RECENT ADVANCES IN CNN MODEL RESEARCH

Xiangyu Zhang
Research-model Team

Outline

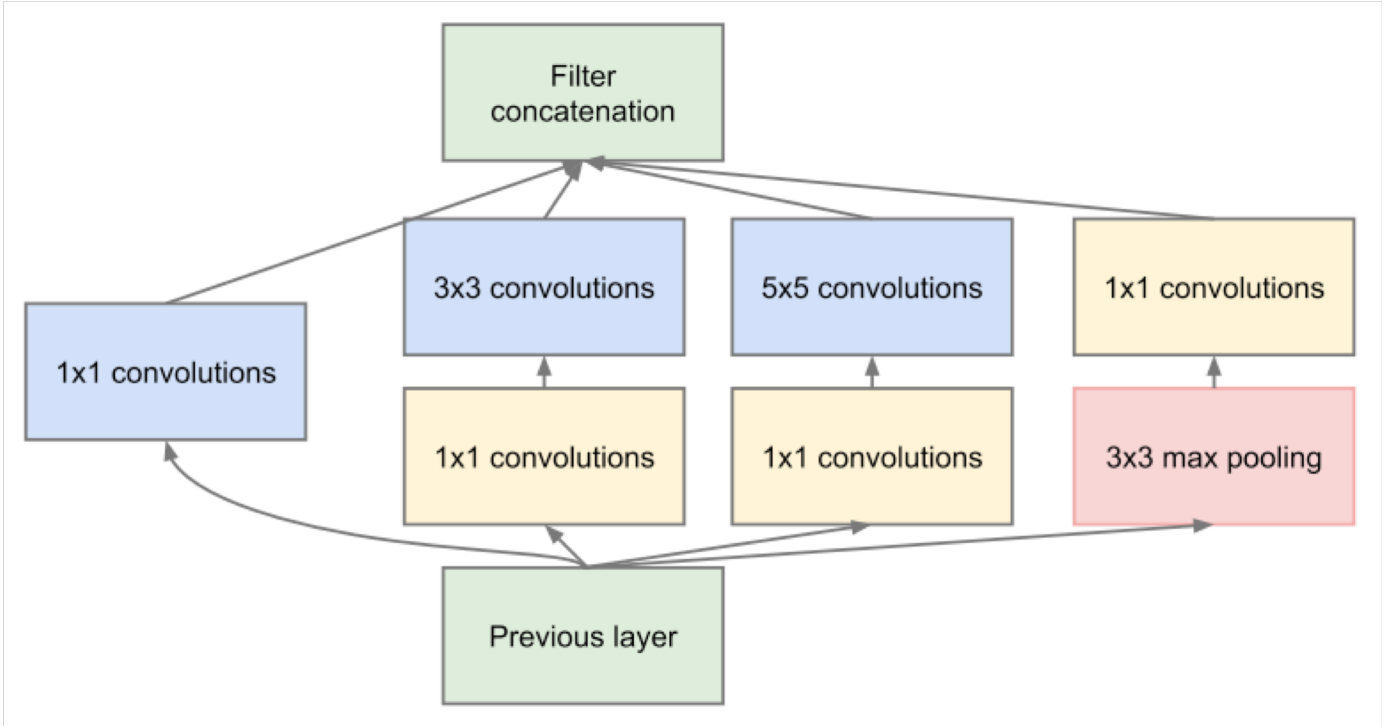
- Architecture related
- Other topics

Outline

- Architecture related
  - *Tensor decomposition*
  - *Channel shuffle*
  - *Inverted bottleneck*
  - *Spatial operations*
  - *Densely connections*
  - *Attention blocks*
  - *Normalization*
  - *Inference efficiency*
- Other topics

Tensor Decomposition

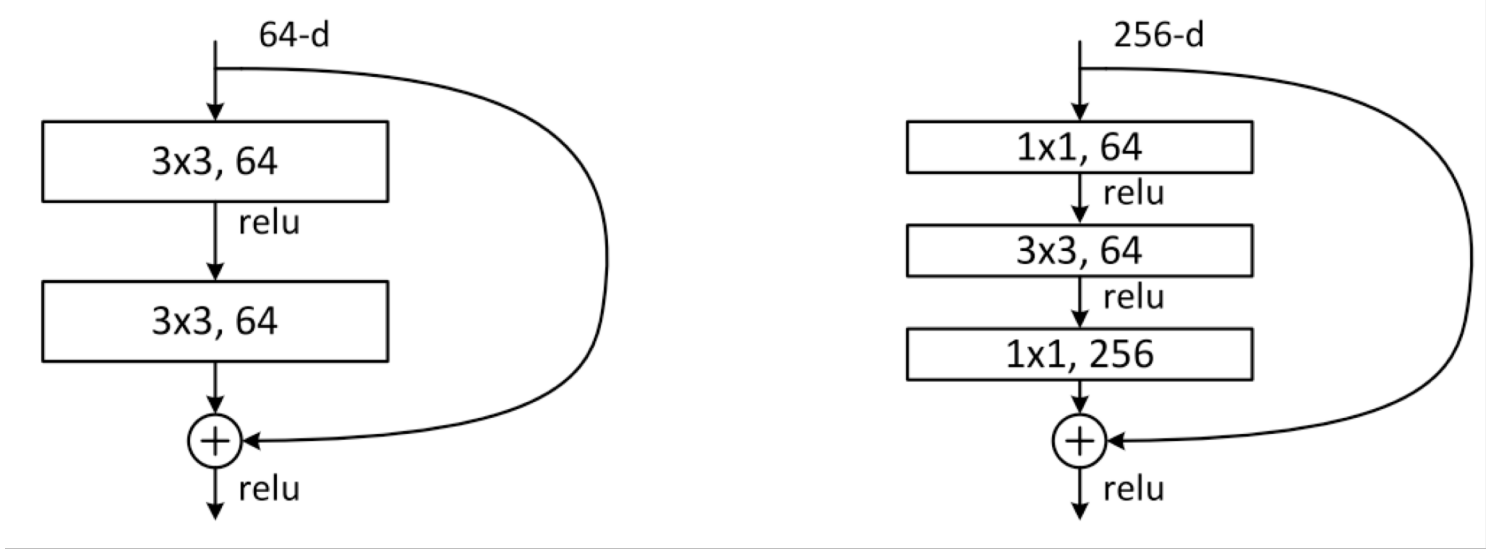
- 1x1 + KxK
  - *GoogleNet*



Szegedy, Christian, et al. "Going deeper with convolutions." Cvpr, 2015.

Tensor Decomposition

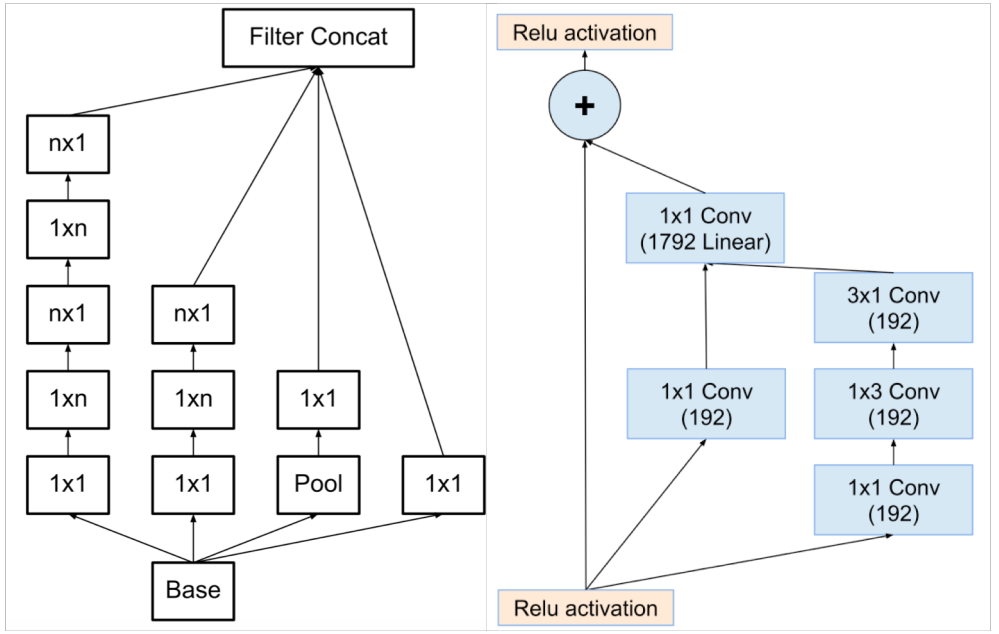
- 1x1 + KxK + 1x1
  - *Bottlenecks*

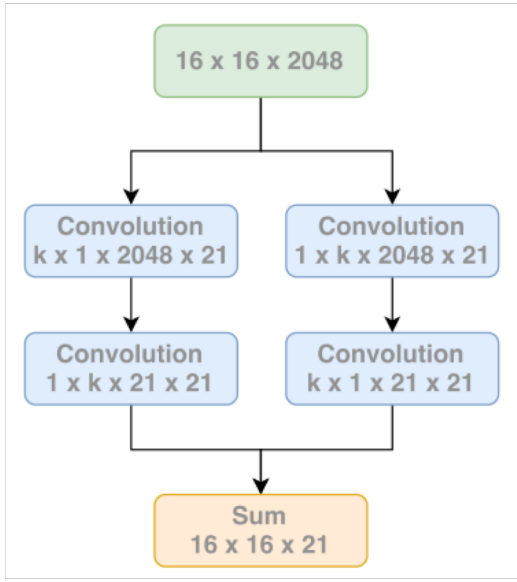


He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition.

Tensor Decomposition

- 1xM + Mx1
  - *Inception v3, v4*
  - *“Large kernel matters”*





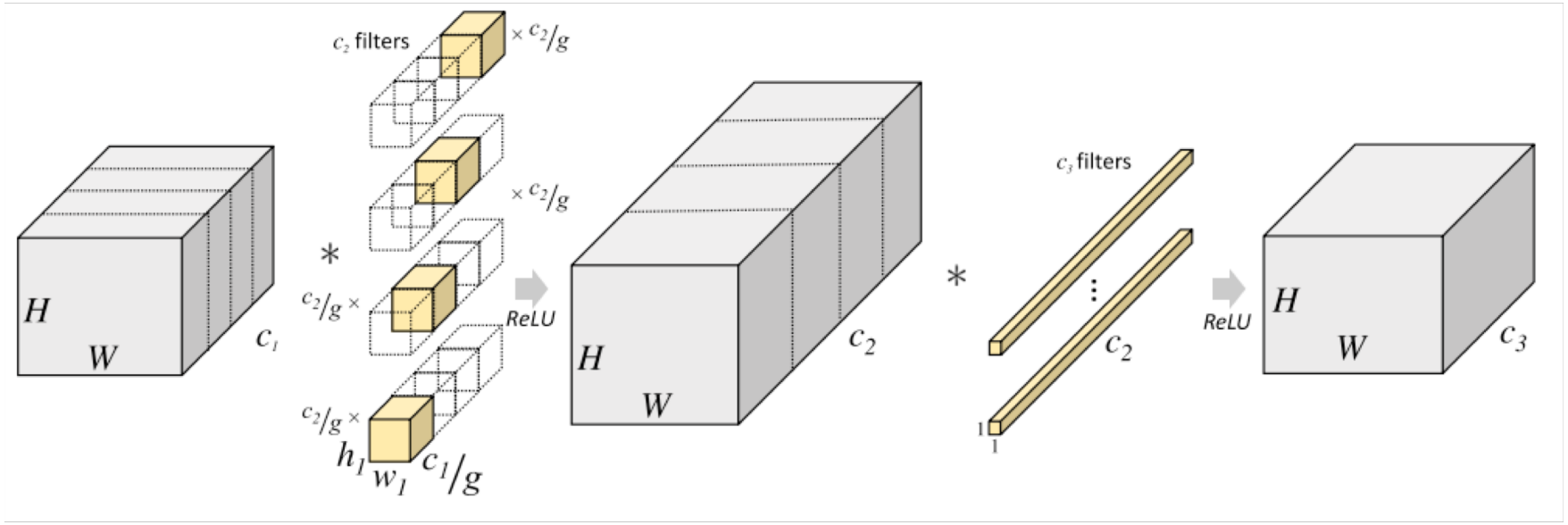
Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Cvpr*. 2016.

Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." *AAAI*. Vol. 4. 2017.

Peng, Chao, et al. "Large Kernel Matters--Improve Semantic Segmentation by Global Convolutional Network.” *Cvpr 2017*

Tensor Decomposition

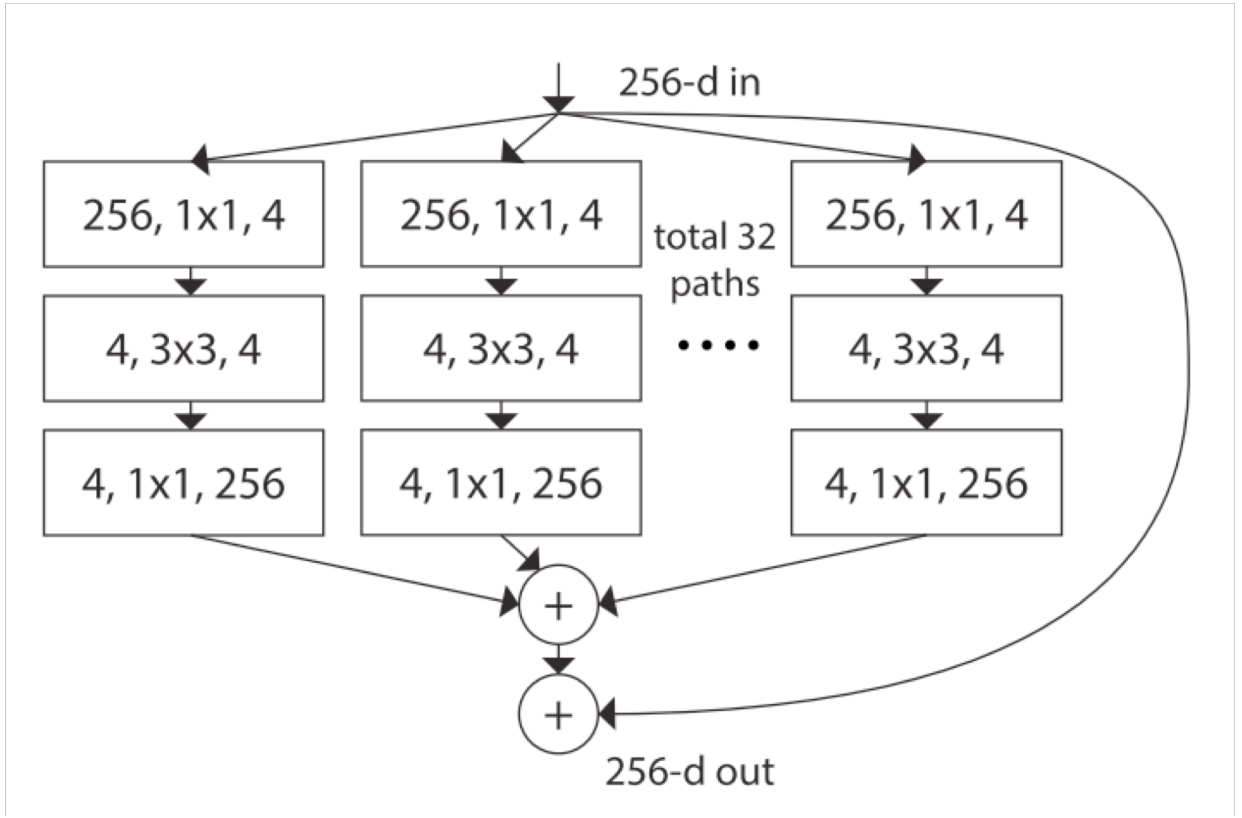
- GConv + 1x1
  - *DeepRoots*



Ioannou, Yani, et al. "Deep roots: Improving CNN efficiency with hierarchical filter groups." (2017).

