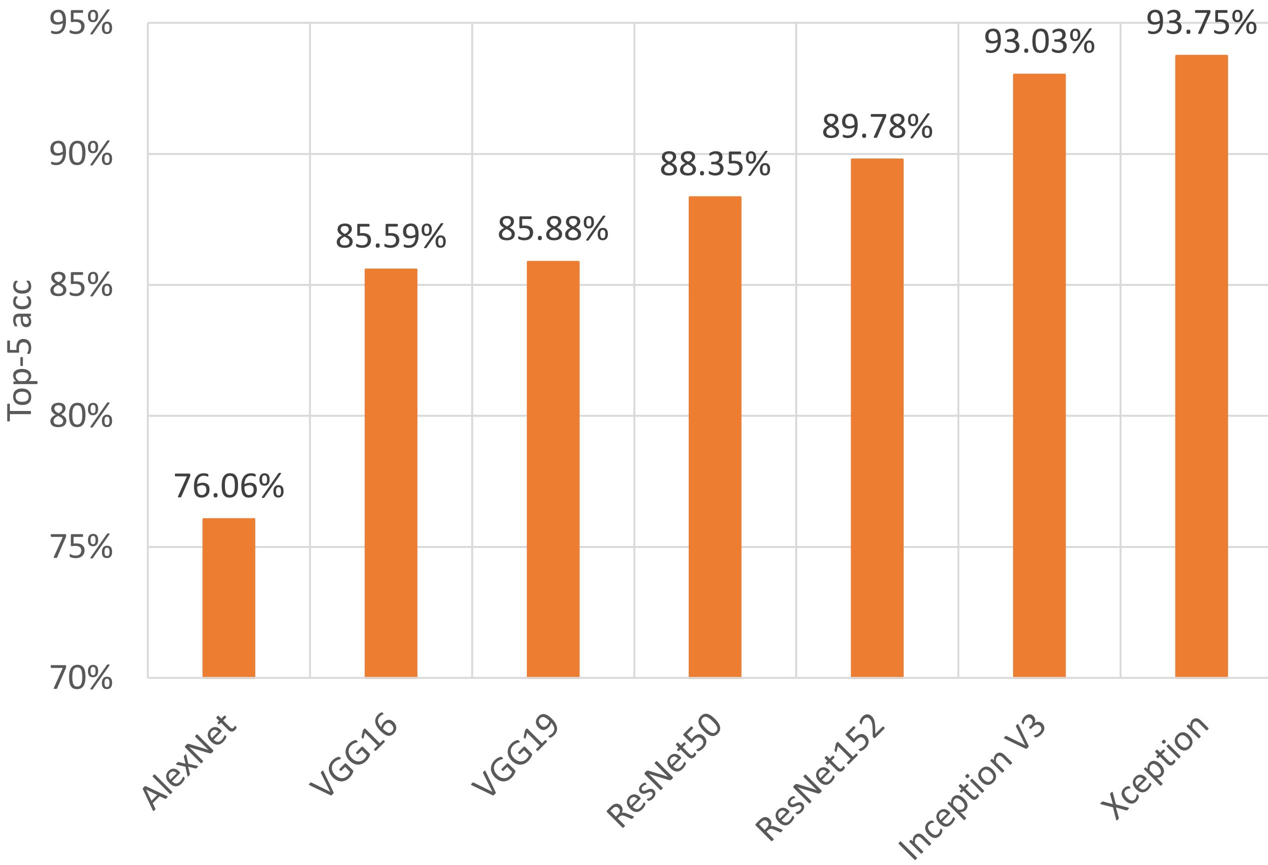
- Software
  - Python3
  - Tensorflow 1.4
  - Keras 2.0
- Hardware
  - 2 GHz 2 Cores Intel Core i5 CPU
  - Nvidia 1080 GPU

Analysis Topics

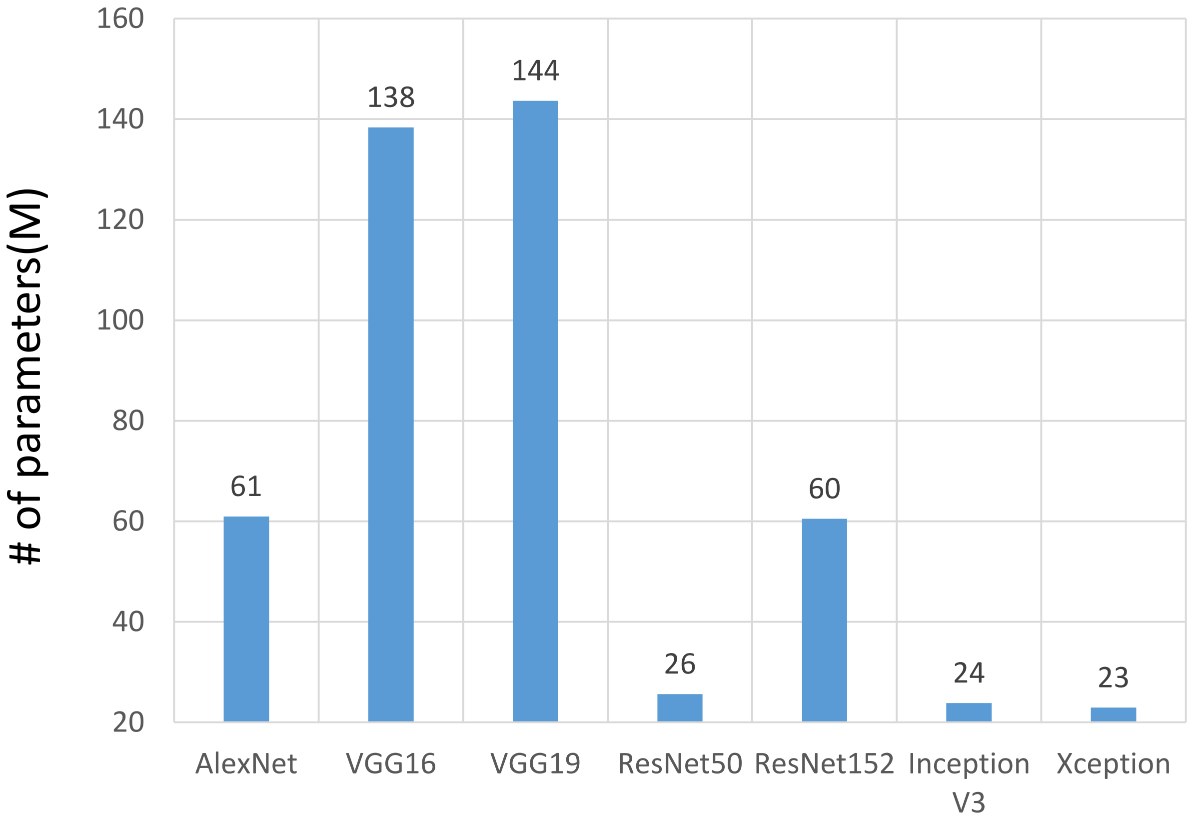
- Accuracy
  - Top-1 accuracy
  - Top-5 accuracy
- # of parameters
- # of operations (G-FLOPS)
- Inference time per image
- Power

