ODTÜ
METU

TÜBİTAK

EURO 4SEE

Optimizing Deep Learning Systems for Hardware

Assoc. Prof. Erdem AKAGÜNDÜZ, METU

ncc@ulakbim.gov.tr

Next: Part III.b

- PIII.a : Model compression
- PIII.b : Efficient Architectures

Part III.b: Efficient Architectures

- a) Model Compression
  - Pruning
  - Quantization
  - Knowledge Distillation
- b) Efficient architectures
  - Case: Efficient CNNs

Efficient Architecture?

- Definition: An efficient architecture is a model design that
  - reduces computational cost, memory usage, and latency,
  - **through structural choices**,
  - (hopefully) without significantly sacrificing accuracy.
- Architectural Efficiency
  - Smarter building blocks (i.e. layers)
  - Better use of compute
  - Hardware-aware design

Efficient Dimensions

- Compute Efficiency
  - Minimize the number of operations (e.g., FLOPs)
    → Faster inference/training, lower energy cost

- Memory Efficiency
  - Reduce model size (parameters) and runtime memory (activations)
  → Enables deployment on edge devices, fits more in (GPU) memory

- Latency & Hardware Friendliness
  - Optimize for speed on real hardware
  → Fewer sequential operations, better dataflow, hardware-aligned ops

Efficient Dimensions

TÜRKİYE EURO 4SEE

- Compute Efficiency
    - Minimize the number of operations (e.g., FLOPs)
      → Faster inference/training, lower energy cost
- Memory Efficiency
    - Reduce model size (parameters) and runtime memory (activations)
    → Enables deployment on edge devices, fits more in (GPU) memory
- Latency & Hardware Friendliness
    - Optimize for speed on real hardware
    → Fewer sequential operations, better dataflow, hardware-aligned ops

through structural choices!

CNNs: Early Efficiency Tricks

- MobileNet Family
  - MobileNetV1: depthwise separable conv
  - MobileNetV2: inverted residuals & linear bottlenecks

- Key idea: reduce multiply-adds without losing too much accuracy

MobileNet V1

arXiv > cs > arXiv:1704.04861

Computer Science > Computer Vision and Pattern Recognition

(Submitted on 17 Apr 2017)

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

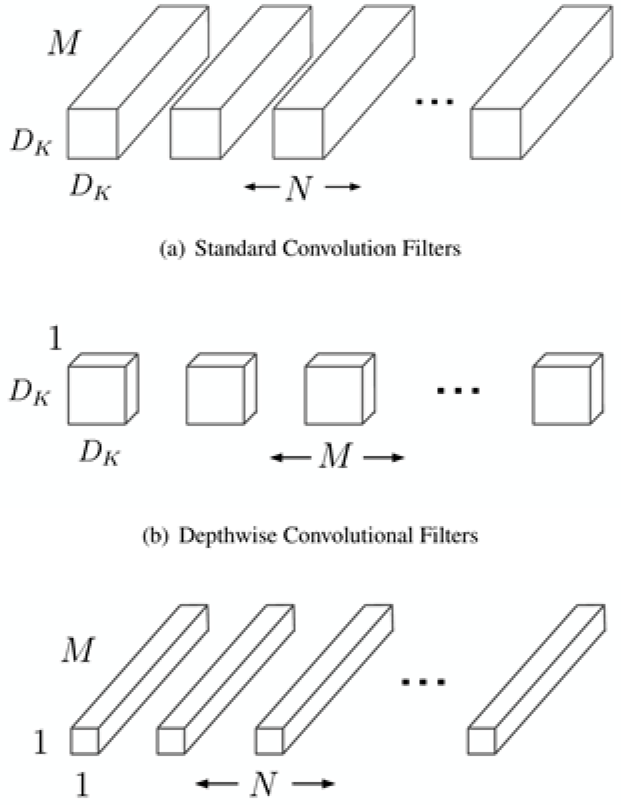
Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam

- MobileNet V1: depthwise separable convolution
  - Depthwise convolution

    (one 3×3 filter per input channel, no mixing)
  - Pointwise convolution

    (a 1×1 convolution to mix channels)

- Standard convolution (kernel size k×k, input channels M, output channels N) has cost:
  - **k×k×M×N**
- Depthwise separable convolution has cost:
  - **k×k×M + 1×1×M×N**



