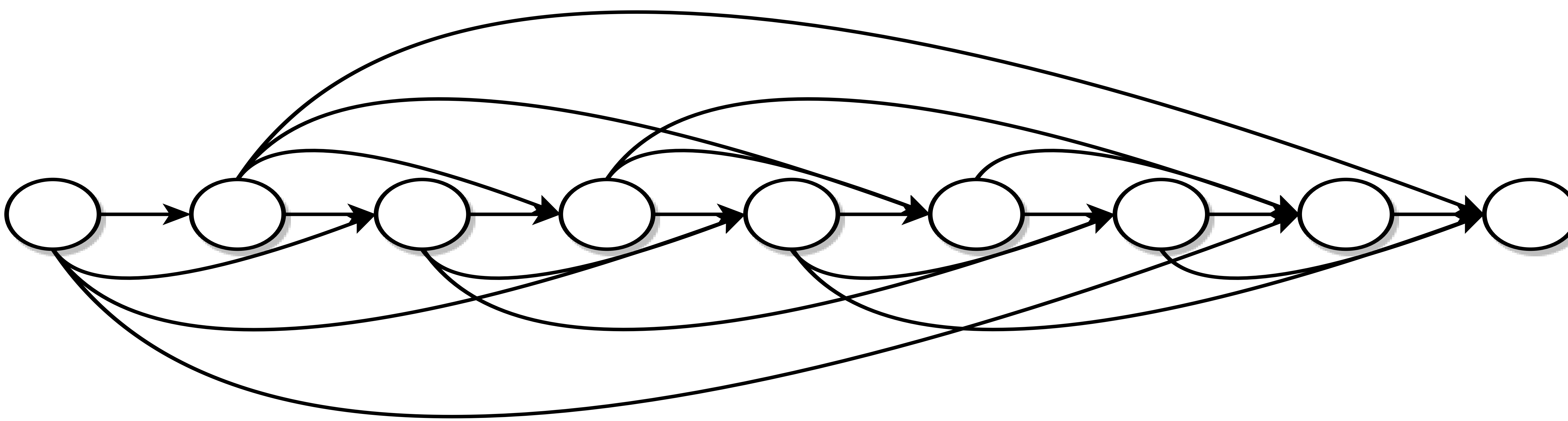
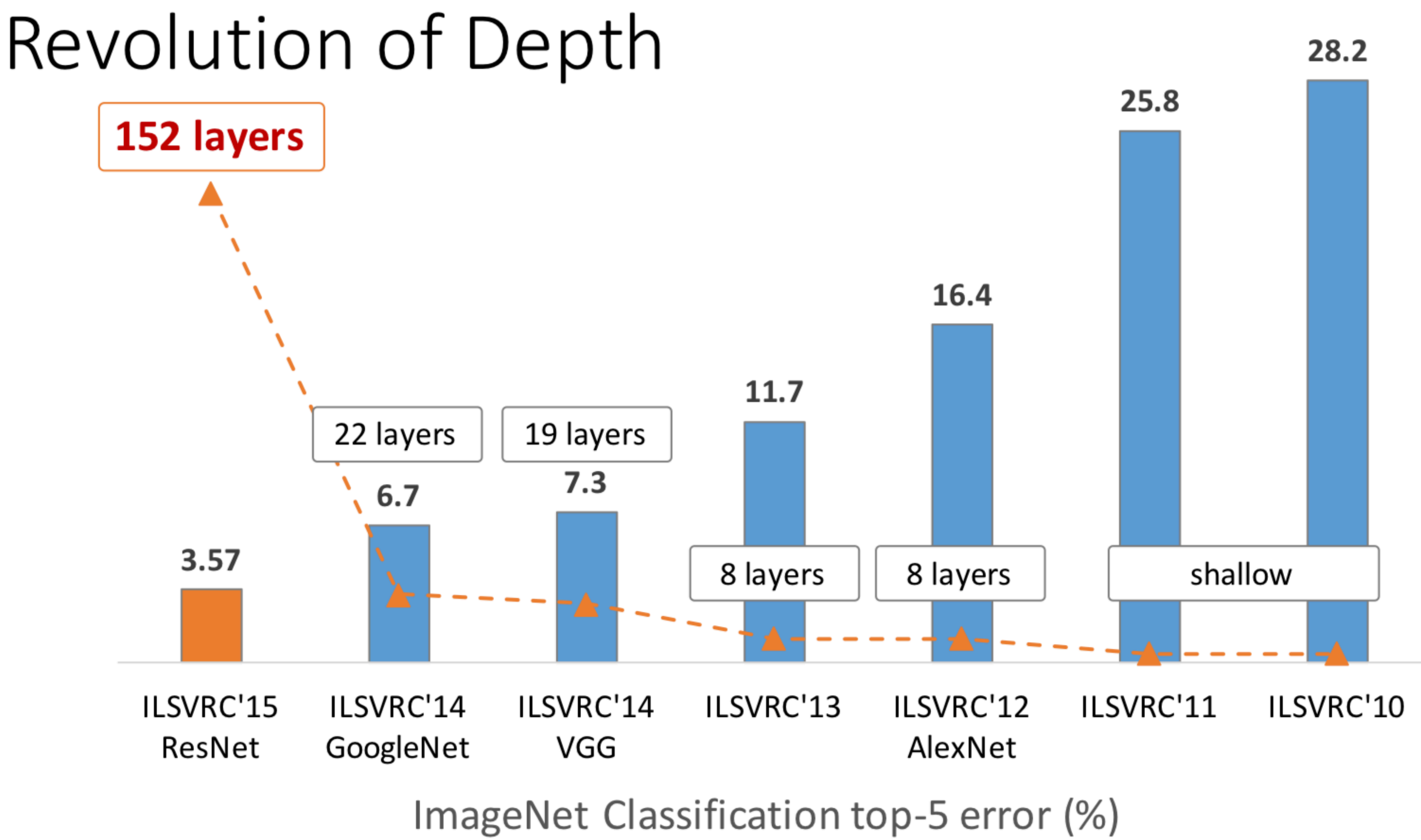
Sparsely Aggregated Convolutional Networks

Ligeng Zhu, Ruizhi Deng, Zhiwei Deng,
Greg Mori, Ping Tan

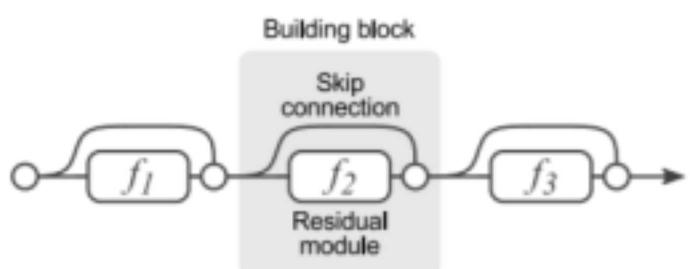


Power of Skip Connections



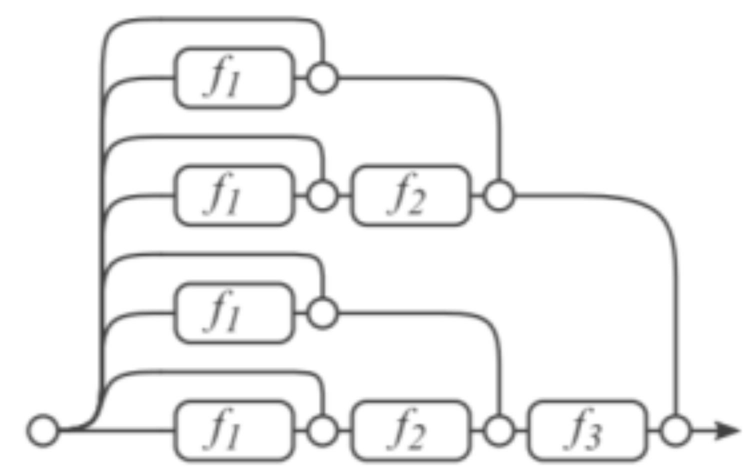
Residual Networks Behave Like Ensembles of Relatively Shallow Networks. (NIPS 2016)

