Best paper award CVPR 2017

DENSELY CONNECTED CONVOLUTIONAL NETWORKS

Cornell University

Tsinghua University

Facebook AI Research

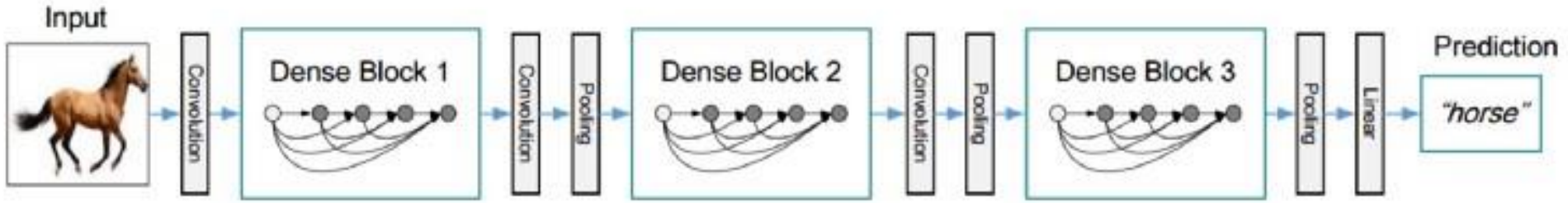


Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.

Introduction

- Originally introduced over 25 years ago

  Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

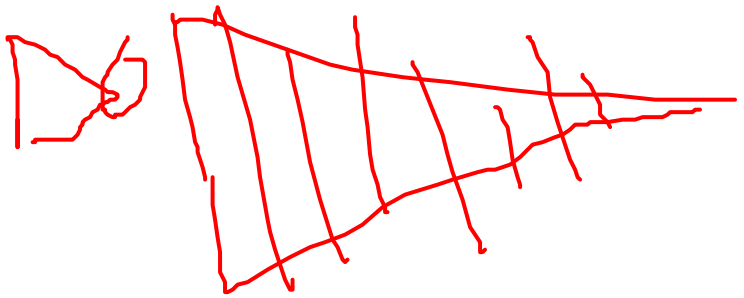
- Improvements in computer hardware and network structure have enabled the training of truly deep CNNs ← dominant ML approach
- LeNet – 5 layers
- VGG – 19 layers
- Highway network cross 100 layers

  https://arxiv.org/pdf/1505.00387.pdf

- ResNet
- FractalNets

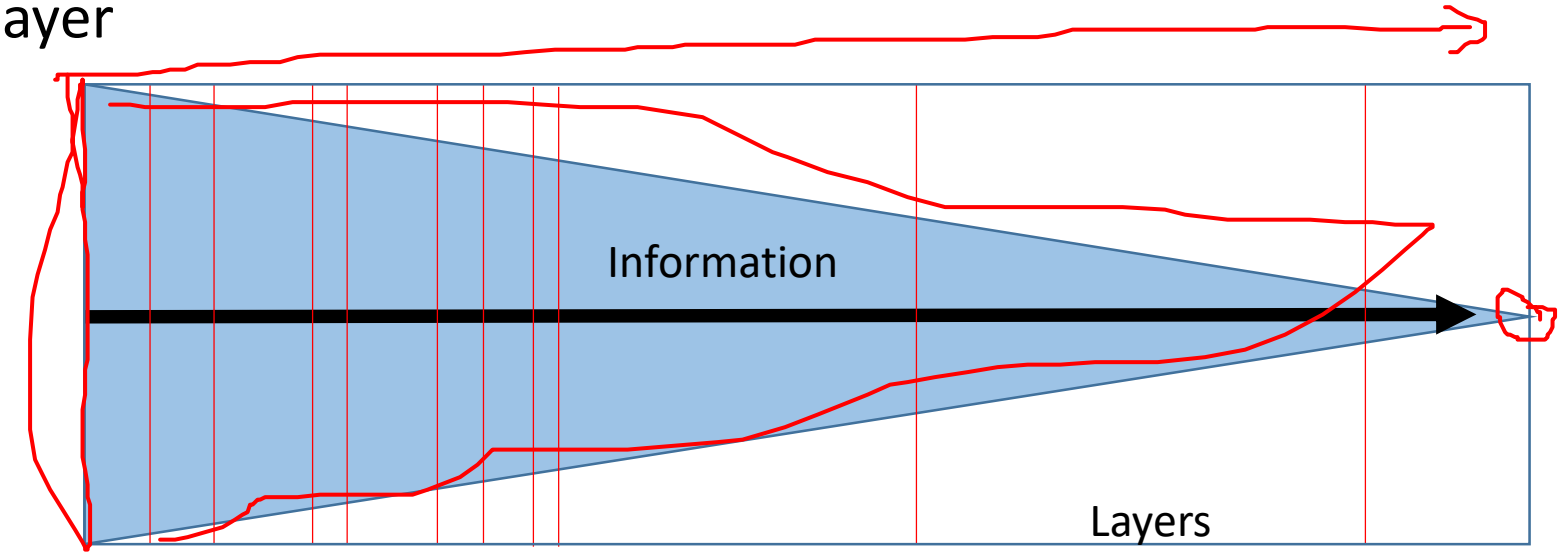
G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648, 2016.

Introduction

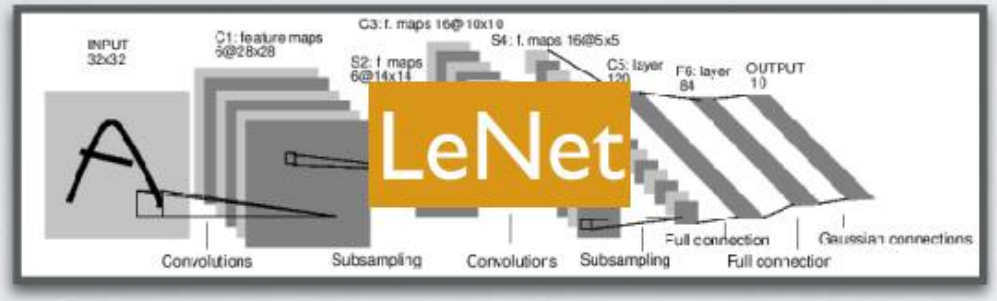


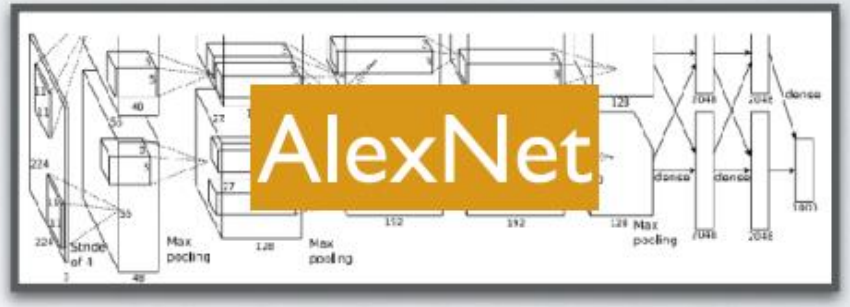
- Highway and ResNet Network
  - Bypass signal from one layer to the next *via* identity connections

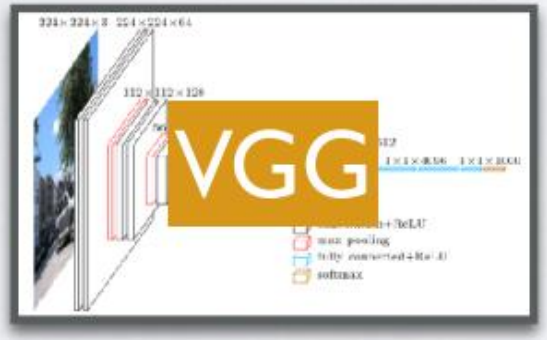
- CNN ← deep and ← deeper ← emerges new problem
  - The input information passes through and can vanish or simply washed out till it reaches the last layer

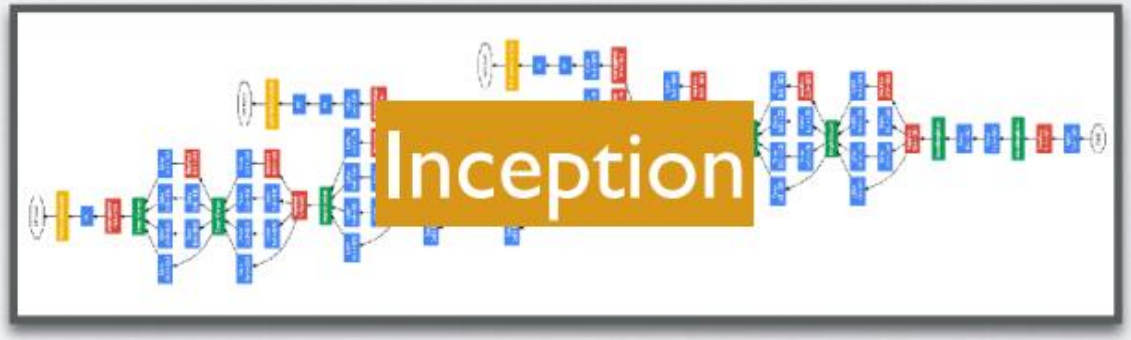


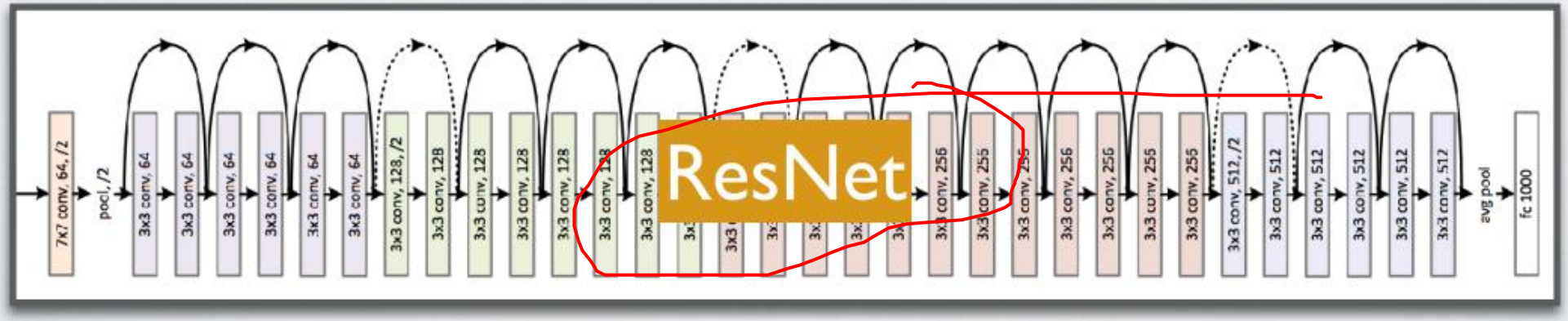
CONVOLUTIONAL NETWORKS











Convolutional Network

- Deep Residual Learning for Image Recognition — **ResNet** (Microsoft Research)
- Wide Residual Networks (Université Paris-Est, École des Ponts ParisTech)
- Aggregated Residual Transformations for Deep Neural Networks — **ResNeXt** (Facebook AI Research)

