Towards Faster and Lighter Neural Network

심 재 헌

Vision & Display Systems Lab.

Dept. of Electronic Engineering, Sogang University

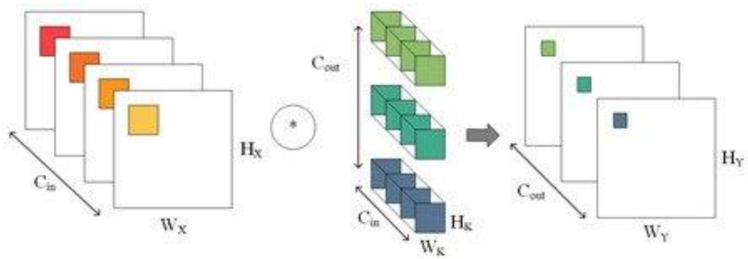
Outline

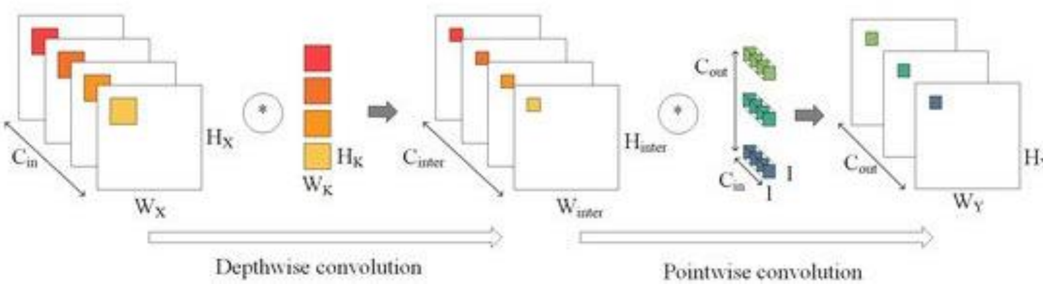
- Background
- MobileNeXt: Rethinking MobileNet
- Revisiting VGG structure
- RepVGG
- Conclusion

Background

- MobileNetV1: Depthwise Separable Convolution
  - Divide Standard Convolution into Depthwise & Pointwise Convolution

  - Parameters
    - Standard: $W_K \times H_K \times C_{out} \times C_{in}$
    - Depthwise: $(W_K \times H_K + C_{out}) \times C_{in}$
  - Computation
    - Standard: $W_K \times H_K \times C_{out} \times C_{in} \times W_Y \times H_Y$
    - Depthwise: $(W_K \times H_K + C_{out}) \times C_{in} \times W_Y \times H_Y$





Background

- ResNet: Bottleneck Layer
  - Used in ResNet
    - Reduce & Increase dimension within skip layer
    - To reduce computation as layers get deeper

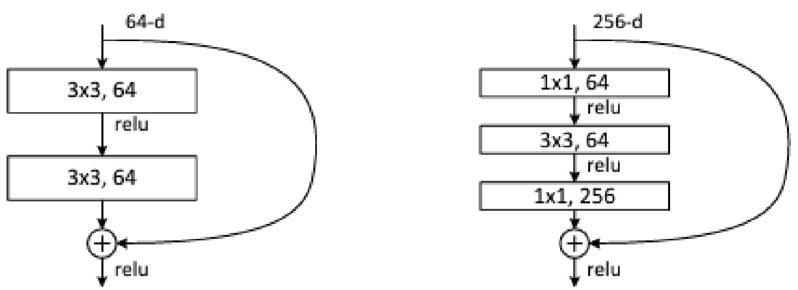
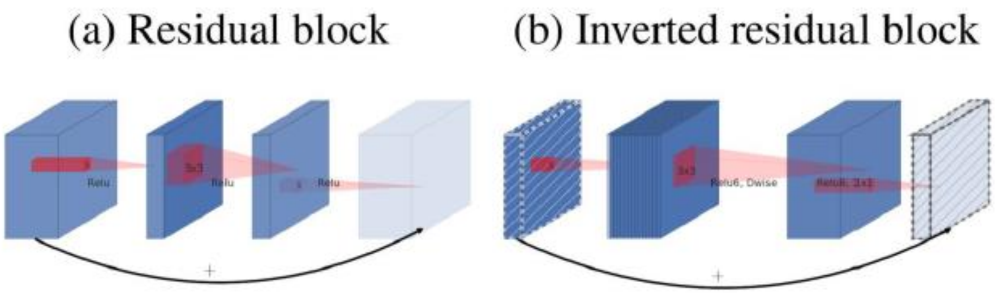


