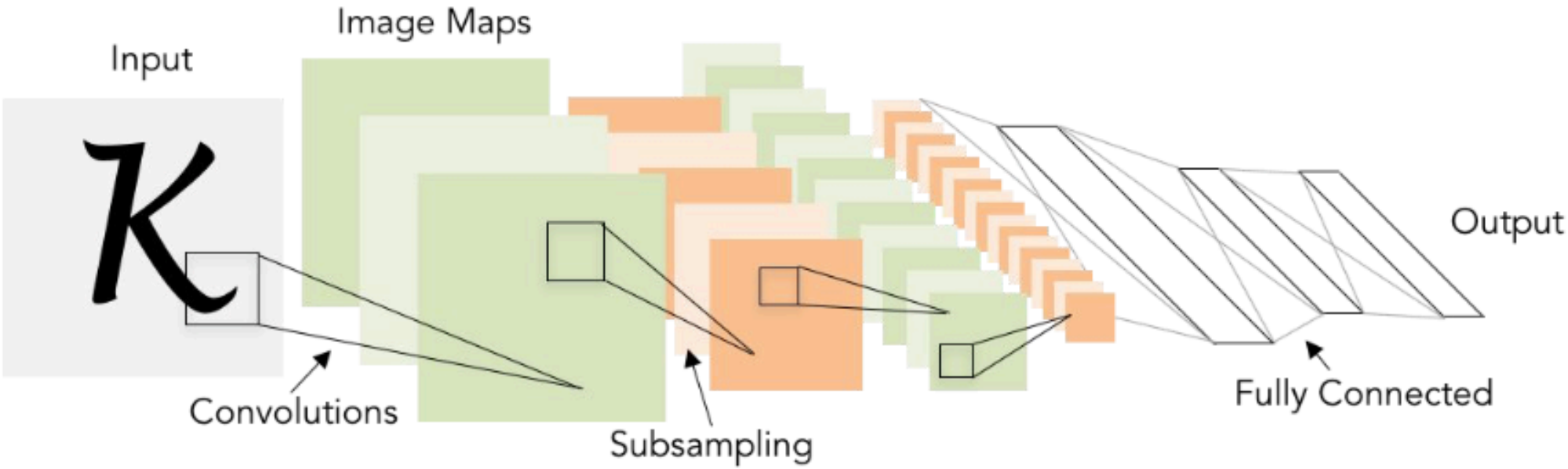
FILIPE ROLIM CORDEIRO (FILIPE.ROLIM@GMAIL.COM)

Imagenet Challenge

LeNet-5

- [LeCun et al., 1998]



- Filtros convolutivos 5x5, aplicados com stride 1
- Camadas de pooling com filtro 2x2 aplicado com stride 2
- [CONV-POOL-CONV-POOL-FC-FC]

LeNet5 – Especificações

MNIST - 60,000 training, 10,000 testing

Input is 32x32 image

8 layers

60,000 parameters

Few hours to train on a laptop

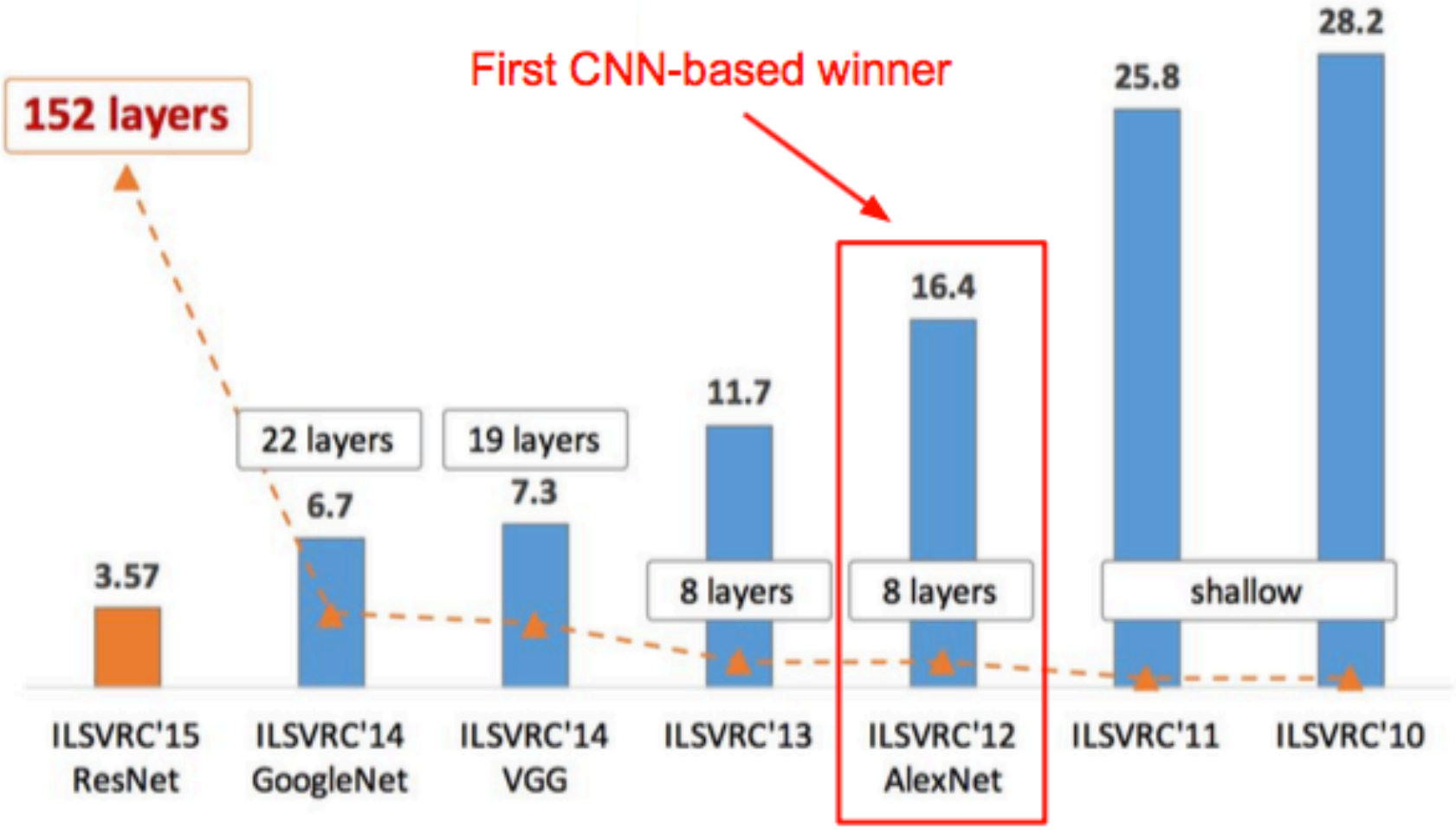
Implementando LeNet

- Código
  - Lenet.ipynb

Layer Type	Output Size	Filter Size / Stride
INPUT IMAGE	$28 \times 28 \times 1$	
CONV	$28 \times 28 \times 20$	$5 \times 5, K = 20$
ACT	$28 \times 28 \times 20$	
POOL	$14 \times 14 \times 20$	2×2
CONV	$14 \times 14 \times 50$	$5 \times 5, K = 50$
ACT	$14 \times 14 \times 50$	
POOL	$7 \times 7 \times 50$	2×2
FC	500	
ACT	500	
FC	10	
SOFTMAX	10	

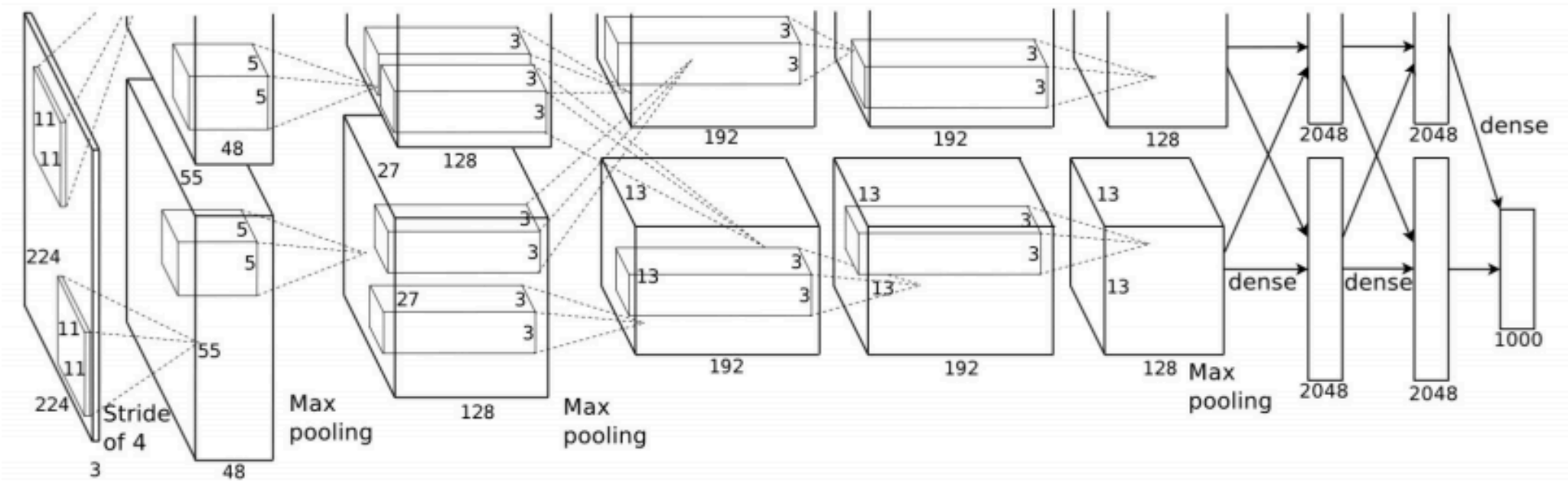
Arquiteturas

- Vencedores da ILSVRC (ImageNet Large Scale Visual Recognition Challenge)



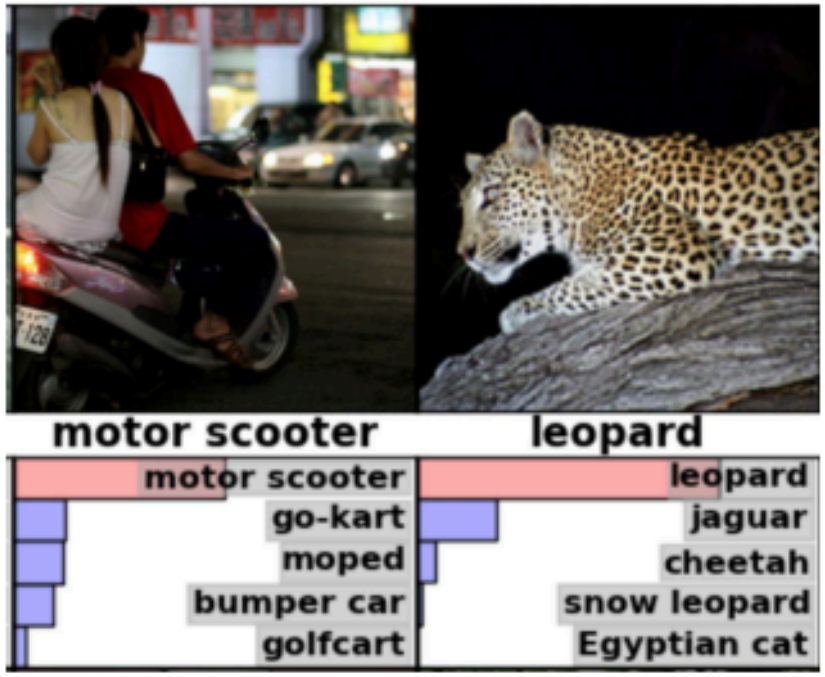
AlexNet (2012)

- Venceu a ILSVRC
- Alcançou top-5 error de 15.4%, contra anterior de 26.2%



AlexNet- Especificações

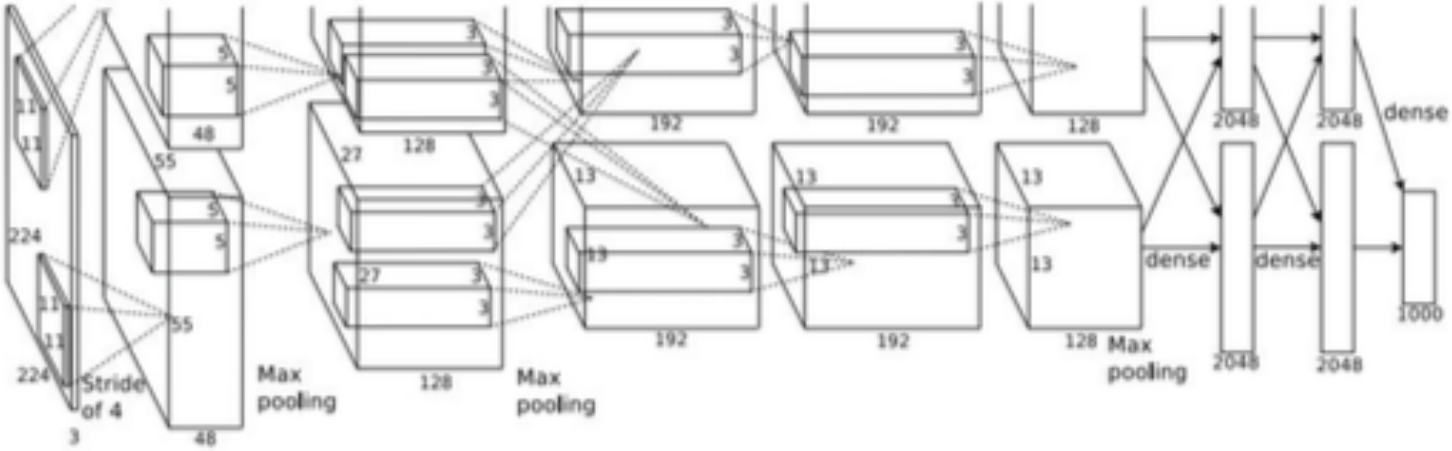
- 1000 categorias do ImageNet
- 1.2 milhões de imagens de treino
- 50.000 imagens de validação
- 150.000 imagens de teste



- 60M Parâmetros
- Treinado em duas GPUs GTX entre 5 e 6 dias

AlexNet

[Krizhevsky et al. 2012]

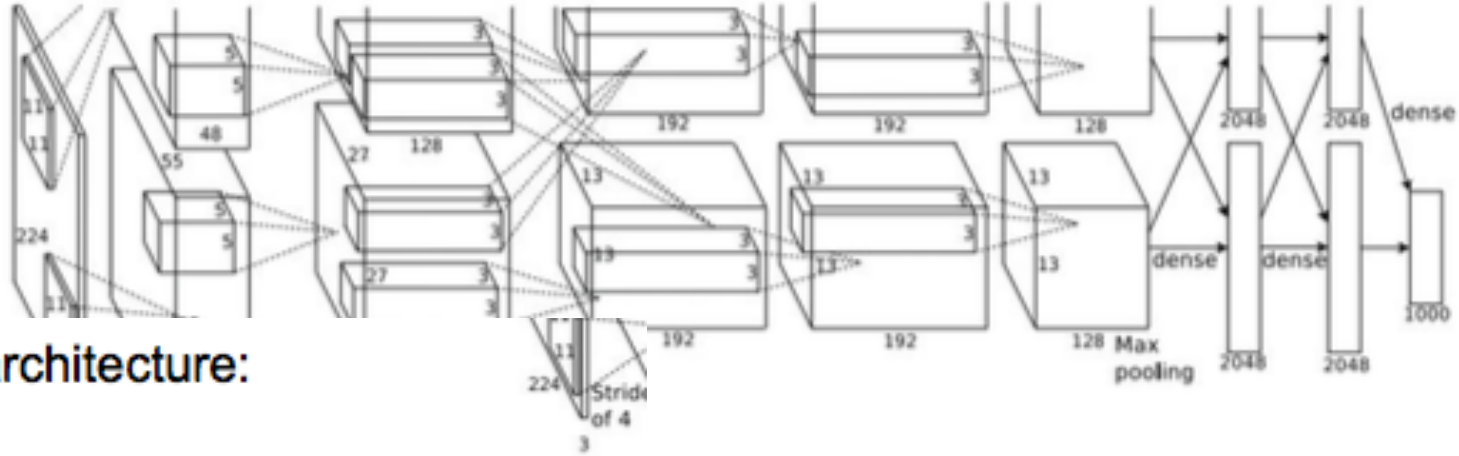


Architecture:

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

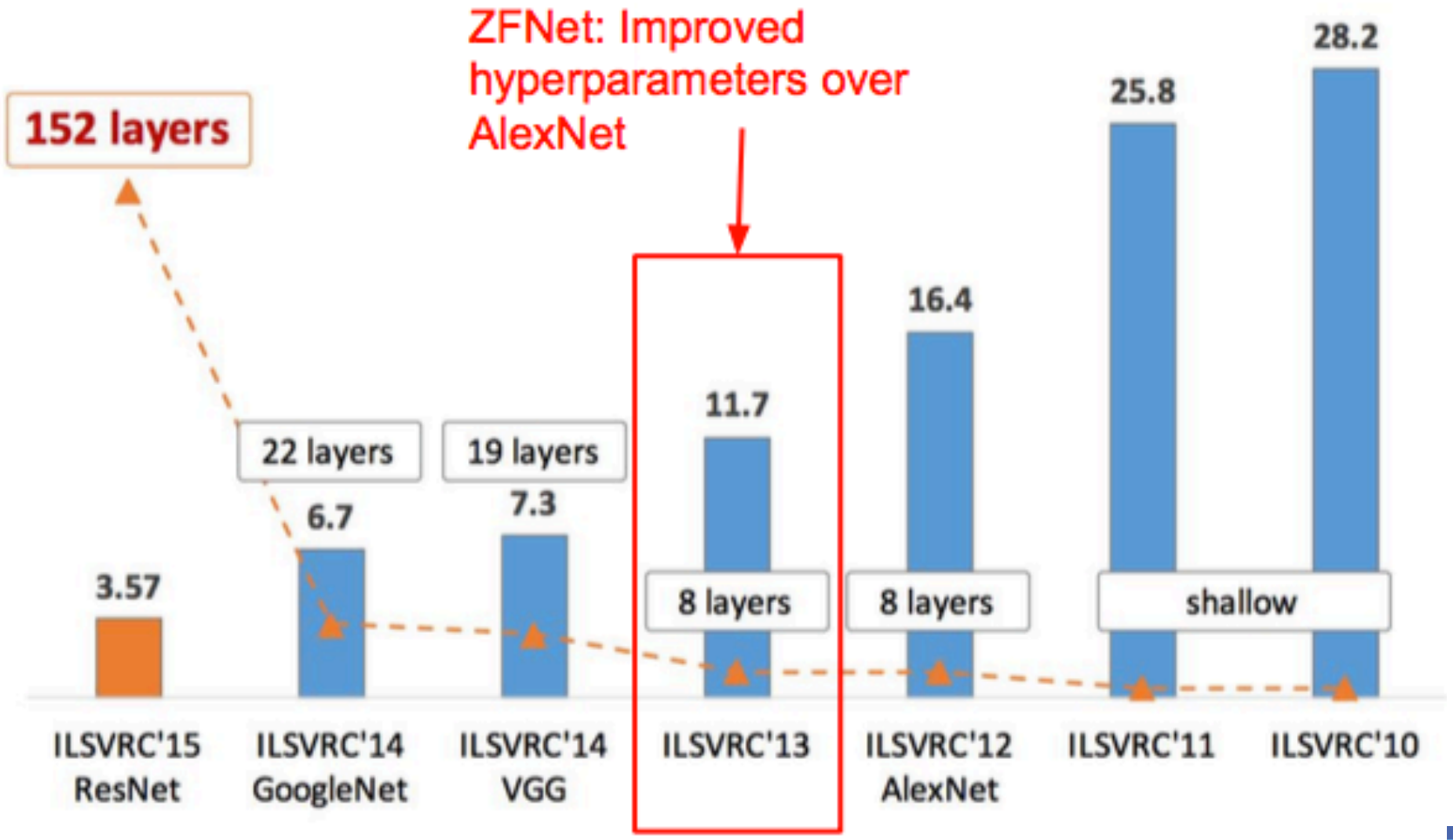
AlexNet

[Krizhevsky et al. 2012]



