Глубокое обучение

Свёрточные сети

План лекции

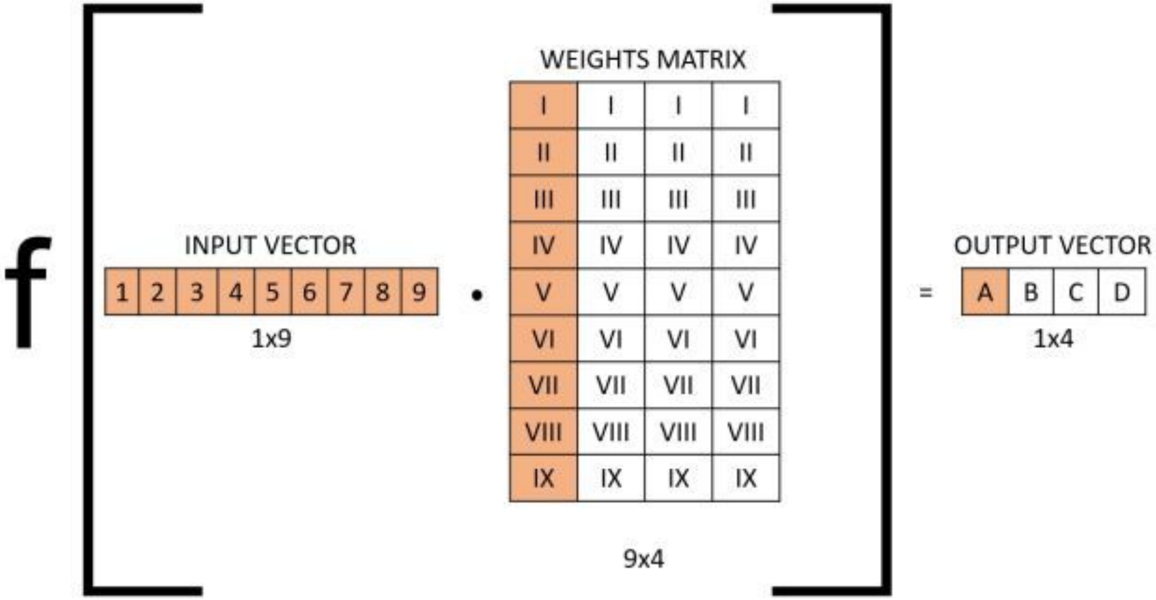
- Свёрточный слой
    - Зачем
    - Идея
    - Параметры
    - im2col
    - Receptive field
- Pooling
- Датасеты картиночной классификации
- Архитектуры свёрточных нейросетей
    - LeNet
    - AlexNet
    - VGG
    - Inception
    - ResNet
    - MobileNet

Зачем

- количество параметров

В первом линейном слое `HxWxCxCout` параметров.
При этом если `Cout` сделать маленьким, то качество будет не очень, особенно на больших картинках типа 1980x1080.
Если большим, то перепараметризуем сетку, т.е. столкнёмся с переобучением, сложной оптимизацией и т.д.

Linear layer



Зачем

- количество параметров
- структура данных никак не учитывается (хотим устойчивости относительно простых изменений)

(собачка)

Зачем

- количество параметров
- структура данных никак не учитывается (хотим устойчивости относительно простых изменений)

(собачка)

Зачем

- количество параметров
- структура данных никак не учитывается (хотим устойчивости относительно простых изменений)

(собачка)

(тоже собачка)

Зачем

- количество параметров
- структура данных никак не учитывается (хотим устойчивости относительно простых изменений)

(собачка)

(тоже собачка)

(и тут)

Зачем

- количество параметров
- структура данных никак не учитывается (хотим устойчивости относительно простых изменений)

(собачка)

(тоже собачка)

(и тут)

(снова она)

Зачем

- количество параметров
- структура данных никак не учитывается (хотим устойчивости относительно простых изменений)

(собачка)

(тоже собачка)

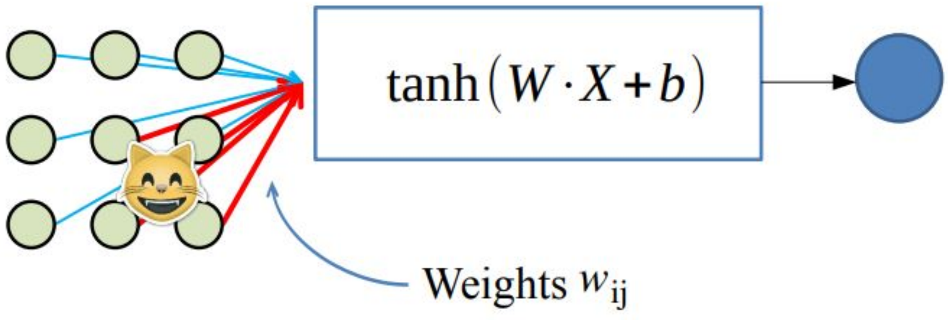
(и тут)

(снова она)

(и даже тут)

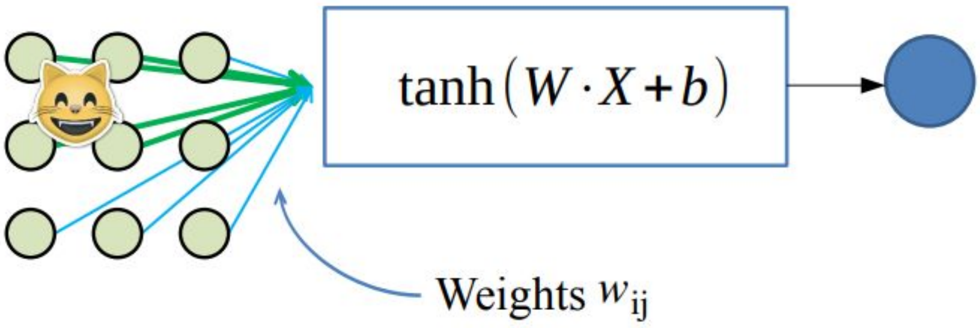
Начнём с последней проблемы

Motivation image recognition



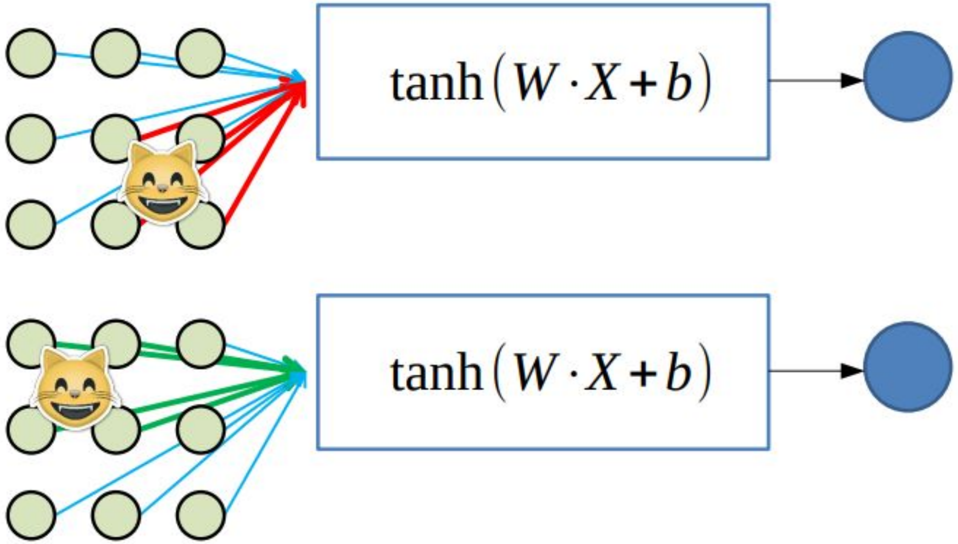
On this object, you will train red weights to react on cat face

Motivation image recognition



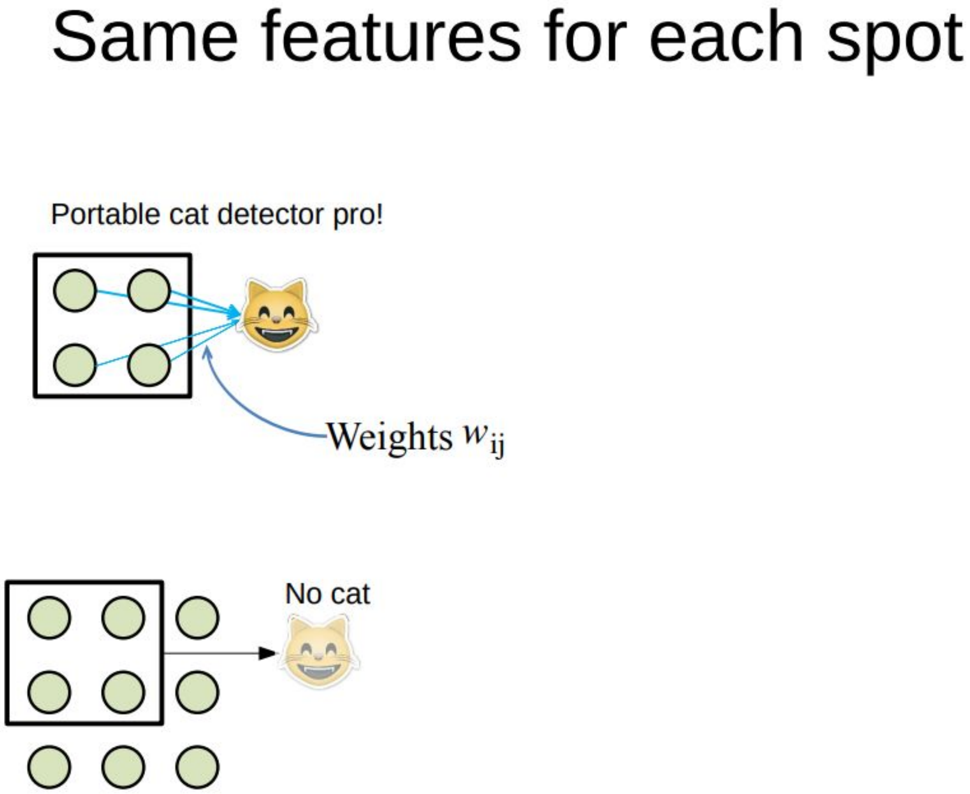
On this object, you will train green weights to react on cat face