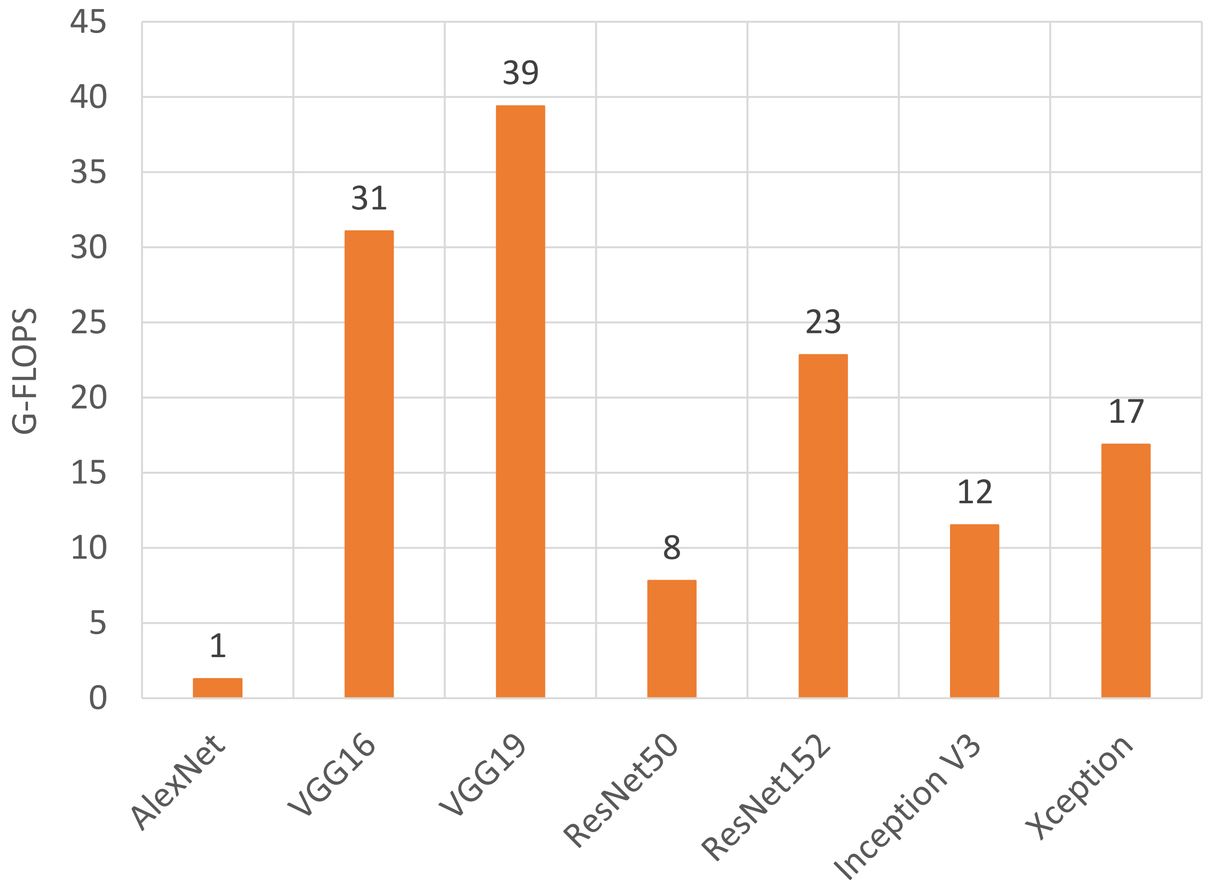
Accuracy


80%
75%
70%
65%
60%
55%
50%
Top-1 Acc
52.09%
64.27%
64.73%
67.99%
70.63%
76.27%
77.52%
AlexNet
VGG16
VGG19
ResNet50
ResNet152
Inception V3
Xception


95%
90%
85%
80%
75%
70%
Top-5 acc
76.06%
85.59%
85.88%
88.35%
89.78%
93.03%
93.75%
AlexNet
VGG16
VGG19
ResNet50
ResNet152
Inception V3
Xception