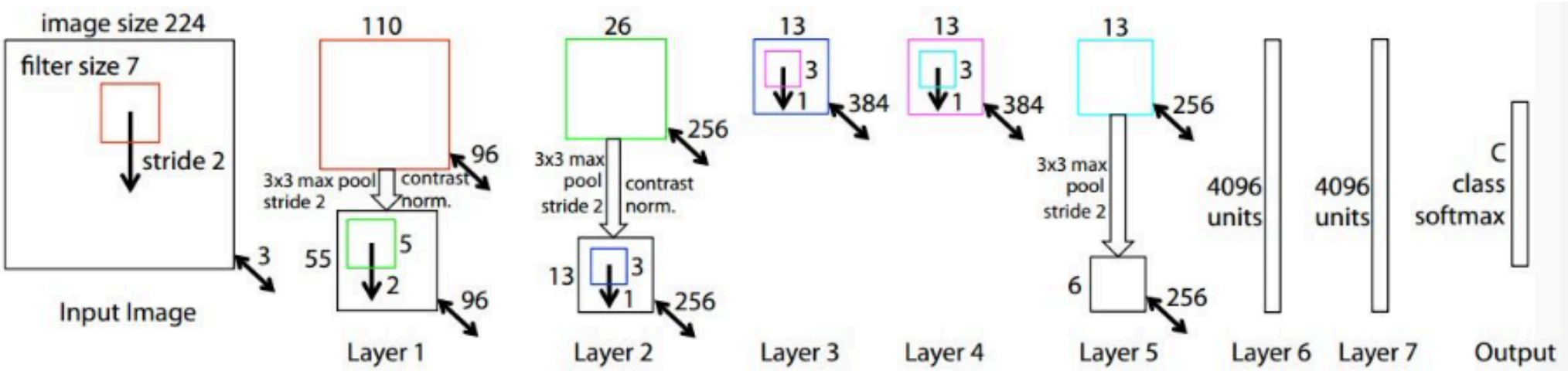
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

- Vencedores da ILSVRC (ImageNet Large Scale Visual Recognition Challenge)



ZFNET

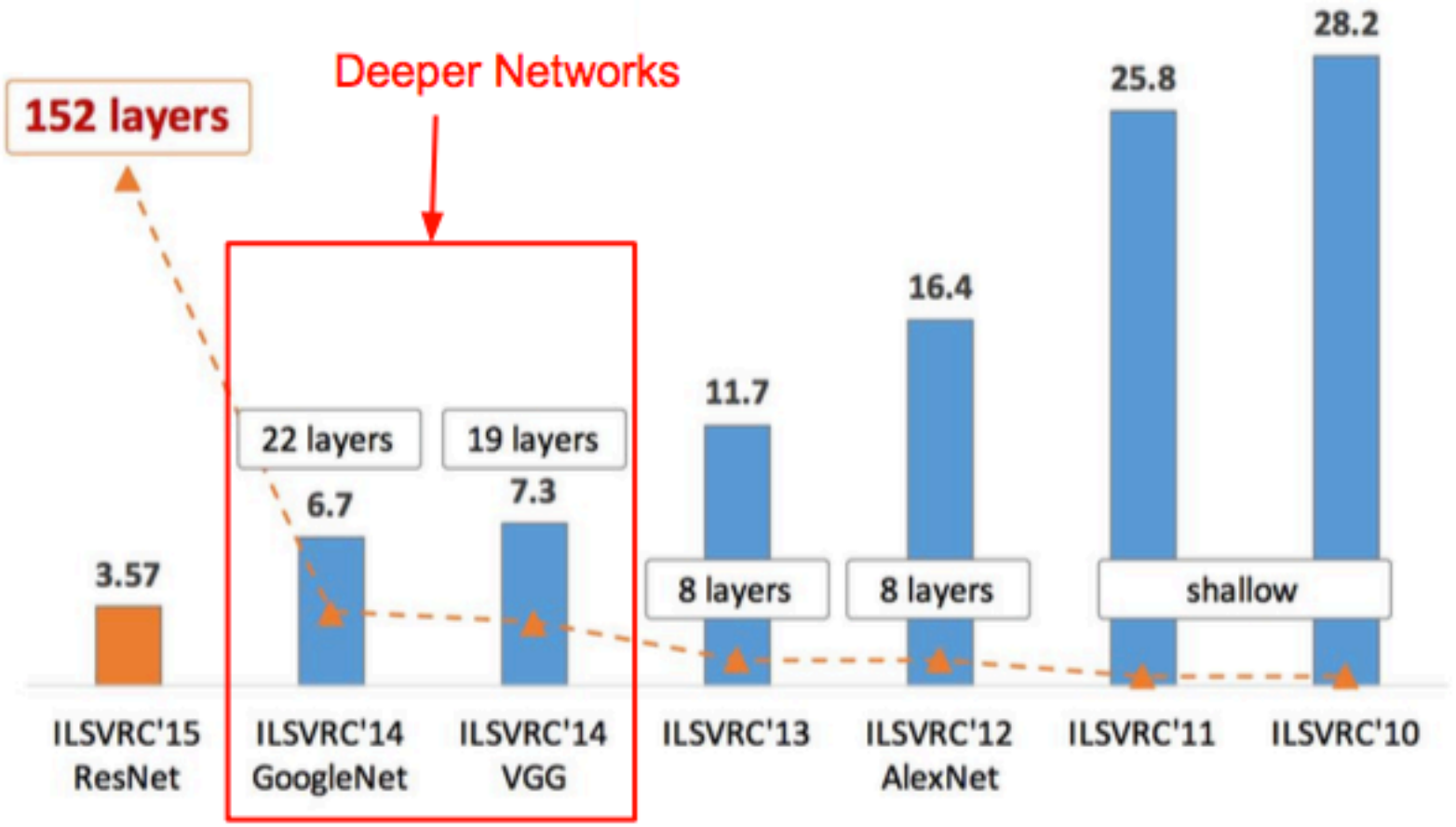
- [Zeiler and Fergus, 2013]



AlexNet but:
CONV1: change from (11x11 stride 4) to (7x7 stride 2)
CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ImageNet top 5 error: 16.4% -> 11.7%

- Vencedores da ILSVRC (ImageNet Large Scale Visual Recognition Challenge)



VGG Net - 2014

"Simple and deep"

Top-5 error rate of 7.3% on ImageNet

16 layer CNN - Best result - Conf. D

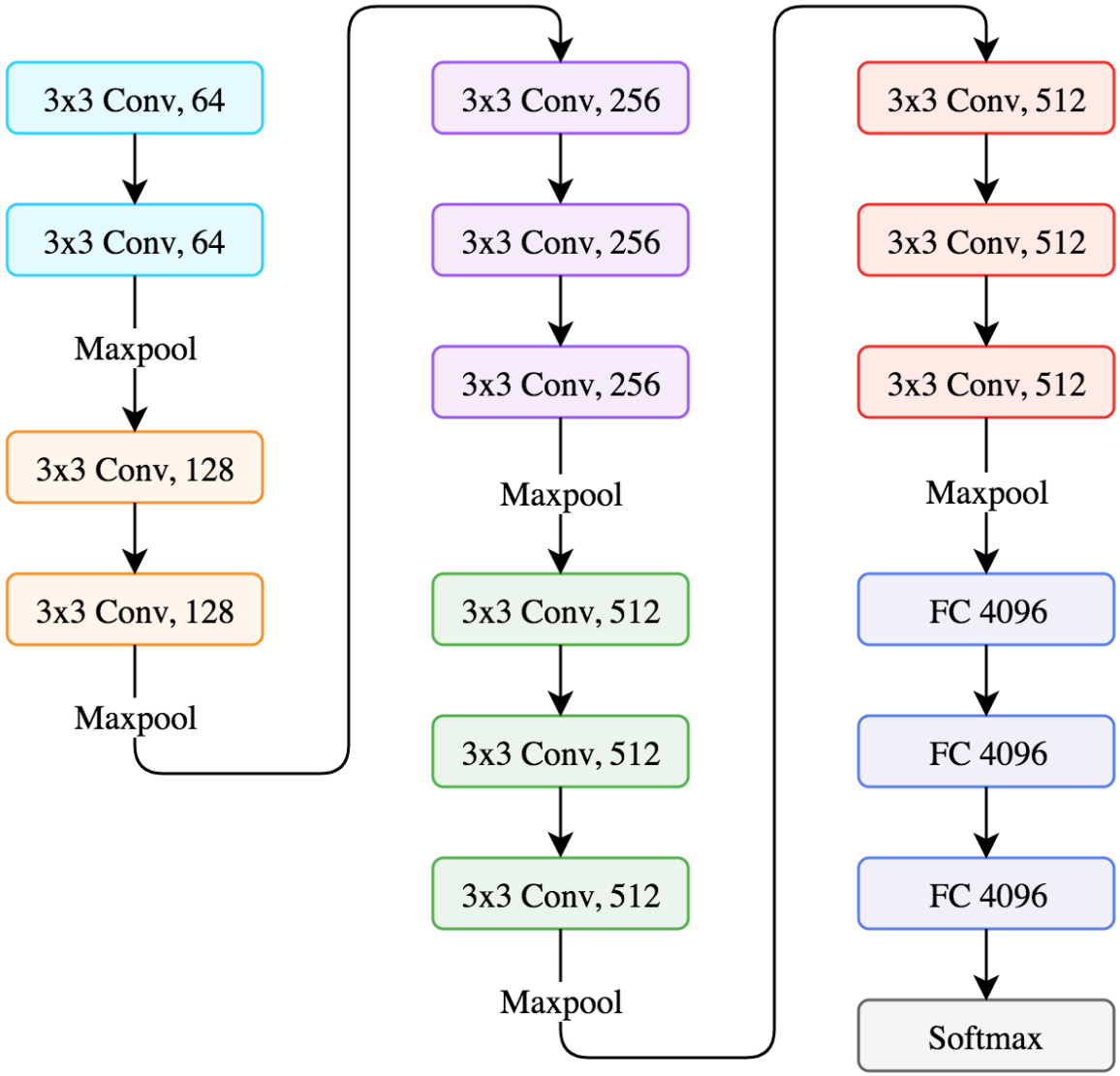
138 M parameters

Trained on 4 Nvidia Titan Black GPUs

for two to three weeks.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG-16

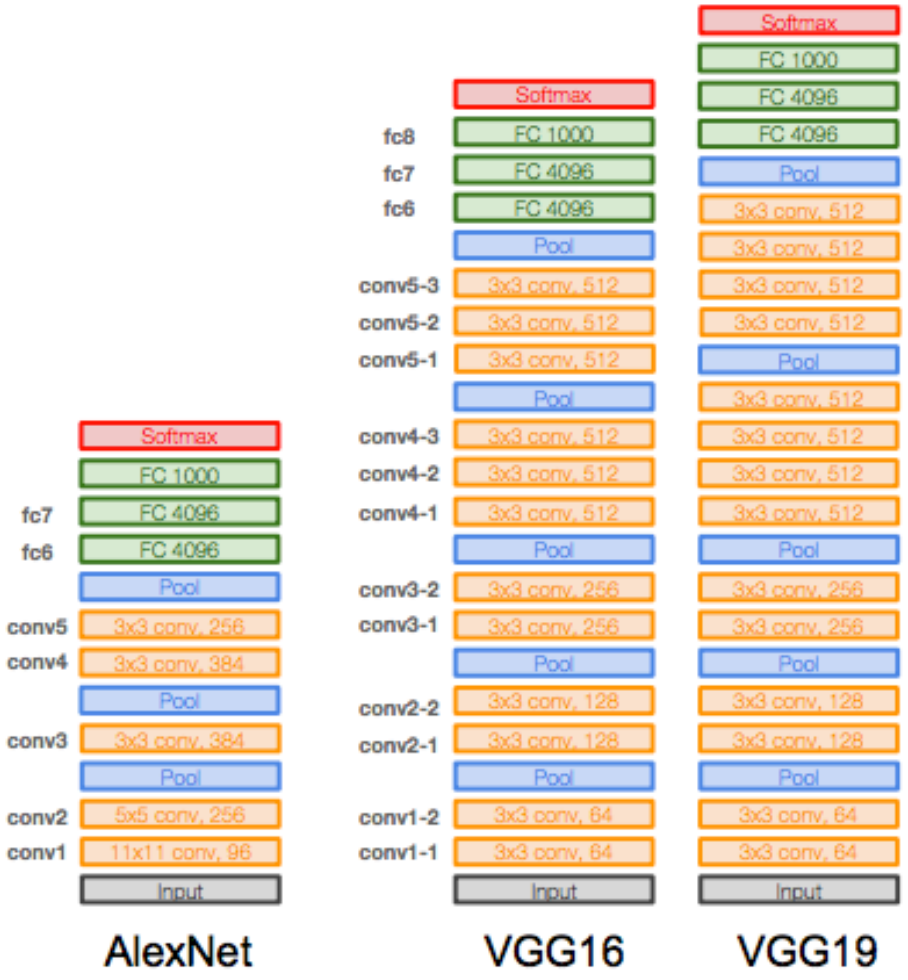

3x3 Conv, 64
3x3 Conv, 64
Maxpool
3x3 Conv, 128
3x3 Conv, 128
Maxpool
3x3 Conv, 256
3x3 Conv, 256
3x3 Conv, 256
Maxpool
3x3 Conv, 512
3x3 Conv, 512
3x3 Conv, 512
Maxpool
3x3 Conv, 512
3x3 Conv, 512
3x3 Conv, 512
Maxpool
FC 4096
FC 4096
FC 4096
Softmax

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

Details:

- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks

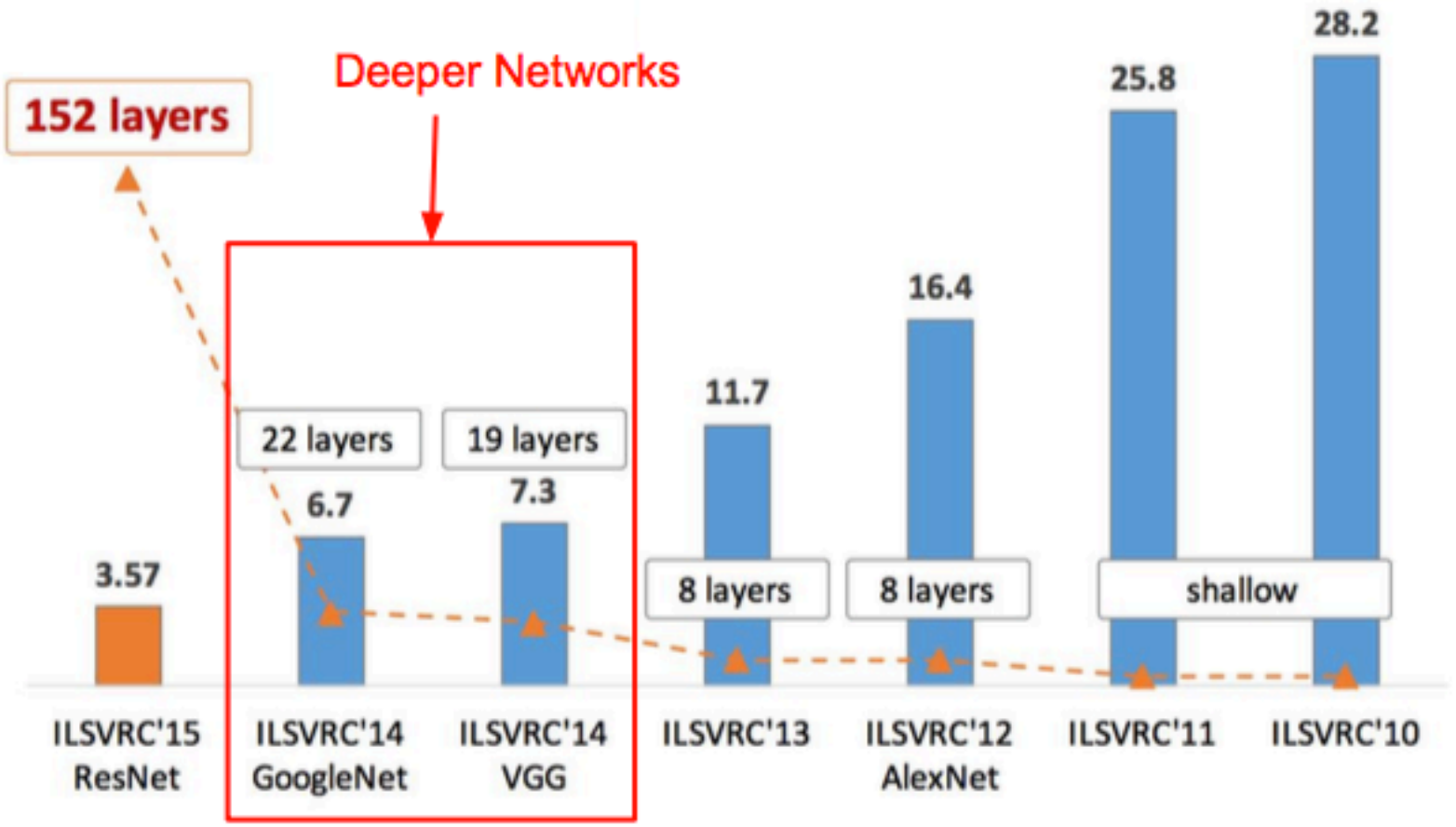


Exercício

- Implementar minivgg
  - Codigo: lenet.ipynb
    - Criar nova função
  - Depois, testar na base cifar

Layer Type	Output Size	Filter Size / Stride
INPUT IMAGE	$32 \times 32 \times 3$	
CONV	$32 \times 32 \times 32$	$3 \times 3, K = 32$
ACT	$32 \times 32 \times 32$	
BN	$32 \times 32 \times 32$	
CONV	$32 \times 32 \times 32$	$3 \times 3, K = 32$
ACT	$32 \times 32 \times 32$	
BN	$32 \times 32 \times 32$	
POOL	$16 \times 16 \times 32$	2×2
DROPOUT	$16 \times 16 \times 32$	
CONV	$16 \times 16 \times 64$	$3 \times 3, K = 64$
ACT	$16 \times 16 \times 64$	
BN	$16 \times 16 \times 64$	
CONV	$16 \times 16 \times 64$	$3 \times 3, K = 64$
ACT	$16 \times 16 \times 64$	
BN	$16 \times 16 \times 64$	
POOL	$8 \times 8 \times 64$	2×2
DROPOUT	$8 \times 8 \times 64$	
FC	512	
ACT	512	
BN	512	
DROPOUT	512	
FC	10	
SOFTMAX	10	

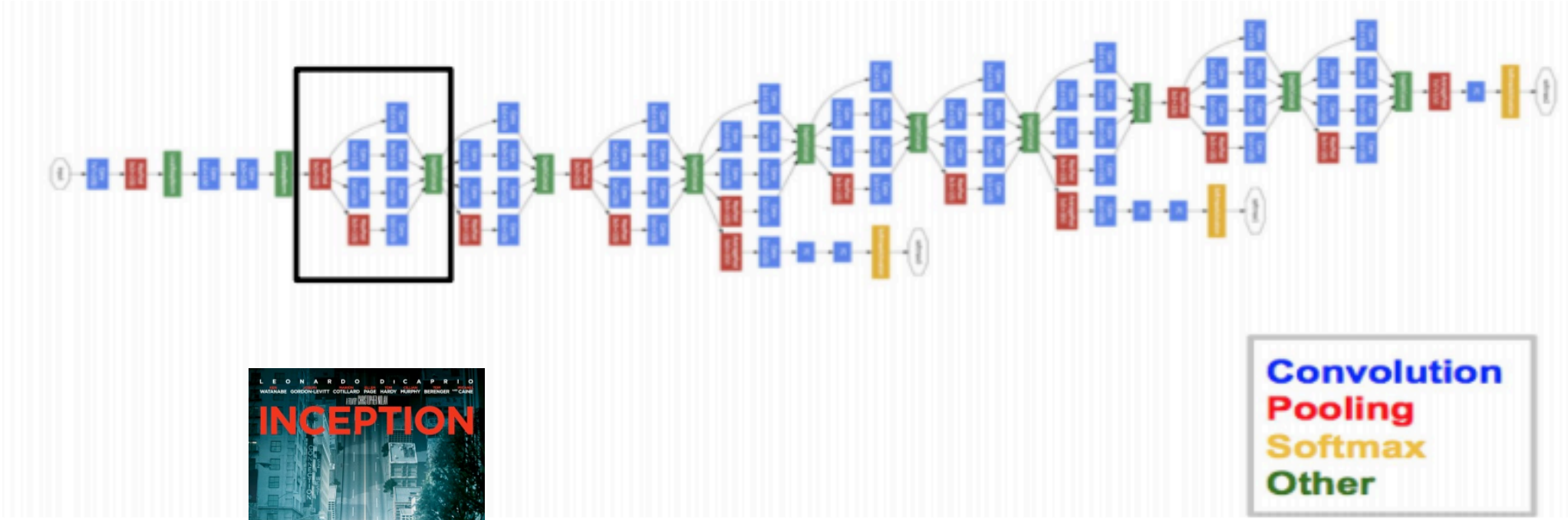
- Vencedores da ILSVRC (ImageNet Large Scale Visual Recognition Challenge)



