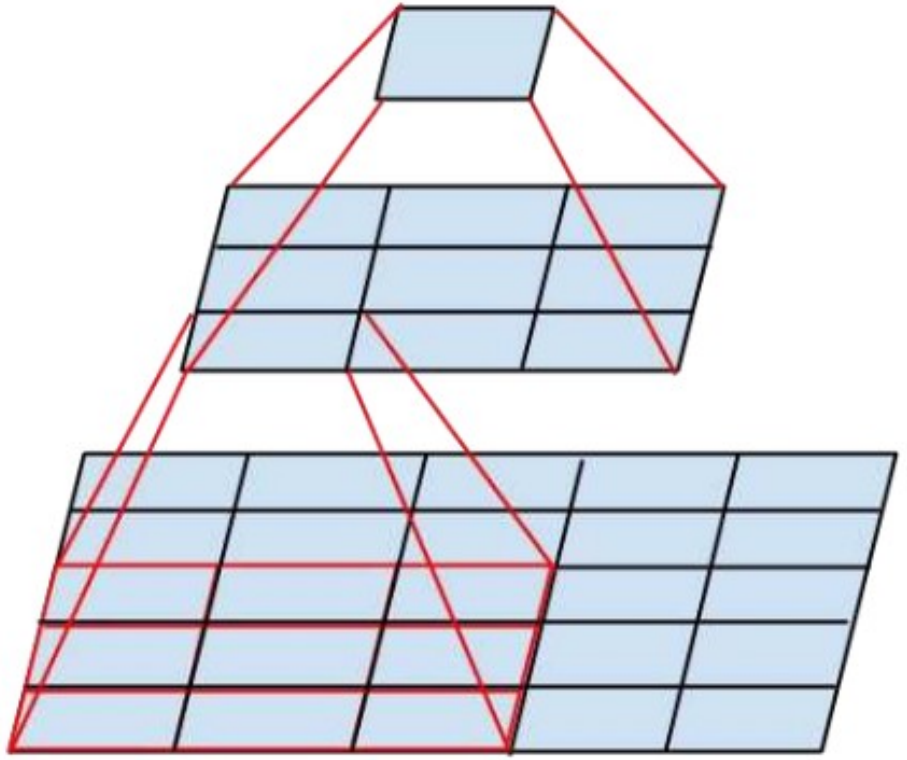
Архитектуры CNN или объять необъятное

CNN



CNN vs FCN

- CNN – свертка – фильтр, применяемый к каждому окну изображения
- CNN может работать с данными разной размерности
- CNN имеет на порядок меньше параметров, данные для фильтра это окна, поэтому данных очень много

CNN vs RNN

- Естественно для D > 1
- Параллельная обработка всего сигнала – нет состояния – только между слоями
- Ограничение на “receptive field” решается за счет down/up scaling и dilation

CNN+

- MaxPooling
- AvgPooling
- LRN – Local Response Normalization
- Deconvolution/TransposedConvolution
- LocallyConnected

Параметры свертки

- kernel
- stride
- padding
- rate (dilation)

Kernel

Conv<N>D(nchannels, kernel=(m,...,n))

Ядро это тензор

- Размерность?
- Количество параметров?

Kernel

Conv<N>D(nchannels, kernel=(m,...,n))

Ядро это тензор

- Размерность N+2: nchannels, m,...,n, src_channels
- Количество параметров: nchannels*m*...*n* src_channels

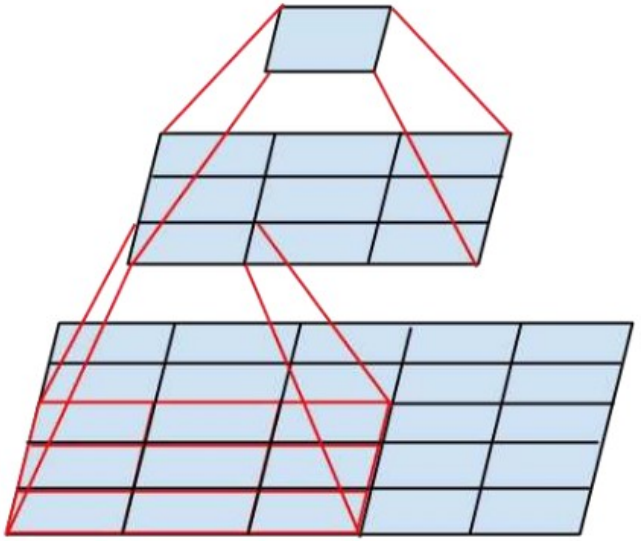
например для фильтра 3*3: 64*3*3*64=36864 параметров

для изображения 128*128: 128*128*36864=6e8 умножений

Факторизация ядра

Пространственная:

conv5x5 -> conv3x3 conv3x3



5*5 = 25

3*3+3*3 = 18

меньше на 28%

Чем мы за это платим?

Факторизация ядра

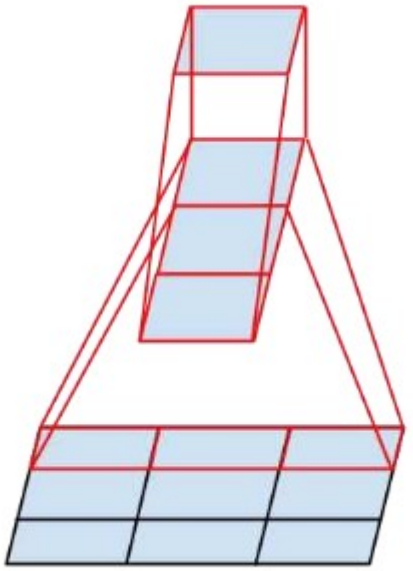
Анизотропная:

conv3x3 -> conv1x3 conv3x1

3*3 = 9

3*1+1*3 = 6

меньше на 30%



Факторизация ядра

Анизотропная:

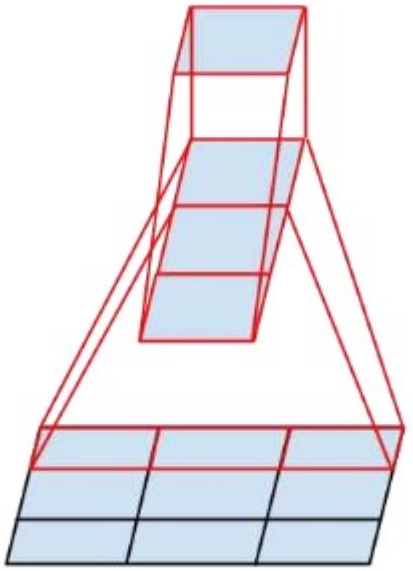
conv3x3 -> conv1x3 conv3x1

3*3 = 9

3*1+1*3 = 6

меньше на 30%

Что еще можно факторизовать?



