Densely Connected Convolutional Networks

作者：Gao Huang，Zhuang Liu，Laurens van der Maaten，Kilian Q. Weinberger

来源：CVPR17

汇报人：黄波 19210240246

Background

CNN层数增加

梯度消失

Batch Normalization

ResNet和Highway Networks构造恒等映射设置旁路

随机深度网络对网络中的一些层进行失活

Fractal Nets将不同深度的网络并行化

将feature map进行跨网络层的连接

Create short paths from early layers to later layers

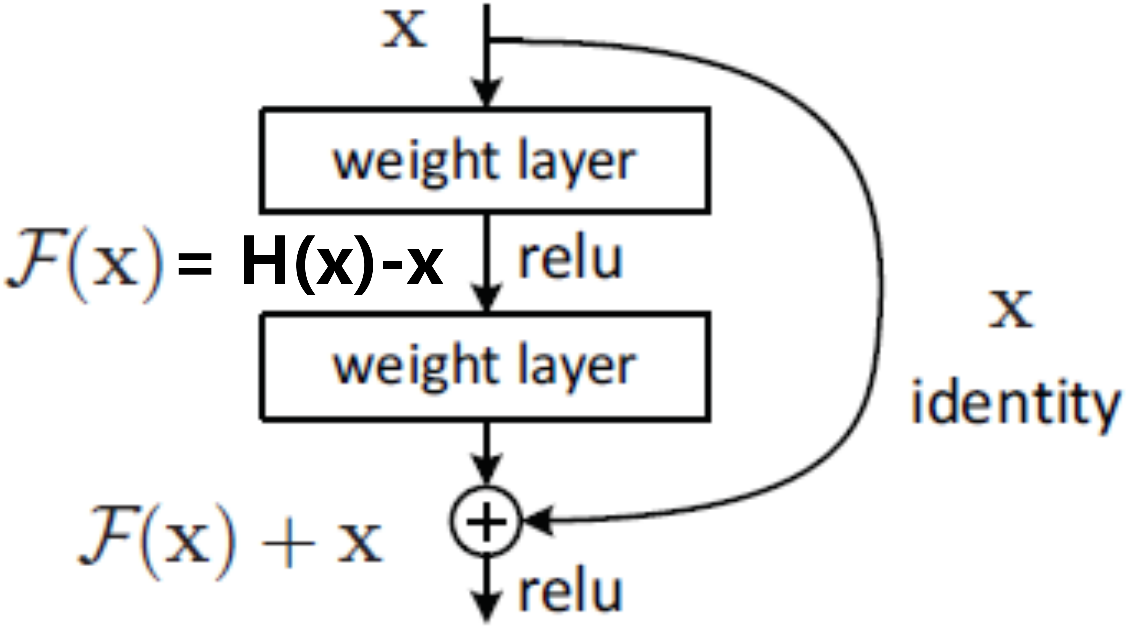
Background

- **ResNet**

$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$$

An advantage of ResNets is that the gradient can flow directly through the identity function from later layers to the earlier layers.

However, the identity function and the output are combined by summation, which may impede the information flow in the network.