- Skip connection matters!
    - ResNet = a collection of many paths



(a) Conventional 3-block residual network

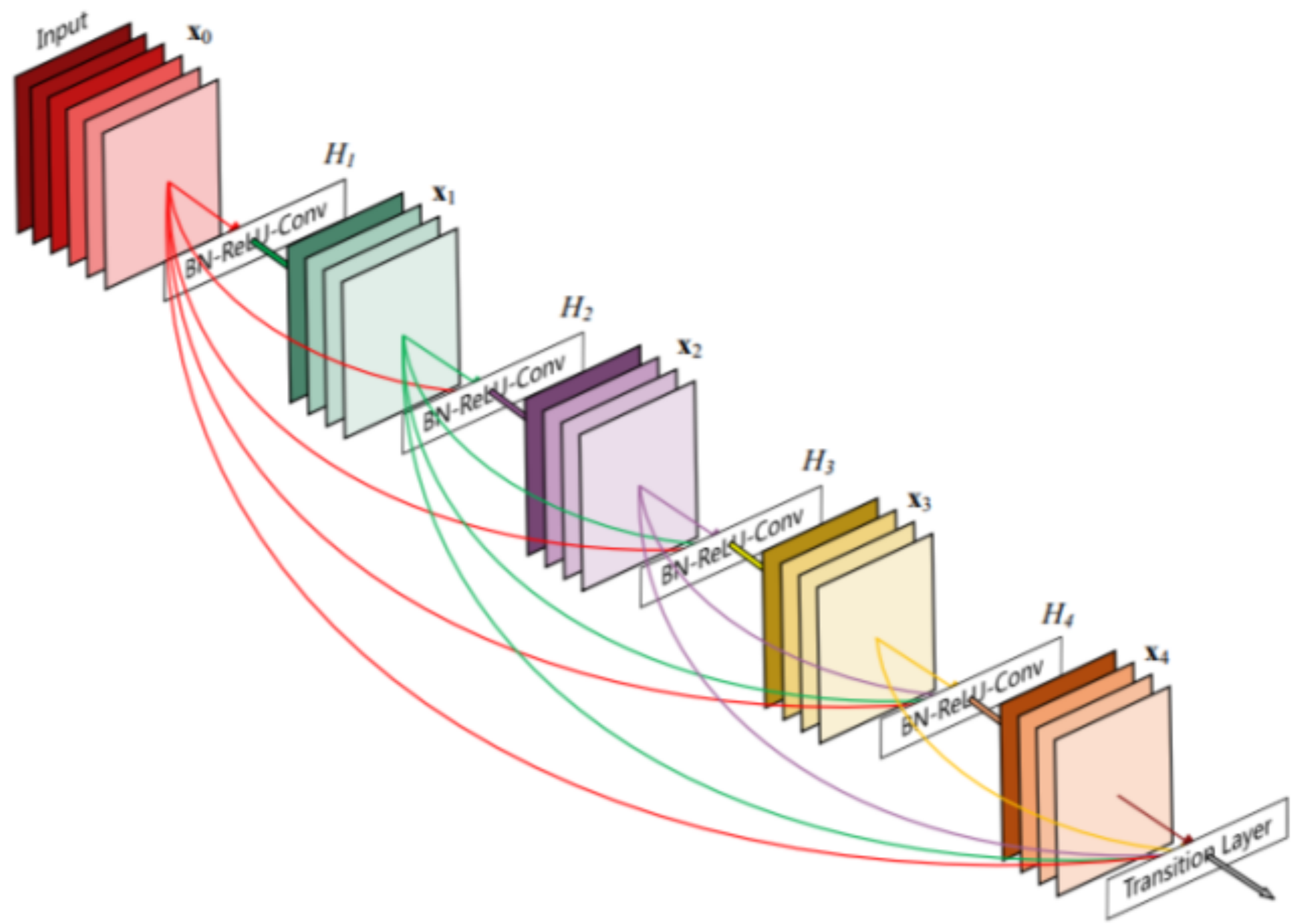
=



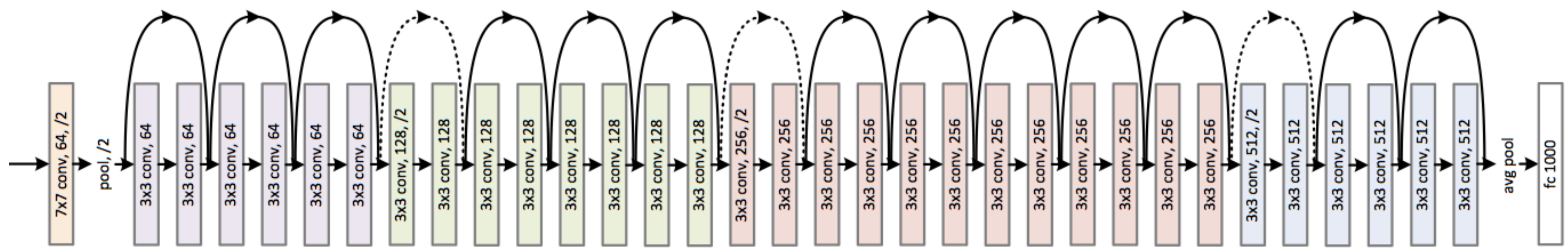
(b) Unraveled view of (a)

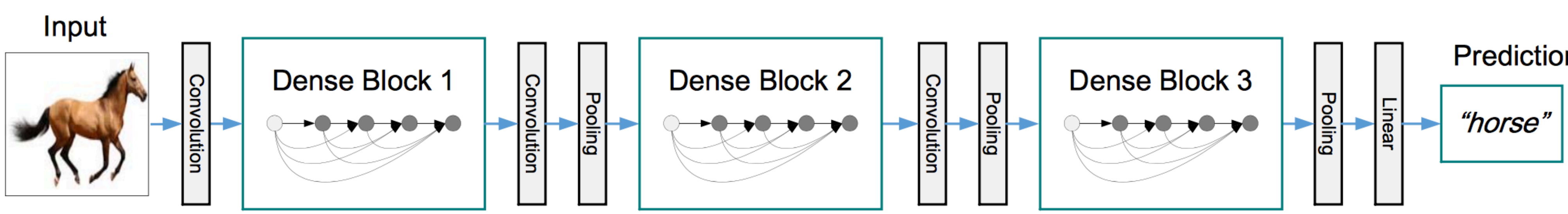
Why DenseNet Further Improves?

Cifar-10	param	error
Dense-40-12	1.0M	7.00
Dense-100-12	7.0M	5.77
Dense-100-24	27.2M	**5.83**
Res-164	1.7M	11.26
Res-1001	10.2M	10.56



Compare Dense & Res





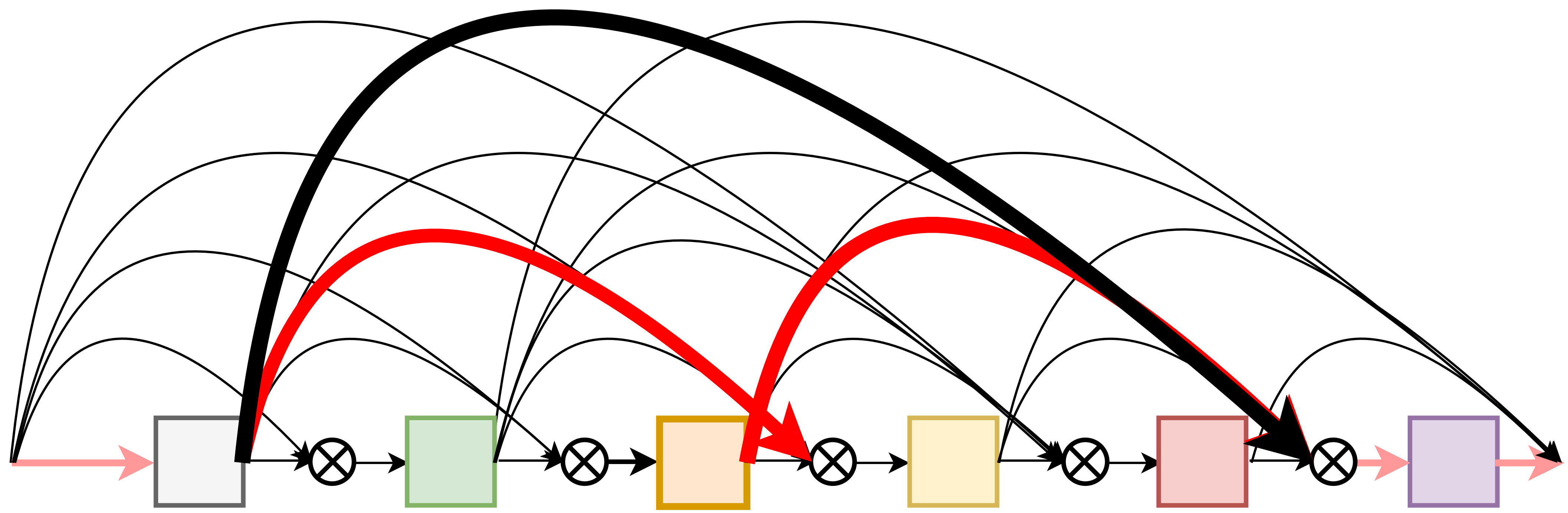
DenseNet has much more paths than ResNet. (**Dense**)

True ?

