The Ramifications of Making Deep Neural Networks (DNNs) Compact

Nandan Kumar Jha, Sparsh Mittal, Govardhan Mattela
{cs17mtech11010,sparsh,cs17resch01004}@iith.ac.in

Department of Computer Science and Engineering
IIT Hyderabad

January 9, 2019

Outline

- Conventional DNNs : A rudimentary introduction
- Compact DNNs : Pitfalls and fallacies
- Desiderata of compact DNNs
- Conclusion

ImageNet era : 2010 - 2017

Conventional DNNs are large and over-parameterized.

ImageNet era : 2010 - 2017

Conventional DNNs are large and over-parameterized.

Conventional DNNs [e.g., AlexNet, VGGNet, Inception-v3/v4 etc.]

- Brute-force approach
  - large amount of training data
  - large model
- Little consideration about efficiency

ImageNet era : 2010 - 2017

Conventional DNNs are large and over-parameterized.

Conventional DNNs [e.g., AlexNet, VGGNet, Inception-v3/v4 etc.]

- Brute-force approach
  - large amount of training data
  - large model
- Little consideration about efficiency

Challenges : More computations, slow to deploy, high bandwidth requirement and energy inefficient

ImageNet era : 2010 - 2017

Conventional DNNs are large and over-parameterized.

Conventional DNNs [e.g., AlexNet, VGGNet, Inception-v3/v4 etc.]

- Brute-force approach
  - large amount of training data
  - large model
- Little consideration about efficiency

Challenges : More computations, slow to deploy, high bandwidth requirement and energy inefficient

" DNNs are over-parameterized. We use SGD, a convex optimization method, to optimize a DNN, which is a highly non-convex problem. Redundancy is needed to avoid getting stuck at a local minima."

- Han et al., " Bandwidth-Efficient Deep Learning ", DAC 2018

Post ImageNet era

Compact models : Prune large models or design compact DNNs

Prune large models

- Reduces model size
  - low storage requirement
  - can be accommodated on-chip
- Reduces computational complexity
  - faster inference
  - energy efficient
- Enable deployment on resource constrained platforms

Post ImageNet era

Compact models : Prune large models or design compact DNNs

Prune large models

- Reduces model size
  - low storage requirement
  - can be accommodated on-chip
- Reduces computational complexity
  - faster inference
  - energy efficient
- Enable deployment on resource constrained platforms

Challenges: Exhaustive retraining, unstructured/irregular sparsity, inefficient on general purpose processors

Post ImageNet era

Compact models : Prune large models or design compact DNNs

Prune large models

- Reduces model size
  - low storage requirement
  - can be accommodated on-chip
- Reduces computational complexity
  - faster inference
  - energy efficient
- Enable deployment on resource constrained platforms

Challenges: Exhaustive retraining, unstructured/irregular sparsity, inefficient on general purpose processors

Compact Neural Network algorithms

- Deeper but narrower
- Sparse but regular
- Reduces computations and #parameters
- Efficient on general purpose hardware

Compact DNNs

Model Name	#Param (M)	#MACs (M)	Top-1
AlexNet	60.97	723	57.1
SqueezeNet V1.0	1.25	848	57.5
SqueezeNet V1.1	1.24	349	57.1
1.0-G-SqNxt-23	0.54	221	57.16
1.0-SqNxt-23	0.72	273	59.05
1.0-SqNxt-23v5	0.93	225	59.24
2.0-SqNxt-23	2.36	726	67.18
2.0-SqNxt-23v5	3.22	703	67.44
1.0 MobileNet-224	4.23	574	70.6
DenseNet-121	7.98	3080	75.0
GoogLeNet	7.00	1590	71.0
InceptionV2	11.2	2220	76.6