Tensor Decomposition

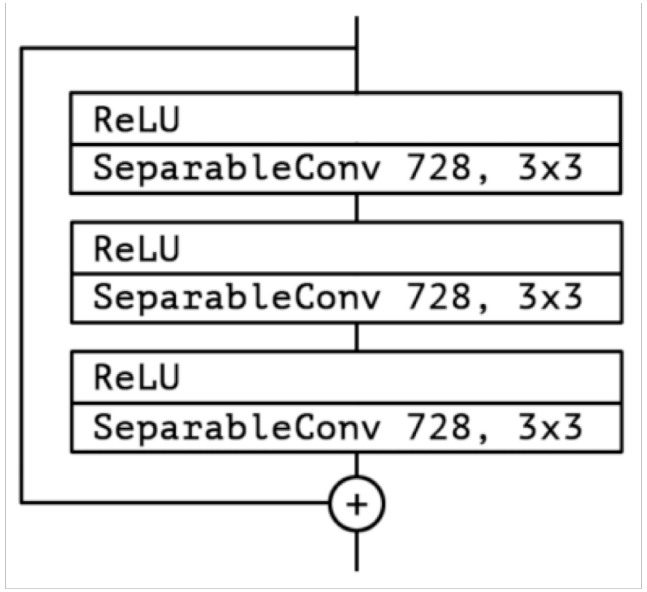
- 1x1 + GConv + 1x1
  - *ResNeXt*
- Applications
  - *For competitions (with SE)*
  - *Liveness*

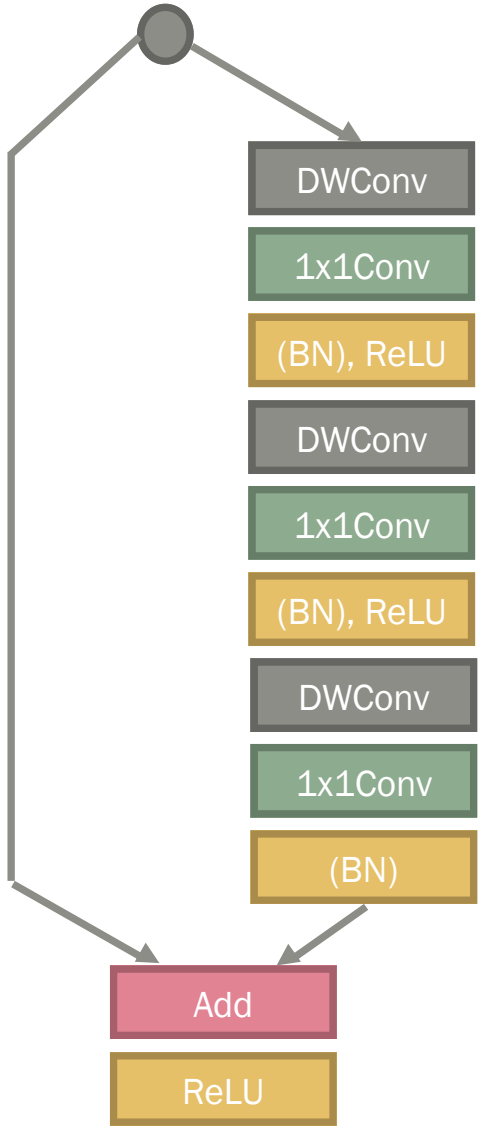


Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." *(CVPR)*

Tensor Decomposition

- DWConv + 1x1
  - *Xception*
  - *Light-head DET*
  - *Waterfall Xception*





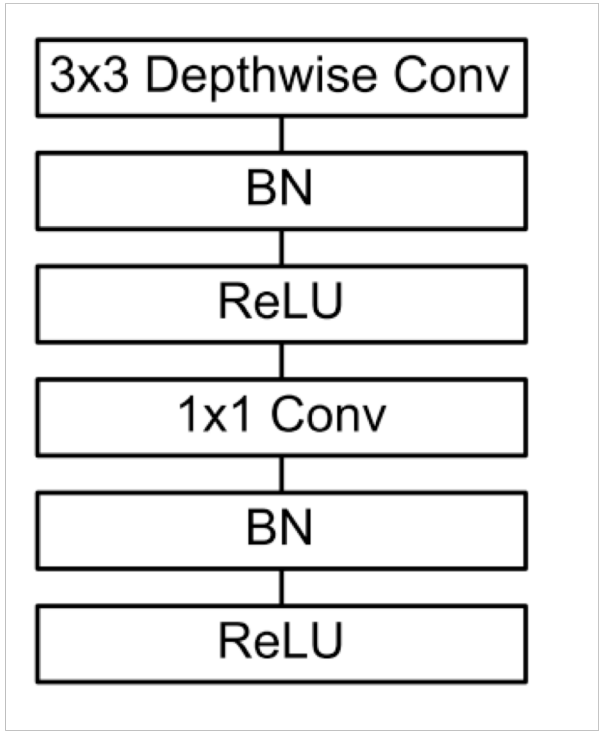
- Applications
  - *Large RF required (e.g. segmentation)*

Chollet, François. "Xception: Deep learning with depthwise separable convolutions."

Li, Zeming, et al. "Light-Head R-CNN: In Defense of Two-Stage Object Detector."

Tensor Decomposition

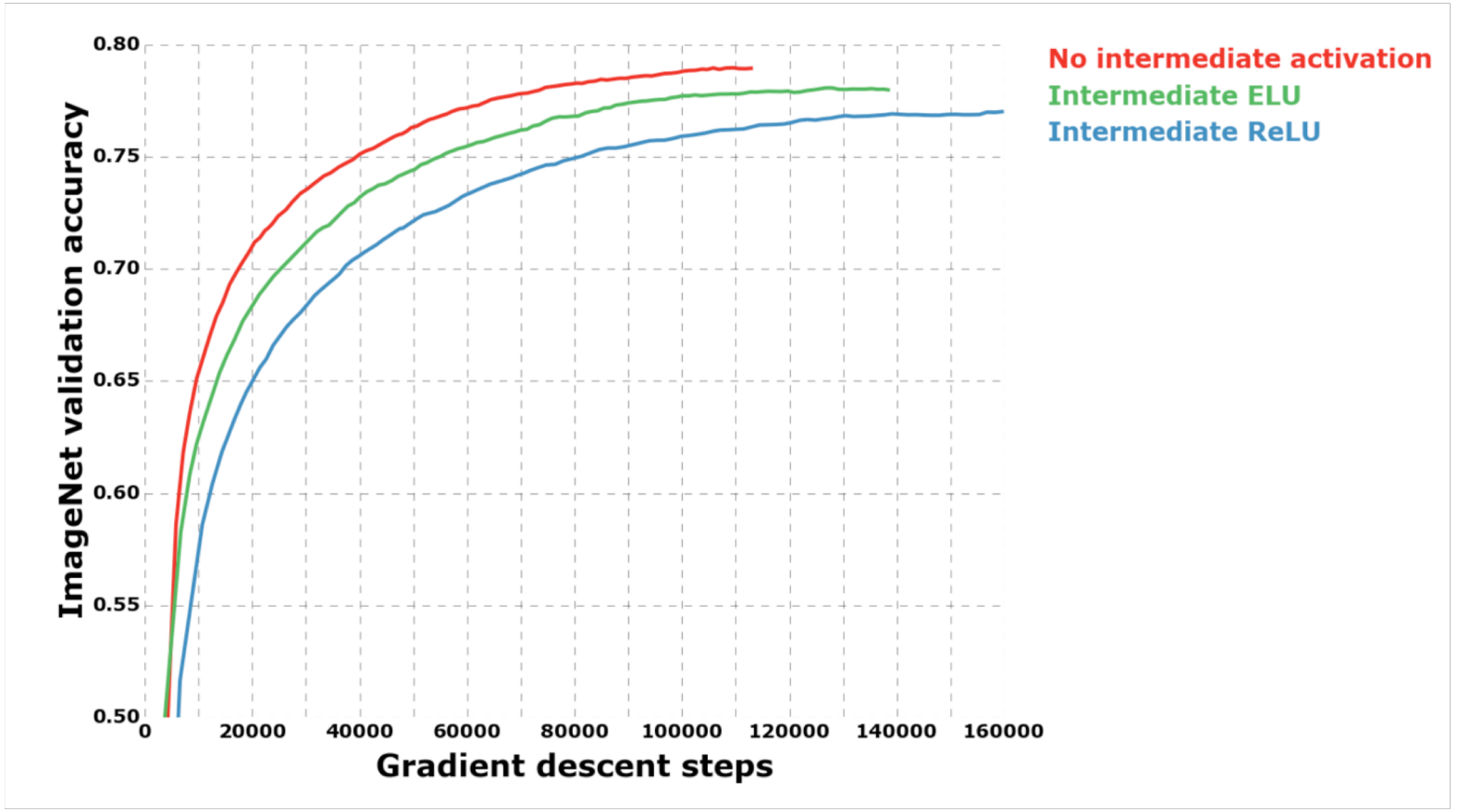
- DWConv + 1x1
  - *MobileNet v1*
- Applications
  - *None, unless none of other components supported*



Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications."

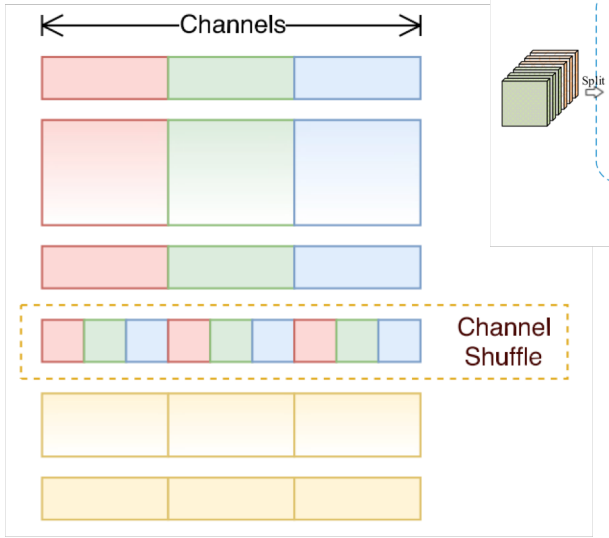
Tensor Decomposition

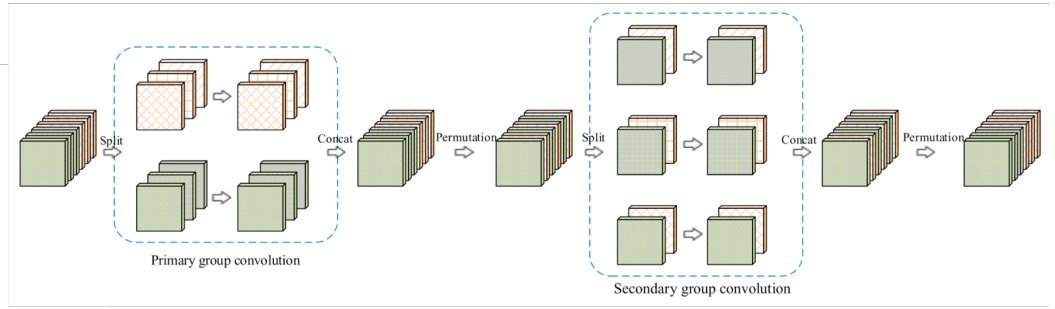
- Some details
  - *With/without BN?*
  - *With/without ReLU?*

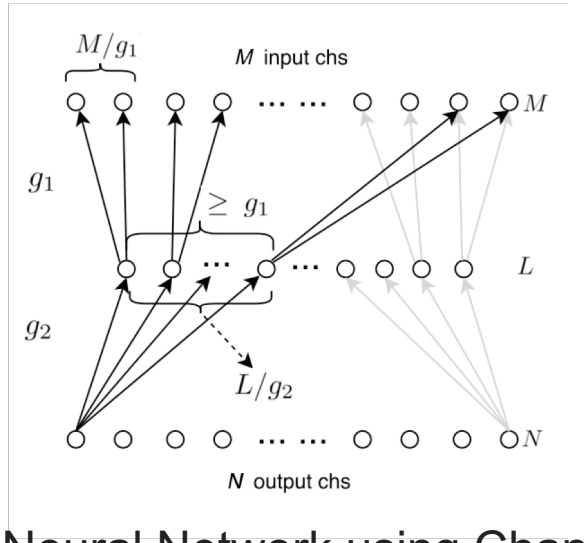


Channel Shuffle

- GConv + Channel Shuffle
  - *ShuffleNet*
  - *IGCNet*
  - *CLCNet*







Zhang, X.*, Zhou, X.*, Lin, M. and Sun, J., 2017. Shufflenet: An extremely efficient convolutional neural network for mobile devices

Zhang, Dong-Qing. "clcNet: Improving the Efficiency of Convolutional Neural Network using Channel Local Convolutions."

Zhang, T., et al. Interleaved Group Convolutions for Deep Neural Networks

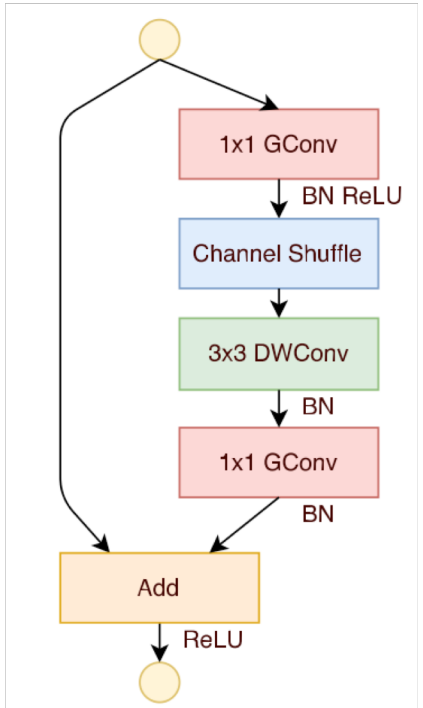
Channel Shuffle

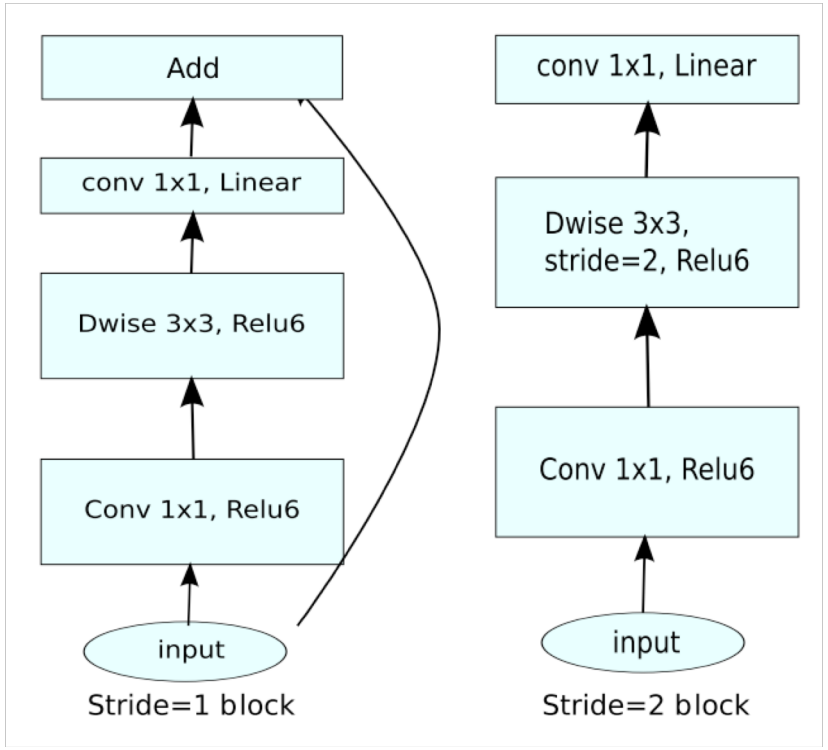
- Network configuration
  - *"fast downsampling"*

Layer	Output size	KSize	Stride	Repeat	Output channels (g groups)				
					$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 8$
Image	224×224				3	3	3	3	3
Conv1	112×112	3×3	2	1	24	24	24	24	24
MaxPool	56×56	3×3	2						
Stage2[1]	28×28		2	1	144	200	240	272	384
	28×28		1	3	144	200	240	272	384
Stage3	14×14		2	1	288	400	480	544	768
	14×14		1	7	288	400	480	544	768
Stage4	7×7		2	1	576	800	960	1088	1536
	7×7		1	3	576	800	960	1088	1536
GlobalPool	1×1	7×7							
FC					1000	1000	1000	1000	1000
Complexity[2]					143M	140M	137M	133M	137M

Bottleneck vs. Inverted Bottleneck

- MobileNet v2
  - *Linear bottleneck*
  - *Inverted residual*
  - *ReLU6*





Sandler, Mark, et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation."