(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

MobileNet V1

arXiv > cs > arXiv:1704.04861

Computer Science > Computer Vision and Pattern Recognition

(Submitted on 17 Apr 2017)

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

- MobileNet V1: depthwise separable convolution
  - Depthwise convolution

    (one 3×3 filter per input channel, no mixing)
  - Pointwise convolution

    (a 1×1 convolution to mix channels)

- Standard convolution (kernel size k×k, input channels M, output channels N) has cost:
  - **k×k×M×N**
- Depthwise separable convolution has cost:
  - **k×k×M + 1×1×M×N**

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet V1

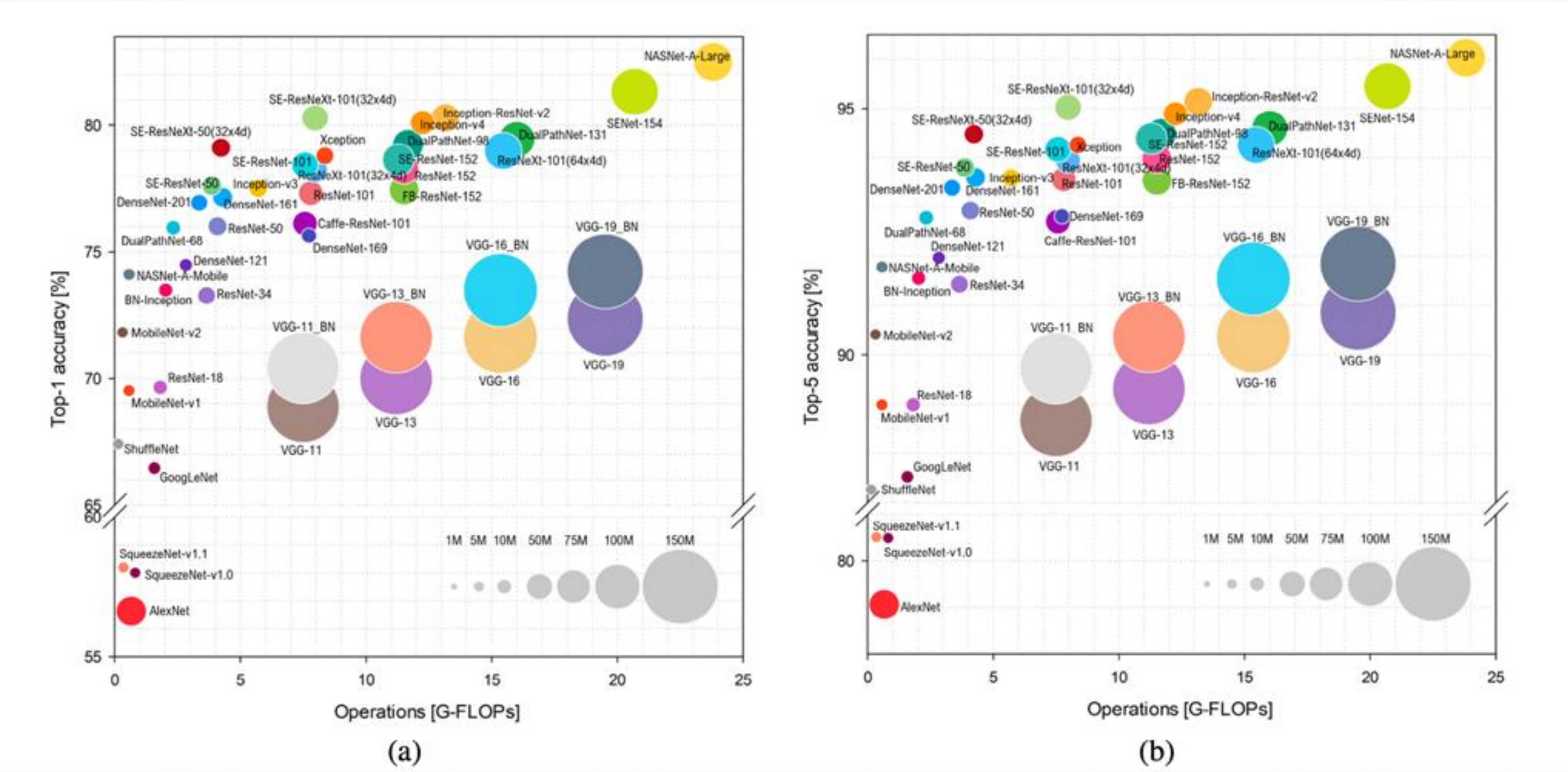
arXiv > cs > arXiv:1704.04861

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 17 Apr 2017]