Compare Dense & Res

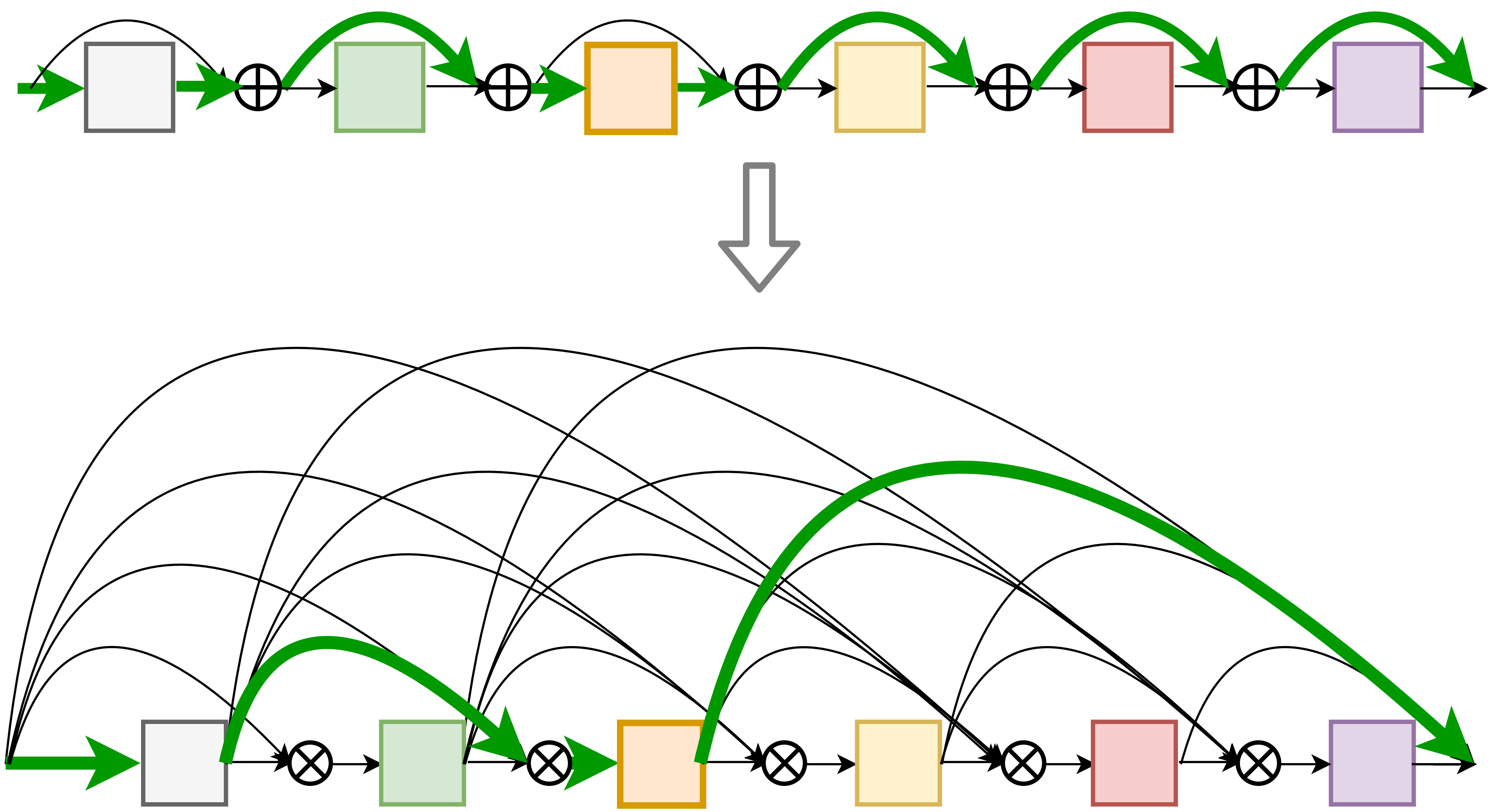
- No, the number of paths in DenseNet and ResNet have similar patterns.

- Because no consecutive skip connections can be taken.



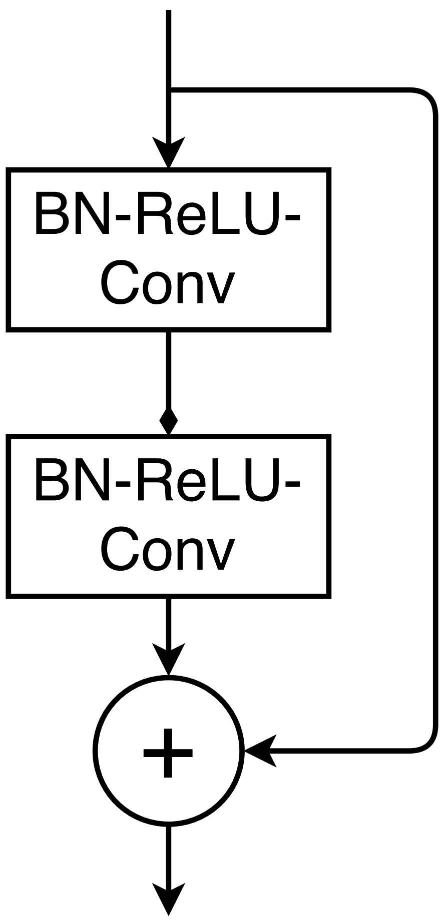
Compare Dense & Res

- There's a bijection between paths of DenseNet and paths of ResNet.



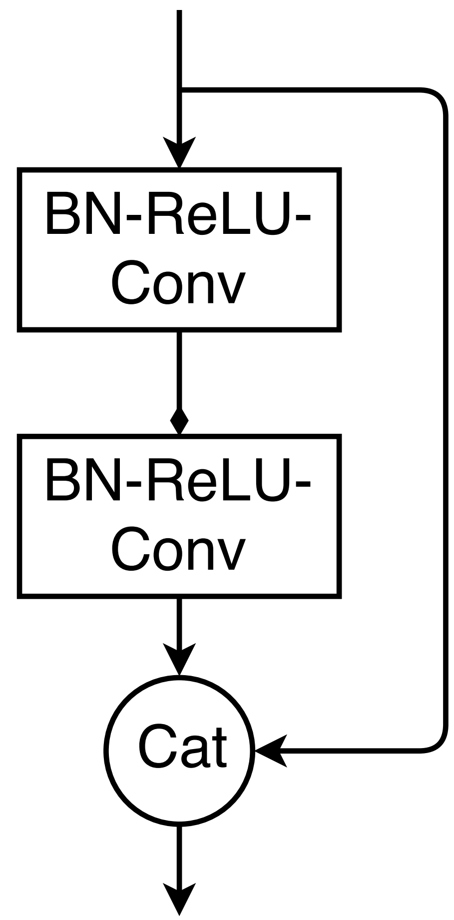
Aggregation View

- So, what makes Dense better?




```
# ResNet pre-activation
def ResidualBlock(x):
    x1 = BN_ReLU_Conv(x)
    x2 = BN_ReLU_Conv(x1)
    return x + x2

for i in range(N):
    model.add(ResidualBlock)
```

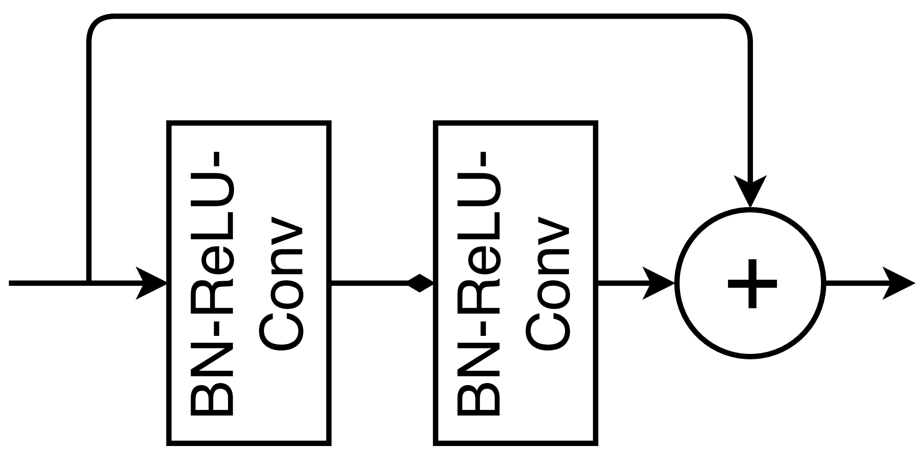


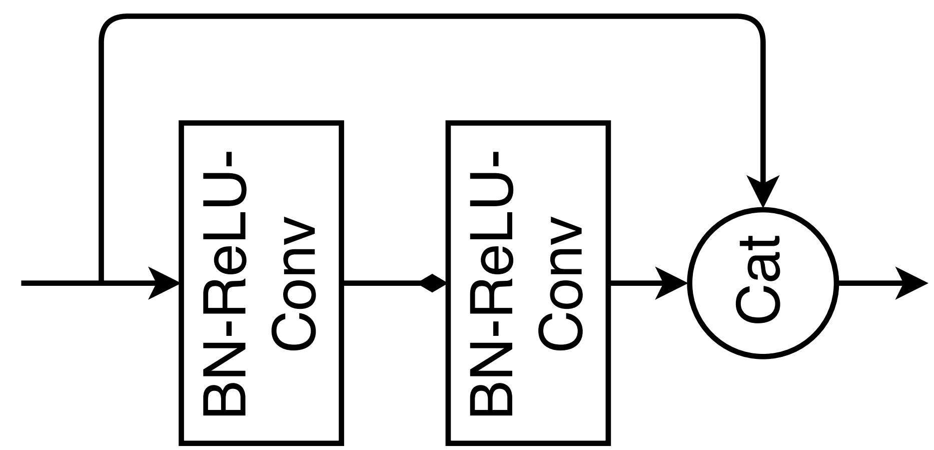

```
# DenseNet BC structure
def DenseBlock(x):
    x1 = BN_ReLU_Conv(x)
    x2 = BN_ReLU_Conv(x1)
    return Concat([x, x2])

for i in range(N):
    model.add(DenseBlock)
```

Aggregation View

- Features are densely aggregated in both Res and Dense.



