Compact Network Design

Jinyang Guo（郭晋阳）
jinyangguo@buaa.edu.cn

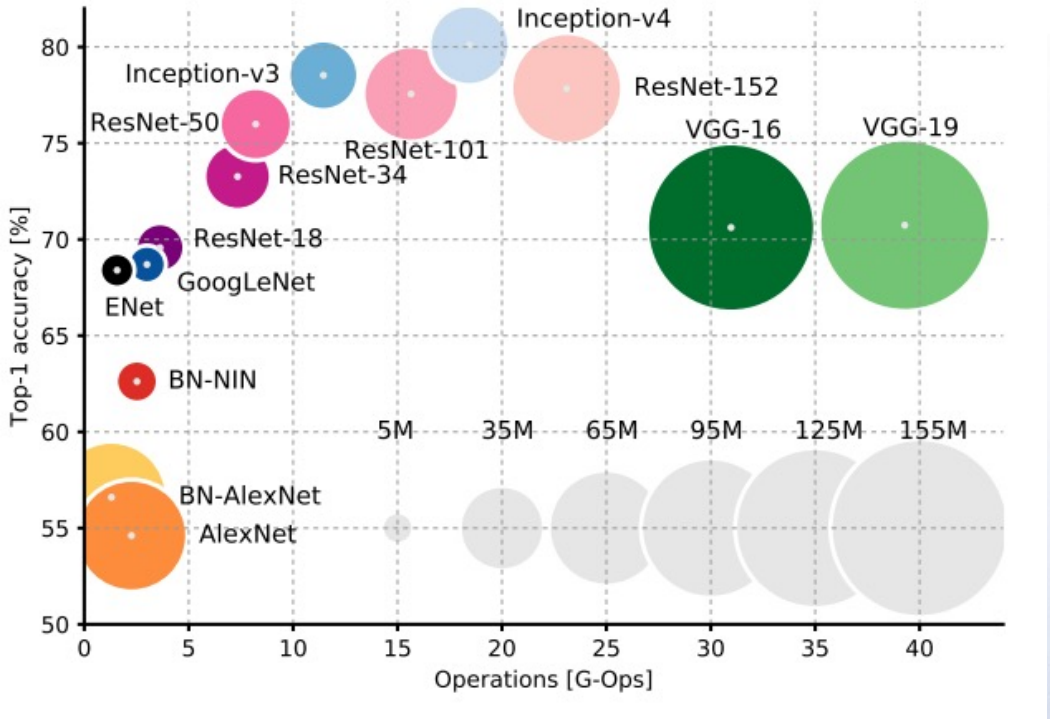
Last lecture

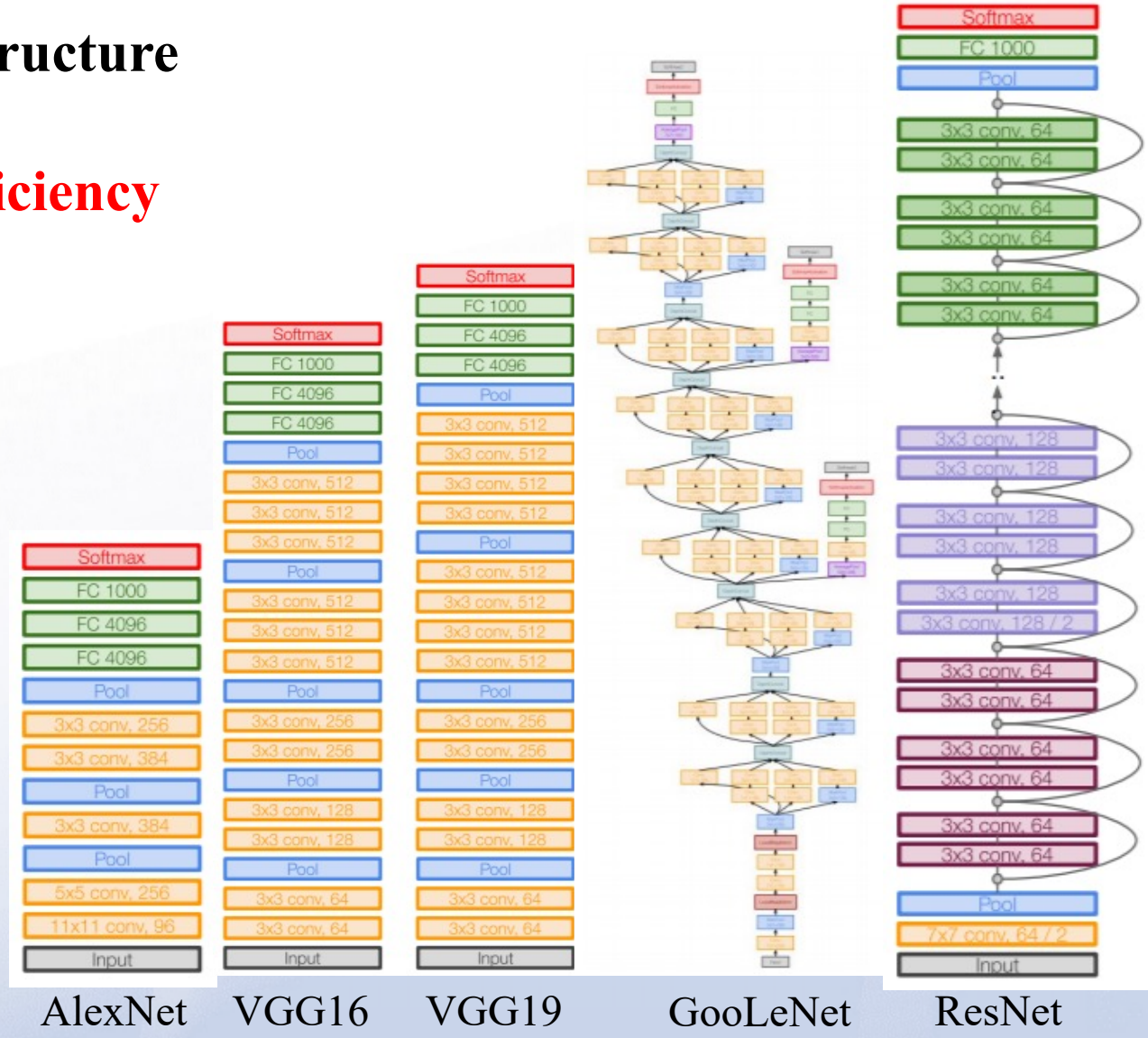
- Knowledge distillation
- Make network smaller

目录

Introduction

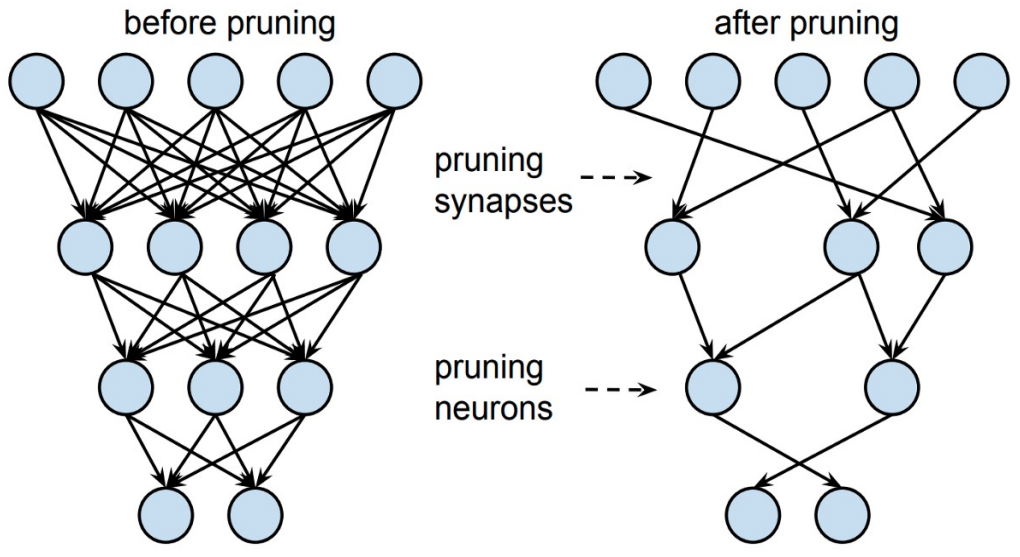
- Model size grows up quickly, and model structure becomes more complex
- Better performance but computation inefficiency and storage inefficiency





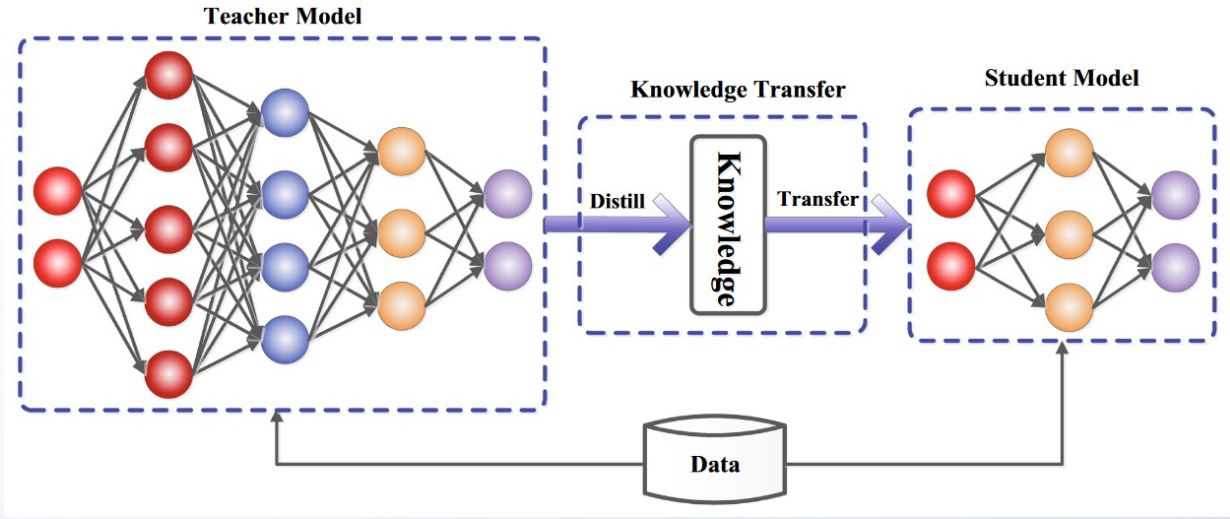
Introduction

➢Network Pruning



remove unimportant weights/neurons

➢Knowledge Distillation



transfer knowledge from a teacher model to a student model

Introduction

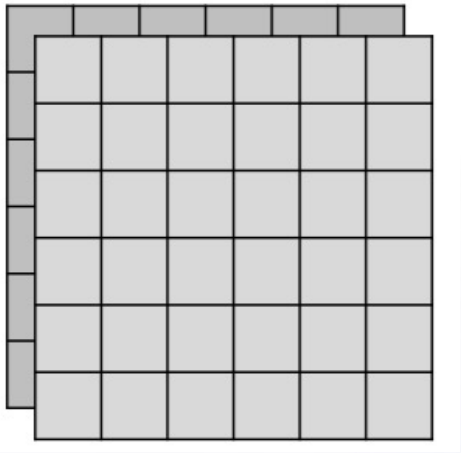
➢Compact Network Design

- ➢ a different model compression technique, used to **construct a lightweight network**
- ➢ **redesign the special structure** of filters, layers and networks
- ➢ e.g. smaller filter size
  depthwise convolution
  pointwise convolution
  group convolution
  …

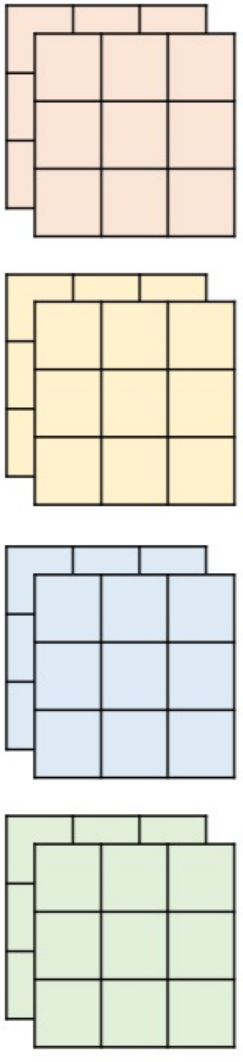
Introduction

➢ **Review: Standard Convolution**

Input feature map

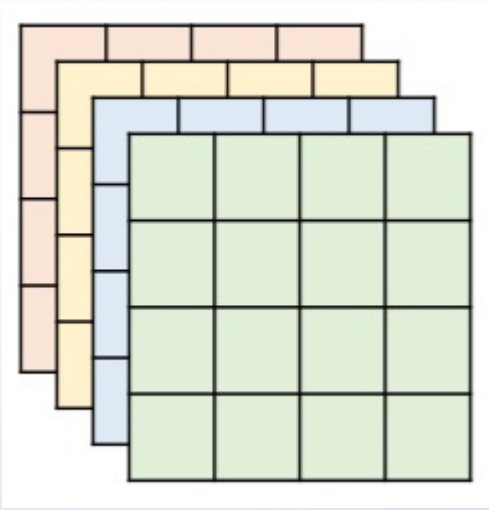


2 channels



Filters

$3 \times 3 \times 2 \times 4 = 72$
parameters



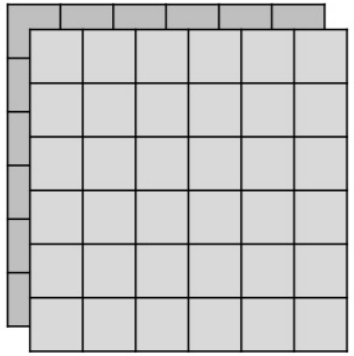
Output feature map
4 channels

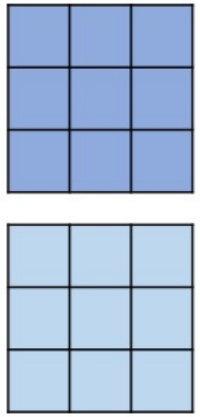
https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/tiny_v7.pdf

Introduction

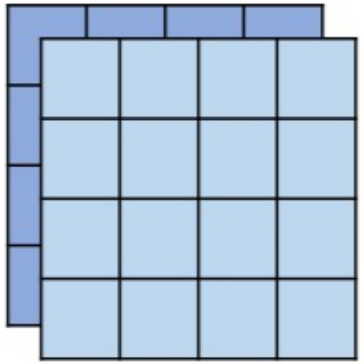
➢Depthwise Separable Convolution

1. Depthwise Convolution

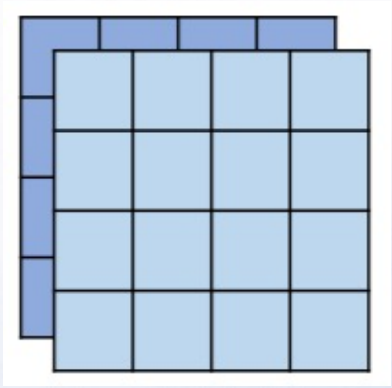


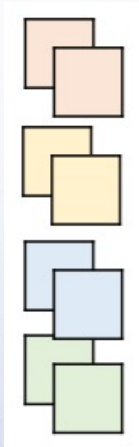


$3 \times 3 \times 2 = 18$ parameters



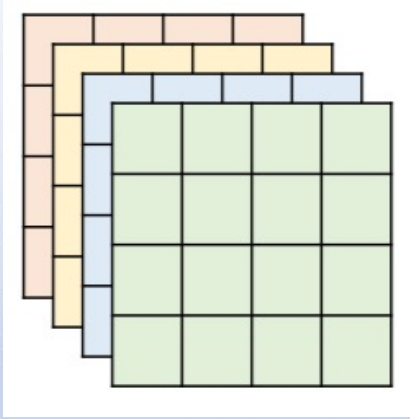
2. Pointwise Convolution





1×1 filter

$2 \times 4 = 8$ parameters



https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/tiny_v7.pdf

目录

Static compact network

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

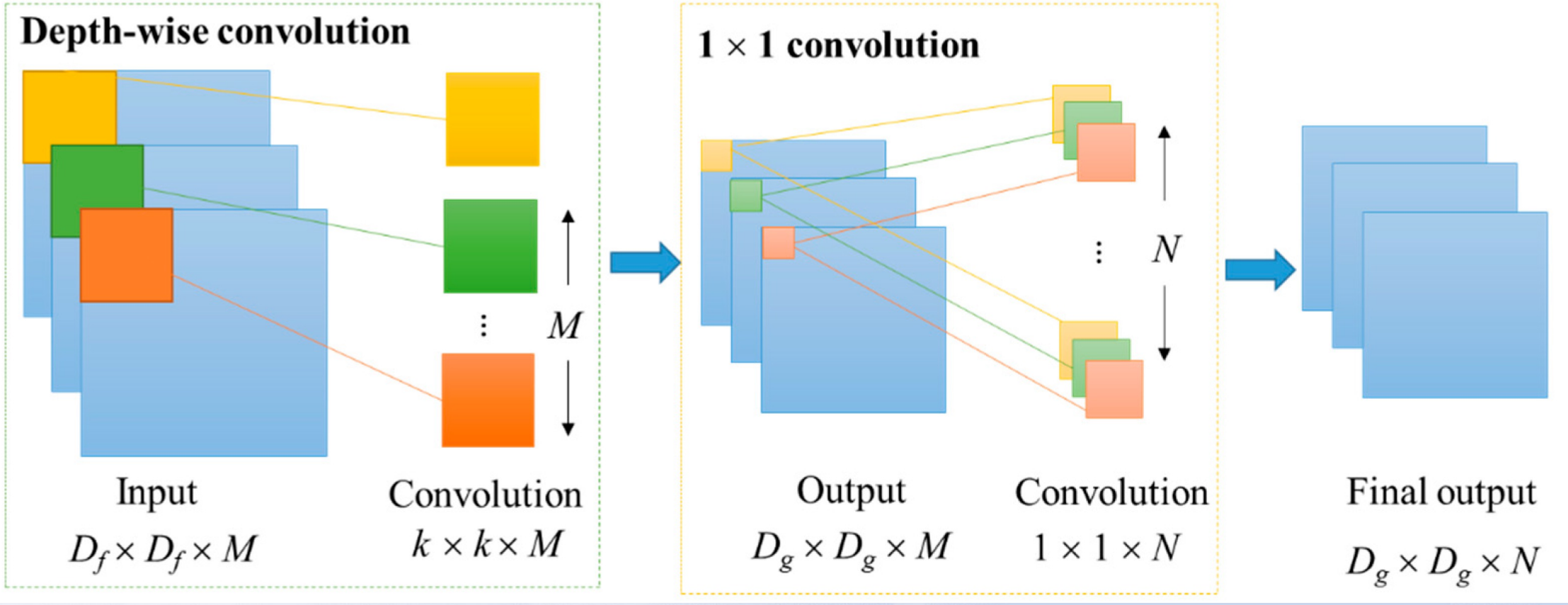
Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko
Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam

Google Inc.

