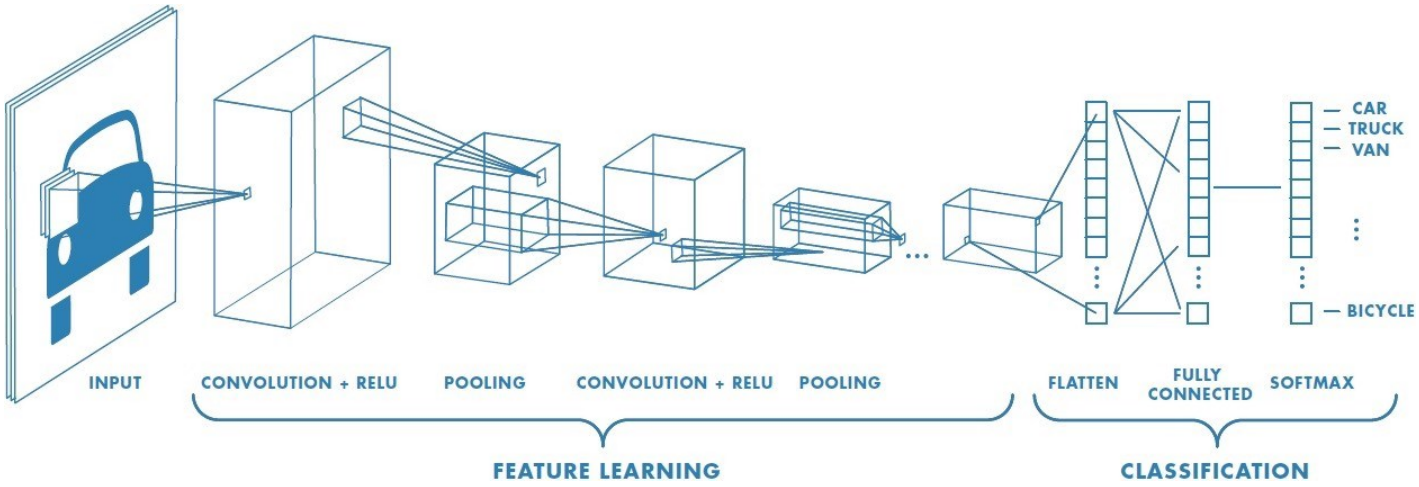
Modern CNNs I



Fast Campus
Start Deep Learning with TensorFlow

DenseNet

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

Abstract

Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections—one between each layer and its subsequent layer—our network has $\frac{L(L+1)}{2}$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage fea-

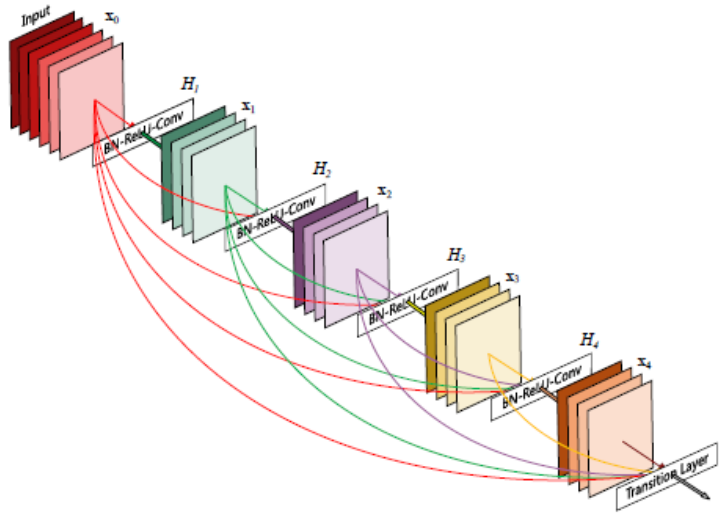
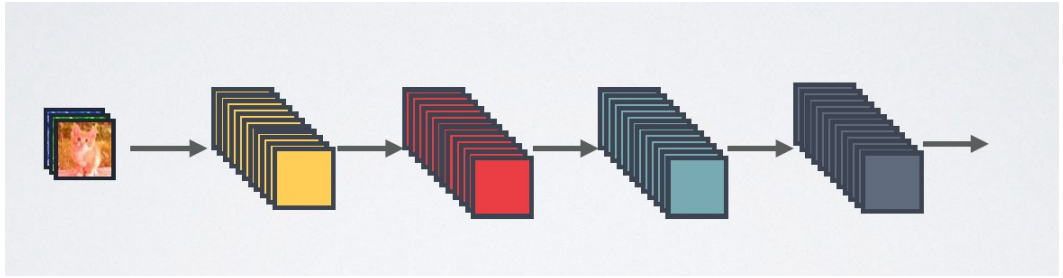


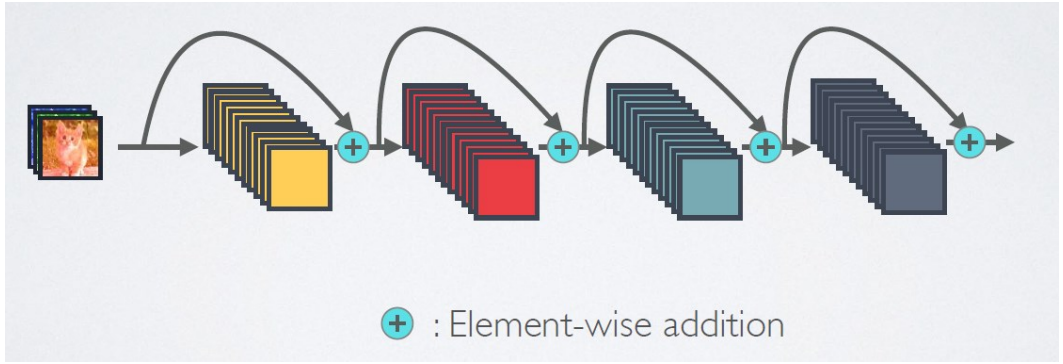
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Dense Connectivity

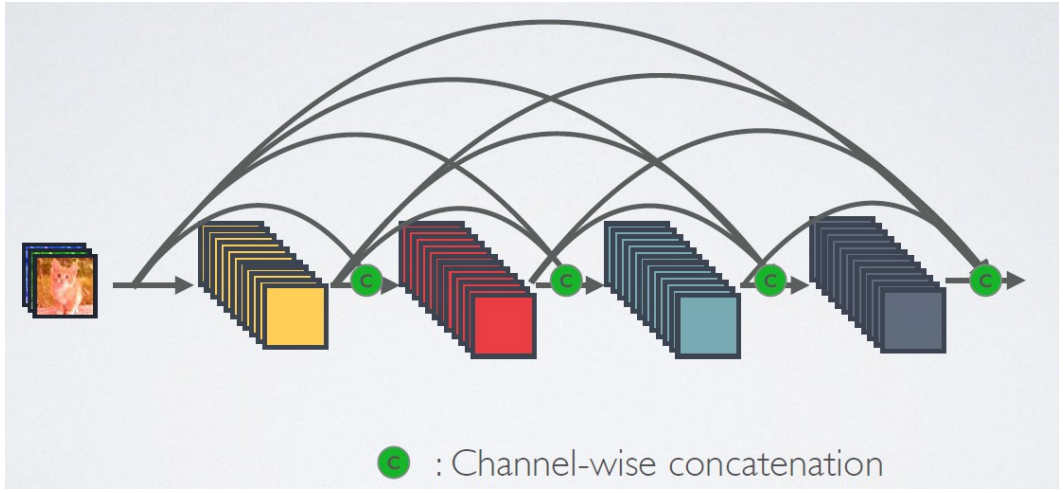
- Standard Connectivity



- ResNet Connectivity

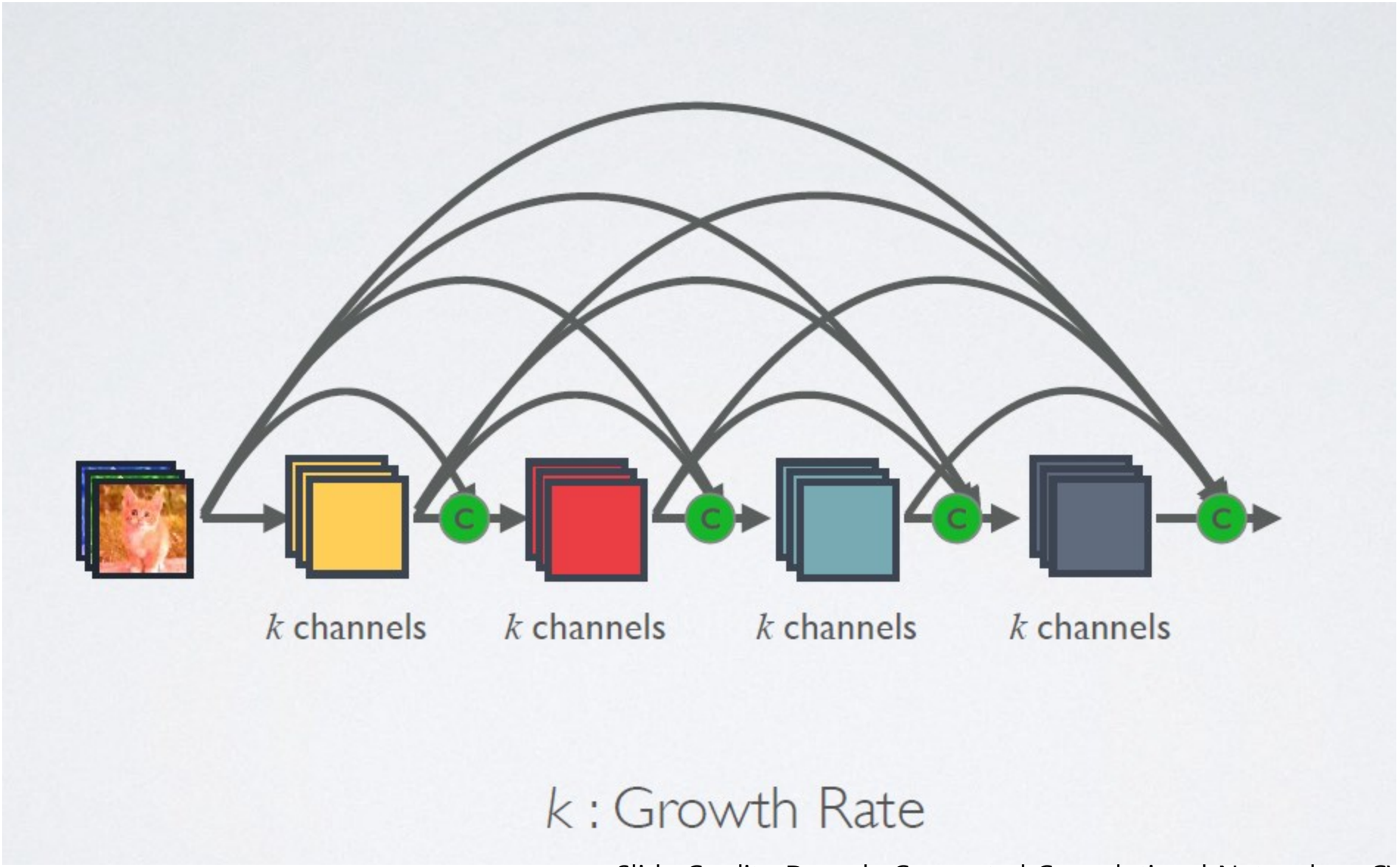


- Dense Connectivity



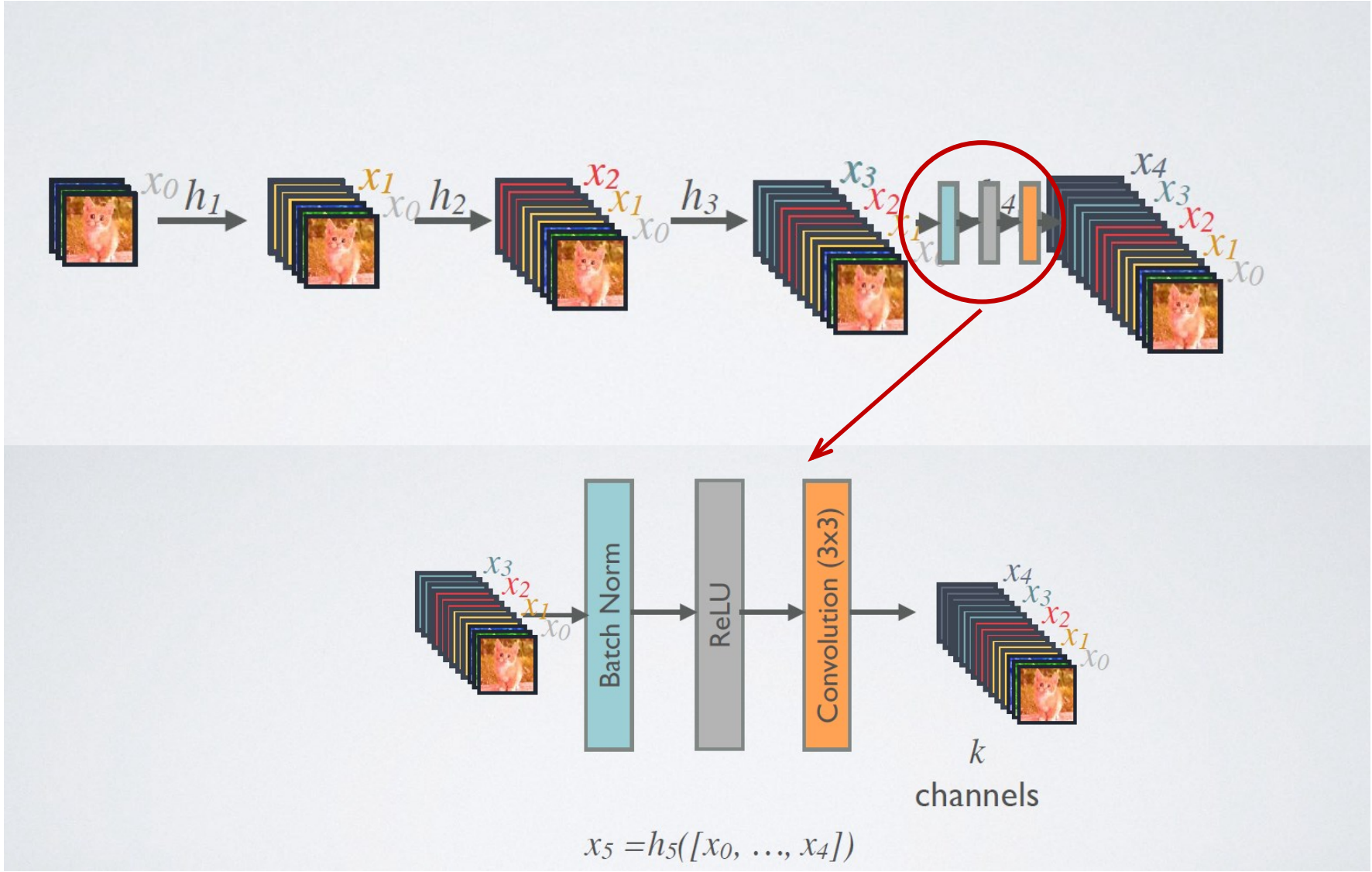
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Dense and Slim



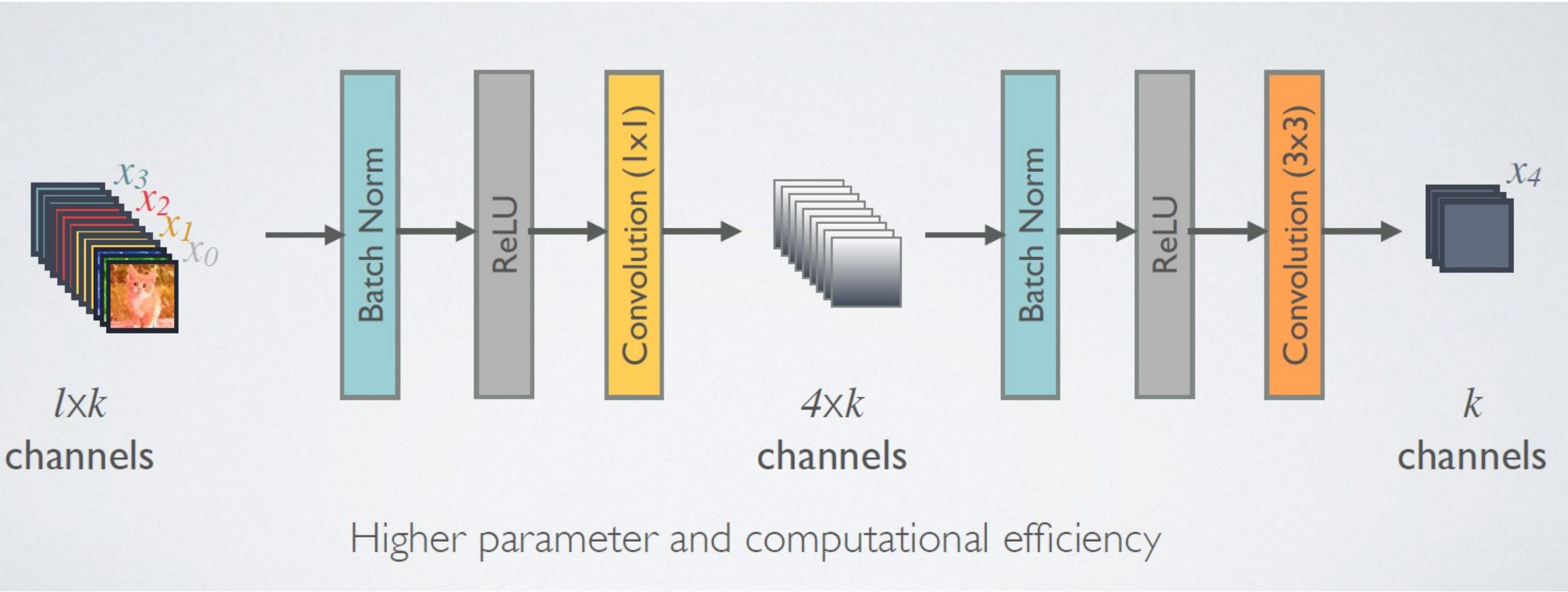
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Forward Propagation