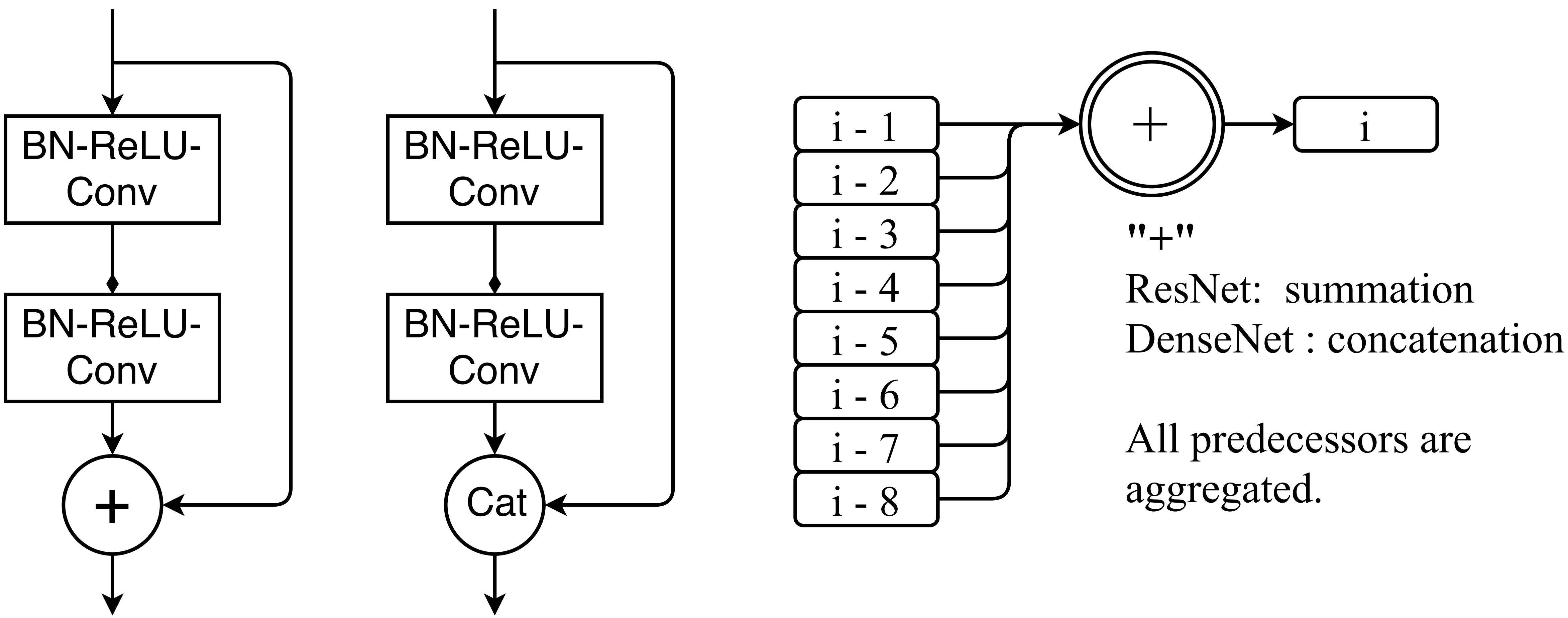


$$\begin{aligned} x_{\ell+1} &= F_\ell(x_\ell) + x_\ell \\ &= F_\ell(x_\ell) + F_{\ell-1}(x_{\ell-1}) + x_{\ell-1} \\ &= F_\ell(x_\ell) + F_{\ell-1}(x_{\ell-1}) + ... + F_1(x_1) \\ &= y_{\ell-1} + y_{\ell-2} + ... + y_1. \end{aligned}$$

$$\begin{aligned} x_{\ell+1} &= F_\ell(x_\ell) \oplus x_\ell \\ &= F_\ell(x_\ell) \oplus F_{\ell-1}(x_{\ell-1}) \oplus x_{\ell-1} \\ &= F_\ell(x_\ell) \oplus F_{\ell-1}(x_{\ell-1}) \oplus ... \oplus F_1(x_1) \\ &= y_{\ell-1} \oplus y_{\ell-2} \oplus ... \oplus y_1. \end{aligned}$$

Aggregation View

- Features are densely aggregated in both Res and Dense.



Aggregation View

- Concatenation is a better way of aggregation.

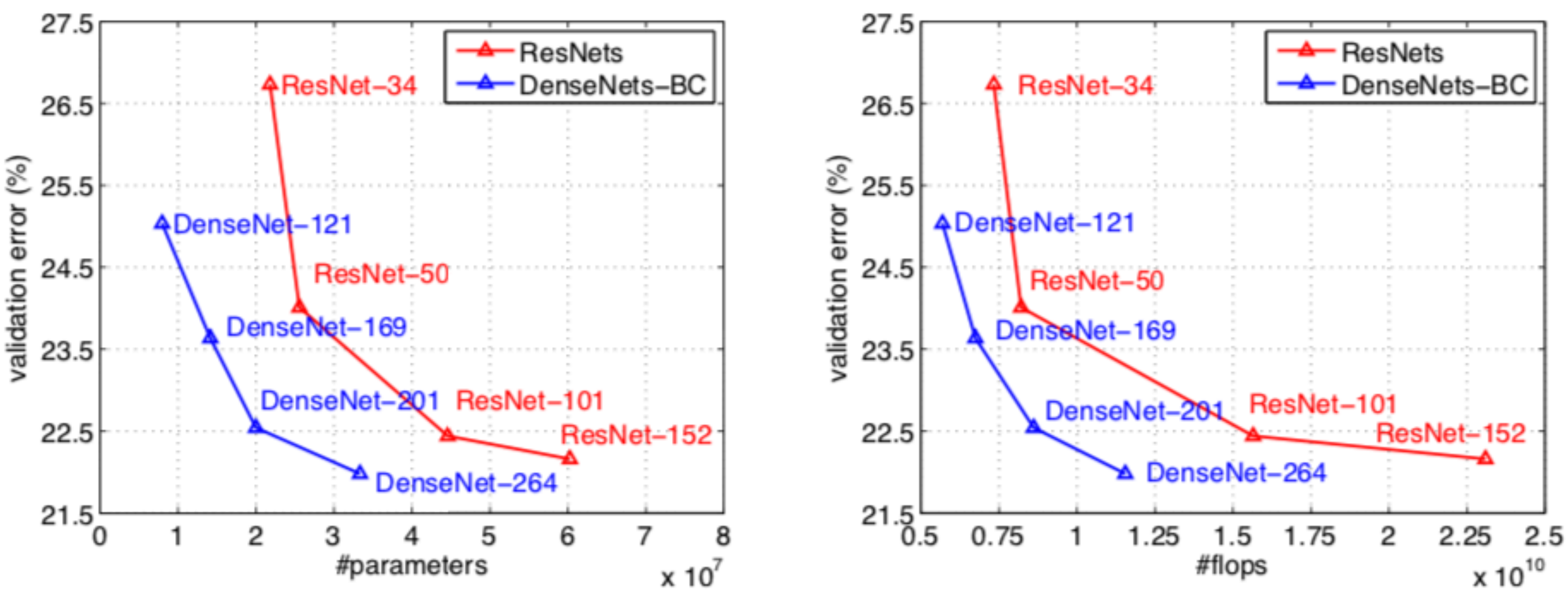


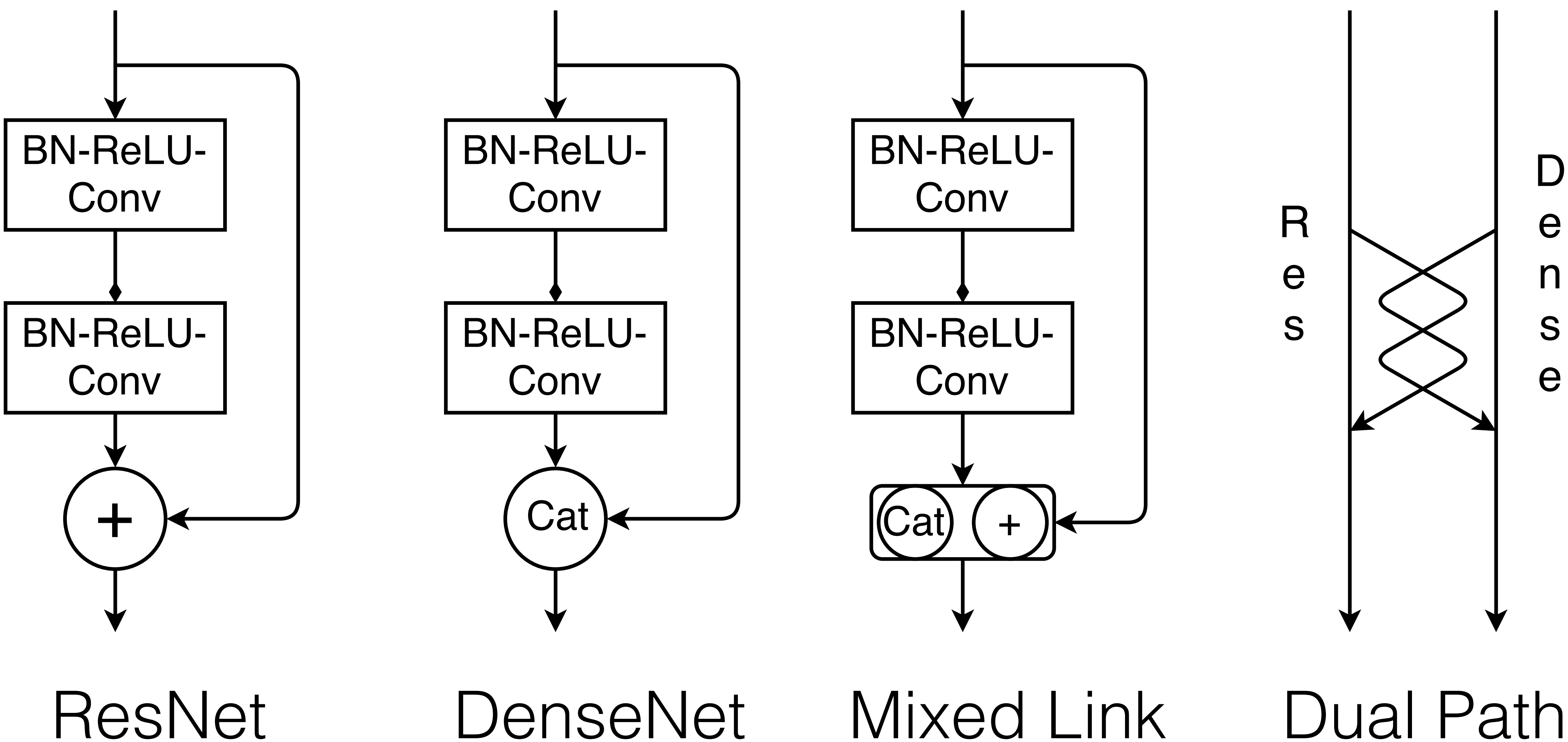
Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

Aggregation View

- ResNet > Plain:
    - Utilize more previous layers
- DenseNet > ResNet
    - Concatenation is a better way of aggregation.

