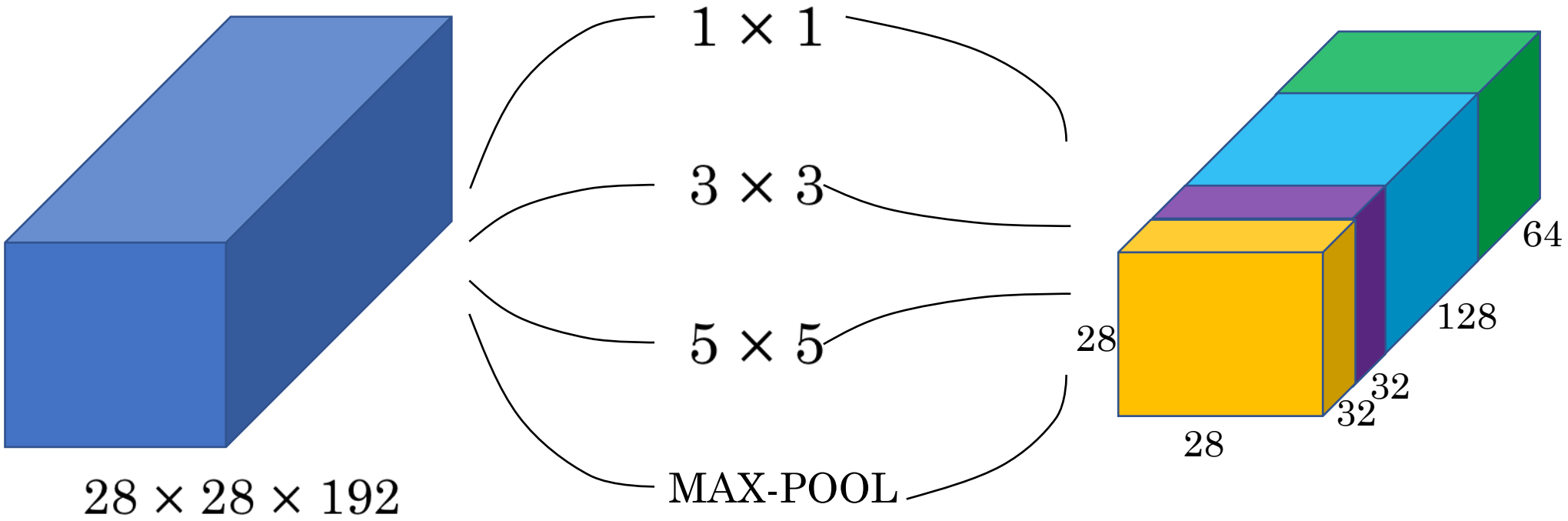
Estudos de Caso

Motivação para a Rede *Inception*

Motivação para a rede *Inception*

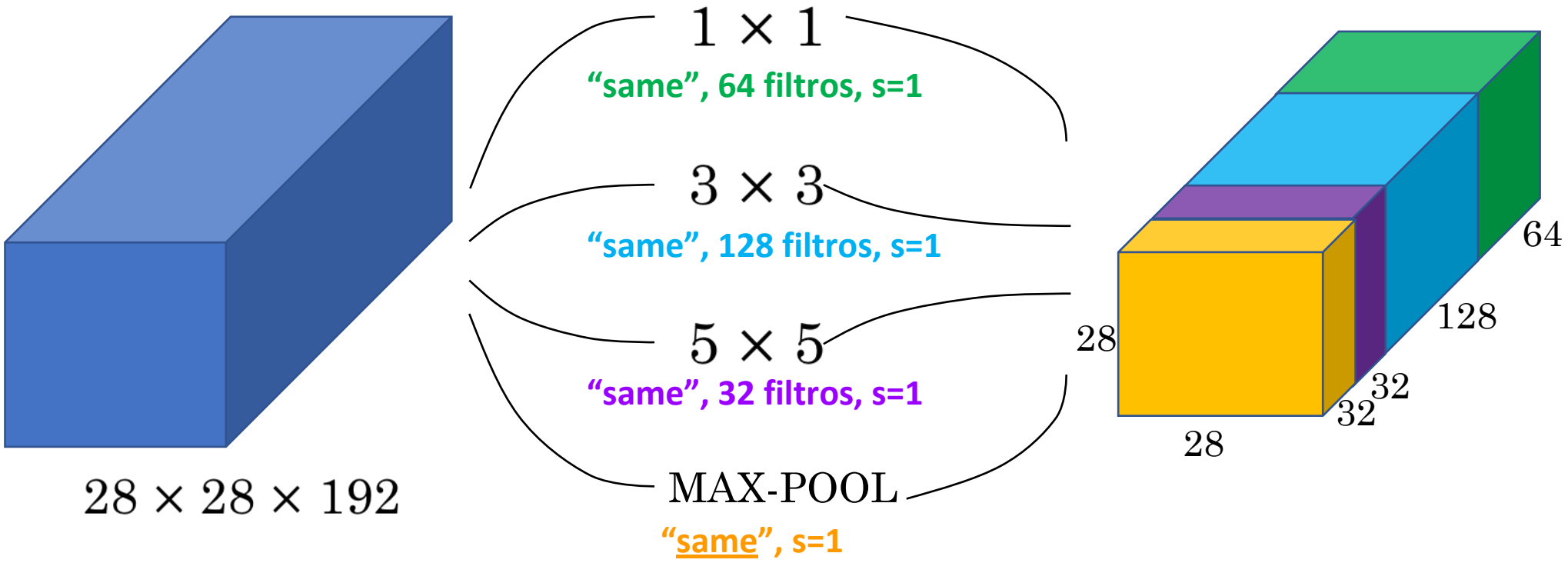
- Quando você cria uma CNN, você tem que decidir sobre todas as camadas
  - Você escolherá uma Conv 3 x 3 ou Conv 5 x 5 ou talvez uma camada de max *pooling*
  - Você tem tantas escolhas!
- O que a *inception* nos diz é: por que não usar todos eles de uma só vez?

Motivação para a rede *Inception*



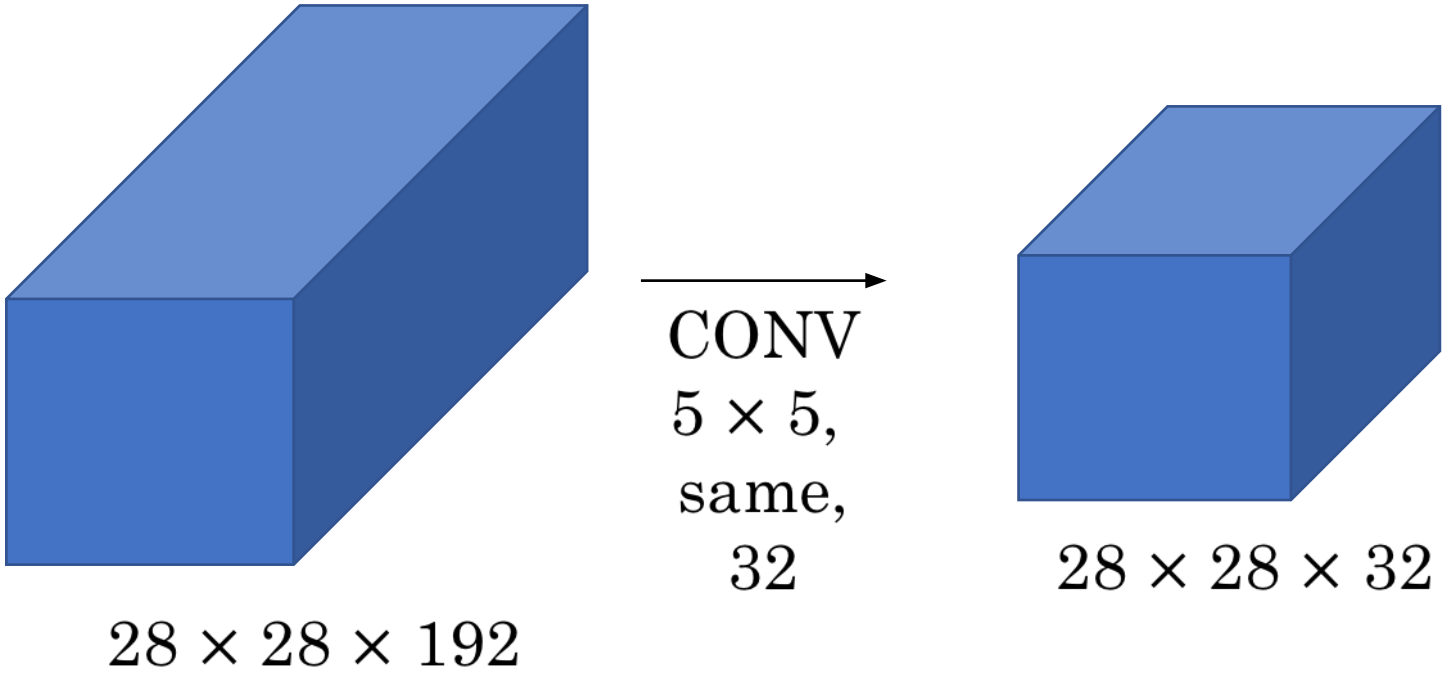
[Szegedy et al. 2014. Going deeper with convolutions]

Motivação para a rede *Inception*



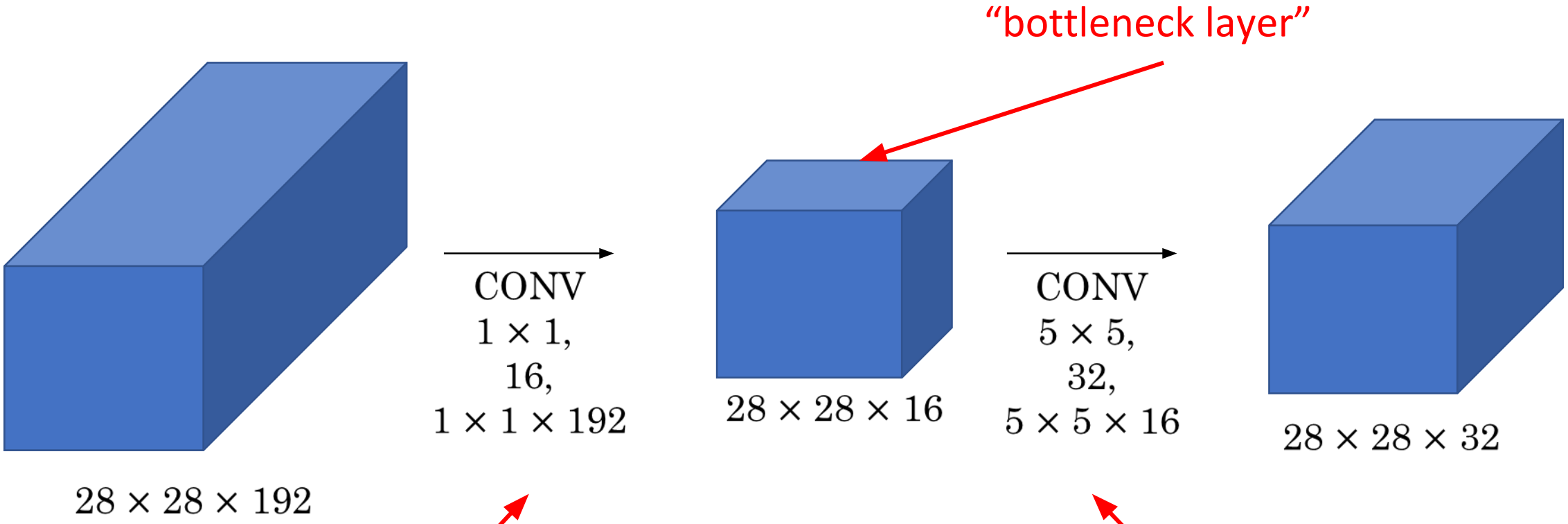
[Szegedy et al. 2014. Going deeper with convolutions]

O problema do custo computacional



Número de multiplicações:
(28 * 28 * 32) * (5 * 5 * 192) = 120M!

Usando convoluções 1x1



Número de multiplicações:
(28 * 28 * 16) * (1 * 1 * 192) = 2.4M

Número de multiplicações:
(28 * 28 * 32) * (5 * 5 * 16) = 10M

12.4M << 120M

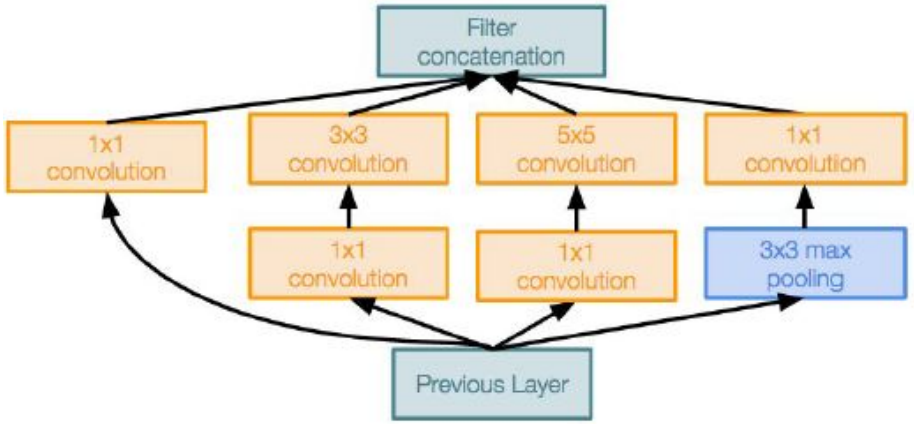
Estudos de Caso

Inception network (GoogLeNet)

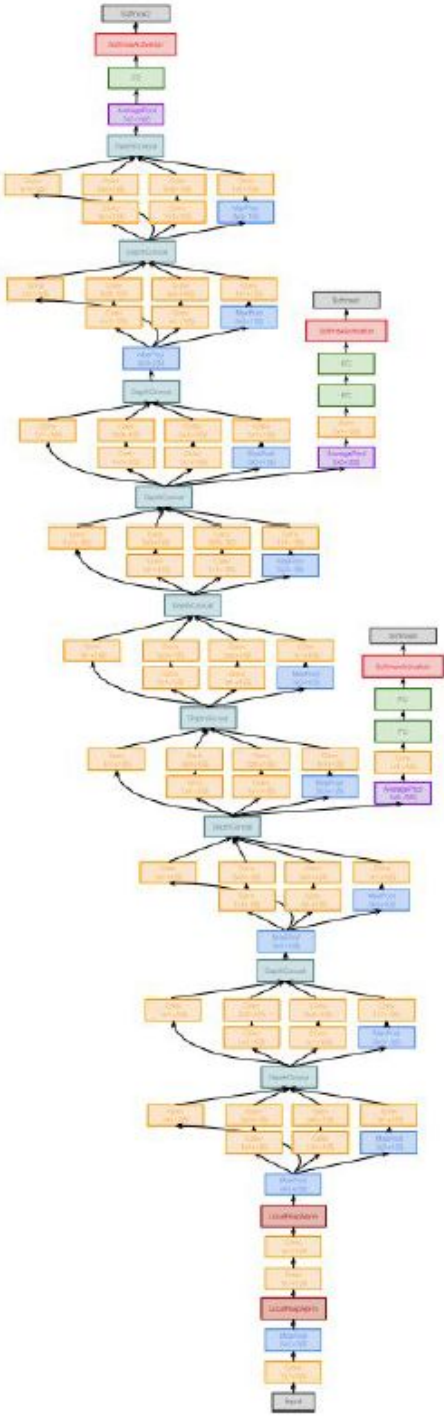
GoogLeNet

Redes mais profundas, mas com eficiência computacional

- 22 camadas
- Módulo “Inception” (eficiente)
- Sem camadas FC
- “Apenas” 5M parâmetros
  - 12x menos que a AlexNet
- Vencedor da tarefa de classificação do ILSVRC´14 (6.7% top 5 error)

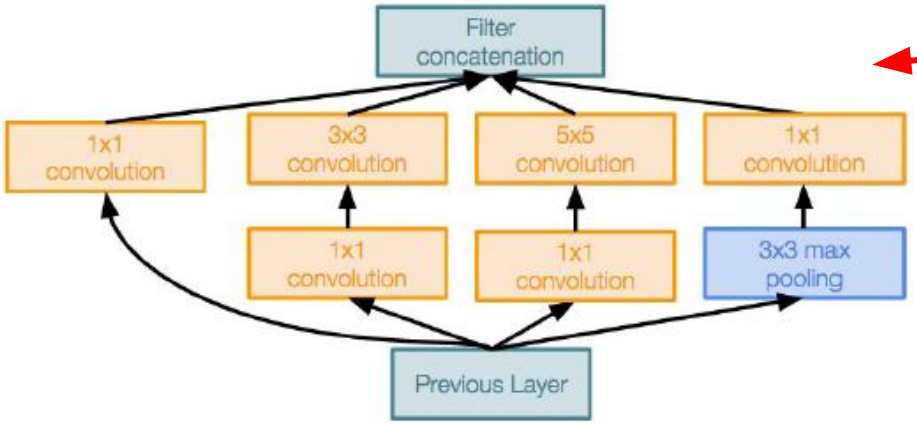


Módulo *inception*

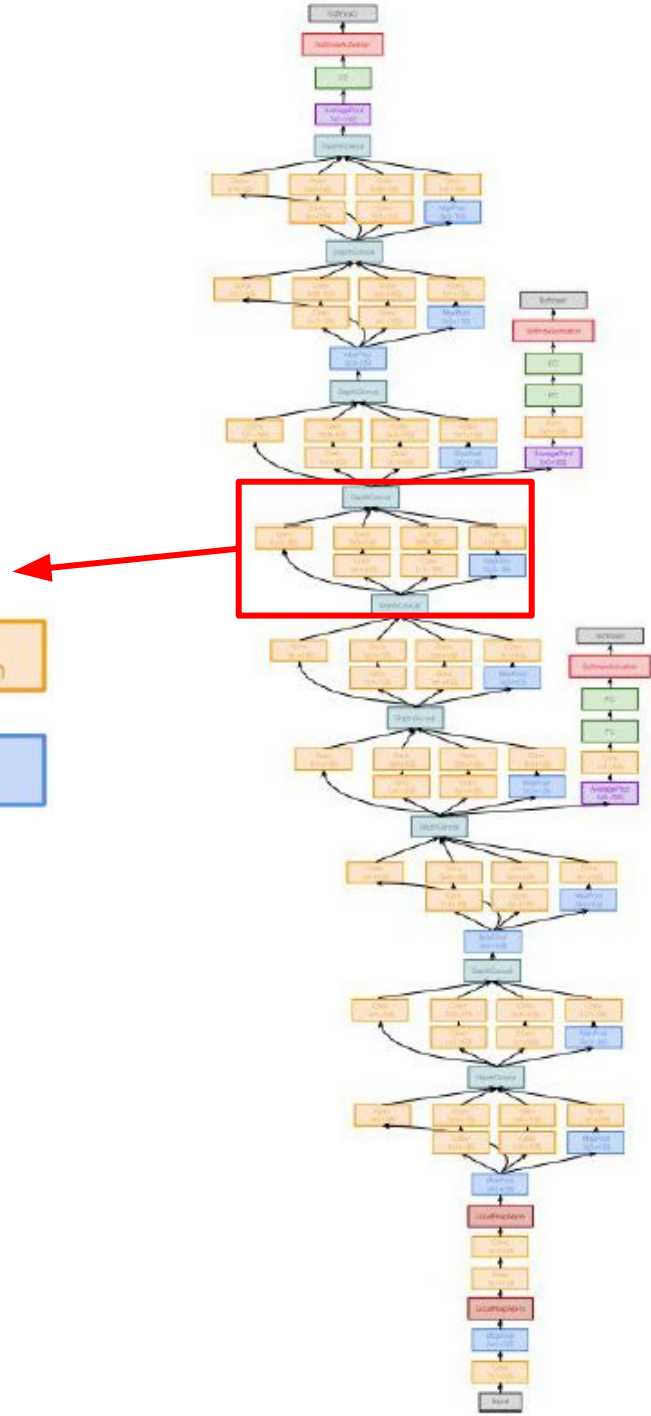


GoogLeNet

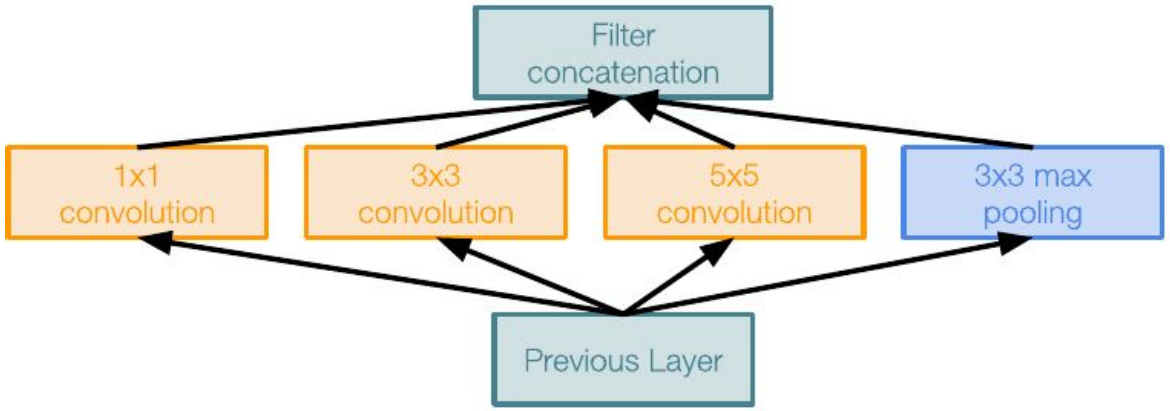
"Módulo *Inception*": projeto de uma boa topologia de rede local (*network within network*) e depois empilha esses módulos um em cima do outro



Módulo i*nception*



GoogLeNet



Naive Inception module

Aplica operações de filtros paralelas na entrada da camada anterior:

- Múltiplos tamanhos de campos receptivos para convolução (1x1, 3x3, 5x5)
- Operação de pooling (3x3)

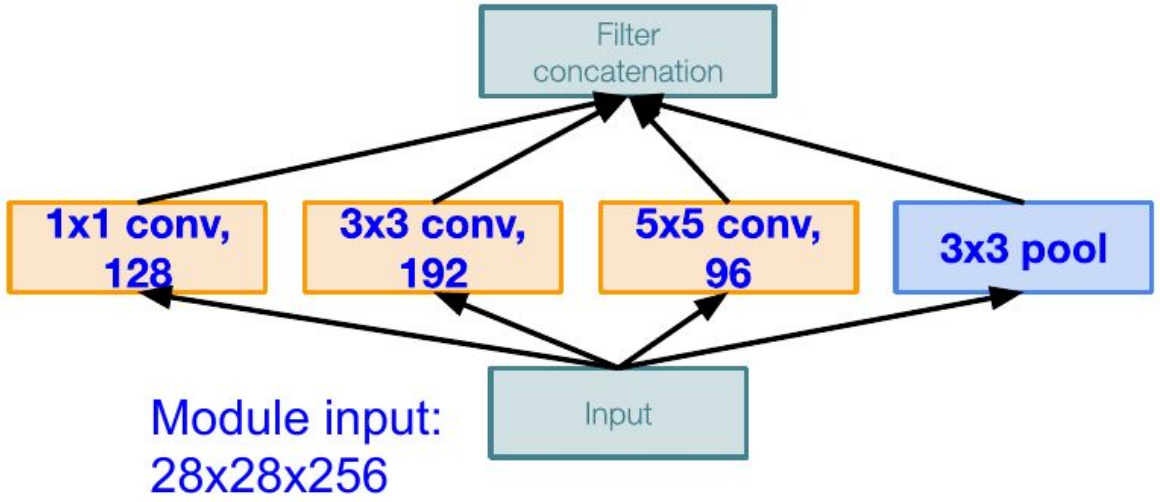
Empilha (concatena) todas as saídas dos filtros juntos, "*depth-wise*"

Q: Qual é o problema disso?

- Dica: complexidade computacional

GoogLeNet

Exemplo:



Naive Inception module

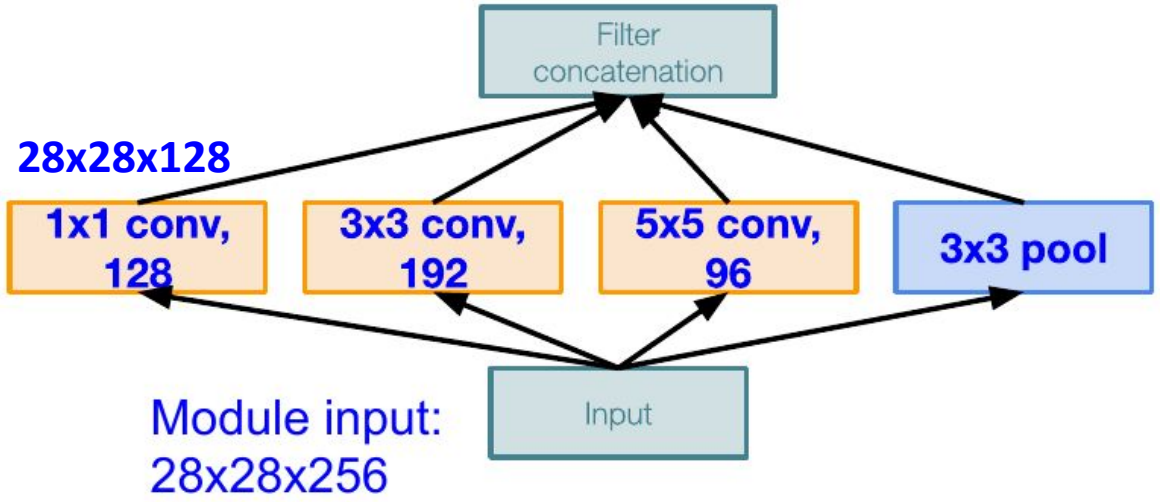
Q: Qual é o problema disso?

- Dica: complexidade computacional

Q1: Qual é o tamanho de saída da conv 1x1, com 128 filtros?

GoogLeNet

Exemplo:



Naive Inception module

Q: Qual é o problema disso?

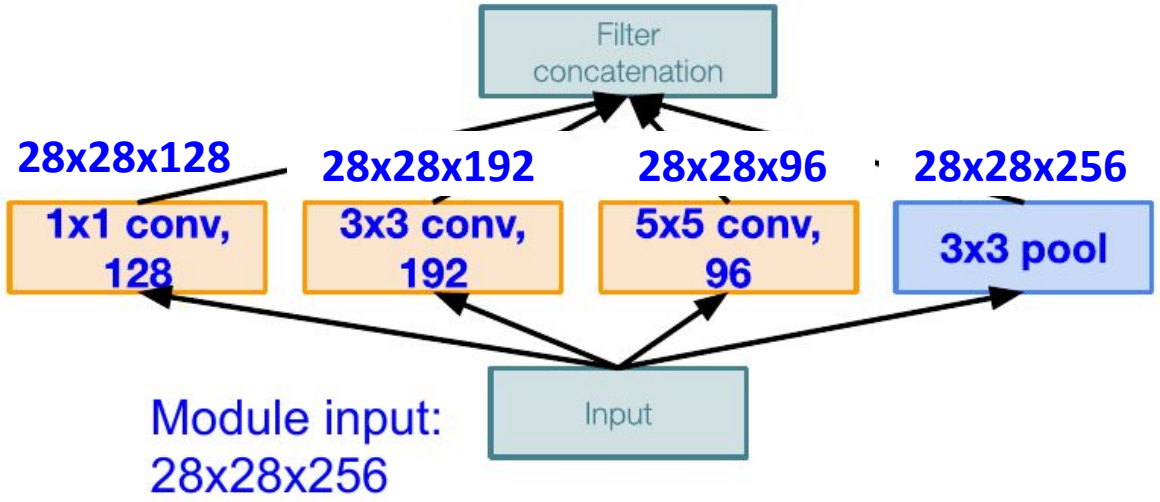
- Dica: complexidade computacional

Q1: Qual é o tamanho de saída da conv 1x1, com 128 filtros?

A1: 28x28x128

GoogLeNet

Exemplo:



Naive Inception module

Q: Qual é o problema disso?

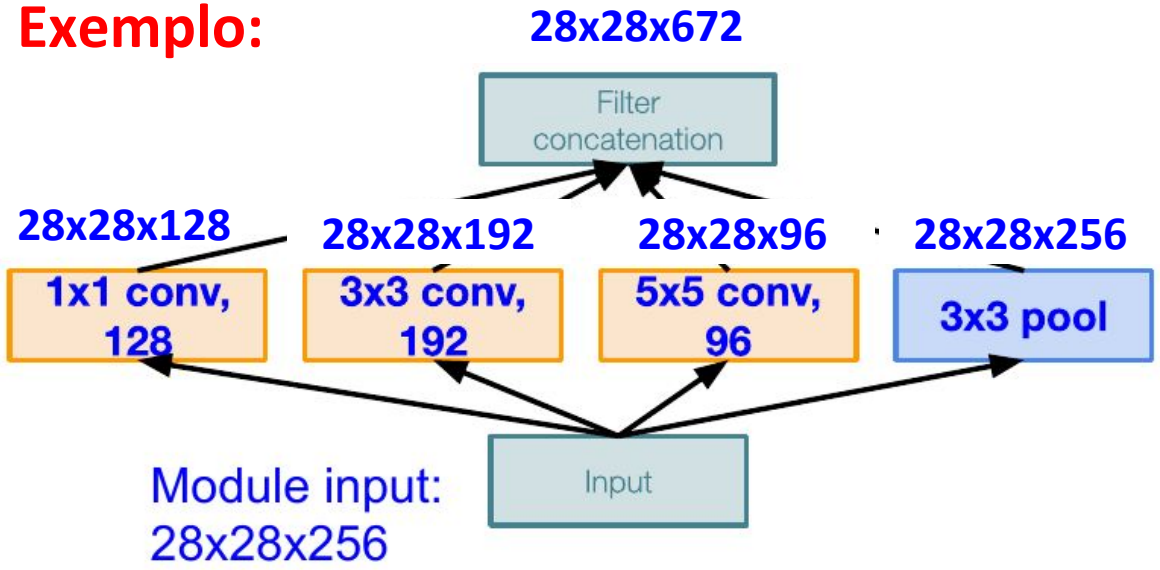
- Dica: complexidade computacional

Q1: Qual é o tamanho de saída da conv 1x1, com 128 filtros?

Q2: Qual é o tamanho das outras saídas?

GoogLeNet

Exemplo:



Naive Inception module

Q: Qual é o problema disso?

- Dica: complexidade computacional

Q1: Qual é o tamanho de saída da conv 1x1, com 128 filtros?

Q2: Qual é o tamanho das outras saídas?