Motivation image recognition

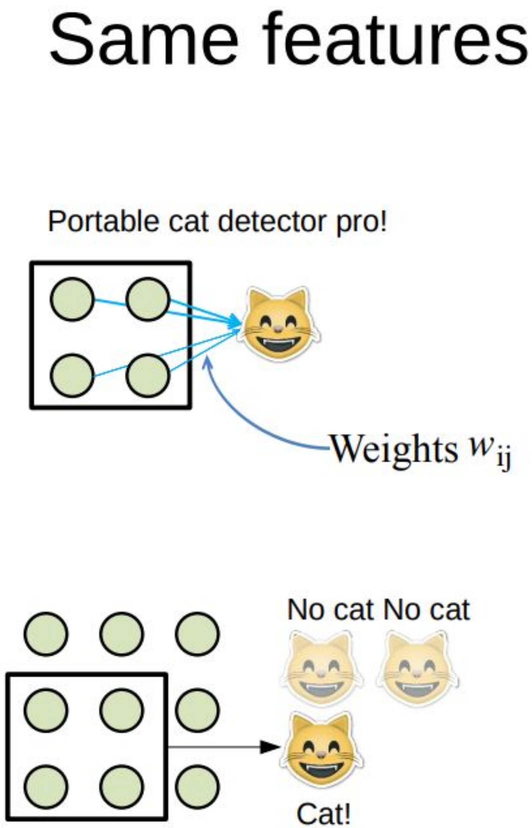


You network will have to learn those two cases separately!
Worst case: one neuron per position. 1

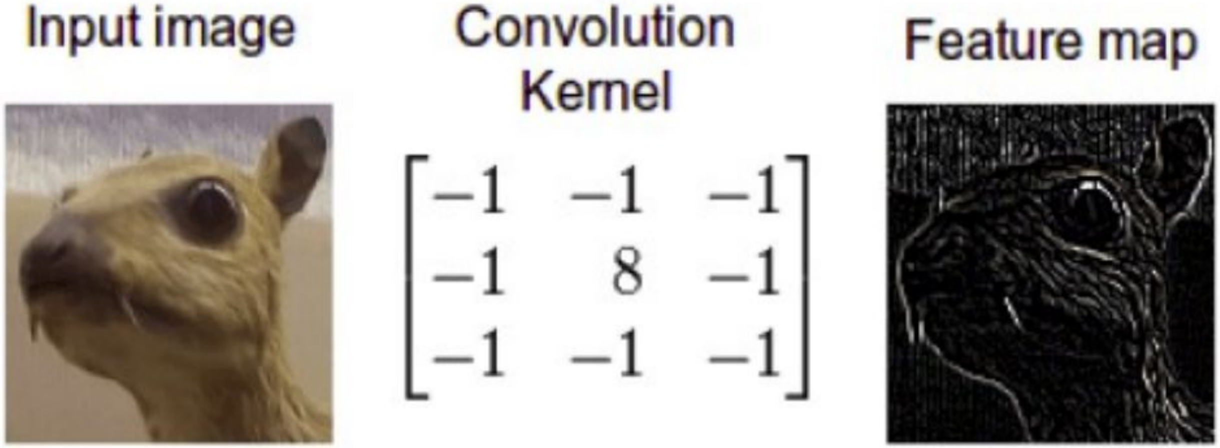
Motivation image recognition



Motivation image recognition

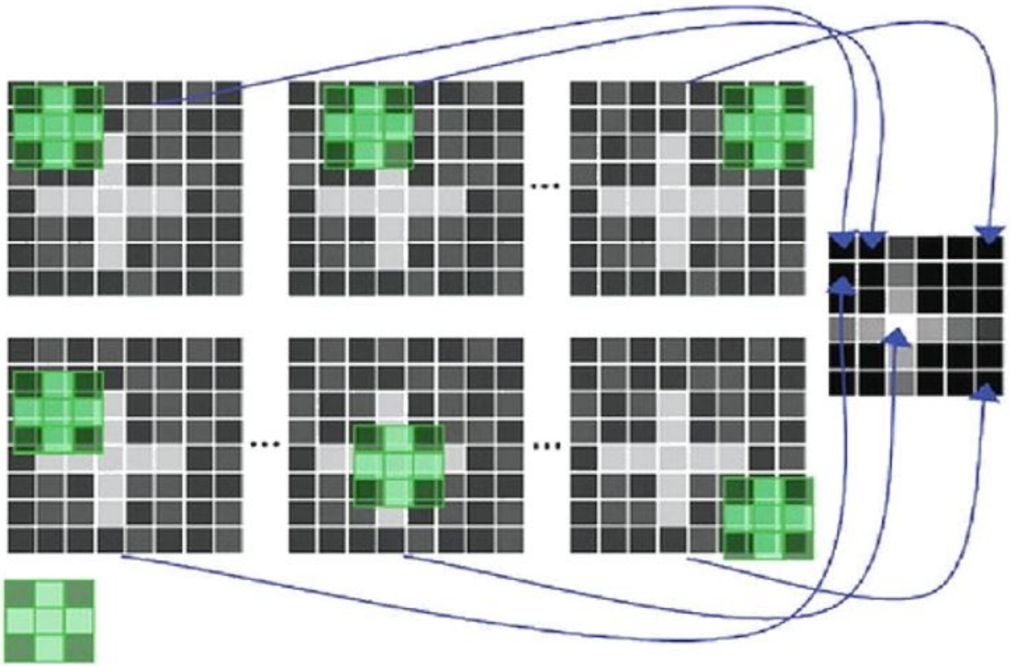


Convolution



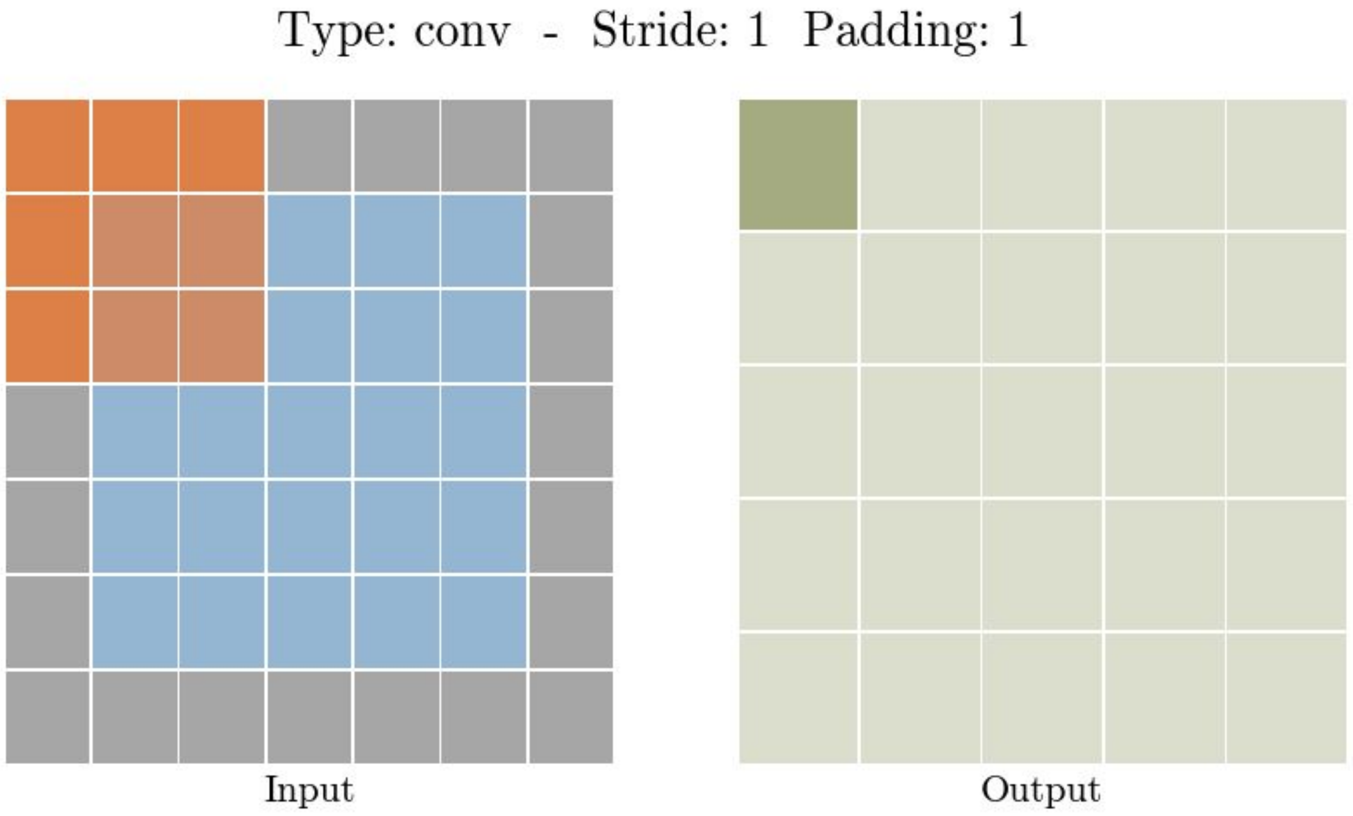
Intuition: how **edge-like** is this square?