GoogLeNet / Inception (2014)

- Vencedora da ILSVRC 2014 com erro de 6.7% (4M parâmetros comparado com 60M do AlexNet)
- Treinado em uma semana (algumas GPUs high-end)

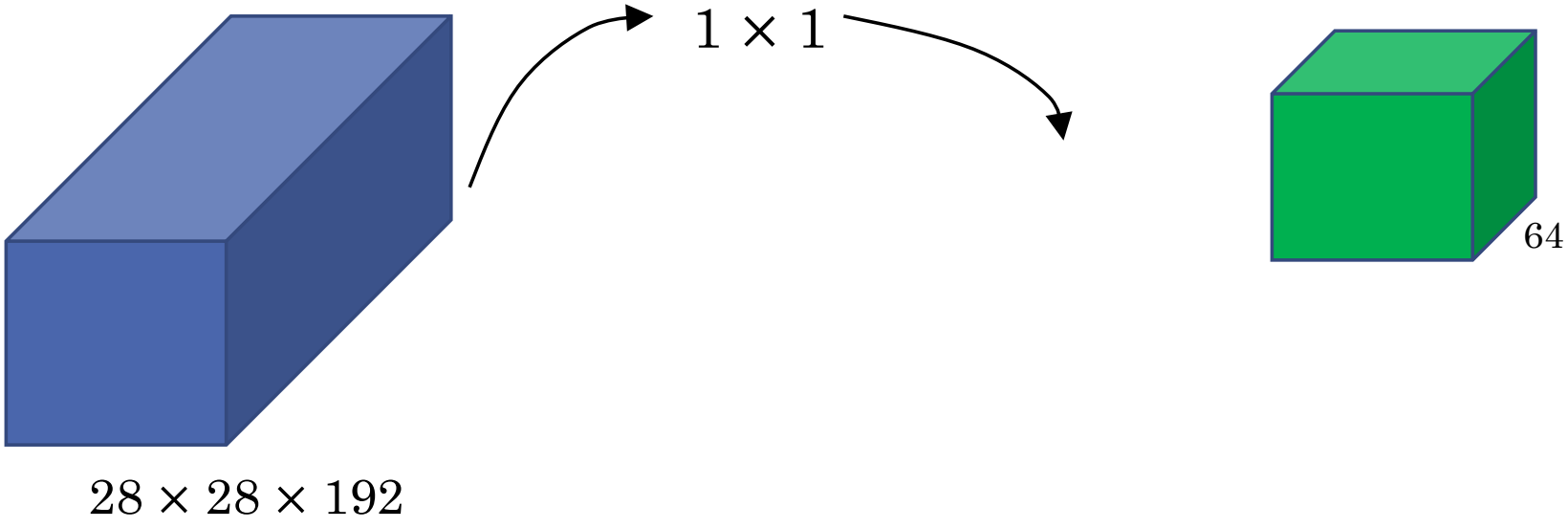
O Módulo Inception



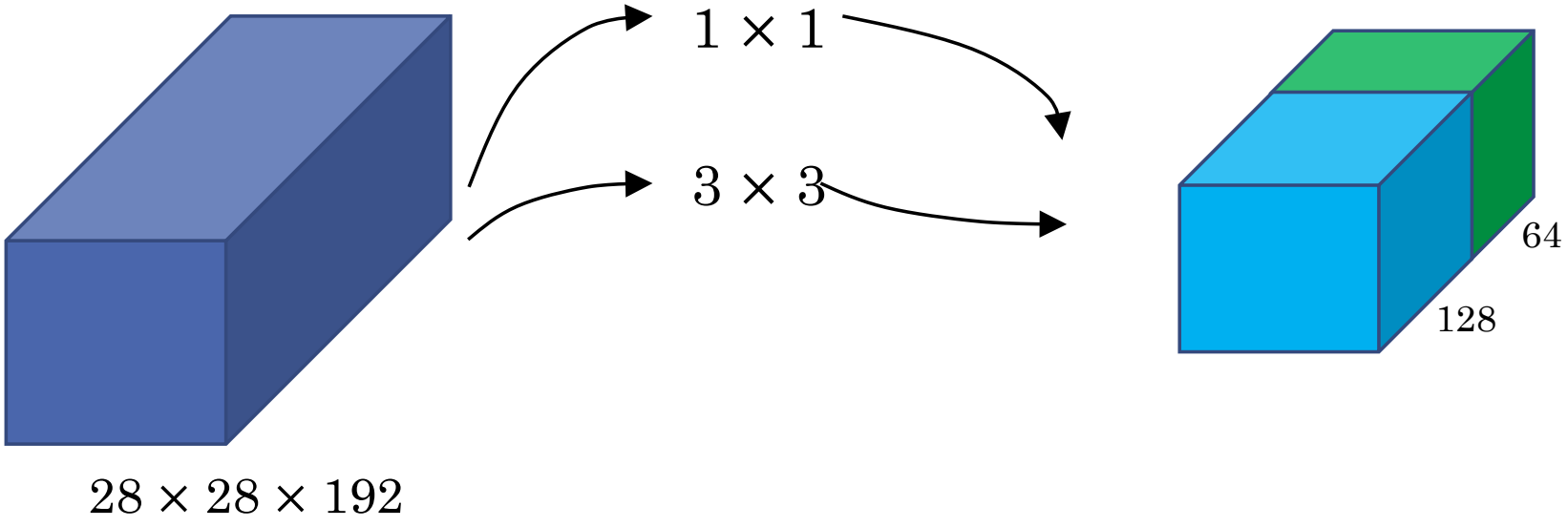
GoogLeNet – Principais ideias

- Usa 9 módulos inceptions
- Não usa camadas FC
  - Usa average pool para transformar 7x7x1024 em 1x1x1024
- Usa 12x menos parâmetros que AlexNet