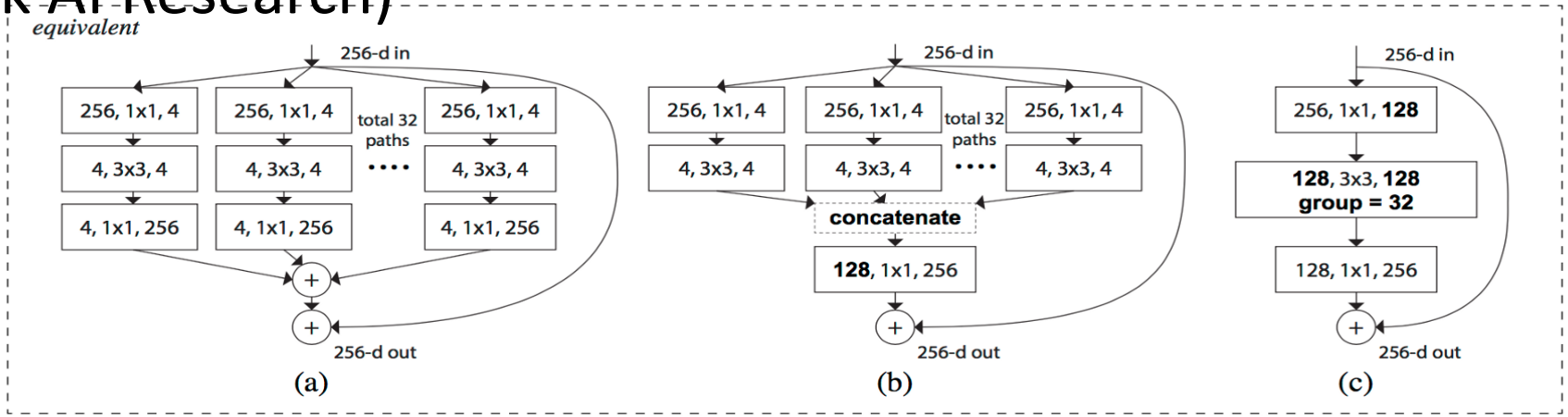
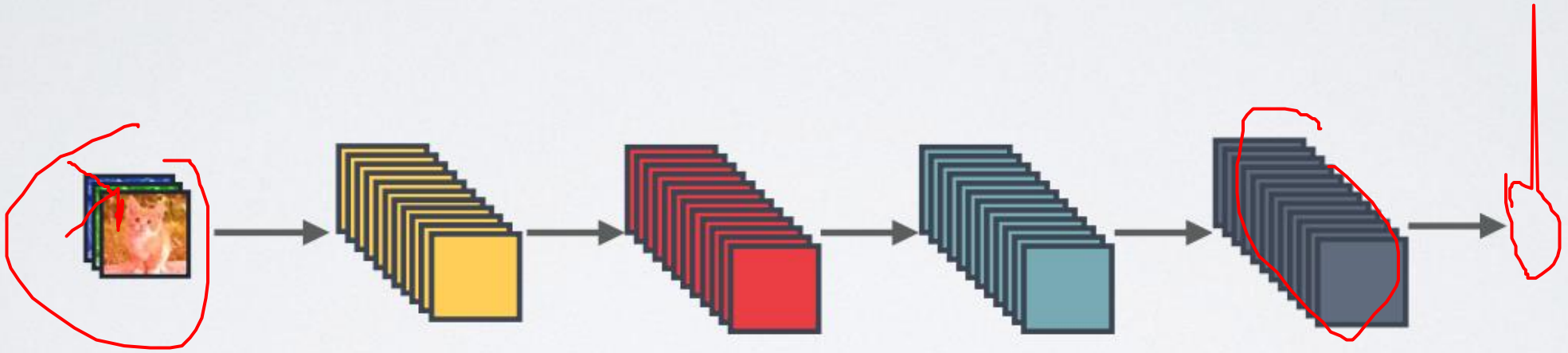
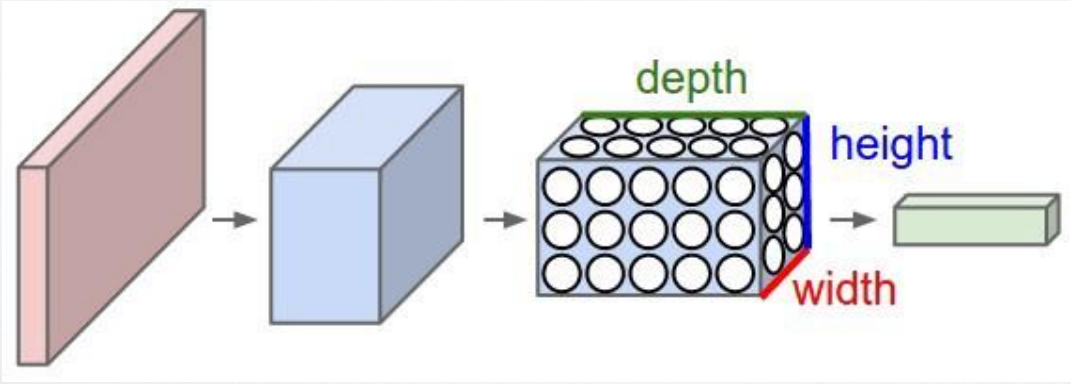


Figure 3. Equivalent building blocks of ResNeXt. **(a)**: Aggregated residual transformations, the same as Fig. 1 right. **(b)**: A block equivalent to (a), implemented as early concatenation. **(c)**: A block equivalent to (a,b), implemented as grouped convolutions [24]. Notations in **bold** text highlight the reformulation changes. A layer is denoted as (# input channels, filter size, # output channels).

STANDARD CONNECTIVITY

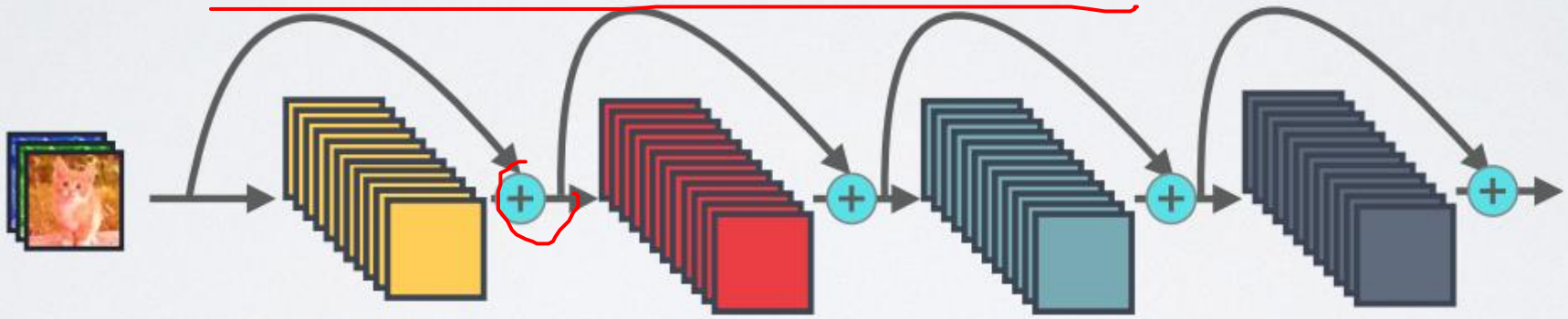




RESNET CONNECTIVITY

Identity mappings promote gradient propagation.

ResNet architecture has a fundamental building block (Identity) where you merge (additive) a previous layer into a future layer.



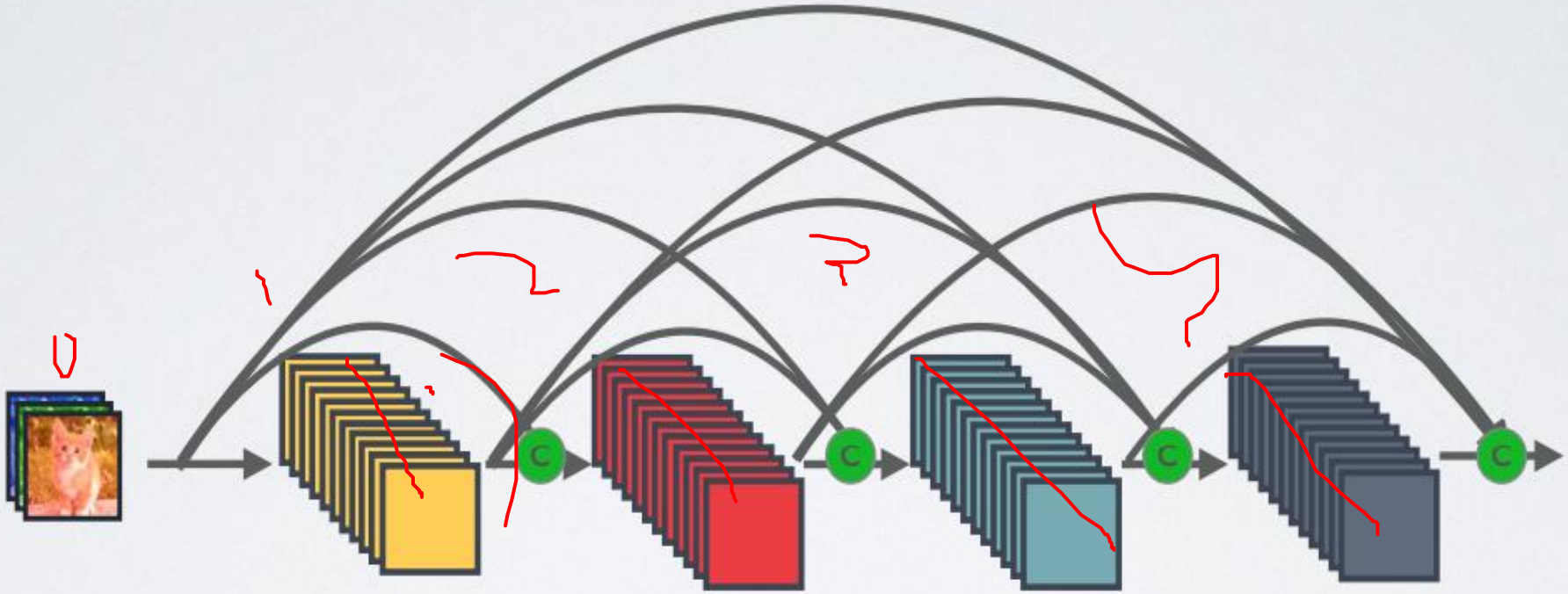
⊕ : Element-wise addition

Deep residual learning for image recognition: [He, Zhang, Ren, Sun] (CVPR 2015)

DenseNets are simpler and more efficient ResNet.

DENSE CONNECTIVITY

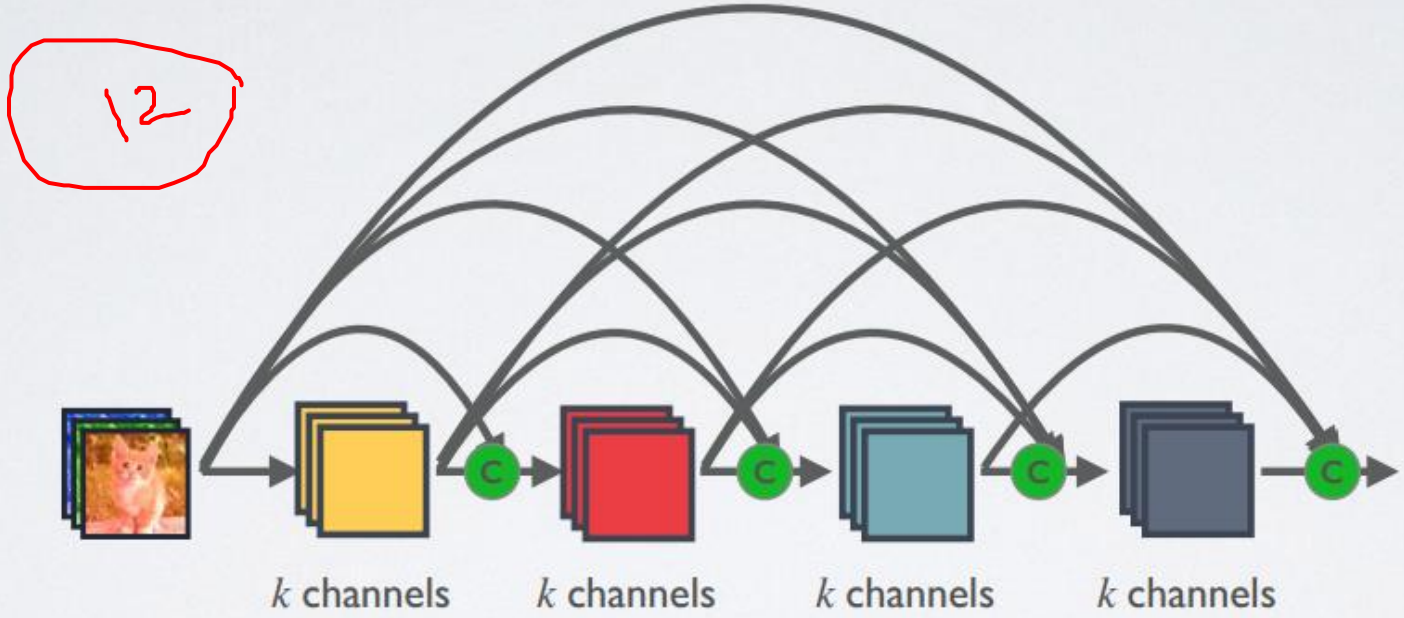
DenseNet proposes concatenating outputs from the previous layers in a feed-forward fashion instead of using the summation.



C : Channel-wise concatenation

to ensure maximum information flow between layers in the network

DENSE AND SLIM



k : Growth Rate

Connect all layers (with matching feature-map sizes) directly with each other.

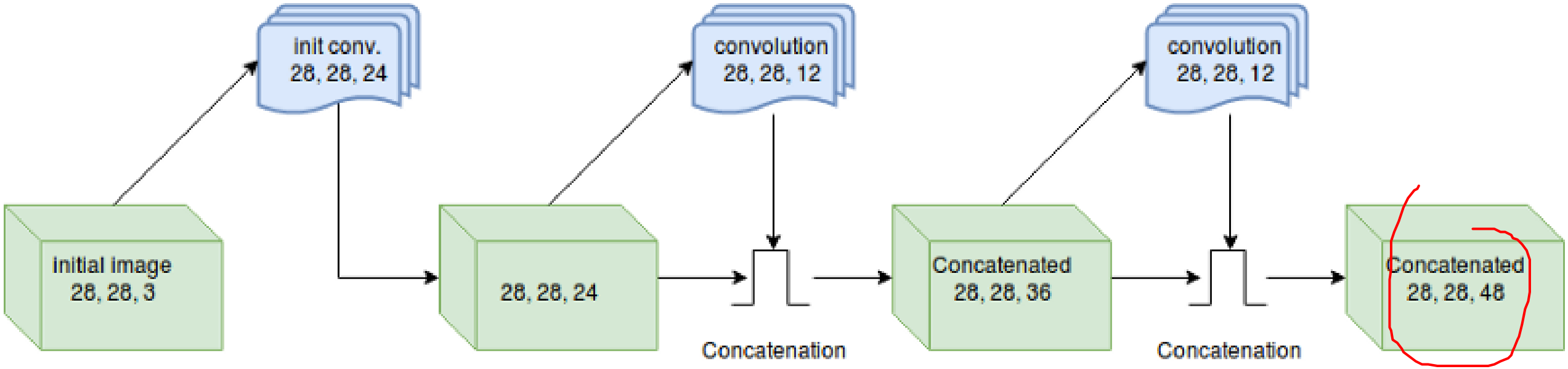
DenseNet

- Ex. Let's imagine we have an image with shape (28, 28, 3).
- First, we spread image to initial 24 channels and receive the image (28, 28, 24).
- Every next convolution layer will generate k=12 features, and remain width and height the same.
- The output from L_i layer will be (28, 28, 12).