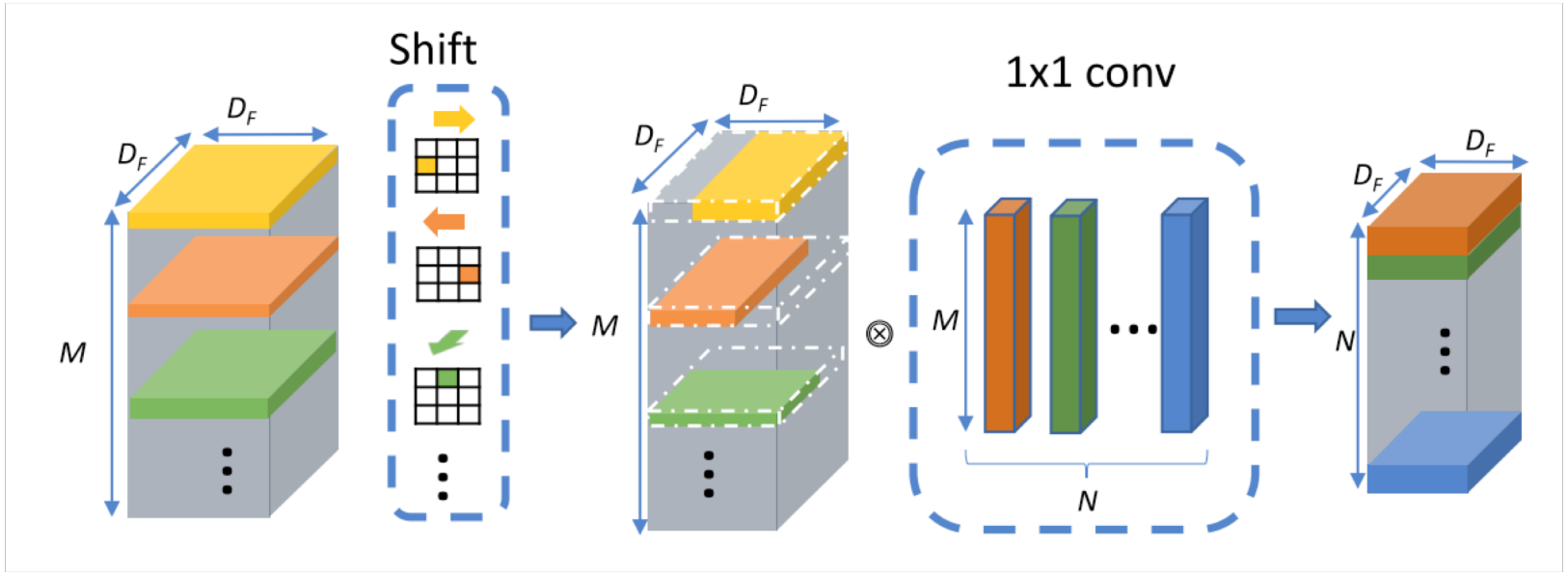
Bottleneck vs. Inverted Bottleneck

- Trick in MobileNet v2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$28^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times k$	conv2d 1x1	-	k	-	

Spatial Operators

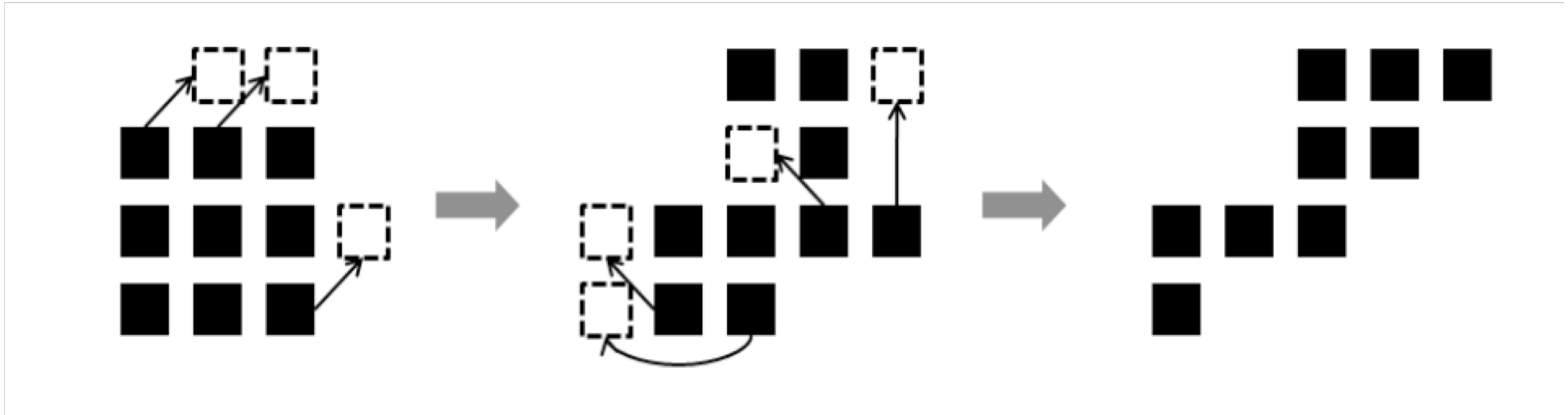
- “Shift” operator



Wu, Bichen, et al. "Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions."

Spatial Operators

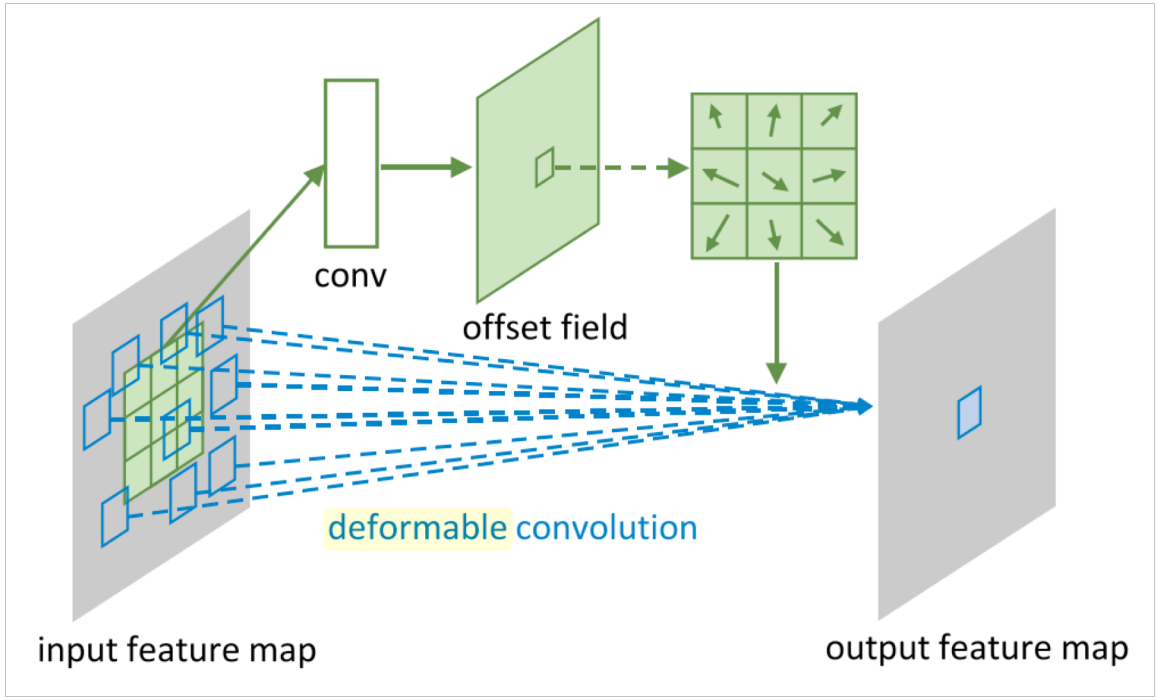
- Irregular convolution



Ma, Jiabin, Wei Wang, and Liang Wang. "Irregular Convolutional Neural Networks."

Spatial Operators

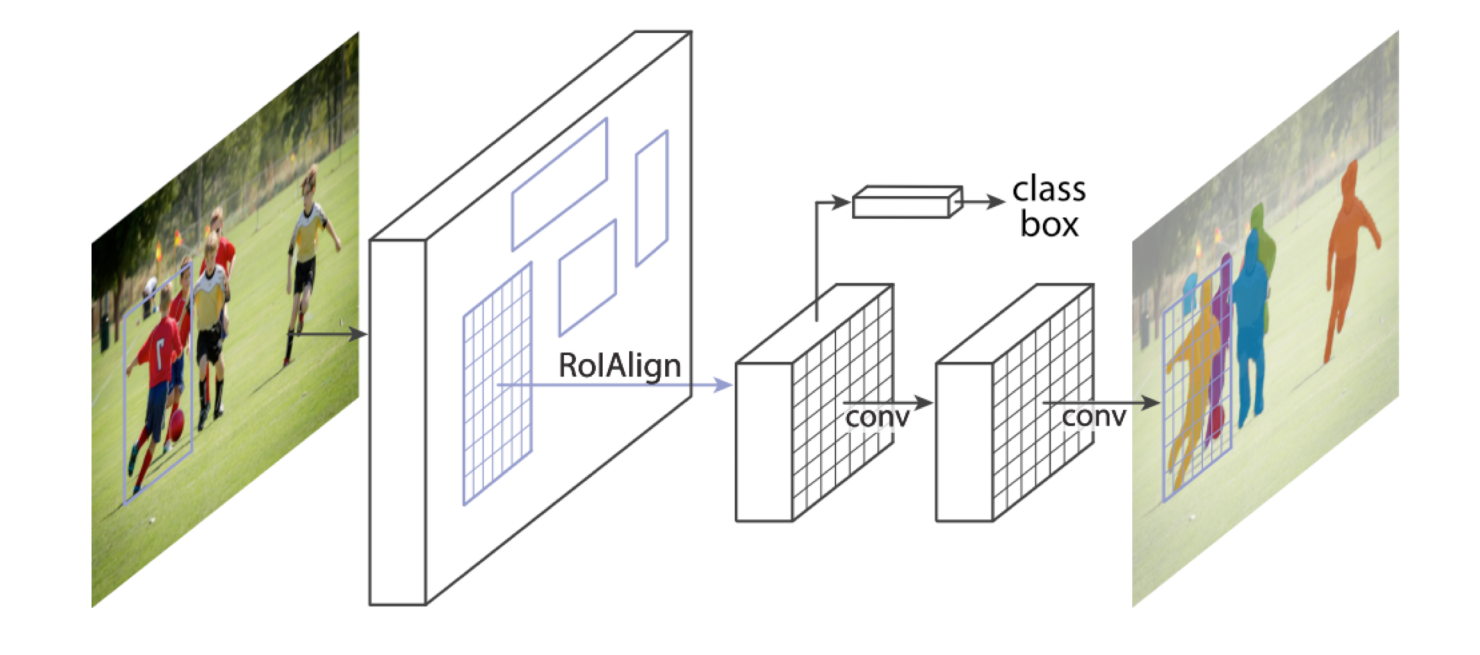
- Deformable Convolution/Pooling



Dai, Jifeng, et al. "Deformable convolutional networks."

Spatial Operators

- ROIAlign



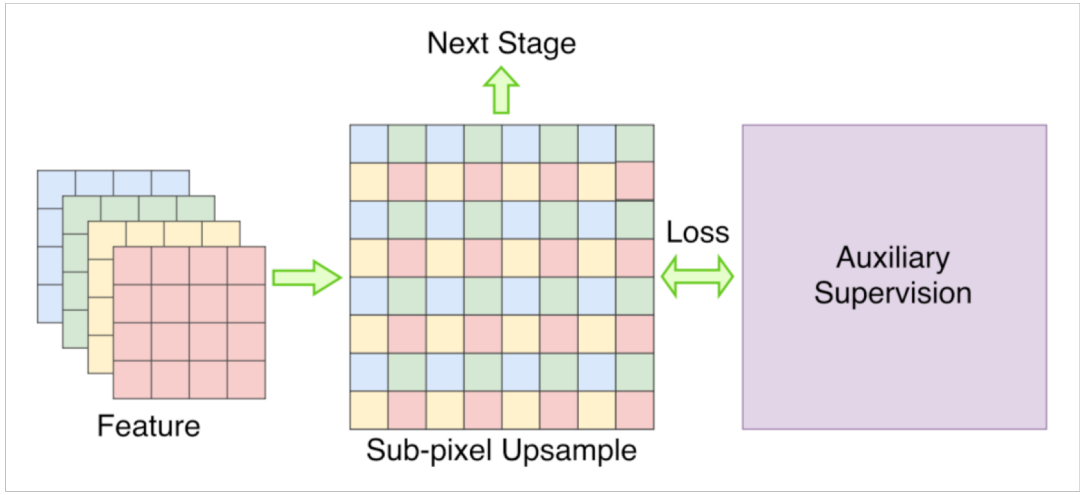
He, Kaiming, et al. "Mask r-cnn." *(ICCV)*

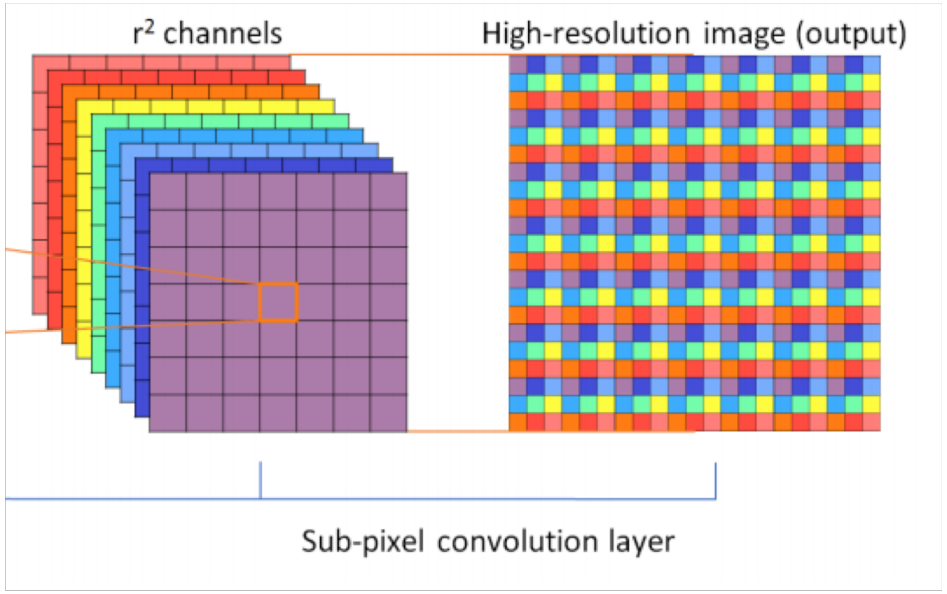
Spatial Operators

- Upsampling
  - *Deconvolution*
  - *Resize*
  - *“Reshape”*
  - *Unpooling*
  - *…*

Spatial Operators

- Upsampling
  - *Deconvolution*
  - *Resize*
  - *"Reshape"*
  - *Unpooling*
  - *…*





Shi, W., et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. (2016)

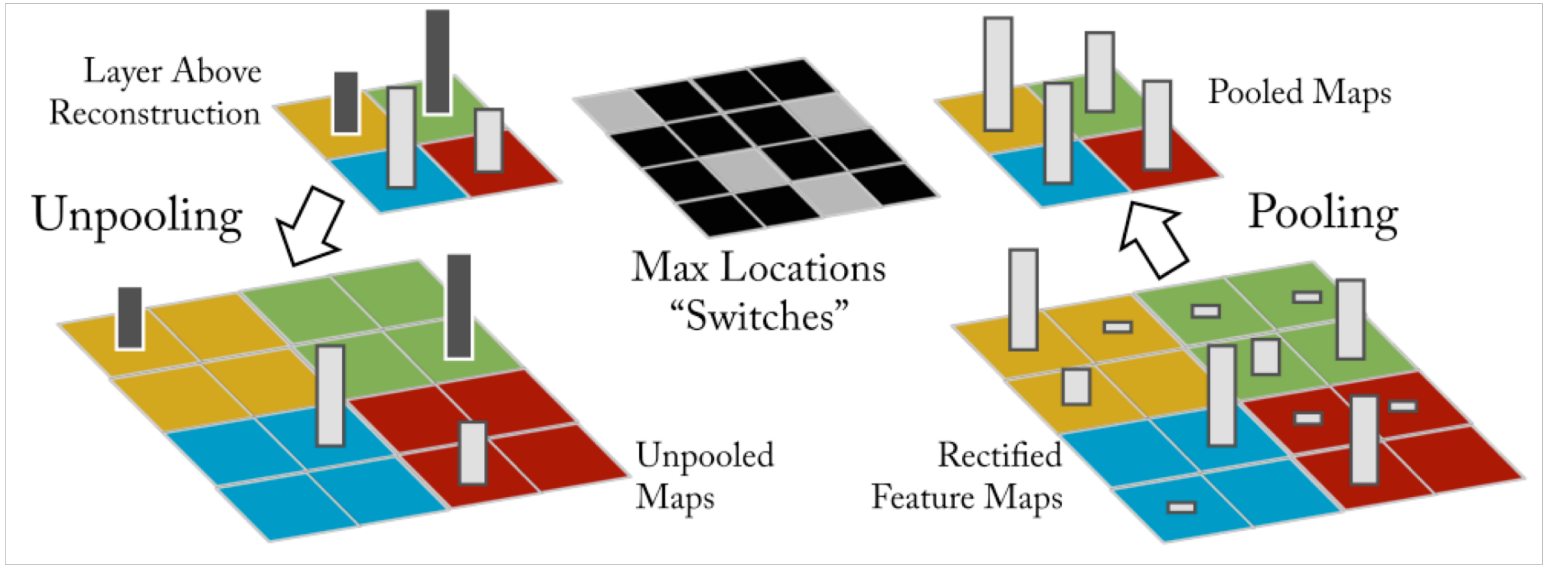
Aitken, A., et al.: Checkerboard 684 artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. (2017)

Zhenli Zhang, et al. ExFuse: Enhancing Feature Fusion for Semantic Segmentation.

Spatial Operators

- Upsampling
  - *Deconvolution*
  - *Resize*
  - *“Reshape”*
  - *Unpooling*

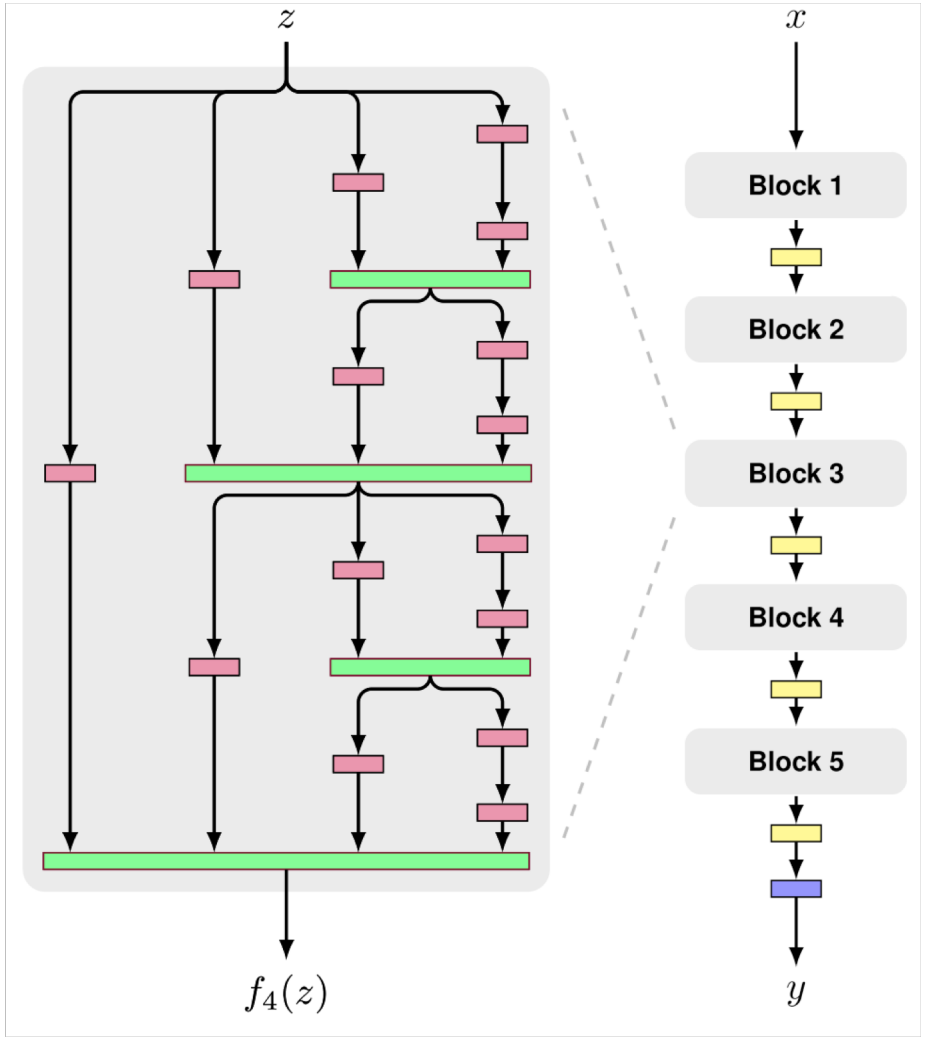
  …



Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." *European conference on computer vision*. Springer, Cham, 2014.