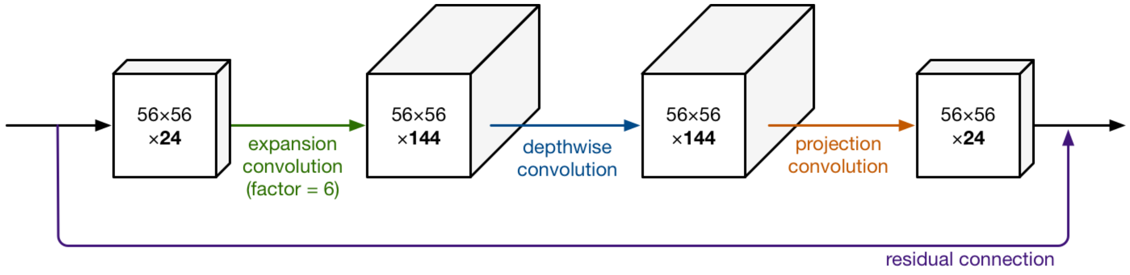
Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Background

- MobileNetV2: Inverted Residual Block
  - Residual (Bottleneck) vs Inverted Residual
    - Reverse classical Residual Block

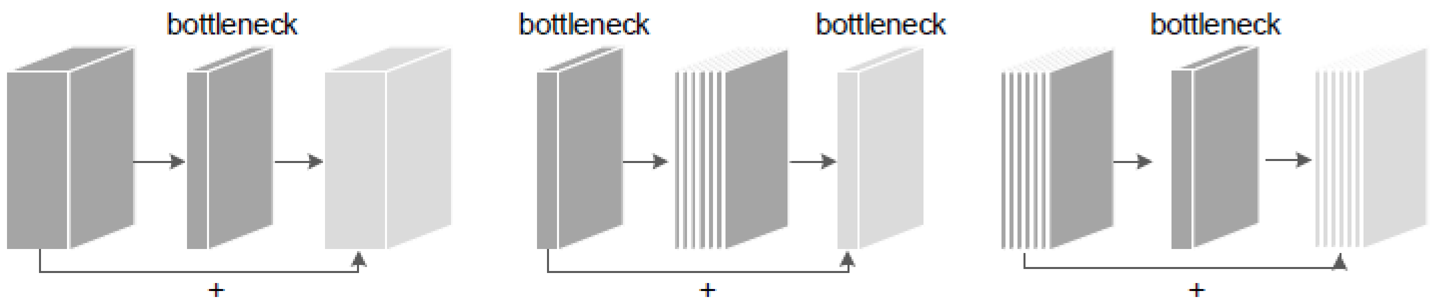


  - Inverted Residual Block Structure
    - Now a standard for many light-weight models (ex. ShuffleNet, EfficientNet, MobileNetV3 etc.)



MobileNeXt: Rethinking MobileNet

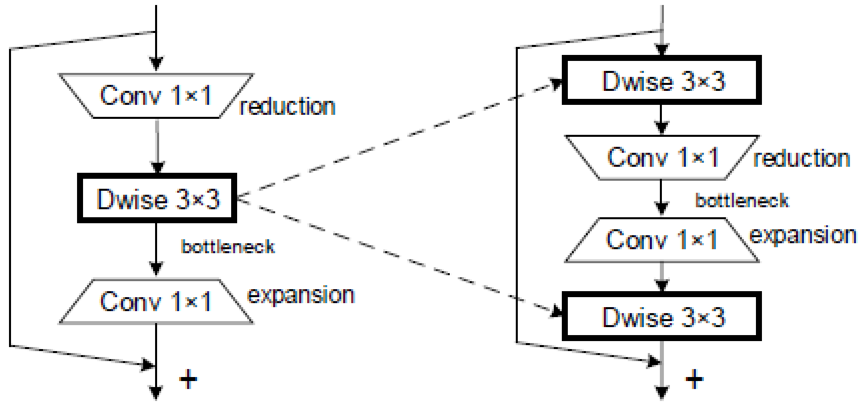
- Problems with MobileNetV2
  - Identity mapping (Skip connection) between low-dimensional representations
    - Inevitable Information Loss
    - Weakened propagation capability of gradients across layers
- Solution: Sandglass Block
  - Move identity mapping to high-dimensional representations
    - Propagates more gradients during training
    - Improved model performance



Bottleneck　　Inverted Residual　　Sandglass

MobileNeXt: Rethinking MobileNet

- Sandglass Block
  - Rethinking the positions of expansion and reduction layers
    - High-dimensional shortcuts
      - Wider shortcuts → more gradients propagate through multiple layers
    - Learning expressive spatial features
      - Less spatial information encoded when Depthwise conv is placed in the bottleneck
      - Depthwise convolution added at the beginning & end of bottleneck
      - Richer feature representation