- But input to the L_{i+1} will be (28, 28, 24+12), for L_{i+2} (28, 28, 24 + 12 + 12) and so on.

DenseNet

After a while, we receive the image with same width and height, but with plenty of features (28, 28, 48).

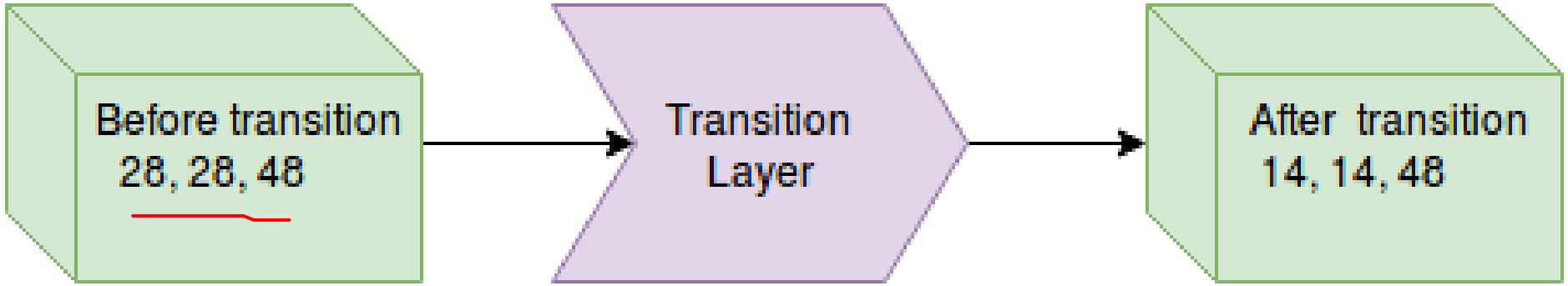


Dense block is where all preceding layers are connected to the current layer

All these N layers are named Block in the paper. There's also batch normalization, nonlinearity and dropout inside the block.

DenseNet

- To reduce the size, DenseNet uses transition layers.
- These layers contain convolution with kernel size = 1 followed by 2x2 average pooling with stride = 2.



- Now we can again pass the image through the block with N convolutions.

DenseNet

- As a direct consequence of the input concatenation, the feature maps learned by any of the DenseNet layers can be accessed by all subsequent layers. This encourages feature reuse throughout the network, and leads to more compact models.

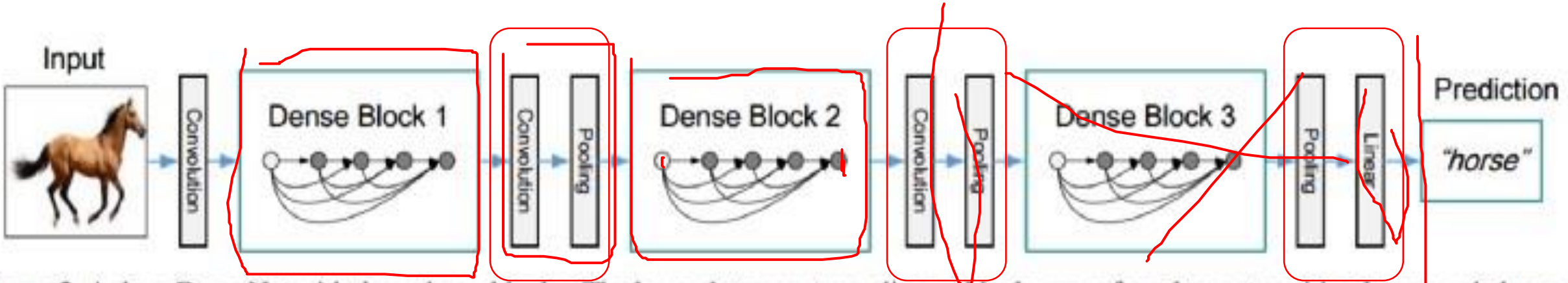
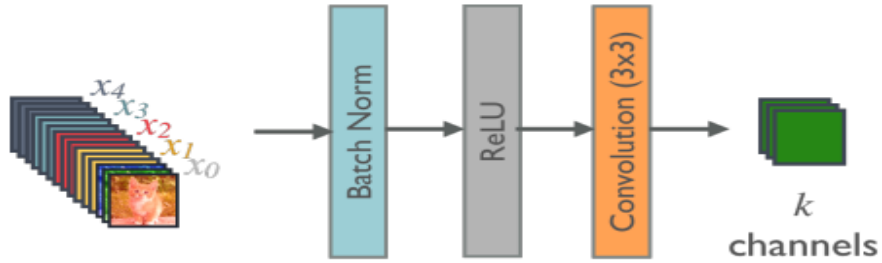


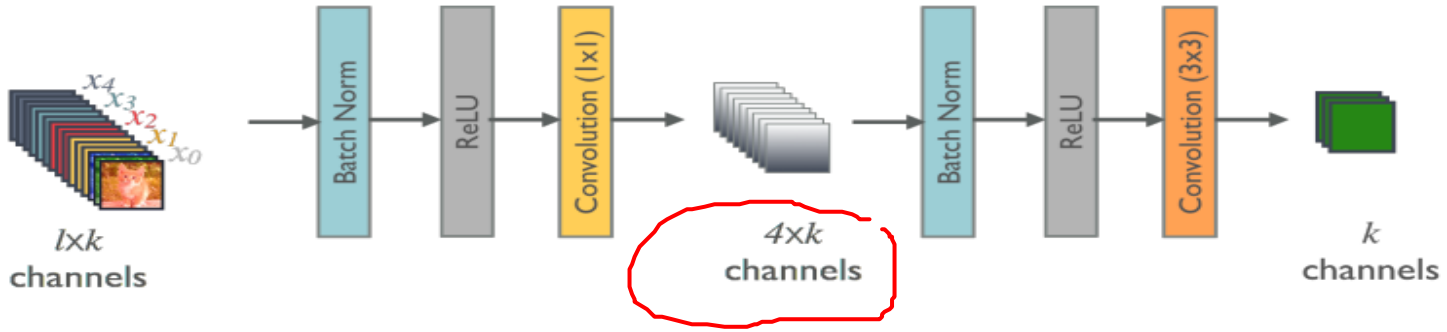
Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.

Composite Layer

- Each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision.



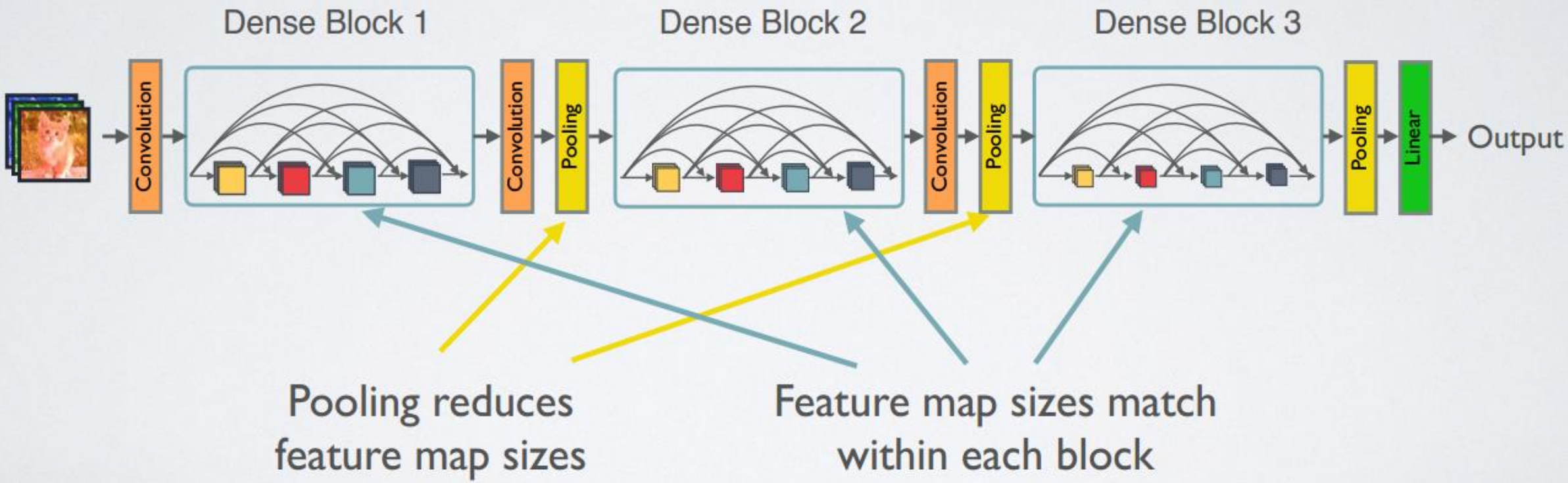
Basic Structure



Bottleneck Structure for higher efficiency

Composite Layer

This introduces L(L+1)/2 connections in an L-layer network, instead of just L, as in traditional architectures.

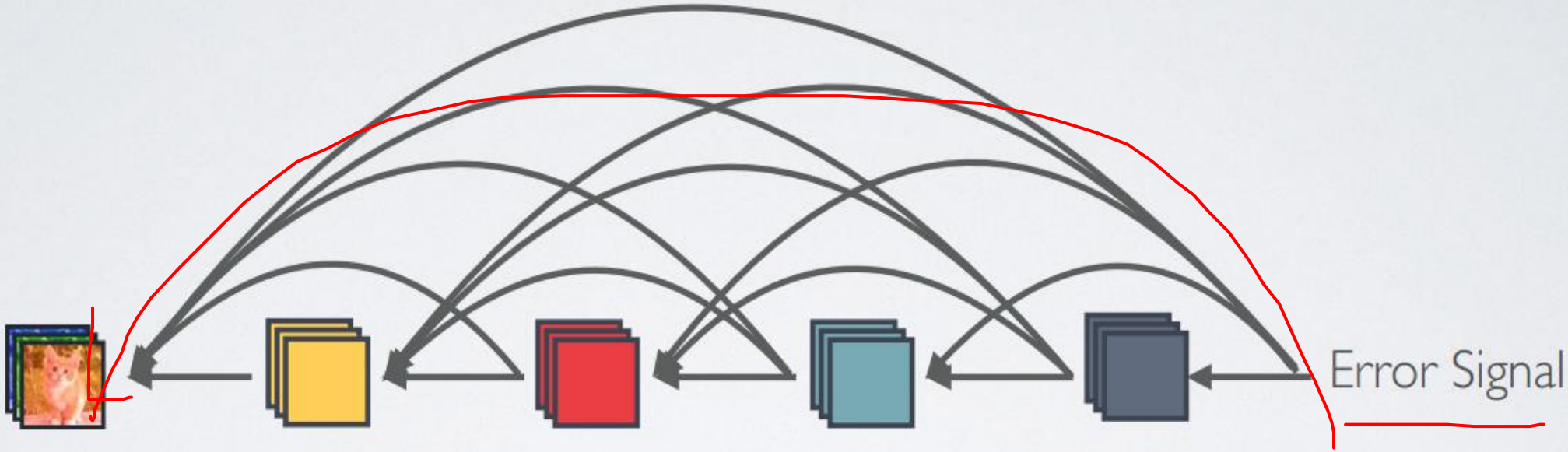


DenseNet Architecture

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

ADVANTAGE 1: STRONG GRADIENT FLOW

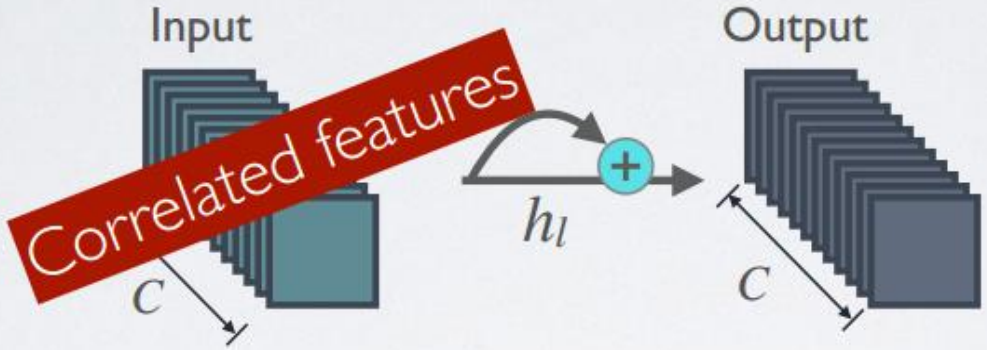


Implicit "deep supervision"

Deeply supervised Net: [Lee, Xie, Gallagher, Zhang, Tu] (2015)

ADVANTAGE 2: PARAMETER & COMPUTATIONAL EFFICIENCY

ResNet connectivity:



DenseNet connectivity:

#parameters:

$O(CxC)$

$k<<C$

$O(lxkxk)$

k: Growth rate

Memory-Efficient Implementation of DenseNets

Geoff Pleiss* Cornell University geoff@cs.cornell.edu

Danlu Chen* Fudan University dc688@cornell

Gao Huang, Tongcheng Li Cornell University

Laurens van der Maaten Facebook AI Research lvdmaaten@fb.com

Abstr

The DenseNet architecture [9] is highly c
feature reuse. However, a naïve DenseNet i
amount of GPU memory: If not properly m
tion [7] and contiguous convolution operati
quadratically with network depth. In this t
to reduce the memory consumption of Den
using shared memory allocations, we redu
maps from *quadratic* to *linear*. Without t
possible to train extremely deep DenseNet
be trained on a single GPU, up from $4M$.
ters), which previously would have been in
a single workstation with 8 NVIDIA Tesla
classification dataset, this large DenseNe
top-1 error of 20.26%.

