

Estudos de Caso

Por que ver
estudos de caso?

Por que ver estudos de caso?

- Aprendemos sobre camadas **Conv**, de *pooling* e camadas **totalmente conectadas**
- Acontece que os pesquisadores de visão computacional passaram os últimos anos aprendendo a unir essas camadas
- Para obter algumas intuições você tem que ver os exemplos que foram feitos
- Algumas arquiteturas de redes neurais que funcionam bem em algumas tarefas também podem funcionar bem em outras tarefas

Agenda

Redes clássicas

- LeNet-5
- AlexNet
- VGG

Inception

ResNet

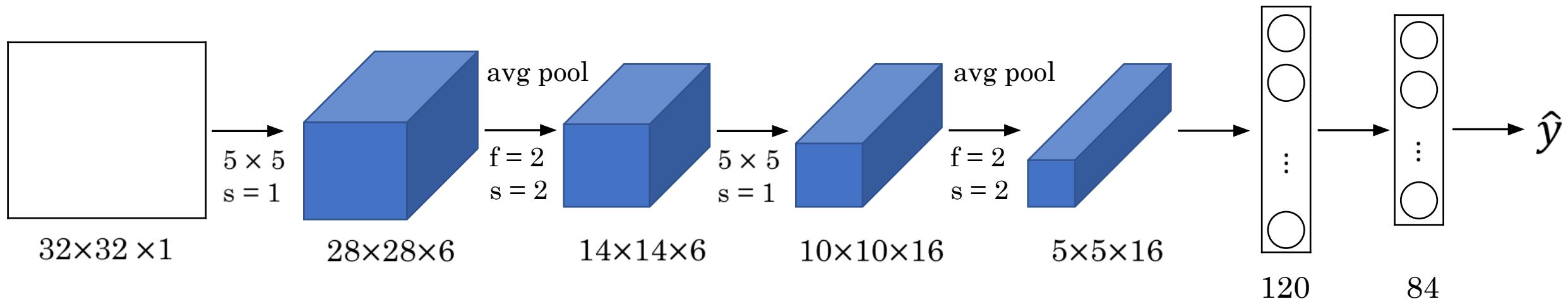
Por que ver estudos de caso?

- A primeira rede a bater o classificador humano na competição da ImageNet é chamada ResNet e tem **152 camadas!**
- Há também uma arquitetura chamada **Inception** que foi feita pelo **Google** e que é muito útil, tanto para aprender quanto para aplicar em suas tarefas
- **Aprender e experimentar** os modelos mencionados pode impulsioná-lo e dar-lhe muitas ideias para resolver a sua tarefa

Estudos de Caso

Classic networks

LeNet - 5

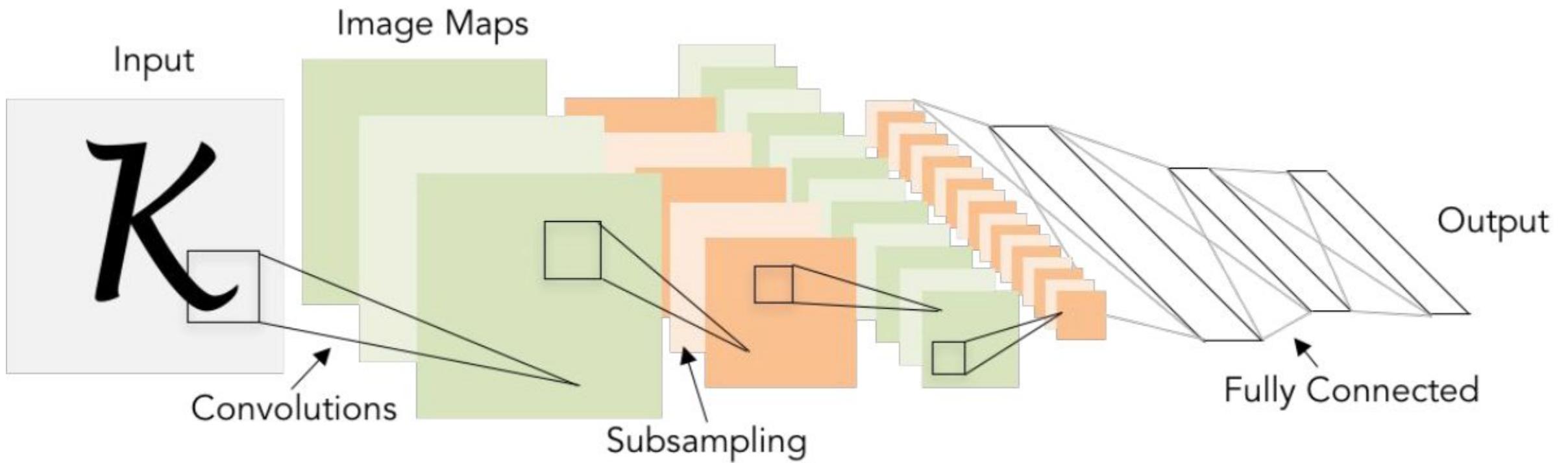


LeNet - 5

- Este modelo foi publicado em **1998**
 - A última camada **não estava usando** o softmax naquela época
- Tem **60k** parâmetros.
- As dimensões da imagem diminuem conforme o número de canais aumenta
- Conv => Pool => Conv => Pool => FC => FC => softmax
 - Esse tipo de arranjo é bastante comum
- A função de ativação usada foi **Sigmoid** e **Tanh**
- A implementação moderna usa o **RELU** na maioria dos casos

[\[LeCun et al., 1998. Gradient-based learning applied to document recognition\]](#)

LeNet - 5



AlexNet

- Nomeada assim por causa de **Alex Krizhevsky**, que foi o primeiro autor deste trabalho
- Os outros autores incluem **Geoffrey Hinton**
- O objetivo do modelo foi o desafio **ImageNet**, que classifica as imagens em **1000 classes**
- [\[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks\]](#)

AlexNet

- Conv => Max-pool => Conv => Max-pool => Conv => Conv => Conv => Max-pool => Flatten => FC => FC => Softmax
- Semelhante ao **LeNet-5**, mas maior
 - Tem **60 milhões** de parâmetros, enquanto LeNet-5 tem **60k**
 - Usa a função de ativação **RELU**
 - O artigo original contém várias GPUs e normalização de resposta local (RN)
 - Múltiplas GPUs foram usadas porque as GPUs não eram tão rápidas naquela época
 - Pesquisadores provaram que a normalização da Resposta Local **não ajuda muito**, por enquanto, não se incomode em entendê-la ou implementá-la
- Convenceu os pesquisadores de VC sobre a importância de DL

AlexNet

Arquitetura:

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

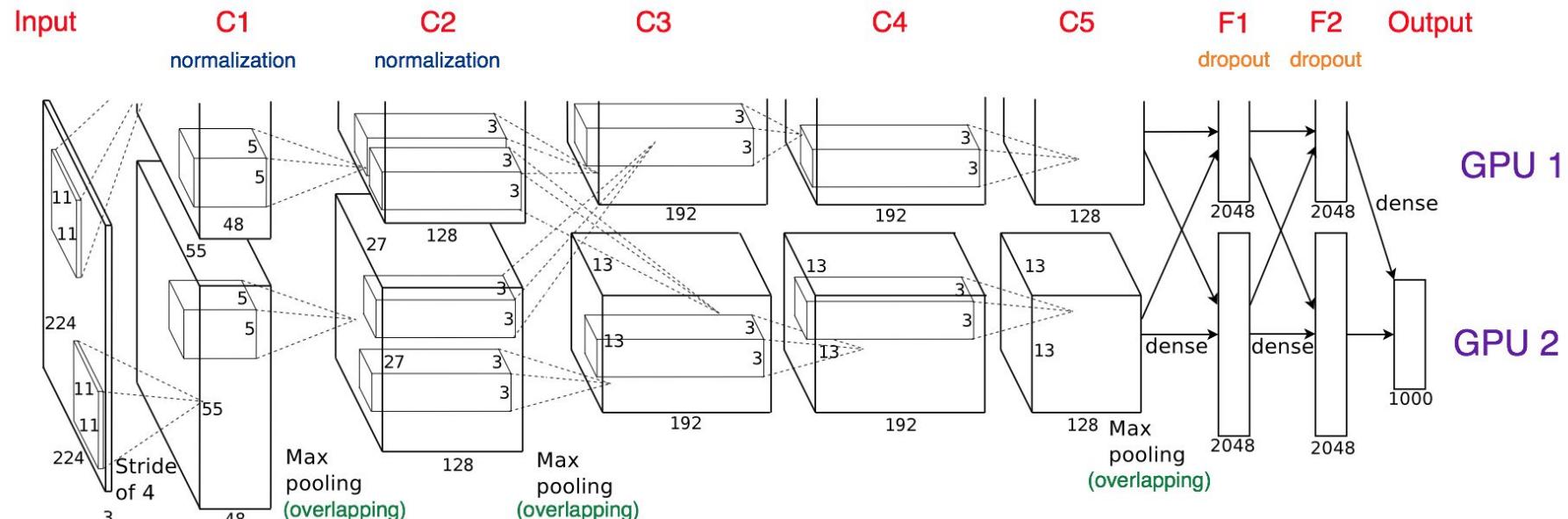
CONV5

MAX POOL3

FC6

FC7

FC8



AlexNet

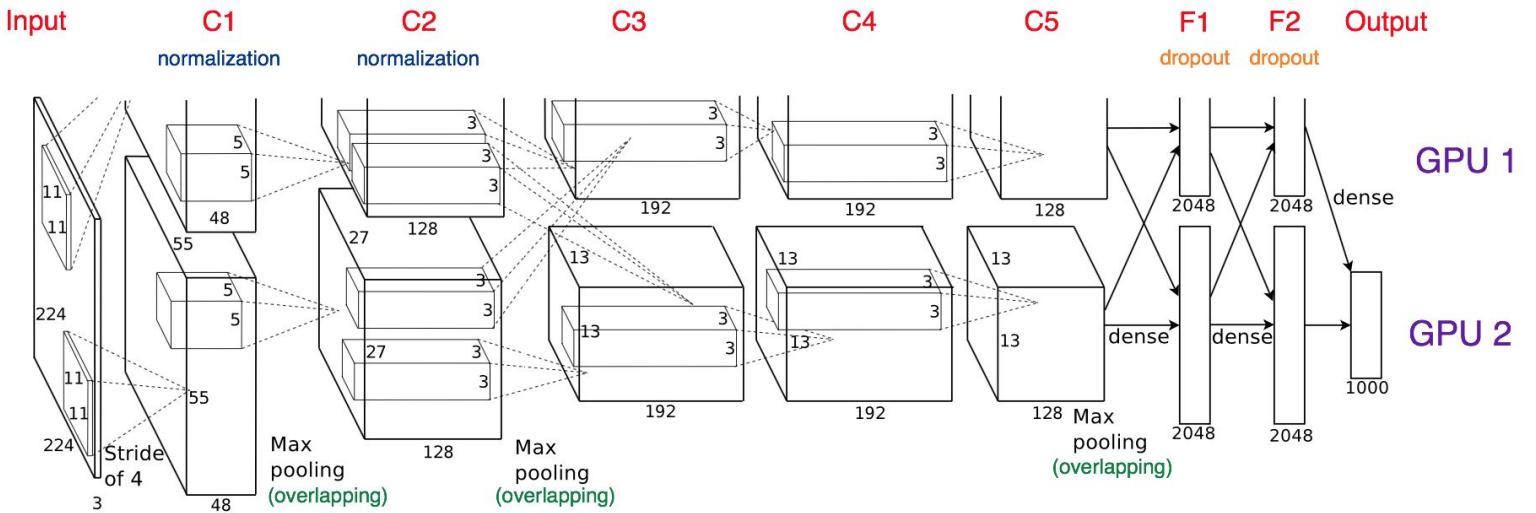


Imagen de entrada: 227 x 227 x 3

Primeira camada (CONV1): 96 filtros 11 x 11 aplicados com um *stride* de 4

Q: qual é o tamanho do volume de saída? Dica: $(227-11)/4+1 = 55$

AlexNet

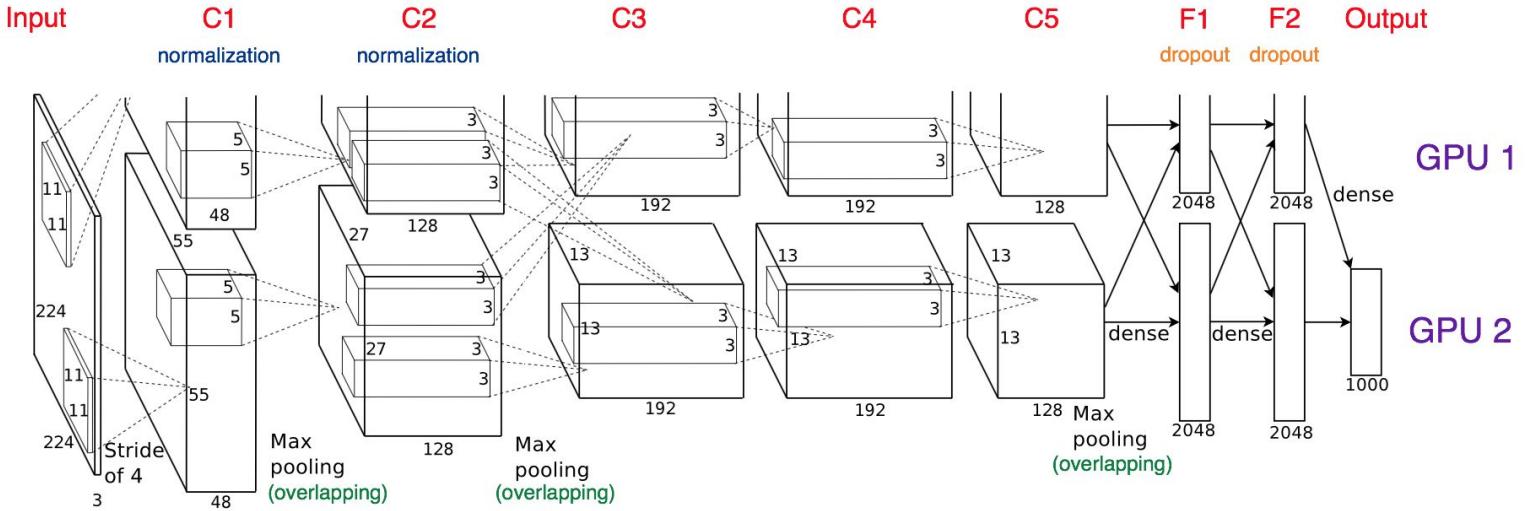


Imagen de entrada: 227 x 227 x 3

Primeira camada (CONV1): 96 filtros 11 x 11 aplicados com um *stride* de 4

Q: qual é o tamanho do volume de saída? Dica: $(227-11)/4+1 = 55$

A: (55 x 55 x 96)

Q: Qual é número total de parâmetros desta camada?