{howarda,menglong,bochen,dkalenichenko,weijunw,weyand,anm,hadam}@google.com

Static compact network

- **Problem:** popular CNNs are becoming larger and more computationally expensive
- **Method:** introduce depthwise separable convolution



MobileNets: Efficient Convolution Neural Networks for Mobile Vision Applications, 2017

Static compact network

- **Result: higher accuracy than VGG, with 27× computation reduction and 32× parameters reduction**

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

higher accuracy than VGG-16

Table 13. COCO object detection results comparison using different frameworks and network architectures. mAP is reported with COCO primary challenge metric (AP at IoU=0.50:0.05:0.95)

Framework Resolution	Model	mAP	Billion Mult-Adds	Million Parameters
SSD 300	deeplab-VGG	21.1%	34.9	33.1
	Inception V2	22.0%	3.8	13.7
	MobileNet	19.3%	1.2	6.8
Faster-RCNN 300	VGG	22.9%	64.3	138.5
	Inception V2	15.4%	118.2	13.3
	MobileNet	16.4%	25.2	6.1
Faster-RCNN 600	VGG	25.7%	149.6	138.5
	Inception V2	21.9%	129.6	13.3
	Mobilenet	19.8%	30.5	6.1

also applicable to detection task

Static compact network

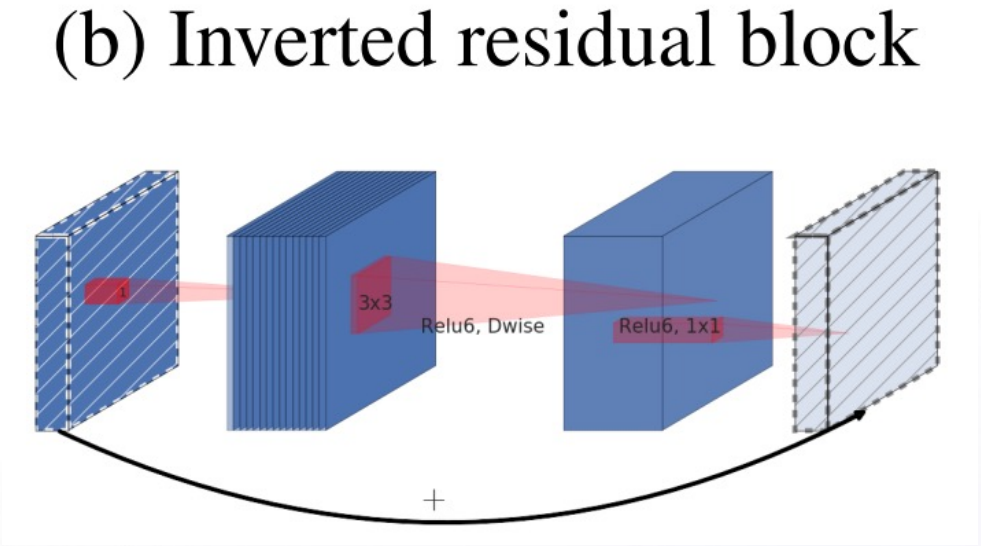
MobileNetV2: Inverted Residuals and Linear Bottlenecks

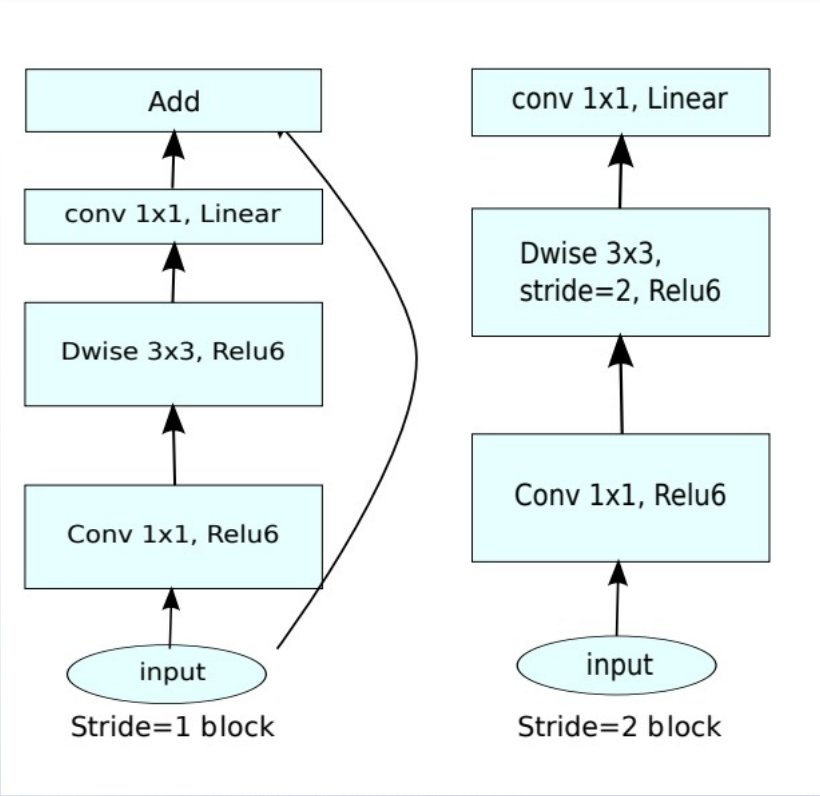
Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen
Google Inc.
{sandler, howarda, menglong, azhmogin, lcchen}@google.com

Mobilenetv2: Inverted residuals and linear bottlenecks, 2018

Static compact network

- **Problem:** existing neural networks require high computational resources
- **Method:** introduce **inverted residual with linear bottleneck**





The inverted residual structure first increases and then decreases the dimension of the input feature map, which is completely opposite to the operation process of the residual structure.

Mobilenetv2: Inverted residuals and linear bottlenecks, 2018

Static compact network

➢ **Result: higher accuracy and less computation than MobileNetV1**

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table 4: Performance on ImageNet, comparison for dif-

faster inference speed

Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Table 6: Performance comparison of MobileNetV2 + SSDLite and other realtime detectors on the COCO dataset object detection task. MobileNetV2 + SSDLite

also applicable to detection task

Mobilenetv2: Inverted residuals and linear bottlenecks, 2018

Static compact network

Searching for MobileNetV3

Andrew Howard[1] **Mark Sandler**[1] **Grace Chu**[1] **Liang-Chieh Chen**[1] **Bo Chen**[1] **Mingxing Tan**[2]
Weijun Wang[1] **Yukun Zhu**[1] **Ruoming Pang**[2] **Vijay Vasudevan**[2] **Quoc V. Le**[2] **Hartwig Adam**[1]
[1]Google AI, [2]Google Brain
{howarda, sandler, cxy, lcchen, bochen, tanmingxing, weijunw, yukun, rpang, vrv, qvl, hadam}@google.com

Searching for MobileNetV3, 2019

Static compact network

- **Problem:** the manual design of MobileNetV2 has limitations and does not address the requirements of specific hardware
- **Method:** use **platform-aware NAS** to search the optimal network structure, and propose a new activation function **h-wish**

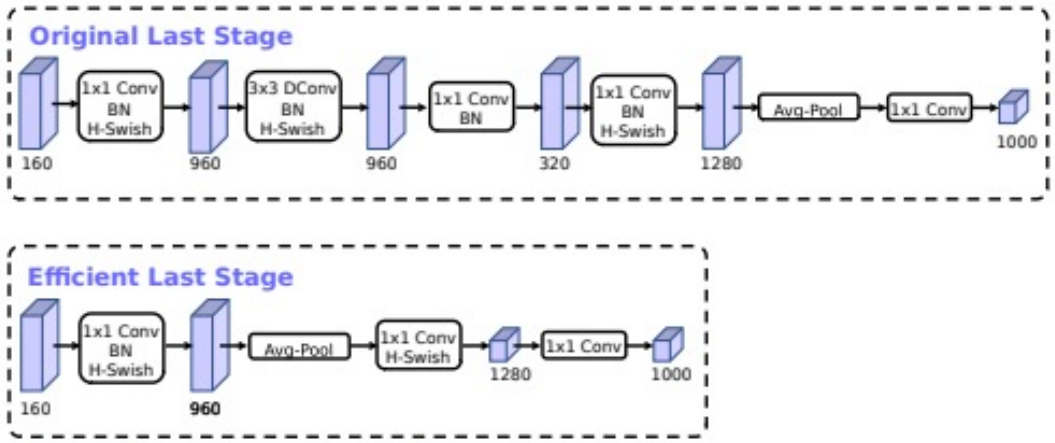
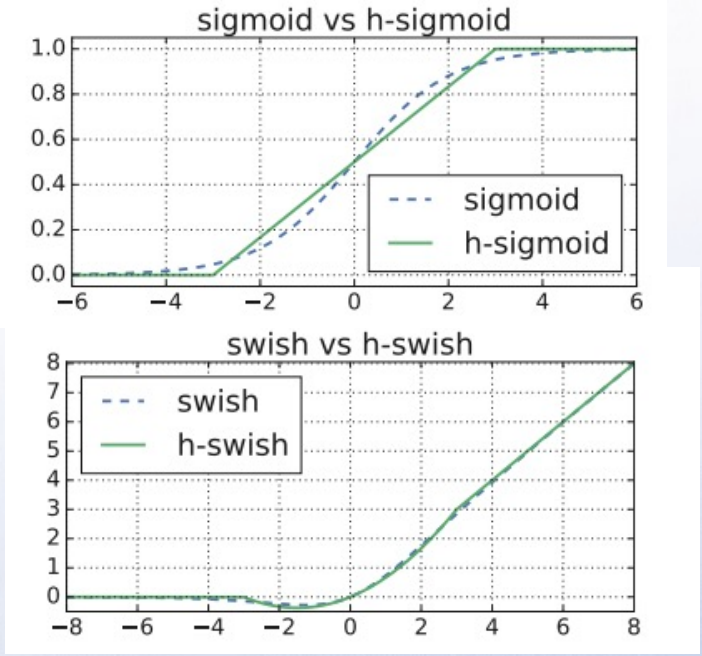


Figure 5. Comparison of original last stage and efficient last stage. This more efficient last stage is able to drop three expensive layers at the end of the network at no loss of accuracy.

get Original Last Stage through NAS, and then simplify it to get Efficient Last Stage



h-swish

Searching for MobileNetV3, 2019

Static compact network

➢ **Result: higher performance and lower latency**

Network	Top-1	MAdds	Params	P-1	P-2	P-3
V3-Large 1.0	**75.2**	**219**	5.4M	**51**	**61**	**44**
V3-Large 0.75	73.3	155	4.0M	39	46	40
MnasNet-A1	75.2	315	3.9M	71	86	61
Proxyless[5]	74.6	320	4.0M	72	84	60
V2 1.0	72.0	300	3.4M	64	76	56
V3-Small 1.0	**67.4**	**56**	2.5M	**15.8**	**19.4**	**14.4**
V3-Small 0.75	65.4	44	2.0M	12.8	15.6	11.7
Mnas-small [43]	64.9	65.1	1.9M	20.3	24.2	17.2
V2 0.35	60.8	59.2	1.6M	16.6	19.6	13.9

Table 3. Floating point performance on the Pixel family of phones (P-n denotes a Pixel-n phone). All latencies are in ms and are

achieves 3.2% higher accuracy on ImageNet compared to MobileNetV2 while reducing latency by 20%

Backbone	mAP	Latency (ms)	Params (M)	MAdds (B)
V1	22.2	228	5.1	1.3
V2	22.1	162	4.3	0.80
MnasNet	23.0	174	4.88	0.84
V3	22.0	137	4.97	0.62
V3†	22.0	119	3.22	0.51
V2 0.35	13.7	66	0.93	0.16
V2 0.5	16.6	79	1.54	0.27
MnasNet 0.35	15.6	68	1.02	0.18
MnasNet 0.5	18.5	85	1.68	0.29
V3-Small	16.0	52	2.49	0.21
V3-Small†	16.1	43	1.77	0.16

Table 6. Object detection results of SSDLite with different backbones on COCO test set. †: Channels in the blocks between $C4$

Object detection results

Static compact network

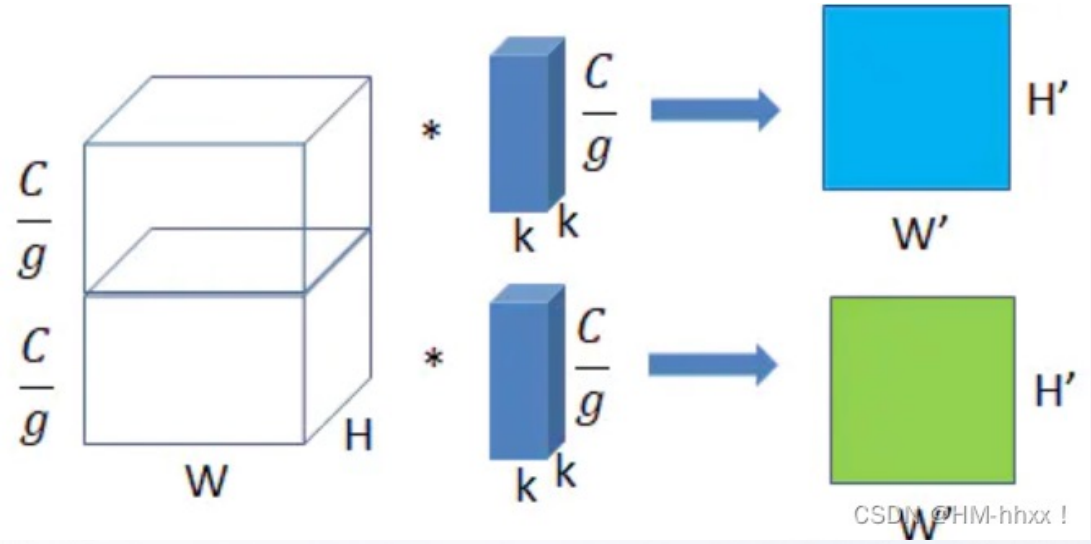
ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

Xiangyu Zhang* Xinyu Zhou* Mengxiao Lin Jian Sun
Megvii Inc (Face++)
{zhangxiangyu,zxy,linmengxiao,sunjian}@megvii.com

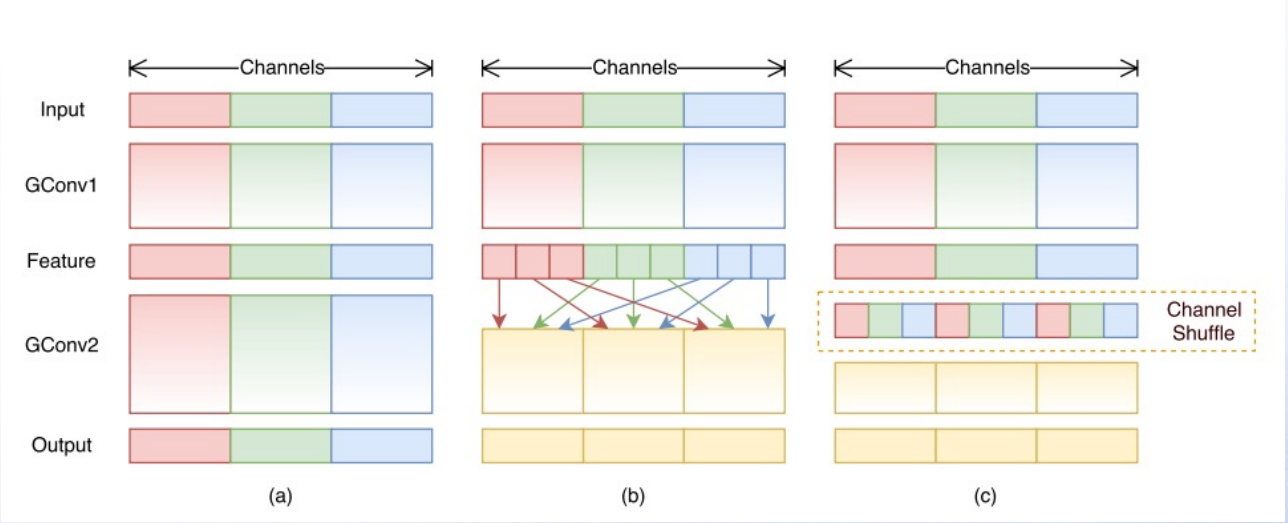
ShuffleNet: An extremely efficient convolutional neural network for mobile devices, 2018

Static compact network

- **Problem:** deep separable convolution doesn't consider the connections between channels
- **Method:** utilize pointwise group convolution and channel shuffle



Group convolution



divide the channels in each group into several subgroups, and then implement a channel shuffle operation

ShuffleNet: An extremely efficient convolutional neural network for mobile devices, 2018

Static compact network

➢ **Result: significant acceleration while maintaining the accuracy**

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**26.3**	3.1
ShuffleNet 2× (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× ($g = 4$)	38	**41.6**	7.8
ShuffleNet 0.5× (shallow, $g = 3$)	40	42.8	6.6

Table 5. ShuffleNet vs. MobileNet [12] on ImageNet Classification

achieves ~13× actual speedup over AlexNet

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**26.3**	3.1
ShuffleNet 2× (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× ($g = 4$)	38	**41.6**	7.8
ShuffleNet 0.5× (shallow, $g = 3$)	40	42.8	6.6

Table 5. ShuffleNet vs. MobileNet [12] on ImageNet Classification

Model	mAP [.5, .95] (300× image)	mAP [.5, .95] (600× image)
ShuffleNet 2× ($g = 3$)	**18.7%**	**25.0%**
ShuffleNet 1× ($g = 3$)	14.5%	19.8%
1.0 MobileNet-224 [12]	16.4%	19.8%
1.0 MobileNet-224 (our impl.)	14.9%	19.3%

Table 7. Object detection results on MS COCO (*larger numbers represents better performance*). For MobileNets we compare two results: 1) COCO detection scores reported by [12]; 2) finetuning from our reimplemented MobileNets, whose training and finetuning settings are exactly the same as that for ShuffleNets.