Inverted Residual Block

Sandglass Block

MobileNeXt: Rethinking MobileNet

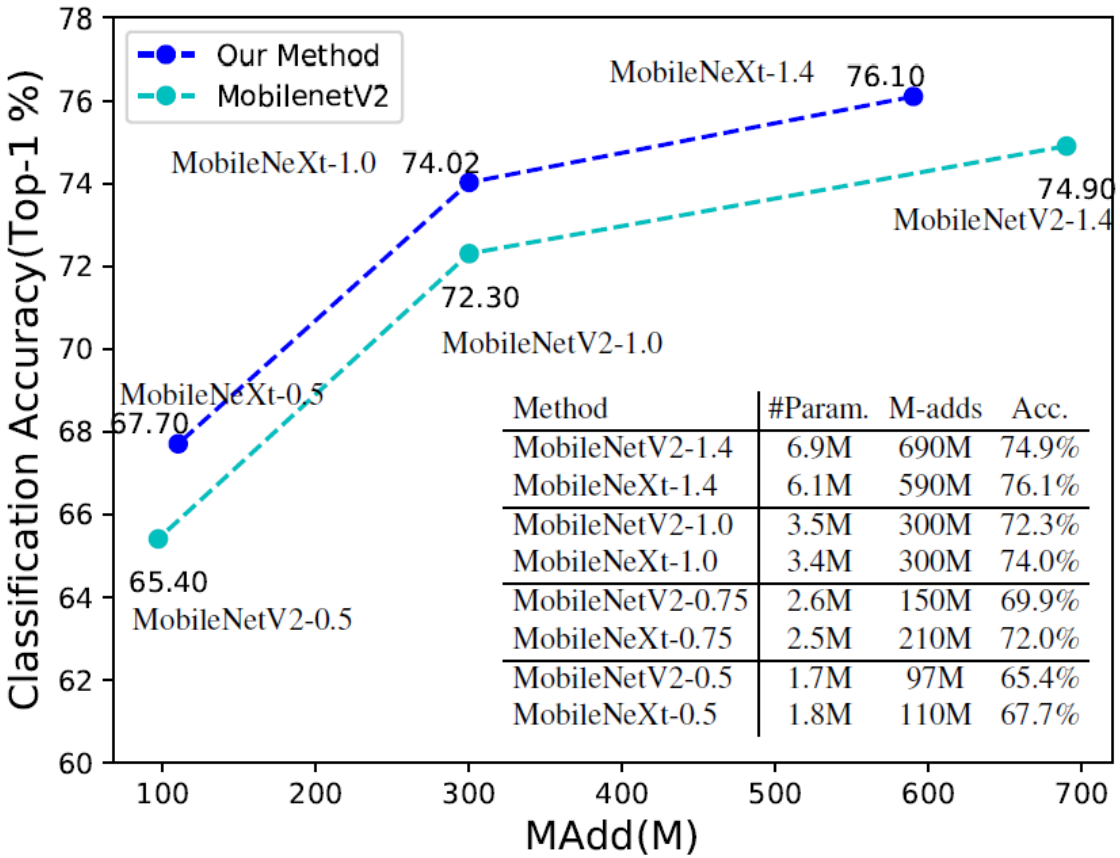
- MobileNeXt Architecture
  - Identity tensor multiplier
    - No need to keep the whole identity tensor in the skip layer
    - New hyper parameter: Identity tensor multiplier $\alpha \in [0,1]$
      - Control which portion of the residual path is preserved

$$G = \phi(F) + F \quad \Rightarrow \quad \begin{aligned} G_{1:\alpha M} &= \phi(F)_{1:\alpha M} + F_{1:\alpha M} \\ G_{\alpha M:M} &= \phi(F)_{\alpha M:M} \end{aligned}$$

  - Advantages
    - Reduced element-wise addition
      - better latency & almost no performance drop
    - MAC (Memory Access Cost)
      - Need to store input feature for skip connection
      - Less channel, less MAC

Experiments

- Comparison to MobileNetV2 on ImageNet Classification



Method	#Param.	M-adds	Acc.
MobileNetV2-1.4	6.9M	690M	74.9%
MobileNeXt-1.4	6.1M	590M	76.1%
MobileNetV2-1.0	3.5M	300M	72.3%
MobileNeXt-1.0	3.4M	300M	74.0%
MobileNetV2-0.75	2.6M	150M	69.9%
MobileNeXt-0.75	2.5M	210M	72.0%
MobileNetV2-0.5	1.7M	97M	65.4%
MobileNeXt-0.5	1.8M	110M	67.7%