AlexNet

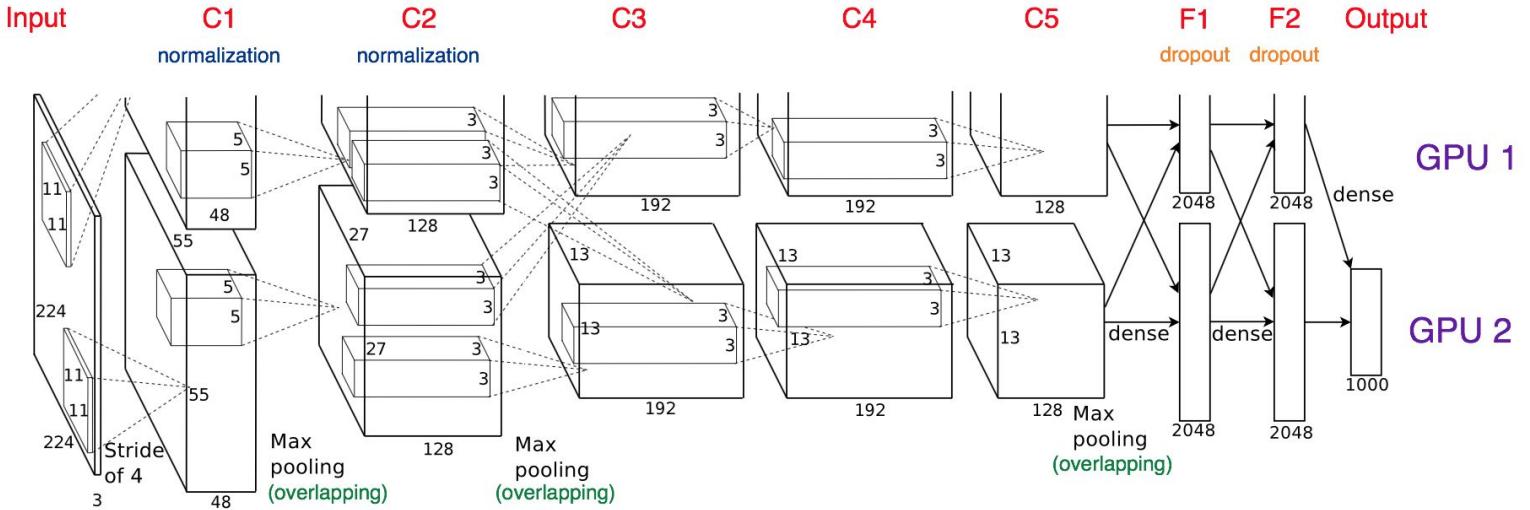


Imagen de entrada: 227 x 227 x 3

Primeira camada (CONV1): 96 filtros 11 x 11 aplicados com um *stride* de 4

Q: qual é o tamanho do volume de saída? Dica: $(227-11)/4+1 = 55$

A: (55 x 55 x 96)

Q: Qual é número total de parâmetros desta camada?

A: $(11 * 11 * 3)*96 + 96 = 35k$

AlexNet

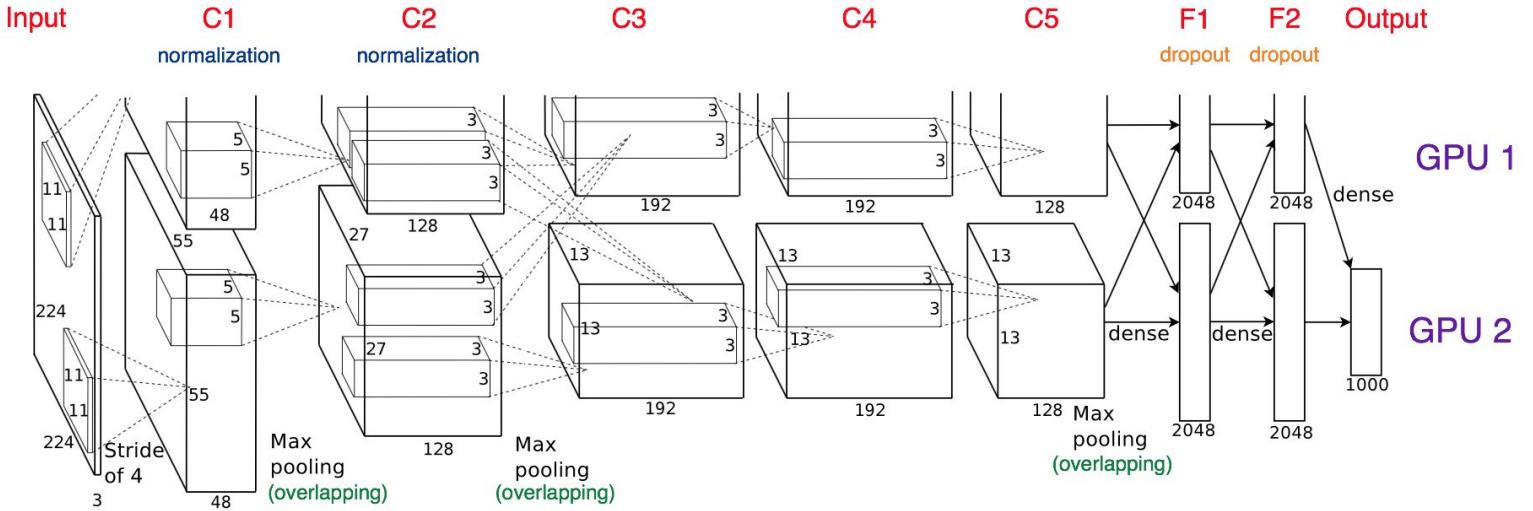


Imagen de entrada: 227 x 227 x 3

Depois de CONV1: volumes de tamanho (55 x 55 x 96)

Segunda camada (MAX POOL1): filtros 3x3 aplicados com um *stride* de 2

Q: qual é o tamanho do volume de saída? Dica: $(55-3)/2+1 = 27$

AlexNet

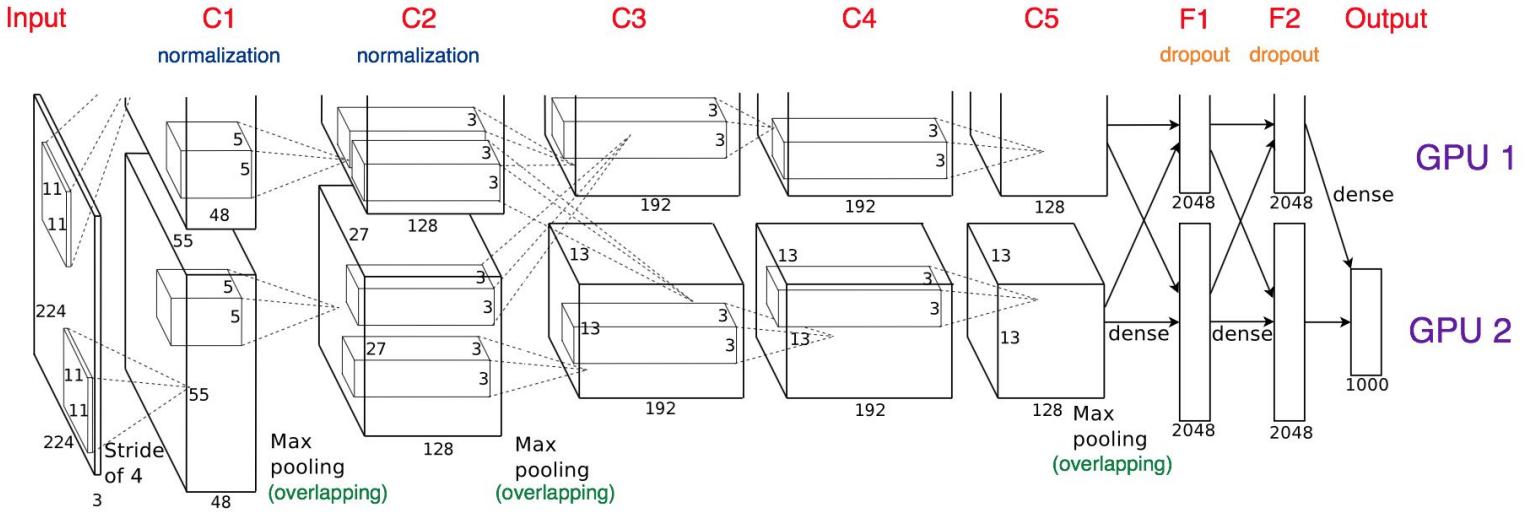


Imagen de entrada: 227 x 227 x 3

Depois de CONV1: volumes de tamanho (55 x 55 x 96)

Segunda camada (MAX POOL1): filtros 3x3 aplicados com um *stride* de 2

Q: qual é o tamanho do volume de saída?

A: (27 x 27 x 96)

Q: Qual é número total de parâmetros desta camada?

AlexNet

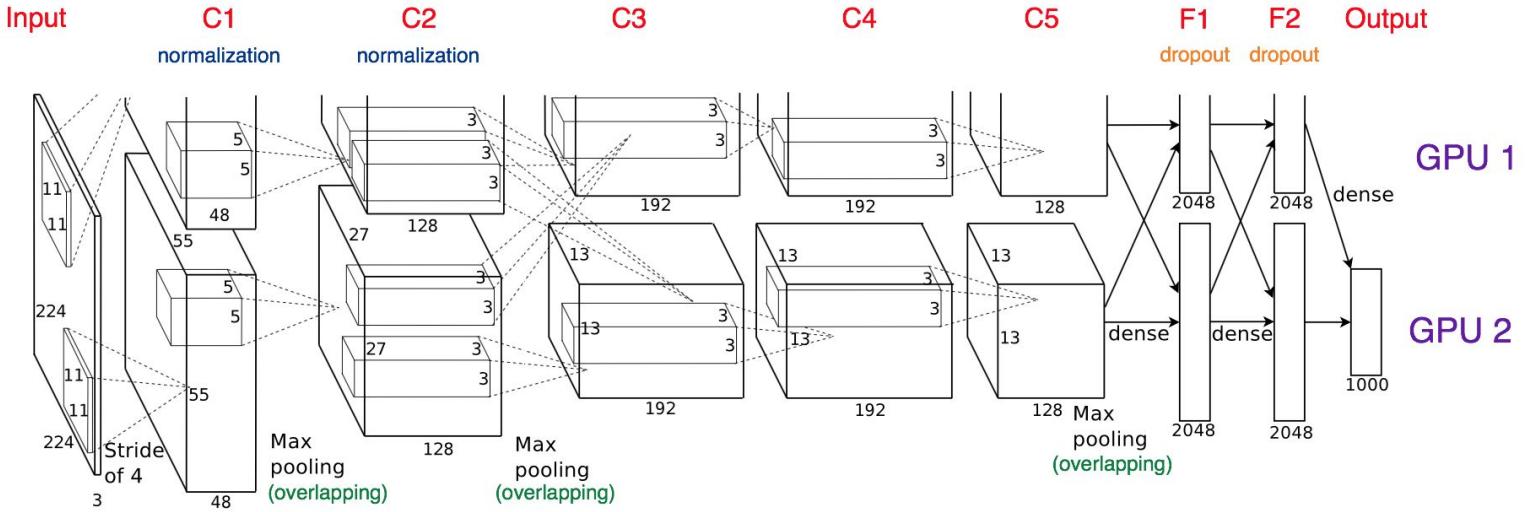


Imagen de entrada: 227 x 227 x 3

Depois de CONV1: volumes de tamanho (55 x 55 x 96)