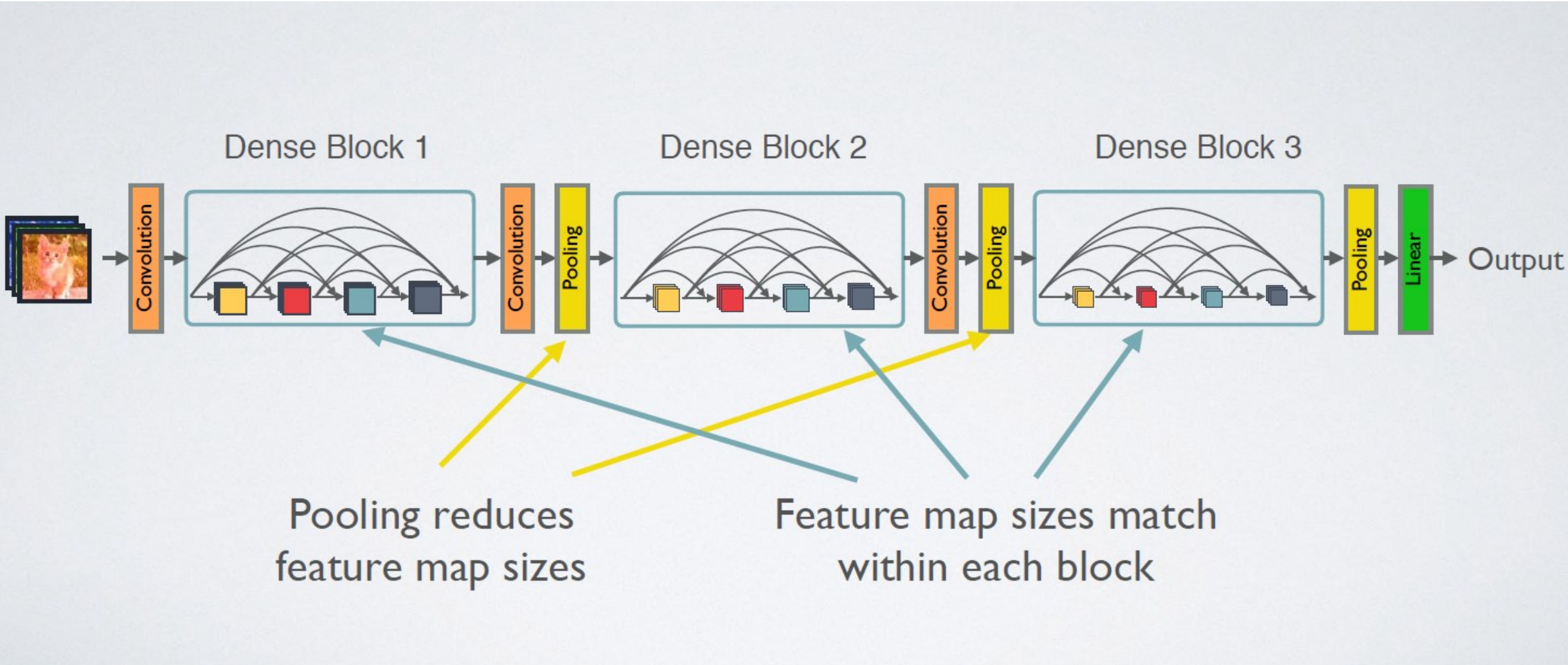
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Composite Layer in DenseNet with Bottleneck Layer



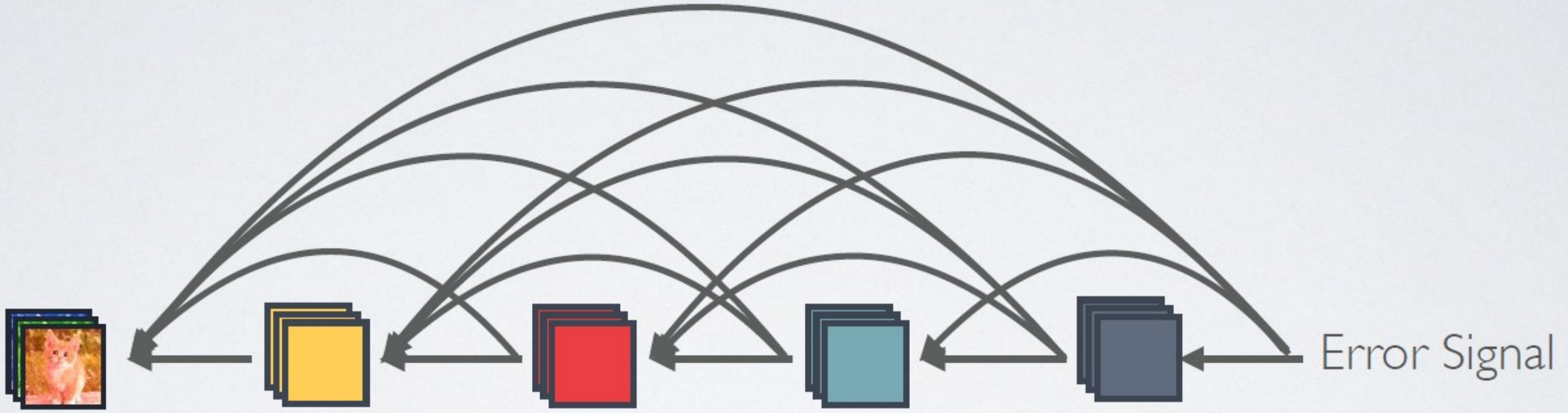
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

DenseNet



Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Advantage 1 : Strong Gradient Flow

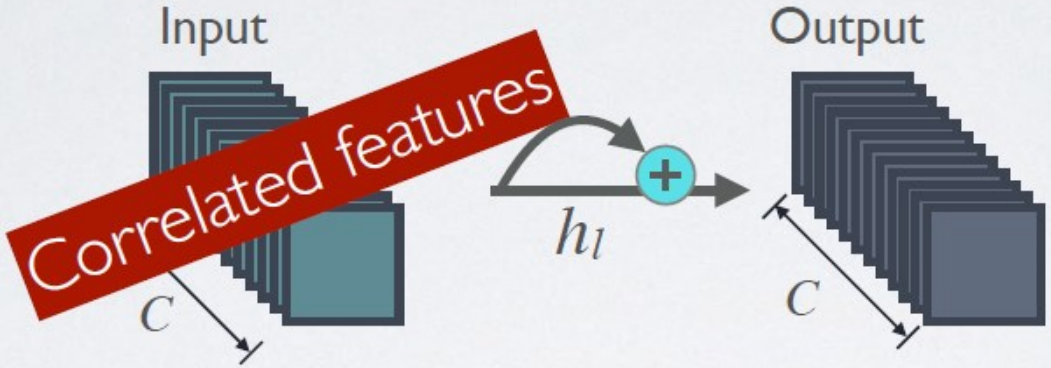


Implicit "deep supervision"

Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Advantage 2 : Parameter & Computational Efficiency

ResNet connectivity:



#parameters:

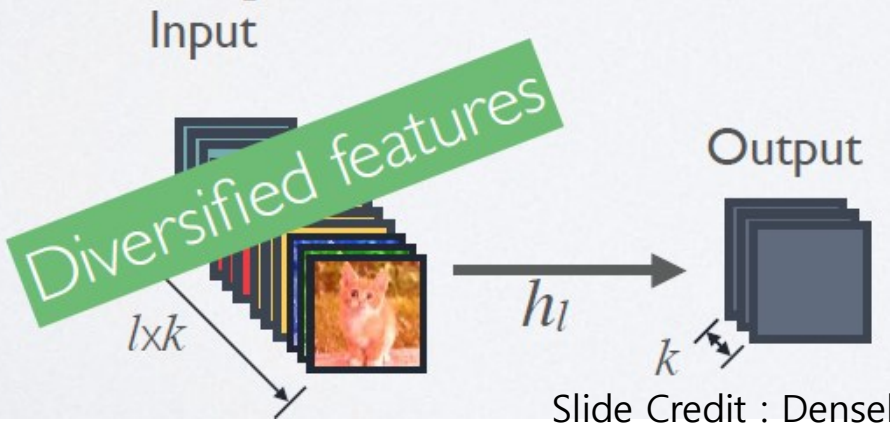
$O(C \times C)$

$k \ll C$

$O(l \times k \times k)$

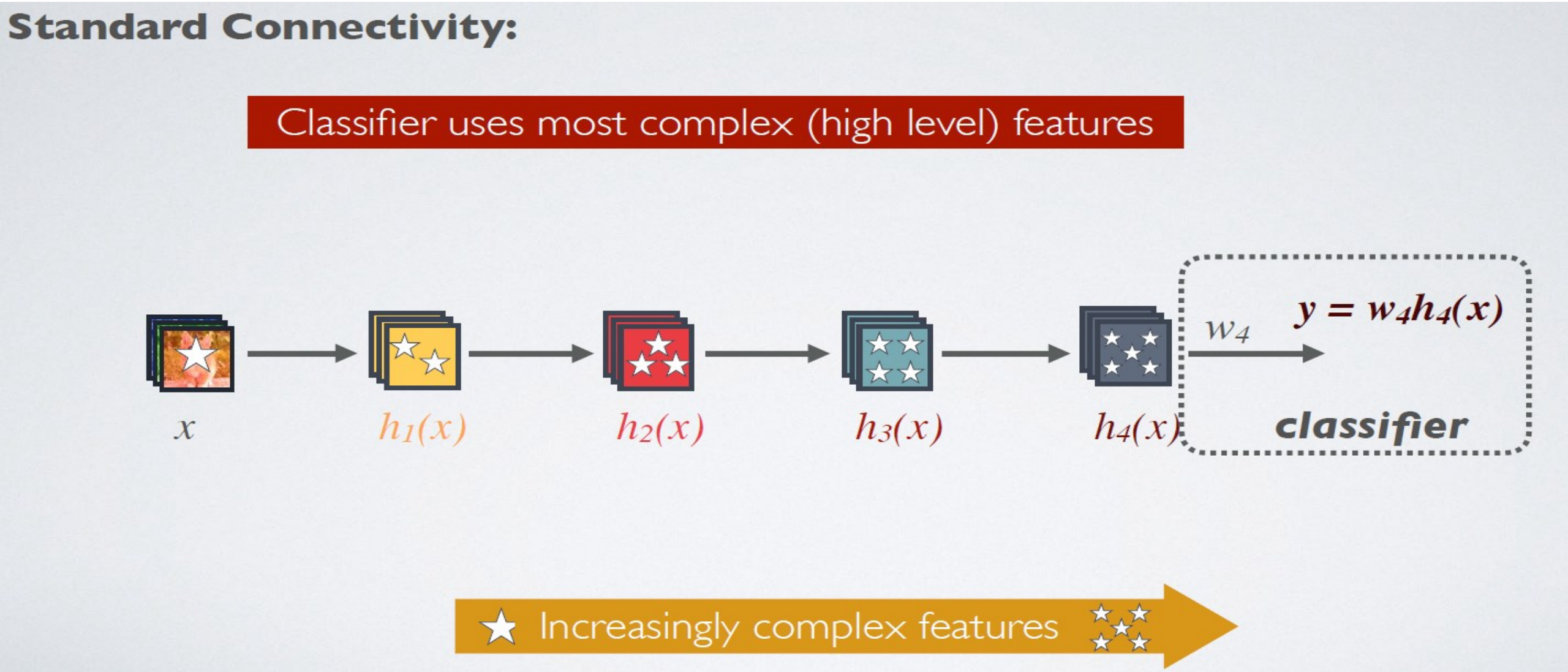
k: Growth rate

DenseNet connectivity:



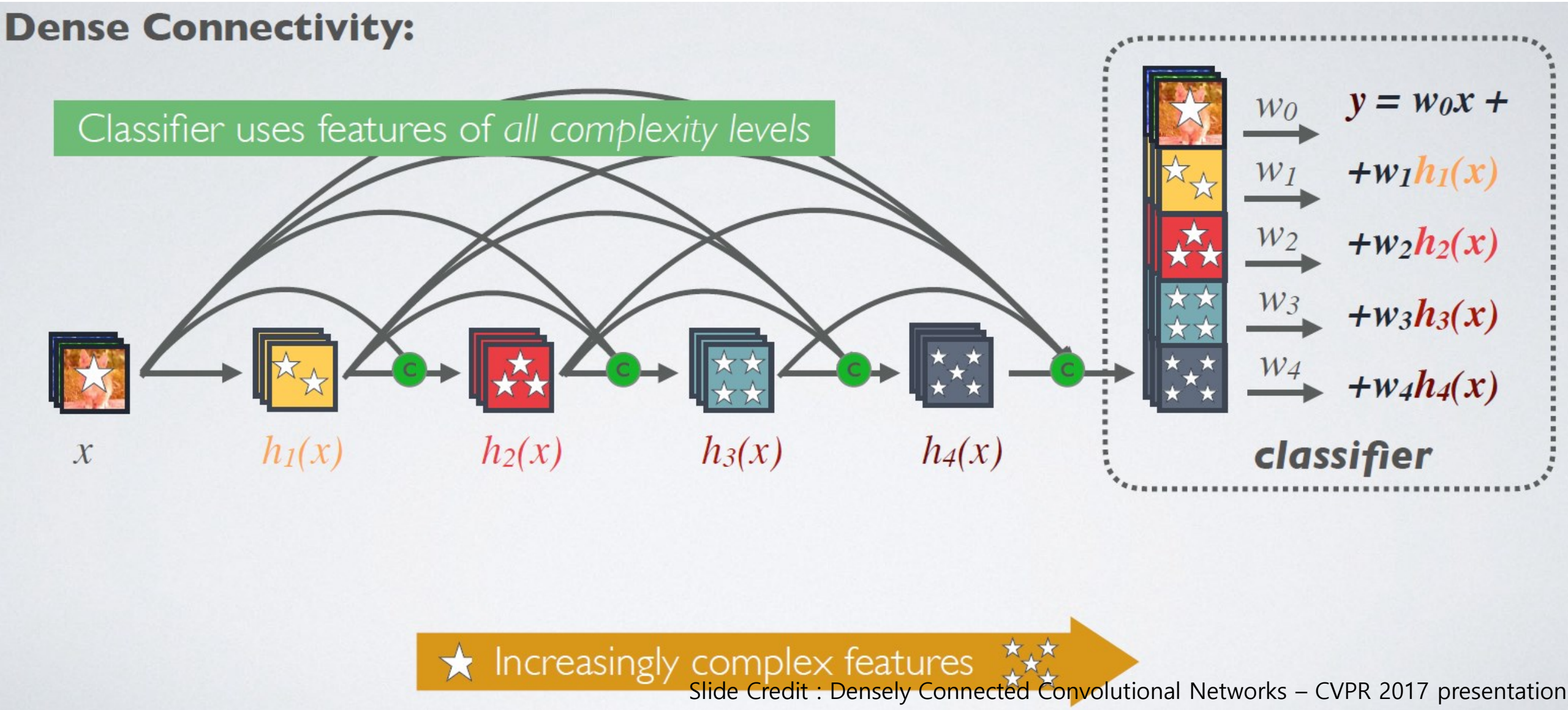
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Advantage 3 : Maintains Low Complexity Features



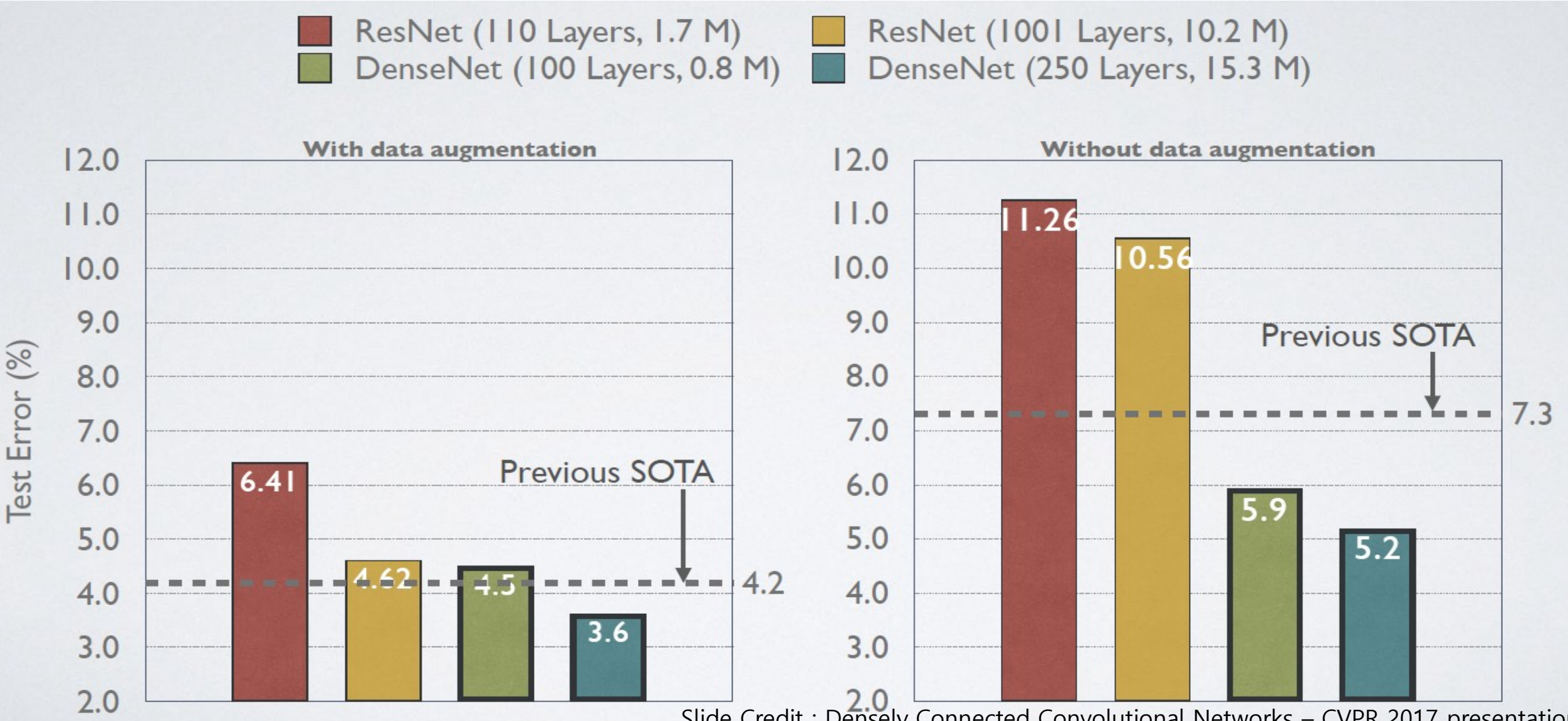
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Advantage 3 : Maintains Low Complexity Features