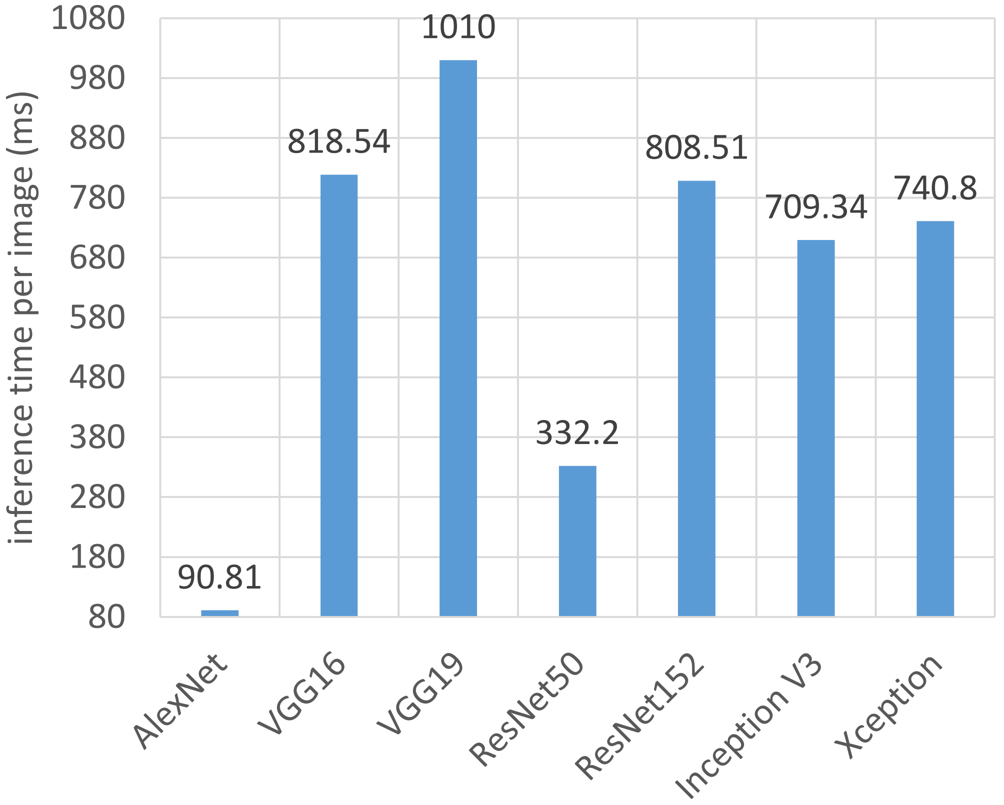
of parameters & # of operations

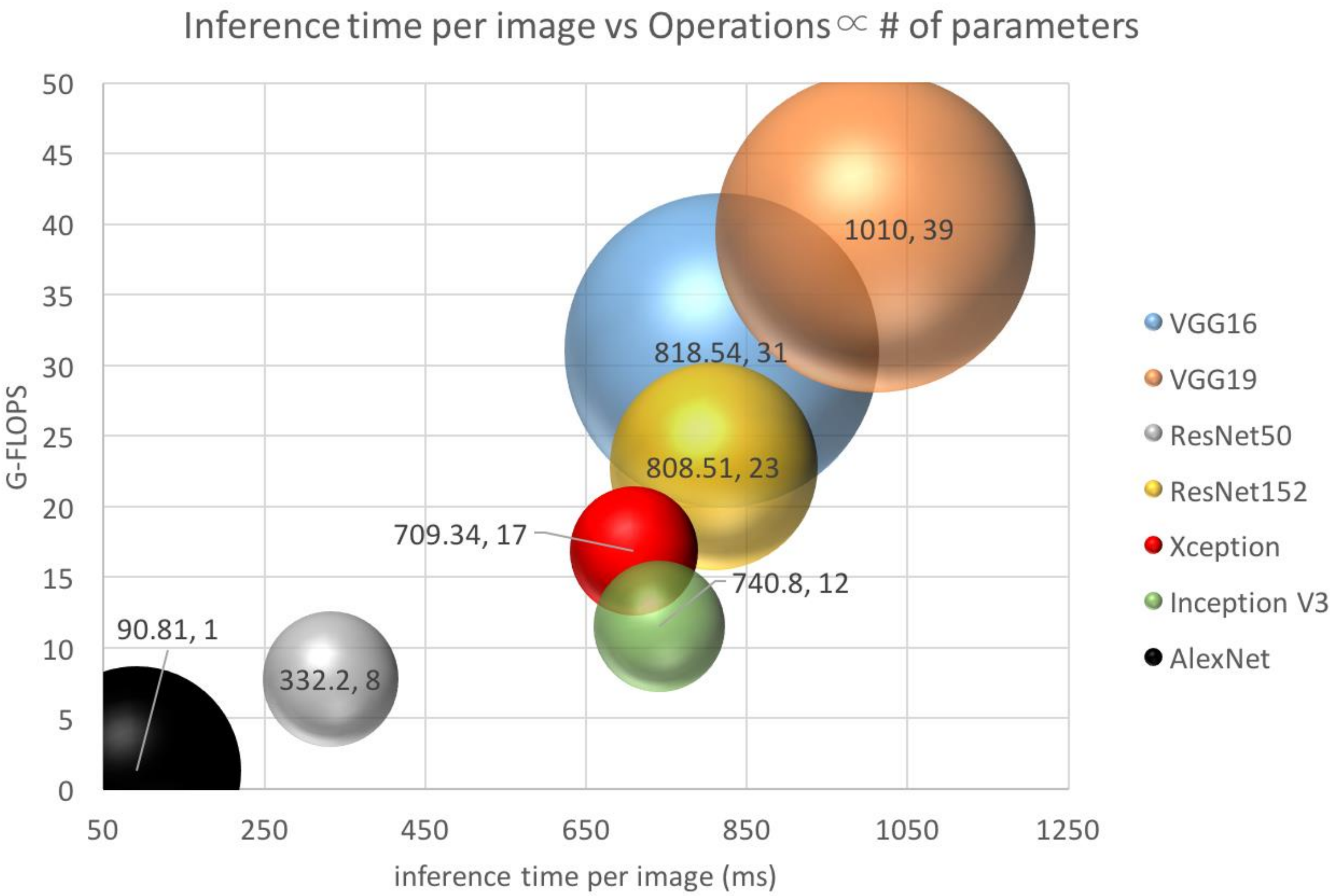




Inference Time

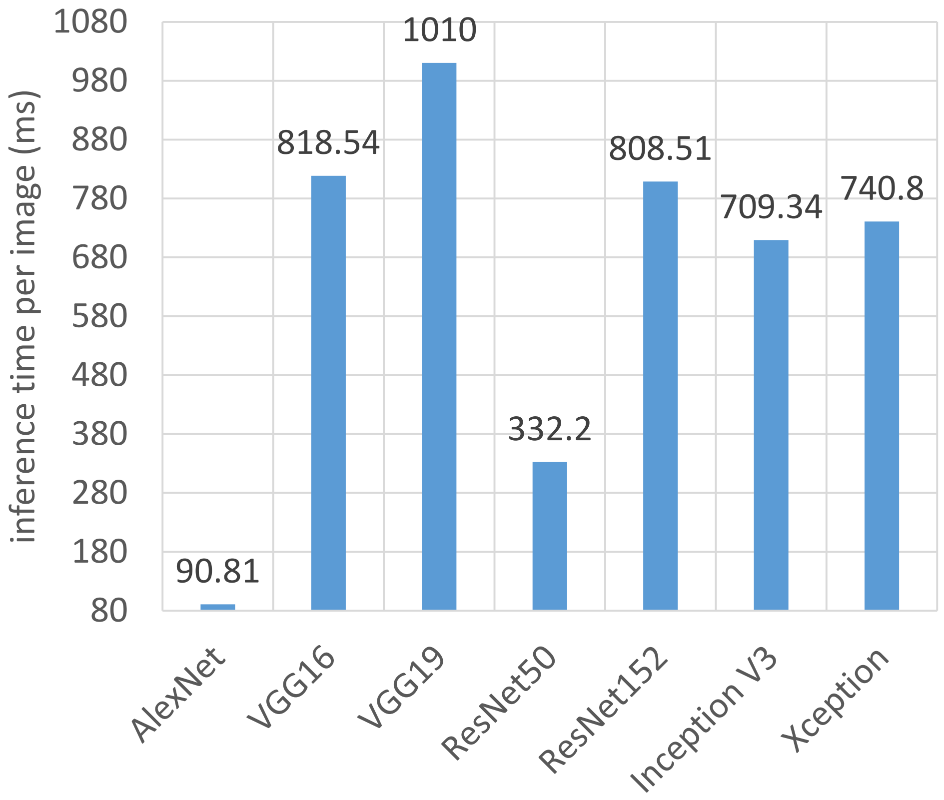
AVLSI_1061

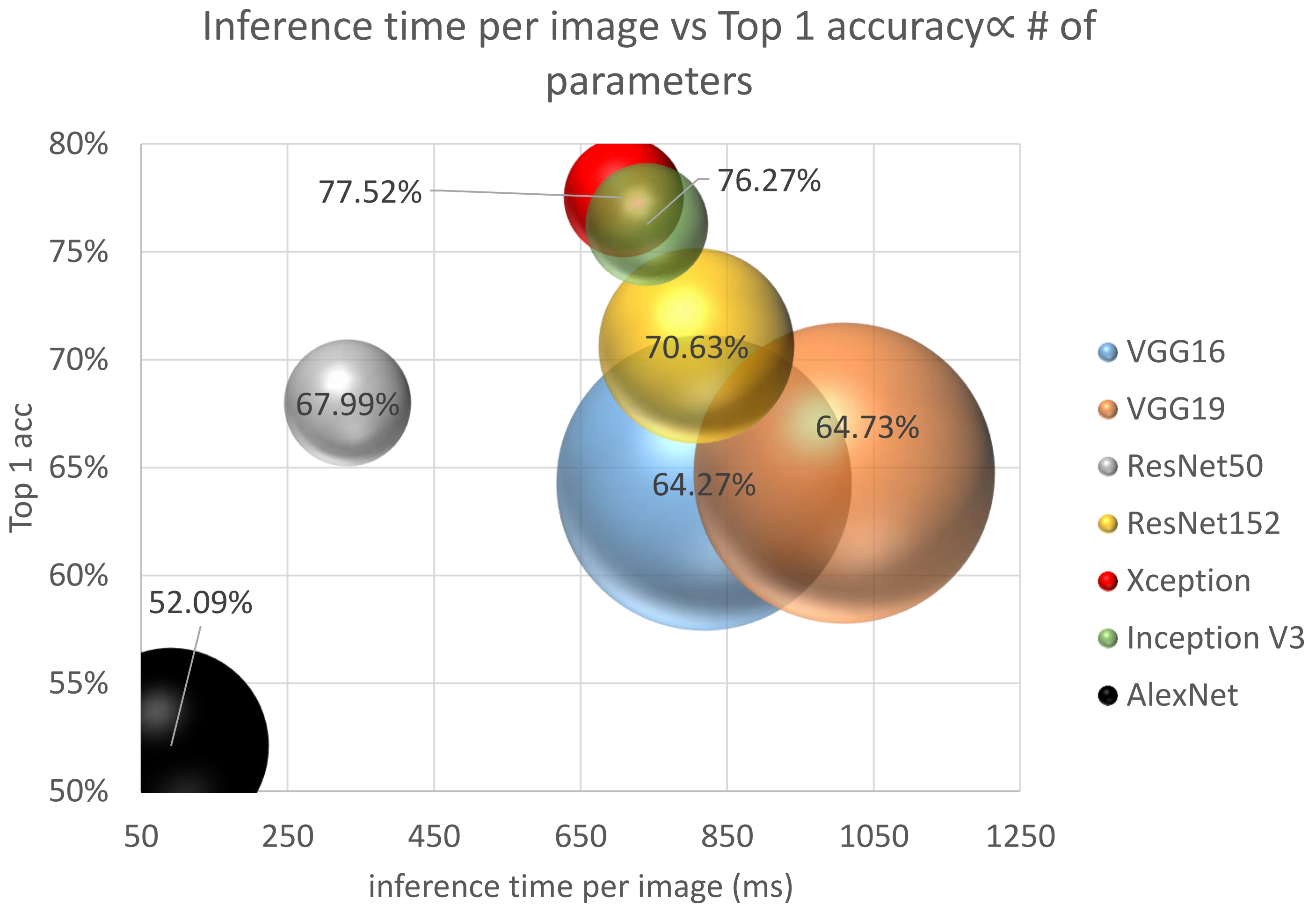




☞ Inference time are measured on 2 GHz 2 Cores Intel Core i5 CPU