Segunda camada (MAX POOL1): filtros 3x3 aplicados com um *stride* de 2

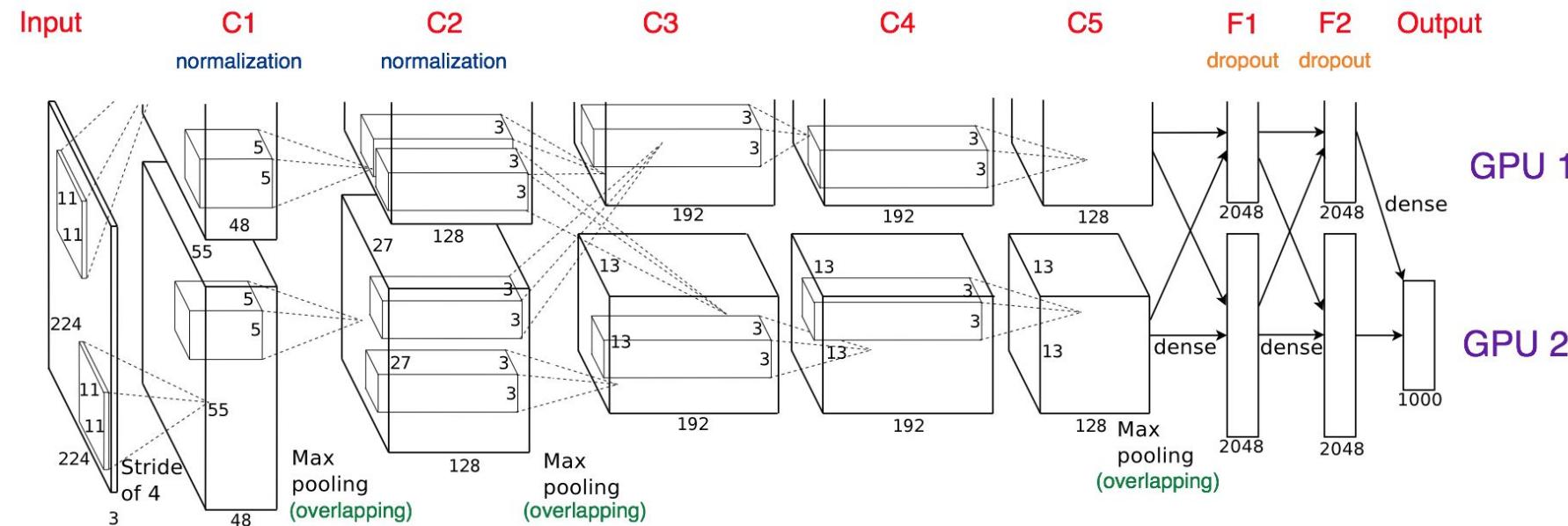
Q: qual é o tamanho do volume de saída? Dica: $(55-3)/2+1 = 27$

A: (27 x 27 x 96)

Q: Qual é número total de parâmetros desta camada?

A: 0!

AlexNet



Arquitetura completa:

[227x227x3] ENTRADA

[55x55x96] CONV1: 96 filtros 11x11 com **stride 4, pad 0**

[27x27x96] MAX POOL1: filtros 3x3 com **stride 2**

[27x27x96] NORM1: camada de normalização

[27x27x256] CONV2: 256 filtros 5x5 com **stride 1, pad 2**

[13x13x256] MAX POOL2: filtros 3x3 com **stride 2**

[13x13x256] NORM2: camada de normalização

[13x13x384] CONV3: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x384] CONV4: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x256] CONV5: 256 filtros 3x3 com **stride 1, pad 1**

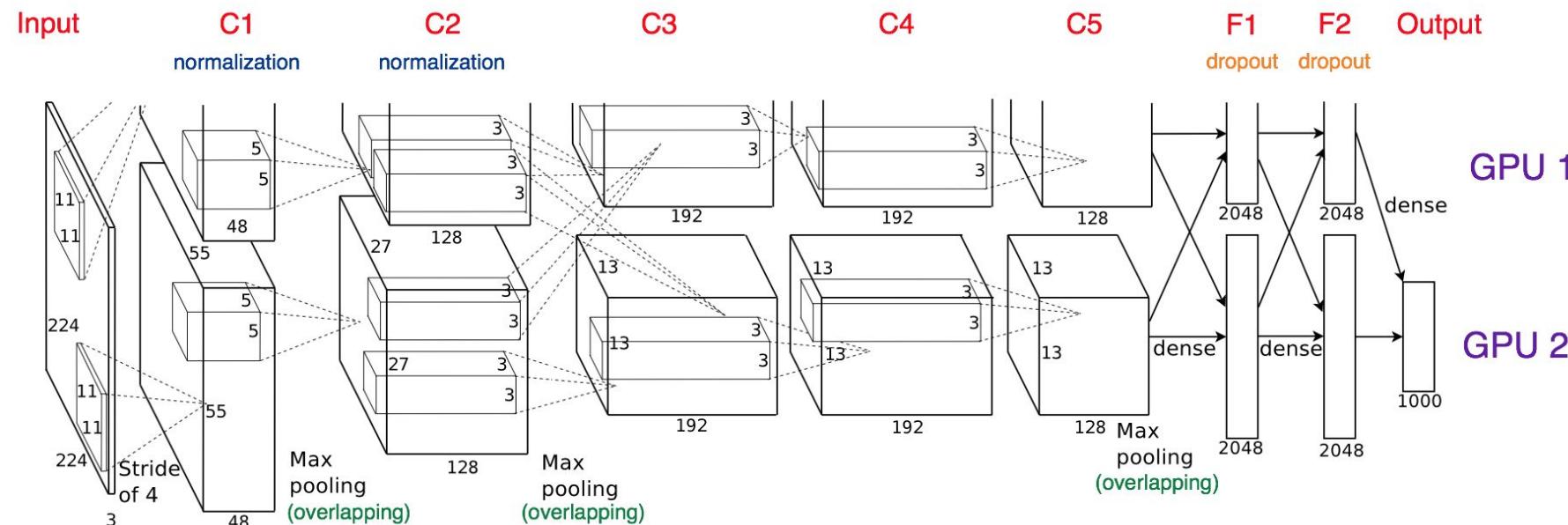
[6x6x256] MAX POOL3: filtros 3x3 com **stride 2**

[4096] FC6: 4096 neurônios

[4096] FC7: 4096 neurônios

[1000] FC8: 1000 neurônios (valores das classes)

AlexNet



Arquitetura completa:

[227x227x3] ENTRADA

[55x55x96] CONV1: 96 filtros 11x11 com **stride 4, pad 0**

[27x27x96] MAX POOL1: filtros 3x3 com **stride 2**

[27x27x96] NORM1: camada de normalização

[27x27x256] CONV2: 256 filtros 5x5 com **stride 1, pad 2**

[13x13x256] MAX POOL2: 96 filtros 3x3 com **stride 2**

[13x13x256] NORM2: camada de normalização

[13x13x384] CONV3: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x384] CONV4: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x256] CONV5: 256 filtros 3x3 com **stride 1, pad 1**

[6x6x256] MAX POOL3: filtros 3x3 com **stride 2**

[4096] FC6: 4096 neurônios

[4096] FC7: 4096 neurônios

[1000] FC8: 1000 neurônios (valores das classes)

Detalhes/Retrospectiva:

- Primeira uso da ReLU
- Usaram camadas NORM (não usa-se mais)
- Aumento de dados pesado!
- Dropout 0.5
- Tamanho do batch: 128
- SGD Momentum 0.9
- Taxa de aprendizagem de 0.01, e reduzida manualmente quando a acc estabiliza
- L2 weight decay 0.0005
- Combinação de 7 CNNs: 18.2% -> 15.4%

AlexNet

[55x55x48] x 2

Arquitetura completa:

[227x227x3] ENTRADA

[55x55x96] CONV1: 96 filtros 11x11 com **stride 4, pad 0**

[27x27x96] MAX POOL1: filtros 3x3 com **stride 2**

[27x27x96] NORM1: camada de normalização

[27x27x256] CONV2: 256 filtros 5x5 com **stride 1, pad 2**

[13x13x256] MAX POOL2: 96 filtros 3x3 com **stride 2**

[13x13x256] NORM2: camada de normalização

[13x13x384] CONV3: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x384] CONV4: 384 filtros 3x3 com **stride 1, pad 1**

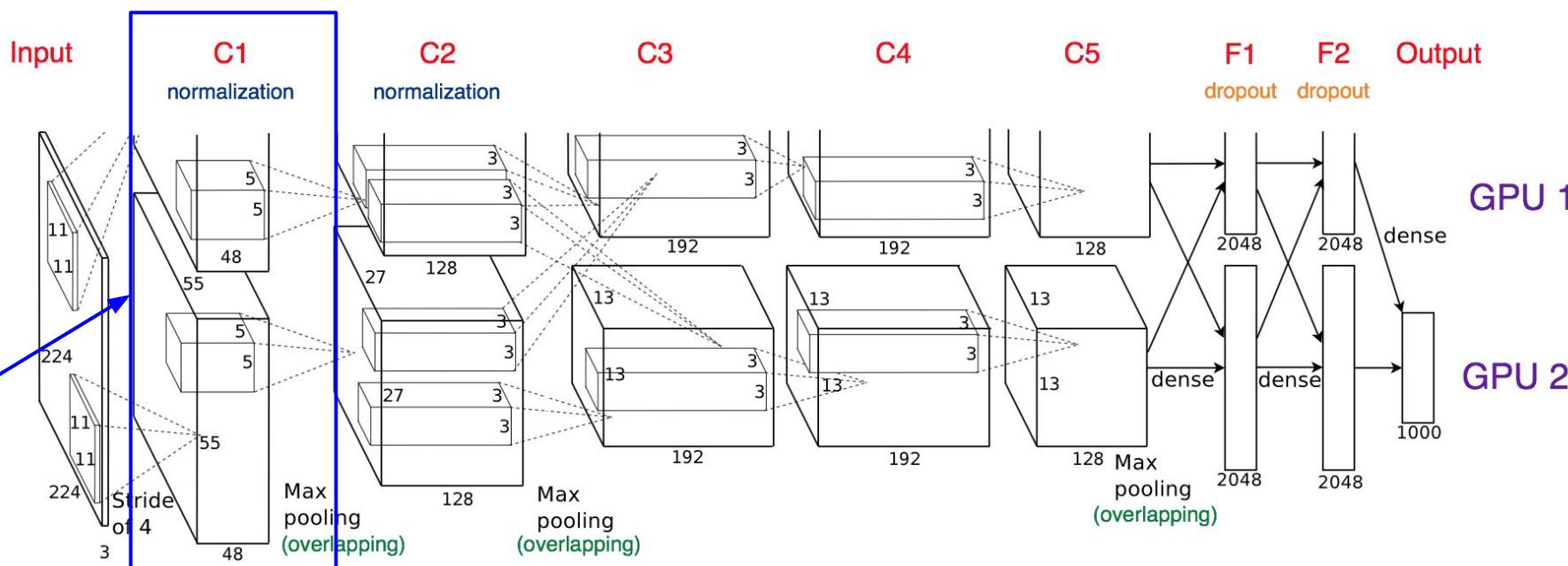
[13x13x256] CONV5: 256 filtros 3x3 com **stride 1, pad 1**

[6x6x256] MAX POOL3: filtros 3x3 com **stride 2**

[4096] FC6: 4096 neurônios

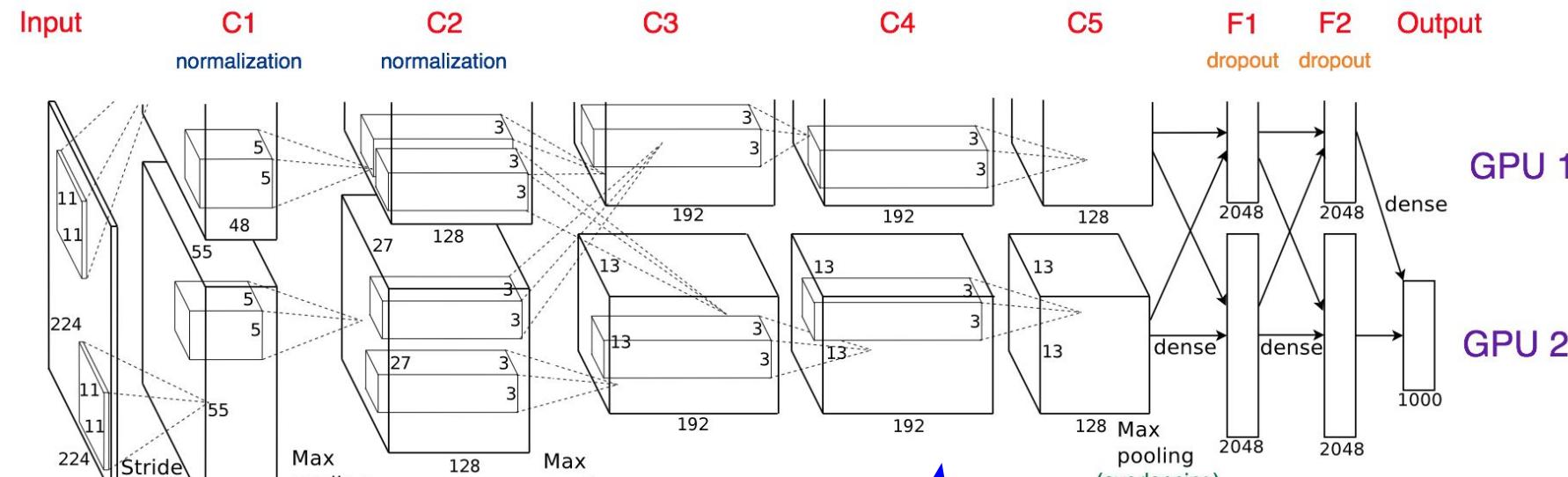
[4096] FC7: 4096 neurônios

[1000] FC8: 1000 neurônios (valores das classes)



Nota histórica: treinada em uma GPU GTX 580 com apenas 3GB de memória. Como a memória foi insuficiente, foram necessárias 2 GPUs, metade dos neurônios (mapa de features) em cada GPU.