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



MobileNet V1

arXiv > cs > arXiv:1704.04861

Computer Science > Computer Vision and Pattern Recognition

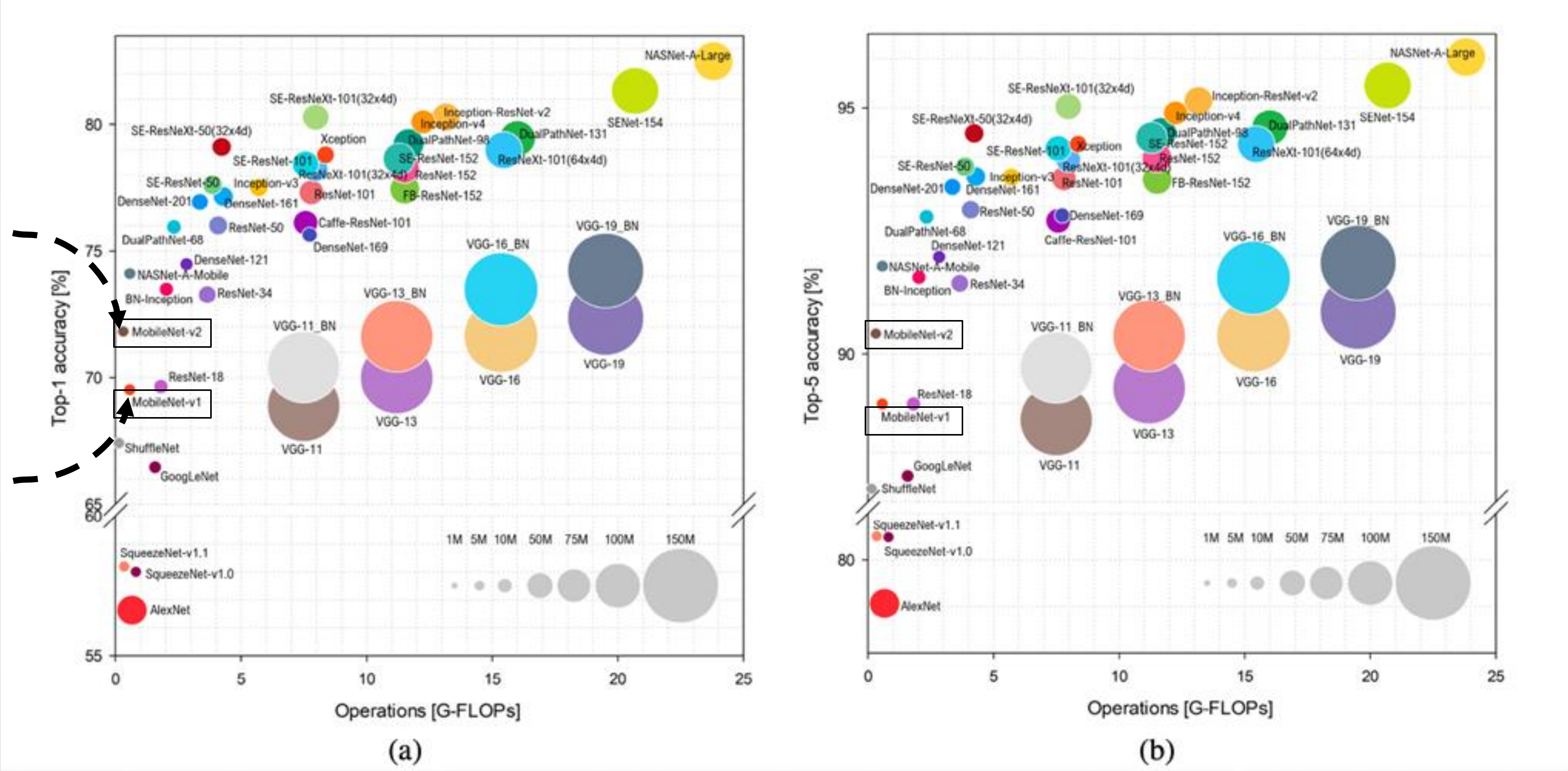
(Submitted on 17 Apr 2017)