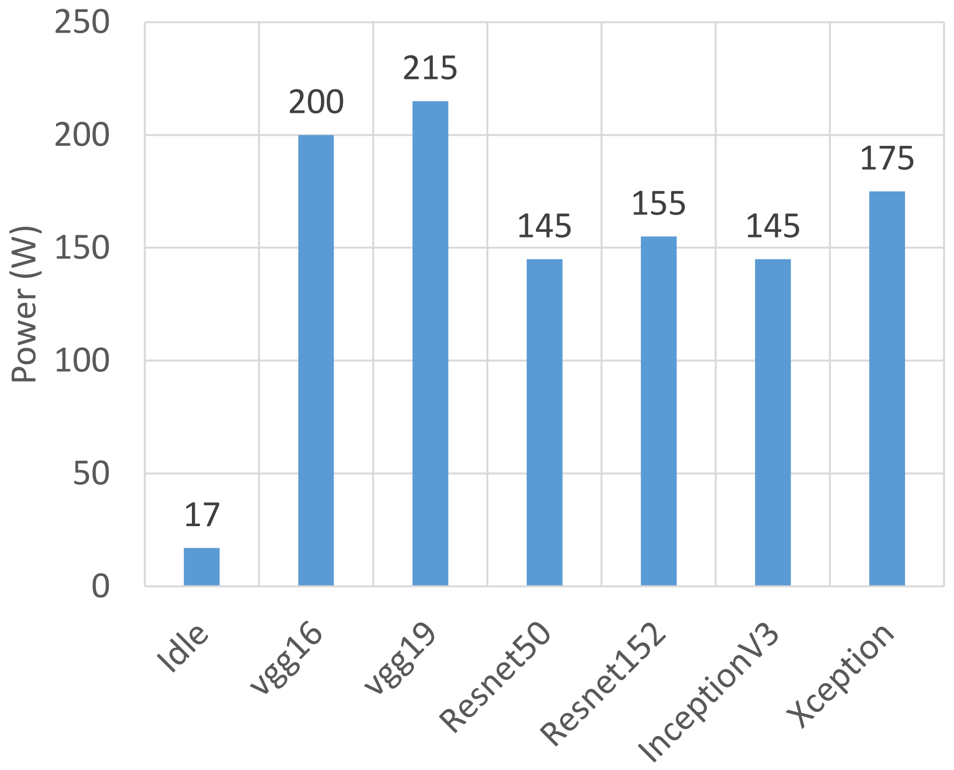
Inference Time v.s. Accuracy

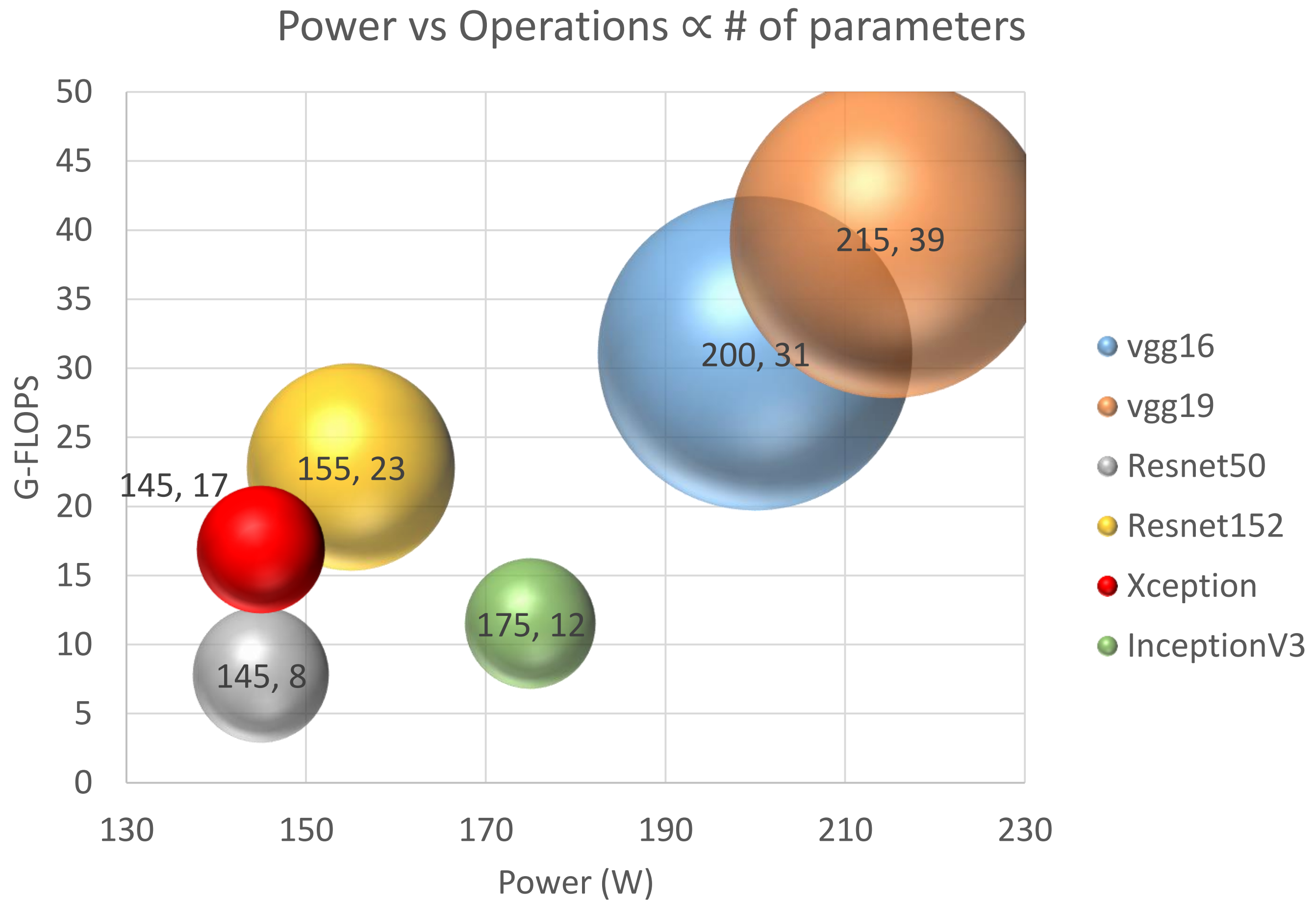




☞ Inference time are measured on 2 GHz 2 Cores Intel Core i5 CPU

Power





☞ Power are measured on Nvidia GeForce 1080