AlexNet



Arquitetura completa:

[227x227x3] ENTRADA

[55x55x96] CONV1: 96 filtros 11x11 com **stride 4, pad 0**

[27x27x96] MAX POOL1: filtros 3x3 com **stride 2**

[27x27x96] NORM1: camada de normalização

[27x27x256] CONV2: 256 filtros 5x5 com **stride 1, pad 2**

[13x13x256] MAX POOL2: 96 filtros 3x3 com **stride 2**

[13x13x256] NORM2: camada de normalização

[13x13x384] CONV3: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x384] CONV4: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x256] CONV5: 256 filtros 3x3 com **stride 1, pad 1**

[6x6x256] MAX POOL3: filtros 3x3 com **stride 2**

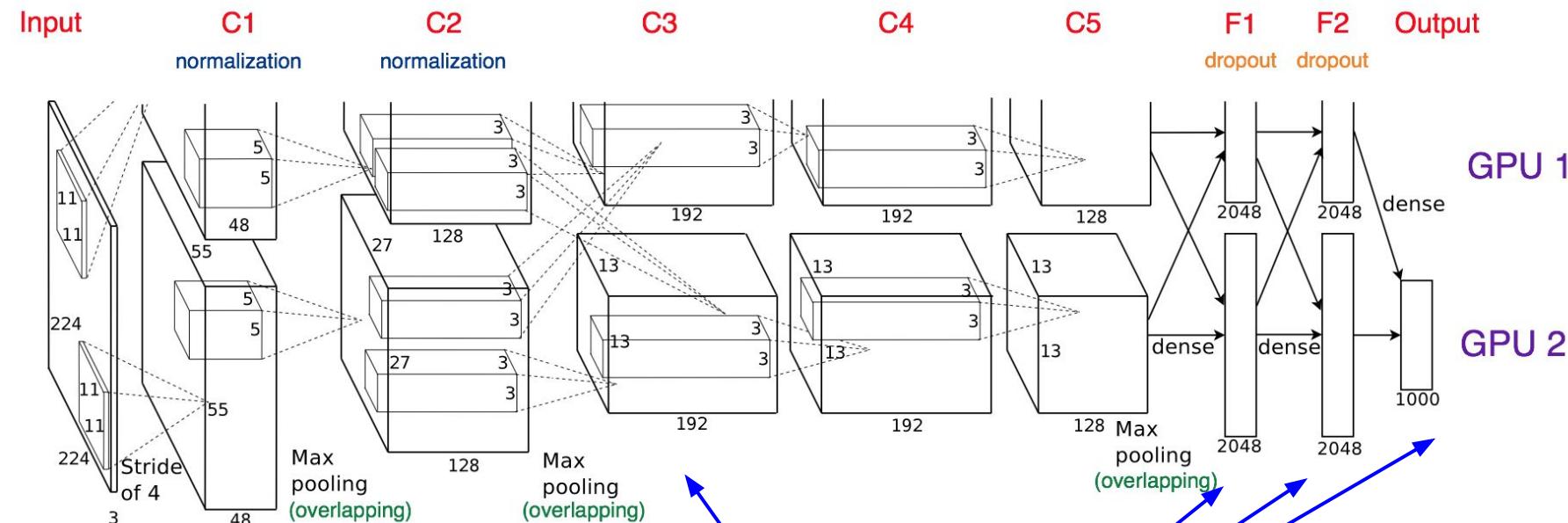
[4096] FC6: 4096 neurônios

[4096] FC7: 4096 neurônios

[1000] FC8: 1000 neurônios (valores das classes)

CONV1, CONV2, CONV4, CONV5:
Conectadas somente aos mapas de features da mesma GPU

AlexNet



Arquitetura completa:

[227x227x3] ENTRADA

[55x55x96] CONV1: 96 filtros 11x11 com **stride 4, pad 0**

[27x27x96] MAX POOL1: filtros 3x3 com **stride 2**

[27x27x96] NORM1: camada de normalização

[27x27x256] CONV2: 256 filtros 5x5 com **stride 1, pad 2**

[13x13x256] MAX POOL2: 96 filtros 3x3 com **stride 2**

[13x13x256] NORM2: camada de normalização

[13x13x384] CONV3: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x384] CONV4: 384 filtros 3x3 com **stride 1, pad 1**

[13x13x256] CONV5: 256 filtros 3x3 com **stride 1, pad 1**

[6x6x256] MAX POOL3: filtros 3x3 com **stride 2**

[4096] FC6: 4096 neurônios

[4096] FC7: 4096 neurônios

[1000] FC8: 1000 neurônios (valores das classes)

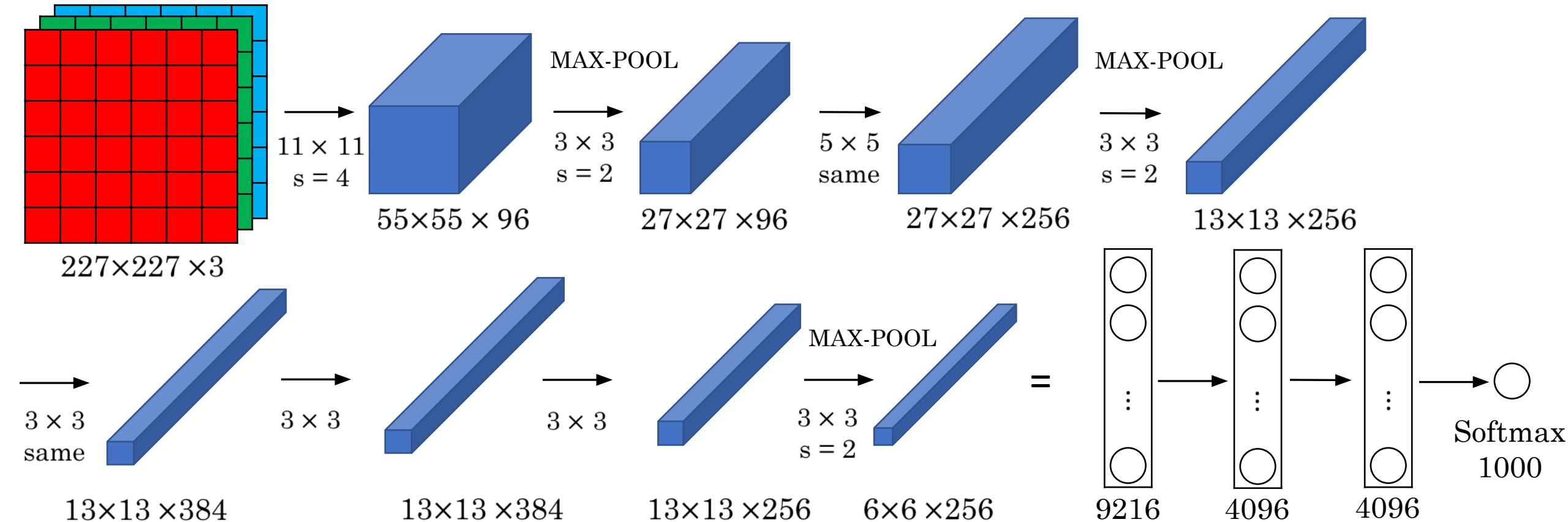
CONV3, FC6, FC7, FC8:

Conexões com todos os mapas de features em camadas anteriores, comunicação entre as GPUs

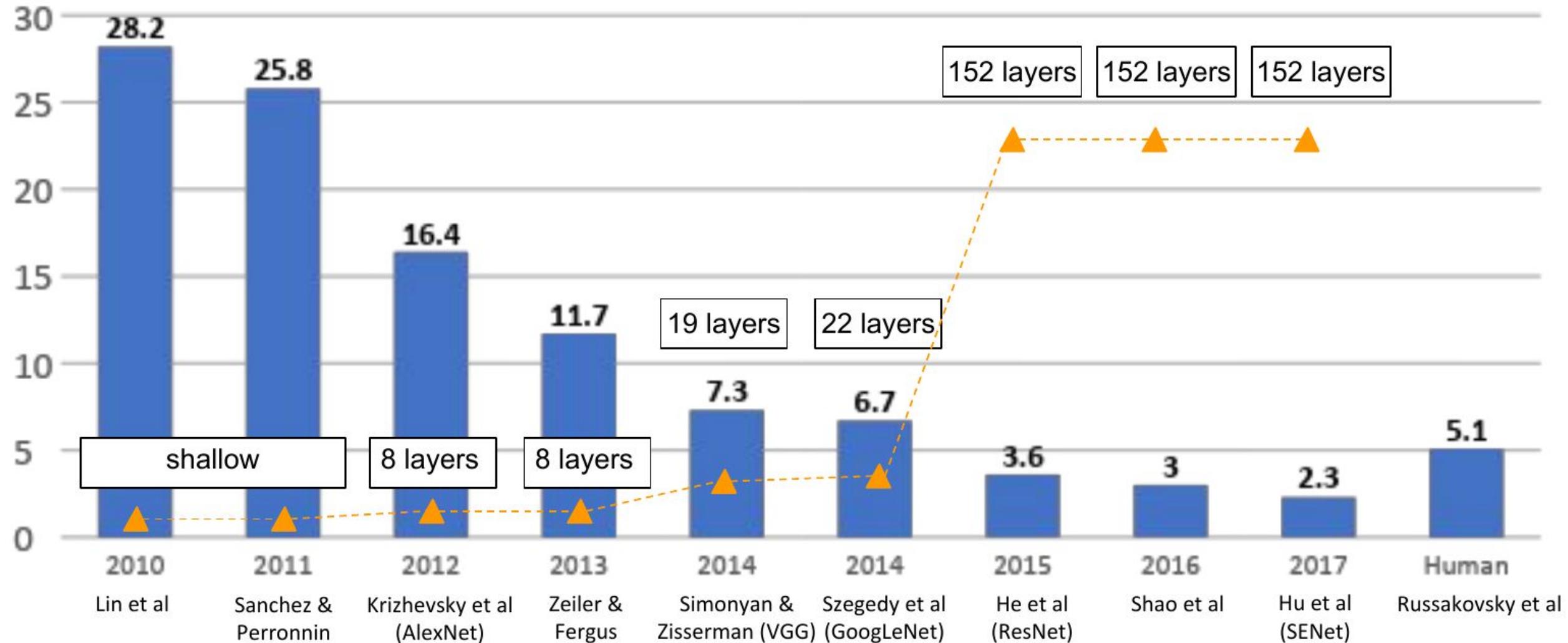
AlexNet

- Conv => Max-pool => Conv => Max-pool => Conv => Conv => Conv => Max-pool => Flatten => FC => FC => Softmax
- Semelhante ao **LeNet-5**, mas maior
 - Tem **60 milhões** de parâmetros, enquanto LeNet-5 tem **60k**
 - Usa a função de ativação **RELU**
 - O artigo original contém várias GPUs e normalização de resposta local (RN)
 - Múltiplas GPUs foram usadas porque as GPUs não eram tão rápidas naquela época
 - Pesquisadores provaram que a normalização da Resposta Local **não ajuda muito**, por enquanto, não se incomode em entendê-la ou implementá-la
- Convenceu os pesquisadores de VC sobre a importância de DL

