INF-0618
Tópicos em Aprendizado de Máquina II

Aula 4 – Arquiteturas CNN

Profa. Fernanda Andaló

2018

Instituto de Computação - Unicamp

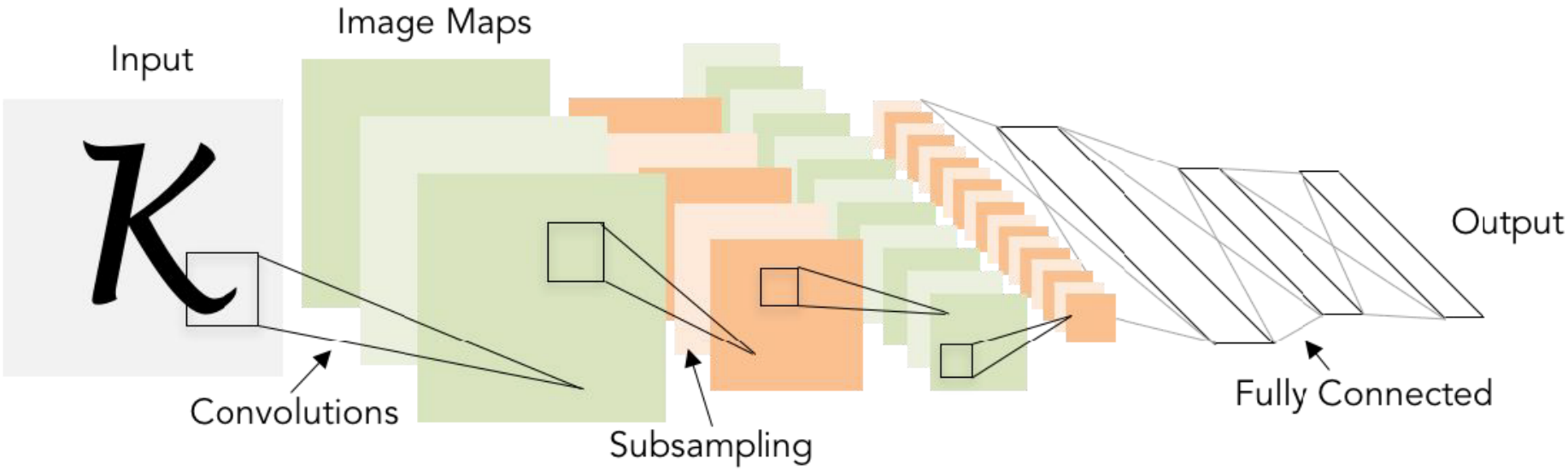
Arquiteturas CNN

Arquiteturas CNN

- LeNet
- AlexNet
- ZeNet
- VGGNet
- GoogLeNet (Inception-v1)
- ResNet
- SqueezeNet

LeNet-5

[LeCun et al., 1998]



Conv filters: 5x5, stride 1
Pooling: 2x2, stride 2
Arquitetura: [CONV-POOL-CONV-POOL-FC-FC]

AlexNet

[Krizhevsky et al., 2012]

Arquitetura:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

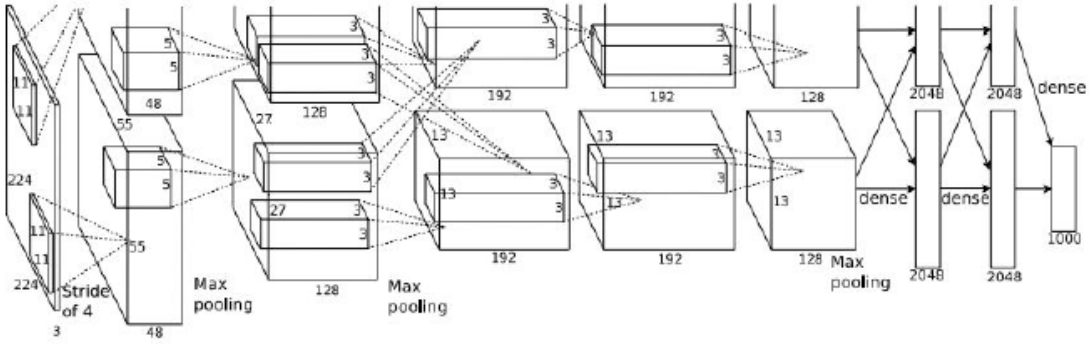
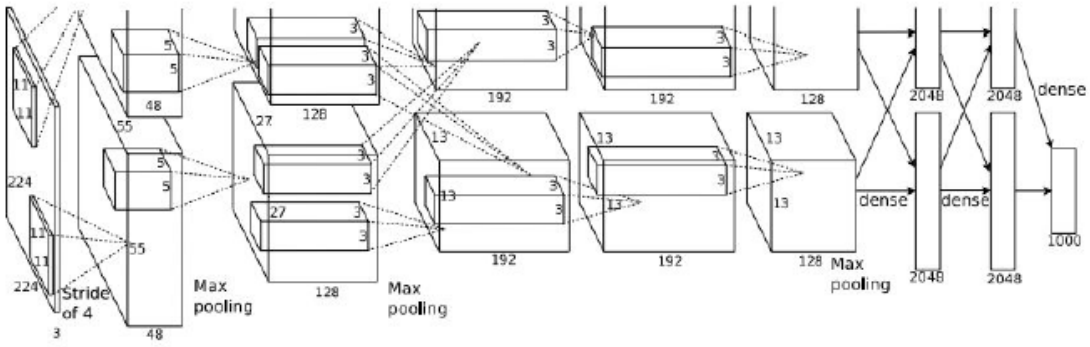


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

AlexNet

[Krizhevsky et al., 2012]



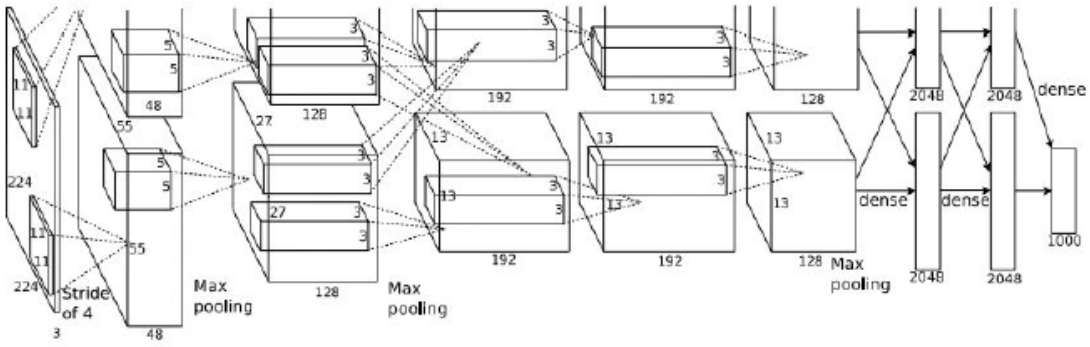
Input: imagens 227x227x3

Primeiro layer (CONV1): 96 filtros 11x11, com stride de 4.

=> Qual o tamanho do volume de saida? Dica: (227-11)/4+1 = 55

AlexNet

[Krizhevsky et al., 2012]



Input: imagens 227x227x3

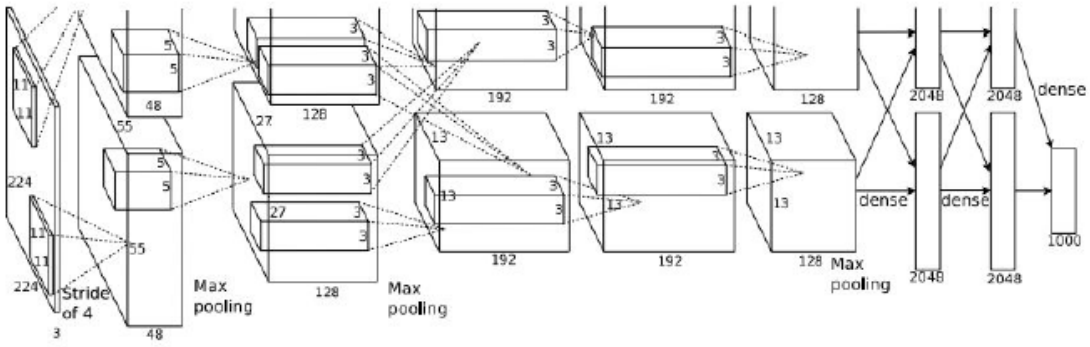
Primeiro layer (CONV1): 96 filtros 11x11, com stride de 4.

=> Volume de saída: [55x55x96]

=> Qual o total de parâmetros neste layer?

AlexNet

[Krizhevsky et al., 2012]



Input: imagens 227x227x3

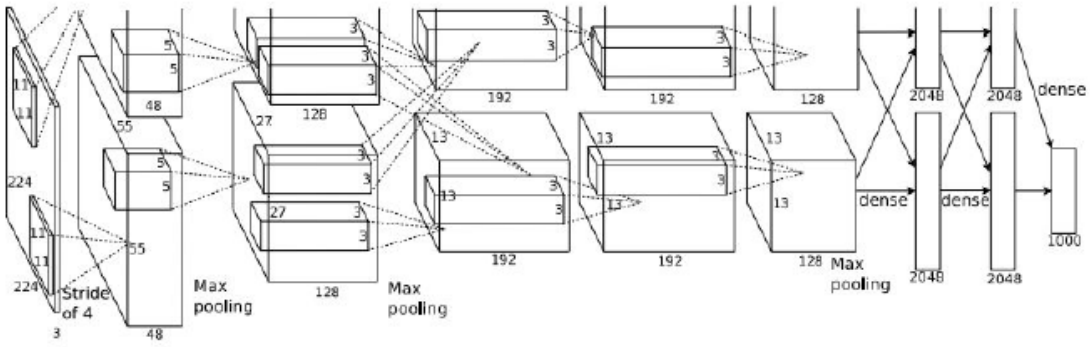
Primeiro layer (CONV1): 96 filtros 11x11, com stride de 4.

=> Volume de saída: [55x55x96]

=> Parâmetros: (11*11*3)*96 = 35K

AlexNet

[Krizhevsky et al., 2012]



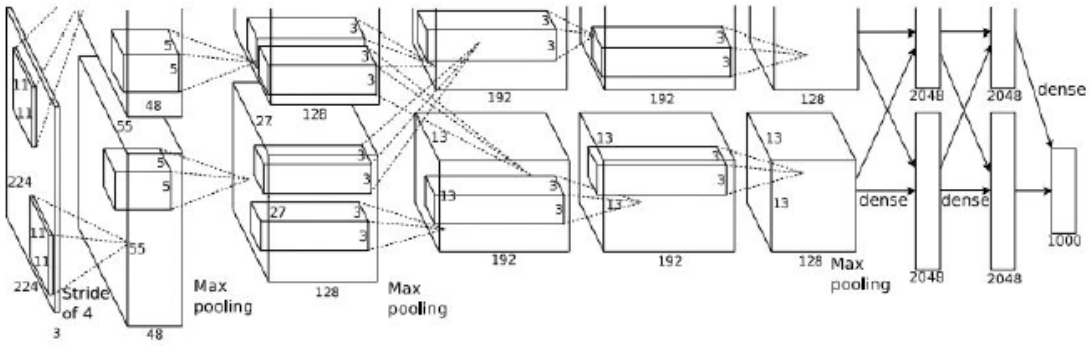
Input: imagens 227x227x3
Depois da CONV1: 55x55x96

Segundo layer (POOL1): filtros 3x3, com stride 2

=> Qual o tamanho do volume de saída? Dica: (55-3)/2+1 = 27

AlexNet

[Krizhevsky et al., 2012]



Input: imagens 227x227x3
Depois da CONV1: 55x55x96

Segundo layer (POOL1): filtros 3x3, com stride 2

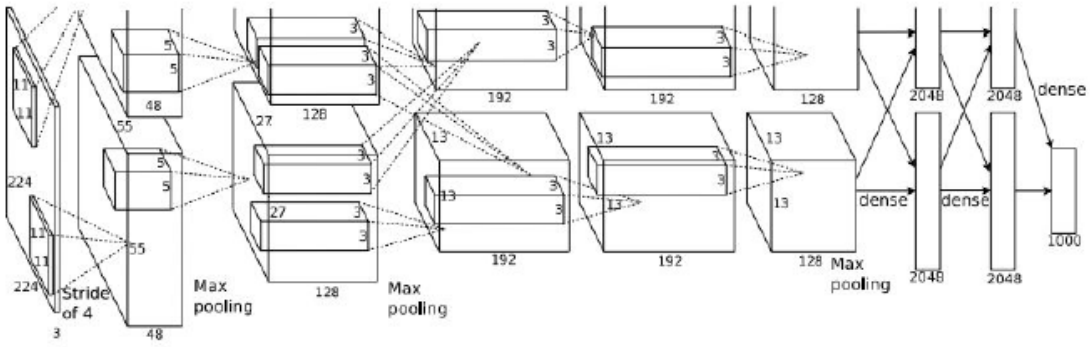
=> Volume de saída: [27x27x96]

=> Qual o total de parâmetros neste layer?

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

AlexNet

[Krizhevsky et al., 2012]



Input: imagens 227x227x3
Depois da CONV1: 55x55x96

Segundo layer (POOL1): filtros 3x3, com stride 2

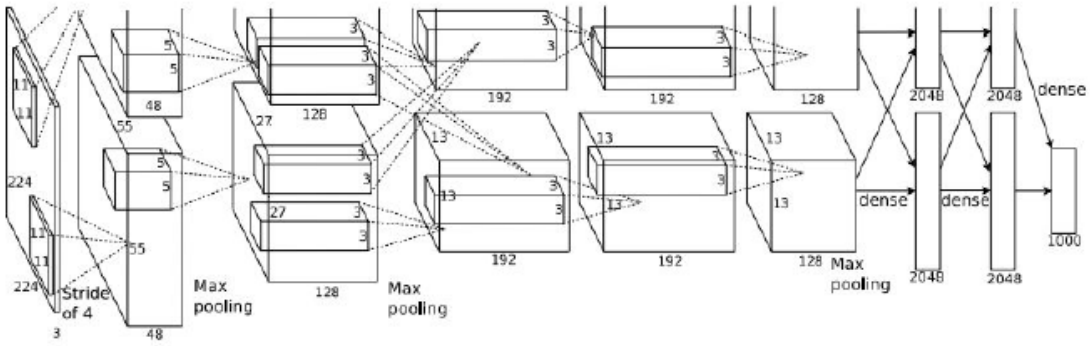
=> Volume de saída: [27x27x96]

=> Parâmetros: 0

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

AlexNet

[Krizhevsky et al., 2012]

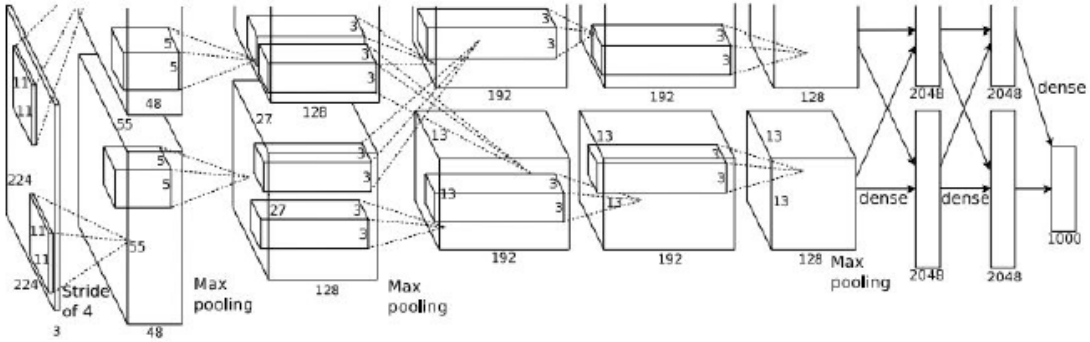


Input: imagens 227x227x3
Depois da CONV1: 55x55x96
Depois da POOL1: 27x27x96
...

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

AlexNet

[Krizhevsky et al., 2012]

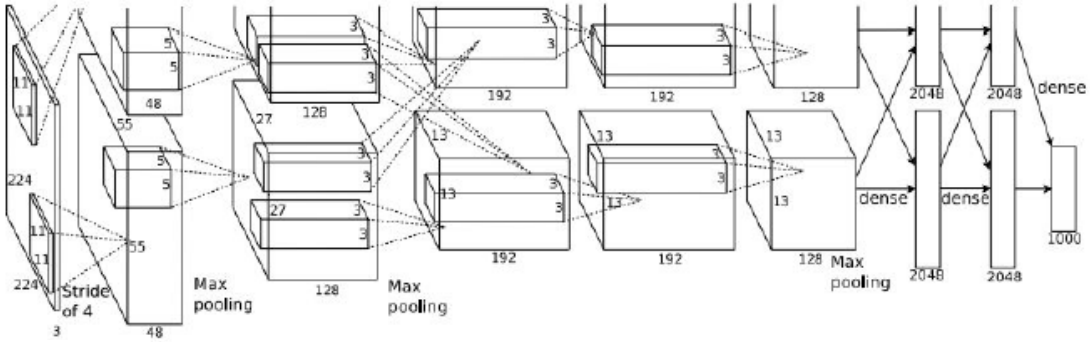


Arquitetura simplificada:

```
[227x227x3]     INPUT
[55x55x96]      CONV1: 96 filtros 11x11, stride 4, pad 0
[27x27x96]      MAX POOL1: filtros 3x3, stride 2
[27x27x96]      NORM1: layer de normalização
[27x27x256]     CONV2: 256 filtros 5x5, stride 1, pad 2
[13x13x256]     MAX POOL2: filtros 3x3, stride 2
[13x13x256]     NORM2: layer de normalização
[13x13x384]     CONV3: 384 filtros 3x3, stride 1, pad 1
[13x13x384]     CONV4: 384 filtros 3x3, stride 1, pad 1
[13x13x256]     CONV5: 256 filtros 3x3, stride 1, pad 1
[6x6x256]       MAX POOL3: filtros 3x3, stride 2
[4096]          FC6: 4096 neurônios
[4096]          FC7: 4096 neurônios
[1000]          FC8: 1000 neurônios (classes)
```

AlexNet

[Krizhevsky et al., 2012]



Arquitetura simplificada:

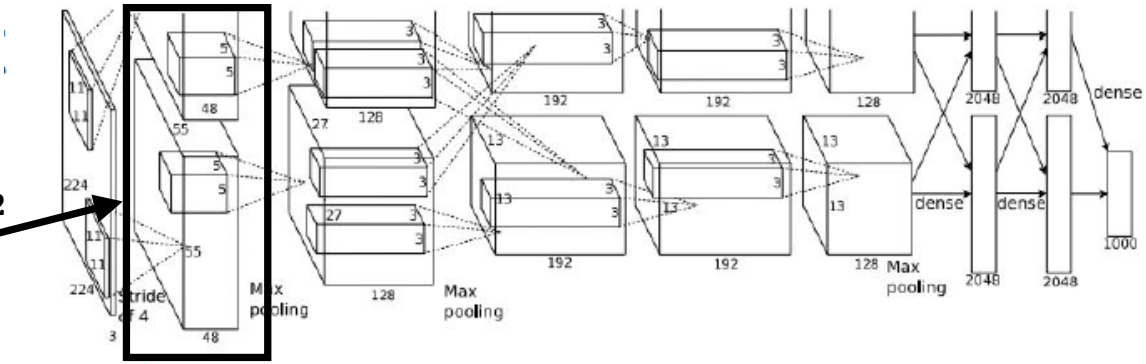
```
[227x227x3]    INPUT
[55x55x96]     CONV1: 96 filtros 11x11, stride 4, pad 0
[27x27x96]     MAX POOL1: filtros 3x3, stride 2
[27x27x96]     NORM1: layer de normalização
[27x27x256]    CONV2: 256 filtros 5x5, stride 1, pad 2
[13x13x256]    MAX POOL2: filtros 3x3, stride 2
[13x13x256]    NORM2: layer de normalização
[13x13x384]    CONV3: 384 filtros 3x3, stride 1, pad 1
[13x13x384]    CONV4: 384 filtros 3x3, stride 1, pad 1
[13x13x256]    CONV5: 256 filtros 3x3, stride 1, pad 1
[6x6x256]      MAX POOL3: filtros 3x3, stride 2
[4096]         FC6: 4096 neurônios
[4096]         FC7: 4096 neurônios
[1000]         FC8: 1000 neurônios (classes)
```