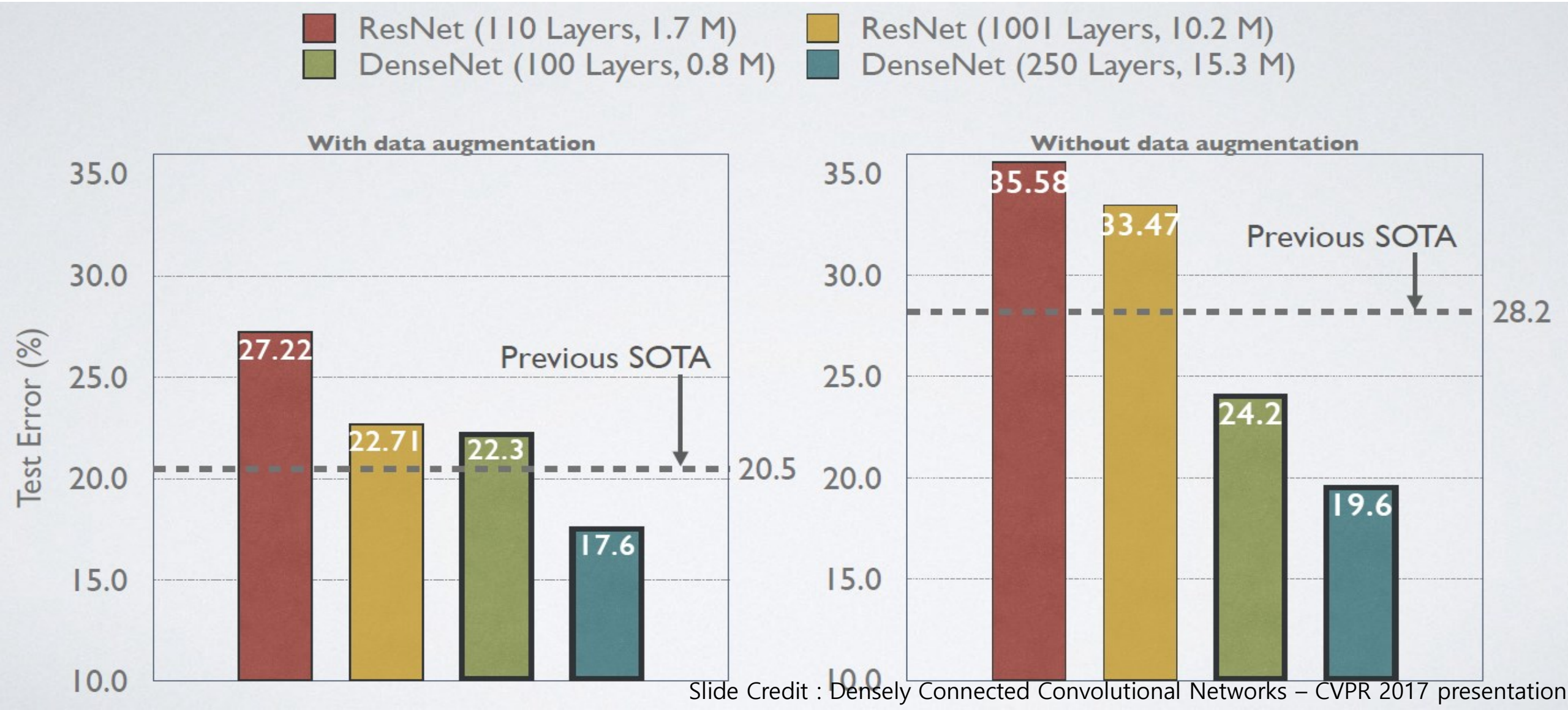
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Results on CIFAR-10



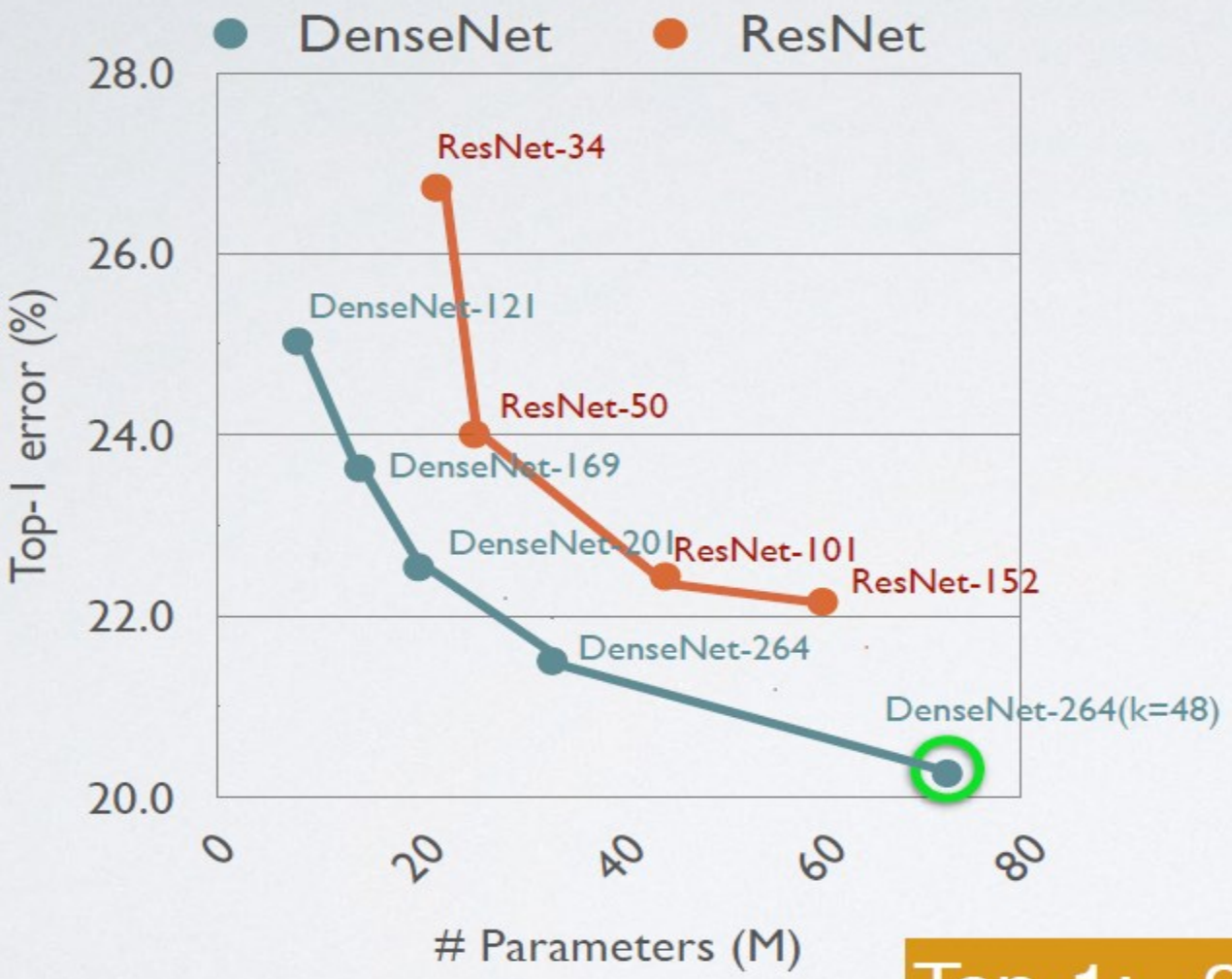
Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

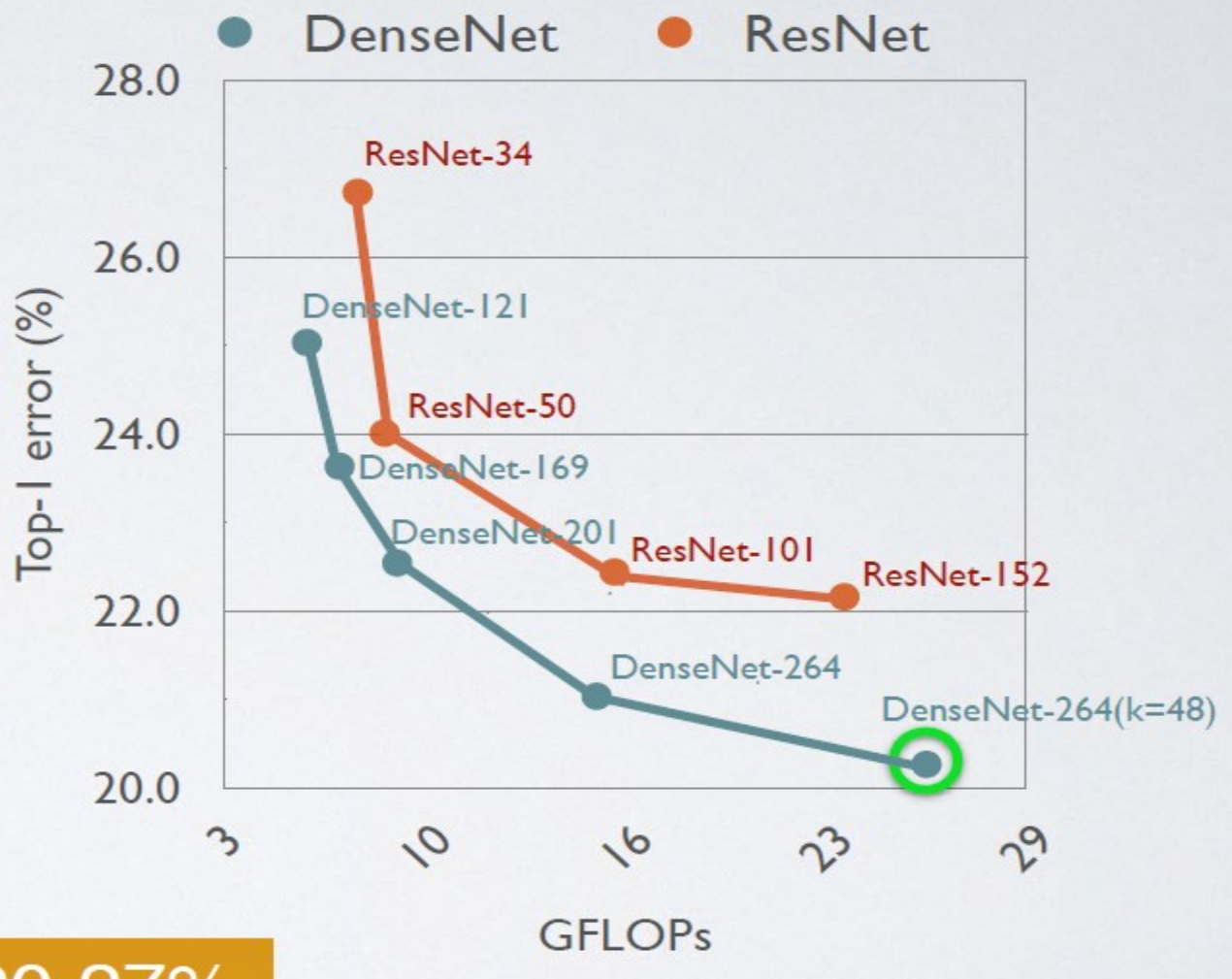
Results on CIFAR-100



Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

Results on ImageNet





Top-1: 20.27%
Top-5: 5.17%

Slide Credit : Densely Connected Convolutional Networks – CVPR 2017 presentation

ResNeXt

Aggregated Residual Transformations for Deep Neural Networks

Saining Xie[1] Ross Girshick[2] Piotr Dollár[2] Zhuowen Tu[1] Kaiming He[2]
[1]UC San Diego [2]Facebook AI Research
{s9xie,ztu}@ucsd.edu {rbg,pdollar,kaiminghe}@fb.com

Abstract

We present a simple, highly modularized network architecture for image classification. Our network is constructed by repeating a building block that aggregates a set of transformations with the same topology. Our simple design results in a homogeneous, multi-branch architecture that has only a few hyper-parameters to set. This strategy exposes a new dimension, which we call "cardinality" (the size of the set of transformations), as an essential factor in addition to the dimensions of depth and width. On the ImageNet-1K dataset, we empirically show that even under the restricted condition of maintaining complexity, increasing cardinality is able to improve classification accuracy. Moreover, in-

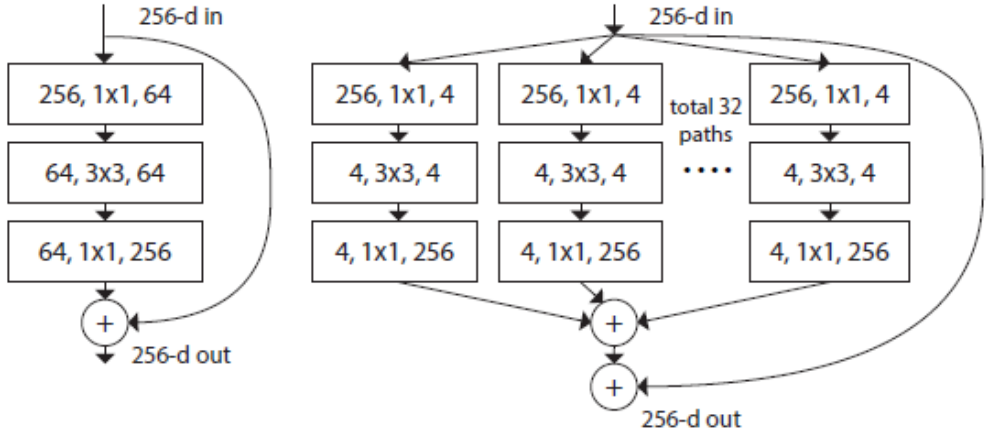


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

ImageNet 2016 Results

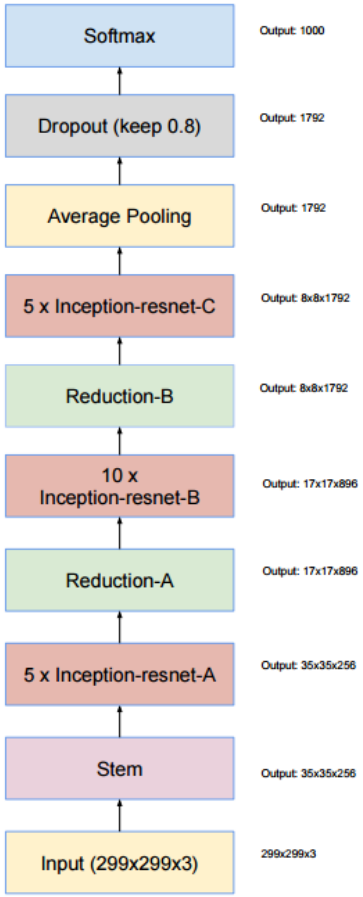
Team name	Entry description	**Classification error**	Localization error
Trimps-Soushen	Ensemble 2	0.02991	0.077668
Trimps-Soushen	Ensemble 3	0.02991	0.077087
Trimps-Soushen	Ensemble 4	0.02991	0.077429
ResNeXt	Ensemble C, weighted average, tuned on val. [No bounding box results]	0.03031	0.737308
CU-DeepLink	GrandUnion + Fused-scale EnsembleNet	0.03042	0.098892
CU-DeepLink	GrandUnion + Multi-scale EnsembleNet	0.03046	0.099006
CU-DeepLink	GrandUnion + Basic Ensemble	0.03049	0.098954
ResNeXt	Ensemble B, weighted average, tuned on val. [No bounding box results]	0.03092	0.737484
CU-DeepLink	GrandUnion + Class-reweighted Ensemble	0.03096	0.099369
CU-DeepLink	GrandUnion + Class-reweighted Ensemble with Per-instance Normalization	0.03103	0.099349
ResNeXt	Ensemble C, weighted average. [No bounding box results]	0.03124	0.737526
Trimps-Soushen	Ensemble 1	0.03144	0.079068
ResNeXt	Ensemble A, simple average. [No bounding box results]	0.0315	0.737505
SamExynos	3 model only for classification	0.03171	0.236561
ResNeXt	Ensemble B, weighted average. [No bounding box results]	0.03203	0.737681
KAISTNIA_ETRI	Ensembles A	0.03256	0.102015
KAISTNIA_ETRI	Ensembles C	0.03256	0.102056
KAISTNIA_ETRI	Ensembles B	0.03256	0.100676

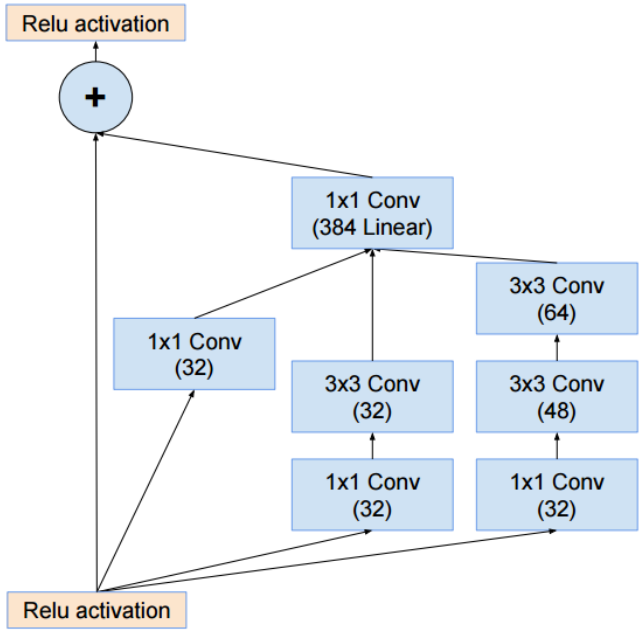
Growing Number of Hyper-parameters

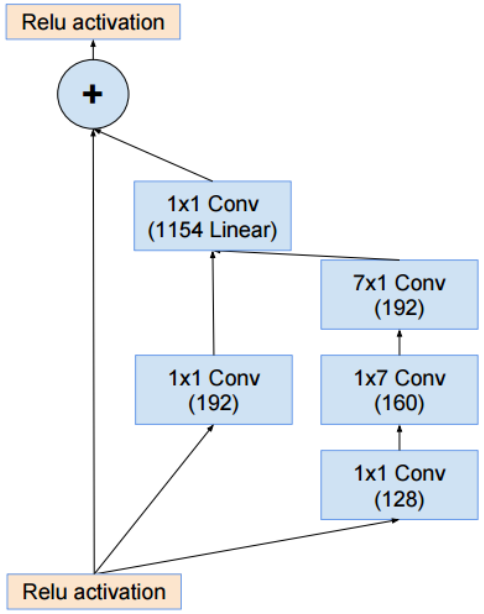
- VGGNet exhibit a simple yet effective strategy of constructing very deep network – stacking building blocks of the same shape
- ResNet inherit this strategy with stacking modules of the same topology
- Unlike VGGNet, the family of Inception models have demonstrated that carefully designed topologies are able to achieve compelling accuracy
  - Important common property is split-transform-merge strategy
  - Split – 1x1 conv, transform – 3x3, 5x5 conv, merge - concatenation

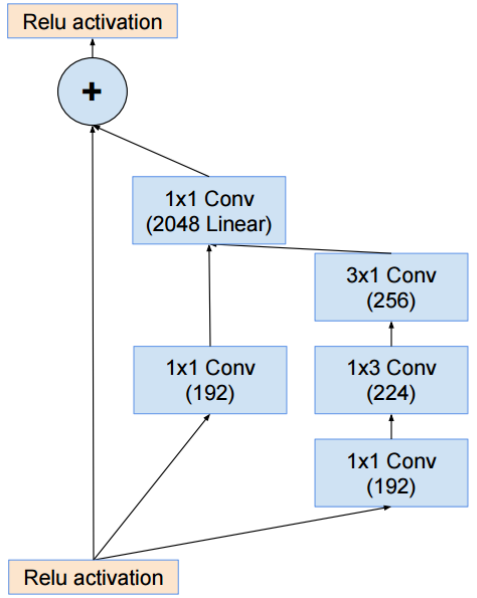
Inception Learns ResNet

- Inception + ResNet





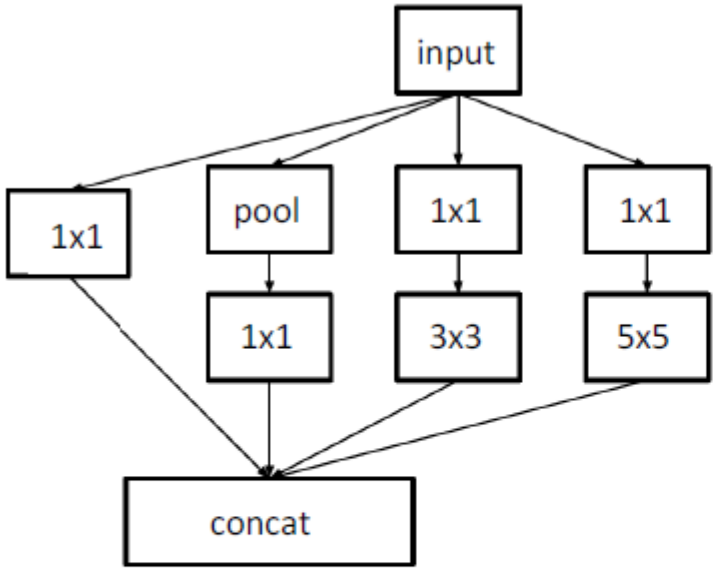




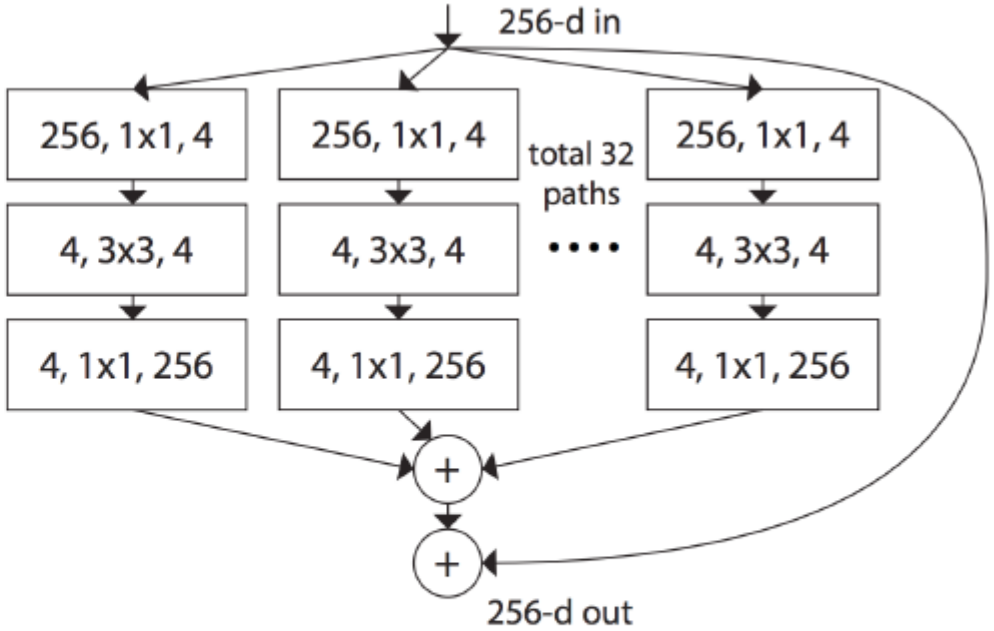
"Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning"

ResNet Learns Inception??