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

NC TÜRKİYE

EURO CC 4SEE



MobileNet V1

arXiv > cs > arXiv:1704.04861

Computer Science > Computer Vision and Pattern Recognition

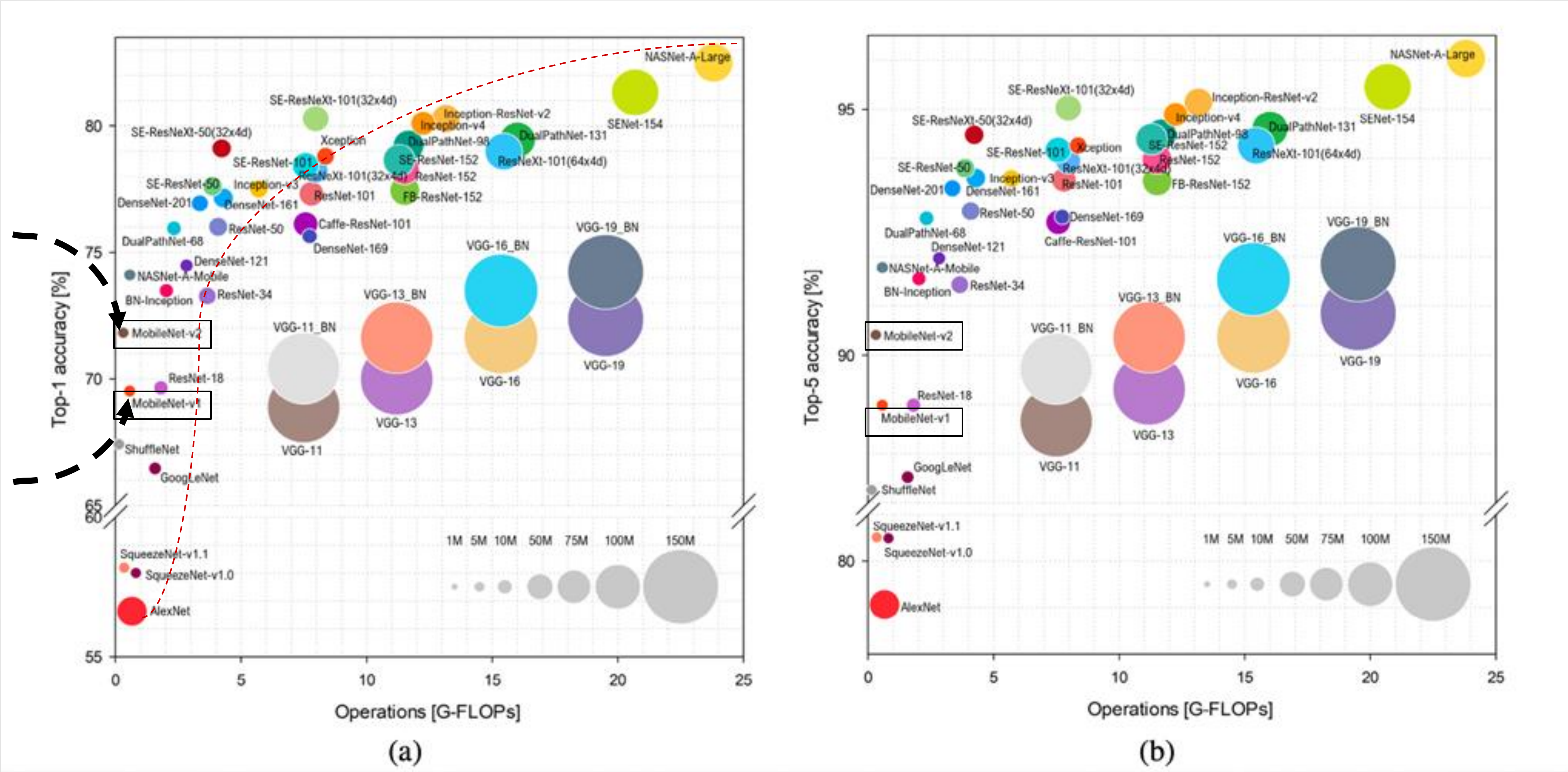
[Submitted on 17 Apr 2017]