Detalhes:

- primeiro uso de ReLU
- layers de norm: não são mais usados
- data augmentation pesado
- dropout 0.5
- batch 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduzido por 10 sempre que a acurácia de validação ficava em um plateau
- L2 weight decay 5e-4
- Ensemble de 7 CNNs: 18.2% -> 15.4%

AlexNet

[Krizhevsky et al., 2012]



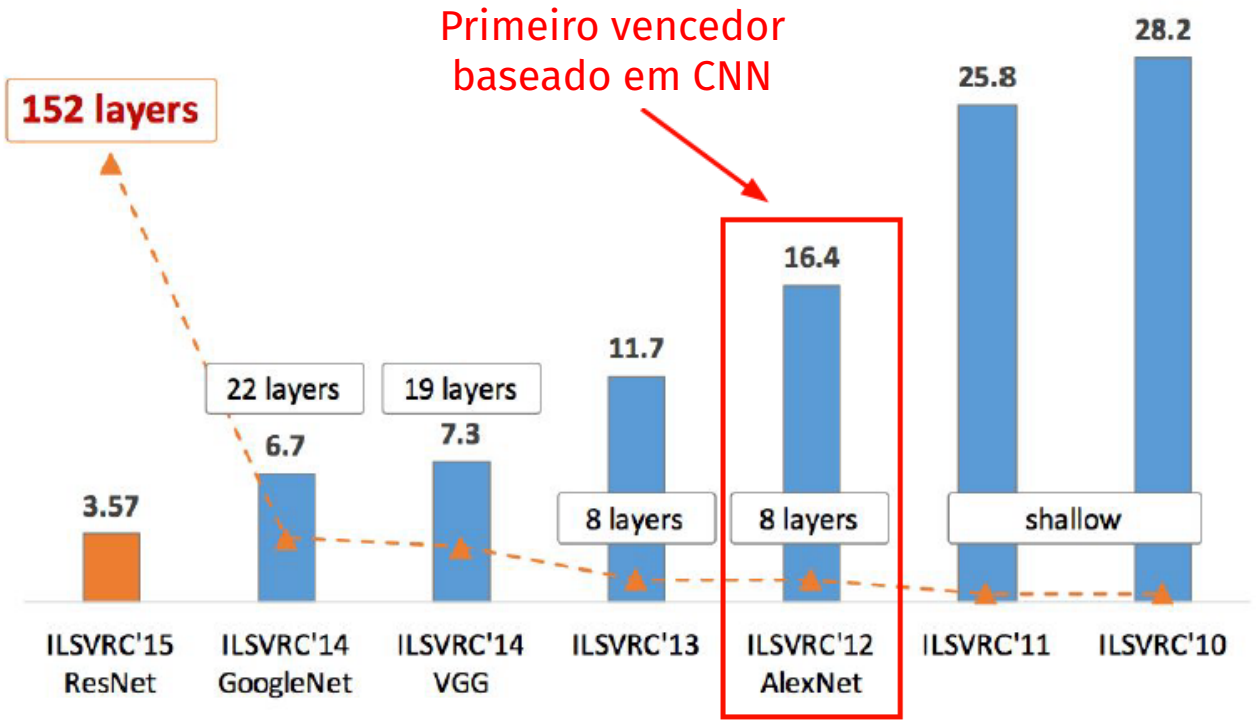
Arquitetura simplificada:

```
[227x227x3]    INPUT
[55x55x96]     CONV1: 96 filtros 11x11, stride 4, pad 0
[27x27x96]     MAX POOL1: filtros 3x3, stride 2
[27x27x96]     NORM1: layer de normalização
[27x27x256]    CONV2: 256 filtros 5x5, stride 1, pad 2
[13x13x256]    MAX POOL2: filtros 3x3, stride 2
[13x13x256]    NORM2: layer de normalização
[13x13x384]    CONV3: 384 filtros 3x3, stride 1, pad 1
[13x13x384]    CONV4: 384 filtros 3x3, stride 1, pad 1
[13x13x256]    CONV5: 256 filtros 3x3, stride 1, pad 1
[6x6x256]      MAX POOL3: filtros 3x3, stride 2
[4096]         FC6: 4096 neurônios
[4096]         FC7: 4096 neurônios
[1000]         FC8: 1000 neurônios (classes)
```

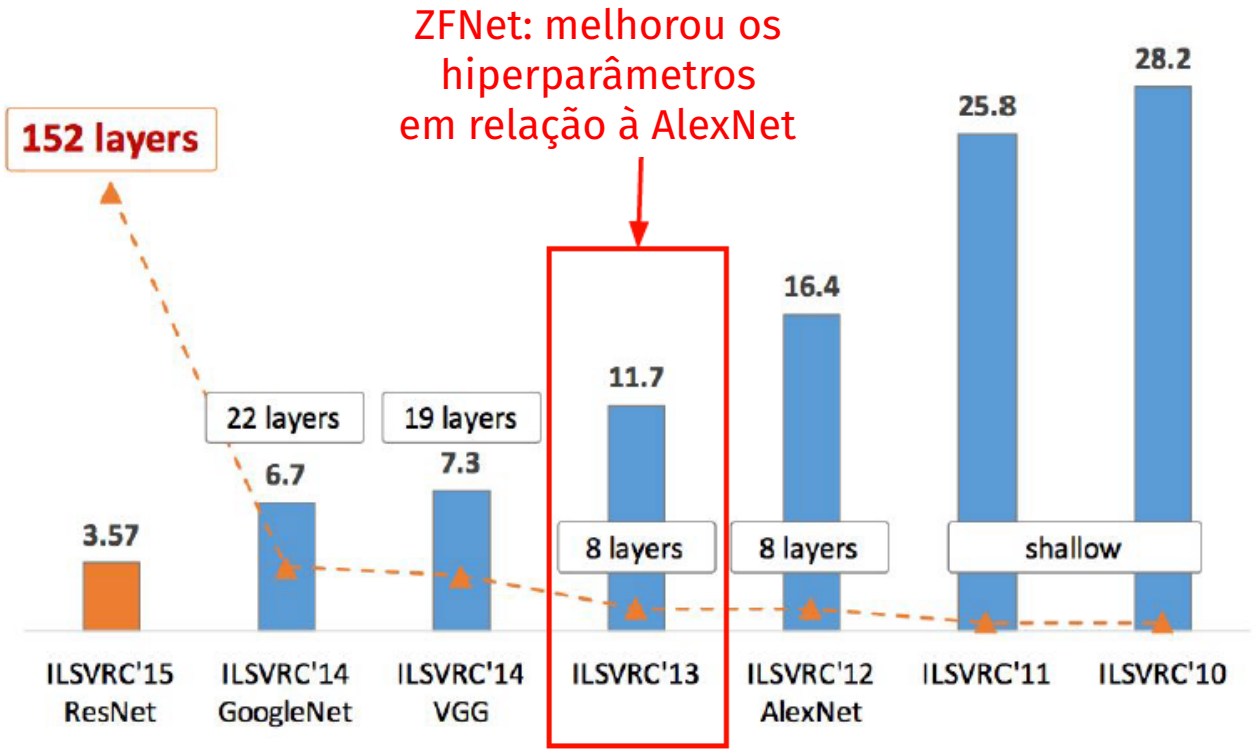
- Treinada em 2 GPUs com apenas 3GB
- Metade dos neurônios (mapas de ativação) em cada

Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vencedores do ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

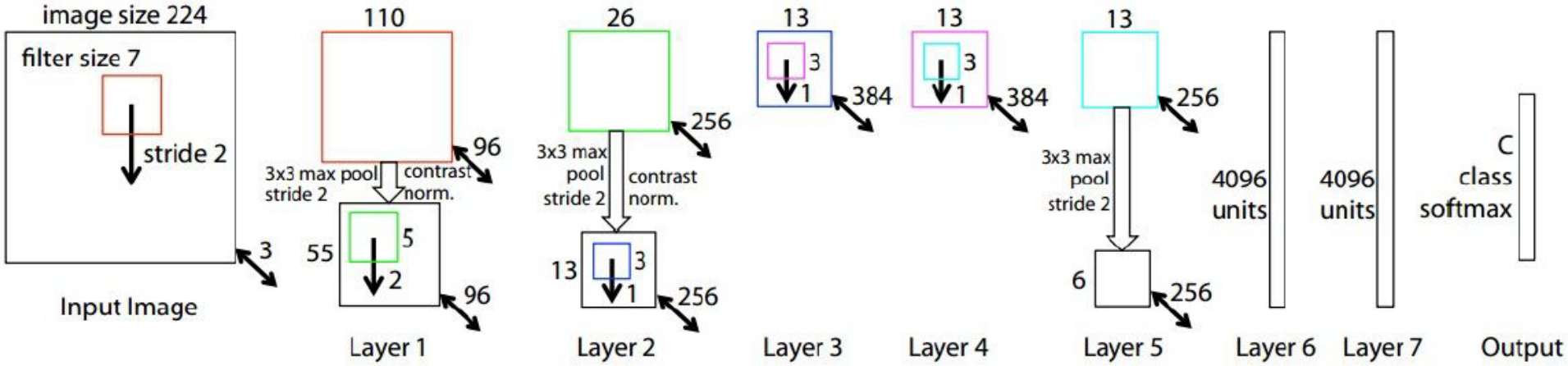


Vencedores do ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



ZeNet

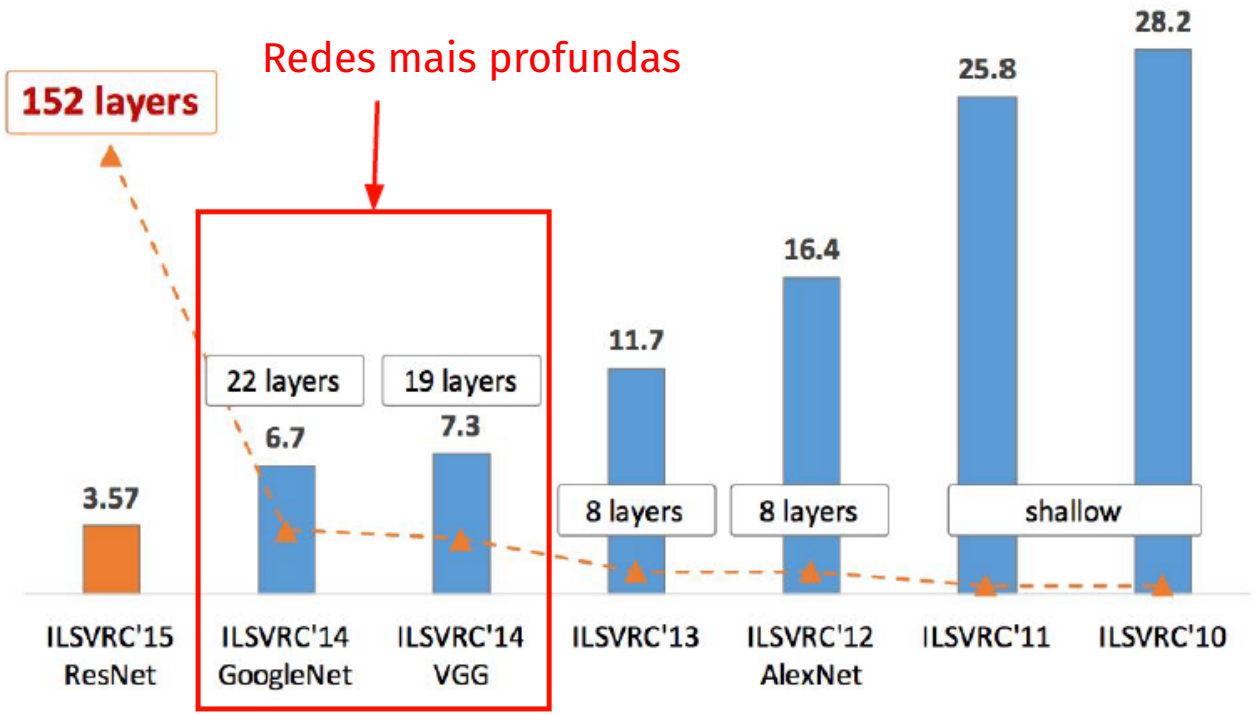
[Zeiler e Fergus, 2013]



AlexNet, porém:
CONV1: muda de (11x11, stride 4) para (7x7 stride 2)
CONV3,4,5: ao invés de 384, 384 e 356 filtros, muda para 512, 1024, 512

ImageNet top error: 16.4% -> 11.7%

Vencedores do ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



VGGNet

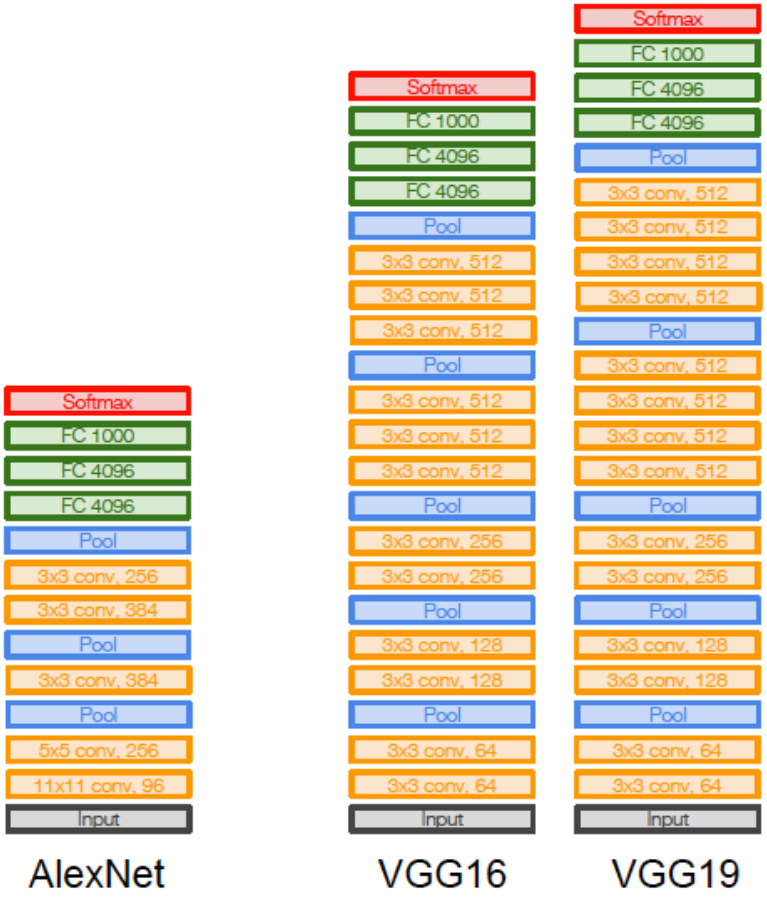
[Simonyan and Zisserman, 2014]

Filtros menores, rede mais profunda

AlexNet 8 layers -> VGGNet 16 a 19 layers

Apenas 3x3 CONV, stride 1, pad 1
e 2x2 MAX POOL, stride 2

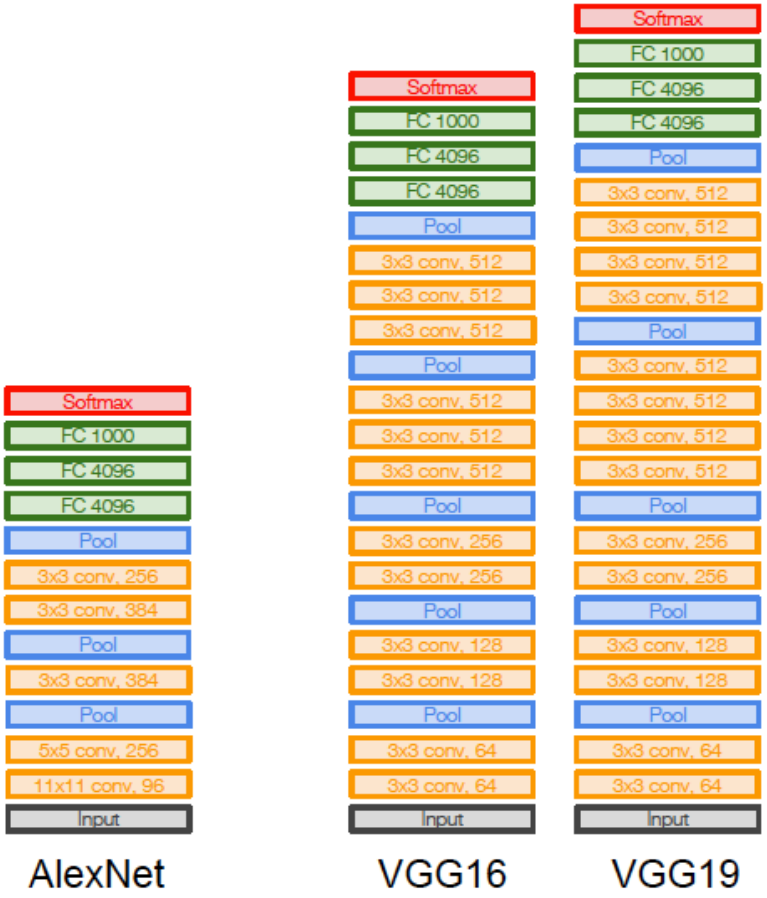
ZFNet 11.7% top error (2013) ->
VGGNet 7.3% top error (2014)



VGGNet

[Simonyan and Zisserman, 2014]

=> Por que usar filtros menores e rede mais profunda?



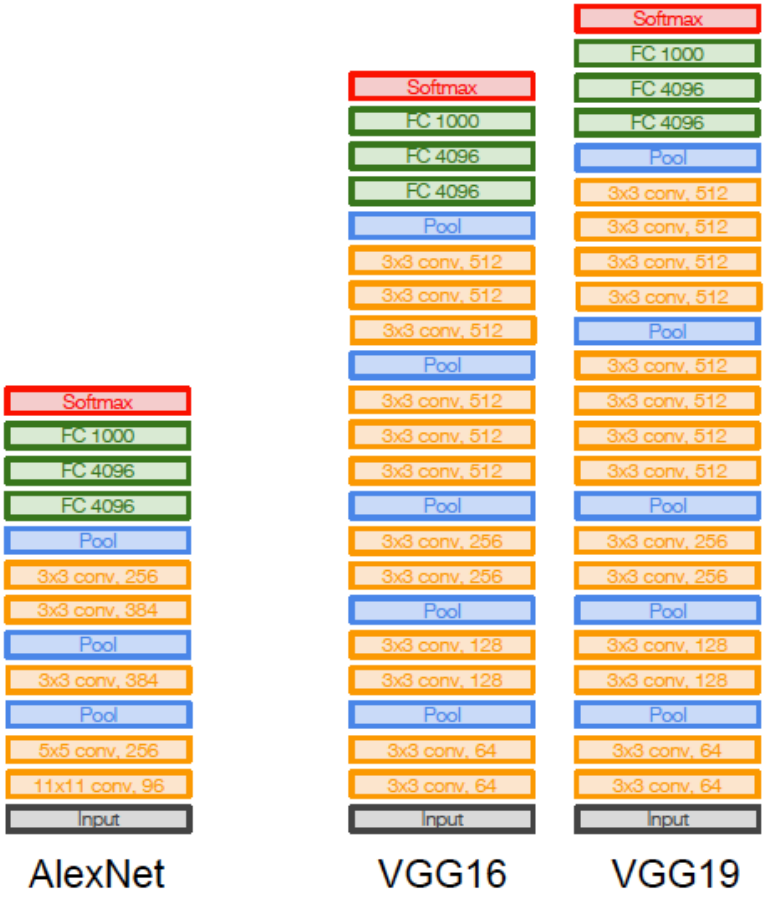
VGGNet

[Simonyan and Zisserman, 2014]

=> Por que usar filtros menores e rede mais profunda?