Факторизация ядра

Поканальная depthwise:

convNx3x3xN -> N*conv1x3x3x1

меньше в N раз

См MobileNet

Факторизация ядра

Абстрактная – факторизуем сам тензор:

kernel_Nx3x3xN

->kernel_NxM * kernel_M_x3x3xN

M < N

Получаем NiN – Network in Network:

model.add(Conv2d(M,3,3)))

model.add(Conv2d(N, (1,1)))

Факторизация ядра

Поканальная depthwise:

convNx3x3xN -> N*conv1x3x3x1

меньше в N раз

Как еще можно факторизовать?

Факторизация ядра

Абстрактная – факторизуем сам тензор:

kernel_Nx3x3xN

→ kernel_NxM * kernel_M_x3x3xN

M < N

Получаем NiN – Network in Network:

model.add(Conv2d(M, (3,3)))

model.add(Conv2d(N, (1,1)))

Также используется в ResNet

Факторизация ядра

Абстрактная – факторизуем сам тензор:

kernel_Nx3x3xN

→ kernel_N_x3x3xM * kernel_MxN

M < N

model.add(Conv2d(M, (1,1)))

model.add(Conv2d(N, (3,3)))

Используется в YOLO (darknet), Inception, ResNet

Факторизация ядра

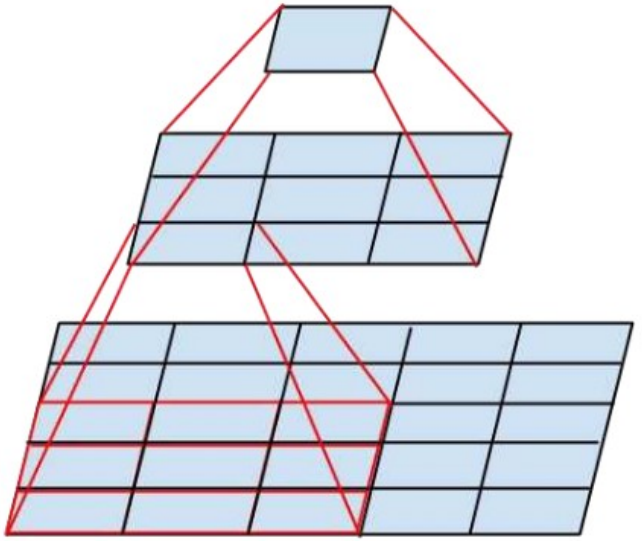
В каком случае меньше параметров?

kernel_Nx3x3xN

→ kernel_N_x3x3xM * kernel_MxN

→ kernel_NxM * kernel_M_x3x3xN

Receptive Field



Receptive Field

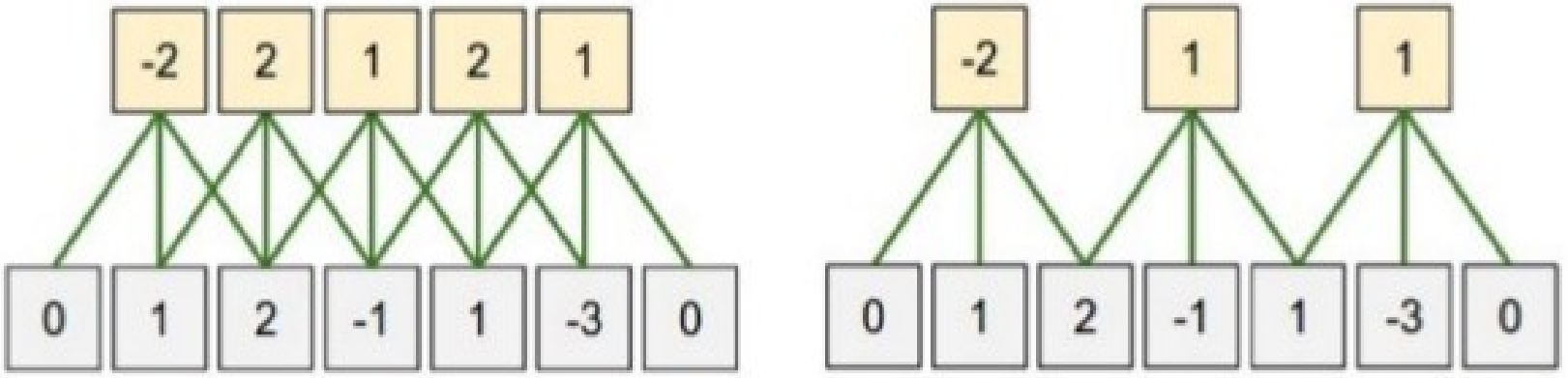
- Сколько пикселей исходного изображения “видит” нейрон N-слойной сверточной сети 3x3?

Receptive Field

- Сколько пикселей исходного изображения “видит” нейрон N-слойной сверточной сети 3x3?
- Всего N+2 x N+2
- Для слона потребуется очень много слоев

RF как увеличить?

Stride – будем сдвигать окно не на 1, а на 2

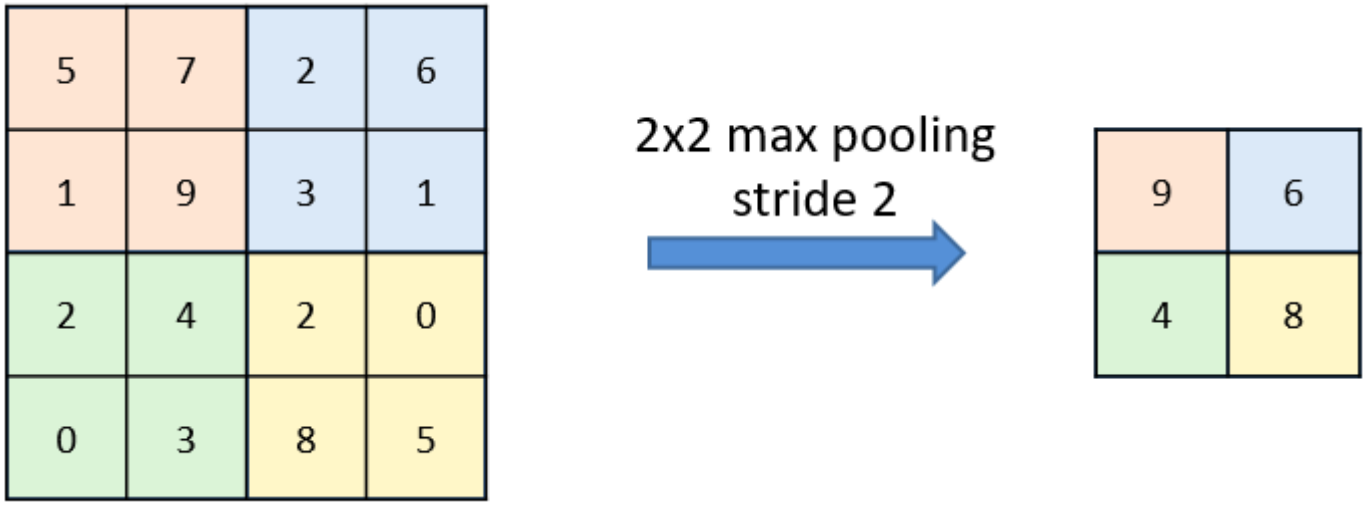


\+ Понижаем разрешение

\- Пропускаем высокочастотный сигнал

RF как увеличить?