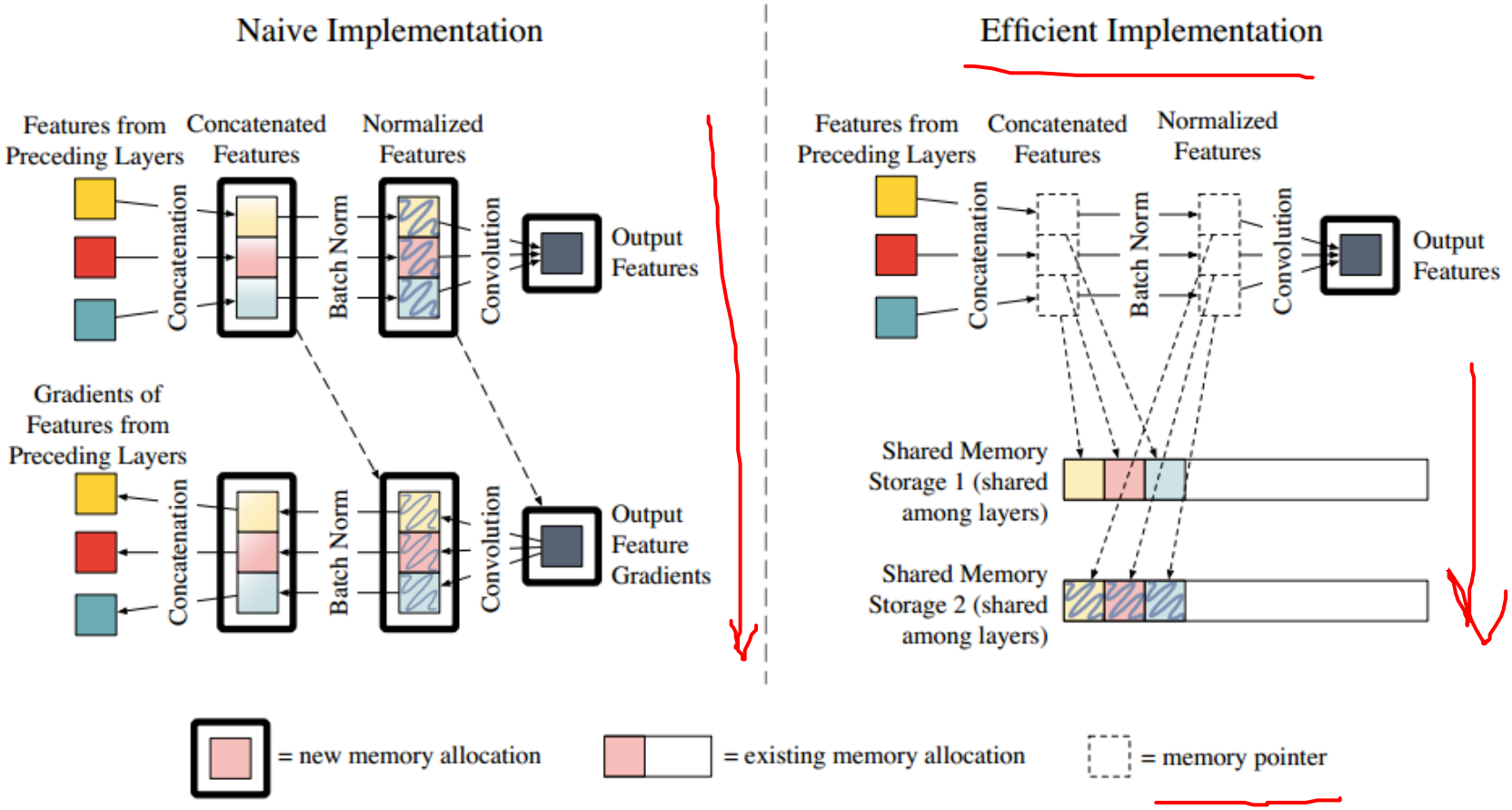
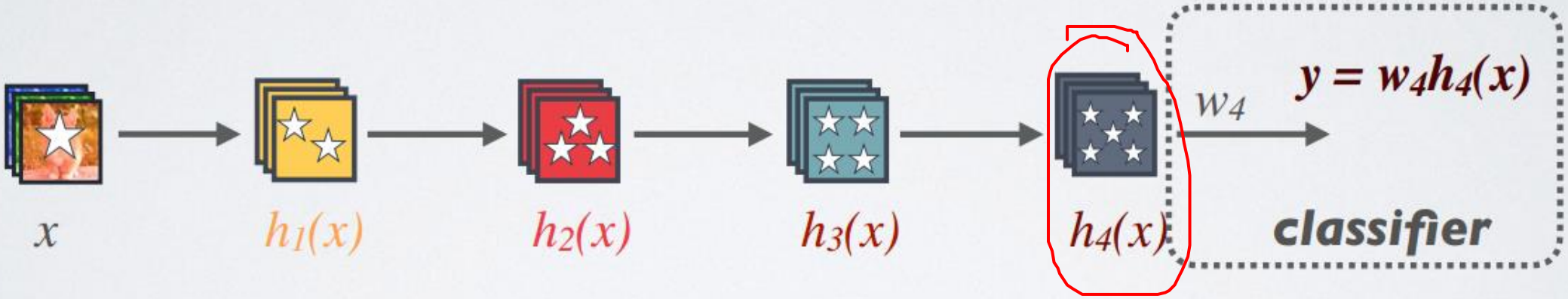


Figure 3: DenseNet layer forward pass: original implementation (**left**) and efficient implementation (**right**). Solid boxes correspond to tensors allocated in memory, where as translucent boxes are pointers. Solid arrows represent computation, and dotted arrows represent memory pointers. The efficient implementation stores the output of the concatenation, batch normalization, and ReLU layers in temporary storage buffers, whereas the original implementation allocates new memory.

ADVANTAGE 3: MAINTAINS LOW COMPLEXITY FEATURES

Standard Connectivity:

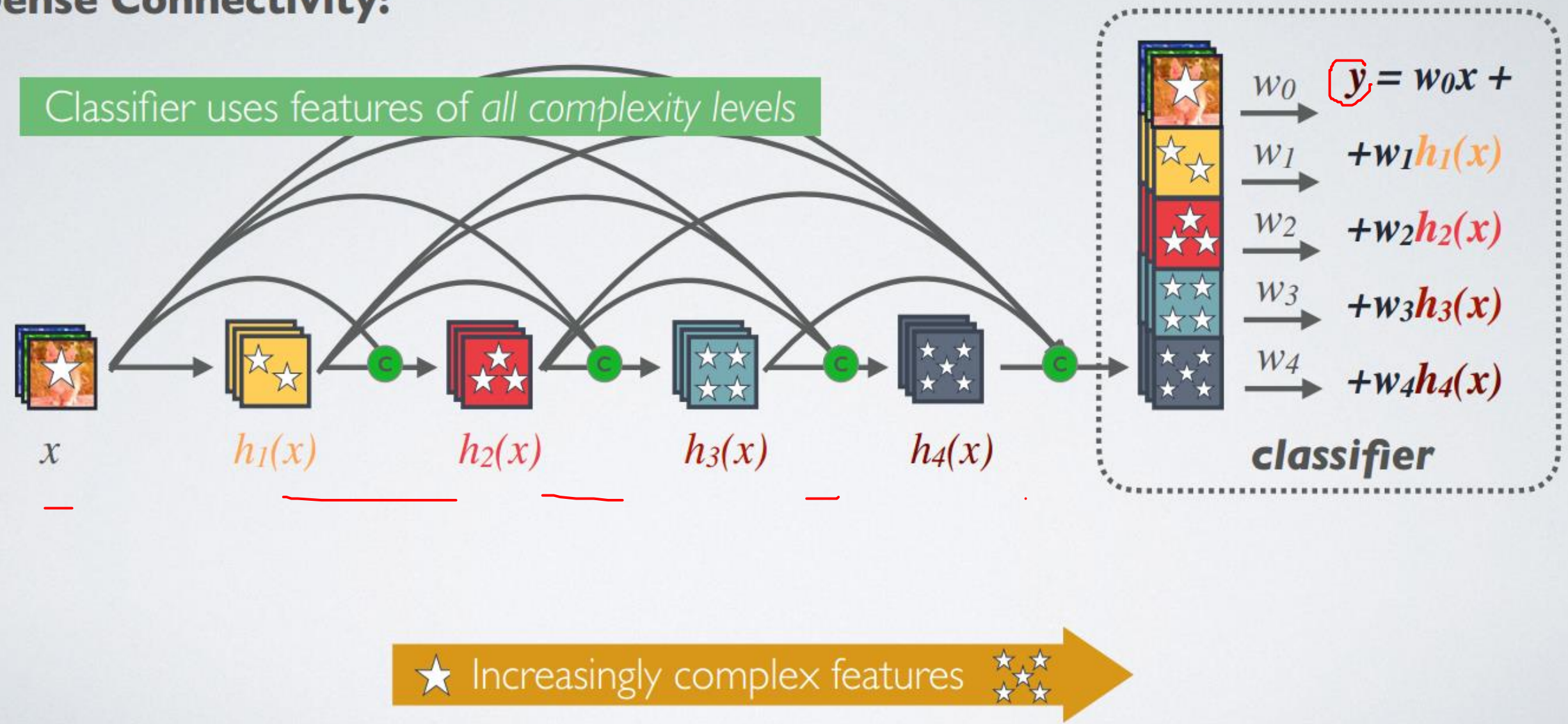
Classifier uses most complex (high level) features



Increasingly complex features

ADVANTAGE 3: MAINTAINS LOW COMPLEXITY FEATURES

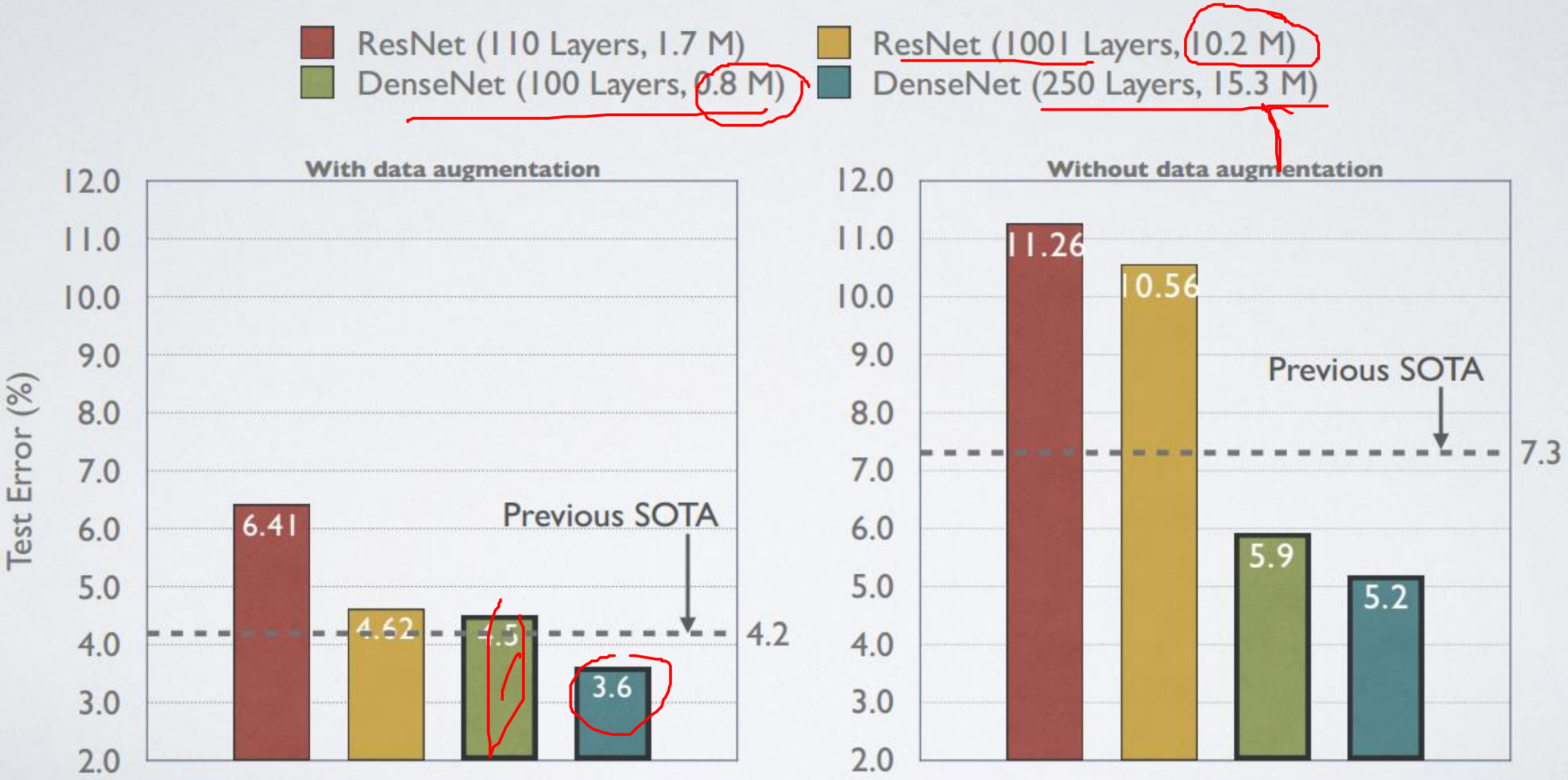
Dense Connectivity:



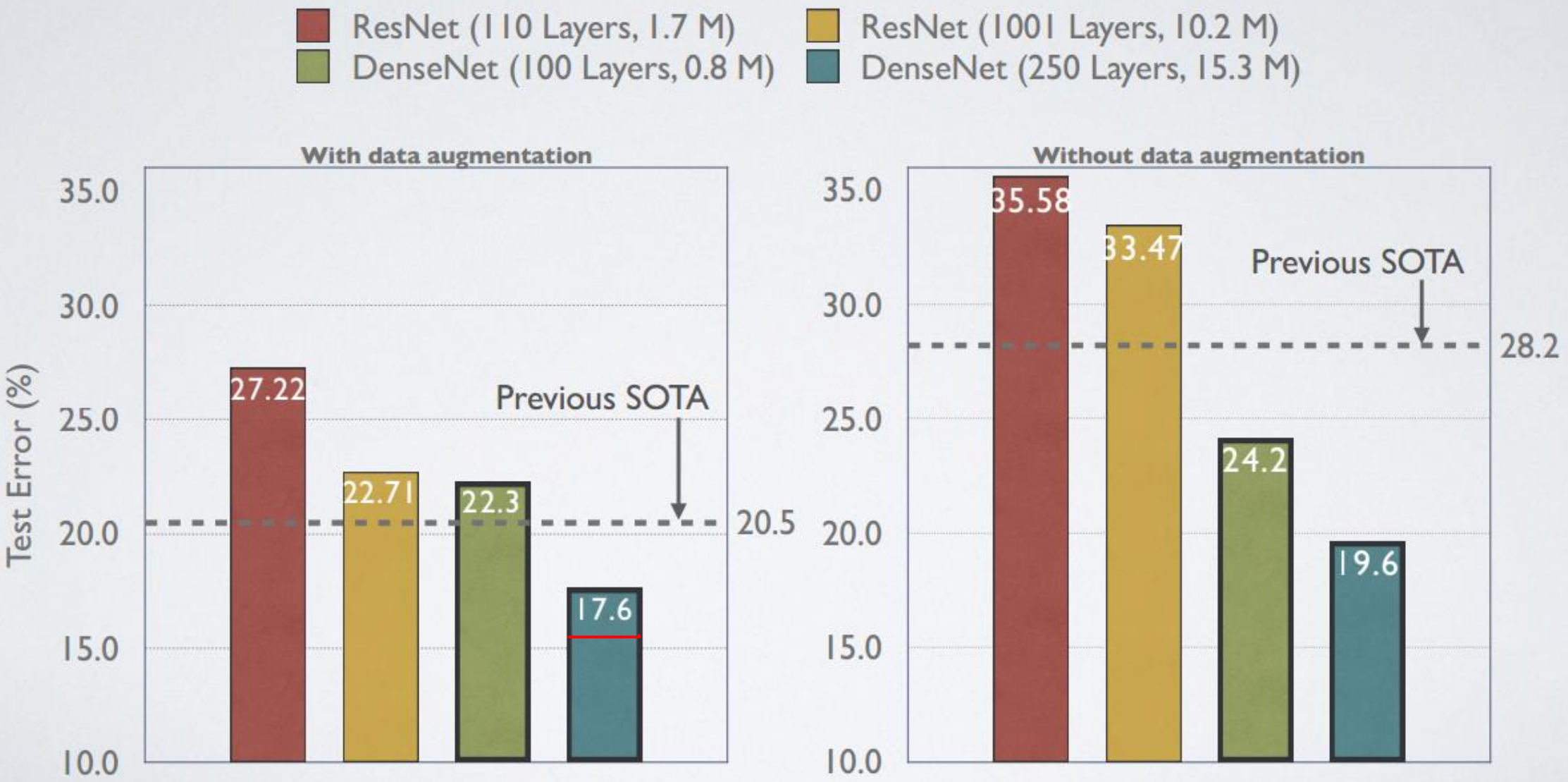
Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Table 2: Error rates (%) on CIFAR and SVHN datasets. k denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

