RepVGG: Making VGG-style ConvNets Great Again

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, Jian Sun

Conference on Computer Vision and Pattern Recognition 2021

Last revised on arXiv on: 29 March 2021

Repository: https://github.com/DingXiaoH/RepVGG

Agenda

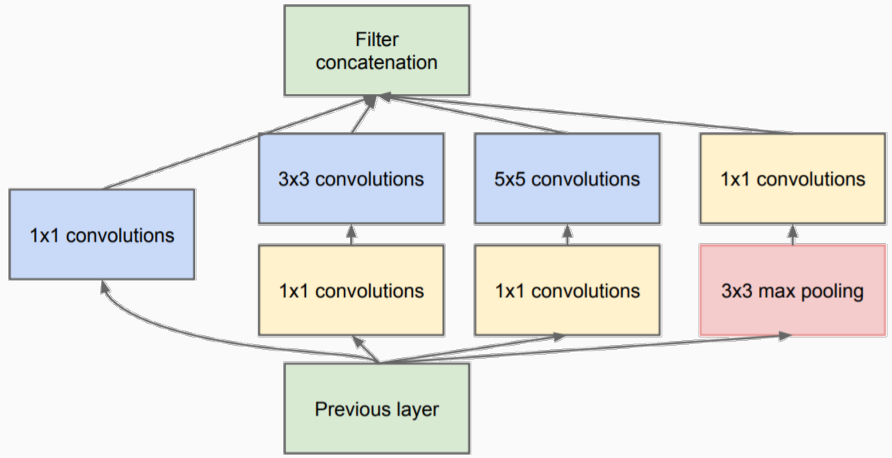
UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Motivation

VGG [1]

- Prefer deep CNNs with small receptive fields over shallow CNNs with bigger receptive fields
- Uses only simple convolutional, max-pooling and fully-connected layers
- ImageNet top-5 test error: 7.32

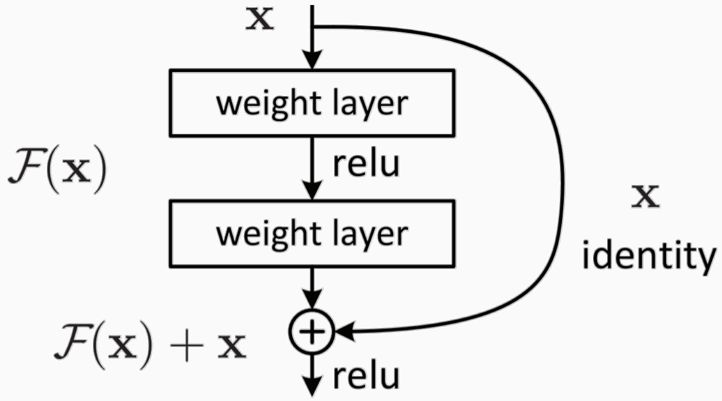
Inception (GoogLeNet) [2]



- ImageNet top-5 test error: 6.67

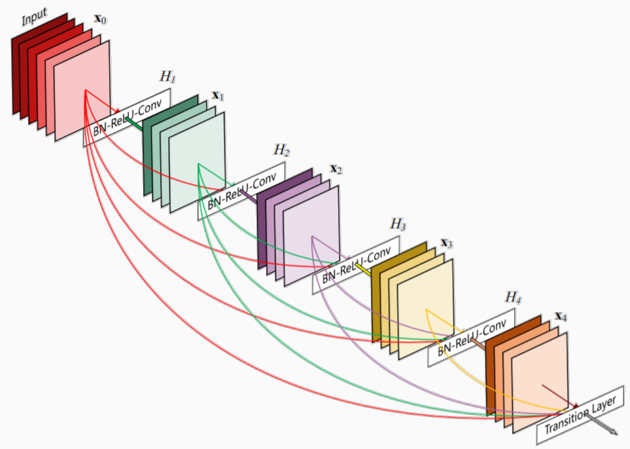
Motivation

ResNet [3]