Mais profunda: mais não-linearidade

Filtros menores: menos parâmetros

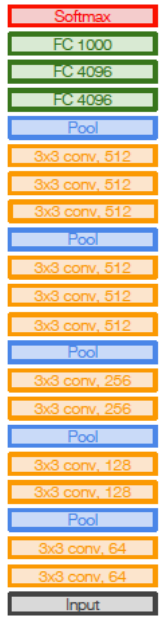


VGGNet

[Simonyan and Zisserman, 2014]

não contando o bias

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

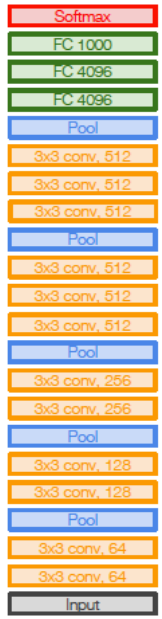
VGGNet

[Simonyan and Zisserman, 2014]

não contando o bias

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Memória total: 24M * 4 bytes ~= 96MB / image
Parâmetros totais: 138M



VGG16

VGGNet

[Simonyan and Zisserman, 2014]

não contando o bias

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Maioria da memória está nas CONV anteriores

Maioria dos parâmetros está nas camadas fully-connected

Memória total: 24M * 4 bytes ~= 96MB / image
Parâmetros totais: 138M

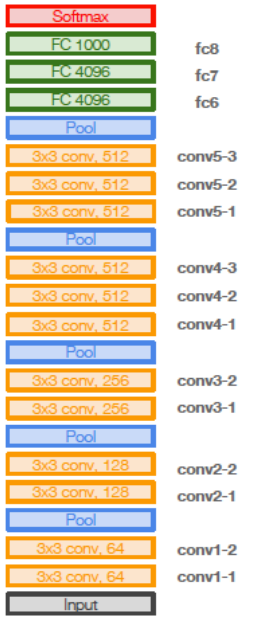
VGGNet

[Simonyan and Zisserman, 2014]

não contando o bias

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Memória total: 24M * 4 bytes ~= 96MB / image
Parâmetros totais: 138M



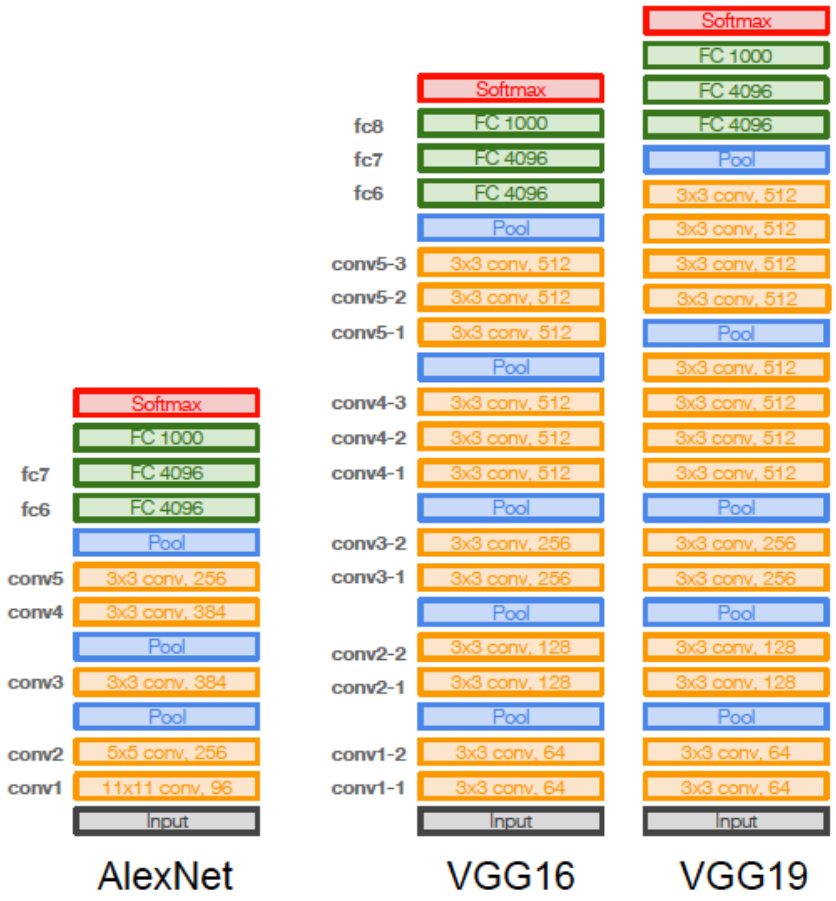
VGG16

nomes comuns

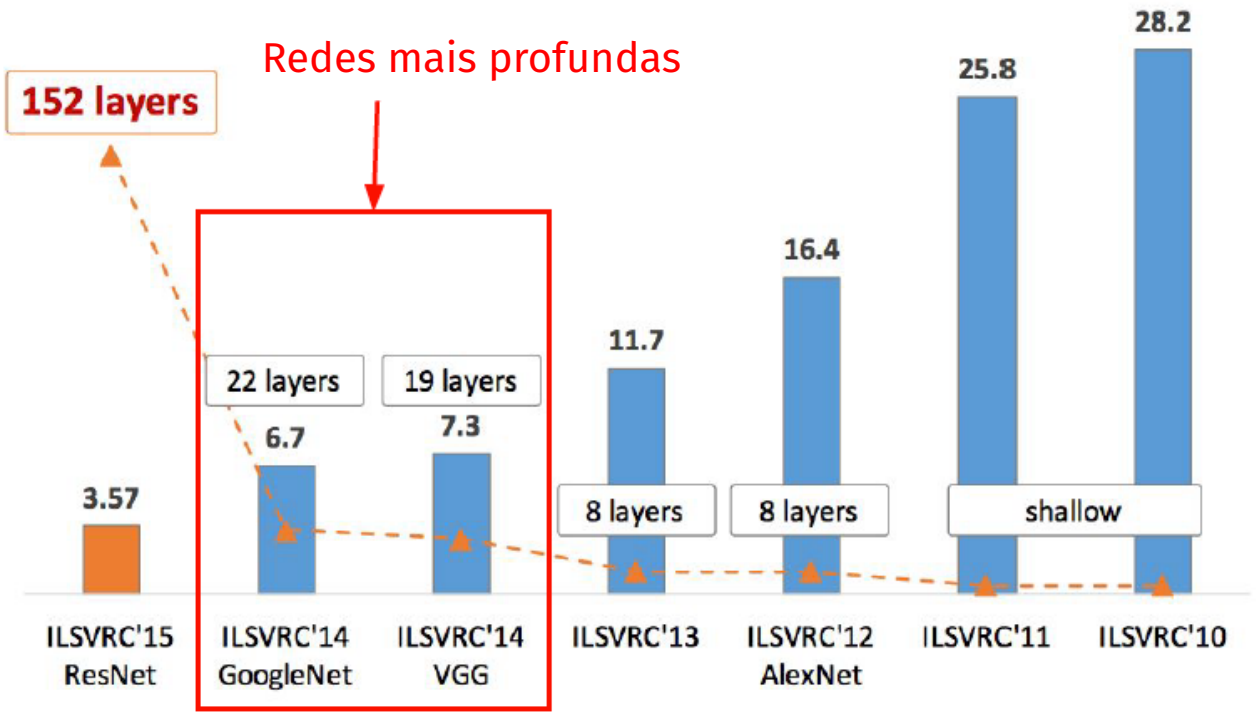
VGGNet

[Simonyan and Zisserman, 2014]

- ILSVRC ’14: 2º lugar em classificação, 1º lugar em localização
- Treinamento similar à AlexNet
- Features geradas pela FC7 generalizam bem para outras tarefas



Vencedores do ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



GoogLeNet

[Szegedy et al., 2014]

Rede mais profunda, com eficiência computacional

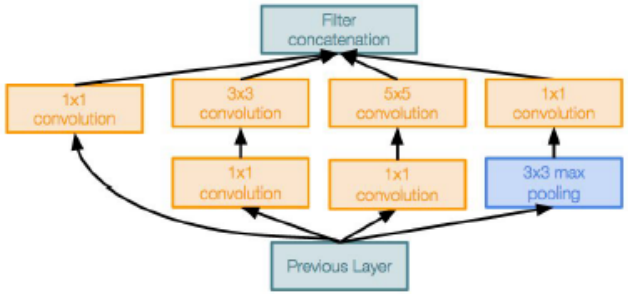
22 layers

Módulo Inception

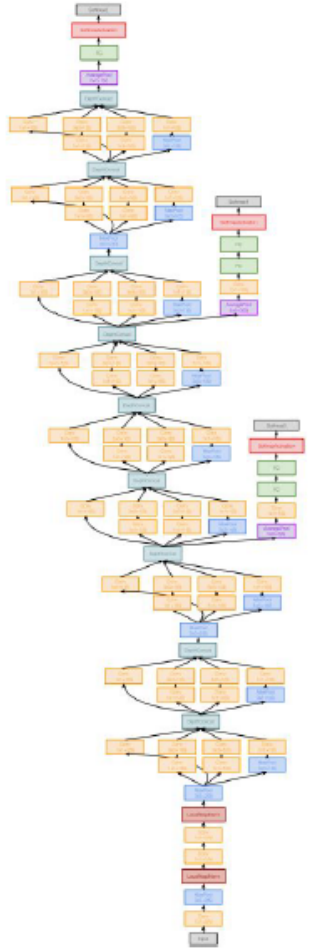
Sem layers FC

Apenas 5 milhões de parâmetros (12x menos que a AlexNet)

Campeão na tarefa de classificação na ILSVRC ’14: 6.7% top error



Inception module

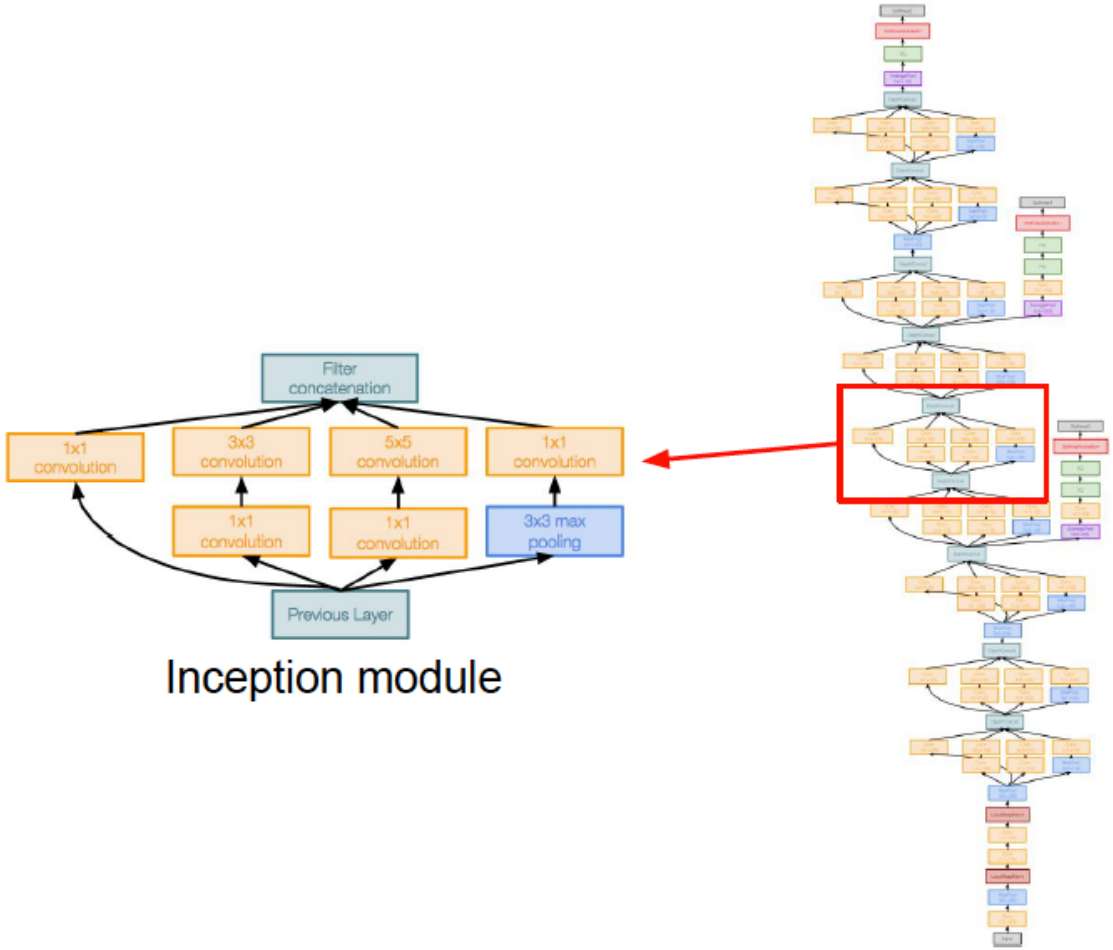


GoogLeNet

[Szegedy et al., 2014]

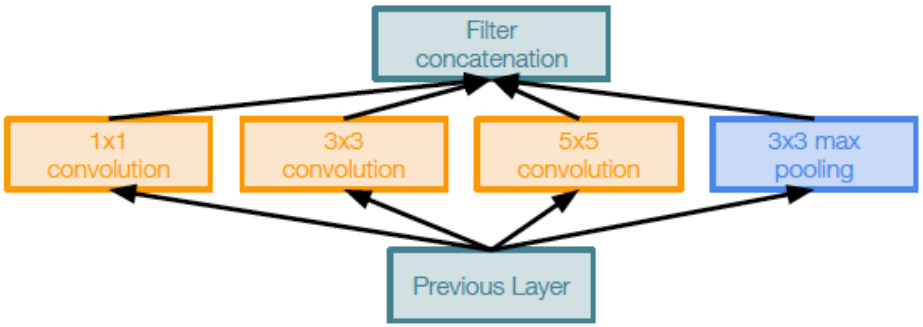
Módulo **inception**:

criar uma topologia de rede local (rede dentro de uma rede) e empilhar os módulos.



GoogLeNet

[Szegedy et al., 2014]



Módulo inception “ingênuo"

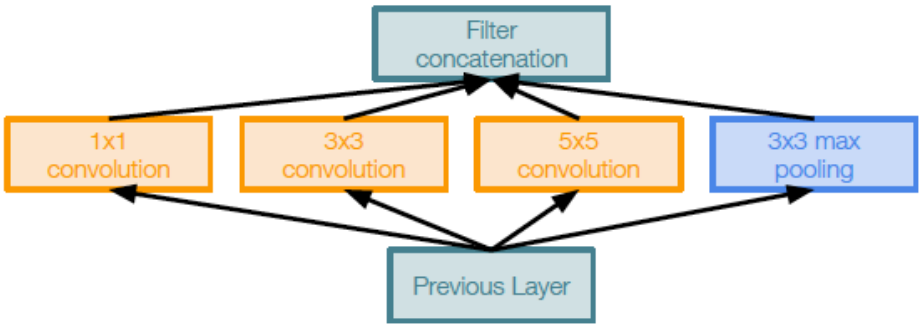
Aplicar filtros paralelos ao input da camada anterior:

- Múltiplos campos receptivos para convolução
  - 1x1
  - 3x3
  - 5x5
- Operações de pooling 3x3

Concatenar todos os outputs dos filtros (nos canais)

GoogLeNet

[Szegedy et al., 2014]



Módulo inception “ingênuo"

Aplicar filtros paralelos ao input da camada anterior:

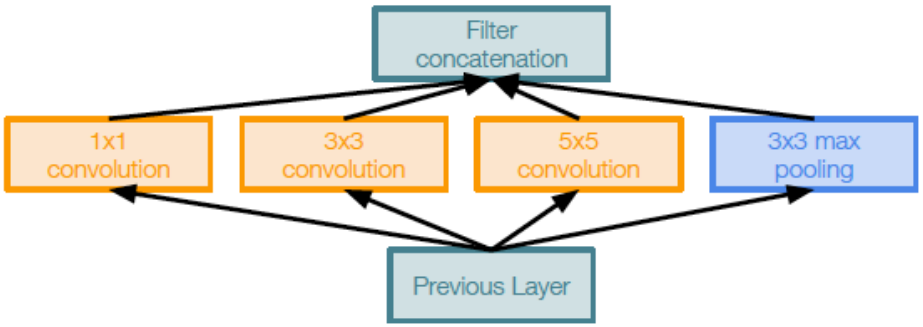
- Múltiplos campos receptivos para convolução
  - 1x1
  - 3x3
  - 5x5
- Operações de pooling 3x3

Concatenar todos os outputs dos filtros (nos canais)

=> Qual o problema com essa abordagem?

GoogLeNet

[Szegedy et al., 2014]



Módulo inception “ingênuo"

Aplicar filtros paralelos ao input da camada anterior:

- Múltiplos campos receptivos para convolução
  - 1x1
  - 3x3
  - 5x5
- Operações de pooling 3x3

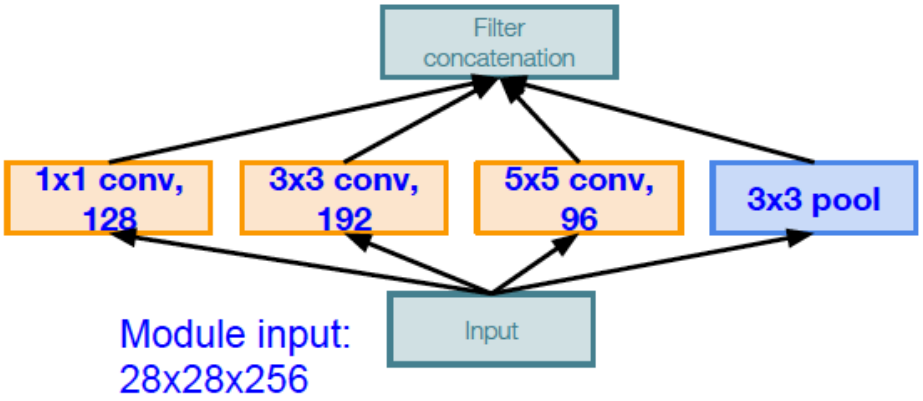
Concatenar todos os outputs dos filtros (nos canais)

=> Complexidade computacional!

GoogLeNet

[Szegedy et al., 2014]

Exemplo:



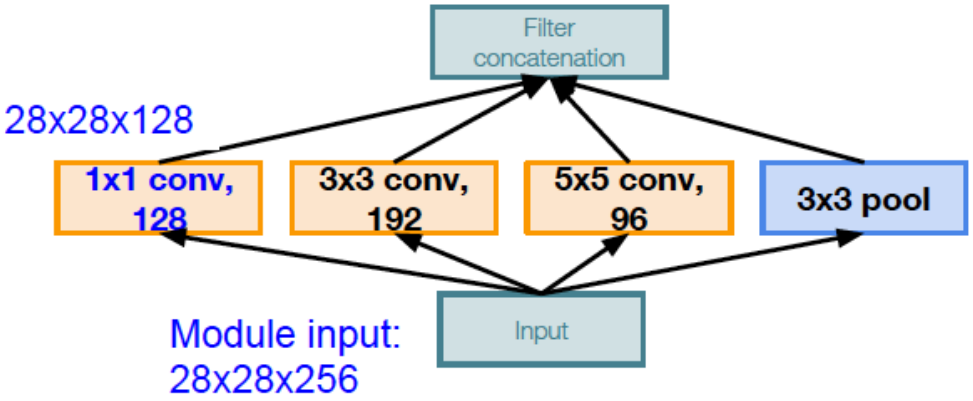
Módulo inception “ingênuo"

=> Qual o tamanho do output da CONV 1x1, com 128 filtros?