Densely Connections

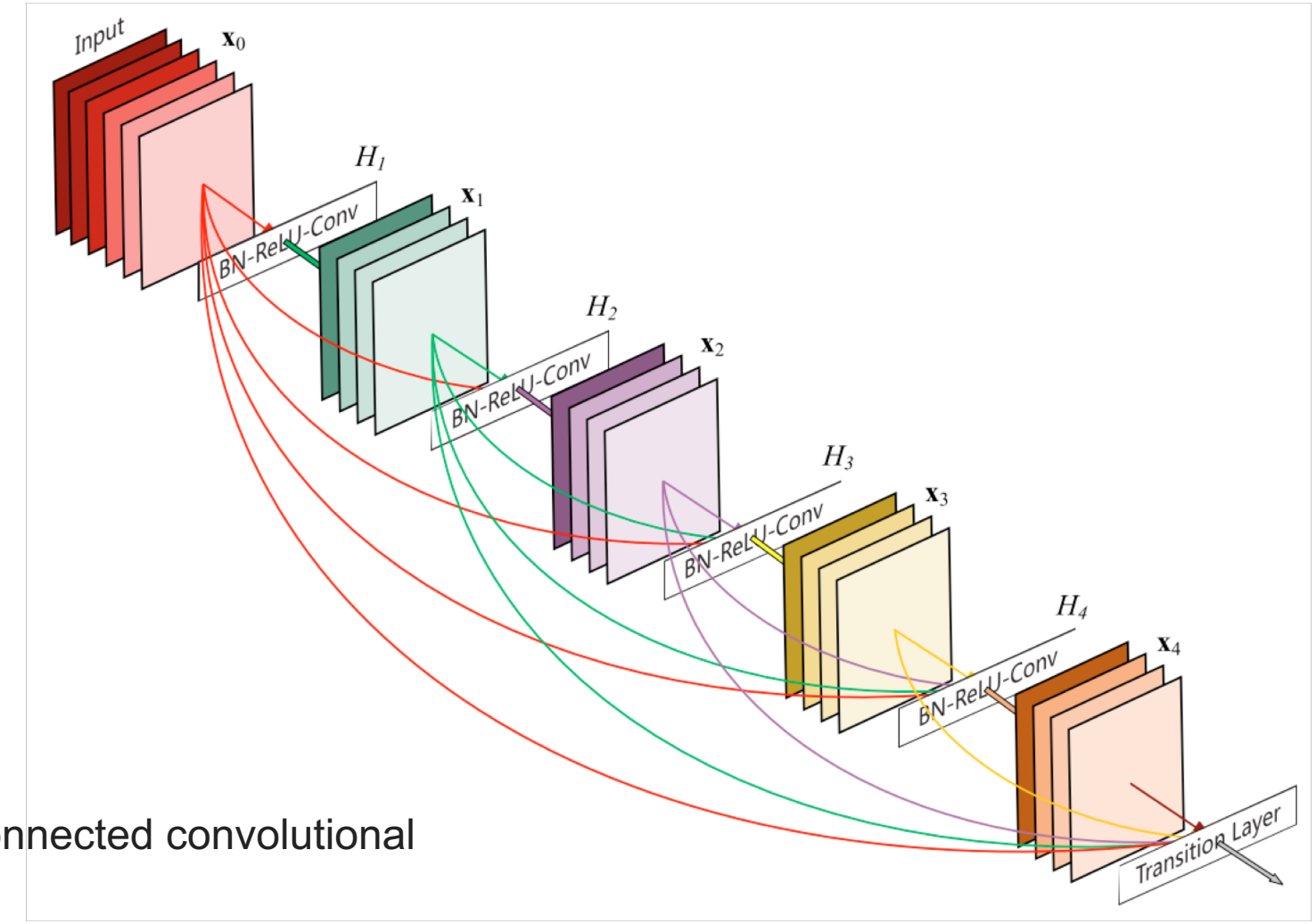
- FractalNet



Larsson, Gustav, Michael Maire, and Gregory Shakhnarovich. "Fractalnet: Ultra-deep neural networks without residuals."

Densely Connections

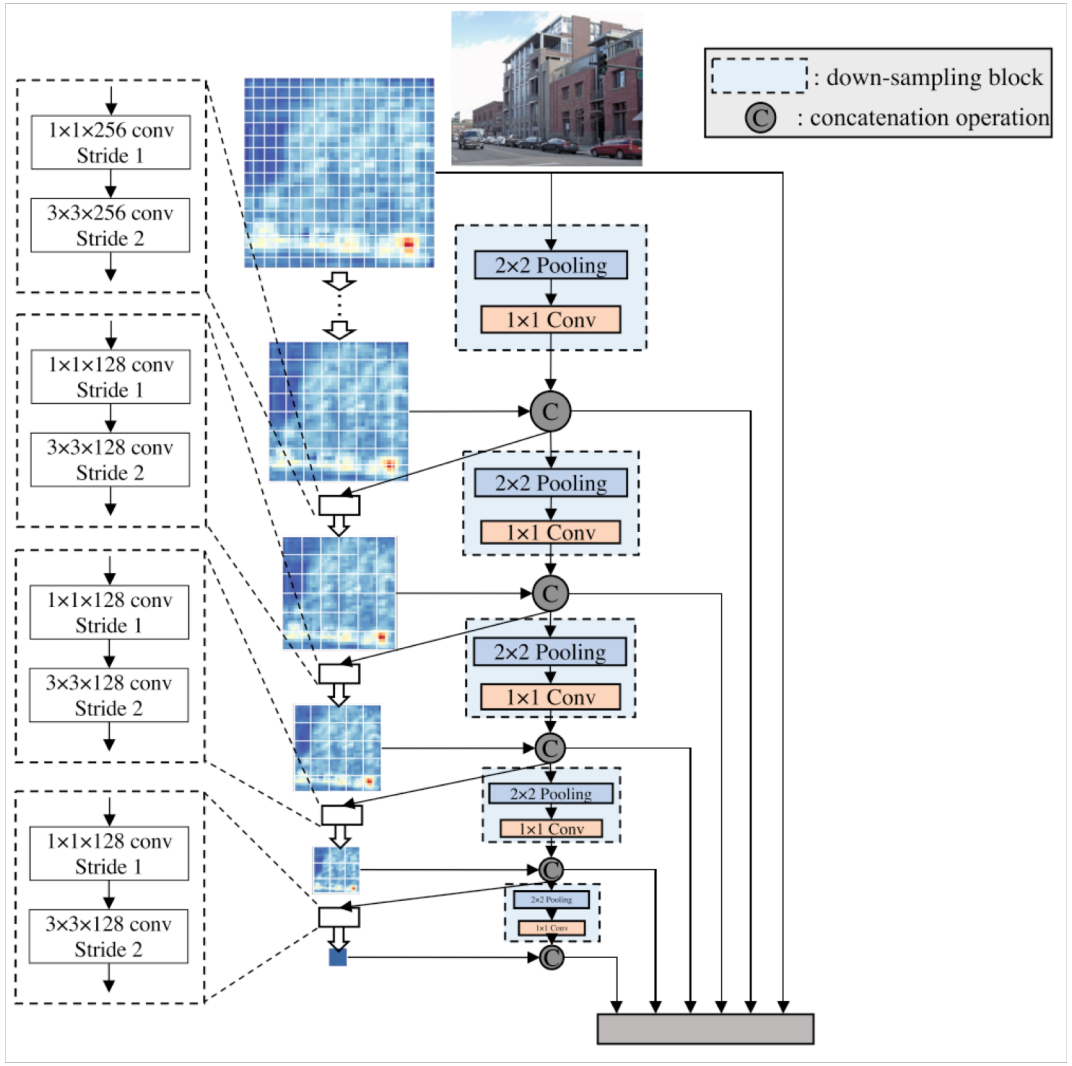
- DenseNet



Huang, Gao, et al. "Densely connected convolutional networks." *CVPR 2017.*

Densely Connections

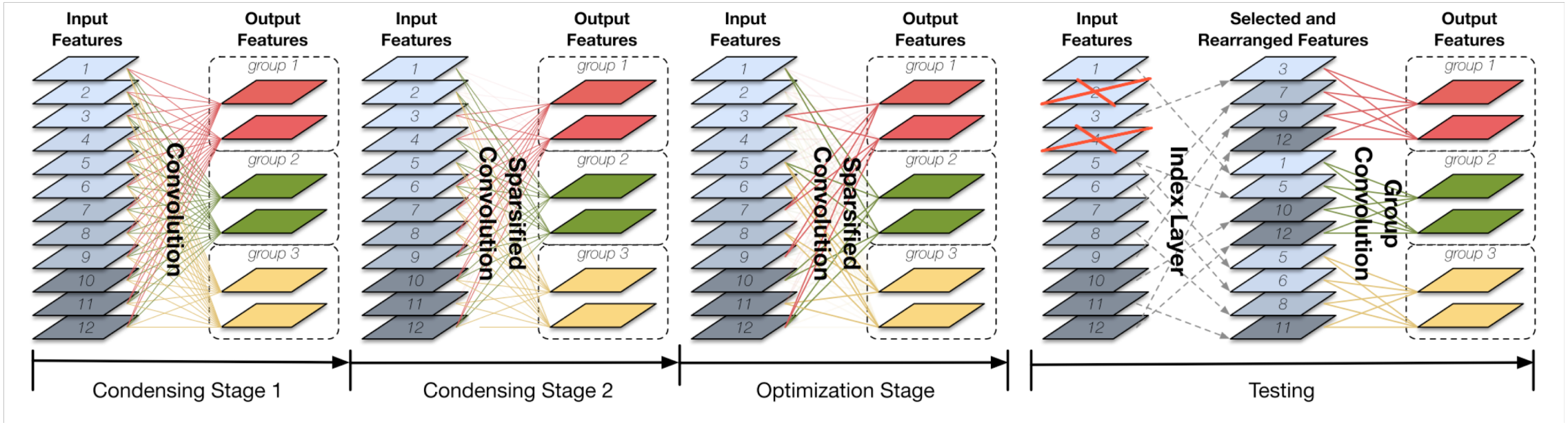
- DenseNet Applications
  - *DSOD*



Shen, Zhiqiang, et al. "Dsod: Learning deeply supervised object detectors from scratch." ICCV 2017

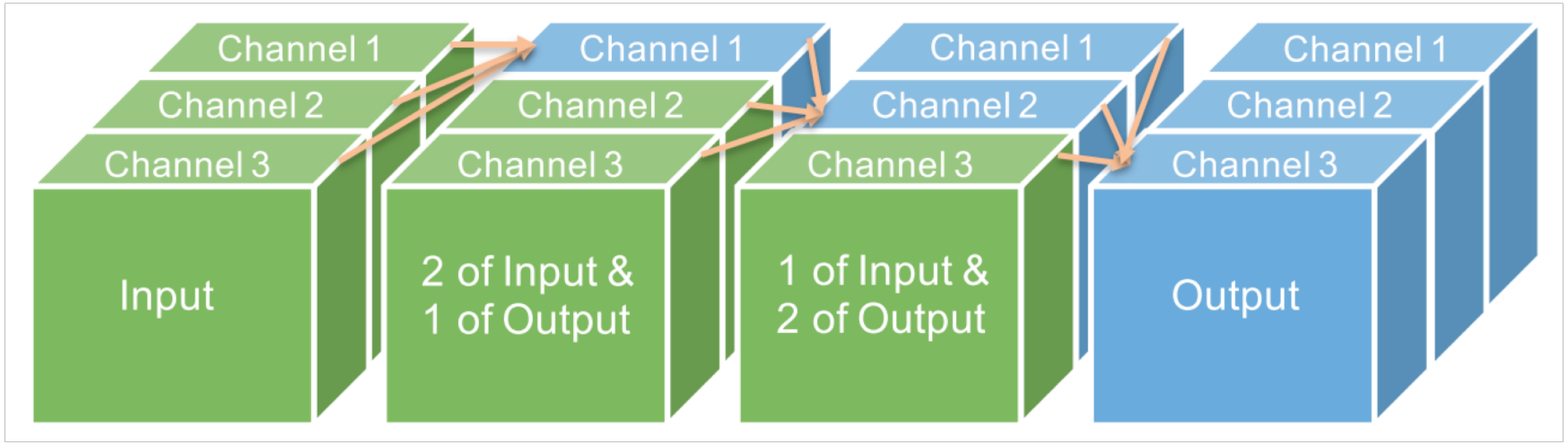
Densely Connections

- DenseNet Optimizations
  - *CondenseNet*



Huang, Gao, et al. "CondenseNet: An Efficient DenseNet using Learned Group Convolutions."

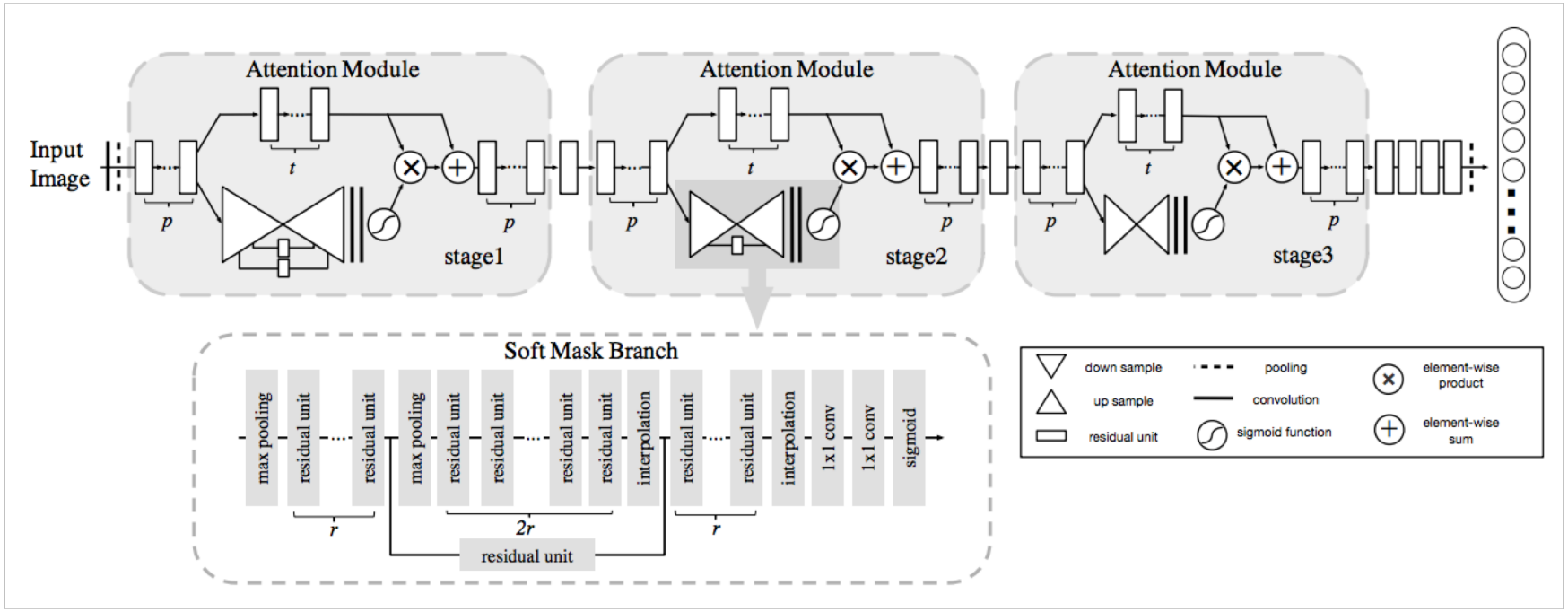
Feature Reuse



Qiao, Siyuan, et al. "Gradually Updated Neural Networks for Large-Scale Image Recognition."