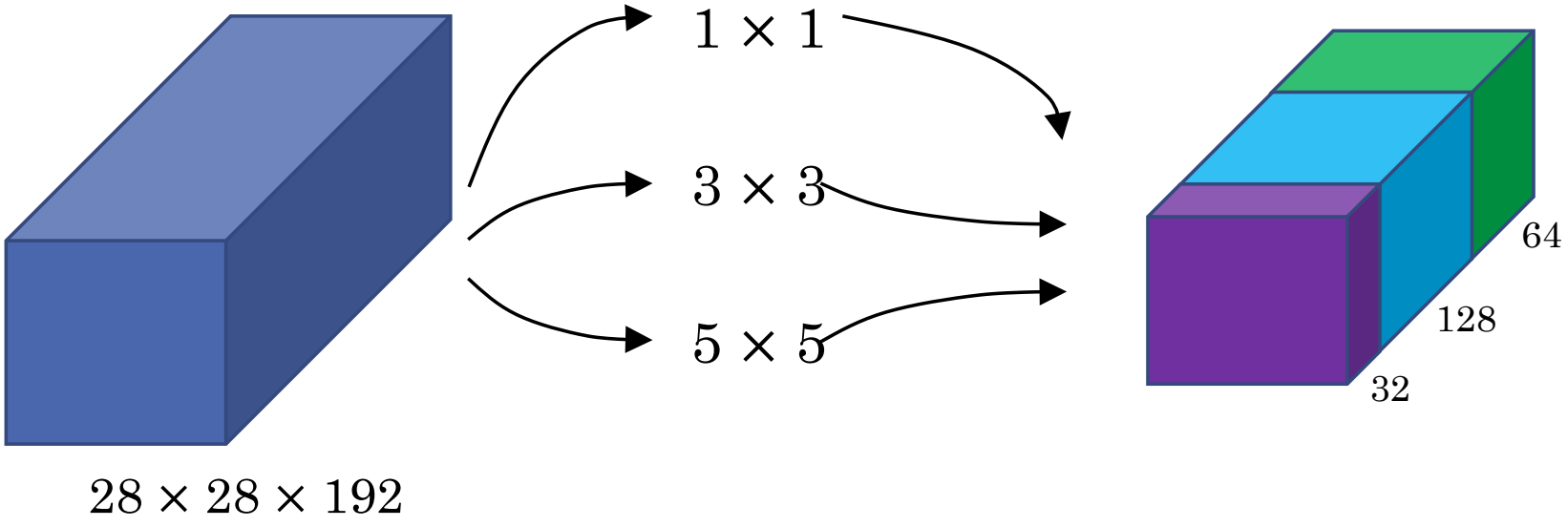
Módulo Inception



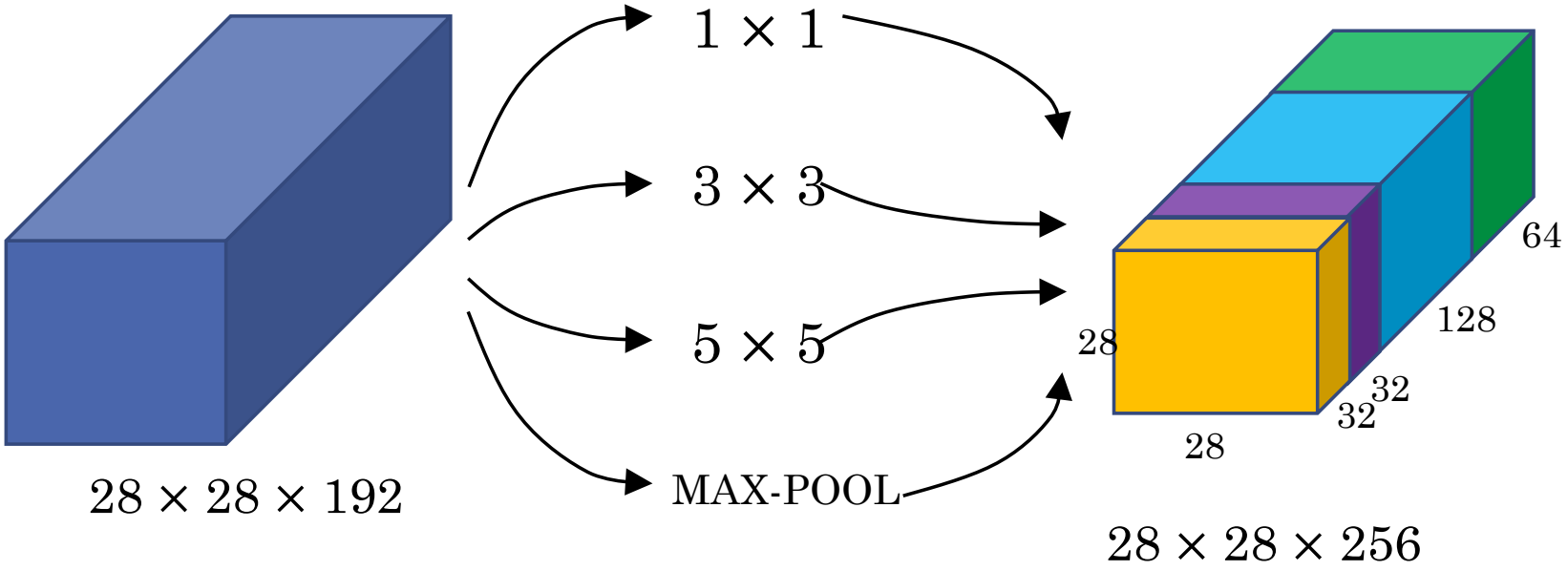
Módulo Inception



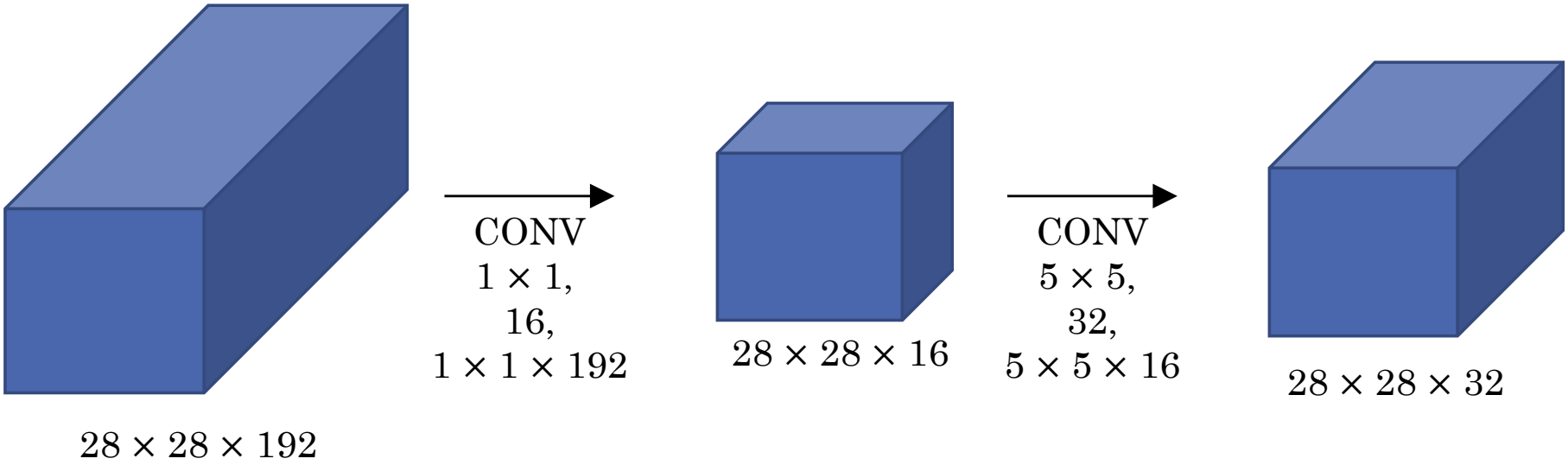
Módulo Inception



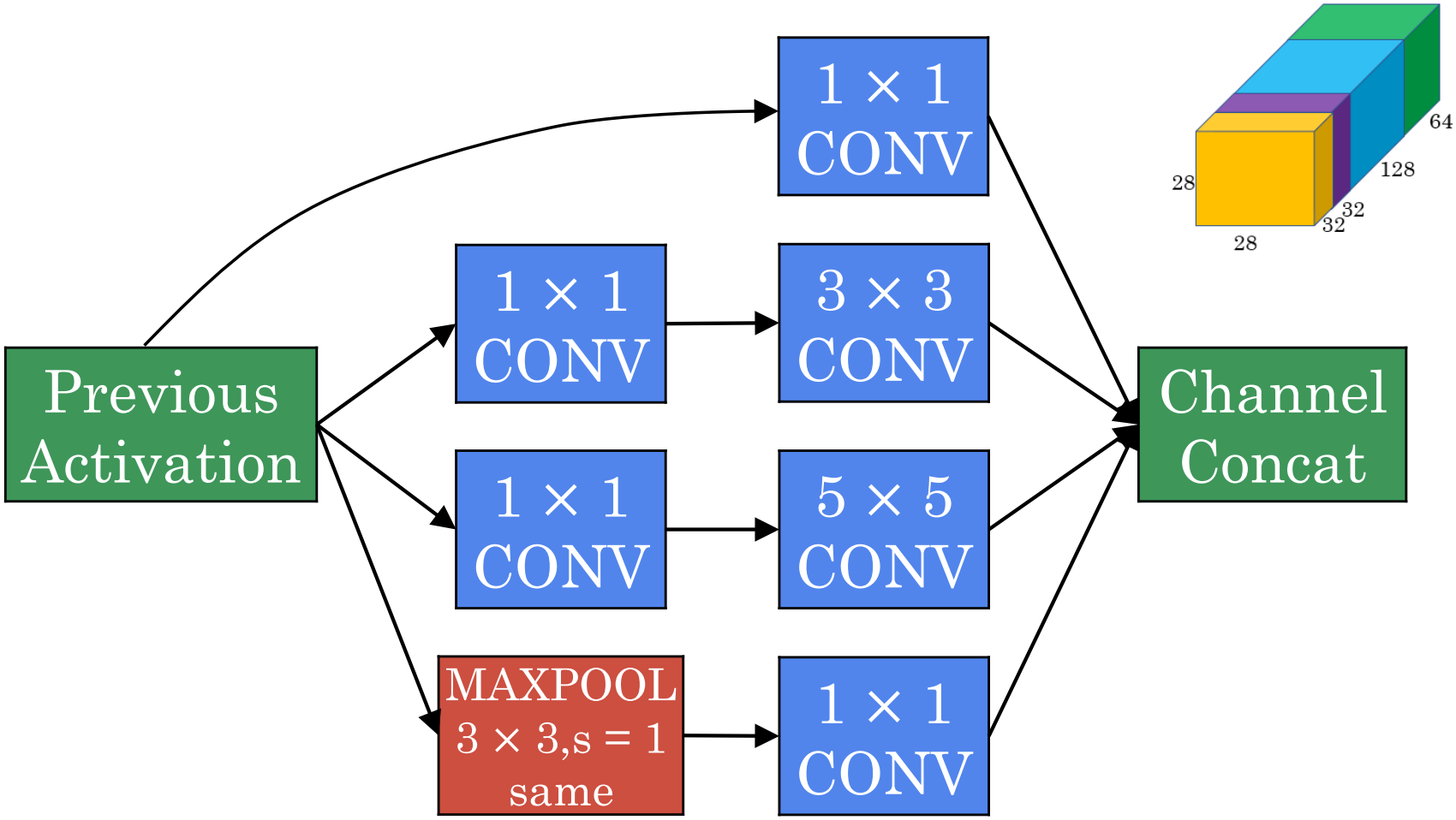
Módulo Inception



Usando convolução 1x1



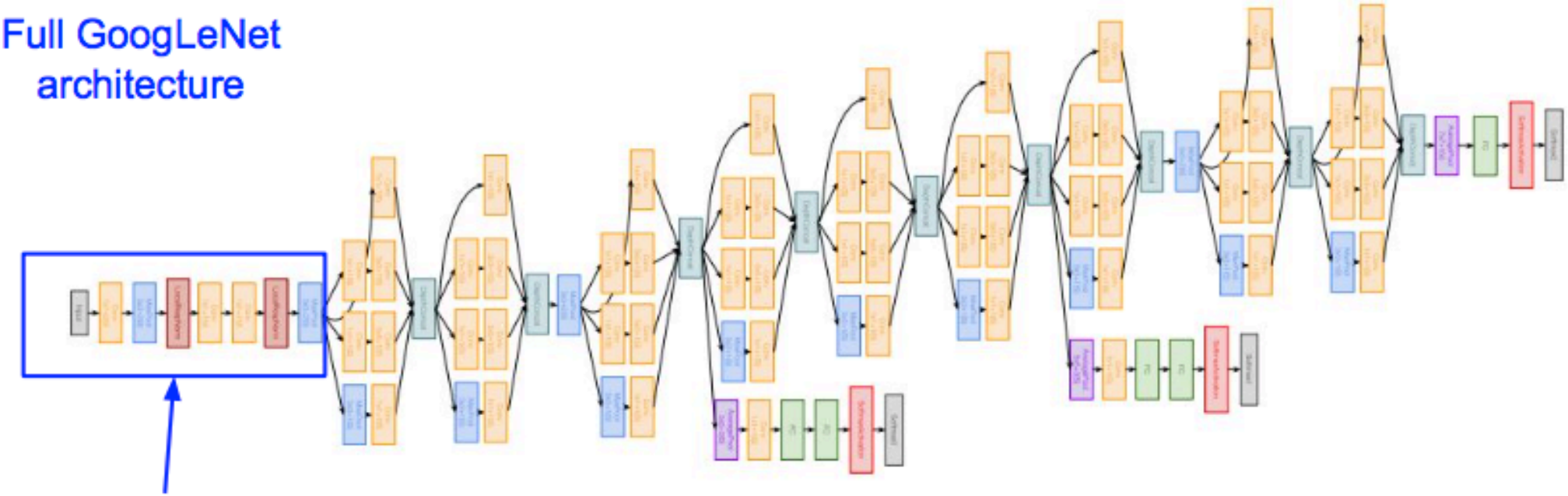
Módulo Inception



Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

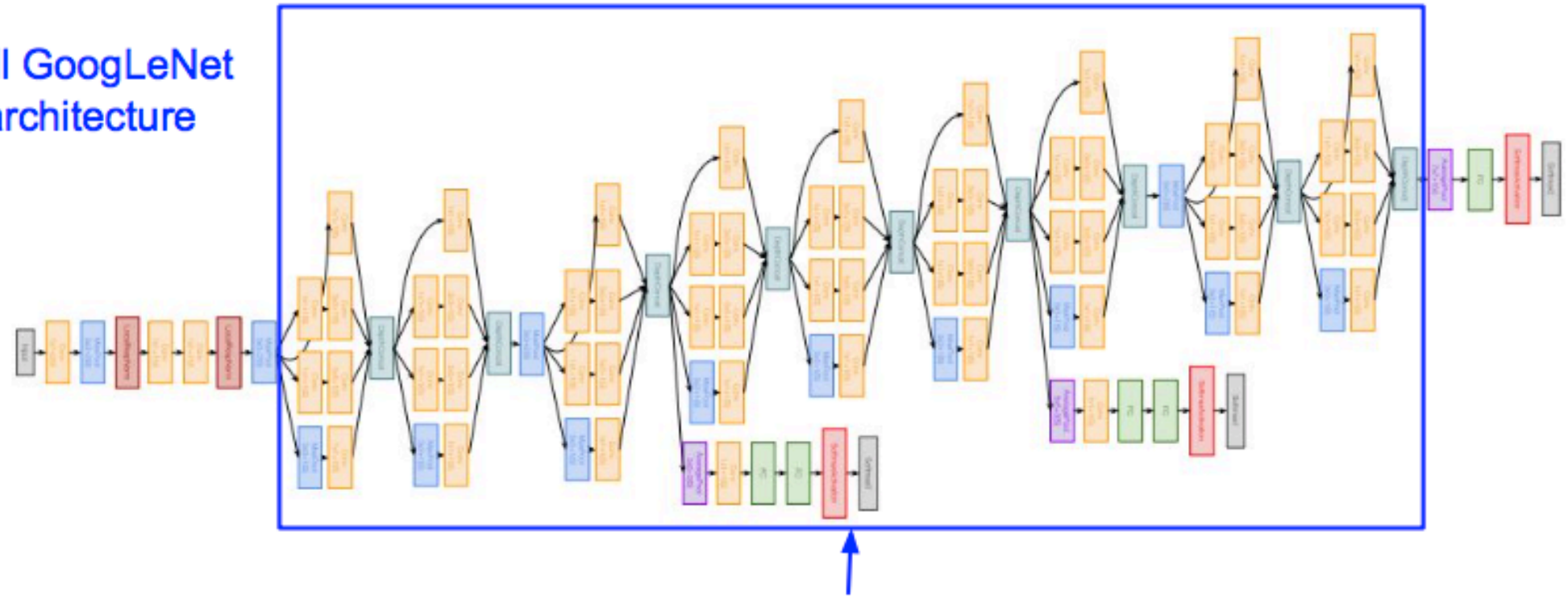


Stem Network: Conv-Pool- 2x Conv-Pool

Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

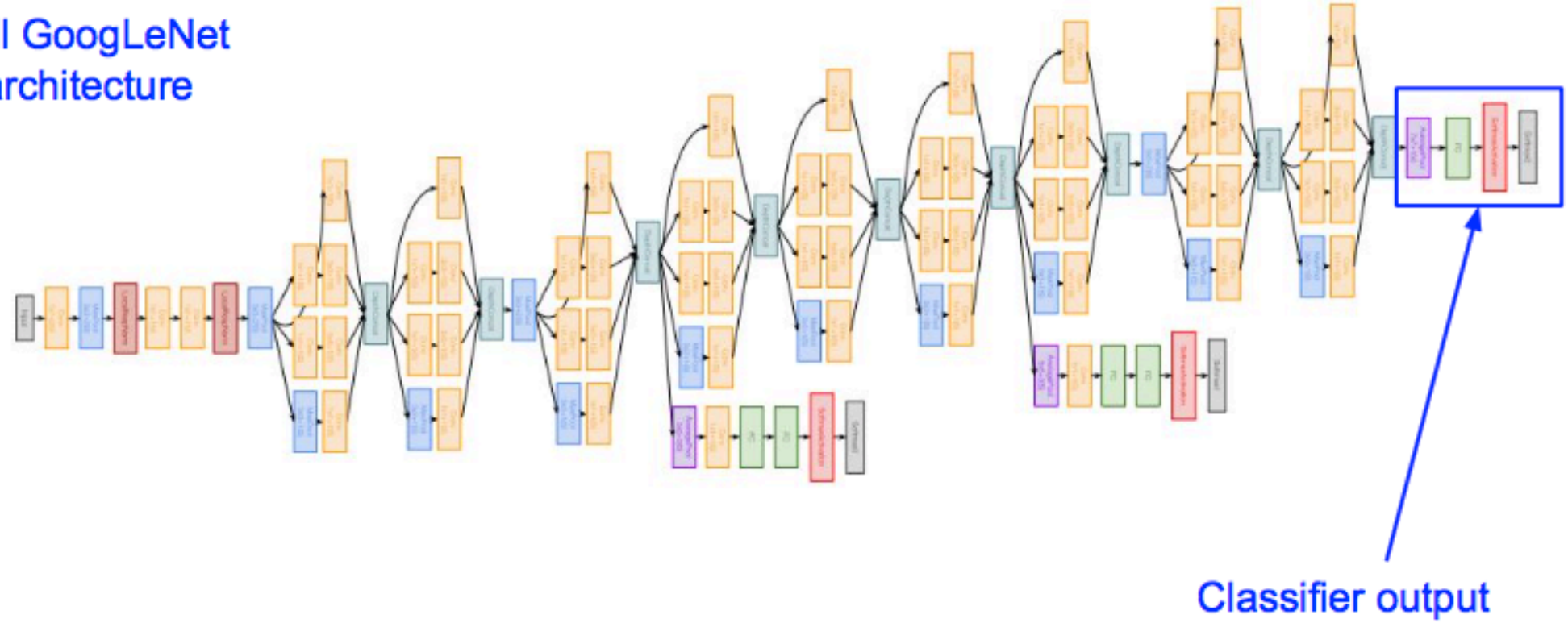


Stacked Inception Modules

Case Study: GoogLeNet

[Szegedy et al., 2014]

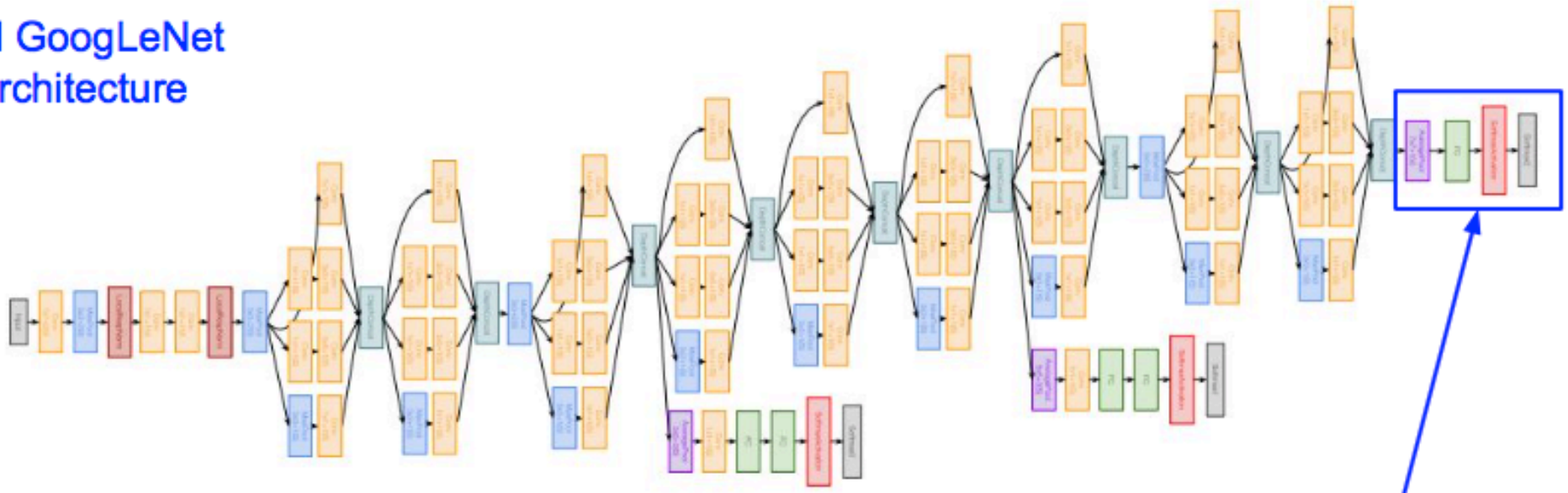
Full GoogLeNet architecture



Case Study: GoogLeNet

[Szegedy et al., 2014]

Full GoogLeNet architecture

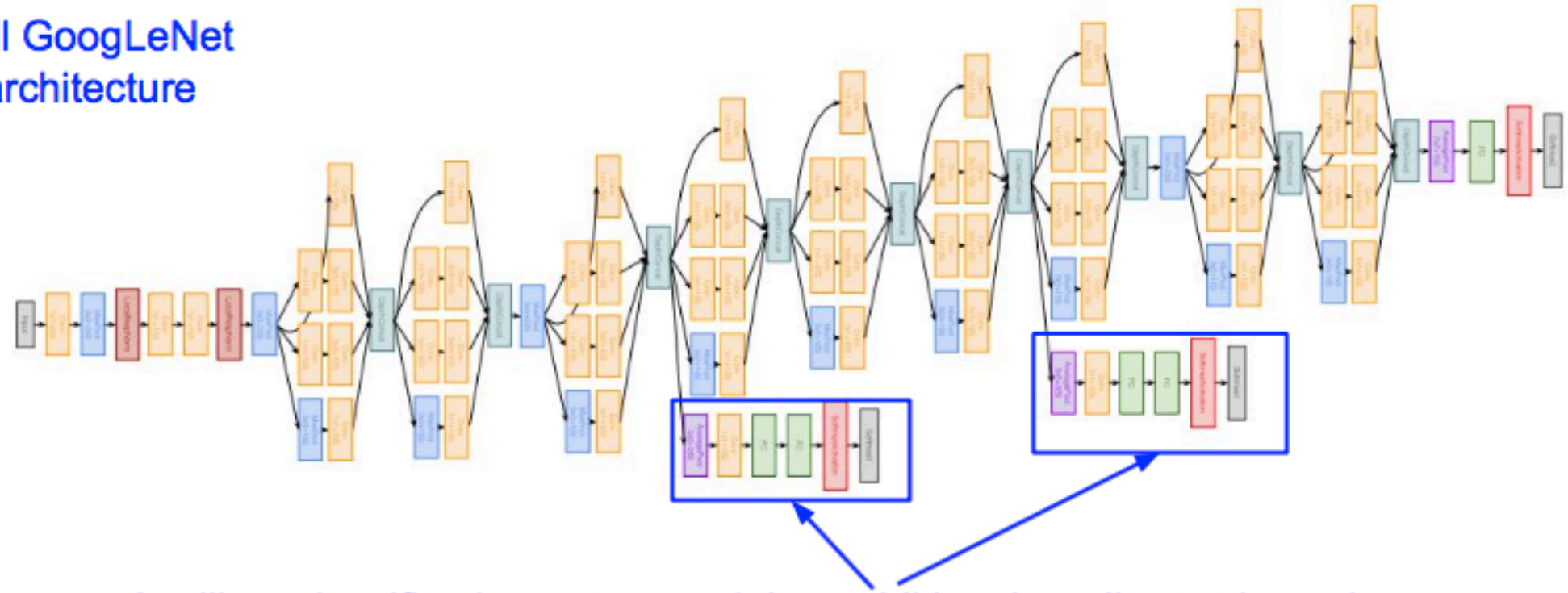


Classifier output
(removed expensive FC layers!)

Case Study: GoogLeNet

[Szegedy et al., 2014]

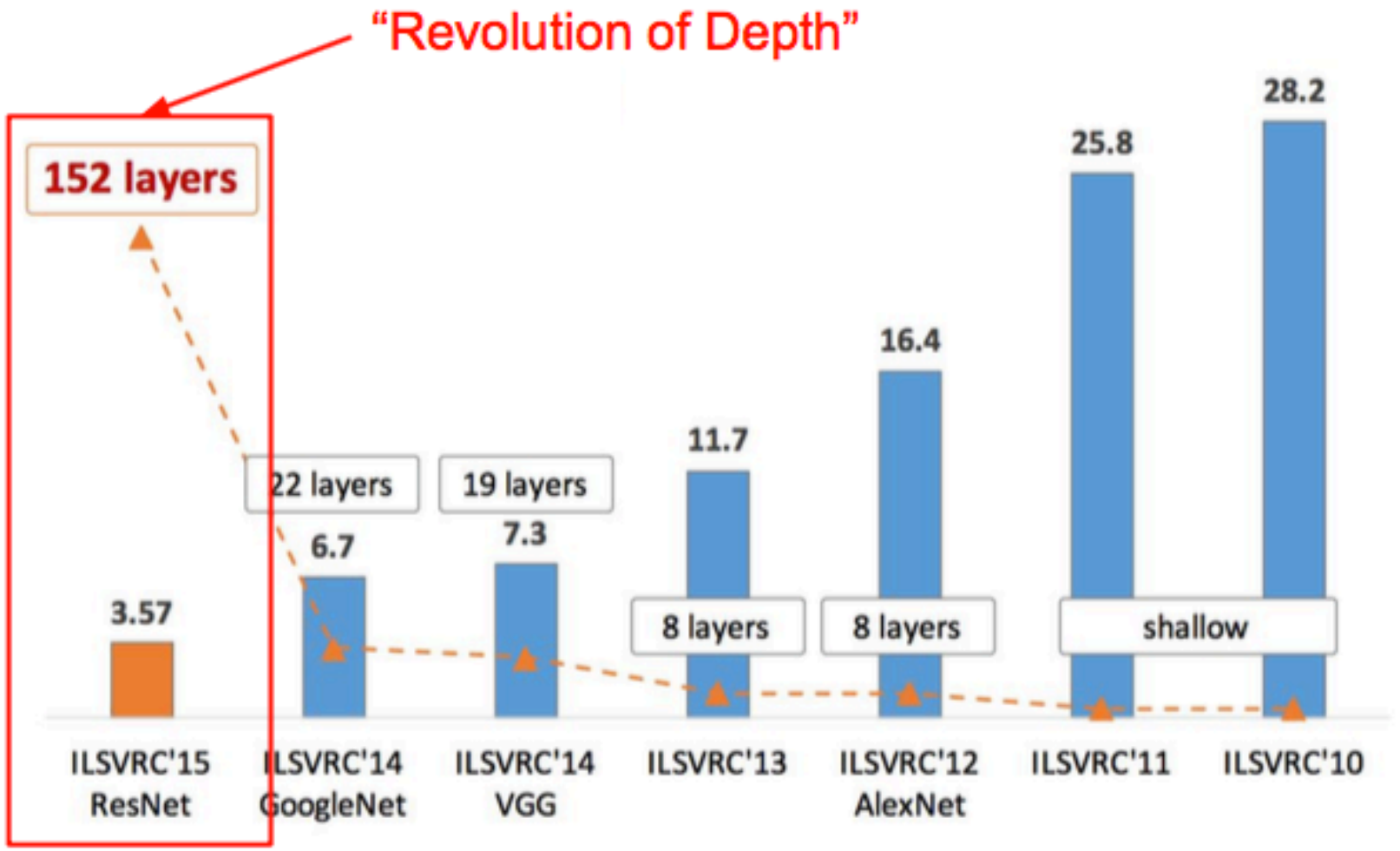
Full GoogLeNet architecture



Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

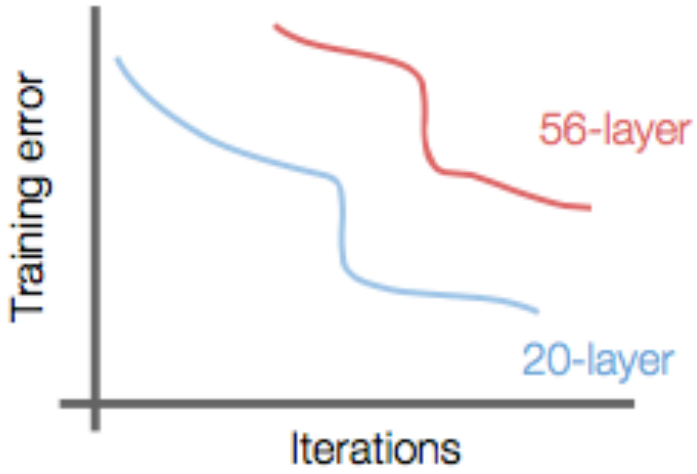

WE NEED TO GO
DEEPER

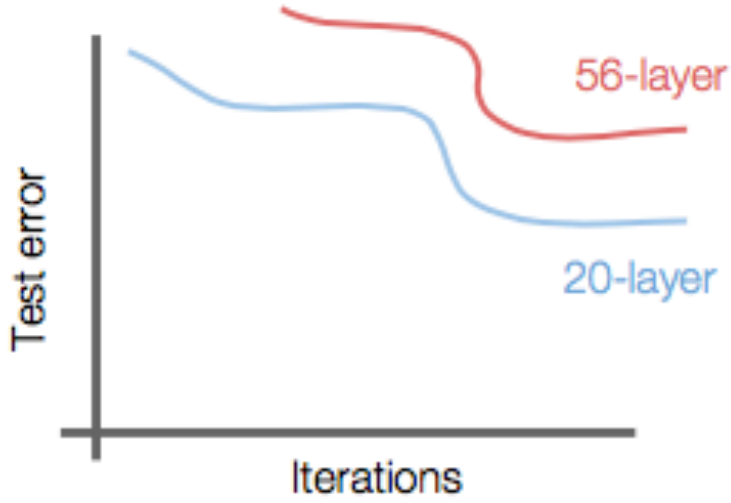
- Vencedores da ILSVRC (ImageNet Large Scale Visual Recognition Challenge)



ResNet (Microsoft) [He et al., 2015]

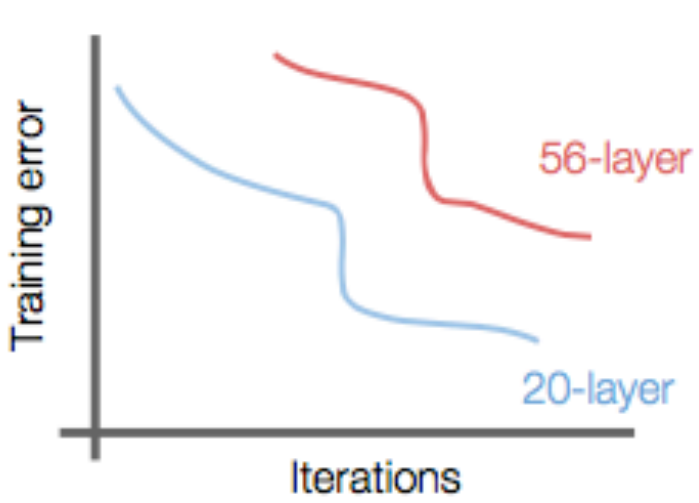
- O que acontece quando continuamos a empilhar mais camadas profundas na CNN?