GoogLeNet

[Szegedy et al., 2014]

Exemplo:



Módulo inception “ingênuo"

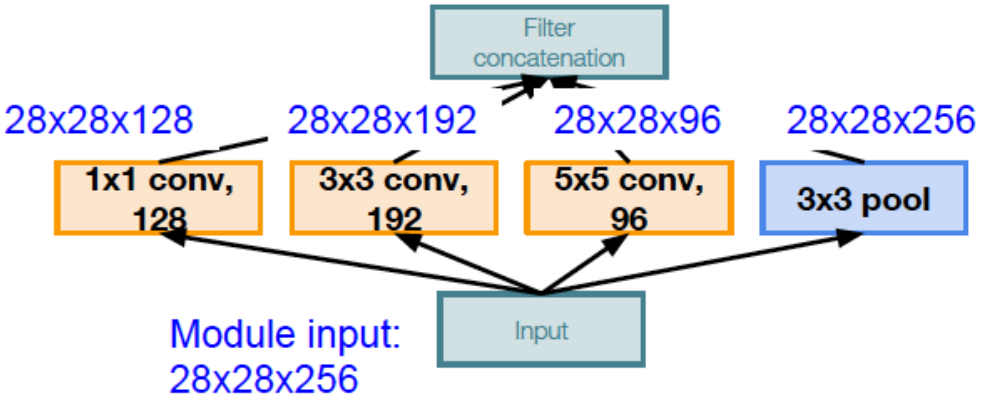
=> Qual o tamanho do output da CONV 1x1, com 128 filtros?

=> Qual o tamanho do output das outras operações?

GoogLeNet

[Szegedy et al., 2014]

Exemplo:



Módulo inception “ingênuo"

=> Qual o tamanho do output da CONV 1x1, com 128 filtros?

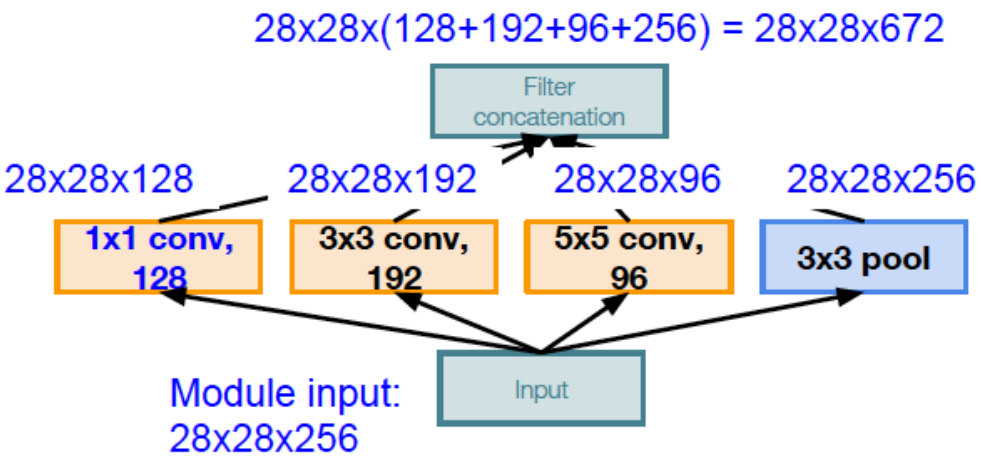
=> Qual o tamanho do output das outras operações?

=> Qual o tamanho do output depois da concatenação?

GoogLeNet

[Szegedy et al., 2014]

Exemplo:



Módulo inception “ingênuo"

=> Qual o tamanho do output da CONV 1x1, com 128 filtros?

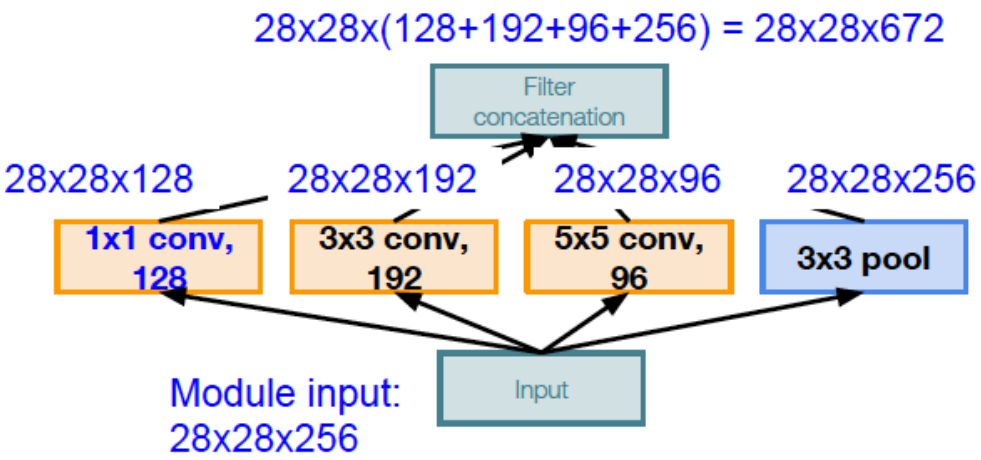
=> Qual o tamanho do output das outras operações?

=> Qual o tamanho do output depois da concatenação?

GoogLeNet

[Szegedy et al., 2014]

Exemplo:



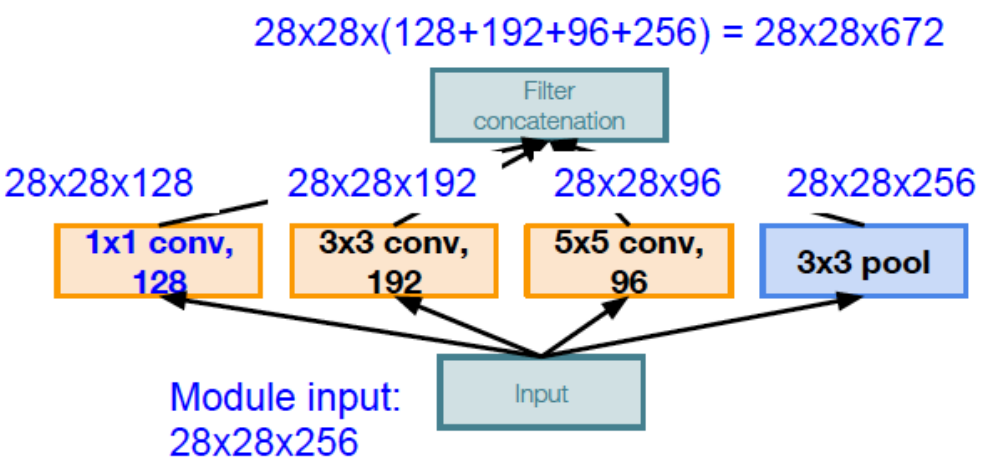
Módulo inception “ingênuo"

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

GoogLeNet

[Szegedy et al., 2014]

Exemplo:



Módulo inception “ingênuo"

Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops

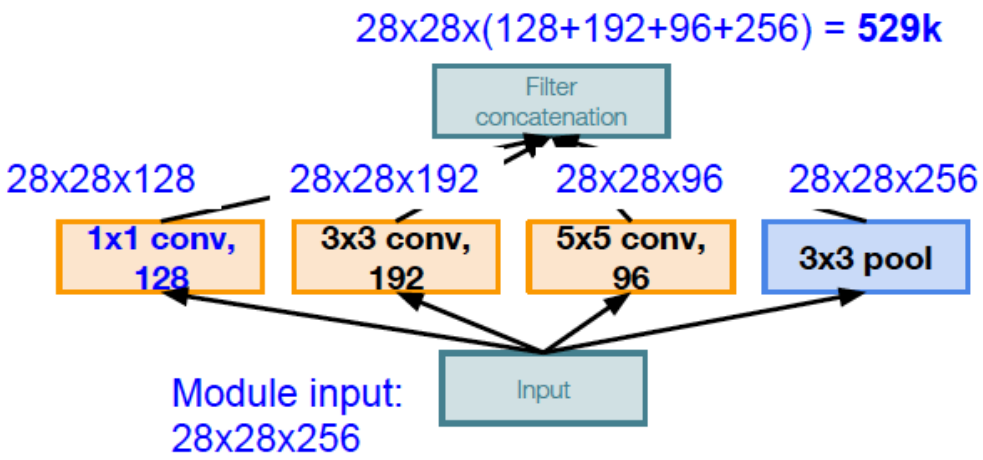
Muito caro computacionalmente!

O pooling preserva a profundidade e, assim, a profundidade após a concatenação apenas cresce a cada layer!

GoogLeNet

[Szegedy et al., 2014]

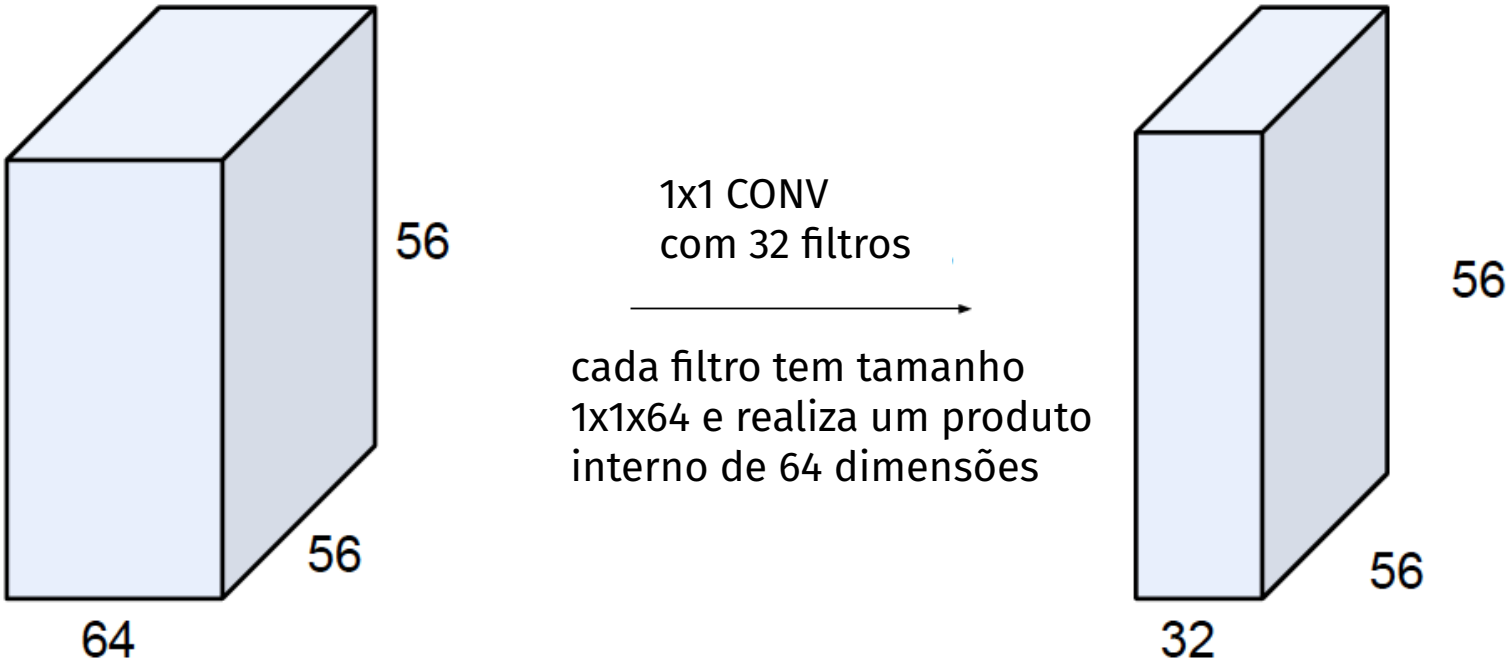
Exemplo:



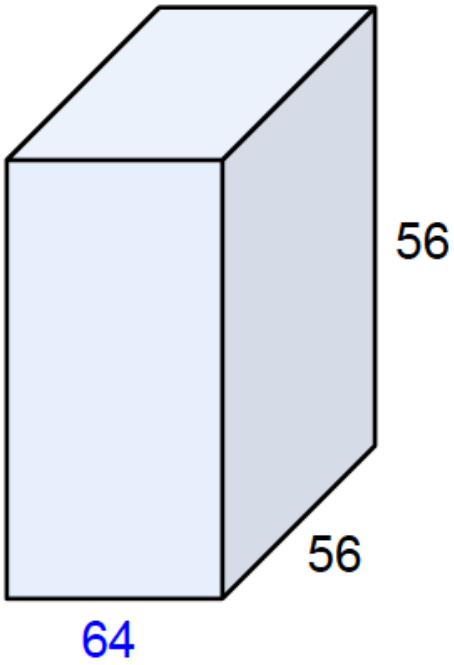
Módulo inception “ingênuo"

Solução: camadas de “gargalo"
que usam convoluções 1x1
para reduzir a profundidade
das features

Lembrete: convoluções 1x1



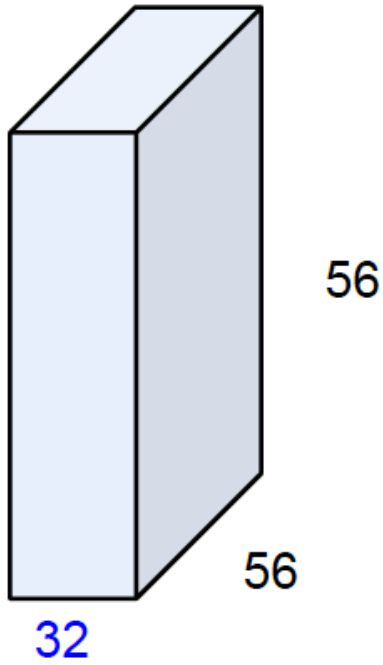
Lembrete: convoluções 1x1



1x1 CONV
com 32 filtros

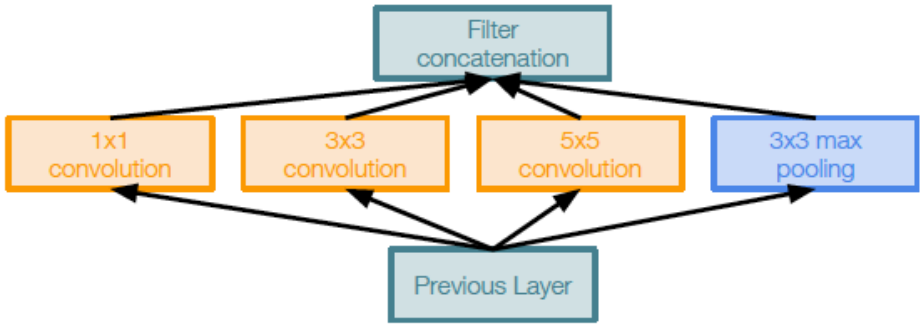
cada filtro tem tamanho
1x1x64 e realiza um produto
interno de 64 dimensões

projeta a profundidade
para dimensões menores
(combinação de mapas
de ativação)

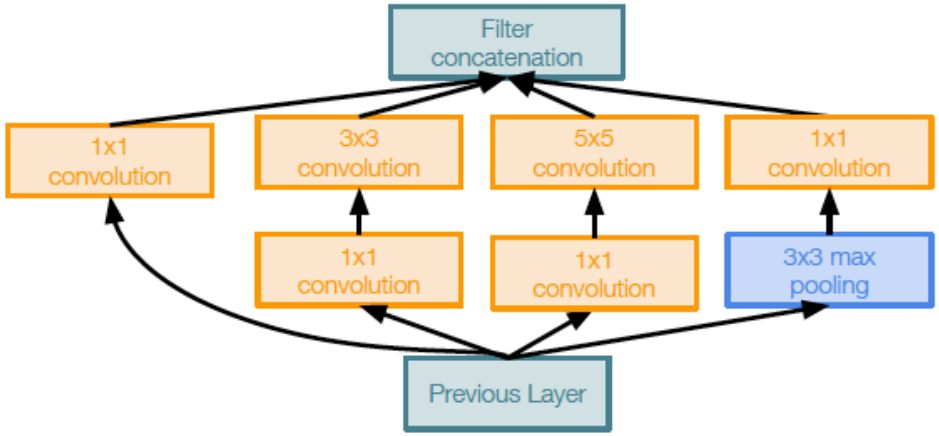


GoogLeNet

[Szegedy et al., 2014]



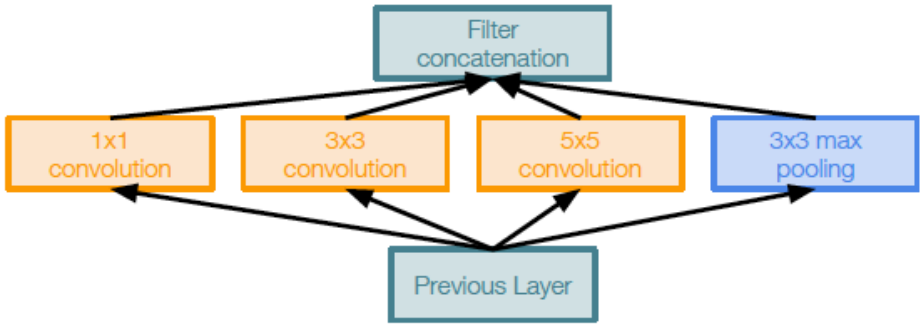
Módulo inception “ingênuo"



Módulo inception com
redução de dimensionalidade

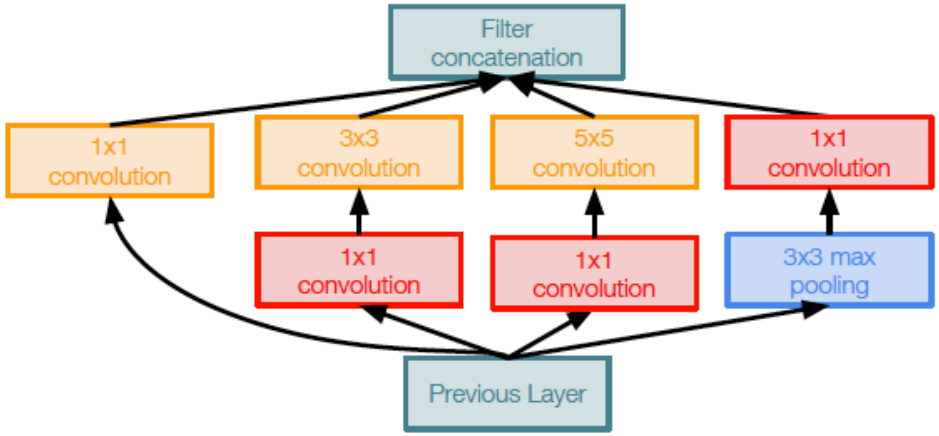
GoogLeNet

[Szegedy et al., 2014]



Módulo inception “ingênuo"