Attention Blocks

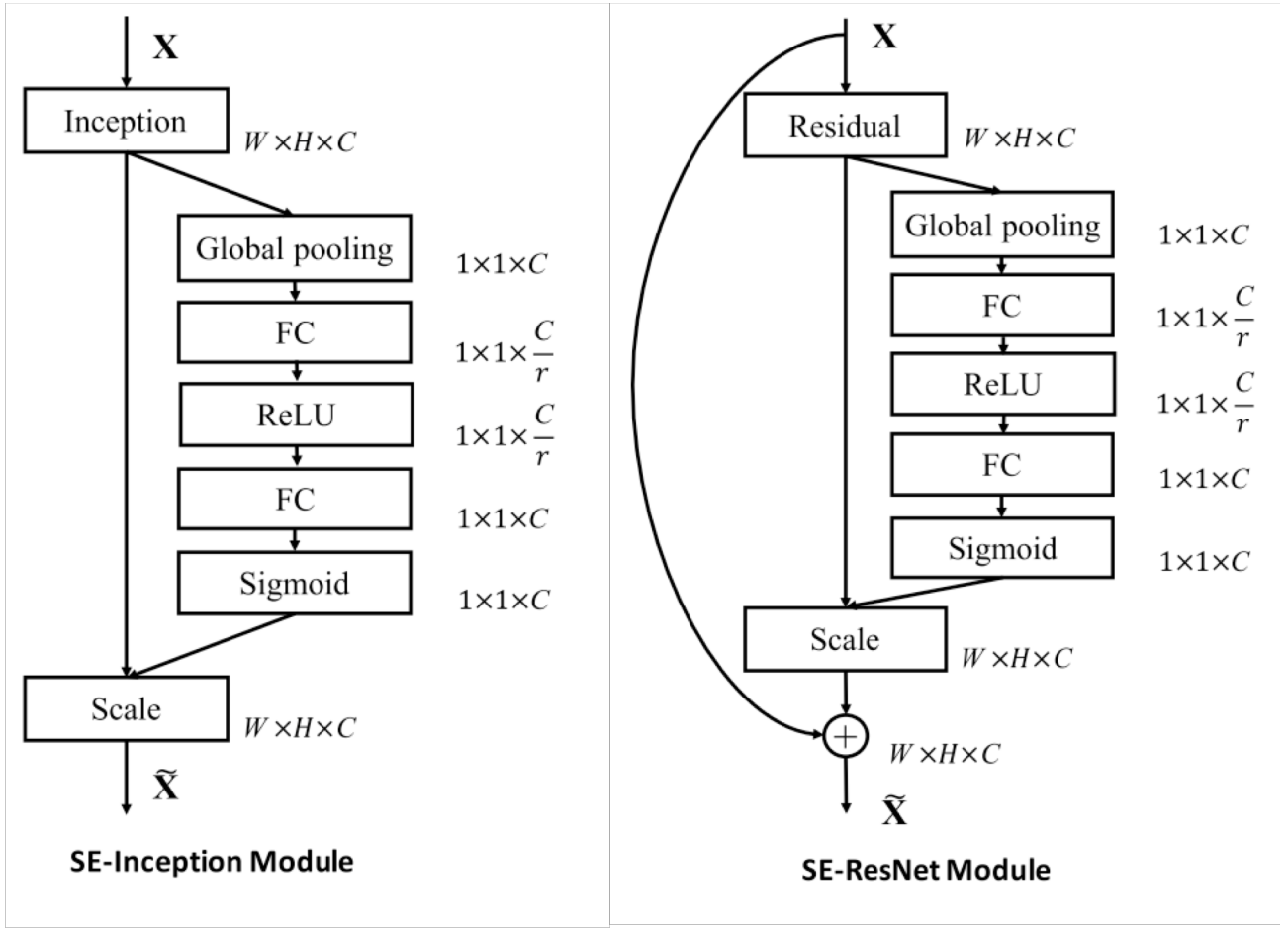
- Spatial attention
  - *Residual Attention Net*



Wang, Fei, et al. "Residual attention network for image classification."

Attention Blocks

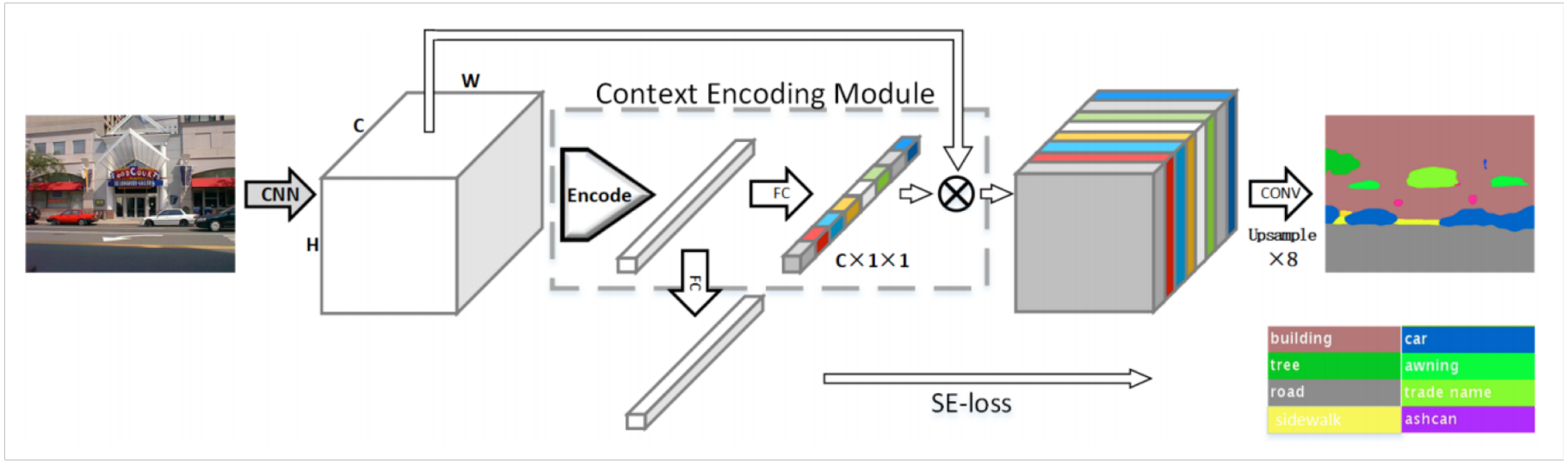
- Channel Attention
  - *SENet*
  - *With/without BN?*
- Applications
  - *Recommended*



Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks."

Attention Blocks

- Applications
  - *EncNet*



Hang Zhang, et al. Context Encoding for Semantic Segmentation

Normalization

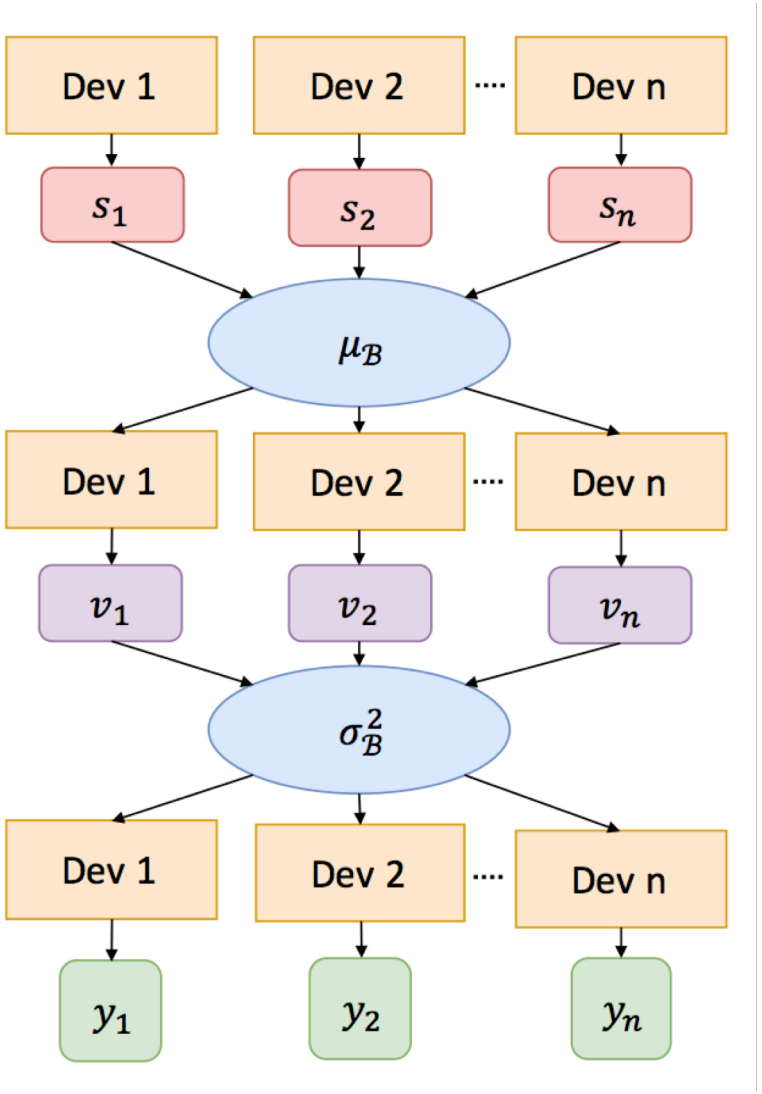
- Batch Normalization/Batch Renormalization
- Multi-GPU Batch Normalization
- Layer Normalization
- Weight Normalization
- Gradient Normalization
- Group Normalization

...

Normalization

- Batch Normalization/Batch Renormalization
- Multi-GPU Batch Normalization
- Layer Normalization
- Weight Normalization
- Gradient Normalization
- Group Normalization

...

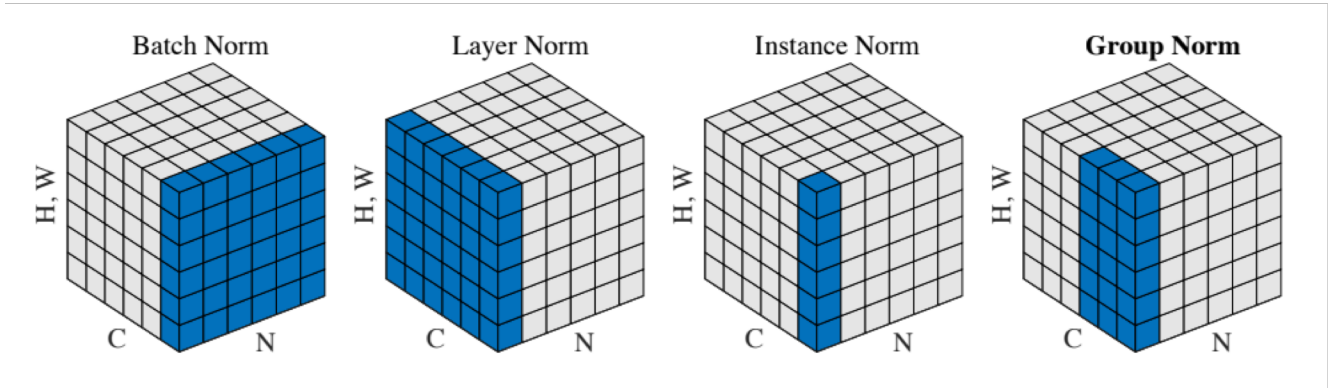


Peng, Chao, et al. "MegDet: A Large Mini-Batch Object Detector."

Normalization

- Batch Normalization/Batch Renormalization
- Multi-GPU Batch Normalization
- Layer Normalization
- Weight Normalization
- Gradient Normalization
- Group Normalization

...



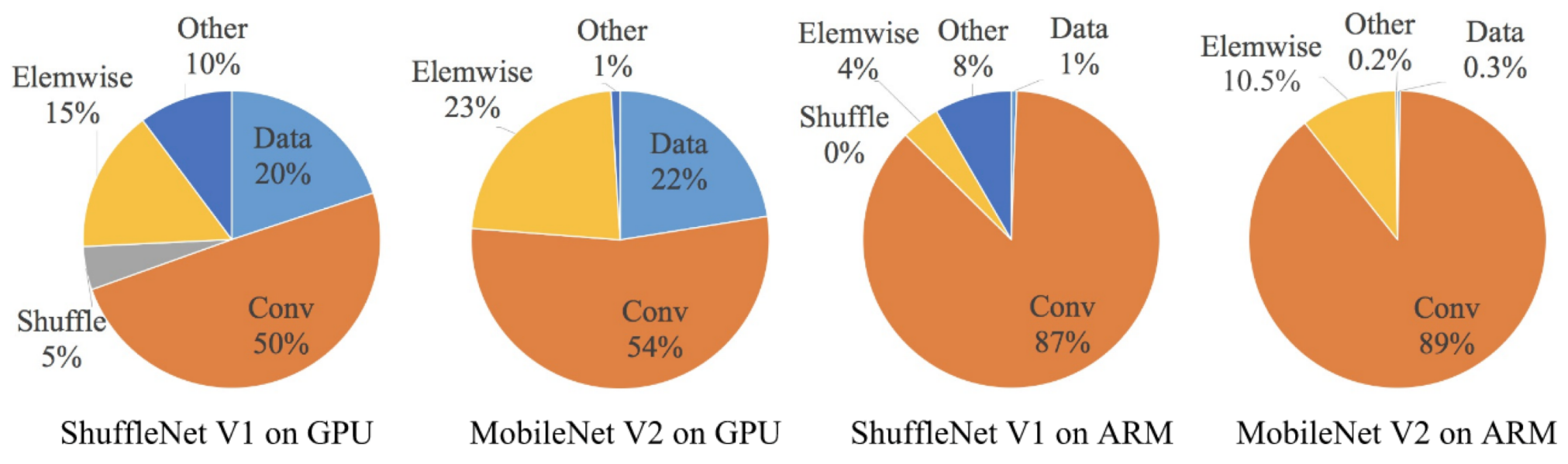
Yuxin Wu, et al. Group normalization

Normalization

- Best practice
  - *Large batch: BN*
  - *Small batch: Multi-gpu BN > GN/BRN > BN/LN*
  - *Weight/meta network: LN/WN*
  - *GAN/Multi-task: Gradient normalization*

Analysis: Inference Efficiency

- Actual running time
  - *Convolutions and elementwise operators*



Analysis: Inference Efficiency

- Structure efficiency
  - *Convolutional kernel shape*

		GPU (Batches/sec.)			CPU (Images/sec.)		
Input size	Channel (c) when c1=c2	c=128	c=256	c=512	c=32	c=64	c=128
56x56	c1=c2	1480	723	232	76.2	21.7	5.3
	c1=2c2	1296	586	206	72.9	20.5	5.1
	c1=6c2	876	489	189	69.1	17.9	4.6
	c1=12c2	748	392	163	57.6	15.1	4.4
28x28	c1=c2	2314	1504	765	263.9	82.6	23.3
	c1=2c2	2239	1401	675	212.8	78.5	22.6
	c1=6c2	1931	1075	573	198.8	75.5	20.7
	c1=12c2	1646	975	521	194.9	72.5	19.5

Analysis: Inference Efficiency

- Structure efficiency
  - *Group convolutions*

		GPU (Batches/sec.)			CPU (Images/sec.)		
Input size	Channel (c) when g=1	c=128	c=256	c=512	c=64	c=128	c=256
56x56	g=1	2451	1289	437	40.0	10.2	2.3
	g=2	1725	873	341	35.0	9.5	2.2
	g=4	1026	644	338	32.9	8.7	2.1
	g=8	634	445	230	27.8	7.5	1.8
28x28	g=1	3546	2523	1404	146.8	44.3	10.6
	g=2	2650	1874	976	137.4	42.2	10.2
	g=4	1766	1265	717	137.2	39.4	10.1
	g=8	1097	807	499	122.9	33.9	9.6

Analysis: Inference Efficiency

- Structure efficiency
  - *Fragments*

		GPU (Batches/sec.)			CPU (Images/sec.)		
Input size	Channel (c) for 1-fragment	c=128	c=256	c=512	c=64	c=128	c=256
56x56	1-fragment	2446	1274	434	40.2	10.1	2.3
	2-fragment-series	1790	909	336	38.6	10.1	2.2
	4-fragment-series	752	745	349	38.4	10.1	2.3
	2-fragment-parallel	1537	803	320	33.4	9.1	2.2
	4-fragment-parallel	691	572	292	35.0	8.4	2.1
28x28	1-fragment	3572	2508	1377	169.8	44.3	9.4
	2-fragment-series	2377	1763	960	155.3	42.9	9.1
	4-fragment-series	1269	1134	776	156.3	41.9	9.4
	2-fragment-parallel	2097	1587	887	146.2	40.7	8.7
	4-fragment-parallel	1045	974	651	144.5	40.3	8.5