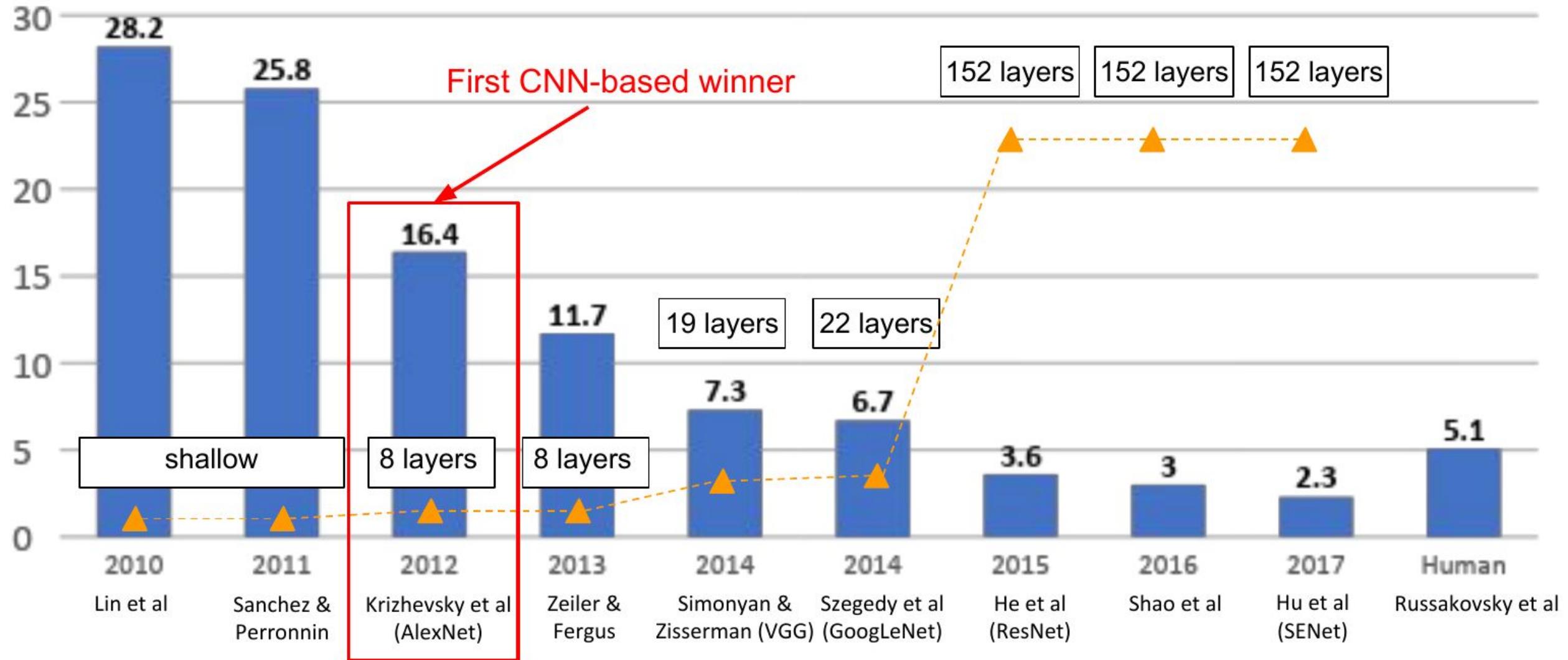
AlexNet



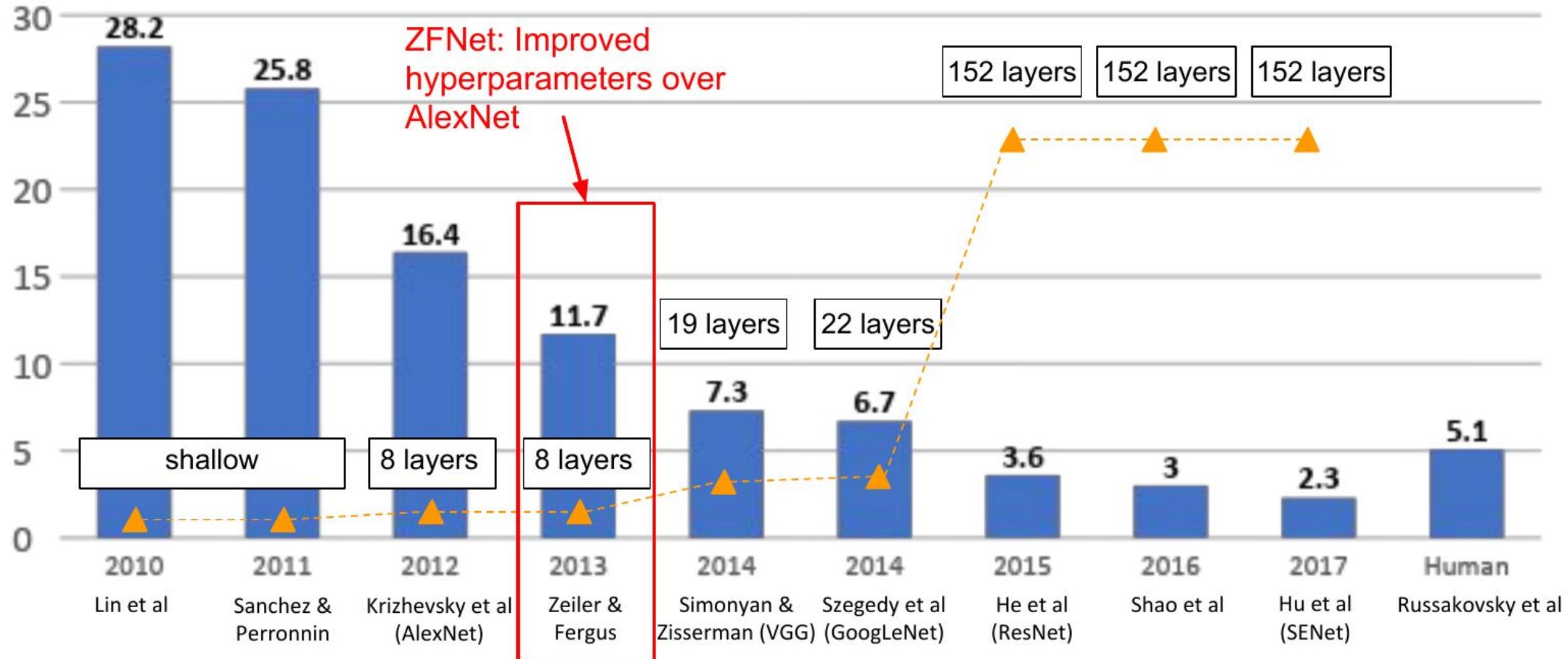
ImageNet Challenge (vencedores)



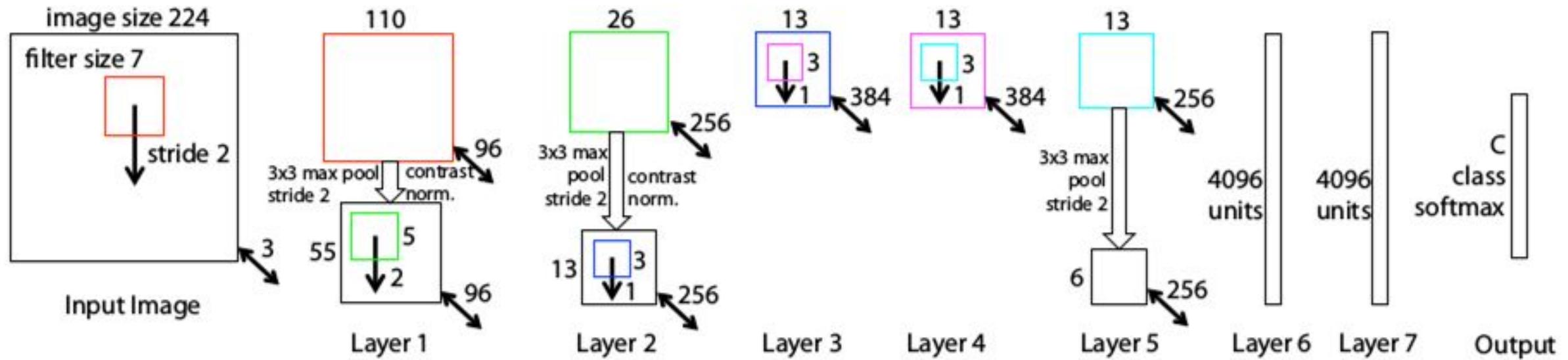
ImageNet Challenge (vencedores)



ImageNet Challenge (vencedores)



ZFNet

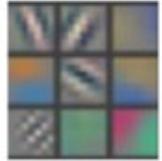


ZF Net Architecture

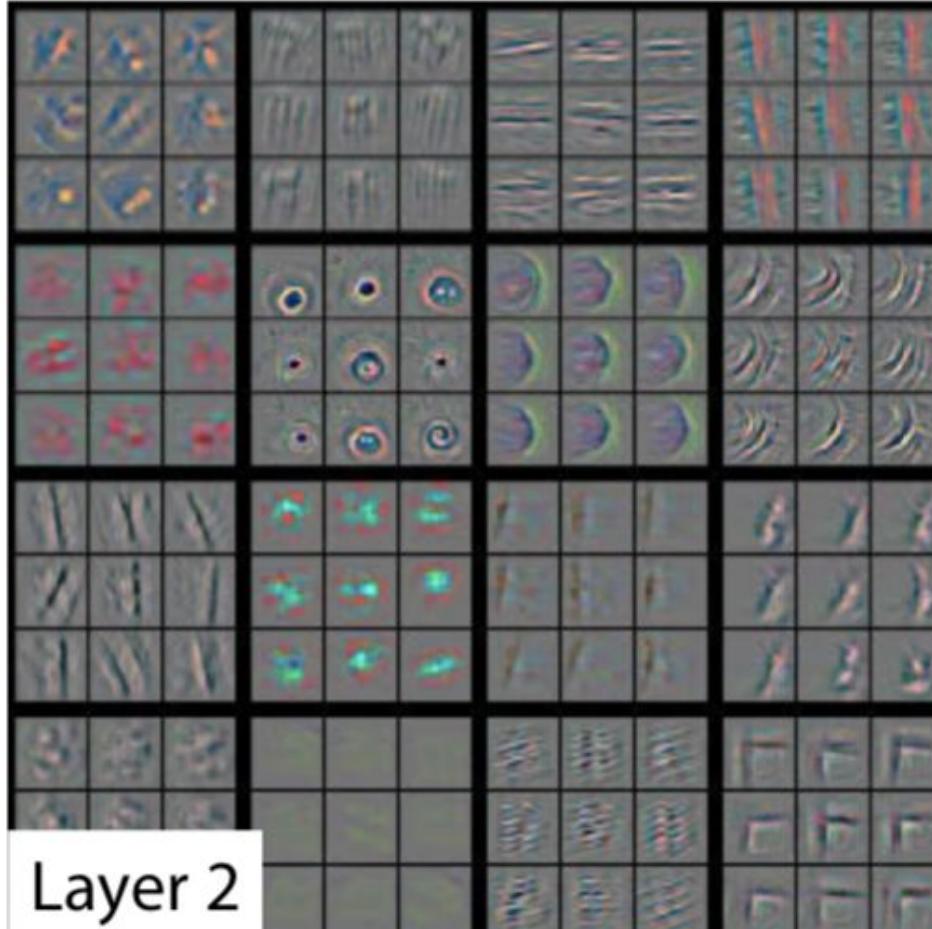
Basicamente = AlexNet, mas:

- **CONV1:** trocou filtros (11x11 stride 4) para (7x7 stride 2)
- **# de filtros/neurônios:** 48-128-192-192-128-2048-2048 (ImageNet) x 2
- **ImageNet top 5 error:** 16.4% -> 11.7%
- **Permitiu** a visualização das features