- Multi-branch + ResNet

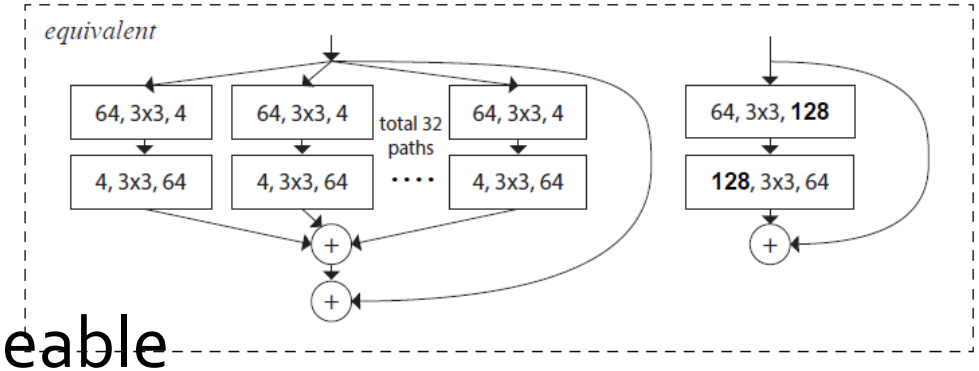


Inception:
heterogeneous multi-branch

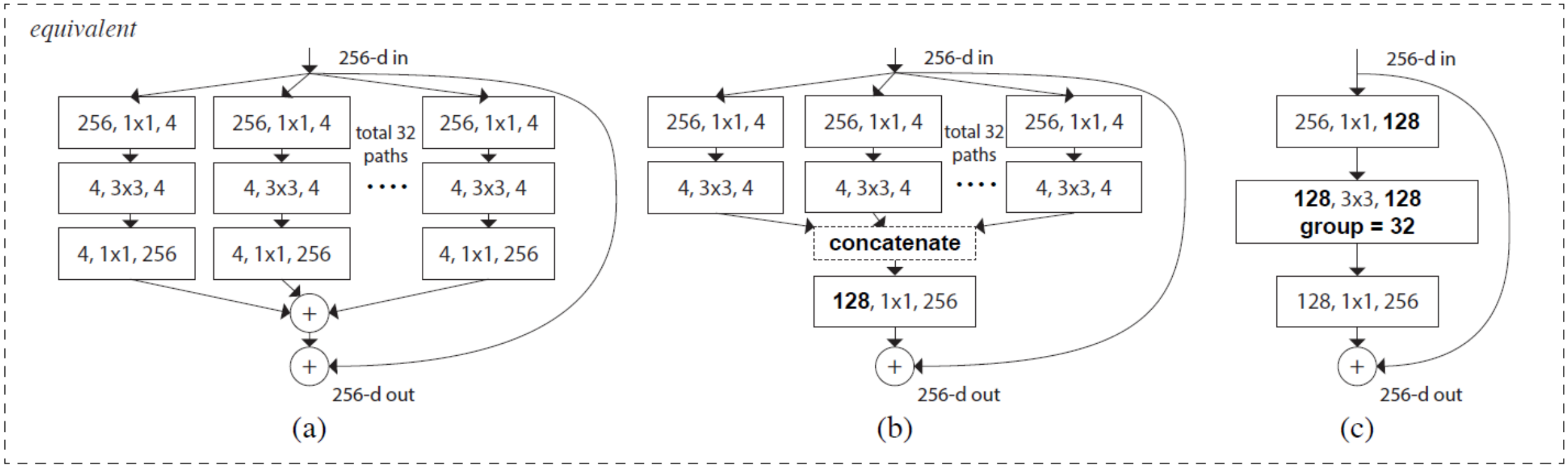


ResNeXt:
uniform multi-branch

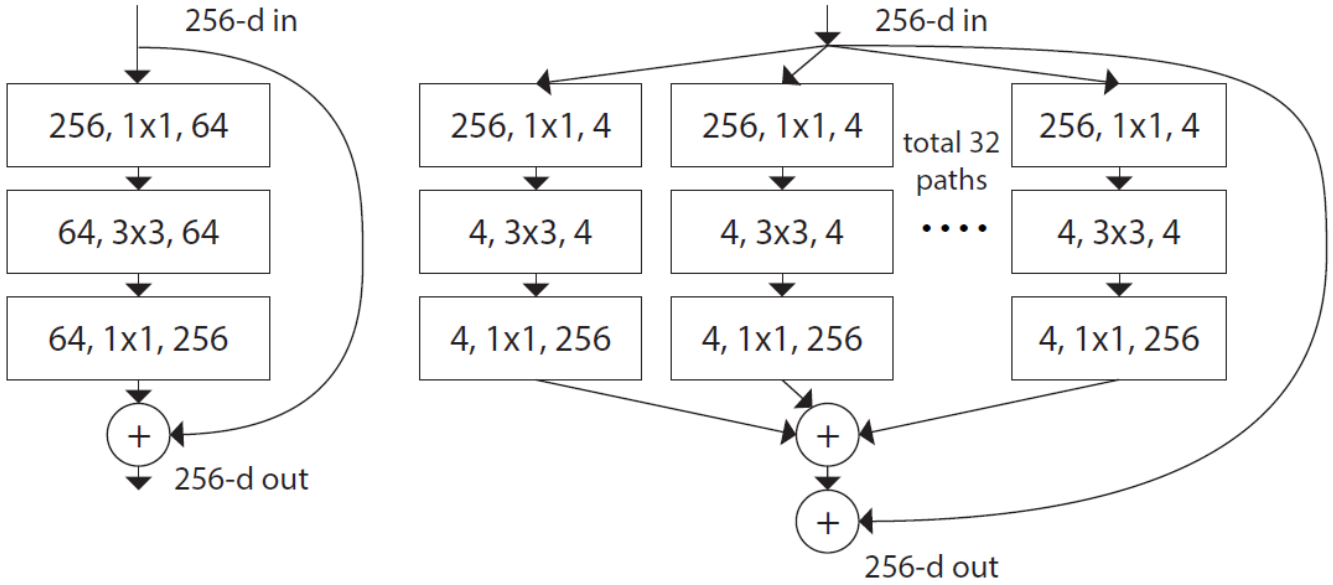
ResNeXt



- Concatenation and Addition are interchangeable
- Uniform multi-branching can be done by group-conv



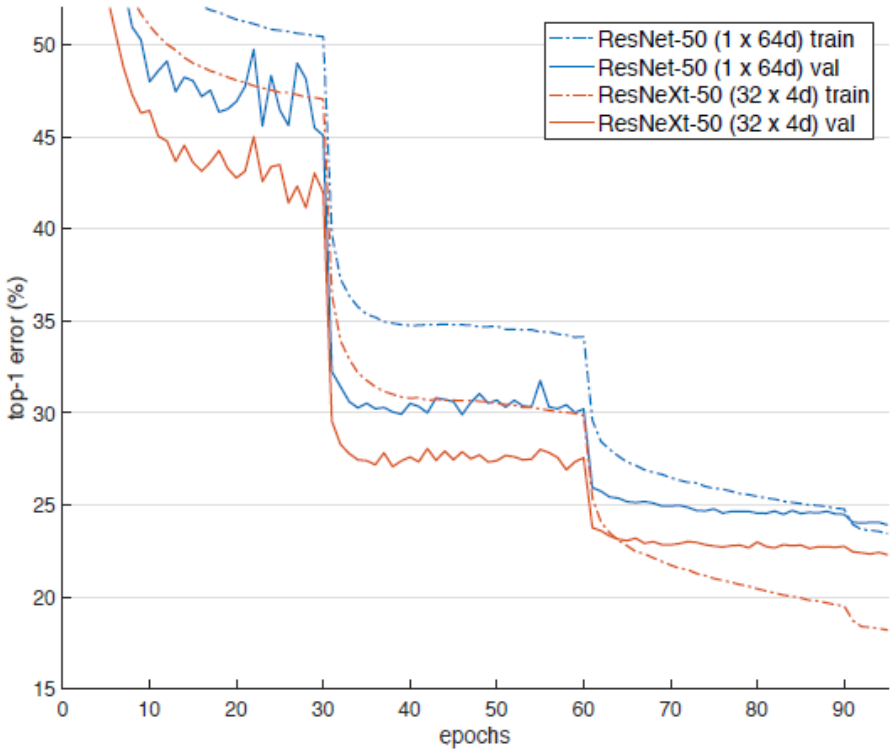
Model Capacity (# of parameters)

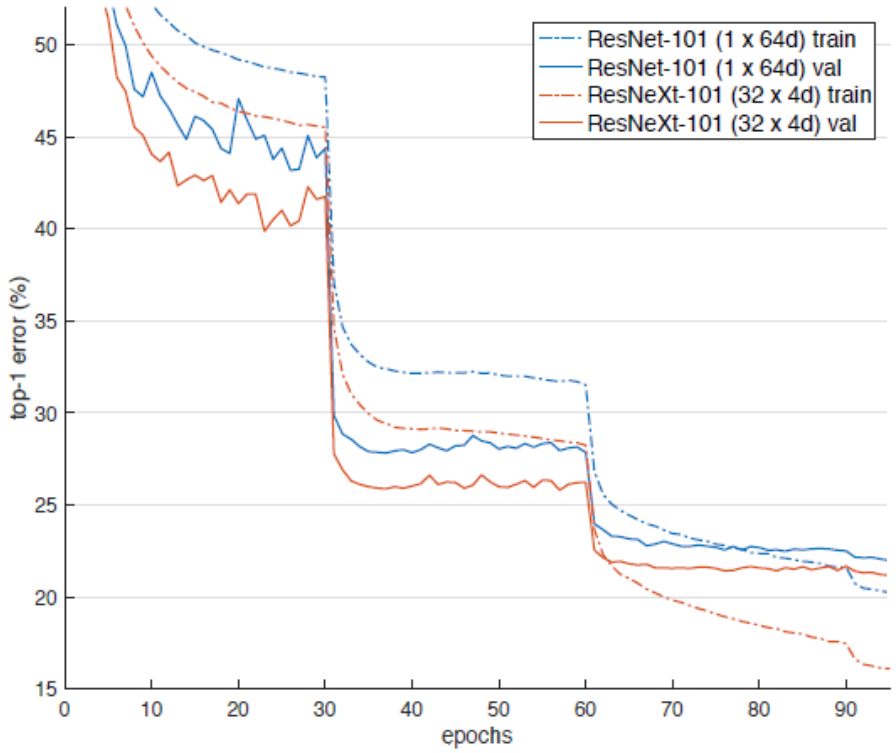


- Original ResNet(left) : 256x64 + 3x3x64x64 + 64x256 = 70k
- ResNeXt(right – with bottleneck width d and cardinality C) :

  C x (256xd + 3x3xdxd + dx256) = 70k, when C =32 and d=4

Results – Cardinality vs Width





	setting	top-1 error (%)
ResNet-50	1 × 64d	23.9
ResNeXt-50	2 × 40d	23.0
ResNeXt-50	4 × 24d	22.6
ResNeXt-50	8 × 14d	22.3
ResNeXt-50	32 × 4d	**22.2**
ResNet-101	1 × 64d	22.0
ResNeXt-101	2 × 40d	21.7
ResNeXt-101	4 × 24d	21.4
ResNeXt-101	8 × 14d	21.3
ResNeXt-101	32 × 4d	**21.2**

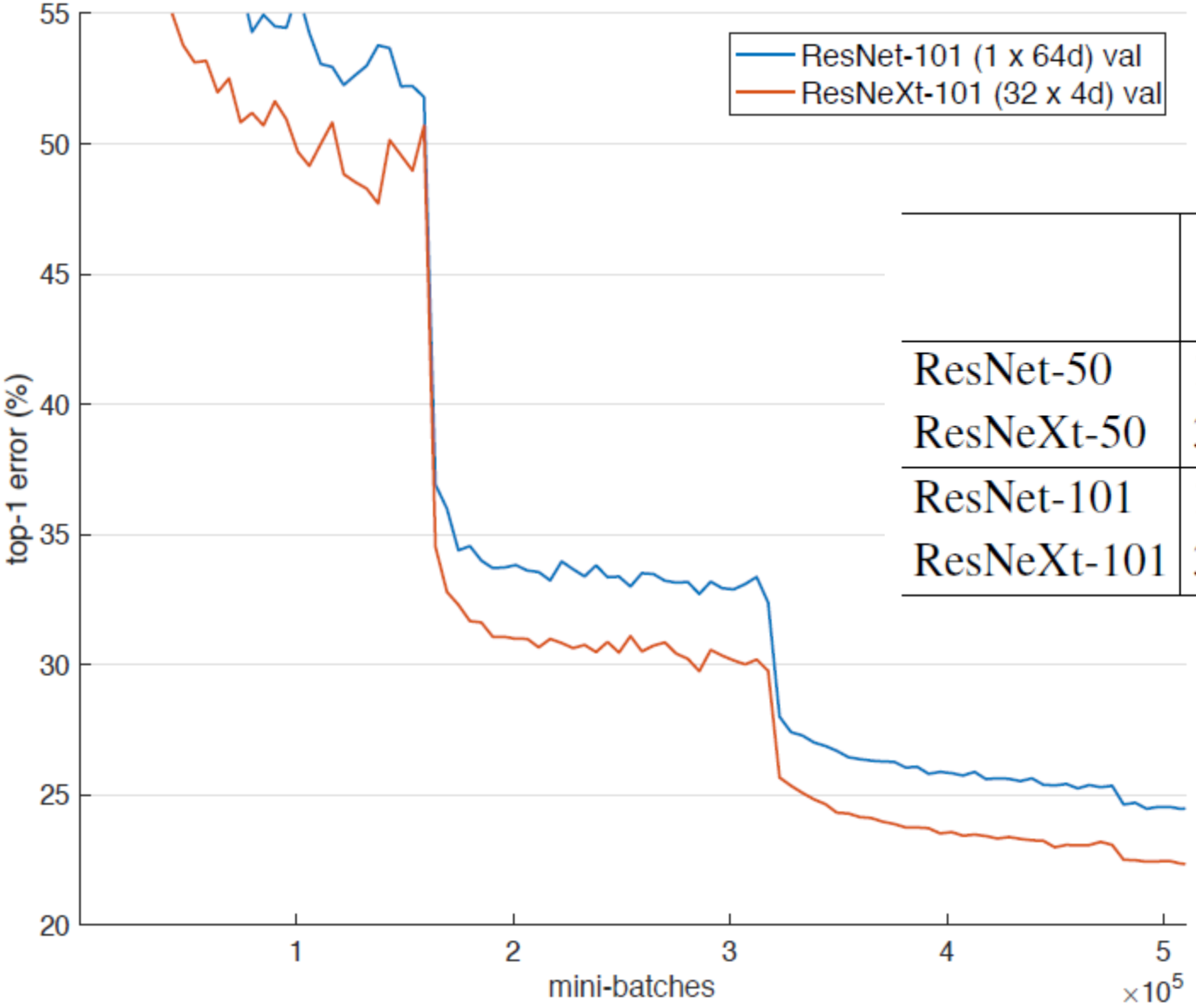
Results – Increasing Cardinality vs Deeper/Wider

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Results – vs SOTA Networks

	224×224		320×320 / 299×299	
	top-1 err	top-5 err	top-1 err	top-5 err
ResNet-101 [14]	22.0	6.0	-	-
ResNet-200 [15]	21.7	5.8	20.1	4.8
Inception-v3 [39]	-	-	21.2	5.6
Inception-v4 [37]	-	-	20.0	5.0
Inception-ResNet-v2 [37]	-	-	19.9	4.9
ResNeXt-101 (**64** × **4d**)	20.4	5.3	**19.1**	**4.4**

Results – ImageNet-5K



	setting	5K-way classification		1K-way classification	
		top-1	top-5	top-1	top-5
ResNet-50	1 × 64d	45.5	19.4	27.1	8.2
ResNeXt-50	32 × 4d	**42.3**	**16.8**	**24.4**	**6.6**
ResNet-101	1 × 64d	42.4	16.9	24.2	6.8
ResNeXt-101	32 × 4d	**40.1**	**15.1**	**22.2**	**5.7**

Squeeze-and-Excitation Networks

Squeeze-and-Excitation Networks

Jie Hu[0000−0002−5150−1003] Li Shen[0000−0002−2283−4976] Samuel Albanie[0000−0001−9736−5134]
Gang Sun[0000−0001−6913−6799] Enhua Wu[0000−0002−2174−1428]

Abstract—The central building block of convolutional neural networks (CNNs) is the convolution operator, which enables networks to construct informative features by fusing both spatial and channel-wise information within local receptive fields at each layer. A broad range of prior research has investigated the spatial component of this relationship, seeking to strengthen the representational power of a CNN by enhancing the quality of spatial encodings throughout its feature hierarchy. In this work, we focus instead on the channel relationship and propose a novel architectural unit, which we term the "Squeeze-and-Excitation" (SE) block, that adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. We show that these blocks can be stacked together to form SENet architectures that generalise extremely effectively across different datasets. We further demonstrate that SE blocks bring significant improvements in performance for existing state-of-the-art CNNs at minimal additional computational cost. Squeeze-and-Excitation Networks formed the foundation of our ILSVRC 2017 classification submission which won first place and reduced the top-5 error to 2.251%, surpassing the winning entry of 2016 by a relative improvement of ∼25%. Models and code are available at https://github.com/hujie-frank/SENet.