- Solved the degradation problem by introducing shortcut connections
- ImageNet top-5 test error: 3.57

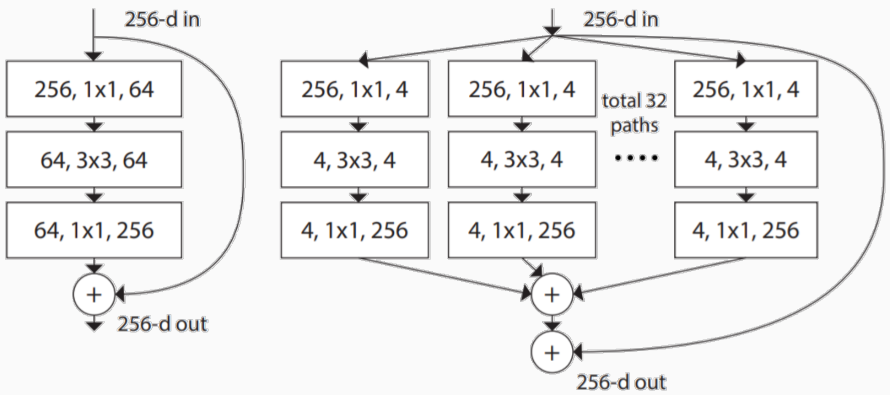
DenseNet [4]



- Stronger feature propagation through densely connected layers
- Outperforms ResNet on ImageNet val. dataset

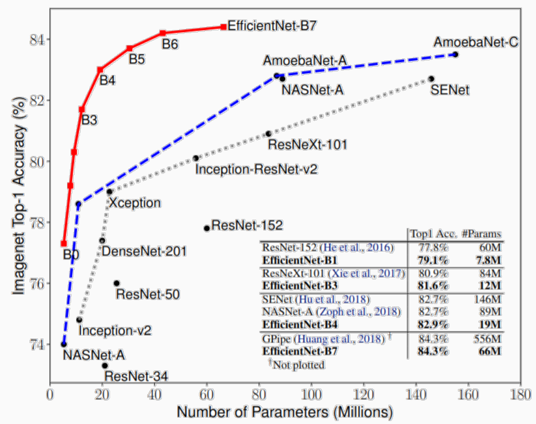
Motivation

ResNeXt [5]



- Cardinality dimension: size of the set of transformations
- Increase cardinality instead of depth or width

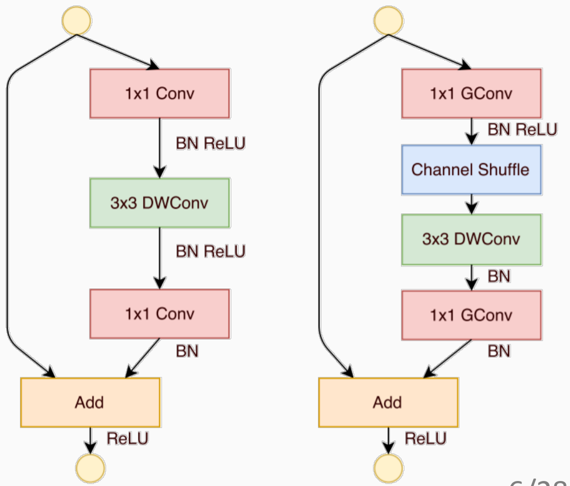
EfficientNet [6]



- Compound scaling method: uniform scaling in depth, width and resolution
- ImageNet top-1 accuracy: 84.3 (B7)

Motivation

- **Xception [7]:** depthwise separable convolutional layers
- **MobileNet [8]:** depthwise separable convolutional layers using pointwise convolutions, width multiplier, resolution multiplier
- **ShuffleNet [9]:** depthwise separable convolutional layers, pointwise group convolution, channel shuffling
- **NASNet [11]:** utilize neural architecture search
- **RegNet [12]:** design network design spaces
- **Evolutionary algorithms [13]**
- **...**