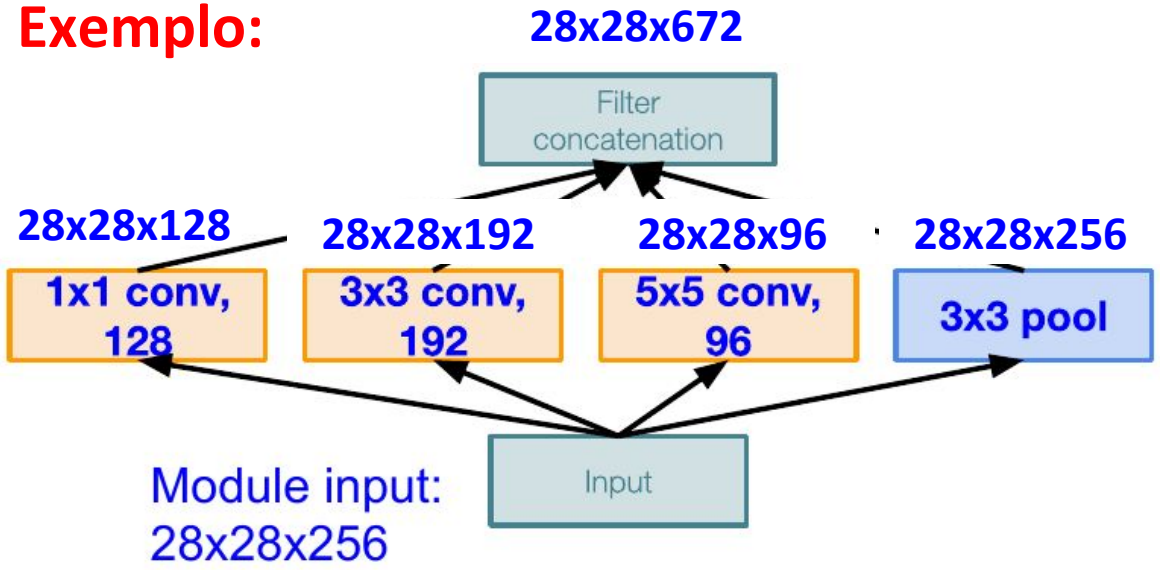
Q3: Qual é o tamanho da saída depois da concatenação?

A3: 28x28x(28 + 192 + 96 + 256)

= **28x28x672**

GoogLeNet

Exemplo:



Naive Inception module

Q1: Qual é o tamanho de saída da conv 1x1, com 128 filtros?

Q2: Qual é o tamanho das outras saídas?

Q3: Qual é o tamanho da saída depois da concatenação?

Q: Qual é o problema disso?

- Dica: complexidade computacional

Número de operações (*) nas convs:

[1x1 conv, 128]: 28*28*128*1*1*256

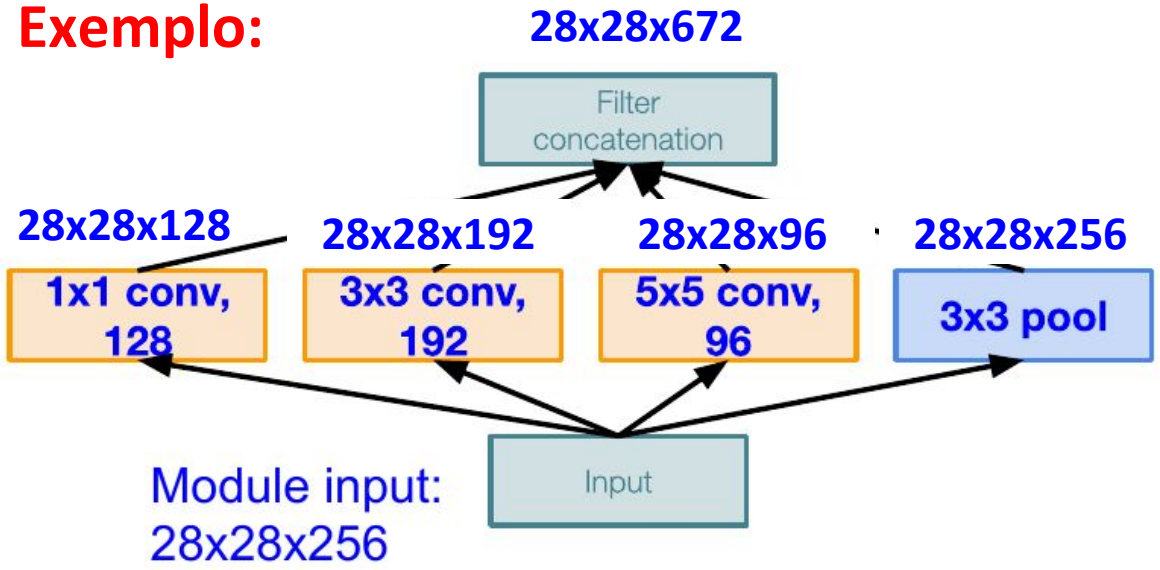
[3x3 conv, 192]: 28*28*192*3*3*256

[5x5 conv, 96]: 28*28*96*5*5*256

Total: 854M multiplicações

GoogLeNet

Exemplo:



Naive Inception module

Q: Qual é o problema disso?

- Dica: complexidade computacional

Número de operações (*) nas convs:

[1x1 conv, 128]: 28*28*128*1*1*256

[3x3 conv, 192]: 28*28*192*3*3*256

[5x5 conv, 96]: 28*28*96*5*5*256

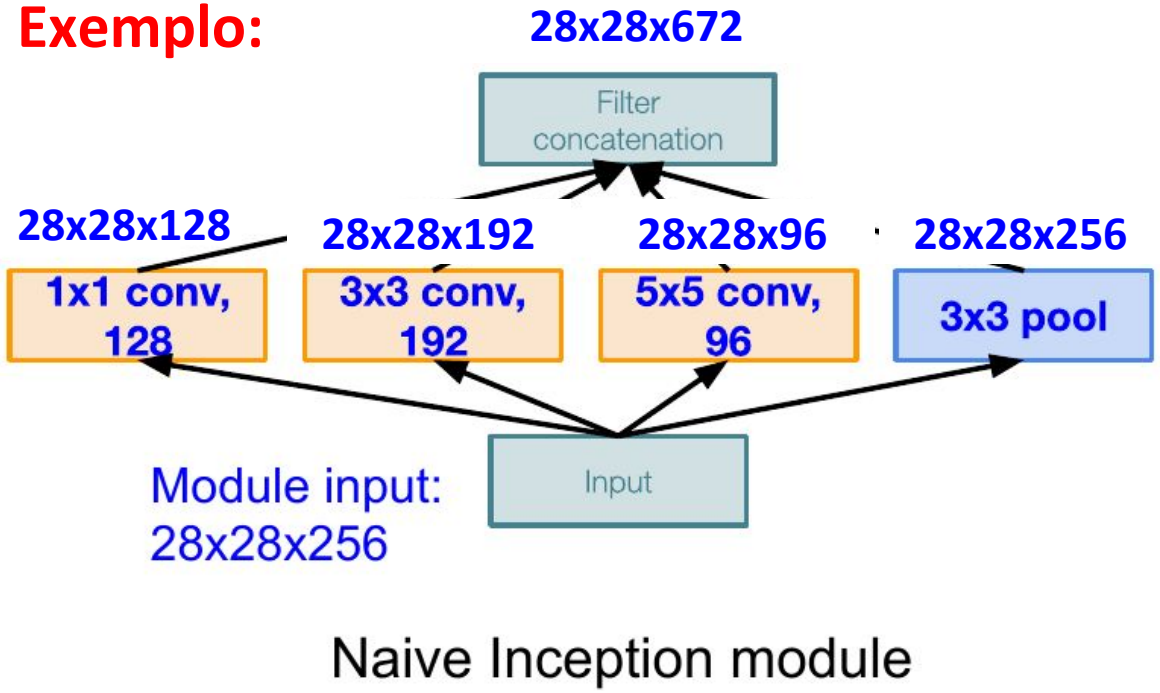
Total: 854M multiplicações

Muito caro de computar

Camadas de pooling também preservam a profundidade do volume de entrada, então a profundidade de saída pode apenas aumentar

GoogLeNet

Exemplo:



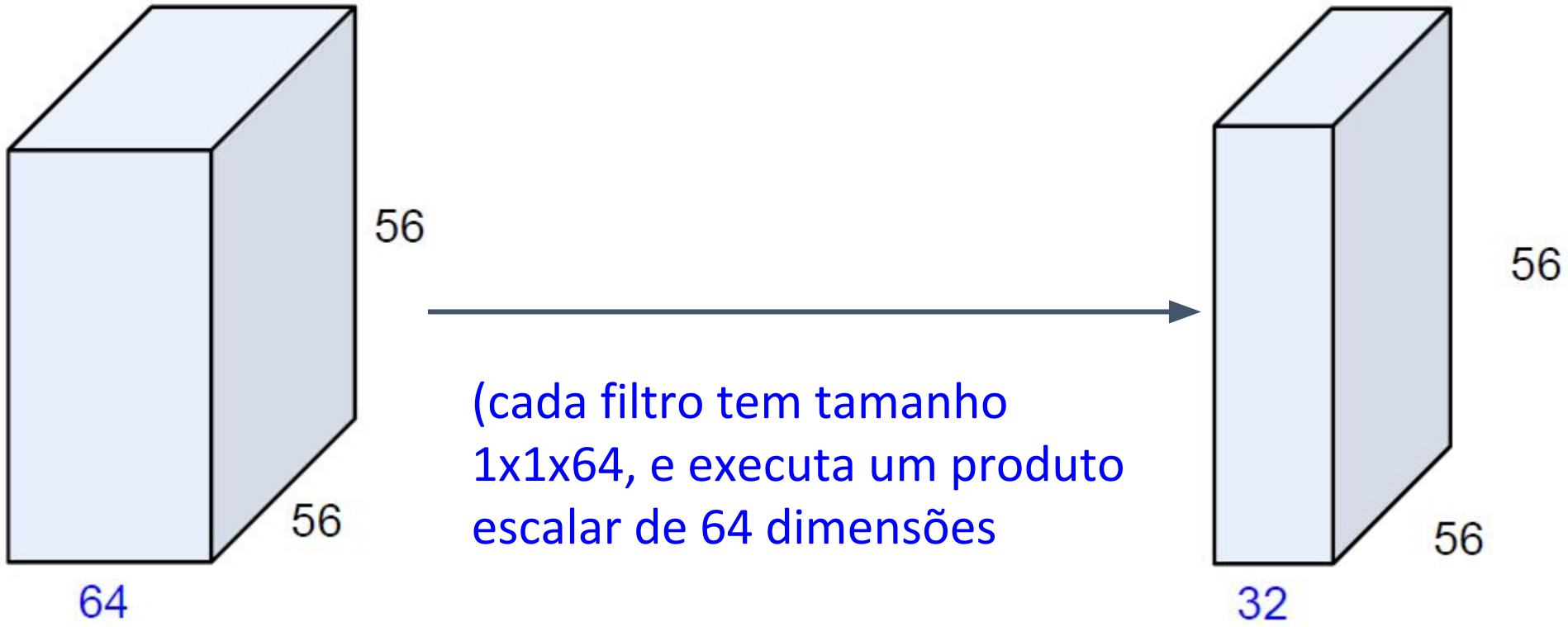
Naive Inception module

Q: Qual é o problema disso?

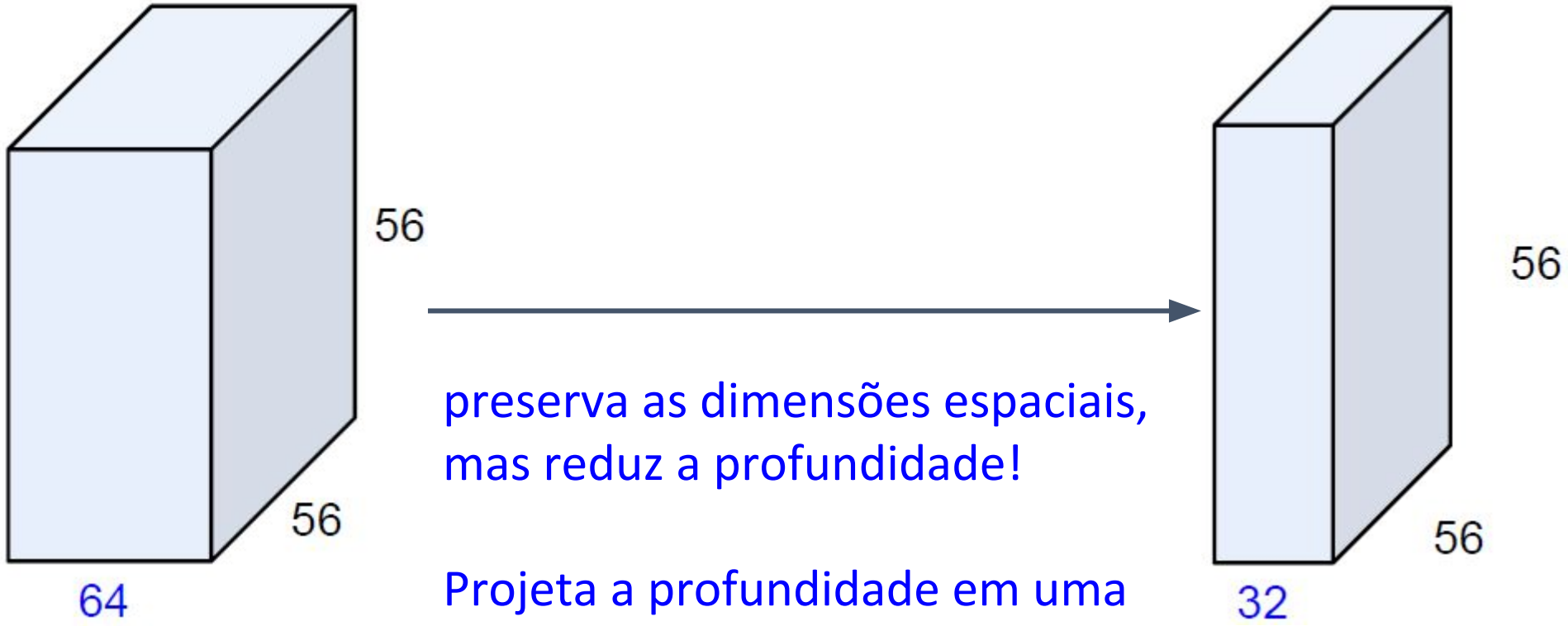
- Dica: complexidade computacional

Soulção: camadas “bottleneck” que usam convoluções 1x1 para reduzir a profundidade do volume de saída

Lembrete: convoluções 1x1



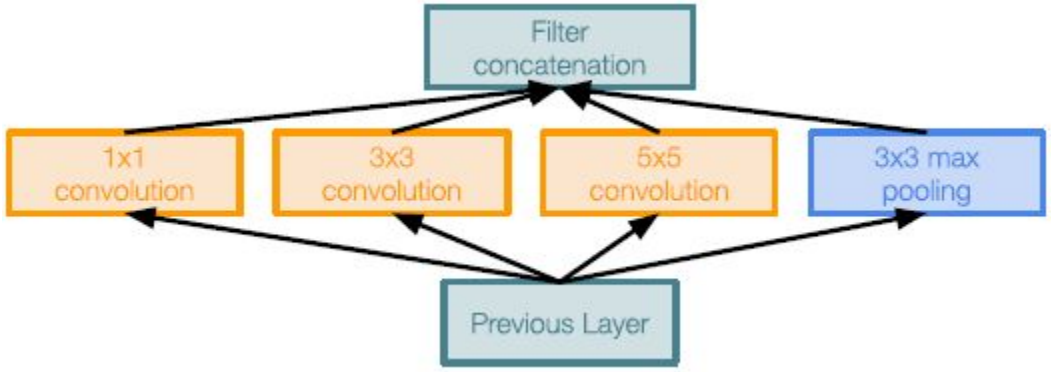
Lembrete: convoluções 1x1



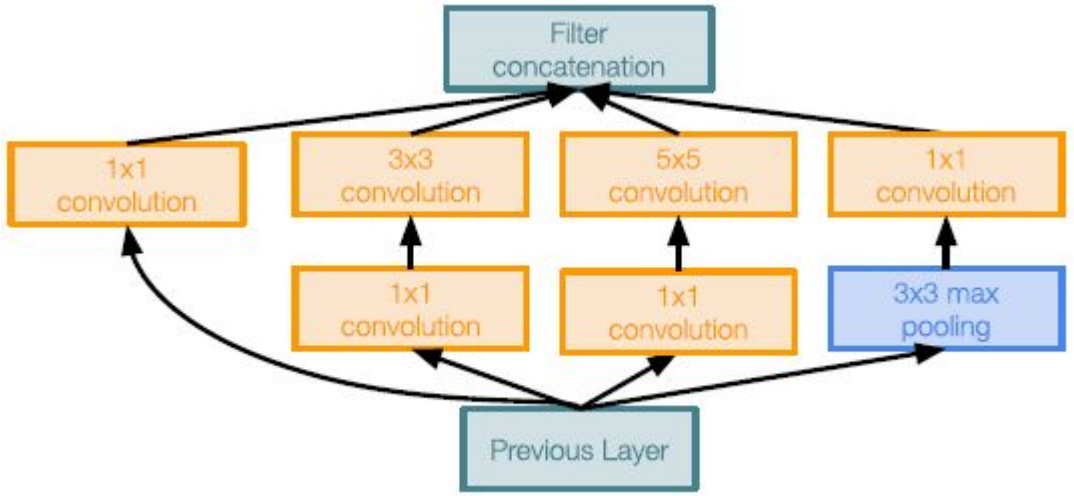
preserva as dimensões espaciais, mas reduz a profundidade!

Projeta a profundidade em uma dimensão menor (combinação dos mapas de features)

GoogLeNet



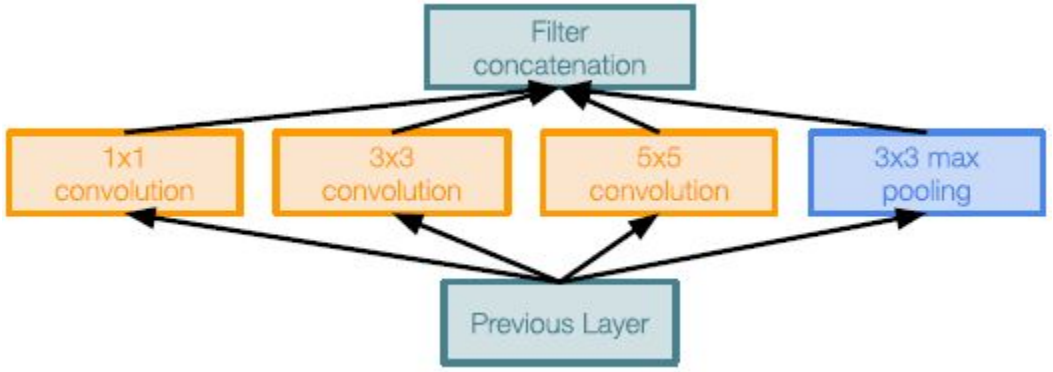
Módulo *inception* ingênuo
(*naive inception module*)



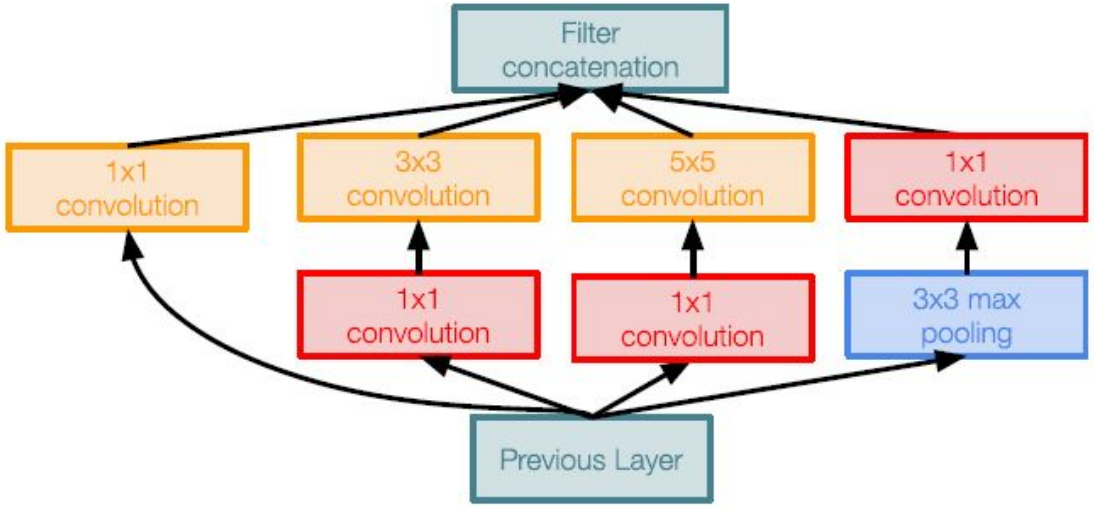
Módulo *inception* com
redução de dimensionalidade

GoogLeNet

camadas 1x1 conv bottleneck

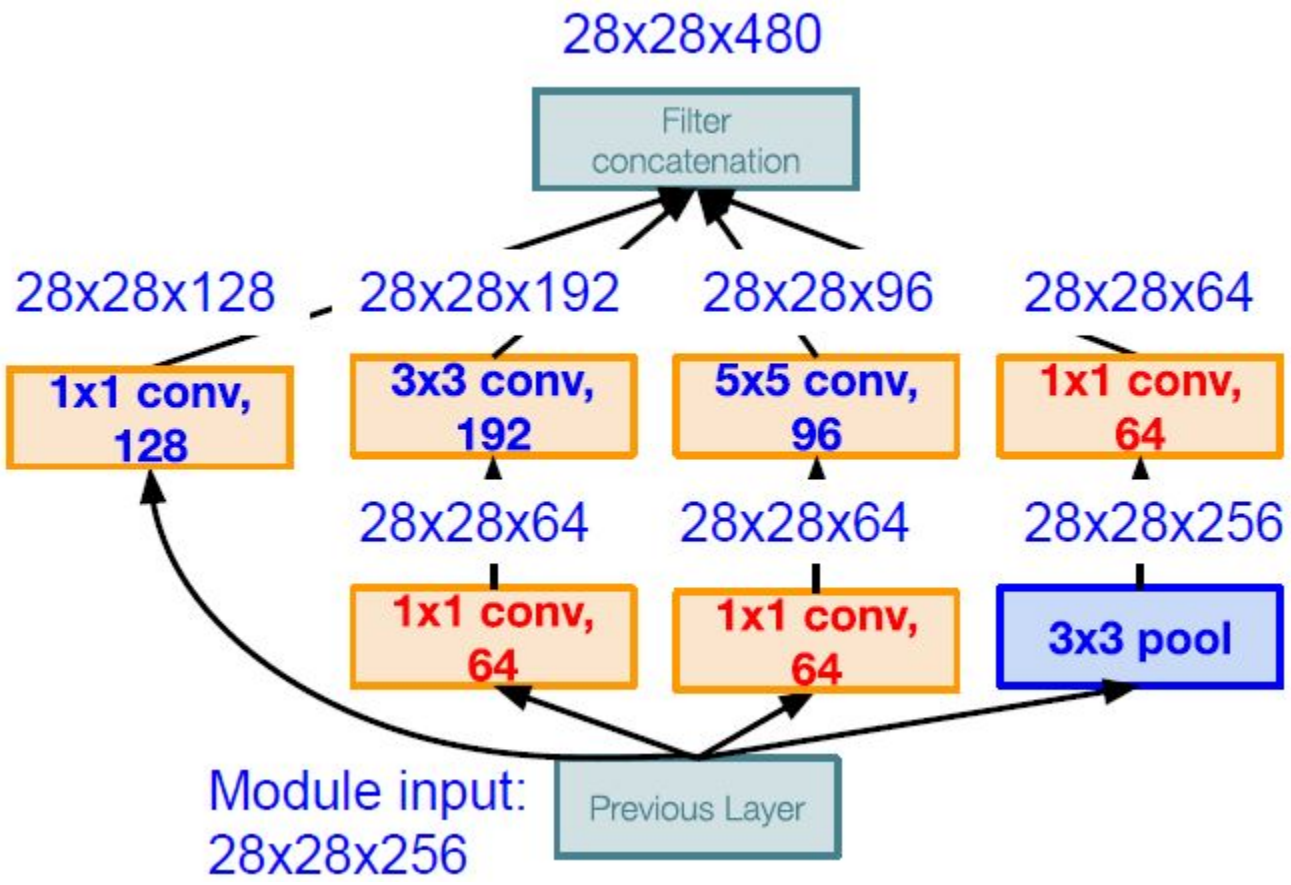


Módulo *inception* ingênuo
(*naive inception module*)



Módulo *inception* com
redução de dimensionalidade

GoogLeNet



Módulo *inception* com redução de dimensionalidade

Número de operações (*) nas convs:

[1x1 conv, 64]: 28*28*64*1*1*256

[1x1 conv, 64]: 28*28*64*1*1*256

[1x1 conv, 128]: 28*28*128*1*1*256

[3x3 conv, 192]: 28*28*192*3*3***64**

[5x5 conv, 96]: 28*28*96*5*5***64**

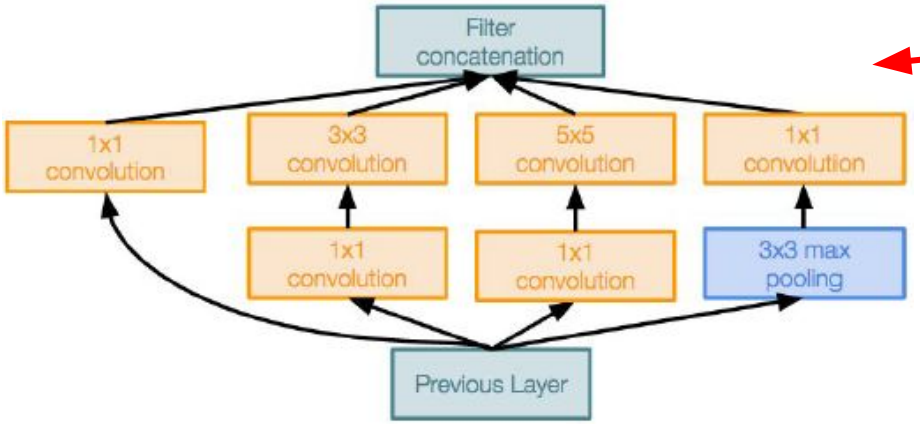
[1x1 conv, 64]: 28*28*64*1*1*256

Total: 358M operações

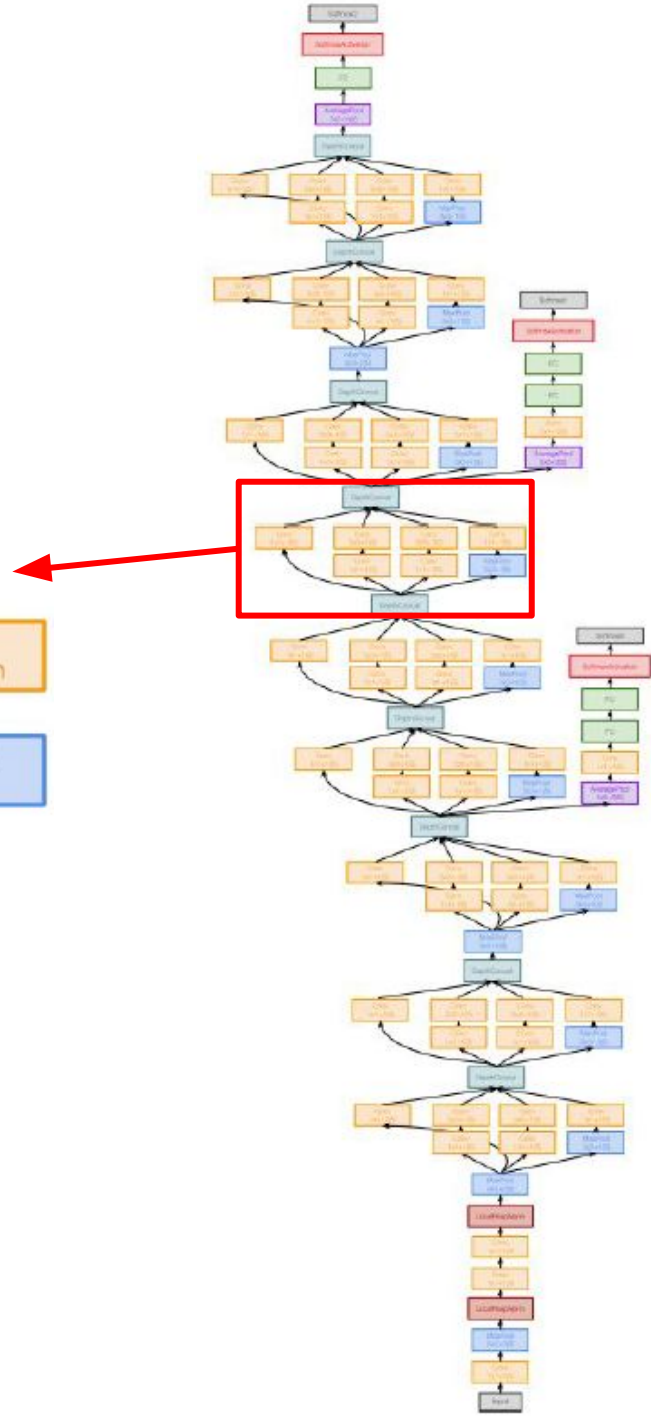
- Bem menos, comparado com as **854M** de operações da versão ingênua
- Também pode reduzir a profundidade depois da camada de *pooling*

GoogLeNet

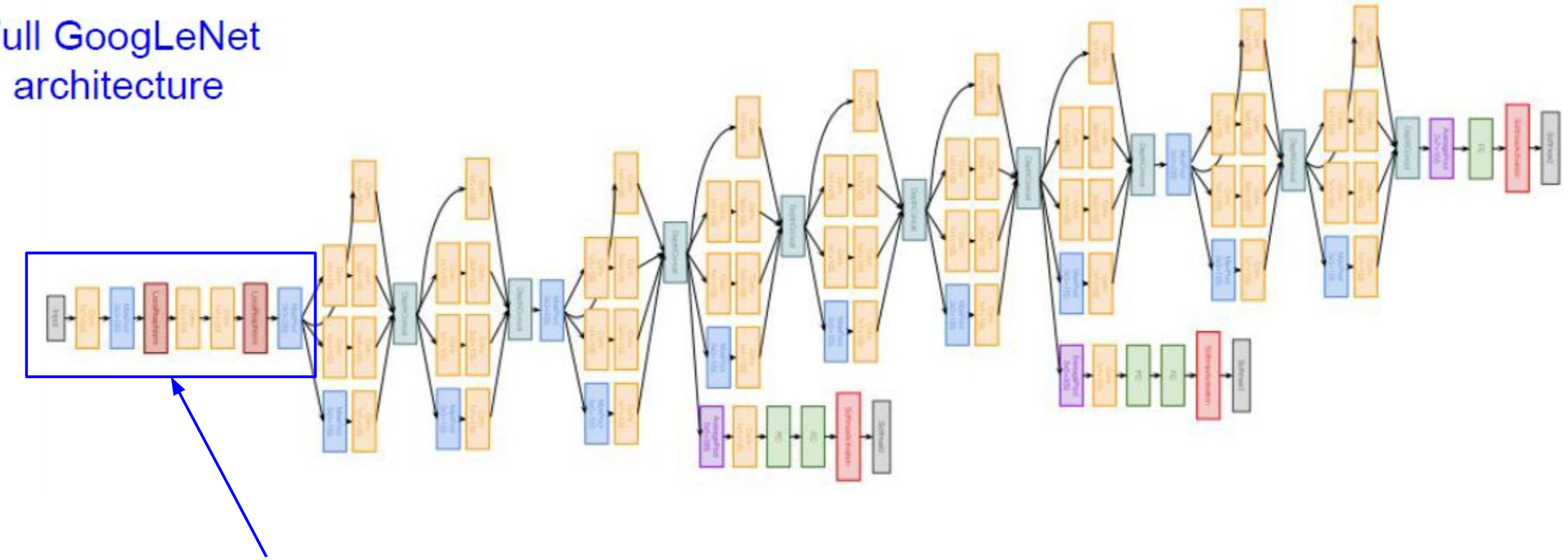
Empilha módulos “inception” com redução de dimensionalidade um em cima do outro