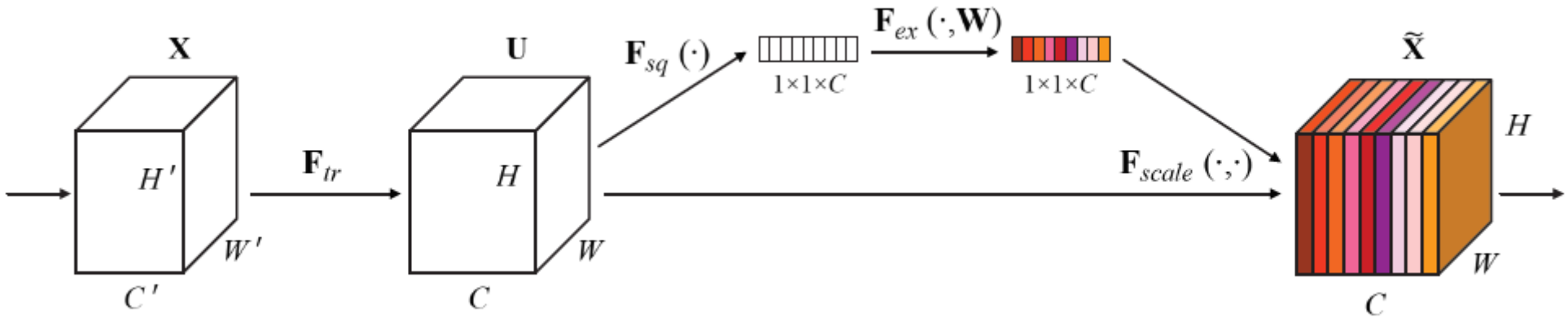
Index Terms—Squeeze-and-Excitation, Image classification, Convolutional Neural Network.

Squeeze-and-Excitation Networks

Team name	Entry description	**Classification error**	Localization error
WMW	Ensemble C [No bounding box results]	0.02251	0.590987
WMW	Ensemble E [No bounding box results]	0.02258	0.591018
WMW	Ensemble A [No bounding box results]	0.0227	0.591153
WMW	Ensemble D [No bounding box results]	0.0227	0.591039
WMW	Ensemble B [No bounding box results]	0.0227	0.59106
Trimps-Soushen	Result-1	0.02481	0.067698
Trimps-Soushen	Result-2	0.02481	0.06525
Trimps-Soushen	Result-3	0.02481	0.064991
Trimps-Soushen	Result-4	0.02481	0.065261
Trimps-Soushen	Result-5	0.02481	0.065302
NUS-Qihoo_DPNs (CLS-LOC)	[E2] CLS:: Dual Path Networks + Basic Ensemble	0.0274	0.088093
NUS-Qihoo_DPNs (CLS-LOC)	[E1] CLS:: Dual Path Networks + Basic Ensemble	0.02744	0.088269
BDAT	provide_class	0.02962	0.086942
BDAT	provide_box	0.03158	0.081392
MIL_UT	Ensemble of 9 models (classification-only)	0.03205	0.596164
SIIT_KAIST-SKT	ensemble 2	0.03226	0.128924
MIL_UT	Ensemble of 10 models (classification-only)	0.03228	0.596174

Squeeze-and-Excitation Blocks

- Improving the quality of representations produced by a network by explicitly modelling the interdependencies between the channels of its convolutional features

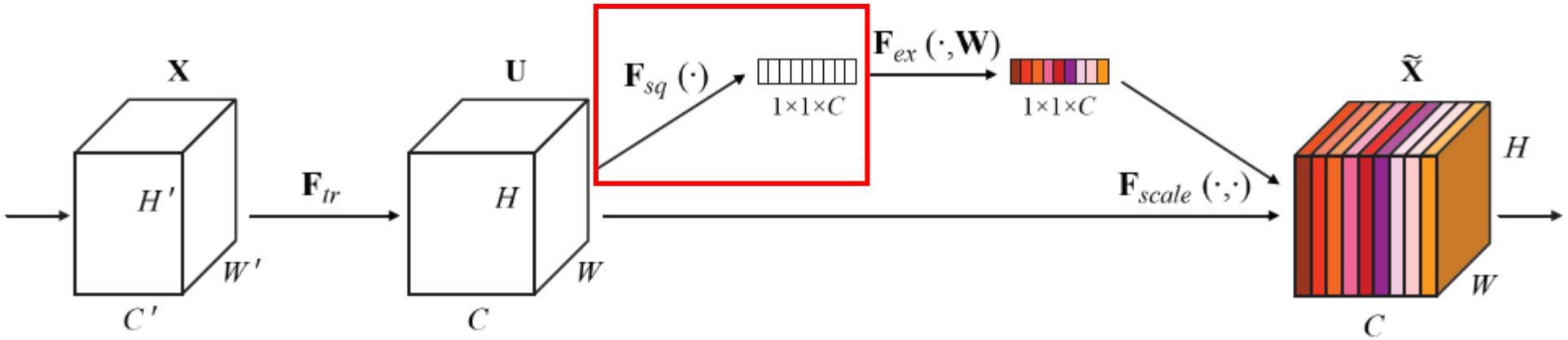


Standard Convolution

- In standard convolution case, the output is produced by a summation through all channels
- Channel dependencies are implicitly embedded in output feature maps
- But, they are entangled with the local spatial correlation captured by filters

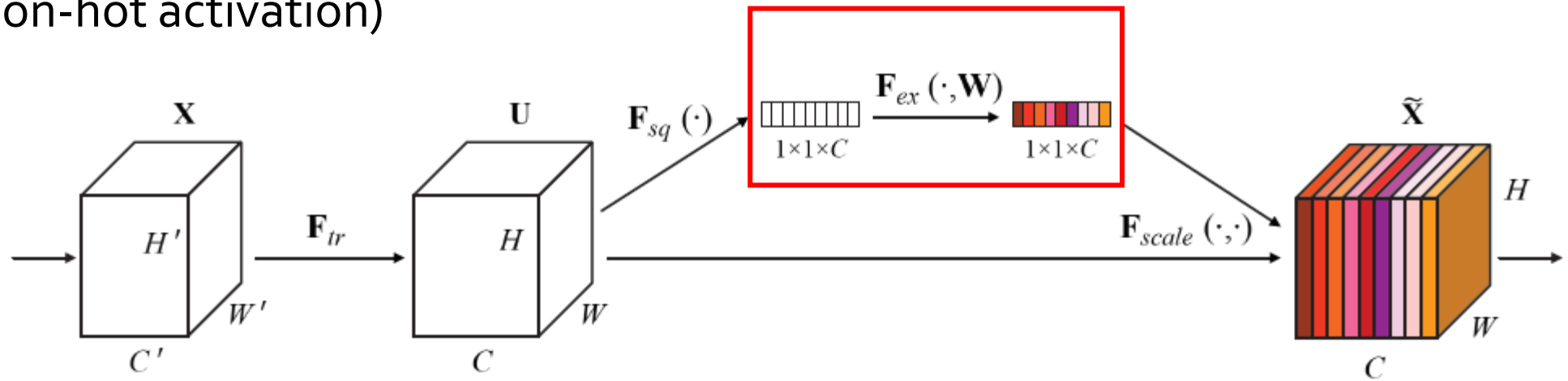
Squeeze : Global Information Embedding

- Authors propose to squeeze global spatial information into a channel descriptor.
- This is achieved by using global average pooling to generate channel-wise statistics.
- The output of the transformation(GAP) can be interpreted as a collection of the local descriptors whose statistics are expressive for whole image



Excitation : Adaptive Recalibration

- To make use of the information aggregated in the squeeze operation, authors follow it with a second operation which aims to fully captured channel-wise dependencies.
- The function must meet two criteria
  - It must be flexible(it must be capable of learning a nonlinear interaction btw channels)
  - It must learn a non-mutually-exclusive relationship(rather than enforcing a on-hot activation)

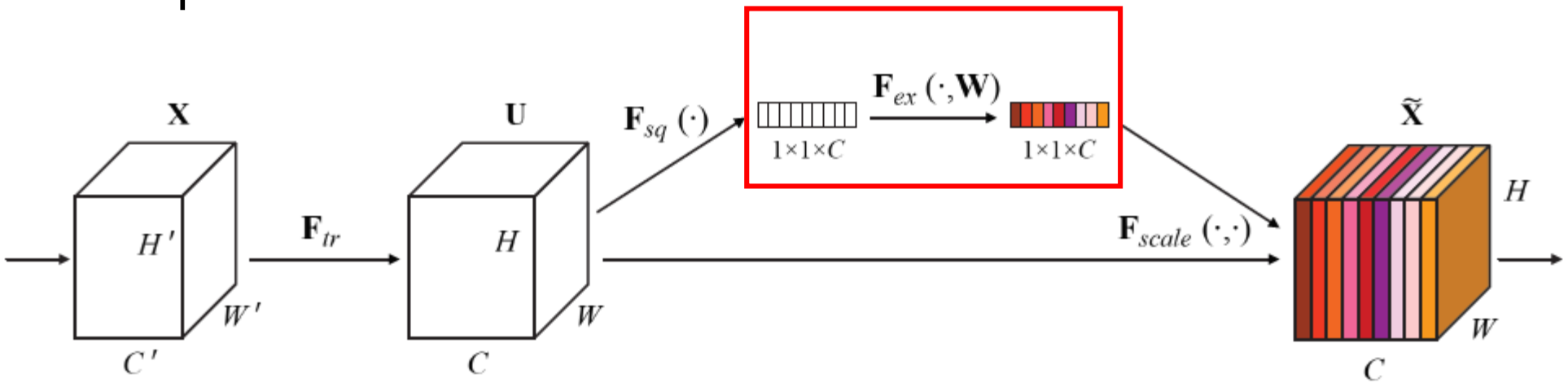


Excitation : Adaptive Recalibration

- To meet these criteria, authors opt to employ a simple gating mechanism with a sigmoid activation (σ : sigmoid, δ : ReLU)

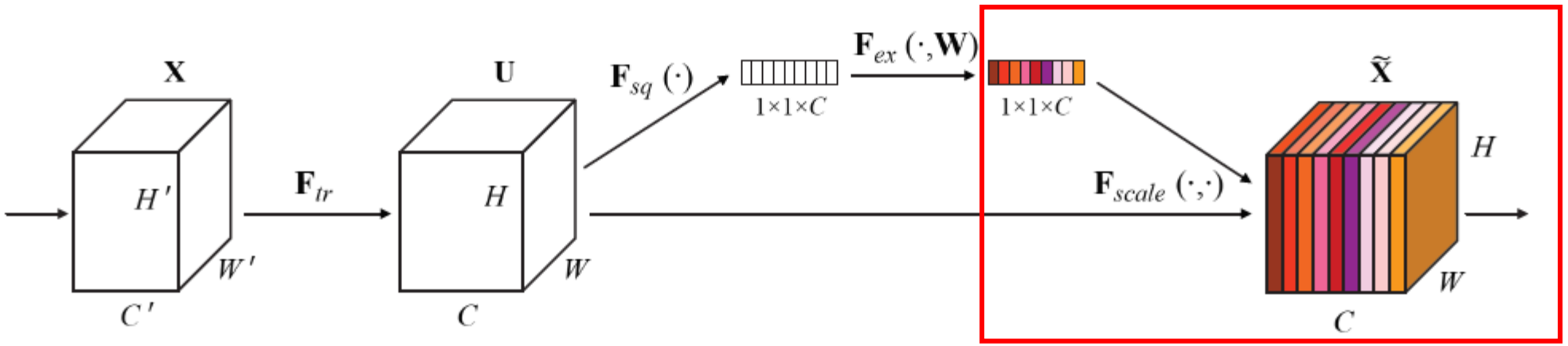
$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2\delta(\mathbf{W}_1\mathbf{z})),$$

- Forming a bottleneck with two fully connected layers around the non-linearity. A dimensionality-reduction layer with parameters W1 and reduction ratio r, a ReLU and then a dimensionality-increasing layer with parameters W2



Excitation : Adaptive Recalibration

- The final output of the block is obtained by rescaling the transformation output with the activations – channel-wise multiplication



Instantiation

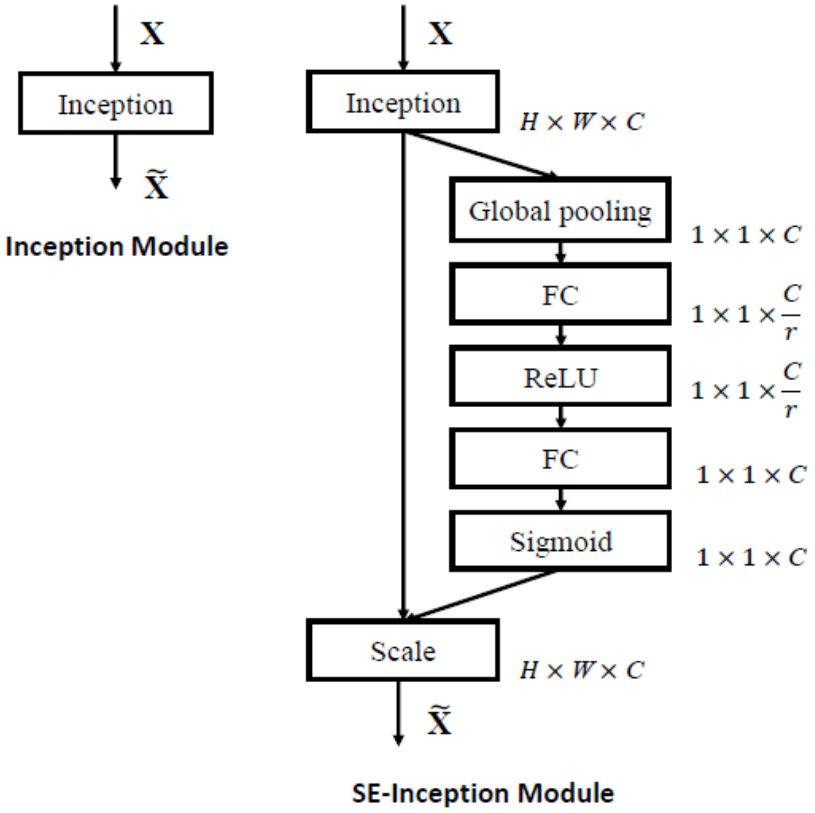


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

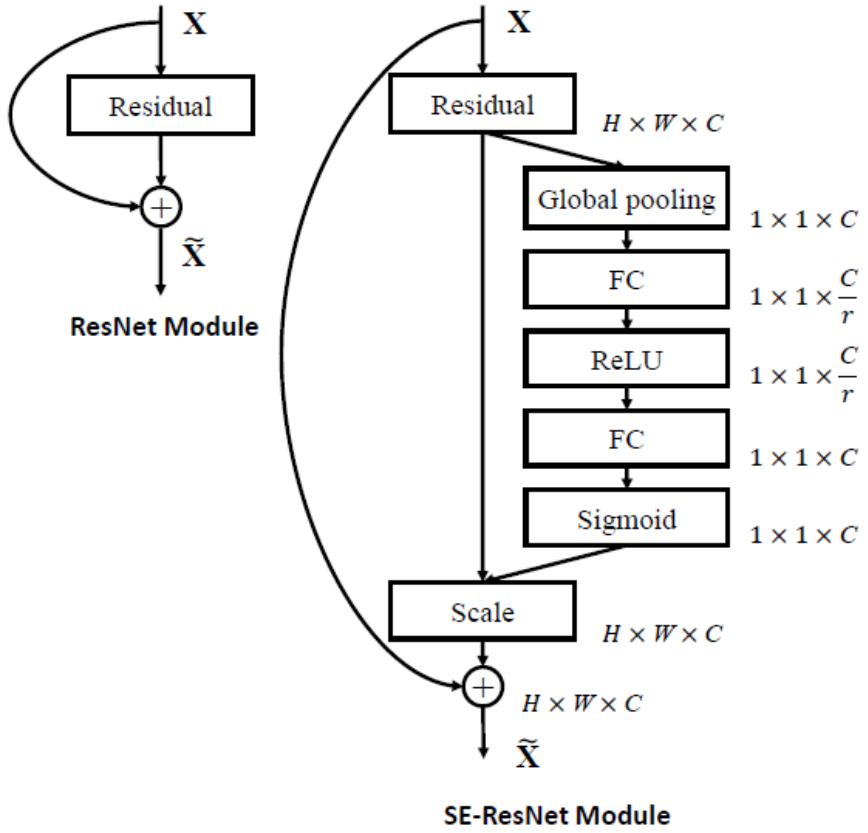


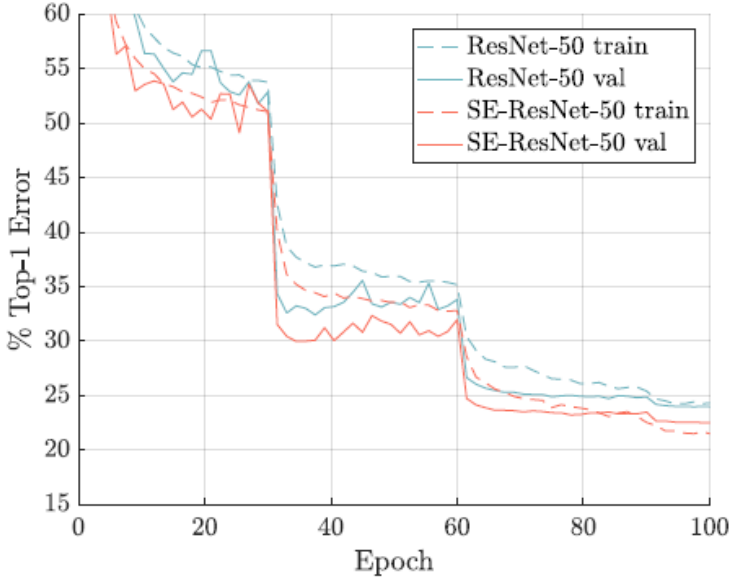
Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

Example Architecture

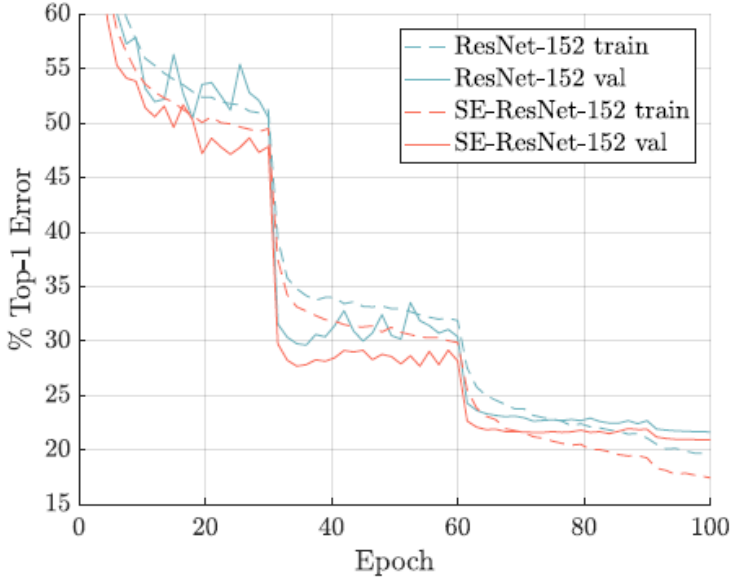
TABLE 1
(Left) ResNet-50. (Middle) SE-ResNet-50. (Right) SE-ResNeXt-50 with a 32×4d template. The shapes and operations with specific parameter settings of a residual building block are listed inside the brackets and the number of stacked blocks in a stage is presented outside. The inner brackets following by *fc* indicates the output dimension of the two fully connected layers in an SE module.