Convolution



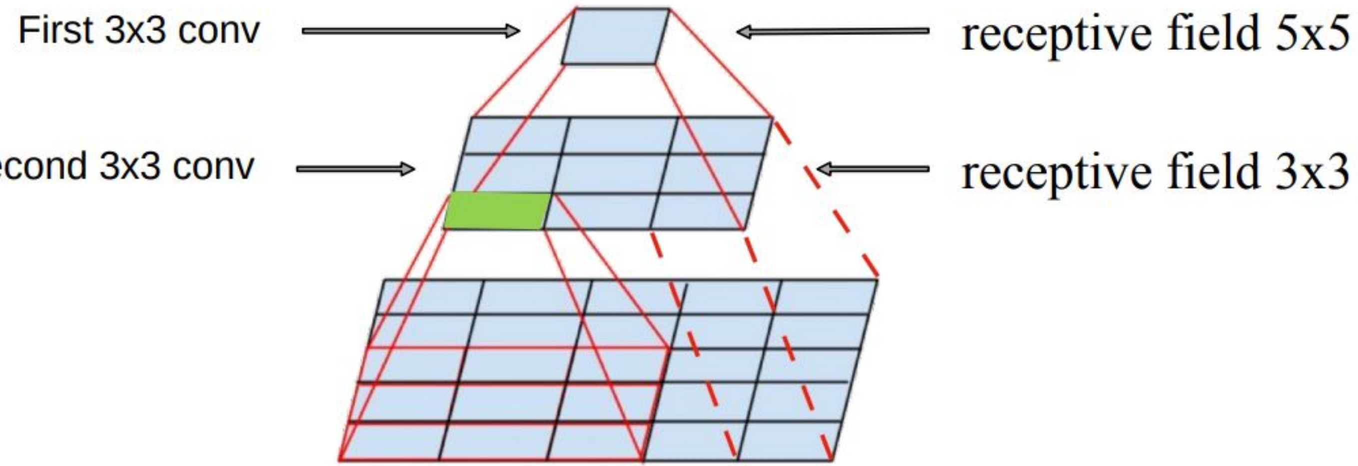
apply same filter to all patches

Convolution



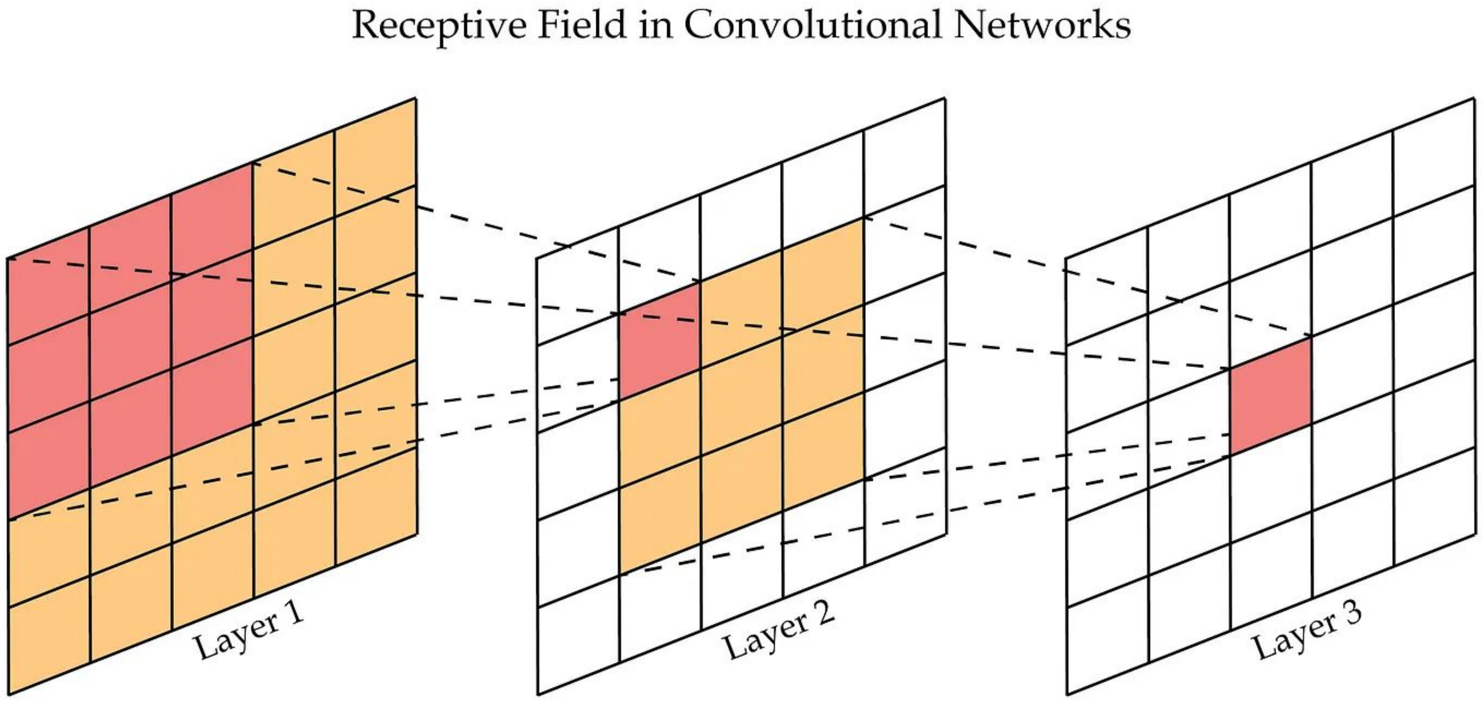
link

Receptive field

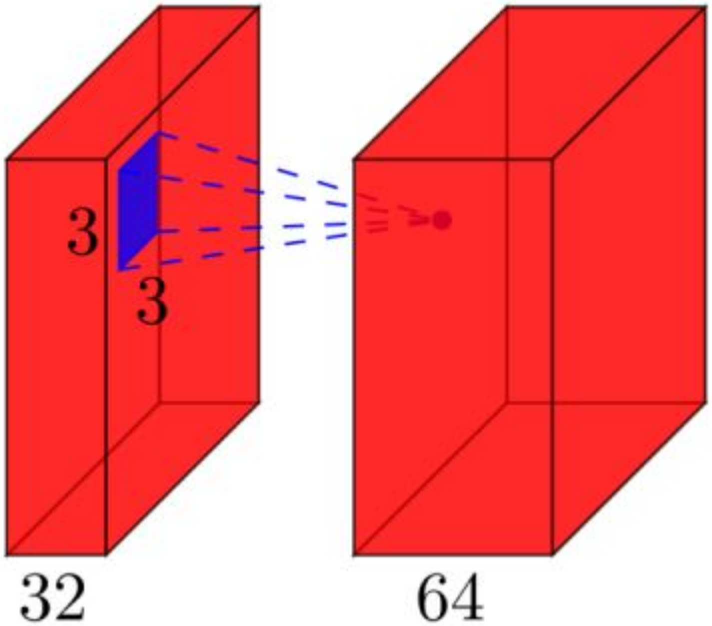


We can recognize larger objects by stacking several small convolutions!

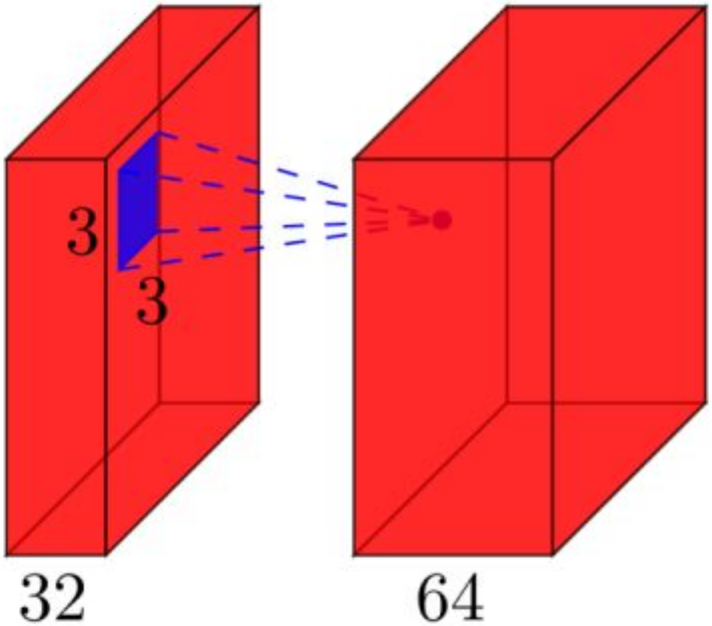
Receptive field



Num of params?



Num of params?