Better performance on both classification and detection tasks

Static compact network

ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design

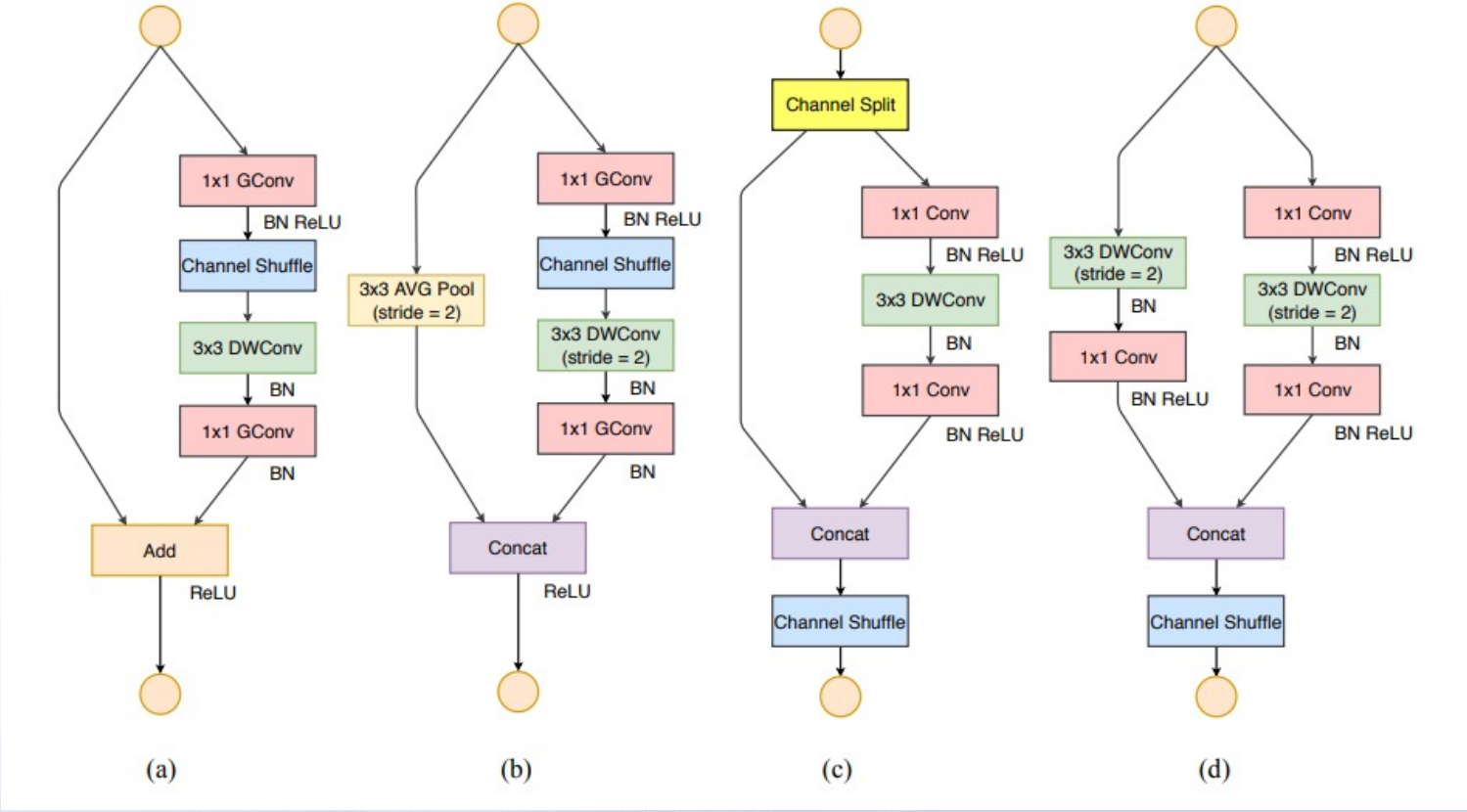
Ningning Ma[⋆1,2][0000−0003−4628−8831], Xiangyu Zhang[⋆1][0000−0003−2138−4608], Hai-Tao Zheng[2][0000−0001−5128−5649], and Jian Sun[1][0000−0002−6178−4166]

[1] Megvii Inc (Face++)
{maningning,zhangxiangyu,sunjian}@megvii.com
[2] Tsinghua University
zheng.haitao@sz.tsinghua.edu.cn

Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018

Static compact network

- **Problem:** existing neural network architecture design is mostly guided by FLOPs, ignoring the actual speed and efficiency
- **Method:** propose 4 principles: equivalent macro operations, appropriate number of channels, multipath, and reasonable complexity



Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018

Static compact network

➢ **Result: higher performance and lower latency**

Model	mmAP(%)				GPU Speed (Images/sec.)			
FLOPs	40M	140M	300M	500M	40M	140M	300M	500M
Xception	21.9	29.0	31.3	32.9	178	131	101	83
ShuffleNet v1	20.9	27.0	29.9	32.9	152	85	76	60
MobileNet v2	20.7	24.4	30.0	30.6	146	111	94	72
ShuffleNet v2 (ours)	22.5	29.0	31.8	33.3	**188**	**146**	**109**	**87**
ShuffleNet v2* (ours)	**23.7**	**29.6**	**32.2**	**34.2**	183	138	105	83

Table 7: Performance on COCO object detection. The input image size is 800 × 1200.

significant improvement on detection task

Model	Complexity (MFLOPs)	Top-1 err. (%)	GPU Speed (Batches/sec.)	ARM Speed (Images/sec.)
ShuffleNet v2 0.5× (ours)	41	**39.7**	417	**57.0**
0.25 MobileNet v1 [13]	41	49.4	**502**	36.4
0.4 MobileNet v2 [14] (our impl.)*	43	43.4	333	33.2
0.15 MobileNet v2 [14] (our impl.)	39	55.1	351	33.6
ShuffleNet v1 0.5× (g=3) [15]	38	43.2	347	56.8
DenseNet 0.5× [6] (our impl.)	42	58.6	366	39.7
Xception 0.5× [12] (our impl.)	40	44.9	384	52.9
IGCV2-0.25 [27]	46	45.1	183	31.5
ShuffleNet v2 1× (ours)	146	**30.6**	341	**24.4**
0.5 MobileNet v1 [13]	149	36.3	**382**	16.5
0.75 MobileNet v2 [14] (our impl.)**	145	32.1	235	15.9
0.6 MobileNet v2 [14] (our impl.)	141	33.3	249	14.9
ShuffleNet v1 1× (g=3) [15]	140	32.6	213	21.8
DenseNet 1× [6] (our impl.)	142	45.2	279	15.8
Xception 1× [12] (our impl.)	145	34.1	278	19.5
IGCV2-0.5 [27]	156	34.5	132	15.5
IGCV3-D (0.7) [28]	210	31.5	143	11.7
ShuffleNet v2 1.5× (ours)	299	**27.4**	255	**11.8**
0.75 MobileNet v1 [13]	325	31.6	**314**	10.6
1.0 MobileNet v2 [14]	300	28.0	180	8.9
1.0 MobileNet v2 [14] (our impl.)	301	28.3	180	8.9
ShuffleNet v1 1.5× (g=3) [15]	292	28.5	164	10.3
DenseNet 1.5× [6] (our impl.)	295	39.9	274	9.7
CondenseNet (G=C=8) [16]	274	29.0	-	-
Xception 1.5× [12] (our impl.)	305	29.4	219	10.5
IGCV3-D [28]	318	27.8	102	6.3

40% faster than ShuffleNet v1
16% faster than MobileNet v2

Static compact network

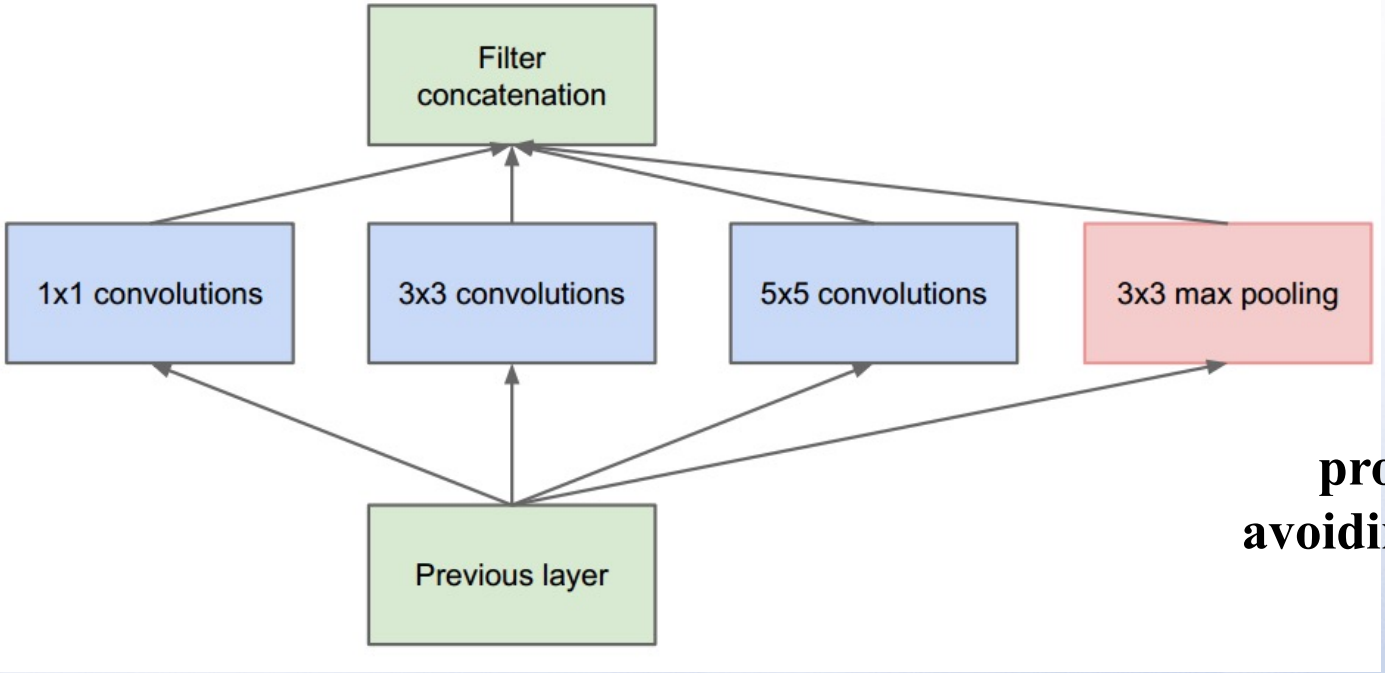
Going Deeper with Convolutions

Christian Szegedy[1], Wei Liu[2], Yangqing Jia[1], Pierre Sermanet[1], Scott Reed[3],
Dragomir Anguelov[1], Dumitru Erhan[1], Vincent Vanhoucke[1], Andrew Rabinovich[4]
[1]Google Inc. [2]University of North Carolina, Chapel Hill
[3]University of Michigan, Ann Arbor [4]Magic Leap Inc.
[1]{szegedy,jiayq,sermanet,dragomir,dumitru,vanhoucke}@google.com
[2]wliu@cs.unc.edu, [3]reedscott@umich.edu, [4]arabinovich@magicleap.com

Going deeper with convolutions, 2015

Static compact network

- **Problem:** traditional deep networks improve performance by increasing the number of layers, but this increases the risk of overfitting and requires more computational resources
- **Method:** propose **Inception** module, which performs multi-scale convolution in parallel and combine the



providing multi-scale features while avoiding significant computational burden

Going deeper with convolutions, 2015

Static compact network

➢ **Result: first place in the ILSVRC-2014**

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Table 2: Classification performance

Number of models	Number of Crops	Cost	Top-5 error	compared to base
1	1	1	10.07%	base
1	10	10	9.15%	-0.92%
1	144	144	7.89%	-2.18%
7	1	7	8.09%	-1.98%
7	10	70	7.62%	-2.45%
7	144	1008	6.67%	-3.45%

Table 3: GoogLeNet classification performance break down

Team	Year	Place	mAP	external data	ensemble	approach
UvA-Euvision	2013	1st	22.6%	none	?	Fisher vectors
Deep Insight	2014	3rd	40.5%	ImageNet 1k	3	CNN
CUHK DeepID-Net	2014	2nd	40.7%	ImageNet 1k	?	CNN
GoogLeNet	2014	1st	43.9%	ImageNet 1k	6	CNN

Table 4: Detection performance

Team	mAP	Contextual model	Bounding box regression
Trimps-Soushen	31.6%	no	?
Berkeley Vision	34.5%	no	yes
UvA-Euvision	35.4%	?	?
CUHK DeepID-Net2	37.7%	no	?
GoogLeNet	38.02%	no	no
Deep Insight	40.2%	yes	yes

Table 5: Single model performance for detection

Static compact network

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

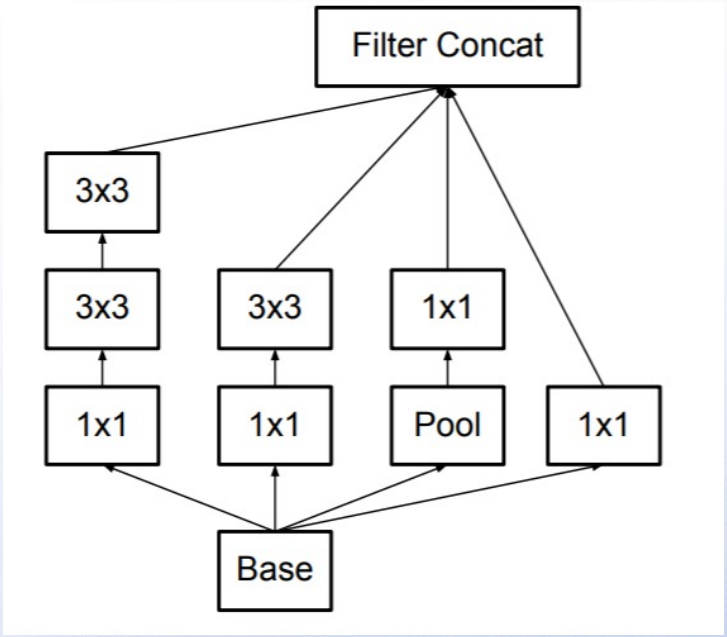
Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015

Static compact network

- **Problem:** as the depth of the network increases, the vanishing or exploding gradients will affect the training stability of the network
- **Method:** introduce **Batch Normalization** into Inception module, and change the network architecture

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

BN layer, which avoids gradients vanishing, accelerates convergence and saving training resources



replace **5x5** convolution by two continuous **3x3** convolution to reduce the computation

Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015

Static compact network

➢ **Result:** **reduce the training time and improve the performance**

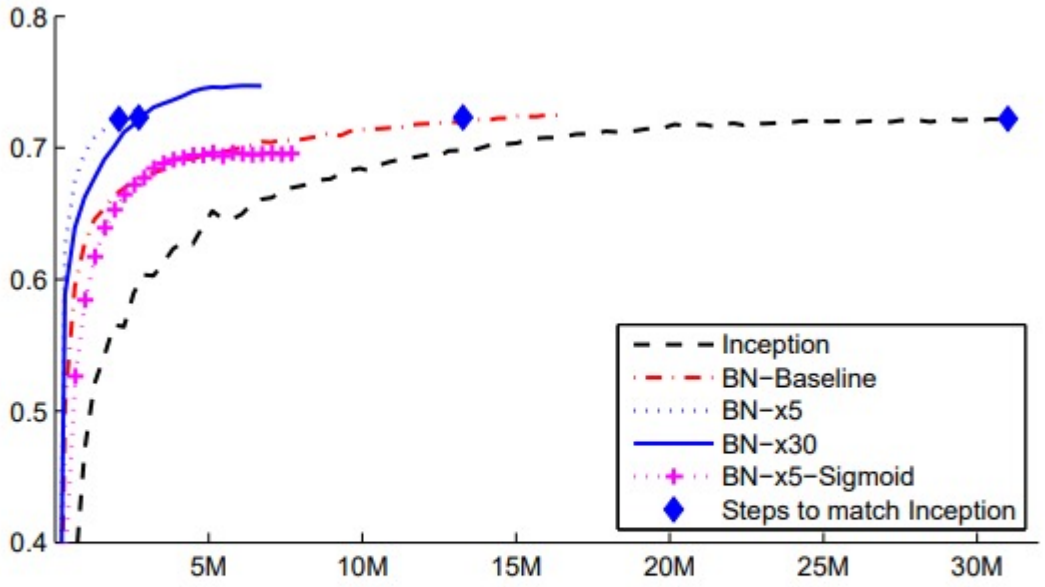


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

5× speed up convergence, higher accuracy

Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015

Static compact network

Rethinking the Inception Architecture for Computer Vision

Christian Szegedy
Google Inc.
szegedy@google.com

Vincent Vanhoucke
vanhoucke@google.com

Sergey Ioffe
sioffe@google.com

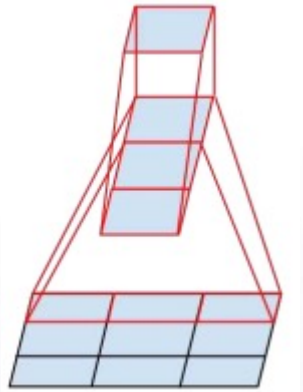
Jon Shlens
shlens@google.com

Zbigniew Wojna
University College London
zbigniewwojna@gmail.com

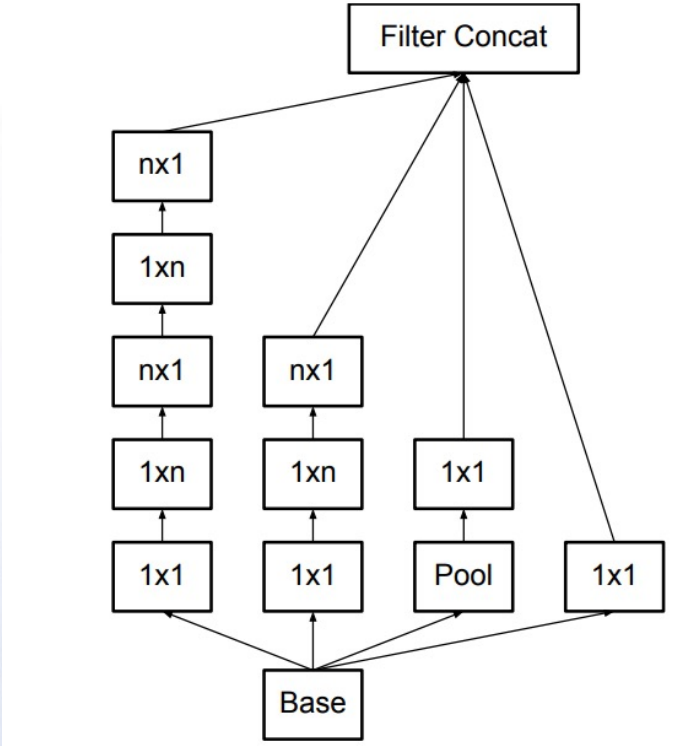
Rethinking the inception architecture for computer vision, 2016

Static compact network

- **Problem:** the model may still not capture enough information or features in deeper representations
- **Method:** further optimize and expand the Inception module, introduce **Factorization**



filter factorization



factorize a 7x7 convolution into a 1x7 convolution and a 7x1 convolution

Rethinking the inception architecture for computer vision, 2016

Static compact network

➢ **Result:** significant reduction in error

Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

Table 4. Single-model, multi-crop experimental results comparing the cumulative effects on the various contributing factors. We compare our numbers with the best published single-model inference results on the ILSVRC 2012 classification benchmark.

2.6% performance improvement on ImageNet compared to VGG

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet [18]	2	-	23.7%	6.8%
GoogLeNet [20]	7	144	-	6.67%
PReLU [6]	-	-	-	4.94%
BN-Inception [7]	6	144	20.1%	4.9%
Inception-v3	4	144	**17.2%**	**3.58%***

Table 5. Ensemble evaluation results comparing multi-model, multi-crop reported results. Our numbers are compared with the best published ensemble inference results on the ILSVRC 2012 classification benchmark. *All results, but the top-5 ensemble result reported are on the validation set. The ensemble yielded 3.46% top-5 error on the validation set.

further improve performance through ensemble

Static compact network

Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

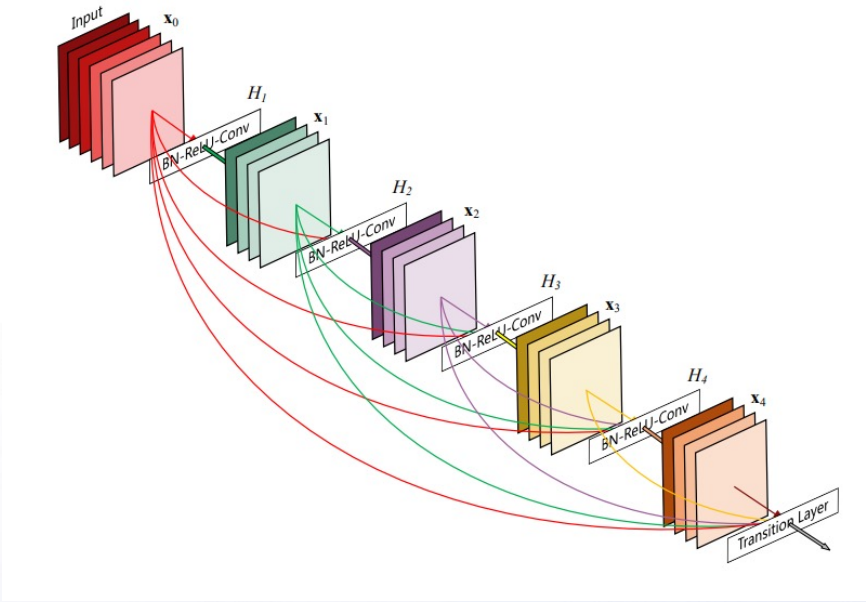
Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu

Densely connected convolutional networks, 2017

Static compact network

- **Problem:** as convolutional networks become deeper, information and gradients can vanish, hindering optimization
- **Method:** introduce **dense connections**, which better utilize shallow features and save network parameters



each layer takes **all preceding feature maps** as input

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

network structure: composed of DenseBlock and Transition

Densely connected convolutional networks, 2017

Static compact network

➢ **Result:** outperforms ResNets significantly

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

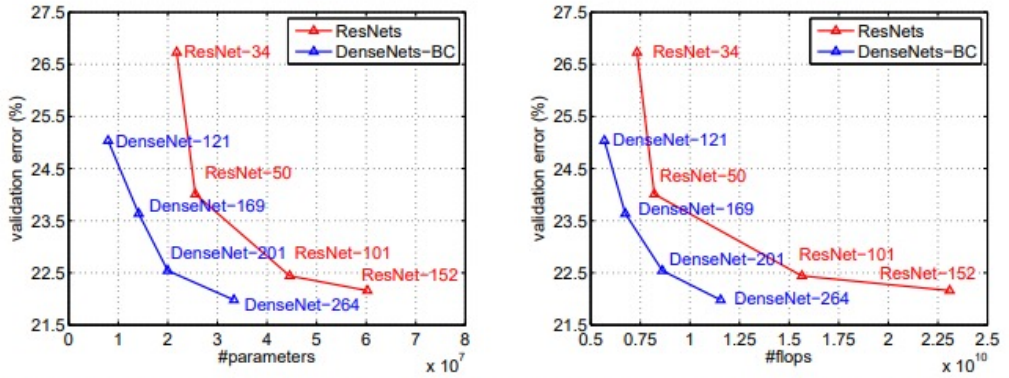


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

significant improvements on ImageNet

Static compact network

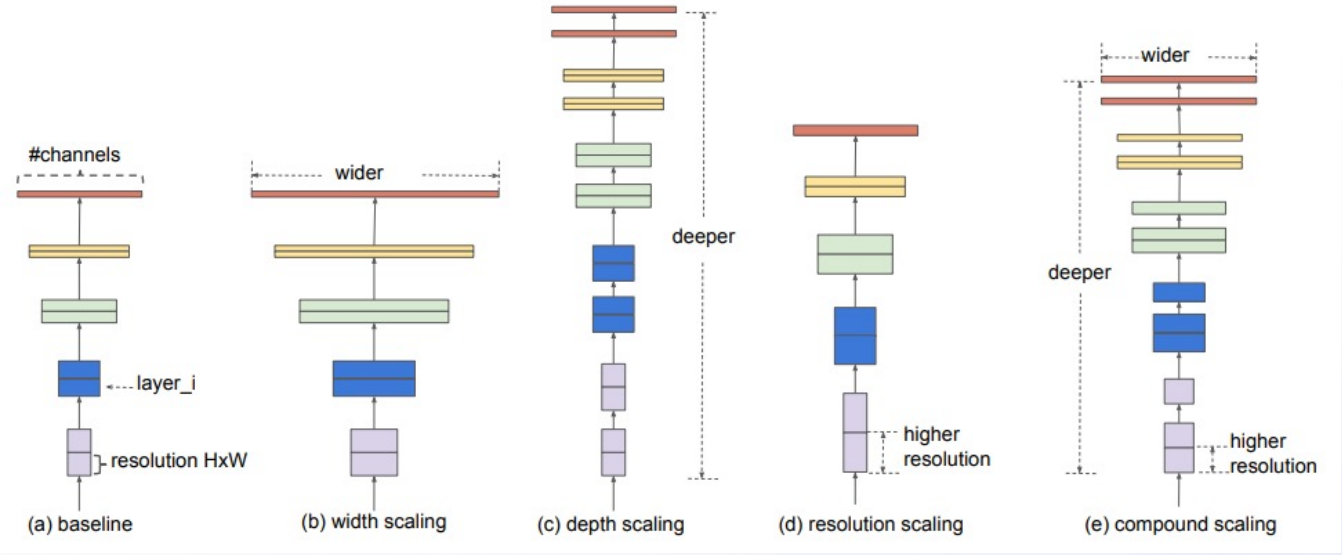
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan [1] **Quoc V. Le** [1]

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2019

Static compact network

- **Problem:** is there a principled method to scale up ConvNets that can achieve better accuracy and efficiency
- **Method:** propose a **compound scaling method** and use neural architecture search to design a new baseline network **EfficientNet**



Model Scaling

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

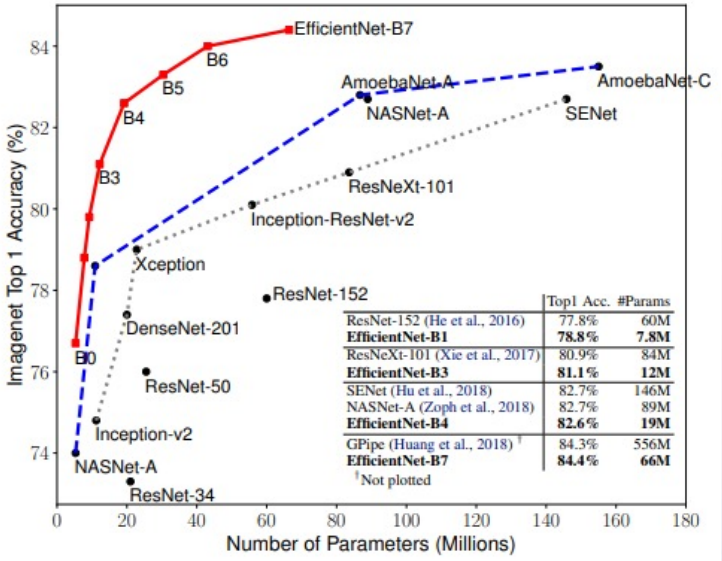
EfficientNet-B0 baseline network

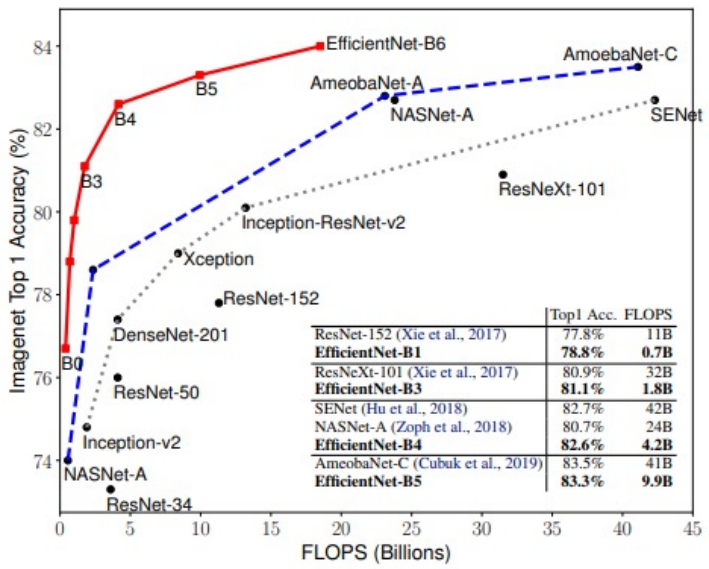
Observation 1: Scaling up any dimension of network width, depth, or resolution improves accuracy, but the accuracy gain diminishes for bigger models. **Observation 2:** In order to pursue better accuracy and efficiency, it is critical to balance all dimensions of network width, depth, and resolution during ConvNet scaling.

Static compact network

➢ **Result: significantly outperform other ConvNets**

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**78.8%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**79.8%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.1%**	**95.5%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-





achieves state-of-the-art **84.4%** top-1 accuracy on ImageNet, while being **8.4x** smaller and **6.1x** faster on inference than the best existing ConvNet

Static compact network

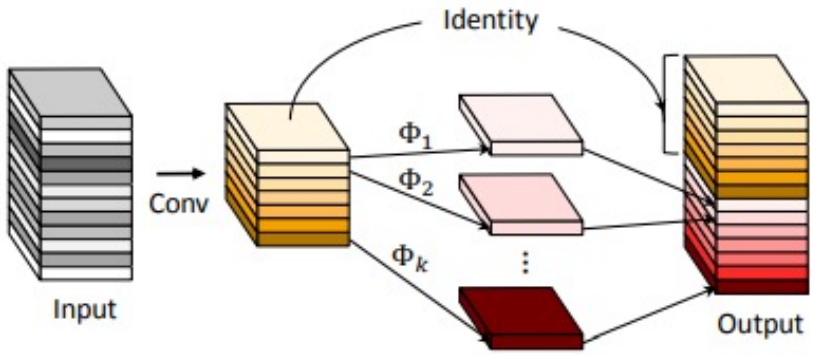
GhostNet: More Features from Cheap Operations

Kai Han[1] Yunhe Wang[1] Qi Tian[1*] Jianyuan Guo[2] Chunjing Xu[1] Chang Xu[3]
[1]Noah's Ark Lab, Huawei Technologies. [2]Peking University.
[3]School of Computer Science, Faculty of Engineering, University of Sydney.
{kai.han,yunhe.wang,tian.qi1,xuchunjing}@huawei.com jyguo@pku.edu.cn c.xu@sydney.edu.au

GhostNet: More Features from Cheap Operations, 2020

Static compact network

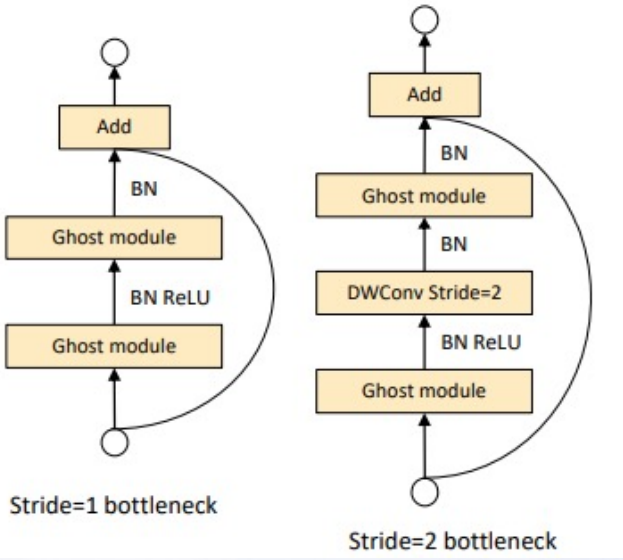
- **Problem:** can we accept the redundancy in feature maps instead of avoiding it
- **Method:** propose a **Ghost module** to generate more feature maps from **cheap operations**



(b) The Ghost module.

Ghost module

apply a series of linear transformations with cheap cost to generate many ghost feature maps based on a set of intrinsic feature maps



Ghost bottleneck

Input	Operator	#exp	#out	SE	Stride
$224^2 \times 3$	Conv2d 3×3	-	16	-	2
$112^2 \times 16$	G-bneck	16	16	-	1
$112^2 \times 16$	G-bneck	48	24	-	2
$56^2 \times 24$	G-bneck	72	24	-	1
$56^2 \times 24$	G-bneck	72	40	1	2
$28^2 \times 40$	G-bneck	120	40	1	1
$28^2 \times 40$	G-bneck	240	80	-	2
$14^2 \times 80$	G-bneck	200	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	184	80	-	1
$14^2 \times 80$	G-bneck	480	112	1	1
$14^2 \times 112$	G-bneck	672	112	1	1
$14^2 \times 112$	G-bneck	672	160	1	2
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	G-bneck	960	160	-	1
$7^2 \times 160$	G-bneck	960	160	1	1
$7^2 \times 160$	Conv2d 1×1	-	960	-	1
$7^2 \times 960$	AvgPool 7×7	-	-	-	-
$1^2 \times 960$	Conv2d 1×1	-	1280	-	1
$1^2 \times 1280$	FC	-	1000	-	-

GhostNet

Static compact network

➢ Result

Table 5. Comparison of state-of-the-art methods for compressing VGG-16 and ResNet-56 on CIFAR-10. - represents no reported results available.

Model	Weights	FLOPs	Acc. (%)
VGG-16	15M	313M	93.6
ℓ_1-VGG-16 [31, 37]	5.4M	206M	93.4
SBP-VGG-16 [18]	-	136M	92.5
Ghost-VGG-16 (s=2)	7.7M	158M	**93.7**
ResNet-56	0.85M	125M	93.0
CP-ResNet-56 [18]	-	63M	92.0
ℓ_1-ResNet-56 [31, 37]	0.73M	91M	92.5
AMC-ResNet-56 [17]	-	63M	91.9
Ghost-ResNet-56 (s=2)	0.43M	63M	**92.7**

Table 6. Comparison of state-of-the-art methods for compressing ResNet-50 on ImageNet dataset.