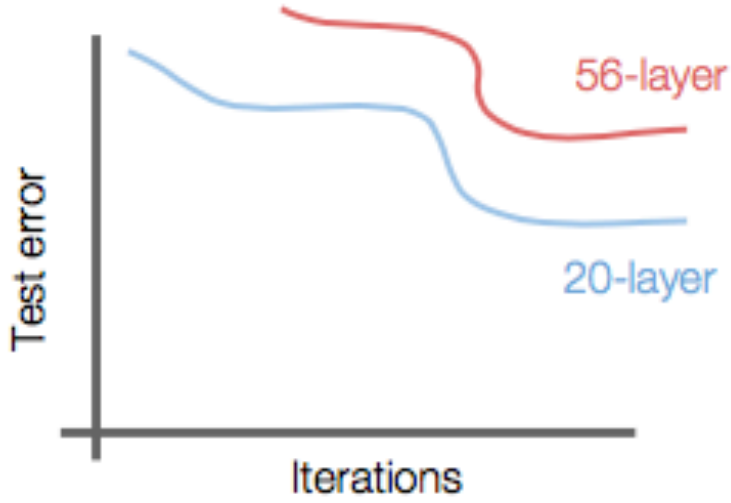




ResNet (Microsoft) [He et al., 2015]

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





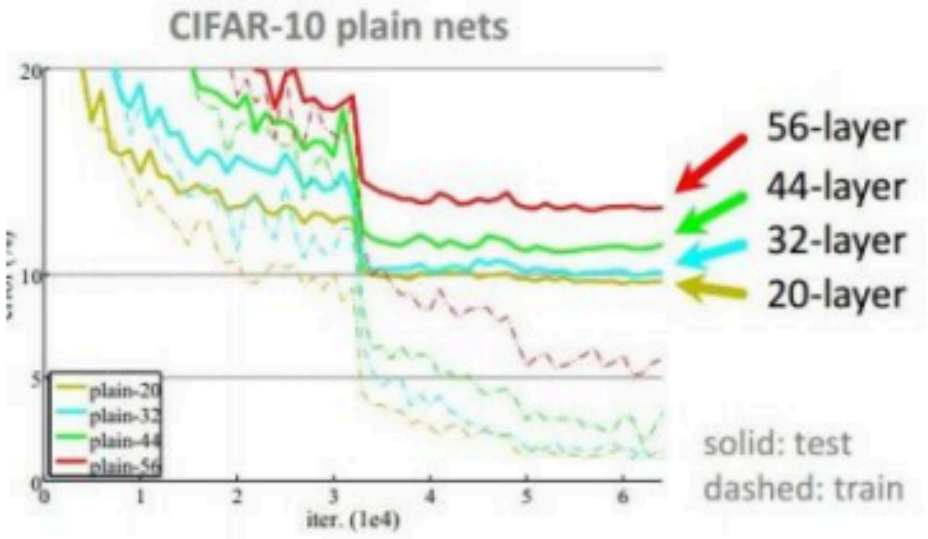
O modelo mais profundo é pior, mas não é por causa de overfitting

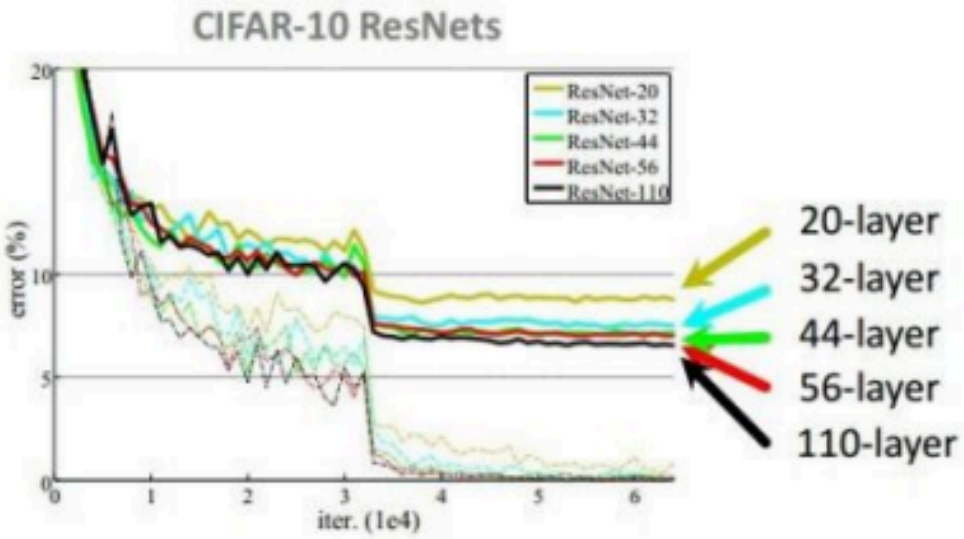
ResNet (Microsoft) [He et al., 2015]

- Resultados
  - Capaz de treinar uma *very deep network* (152 camadas) sem perder desempenho
  - Primeiro lugar em todas as competições do ILSVRC e COCO
  - 3.6% de erro no ILSVRC (melhor que desempenho humano)

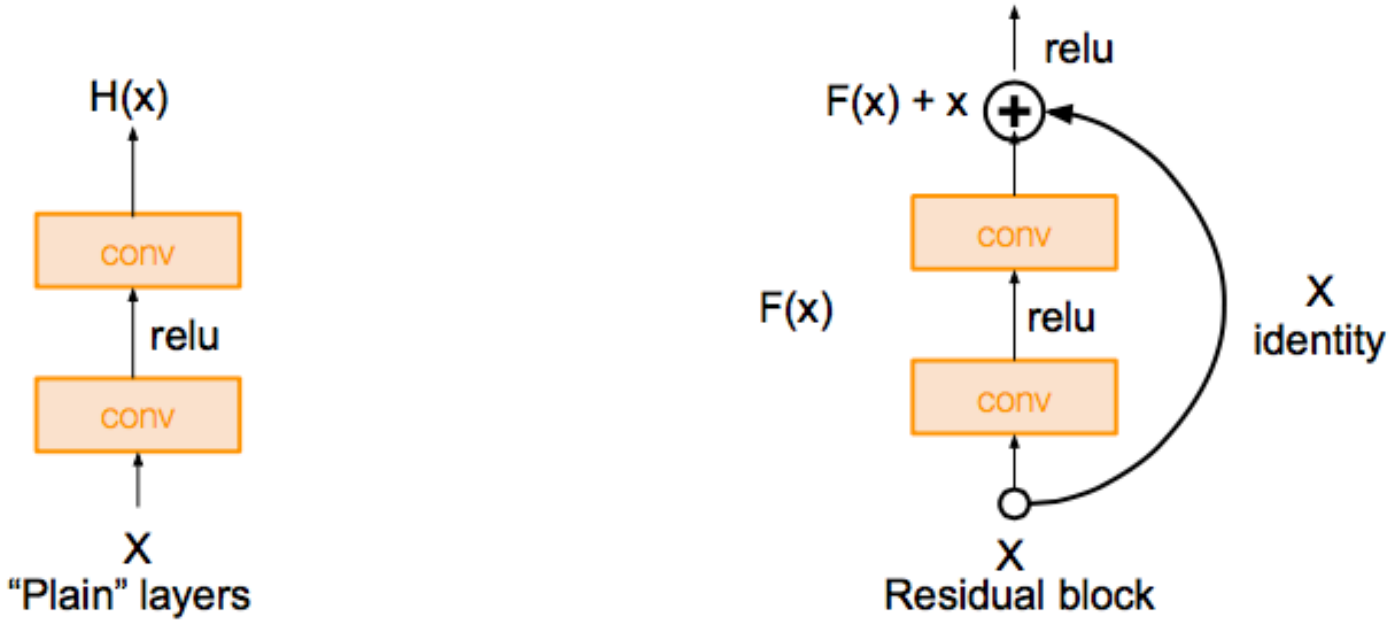
Going deeper

Performance of ResNets versus plain-nets as depth is increased





Bloco Residual

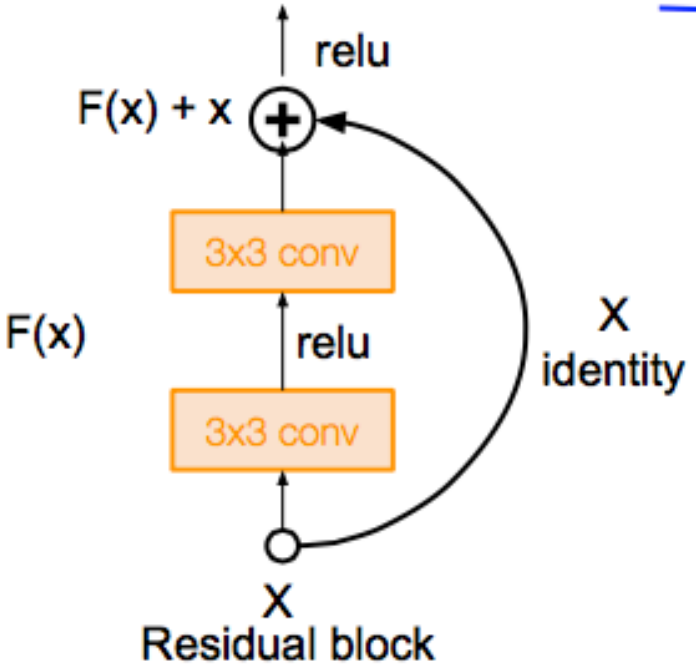


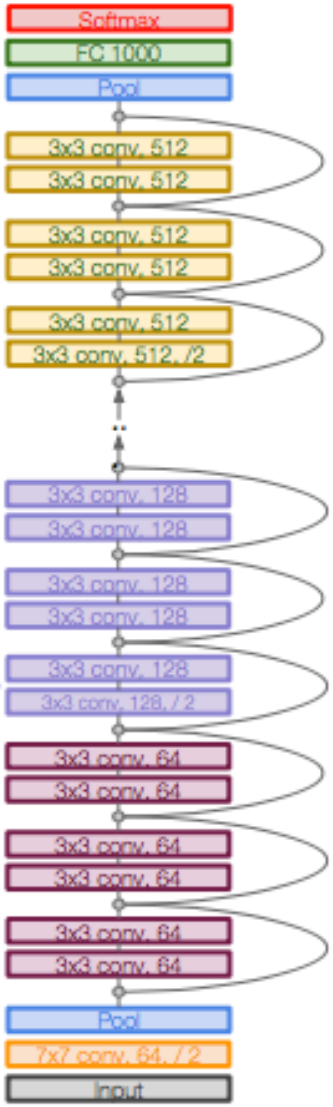
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers



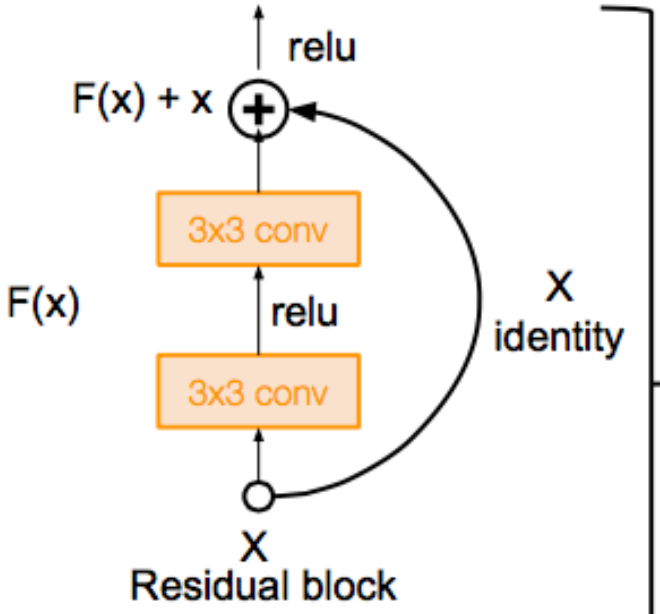


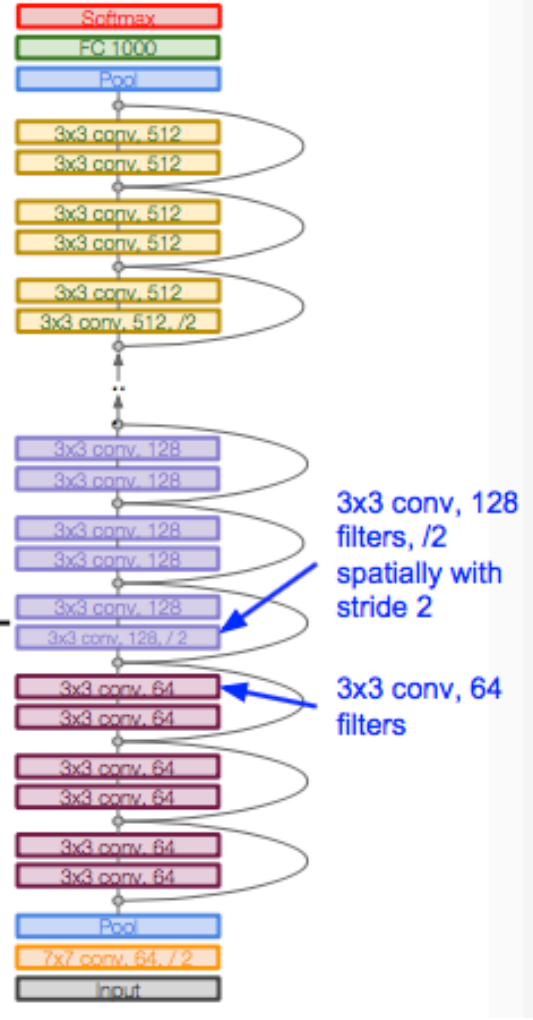
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)



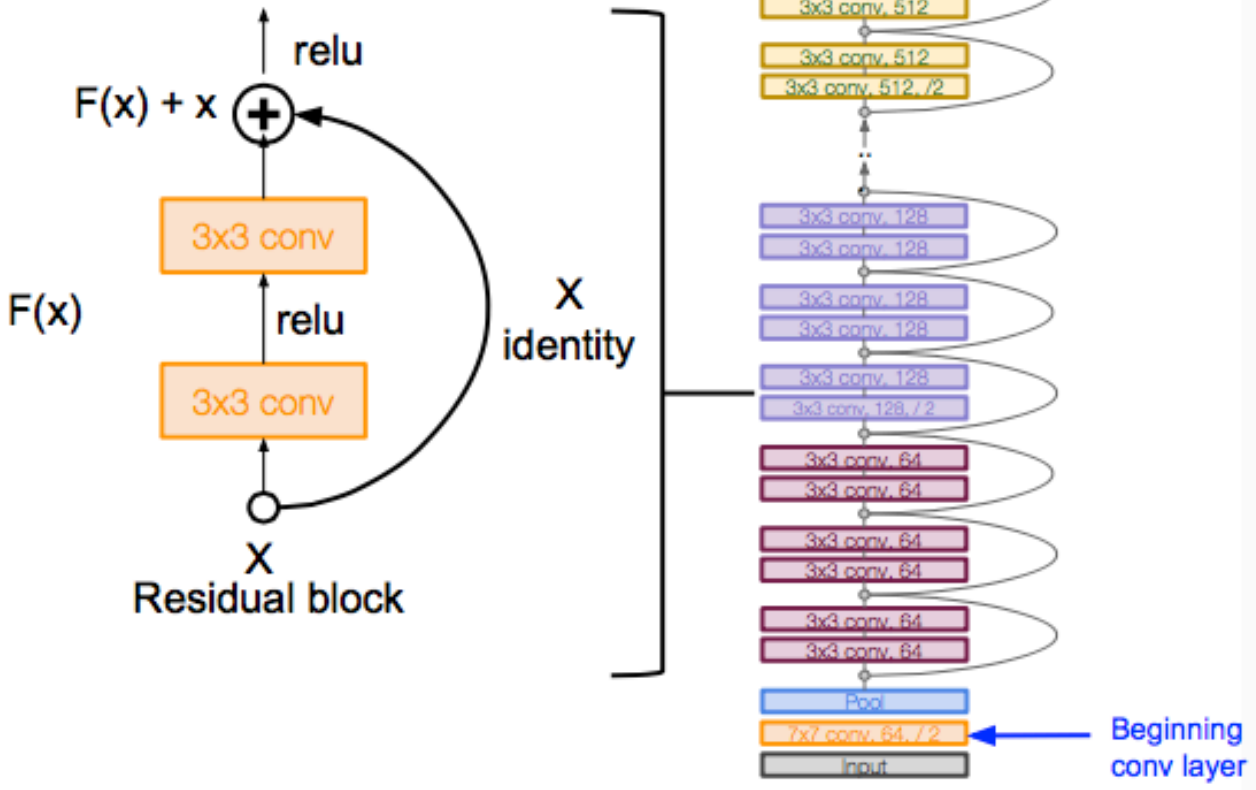


Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning

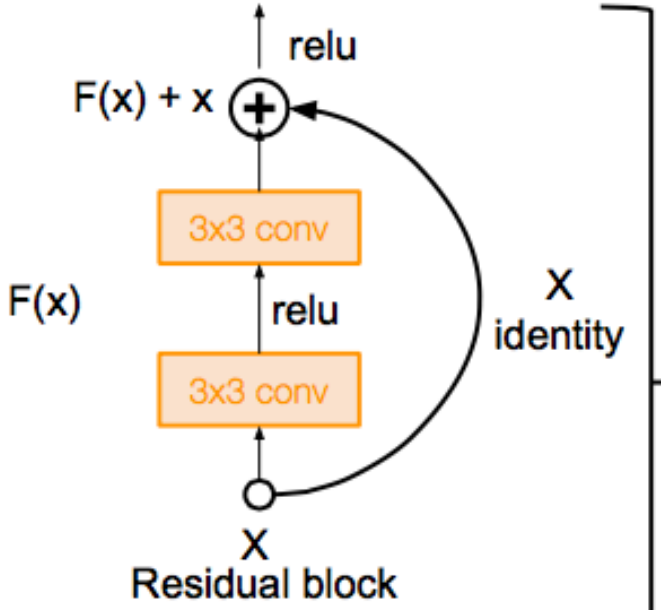


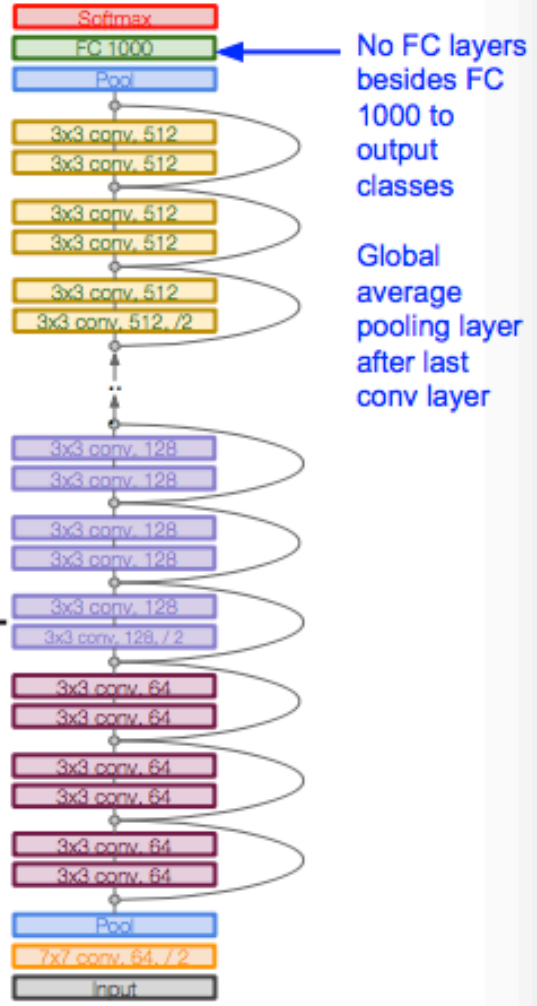
Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)