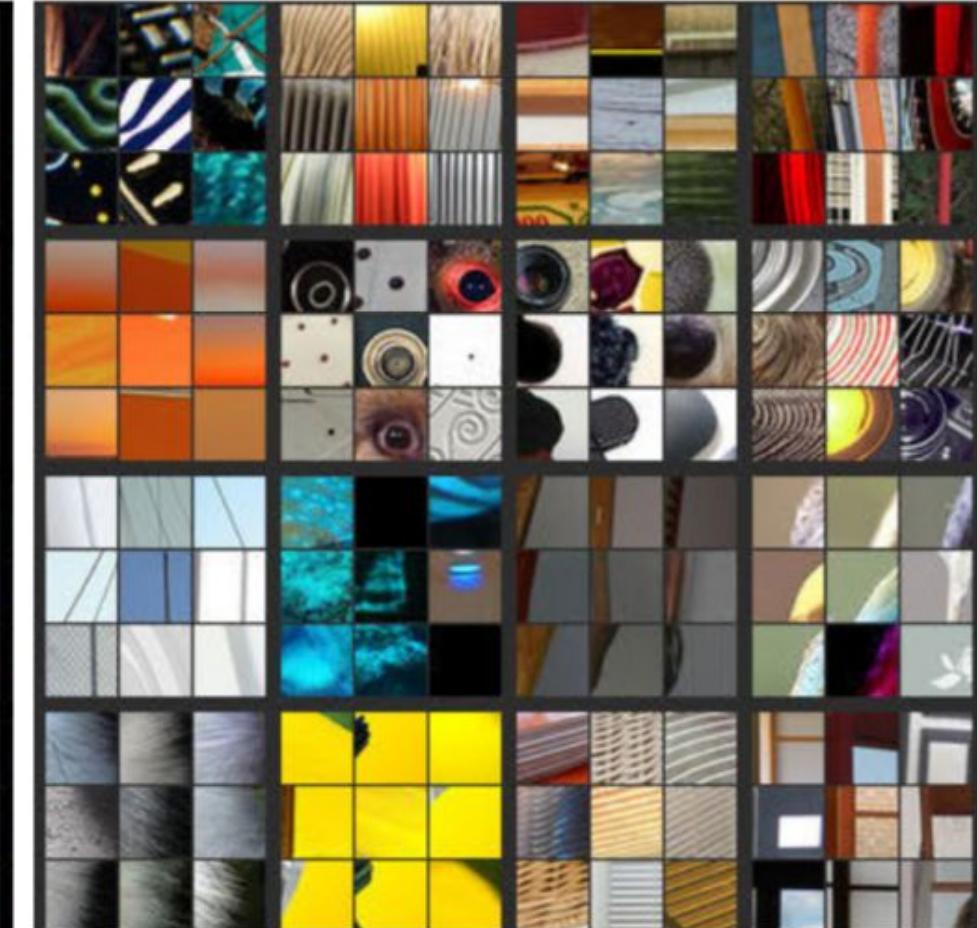
ZFNet



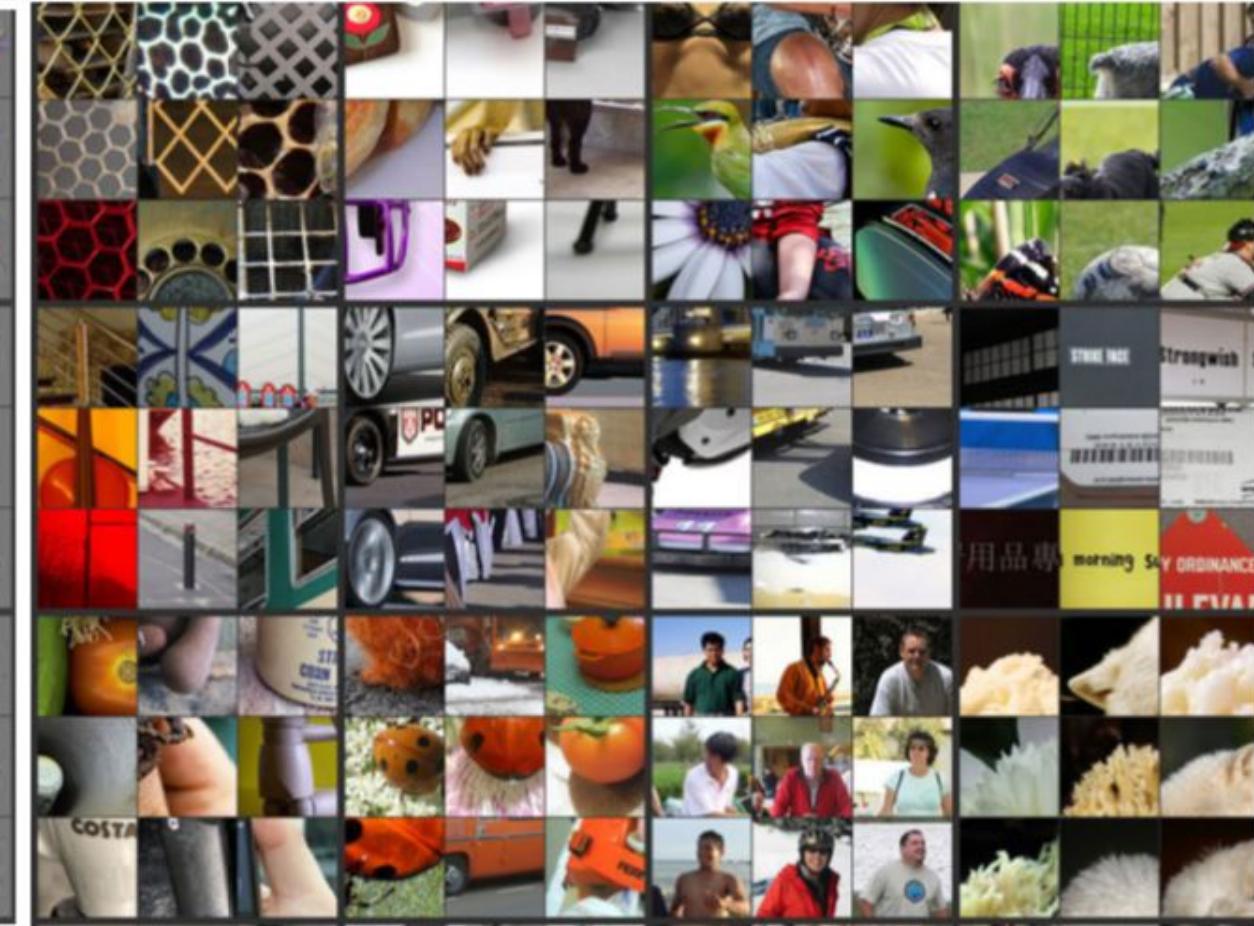
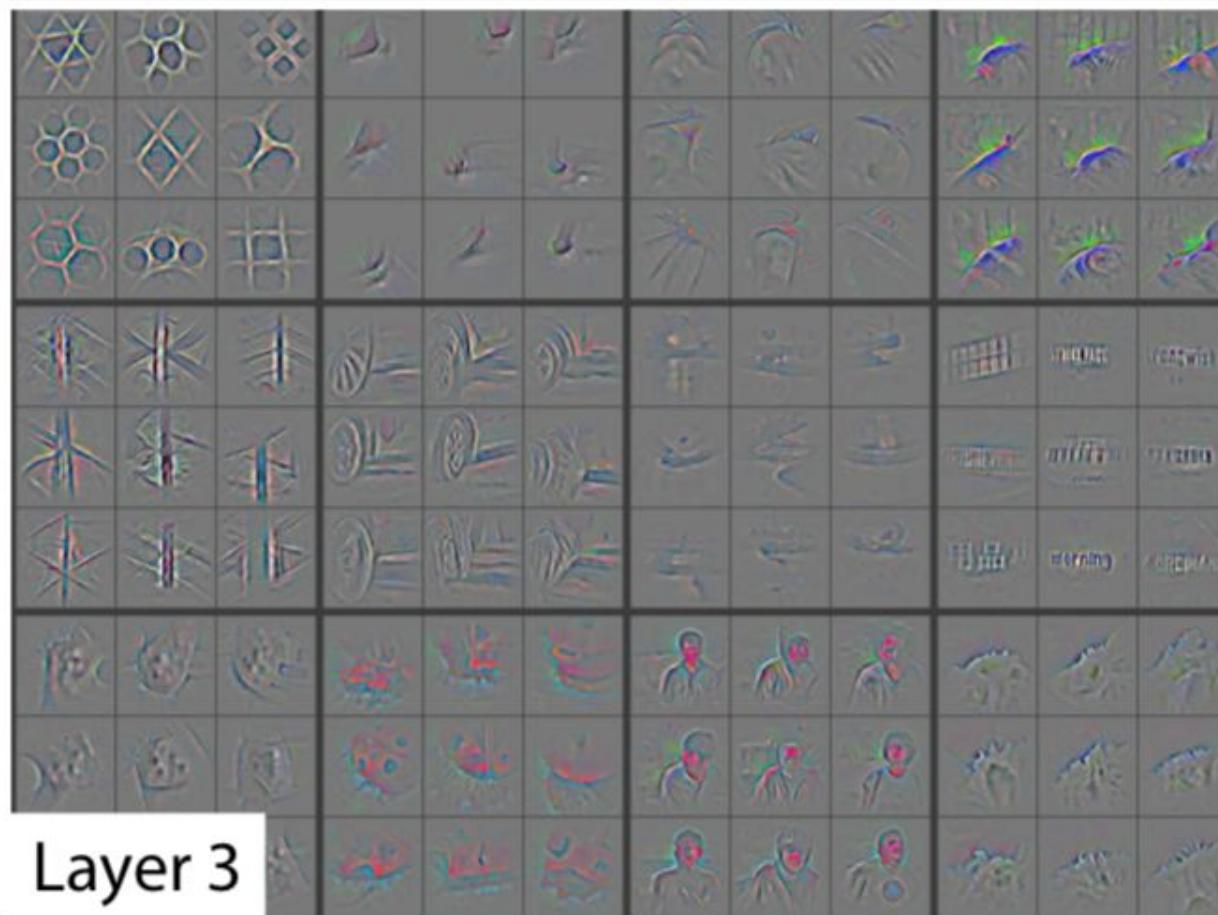
Layer 1

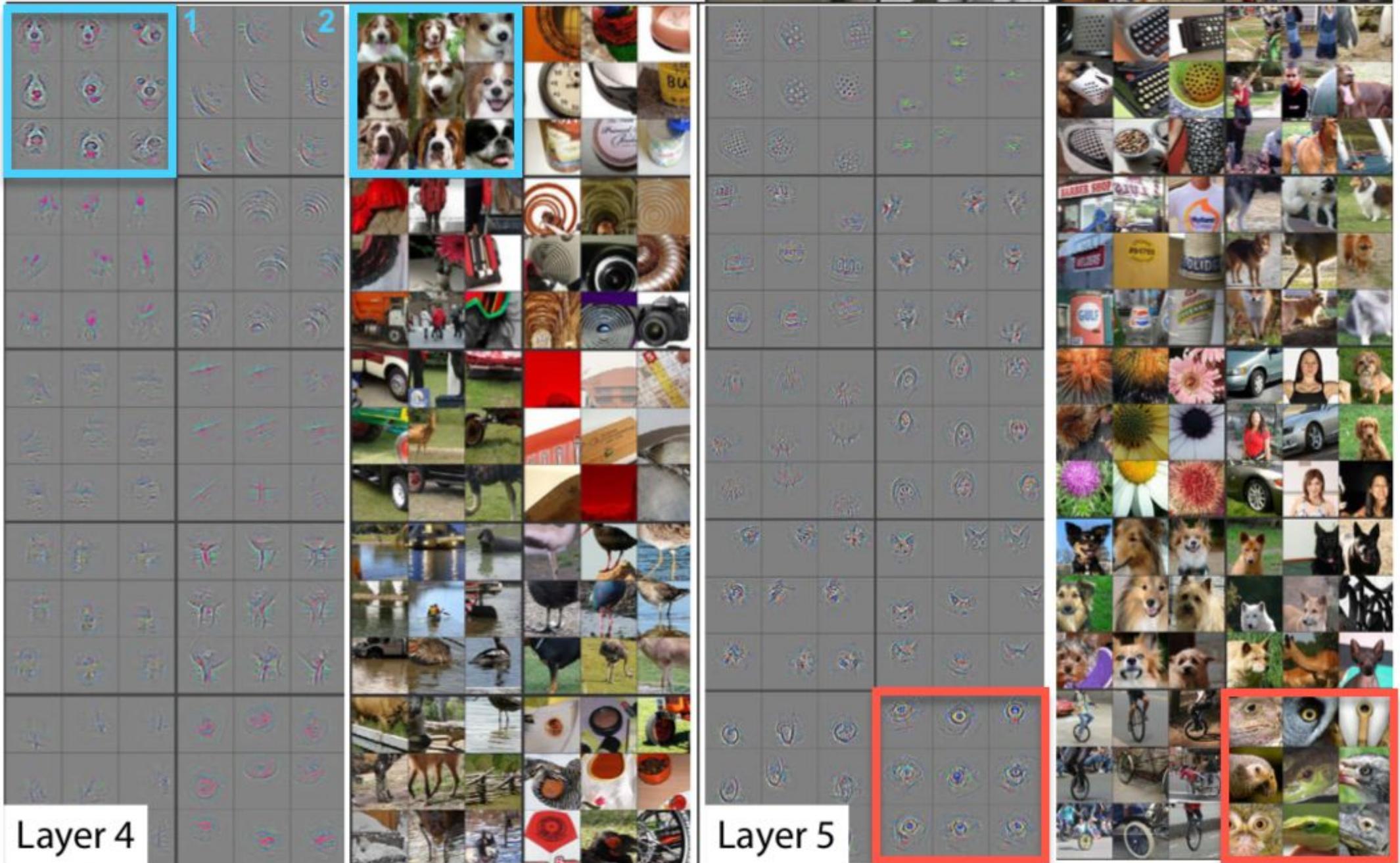


Layer 2

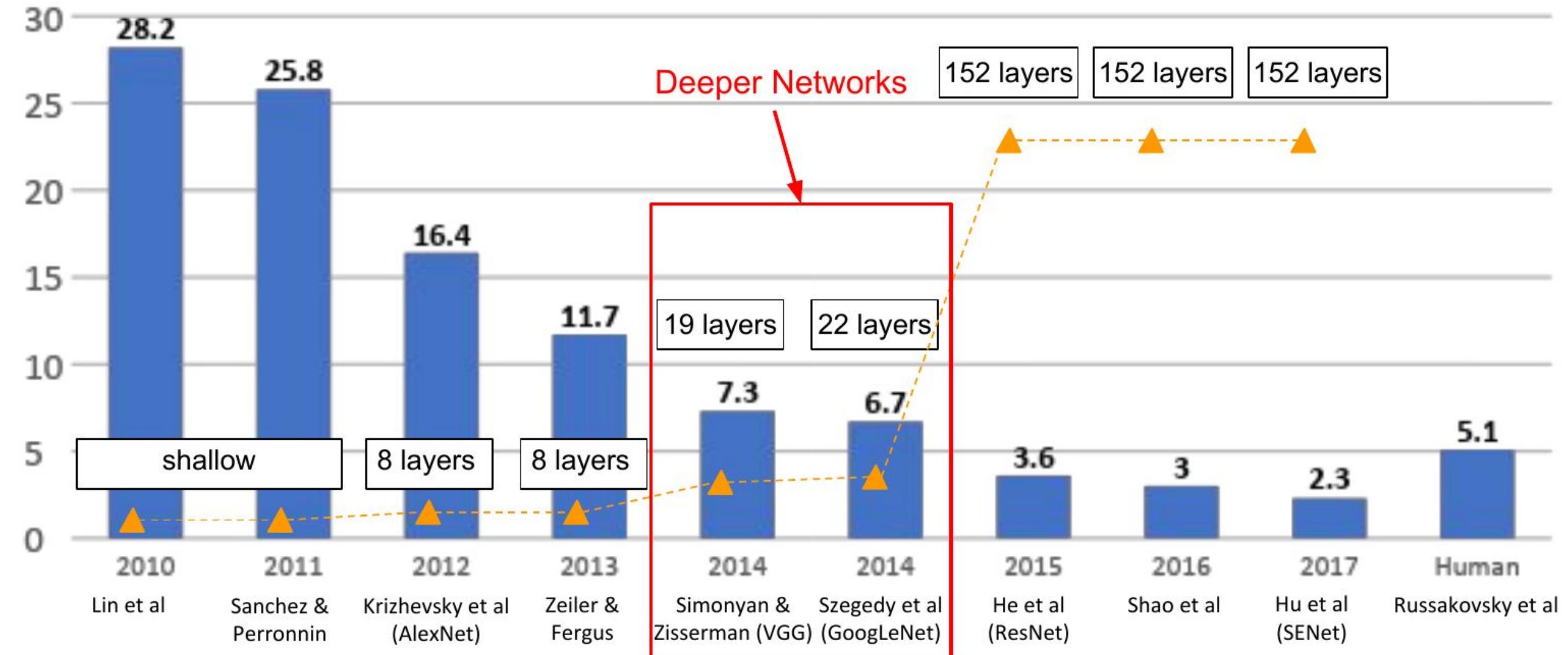


ZFNet





ImageNet Challenge (vencedores)