Model	Weights (M)	FLOPs (B)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNet-50 [16]	25.6	4.1	75.3	92.2
Thinet-ResNet-50 [39]	16.9	2.6	72.1	90.3
NISP-ResNet-50-B [59]	14.4	2.3	-	90.8
Versatile-ResNet-50 [49]	11.0	3.0	74.5	91.8
SSS-ResNet-50 [23]	-	2.8	74.2	91.9
Ghost-ResNet-50 (s=2)	13.0	2.2	**75.0**	**92.3**
Shift-ResNet-50 [53]	6.0	-	70.6	90.1
Taylor-FO-BN-ResNet-50 [41]	7.9	1.3	71.7	-
Slimmable-ResNet-50 0.5× [58]	6.9	1.1	72.1	-
MetaPruning-ResNet-50 [36]	-	1.0	73.4	-
Ghost-ResNet-50 (s=4)	6.5	1.2	**74.1**	**91.9**

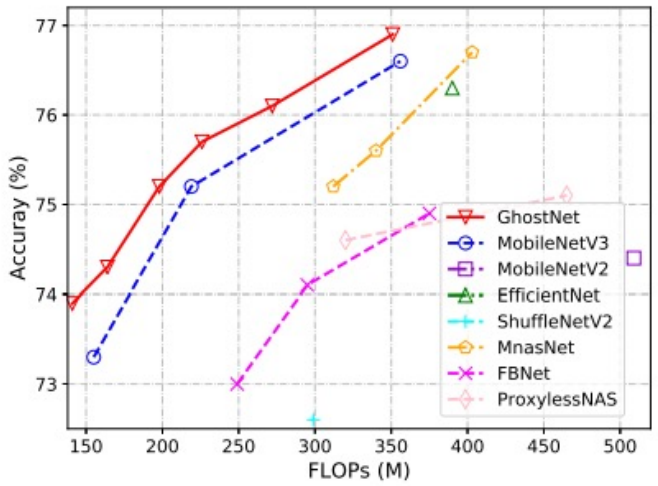


Figure 6. Top-1 accuracy *v.s.* FLOPs on ImageNet dataset.

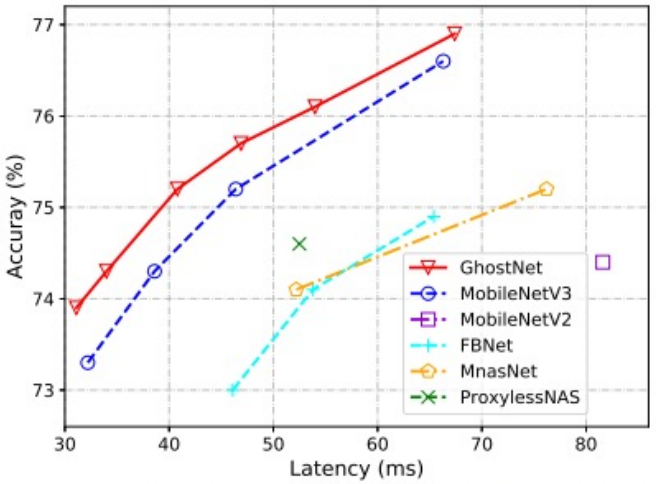


Figure 7. Top-1 accuracy *v.s.* latency on ImageNet dataset.

GhostNet can achieve **higher recognition performance** than MobileNetV3 with **similar computational cost** on the ImageNet dataset.

Ghost module can be taken as a **plug-and-play** component to compress existing convolutional neural networks.

Static compact network

GhostNetV2: Enhance Cheap Operation with Long-Range Attention

Yehui Tang[1,2], **Kai Han**[2], **Jianyuan Guo**[2,3], **Chang Xu**[3], **Chao Xu**[1], **Yunhe Wang**[2*]
[1]School of Artificial Intelligence, Peking University [2]Huawei Noah's Ark Lab
[3]School of Computer Science, University of Sydney
yhtang@pku.edu.cn, {kai.han, yunhe.wang}@huawei.com

GhostNetV2: Enhance Cheap Operation with Long-Range Attention, 2022

Static compact network

- **Problem:** GhostNet has weak ability to capture spatial information which prevents performance from being further improved
- **Method:** propose **DFC attention** to capture the long-range spatial information and then construct **GhostNetV2**

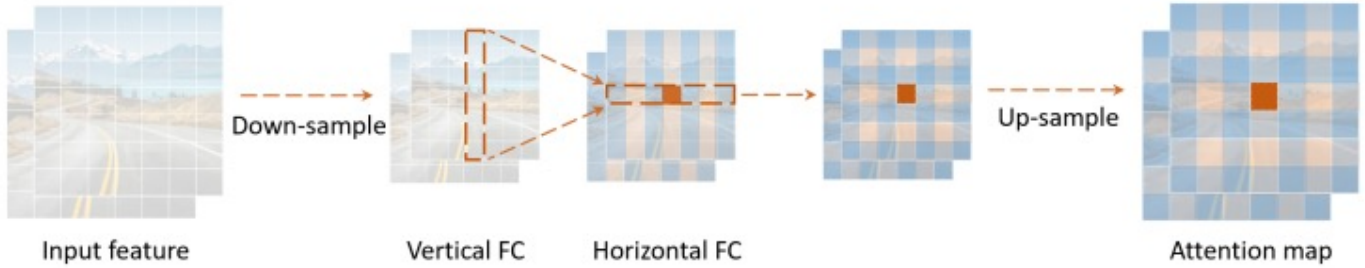
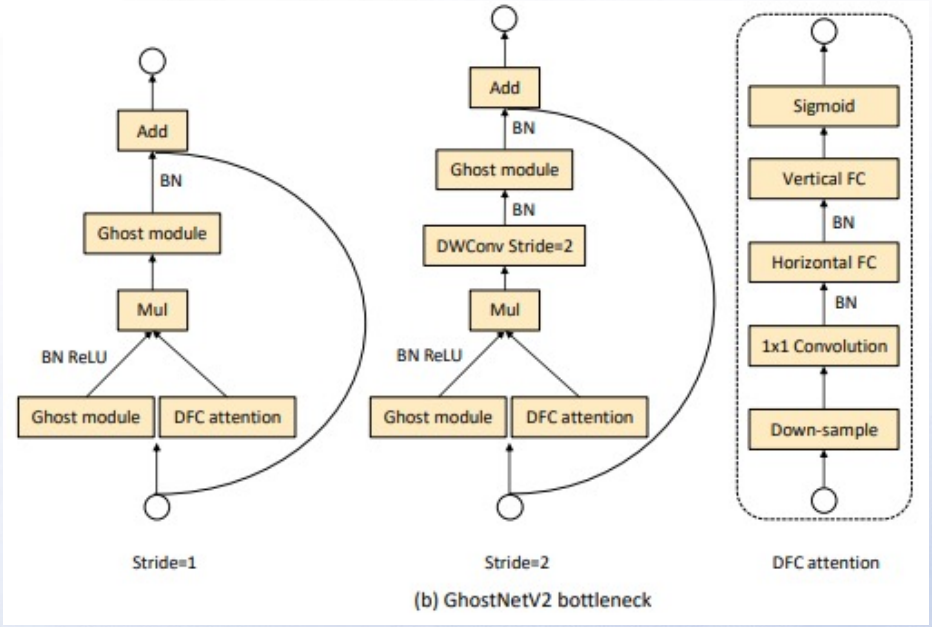


Figure 3: The information flow of DFC attention. The horizontal and vertical FC layers capture the long-range information along the two directions, respectively.

Only **fully connected (FC) layers** participate in generating the attention maps for simplicity.



use DFC attention to enhance the expanded features to improve expressiveness ability

Static compact network

➢ **Result: achieves a better trade-off between accuracy and speed**

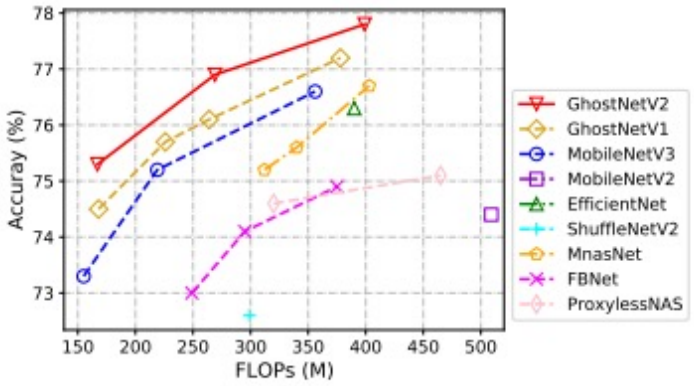


Figure 1: Top-1 accuracy *vs.*FLOPs on ImageNet dataset.

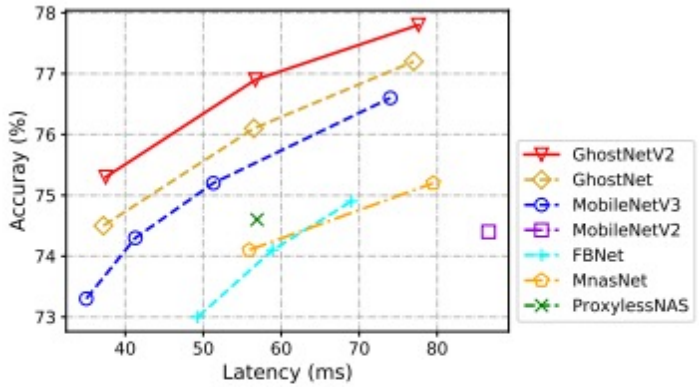


Figure 2: Top-1 accuracy *vs.*latency on ImageNet dataset.

achieves **75.3%** top-1 accuracy on ImageNet with **167M** FLOPs, significantly suppressing GhostNetV1 (74.5%) with a similar computational cost

Table 3: Results of object detection on MS COCO dataset. YOLOv3 [26] is used as the detection head.

Backbone	Resolution	Backbone FLOPs (M)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MobileNetV2 1.0× [28]	320 × 230	613	22.2	41.9	21.4	6.0	23.6	35.8
GhostNet V1 1.1×		338	21.8	41.2	20.8	5.7	22.3	37.3
GhostNetV2 1.0×		342	**22.3**	41.4	21.9	6.0	22.8	38.1
MobileNetV2 1.0× [28]	416 × 416	1035	23.9	45.4	22.6	10.6	25.1	34.9
GhostNet V1 1.1×		567	23.4	45.2	21.9	9.8	24.4	34.9
GhostNetV2 1.0×		571	**24.1**	45.7	23.0	10.4	25.0	36.1

Results of object detection on MS COCO dataset

Static compact network

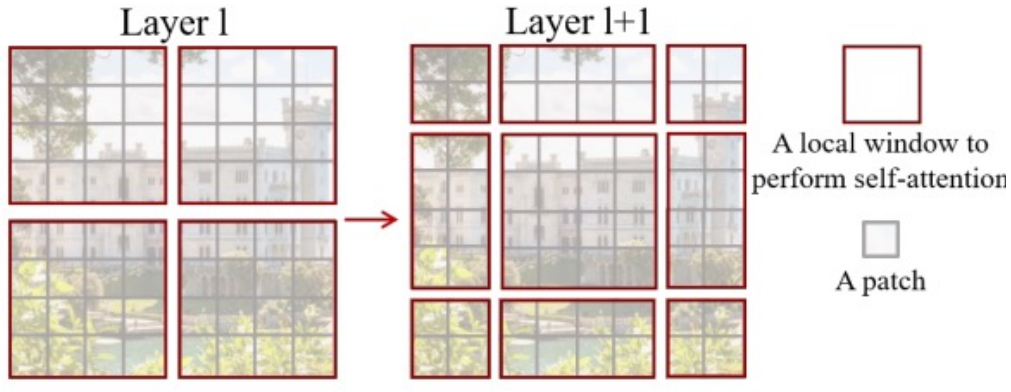
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu[1,2†*] Yutong Lin[1,3†*] Yue Cao[1*] Han Hu[1*‡] Yixuan Wei[1,4†]
Zheng Zhang[1] Stephen Lin[1] Baining Guo[1]
[1]Microsoft Research Asia [2]University of Science and Technology of China
[3]Xian Jiaotong University [4]Tsinghua University
{v-zeliul,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com

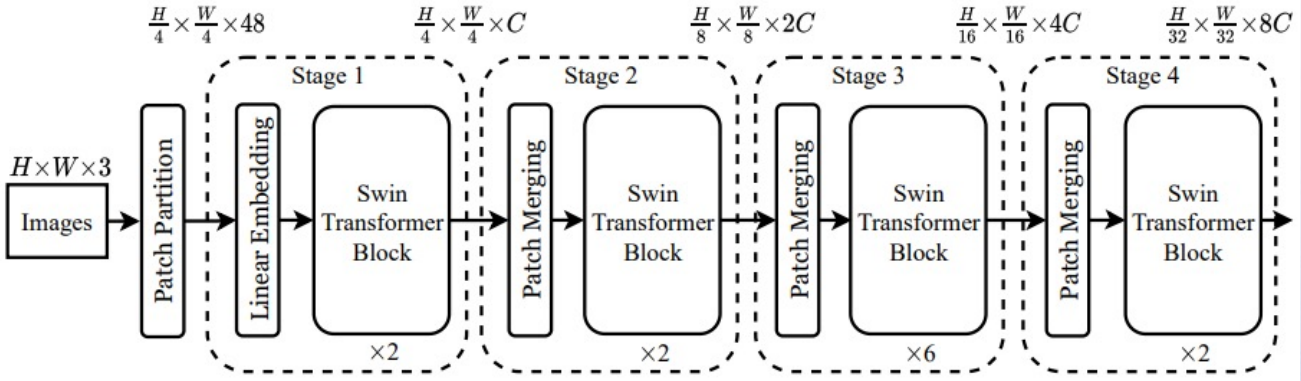
Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, 2021

Static compact network

- **Problem:** previous methods use a global attention mechanism, leading to massive computation burden
- **Method:** propose **shifted windows** to reduce computation



the shifted window approach



the architecture of a Swin Transformer

Static compact network

➢ **Result: higher performance and lower latency**

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [44]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [44]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [44]	224^2	84M	16.0G	334.7	82.9
ViT-B/16 [19]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [19]	384^2	307M	190.7G	27.3	76.5
DeiT-S [57]	224^2	22M	4.6G	940.4	79.8
DeiT-B [57]	224^2	86M	17.5G	292.3	81.8
DeiT-B [57]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

2.4% higher accuracy than ViT, with similar inference throughput

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

also applicable to detection tasks

Static compact network

MOBILEVIT: LIGHT-WEIGHT, GENERAL-PURPOSE, AND MOBILE-FRIENDLY VISION TRANSFORMER

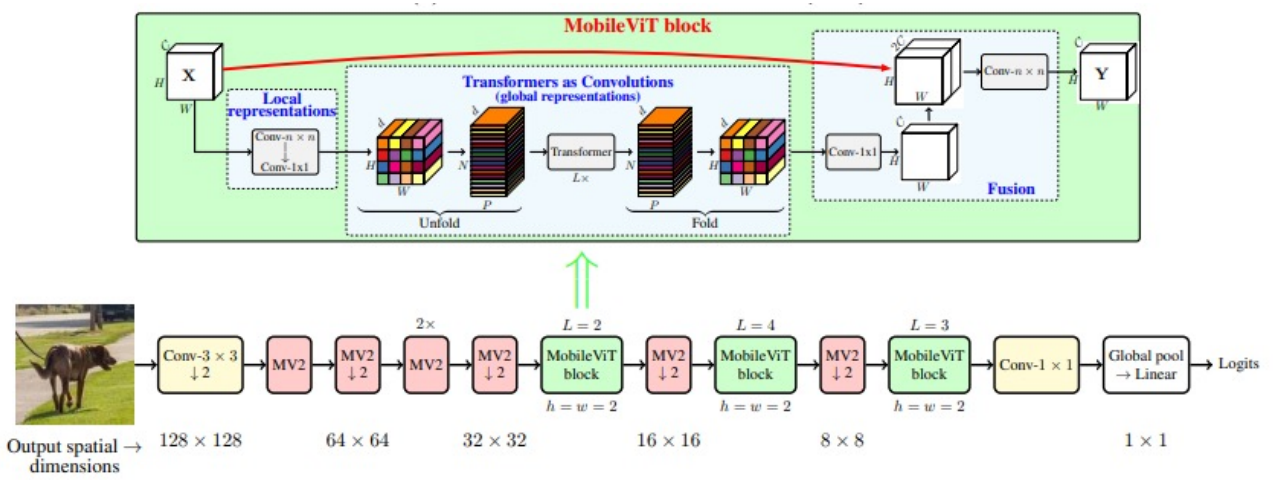
Sachin Mehta
Apple

Mohammad Rastegari
Apple

MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer, 2021

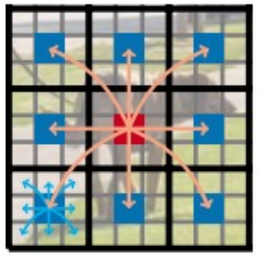
Static compact network

- **Problem:** previous light-weight ViTs exhibit substandard optimizability and poor generalization compared to CNNs for vision tasks
- **Method:** introduce a light-weight ViT **MobileViT** that **combines the strengths of CNNs and ViTs** by modeling transformers as convolutions



(b) **MobileViT**. Here, Conv-$n \times n$ in the MobileViT block represents a standard $n \times n$ convolution and MV2 refers to MobileNetv2 block. Blocks that perform down-sampling are marked with $\downarrow 2$.