MaxPooling



+ Shift invariance

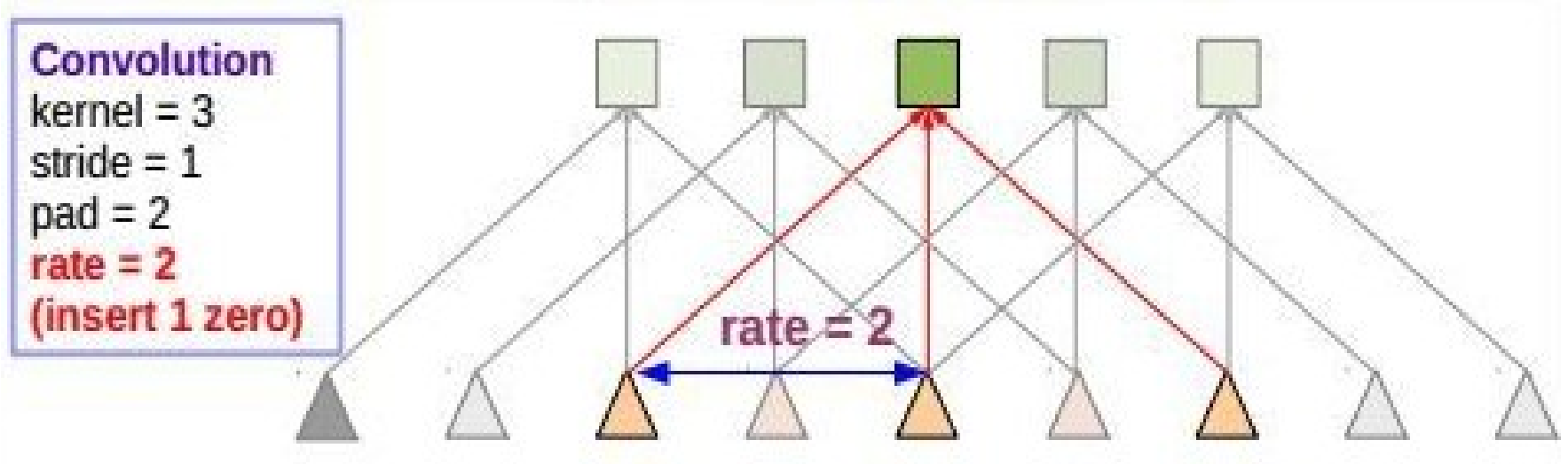
- Выкидываем ¾ вычислений предыдущего слоя

RF как увеличить?

Если нежелательно понижать разрешение?

dilation он же atrous он же rate

В ядро даем не соседние пикселы, а через N



RF dilation

Какой receptive field у N слоев сверток с dilation=2 ?

RF dilation

Какой receptive field у N слоев сверток с dilation=2 ?

2*N

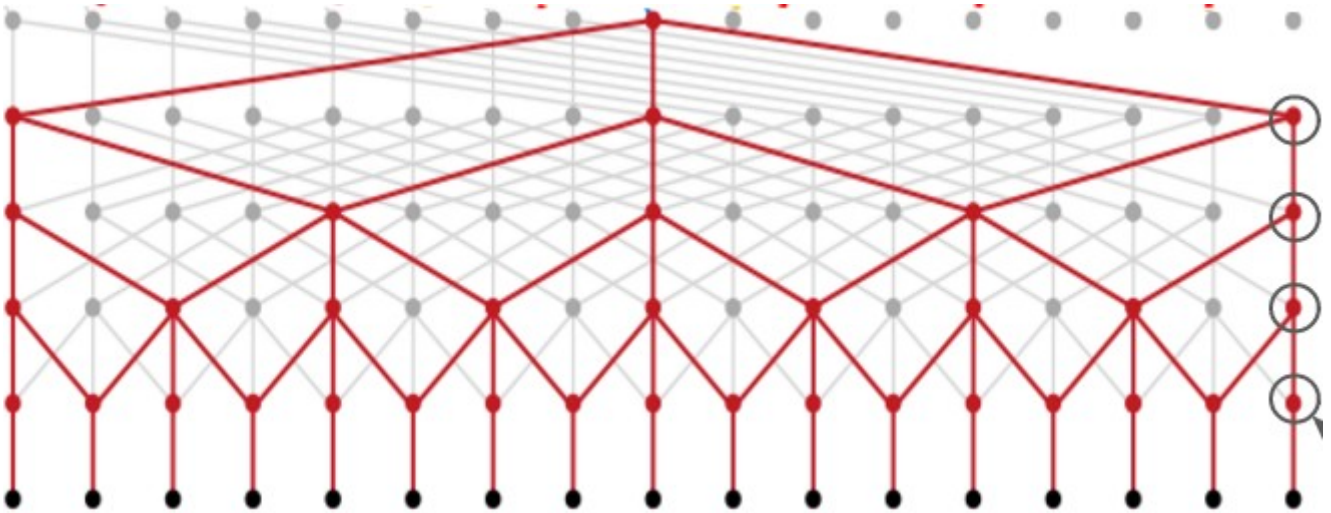
Можно ли больше?

RF dilation

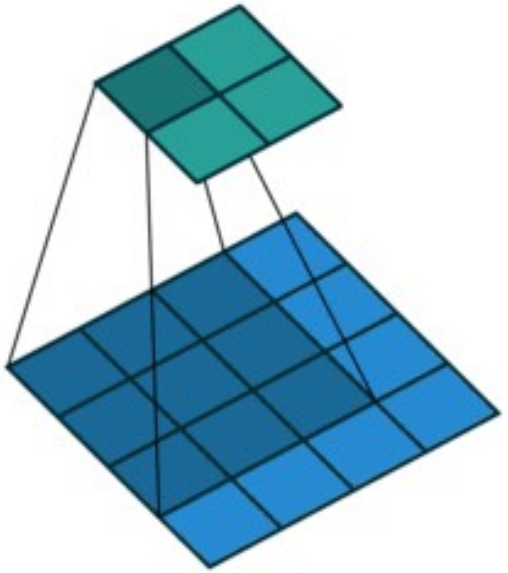
Какой receptive field у N слоев сверток с dilation=2 ?

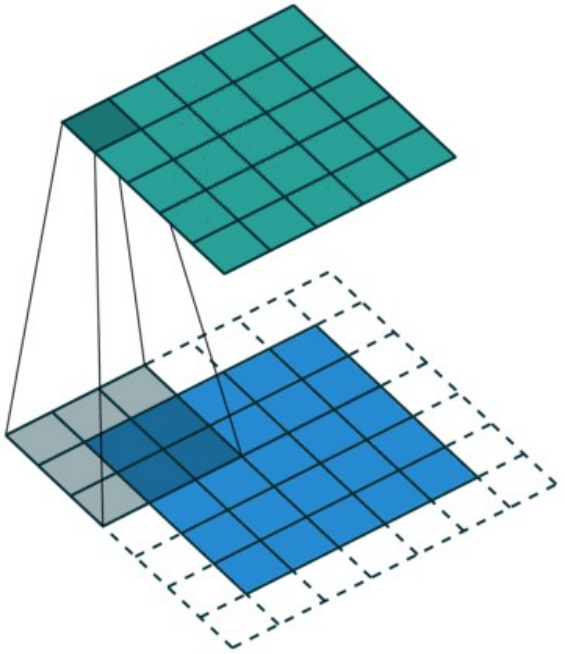
2*N

Экспоненциально увеличим dilation 2,4,8,16...