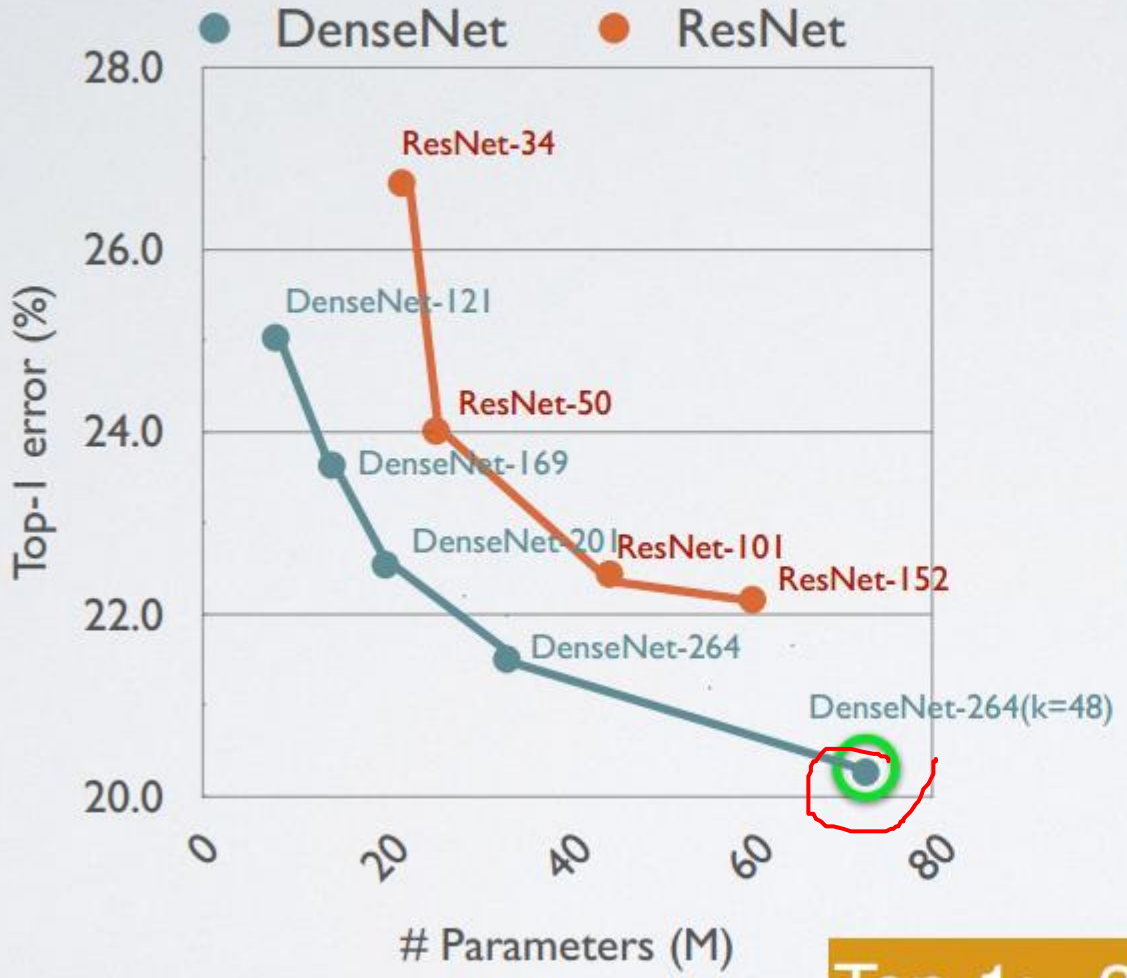
RESULTS ON **CIFAR-10**

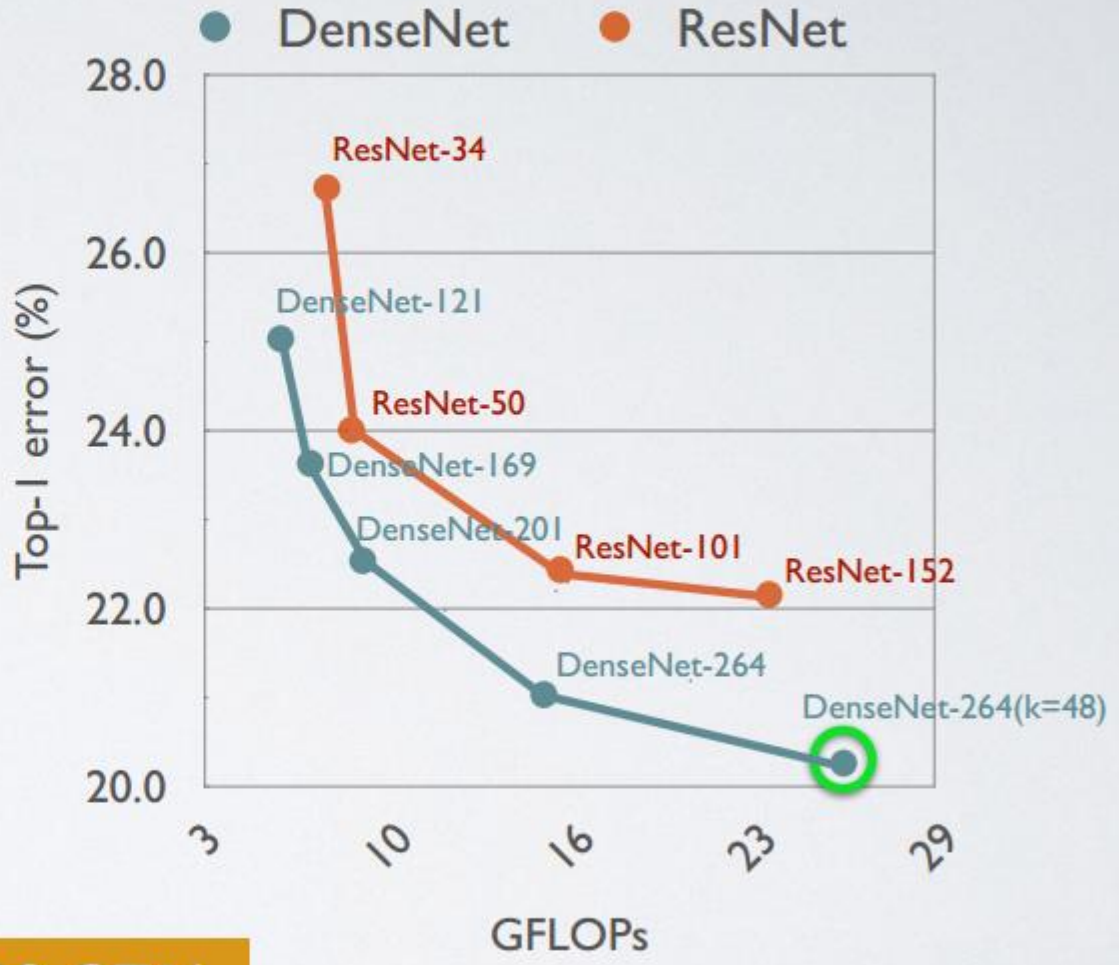


RESULTS ON **CIFAR-100**



RESULTS ON **IMAGENET**





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

Results on ImageNet

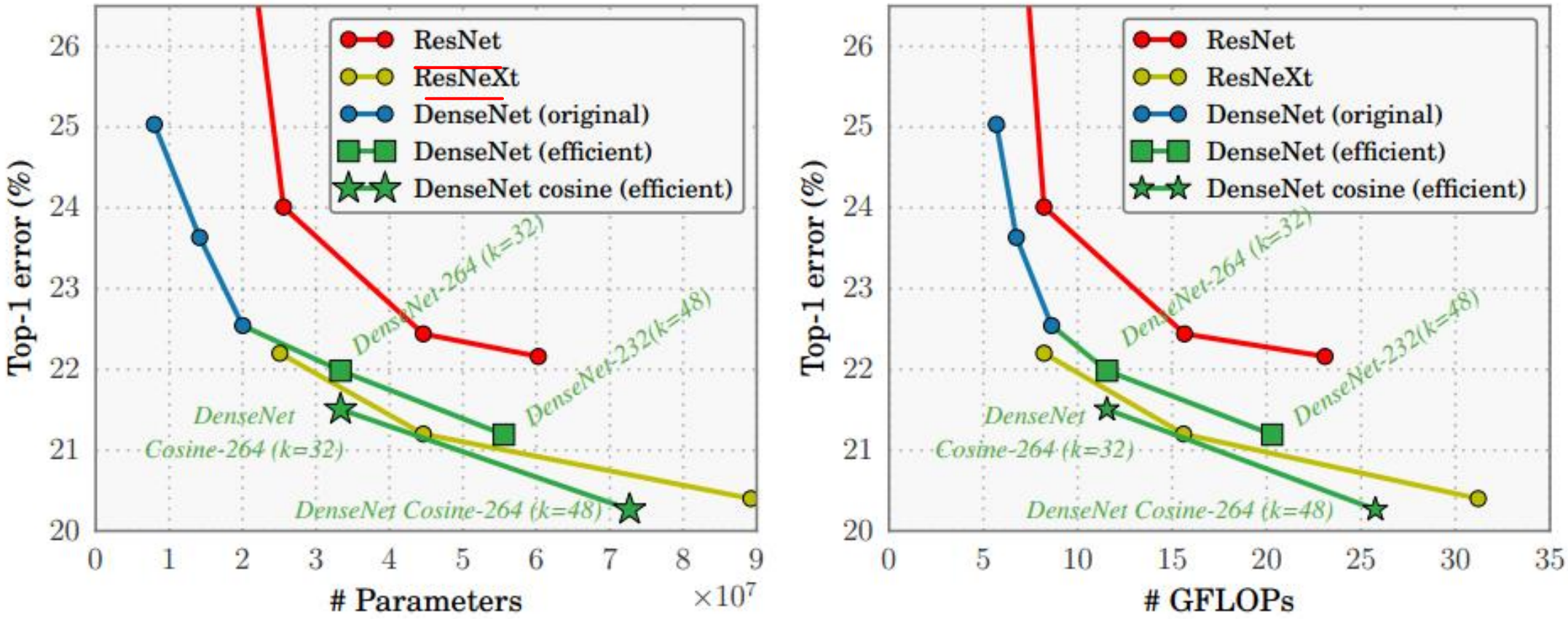


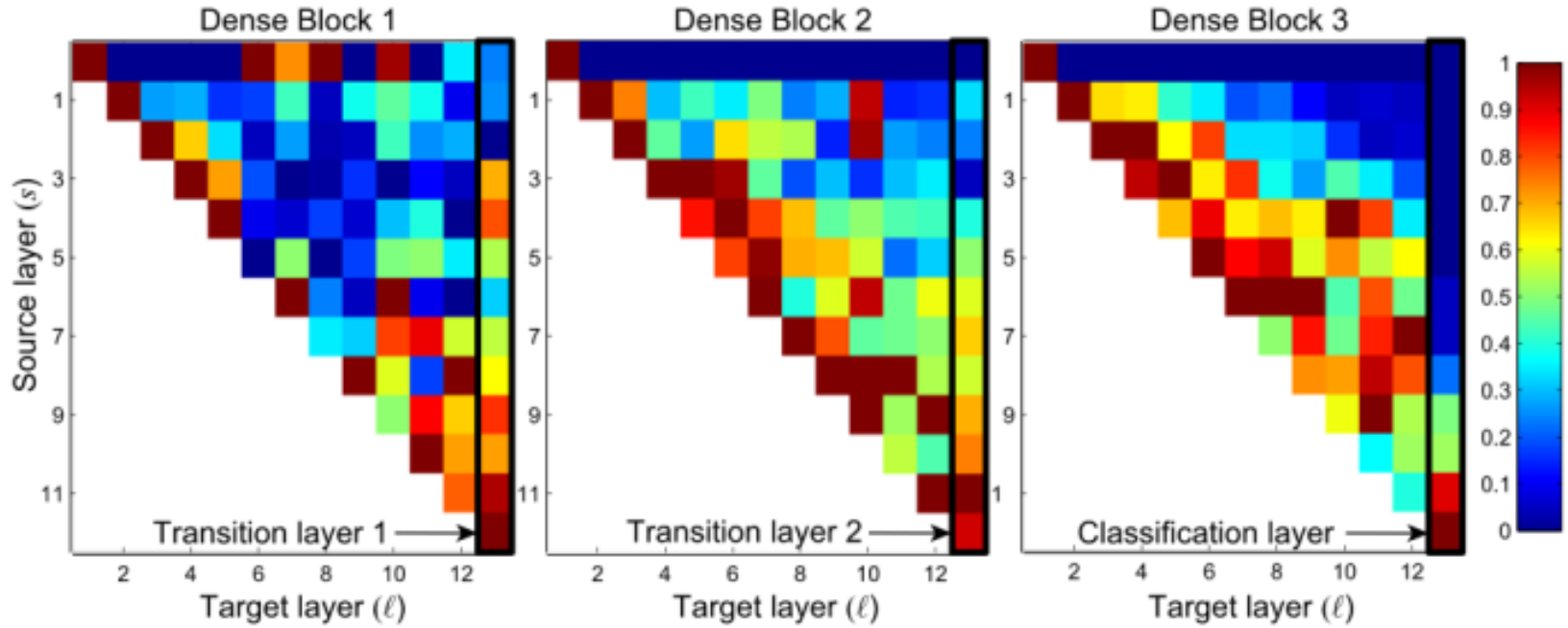
Figure 6: Top-1 classification error on ImageNet. The DenseNet models were trained on 8 NVIDIA Tesla M40 GPUs. Results in stars were not possible to train without the efficient implementation.

Results Others

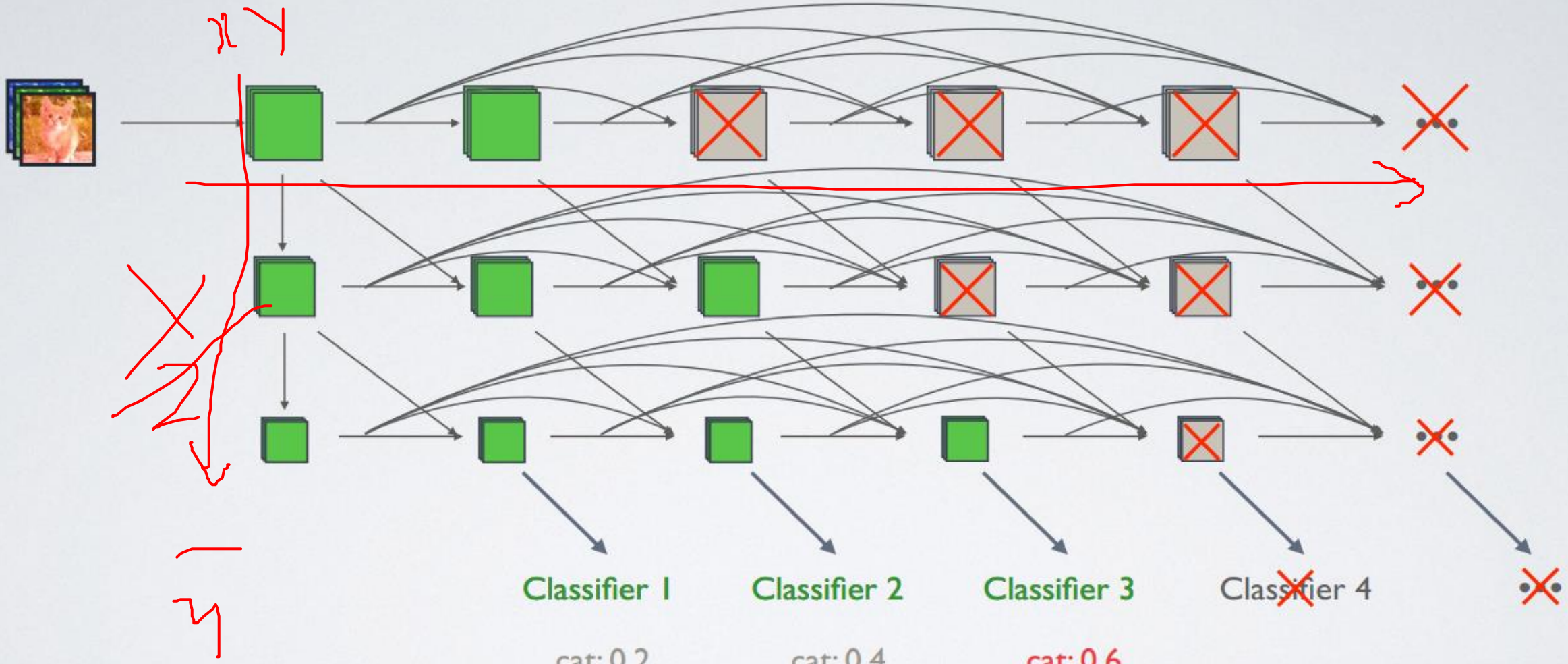
Efficient Models on ImageNet

Model	MFLOPs	Top-1 Err.
MobileNet	569	29.4%
DenseNet*	407	28.6%

Weight scales from source to target layers

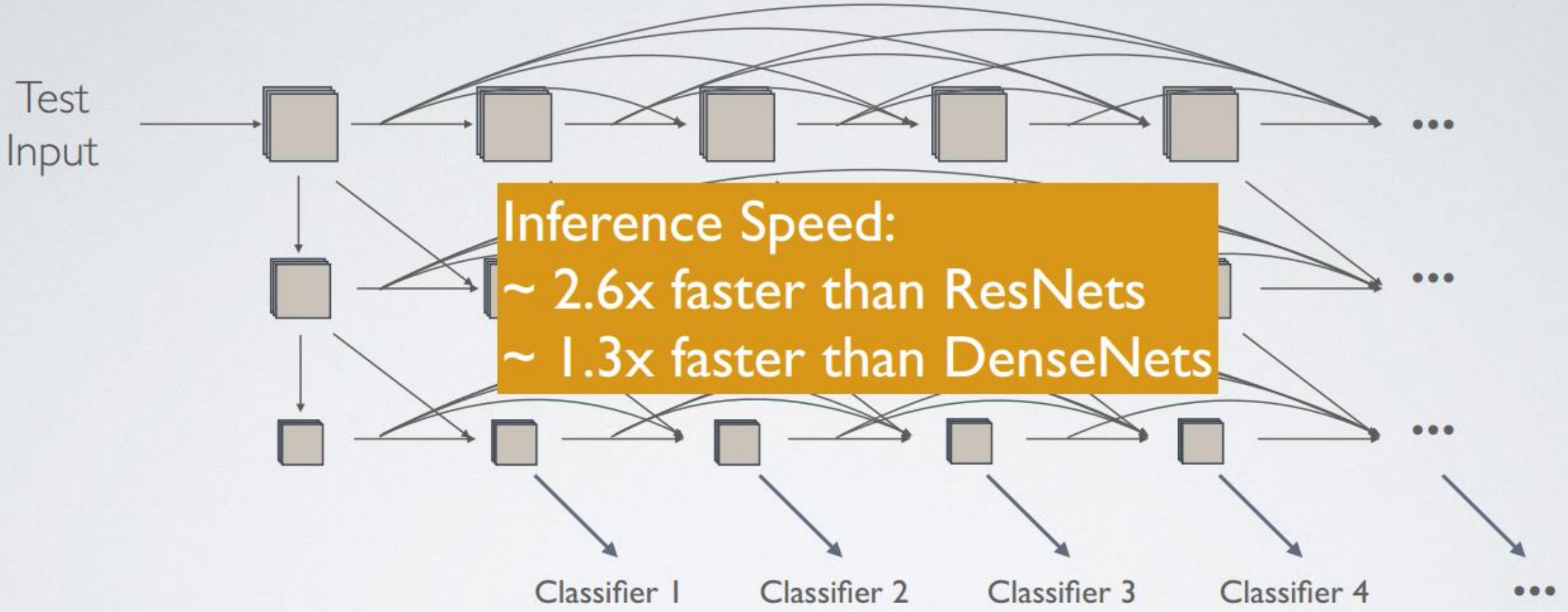


MULTI-SCALE DENSENET (Preview)



Multi-Scale DenseNet: [Huang, Chen, Li, Wu, van der Maaten, Weinberger] (arXiv Preprint: 1703.09844)

MULTI-SCALE DENSENET (Preview)



Applications of DenseNet

- DenseNet Semantic Segmentation: https://arxiv.org/pdf/1611.09326v2.pdf
- DenseNet for Optical flow: https://arxiv.org/pdf/1707.06316v1.pdf
- DenseNet for Object Detection: the next paper

DSOD: Learning Deeply Supervised Object Detectors from Scratch

Zhiqiang Shen*[1], Zhuang Liu*[2], Jianguo Li[3], Yu-Gang Jiang[1], Yurong Chen[3], Xiangyang Xue[1]
[1]Fudan University, [2]Tsinghua University, [3]Intel Labs China
{zhiqiangshen13, ygj, xyxue}@fudan.edu.cn, liuzhuangthu@gmail.com
{jianguo.li, yurong.chen}@intel.com

Abstract

We present Deeply Supervised Object Detector (DSOD), a framework that can learn object detectors from scratch. State-of-the-art object objectors rely heavily on the off-the-shelf networks pre-trained on large-scale classification datasets like ImageNet, which incurs learning bias due to the difference on both the loss functions and the category distributions between classification and detection tasks. Model fine-tuning for the detection task could alleviate this bias to some extent but not fundamentally. Besides, transferring pre-trained models from classification to detection between discrepant domains is even more difficult (e.g. RGB to depth images). A better solution to tackle

vision tasks, such as image classification [17, 28, 32, 9, 10], object detection [5, 4, 27, 19, 21, 25], image segmentation [23, 8, 2, 36], etc. In the past several years, many innovative CNN network structures have been proposed. Szegedy *et al.* [32] propose an "Inception" module which concatenates features maps produced by various sized filters. He *et al.* [9] propose residual learning blocks with skip connections, which enable training *very deep* networks with more than 100 layers. Huang *et al.* [10] propose DenseNets with dense layer-wise connections. Thanks to these excellent network structures, the accuracy of many vision tasks has been greatly improved. Among them, object detection is one of the fastest moving areas due to its wide applications in surveillance, autonomous driving, etc.