Motivation

Drawbacks - Speed

- Memory Access Costs (MAC) high for branch additions/concatenations, groupwise convolutions, depthwise separable convolutions and channel shuffling
- Degree of parallelism measured by the number of fragmented operators introduces synchronization overheads

➡ FLOPs cannot be used as a measure for speed [10]

Motivation

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Drawbacks – Memory Efficiency

- Multi-branch topology like residual architectures keep results of every branch until addition/concatenation

➡ Less computing units to be integrated onto the chip

Motivation

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Drawbacks – Flexibility

- Introduce architectural constraints: shape matching within residual blocks for final branch addition
- Limit the application of channel pruning

➡ Difficult to implement and customize

Motivation

Idea: RepVGG

- Multi-branch topology during training
- VGG-like plain inference-time architecture
- Transformation by structural re-parameterization
- Advantages:
  - VGG-like plain feed-forward topology using only 3x3 conv and ReLU
  - No automatic search, manual refinement, compound scaling or other heavy design methods
  - Fewer types of operators enable more computing units integrated onto the chip
  - Good accuracy-speed trade-off

Fundamentals

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Winograd's minimal filtering algorithm [14]:

$$F(2,3) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

$$m_1 = (d_0 - d_2)g_0 \qquad m_2 = (d_1 + d_2)\frac{g_0 + g_1 + g_2}{2}$$

$$m_4 = (d_1 - d_3)g_2 \qquad m_3 = (d_2 - d_1)\frac{g_0 - g_1 + g_2}{2}$$

Fundamentals

Winograd's minimal filtering algorithm [14]:

$$\boxed{Y = A^T\left[(Gg) \odot (B^T d)\right]}$$

$$A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$

$$B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$g = \begin{bmatrix} g_0 & g_1 & g_2 \end{bmatrix}^T$$

$$d = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \end{bmatrix}^T$$

Fundamentals

<u>Winograd‘s minimal filtering algorithm [14]:</u>

$$Y = A^T \left[[GgG^T] \odot [B^T dB] \right] A$$

- F(2x2,3x3) uses 16 mults, whereas normal convolution takes 36 mults
- Speedup by factor 2.25
- 3x3 concolutions highly optimized by modern computing libraries like cuDNN (by using Winograd‘s algorithm)

Training-Time Model

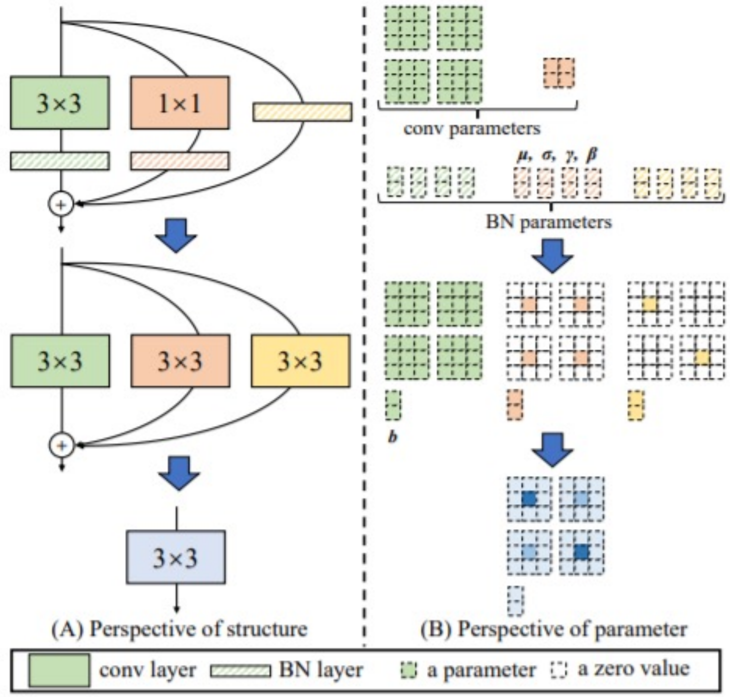


Figure 4: Structural re-parameterization of a RepVGG block. For the ease of visualization, we assume $C_2 = C_1 = 2$, thus the 3×3 layer has four 3×3 matrices and the kernel of 1×1 layer is a 2×2 matrix.