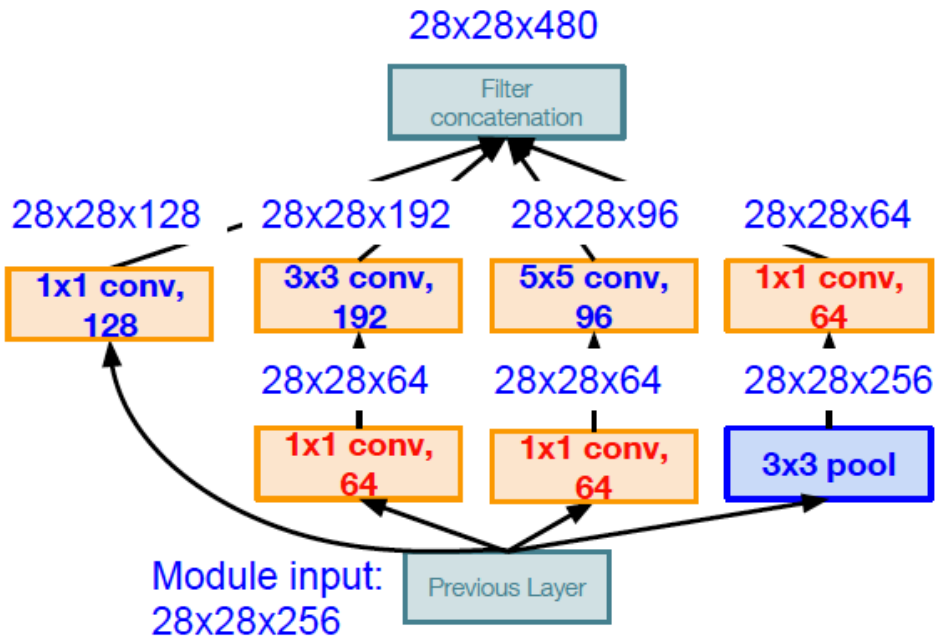
1x1 conv “bottleneck” layers



Módulo inception com redução de dimensionalidade

GoogLeNet

[Szegedy et al., 2014]



Módulo inception com redução de dimensionalidade

Utilizar os mesmos layers paralelos do exemplo “ingênuo” e adicionar conv 1x1, 64 filtros “gargalo”:

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops

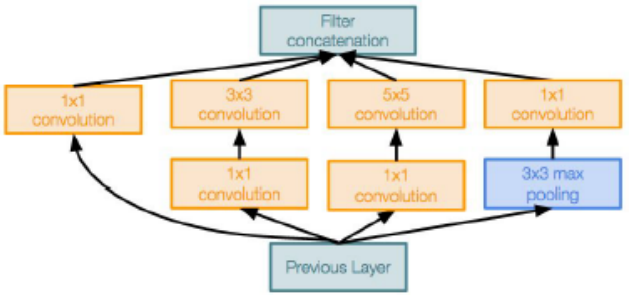
Comparado aos 854M de operações da versão “ingênua”

O “gargalo" também diminui os canais depois do pooling

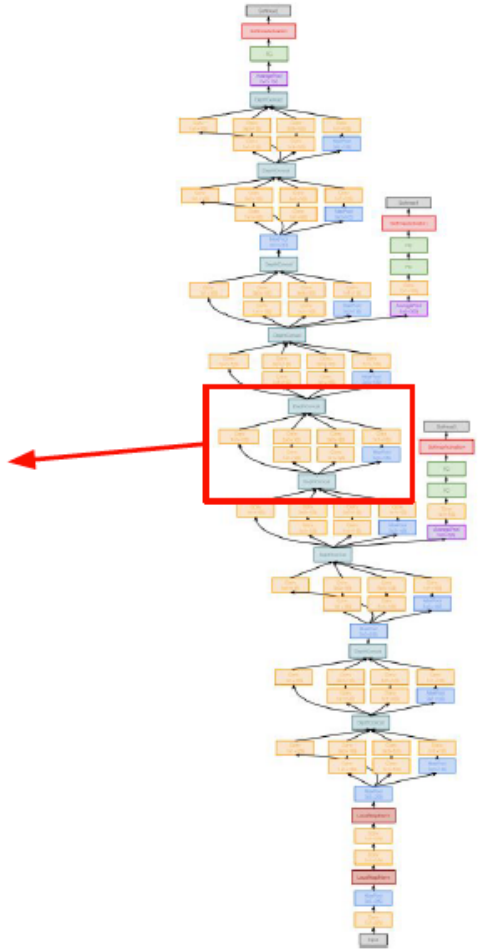
GoogLeNet

[Szegedy et al., 2014]

Empilhar módulos inception
com redução de dimensionalidade



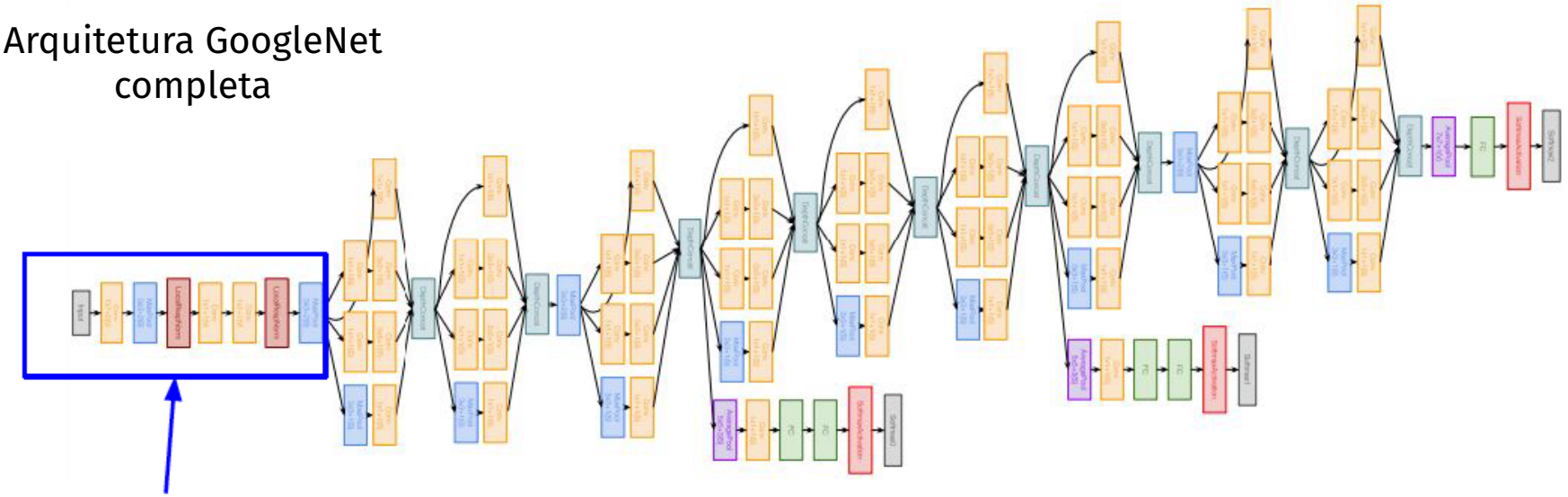
Inception module



GoogLeNet

[Szegedy et al., 2014]

Arquitetura GoogleNet completa

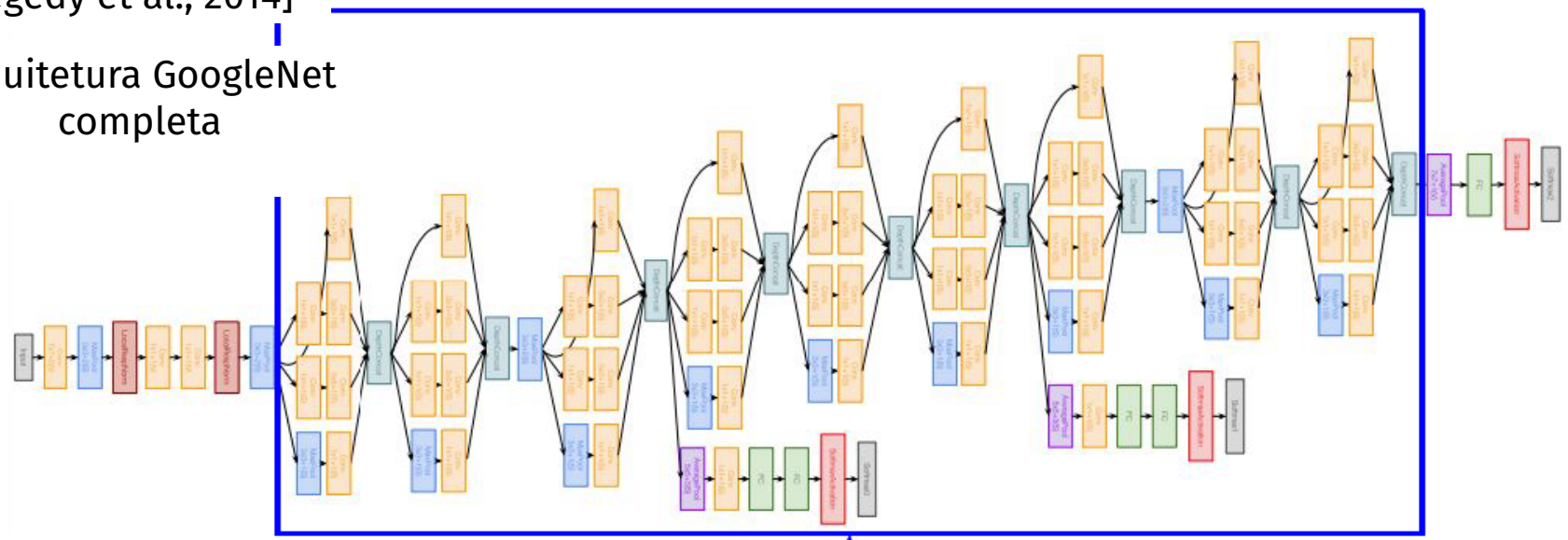


Rede inicial:
CONV-POOL-2x CONV-POOL

GoogLeNet

[Szegedy et al., 2014]

Arquitetura GoogleNet completa

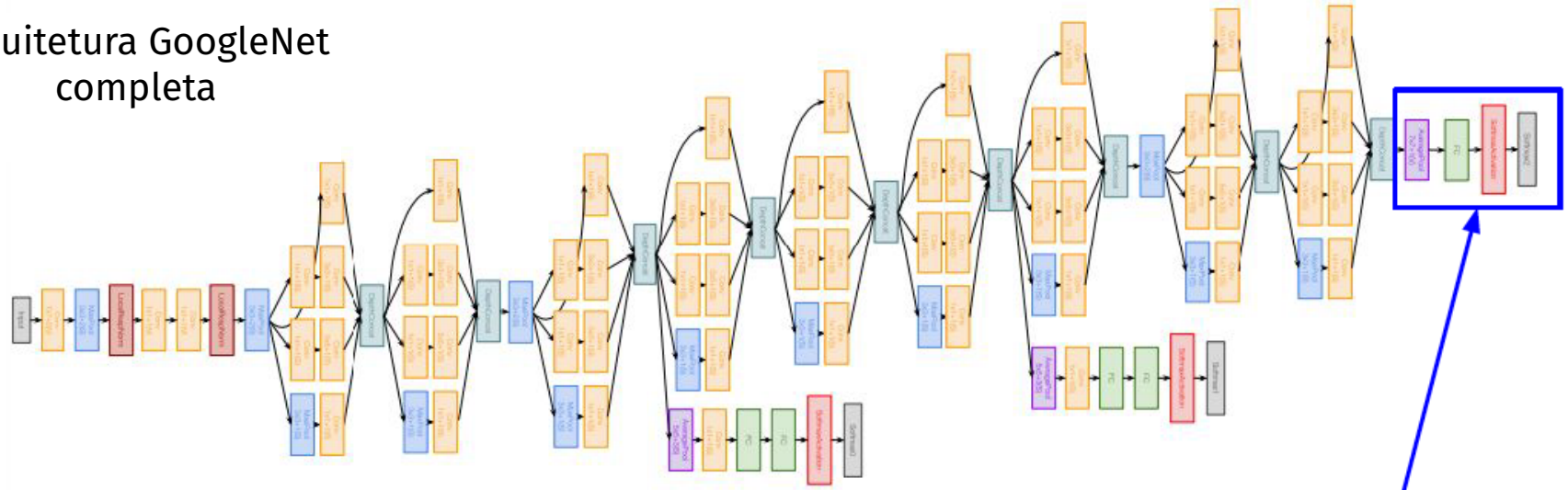


Módulos inception empilhados

GoogLeNet

[Szegedy et al., 2014]

Arquitetura GoogleNet completa

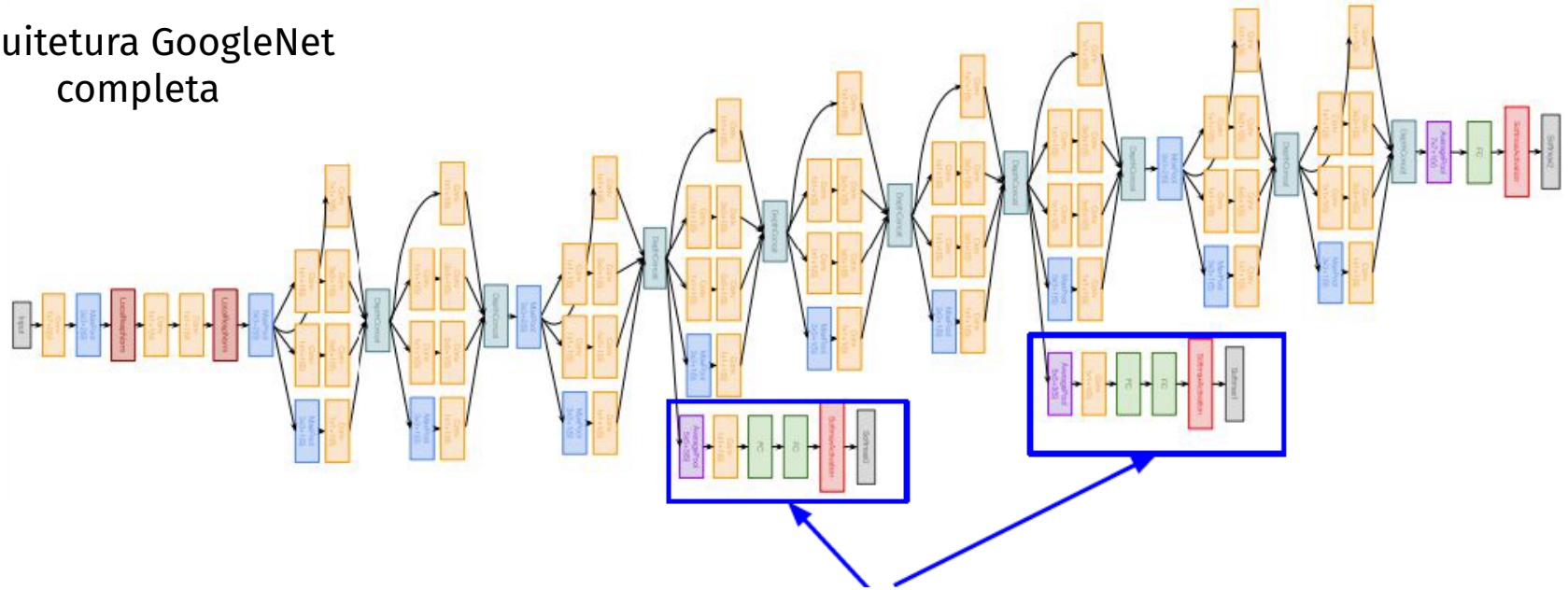


Layers de classificação (remoção dos FCs caros)

GoogLeNet

[Szegedy et al., 2014]

Arquitetura GoogleNet completa

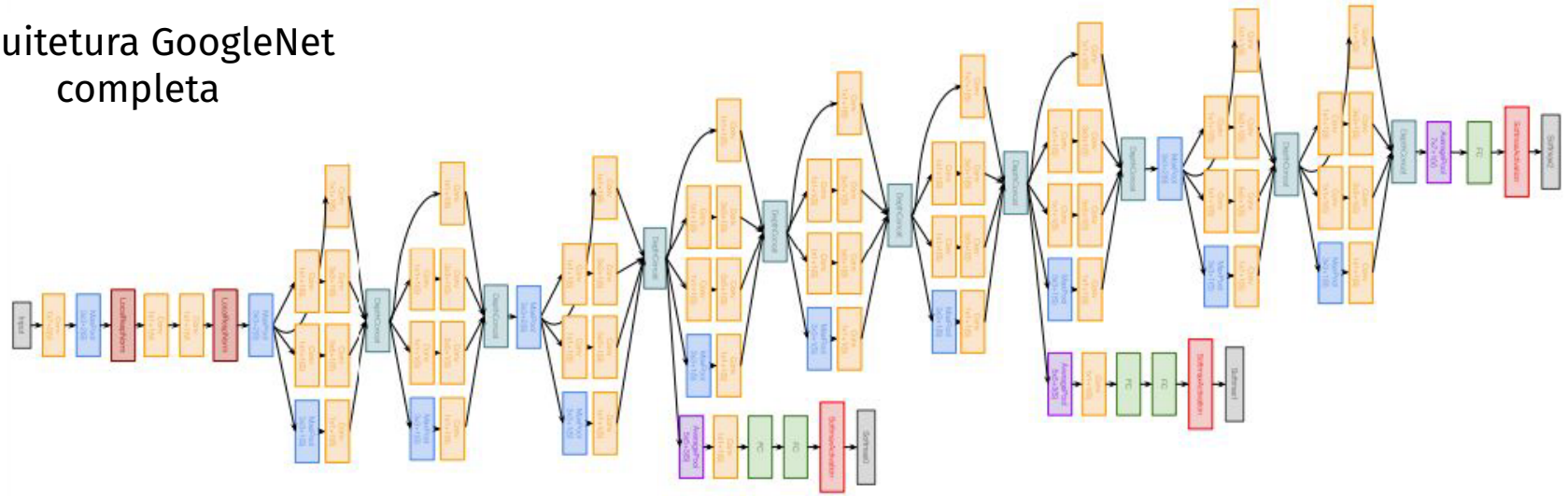


Camadas auxiliares de classificação para injetar gradientes adicionais em camadas mais cedo na rede (AvgPool-1x1 CONV-FC-FC-Softmax)

GoogLeNet

[Szegedy et al., 2014]

Arquitetura GoogleNet completa



Total de 22 layers com pesos (incluindo cada layer paralelo no módulo inception)

GoogLeNet

[Szegedy et al., 2014]

Rede mais profunda, com eficiência computacional

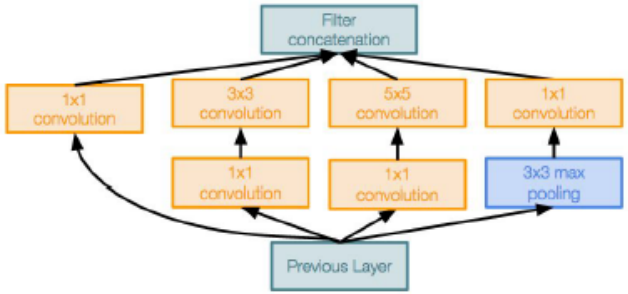
22 layers

Módulo Inception

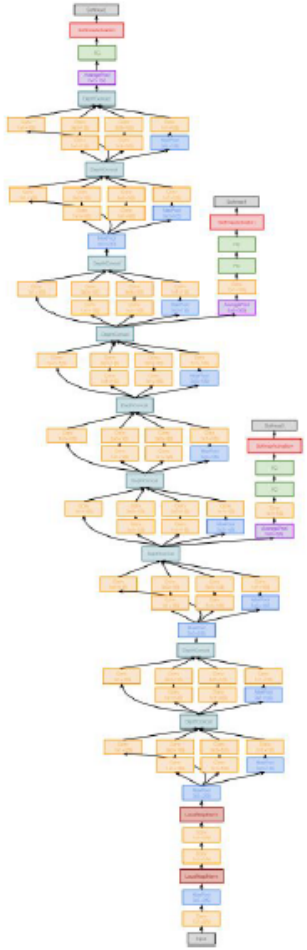
Sem layers FC

Apenas 5 milhões de parâmetros (12x menos que a AlexNet)