MobileViT block that encodes both local and global information in a tensor effectively, by replacing local processing in convolutions with global processing using transformers



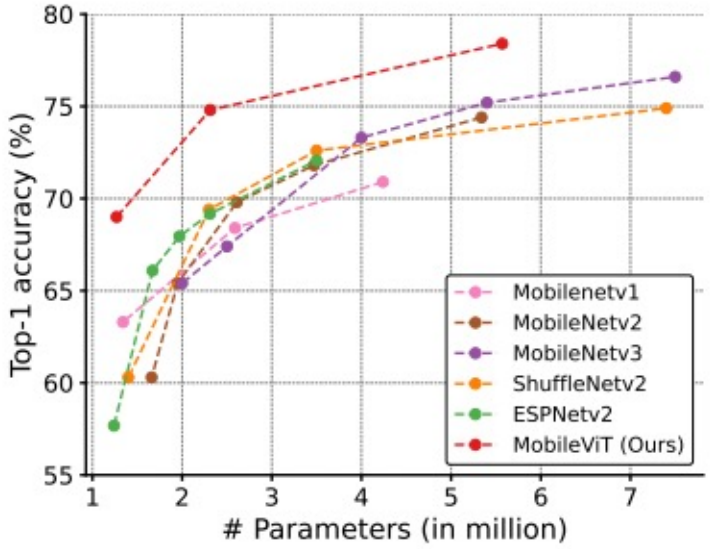
Every pixel sees every other pixel in the MobileViT block.

Layer	Output size	Output stride	Repeat	Output channels		
				XXS	XS	S
Image	256×256	1				
Conv-3 × 3, ↓ 2	128×128	2	1	16	16	16
MV2			1	16	32	32
MV2, ↓ 2	64×64	4	1	24	48	64
MV2			2	24	48	64
MV2, ↓ 2	32×32	8	1	48	64	96
MobileViT block ($L = 2$)			1	48 ($d = 64$)	64 ($d = 96$)	96 ($d = 144$)
MV2, ↓ 2	16×16	16	1	64	80	128
MobileViT block ($L = 4$)			1	64 ($d = 80$)	80 ($d = 120$)	128 ($d = 192$)
MV2, ↓ 2	8×8	32	1	80	96	160
MobileViT block ($L = 3$)			1	80 ($d = 96$)	96 ($d = 144$)	160 ($d = 240$)
Conv-1 × 1			1	320	384	640
Global pool	1×1	256	1			
Linear				1000	1000	1000
Network Parameters				1.3 M	2.3 M	5.6 M

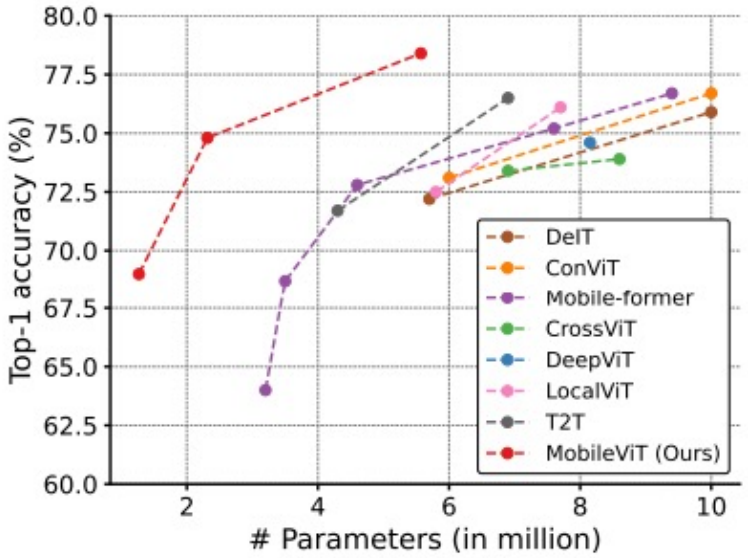
MobileViT architecture

Static compact network

➢ **Result: outperforms state-of-the-art light-weight CNNs and ViTs while being lighter**



Comparison with light-weight CNNs



Comparison with ViTs

Model	# Params. ⇓	Top-1 ⇑
MobileNetv1	2.6 M	68.4
MobileNetv2	2.6 M	69.8
MobileNetv3	2.5 M	67.4
ShuffleNetv2	2.3 M	69.4
ESPNetv2	2.3 M	69.2
MobileViT-XS (Ours)	2.3 M	**74.8**

(b) Comparison with light-weight CNNs (similar parameters)

Model	# Params. ⇓	Top-1 ⇑
DenseNet-169	14 M	76.2
EfficientNet-B0	5.3 M	76.3
ResNet-101	44.5 M	77.4
ResNet-101-SE	49.3 M	77.6
MobileViT-S (Ours)	5.6 M	**78.4**

(c) Comparison with heavy-weight CNNs

Row #	Model	Augmentation	# Params. ⇓	Top-1 ⇑
R1	DeIT	Basic	5.7 M	68.7
R2	T2T	Advanced	4.3 M	71.7
R3	DeIT	Advanced	5.7 M	72.2
R4	PiT	Basic	10.6 M	72.4
R5	Mobile-former	Advanced	4.6 M	72.8
R6	PiT	Advanced	4.9 M	73.0
R7	CrossViT	Advanced	6.9 M	73.4
R8	MobileViT-XS (Ours)	Basic	2.3 M	**74.8**
R9	CeiT	Advanced	6.4 M	76.4
R10	DeIT	Advanced	10 M	75.9
R11	T2T	Advanced	6.9 M	76.5
R12	ViL	Advanced	6.7 M	76.7
R13	LocalVit	Advanced	7.7 M	76.1
R14	Mobile-former	Advanced	9.4 M	76.7
R15	PVT	Advanced	13.2 M	75.1
R16	ConViT	Advanced	10 M	76.7
R17	PiT	Advanced	10.6 M	78.1
R18	BoTNet	Basic	20.8 M	77.0
R19	BoTNet	Advanced	20.8 M	78.3
R20	MobileViT-S (Ours)	Basic	5.6 M	**78.4**

On the ImageNet-1k dataset, MobileViT achieves **top-1 accuracy of 78.4%** with about **6 million parameters**, which is **3.2%** and **6.2%** more accurate than MobileNetv3 (CNN-based) and DeIT (ViT-based) for a similar number of parameters.

Static compact network

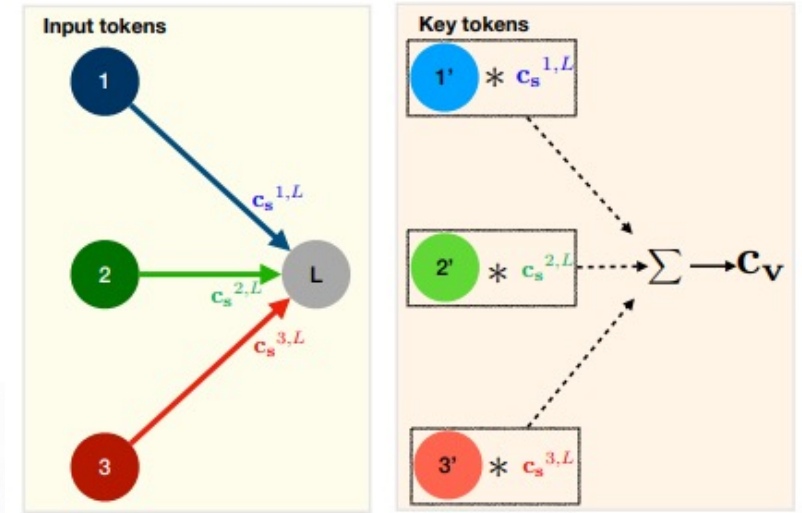
Separable Self-attention for Mobile Vision Transformers

Sachin Mehta
Apple

Mohammad Rastegari
Apple

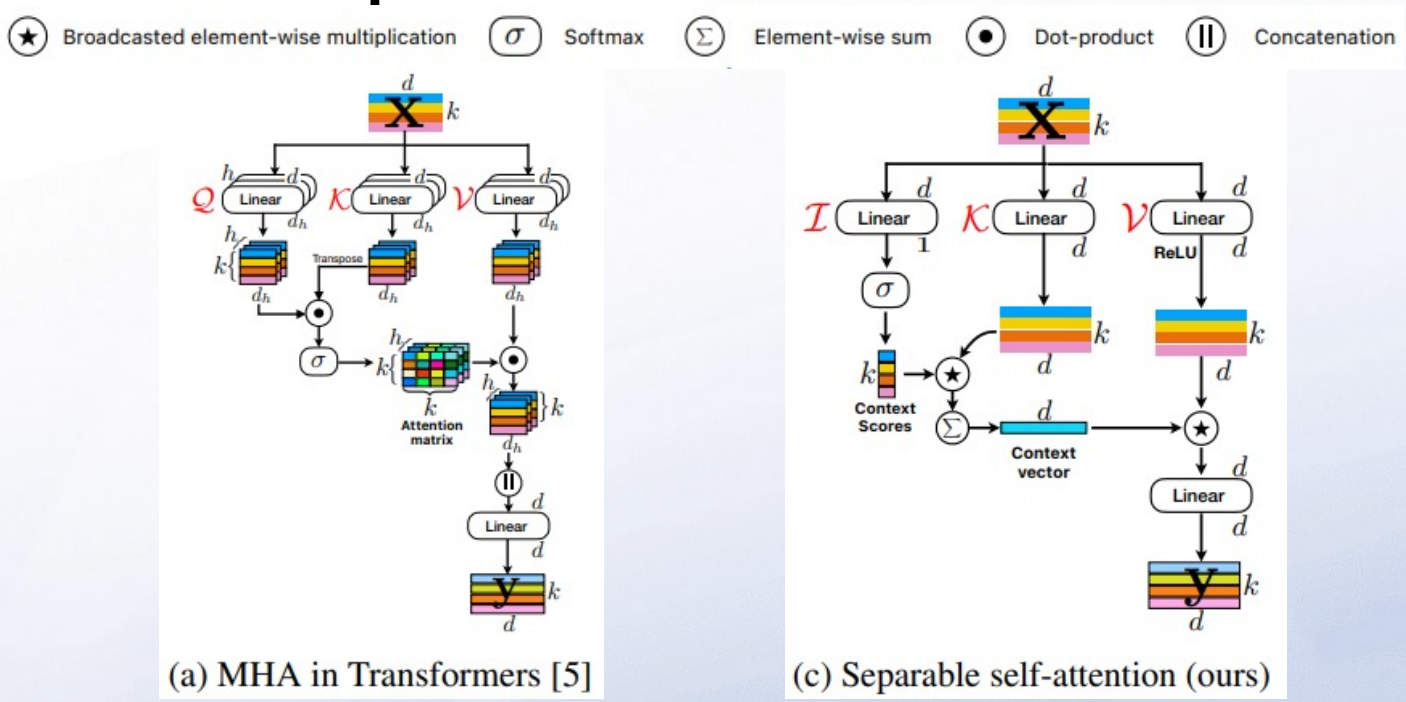
Static compact network

- **Problem:** MobileViT has high latency due to the quadratic computational cost of multi-headed self-attention (MHA)
- **Method:** propose a **separable self-attention** method with linear complexity, and obtain **MobileViTv2** by replacing MHA with separable self-attention in MobileViT



(b) Proposed separable self-attention method

compute **context scores** with respect to a latent token L, then use them re-weight the input tokens and produce a **context vector**



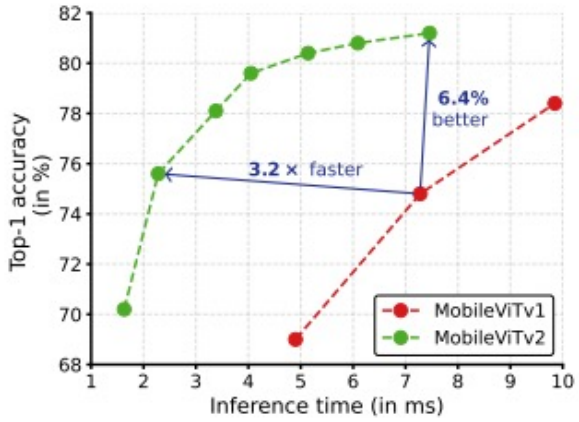
(a) MHA in Transformers [5]

(c) Separable self-attention (ours)

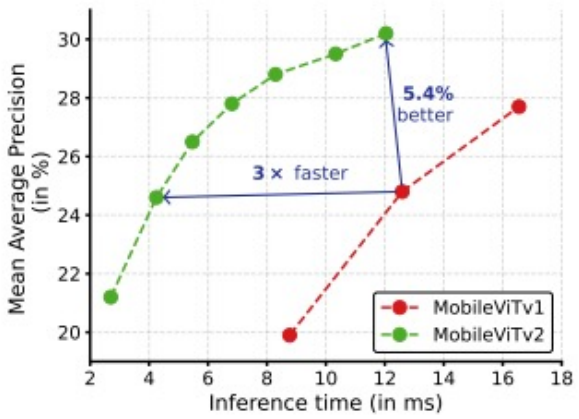
replaces the computationally expensive operations (e.g., batch-wise matrix multiplication) in **MHA** with **element-wise operations** (e.g., summation and multiplication), for faster computation

Static compact network

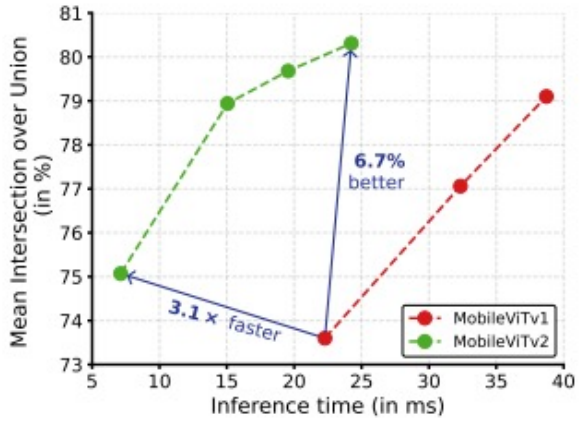
➢ **Result:** **state-of-the-art on several mobile vision tasks**



(a) ImageNet-1k classification

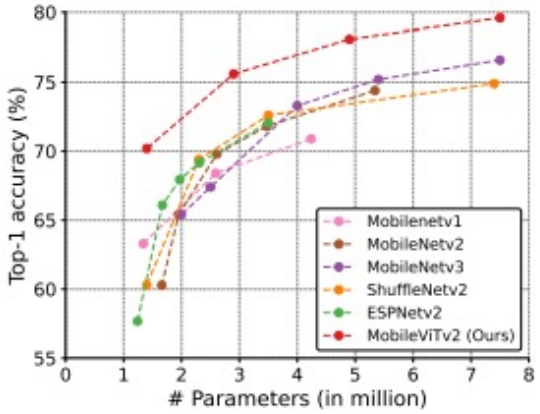


(b) MS-COCO object detection

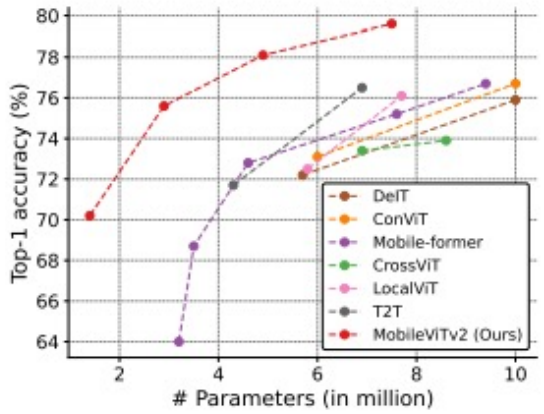


(c) PASCAL VOC segmentation

Faster and better than MobileViTv1 models across different tasks



(a) Comparison with light-weight CNNs



(b) Comparison with light-weight ViTs

Comparisons with light-weight networks on the ImageNet-1k dataset

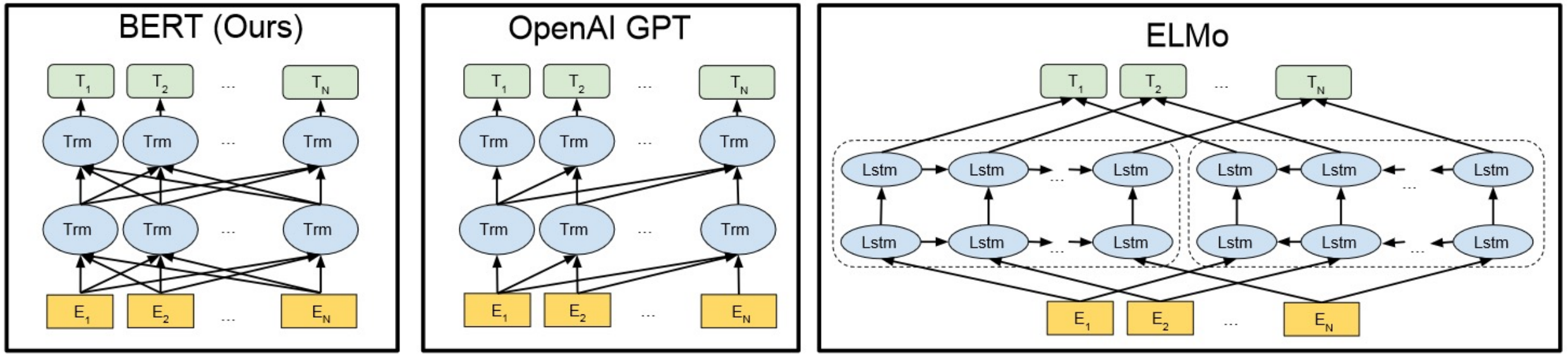
Static compact network

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Static compact network

- **Problem:** Standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training.
- **Method:** propose BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers



BERT uses a **bidirectional** Transformer. Only BERT representations are jointly conditioned on both left and right context in all layers.

OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-toleft LSTMs to generate features for downstream tasks.