$$\boxed{y = x + g(x) + f(x)}$$

x: ResNet-like identity (dimensions must match)
g(x): 1x1 convolution (dimensions do not need to match)
f(x): residual function

Structural Re-Parameterization

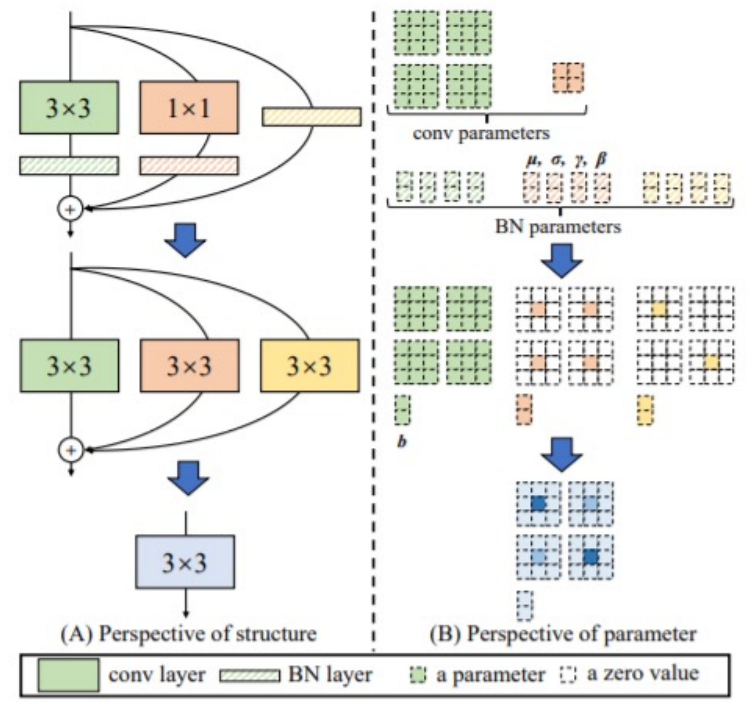


Figure 4: Structural re-parameterization of a RepVGG block. For the ease of visualization, we assume $C_2 = C_1 = 2$, thus the 3×3 layer has four 3×3 matrices and the kernel of 1×1 layer is a 2×2 matrix.

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

$$\mathrm{bn}(\mathrm{M}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\beta})_{:,i,:,:} = (\mathrm{M}_{:,i,:,:} - \boldsymbol{\mu}_i)\frac{\boldsymbol{\gamma}_i}{\boldsymbol{\sigma}_i} + \boldsymbol{\beta}_i \,. \tag{2}$$

$$\mathrm{W}'_{i,:,:,:} = \frac{\boldsymbol{\gamma}_i}{\boldsymbol{\sigma}_i}\mathrm{W}_{i,:,:,:} \,, \quad \mathbf{b}'_i = -\frac{\boldsymbol{\mu}_i \boldsymbol{\gamma}_i}{\boldsymbol{\sigma}_i} + \boldsymbol{\beta}_i \,. \tag{3}$$

$$\mathrm{bn}(\mathrm{M} * \mathrm{W}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\gamma}, \boldsymbol{\beta})_{:,i,:,:} = (\mathrm{M} * \mathrm{W}')_{:,i,:,:} + \mathbf{b}'_i \,. \tag{4}$$

Architectures

Table 2: Architectural specification of RepVGG. Here $2 \times 64a$ 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	$1 \times 512b$	$1 \times 512b$

- Interleave groupwise 3x3 convolution layer every two layers
  - trade accuracy for efficiency but still maintain inter-channel information exchange
  - Group factor: 1,2 or 4

Experiments

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

Table 5: Results on ImageNet trained in 200 epochs with Autoaugment [5], label smoothing and mixup.