Aggregation View

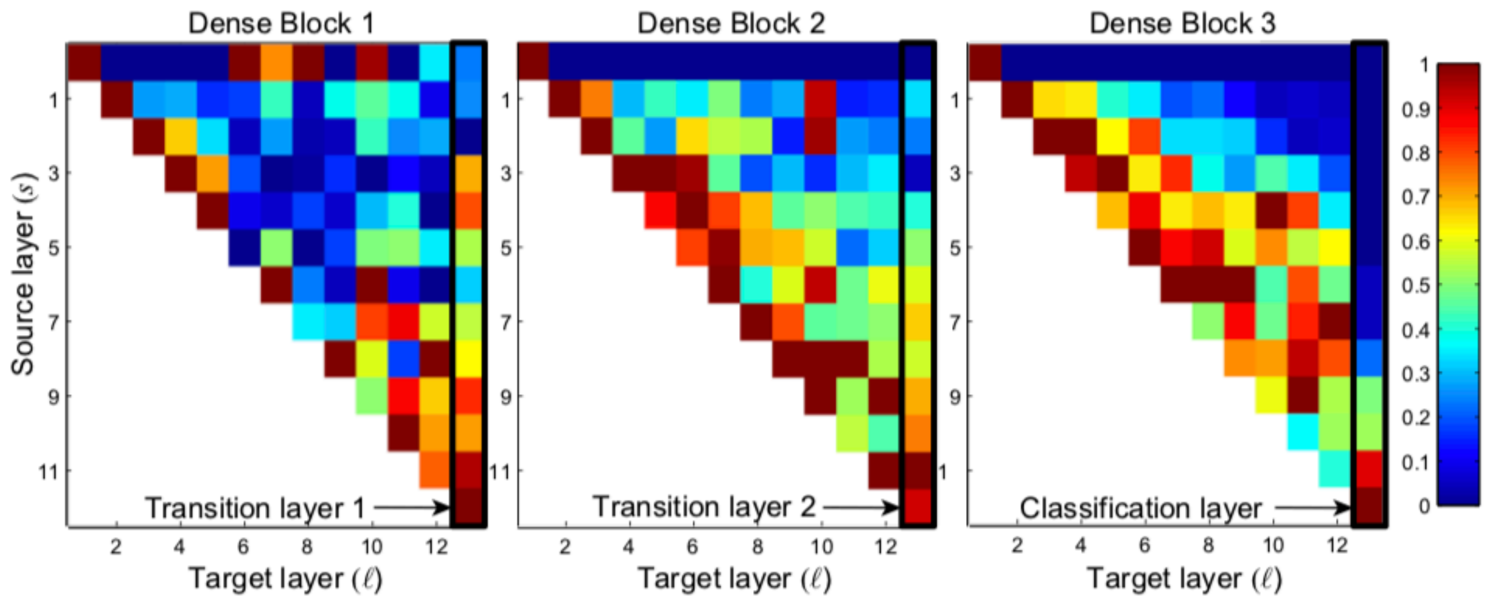
- More variations under aggregation view



Cons of Concatenation

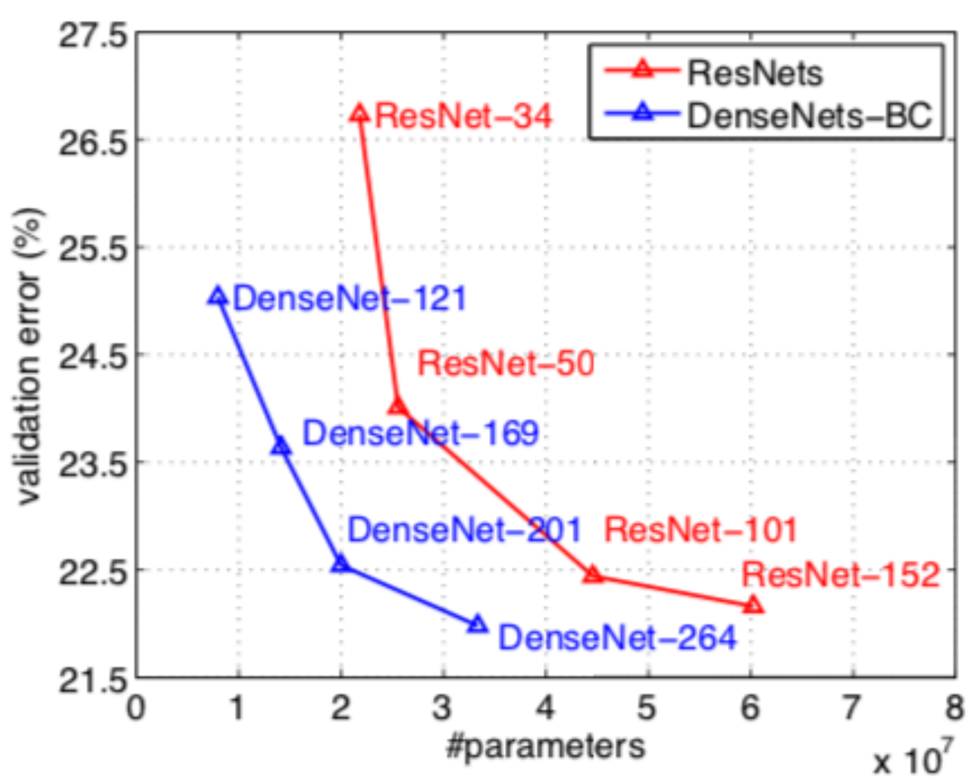
- Disadvantage :
    - Exploding parameters in deep networks-> O(n^2)
    - Redundant inputs in deeper layers

Dense-40-12	1.0M
Dense-100-12	7.0M
Dense-100-24	27.2M



Cons of Summation

- Disadvantage :
  - Information loss during aggregation



Cifar-10	param	error
Res-32	0.46M	7.51
Res-44	0.66M	7.17
Res-56	0.85M	6.97
Res-110	1.7M	**6.43**
Res-1202	19.4M	7.93

Thinking on Cat and Sum

- ResNet and DenseNet are both dense aggregation structure.
- Summation appears to be powerful on gradients, **BUT**
    - Information loss leads to parameter deficiency
- Concat is a better way of aggregations, **BUT**
    - Blowing params and redundancy
- Any way to utilize both advantages without bringing new troubles?

Thinking on Cat and Sum

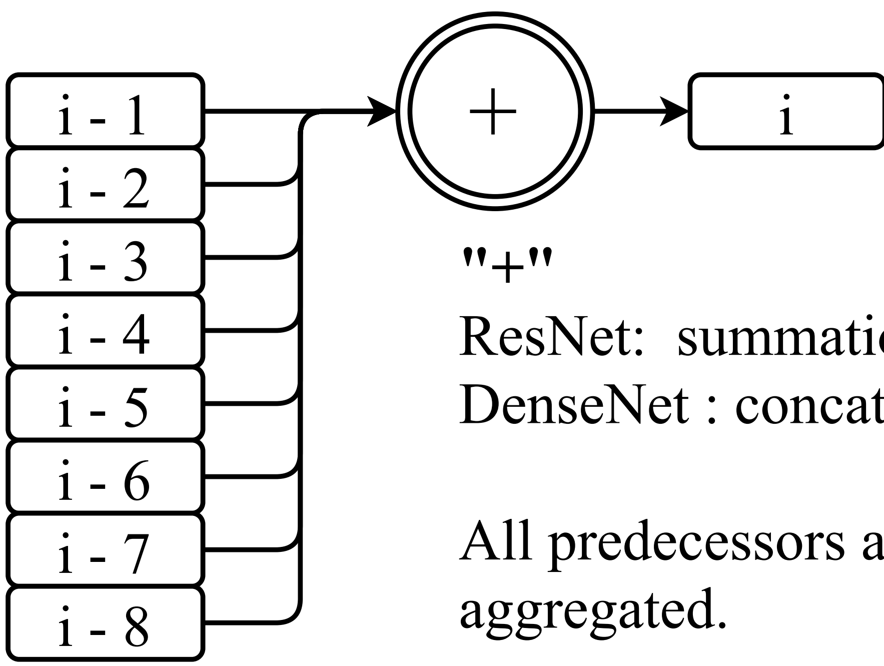
- Improvement on aggregation operators?
    - Combine both ? (Mixed link and dual path)
    - Others operators, e.g. + - * % mod
- Improvement on aggregation pattern?
    - Worthy trying