Немного про padding





valid – понижает размер на kernel-1

same – дополняет нулями (бывают граничные эффекты)

Activations

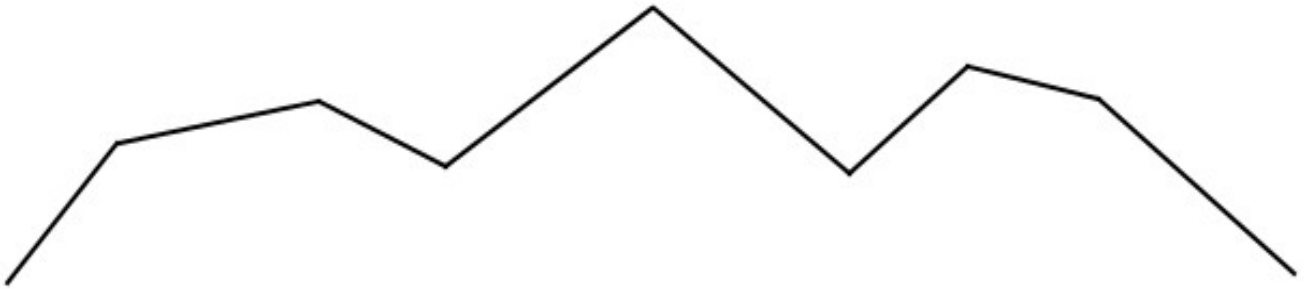
Relu

Leaky Relu, PRelu

Elu

Selu

Relu



1) Любая кусочно линейная функция представима так:

$\sum_i w_i * \max(x + b_i, 0)$

2) sparsity

3) grad(relu) = 1

Relu

Недостатки:

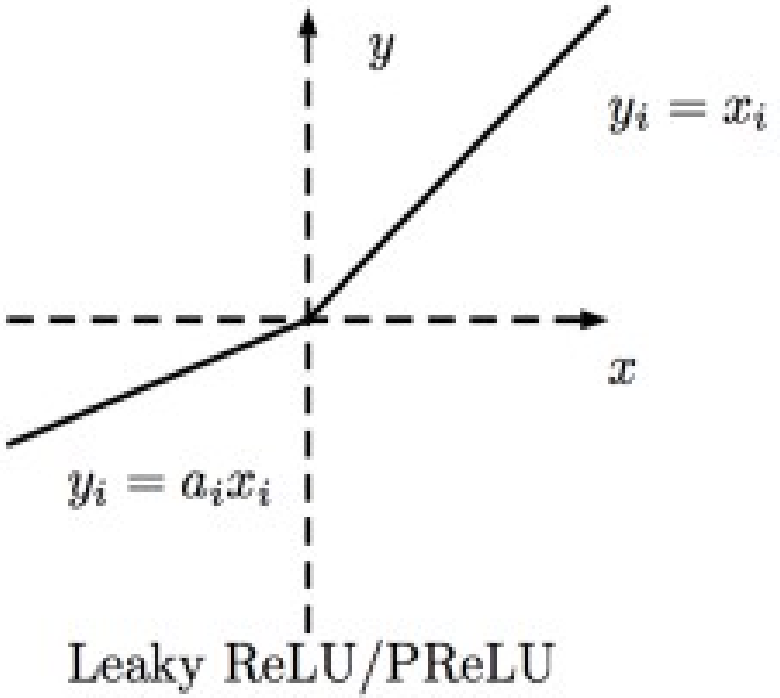
grad(relu) = 0

Dying: relu Wx

- Где x > 0 (предыдущий relu или вход)
- Если случится W < 0 (инициализация или неудачный шаг градиента), то:

  grad(Wx) = 0 → W < 0 навсегда

Leaky Relu, PRelu



Elu/Selu

1) mean(f(Wx)) = 0
Srelu = max(x, -1)