DenseNet

- **Dense Connectivity**

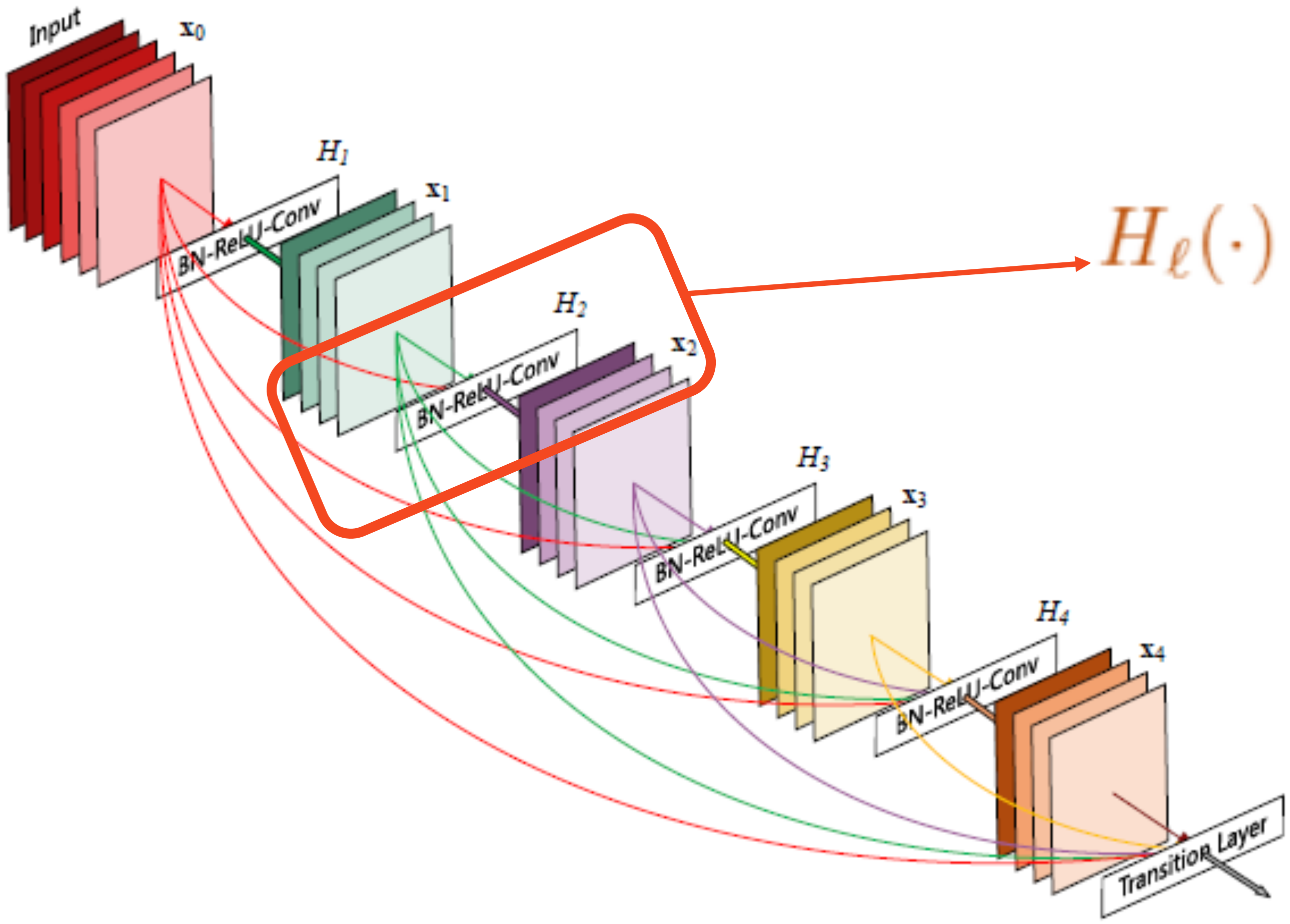
The ℓ^{th} layer receives the feature-maps of all preceding layers, $\mathbf{x}_0, \ldots, \mathbf{x}_{\ell-1}$ as input:

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$$

where $[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]$ refers to the concatenation of the feature-maps produced in layers $0, \ldots, \ell-1$

DenseNet

- **Dense Connectivity**



DenseNet

- **Dense Block**

The concatenation operation used in the equation is not viable when the size of feature-maps changes

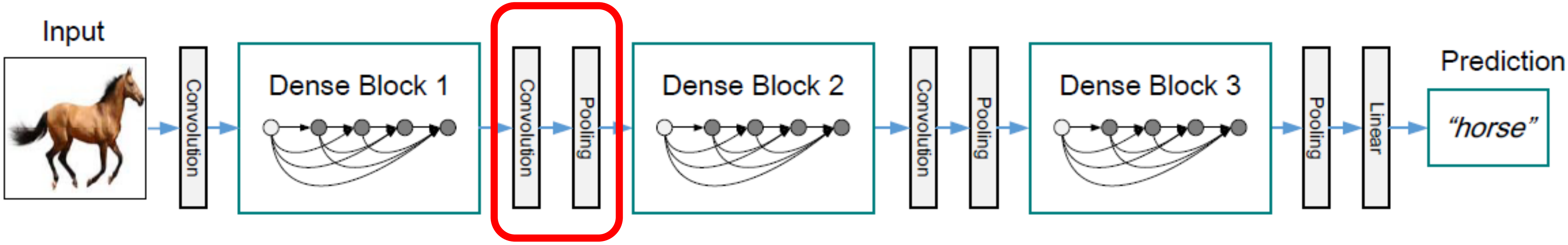
$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$$