Módulo *inception*



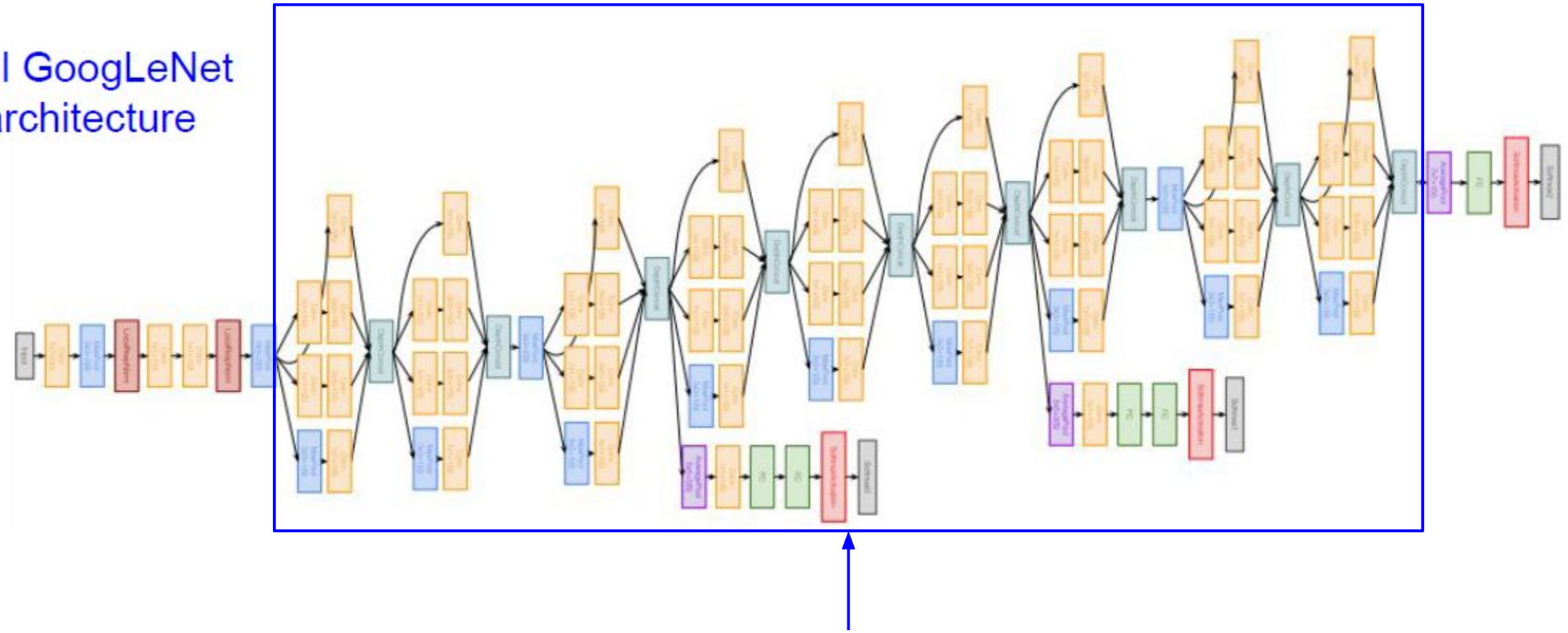
GoogLeNet



Traditional CNN:

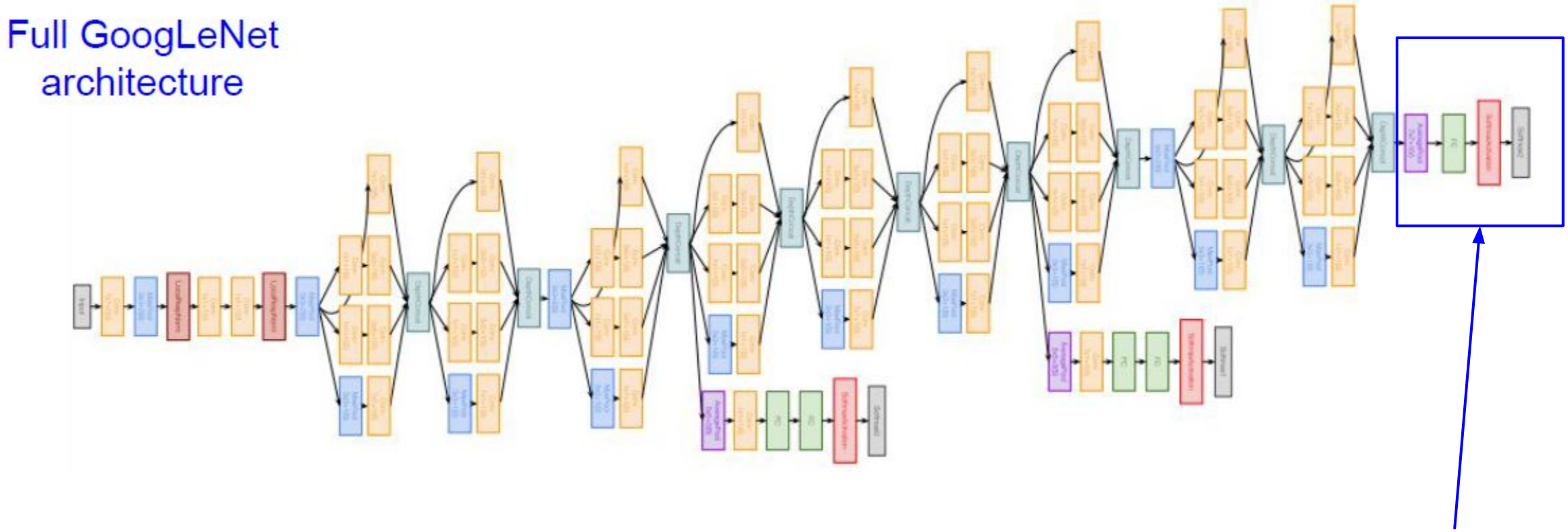
CONV-Pool-CONV-CONV-Pool

GoogLeNet



Módulos “Inception”
empilhados

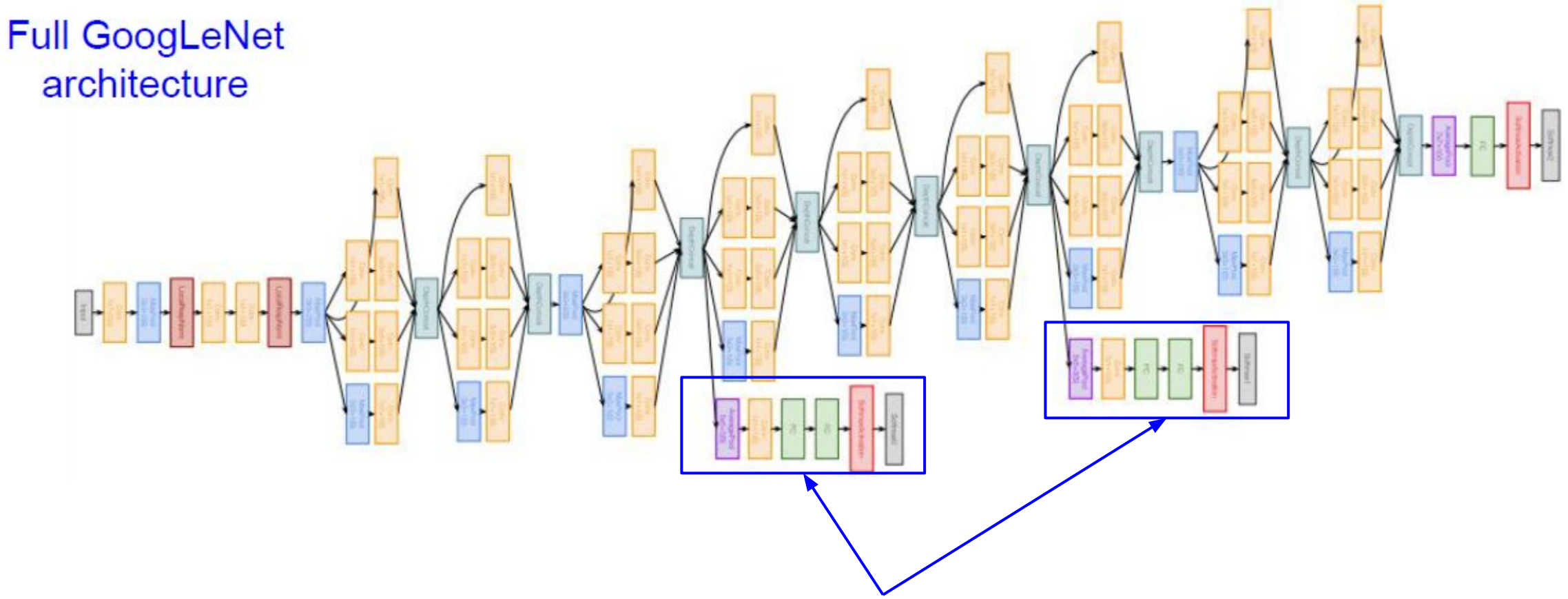
GoogLeNet



Saída do classificador

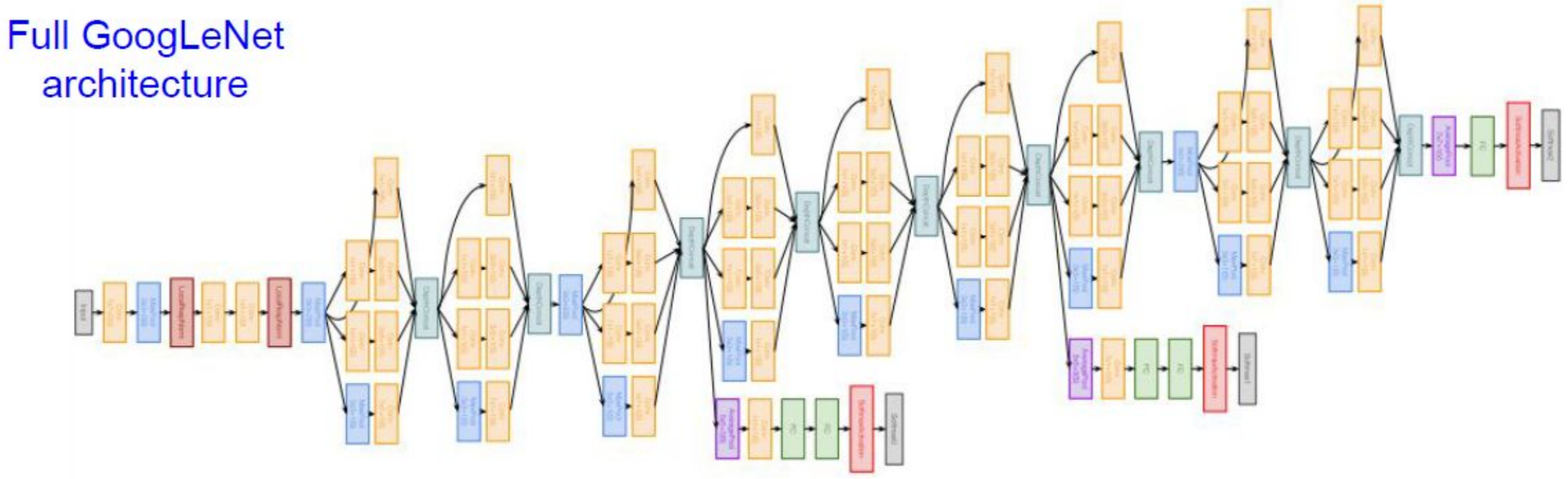
Nota: depois da última camada de convolução, uma camada de *average pooling* global é usada para calcular a média entre todos os mapas de features, antes da última camada FC. Não precisamos mais de múltiplas camadas FCs, o que é muito caro!

GoogLeNet



Saídas de classificação auxiliares para injetar gradientes adicionais em camadas mais baixas (AvgPool-1x1Conv-FC-FC-Softmax)

GoogLeNet



22 camadas no total com pesos (parâmetros):

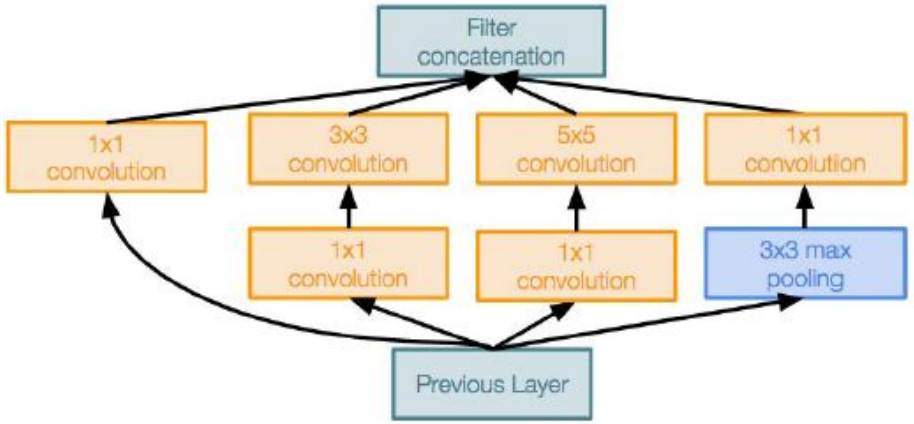
camadas paralelas contam como 1 camada -> 2 camadas por módulo Inception

Não conte camadas de saída auxiliares

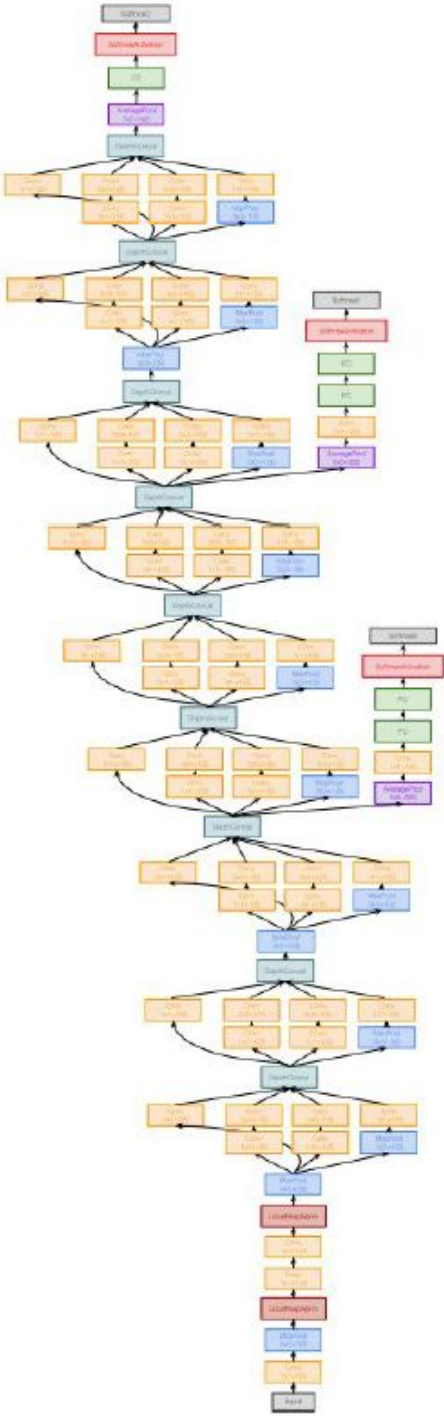
GoogLeNet

Redes mais profundas, mas com eficiência computacional

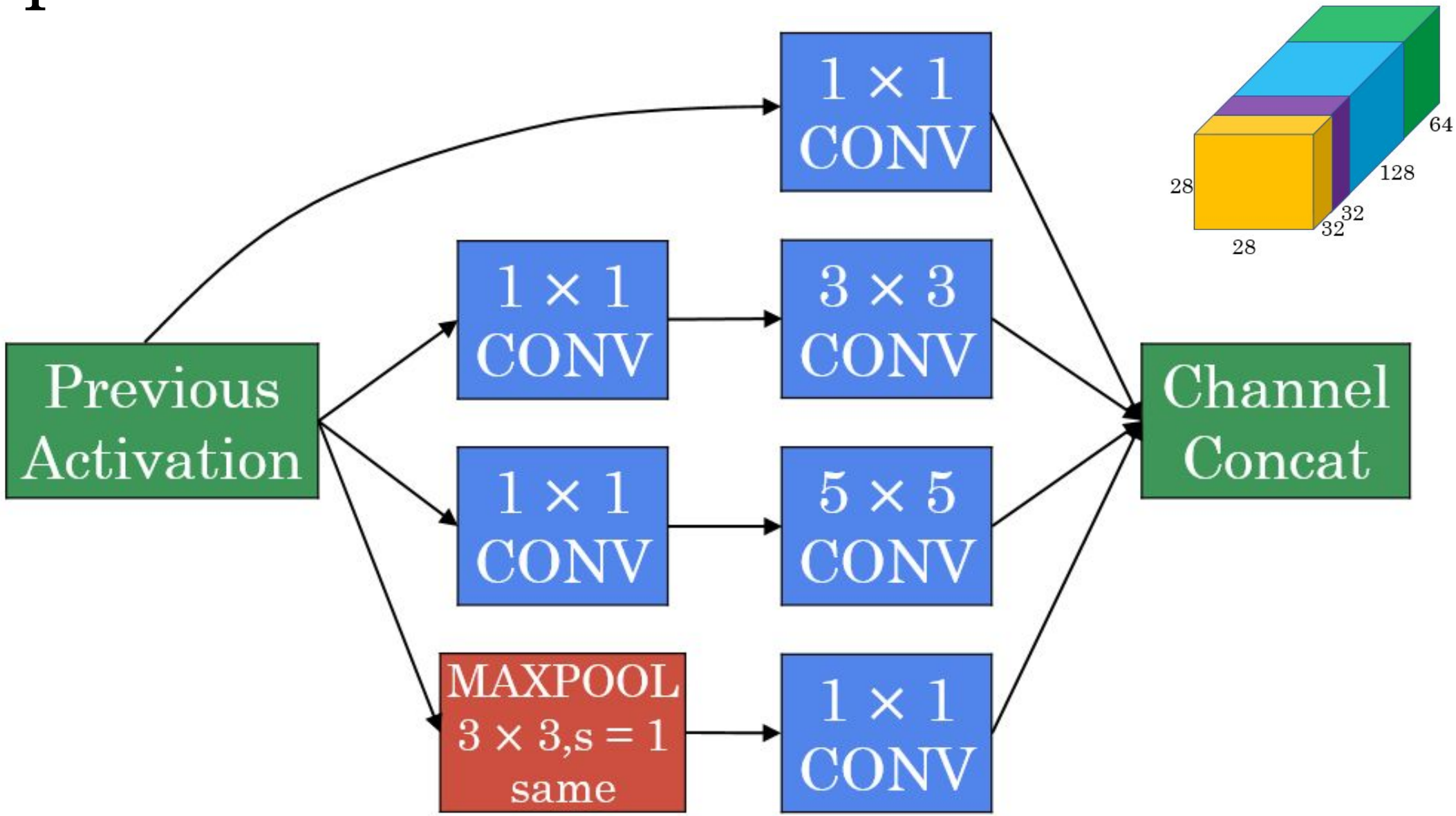
- 22 camadas
- Módulo “Inception” (eficiente)
- Sem camadas FC
- “Apenas” 5M parâmetros
  - 12x menos que a AlexNet
- Vencedor da tarefa de classificação do ILSVRC´14 (6.7% top 5 error)



Módulo *inception*



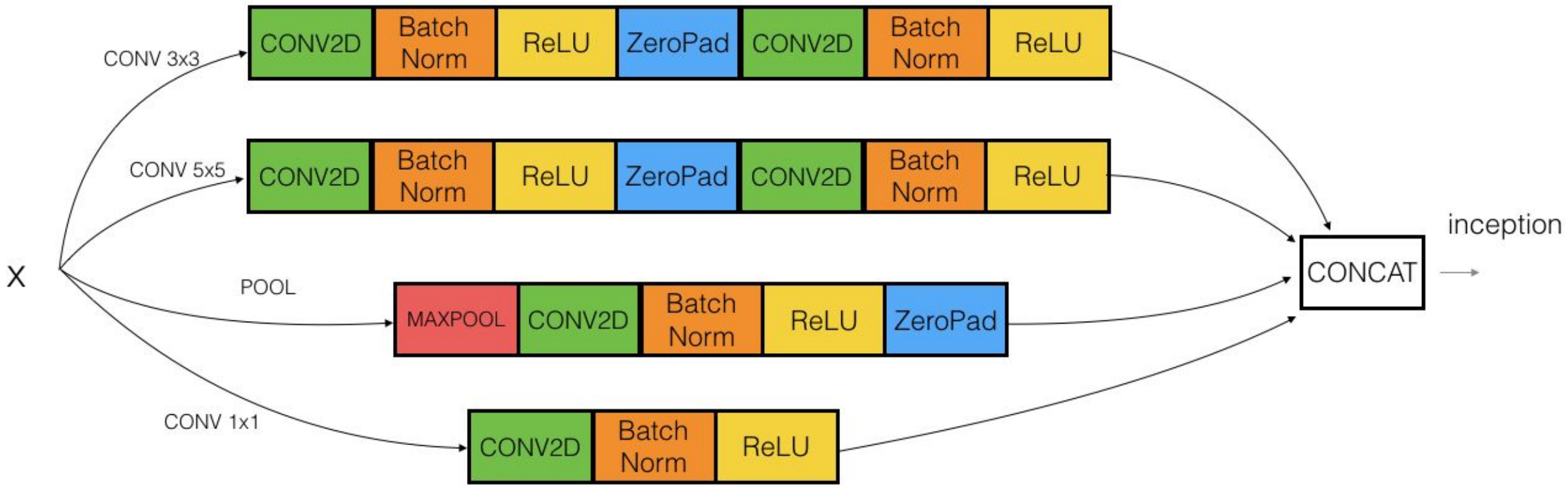
Inception module



Módulo *Inception*

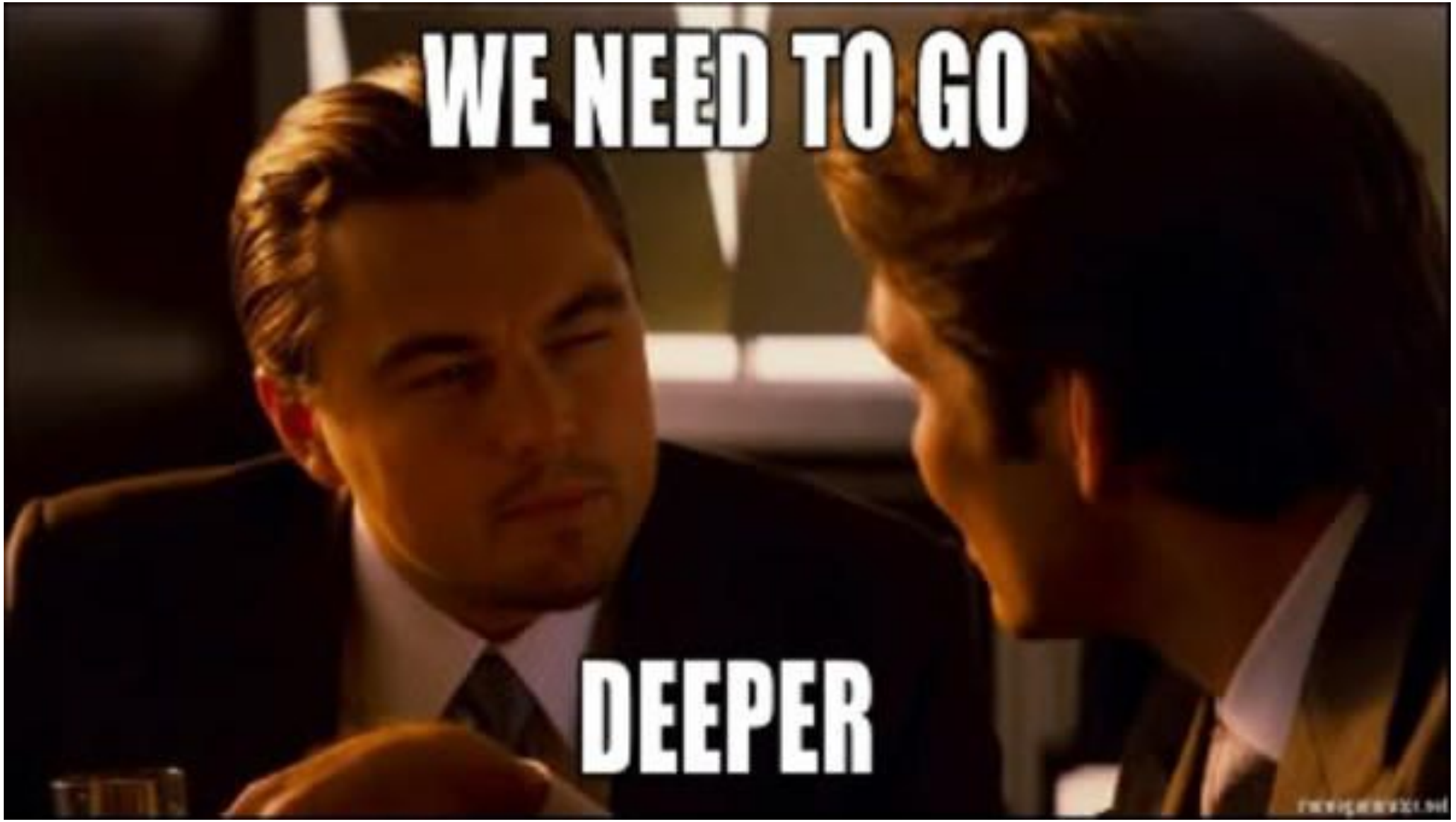
- Dica: o ***max-pooling*** é do tipo “same” aqui
- A entrada para o módulo inception é **28 x 28 x 192** e a saída é **28 x 28 x 256**
- Faremos todos os **Convs** e **pools** que podemos querer e deixaremos a NN aprender e decidir qual deles quer usar mais
- [Szegedy et al. 2014. Going deeper with convolutions]

Exemplo em Keras



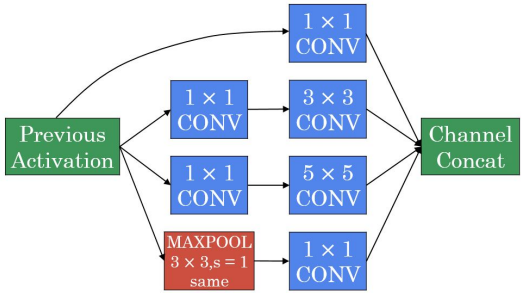
Rede inception (GoogleNet)

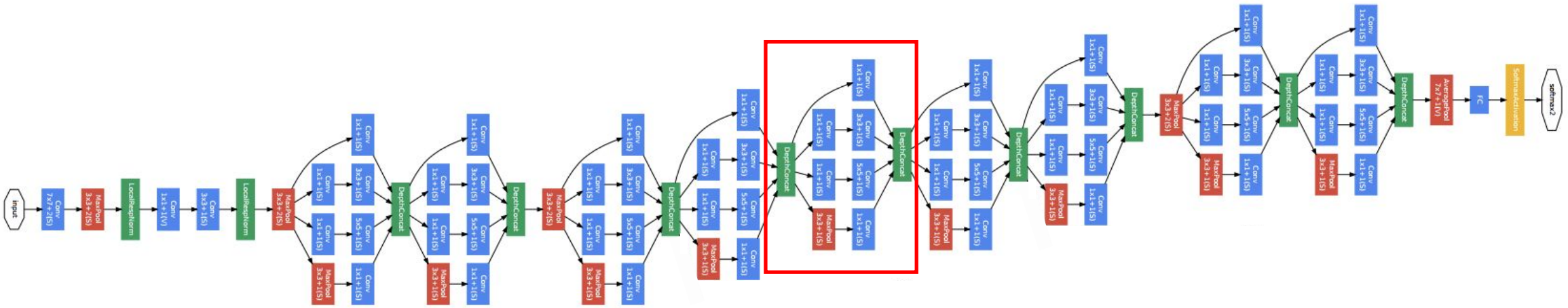
- A rede **Inception** consiste em blocos concatenados do módulo Inception
- O início do nome foi tirado de uma imagem memética tirada do filme **Inception**



http://knowyourmeme.com/memes/we-need-to-go-deeper

Rede inception



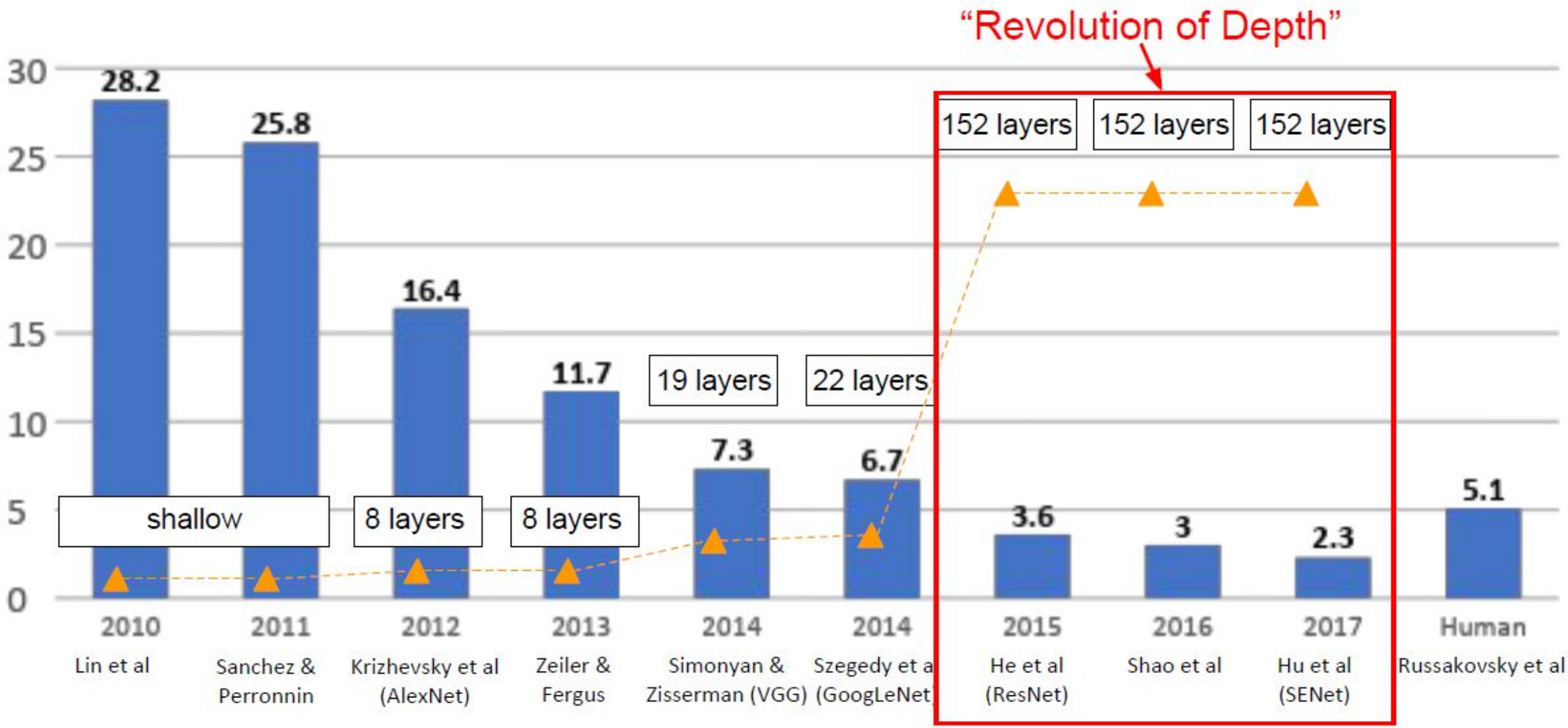


[Szegedy et al., 2014, Going Deeper with Convolutions]