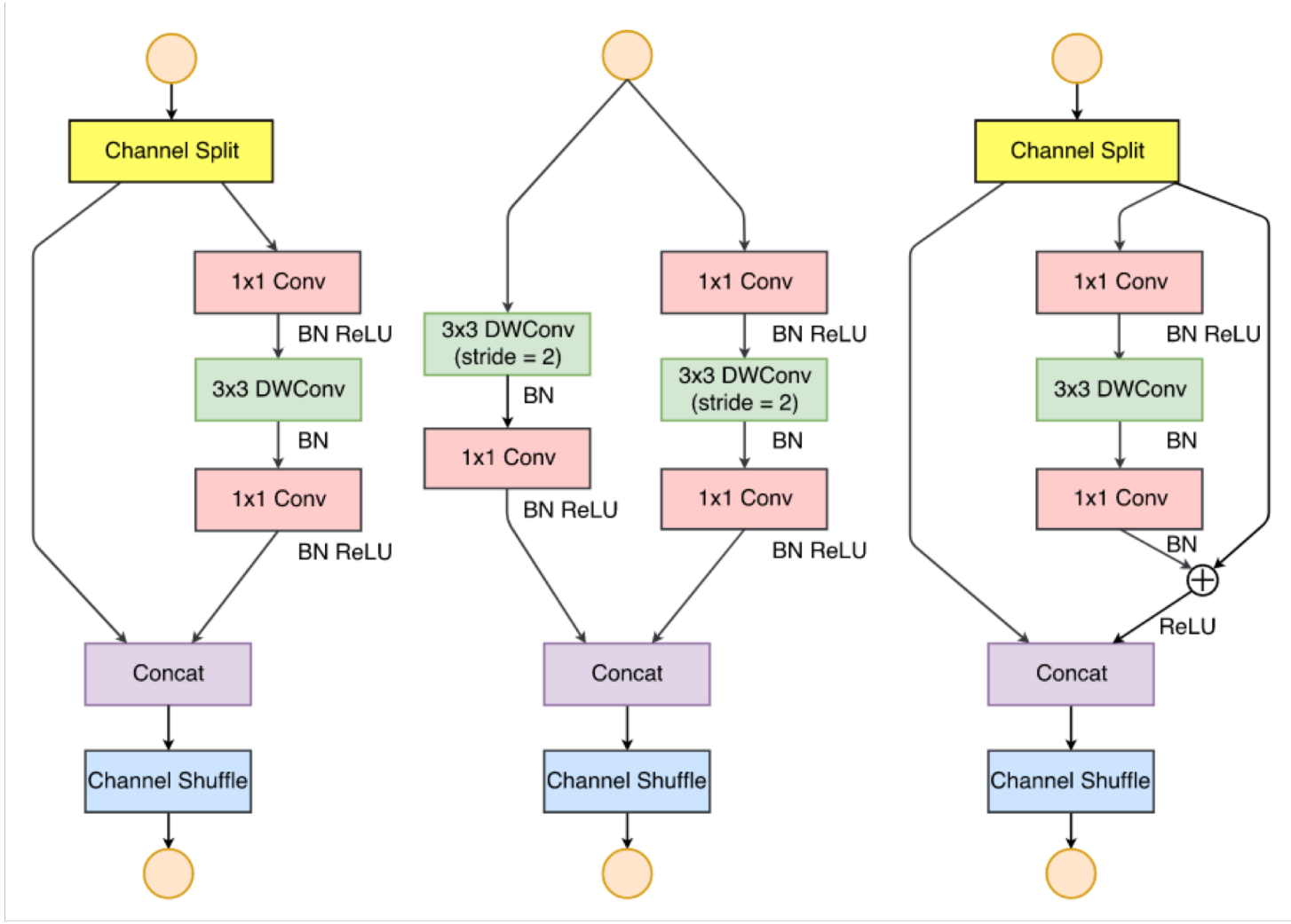
Analysis: Inference Efficiency

- Structure efficiency
  - *ReLUs and residual-adds*

		GPU (Batches/sec.)			CPU (Images/sec.)		
Input size	Channel (c)	c=32	c=64	c=128	c=32	c=64	c=128
56x56	bottleneck	2427	2066	1436	56.7	16.9	5.0
	bottleneck (without shortcut)	2647	2256	1735	61.9	18.8	5.2
	bottleneck (without ReLU)	2672	2121	1458	57.3	18.2	5.1
	bottleneck (without shortcut and ReLU)	2842	2376	1782	66.3	20.2	5.4
28x28	bottleneck	2927	3120	2845	261.1	74.7	21.6
	bottleneck (without shortcut)	3032	3499	3102	274.0	79.7	23.5
	bottleneck (without ReLU)	3137	3562	2905	274.7	76.1	22.4
	bottleneck (without shortcut and ReLU)	3342	3719	3118	284.9	81.6	24.3

ShuffleNet v2

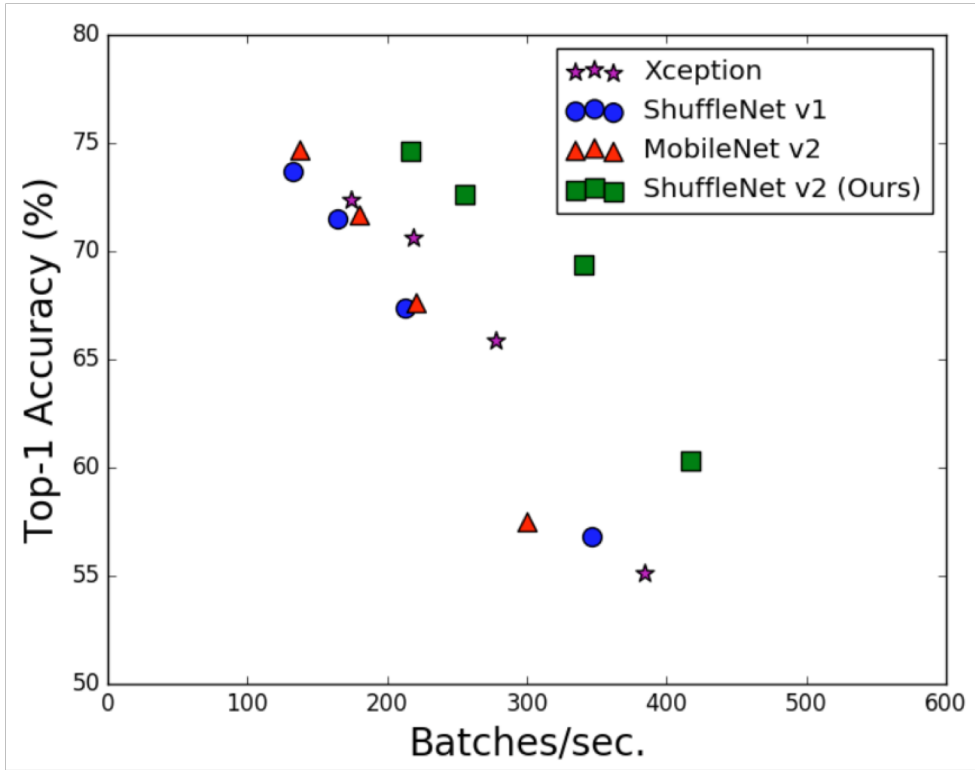
- Inference-efficient structrue

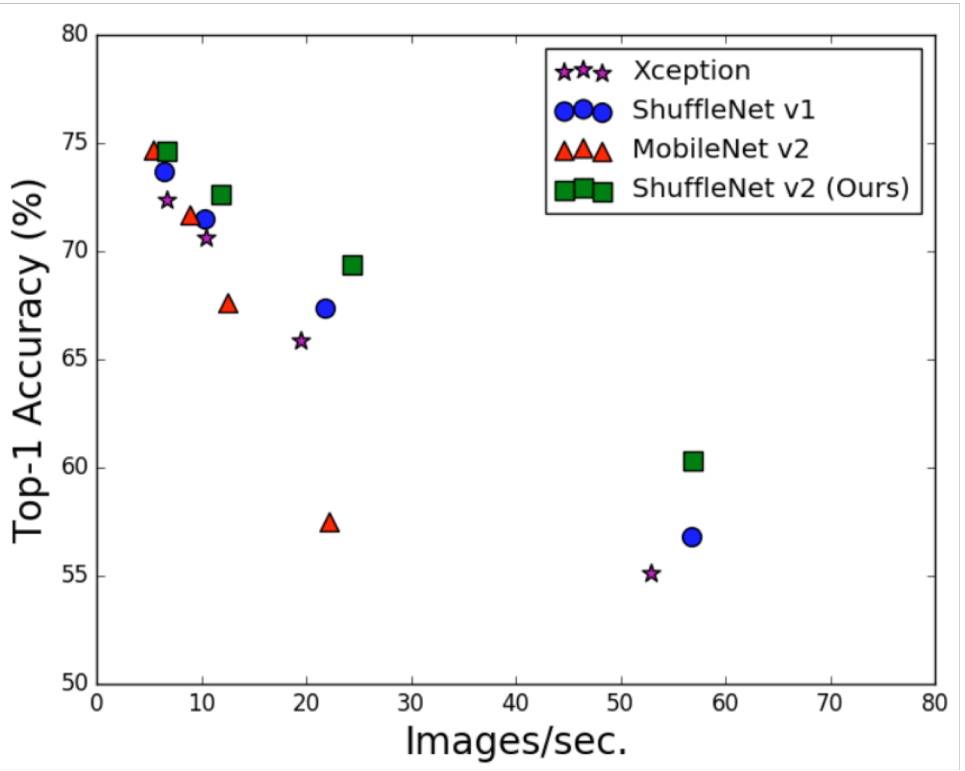


Ninging Ma, Xiangyu Zhang, Jian Sun. Light-weight CNN Architecture Design for Fast Inference

ShuffleNet v2

- Better accuracy/actual speed trade-offs





ShuffleNet v2

- Known issues
  - *Limited RF*
- Applications
  - *Classification-oriented tasks*
  - *GPU efficiency*
  - *Competition (not sure)*

Model	mmAP (%)		GPU Speed (Batches/sec.)	
FLOPs	300M	500M	300M	500M
Xception	31.3	32.9	25.3	20.8
ShuffleNet v1	29.9	32.9	19.0	15.0
MobileNet v2	30.0	30.6	23.5	18.0
ShuffleNet v2 (ours)	31.8	33.3	**27.3**	**21.8**
ShuffleNet v2* (ours)	**33.0**	**34.8**	21.3	16.5

Conclusion: Model Design

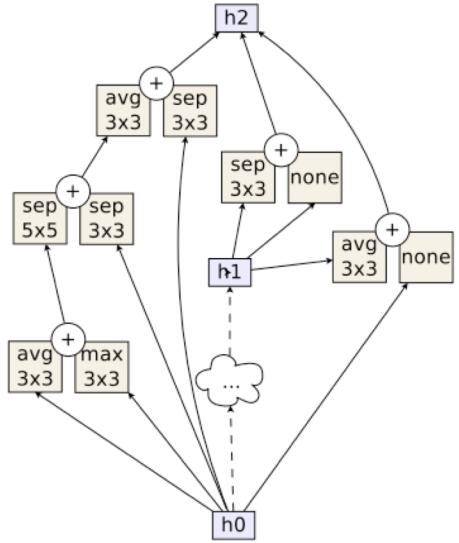
- Task specific
  - *Cls or Loc*
  - *Representation or Receptive Field*
- Shortcut matters!
- Feature reuse
- Attention if needed
- Inference efficiency

Outline

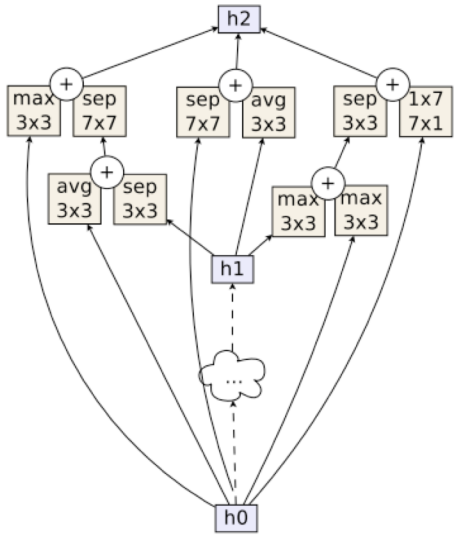
- Architecture related
- Other topics
  - *Model search*
  - *Bilinear feature*
  - *Large batch training*

Model Search

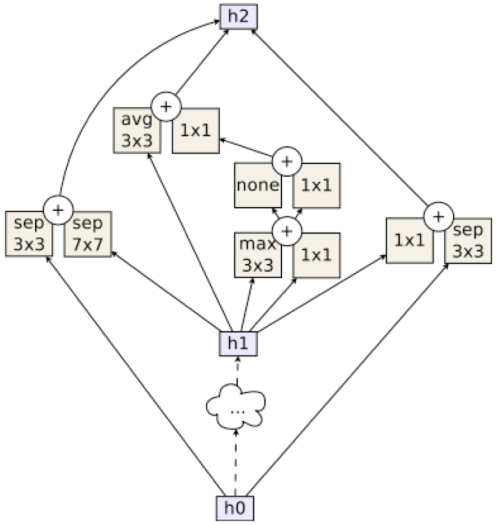
- Automatic model structure generation
- Agents
  - *Greedy*
  - *RL*
  - *Evolutionary/Genetic*
  - *Model Pruning*
- Usually very costly


h2
avg 3x3
sep 3x3
sep 3x3
none
sep 5x5
sep 3x3
avg 3x3
none
h1
avg 3x3
max 3x3
h0

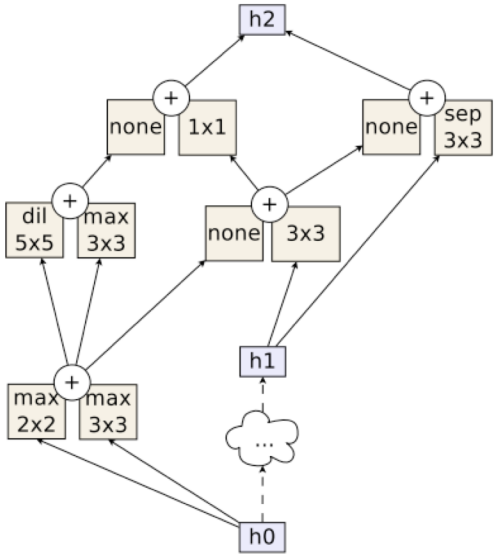
(a)


h2
max 3x3
sep 7x7
sep 7x7
avg 3x3
sep 3x3
1x7 7x1
avg 3x3
sep 3x3
max 3x3
max 3x3
h1
h0

(b)


h2
avg 3x3
1x1
none
1x1
sep 3x3
sep 7x7
max 3x3
1x1
1x1
sep 3x3
h1
h0

(c)

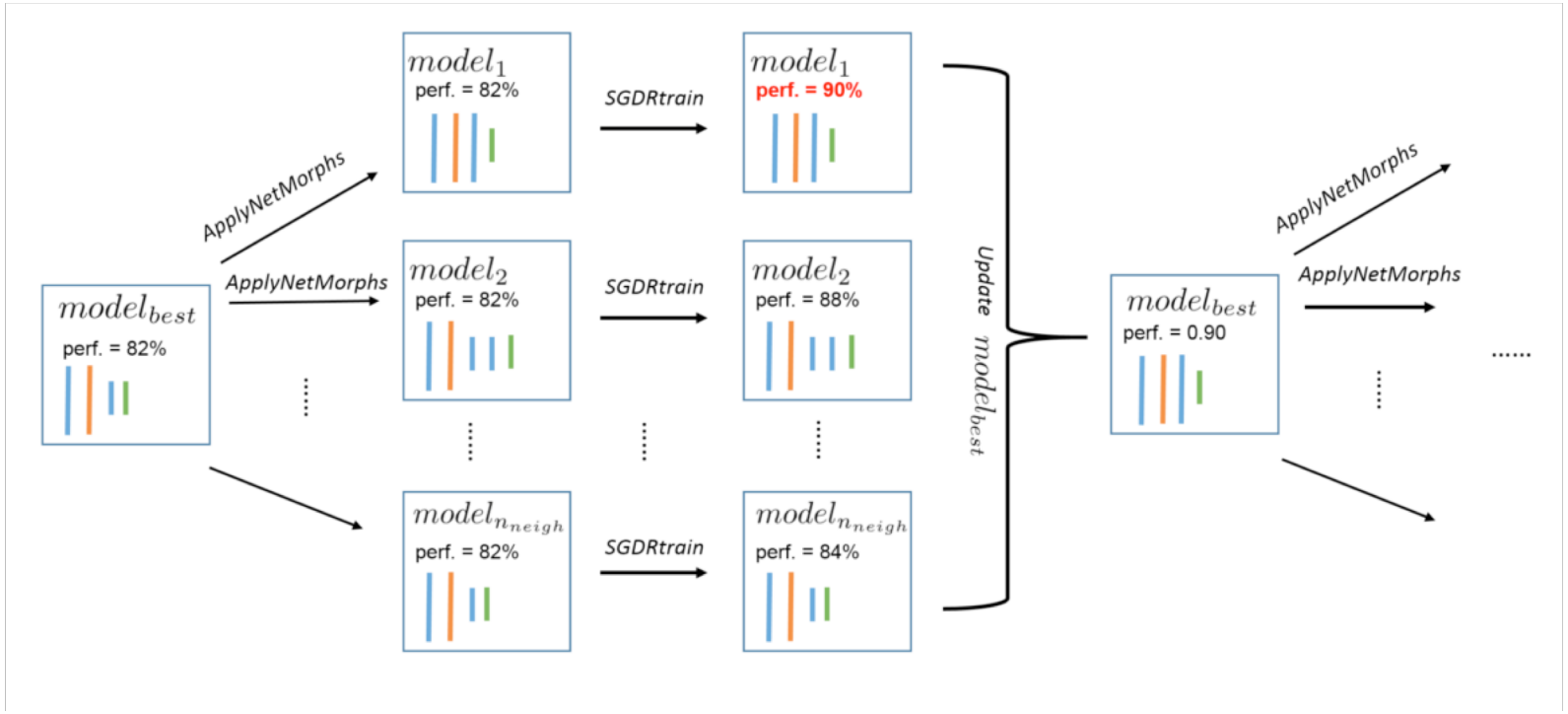

h2
none
1x1
none
sep 3x3
dil 5x5
max 3x3
none
3x3
h1
max 2x2
max 3x3
h0

(d)


add
sep 5x5

Model Search

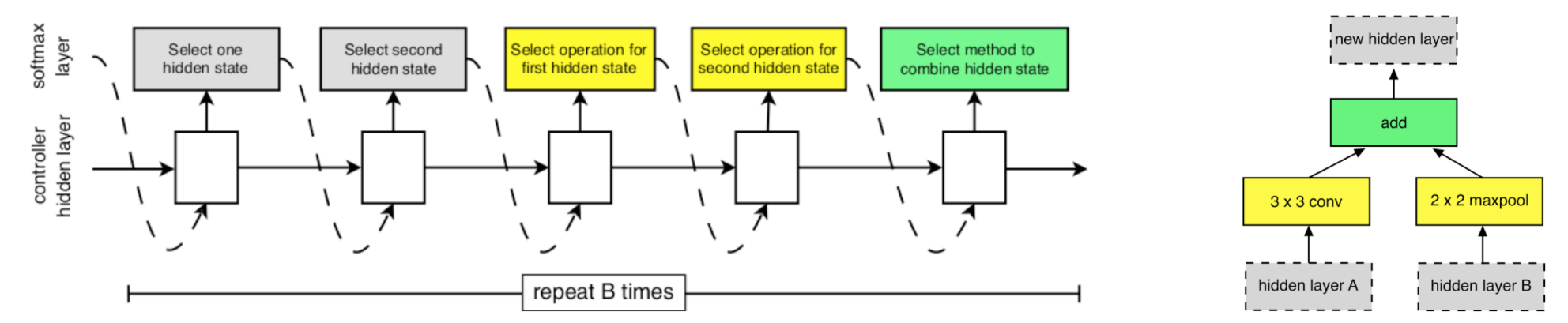
- Greedy
  - *Net2net based*
  - *Hill climbing*



Elsken, Thomas, Jan-Hendrik Metzen, and Frank Hutter. "Simple and efficient architecture search for convolutional neural networks."