Static compact network

➢ **Result:** **state-of-the-art on several NLP tasks**

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

BERT outperforms on all tasks by a substantial margin, obtaining 7.0% respective average accuracy improvement over the prior state of the art.

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Comparisons with other systems on the SQuAD 2.0 task and SWAG dataset

Static compact network

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

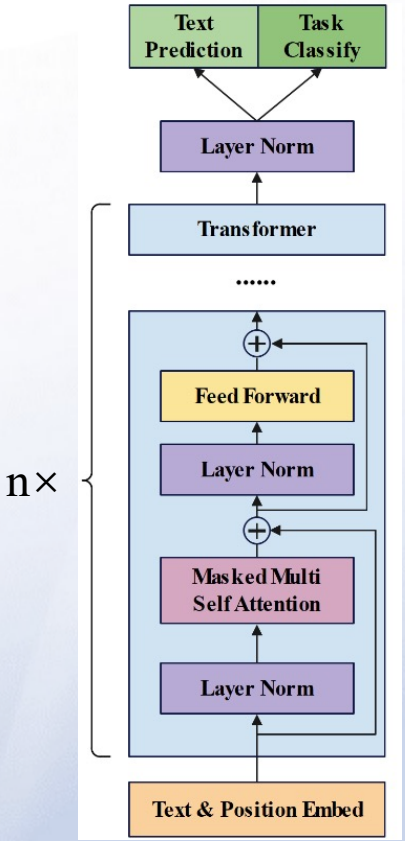
OpenAI

Language Models are Few-Shot Learners, 2020

Static compact network

- **Problem:** While the architecture is task-agnostic, there is still a need for task-specific datasets and task-specific fine-tuning.
- **Method:** propose GPT3: a 175 billion parameter autoregressive language model

meta-learning—which in the context of language models means the model develops a broad set of skills and pattern recognition abilities at training time, and then uses those abilities at inference time to rapidly adapt to or recognize the desired task.

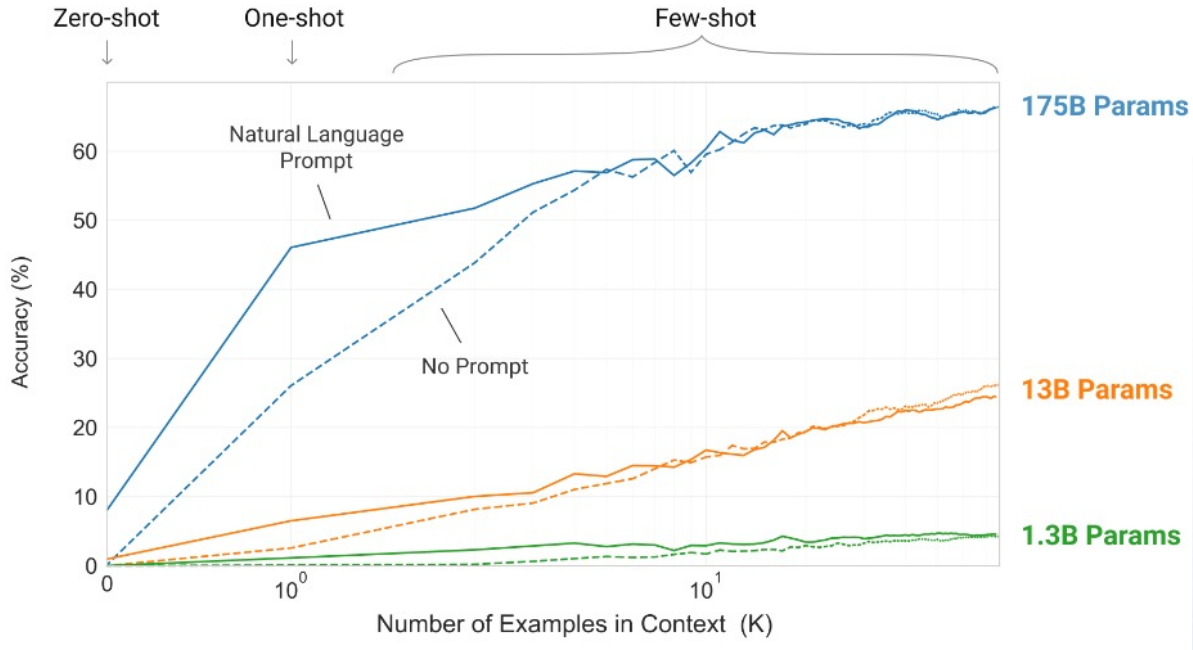


model architecture—the same as GPT-2, with the exception that it uses alternating dense and locally banded sparse attention patterns in the layers of the transformer.

Language Models are Few-Shot Learners, 2020

Static compact network

- **Result:** **strong performance** on NLP tasks and benchmarks in the zero-shot, one-shot, and few-shot settings



Larger models make increasingly efficient use of in-context information.

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Performance on cloze and completion tasks. GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets.

Static compact network

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron,* Thibaut Lavril,* Gautier Izacard,* Xavier Martinet
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin
Edouard Grave,* Guillaume Lample*

Meta AI

LLaMA: Open and Efficient Foundation Language Models, 2023

Static compact network

- **Problem:** Most existing models rely on data which is either **not publicly available** or **undocumented**
- **Method:** present an overview of the modifications to the transformer and training method

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Training dataset is a **mixture** of several sources, that cover a diverse set of domains.

Architecture--based on the transformer

- **Pre-normalization--**normalize the input of each transformer sub-layer with RMSNorm
- **SwiGLU activation function--**replace the ReLU non-linearity by the SwiGLU activation function
- **Rotary Embeddings--**remove the absolute positional embeddings, and add rotary positional embeddings (RoPE)

Static compact network

➢ **Result:** **state-of-the-art on several NLP tasks**

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

Zero-shot performance on Common Sense Reasoning tasks. LLaMA-13B model also outperforms GPT-3 on most benchmarks despite being 10× smaller.

Write a regex to remove all the HTML tags in a python string.

```
import re
def removeHtmlTags(text):
    return re.sub('<[^>]*>', '', text)
```

Write a regexp to extract function definitions in a python file.

Sure! Here's my python code to solve the problem:

```
import re
def extract_functions(text):
    return (re.findall('(?:^|\n)\s*def\s+[a-zA-Z_][a-zA-Z0-9_]*\(.*?\)',
        text))
```

a few examples of generations with LLaMA-I, i.e. LLaMA-65B fine-tuned with the protocol and instruction dataset

目录

Dynamic network architecture

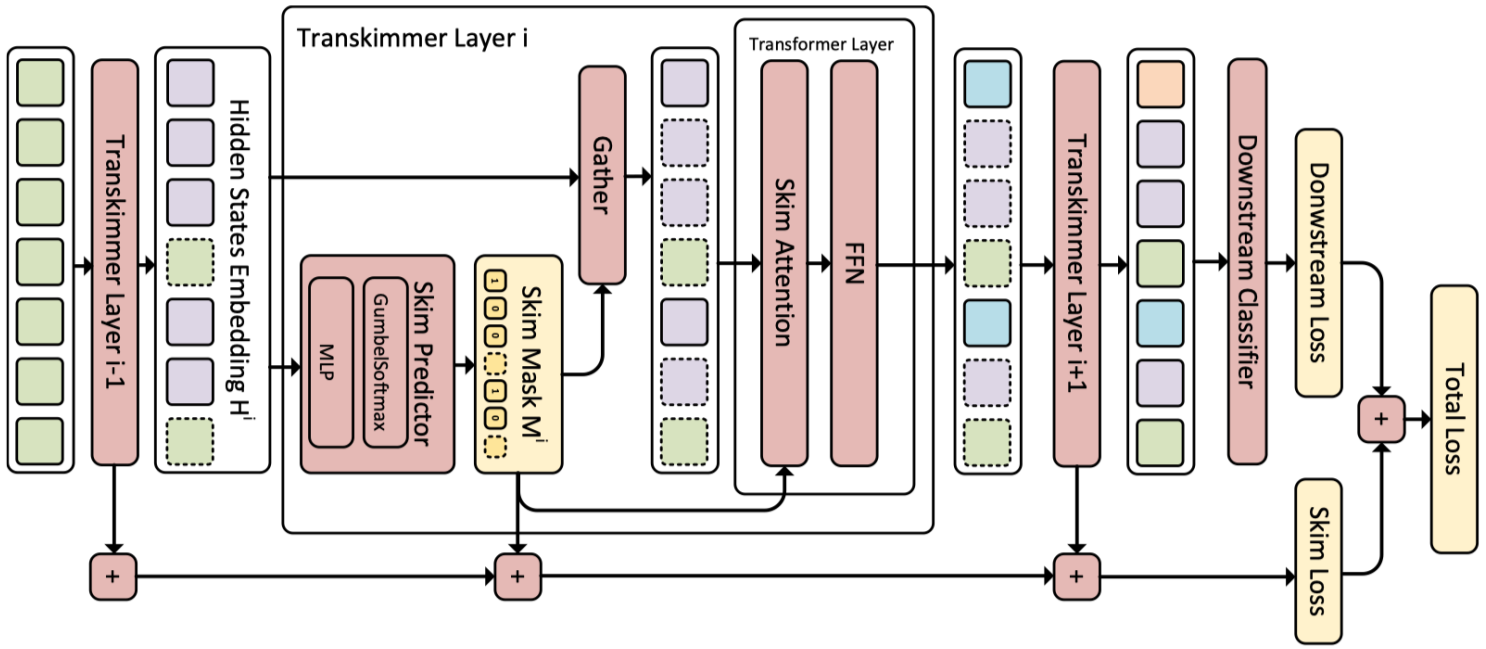
Transkimmer: Transformer Learns to Layer-wise Skim

Yue Guan[1,2,⋆], **Zhengyi Li**[1,2,⋆], **Jingwen Leng**[1,2,⋆], **Zhouhan Lin**[1,§] and **Minyi Guo**[1,2,†]
[1]Shanghai Jiao Tong University, [2]Shanghai Qizhi Institute
⋆{bonboru,hobbit,leng-jw}@sjtu.edu.cn,
§lin.zhouhan@gmail.com, †guo-my@cs.sjtu.edu.cn

Transkimmer: Transformer Learns to Layer-wise Skim, 2022

Dynamic network architecture

- **Problem:** how to skim in NLP
- **Method:** propose **Transkimmer** with a parameterized **skimming predictor**



Architecture of Transkimmer

- identify tokens that are **skimmed** in each layer and **directly forward** them to the final output of the network **without further processing**, to reduce computation
- the predictor outputs a **skimming mask M**
- the attention weights to the skimmed tokens are set to 0

$$SkimAttn(H^i) = Attn(H^i) \cdot M^i$$

Dynamic network architecture

- **Result:** achieves **10.97×** average speedup on GLUE benchmark compared with BERTbaseline with less than **1%** accuracy degradation

	Method	Padding	COLA		RTE		QQP		MRPC		SST-2		MNLI		WNLI		QNLI		STS-B	
			Matthews	FLOPs	Acc.	FLOPs	Acc.	FLOPs	F1	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Acc.	FLOPs	Pearson	FLOPs
$BERT_{base}$	Baseline	-	57.8	1.00×	65.7	1.00×	91.3	1.00×	88.9	1.0×	93.0	1.00×	84.9	1.00×	56.3	1.00×	91.4	1.00×	88.6	1.00×
	DeeBERT	-	-	-	66.7	1.50×	-	-	85.2	1.79×	91.5	1.89×	80.0	1.59×	-	-	87.9	1.79×	-	-
	POWER-BERT	Sequence	52.3	4.50×	67.4	3.40×	90.2	4.50×	88.1	2.70×	92.1	2.40×	83.8	2.60×	-	-	90.1	2.00×	85.1	2.00×
	LAT	Sequence	-	-	-	-	-	-	-	-	**92.8**	2.90×	**84.4**	2.80×	-	-	-	-	-	-
	Transkimmer	No	58.9	1.75×	68.9	2.85×	90.8	2.79×	88.5	3.13×	92.3	1.58×	83.2	2.02×	56.3	5.56×	90.5	2.33×	87.4	3.45×
	Transkimmer	Sequence	**58.9**	**18.9×**	**68.9**	**4.67×**	**90.8**	**11.72×**	**88.5**	**7.45×**	92.3	**10.89×**	83.2	**6.65×**	**56.3**	**18.10×**	**90.5**	**6.01×**	**87.4**	**18.20×**
	DistilBERT	-	55.7	1.98×	58.8	1.98×	90.3	1.98×	88.3	1.98×	90.6	1.98×	87.5	1.98×	53.5	1.98×	89.3	1.98×	87.0	1.98×
	+Transkimmer	No	55.1	3.52×	59.2	4.12×	90.1	4.95×	87.8	9.92×	89.5	5.01×	86.7	4.40×	56.3	10.41×	87.5	4.04×	86.5	3.47×
	ALBERT	-	58.3	0.99×	70.7	0.99×	90.2	0.99×	90.4	0.99×	90.9	0.99×	81.8	0.99×	56.3	0.99×	89.2	0.99×	90.4	0.99×
	+Transkimmer	No	53.4	1.52×	71.5	1.57×	90.2	3.09×	90.6	1.94×	90.1	3.25×	81.5	1.67×	57.7	6.19×	90.1	2.30×	89.8	1.46×
$RoBERTa_{base}$	Baseline	-	61.8	1.00×	78.0	1.00×	90.4	1.00×	92.1	1.00×	94.3	1.00×	87.5	1.00×	56.6	1.00×	92.9	1.00×	90.9	1.00×
	LTP	Batch	-	-	**78.0**	1.81×	89.7	2.10×	91.6	2.10×	**93.5**	2.09×	86.5	1.88×	-	-	**92.0**	1.87×	90.0	1.95×
	Transkimmer	No	**61.3**	**1.52×**	76.2	**1.79×**	**91.0**	**4.92×**	**91.9**	**2.67×**	**93.5**	**2.08×**	**86.7**	**2.19×**	**56.3**	**8.41×**	91.7	**2.85×**	**90.5**	**2.70×**

Table 3: Performance and FLOPs (speedup) on GLUE benchmark with $BERT_{base}$ and $RoBERTa_{base}$ as backbone model. Transkimmer is adopted on DistilBERT and ALBERT to shows its applicability to general model compression methods.

Dynamic network architecture

GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding

Dmitry Lepikhin
lepikhin@google.com

HyoukJoong Lee
hyouklee@google.com

Yuanzhong Xu
yuanzx@google.com

Dehao Chen
dehao@google.com

Orhan Firat
orhanf@google.com

Yanping Huang
huangyp@google.com

Maxim Krikun
krikun@google.com

Noam Shazeer
noam@google.com

Zhifeng Chen
zhifengc@google.com

GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding, 2020

Dynamic network architecture

- **Problem:** inefficient computation and complex parallelization in scaling MoEs
- **Method:** propose conditional computation and automatic sharding