Rede inception

- Algumas vezes, um bloco **Max-Pool** é usado antes do módulo de início para reduzir as dimensões das entradas
- Existem 3 blocos **Softmax** em diferentes posições para levar a rede ao seu objetivo
  - Ajuda a garantir que as ***features*** intermediárias sejam boas o suficiente para a rede aprender
  - Efeito colateral: **softmax0** e **sofmax1** também tem efeito de regularização
- Existem outras versões dessa rede, inception v2, v3 e v4
- Também existe uma rede que usou o módulo inception e a ResNet juntos

ImageNet Challenge (vencedores)



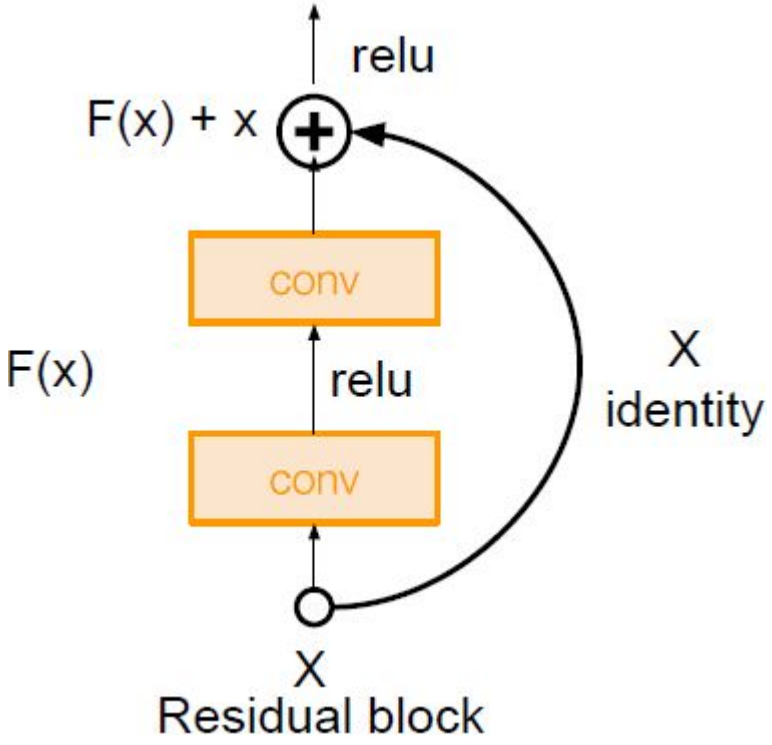
Estudos de Caso

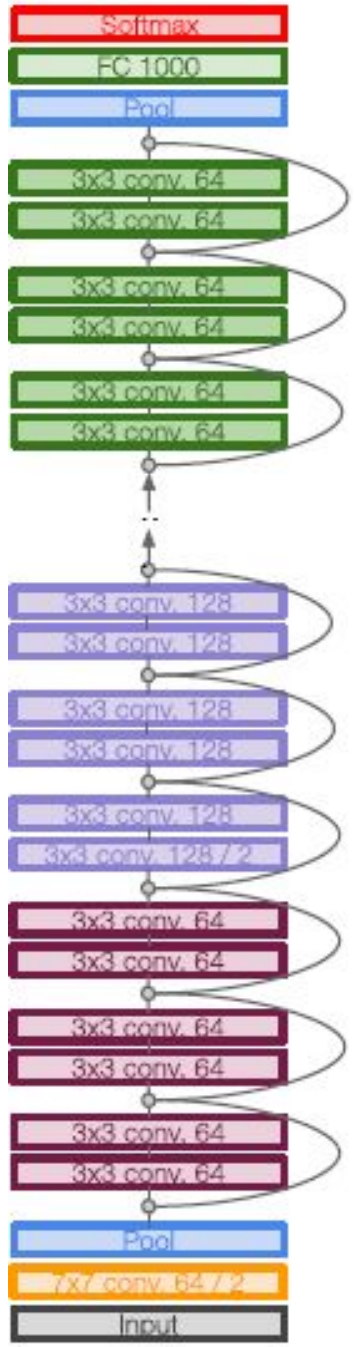
Redes Residuais (ResNets)

ResNets

Redes muito profundas usando conexões residuais

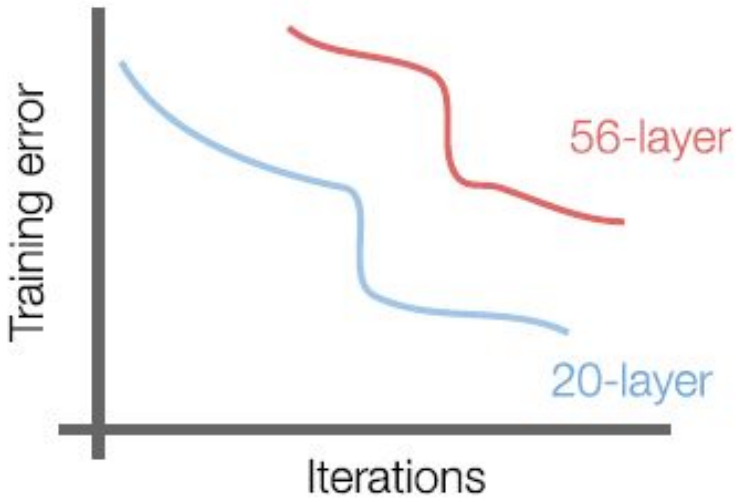
- Modelo de 152 camadas para o ImageNet
- Venceu o ILSVRC'15 (classificação) com 3.57% de top 5 error
- Venceu todas as tarefas de classificação e detecção tanto no ILSVRC'15 e COCO'15

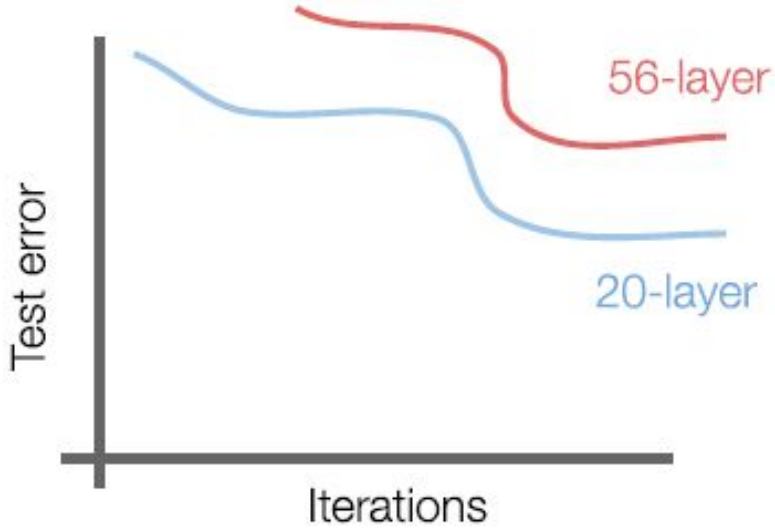




ResNets

- O que acontece quando continuamos a empilhar camadas mais profundas em uma CNN?



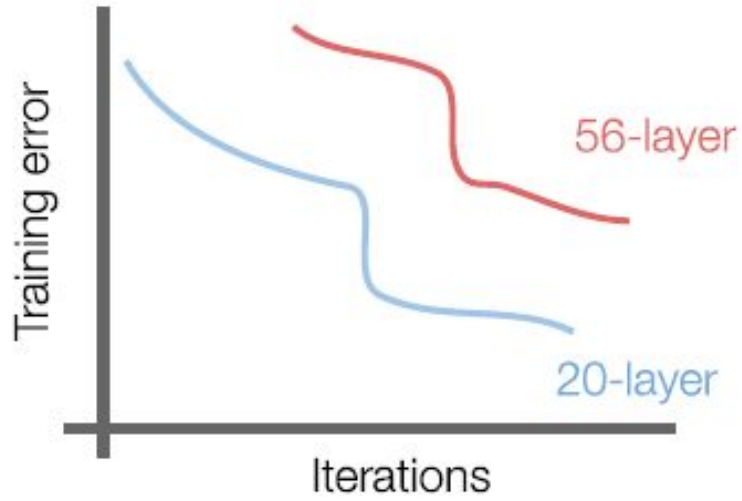


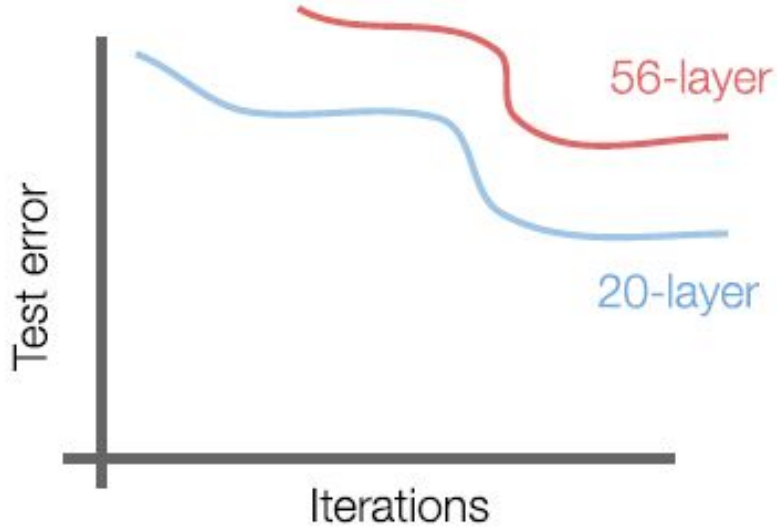
Q: O que tem de estranho com essas curvas?

A: O modelo de 56 camadas é pior tanto no treino quanto no erro.

ResNets

- O que acontece quando continuamos a empilhar camadas mais profundas em uma CNN?





O modelo de 56 camadas é pior tanto no treino quanto no erro.

Q: O que isso significa?

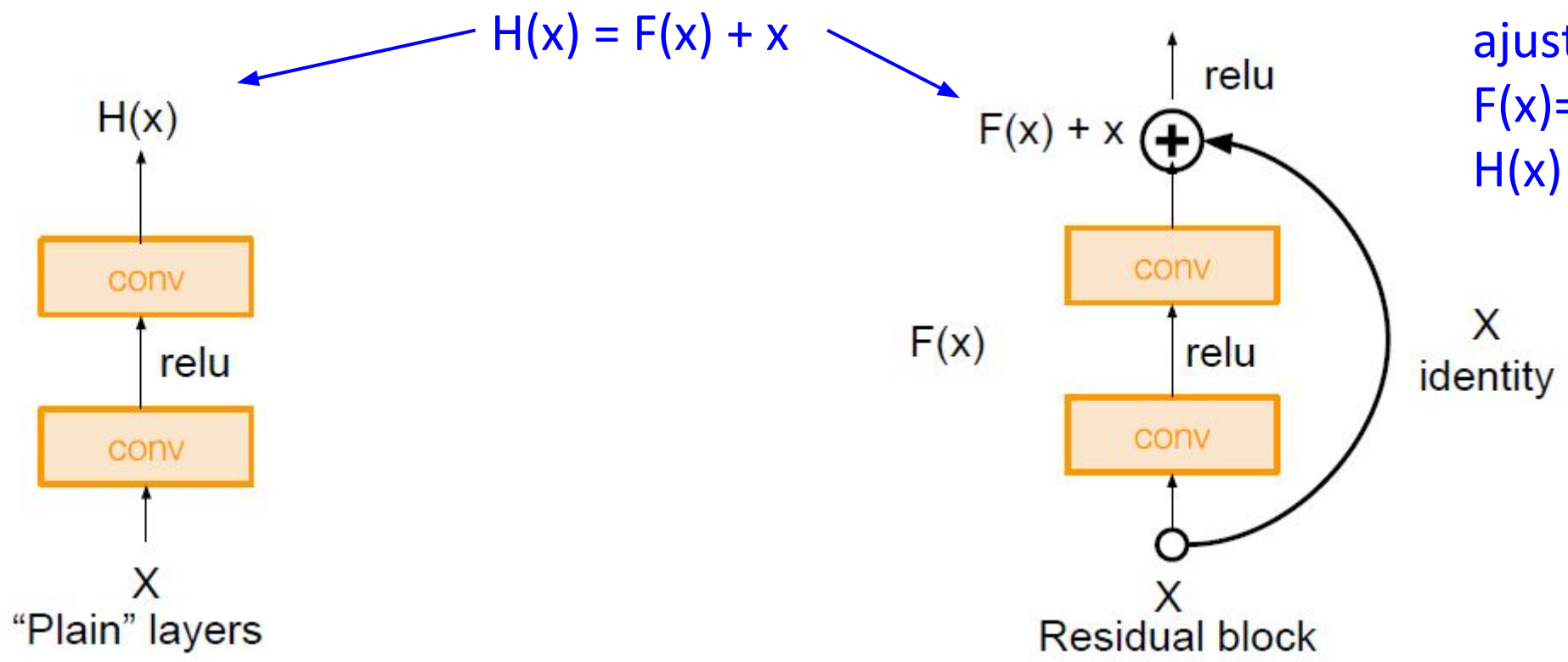
A: O problema não é *overfitting*!

ResNets

- Hipótese: o problema é a otimização
  - Modelos mais profundos são mais difíceis de otimizar
  - Modelos mais profundos deveriam ser pelo menos tão bons
    - Prova por construção: considere um modelo com K camadas, e adicione uma camada extra que faça apenas um *identity mapping*

ResNets

- Solução: Use as camadas da rede para ajustar um mapa residual ao invés de diretamente tentar ajustar o mapeamento subjacente desejado

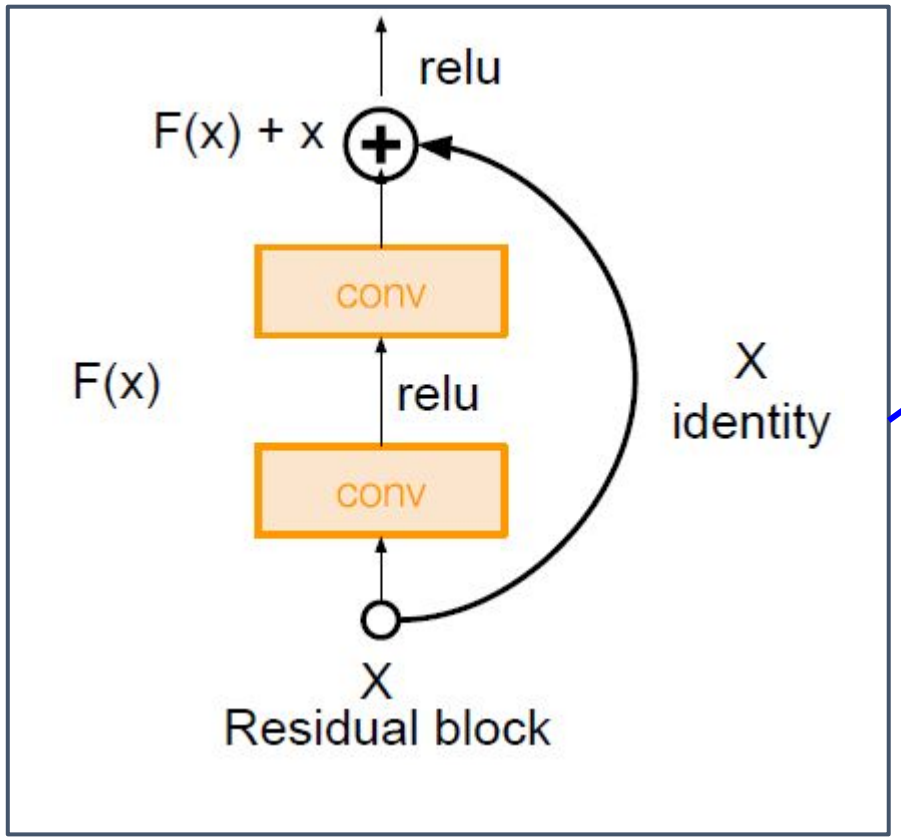


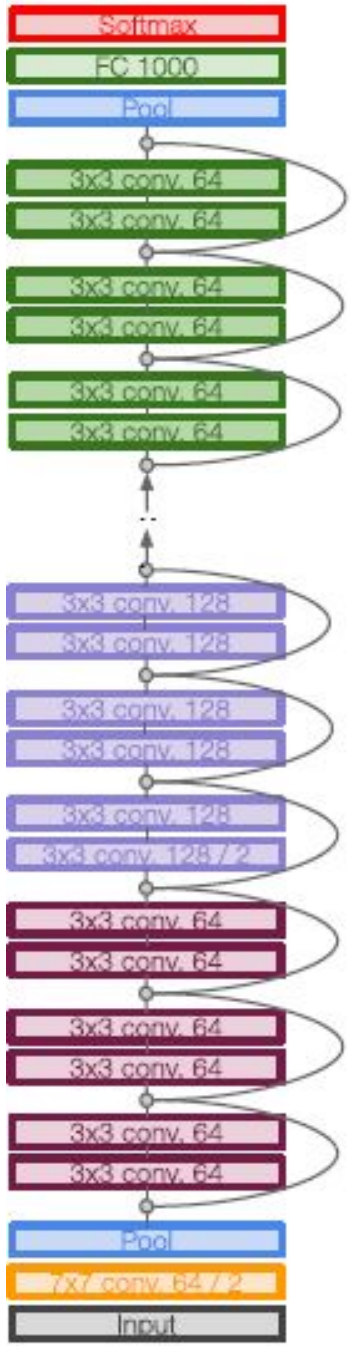
Use as camadas para ajustar os resíduos F(x)=H(x)-x ao invés de H(x) diretamente

ResNets

Arquitetura completa:

- Empilhe blocos residuais
- Cada bloco residual tem duas camadas CONV 3x3

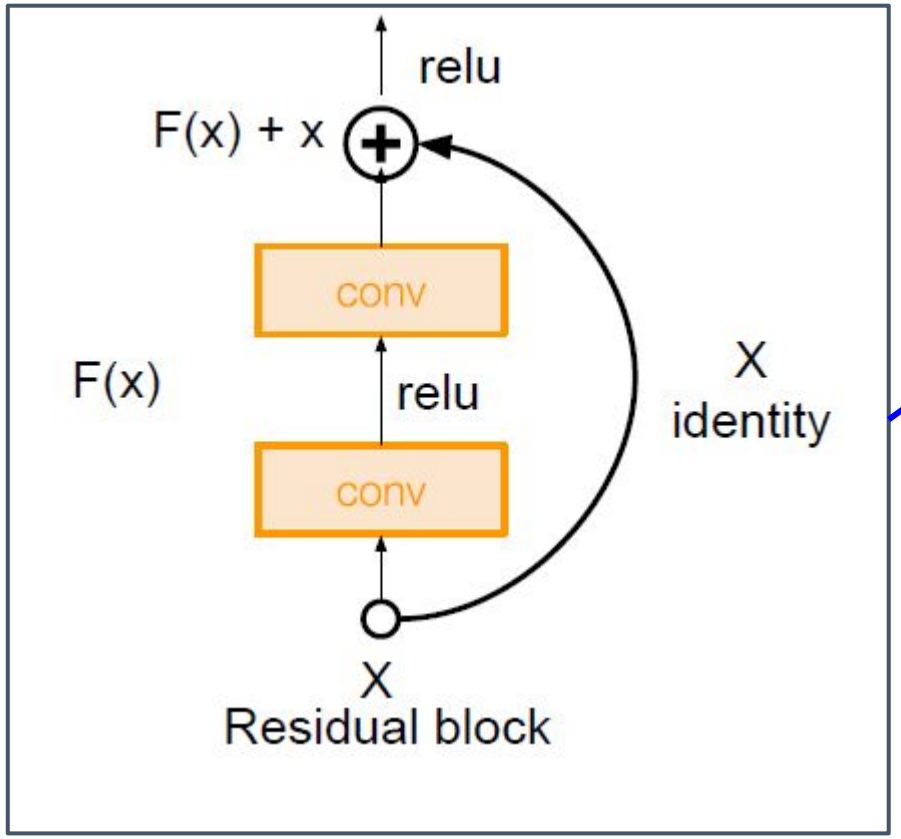


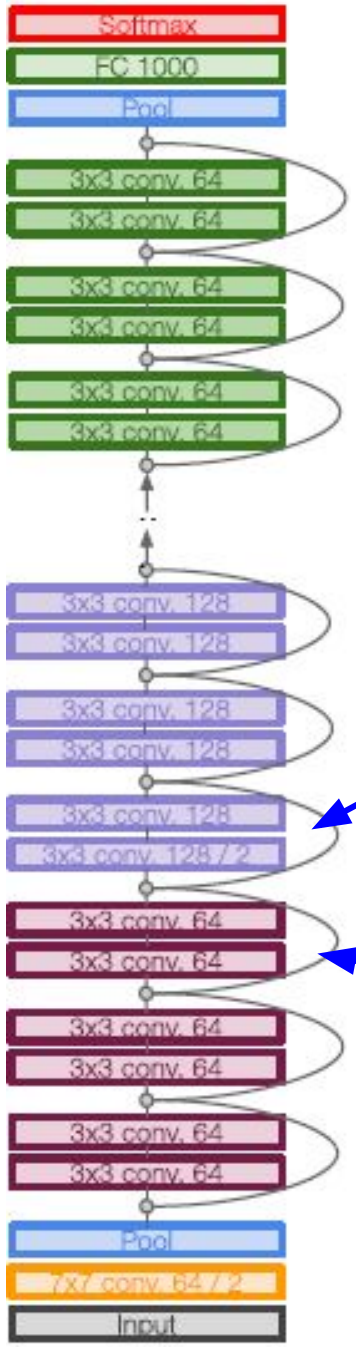


ResNets

Arquitetura completa:

- Empilhe blocos residuais
- Cada bloco residual tem duas camadas CONV 3x3
- Periodicamente dobre o # de filtros e reduza o tamanho espacial usando *stride* 2 (/2 em cada dimensão)
-





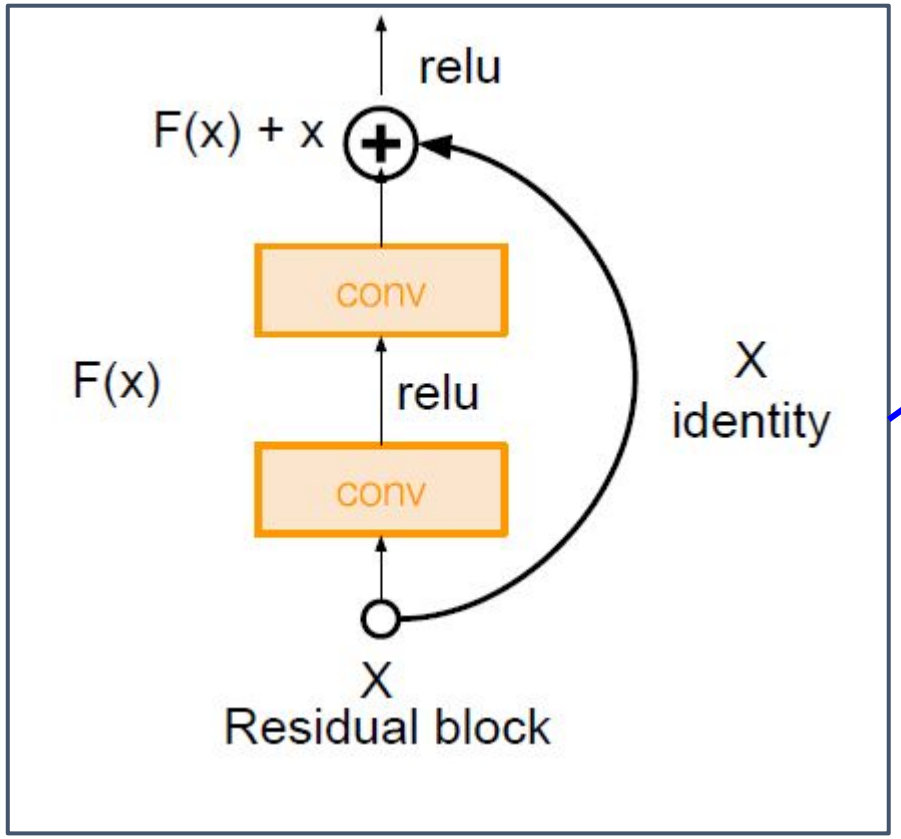
3x3 CONV, 128 filtros, /2 espacialmente com stride 2

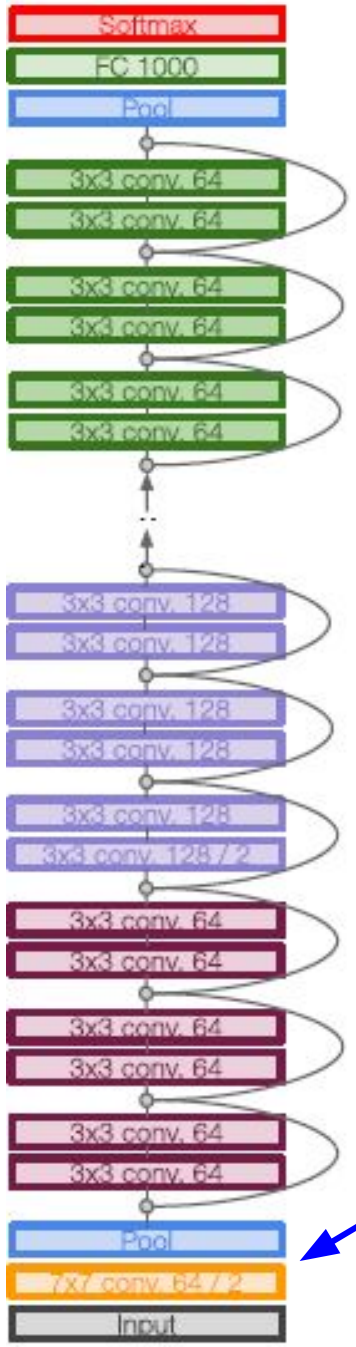
3x3 CONV, 64 filtros

ResNets

Arquitetura completa:

- Empilhe blocos residuais
- Cada bloco residual tem duas camadas CONV 3x3
- Periodicamente dobre o # de filtros e reduza o tamanho espacial usando *stride* 2 (/2 em cada dimensão)
- Camada CONV adicional no começo



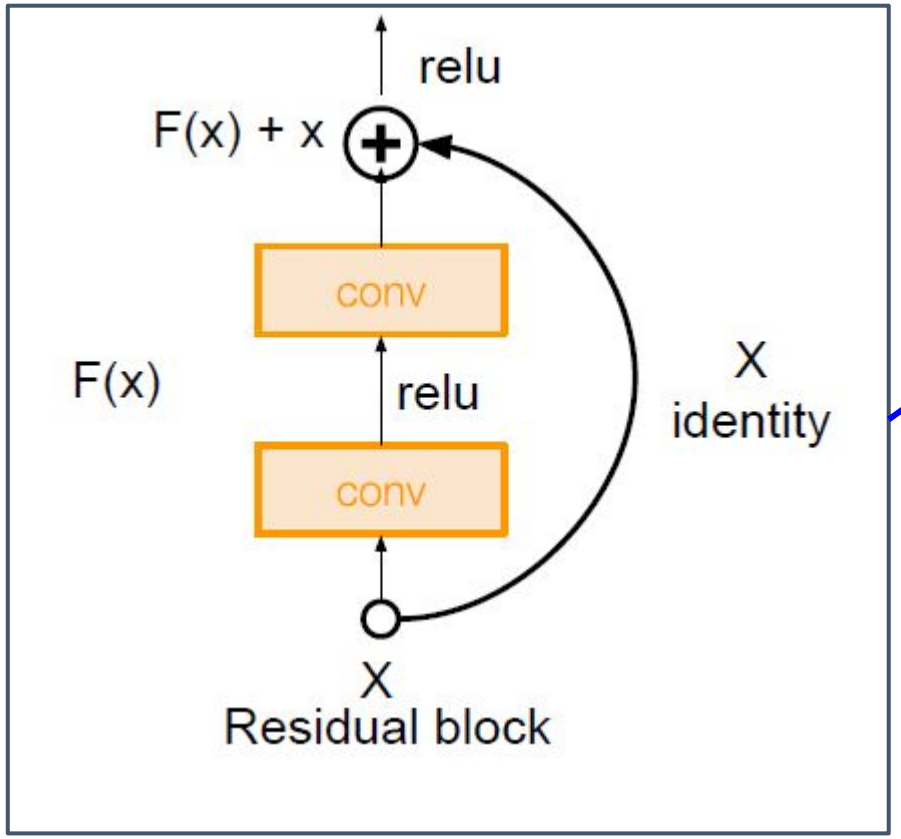


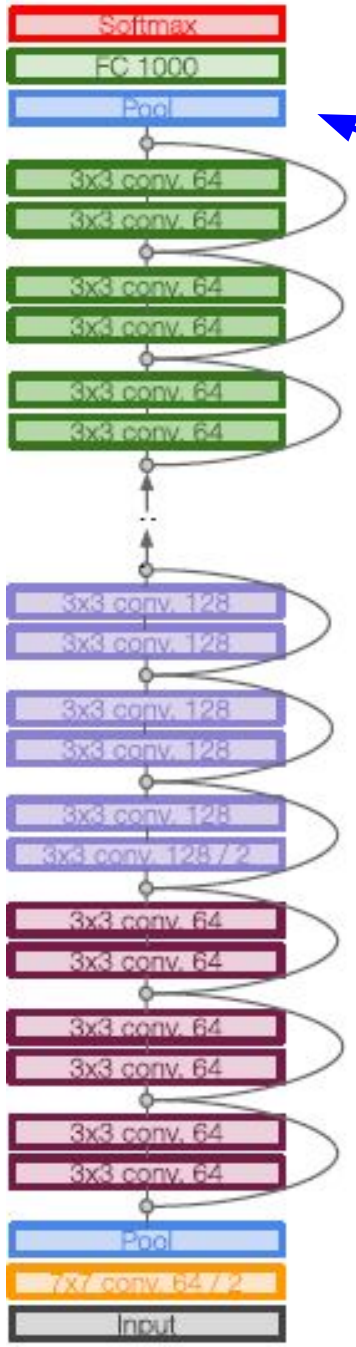
Camada CONV no começo

ResNets

Arquitetura completa:

- Empilhe blocos residuais
- Cada bloco residual tem duas camadas CONV 3x3
- Periodicamente dobre o # de filtros e reduza o tamanho espacial usando *stride* 2 (/2 em cada dimensão)
- Camada CONV adicional no começo
- Sem camadas FC no final (apenas FC 1000 para as classes finais)



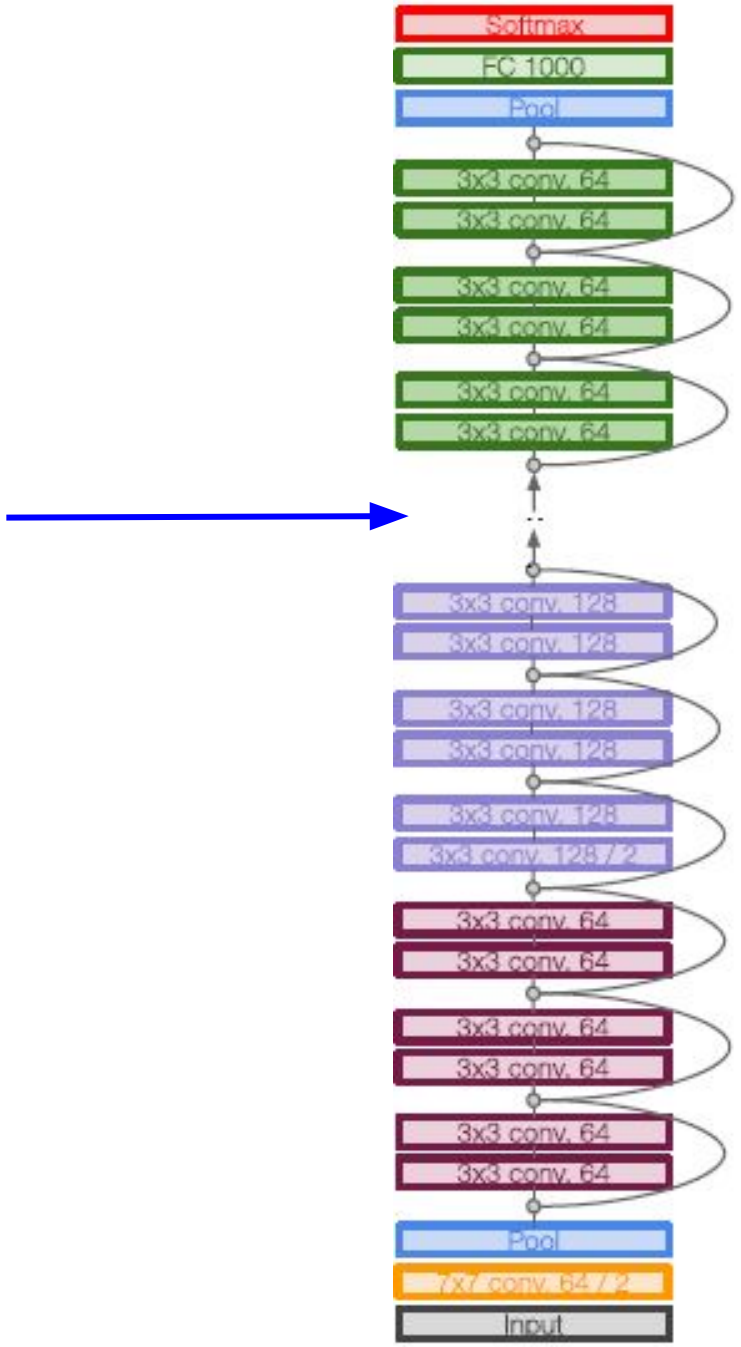


Sem camadas FC, apenas a FC 1000 para as classes finais

Camada de pooling médio depois da última camada CONV

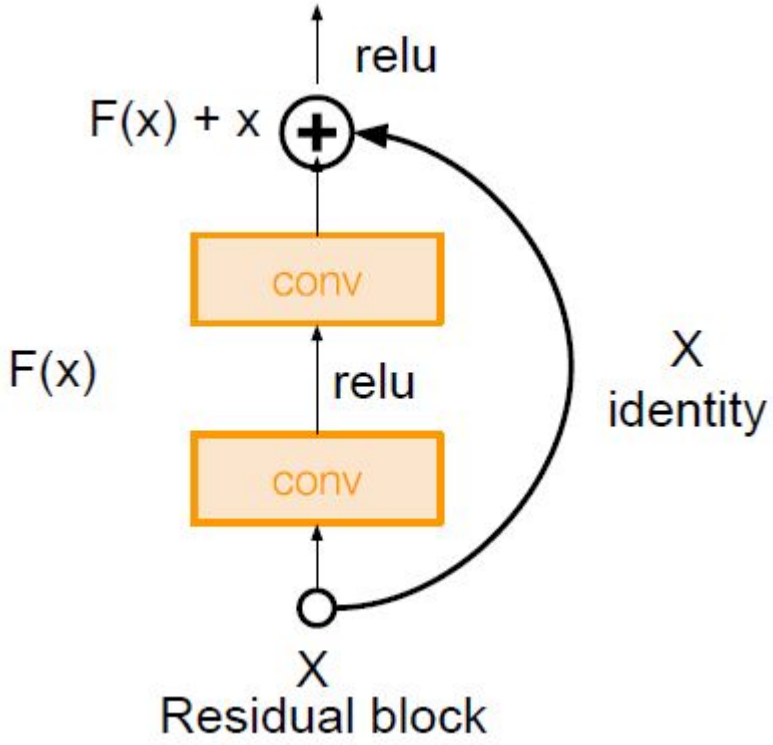
ResNets

Para o ImageNet, foram testadas profundidades de 34, 50, 101 e 152 camadas



ResNets

Para redes mais profundas (ResNet-50+), use uma camada “bottleneck” para melhorar a eficiência



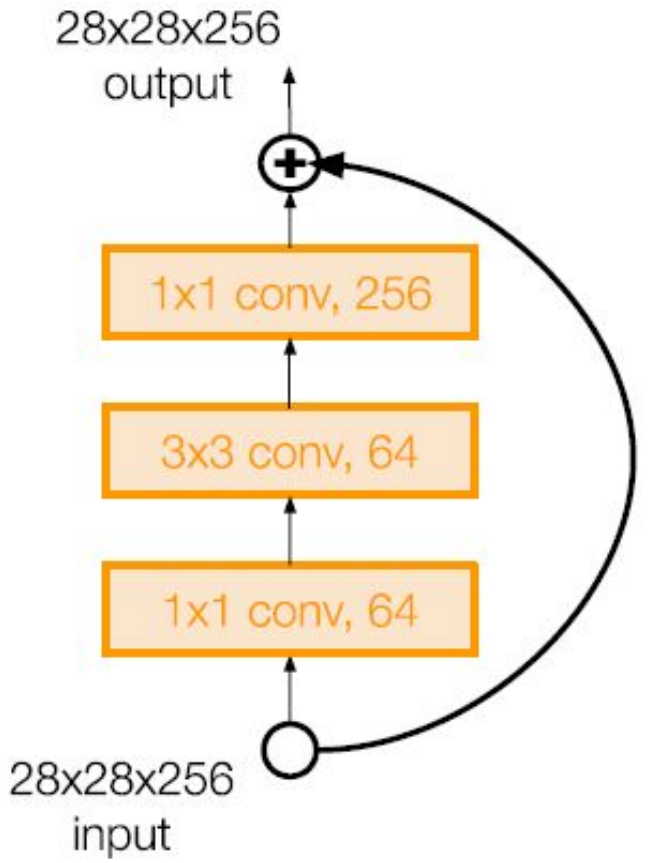
ResNets

Para redes mais profundas (ResNet-50+), use uma camada “bottleneck” para melhorar a eficiência

1x1 CONV, 256 filtros, projetam de volta para 256 mapas de features 28x28x256

3x3 CONV opera sobre apenas 64 mapas de features

1x1 CONV, 64 filtros para projetar em 28x28x64



ResNEts

Resultados experimentais

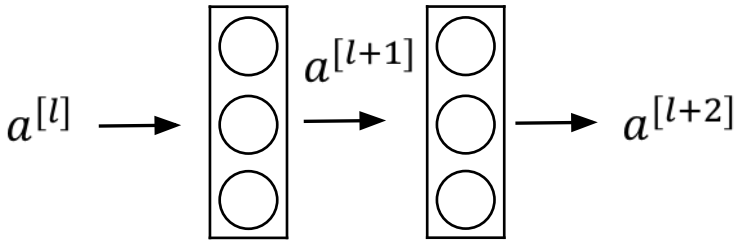
- Capazes de treinar redes muito profundas sem perda de desempenho
  - 152 camadas no ImageNet, 1202 no Cifar
- Redes mais profundas alcançam erro no treino menor, como esperado
- Primeiros lugares em todas competições (2015)

ILSVRC 2015: 3.6% de erro - melhor que performance humana [Russakovsky 2014]

Redes residuais

- NNs muito, muito profundas são difíceis de treinar por causa do problema de desaparecimento e explosão de gradientes
- Para resolver isso, aprenderemos sobre a **conexão *skip***
  - Pega a ativação de uma camada e a alimenta para outra camada
  - Permite treinar NNs grandes, mesmo com camadas maiores que 100

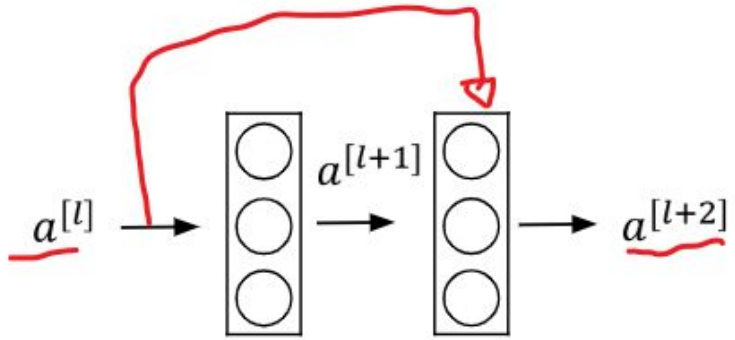
Bloco residual



$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$

[He et al., 2015. Deep residual networks for image recognition]

Bloco residual



$a^{[l]}$ → linear → $z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$

→ ReLU → $a^{[l+1]} = g(z^{[l+1]})$

→ linear → $z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$

→ (+) → ReLU → $a^{[l+2]} = g(z^{[l+2]})$

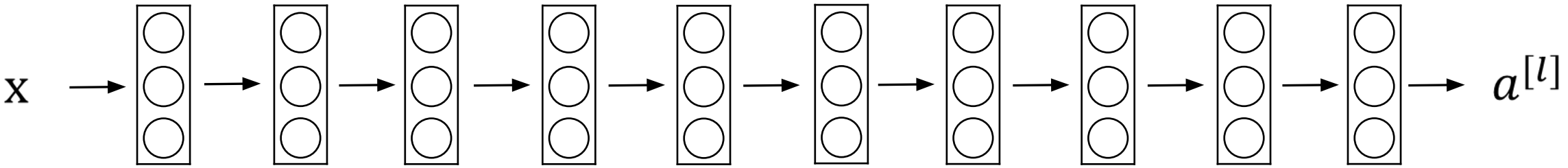
$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

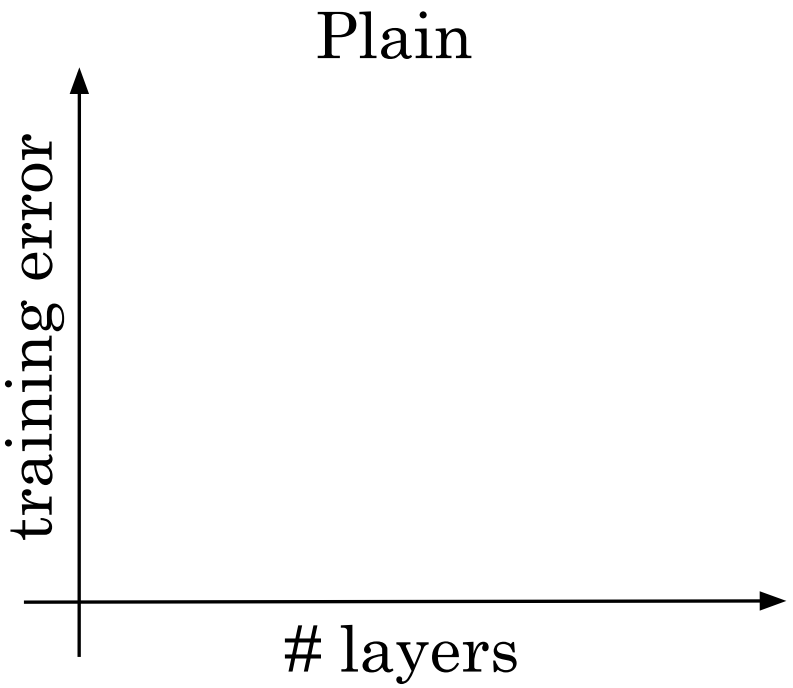
[He et al., 2015. Deep residual networks for image recognition]

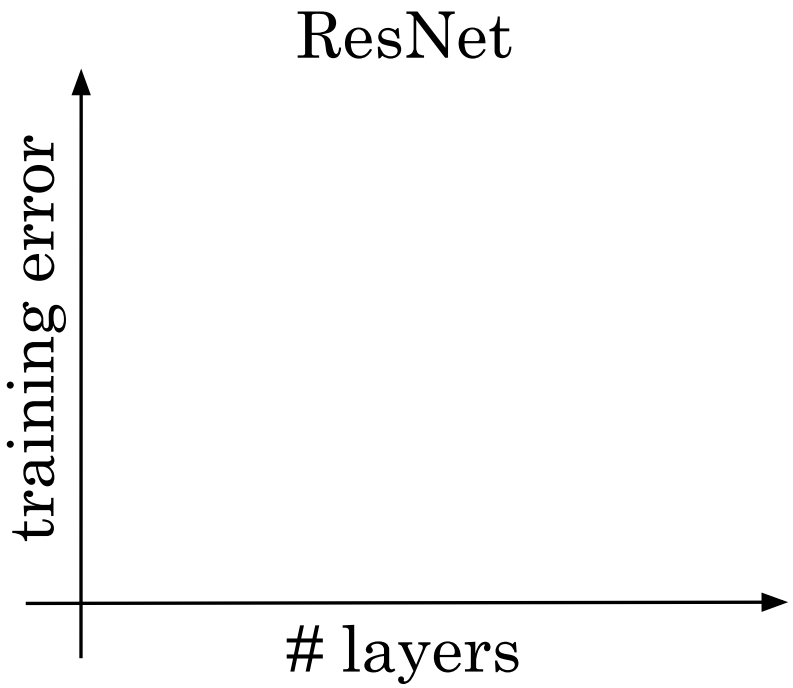
Blocos residuais

- ResNets são compostas de blocos residuais
- Eles adicionam uma conexão de atalho/salto (*shortcut*/*skip*) antes da segunda ativação
- Os autores deste bloco propõem que você pode treinar NNs mais profundas usando o empilhamento destes blocos
- [He et al., 2015. Deep residual networks for image recognition]

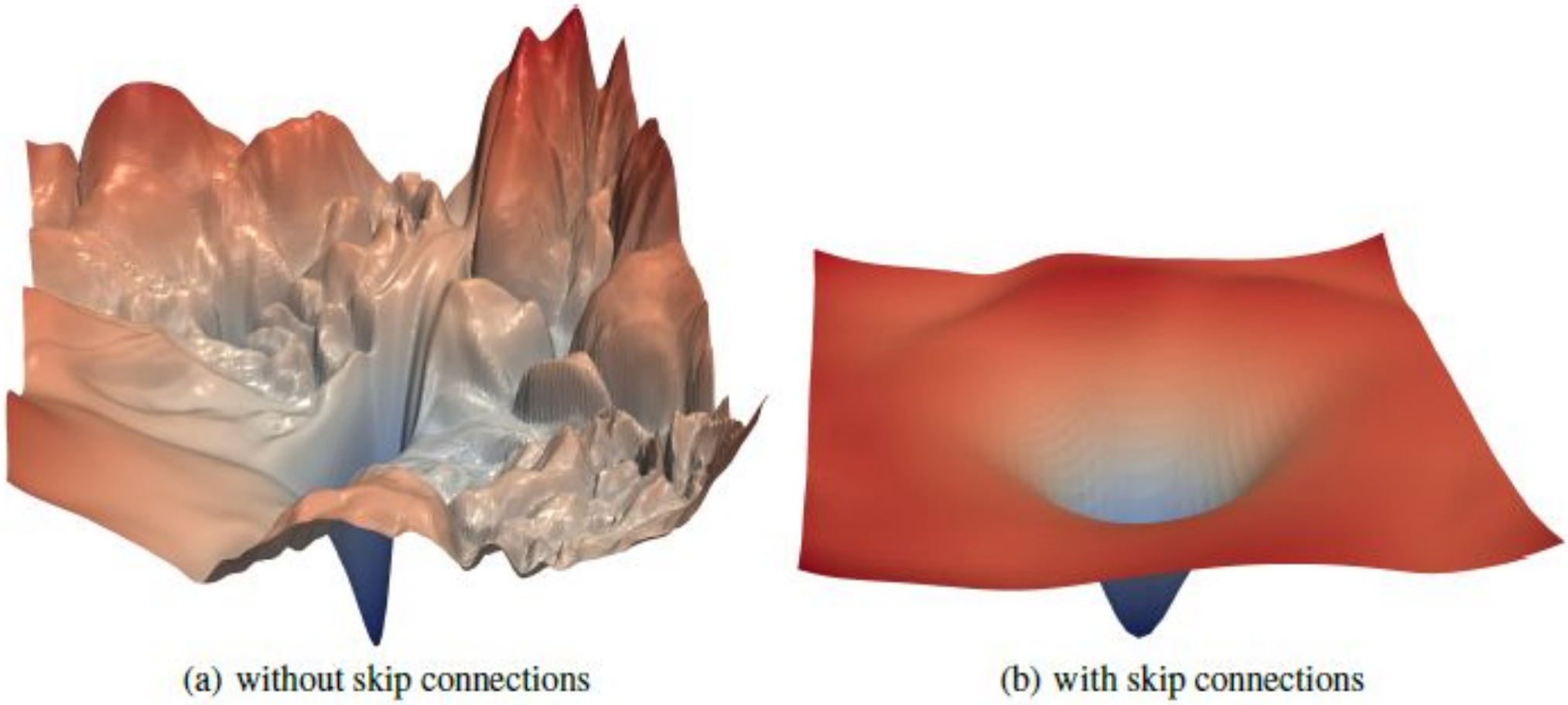
Rede Residual







[He et al., 2015. Deep residual networks for image recognition]



(a) without skip connections

(b) with skip connections

Superfícies das funções de perda

Li, Hao, et al. "Visualizing the loss landscape of neural nets." Advances in Neural Information Processing Systems. 2018.

Rede residual

- São redes compostas de alguns blocos residuais
- Essas redes podem se aprofundar sem prejudicar o desempenho
- Nas redes normais, a teoria nos diz que, se nos aprofundarmos, obteremos uma solução melhor para o nosso problema
- No entanto, por causa dos problemas de gradientes que desaparecem e explodem, o desempenho da rede sofre à medida que se aprofunda
- Graças à Rede Residual, podemos ir mais fundo como queremos agora

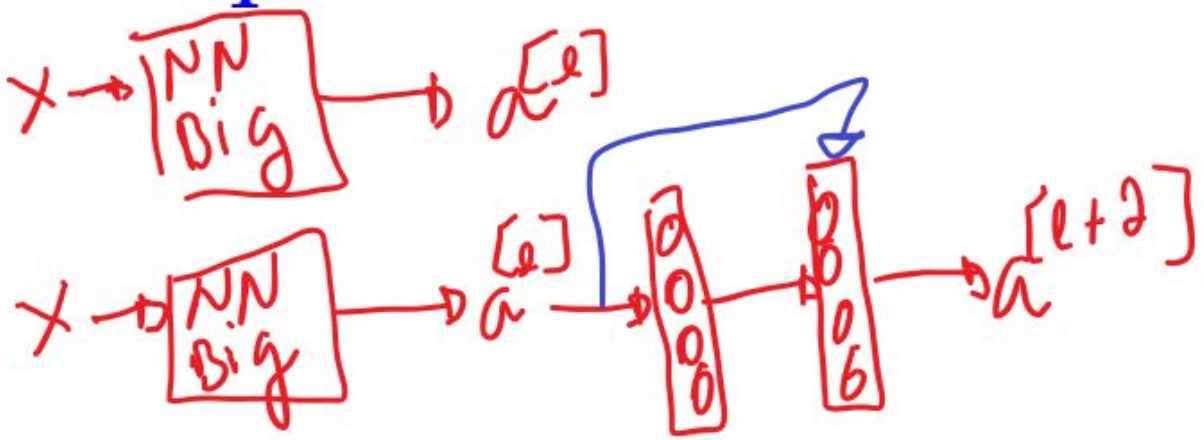
Rede residual

- O desempenho do ResNet aumenta à medida que a rede se aprofunda
- Em alguns casos, ir mais fundo não afeta o desempenho e isso depende do problema em sua mão
- Algumas pessoas estão tentando treinar redes com 1000 camadas, que por enquanto ainda não são usadas na prática

Estudos de Caso

Por que ResNets funcionam?

Por que redes residuais funcionam?



$\in \mathbb{R}^{256\times 128}$

$a^{[l+2]} = g(W^{[l+2]} a^{[l+1]} + b^{[l+2]} + W_s a^{[l]})$

(256) (128)

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$$= g(W^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]}) = g(a^{[l]}) \xrightarrow{\text{se } g \text{ é ReLU}} a^{[l]}$$

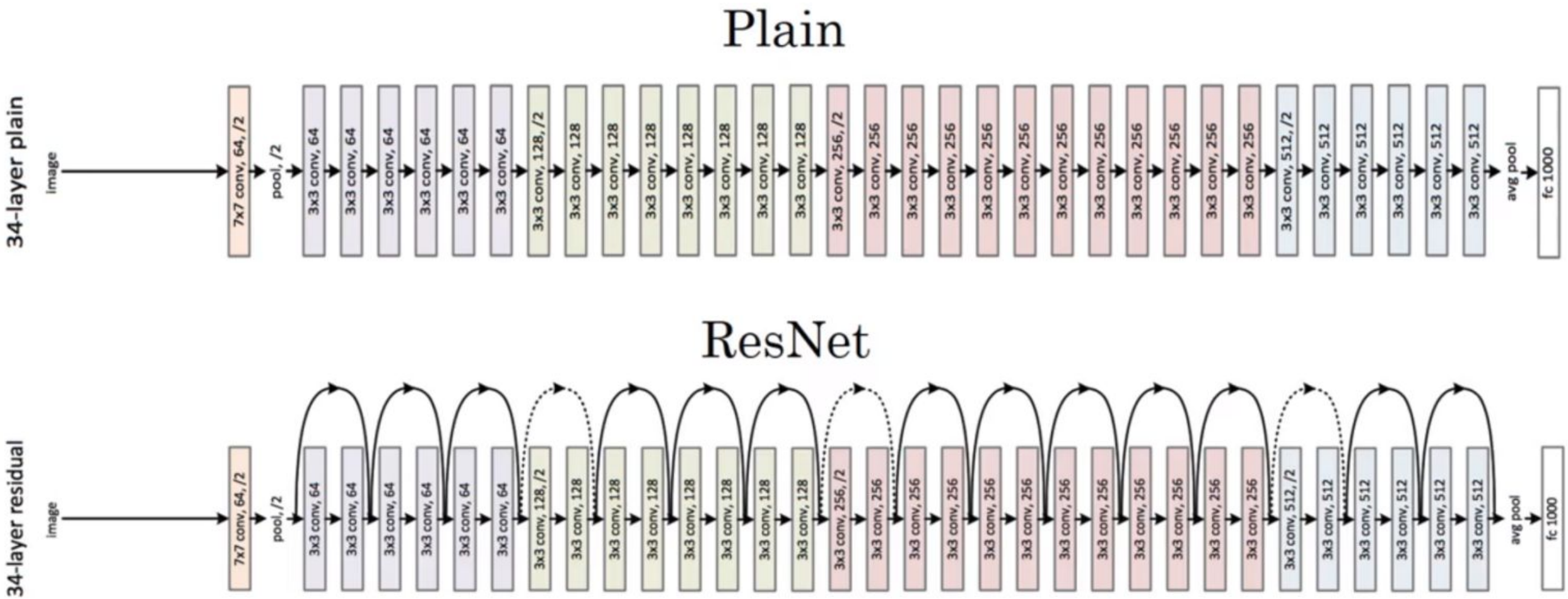
Se usarmos reg. L2 ou weight decay:

$W \to 0$

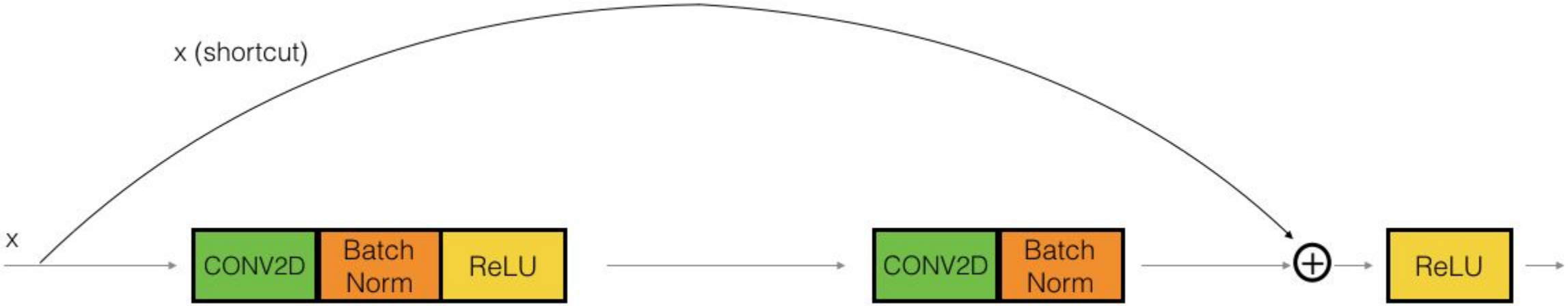
Por que redes residuais funcionam?

- O quão difícil é fazer uma camada (FC ou Conv) aprender uma função identidade?

ResNet

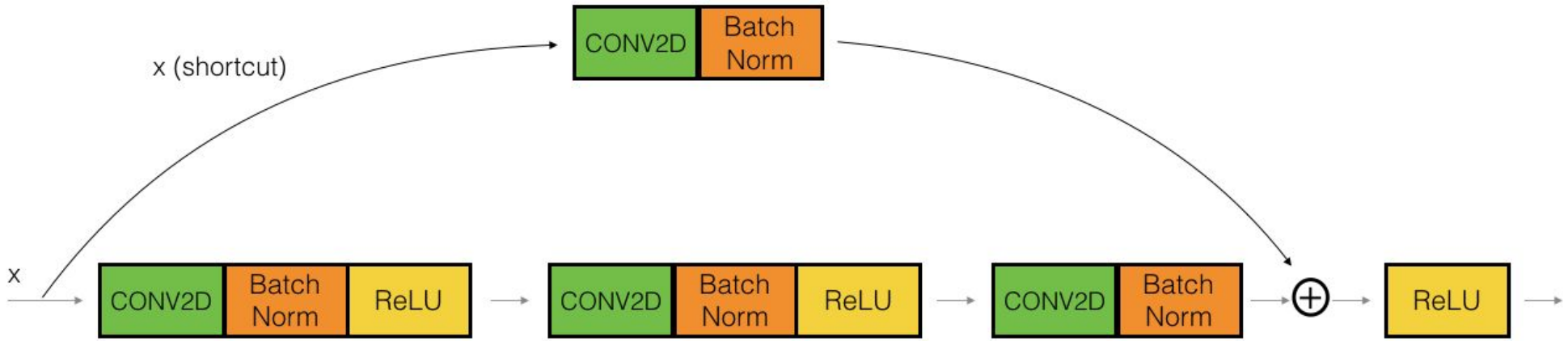

Plain
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
ResNet
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128
3x3 conv, 256
3x3 conv, 512
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000

Bloco identidade (“*identity*”)



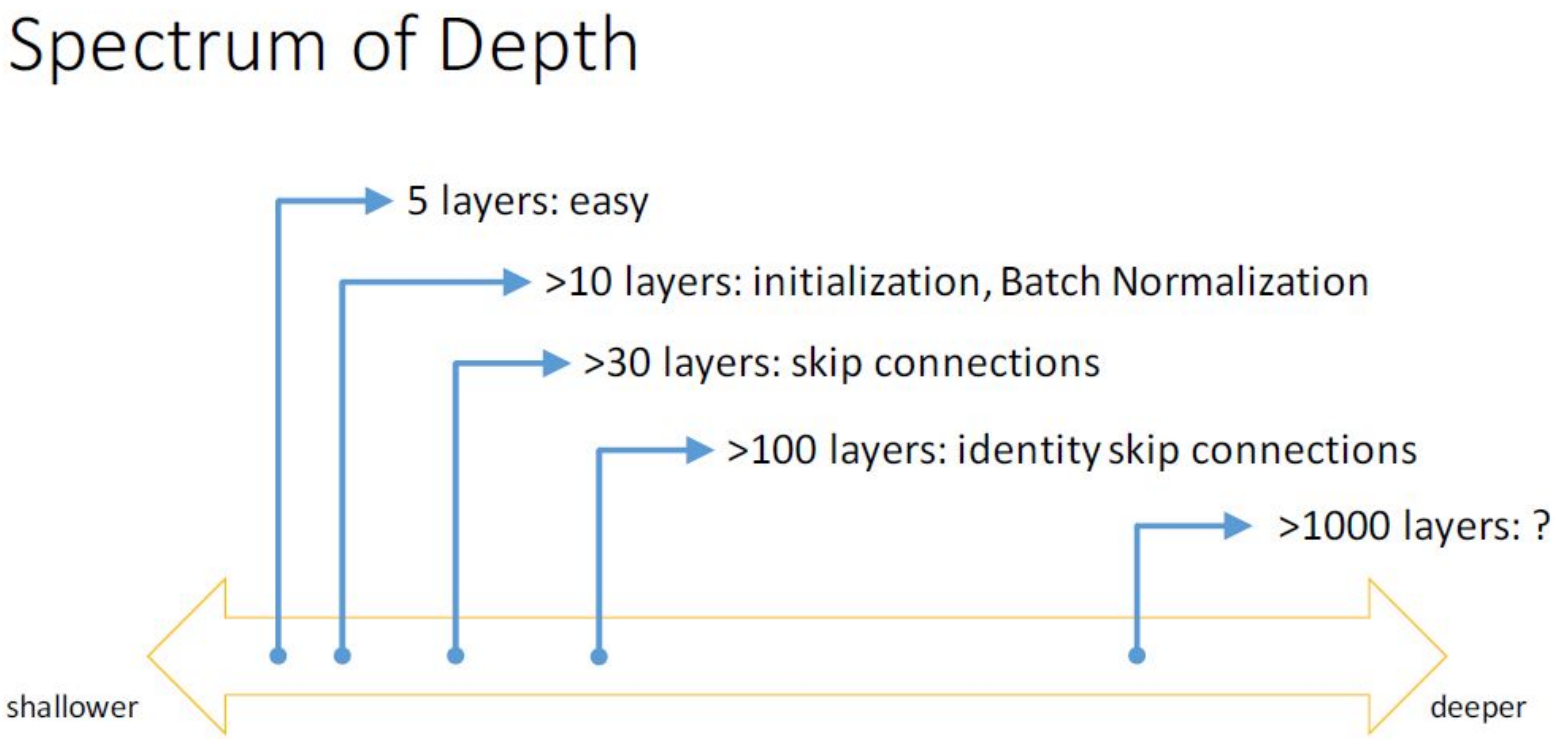
- A **conv** é seguida por um **BN** antes da **RELU**. Dimensões aqui são as mesmas.
- O ***skip*** é dado sobre 2 camadas
  - A conexão de salto pode saltar **n** conexões onde **n > 2**
- Este desenho representa camadas no Keras