Model Search

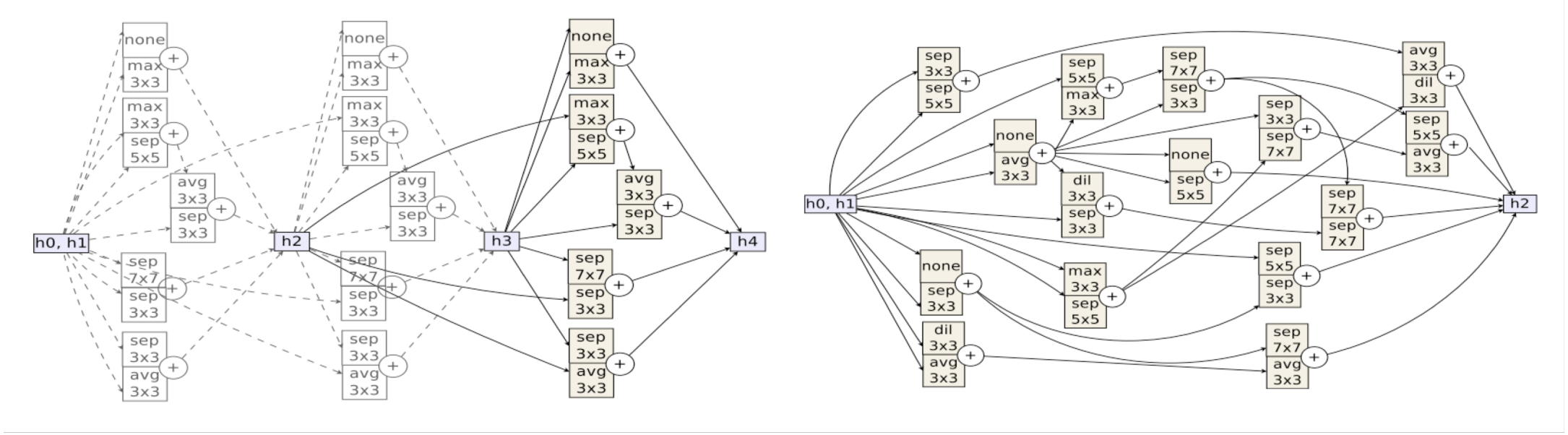
- RL
  - *Search on Cifar-10, test on ImageNet (transferable)*
  - *Generator: RNN*
  - *Reward: score*



Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition."

Model Search

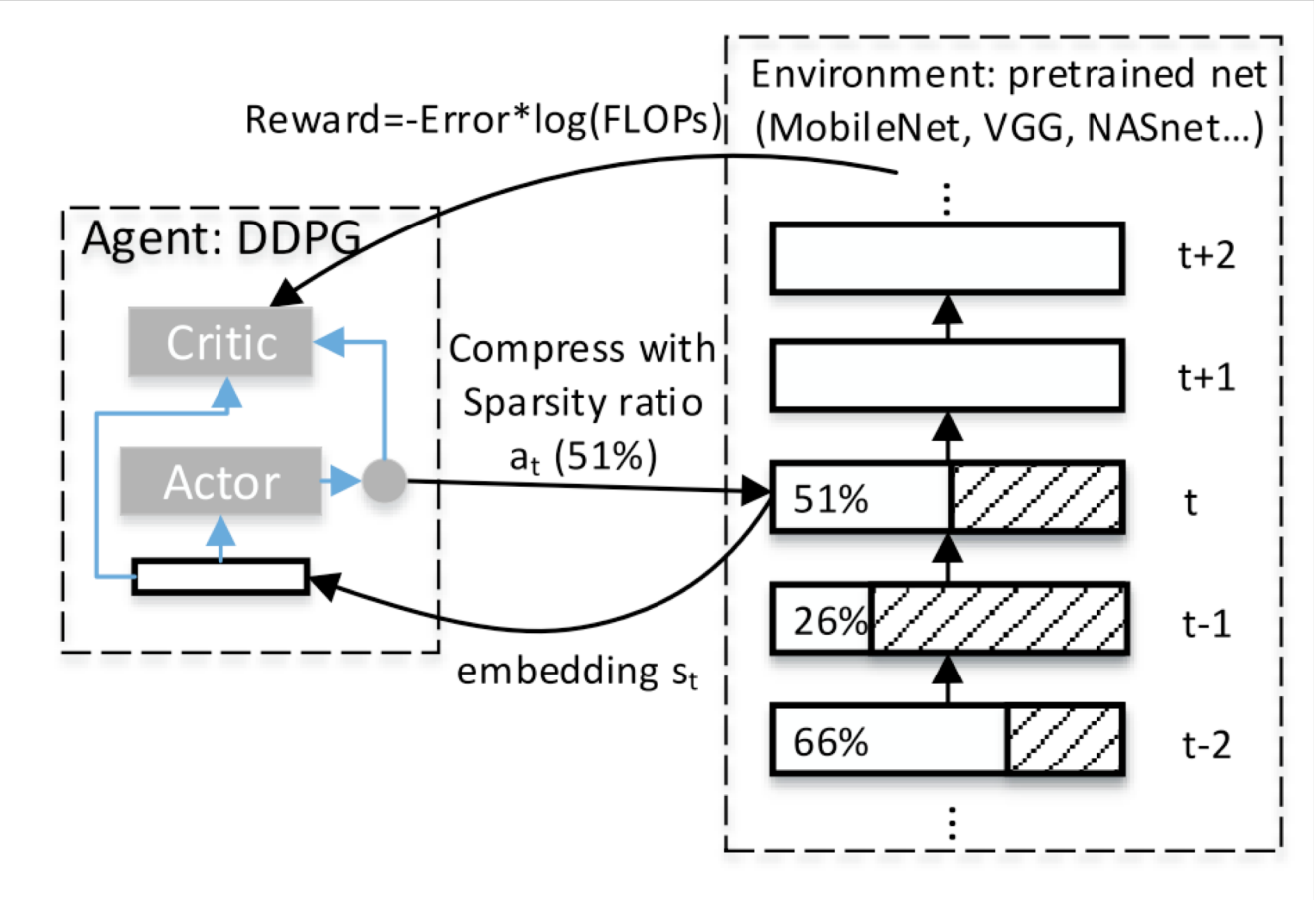
- Evolution
  - *AmoebaNet*



Real, Esteban, et al. "Regularized Evolution for Image Classifier Architecture Search."

Model Search

- Pruning

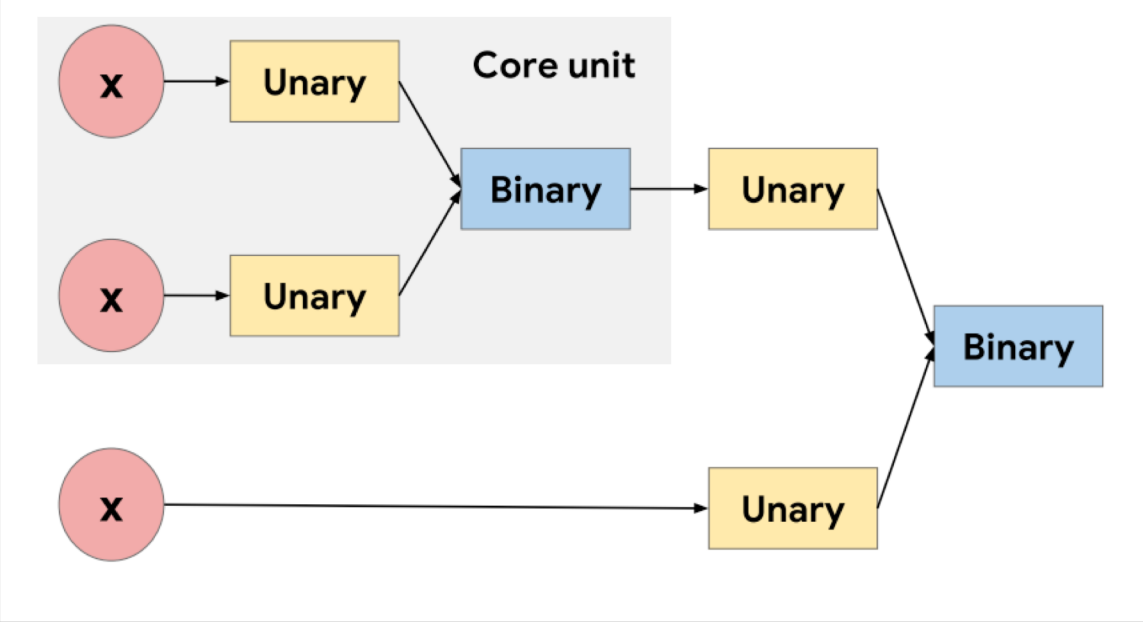


He, Yihui, and Song Han. "ADC: Automated Deep Compression and Acceleration with Reinforcement Learning."

Model Search

- Search for activation functions

Function	RN	WRN	DN
ReLU $[\max(x, 0)]$	93.8	95.3	94.8
$x \cdot \sigma(\beta x)$	94.5	95.5	94.9
$\max(x, \sigma(x))$	94.3	95.3	94.8
$\cos(x) - x$	94.1	94.8	94.6
$\min(x, \sin(x))$	94.0	95.1	94.4
$(\tan^{-1}(x))^2 - x$	93.9	94.7	94.9
$\max(x, \tanh(x))$	93.9	94.2	94.5
$\text{sinc}(x) + x$	91.5	92.1	92.0
$x \cdot (\sinh^{-1}(x))^2$	85.1	92.1	91.1



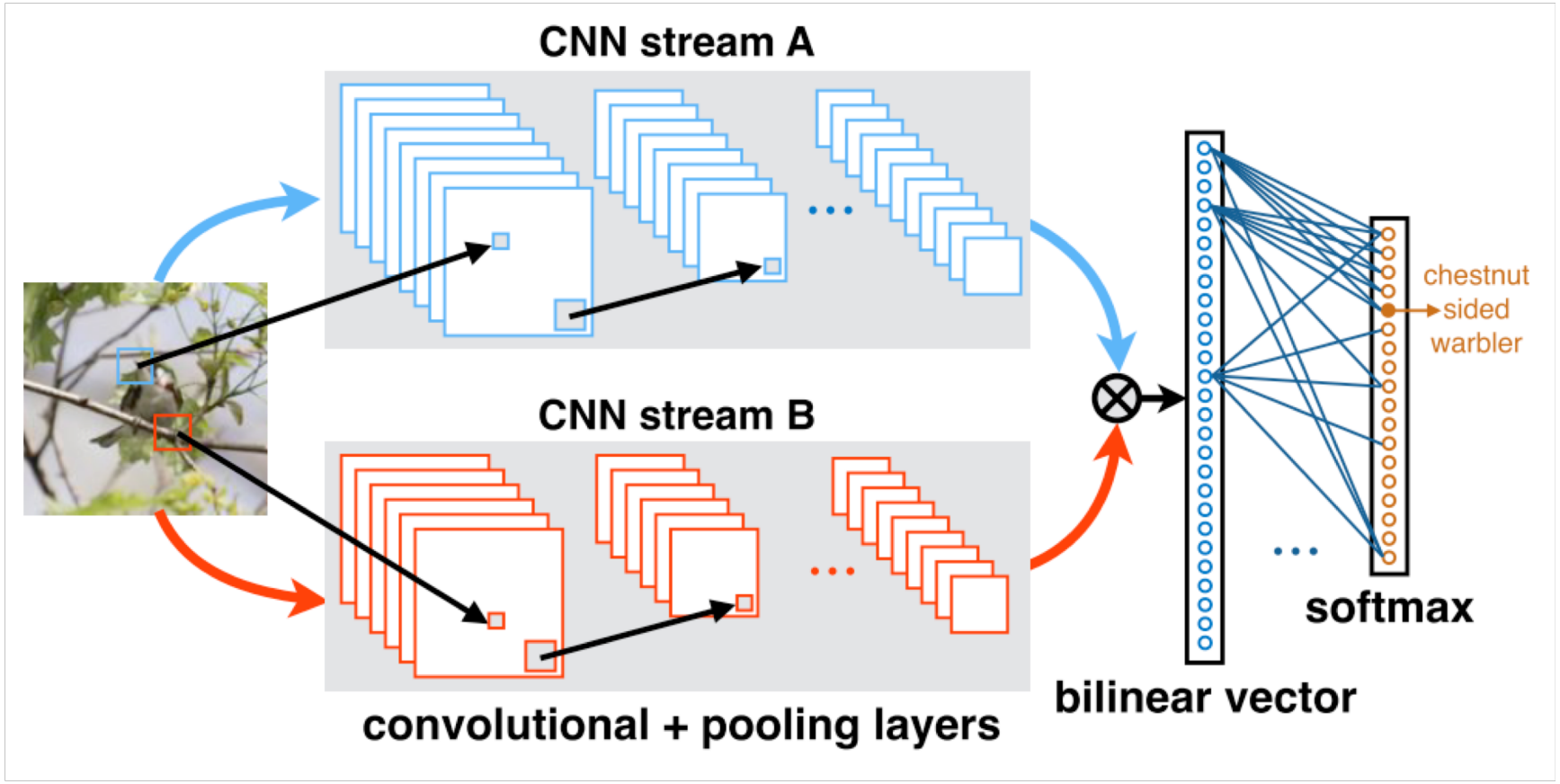
Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." (2018).

Model Search

- Trend
  - *Less cost*
  - *Inference efficiency*

Model	FLOPs	Top-1 err. (%)	GPU Speed (Batches/sec.)	ARM Speed (Images/sec.)
ShuffleNet v2 2x (ours)	591M	25.4	**217**	**6.7**
ShuffleNet v2 2x (ours, with *SE* [18])	597M	**24.6**	161	5.6
MobileNet v2 (1.4) [8]	585M	25.3	137	5.4
NASNet-A [9] (4 @ 1056, our impl.)	564M	26.0	130	4.6
PNASNet-5 [11] (N = 3, F = 54, our impl.)	588M	25.8	115	4.1

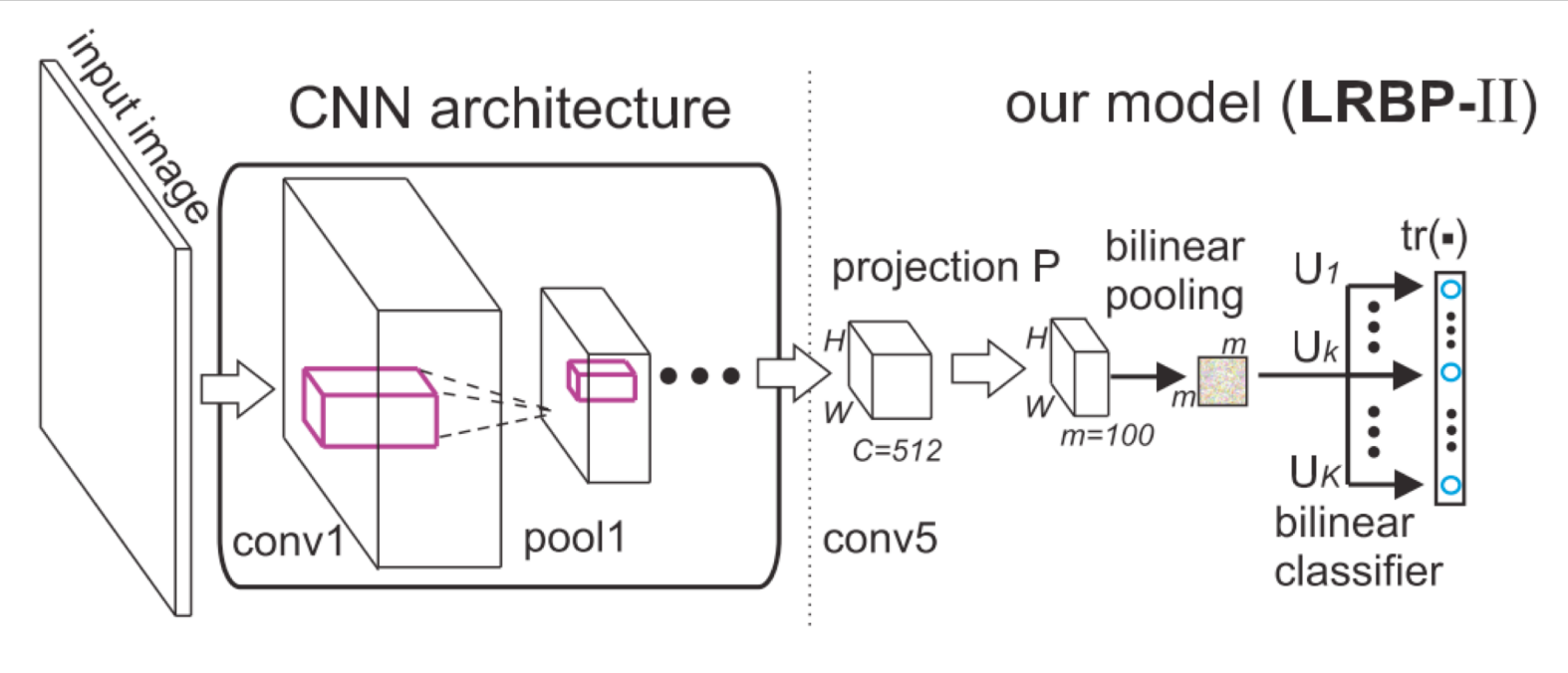
Bilinear Feature



Lin, Tsung-Yu, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear cnn models for fine-grained visual recognition."