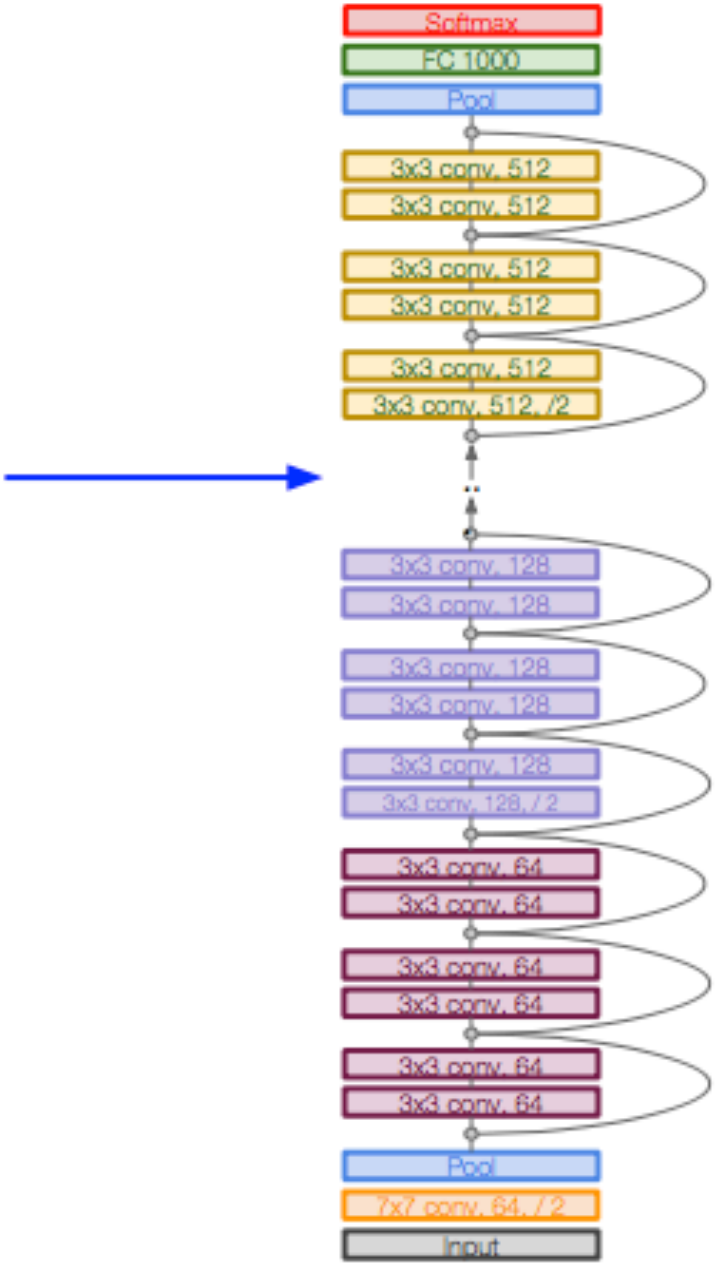




Case Study: ResNet

[He et al., 2015]

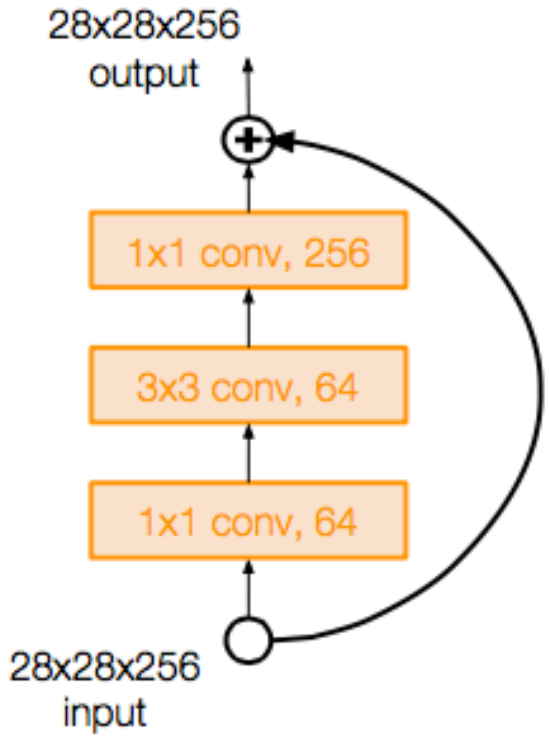
Total depths of 34, 50, 101, or
152 layers for ImageNet



Case Study: ResNet

[He et al., 2015]

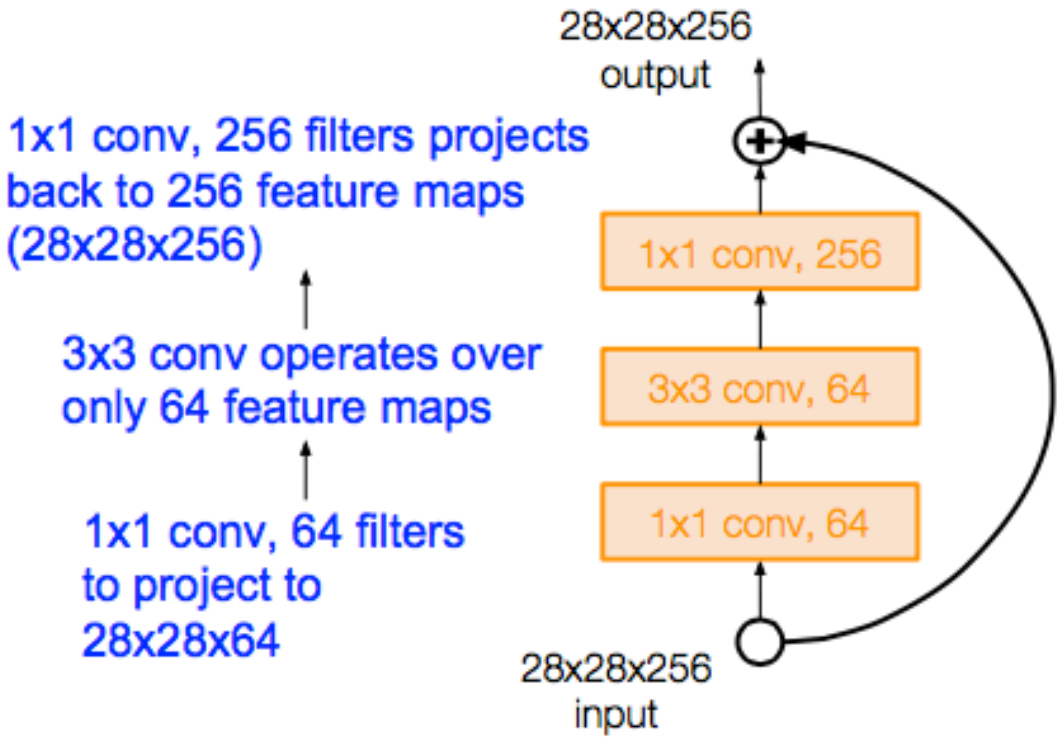
For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



Case Study: ResNet

[He et al., 2015]

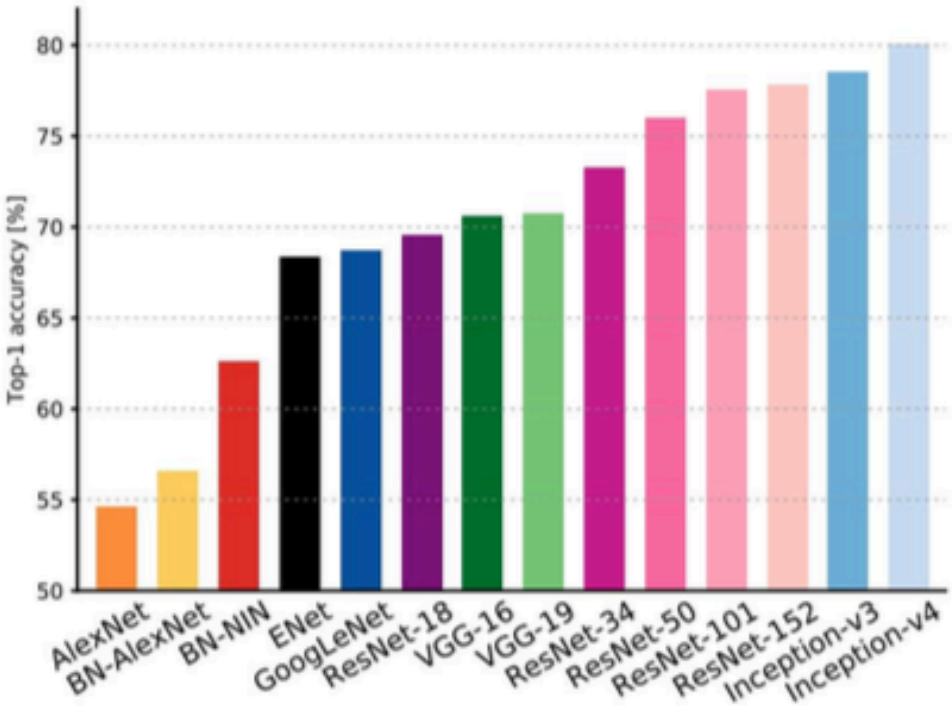
For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

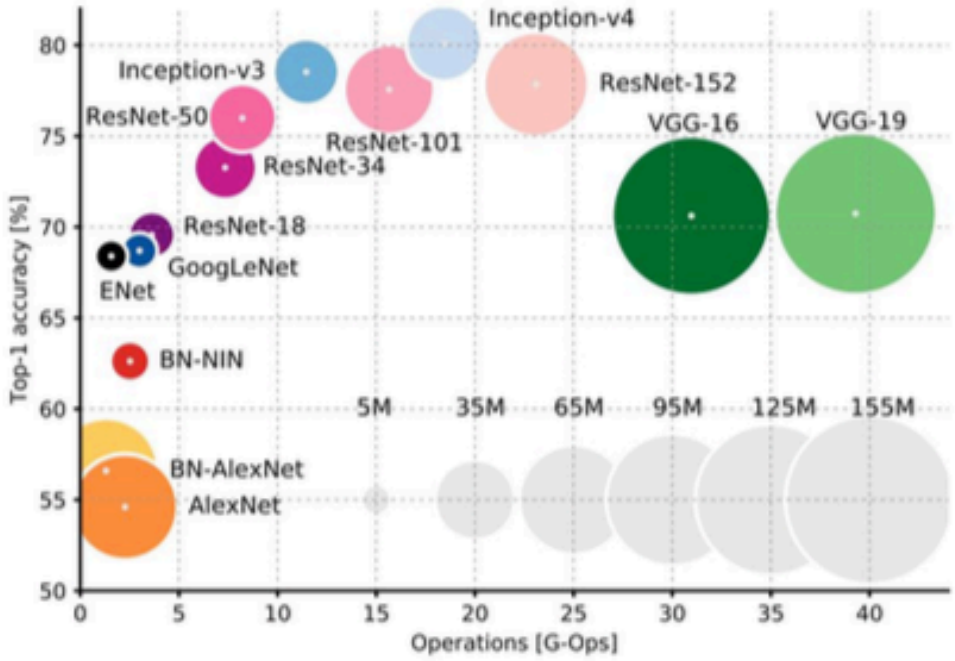


Deeper?

- Tentaram uma rede 1202 camadas, mas deu menor acurácia
  - Provavelmente por causa de overfitting.

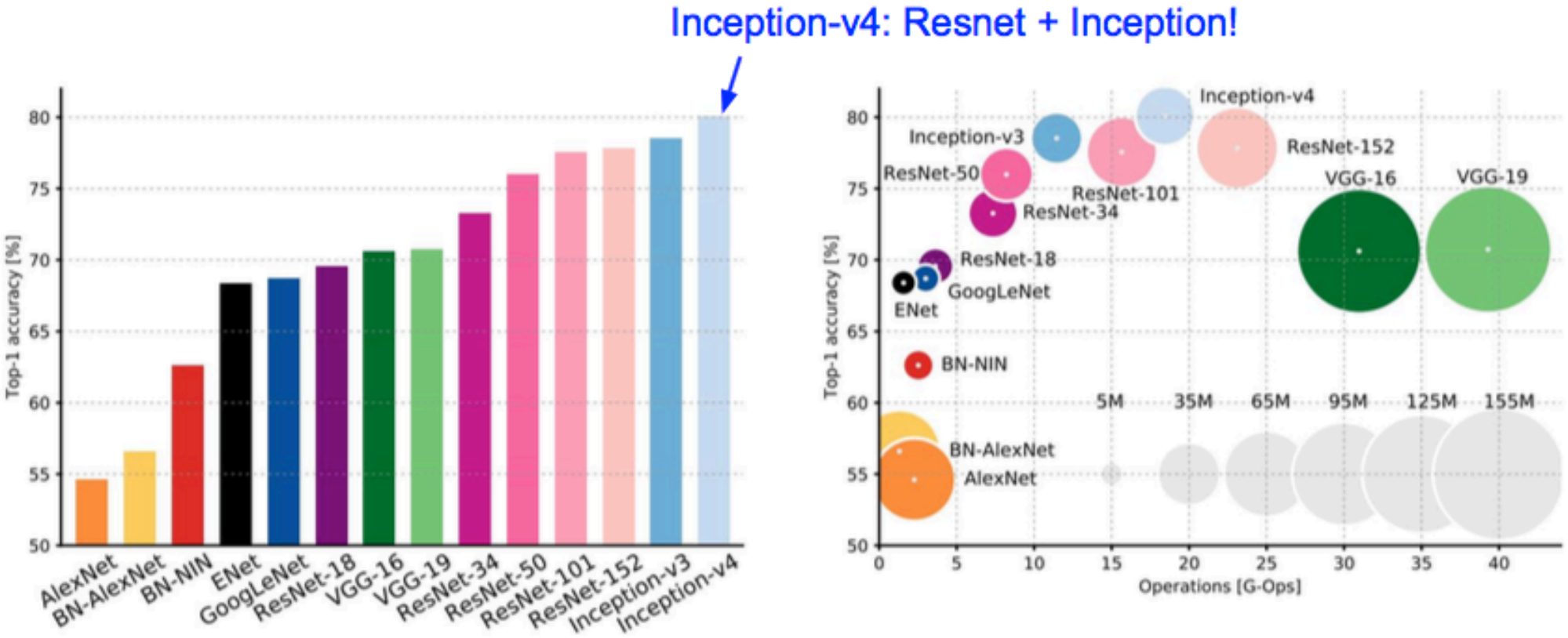
Comparando Complexidade





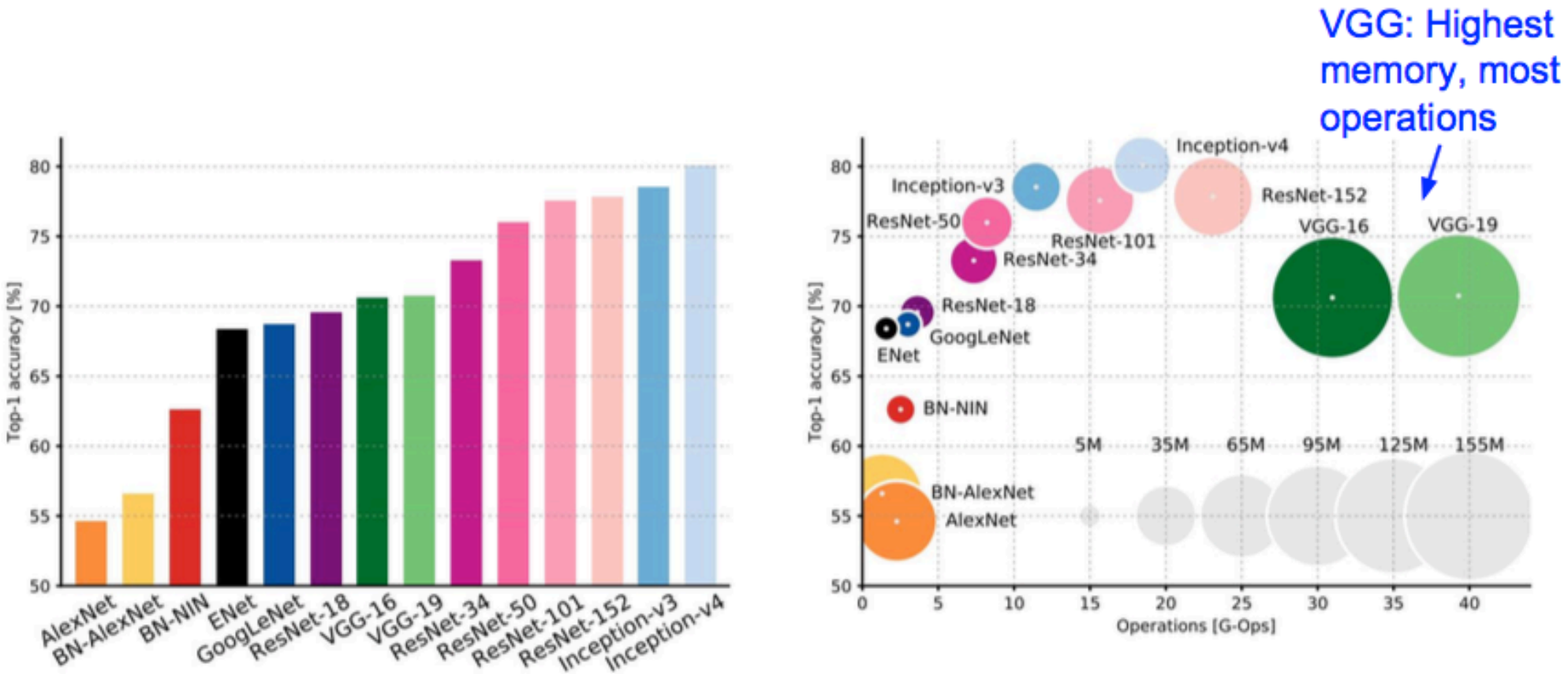
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparando Complexidade



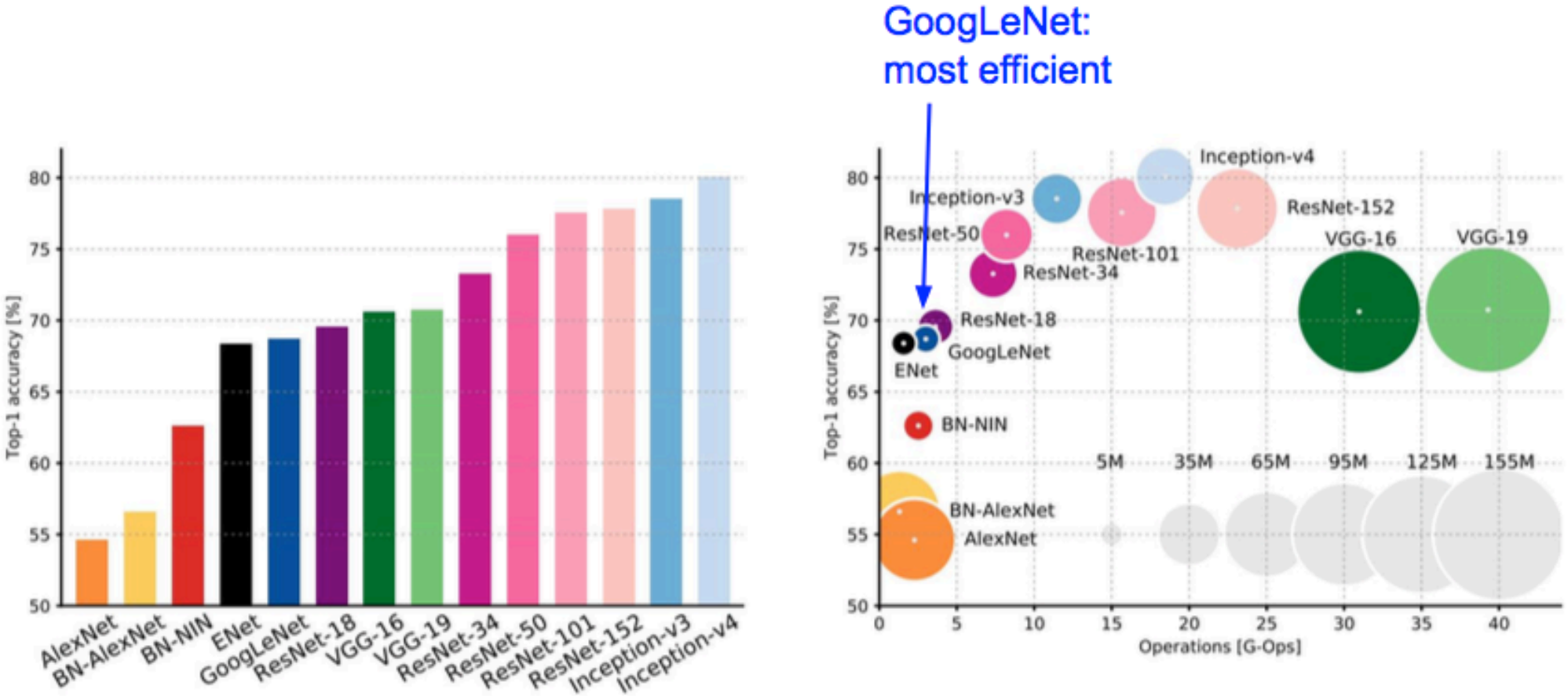
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparando Complexidade