Experiments

- Performance with Quantization

Model	Precision (W/A)	Method	Top-1 Acc. (%)
MobileNetV2	INT8/INT8	Post Training Quant.	65.07
MobileNeXt	INT8/INT8	Post Training Quant.	$68.62_{+3.55}$
MobileNetV2	FP32/FP32	-	72.25
MobileNeXt	FP32/FP32	-	$74.02_{+1.77}$

- Effect of high-dimensional skip layer
  - Add additional depthwise conv to Inverted Residual Block in MobileNetV2

Method	#Dwise convs	Param. (M)	M-Adds (M)	Top-1 Acc. (%)
MobileNetV2	2 (middle)	3.6	340	73.02
MobileNeXt	2 (top, bottom)	3.5	300	74.02

Experiments

- Superiority of Proposed Sandglass Block
  - Replace Inverted Residual Block with Sandglass Block from EfficientNet-b0

Models	Param. (M)	MAdd (M)	Top-1 Acc. (%)
MobilenetV1-1.0 [17]	4.2	575	70.6
MobilenetV2-1.0 [31]	3.5	300	72.3
MnasNet-A1 [34]	3.9	312	75.2
MobilenetV3-L-0.75 [16]	4.0	155	73.3
ProxylessNAS [1]	4.1	320	74.6
FBNet-B [38]	4.5	295	74.1
GhostNet-1.3 [10]	7.3	226	75.7
EfficientNet-b0 [35]	5.3	390	76.3
MobileNeXt-1.0	3.4	300	74.02
MobileNeXt-1.0†	3.94	330	76.05
MobileNeXt-1.1†	**4.28**	420	**76.7**

Experiments

- Latency measurement
  - TF-Lite on Pixel 4XL

MobileNetV2	68ms
MobileNeXt	66ms

- Effect of Identity Tensor Multiplier
  - Higher or almost no degradation up to $\alpha = 1/3$

No.	Models	Tensor multiplier	Param. (M)	Top-1 Acc. (%)	Latency (ms)
1	MobileNeXt	1.0	3.4	74.02	211
2	MobileNeXt	1/2	3.4	74.09	196
3	MobileNeXt	1/3	3.4	73.91	195
4	MobileNeXt	1/6	3.4	73.68	188

Other Applications

- Object Detection
  - MobileNeXt as backbone, tested on Pascal VOC 2007

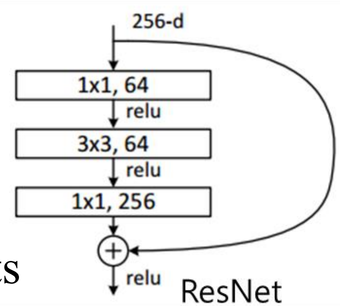
No.	Method	Backbone	Param. (M)	M-Adds (B)	mAP (%)
1	SSD300	VGG [33]	36.1	35.2	77.2
2	SSDLite320	MobileNetV2 [31]	4.3	0.8	71.7
3	SSDLite320	MobileNeXt	4.3	0.8	72.6

- Neural Architecture Search
  - Add Sanglass block to DARTS search space
    - Tested on CIFAR-10

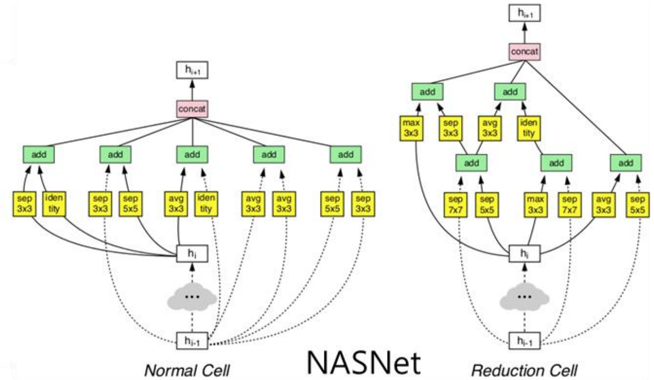
No.	Search Space	Test Error (%)	Param. (M)	Search Method	#Operators
1	DARTS original	3.11	3.25	gradient based	7
2	DARTS + IR Block	3.26	3.29	gradient based	8
3	DARTS + sandglass block	2.98	2.45	gradient based	8