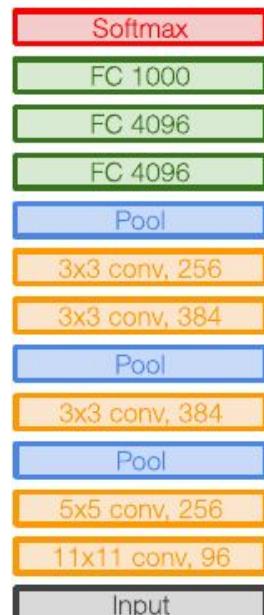


VGG - 16

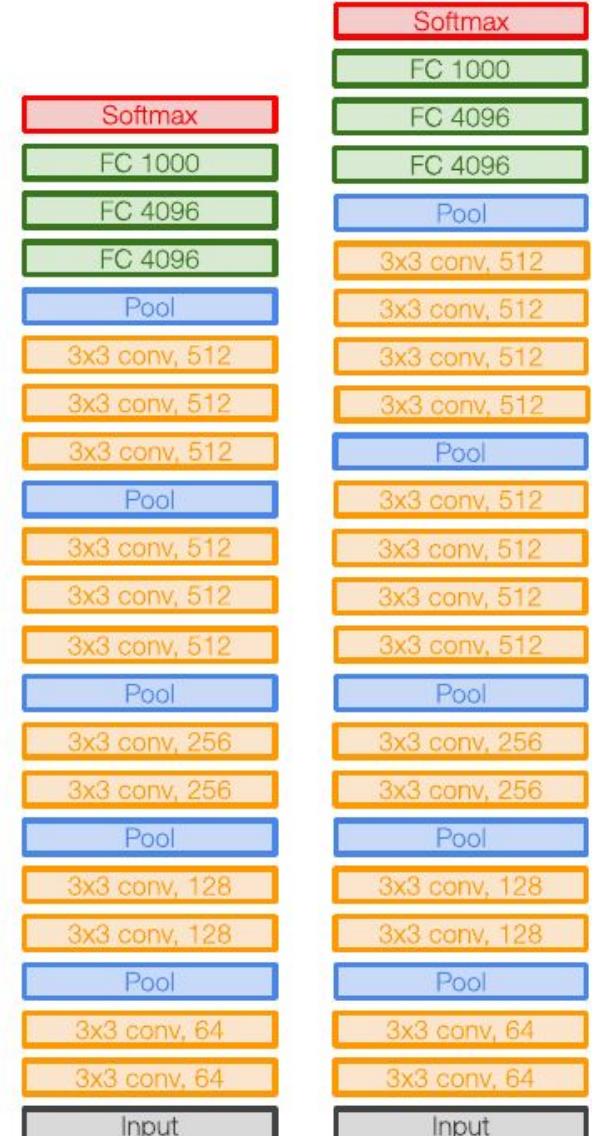
- Uma **modificação** da AlexNet
- Em vez de ter muitos hiperparâmetros, temos uma **rede mais simples**
- Concentra-se em ter apenas os seguintes blocos:
 - **CONV** = filtro 3×3 , stride = 1, “same”
 - **MAX-POOL** = 2×2 , stride = 2
- [\[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition\]](#)

VGG - 16

- Filtros menores, redes mais profundas
- AlexNet: 8 camadas
- VGG: 16 a 19 camadas
- **ImageNet top 5 error:**
 - de 11.7% (ZFNet) para 7.3%



AlexNet



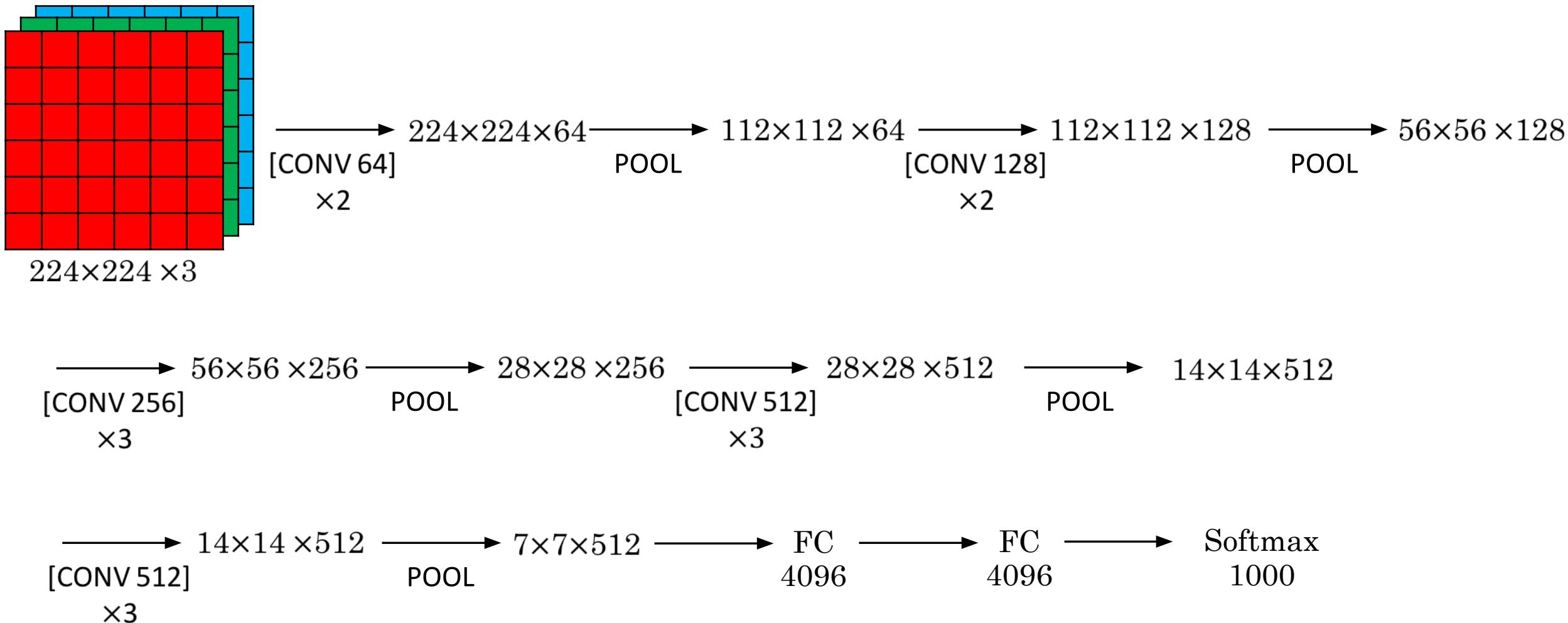
VGG16

VGG19

VGG - 16

CONV = 3×3 filter, s = 1, same

MAX-POOL = 2×2 , s = 2



VGG - 16

- Essa rede é **grande** até mesmo pelos padrões modernos
 - Tem cerca de **138 milhões** de parâmetros
 - A maioria dos parâmetros está nas **camadas totalmente conectadas**
 - Tem uma memória total de **96MB** por imagem apenas para propagação direta!
 - A maior parte da memória está nas **camadas anteriores**
- O **número de filtros** aumenta de 64 para 128 para 256 para 512 (2x)
- ***Pooling*** é o único responsável por encolher as dimensões
- Há outra versão, menos usada, chamada **VGG-19**
- O papel VGG é atraente, ela tenta estipular algumas **regras** sobre o uso de CNNs

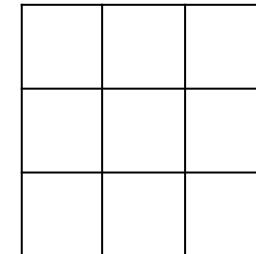
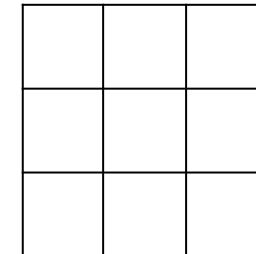
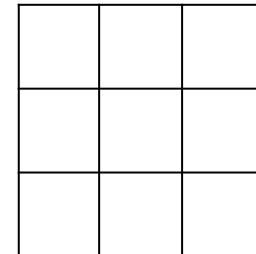
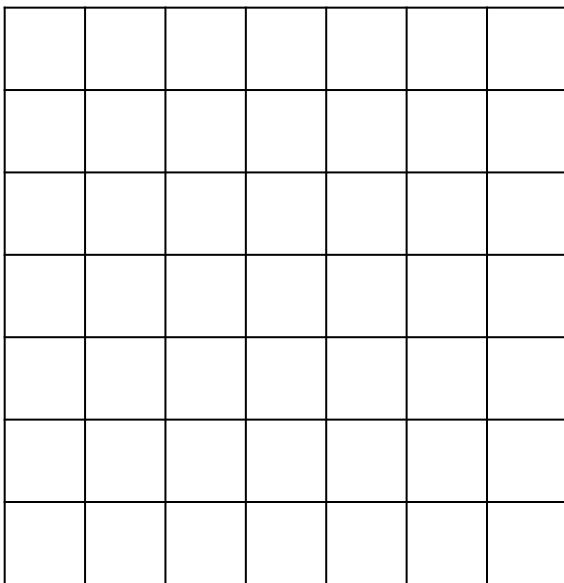
VGGNet

Q: Por que usar filtros menores? (3x3 conv)

A: Empilhar três filtros 3x3 conv (*stride 1*) tem o mesmo **campo receptivo efetivo** (*effective receptive field*) que uma camada conv com filtro 7x7

VGGNet

A: Empilhar três filtros 3x3 conv (*stride 1*) tem o mesmo **campo receptivo efetivo** (*effective receptive field*) que uma camada conv com filtro 7x7



VGGNet

Q: Por que usar filtros menores? (3x3 conv)

A: Empilhar três filtros 3x3 conv (*stride 1*) tem o mesmo **campo receptivo efetivo** (*effective receptive field*) que uma camada conv com filtro 7x7

Vantagens?

VGGNet

Q: Por que usar filtros menores? (3x3 conv)

A: Empilhar três filtros 3x3 conv (*stride 1*) tem o mesmo **campo receptivo efetivo** (*effective receptive field*) que uma camada conv com filtro 7x7

Vantagens:

- Mais profundo, mais não-linearidades
- Menos parâmetros

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150\text{K}$ params: 0 (not counting biases)

Softmax

FC 1000

FC 4096

FC 4096

Pool

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool

3x3 conv, 256

3x3 conv, 256

Pool

3x3 conv, 128

3x3 conv, 128

Pool

3x3 conv, 64

3x3 conv, 64

Input

VGG16

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800\text{K}$ params: 0

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400\text{K}$ params: 0

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200\text{K}$ params: 0

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: 0

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25\text{K}$ params: 0

FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150\text{K}$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800\text{K}$ params: 0

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400\text{K}$ params: 0

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200\text{K}$ params: 0

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: 0

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

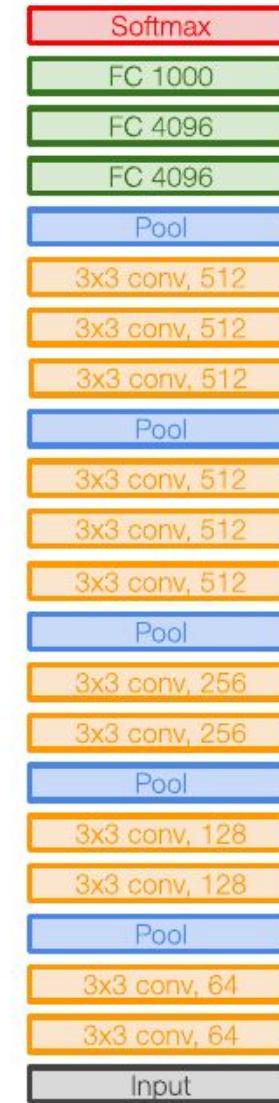
CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25\text{K}$ params: 0

FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$



VGG16

Memória total: 24M * 4bytes = 96MB / imagem (forward pass)

Número total de parâmetros: 138M

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150\text{K}$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800\text{K}$ params: 0

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400\text{K}$ params: 0

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200\text{K}$ params: 0

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: 0

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25\text{K}$ params: 0

FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

Maior gasto de memória está nas primeiras CONVs

A maioria dos parâmetros está nas FCs

Memória total: $24\text{M} * 4\text{bytes} = 96\text{MB} / \text{imagem}$ (forward pass)

Número total de parâmetros: 138M

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
 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

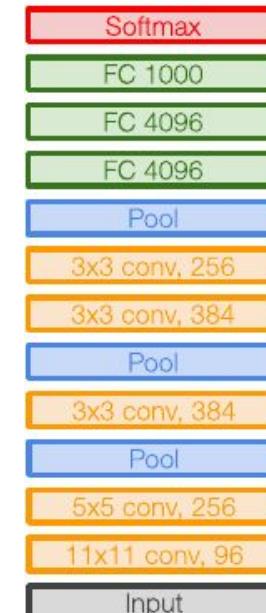
Nomes comuns

Memória total: 24M * 4bytes = 96MB / imagem (forward pass)