Bloco convolucional (“*convolutional*”)



- A **conv** pode ser um ***bottleneck*** 1x1 conv

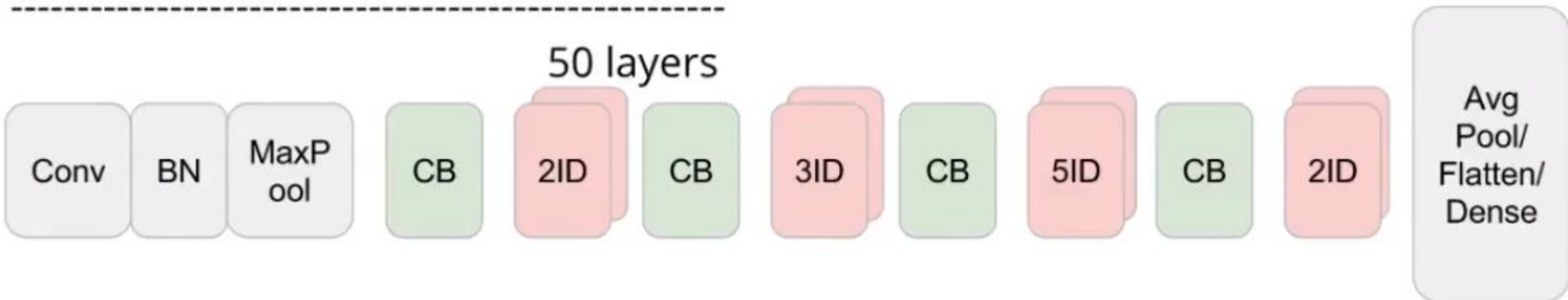
Conceito útil



Fonte: icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

ResNet

16 ResNet Blocks = 16 x 3 = 48 layers
+ 1 Conv Layer + 1 Dense = 2 layers
---
50 layers



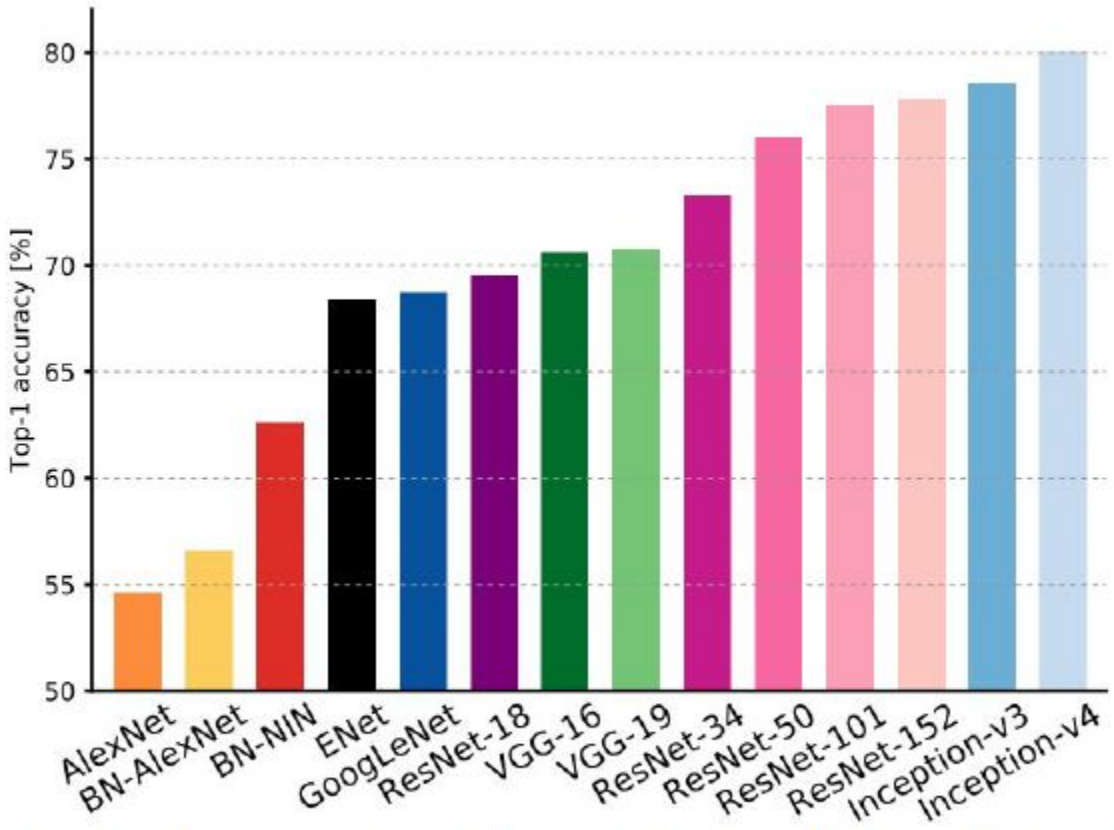
Para explorar

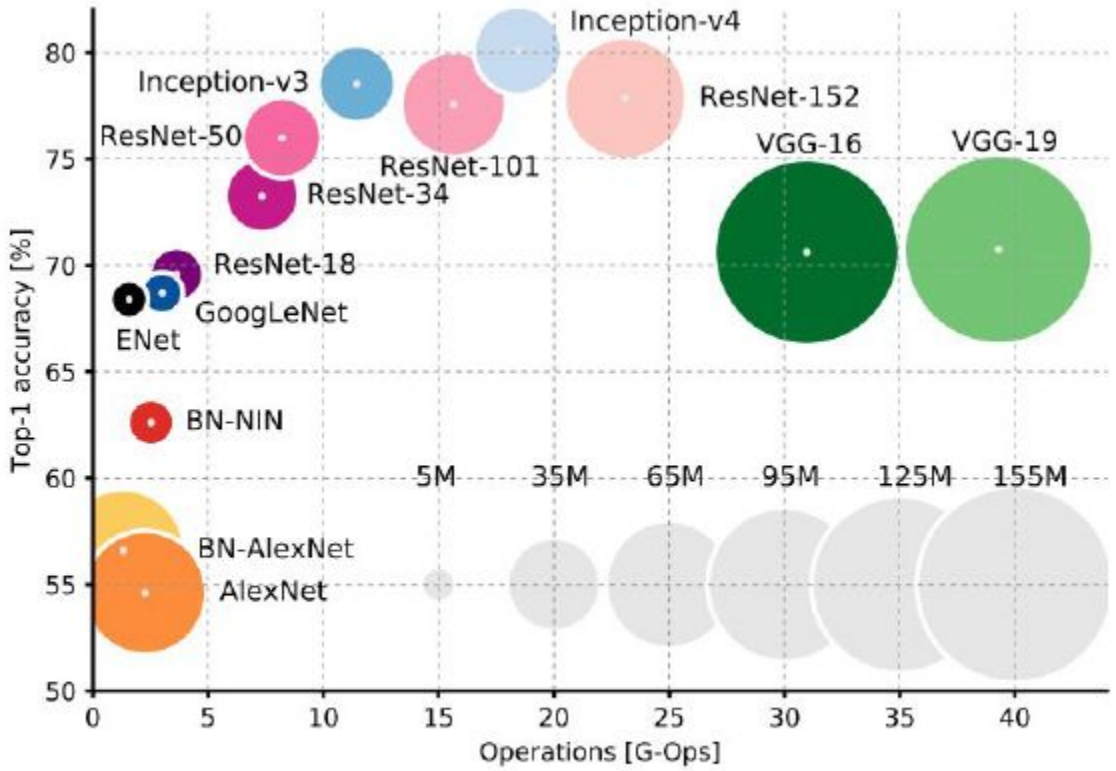
Chu, Brian, Daylen Yang, and Ravi Tadinada. "Visualizing residual networks." arXiv preprint arXiv:1701.02362 (2017).