2) неограниченный отрицательный хвост – чувствительность к шуму

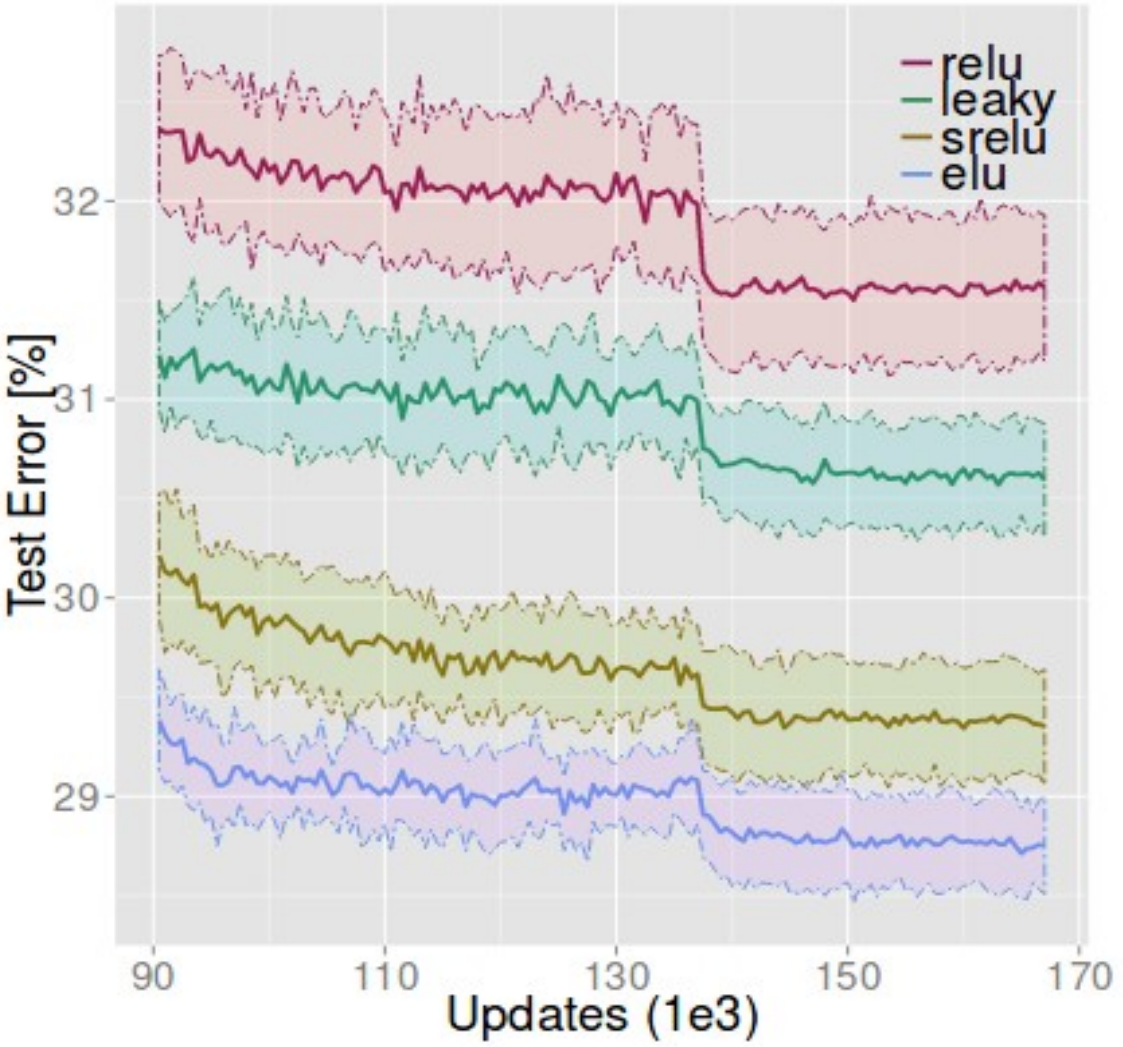
Elu = x|x > 0; exp(x)-1|x < 0

3) var(f(Wx)) = 1

Selu = ax|x > 0; b(exp(x)-1)|x < 0

BatchNorm почти не нужен

Lrelu,Srelu,Elu



Архитектуры CNN

Архитектуры CNN

Alexnet

VGG

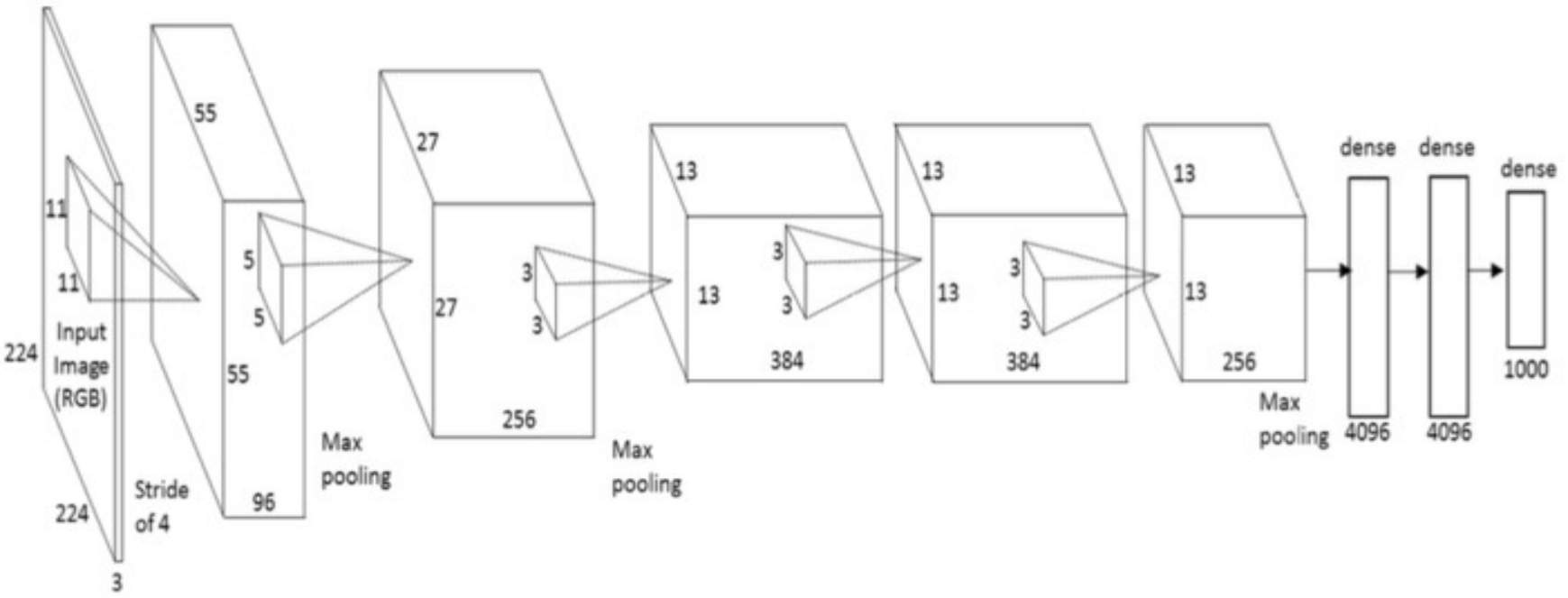
Darknet

Inception

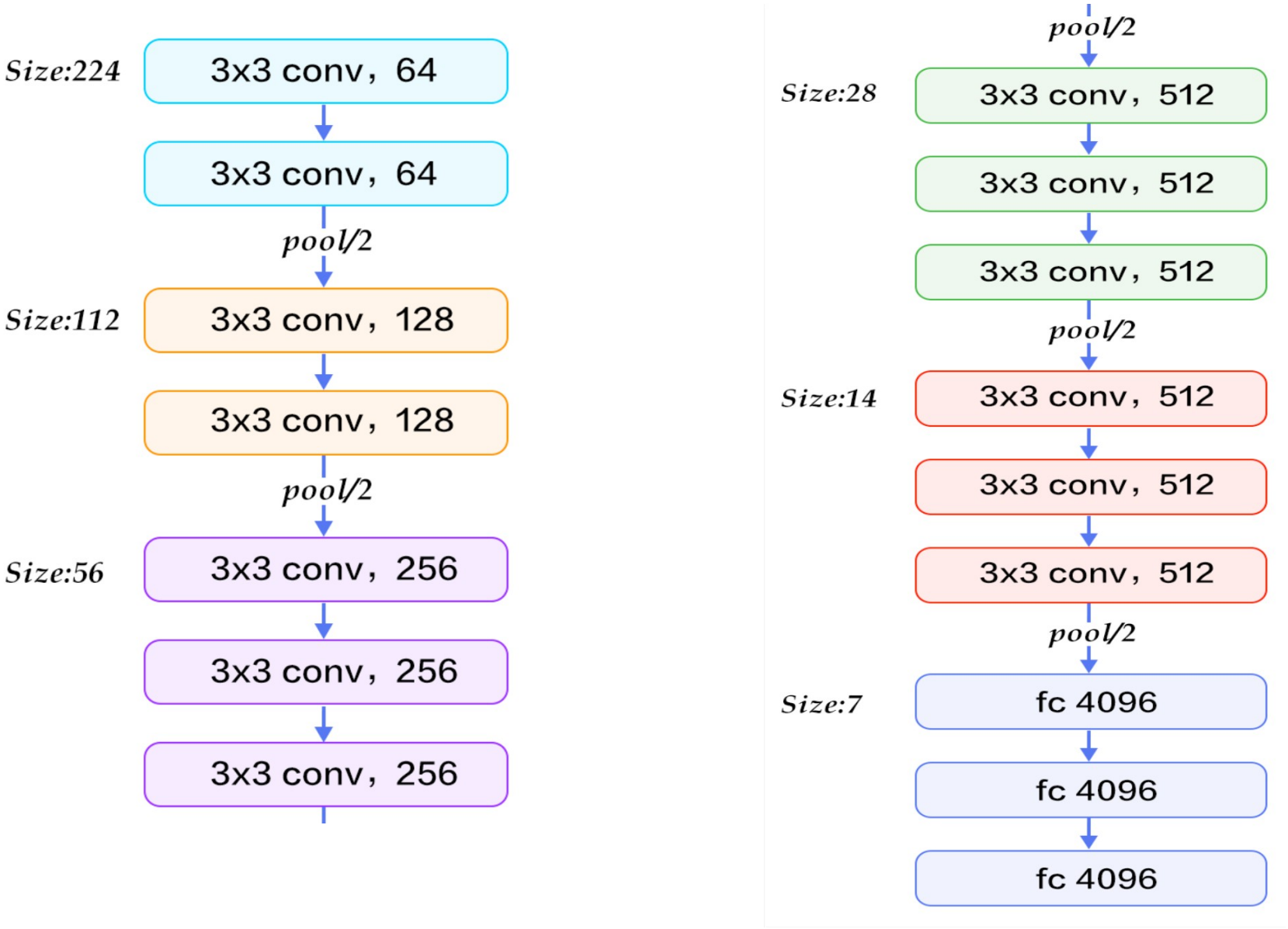
ResNet

Unet

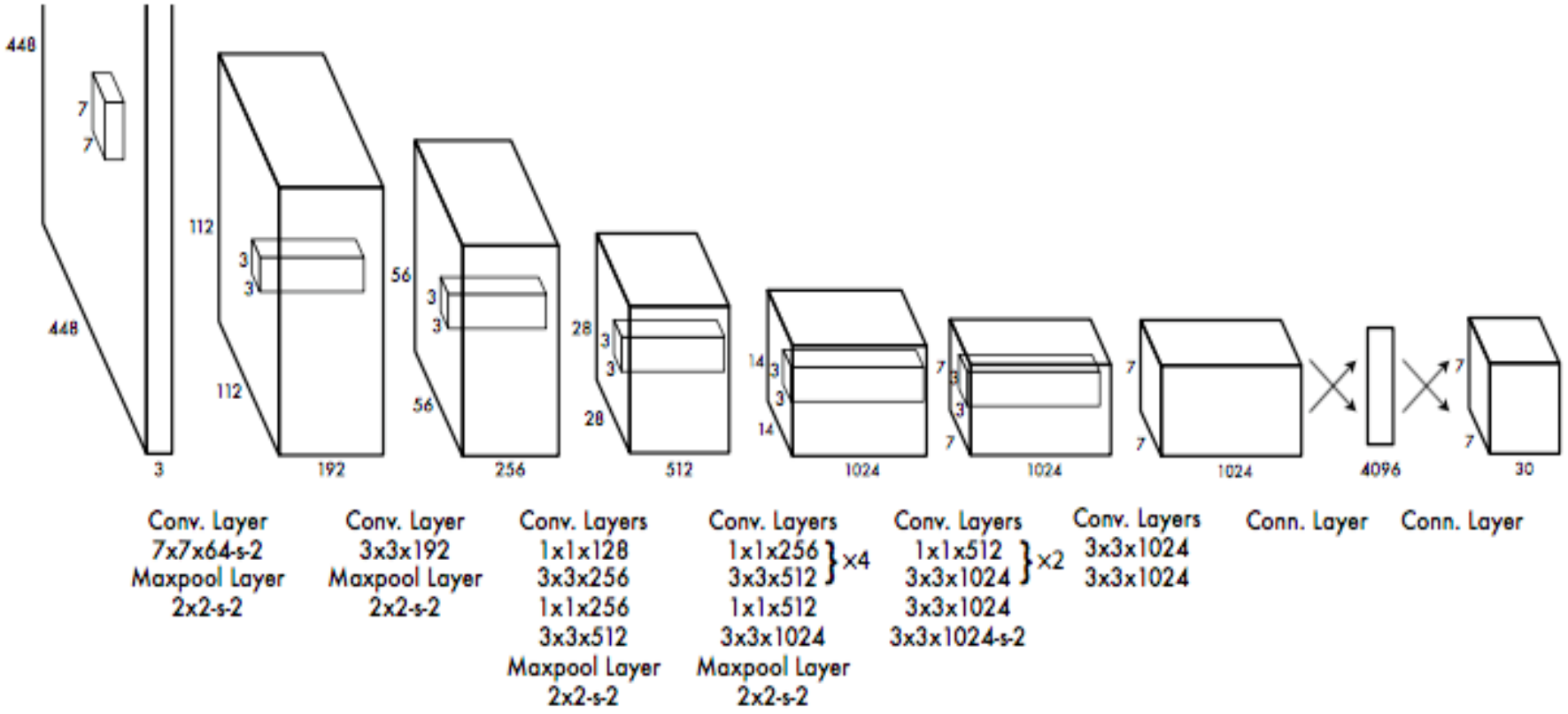
Alexnet


224
Input
Image
(RGB)
224
3
11
11
Stride
of 4
55
55
96
5
5
Max
pooling
27
27
256
3
3
Max
pooling
13
13
384
3
3
13
13
384
3
3
13
13
256
Max
pooling
dense
4096
dense
4096
dense
1000

VGG


Size:224
3x3 conv，64
3x3 conv，64
pool/2
Size:112
3x3 conv，128
3x3 conv，128
pool/2
Size:56
3x3 conv，256
3x3 conv，256
3x3 conv，256
pool/2
Size:28
3x3 conv，512
3x3 conv，512
3x3 conv，512
pool/2
Size:14
3x3 conv，512
3x3 conv，512
3x3 conv，512
pool/2
Size:7
fc 4096
fc 4096
fc 4096