Model	Acc	Speed	Params	FLOPs	MULs
RepVGG-B2g4	79.38	581	55.77	11.3	6.0
RepVGG-B3g4	80.21	464	75.62	16.1	8.4
RepVGG-B3	80.52	363	110.96	26.2	12.9
RegNetX-12GF	80.55	277	46.05	12.1	10.9
EfficientNet-B3	79.31	224	12.19	1.8	-

Table 4: Results trained on ImageNet with simple data augmentation in 120 epochs. The speed is tested on 1080Ti with a batch size of 128, full precision (fp32), and measured in examples/second. We count the theoretical FLOPs and Wino MULs as described in Sect. 2.4. The baselines are our implementations with the same training settings.

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

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Table 6: Ablation studies with 120 epochs on RepVGG-B0. The inference speed w/o re-param (examples/s) is tested with the models before conversion (batch size=128). Note again that all the models have the same final structure.

Identity branch	1×1 branch	Accuracy	Inference speed w/o re-param
		72.39	1810
✓		74.79	1569
	✓	73.15	1230
✓	✓	**75.14**	1061

Table 7: Comparison with variants and baselines on RepVGG-B0 trained in 120 epochs.

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 [39]	73.97
Trivial Re-param	73.51
ACB [10]	73.58
Residual Reorg	74.56

Excursus

Dirac weight parameterization [15]:

$$\hat{\mathbf{W}} = \mathrm{diag}(\mathbf{a})\mathbf{I} + \mathrm{diag}(\mathbf{b})\mathbf{W}_{\mathrm{norm}}$$

$$\mathbf{y} = \sigma\big((\mathbf{I} + \mathbf{W}) \odot \mathbf{x}\big) = \sigma\big(\mathbf{x} + \mathbf{W} \odot \mathbf{x}\big)$$

- No real multi-branch model during training-time
- Does not outperform ResNet

Experiments

Table 6: Ablation studies with 120 epochs on RepVGG-B0. The inference speed w/o re-param (examples/s) is tested with the models before conversion (batch size=128). Note again that all the models have the same final structure.

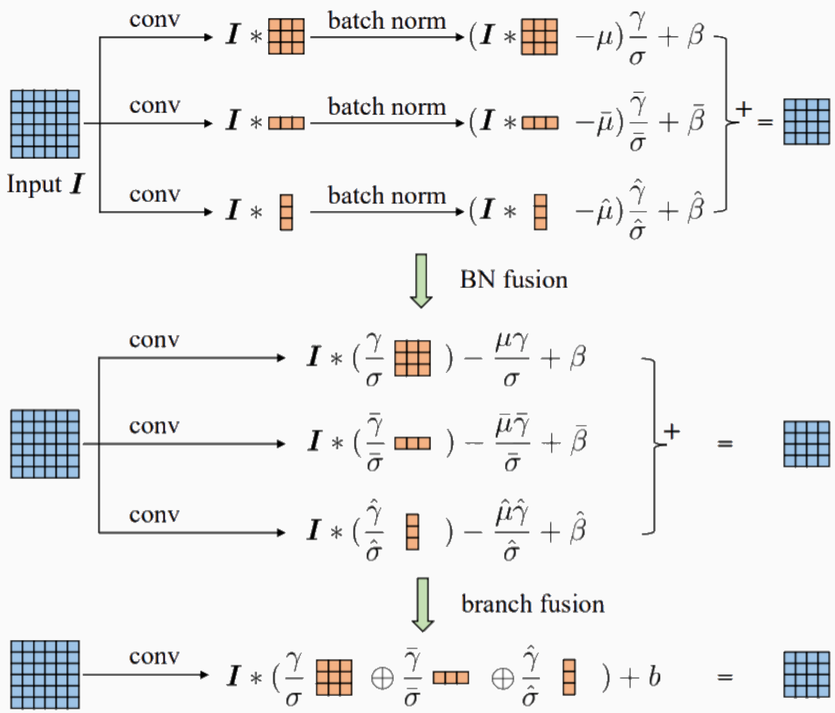
Identity branch	1×1 branch	Accuracy	Inference speed w/o re-param
		72.39	1810
✓		74.79	1569
	✓	73.15	1230
✓	✓	**75.14**	1061

Table 7: Comparison with variants and baselines on RepVGG-B0 trained in 120 epochs.