Motivation

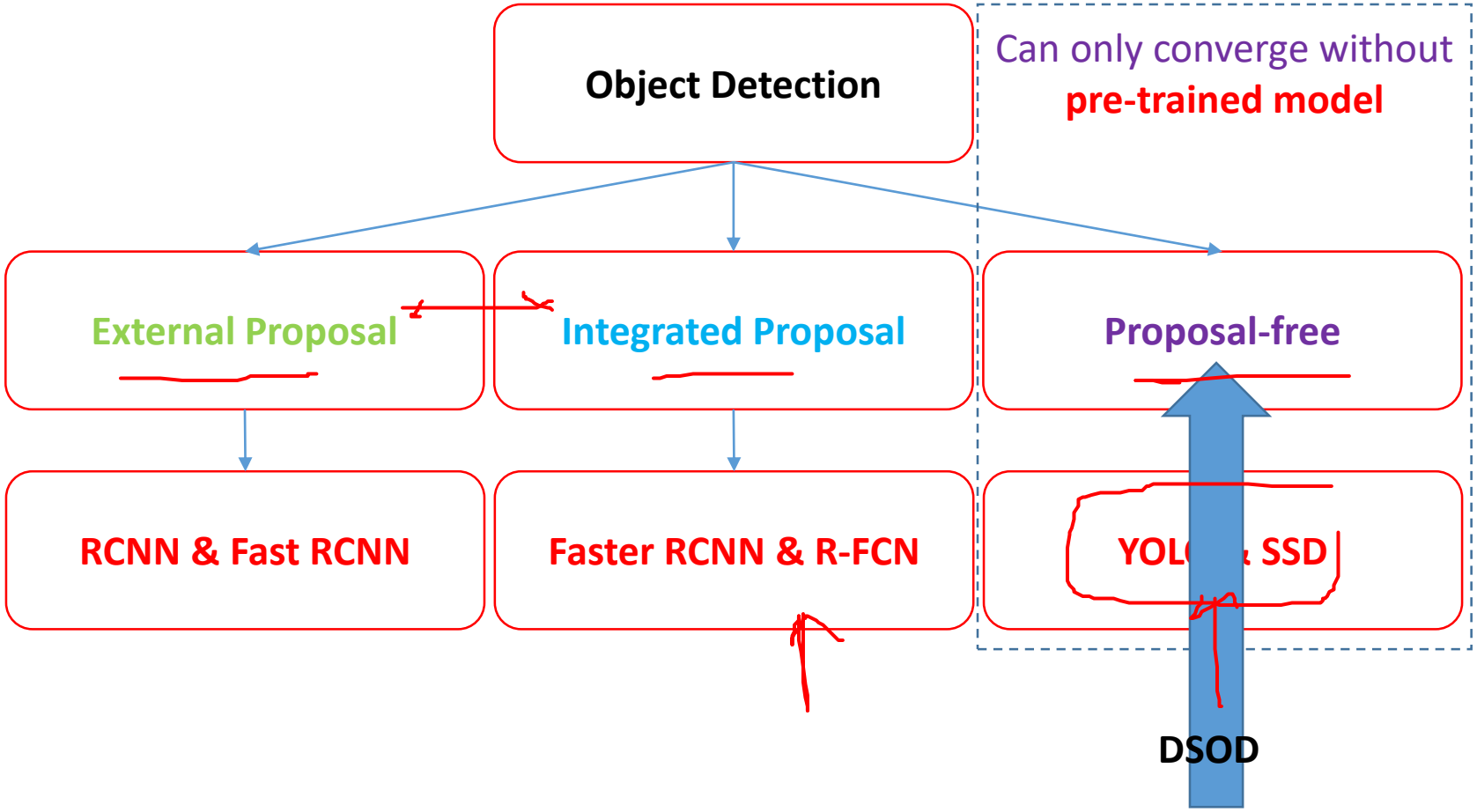
- Two Question:
- First, is it possible to train object detection networks from scratch?
- Second, if the first answer is positive, are there any principles to design a resource efficient network structure for object detection while keeping high detection accuracy?

- State-of-the-art object objectors rely heavily on the off the-shelf networks pre-trained on large-scale classification datasets like ImageNet, which incurs learning bias due to the difference on both the loss functions and the category distributions between classification and detection tasks.

Motivation

- Critical limitation of pre-trained networks
  - **Limited structure design space:** mostly from ImageNet-based classification task – huge number of parameters – therefore very little flexibility
  - **Learning bias:** both loss functions and the category distributions between classification and detections tasks are different – lead to different search/optimization spaces
  - **Domain mismatch:** source domain (ImageNet) has a huge mismatch to the target domain such as depth images, medical images, etc.

Object Detection: Categories



Architecture

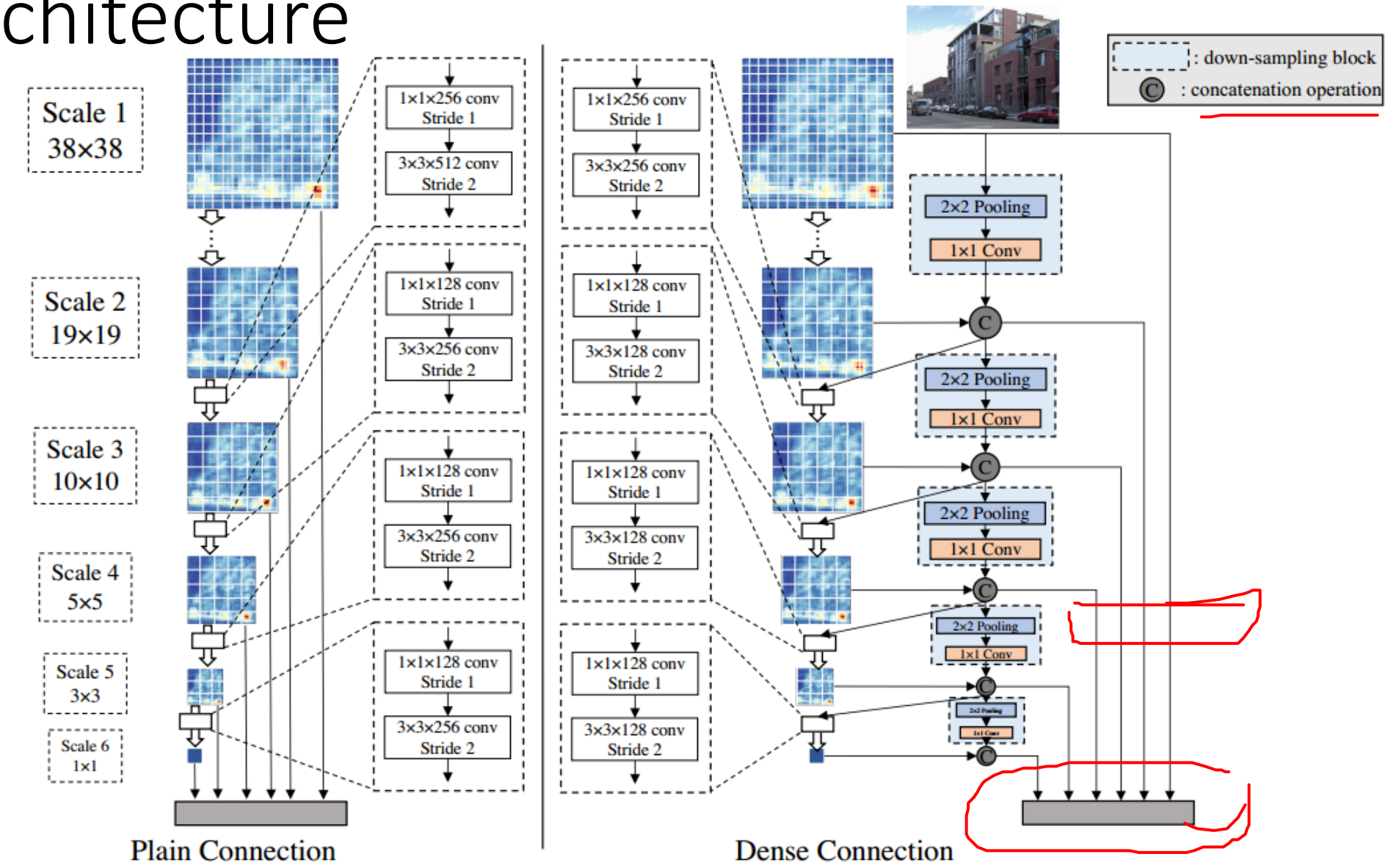


Figure 1: DSOD prediction layers with plain and dense structures (for 300×300 input). Plain structure is introduced by SSD [21] and dense structure is ours. See Section 3 for more details.

DSOD

- DSOD is fairly flexible, so that we can tailor various network structures for different computing platforms such as server, desktop, mobile and even embedded devices.

- Contributions:
  - to the best of our knowledge, world first framework that can train object detection networks from scratch with state-of-the-art performance.
  - set of principles to design efficient object detection networks from scratch through step-by-step ablation studies.
  - achieve state-of-the-art performance on three standard benchmarks (PASCAL VOC 2007, 2012 and MS COCO datasets) with real-time processing speed and more compact models.

DSOD Architecture

- DSOD network divided into two parts:
- the backbone sub-network for feature extraction
- the front-end sub-network for prediction over multi-scale response maps

Layers		Output Size (Input 3×300 × 300)	DSOD
Stem	Convolution	64×150×150	3×3 conv, stride 2
	Convolution	64×150×150	3×3 conv, stride 1
	Convolution	128×150×150	3×3 conv, stride 1
	Pooling	128×75×75	2×2 max pool, stride 2
Dense Block (1)		416×75×75	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)		416×75×75	1×1 conv
		416×38×38	2×2 max pool, stride 2
Dense Block (2)		800×38×38	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 8$
Transition Layer (2)		800×38×38	1×1 conv
		800×19×19	2×2 max pool, stride 2
Dense Block (3)		1184×19×19	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 8$
Transition w/o Pooling Layer (1)		1120×19×19	1×1 conv
Dense Block (4)		1568×19×19	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 8$
Transition w/o Pooling Layer (2)		1568×19×19	1×1 conv
DSOD Prediction Layers		–	Plain/Dense

Table 1: DSOD architecture (growth rate $k = 48$ in each dense block).

Results

Method	data	backbone network	pre-train	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
ION [1]	07+12+S	VGGNet	✓	76.4	87.5	84.7	76.8	63.8	58.3	82.6	79.0	90.9	57.8	82.0	64.7	88.9	86.5	84.7	82.3	51.4	78.2	69.2	85.2	73.5
Faster RCNN [27]	07++12	ResNet-101	✓	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
R-FCNmulti-sc [19]	07++12	ResNet-101	✓	77.6	86.9	83.4	81.5	63.8	62.4	81.6	81.1	93.1	58.0	83.8	60.8	92.7	86.0	84.6	84.4	59.0	80.8	68.6	86.1	72.9
YOLOv2 [26]	07++12	Darknet-19	✓	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7
SSD300* [21]	07++12	VGGNet	✓	75.8	88.1	82.9	74.4	61.9	47.6	82.7	78.8	91.5	58.1	80.0	64.1	89.4	85.7	85.5	82.6	50.2	79.8	73.6	86.6	72.1
DSOD300	07++12	DS/64-192-48-1	✗	76.3	89.4	85.3	72.9	62.7	49.5	83.6	80.6	92.1	60.8	77.9	65.6	88.9	85.5	86.8	84.6	51.1	77.7	72.3	86.0	72.2
DSOD300	07++12+COCO	DS/64-192-48-1	✗	79.3	90.5	87.4	77.5	67.4	57.7	84.7	83.6	92.6	64.8	81.3	66.4	90.1	87.8	88.1	87.3	57.9	80.3	75.6	88.1	76.7

Table 5: **PASCAL VOC 2012 `test` detection results. 07+12**: 07 `trainval` + 12 `trainval`, **07+12+S**: 07+12 plus segmentation labels, **07++12**: 07 `trainval` + 07 `test` + 12 `trainval`. Result links are DSOD300 (07+12) : `http://host.robots.ox.ac.uk:8080/anonymous/PIOBKI.html`; DSOD300 (07+12+COCO): `http://host.robots.ox.ac.uk:8080/anonymous/I0UUHO.html`.

Method	data	network	pre-train	Avg. Precision, IoU: 0.5:0.95	0.5	0.75	Avg. Precision, Area: S	M	L	Avg. Recall, #Dets: 1	10	100	Avg. Recall, Area: S	M	L
Faster RCNN [27]	trainval	VGGNet	✓	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [1]	train	VGGNet	✓	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
R-FCN [19]	trainval	ResNet-101	✓	29.2	51.5	-	10.3	32.4	43.3	-	-	-	-	-	-
R-FCNmulti-sc [19]	trainval	ResNet-101	✓	29.9	51.9	-	10.8	32.8	45.0	-	-	-	-	-	-
SSD300 (Huang et al.) [11]	< trainval35k	MobileNet	✓	18.8	-	-	-	-	-	-	-	-	-	-	-
SSD300 (Huang et al.) [11]	< trainval35k	Inception-v2	✓	21.6	-	-	-	-	-	-	-	-	-	-	-
YOLOv2 [26]	trainval35k	Darknet-19	✓	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
SSD300* [21]	trainval35k	VGGNet	✓	25.1	43.1	25.8	6.6	25.9	41.4	23.7	35.1	37.2	11.2	40.4	58.4
DSOD300	trainval	DS/64-192-48-1	✗	29.3	47.3	30.6	9.4	31.5	47.0	27.3	40.7	43.0	16.7	47.1	65.0

Table 6: **MS COCO `test-dev 2015` detection results.**

Results

- tried to train object detectors from scratch using the proposal-based framework such as Faster R-CNN and R-FCN. However, the training process failed to converge for all the network structures we attempted (VGGNet, ResNet, DenseNet).