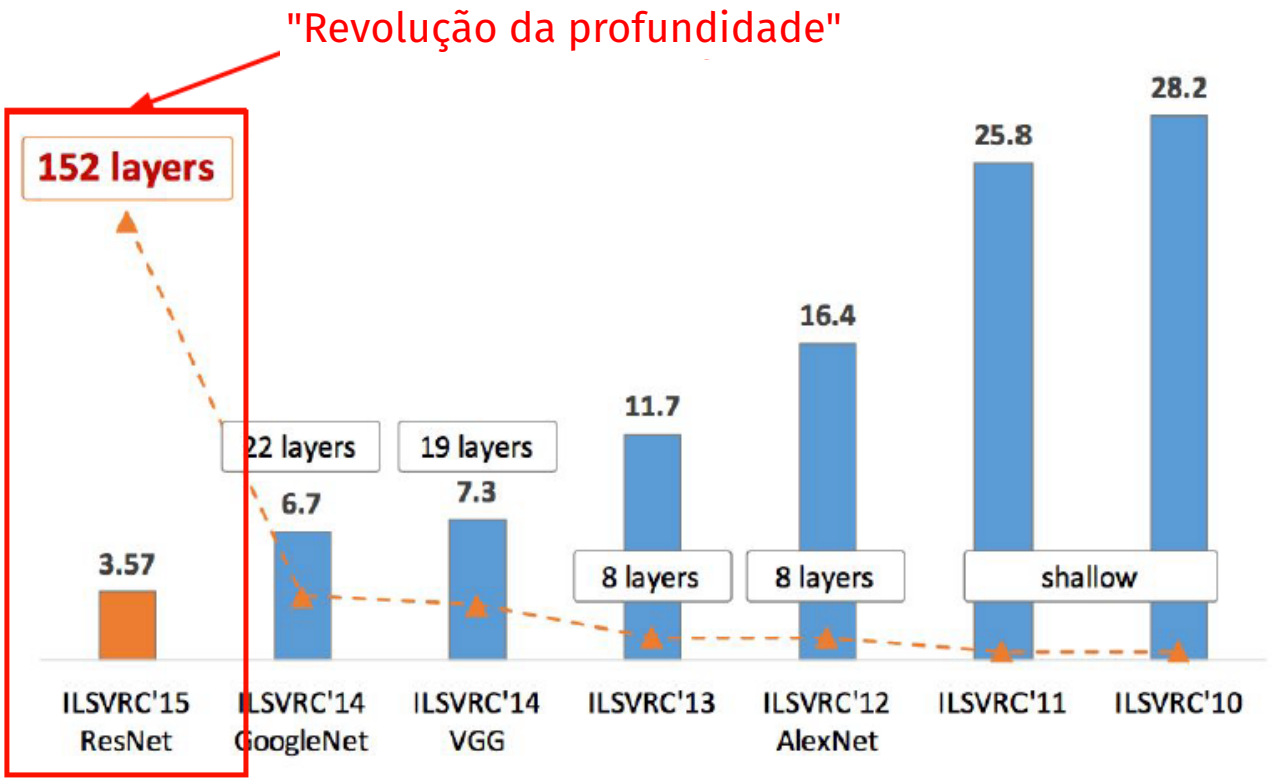
Campeão na tarefa de classificação na ILSVRC ’14: 6.7% top error



Inception module



Vencedores do ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



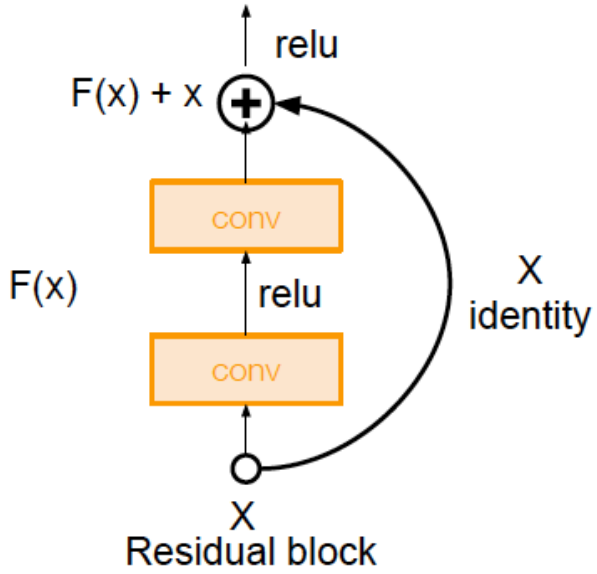
ResNet

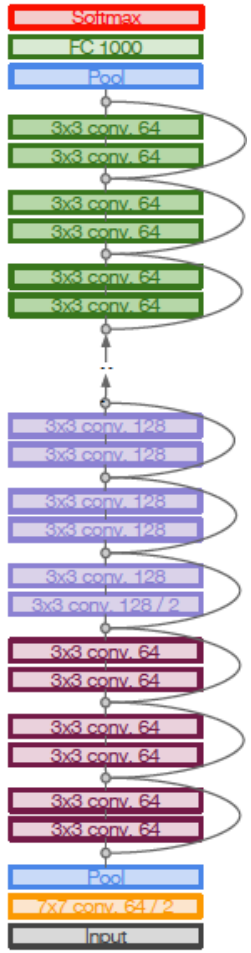
[He et al., 2015]

Rede muito profunda usando conexões residuais

152 layers

Campeão em todas as terefas na ILSVRC ’15: 3.57% top 5 error na classificação





ResNet

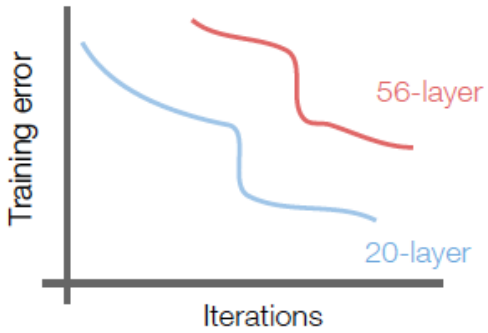
[He et al., 2015]

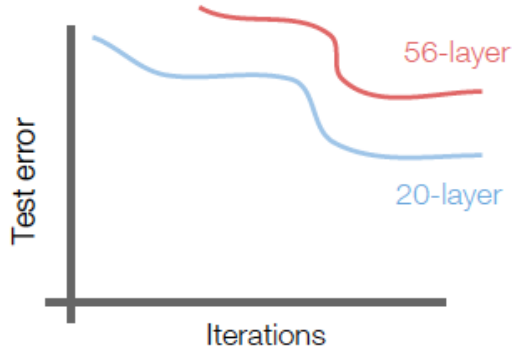
=> O que ocorre se continuarmos empilhando camadas em uma rede neural “comum"?

ResNet

[He et al., 2015]

=> O que ocorre se continuarmos empilhando camadas em uma rede neural “comum"?



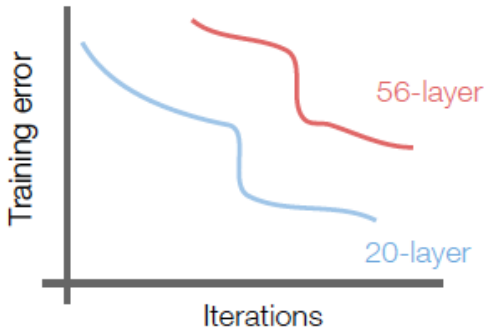


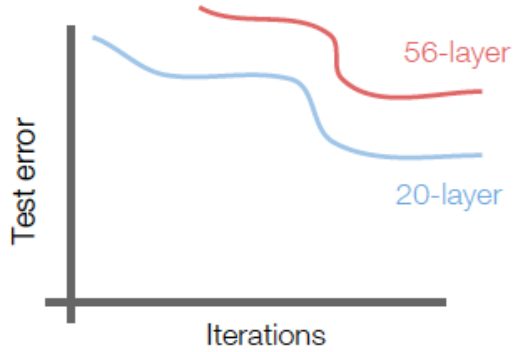
=> O que está estranho nestas curvas de treinamento e teste?

ResNet

[He et al., 2015]

=> O que ocorre se continuarmos empilhando camadas em uma rede neural “comum"?





=> O que está estranho nestas curvas de treinamento e teste?
O modelo de 56 camadas é pior em ambos os casos! Não é overfitting!

ResNet

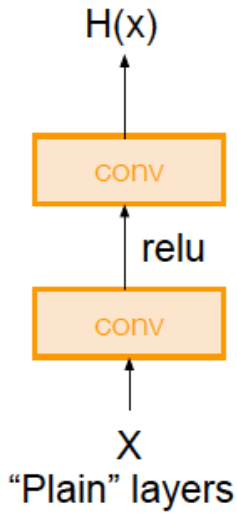
[He et al., 2015]

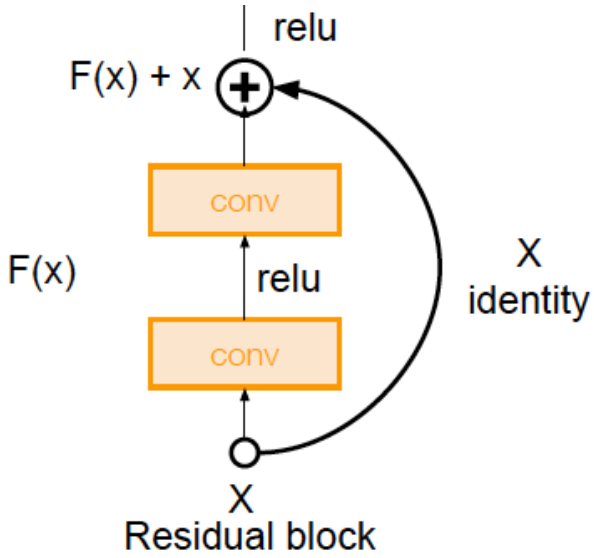
Hipótese: modelos mais profundos são mais difíceis de otimizar.

ResNet

[He et al., 2015]

Solução: Use os layers da rede para calcular um mapa residual, ao invés de calcular o mapa adjacente desejável

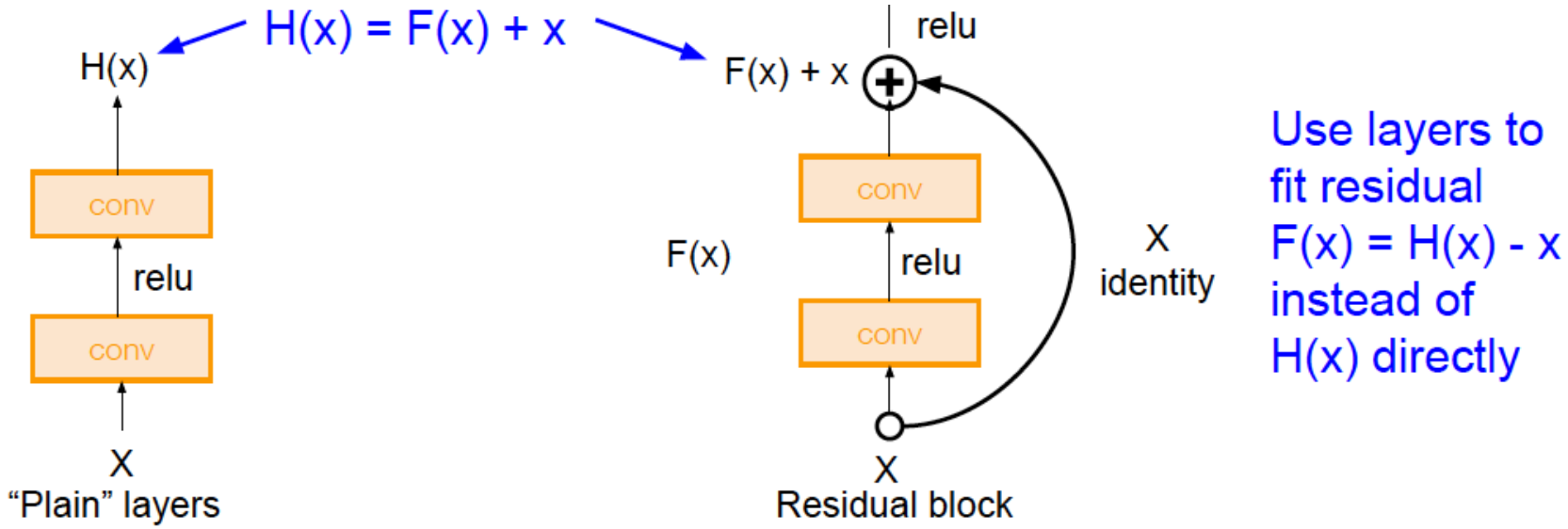




ResNet

[He et al., 2015]

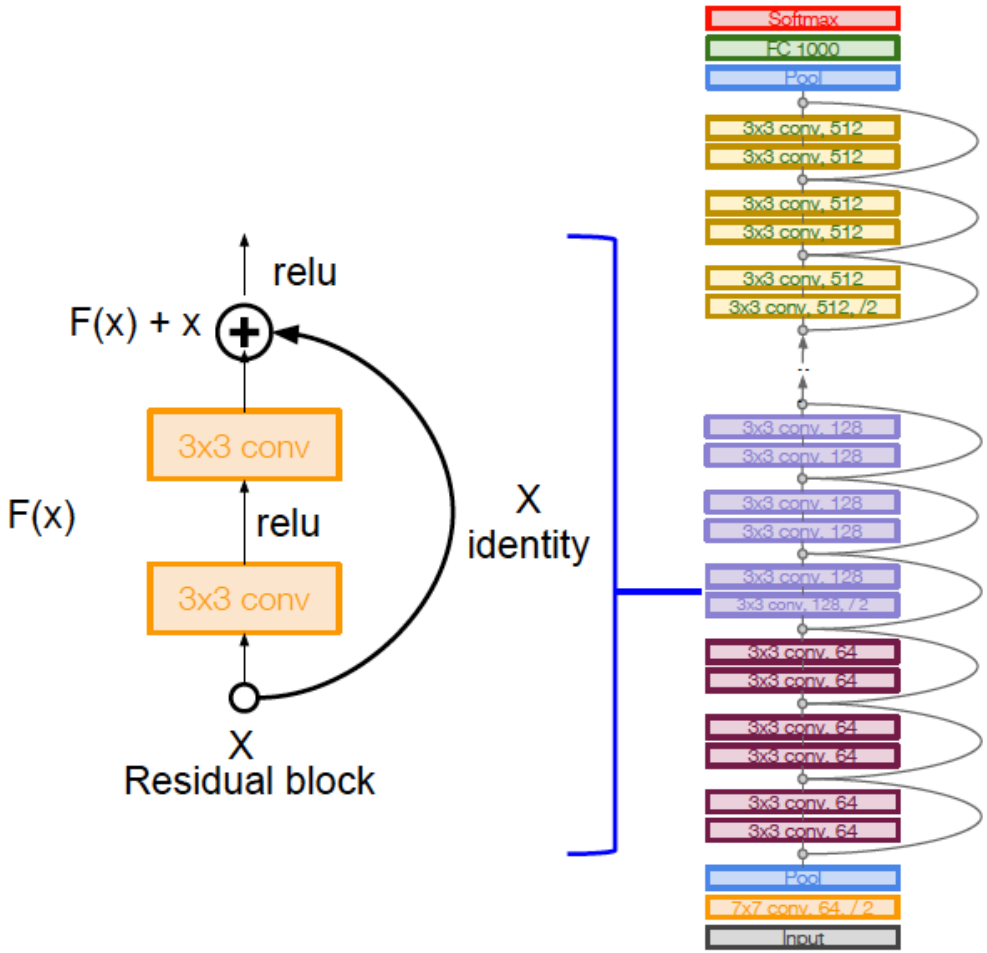
Solução: Use os layers da rede para calcular um mapa residual, ao invés de calcular o mapa adjacente desejável



ResNet

[He et al., 2015]

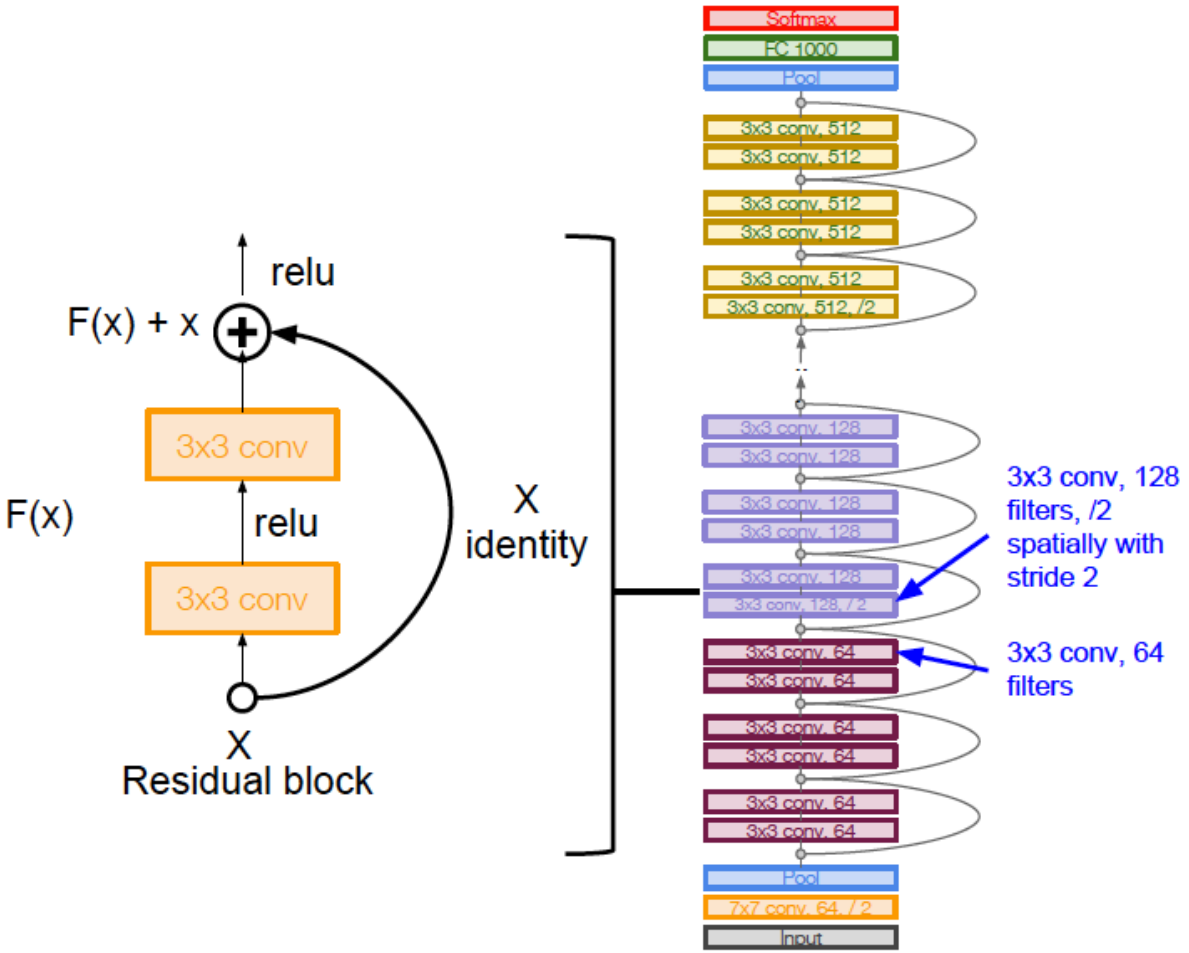
- Empilhe blocos residuais
- Cada bloco tem dois layers CONV 3x3



ResNet

[He et al., 2015]