Bilinear Feature

- Low-rank tricks

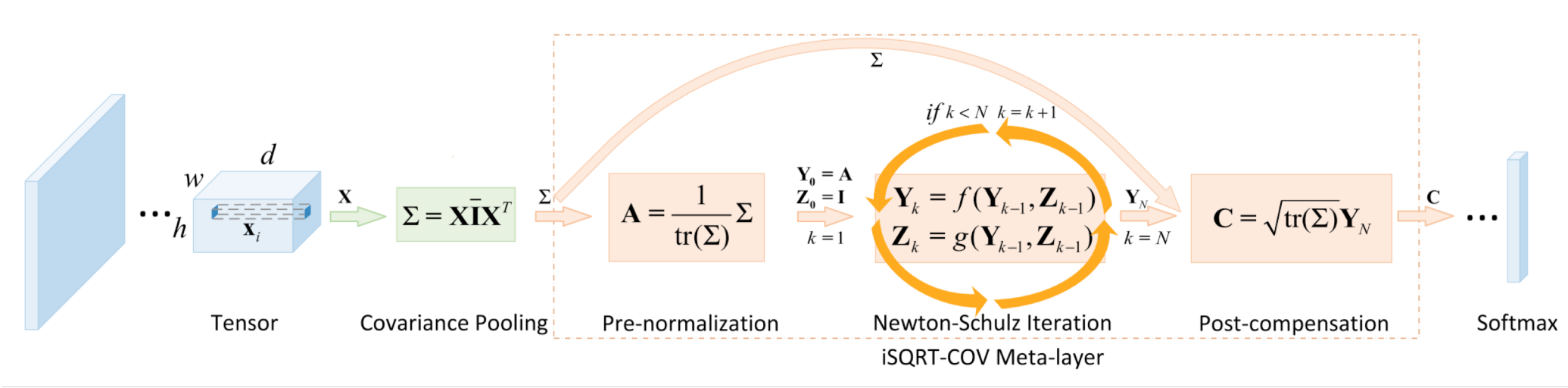


Kong, Shu, and Charless Fowlkes. "Low-rank bilinear pooling for fine-grained classification." *(CVPR)*

Bilinear Feature

- Whitening (SVD, Matrix SQRT, etc.)
  - *MPN-COV*
  - *G2DeNet*
  - *Improved B-CNN*
  - *iSQRT*

Li, Peihua, et al. "Is second-order information helpful for large-scale visual recognition?."
Li, Peihua, et al. "Towards Faster Training of Global Covariance Pooling Networks by Iterative Matrix Square Root Normalization."

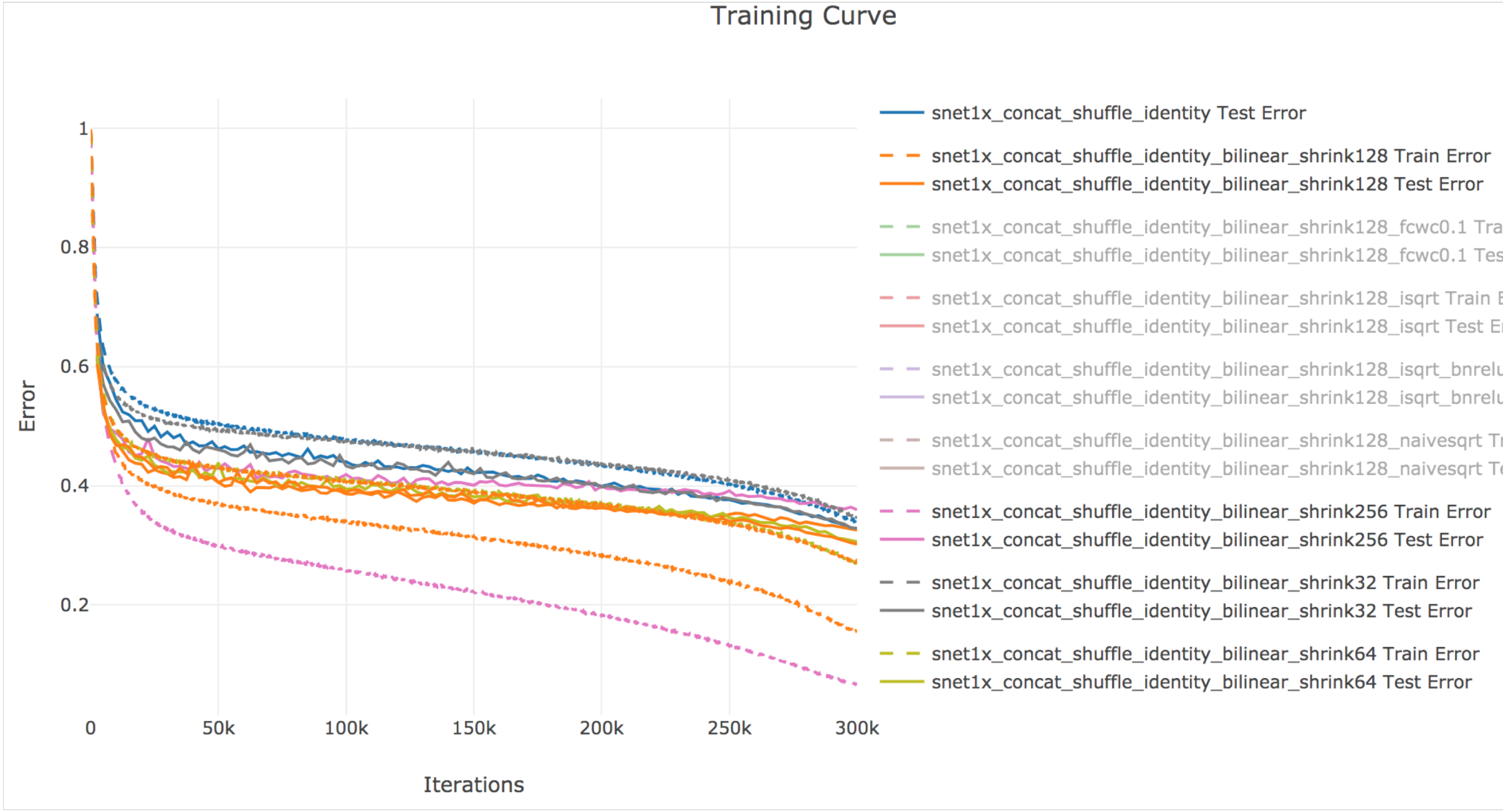


Bilinear Feature

- Best practice
  - *Linear 1x1conv + bilinear pooling + classifier*
  - *N^2 > number of classes*
  - *Add whitening if needed*

- Pros and cons
  - *Stronger classifier under limited number of feature dimensions*
  - *Unlikely to boost features*
  - *Overfit*

ShuffleNet v2 15M:	48.9 -> 44.3
ShuffleNet v2 140M:	31.2 -> 29.6


Training Curve
Error
Iterations
1
0.8
0.6
0.4
0.2
0
50k
100k
150k
200k
250k
300k
snet1x_concat_shuffle_identity Test Error
snet1x_concat_shuffle_identity_bilinear_shrink128 Train Error
snet1x_concat_shuffle_identity_bilinear_shrink128 Test Error
snet1x_concat_shuffle_identity_bilinear_shrink128_fcwc0.1 Tra
snet1x_concat_shuffle_identity_bilinear_shrink128_fcwc0.1 Tes
snet1x_concat_shuffle_identity_bilinear_shrink128_isqrt Train E
snet1x_concat_shuffle_identity_bilinear_shrink128_isqrt Test E
snet1x_concat_shuffle_identity_bilinear_shrink128_isqrt_bnrelu
snet1x_concat_shuffle_identity_bilinear_shrink128_isqrt_bnrelu
snet1x_concat_shuffle_identity_bilinear_shrink128_naivesqrt Tr
snet1x_concat_shuffle_identity_bilinear_shrink128_naivesqrt Te
snet1x_concat_shuffle_identity_bilinear_shrink256 Train Error
snet1x_concat_shuffle_identity_bilinear_shrink256 Test Error
snet1x_concat_shuffle_identity_bilinear_shrink32 Train Error
snet1x_concat_shuffle_identity_bilinear_shrink32 Test Error
snet1x_concat_shuffle_identity_bilinear_shrink64 Train Error
snet1x_concat_shuffle_identity_bilinear_shrink64 Test Error

Large Batch Training

- Distributed training
  - *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*
  - *Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes*
  - *ImageNet Training in Minutes*

Large Batch Training

- Equivalence rules
  - *Linear scaling rule (lr and wc)*

> ***Linear Scaling Rule:*** *When the minibatch size is multiplied by k, multiply the learning rate by k.*

  - *Batch normalization vs. batch size*
  - *Warm up*
  - *Momentum correction*

Goyal, Priya, et al. "Accurate, large minibatch SGD: training imagenet in 1 hour."

Large Batch Training

- Learning rate vs. batch size

$$
\begin{aligned}
g &= \frac{\epsilon}{1-m}\left(\frac{N}{B}-1\right) \\
&\approx \frac{\epsilon N}{B(1-m)}
\end{aligned}
$$

Smith, Samuel L., Pieter-Jan Kindermans, and Quoc V. Le. "Don't Decay the Learning Rate, Increase the Batch Size."

Large Batch Training

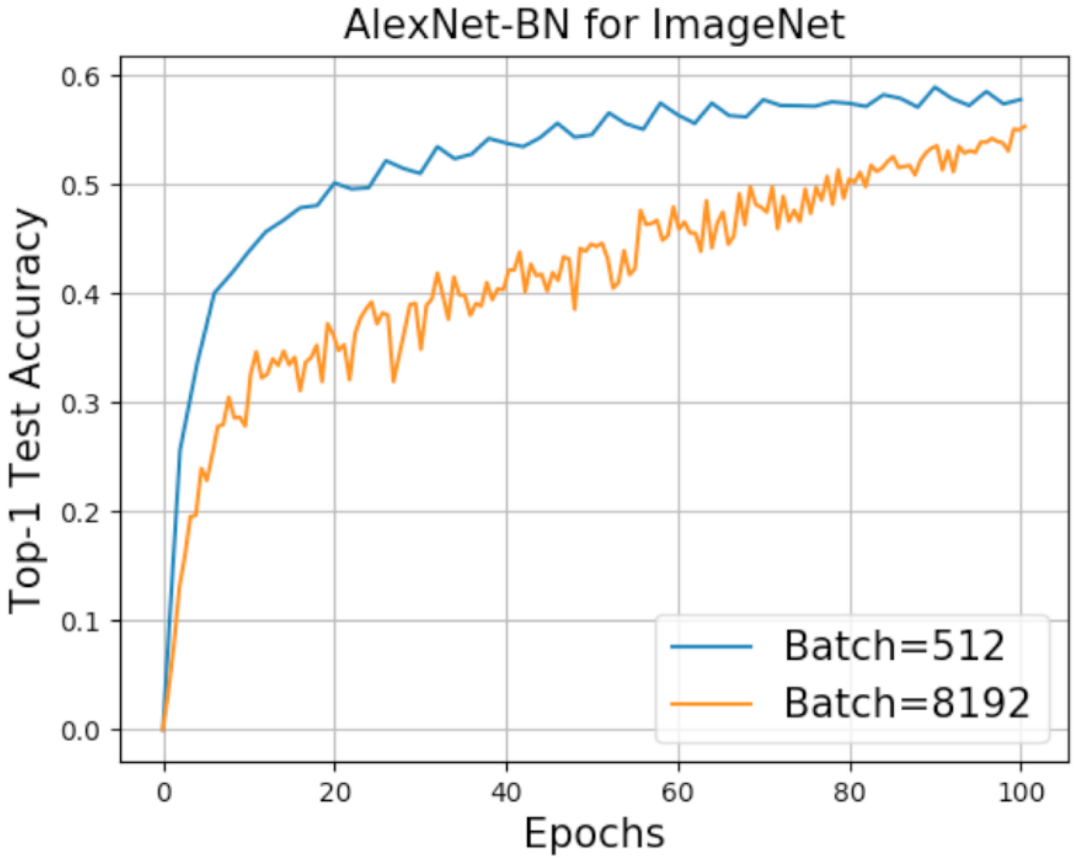
- Layer-wise Adaptive Rate Scaling (LARS)

Algorithm 1 SGD with LARS. Example with weight decay, momentum and polynomial LR decay.

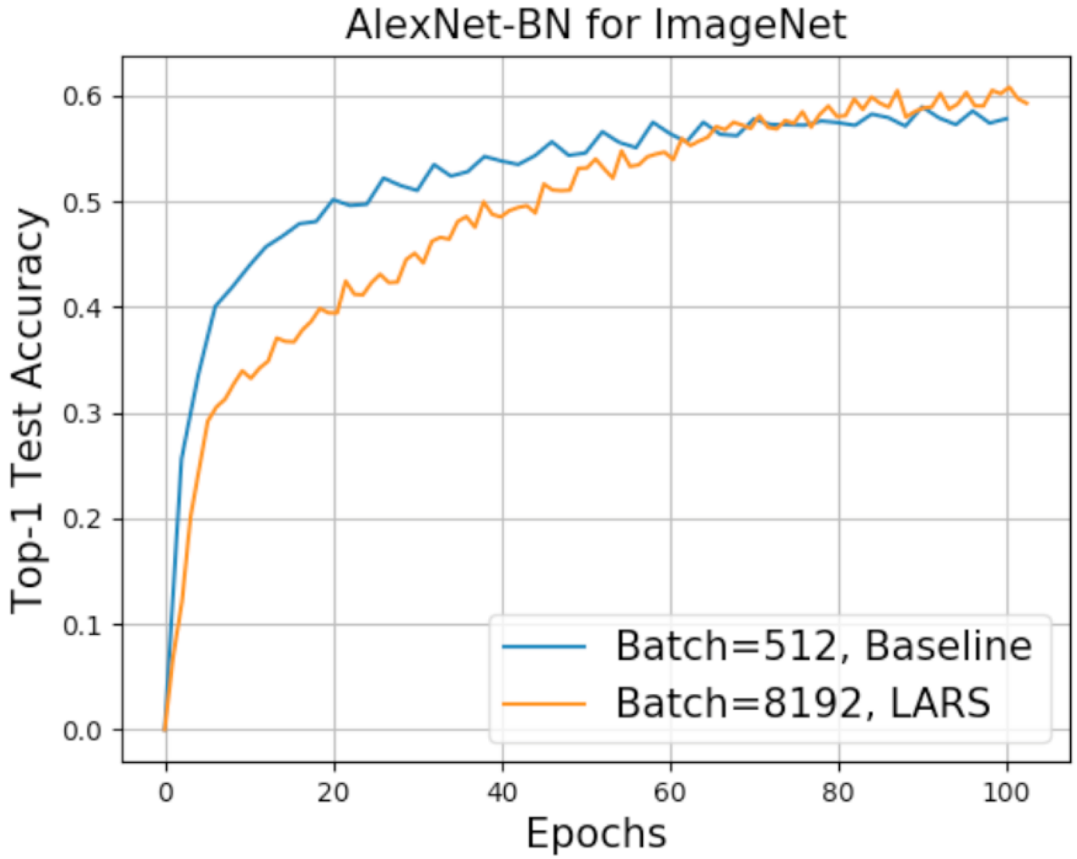
Parameters: base LR γ_0, momentum m, weight decay β, LARS coefficient η, number of steps T
Init: $t = 0, v = 0$. Init weight w_0^l for each layer l
while $t < T$ for each layer l **do**
$g_t^l \leftarrow \nabla L(w_t^l)$ (obtain a stochastic gradient for the current mini-batch)
$\gamma_t \leftarrow \gamma_0 * \left(1 - \frac{t}{T}\right)^2$ (compute the global learning rate)
$\lambda^l \leftarrow \frac{||w_t^l||}{||g_t^l|| + \beta||w_t^l||}$ (compute the local LR λ^l)
$v_{t+1}^l \leftarrow m v_t^l + \gamma_{t+1} * \lambda^l * (g_t^l + \beta w_t^l)$ (update the momentum)
$w_{t+1}^l \leftarrow w_t^l - v_{t+1}^l$ (update the weights)
end while

Yang You, et al. LARGE BATCH TRAINING OF CONVOLUTIONAL NETWORKS

Large Batch Training



(a) Training without LARS



(b) Training with LARS

Large Batch Training

- Trend
  - *Higher performance*
  - *Flexibility*
  - *Multi-task*

THANK YOU

Q & A