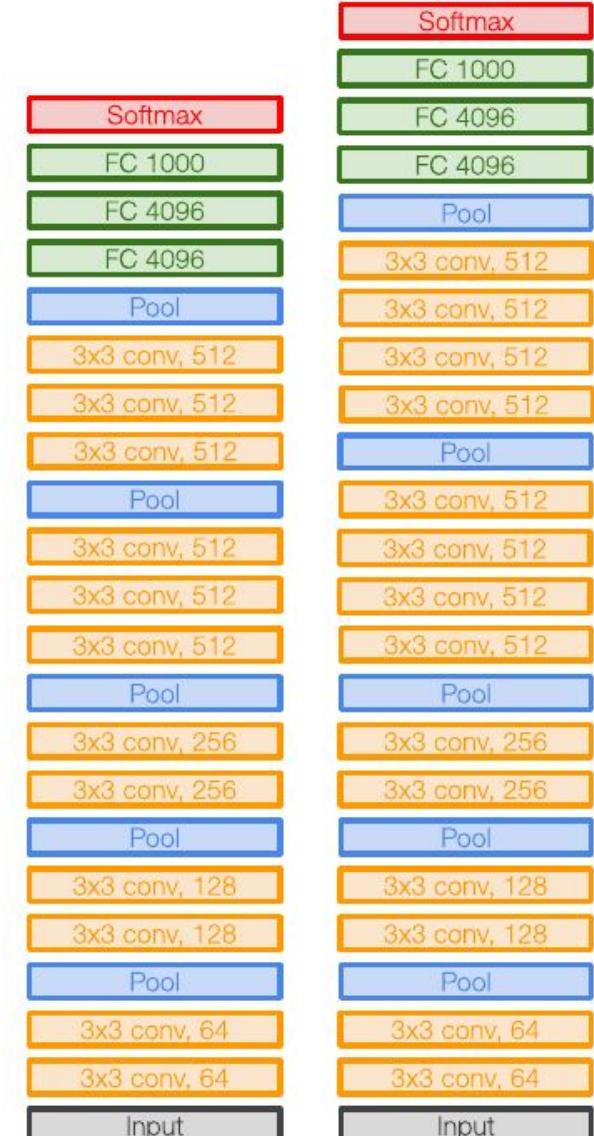
Número total de parâmetros: 138M

VGGNet

- ILSVRC'14: 2º na classificação, 1º na localização
- Procedimento de treinamento similar a [Krizhevsky 2012]
- Sem *Local Response Normalization*
- Use VGG16 ou VGG19 (a segunda apenas um pouco melhor, mas mais memória)
- Use *ensembles* para melhores resultados
- Features FC7 generalizam bem para outras tarefas



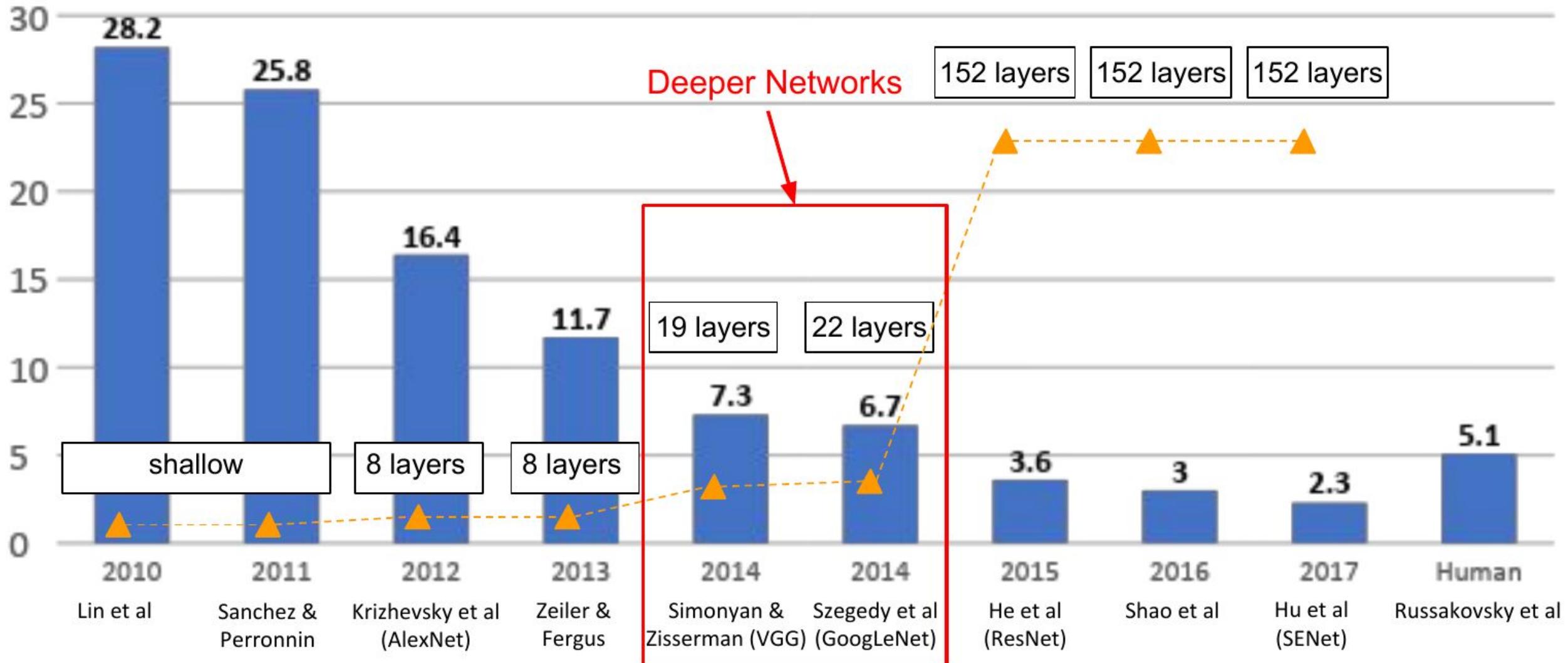
AlexNet



VGG16

VGG19

ImageNet Challenge (vencedores)



Estudos de Caso

Redes na Redes e
Convoluções 1x1

O que faz uma convolução 1x1?

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

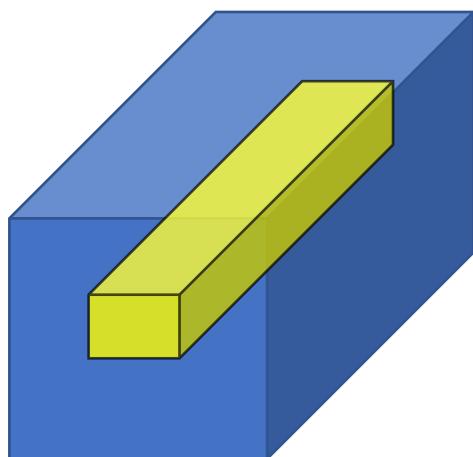
*

2

=

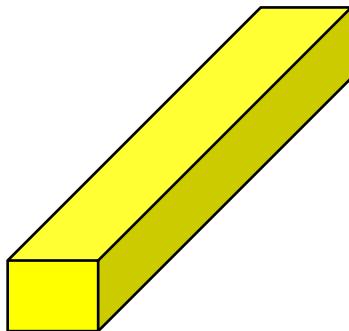
blue	blue	blue	blue	blue
light blue				
white	white	white	white	white
light blue				
white	white	white	white	white

6×6



$6 \times 6 \times 32$

*



$1 \times 1 \times 32$

=

blue	blue	blue	blue	blue
light blue				
white	white	white	white	white
light blue				
white	white	white	white	white

$6 \times 6 \times \# \text{ filters}$

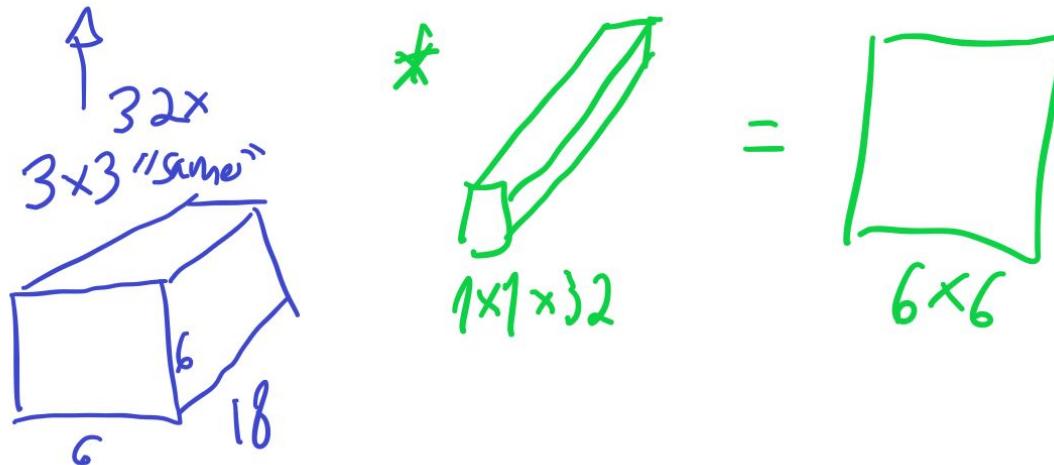
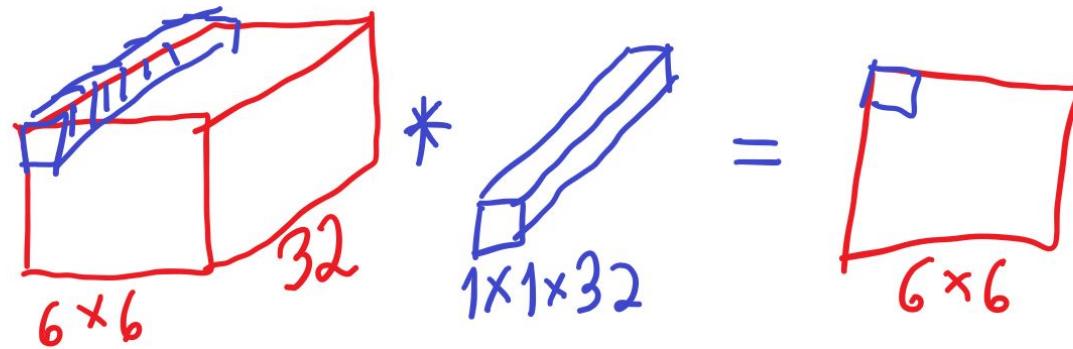
O que faz uma convolução 1x1?

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

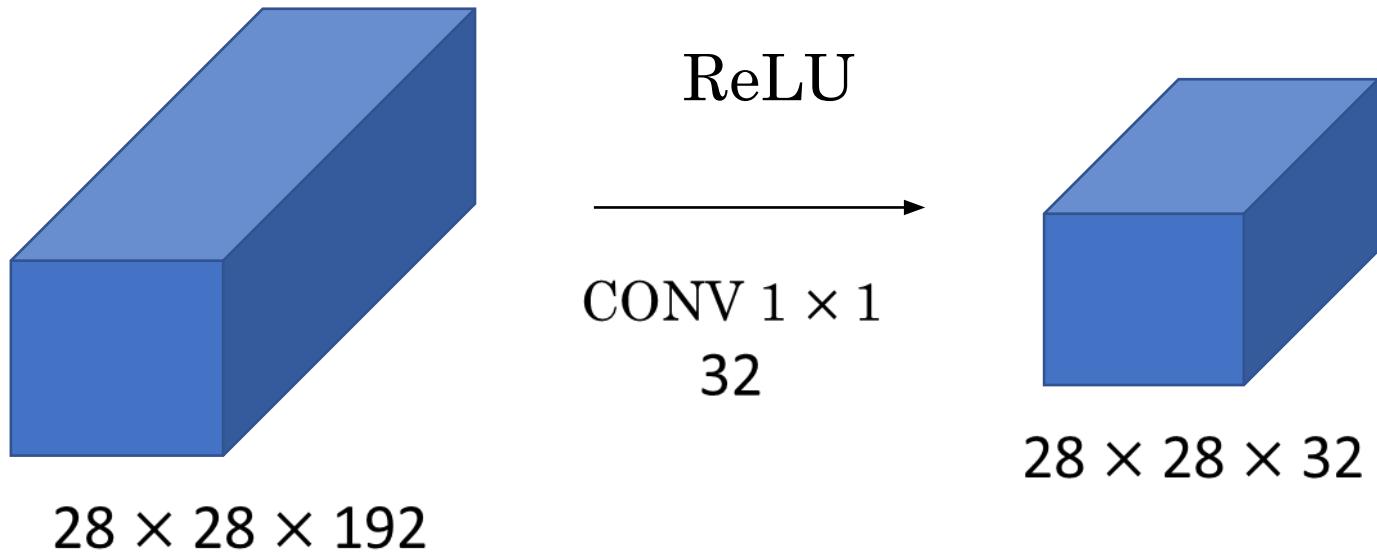
6×6

$$\ast \quad \boxed{2} \quad =$$

Blue	Blue	Blue	Blue	Blue	Blue
Light Blue					
Light Blue					
Light Blue					
Light Blue					



Usando convoluções 1x1



Convolução 1x1

- Uma convolução de 1 x 1 - também chamada de **Rede na Rede (*Network in Network*)** - é muito útil em muitos modelos de CNN
- O que faz uma convolução de 1 X 1?
 - Não é apenas multiplicar por um número?
- [\[Lin et al., 2013. Network in network\]](#)

Convolução 1x1

- Exemplo 1:
 - Entrada: 6x6x1
 - Conv: 1x1x1 um filtro. # A conv 1 x 1
 - Saída: 6x6x1
- Exemplo 2:
 - Entrada: 6x6x32
 - Conv: 1x1x32, 5 filtros. # A conv 1 x 1
 - Saída: 6x6x5

Convolução 1x1

- Tem sido usado em muitas implementações modernas da CNN, como os modelos ResNet e Inception
- Uma convolução de 1×1 é útil quando:
 - Queremos diminuir o número de canais (*feature transformation*)
 - No segundo exemplo anterior, reduzimos a entrada de 32 para 5 canais
 - Mais tarde, veremos que, ao diminuí-lo, podemos economizar muitos cálculos
 - Se tivermos especificado o número de filtros 1×1 Conv para ser o mesmo que o número de entrada de canais, a saída conterá o mesmo número de canais
 - Então o Conv 1×1 atuará como uma não linearidade e aprenderá o operador de não linearidade.

Convolução 1x1: dica final

- Substitua as camadas totalmente conectadas por convoluções 1 x 1, uma vez que Yann LeCun acredita que elas são iguais

“Em Convolutional Nets, não há ‘camadas totalmente conectadas’. Existem apenas camadas de convolução com núcleos de convolução 1x1 e uma tabela de conexões completa.” Yann LeCun