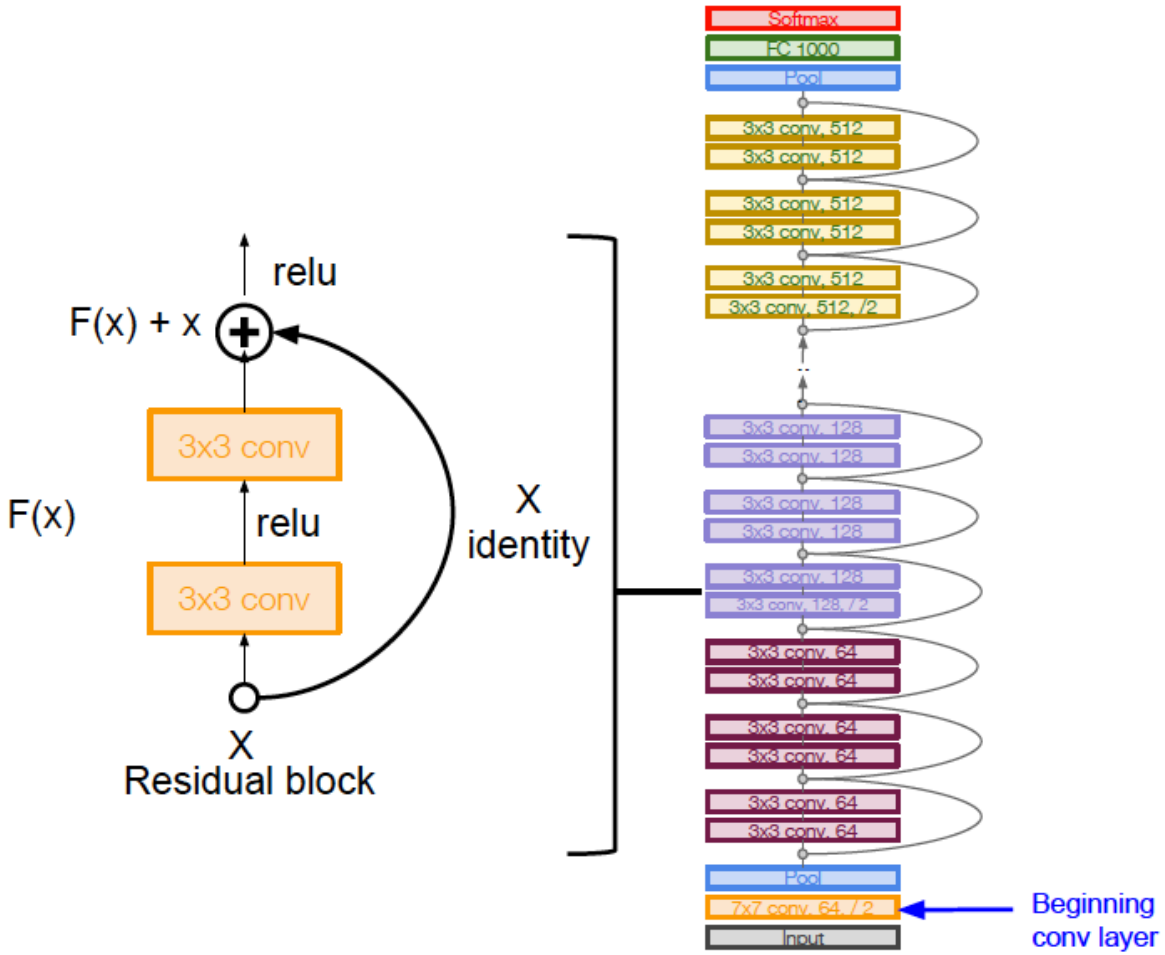
- Empilhe blocos residuais
- Cada bloco tem dois layers CONV 3x3
- Periodicamente dobre o número de filtros e reduza espacialmente com stride 2



ResNet

[He et al., 2015]

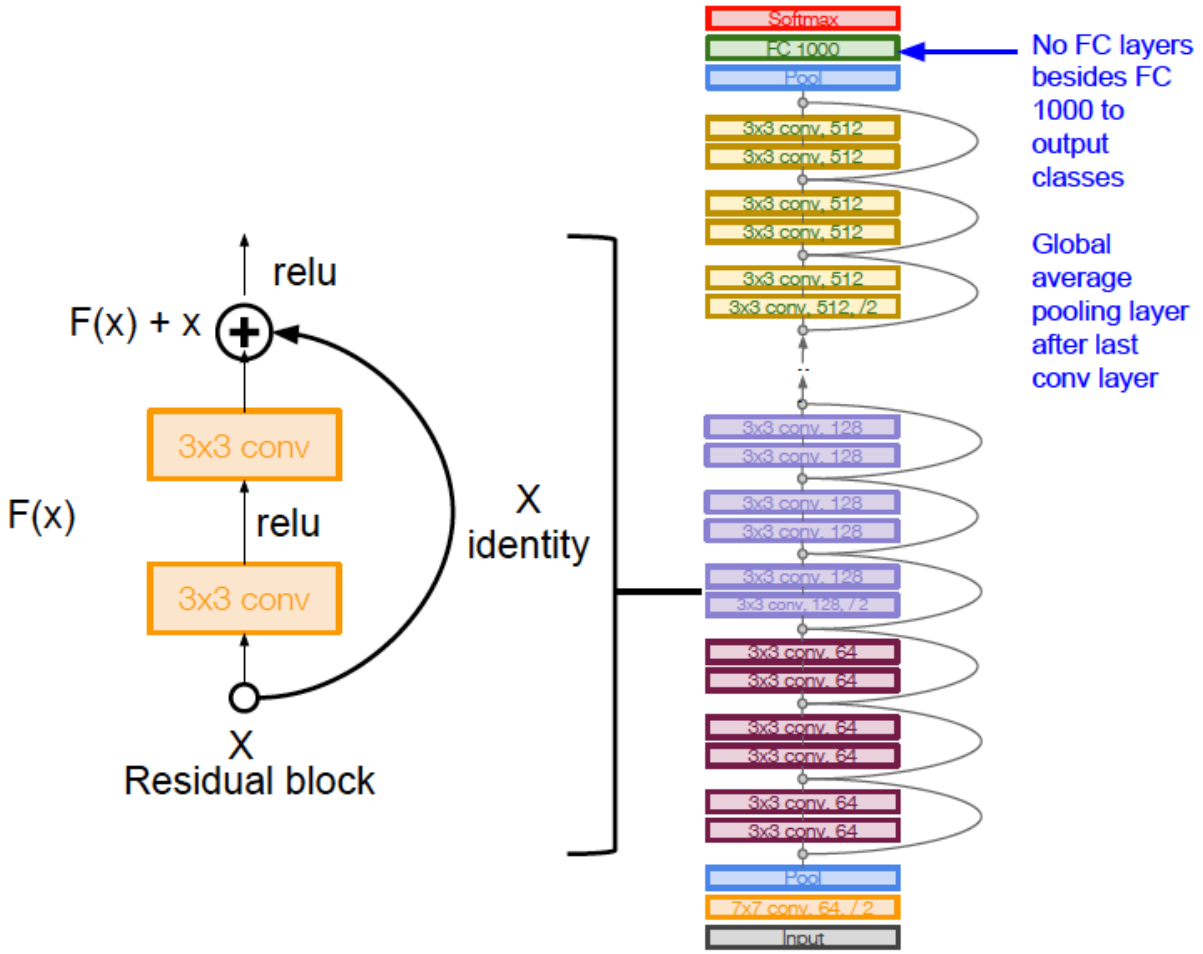
- Empilhe blocos residuais
- Cada bloco tem dois layers CONV 3x3
- Periodicamente dobre o número de filtros e reduza espacialmente com stride 2
- Layer CONV adicional no início



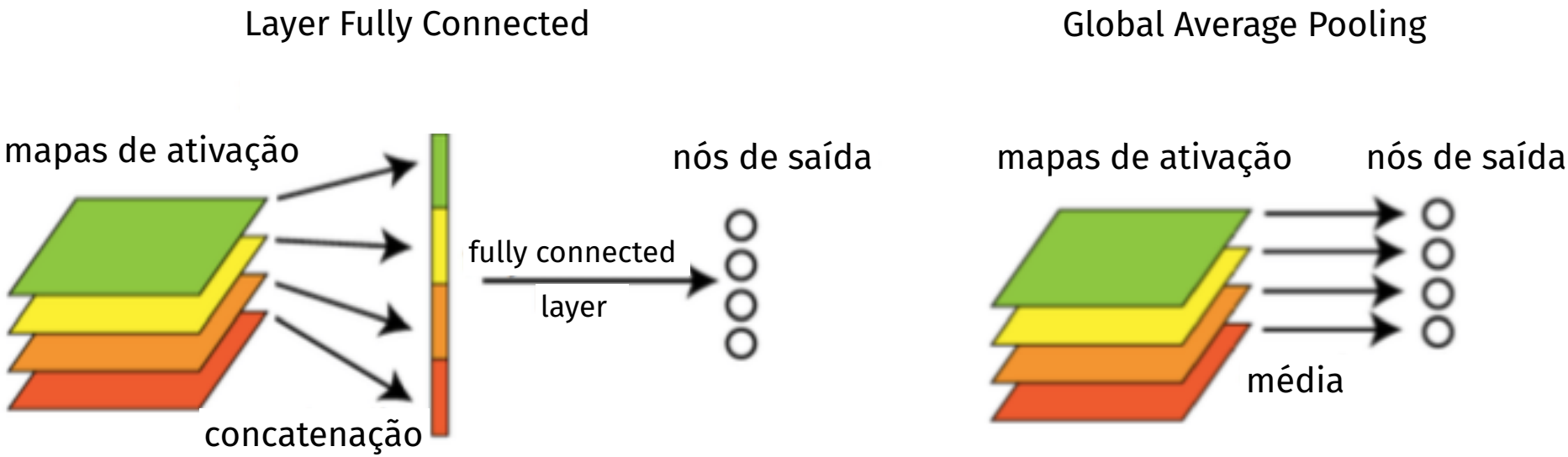
ResNet

[He et al., 2015]

- Empilhe blocos residuais
- Cada bloco tem dois layers CONV 3x3
- Periodicamente dobre o número de filtros e reduza espacialmente com stride 2
- Layer CONV adicional no início
- Apenas um layer FC ao final para o output das classes



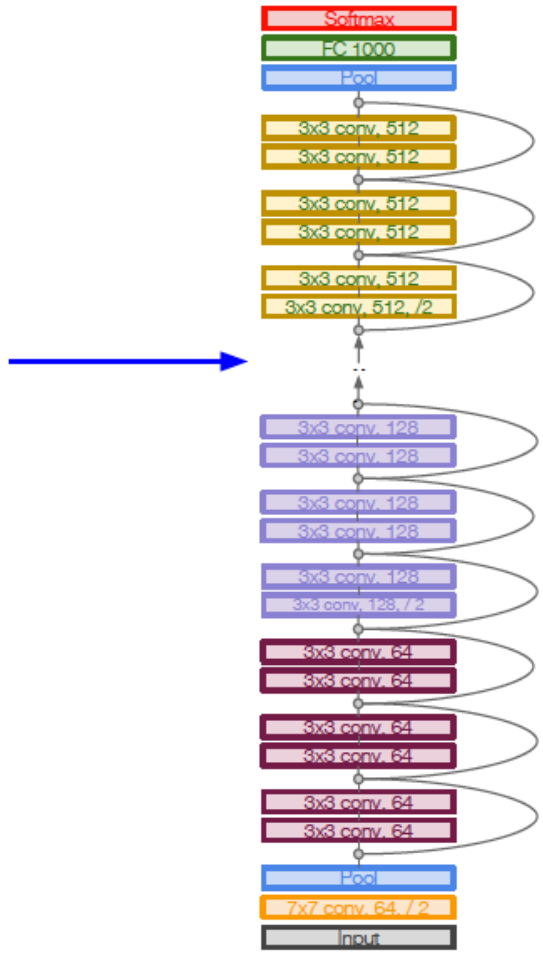
Global Average Pooling



ResNet

[He et al., 2015]

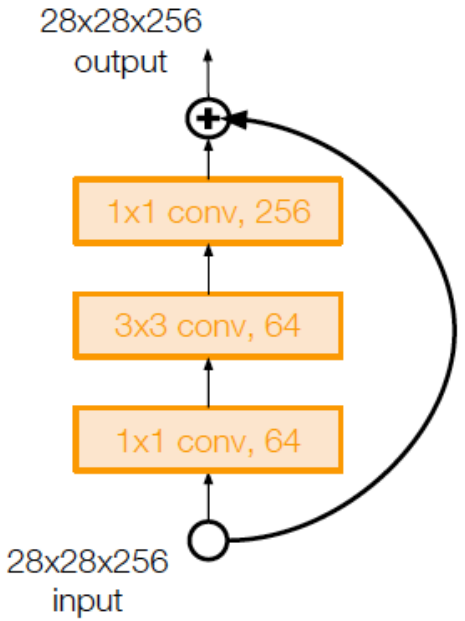
Profundidade: 34, 50, 101, ou 152 layers para a ImageNet



ResNet

[He et al., 2015]

Para redes mais profundas (ResNet-50+): utiliza camada de "gargalo" para eficiência (similar à GoogLeNet)



ResNet

[He et al., 2015]

Treinando a ResNet:

- Batch normalization após cada layer CONV
- Inicialização Xavier
- SGD + Momentum 0.9
- Learning rate: 0.1, dividido por 10 quando a validação fica em um plateau
- Mini-batch de 256
- Weight decay de 1e-5
- Não utiliza dropout

SqueezeNet

[Iandola et al., 2017]

- Módulos fire formados por:
  - uma camada “squeeze" com filtros 1x1
  - uma camada “expand" com filtros 1x1 e 3x3
- Acurácia equivalente à AlexNet na ImageNet, com 50x menos parâmetros
- Modelo 510x menor do que a AlexNet (0.5MB)

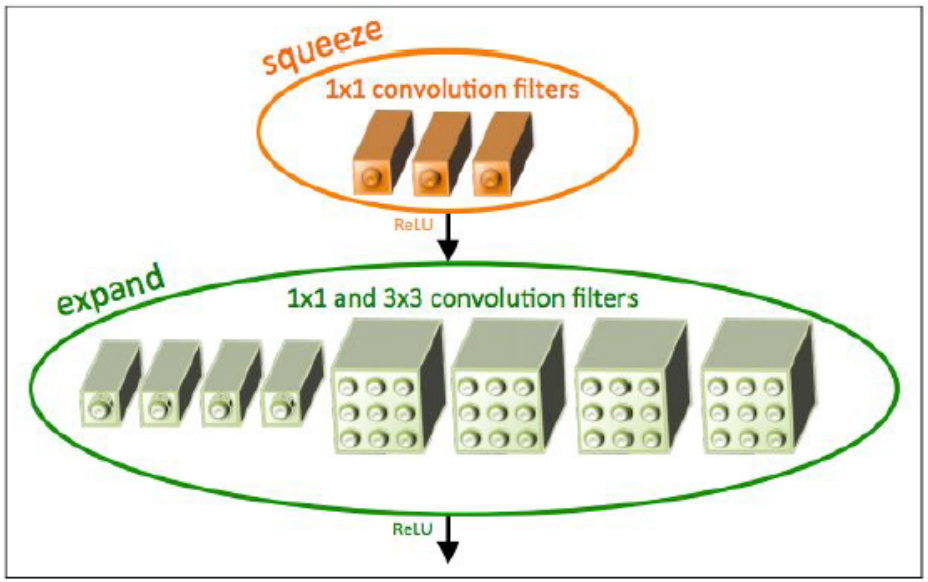
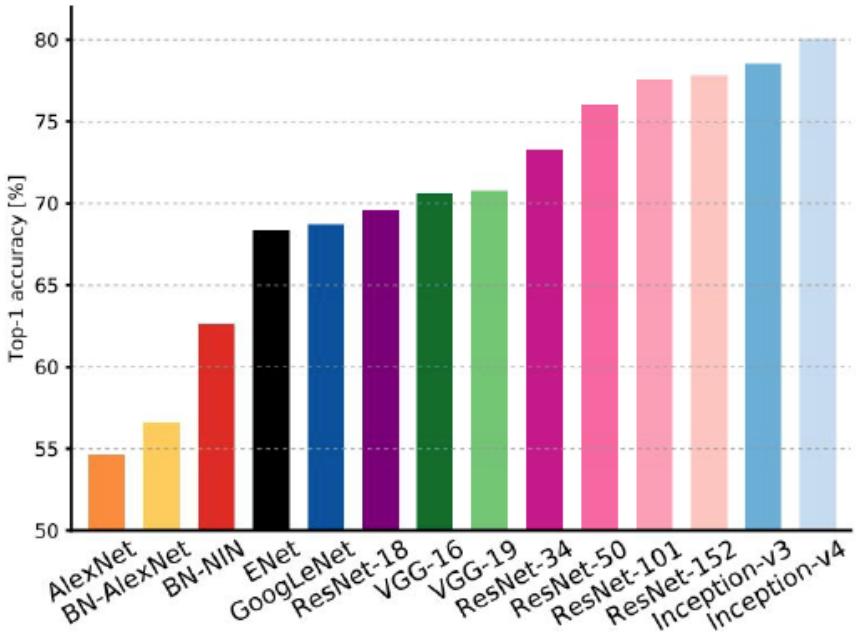
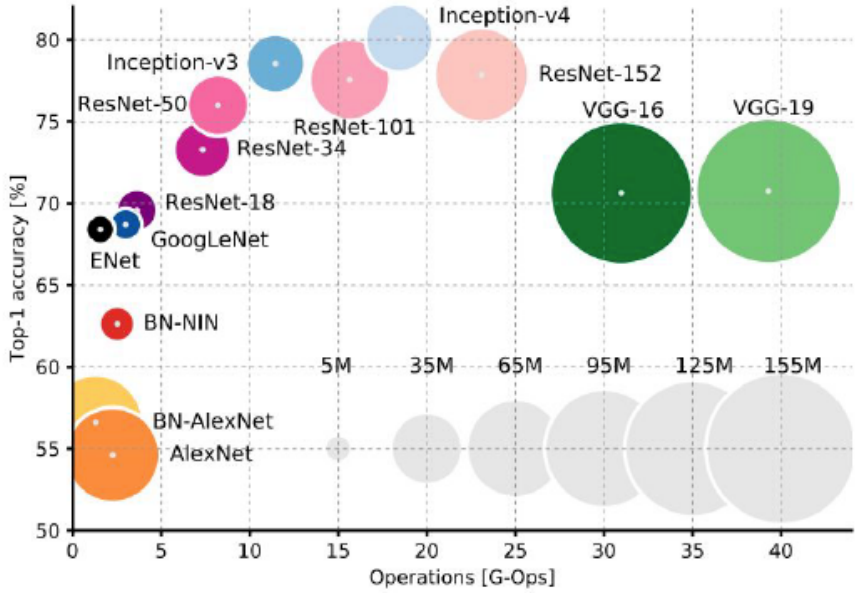


Figure copyright Iandola, Han, Moskewicz, Ashraf, Dally, Keutzer, 2017. Reproduced with permission.