Problems with Recent CNN Architectures

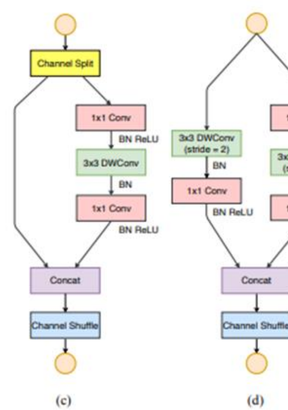
- Complicated Multi-branch Design
  - Residual connection (ResNet)
  - Branch concatenation (Inception, ShuffleNet)
  - Difficult implementation
- Inefficient Convolutional Components
  - Depthwise separable convolution (Xception, MobileNet)
  - Channel shuffle (ShuffleNet)
- FLOPs / # of parameters do not reflect inference time
  - ResNet: less parameters and FLOPS than VGG, yet slower



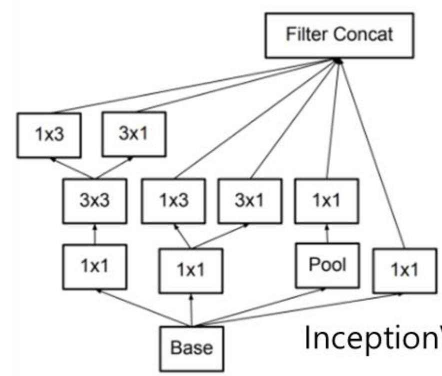
ResNet



NASNet



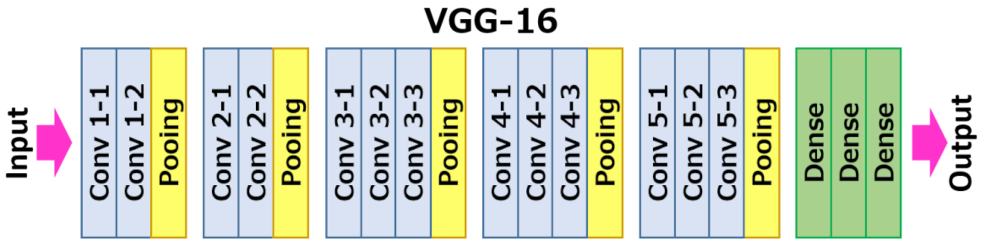
ShuffleNet



InceptionV3

Revisiting VGG structure

- Advantages
  - Simple Structure
    - Only, 3x3 standard convolution and Pooling
    - Single branch (no skip layers)
  - Fast Inference Time
    - Simple operations optimized for hardware platform (e.g. cuDNN,)
    - Less Memory Access Cost
- Drawbacks
  - Low performance
    - Gradient vanishing without skip layer (VGG-16 72% Top 1 Acc on ImageNet)

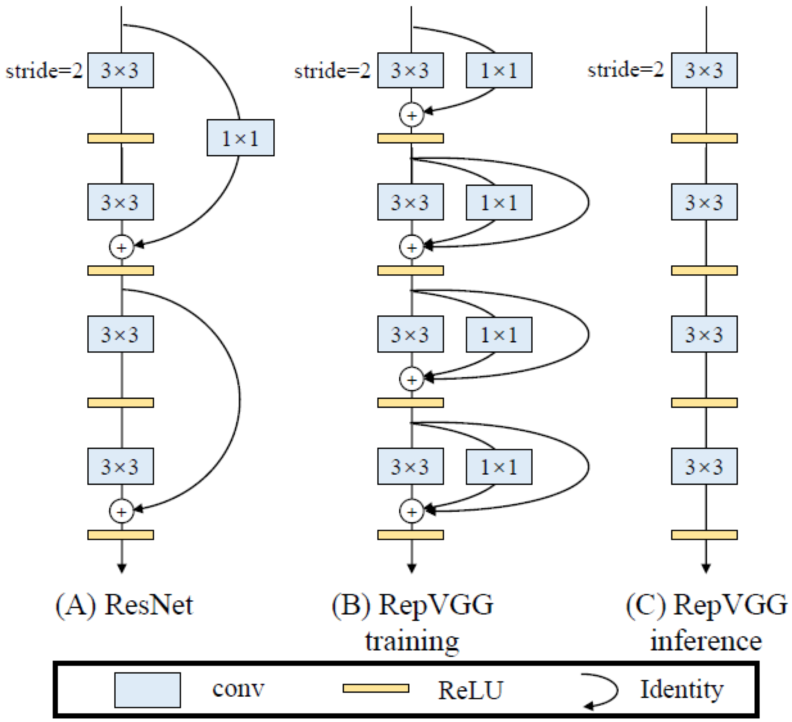


RepVGG

- RepVGG: Making VGG-style ConvNets Great Again

  - Decouple Training and Inference
    - Use multi-branch Ensemble structure for training
      - Skip connection & 1x1 conv
    - Exclude additional branch for fast inference