Our Goal

- Shortest gradient path between layers
  - Better than O(N) [plain]
  - Close to O(1) [ResNet and DenseNet]
- Connections / Params
  - Less than O(N^2) [DenseNet]
  - Close to O(N) [plain, ResNet]

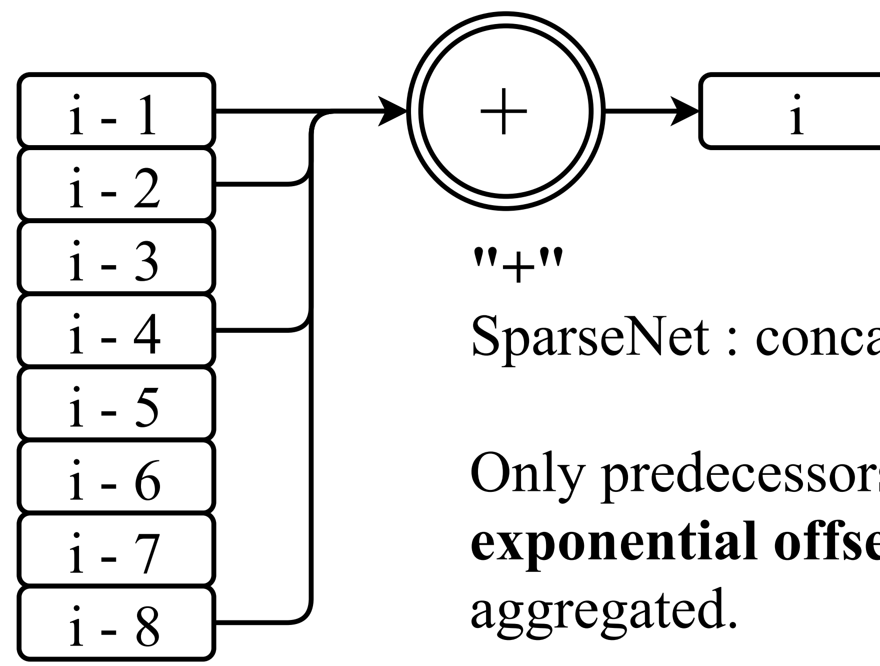
SparseNet

- Use concatenation as aggregation
- Only gather layers with exponential offsets



"+"
ResNet: summation
DenseNet : concatenation

All predecessors are aggregated.



"+"
SparseNet : concatenation

Only predecessors with **exponential offset** are aggregated.

SparseNet

- The total skip connections (params)

$$log_c 1 + log_c 2 + ... + log_c N = log_c N! \approx log_c N^N = O(NlgN)$$

- The gradient flow between any two layers

$$N \text{ offsets} => log_c N \times (c-1) \text{ steps}$$

- For example, when base is 2

$$23 \text{ offsets} => 10111_2 => 4 \text{ steps}$$

$$14 \text{ offsets} => 1110_2 => 3 \text{ steps}$$

SparseNet

- The best choice of base C

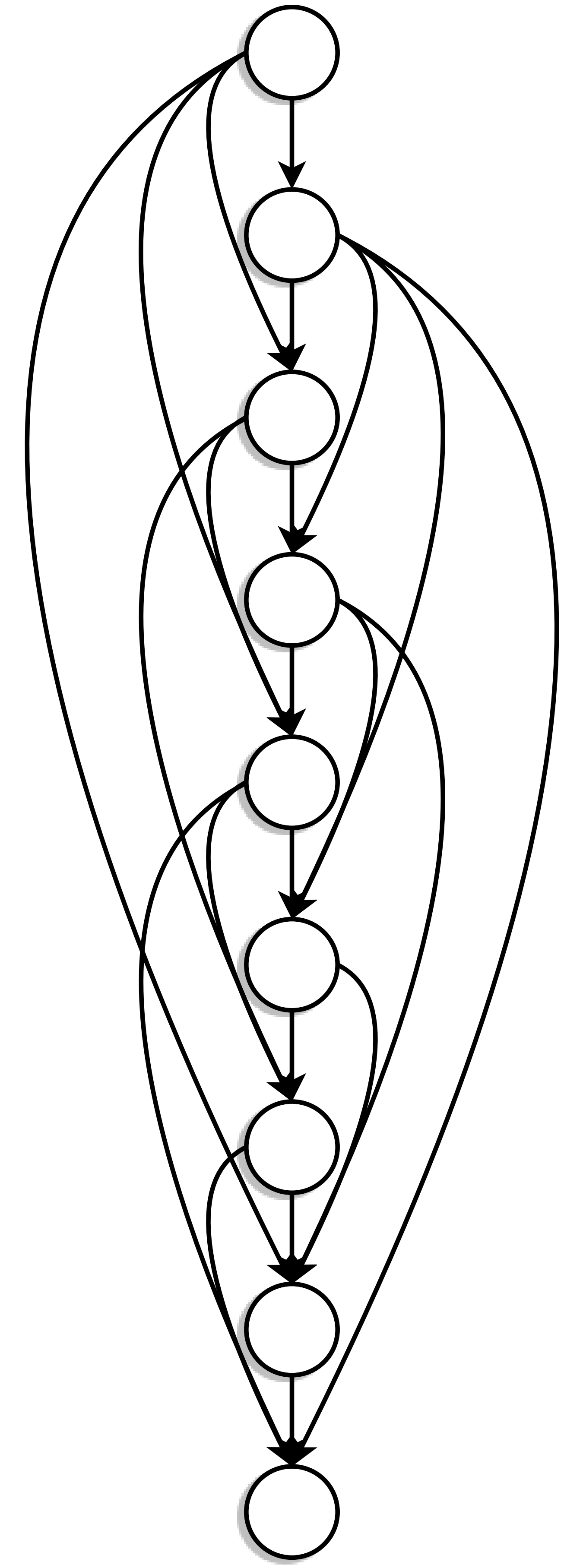
- The gradient path as short as possible

$$
\begin{aligned}
N \text{ offsets} &=> log_c N \times (C-1) \text{ steps} \\
&=> log_2 N \times \frac{(C-1)}{log_2 C} \text{ steps}
\end{aligned}
$$

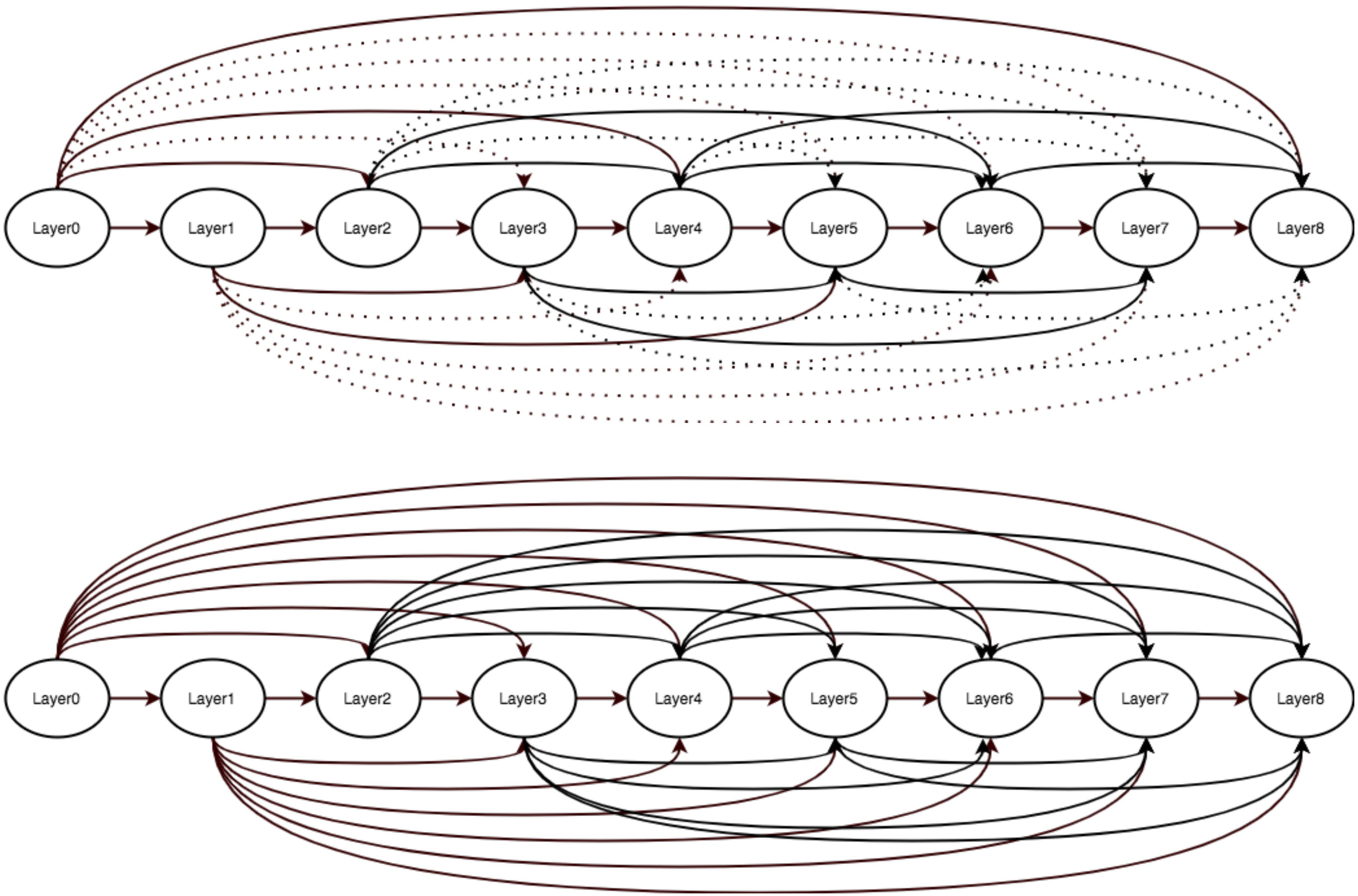
- So, we choose base 2

SparseNet

	Connections	Gradient Path
Plain	O(N)	N
ResNet	O(N * c)	1
DenseNet	O(N ^ 2)	1
SparseNet	O(N * lgN)	lgN