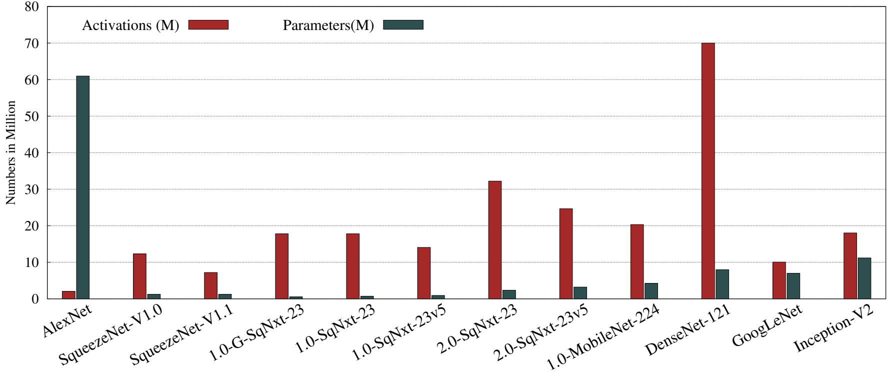
Does less number of parameters implies low memory footprint ?

Does less number of parameters implies low memory footprint ?

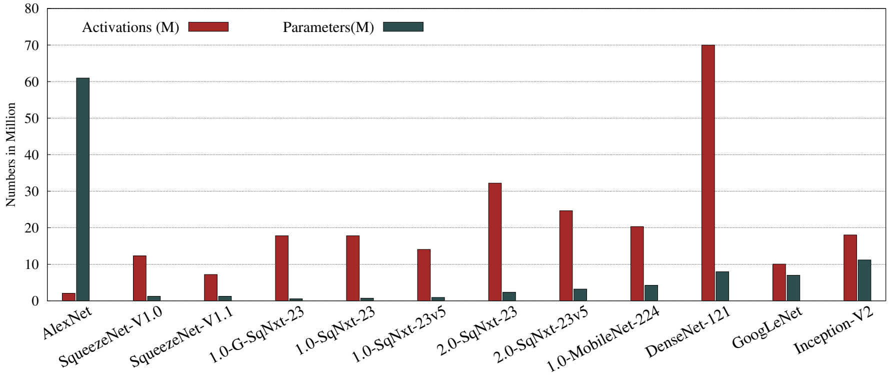
Model Name	#Param (M)	M-F (MB)	comp
AlexNet	60.97	**1015**	1×
SqueezeNet V1.0	1.25	615	51×
SqueezeNet V1.1	1.24	587	51×
1.0-G-SqNxt-23	0.54	**1019**	112×
1.0-SqNxt-23	0.72	885	84×
1.0-SqNxt-23v5	0.93	867	65×
2.0-SqNxt-23	2.36	995	26×
2.0-SqNxt-23v5	3.22	957	19×
1.0 MobileNet-224	4.23	733	14×
DenseNet-121	7.98	**1405**	8×
GoogLeNet	7.00	801	9×
InceptionV2	11.2	987	5×

M-F : Memory-footprint, comp : #Parameters compared to AlexNet.