  - Structural Re-parameterization
    - Transform multi-branch parameters to single-path inference model
    - Combine 3x3, 1x1, skip layer into one 3x3 conv



RepVGG

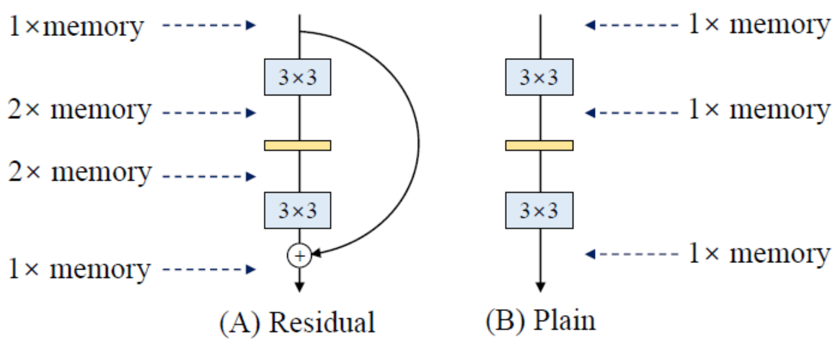
- Simple is Fast, Memory-economical, Flexible
  - Fast
    - VGG-16 has 8.4x FLOPs than EfficientNet-B3, yet 1.8x faster on 1080Ti
    - Winograd Convolution
      - 4/9 number of multiplication in 3x3 convolution
      - Supported in various hardware platforms (NVIDIA cuDNN, Intel MKL)
    - MAC (Memory Access Cost) & parallelism
      - Group conv & Multi-branch: Poor parallelism – high MACs
      - Fragmented operators hurt parallelism (e.g. NASNet-A 13 fragments)

Kernel size	Theoretical FLOPs (B)	Time usage (ms)	Theoretical TFLOPS
1×1	420.9	84.5	9.96
3×3	3788.1	198.8	**38.10**
5×5	10522.6	2092.5	10.57
7×7	20624.4	4394.3	9.38

FLOPs & Time of Convolution Operators on 1080Ti GPU
(TFLOPS: Tera Floating-point Operations Per Second)

RepVGG

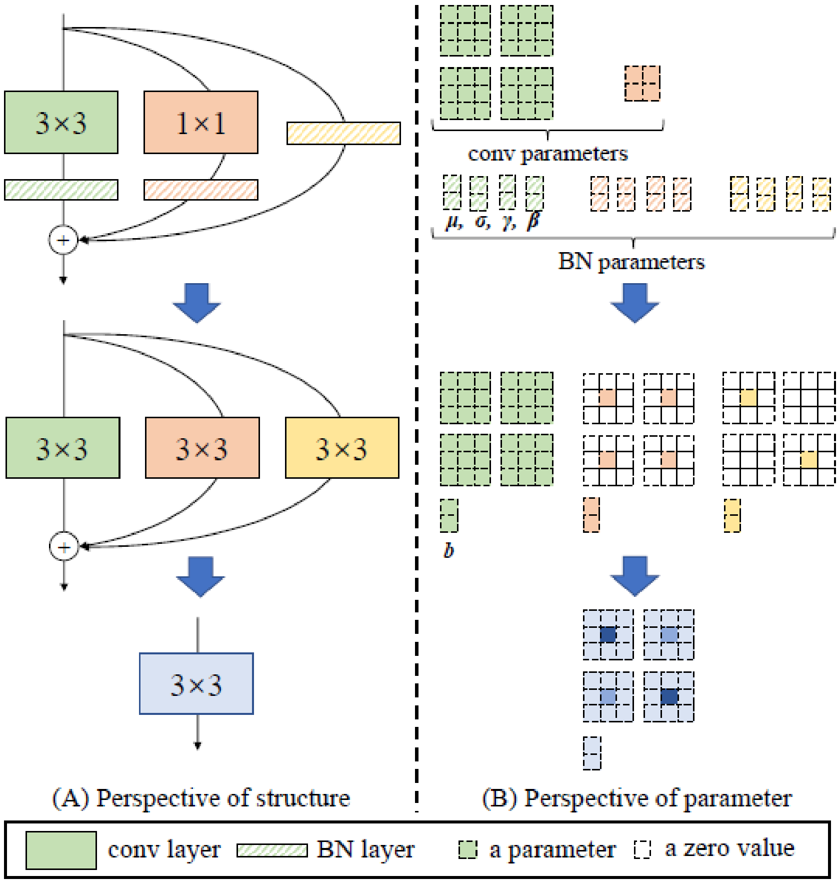
- Simple is Fast, Memory-economical, Flexible
  - Memory-economical
    - Multi-branch keeps input feature until addition / concatenation
    - High memory occupation
  - Flexible
    - Residual block – low flexibility since input and output feature need to be the same size
    - Limits channel pruning