Output size	ResNet-50	SE-ResNet-50	SE-ResNeXt-50 (32 × 4d)
112 × 112	conv, 7 × 7, 64, stride 2		
56 × 56	max pool, 3 × 3, stride 2		
	$\begin{bmatrix} \text{conv}, 1\times1, 64 \\ \text{conv}, 3\times3, 64 \\ \text{conv}, 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 64 \\ \text{conv}, 3\times3, 64 \\ \text{conv}, 1\times1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 128 \\ \text{conv}, 3\times3, 128 \quad C=32 \\ \text{conv}, 1\times1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$
28 × 28	$\begin{bmatrix} \text{conv}, 1\times1, 128 \\ \text{conv}, 3\times3, 128 \\ \text{conv}, 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1\times1, 128 \\ \text{conv}, 3\times3, 128 \\ \text{conv}, 1\times1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1\times1, 256 \\ \text{conv}, 3\times3, 256 \quad C=32 \\ \text{conv}, 1\times1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$
14 × 14	$\begin{bmatrix} \text{conv}, 1\times1, 256 \\ \text{conv}, 3\times3, 256 \\ \text{conv}, 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1\times1, 256 \\ \text{conv}, 3\times3, 256 \\ \text{conv}, 1\times1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1\times1, 512 \\ \text{conv}, 3\times3, 512 \quad C=32 \\ \text{conv}, 1\times1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$
7×7	$\begin{bmatrix} \text{conv}, 1\times1, 512 \\ \text{conv}, 3\times3, 512 \\ \text{conv}, 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 512 \\ \text{conv}, 3\times3, 512 \\ \text{conv}, 1\times1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 1024 \\ \text{conv}, 3\times3, 1024 \quad C=32 \\ \text{conv}, 1\times1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$
1 × 1	global average pool, 1000-d *fc*, softmax		

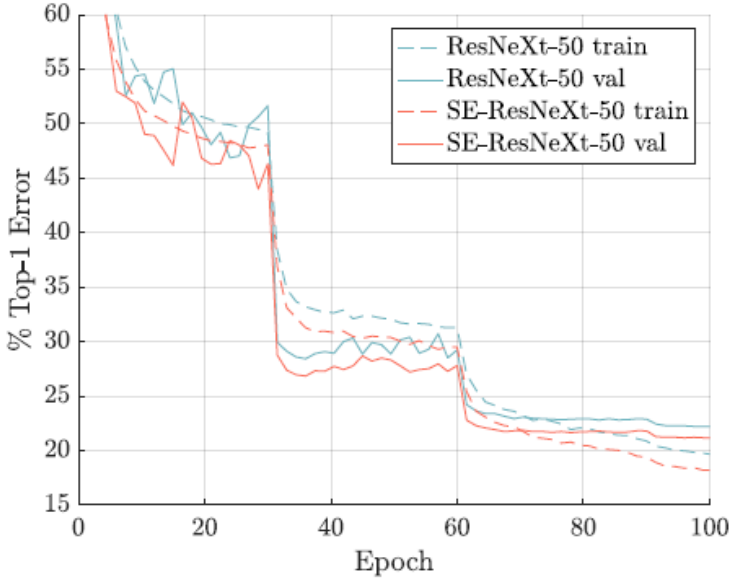
Results



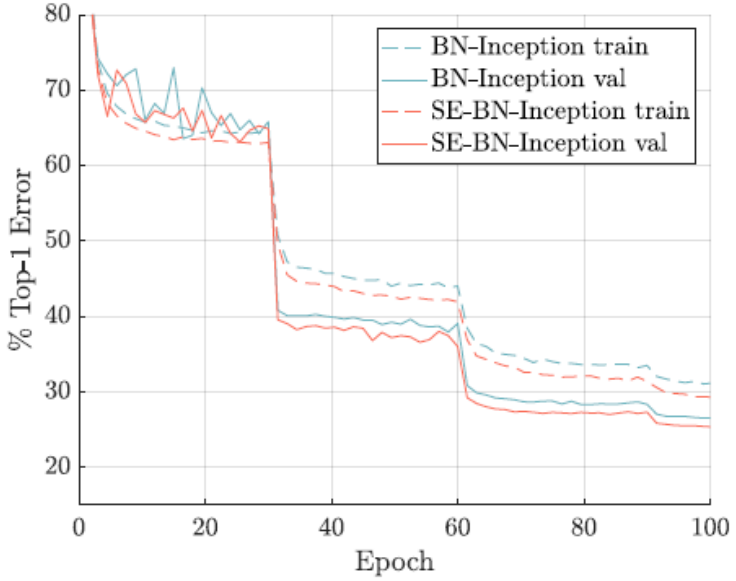
(a) ResNet-50 and SE-ResNet-50



(b) ResNet-152 and SE-ResNet-152



(c) ResNeXt-50 and SE-ResNeXt-50



(d) BN-Inception and SE-BN-Inception

NASNet

Learning Transferable Architectures for Scalable Image Recognition

Barret Zoph
Google Brain
barretzoph@google.com

Vijay Vasudevan
Google Brain
vrv@google.com

Jonathon Shlens
Google Brain
shlens@google.com

Quoc V. Le
Google Brain
qvl@google.com

Abstract

Developing neural network image classification models often requires significant architecture engineering. In this paper, we attempt to automate this engineering process by learning the model architectures directly on the dataset of interest. As this approach is expensive when the dataset is large, we propose to search for an architectural building block on a small dataset and then transfer the block to a larger dataset. Our key contribution is the design of a new search space which enables transferability. In our experiments, we search for the best convolutional layer (or "cell") on the CIFAR-10 dataset and then apply this cell to the ImageNet dataset by stacking together more copies of this

cation represents one of the most important breakthroughs in deep learning. Successive advancements on this benchmark based on convolutional neural networks (CNNs) have achieved impressive results through significant architecture engineering [52, 58, 20, 59, 57, 67].

In this paper, we consider learning the convolutional architectures directly from data with application to ImageNet classification. In addition to being an difficult and important benchmark in computer vision, features derived from ImageNet classifiers are of great importance to many other computer vision tasks. For example, features from networks that perform well on ImageNet classification provide state-of-the-art performance when transferred to other computer vision tasks where labeled data is limited [13].

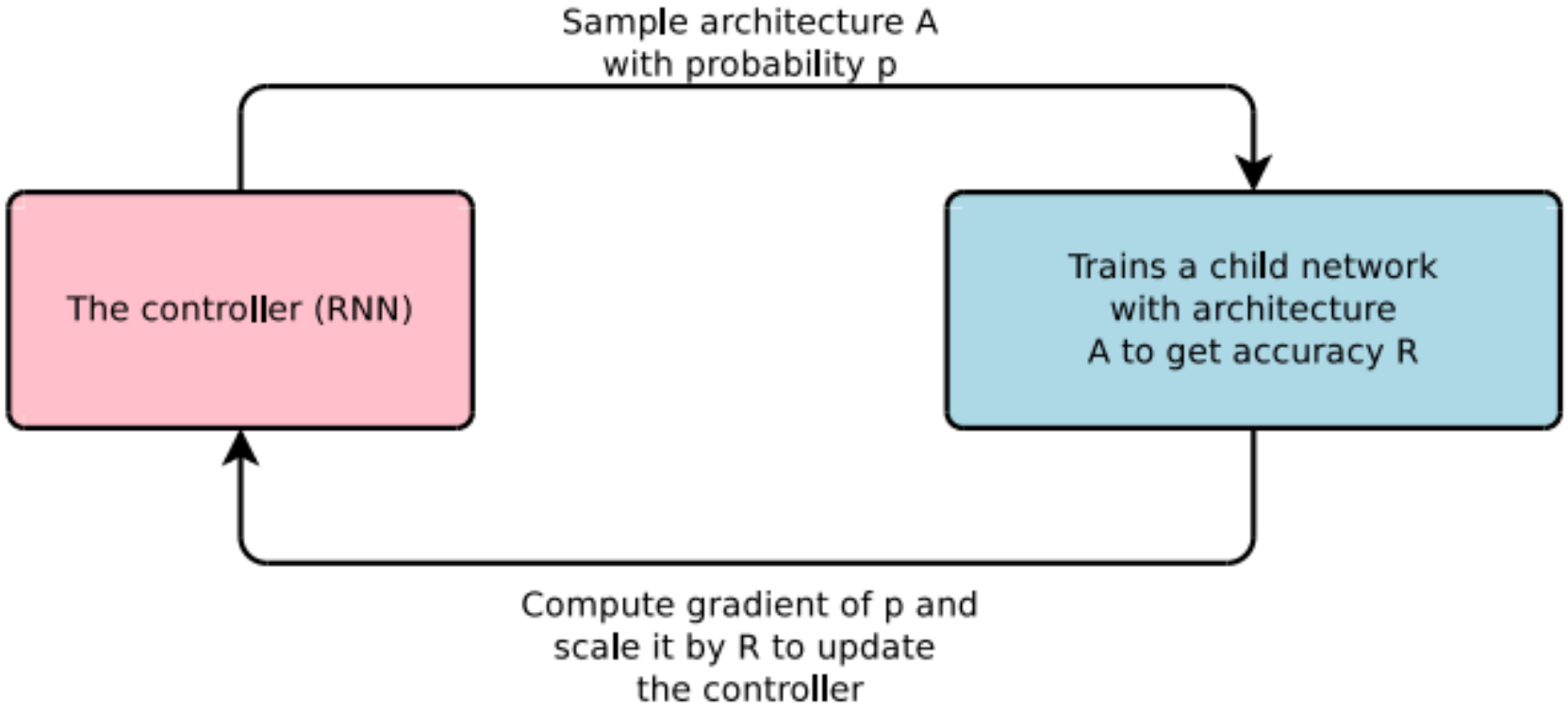
Tesorflow Pre-trained Models

- https://github.com/tensorflow/models/tree/master/research/slim

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50^	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8
ResNet V2 101^	Code	resnet_v2_101_2017_04_14.tar.gz	77.0	93.7
ResNet V2 152^	Code	resnet_v2_152_2017_04_14.tar.gz	77.8	94.1
ResNet V2 200	Code	TBA	79.9*	95.2*
VGG 16	Code	vgg_16_2016_08_28.tar.gz	71.5	89.8
VGG 19	Code	vgg_19_2016_08_28.tar.gz	71.1	89.8
MobileNet_v1_1.0_224	Code	mobilenet_v1_1.0_224_2017_06_14.tar.gz	70.7	89.5
MobileNet_v1_0.50_160	Code	mobilenet_v1_0.50_160_2017_06_14.tar.gz	59.9	82.5
MobileNet_v1_0.25_128	Code	mobilenet_v1_0.25_128_2017_06_14.tar.gz	41.3	66.2
NASNet-A_Mobile_224#	Code	nasnet-a_mobile_04_10_2017.tar.gz	74.0	91.6
NASNet-A_Large_331#	Code	nasnet-a_large_04_10_2017.tar.gz	82.7	96.2

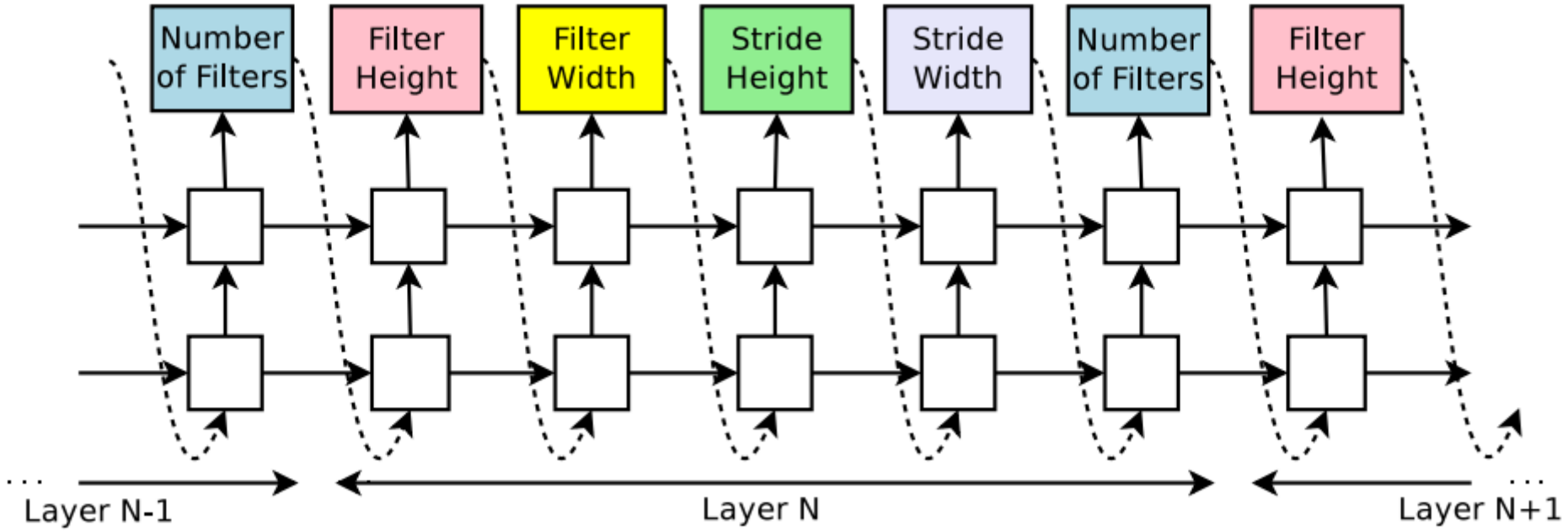
Neural Architecture Search

- B. Zoph and Q. V. Le., "Neural architecture search with reinforcement learning", ICLR-2017



Neural Architecture Search

- How controller RNN samples a simple convolutional network



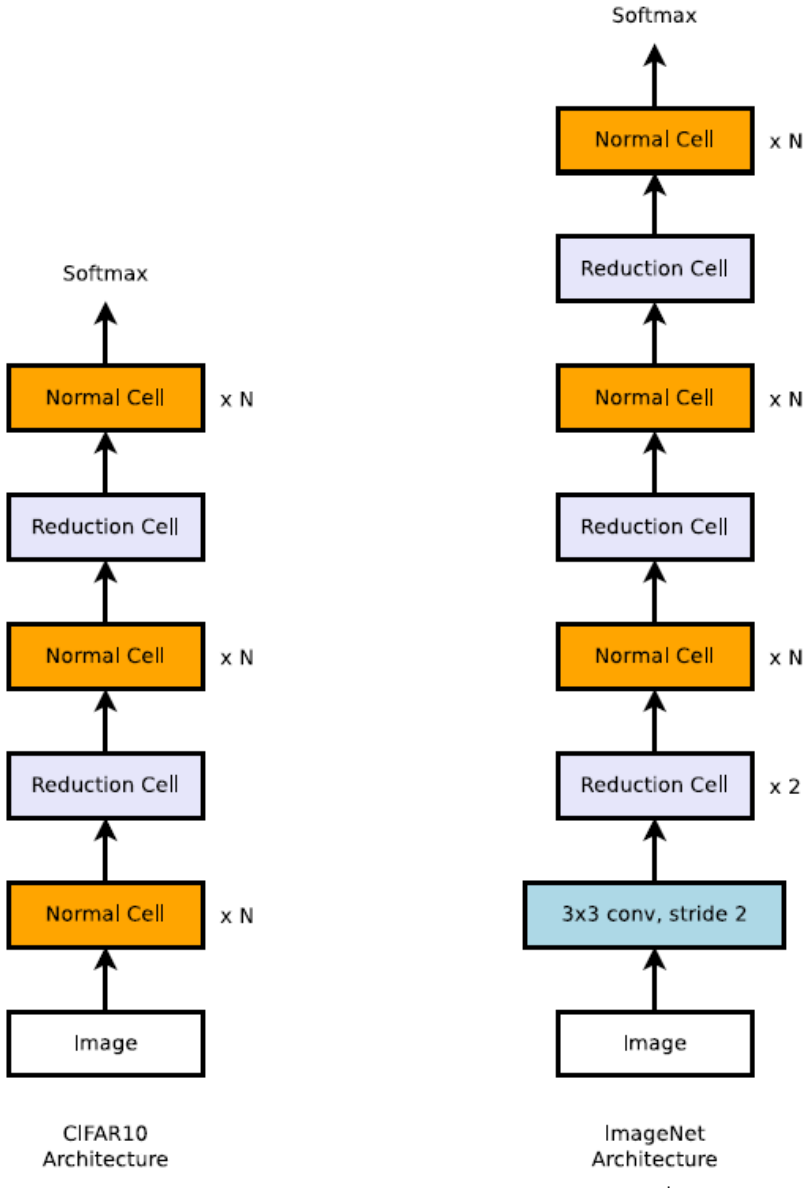
Motivation & Idea

- NAS used 800 GPUs for 28days resulting in 22,400 GPU-hours → too long (ImageNet dataset)
- Many state-of-the-art models have repeated modules
- Searching for a good architecture on the far smaller CIFAR-10 dataset, and automatically transfer the learned architecture to ImageNet
- Achieving this transferability by designing a search space so that the complexity of the architecture is independent of the depth of the network
- All convolutional networks in search space are composed of convolutional layers(or "cells") with identical structure but different weights

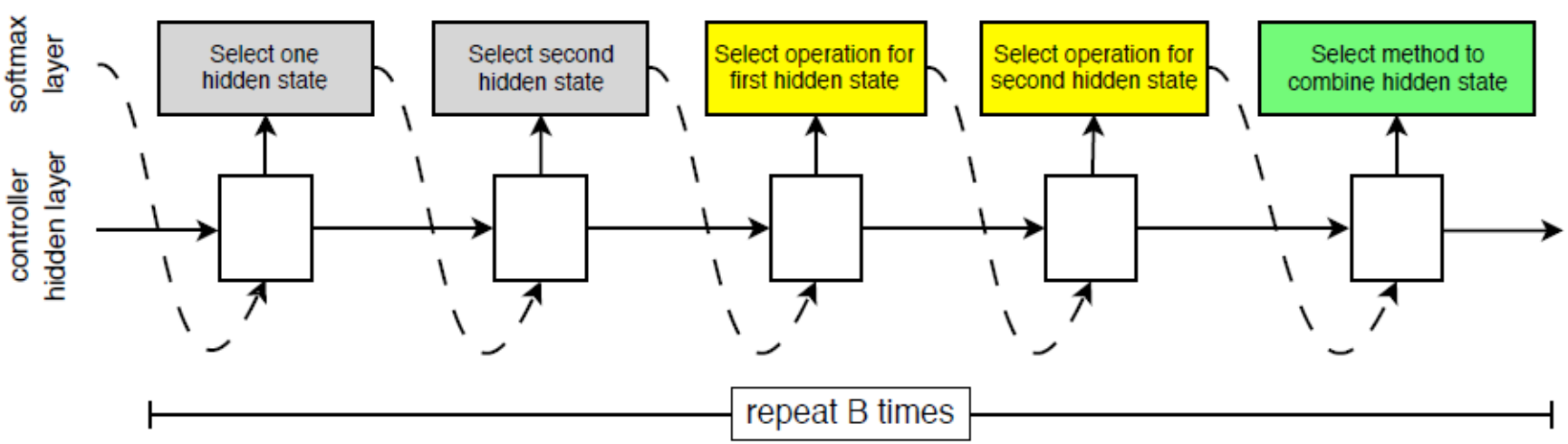
Method

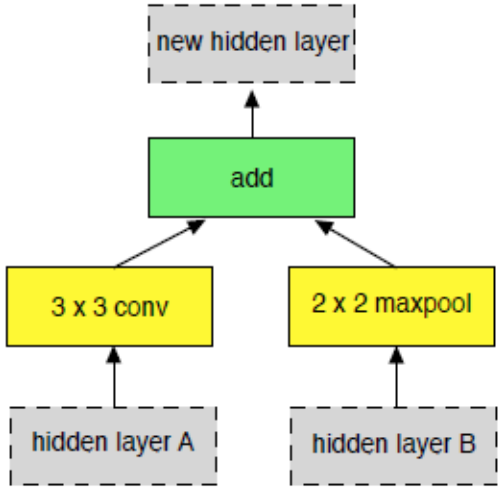
- Overall architectures of the convolutional nets are manually predetermined
  - Normal Cell – convolutional cells that return a feature map of the same dimension
  - Reduction Cell – convolutional cells that return a feature map where the feature map height and width is reduced by a factor of two
- Using common heuristic to double the number of filters in the output whenever the spatial activation size is reduced

Scalable Architectures for Image Classification



Models and Algorithms





Step 1. Select a hidden state from h_i, h_{i-1} or from the set of hidden states created in previous blocks.