HARDWARE-FRIENDLY DESIGN

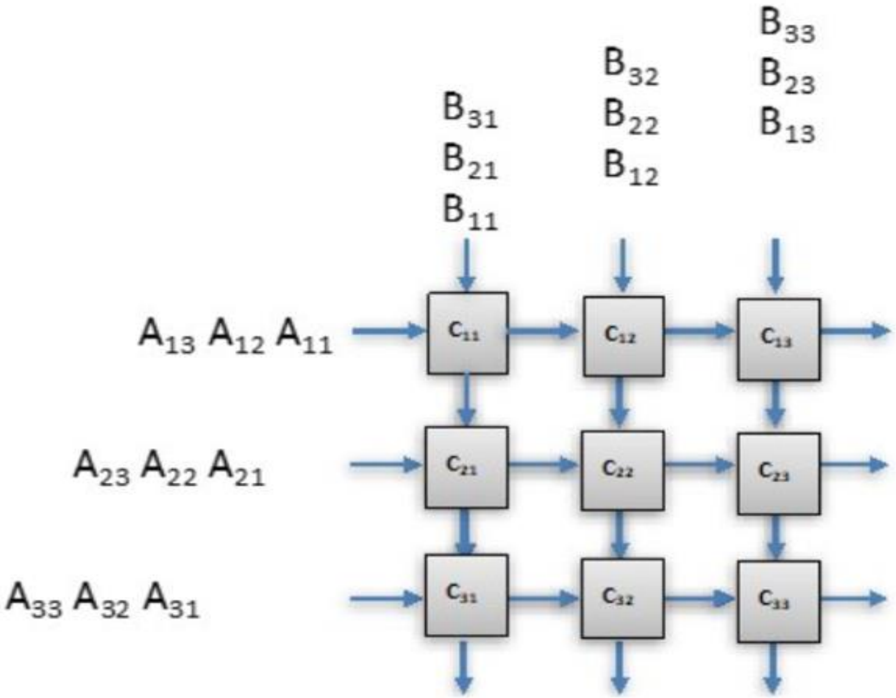
Systolic Array [9]

- Dense Linear Algebra Accelerator

  Accelerate Matrix-Matrix and Matrix-Vector Operation

  → **Systolic Array**

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$



Quantization [9]