Peak memory occupation in residual and plain block

RepVGG

- Structural Re-parameterization



Training

$$\begin{aligned} \mathrm{M}^{(2)} &= \mathrm{bn}(\mathrm{M}^{(1)} * \mathrm{W}^{(3)}, \boldsymbol{\mu}^{(3)}, \boldsymbol{\sigma}^{(3)}, \boldsymbol{\gamma}^{(3)}, \boldsymbol{\beta}^{(3)}) \\ &+ \mathrm{bn}(\mathrm{M}^{(1)} * \mathrm{W}^{(1)}, \boldsymbol{\mu}^{(1)}, \boldsymbol{\sigma}^{(1)}, \boldsymbol{\gamma}^{(1)}, \boldsymbol{\beta}^{(1)}) \\ &+ \mathrm{bn}(\mathrm{M}^{(1)}, \boldsymbol{\mu}^{(0)}, \boldsymbol{\sigma}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\beta}^{(0)}) \,. \end{aligned}$$

$$\boxed{\mathrm{W}'_{i,:,:,:} = \frac{\gamma_i}{\sigma_i}\mathrm{W}_{i,:,:,:}\,, \quad \mathrm{b}'_i = -\frac{\mu_i \gamma_i}{\sigma_i} + \beta_i}$$

Inference

Steps:

1. 1×1 *conv with identity mat* $=$ *identity*
2. 3×3 *conv* $= 1 \times 1$ *conv with zero padding*
3. *Add 3 kernels / 3 biases*

RepVGG

- Model Structure
  - Modification of classic settings of VGG
    - Different models according to width multiplier a, b
    - Only 3x3 conv and ReLU (no Max Pooling)
  - Groupwise convolution for further reduction of parameters and computation
    - RepVGG-A: 3^{rd}, 5^{th}, 7^{th}, … 21^{st} layers
    - RepVGG-B: 3^{rd}, 5^{th}, 7^{th}, … 21^{st}, 23^{rd}, 25^{th}, 27^{th} layers

Table 2: Architectural specification of RepVGG. *E.g.*, $2 \times 64a$ means stage2 has 2 layers each with $64a$ channels.

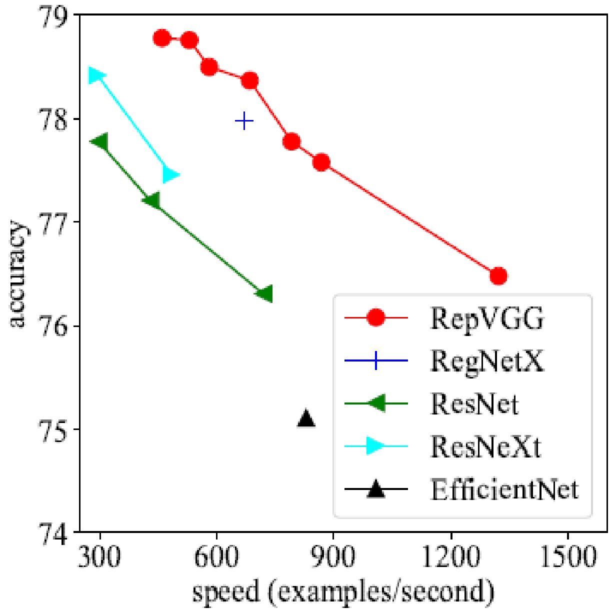
Stage	Output size	RepVGG-A	RepVGG-B
1	112×112	$1 \times \min(64, 64a)$	$1 \times \min(64, 64a)$
2	56×56	$2 \times 64a$	$4 \times 64a$
3	28×28	$4 \times 128a$	$6 \times 128a$
4	14×14	$14 \times 256a$	$16 \times 256a$
5	7×7	$1 \times 512b$	$1 \times 512b$

Table 3: RepVGG models defined by multipliers a and b.