Comparando a complexidade



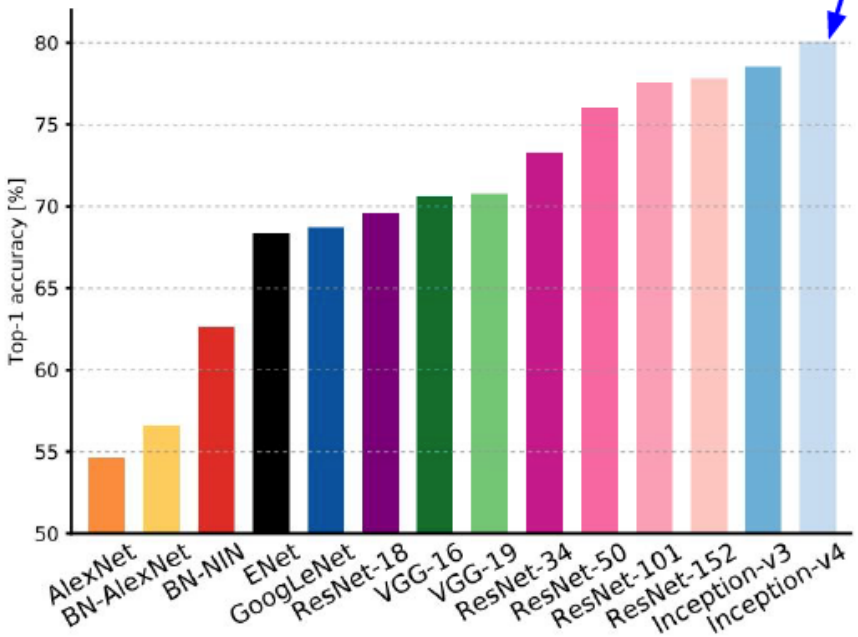


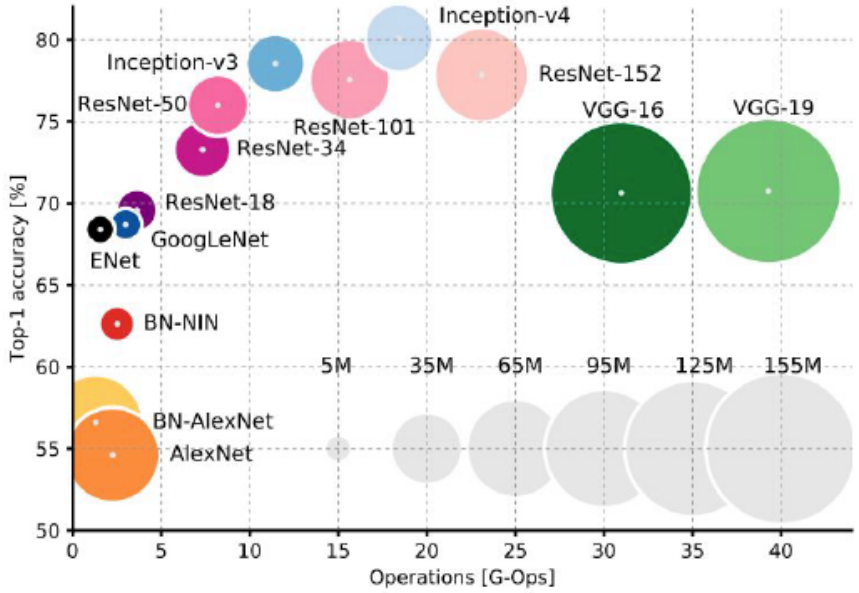
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Arquiteturas CNN

Comparando a complexidade

Inception-v4: ResNet + Inception



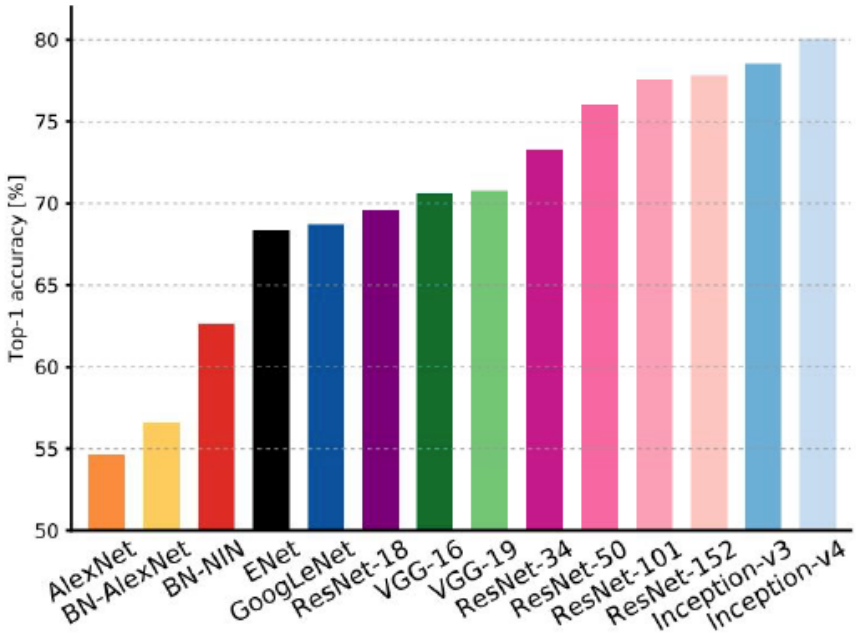


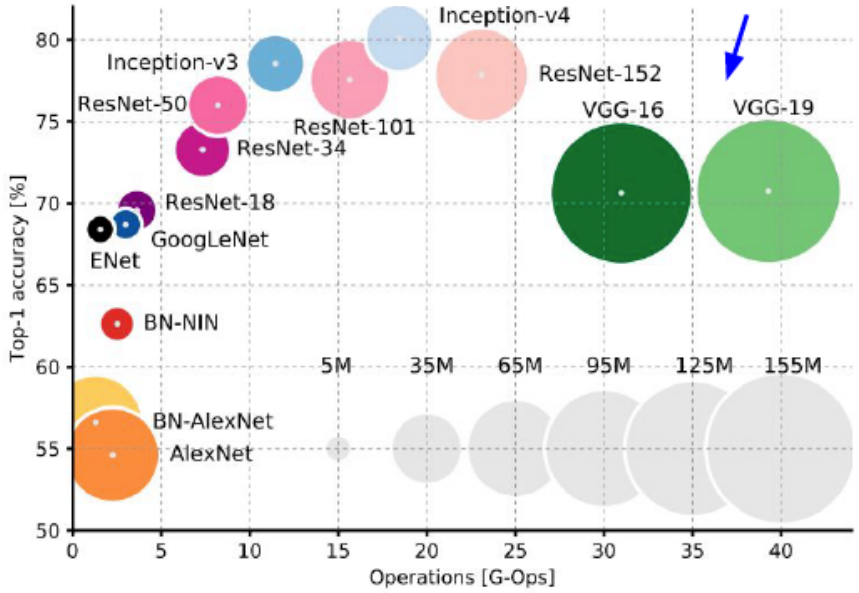
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Arquiteturas CNN

Comparando a complexidade

VGG: maior modelo,
maior número de operações



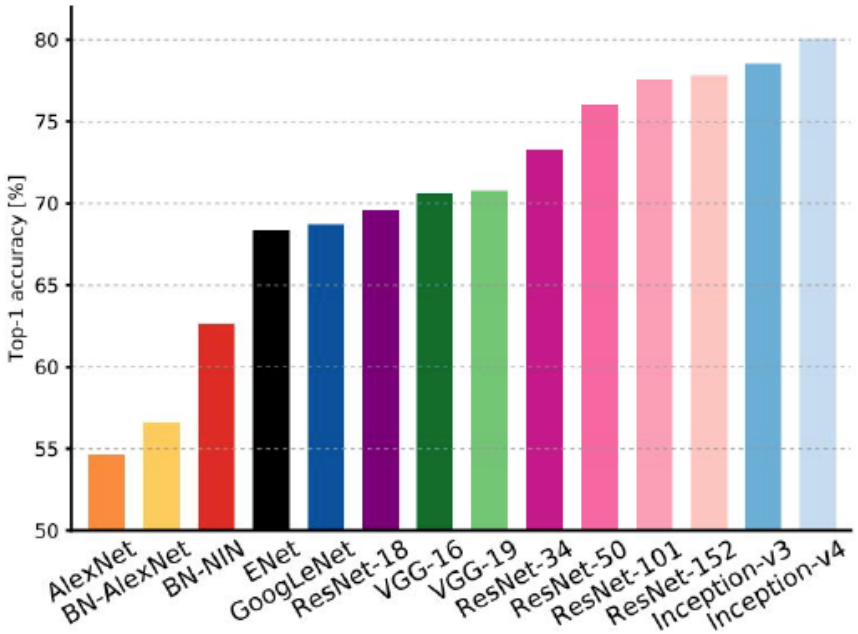


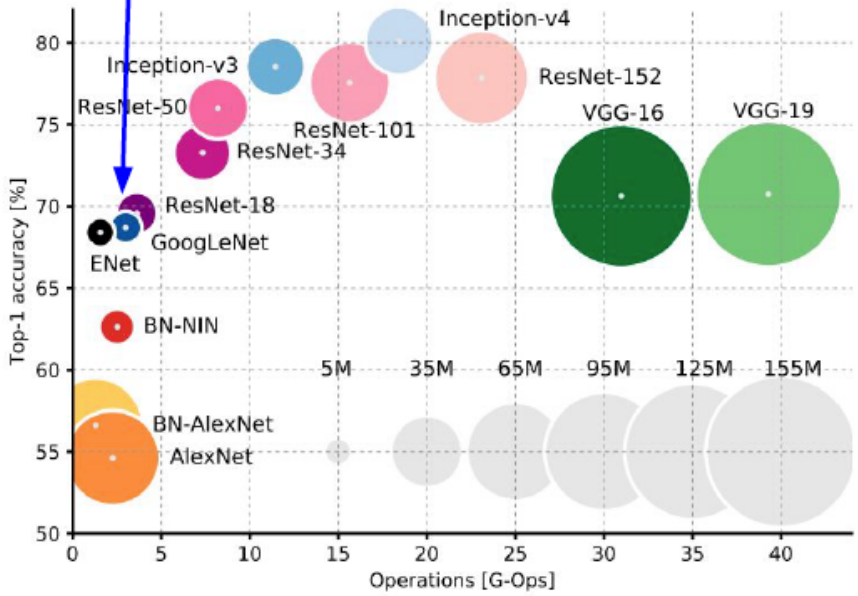
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Arquiteturas CNN

Comparando a complexidade

GoogLeNet:
mais eficiente



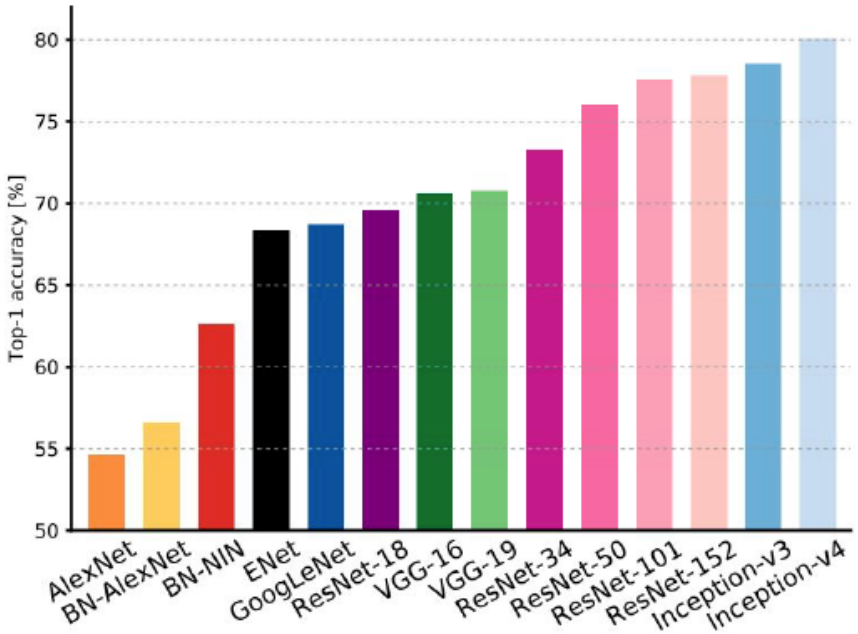


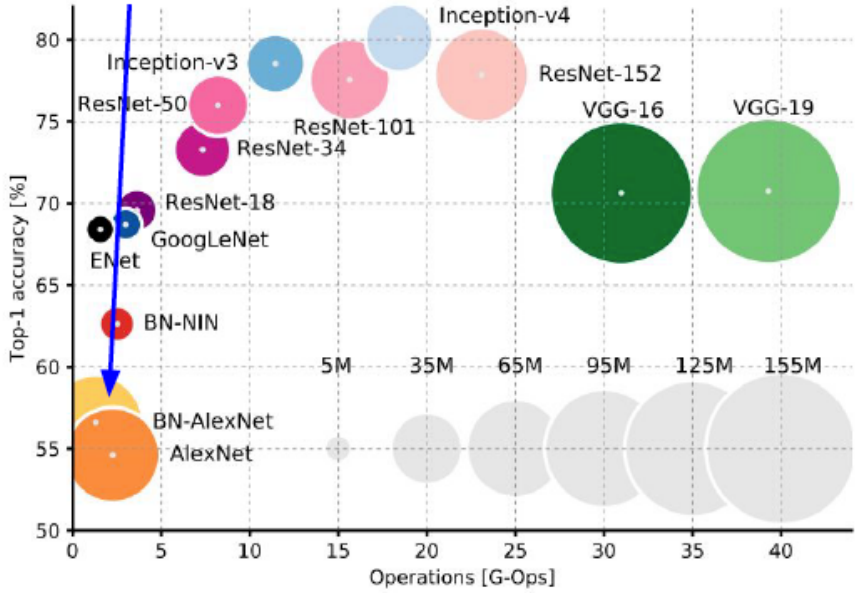
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Arquiteturas CNN

Comparando a complexidade

AlexNet:
arquitetura menor, mas modelo maior
acurácia mais baixa

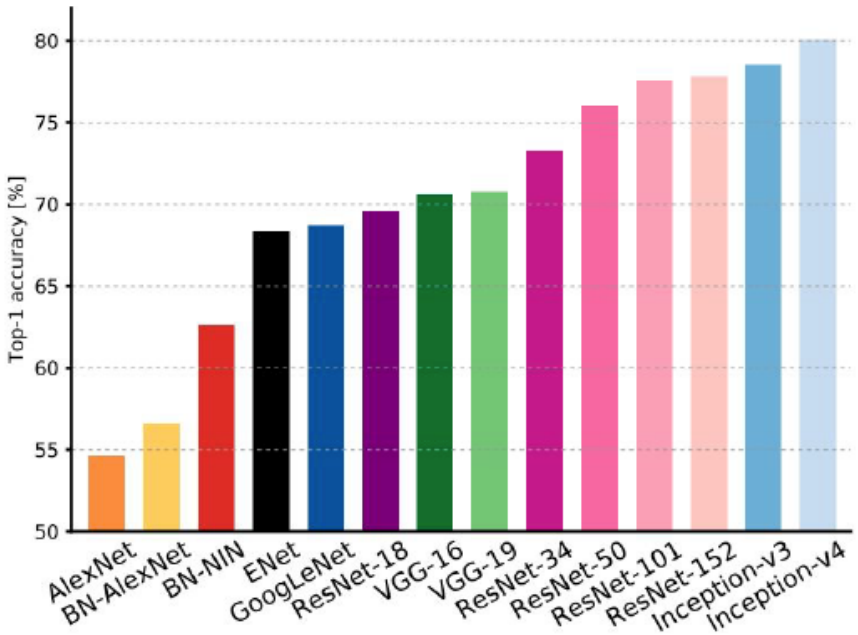




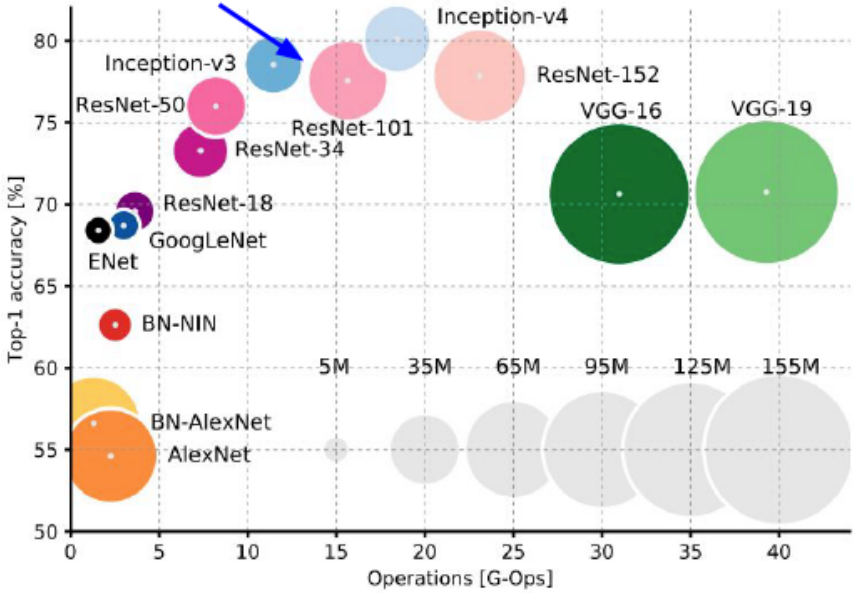
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Arquiteturas CNN

Comparando a complexidade

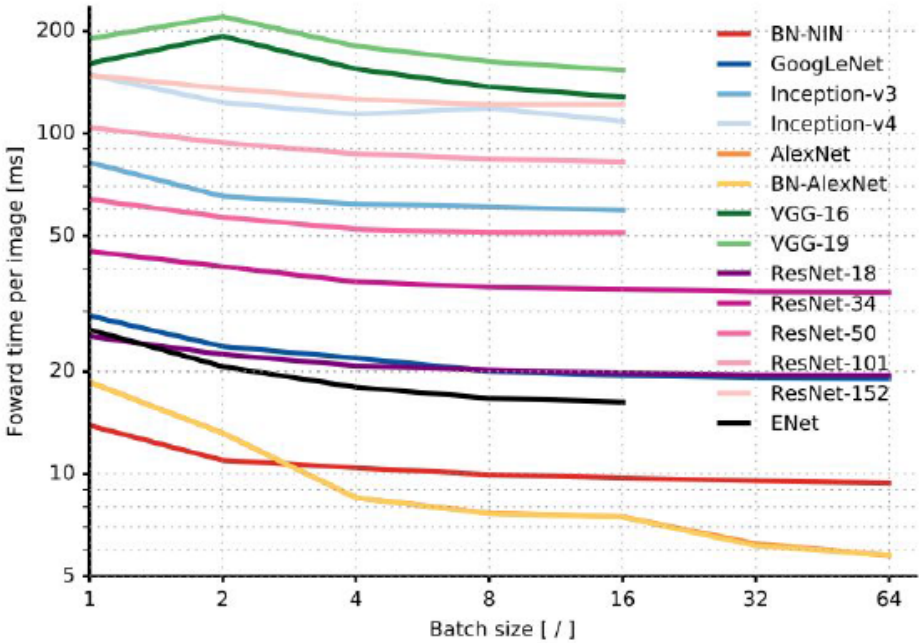


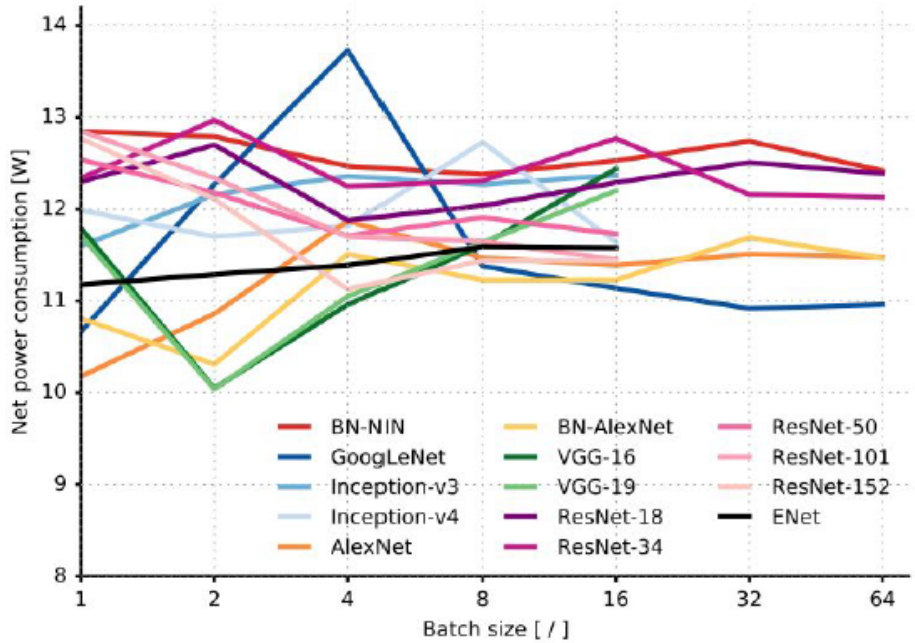
ResNet:
Eficiência depende do modelo, acurácia alta



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

Tempo de um forward pass e consumo de energia





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