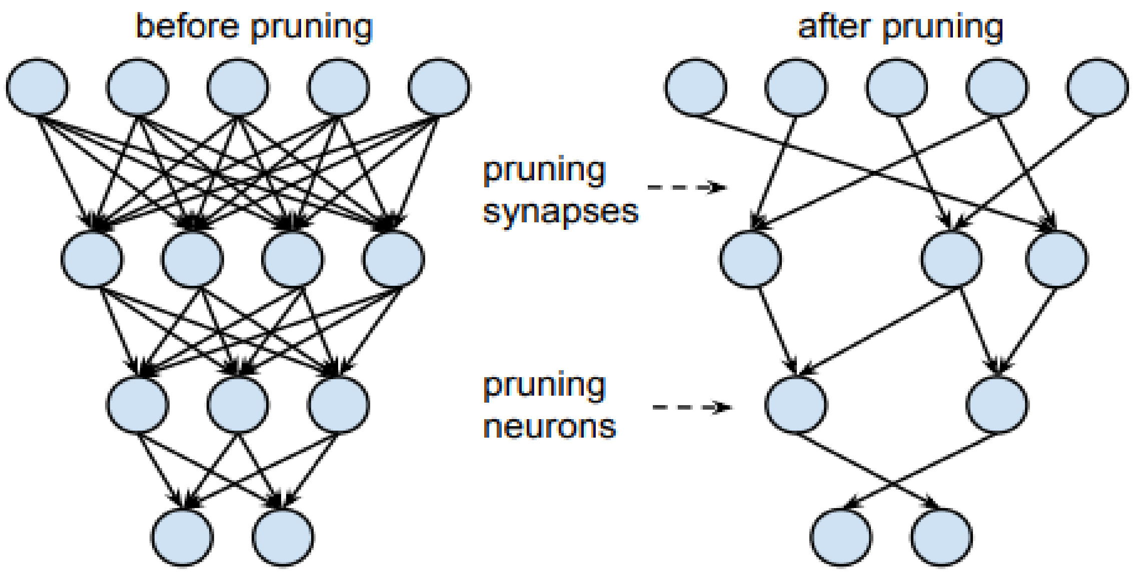
- Systolic Array
- Quantization

Floating point Design → Fixed-point Design

→ Retraining the network

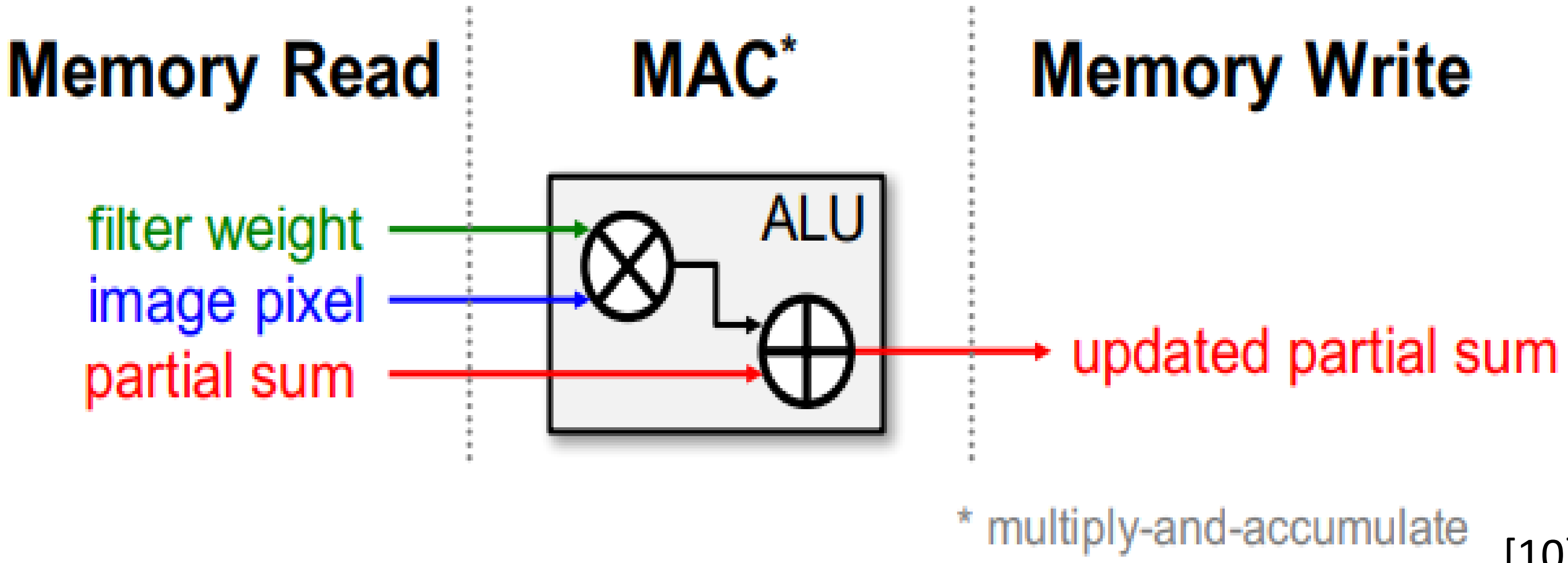
Pruning [9]