An essential part of convolutional networks is down-sampling layers that change the size of feature-maps

To facilitate down-sampling in our architecture we divide the network into multiple densely connected dense blocks

DenseNet

- **Dense Block**

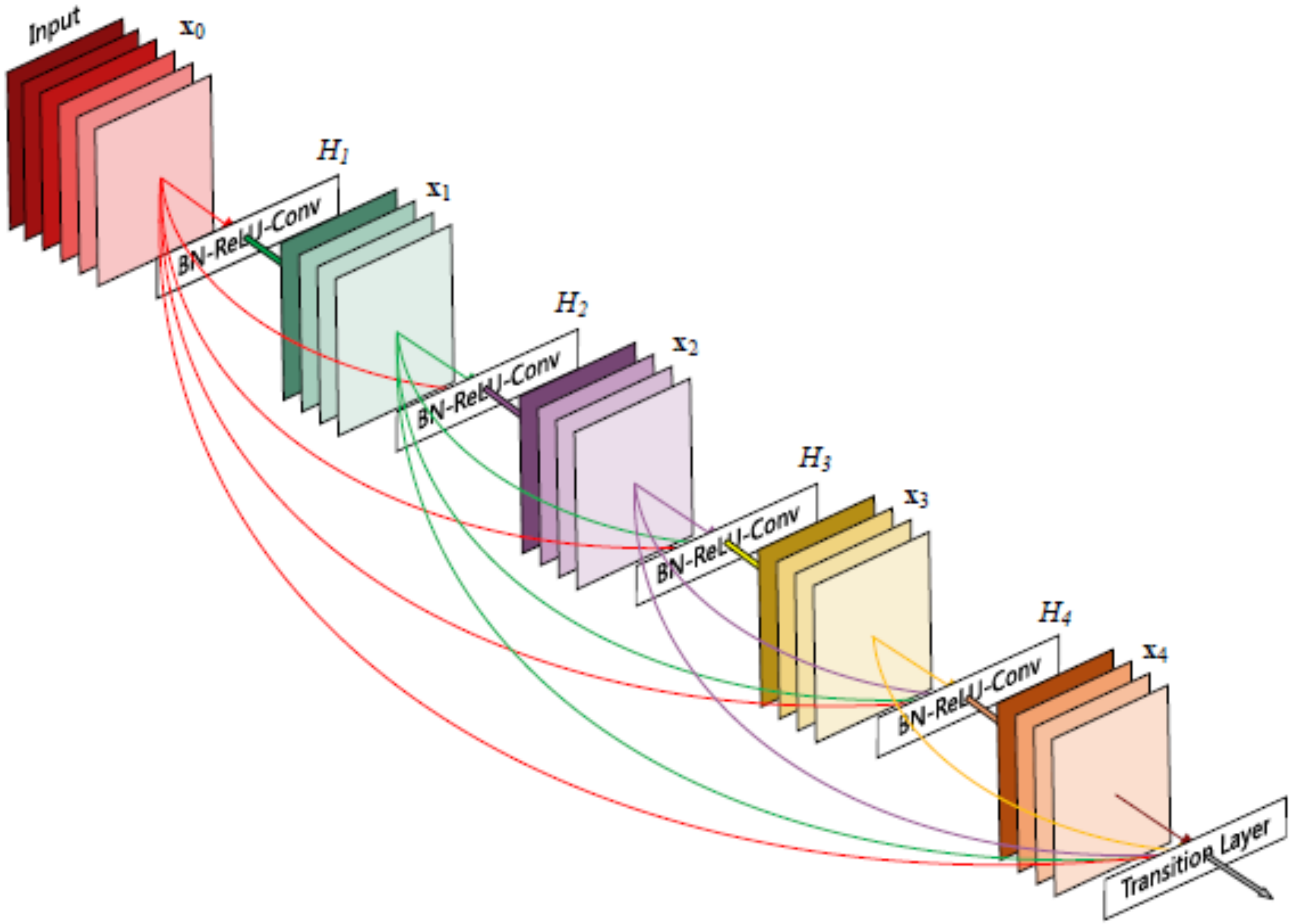


Transition Layer

DenseNet

- **Growth Rate**

If each function H_ℓ produces k feature-maps, it follows that ℓ^{th} the layer has $k_0 + k \times (\ell - 1)$ input feature-maps, where k0 is the number of channels in the input layer.