Yolo

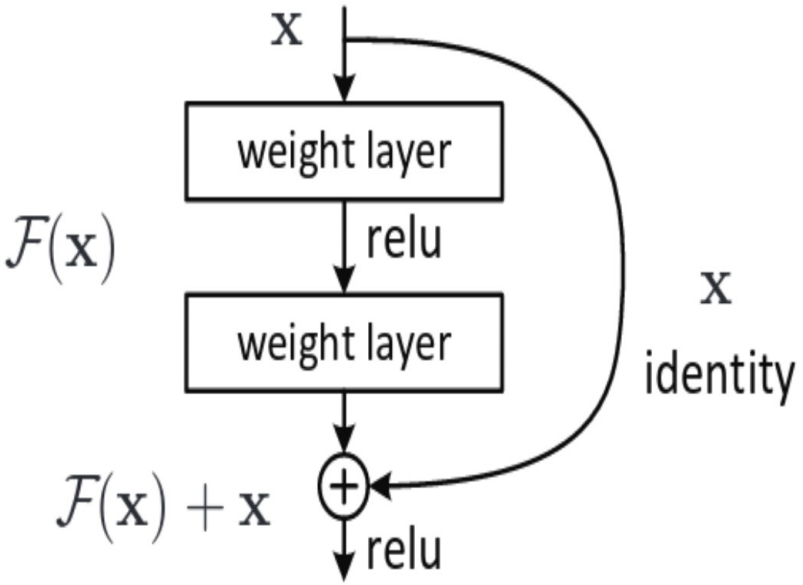

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer

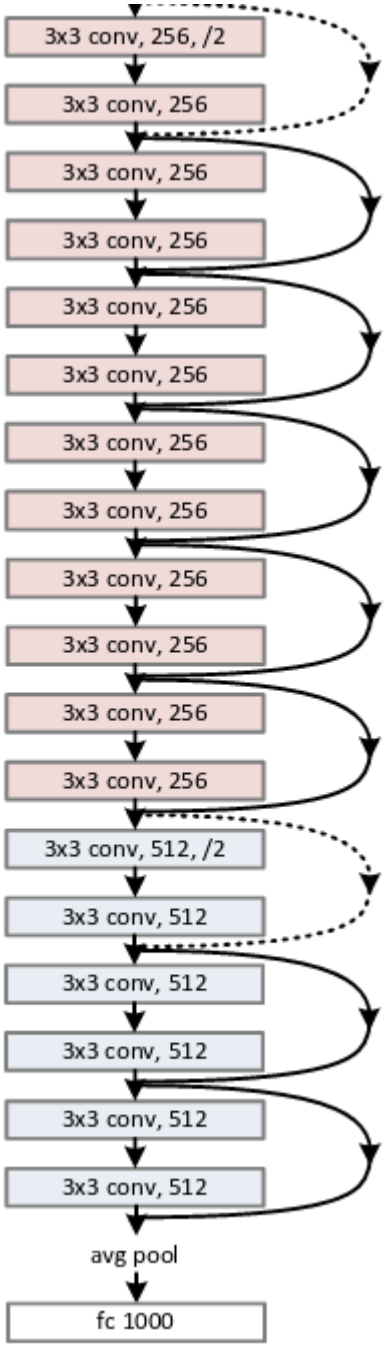
Inception

Несколько идей:

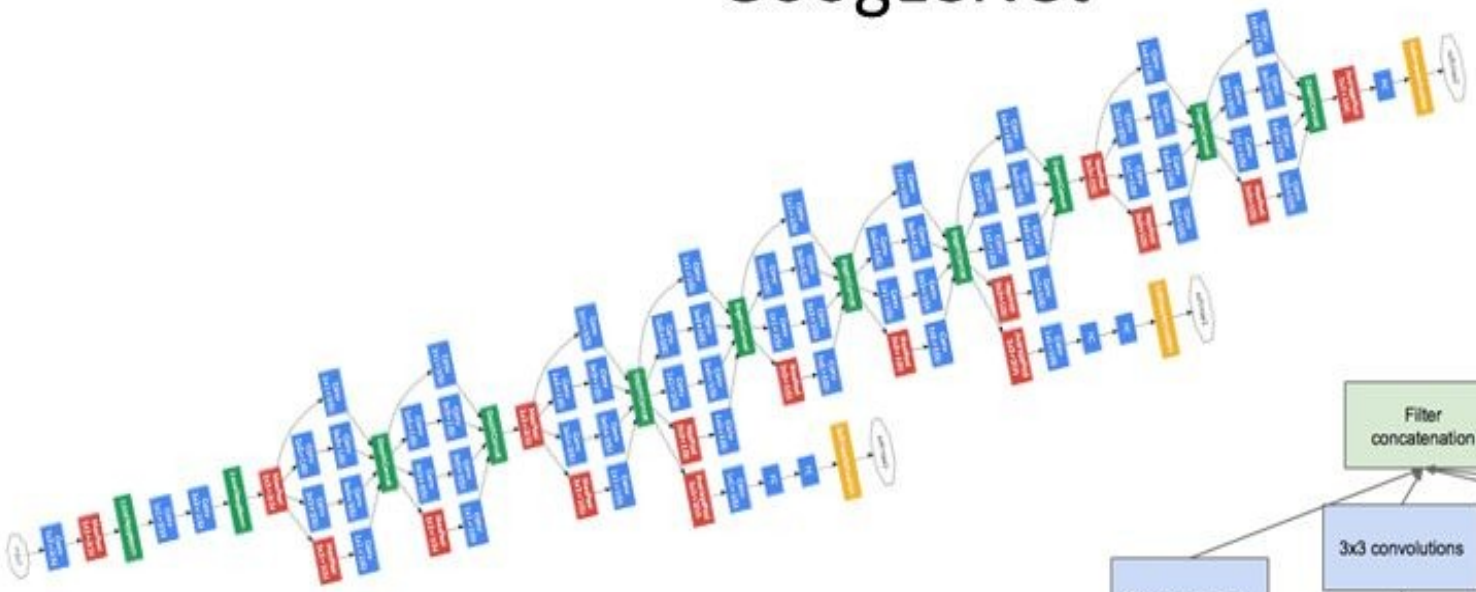
- Делим ядро на 1x1, 3x3, 5x5
- Факторизуем входные каналы с помощью 1x1
- Параллельно с пулингом
- Несколько классифицирующих выходов для протаскивания градентов на нижние слои