(3 * 3 * 32 +1) * 64

Where

3 * 3 - kernel_size

32 - in_channels

1 - bias

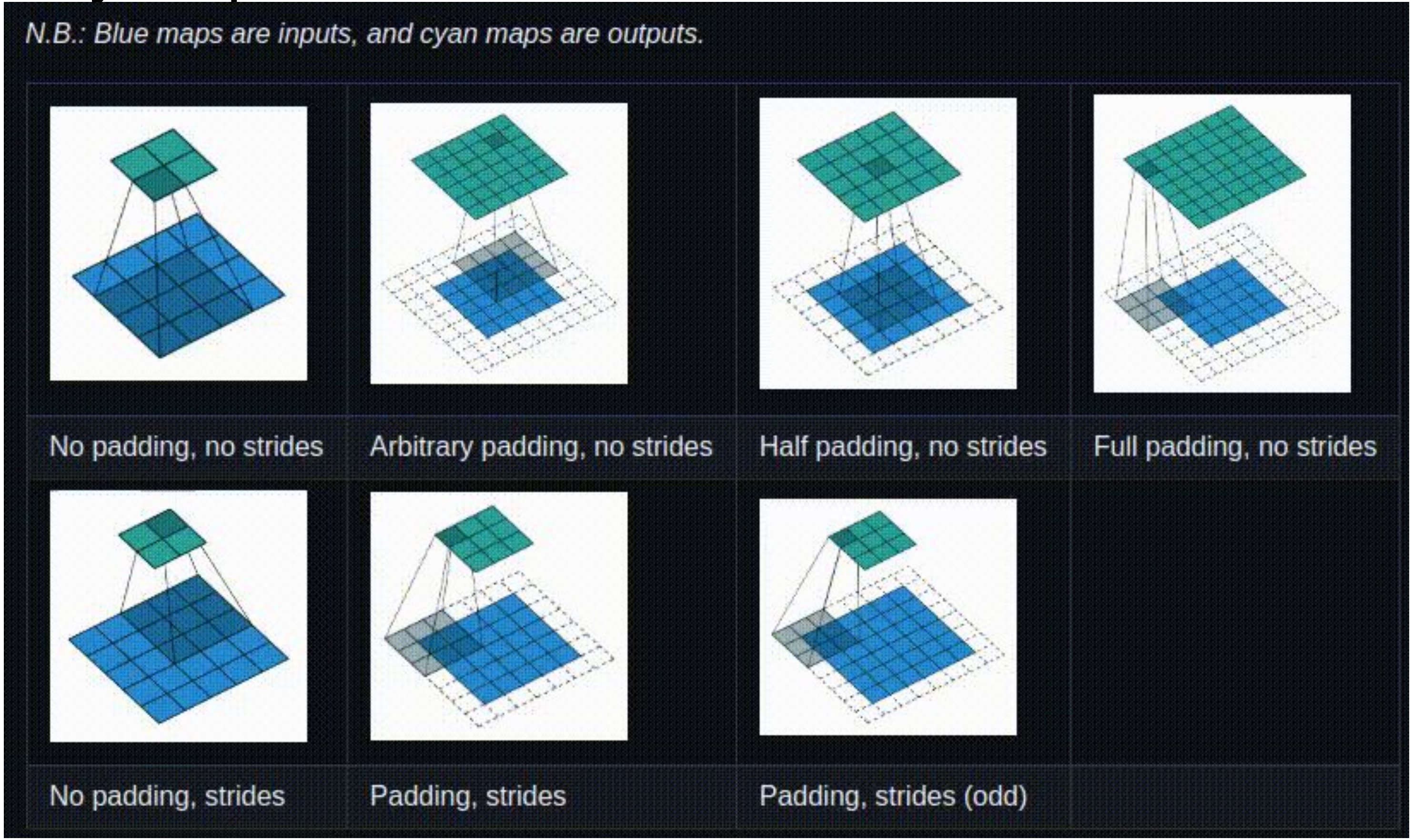
64 - out_channels

Conv layers params

in_channels
out_channels
kernel_size
stride
padding
dilation
bias

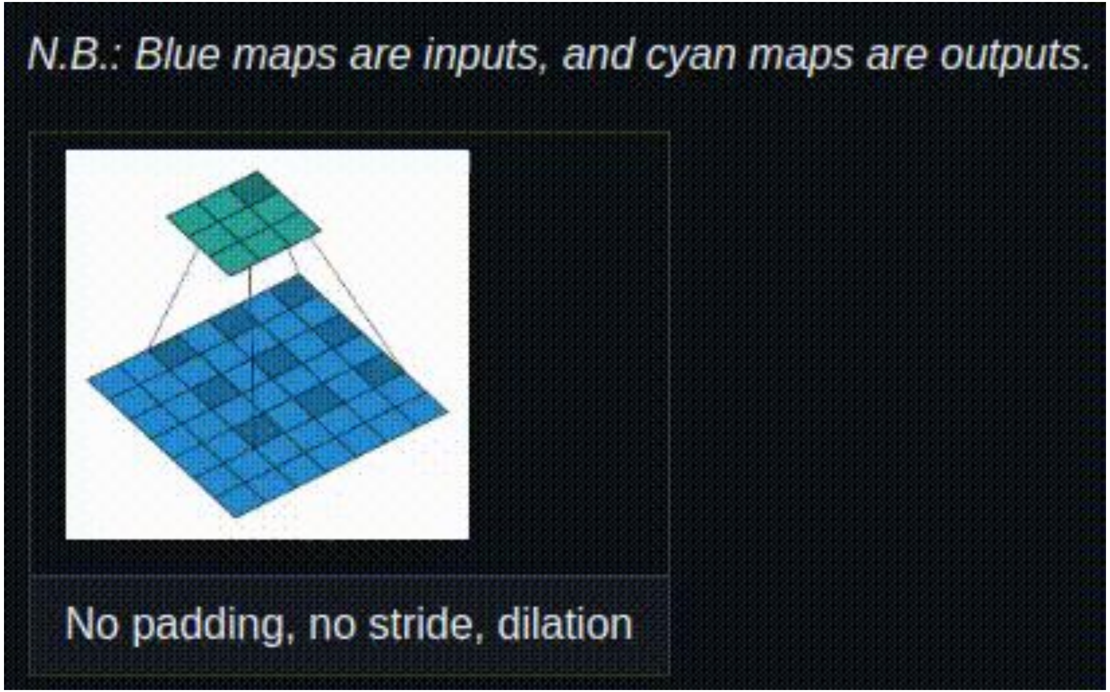
link

Conv layers params

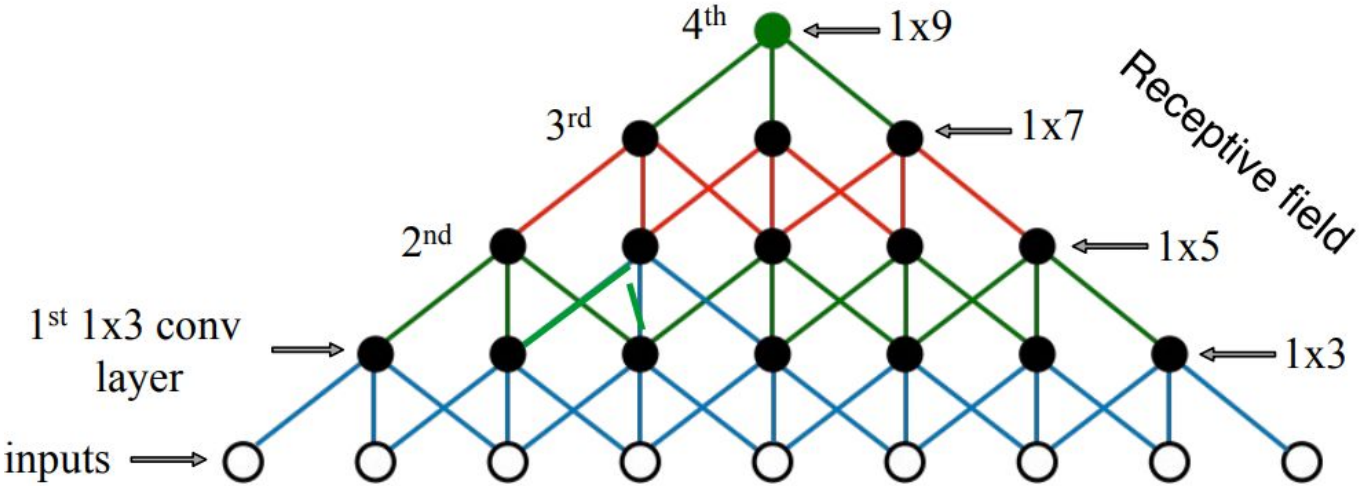


[link]

Dilated conv laye paramsr

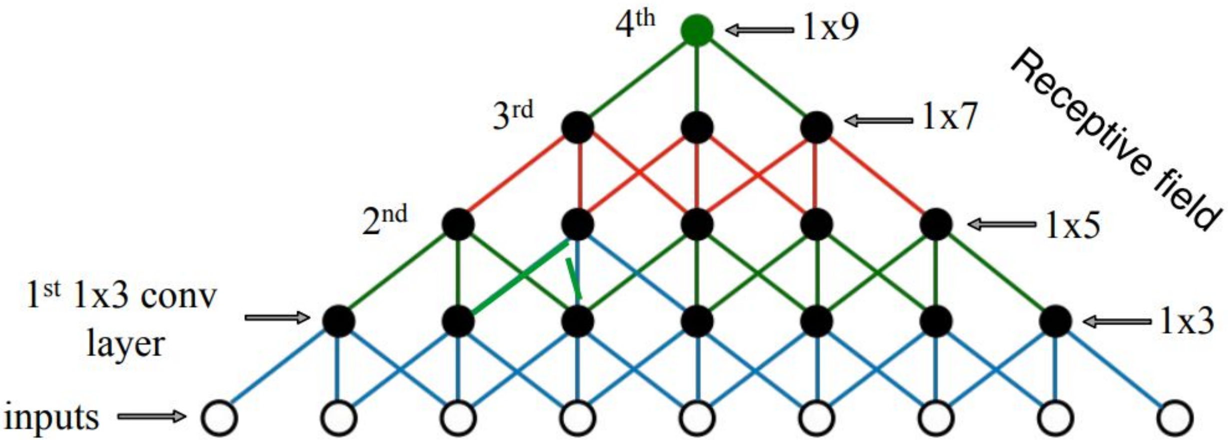


Receptive field



Q: how many 3x3 convolutions we should use to recognize a 100x100px cat

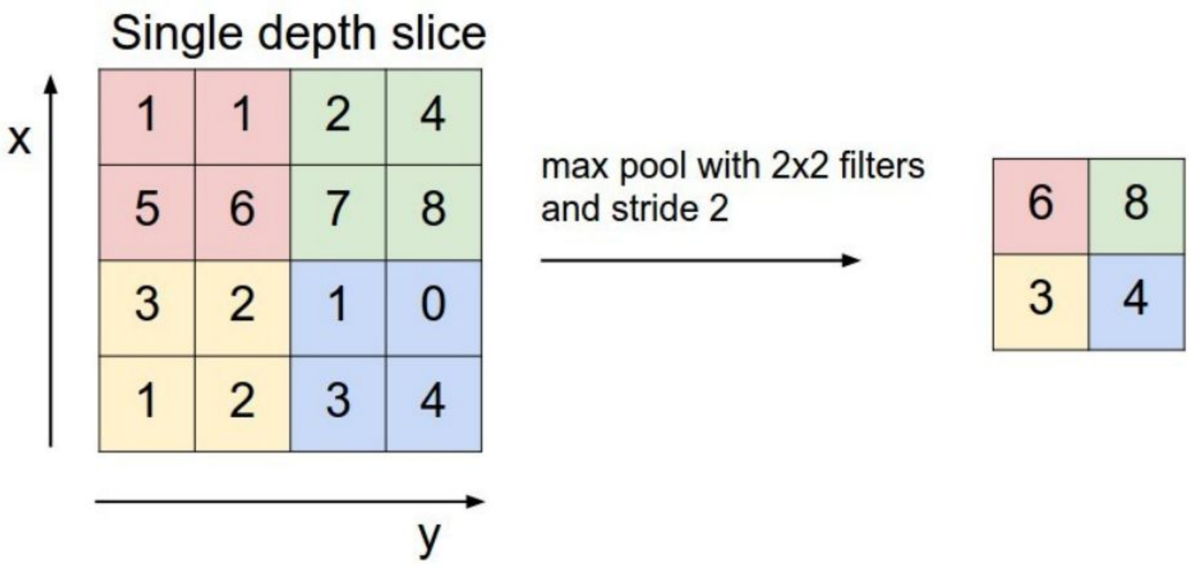
Receptive field



Q: how many 3x3 convolutions we should use to recognize a 100x100px cat

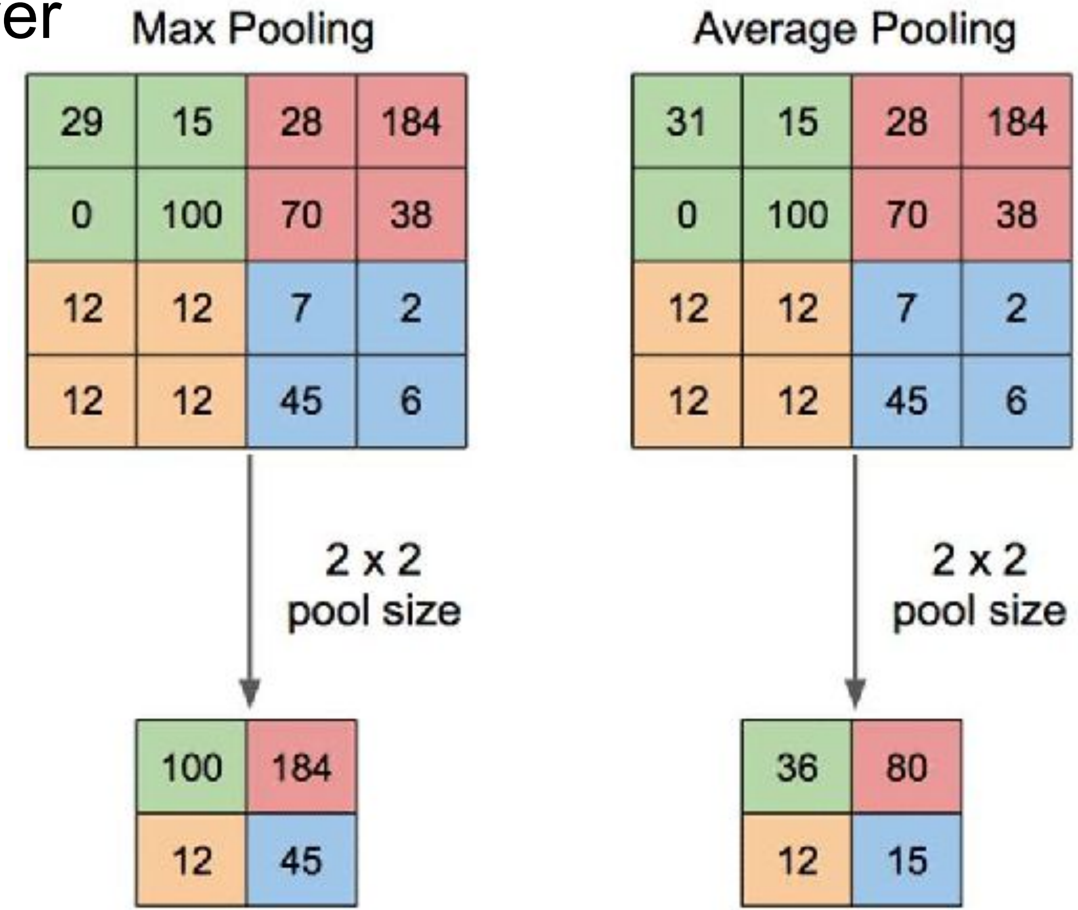
A: around 50... we need to increase receptive field faster!