DenseNet

- **Bottleneck Layers**

$$k_0 + k \times (\ell - 1)$$

DenseBlock3 in DenseNet-169：

(32 - 1) × 32 + k0 > 1000

BN-ReLU-Conv(1×1)-BN-ReLU-Conv(3×3) → **DenseNet-B**

DenseNet

- **Compression**

We can reduce the number of feature-maps at transition layers

DenseNet with Compression：DenseNet-C

DenseNet with Bottleneck and Compression：DenseNet-BC

DenseNet

- **DenseNet architectures**

DenseNet architectures for ImageNet. The growth rate for all the networks is k = 32.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

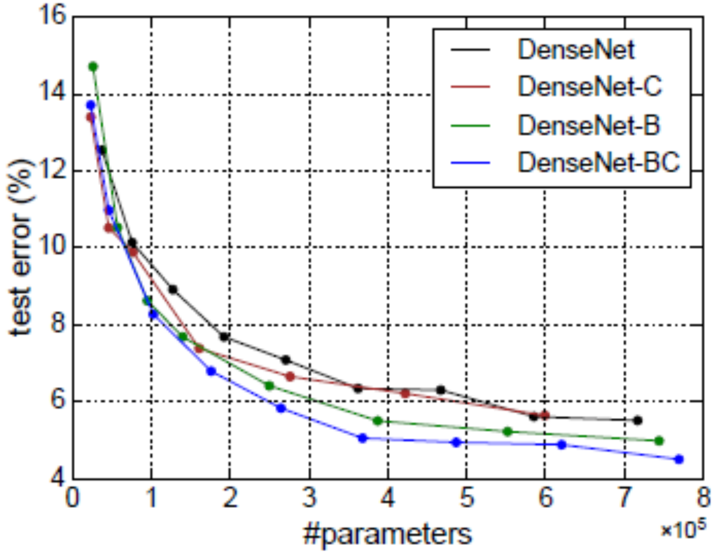
Experiments

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**	-

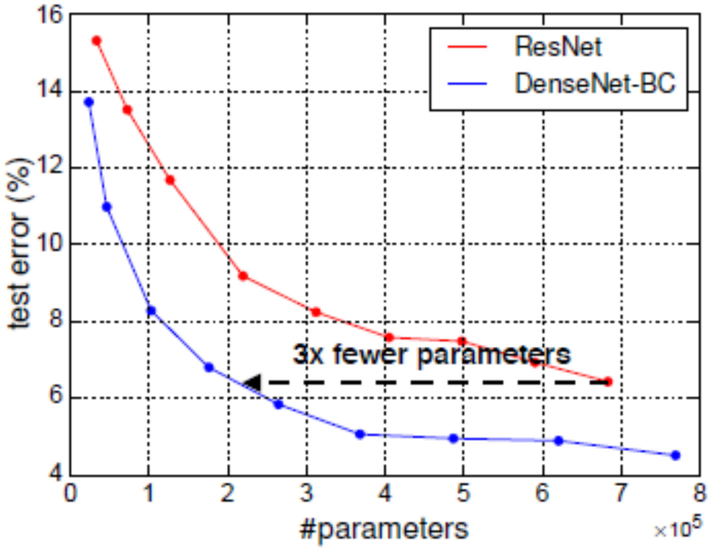
Error rates (%) on CIFAR and SVHN datasets.

Without data augmentation, DenseNet performs better by a large margin.