- DSOD300 achieves 77.7% mAP, which is much better than the SSD300S that is trained from scratch using VGG16 (69.6%) without deep supervision

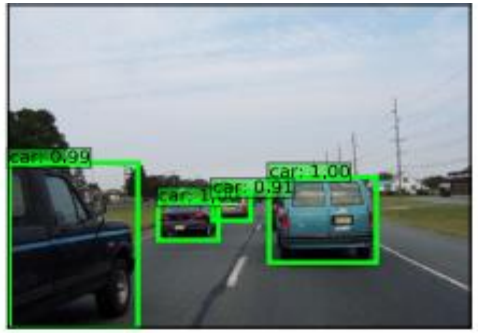
- more and more largescale datasets have been collected and released recently, such as the Open Images dataset [16], which is 7.5x larger in the number of images and 6x larger of categories than that of ImageNet.

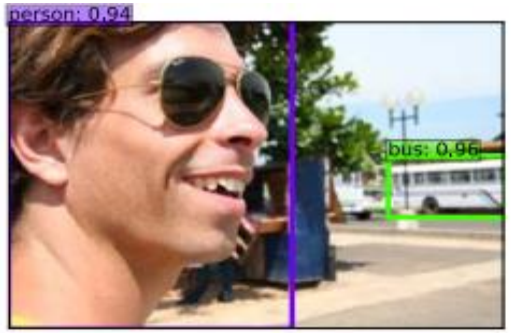
Results

- It is reasonable that our small object detection precision is slightly lower than R-FCN since our input image size (300×300) is much smaller than R-FCN's (～600×1000).

- our smallest dense model (DS/64-64-16-1, with dense prediction layers) achieves 73.6% mAP with only 5.9M parameters, which shows great potential for applications on low-end devices.

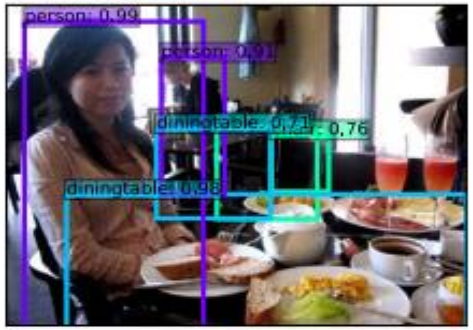
Check the network: http://ethereon.github.io/netscope/


car: 0.99
car: 1.00
car: 0.91

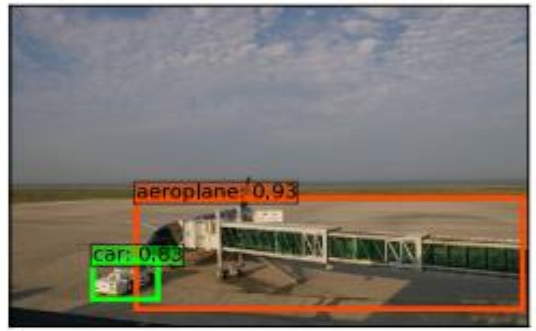

person: 0.94
bus: 0.96


car: 0.99
chair: 0.99


person: 0.99
person: 0.98


person: 0.99
person: 0.91
diningtable: 0.76
diningtable: 0.98


person: 1.00
tvmonitor: 1.00


aeroplane: 0.93
car: 0.83

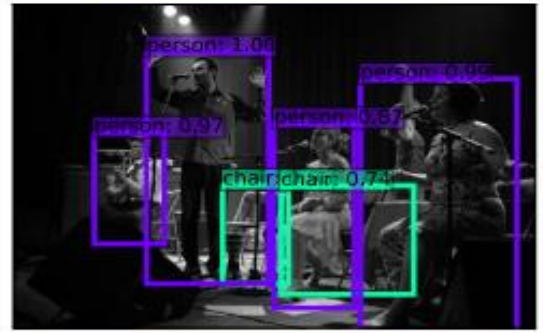

person: 0.67
motorbike: 0.99
bicycle: 0.83

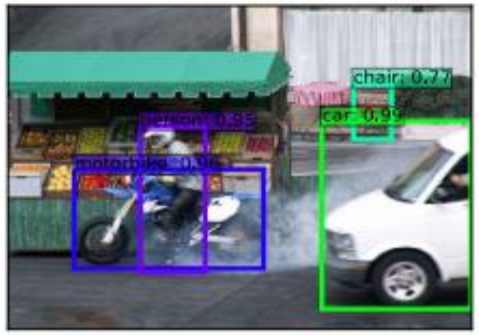

person: 1.00
person: 0.95
person: 0.98

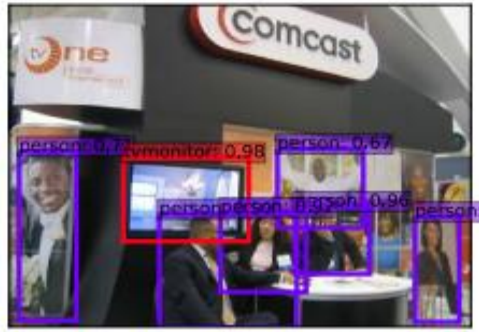

person: 1.00
person: 0.97
person: 1.00
person: 1.00
diningtable: 0.75

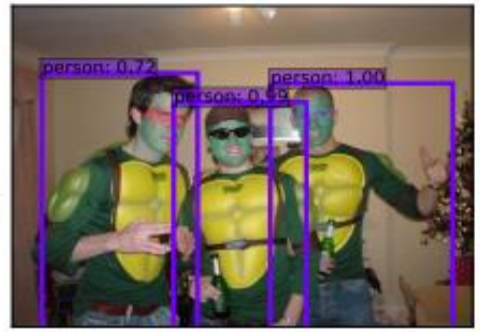

train: 0.99


car: 0.93
dog: 0.99


person: 1.00
person: 0.96
person: 0.97
person: 0.67
chair: chair: 0.74


chair: 0.77
car: 0.99
motorbike: 0.96


Comcast
tvmonitor: 0.98
person: 0.67
person: 0.98

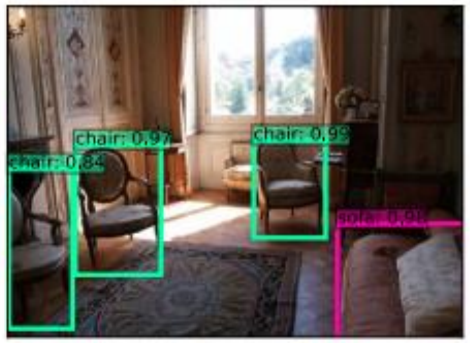

person: 0.72
person: 1.00


pottedplant: 0.85
diningtable: 0.94

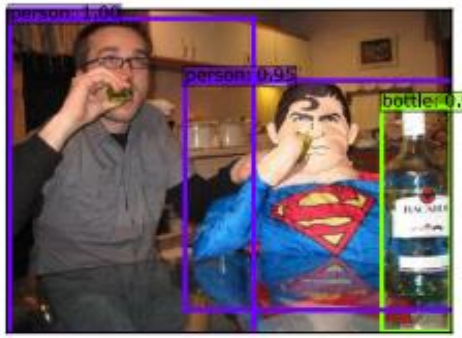

pottedplant: 0.97
person: 1.00


person: 0.99
chair: 0.89
person: 0.99

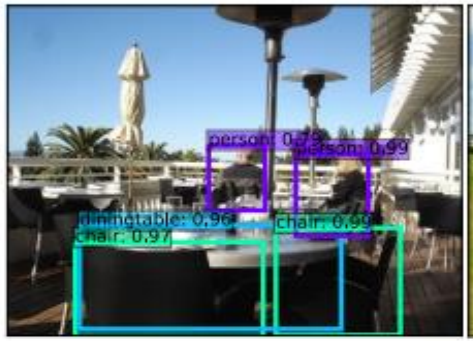

bus: 0.99
person: 0.94
car: 0.99
car: 0.99
FUNG WAH BUS

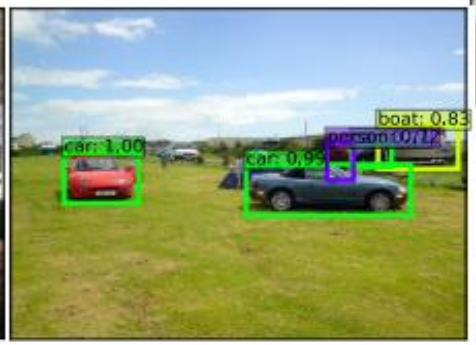

chair: 0.97
chair: 0.99
chair: 0.84

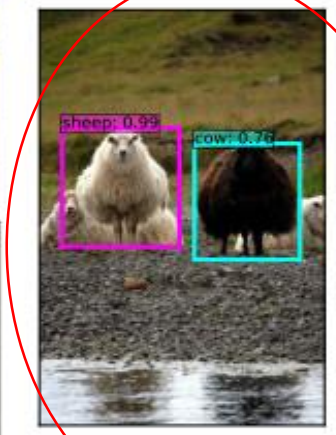

pottedplant: 0.98
pottedplant: 0.98
sofa: 0.95
chair: 0.87

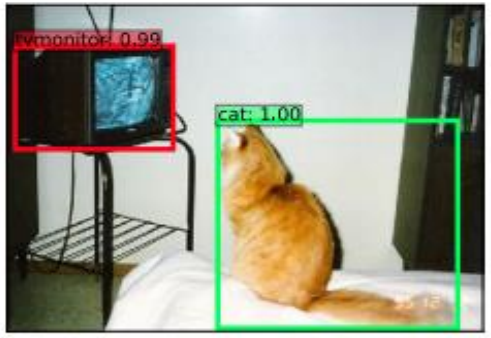

person: 0.95
bottle: 0.98

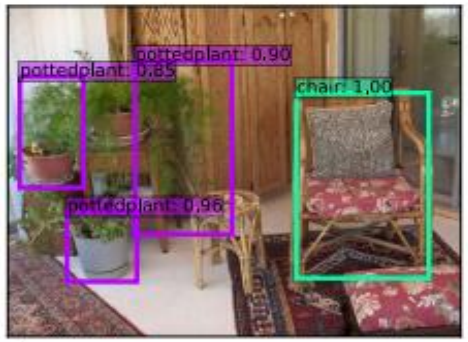

train: 1.00


person: 0.99
diningtable: 0.96
chair: 0.97
chair: 0.99


car: 1.00
car: 0.99
boat: 0.83

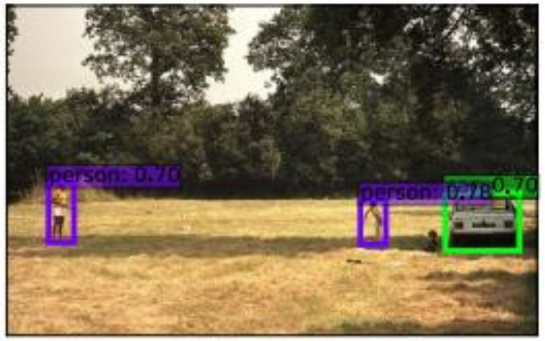

sheep: 0.99
cow: 0.76

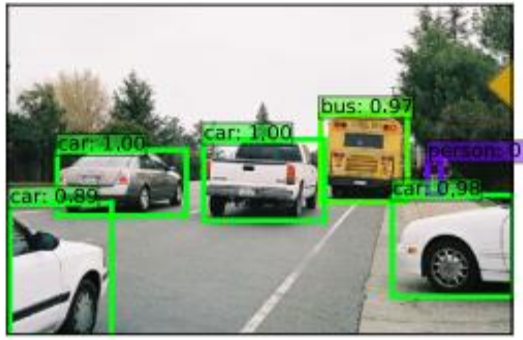

tvmonitor: 0.99
cat: 1.00

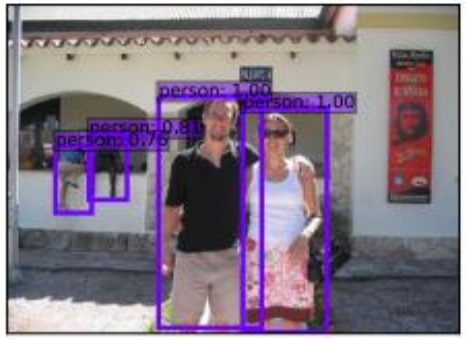

pottedplant: 0.90
pottedplant: 0.85
chair: 1.00
pottedplant: 0.98


car: 0.98
car: 0.83


bicycle: 0.88
bicycle: 0.98


person: 0.70
person: 0.78
0.70


bus: 0.97
car: 1.00
car: 1.00
person: 0.70
car: 0.89
car: 0.98


person: 1.00
person: 1.00
person: 0.81
person: 0.78


car: 0.9

Thanks