Pooling



Intuition: What is the max cat-likelihood over this area?

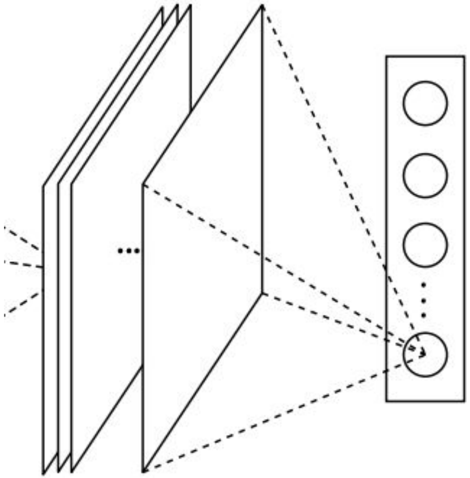
Pooling layer

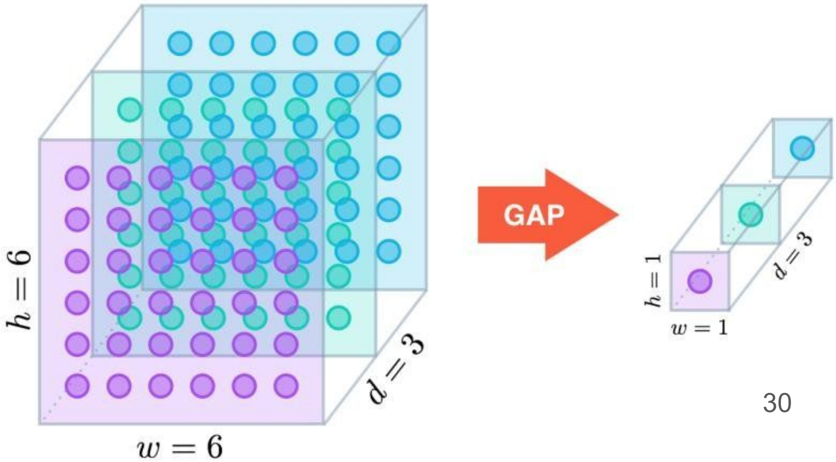


Global poling

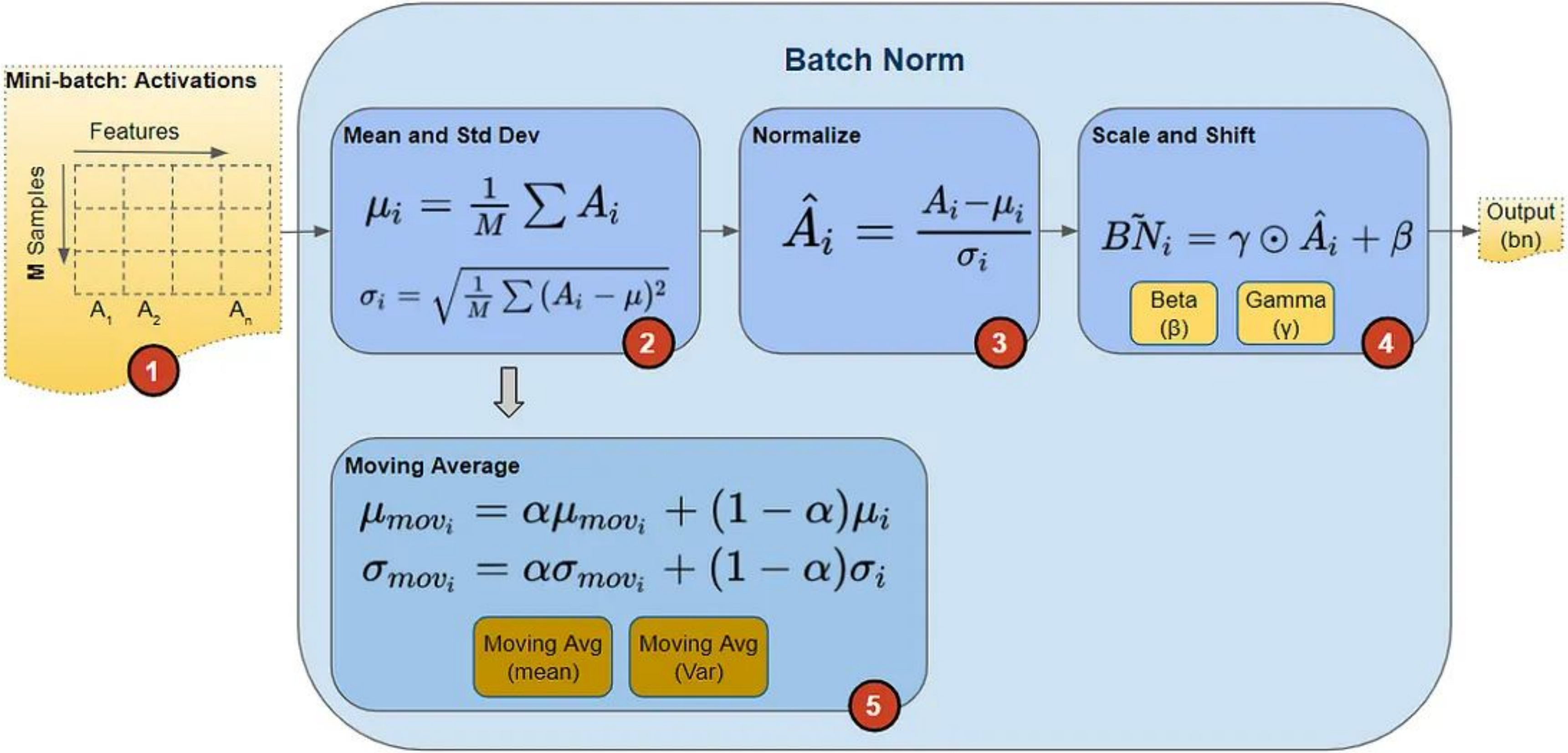
The idea is to generate one feature map for each corresponding category of the classification task in the last conv layer. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer.

Also can be global max pooling or global average pooling.