Avisos práticos ao usar *ConvNets*

Comparação entre arquiteturas

Comparação

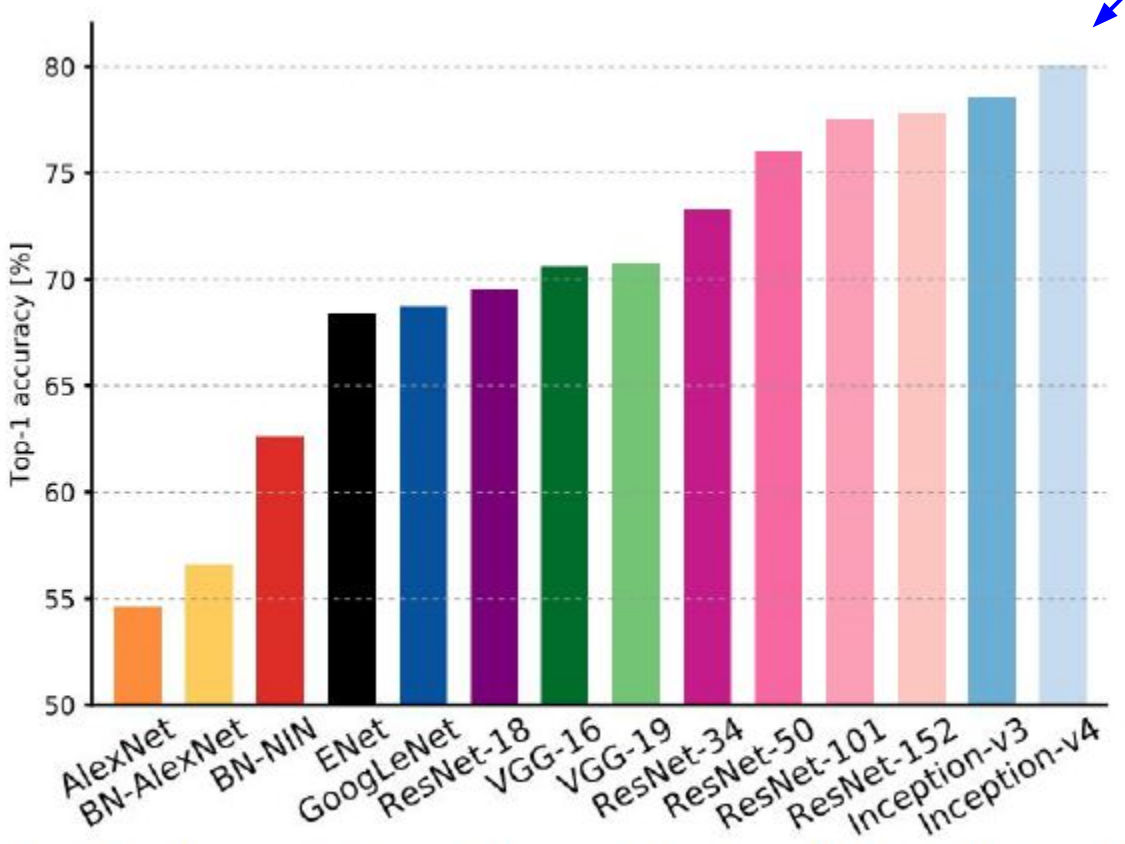


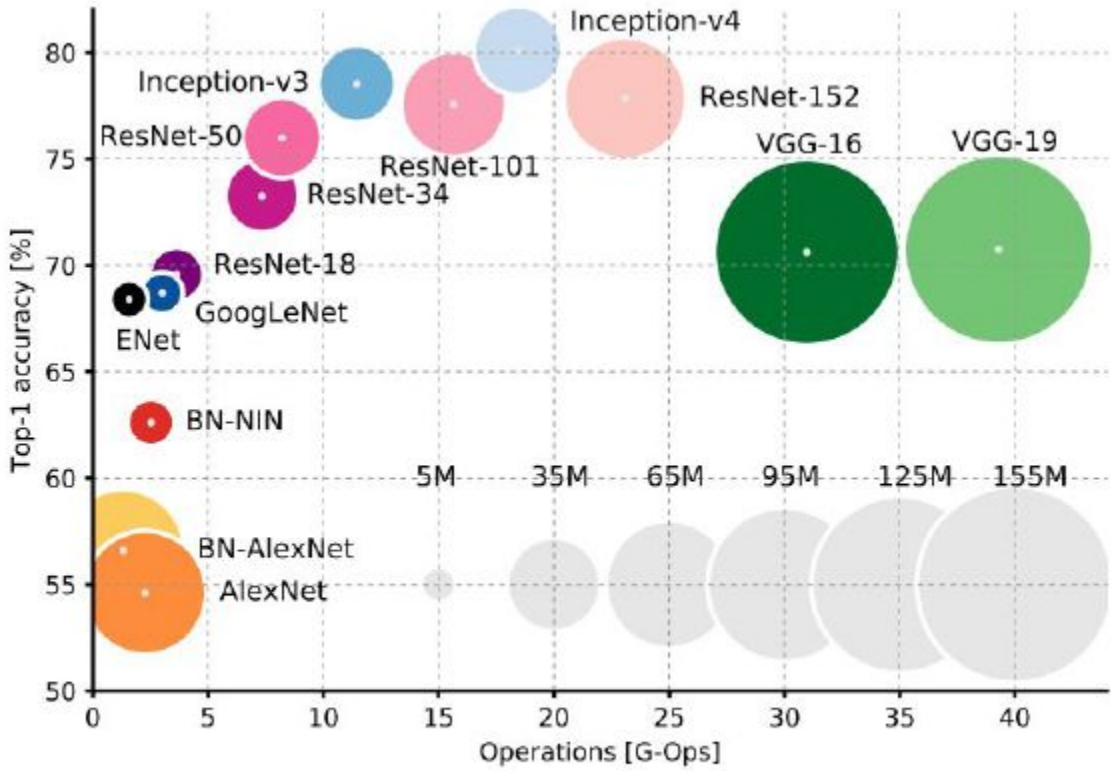


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparação

Inception-v4: ResNet + Inception

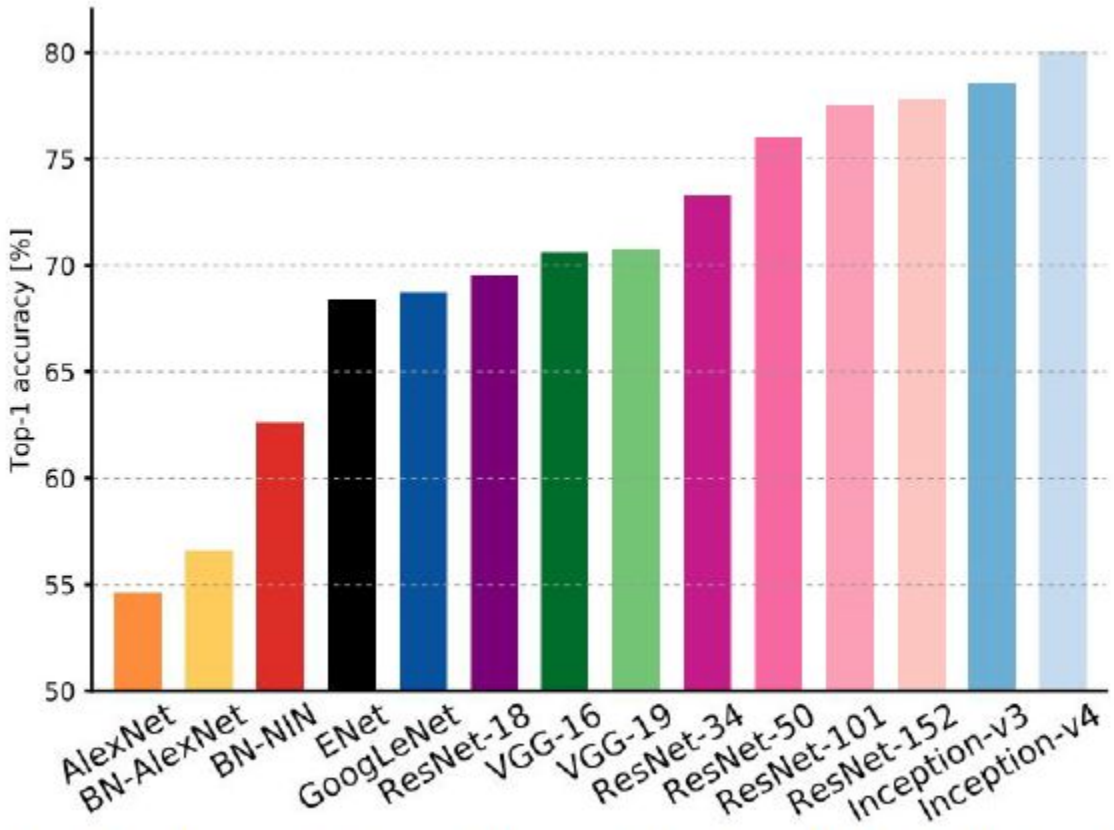


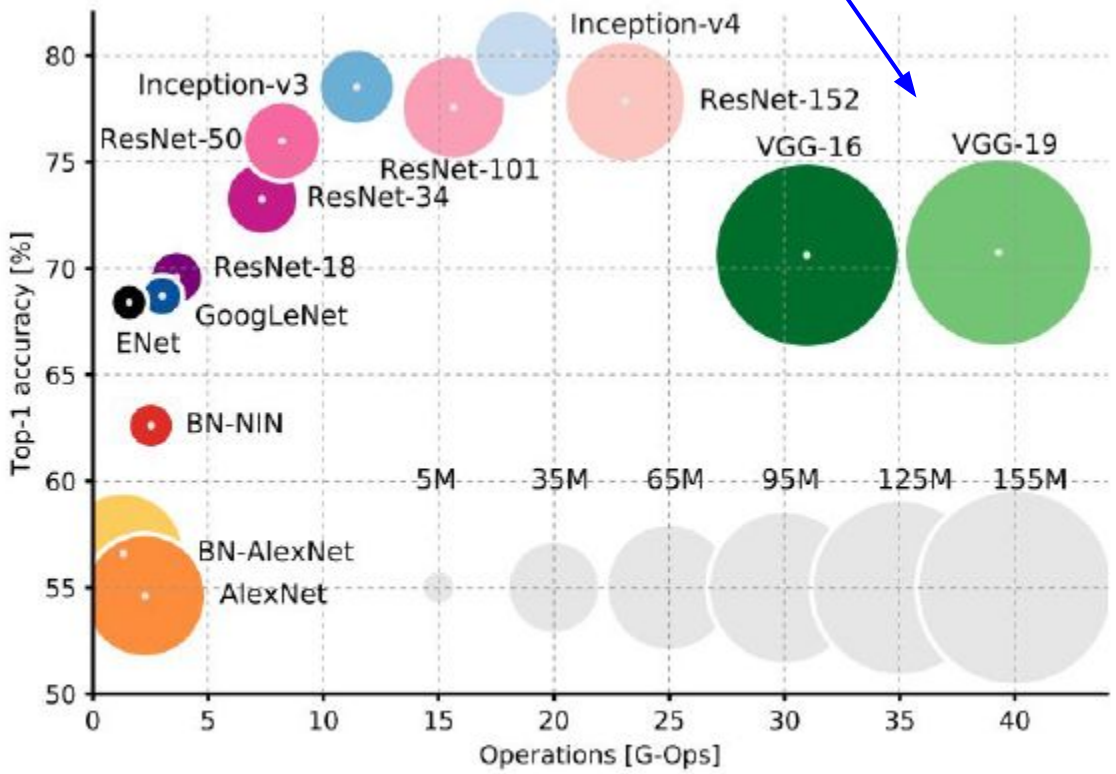


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparação

VGG: maior uso de memória, mais operações

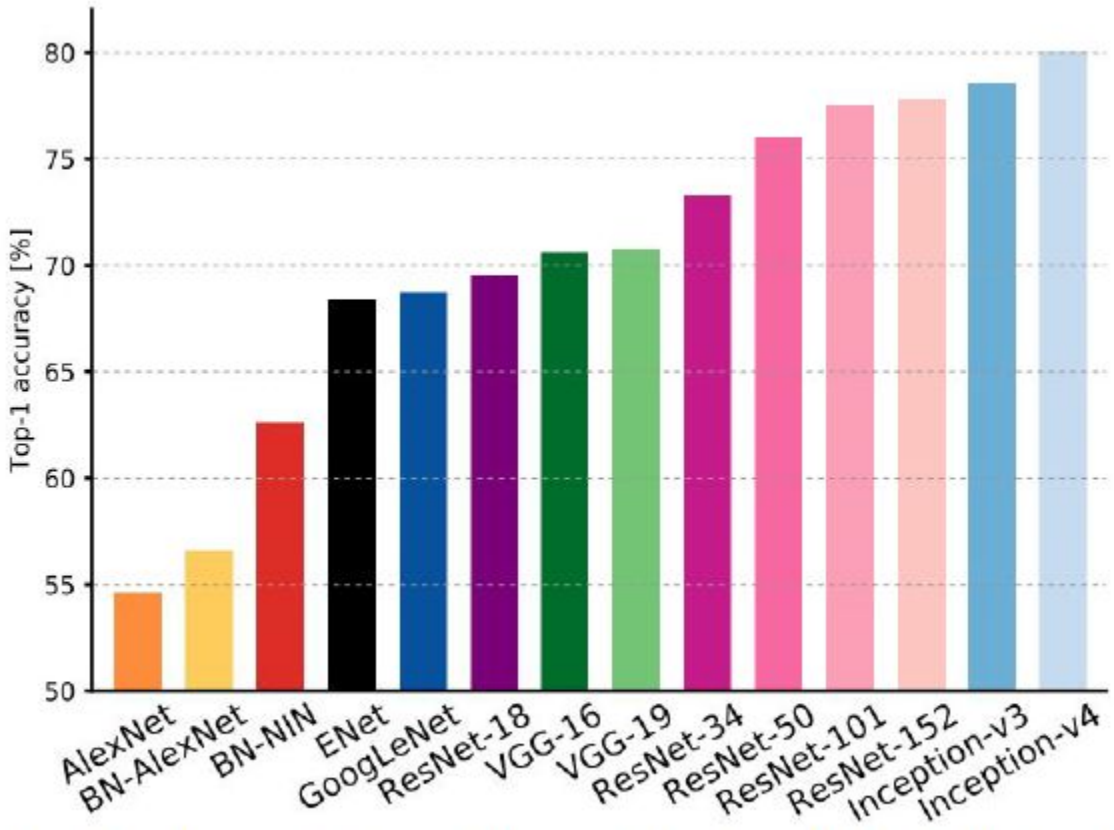


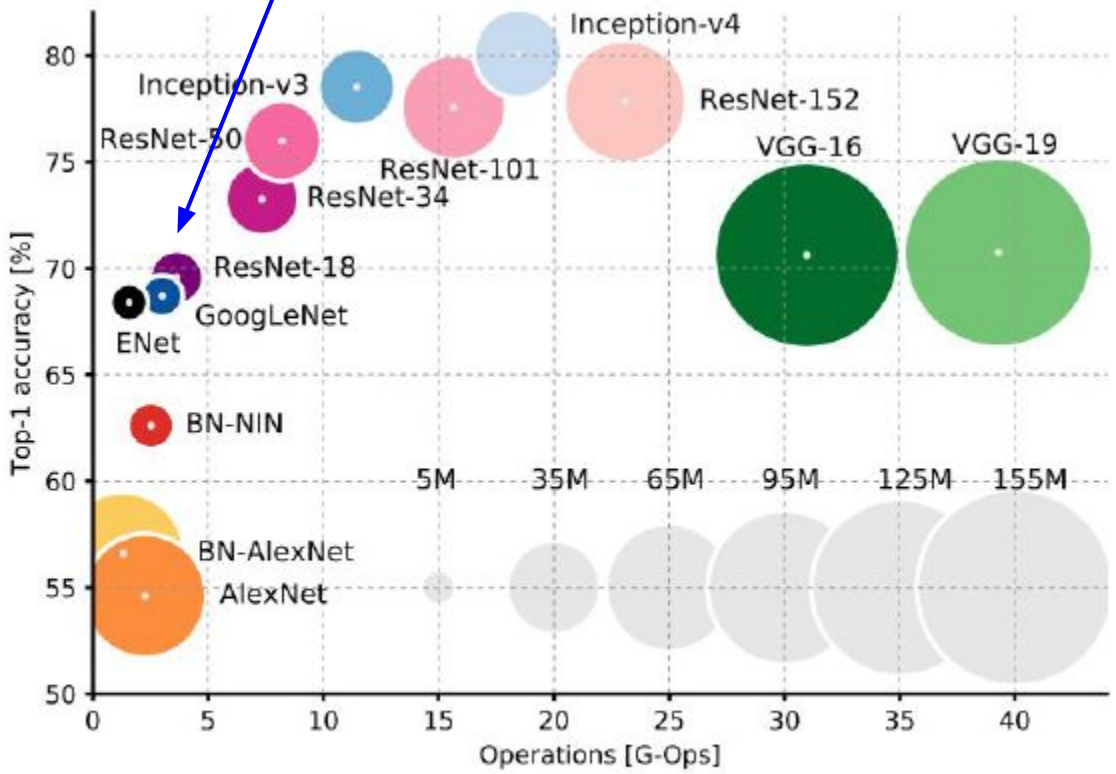


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparação

GoogLeNet: mais eficiente

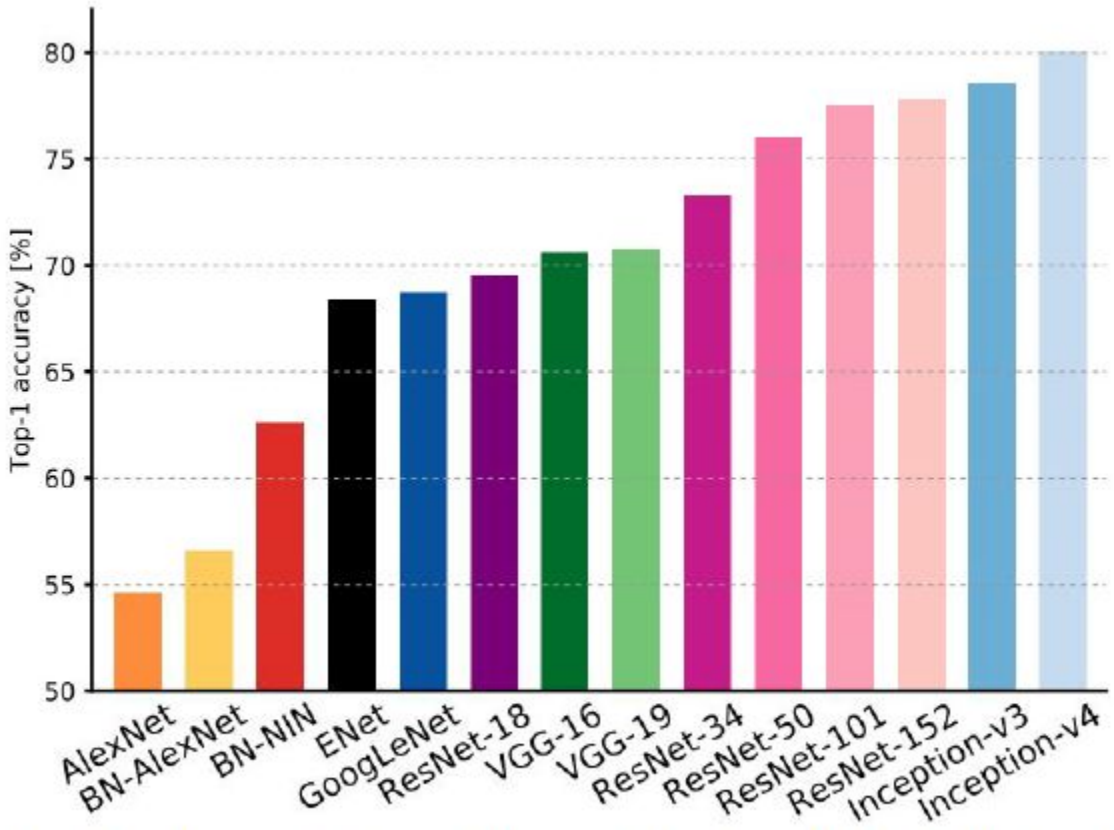


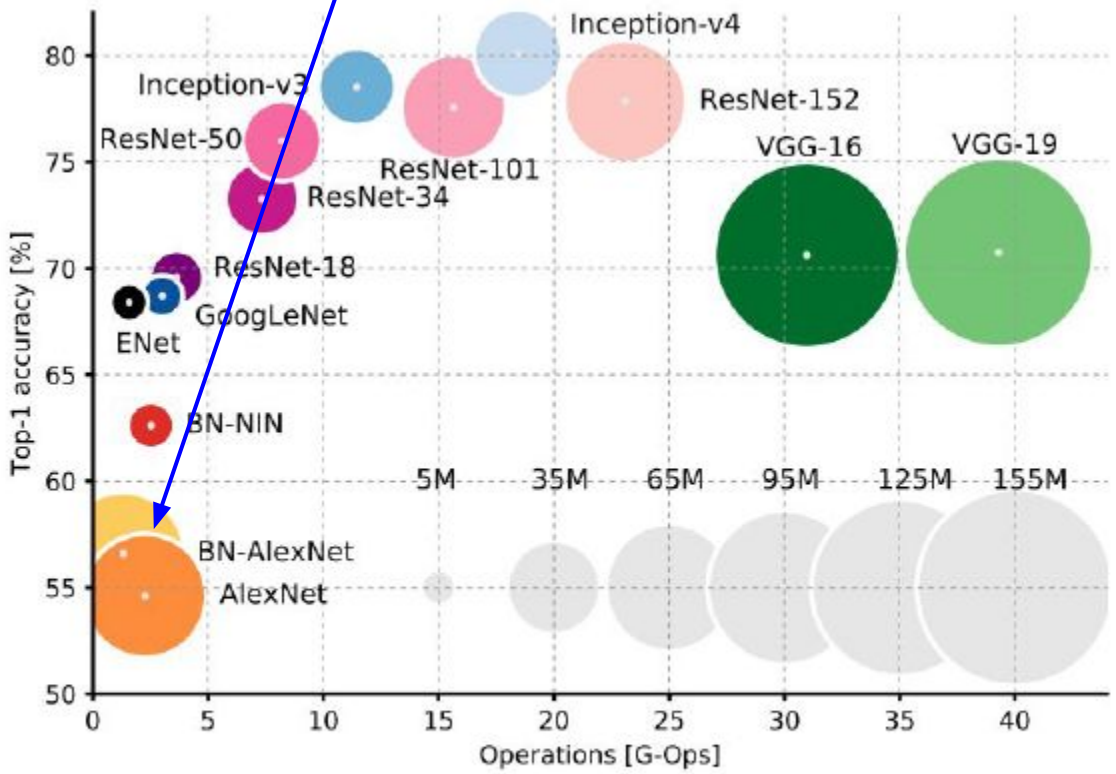


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparação

AlexNet: baixo # de operações, mas pesado em memória e baixa acurácia

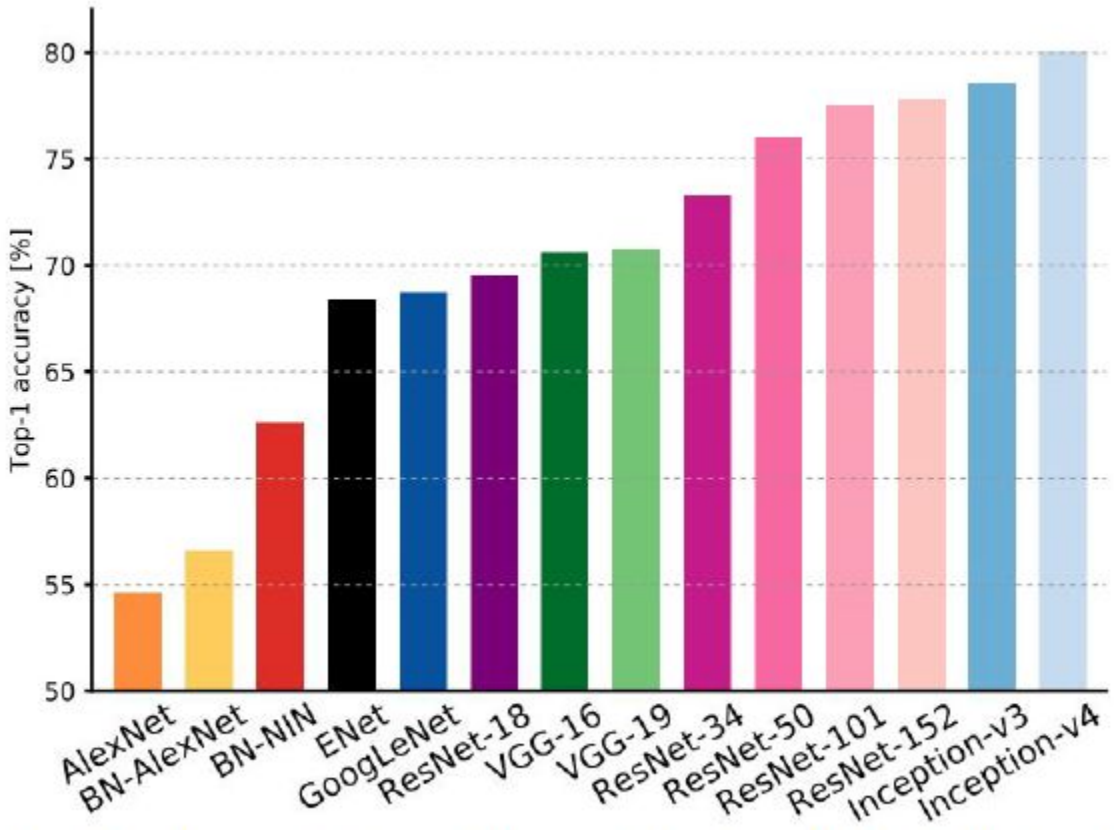


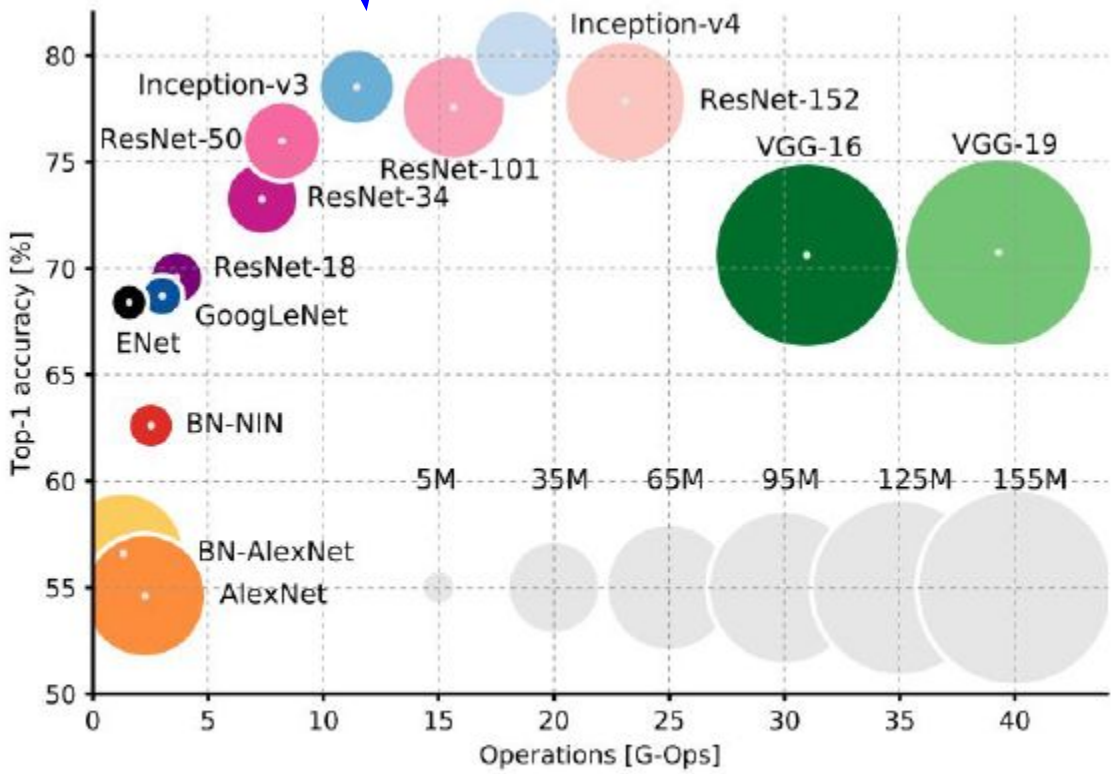


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparação

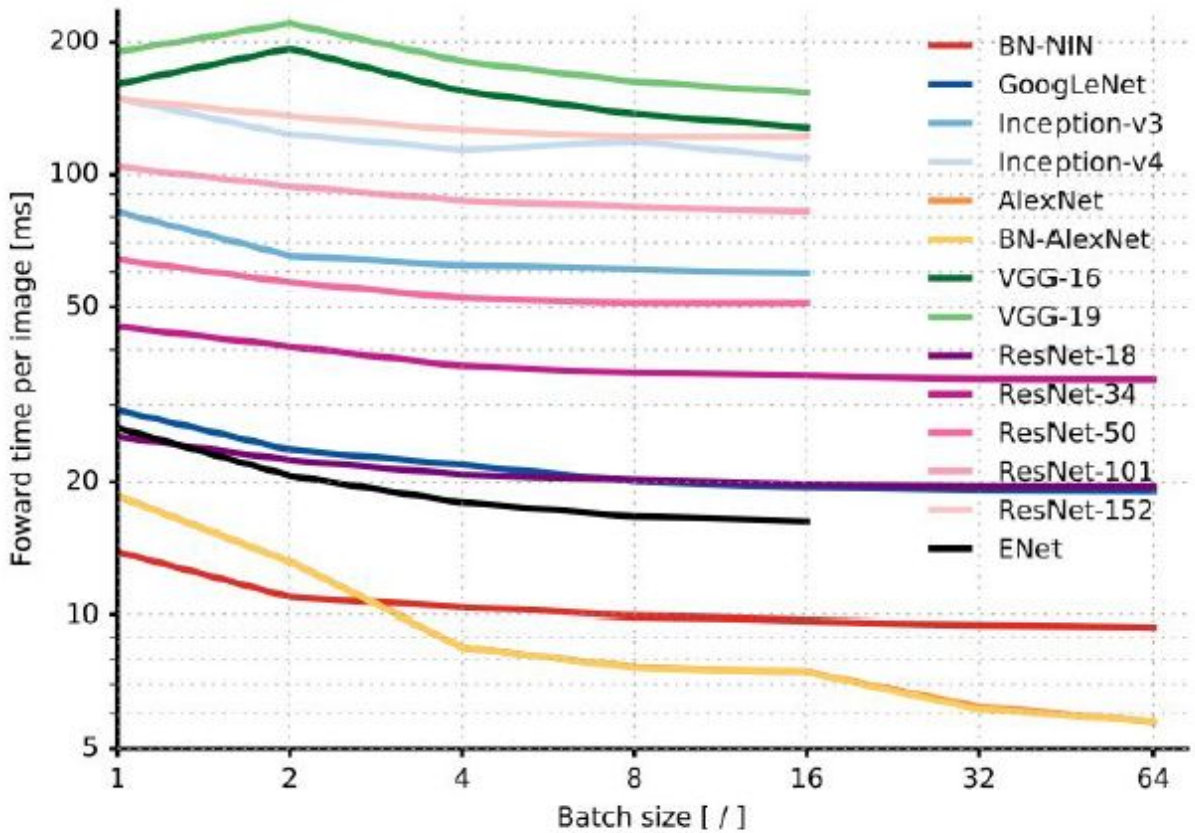
ResNet: eficiência moderada dependendo do modelo, alta acurácia

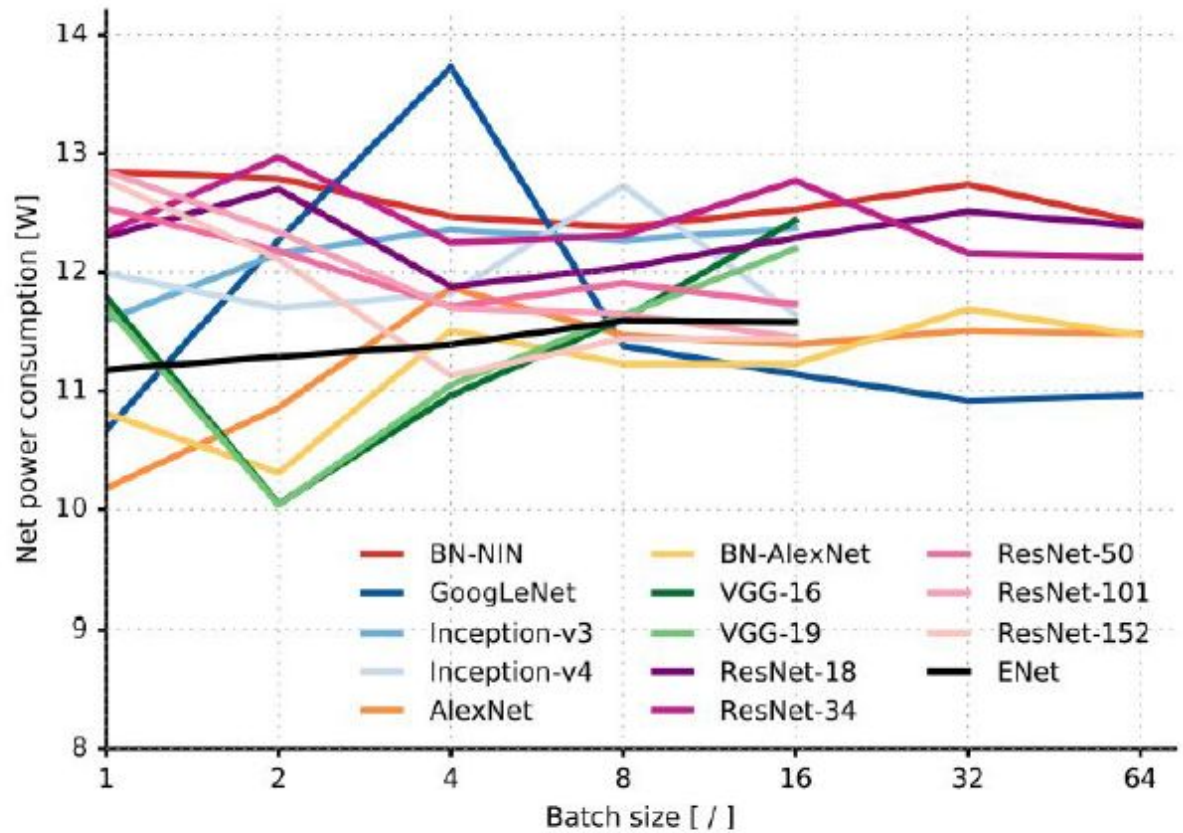




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparação





An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Best paper award

DENSELY CONNECTED CONVOLUTIONAL NETWORKS

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

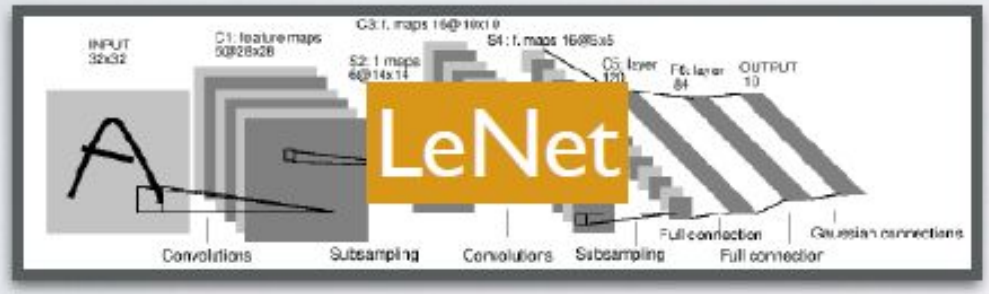
Cornell University

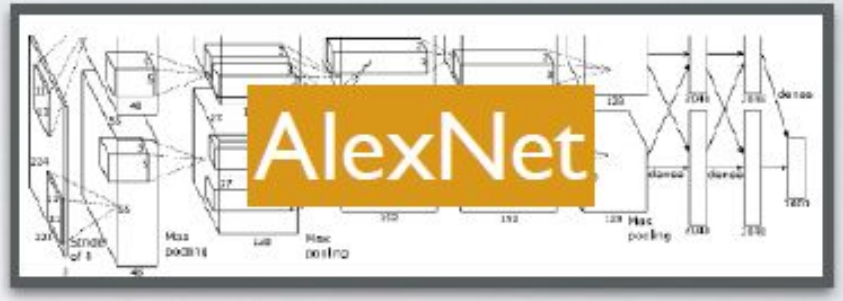
Tsinghua University

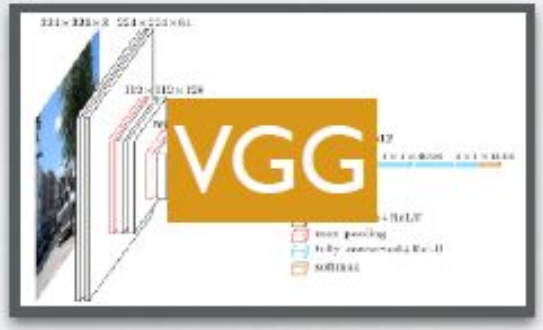
Facebook AI Research

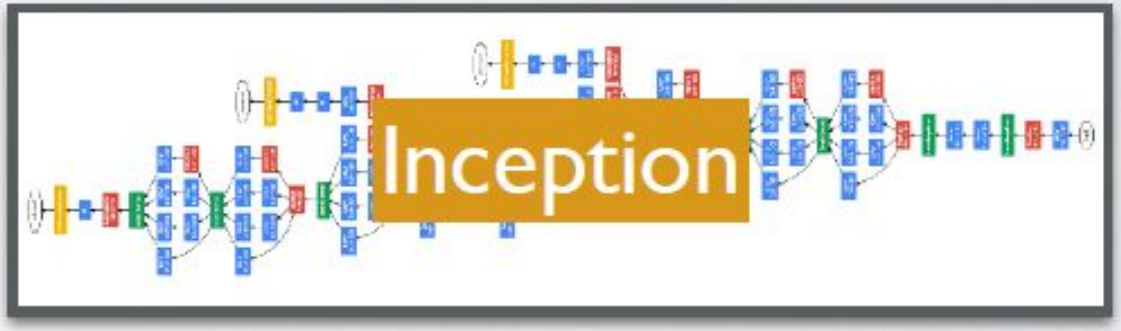
CVPR 2017

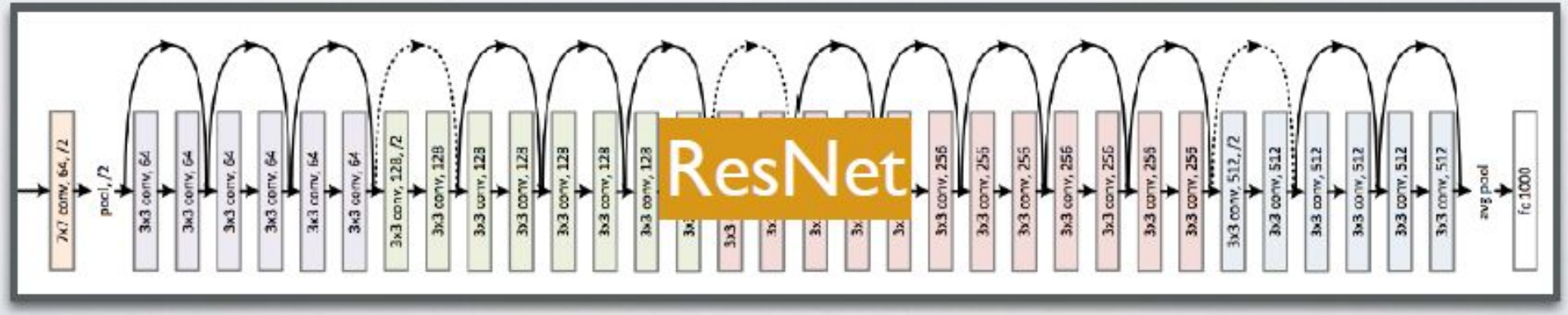
CONVOLUTIONAL NETWORKS



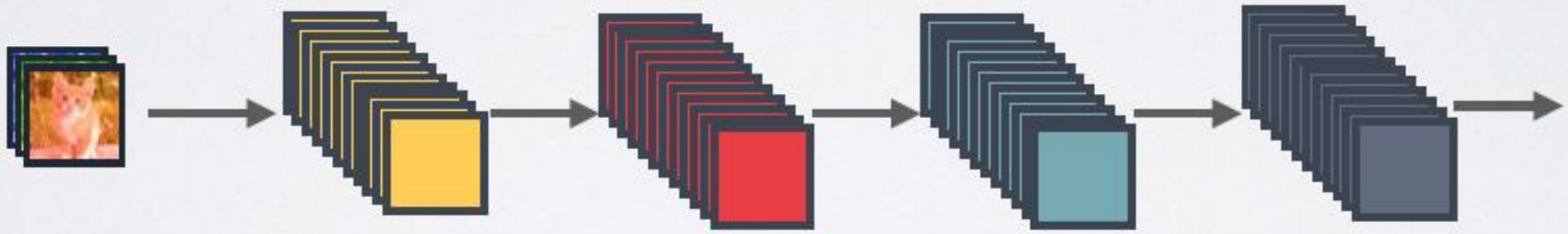






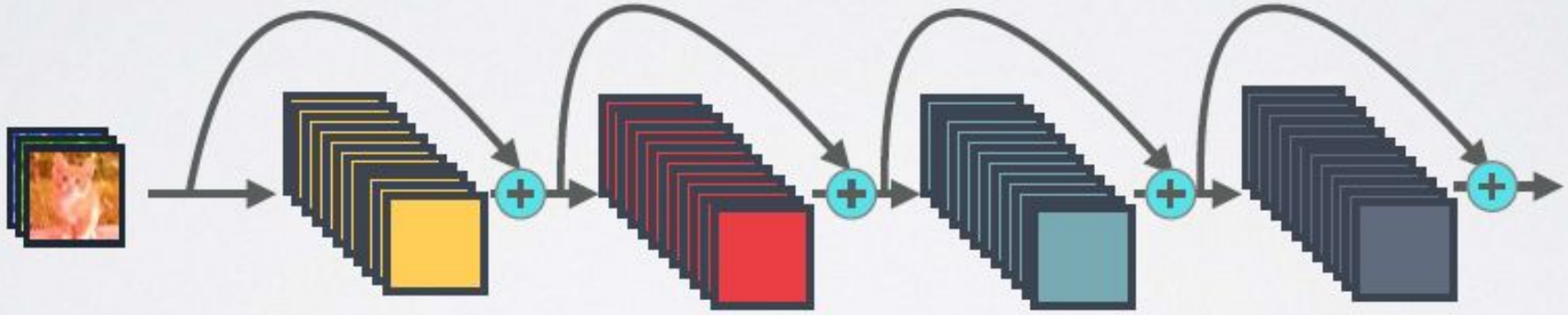


STANDARD CONNECTIVITY



RESNET CONNECTIVITY

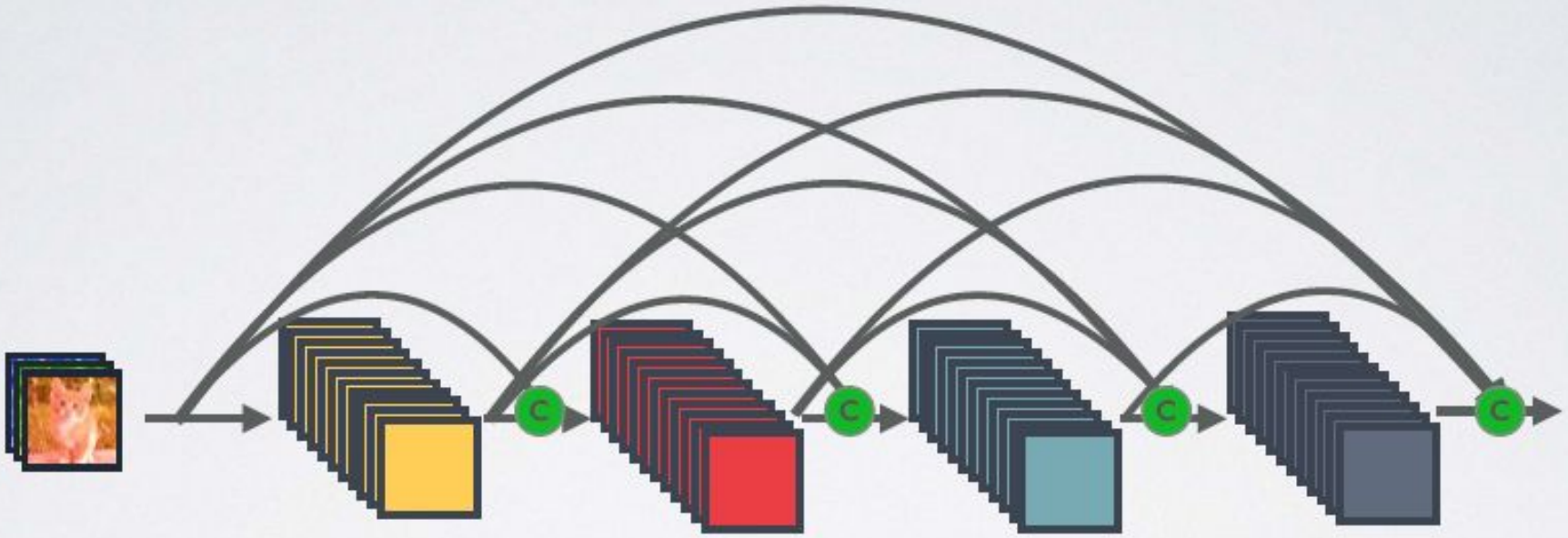
Identity mappings promote gradient propagation.



⊕ : Element-wise addition

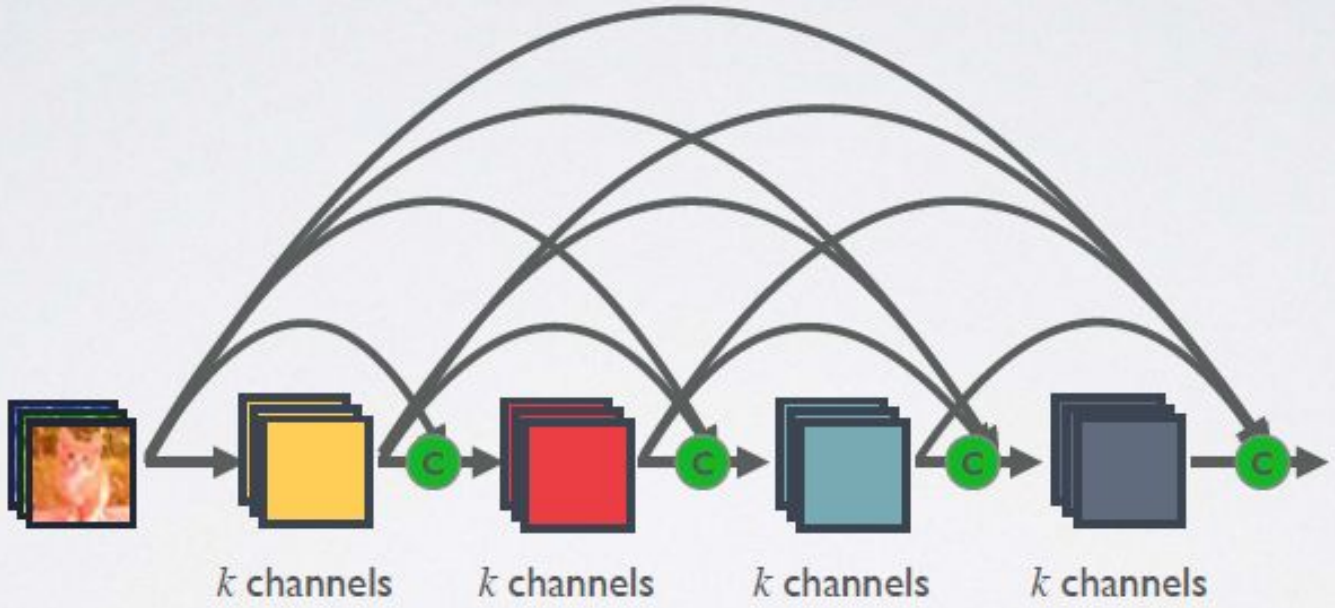
Deep residual learning for image recognition: [He, Zhang, Ren, Sun] (CVPR 2015)

DENSE CONNECTIVITY



C : Channel-wise concatenation

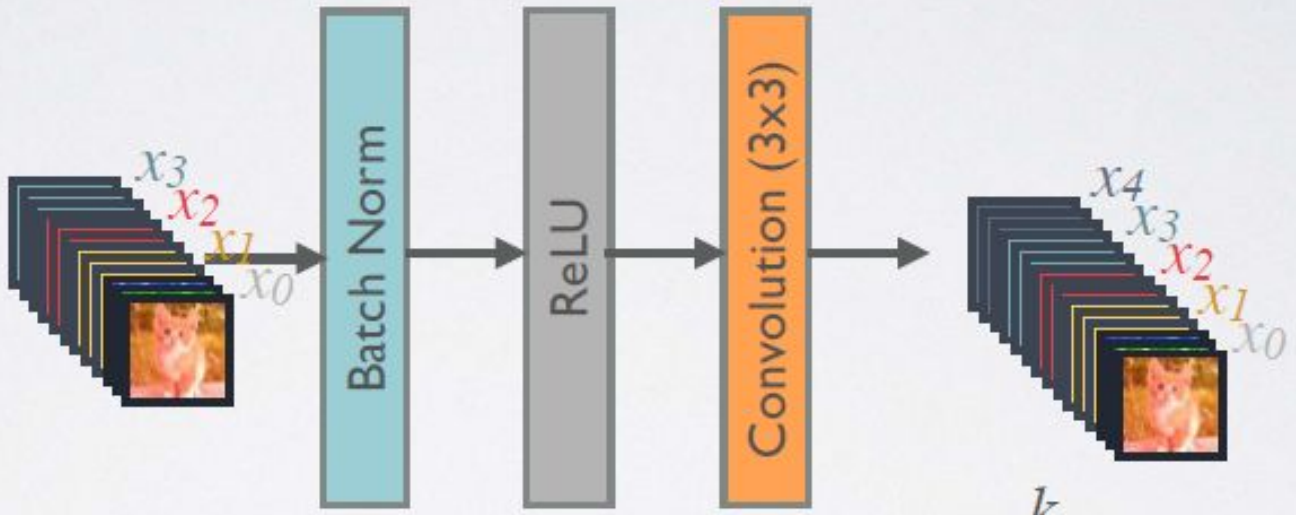
DENSE AND SLIM



k : Growth Rate

FORWARD PROPAGATION

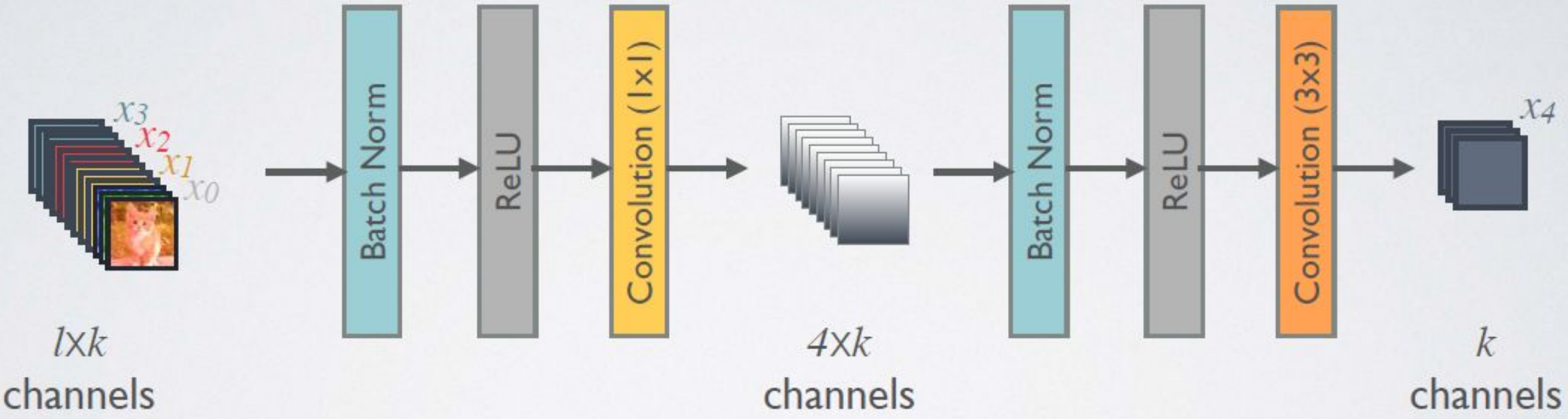
COMPOSITE LAYER IN DENSENET



$$x_5 = h_5([x_0, \ldots, x_4])$$

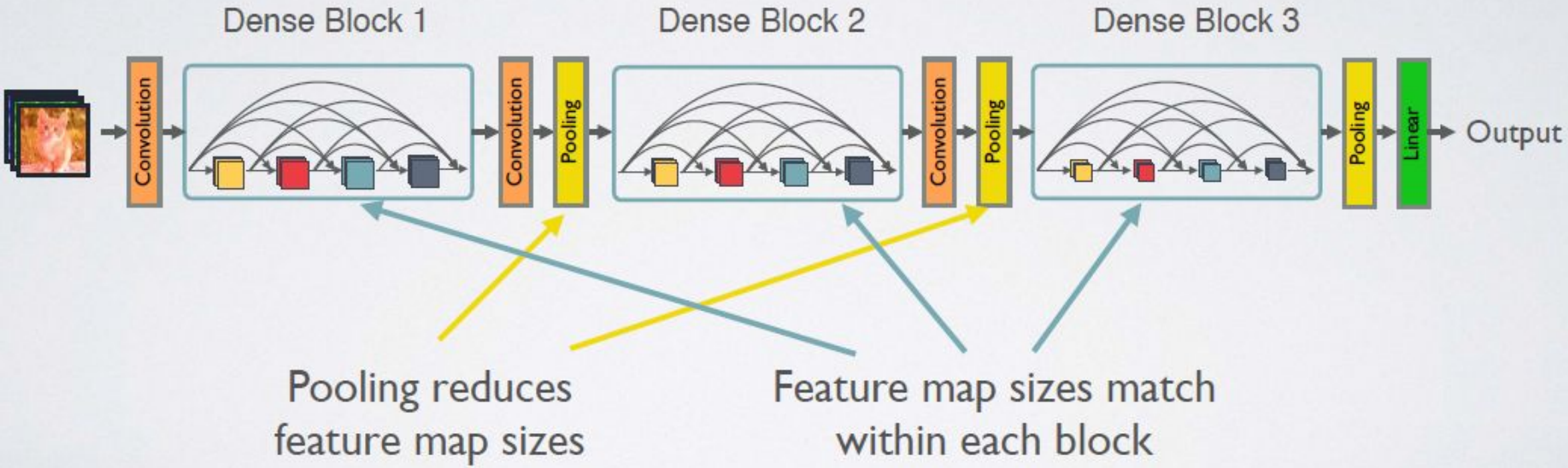
COMPOSITE LAYER IN DENSENET

WITH BOTTLENECK LAYER



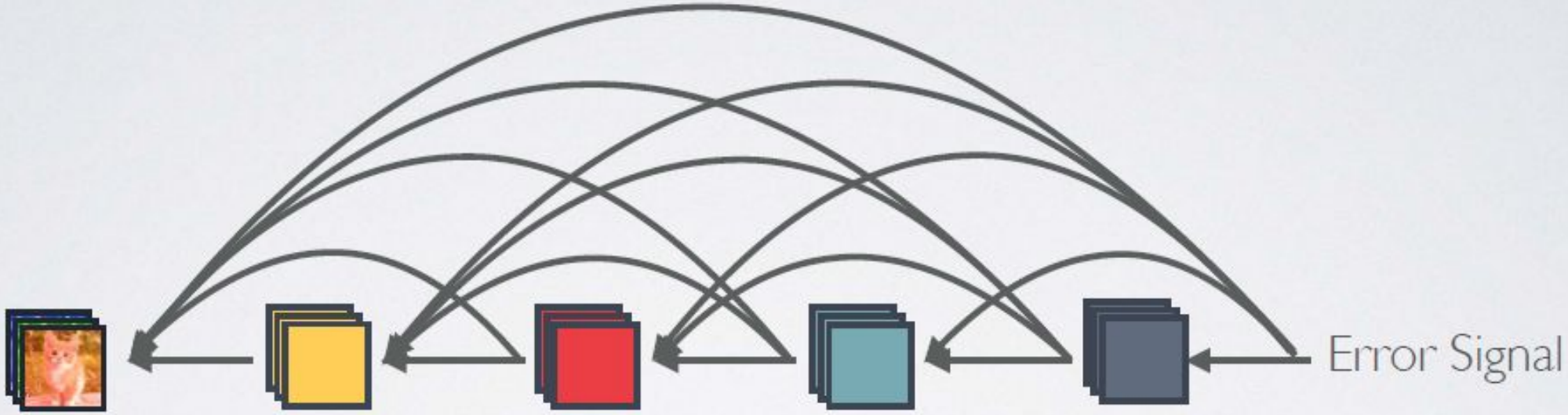
Higher parameter and computational efficiency