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 [39]	73.97
Trivial Re-param	73.51
ACB [10]	73.58
Residual Reorg	74.56

Excursus

Asymmetric Convolution Block (ACB) [16]:



Experiments

Table 6: Ablation studies with 120 epochs on RepVGG-B0. The inference speed w/o re-param (examples/s) is tested with the models before conversion (batch size=128). Note again that all the models have the same final structure.

Identity branch	1×1 branch	Accuracy	Inference speed w/o re-param
		72.39	1810
✓		74.79	1569
	✓	73.15	1230
✓	✓	**75.14**	1061

Table 7: Comparison with variants and baselines on RepVGG-B0 trained in 120 epochs.

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 [39]	73.97
Trivial Re-param	73.51
ACB [10]	73.58
Residual Reorg	74.56

Highlights and Weaknesses

Highlights:

- Proof that plain ConvNets can outperform ResNet-like architectures
- Novel re-parameterization technique based on parameters of a structure to parameterize another structure
- Accuracy-speed trade-off
- Optimized for GPUs and other specialized hardware
- Simple to implement

Highlights and Weaknesses

Weaknesses:

- Parameter inefficient compared to „modern“ architectures like EfficientNet or RegNetX (less favored than mobile-regime MobileNets and ShuffleNets for low-power devices)
- Models they compare with are biased for better positioning
- Additional restrictions: 3x3 convolutions, equal stride, padding

Conclusion

Making VGG-style ConvNets Great Again?

Bibliography

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

[1] Andrew Zisserman, Karen Simonyan. „Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014. https://arxiv.org/pdf/1409.1556.pdf.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. „Going Deeper with Convolutions", 2014. https://arxiv.org/pdf/1409.4842.pdf.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. „Deep Residual Learning for Image Recognition", 2015. https://arxiv.org/pdf/1512.03385.pdf.

[4] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. „Densely Connected Convolutional Networks", 2016. https://arxiv.org/pdf/1608.06993.pdf.

[5] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. „Aggregated Residual Transformations for Deep Neural Networks", 2016. https://arxiv.org/pdf/1611.05431.pdf.

[6] Quoc V. Le Mingxing Tan. „EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", 2019. https://arxiv.org/pdf/1905.11946.pdf.

Bibliography

[7] Franc¸ois Chollet. „Xception: Deep Learning with Depthwise Separable Convolutions", 2016. https://arxiv.org/pdf/1610.02357.pdf.

[8] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. „MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", 2017. https://arxiv.org/pdf/1704.04861.pdf.

[9] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun. „ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices", 2017. https://arxiv.org/pdf/1707.01083.pdf.

[10] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun. „ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design", 2018. https://arxiv.org/pdf/1807.11164.pdf.

[11] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le. „Learning transferable architectures for scalable image recognition", 2017. https://arxiv.org/pdf/1707.07012.pdf.

[12] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, Piotr Dollar. „Designing network design spaces", 2020. https://arxiv.org/pdf/2003.13678.pdf.

Bibliography

[13] Esteban Real, Alok Aggarwal, Yanping Huang, Quoc V Le. „Regularized evolution for image classifier architecture search", 2018. https://arxiv.org/pdf/1802.01548.pdf.

[14] Andrew Lavin, Scott Gray. „Fast Algorithms for Convolutional Neural Networks", 2015. https://arxiv.org/pdf/1509.09308.pdf.

[15] Nikos Komodakis Sergey Zagoruyko. „Diracnets: Training very deep neural networks without skip-connections", 2017. https://arxiv.org/pdf/1706.00388.pdf.

[16] Xiaohan Ding, Yuchen Guo, Guiguang Ding, Jungong Han. „Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks", 2019. https://arxiv.org/pdf/1908.03930.pdf.