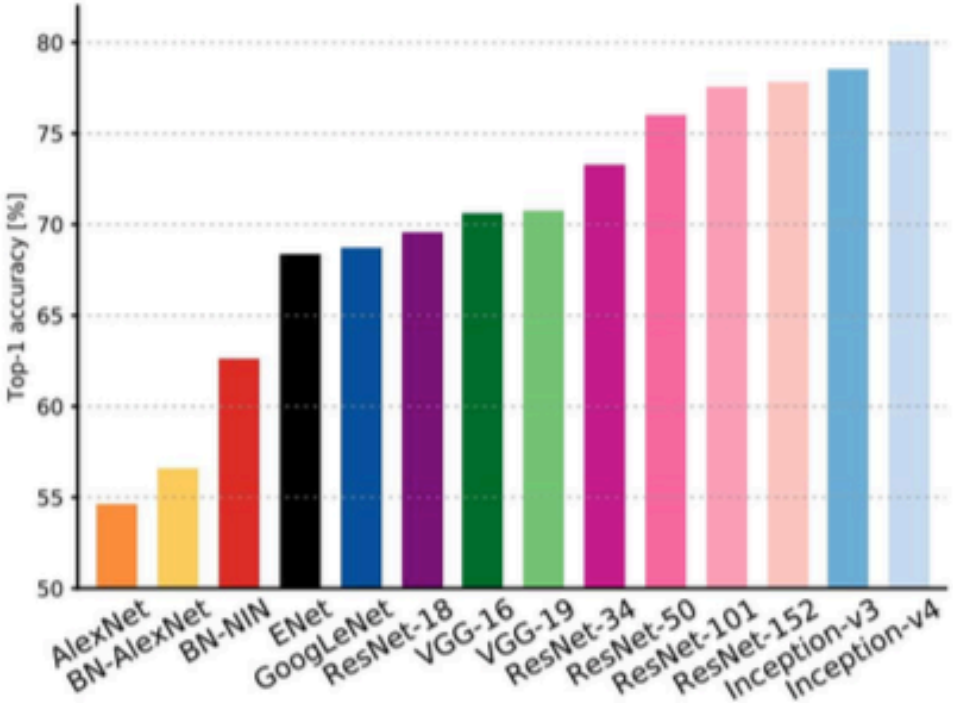
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

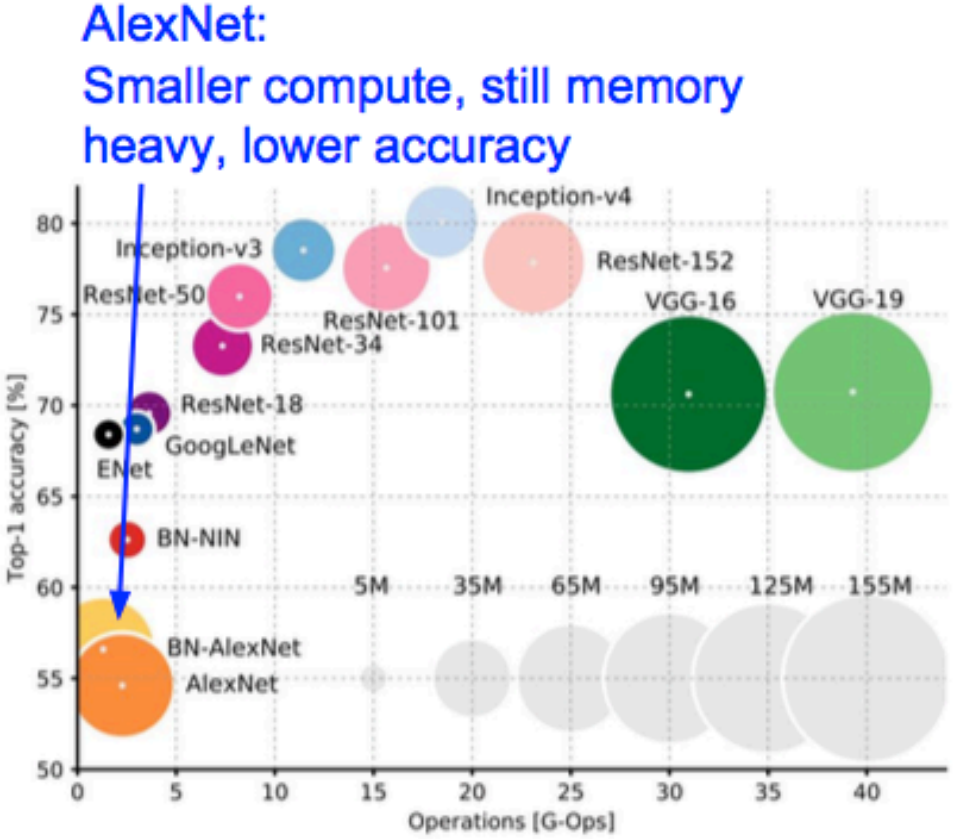
Comparando Complexidade



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

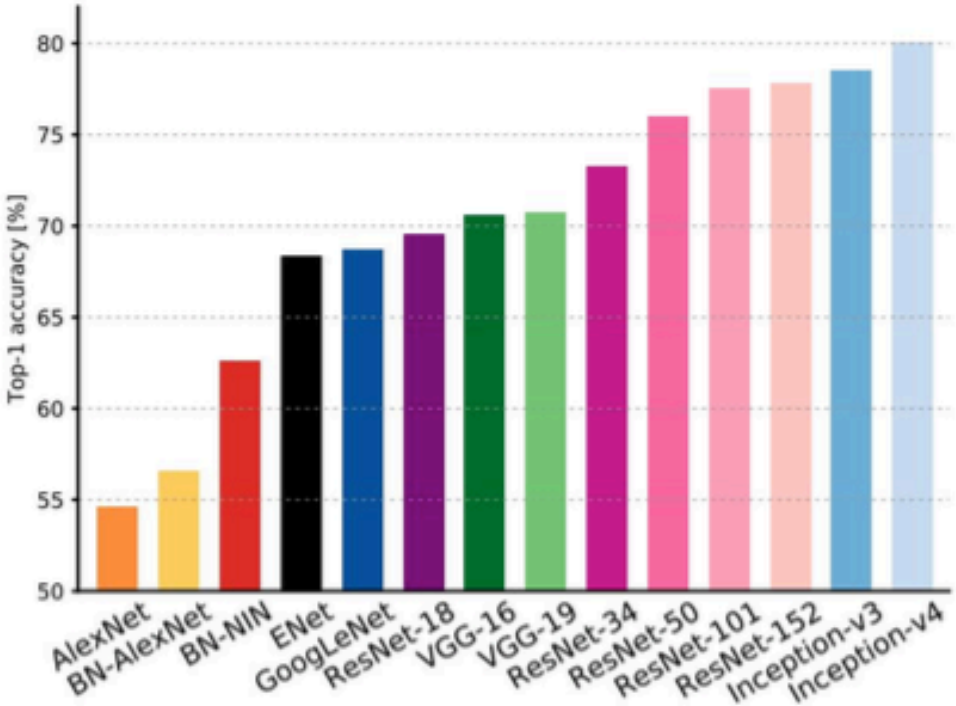
Comparando Complexidade

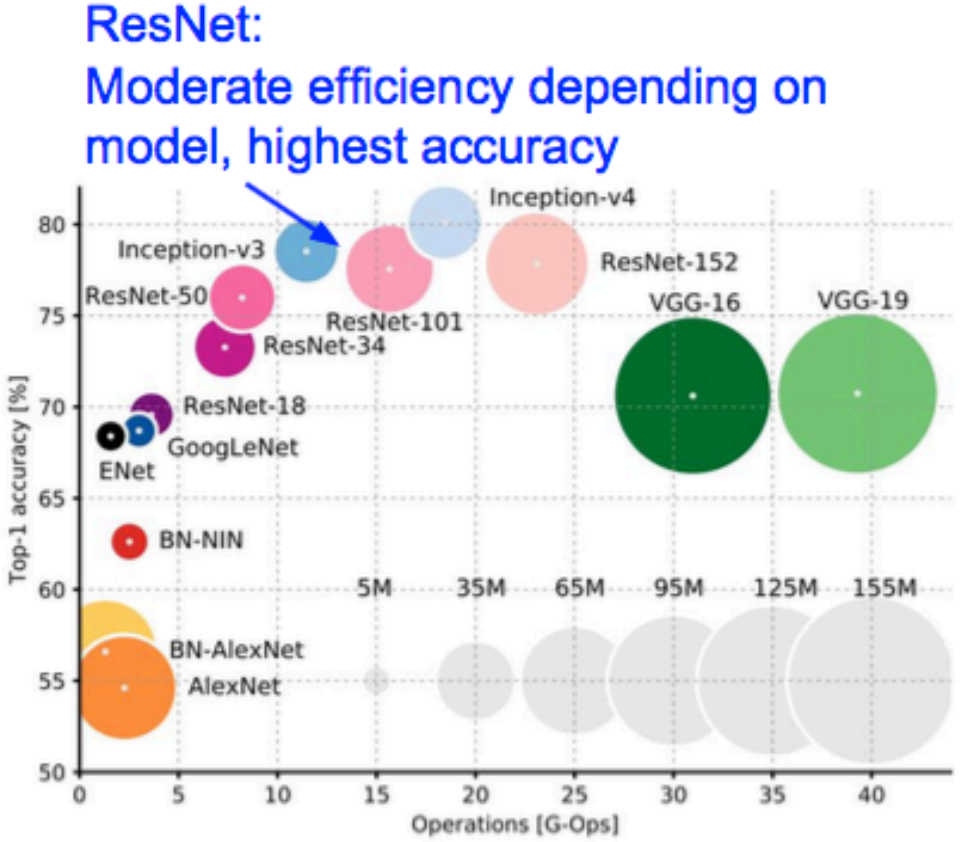




An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Comparando Complexidade



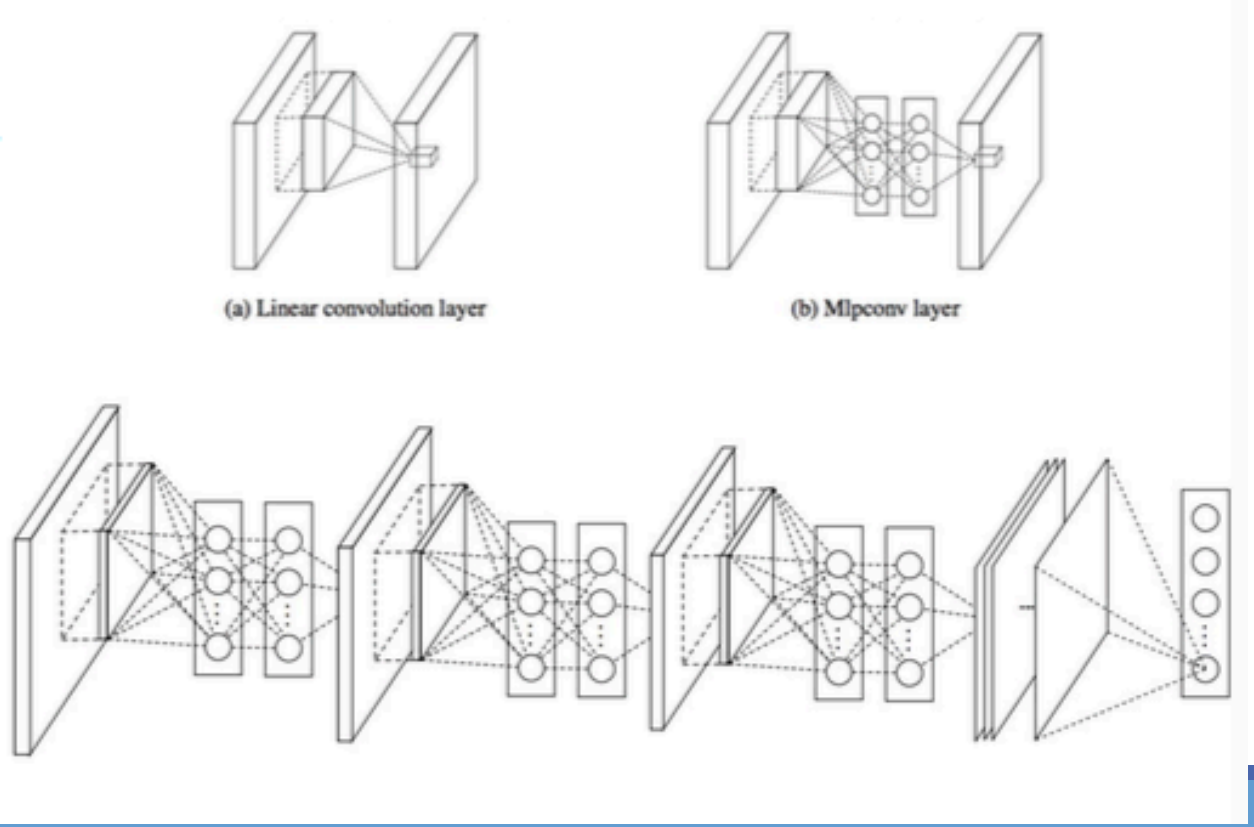


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Outras arquiteturas...

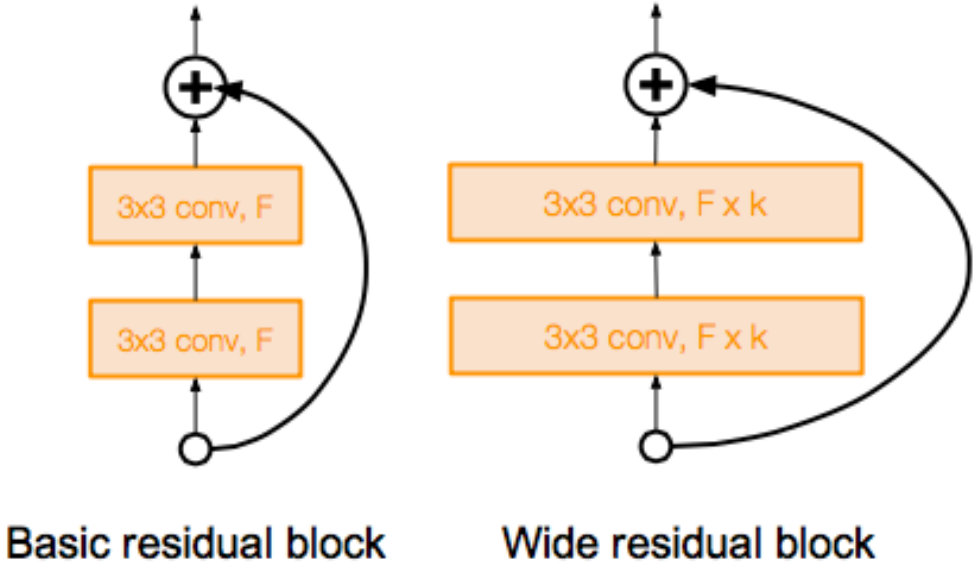
Network in Network (NiN) [Lin et al. 2014]

- Camada **Mlpconv** com ”microrede” dentro de cada camada convolutiva para aprender características mais abstratas para patches locais;
- Microrede usa MLP



Wide Residual Networks [Zagoruyko et al. 2016]

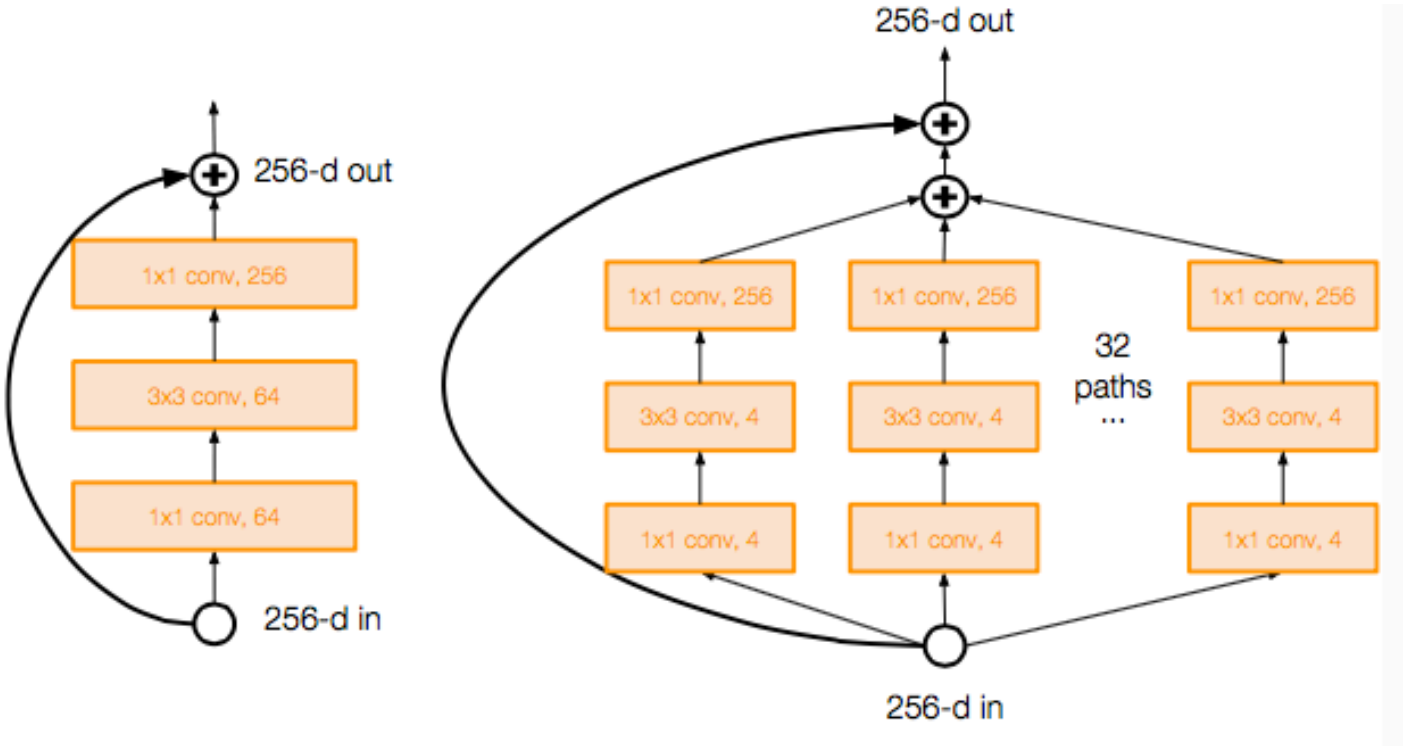
- Usa blocos residuais mais largos (F x k filtros ao invés de F filtros em cada camada)
- Wide ResNet de 50 camadas supera 152-layer ResNet



Basic residual block Wide residual block

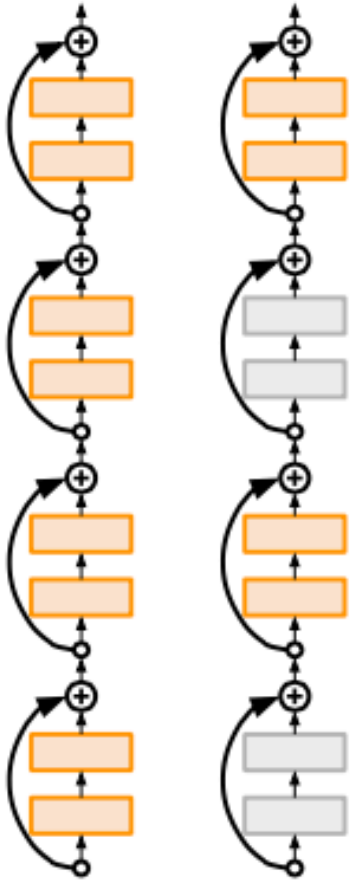
Aggregated Residual Transformations for Deep Neural Networks (ResNetXt)

- [Xie et al. 2016] (criadores do ResNet)
- Adiciona Múltiplos caminhos (similar ao Inception)



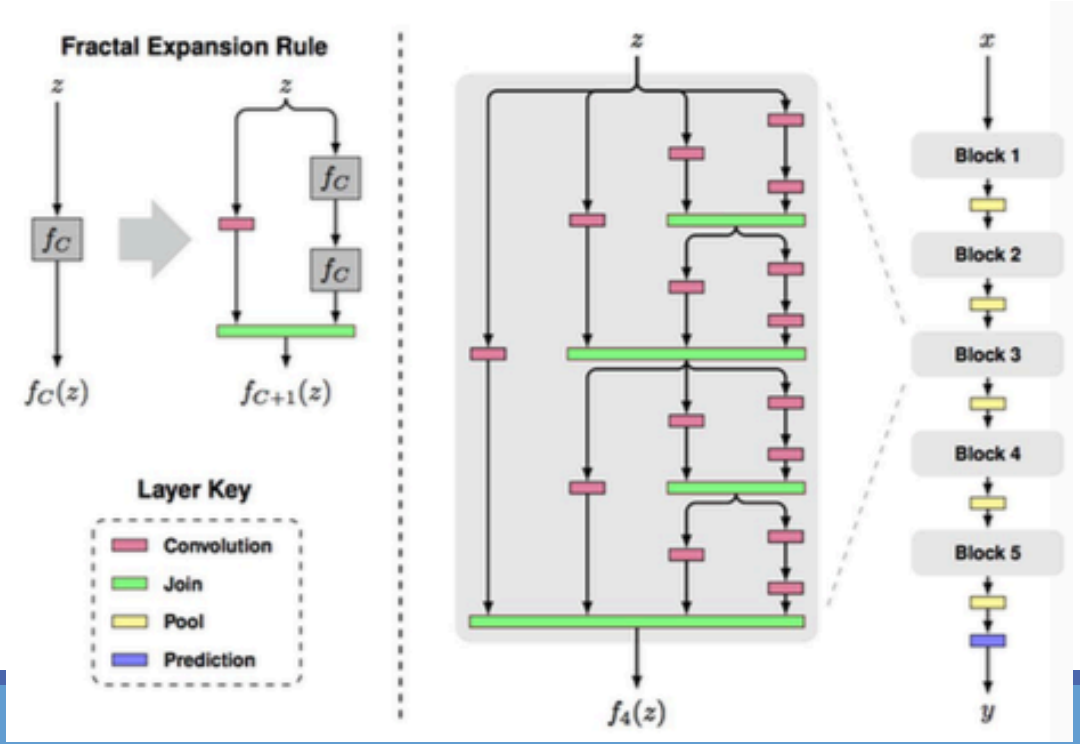
Deep Networks with Stochastic Depth

- [Huang et al. 2016]
- Aleatoriamente desabilita um subconjunto de camadas na etapa de treinamento
- Usa a rede toda na fase de testes.