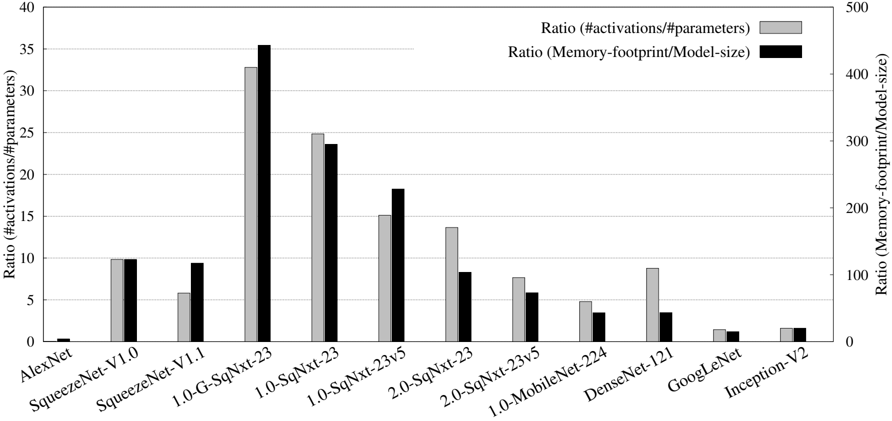
Compact DNNs with fewer parameters



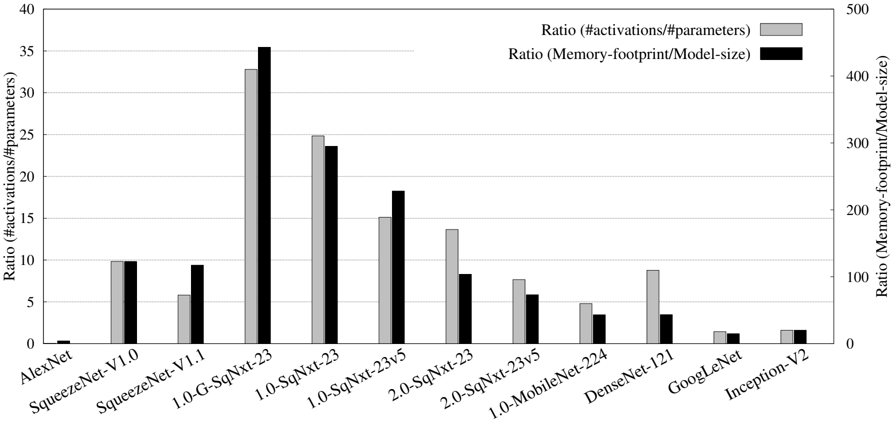
Compact DNNs with fewer parameters



X	Y	**PPMCC(X,Y)**
# Parameters	Memory footprint	0.24
# Activations	Memory footprint	**0.75**
# Parameters + # activations	Memory footprint	**0.82**



Ratio (#activations/#parameters)
Ratio (Memory-footprint/Model-size)
0
5
10
15
20
25
30
35
40
0
100
200
300
400
500
AlexNet
SqueezeNet-V1.0
SqueezeNet-V1.1
1.0-G-SqNxt-23
1.0-SqNxt-23
1.0-SqNxt-23v5
2.0-SqNxt-23
2.0-SqNxt-23v5
1.0-MobileNet-224
DenseNet-121
GoogLeNet
Inception-V2



PPMCC between "memory footprint to model size" ratio and "#activation to #parameters" ratio is 0.96

TL;DR : Memory footprint

"Higher number of activations implies high memory footprint and compact DNNs have large number of activations."

Compact DNNs : Energy Efficiency

$$E_e = \frac{performance}{watt} = \frac{FLOPS}{watt} = \frac{FLOPs}{joule}$$

Compact DNNs : Energy Efficiency

$$E_e = \frac{performance}{watt} = \frac{FLOPS}{watt} = \frac{FLOPs}{joule}$$

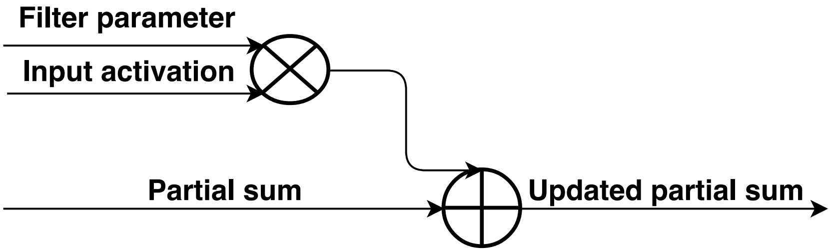
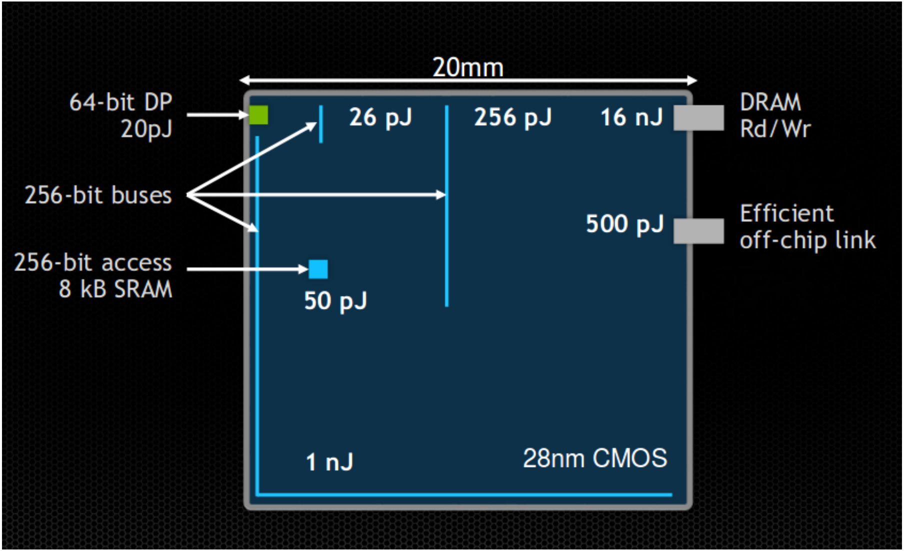