Name	Layers of each stage	a	b
RepVGG-A0	1, 2, 4, 14, 1	0.75	2.5
RepVGG-A1	1, 2, 4, 14, 1	1	2.5
RepVGG-A2	1, 2, 4, 14, 1	1.5	2.75
RepVGG-B0	1, 4, 6, 16, 1	1	2.5
RepVGG-B1	1, 4, 6, 16, 1	2	4
RepVGG-B2	1, 4, 6, 16, 1	2.5	5
RepVGG-B3	1, 4, 6, 16, 1	3	5

Experiments

- ImageNet Classification
  - 1080Ti for speed eval
  - g: group convolution



Model	Top-1 acc	Speed	Params (M)	Theo FLOPs (B)	Wino MULs (B)
RepVGG-A0	72.41	3256	8.30	1.4	0.7
ResNet-18	71.16	2442	11.68	1.8	1.0
RepVGG-A1	74.46	2339	12.78	2.4	1.3
RepVGG-B0	75.14	1817	14.33	3.1	1.6
ResNet-34	74.17	1419	21.78	3.7	1.8
RepVGG-A2	76.48	1322	25.49	5.1	2.7
RepVGG-B1g4	77.58	868	36.12	7.3	3.9
EfficientNet-B0	75.11	829	5.26	0.4	-
RepVGG-B1g2	77.78	792	41.36	8.8	4.6
ResNet-50	76.31	719	25.53	3.9	2.8
RepVGG-B1	78.37	685	51.82	11.8	5.9
RegNetX-3.2GF	77.98	671	15.26	3.2	2.9
RepVGG-B2g4	78.50	581	55.77	11.3	6.0
ResNeXt-50	77.46	484	24.99	4.2	4.1
RepVGG-B2	78.78	460	80.31	18.4	9.1
ResNet-101	77.21	430	44.49	7.6	5.5
VGG-16	72.21	415	138.35	15.5	6.9
ResNet-152	77.78	297	60.11	11.3	8.1
ResNeXt-101	78.42	295	44.10	8.0	7.9

Experiments

- Structural Re-parameterization is the Key
  - DiracNet:
    - $\widehat{W} = diag(a)I + diag(b)W_{norm}$
  - Trivial Re-param:
    - Simpler version of DiracNet: $\widehat{W} = I + W$
  - Asymmetric Conv Block (ACB)
  - Residual Reorg
    - Re-organize parameter in ResNet-like manner

Model	Identity branch	1×1 branch	Accuracy
RepVGG-B0			72.39
RepVGG-B0	✓		74.79
RepVGG-B0		✓	73.15
RepVGG-B0	✓	✓	**75.14**

Variant and baseline	Accuracy
Identity w/o BN	74.18
Post-addition BN	73.52
Full-featured reparam	**75.14**
+ReLU in branch	75.69
DiracNet [38]	73.97
Trivial Re-param	73.51
ACB [9]	73.58
Residual Reorg	74.56

Applications

- Semantic Segmentation
  - PSPNet framework with RepVGG backbone
  - Tested on CityScapes dataset
  - RepVGG-B1g2-fast / B2-fast:
    - 3x3 dilated convolution only in the last 5 layers for fair comparison

Backbone	Mean IoU	Mean pixel acc	Speed
RepVGG-B1g2-fast	78.88	96.19	10.9
ResNet-50	77.17	95.99	10.4
RepVGG-B1g2	78.70	96.27	8.0
RepVGG-B2-fast	79.52	96.36	6.9
ResNet-101	78.51	96.30	6.7
RepVGG-B2	80.57	96.50	4.5

Conclusion

- MoileNeXt: Rethinking Bottleneck Structure for Efficient Mobile Network Design
  - Rethink the previous design rules of Inverted Residual Block
  - Sandglass Block
    - Shortcut connection between high-dimensional inputs
    - Outperform conventional light-weight networks with Inverted Residual Blocks

- RepVGG: Making VGG-style ConvNets Great Again
  - Simple architecture with only 3x3 conv and ReLU
    - Train only Multi-branch structure
    - Structural Re-parameterization
  - Limitations
    - Although fast, simple and practical, less concerns for number of parameters
    - MobileNet, ShuffleNet may be favored for low powered devices

Reference

- Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).
- Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
- Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International Conference on Machine Learning*. PMLR, 2019.
- Zhou, Daquan, et al. "Rethinking bottleneck structure for efficient mobile network design." *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer International Publishing, 2020.
- Zagoruyko, Sergey, and Nikos Komodakis. "Diracnets: Training very deep neural networks without skip-connections." *arXiv preprint arXiv:1706.00388* (2017).
- Lavin, Andrew, and Scott Gray. "Fast algorithms for convolutional neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- Ding, Xiaohan, et al. "Repvgg: Making vgg-style convnets great again." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.