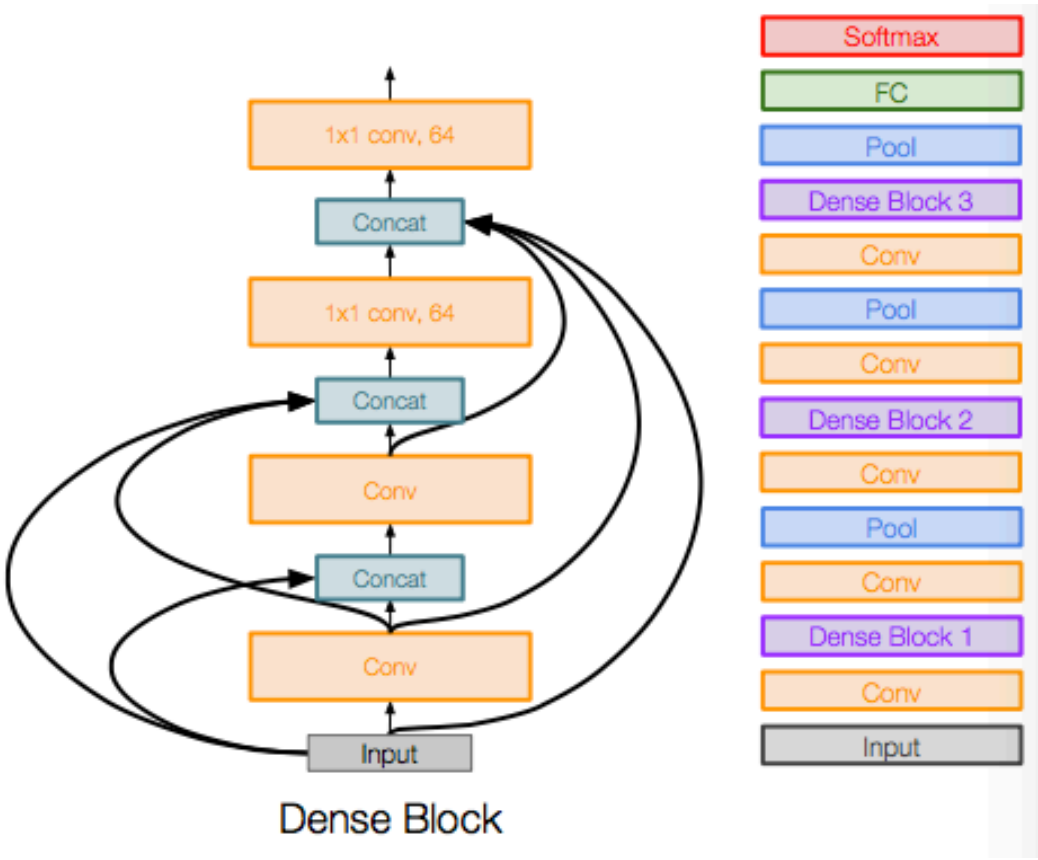
FractalNet: Ultra-Deep Neural Networks without Residuals

- [Larsson et al. 2017]
- Arquitetura com caminhos simples e profundos ao mesmo tempo
- Desabilita caminhos no treinamento
- Utiliza todos os caminhos na fase de teste



Densely Connected Convolutional Networks

- [Huang et al. 2017]
- Blocos Densos onde cada camada é conectada com todas as outras camadas



SqueezeNet

- [Iandola et al. 2017]
- Acurácia do AlexNet com 50x menos parâmetros e modelo < 0.5Mb

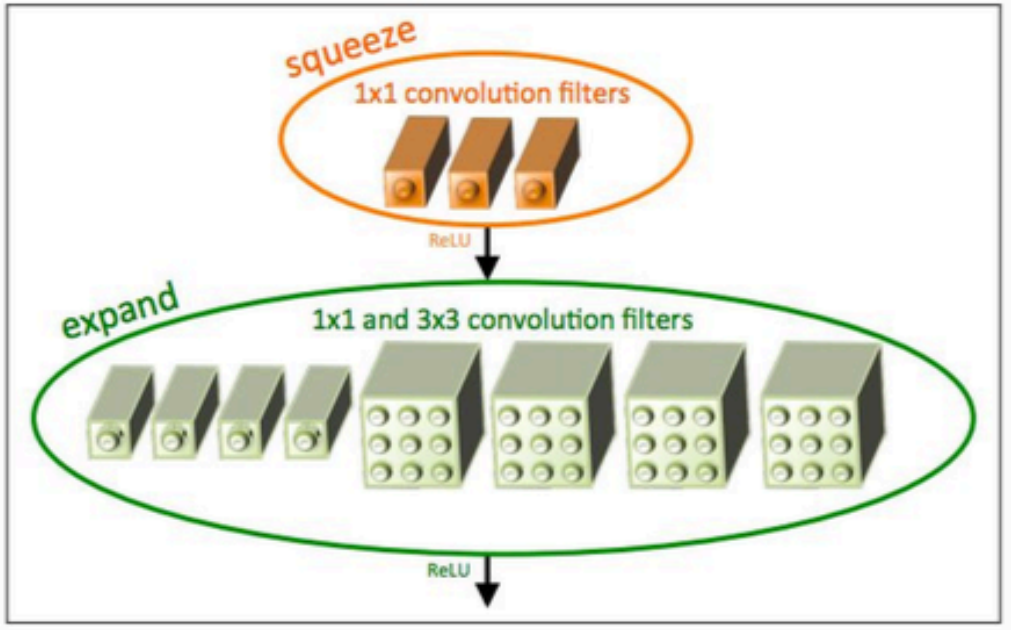


Figure copyright Iandola, Han, Moskewicz, Ashraf, Dally, Keutzer, 2017. Reproduced with permission.

Transfer Learning e Fine Tuning

É preciso treinar a rede toda vez?

- Transfer learning!

Model Zoos (TensorFlow)

- https://github.com/tensorflow/models/tree/master/research/slim

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50^	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8
ResNet V2 101^	Code	resnet_v2_101_2017_04_14.tar.gz	77.0	93.7
ResNet V2 152^	Code	resnet_v2_152_2017_04_14.tar.gz	77.8	94.1
ResNet V2 200	Code	TBA	79.9*	95.2*
VGG 16	Code	vgg_16_2016_08_28.tar.gz	71.5	89.8
VGG 19	Code	vgg_19_2016_08_28.tar.gz	71.1	89.8
MobileNet_v1_1.0_224	Code	mobilenet_v1_1.0_224_2017_06_14.tar.gz	70.7	89.5

Keras

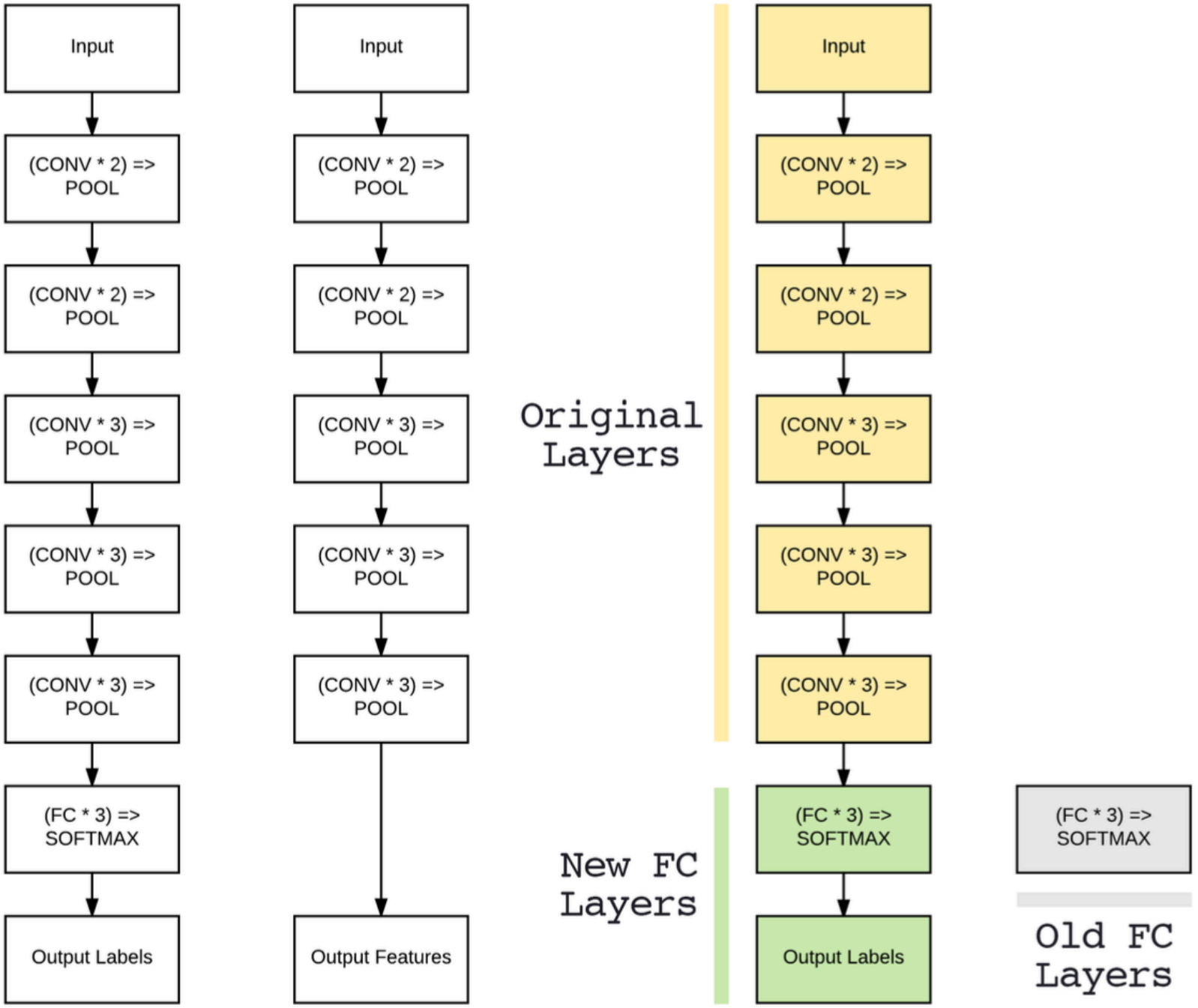
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88

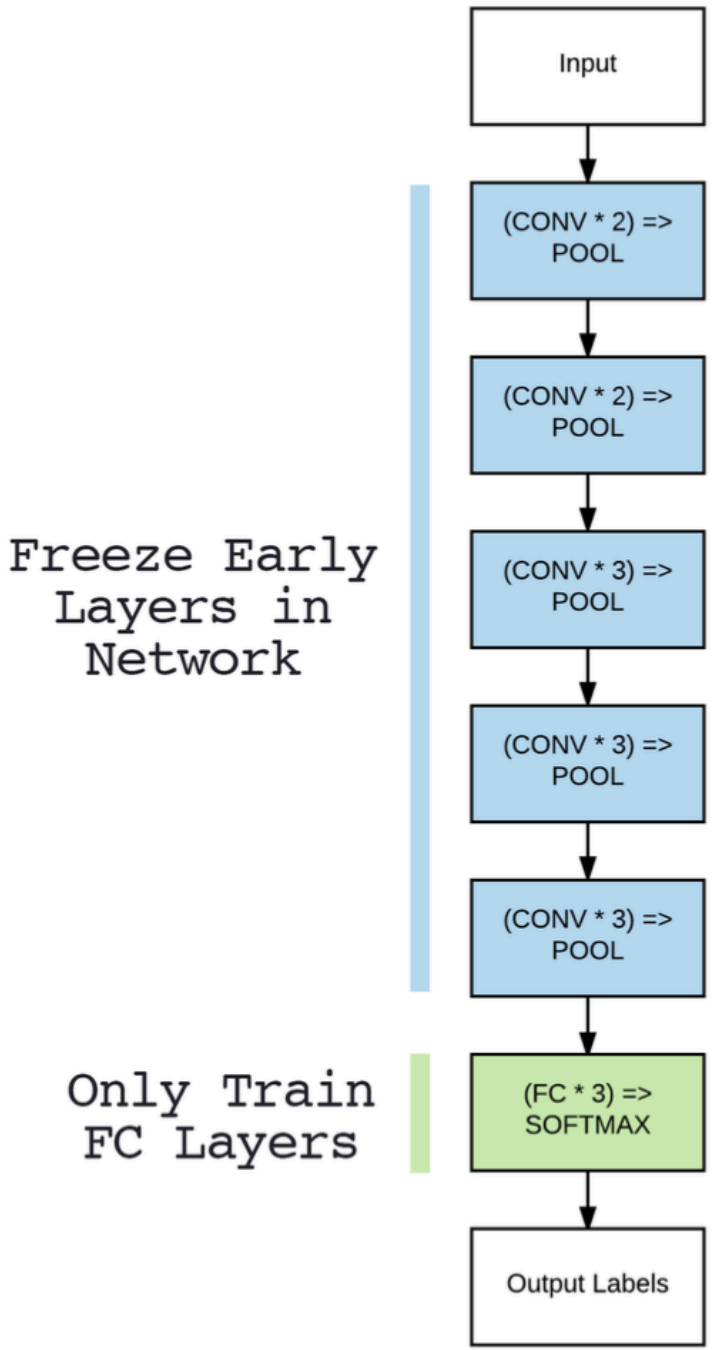
ResNet 50 (resnet50.py)

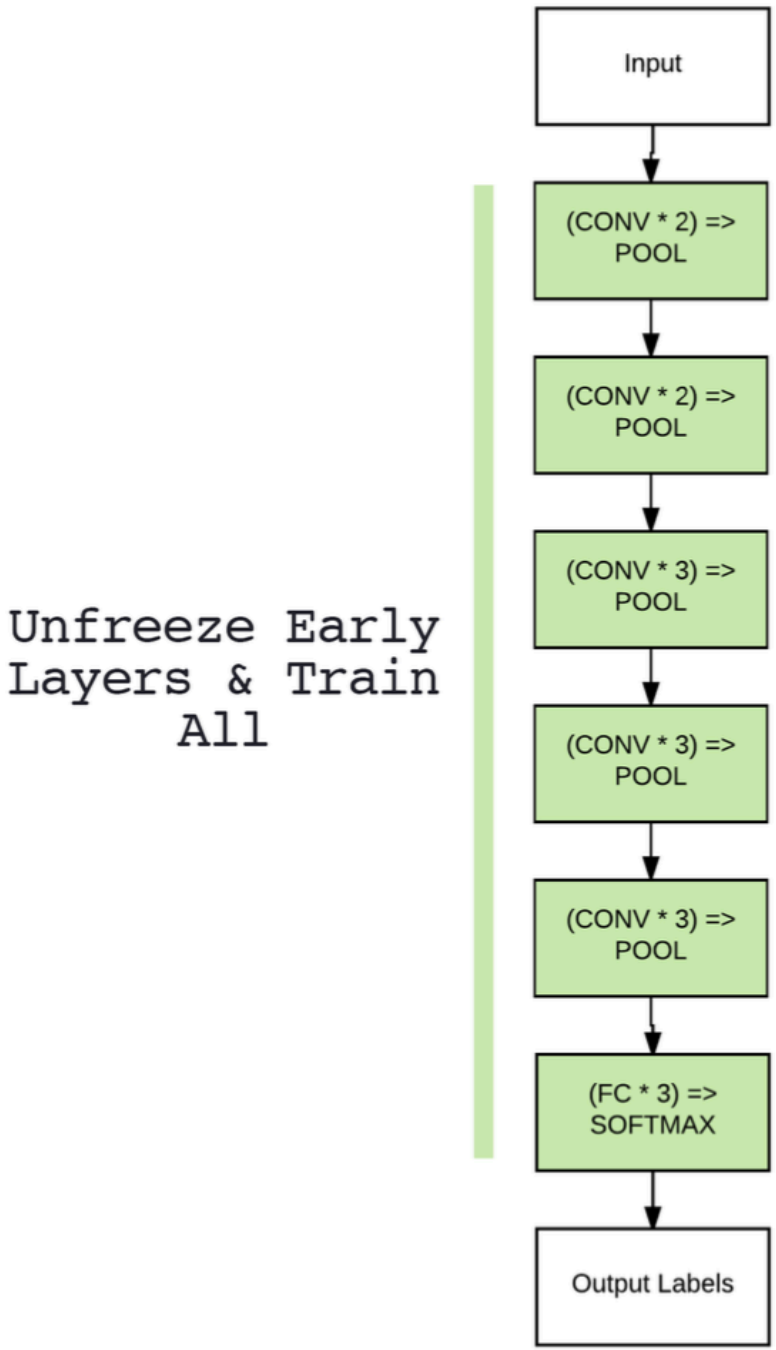
- Exercício

```
1  from keras.applications.resnet50 import ResNet50
2  from keras.preprocessing import image
3  from keras.applications.resnet50 import preprocess_input, decode_predictions
4  import numpy as np
5
6  model = ResNet50(weights='imagenet')
7  img_path = 'elephant.jpg'
8  img = image.load_img(img_path, target_size=(224, 224))
9
10 x = image.img_to_array(img)
11 x = np.expand_dims(x, axis=0)
12 x = preprocess_input(x)
13
14 preds = model.predict(x)
15 P = decode_predictions(preds)
16
17 for (i, (imagenetID, label, prob)) in enumerate(P[0]):
18     print("{}. {}: {:.2f}%".format(i + 1, label, prob * 100))
```

Transfer Learning


Input
(CONV * 2) => POOL
(CONV * 2) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(FC * 3) => SOFTMAX
Output Labels
Input
(CONV * 2) => POOL
(CONV * 2) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
Output Features
Original Layers
New FC Layers
Input
(CONV * 2) => POOL
(CONV * 2) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(FC * 3) => SOFTMAX
Output Labels
(FC * 3) => SOFTMAX
Old FC Layers


Input
(CONV * 2) => POOL
(CONV * 2) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(FC * 3) => SOFTMAX
Output Labels
Freeze Early Layers in Network
Only Train FC Layers


Input
(CONV * 2) => POOL
(CONV * 2) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(CONV * 3) => POOL
(FC * 3) => SOFTMAX
Output Labels
Unfreeze Early Layers & Train All

Como usar transfer learning

	Base Similar	Base diferente
Base pequena	Usar modelo treinado	Treinar mais camadas
Base grande	Treinar camadas FC	treinar próprio modelo

Código

- Transfer_learning_answer.ipynb