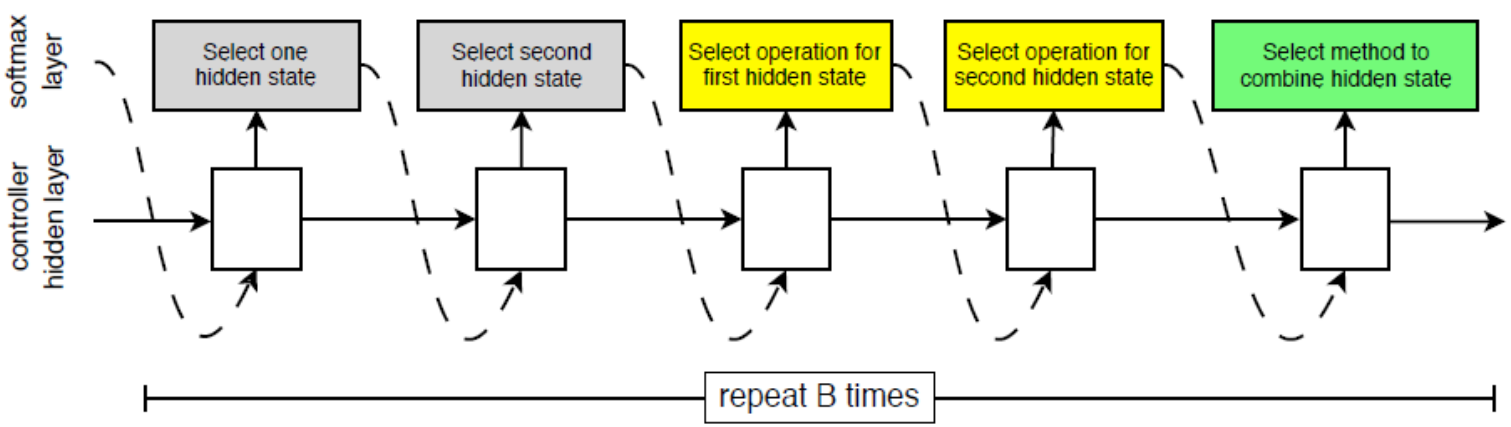
Step 2. Select a second hidden state from the same options as in Step 1.

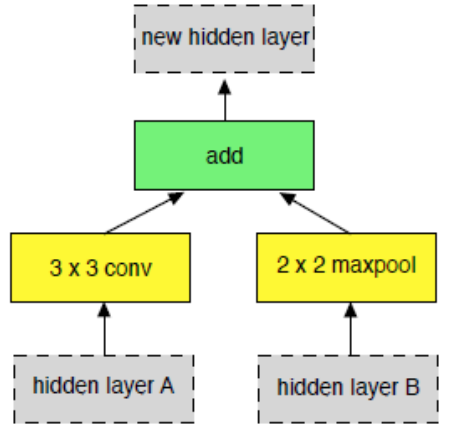
Step 3. Select an operation to apply to the hidden state selected in Step 1.

Step 4. Select an operation to apply to the hidden state selected in Step 2.

Step 5. Select a method to combine the outputs of Step 3 and 4 to create a new hidden state.

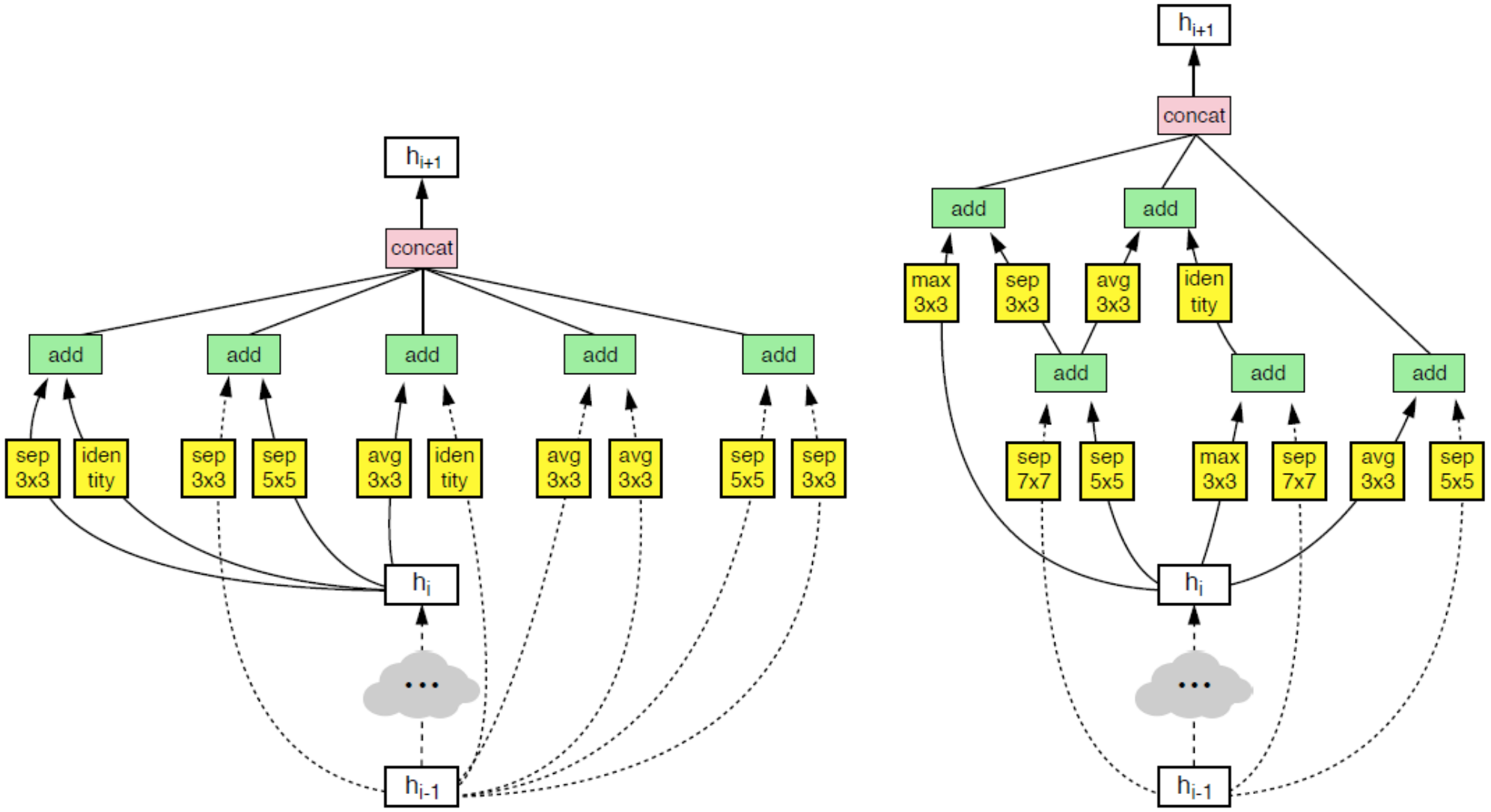
Search Space in a Cell (Step 3 and 4)





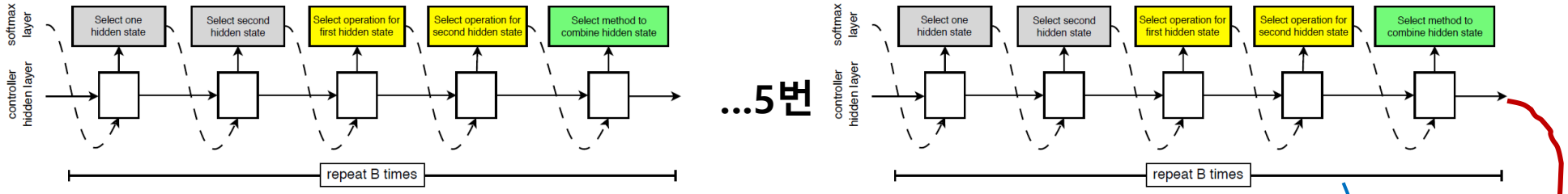
- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

Best Architecture (NASNet-A)



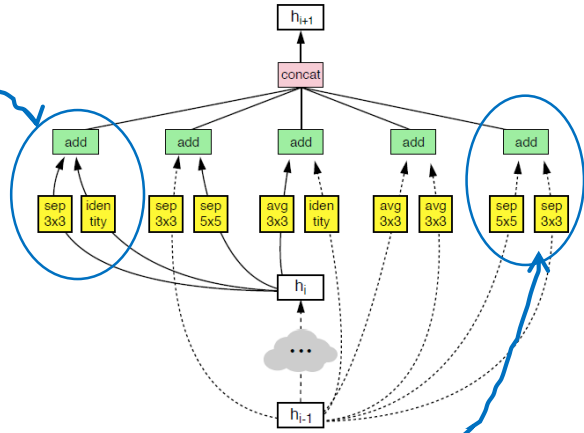
Normal Cell *Reduction Cell*

Best Architecture (NASNet-A)

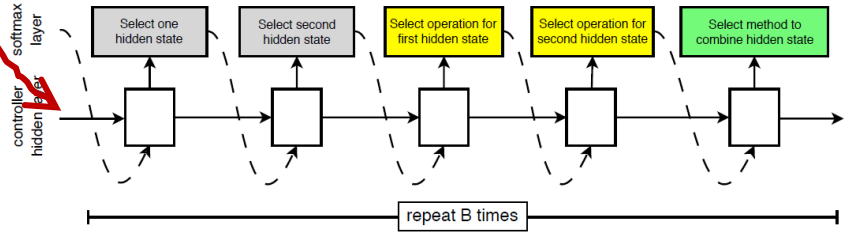


...5번

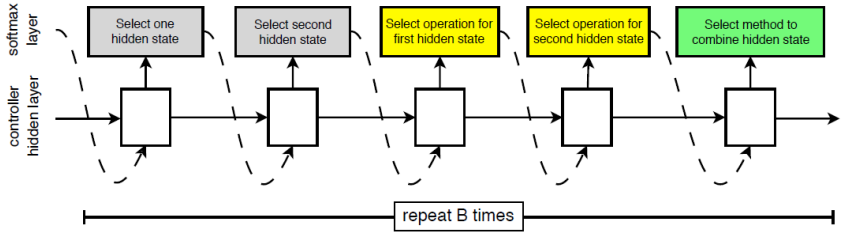
<Normal Cell>



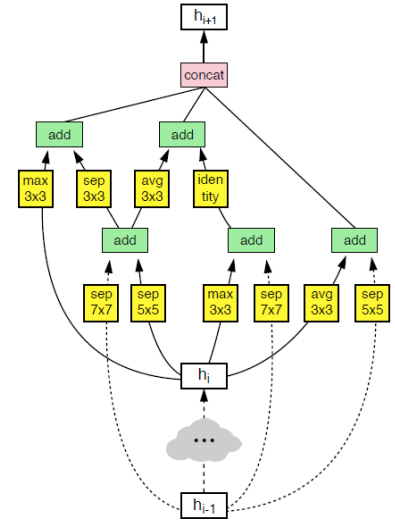
Normal Cell



...5번



<Reduction Cell>



Reduction Cell

Results on CIFAR-10

model	depth	# params	error rate (%)
DenseNet ($L = 40, k = 12$) [26]	40	1.0M	5.24
DenseNet($L = 100, k = 12$) [26]	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) [26]	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) [26]	190	25.6M	3.46
Shake-Shake 26 2x32d [18]	26	2.9M	3.55
Shake-Shake 26 2x96d [18]	26	26.2M	2.86
Shake-Shake 26 2x96d + cutout [12]	26	26.2M	2.56
NAS v3 [70]	39	7.1M	4.47
NAS v3 [70]	39	37.4M	3.65
NASNet-A (6 @ 768)	-	3.3M	3.41
NASNet-A (6 @ 768) + cutout	-	3.3M	2.65
NASNet-A (7 @ 2304)	-	27.6M	2.97
NASNet-A (7 @ 2304) + cutout	-	27.6M	2.40
NASNet-B (4 @ 1152)	-	2.6M	3.73
NASNet-C (4 @ 640)	-	3.1M	3.59

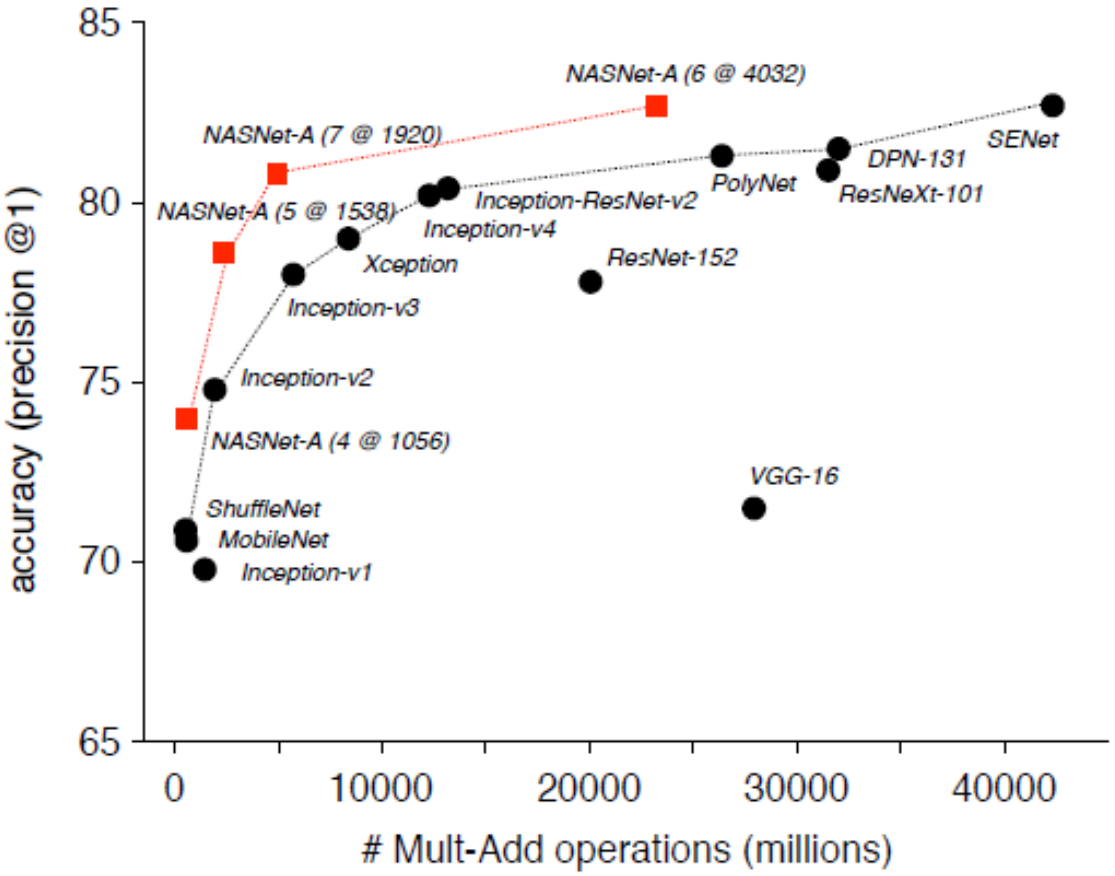
Results on ImageNet (vs SOTA)