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

TÜRKİYE

EURO 4SEE



MobileNet V2


arXiv > cs > arXiv:1801.04381

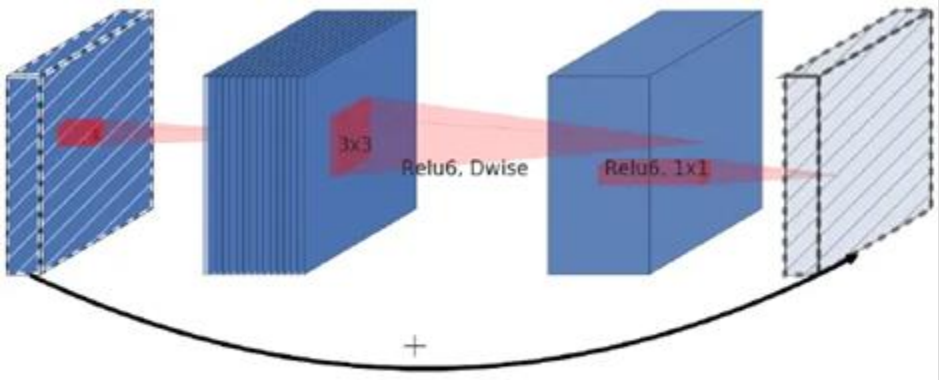
Computer Science > Computer Vision and Pattern Recognition

(Submitted on 13 Jan 2018 (v1), last revised 21 Mar 2019 (this version, v4))

MobileNetV2: Inverted Residuals and Linear Bottlenecks

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen

- **MobileNet V2: Inverted Residuals**
  - Residual blocks connect the beginning and end of a convolutional block with a skip connection.
  - This approach turned out to be essential in order to build networks of great depth.
  - An inverted residual block connects narrow layers with a skip connection while layers in between are wide
  - The authors describe this idea as an inverted residual block because skip connections exist between narrow parts of the network which is opposite of how an original residual connection works.
    - This is more efficient in OPs.



MobileNet V2


arXiv > cs > arXiv:1801.04381

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 13 Jan 2018 (v1), last revised 21 Mar 2019 (this version, v4)]

MobileNetV2: Inverted Residuals and Linear Bottlenecks

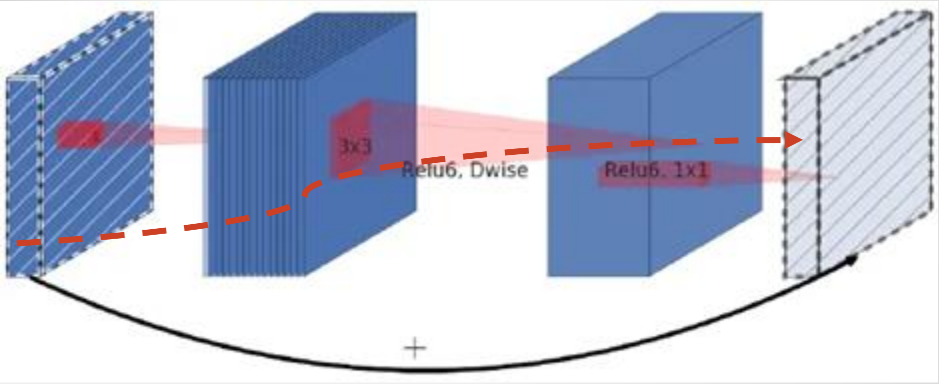
Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen

TÜRKİYE

EURO 4SEE

- **MobileNet V2: Linear Bottlenecks**
  - When we narrow the feature dimension (bottleneck), we reduce redundancy.
  - Applying non-linearities (like ReLU) after such layers can zero out important information.
  - To avoid this, we use a linear activation — that is, we apply no activation function after the bottleneck layer.

Linear
(i.e. no actv. func.)