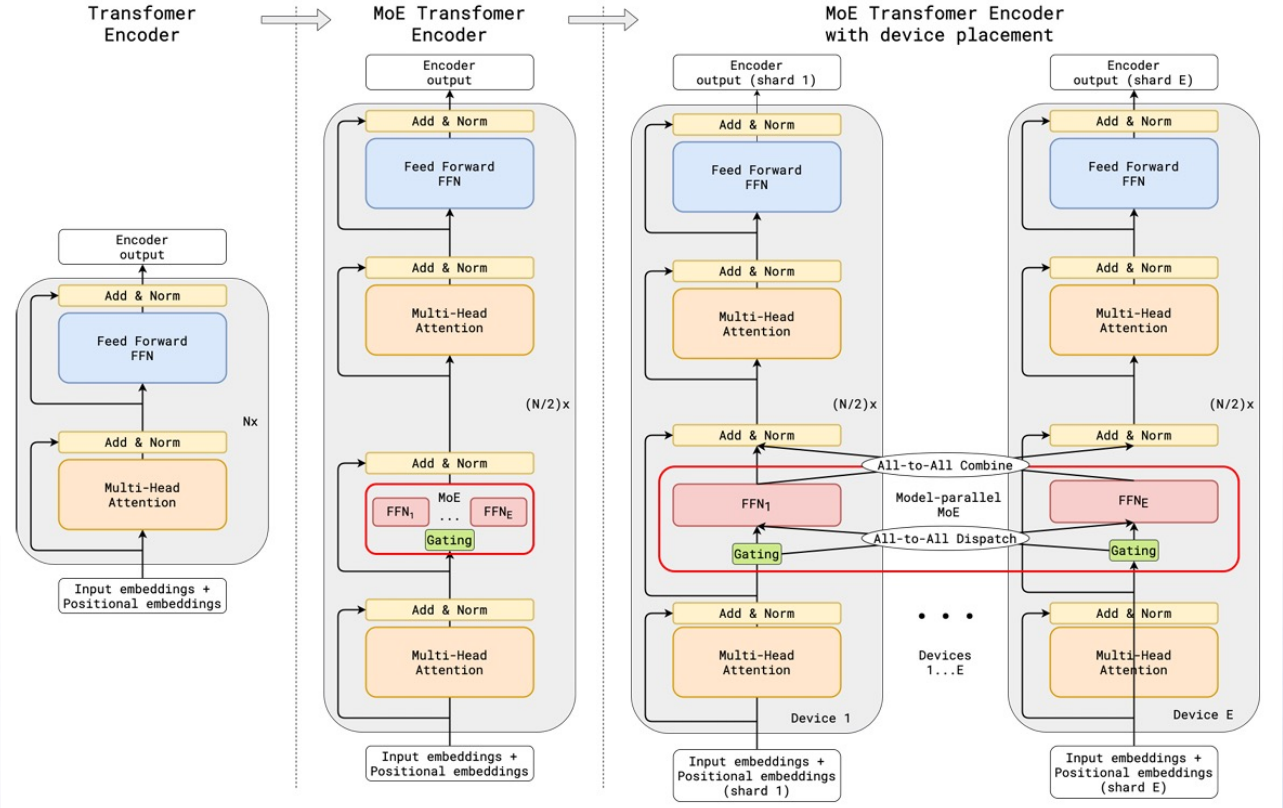


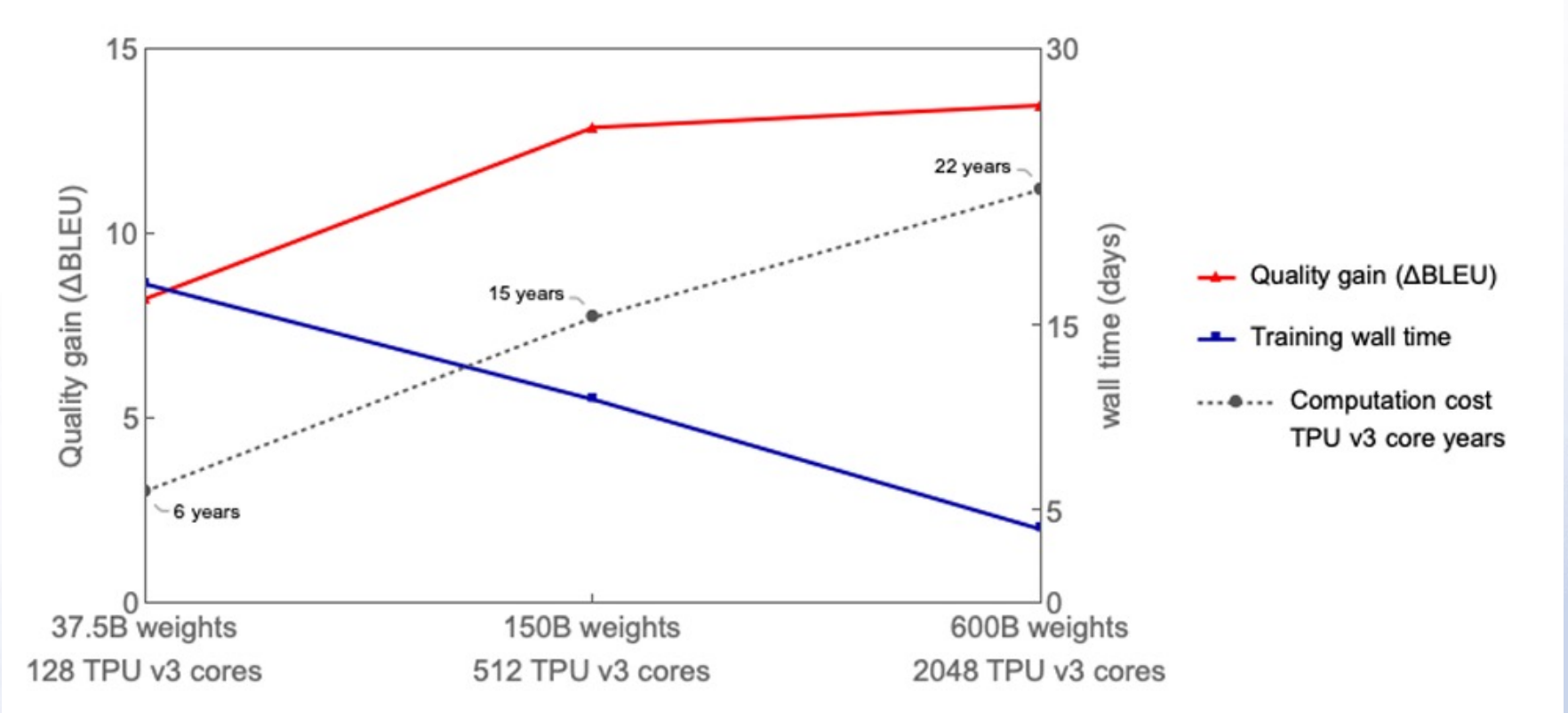
Illustration of scaling of Transformer Encoder with MoE Layers.

When scaling to multiple devices, the MoE layer will be shared across devices, while all other layers will be replicated.

- **Random Routing:** Select top expert always; second with probability proportional to its weight.
- **Expert Capacity:** Set limits on token handling per expert to manage overflow efficiently.

Dynamic network architecture

- **Result:** **Established the parallel paradigm for MoEs. Multilingual translation quality improved as MoE model size grows up to 600B, while the end-to-end training cost only increased sublinearly.**



Dynamic network architecture

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

Noam Shazeer
NOAM@GOOGLE.COM
Google, Mountain View, CA 94043, USA

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, 2021

Dynamic network architecture

- **Problem:** training inefficiency in very large language models
- **Method:** propose **Switch routing** which routes each token to one expert

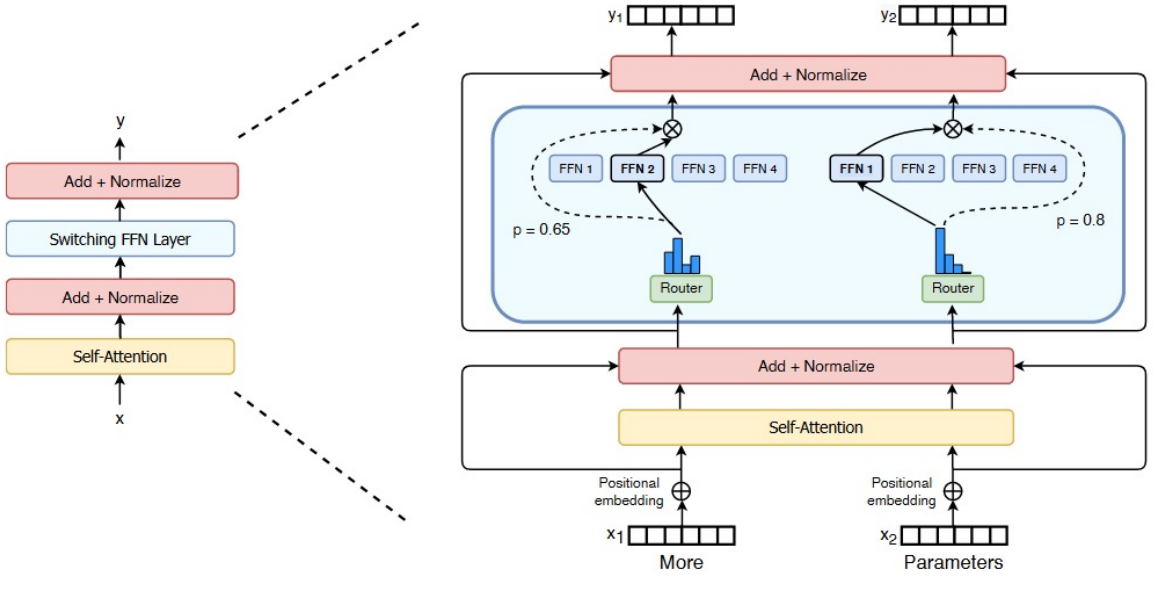


Illustration of a Switch Transformer encoder block.

- Reduced routing computation
- Halved batch size for each expert
- Lowered communication costs
- Guaranteed the quality of the model

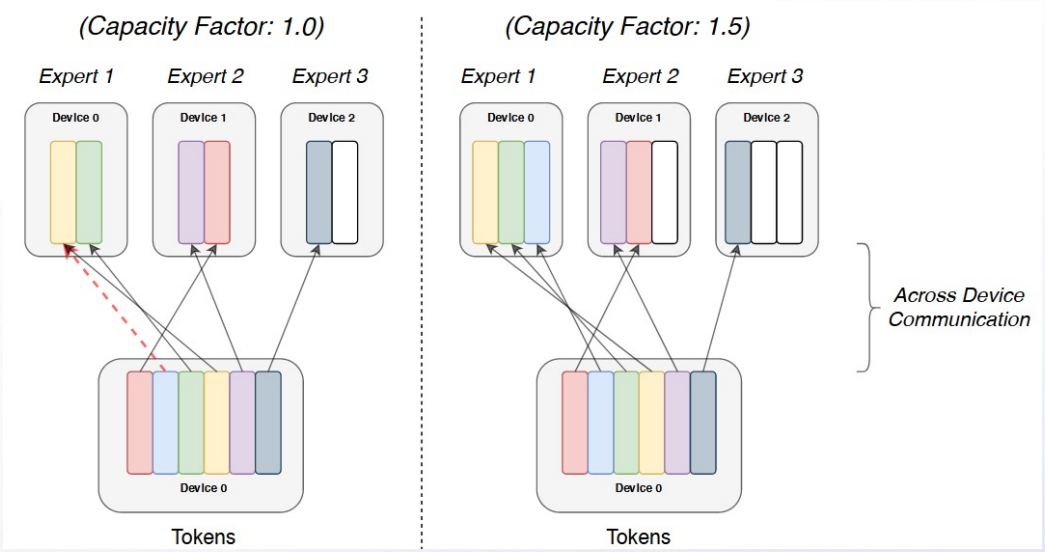


Illustration of token routing dynamics.

$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}}\right) \times \text{capacity factor}$$

- Efficient load distribution among experts to prevent overflow and resource waste.
- 1.25 for best performance.

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, 2021

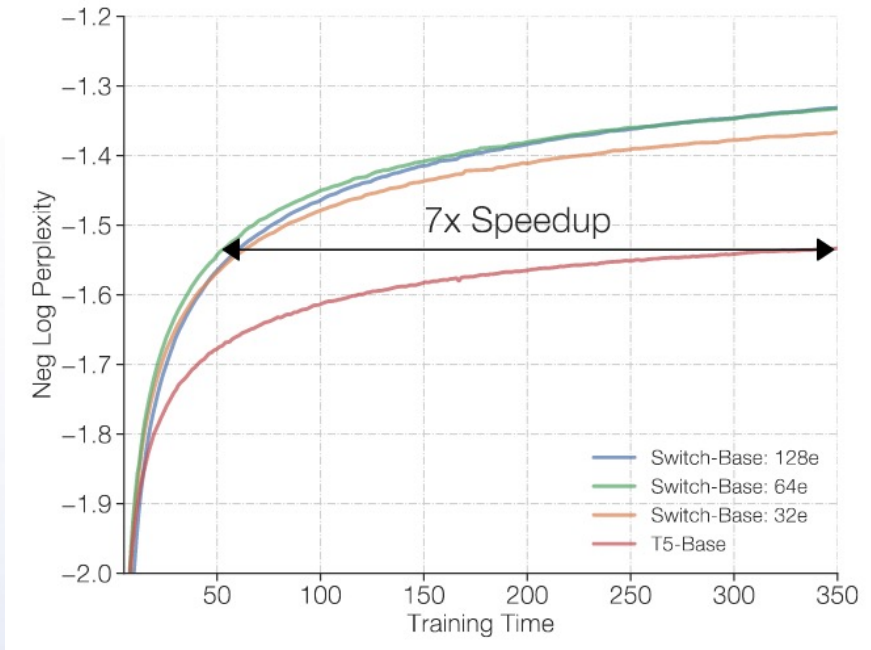
Dynamic network architecture

➢ Result:

Model	Capacity Factor	Quality after 100k steps (↑) (Neg. Log Perp.)	Time to Quality Threshold (↓) (hours)	Speed (↑) (examples/sec)
T5-Base	—	-1.731	Not achieved†	1600
T5-Large	—	-1.550	131.1	470
MoE-Base	2.0	-1.547	68.7	840
Switch-Base	2.0	-1.554	72.8	860
MoE-Base	1.25	-1.559	80.7	790
Switch-Base	1.25	-1.553	65.0	910
MoE-Base	1.0	-1.572	80.1	860
Switch-Base	1.0	-1.561	**62.8**	1000
Switch-Base+	1.0	**-1.534**	67.6	780

Benchmarking Switch versus MoE.

Switch Transformers outperform both carefully tuned dense models and MoE Transformers on a speed-quality basis.



Achieved up to 7 times faster pre-training speed with the same computational resources compared to the T5 model.

Dynamic network architecture

TAMING SPARSELY ACTIVATED TRANSFORMER WITH STOCHASTIC EXPERTS

Simiao Zuo[†,*] **Xiaodong Liu**[◇], **Jian Jiao**[◇], **Young Jin Kim**[◇], **Hany Hassan**[◇], **Ruofei Zhang**[◇], **Tuo Zhao**[†] and **Jianfeng Gao**[◇]
[†]Georgia Institute of Technology [◇]Microsoft
{simiaozuo,tourzhao}@gatech.edu,
{xiaodl,jian.jiao,youki,hanyh,bzhang,jfgao}@microsoft.com

Taming Sparsely Activated Transformer with Stochastic Experts, 2022

Dynamic network architecture

- **Problem:** sparsely activated models are not parameter efficient
- **Method:** propose **THOR (Transformer with Stochastic Experts)** in which experts are **randomly activated** for each input

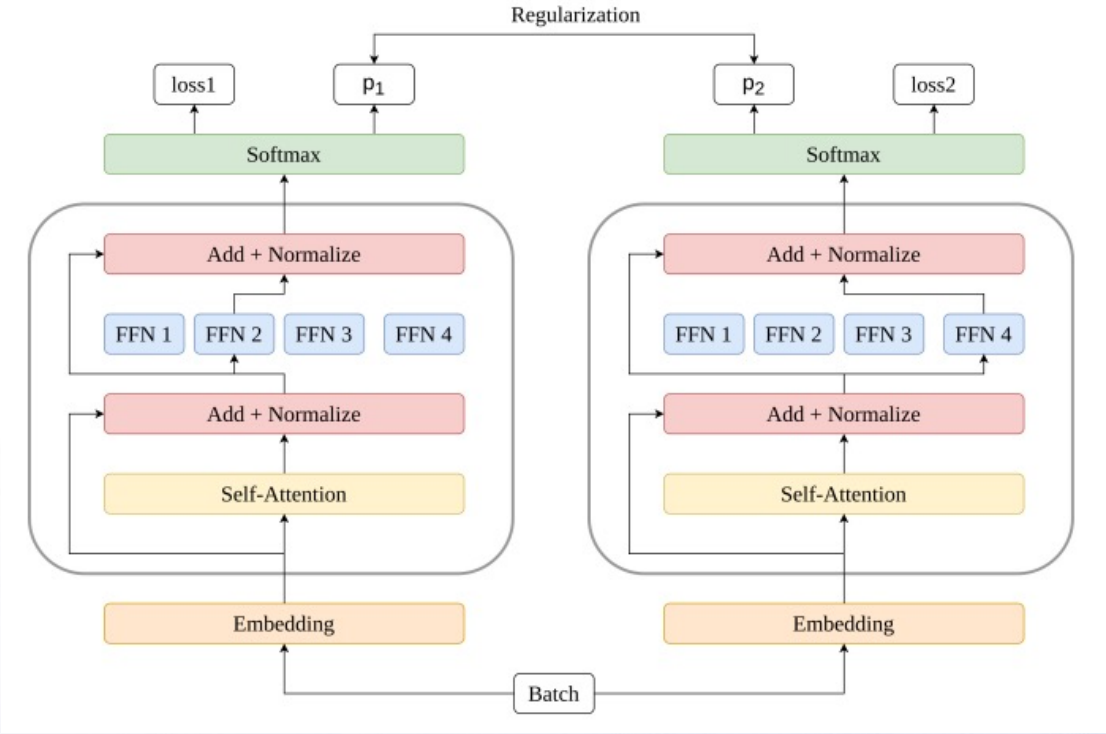


illustration of a training iteration with stochastic experts (FFN)

training objective of THOR

$$\min \sum_{(x,y)\in\mathcal{D}} \ell(x,y) = \mathrm{CE}(p_1;y) + \mathrm{CE}(p_2;y) + \alpha \mathrm{CR}(p_1;p_2),$$

$$\text{where } \mathrm{CR}(p_1;p_2) = \frac{1}{2}\left(\mathrm{KL}(p_1\|p_2) + \mathrm{KL}(p_2\|p_1)\right).$$

minimizing both the cross-entropy loss and a **consistency regularization term**, such that experts can learn from other experts as teachers, so that all the experts make consistent predictions

Dynamic network architecture

➢ **Result:** outperforms strong baselines and achieves new state-of-the-art results

	En-Vi	Vi-En	En-De	De-En	En-Fr	Fr-En
Transformer (Vaswani et al., 2017)	31.3	29.4	28.1	34.8	39.2	38.1
SMART (Jiang et al., 2020)	32.5	30.5	29.3	35.8	40.0	38.8
R3F (Aghajanyan et al., 2020)	32.2	30.7	29.2	35.7	39.7	38.9
Switch (Fedus et al., 2021)	31.7	29.5	28.4	34.6	39.1	38.2
THOR	**34.0**	**33.0**	**31.1**	**37.8**	**40.7**	**40.0**

	En-Ro	Ro-En	En-Lv	Lv-En	En-Cs	Cs-En
Transformer (Vaswani et al., 2017)	23.5	25.0	13.6	15.8	16.1	20.4
SMART (Jiang et al., 2020)	24.6	25.7	14.2	16.3	16.7	21.4
R3F (Aghajanyan et al., 2020)	23.8	25.8	14.4	16.3	16.8	21.6
Switch (Fedus et al., 2021)	23.8	24.4	13.8	16.1	16.1	20.6
THOR	**25.2**	**27.1**	**15.2**	**17.4**	**17.6**	**22.4**

BLEU	En-De	En-Fr
Vaswani et al. (2017)	28.4	41.8
Ott et al. (2018)	29.3	43.2
Wang et al. (2019b)	29.6	—
Wu et al. (2019a)	29.7	43.2
So et al. (2019)	29.8	41.3
Jiang et al. (2020)	29.8	43.4
Wu et al. (2019b)	29.9	43.3
Aghajanyan et al. (2020)	29.4	43.3
Liu et al. (2020c)	30.1	**43.8**
Fedus et al. (2021)	29.3	43.0
THOR	**30.4**	**43.8**

higher accuracy on machine translation tasks

目录

Thanks!