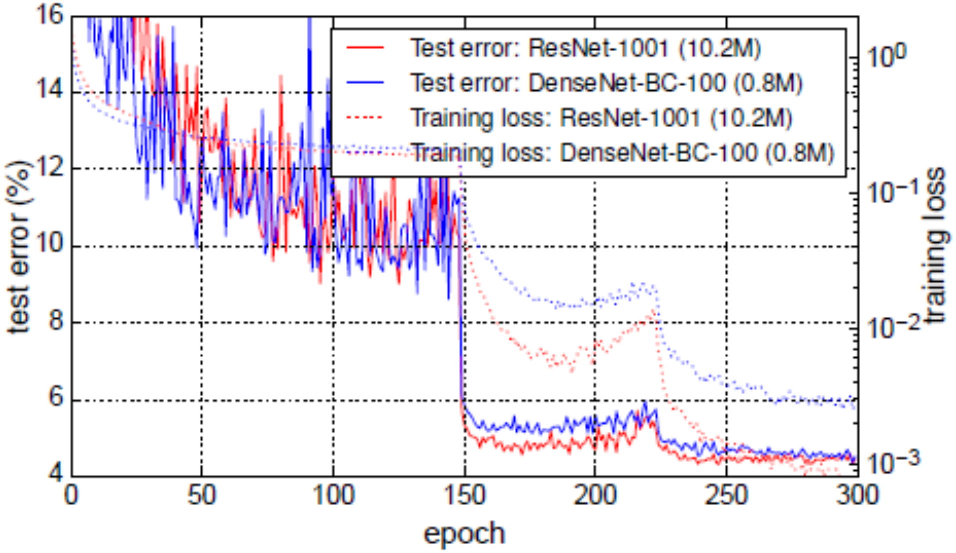
Experiments



Comparison of the parameter efficiency on C10+ between DenseNet variations



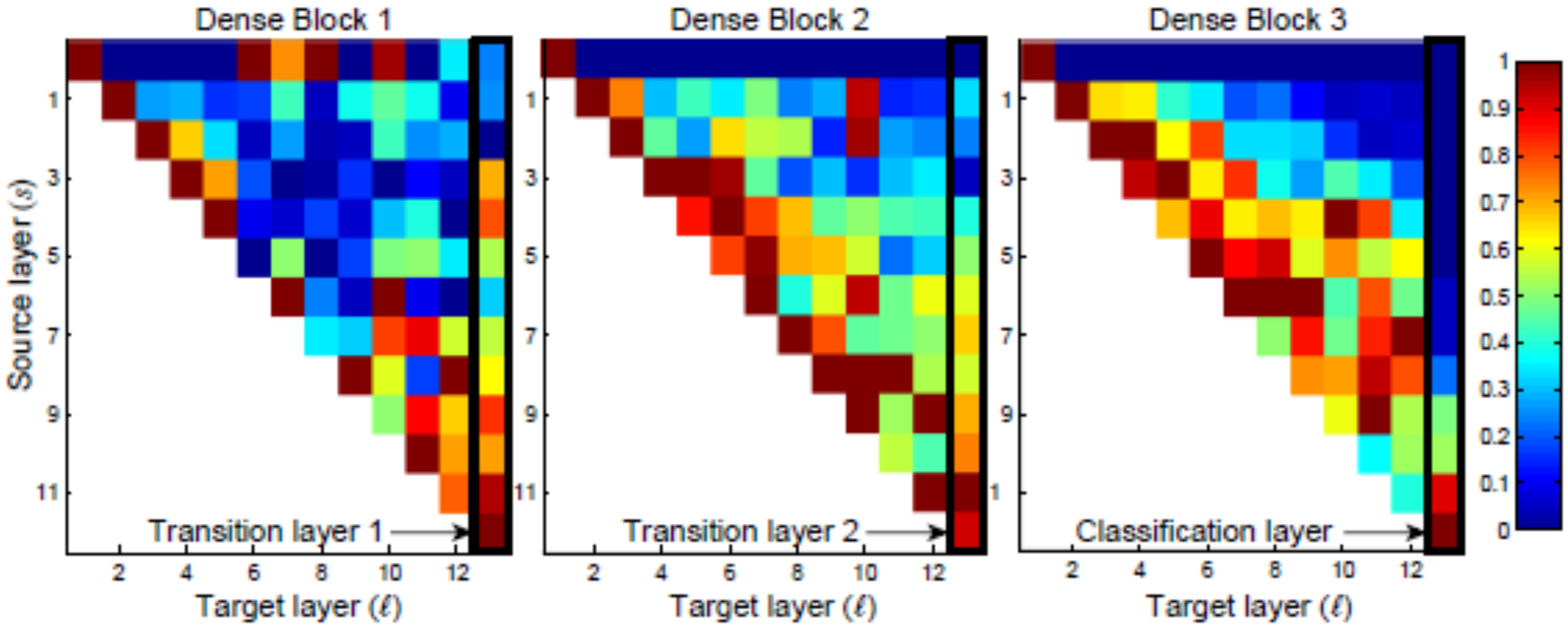
Comparison of the parameter efficiency between DenseNet-BC and (pre-activation) ResNets



Training and testing curves of the 1001-layer pre-activation ResNet and a 100-layer DenseNet

Discussion

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.



Summarize

- alleviate the vanishing-gradient problem
- strengthen feature propagation
- encourage feature reuse
- substantially reduce the number of parameters

Thanks!