DENSENET


Dense Block 1
Dense Block 2
Dense Block 3
Convolution
Convolution
Pooling
Convolution
Pooling
Pooling
Linear
Output
Pooling reduces feature map sizes
Feature map sizes match within each block

ADVANTAGES OF DENSE CONNECTIVITY

ADVANTAGE 1: STRONG GRADIENT FLOW

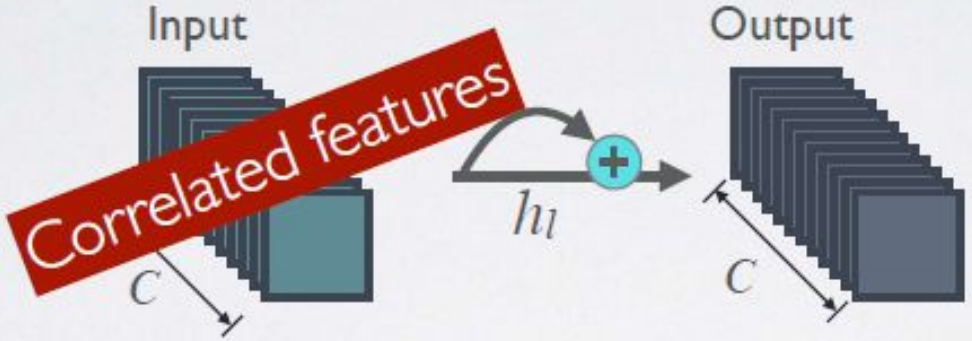


Implicit "deep supervision"

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

ADVANTAGE 2: PARAMETER & COMPUTATIONAL EFFICIENCY

ResNet connectivity:



DenseNet connectivity:

#parameters:

$O(CxC)$

$k<<C$

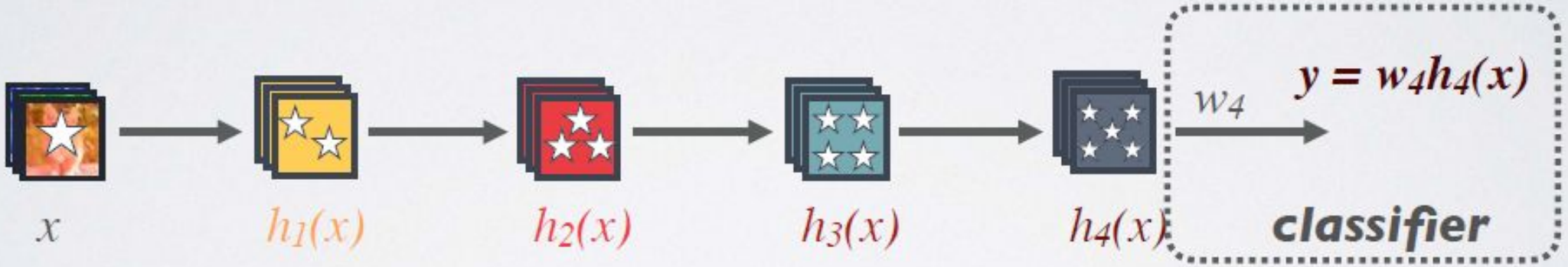
$O(lxkxk)$

k: Growth rate

ADVANTAGE 3: MAINTAINS LOW COMPLEXITY FEATURES

Standard Connectivity:

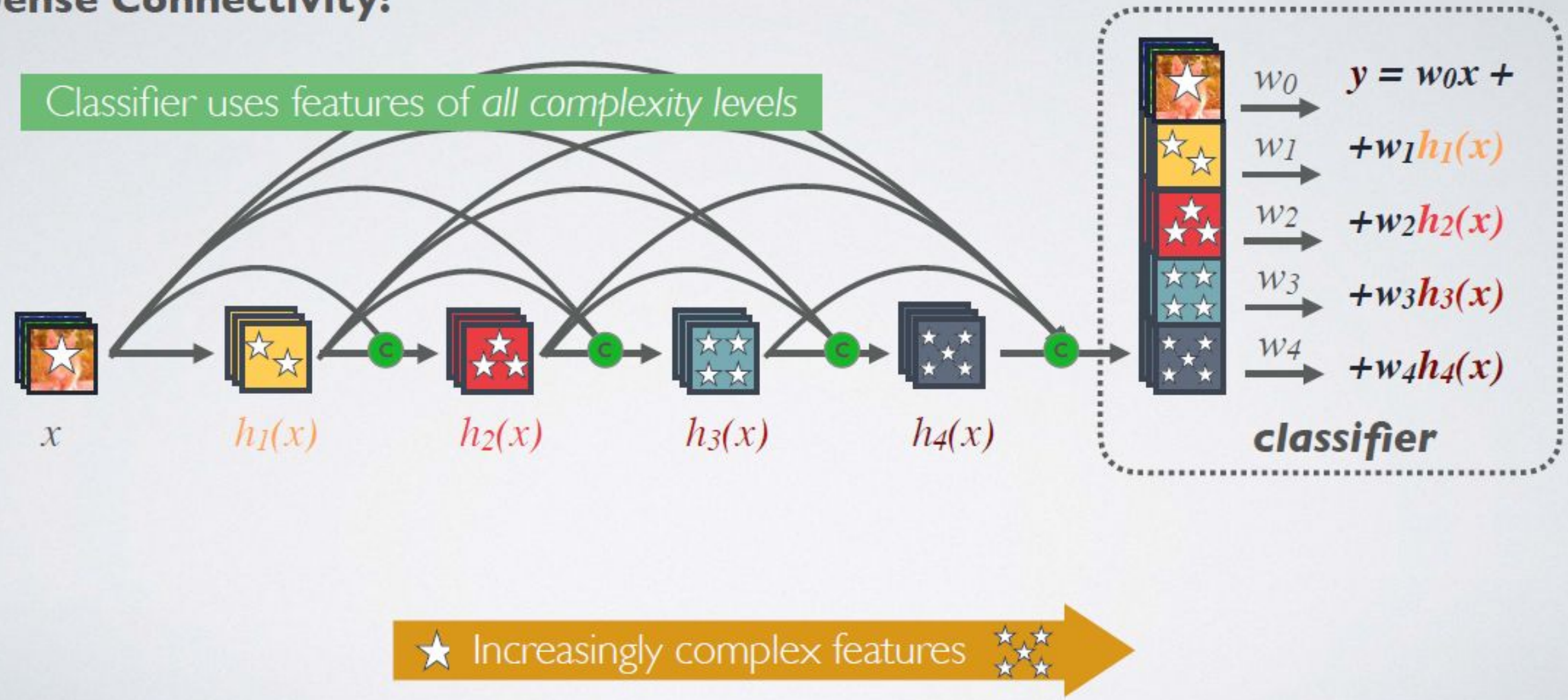
Classifier uses most complex (high level) features



Increasingly complex features

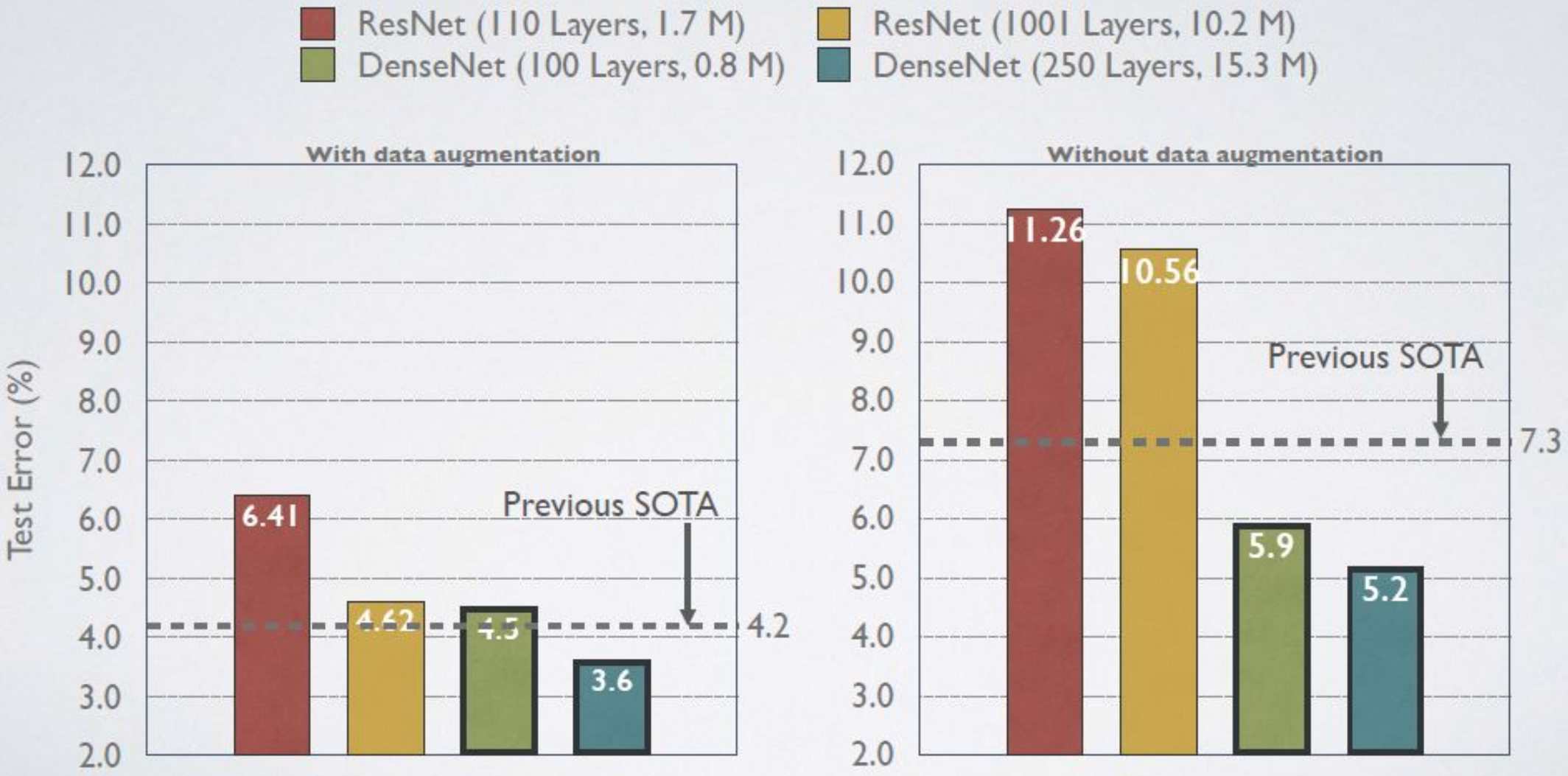
ADVANTAGE 3: MAINTAINS LOW COMPLEXITY FEATURES

Dense Connectivity:

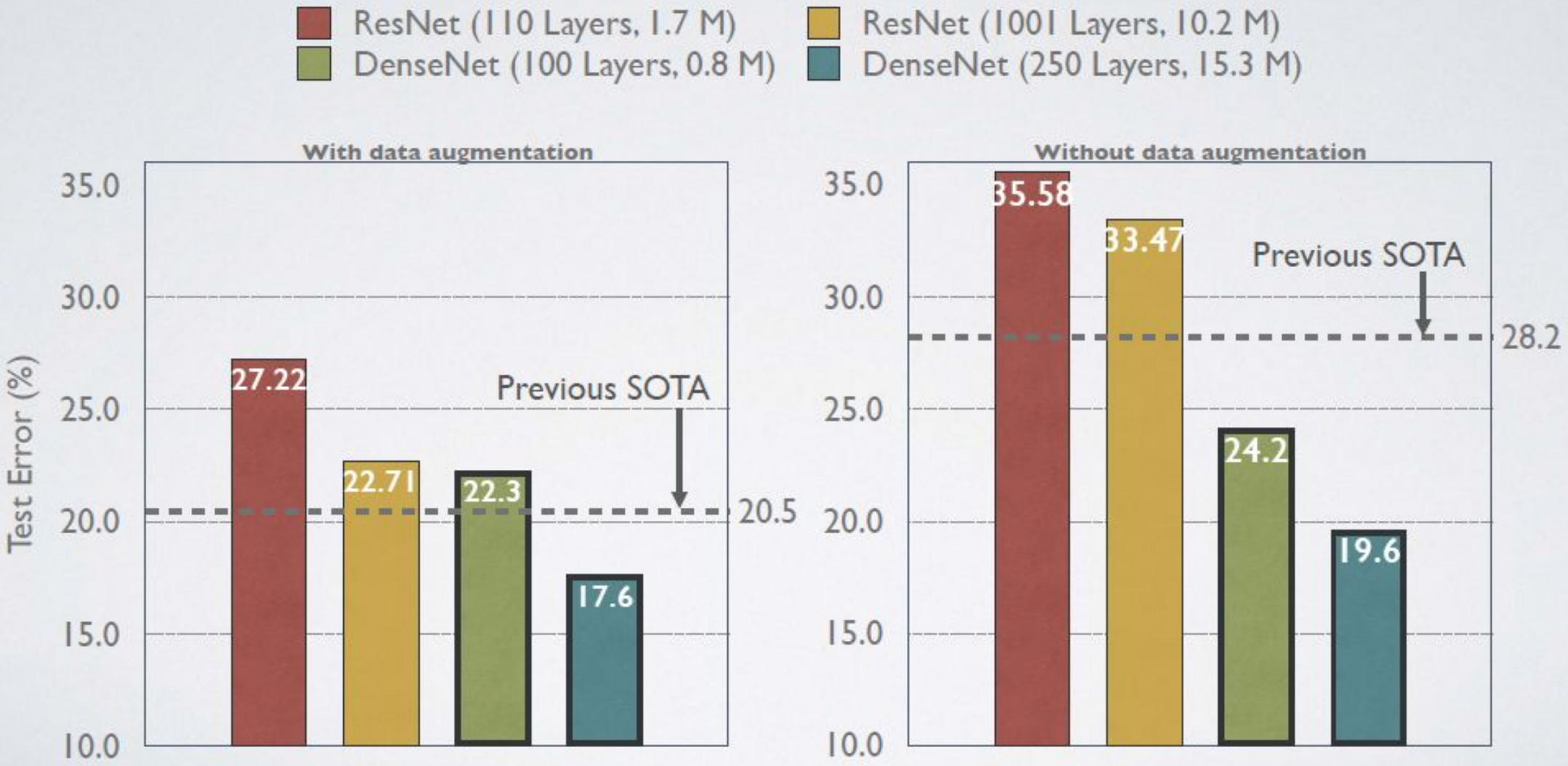


RESULTS

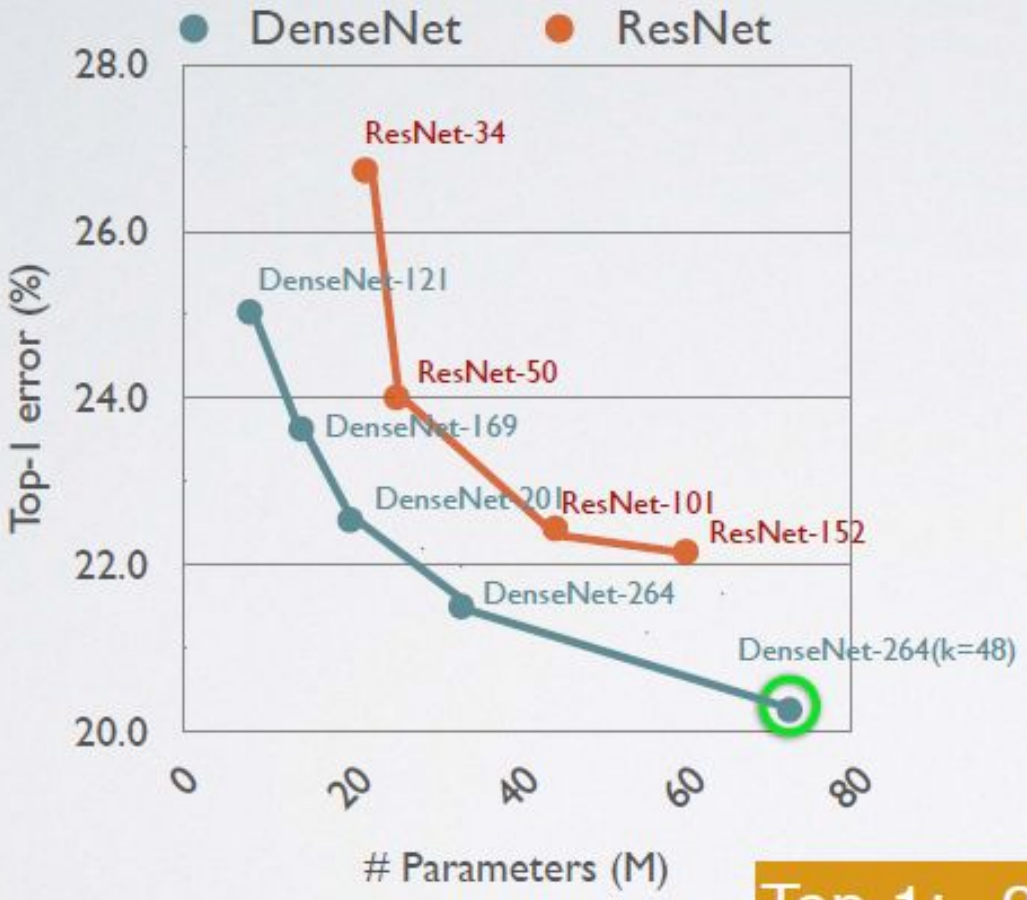
RESULTS ON **CIFAR-10**

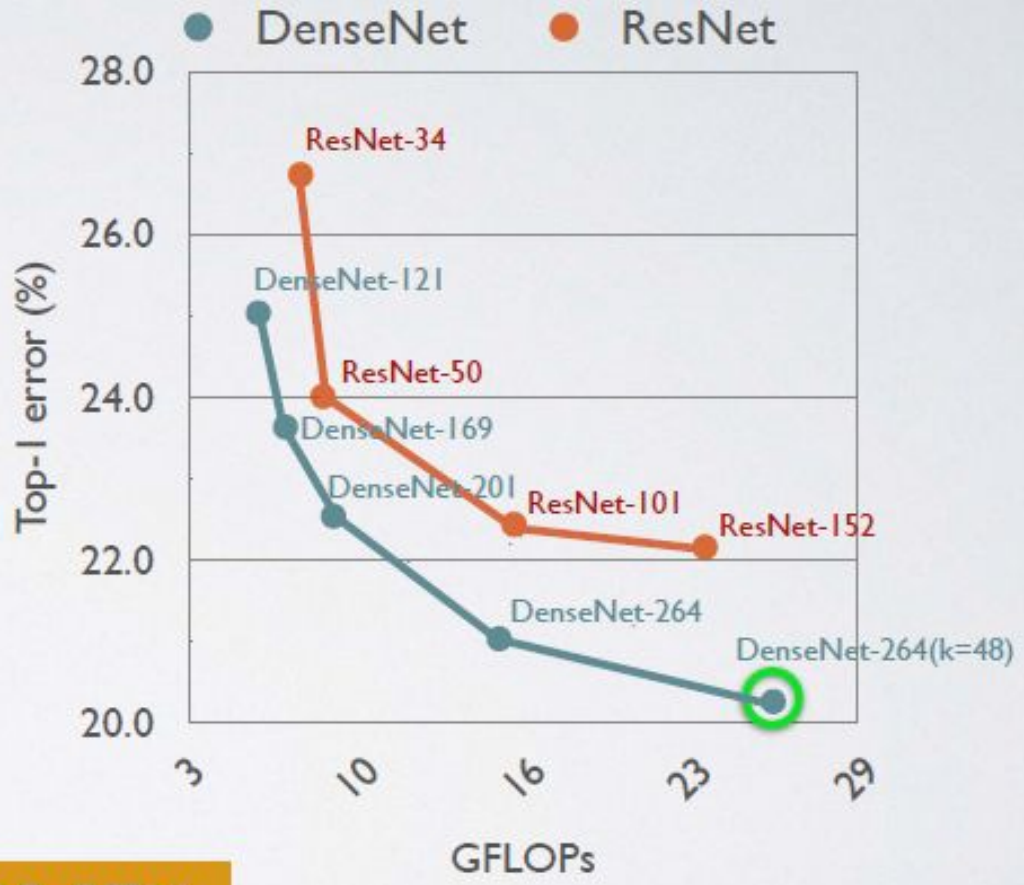


RESULTS ON **CIFAR-100**



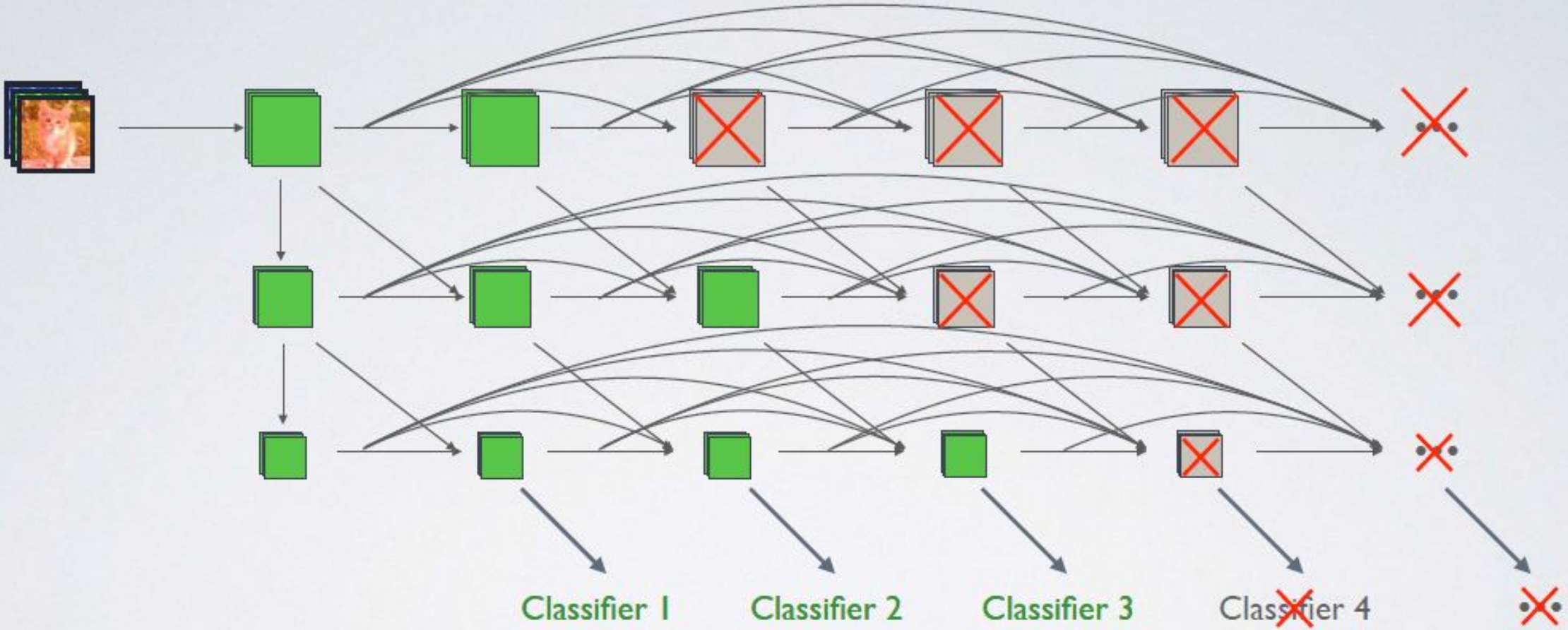
RESULTS ON **IMAGENET**





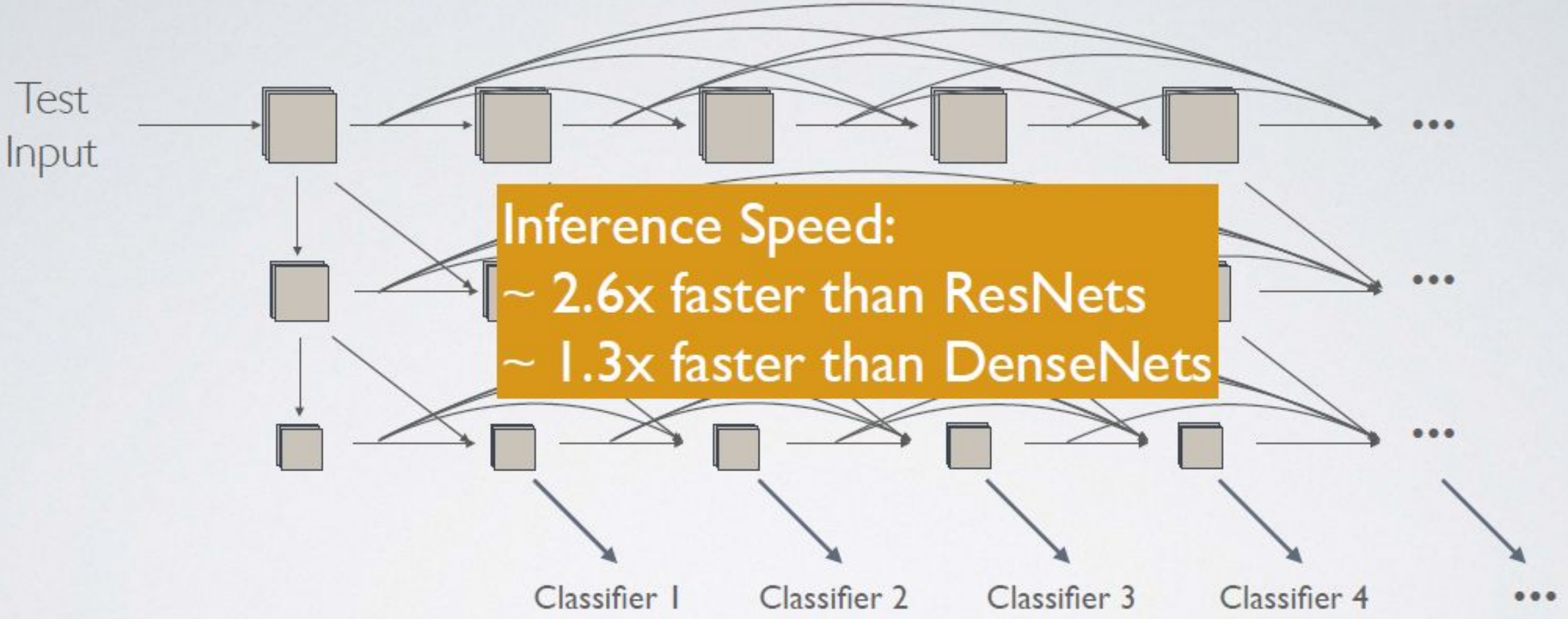
Top-1: 20.27%
Top-5: 5.17%

MULTI-SCALE DENSENET (Preview)



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

MULTI-SCALE DENSENET (Preview)



"Easy" examples

"Hard" examples

REFERENCES

- Kaiming He, et al. "Deep residual learning for image recognition" CVPR 2016
- Chen-Yu Lee, et al. "Deeply-supervised nets" *AISTATS* 2015
- Gao Huang, et al. "Deep networks with stochastic depth" *ECCV* 2016
- Gao Huang, et al. "Multi-Scale Dense Convolutional Networks for Efficient Prediction" *arXiv preprint arXiv:1703.09844* (2017)
- Geoff Pleiss, et al. "Memory-Efficient Implementation of DenseNets", *arXiv preprint arXiv: 1707.06990* (2017)