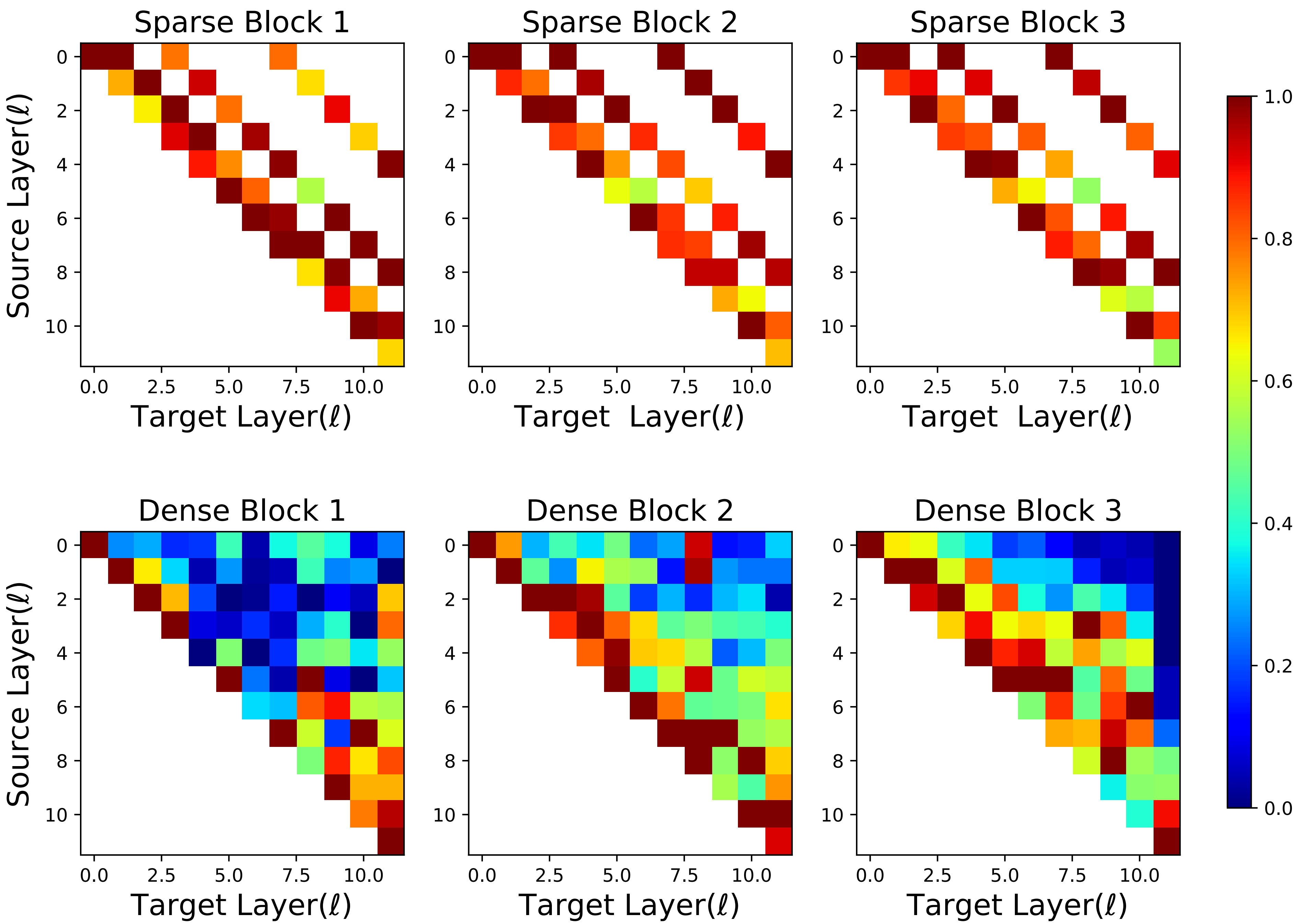


Sparse Compare with Dense



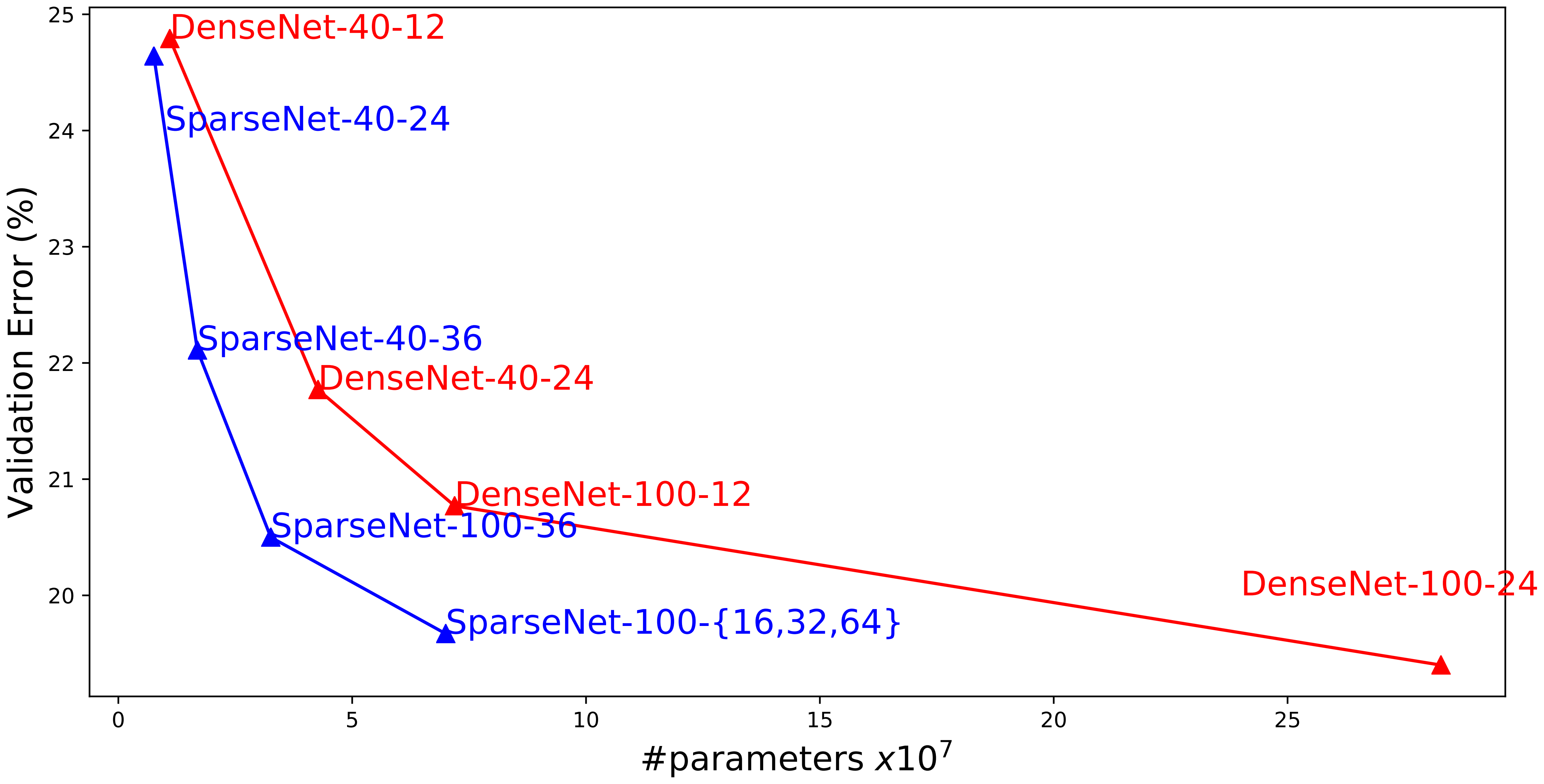
SparseNet

- Better params utilization (almost no redundancy)



SparseNet

- Better param efficiency (CIFAR)