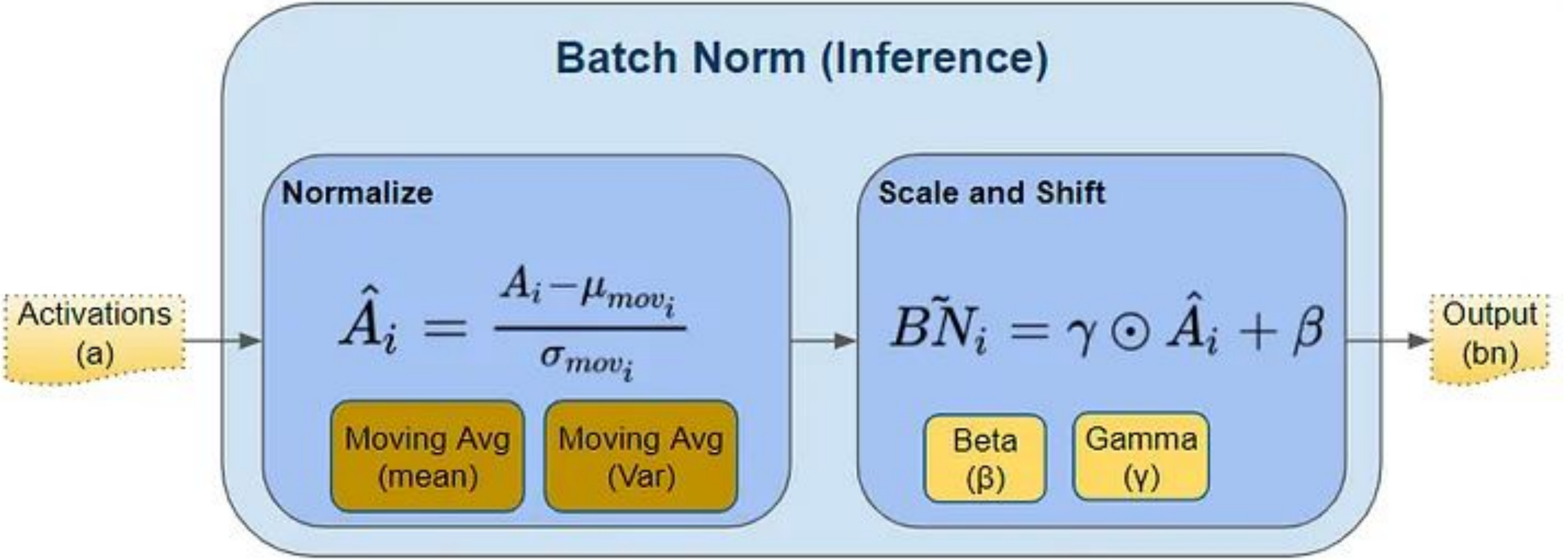


BatchNorm layer



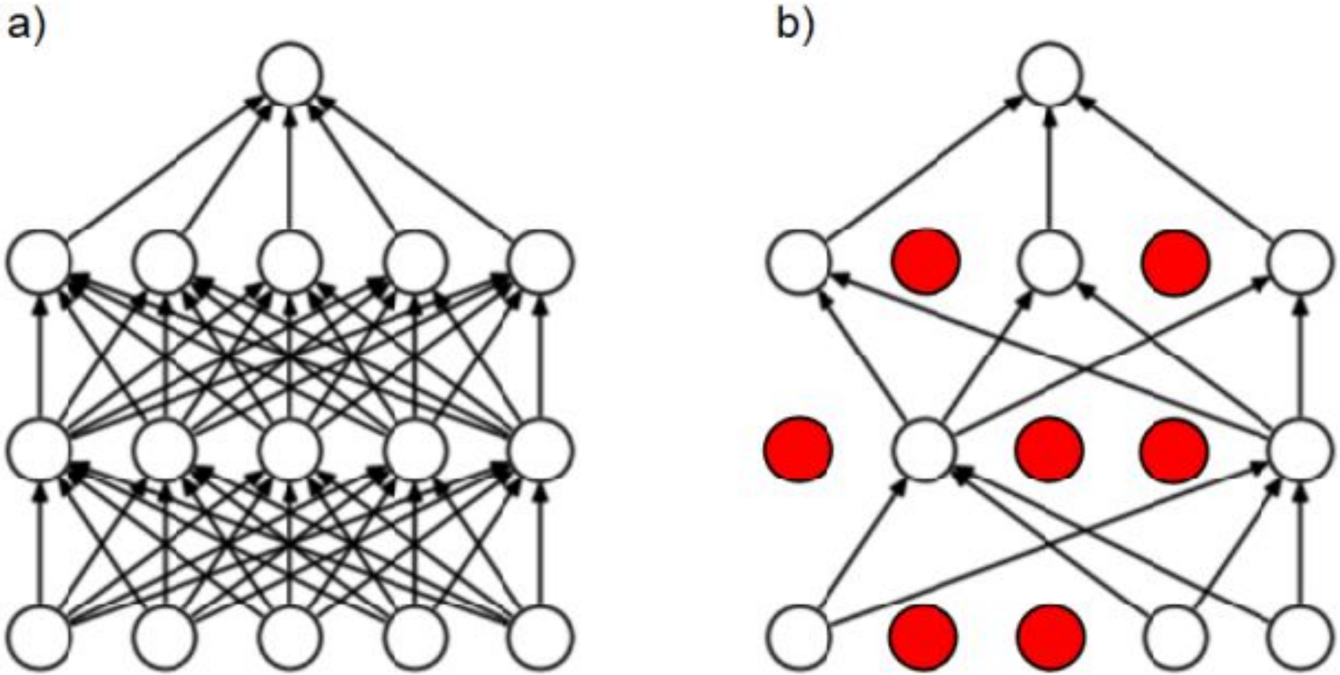
link

BatchNorm layer

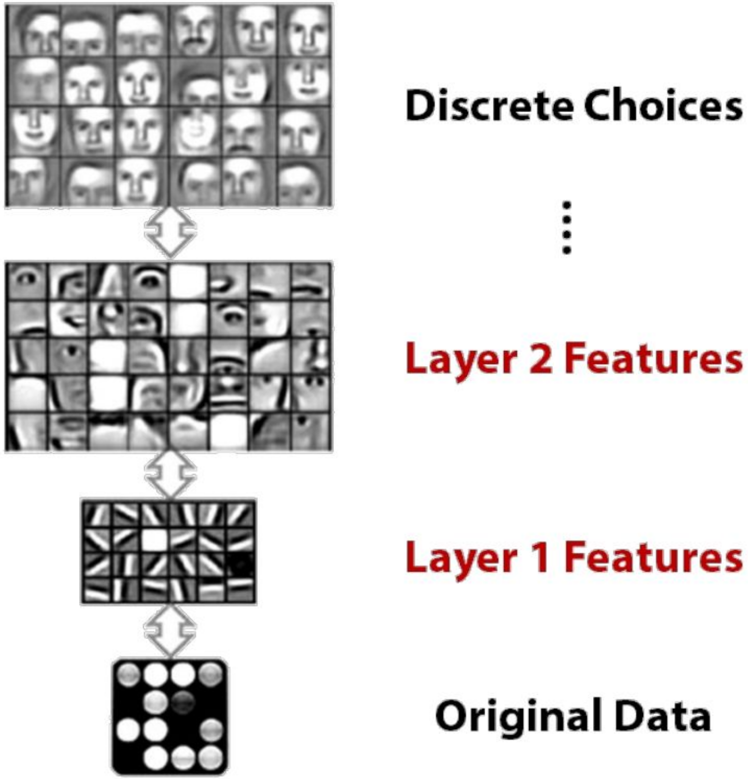


link

Dropout layer



Stack more layers



Problems with large nets

What you sign for if you stack 1000 layers:

- MemoryError(0x...)
- Gradients can vanish
- Gradients can explode
- Activations can vanish
- Activations can explode
- Overfits like crazy

Problems with large nets

What you sign for if you stack 1000 layers:

- MemoryError(0x...)
- **Gradients can vanish**
- **Gradients can explode**

Residual connections

- **Activations can vanish**
- **Activations can explode**

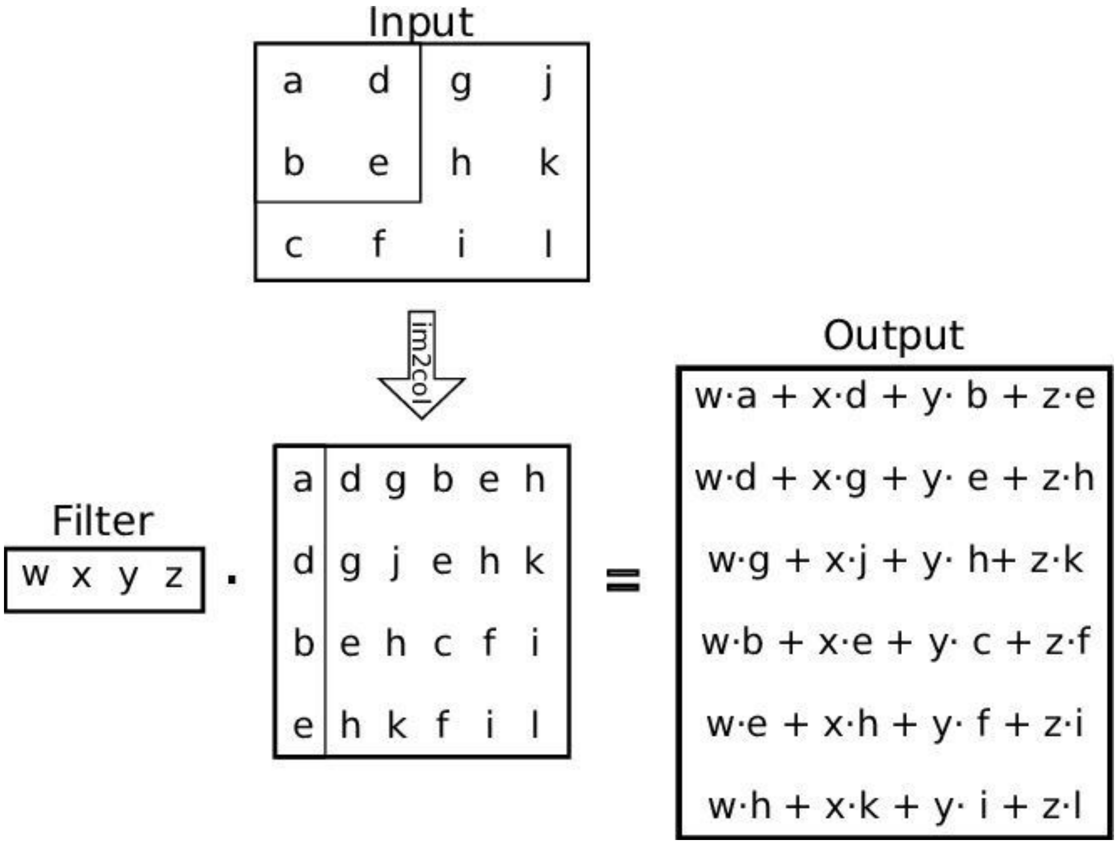
Batch normalization or similar

- Overfits like crazy

De jure just a linear layer (with loads of zeros)

$$\begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

De facto im2col



Data augmentation

- Idea: we can get N times more data by tweaking images.
- If you rotate cat image by 15°, it's still a cat

- Rotate, crop, zoom, flip horizontally, add noize, etc.
- Sound data: add background noizes

Datasets

ImageNet

Data set Details:

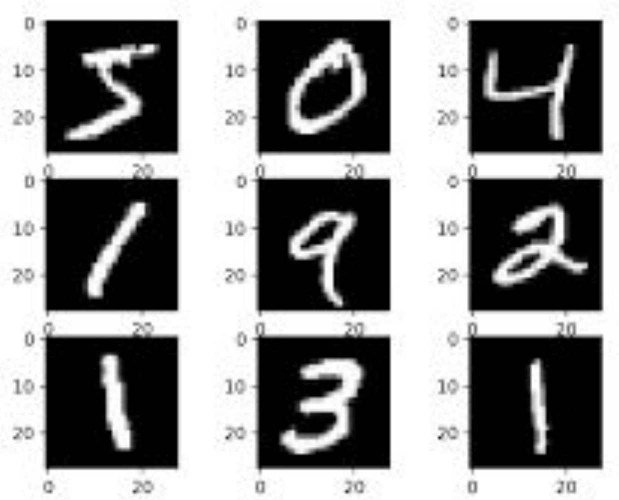
- 14,197,122 images
- 1000 classes
- 224*224

CIFAR-10

- 60,000 color images
- distributed across 10 classes
- 32*32

MNIST

- 60,000 training images and 10,000 test images of 0-9 digits
- 28*28 pixels



Fashion-MNIST

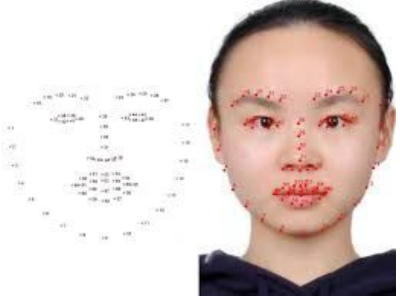
- 70,000 fashion images
- 28×28

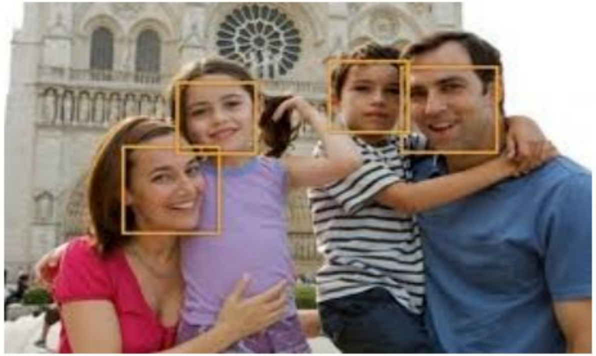
Other CV tasks

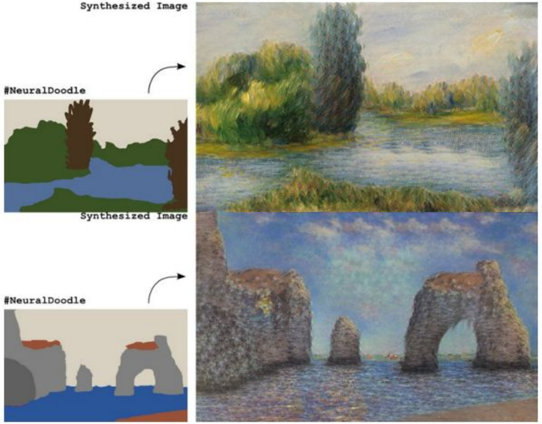
- point coordinates regression
- bounding box regression + classification
- segmentation
- style transferring
- image upscaling