- Systolic Array
- Quantization
- Pruning



Memory access is a critical problem

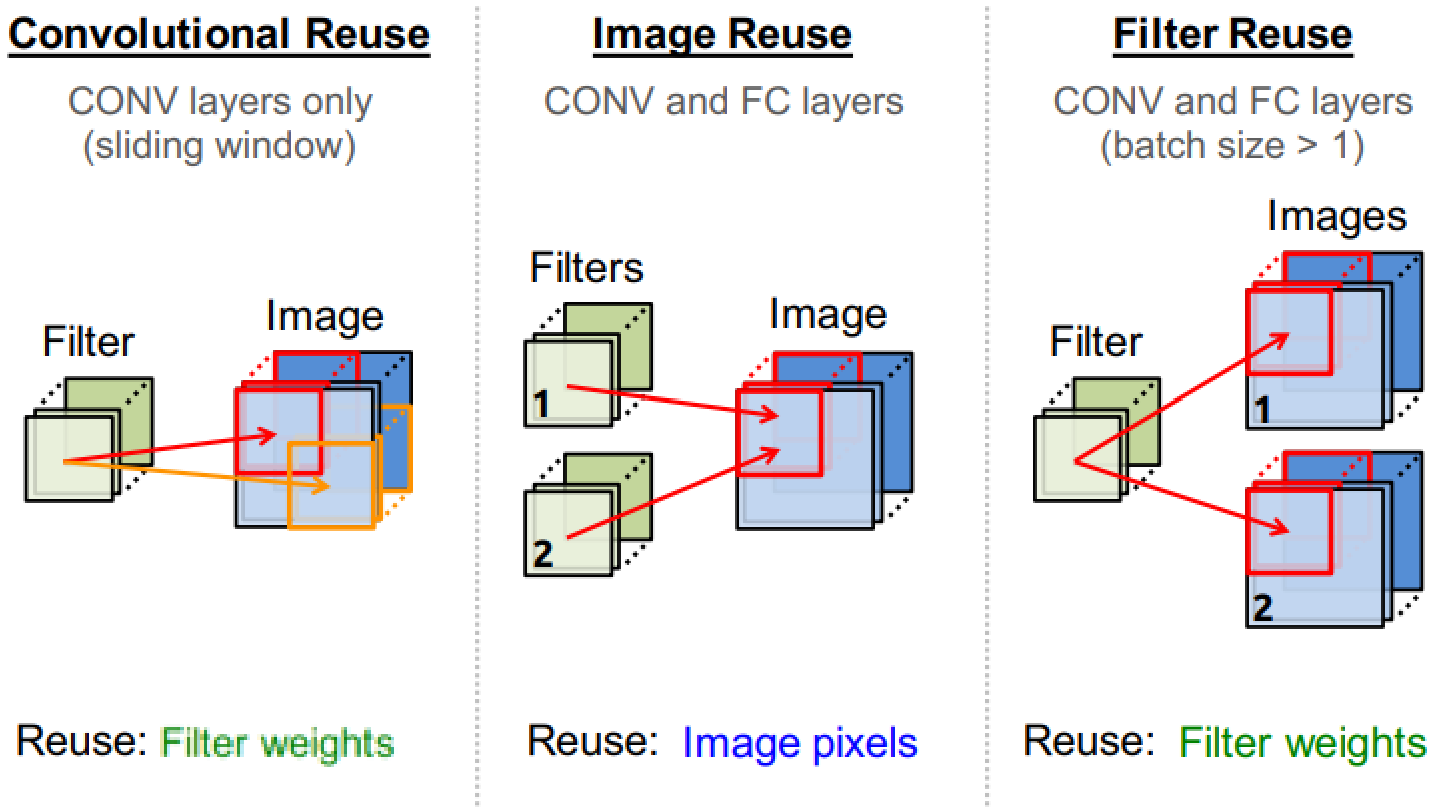
- Systolic Array
- Quantization
- Pruning
- Data Reuse



[10]

Data Reuse [10]

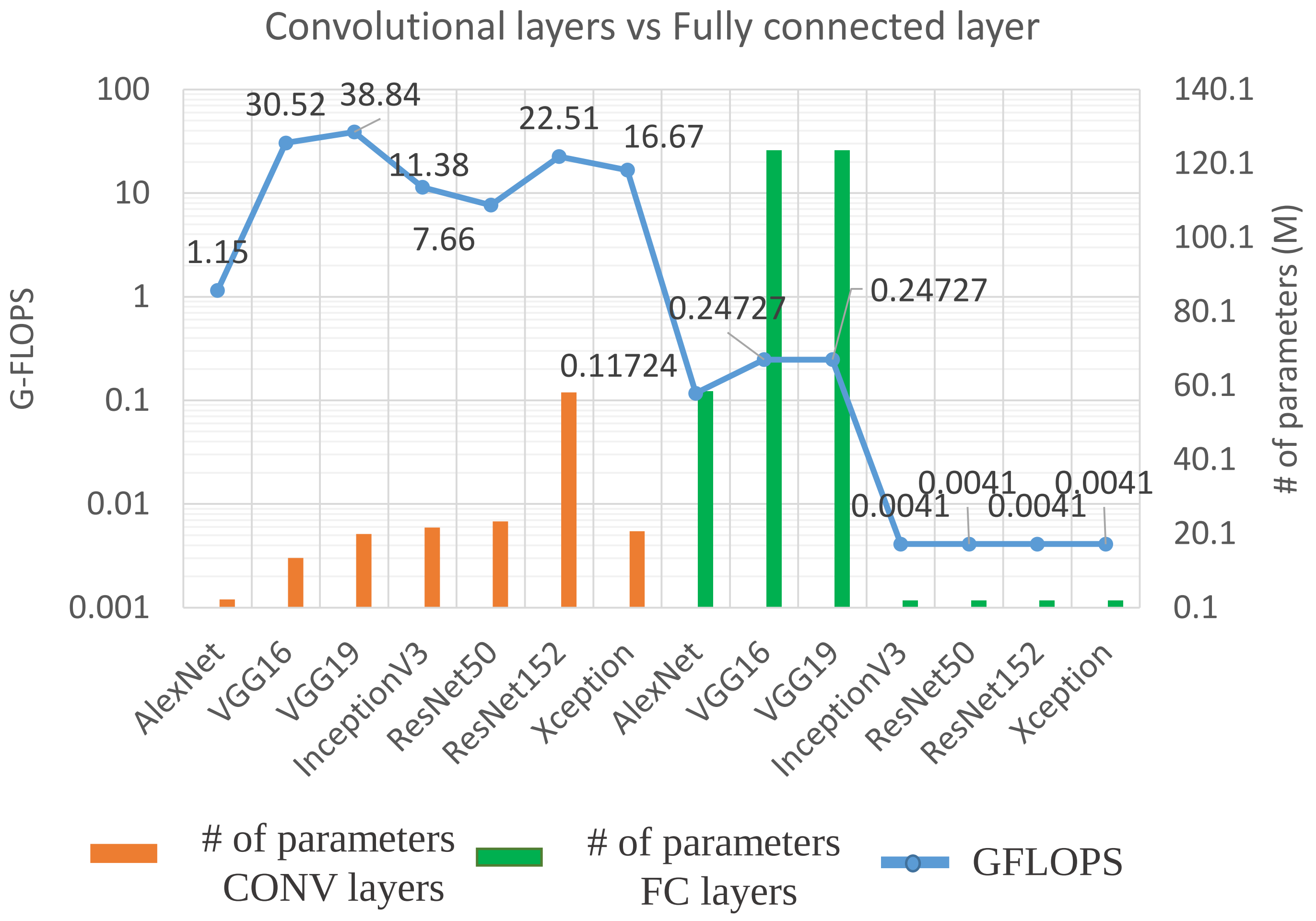
- Systolic Array
- Quantization
- Pruning
- Data Reuse



[10]

of parameters / GFLOPs in FC and CONV layers

AVLSI_1061



- It is observed CONV layers in the AlexNet and VGG require **less parameters** but **more computations** and FC layers require **more parameters** but **less computations** .