MobileNet V1

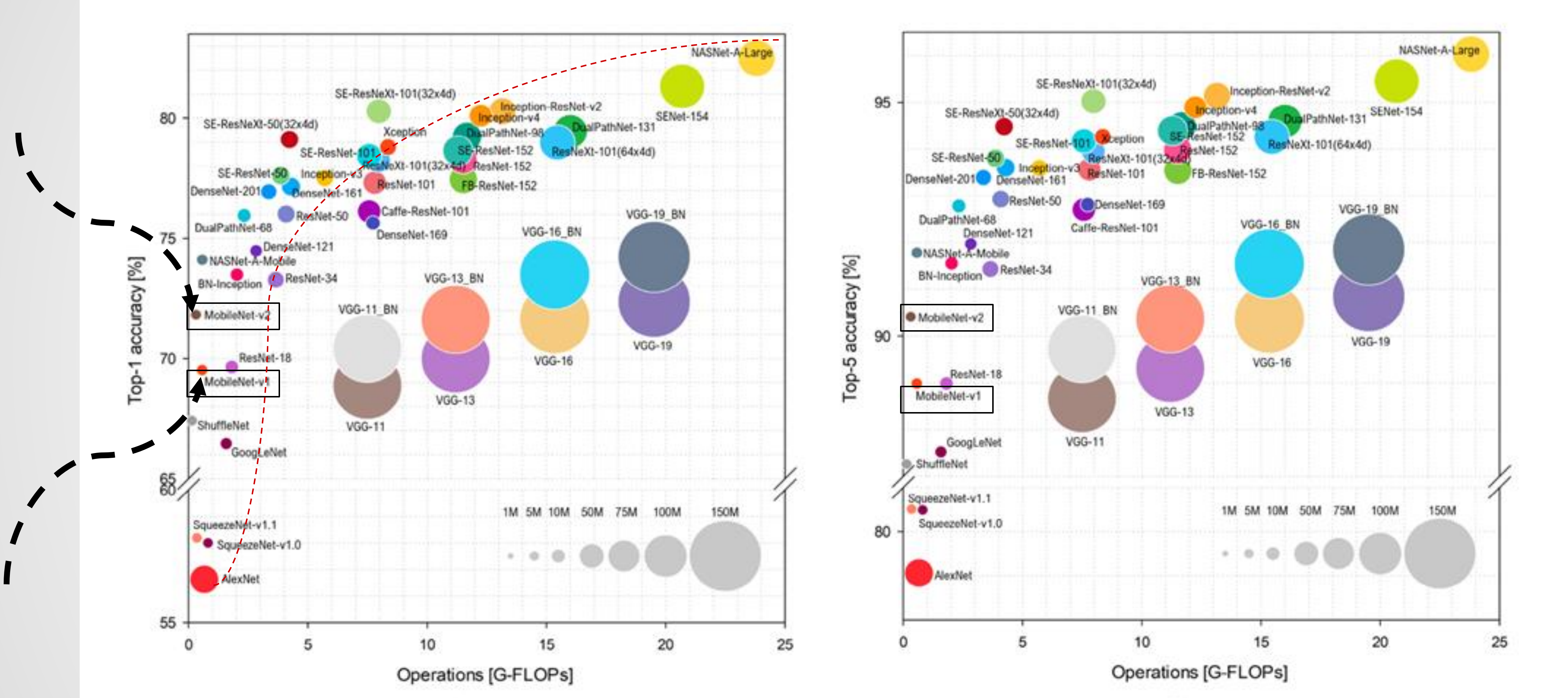
arXiv > cs > arXiv:1704.04861

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 17 Apr 2017]

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



(a) (b)

Efficient Dimensions

TÜRKİYE EURO 4SEE

- Compute Efficiency
  - Minimize the number of operations (e.g., FLOPs)
    → Faster inference/training, lower energy cost
- Memory Efficiency
  - Reduce model size (parameters) and runtime memory (activations)
  → Enables deployment on edge devices, fits more in (GPU) memory
- Latency & Hardware Friendliness
  - Optimize for speed on real hardware
  → Fewer sequential operations, better dataflow, hardware-aligned ops

through structural choices!

Next: Part IV

- Part I : Fundamentals
- Part II : Hardware Types & Memory Hierarchy
- Part III : Model-Level Optimizations
- **Part IV : System-Level Optimizations**
- Part V : Introduction to Scaling Deep Learning in HPC

Thanks!

Co-funded by
the European Union

EuroHPC
Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101191697. The JU receives support from the Digital Europe Programme and Germany, Türkiye, Republic of North Macedonia, Montenegro, Serbia, Bosnia and Herzegovina.