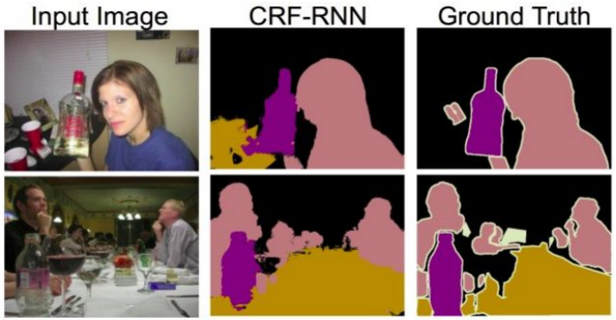
Other CV tasks

- point coordinates regression
- bounding box regression + classification
- segmentation
- style transferring
- image upscaling





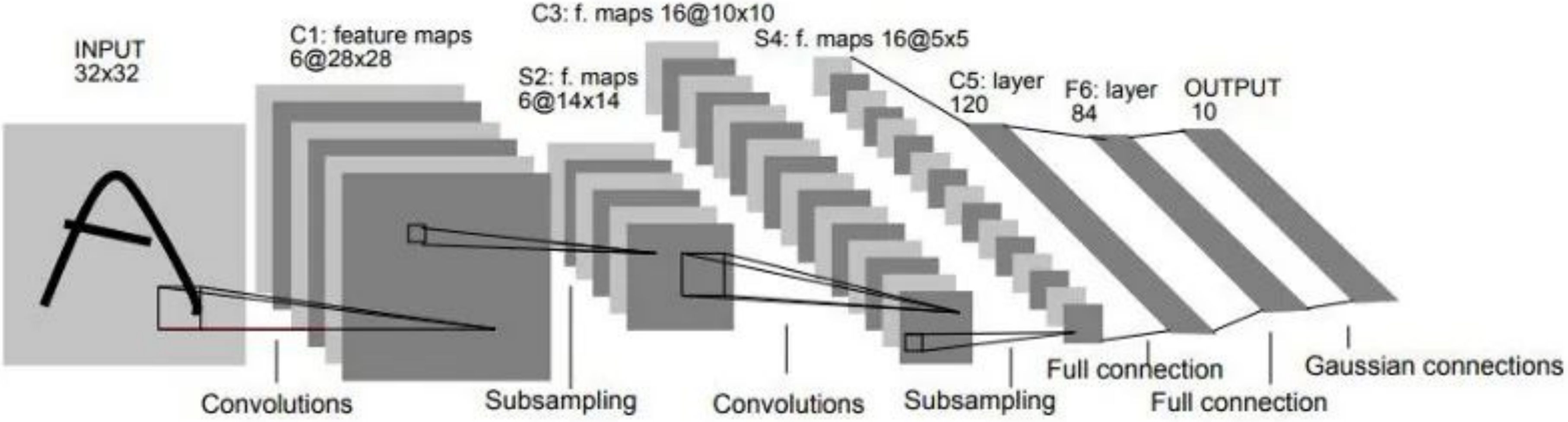




Architectures

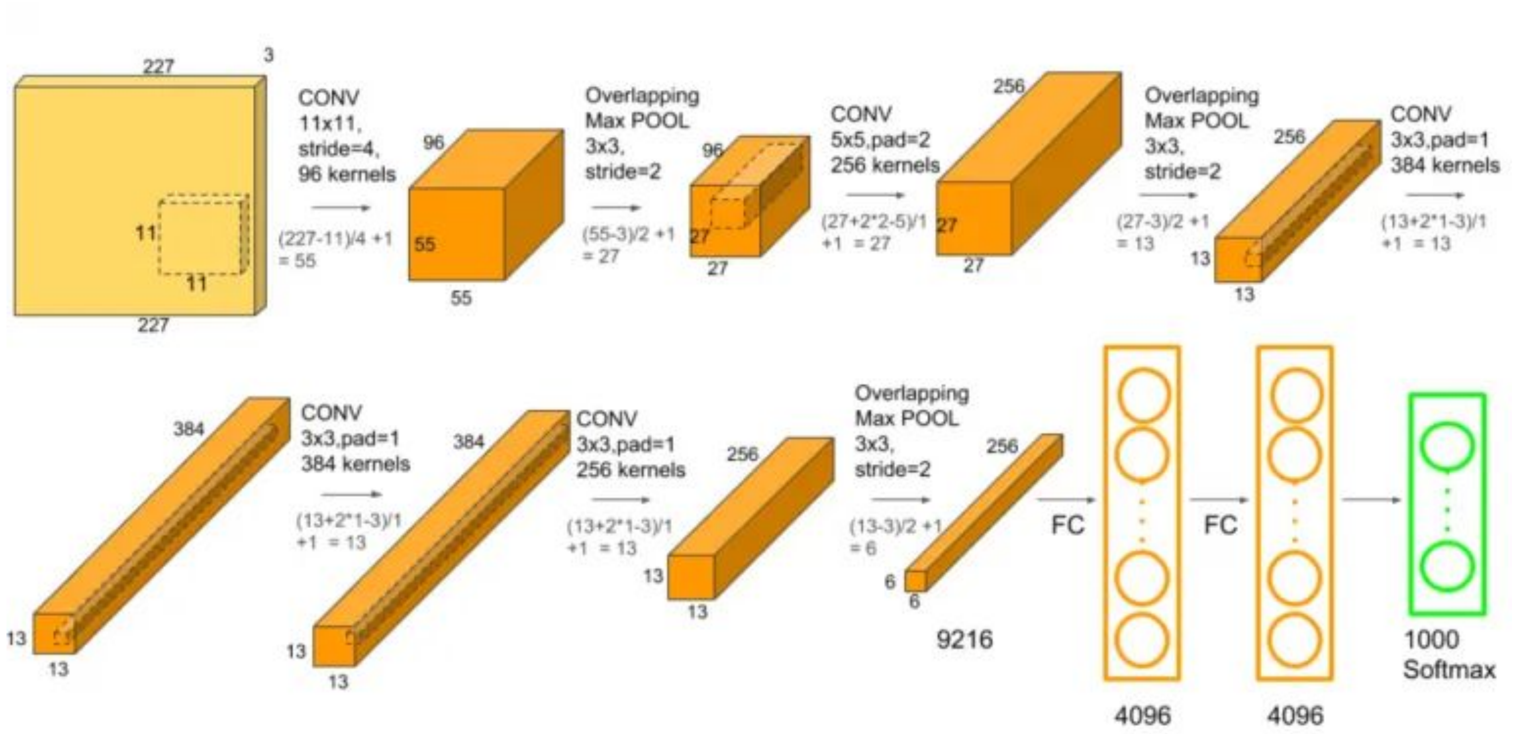
LeNet

LeNet-5, introduced by Yann LeCun and his team in the 1990s, was one of the first successful CNN architectures. Designed for handwritten digit recognition. About 60000 params

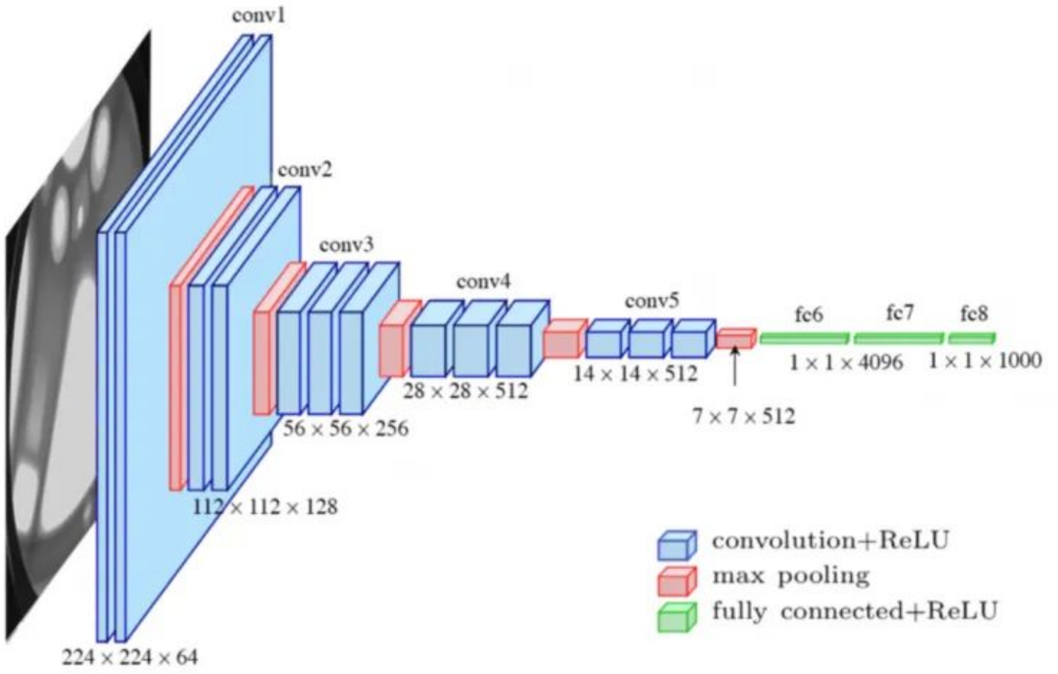


AlexNet

Alex Krizhevsky, Ilya Sutskever (ILSVRC) in 2012 was designed to be used with large-scale image datasets and it achieved state-of-the-art results at the time of its publication. 60 million params, activations - ReLU