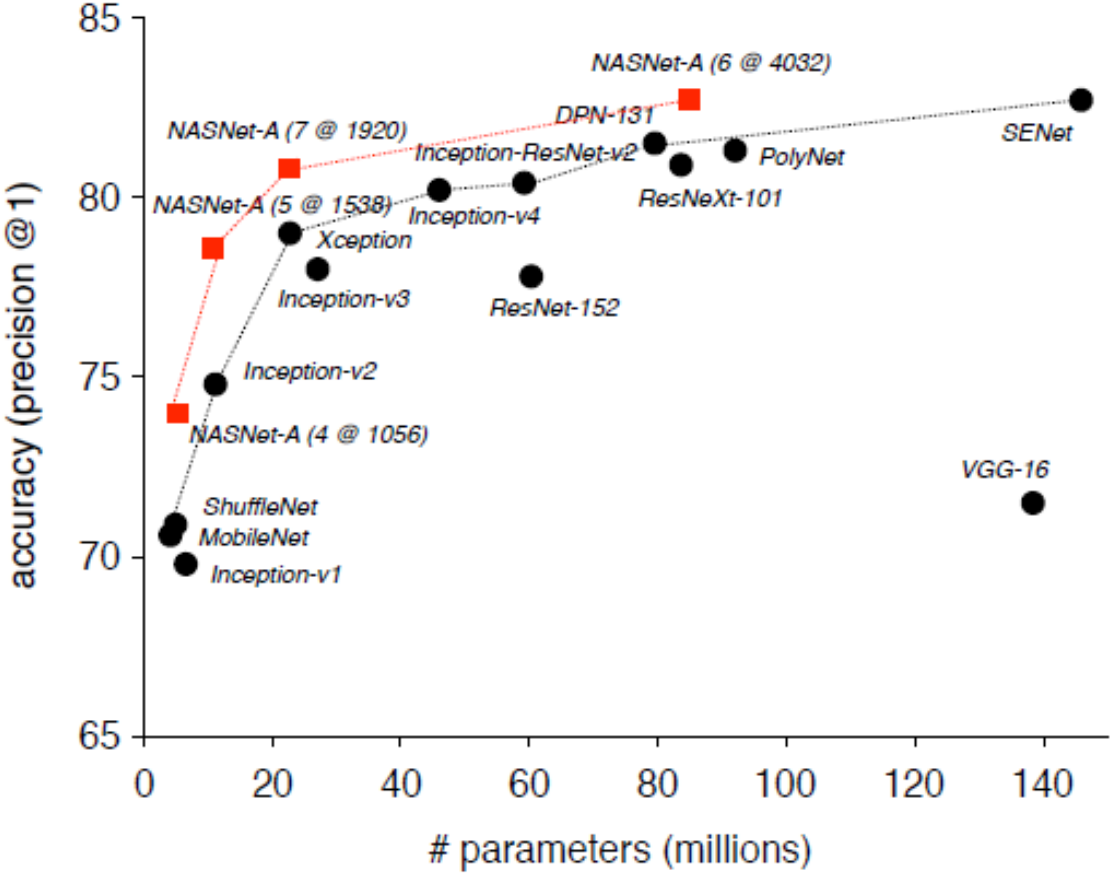
Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [59]	299×299	23.8 M	5.72 B	78.0	93.9
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [57]	299×299	55.8 M	13.2 B	80.4	95.3
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [67]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [68]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

Results on ImageNet in a Resource-Constrained Setting

Model	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V1 [58]	6.6M	1,448 M	69.8	89.9
MobileNet-224 [24]	4.2 M	569 M	70.6	89.5
ShuffleNet (2x) [69]	~ 5M	524 M	70.9	89.8
NASNet-A (4 @ 1056)	**5.3 M**	**564 M**	**74.0**	**91.6**
NASNet-B (4 @ 1536)	5.3M	488 M	72.8	91.3
NASNet-C (3 @ 960)	4.9M	558 M	72.5	91.0

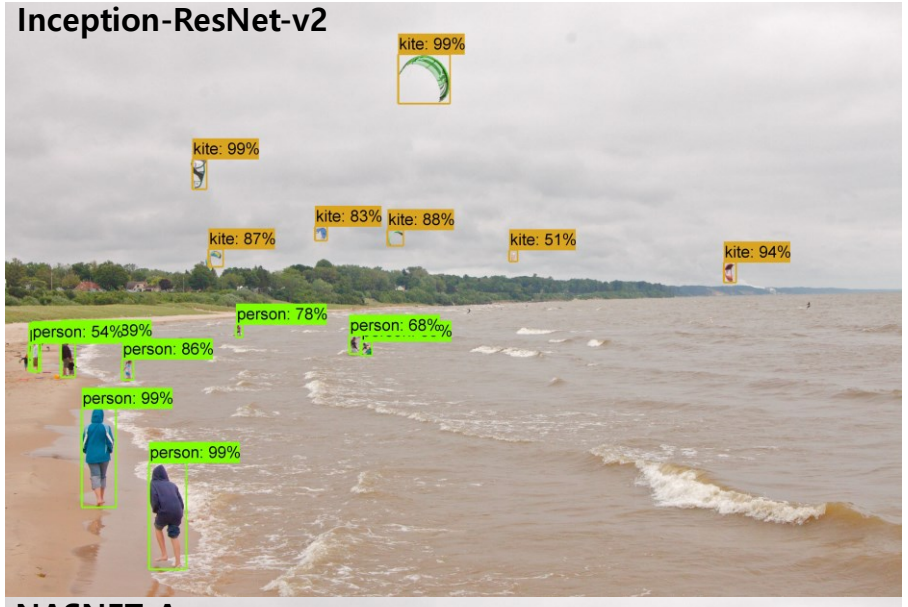
Results on ImageNet

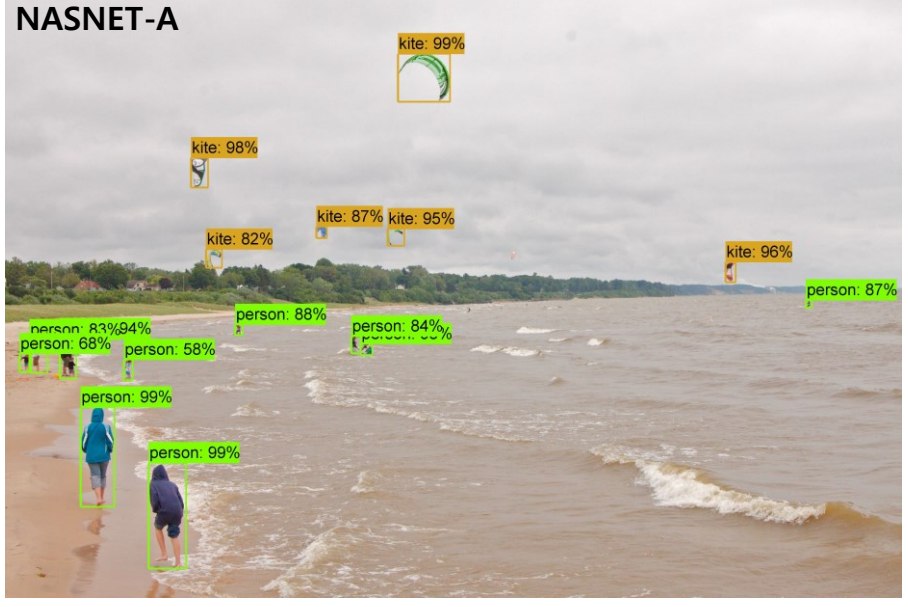




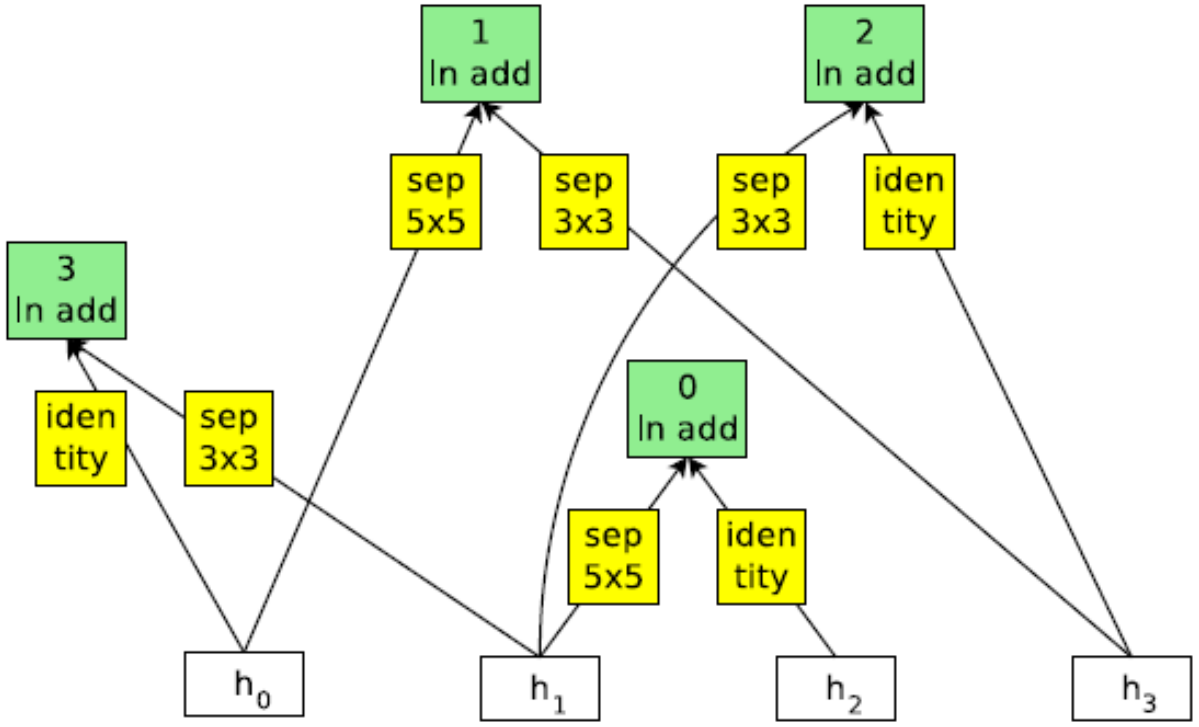
Object Detection Performance on COCO

Model	resolution	mAP (mini-val)	mAP (test-dev)
MobileNet-224 [24]	600 × 600	19.8%	-
ShuffleNet (2x) [69]	600 × 600	24.5%†	-
NASNet-A (4 @ 1056)	600 × 600	**29.6%**	-
ResNet-101-FPN [35]	800 (short side)	-	36.2%
Inception-ResNet-v2 (G-RMI) [28]	600 × 600	35.7%	35.6%
Inception-ResNet-v2 (TDM) [51]	600 × 1000	37.3%	36.8%
NASNet-A (6 @ 4032)	800 × 800	41.3%	40.7%
NASNet-A (6 @ 4032)	1200 × 1200	**43.2%**	**43.1%**
ResNet-101-FPN (RetinaNet) [36]	800 (short side)	-	39.1%

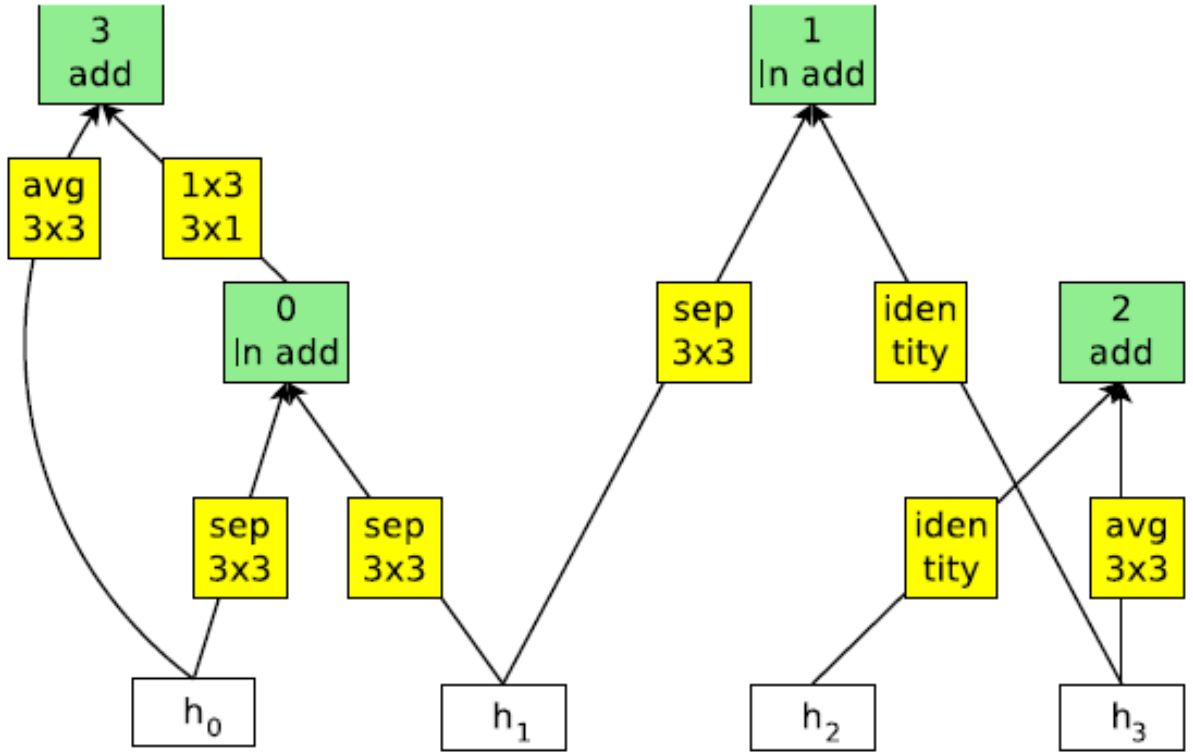




NASNet-B

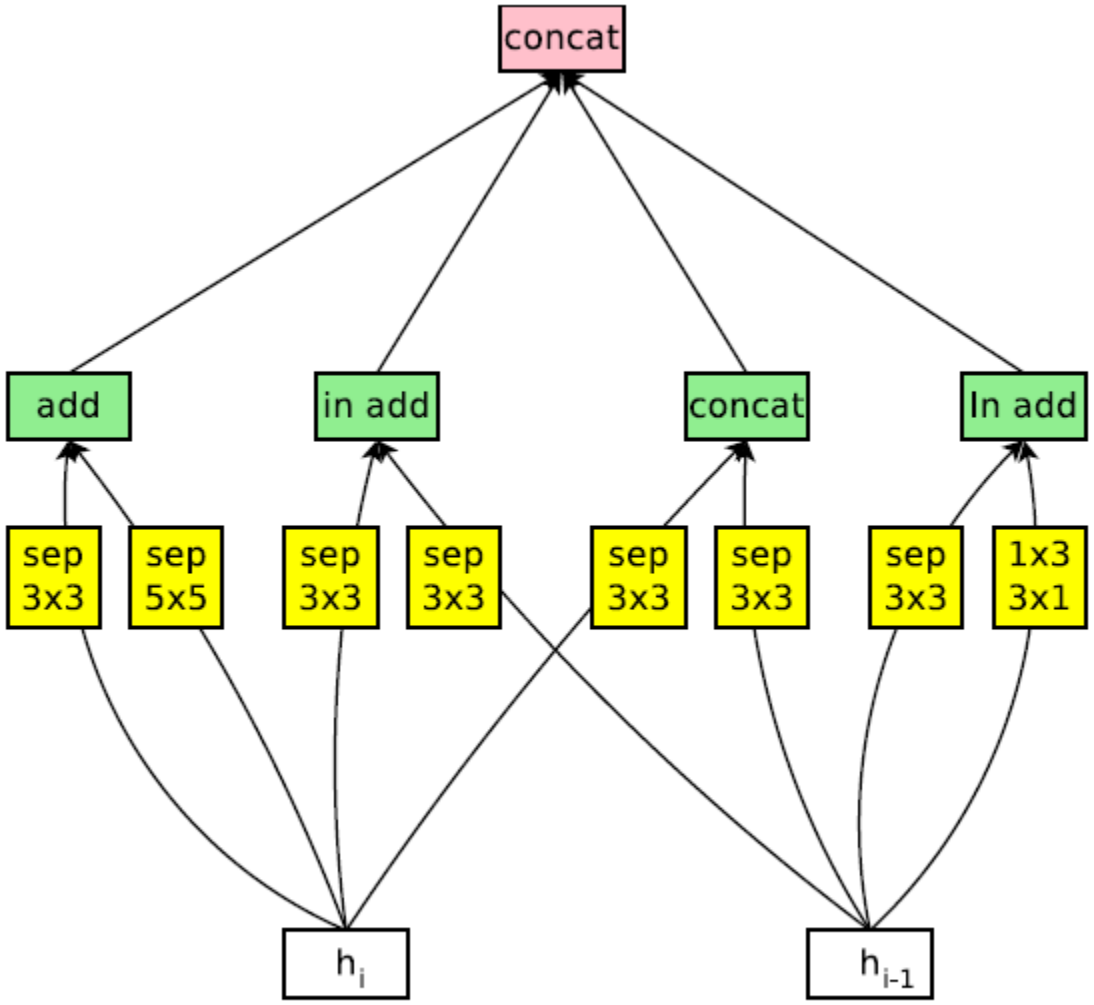


Normal Cell

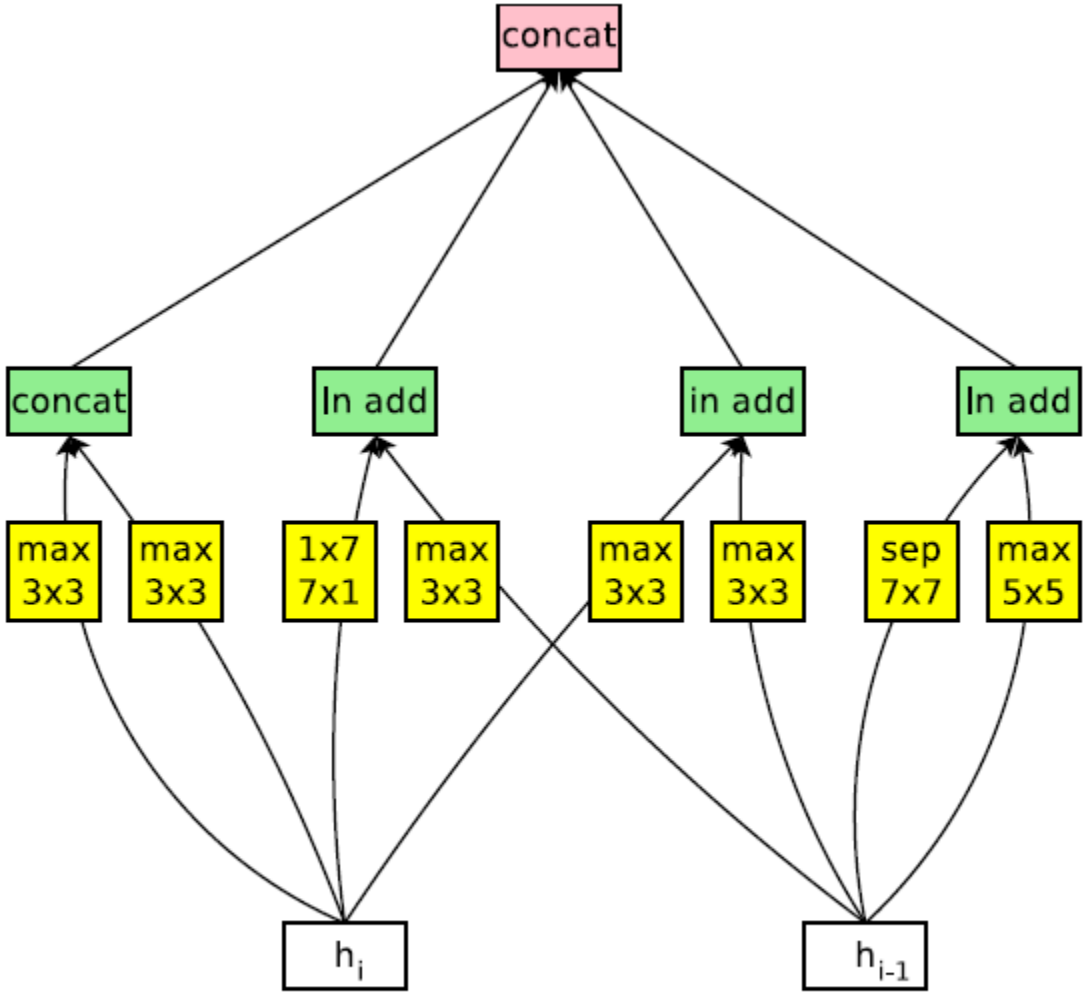


Reduction Cell

NASNet-C



Normal Cell



Reduction Cell