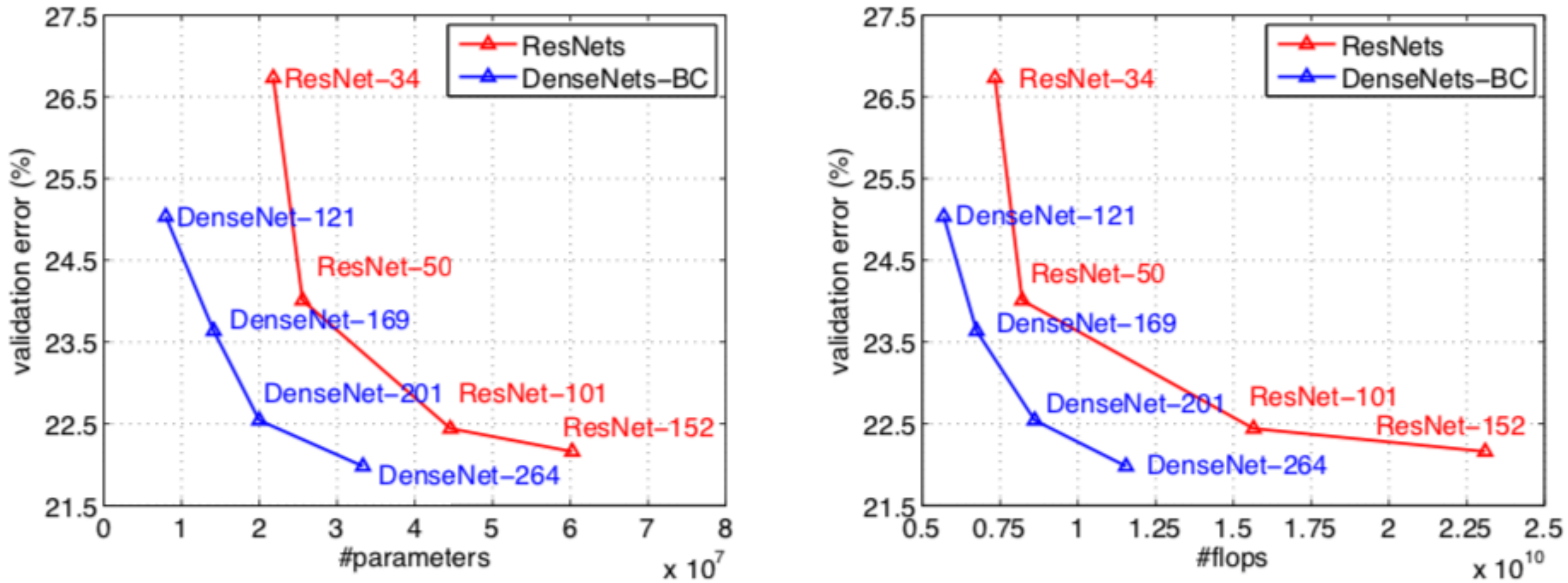
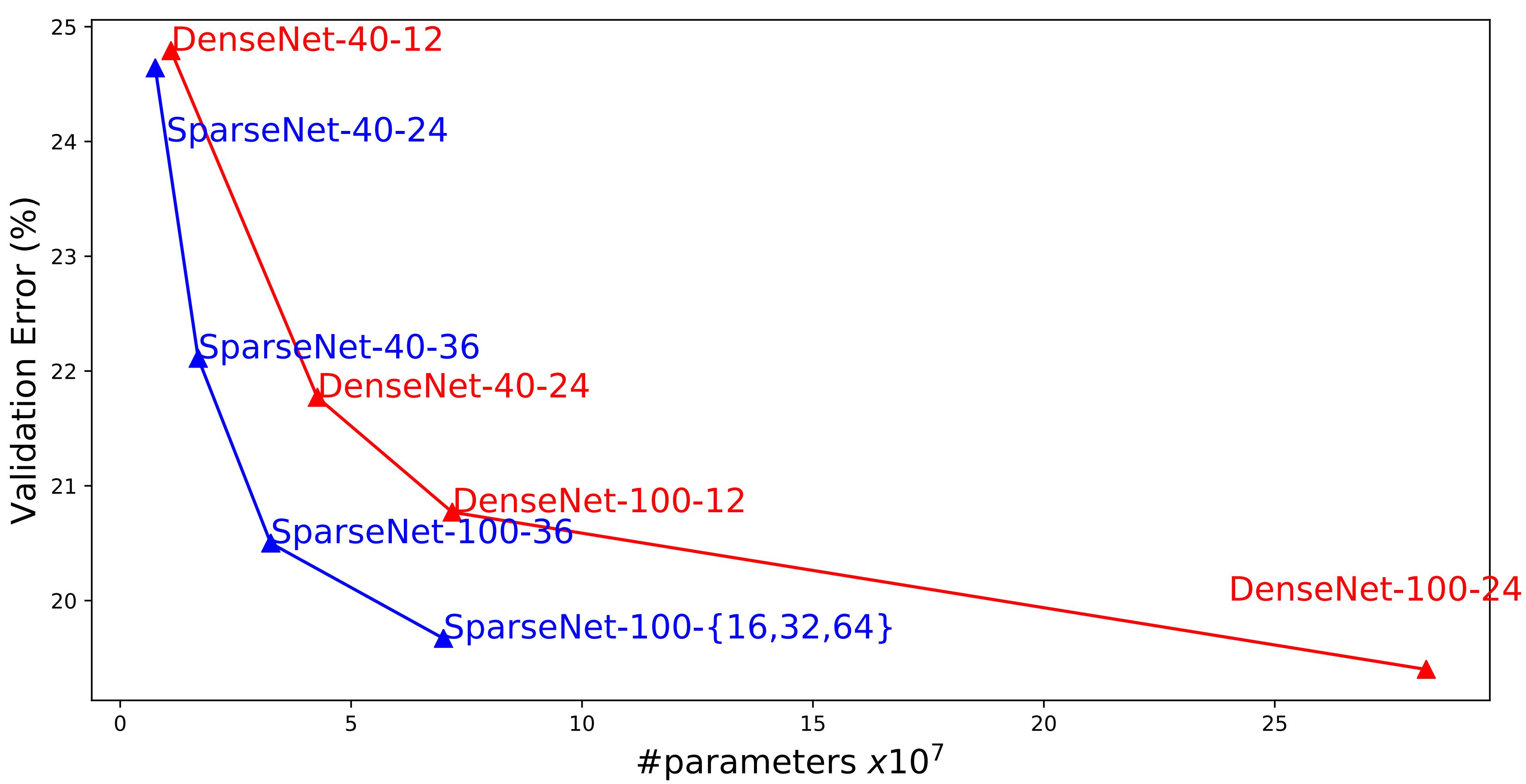


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).



Method	Depth	Params	C10+	C100+
ResNet [6]	110	1.7M	6.61	-
ResNet(pre-activation)[6]	164	1.7M	5.46	24.33
ResNet(pre-activation)[6]	1001	10.2M	4.62	22.71
Wide ResNet [22]	16	11.0M	4.81	22.07
Fractal [12]	21	38.6M	5.52	23.30
DenseNet (k=12)[7]	40	1.1M	5.39*	24.79*
DenseNet (k=12)[7]	100	7.2M	4.28*	20.97*
DenseNet (k=24)[7]	100	28.28M	4.04*	19.61*
DenseNet-BC (k=12)[7]	100	0.8M	4.68*	22.62*
DenseNet-BC (k=24)[7]	250	15.3M	**3.65**	17.6
DenseNet-BC (k=40)[7]	190	25.6M	3.75*	**17.53***
SparseNet (k=24)	40	0.76M	5.13	24.65
SparseNet (k=24)	100	2.52M	4.64	22.41
SparseNet (k=36)	100	5.65M	4.34	20.50
SparseNet (k=16, 32, 64)	100	7.22M	4.11	19.49
SparseNet (k=32, 64, 128)	100	27.72M	3.88	18.80
SparseNet-BC (k=24)	100	1.46M	4.03	22.12
SparseNet-BC (k=36)	100	3.26M	**3.91**	20.31
SparseNet-BC (k=16, 32, 64)	100	4.38M	-	19.71
SparseNet-BC (k=32, 64, 128)	100	16.72M	-	**17.71**

ImageNet

Model	Error	Params	FLOPs	Time(ms)
DenseNet-121-32	25.0*	7.98M	5.7	19.5
DenseNet-169-32	**23.6***	**14.15M**	6.76	32.0
DenseNet-201-32	22.5*	20.01M	8.63	42.6
SparseNet-121-32	25.6	4.51M	3.46	13.5
SparseNet-169-32	24.2	6.23M	3.74	18.8
SparseNet-201-32	**23.1**	**7.22M**	4.13	22.0

Model	Error	Params	FLOPs	Time(ms)
DenseNet-121-32	25.0*	7.98M	5.7	19.5
DenseNet-169-32	**23.6***	**14.15M**	6.76	32.0
DenseNet-201-32	22.5*	20.01M	8.63	42.6
SparseNet-121-32	25.6	4.51M	3.46	13.5
SparseNet-169-32	24.2	6.23M	3.74	18.8
SparseNet-201-32	**23.1**	**7.22M**	4.13	22.0

Network	Top-1 Error	Top-5 Error	Parameters	Pruning Rate
LeNet-300-100	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**×
LeNet-5	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**×
AlexNet	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**×
VGG-16	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13**×
GoogleNet	31.14%	10.96%	7.0M	
GoogleNet Pruned	31.04%	10.88%	**2.0M**	**3.5**×
SqueezeNet	42.56%	19.52%	1.2M	
SqueezeNet Pruned	42.26%	19.34%	**0.38M**	**3.2**×
ResNet-50	23.85%	7.13%	25.5M	
ResNet-50 Pruned	23.65%	6.85%	**7.47M**	**3.4**×

SparseNet

- Analyze Res and Dense in an aggregation view.
- Propose a new aggregation style — **Sparse**
    - Parameters growth : O(nlgn)
    - Gradient between arbitrary layers : O(lgn)
    - Higher parameter efficiency
        - 1/3 ~ 1/5 compared to DenseNet
        - 1/5 ~ 1/15 compared to ResNet

Thank you!

— Ligeng Zhu