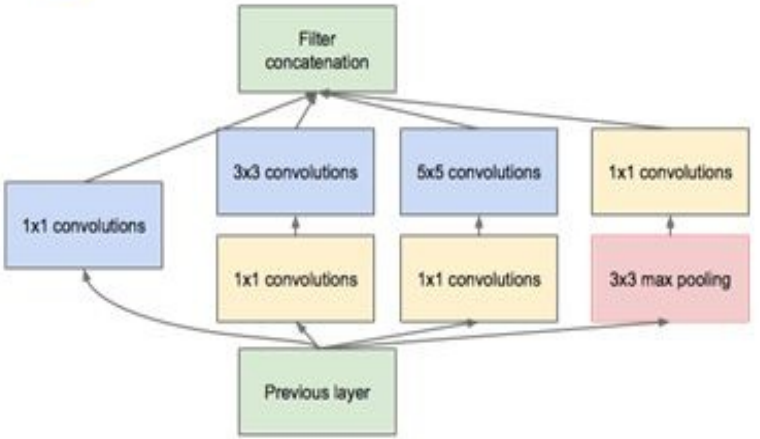
ResNet


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu


3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

GoogLeNet

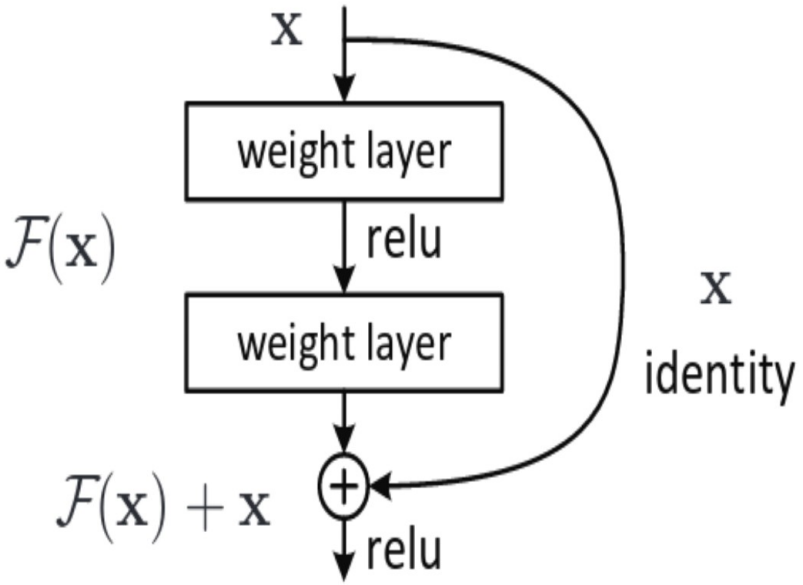


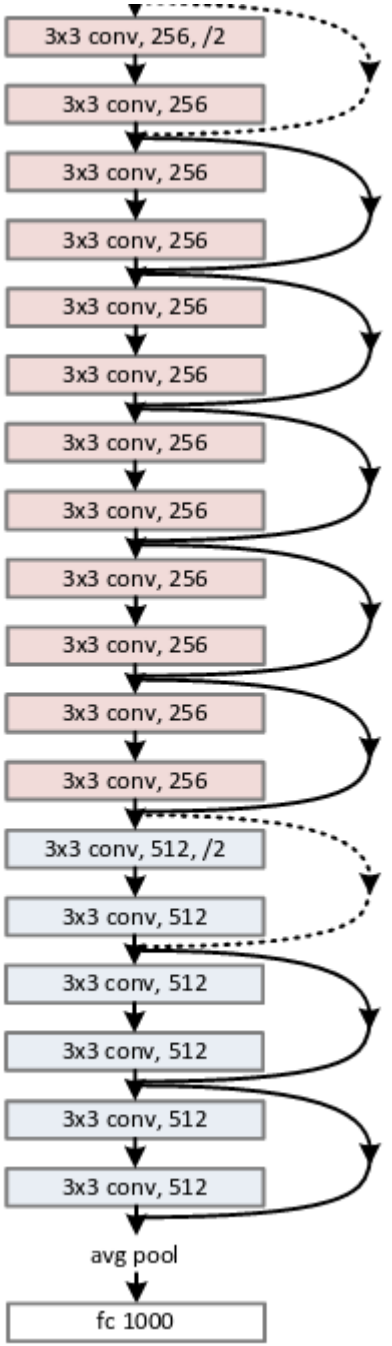


"Inception" module

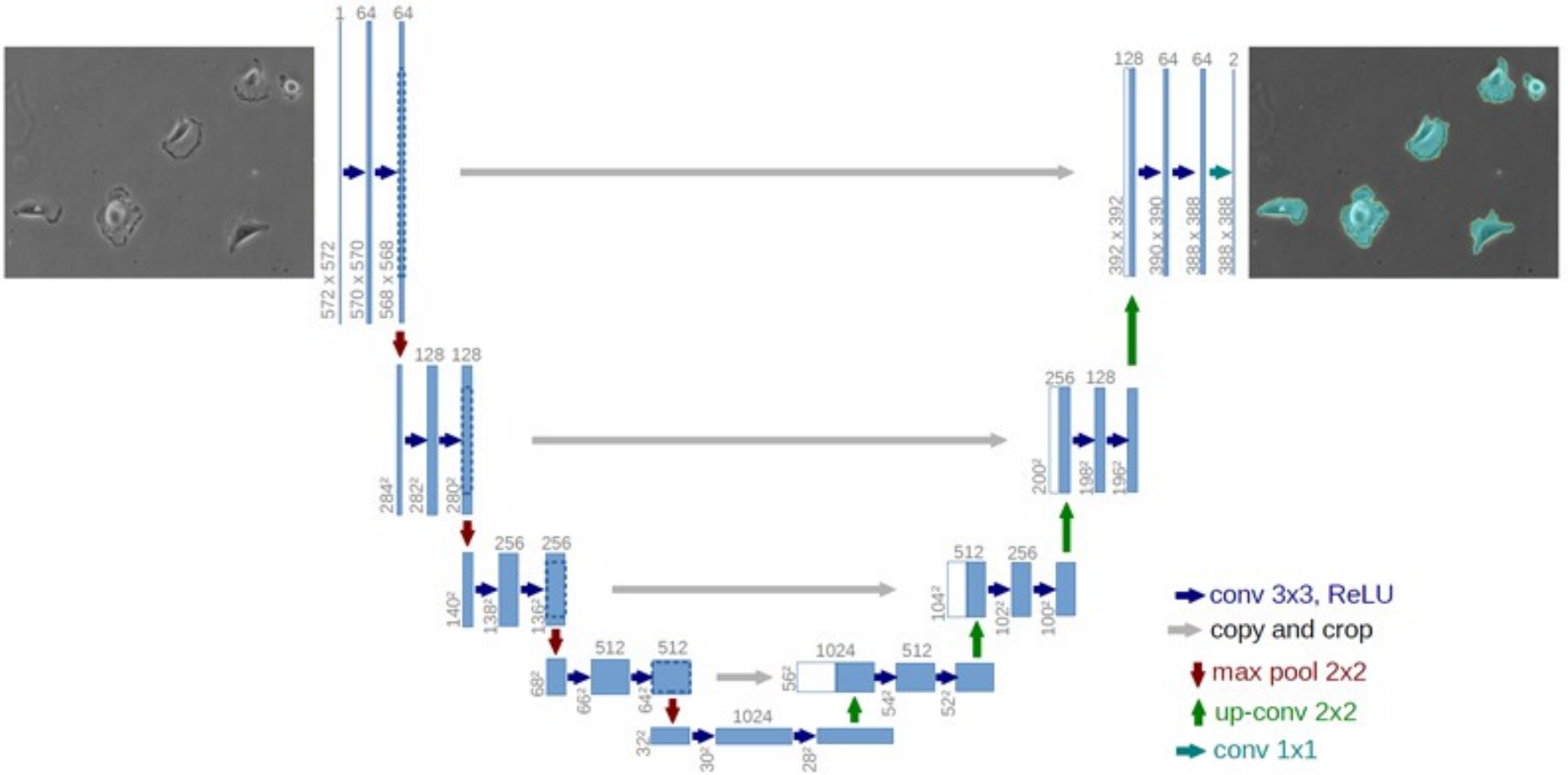
- composition of multi-scale dimension-reduced "Inception" modules
- 1x1 conv for dimensionality reduction
- concatenation across filter scales
- multiple losses for training to depth

ResNet


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu


3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Unet


1 64 64
572 x 572
570 x 570
568 x 568
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
1024
56²
54²
512
52²
1024
30²
28²
512 256
104²
102²
100²
256 128
200²
198²
196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Что еще?

ResNeXt, MobileNet – Depthwise факторизация

WaveNet, ByteNet – Dilation

GCNN – gated convolutions