Figure: One MAC operation

Aside : What dominates energy consumption, computation or communication?

Aside : What dominates energy consumption, computation or communication?

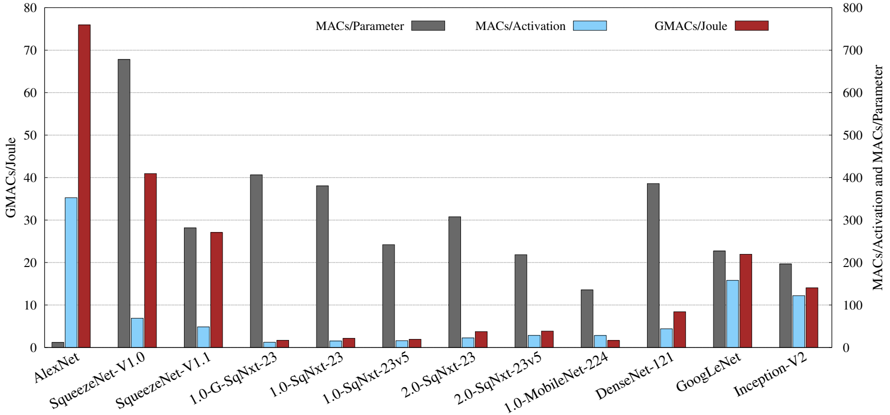


source : Bill Dally, NVIDIA

Data reuse and arithmetic intensity

Model Name	MACs/Param	MACs/Act
AlexNet	12	353
SqueezeNet V1.0	678	69
SqueezeNet V1.1	282	48
1.0-G-SqNxt-23	406	12
1.0-SqNxt-23	381	15
1.0-SqNxt-23v5	242	16
2.0-SqNxt-23	308	23
2.0-SqNxt-23v5	218	29
1.0 MobileNet-224	136	28
DenseNet-121	386	44
GoogLeNet	227	158
InceptionV2	197	122

Energy efficiency and data reuse



PPMCC (Energy efficiency): MACs/parameters = **-0.18**, MACs/activations = **0.88**

TL;DR : Energy efficiency

Compact DNNs lower the data reuse and arithmetic intensity which leads to increase in the bandwidth requirement and result into low energy efficiency

GPU profiling and kernel level analysis

GEMM $\Rightarrow$ dense computation, GEMV $\Rightarrow$ sparse computation

Model Name	Gemv2T(%)	Gemv2N(%)	Gemmk1(%)
AlexNet	7.54	0.18	11.43
SqueezeNet V1.0	0	0	0
SqueezeNet V1.1	0	0	0
1.0-G-SqNxt-23	40.92	5.41	8.64
1.0-SqNxt-23	42.13	5.7	9.35
1.0-SqNxt-23v5	31.68	6.82	10.73
2.0-SqNxt-23	35.10	5.06	7.99
2.0-SqNxt-23v5	24.71	6.53	8.61
1.0 MobileNet-224	**55.47**	**30.5**	0.68
DenseNet-121	10.04	0	5.34
GoogLeNet	0.22	0.16	0.05
InceptionV2	6.38	0.03	3.98