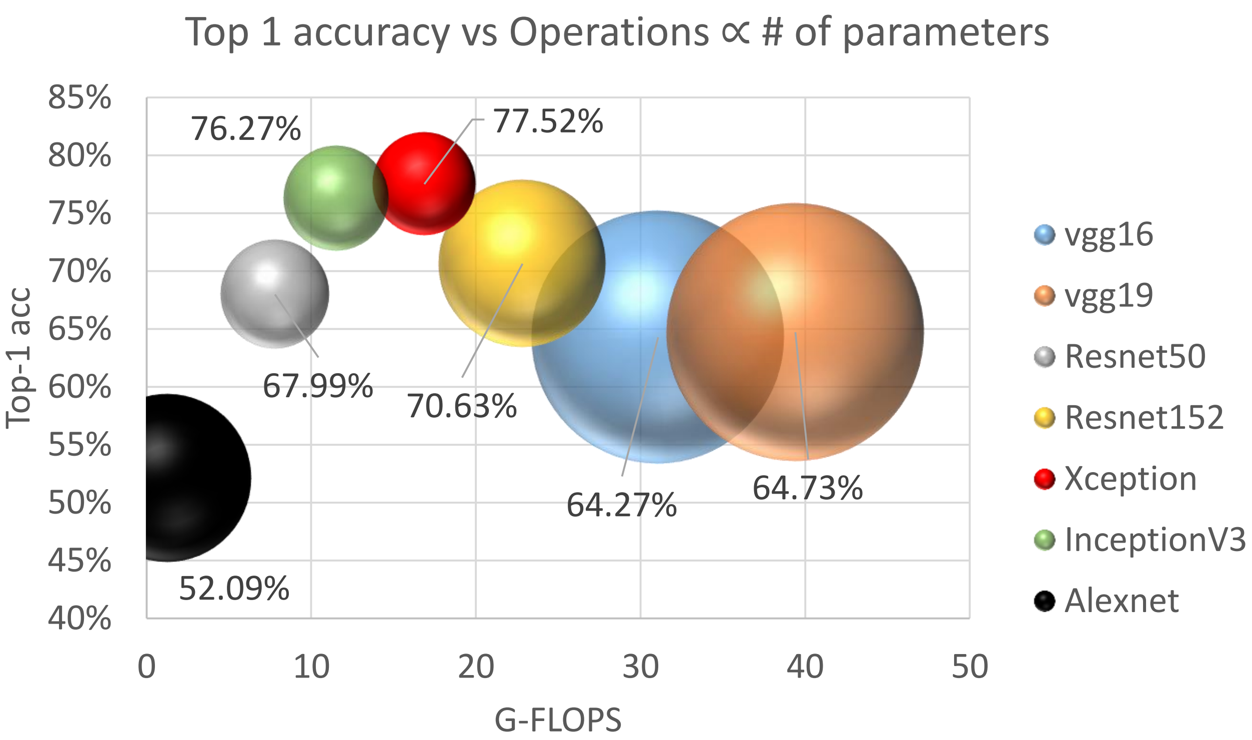
- However, for very deep network (ResNet, Inception, Xception), the critical issue lies on **CONV layers.** (Since there will be only one FC layer)

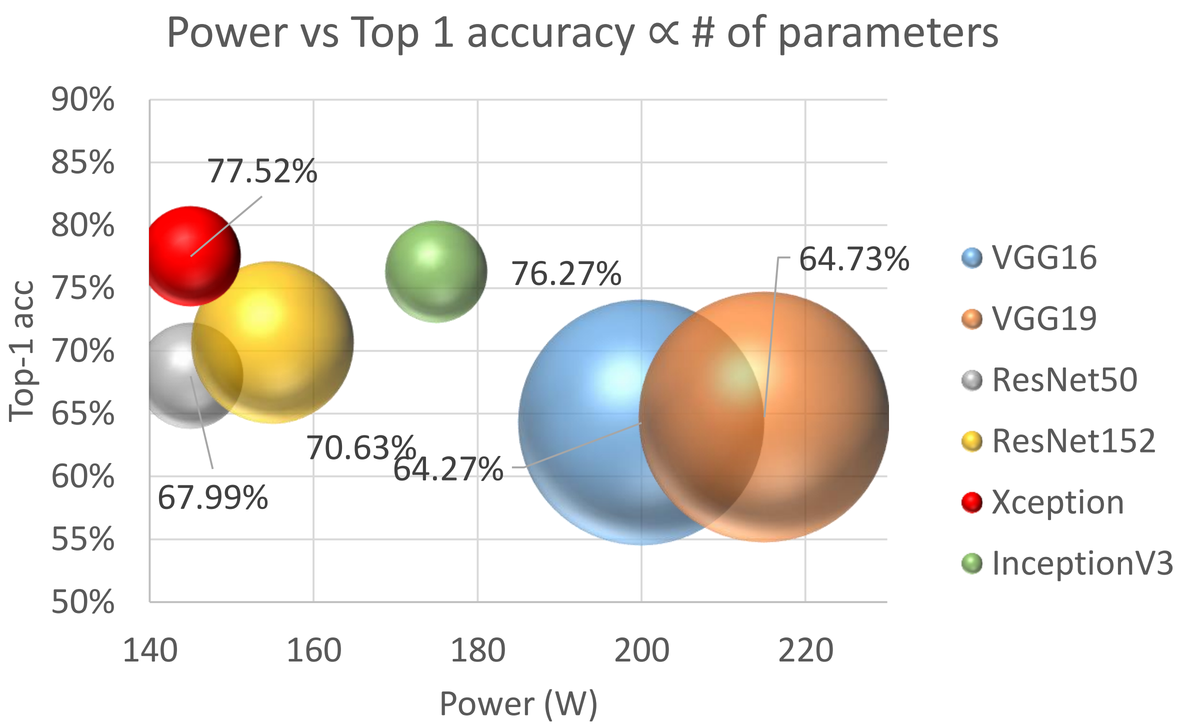
Conclusion

	Top1 Acc	Top5 Acc	# of Parameters	Power(W)	GFLOPs	Inference Time per Image(ms)
AlexNet	52.09%	76.06%	60,965,224	--	1.27	90.81
VGG16	64.27%	85.59%	138,357,544	200	31.06	818.54
VGG19	64.73%	85.88%	143,667,240	215	39.40	1010
ResNet50	67.99%	88.35%	25,636,712	145	7.80	332.2
ResNet152	70.63%	89.78%	60,495,656	155	22.83	808.51
InceptionV3	76.27%	93.75%	23,851,784	175	11.51	740.8
Xception	77.52%	93.03%	22,910,480	145	16.87	709.34

Conclusion

AVLSI_1061





Reference

- [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”