Resource utilization

Attributes/stall reasons	Gemv2T	Gemv2N	Gemmk1
Compute utilization (%)	15	8	5
Bandwidth utilization (%)	0.92	0.094	11
SM utilization	**Poor**	**Very poor**	**Excellent**
Memory dependency (%)	**44.3**	**3.6**	**54.4**
Instruction dependency (%)	**32**	**56.9**	**19**
Synchronization (%)	8.8	25	9.1
Others (%)	15	14.5	17.5

Resource utilization

Attributes/stall reasons	Gemv2T	Gemv2N	Gemmk1
Compute utilization (%)	15	8	5
Bandwidth utilization (%)	0.92	0.094	11
SM utilization	**Poor**	**Very poor**	**Excellent**
Memory dependency (%)	**44.3**	**3.6**	**54.4**
Instruction dependency (%)	**32**	**56.9**	**19**
Synchronization (%)	8.8	25	9.1
Others (%)	15	14.5	17.5

Lower percentage of `Gemv2T` and `Gemv2N` while that of higher `Gemmk1` is better for compute resource utilization.

Compact DNNs : Inference time and throughput

Model Name	#MACs (M)	I.t. (ms)	T.p(FPS)
AlexNet	723	2.1	4000
SqueezeNet V1.0	848	3.8	1479
SqueezeNet V1.1	349	3.5	2778
1.0-G-SqNxt-23	221	24.4	763
1.0-SqNxt-23	273	24.5	770
1.0-SqNxt-23v5	225	23.8	1004
2.0-SqNxt-23	726	28.2	412
2.0-SqNxt-23v5	703	27.7	541
1.0 MobileNet-224	574	29.4	**42**
DenseNet-121	3080	**33.0**	182
GoogLeNet	1590	11.2	1333
InceptionV2	2220	19.2	658

TL;DR : Throughput

Low SM utilization indicates inefficient use of compute resources by MACs. This lowers the throughput, even if number of MACs are less.

Conclusion

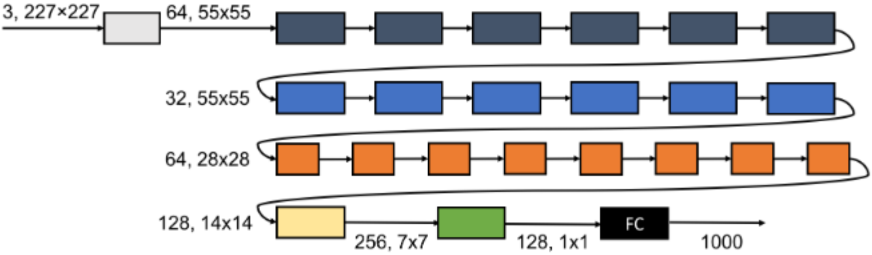
Compact DNNs are narrower which reduces the data reuse and increases bandwidth pressure hence MACs are energy inefficient.

- Reducing only the number of parameters is not sufficient to reduce the memory footprint.
  - Reduce number of activations too.
- Reducing MACs is not sufficient to get high energy efficiency and high throughput.
  - Increases data reuse and resource utilization also.

Thank You

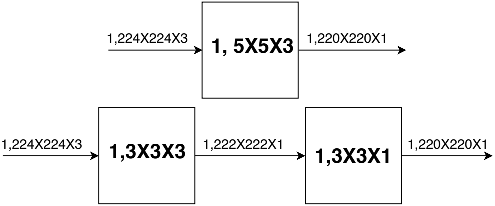
Thank You
Questions?

Additional slides : SqueezeNext architecture



[Gholami et al., CVPR-W'18]

Additional slides : filter factorization



Architecture	#param	#MACs	#Acts	M/P	M/A
factorization	52% ↓	51.3% ↓	102% ↑	1.34% ↑	**61%** ↓