VGGNet16



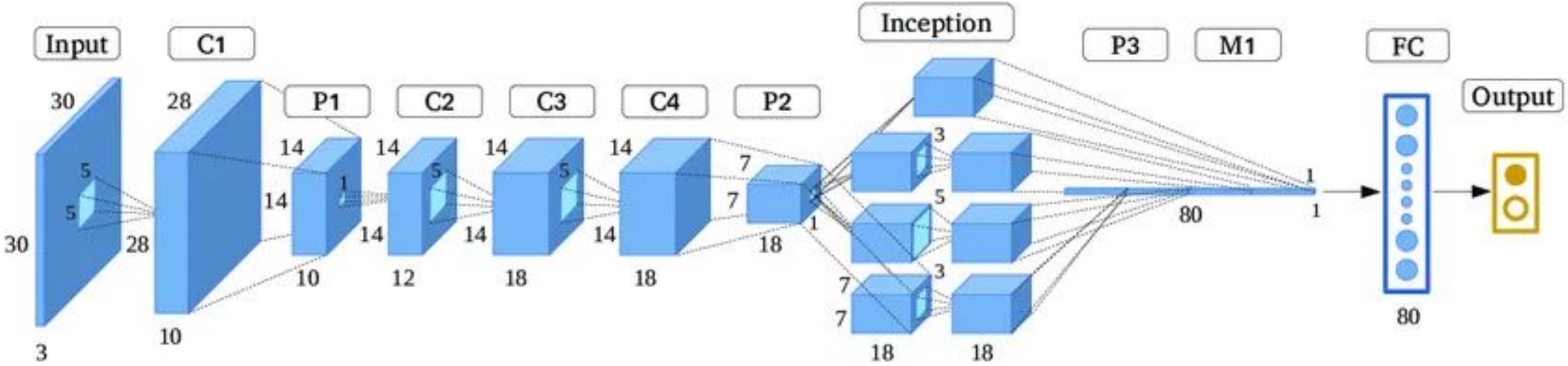
It significantly outperforms AlexNet by substituting several 3x3 kernel-sized filters for the huge kernel-sized filters.

Sota results on ImageNet 92.7 percent in 2014.

About 138 million params.

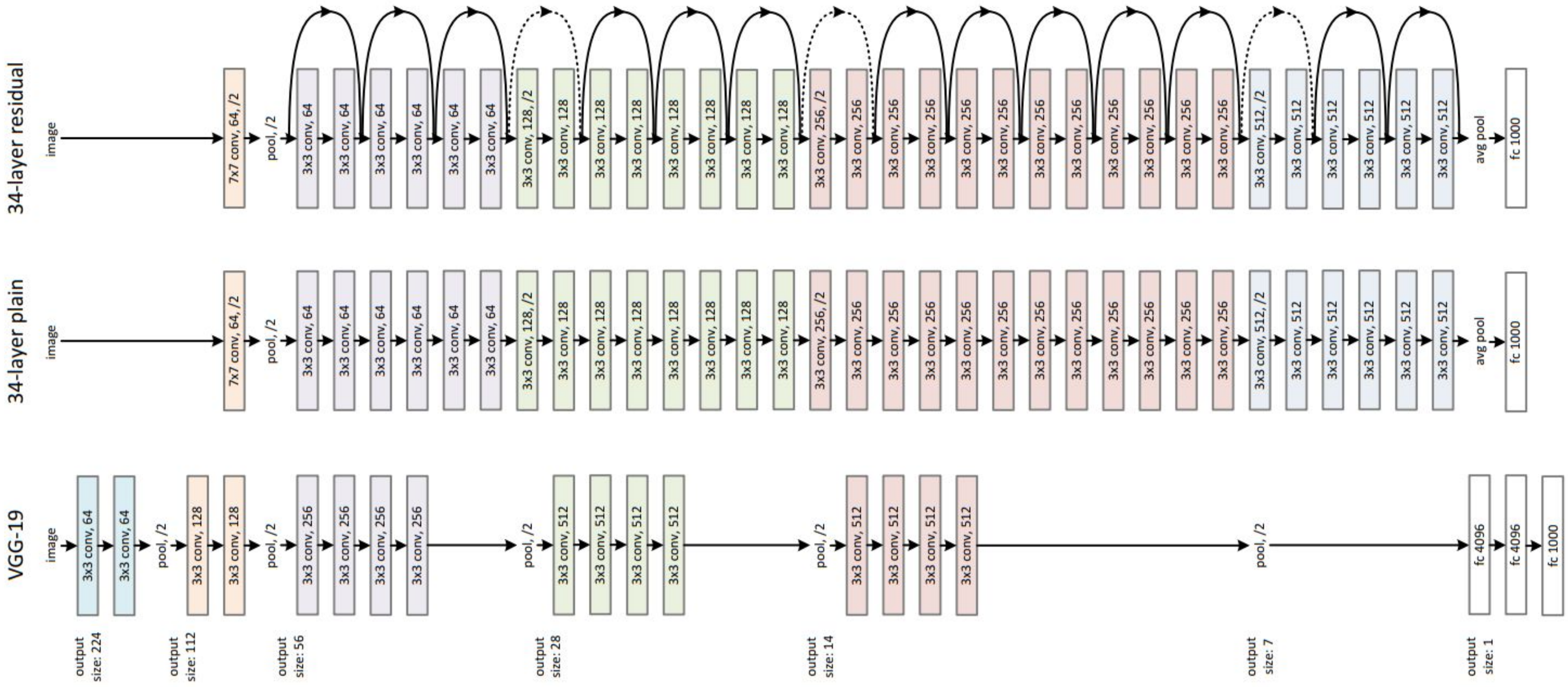
GoogleNet(Inception)

Winner of ILSVRC 2014, introduced the Inception module, which employs parallel convolutional operations with different kernel sizes. This architecture efficiently captures features at multiple scales, promoting better generalization

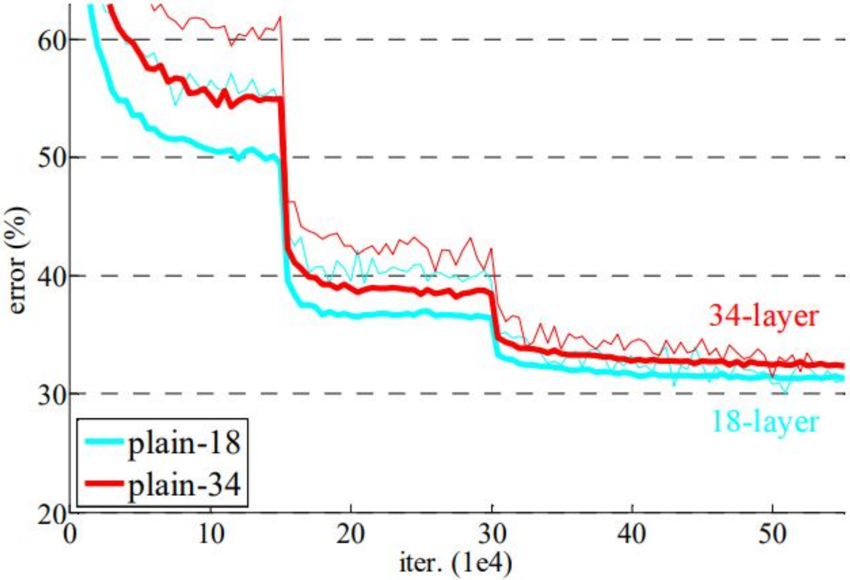


ResNet

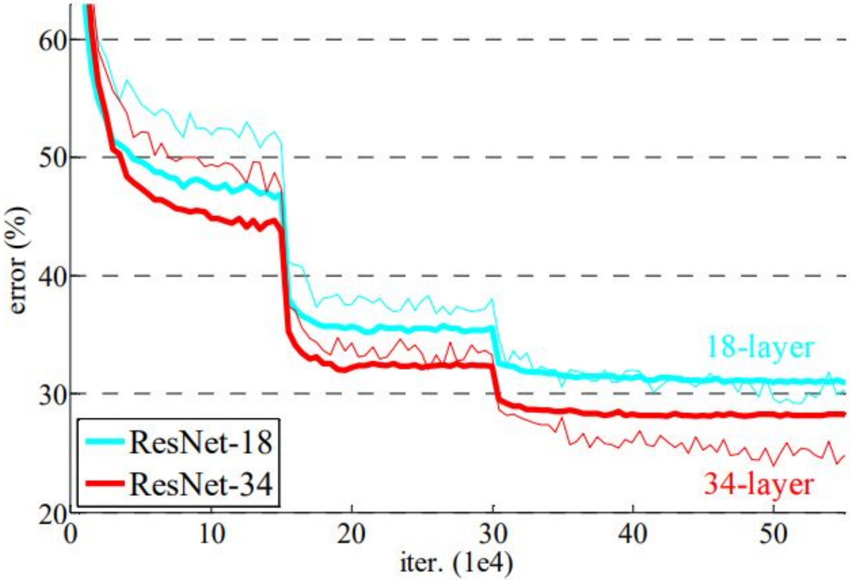
2015-2016. Imagenet 75-79%(resnet34-resnet152) accuracy. ~26 million params



ResNet

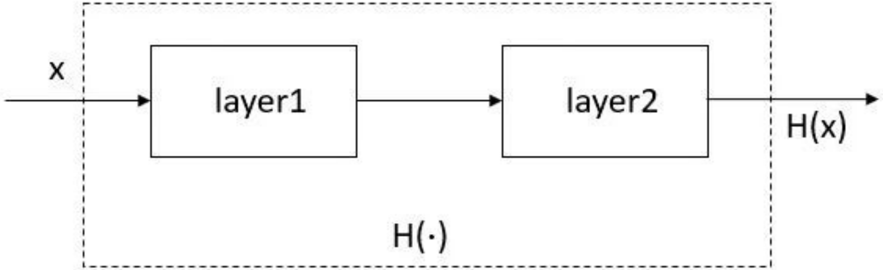


60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
34-layer
18-layer
plain-18
plain-34



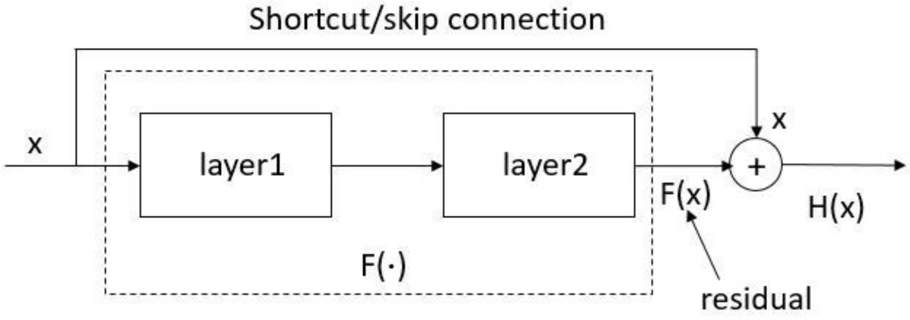
60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
18-layer
34-layer
ResNet-18
ResNet-34

ResNet



$$\left|\frac{\partial H(x)}{\partial x}\right| \ll 1$$

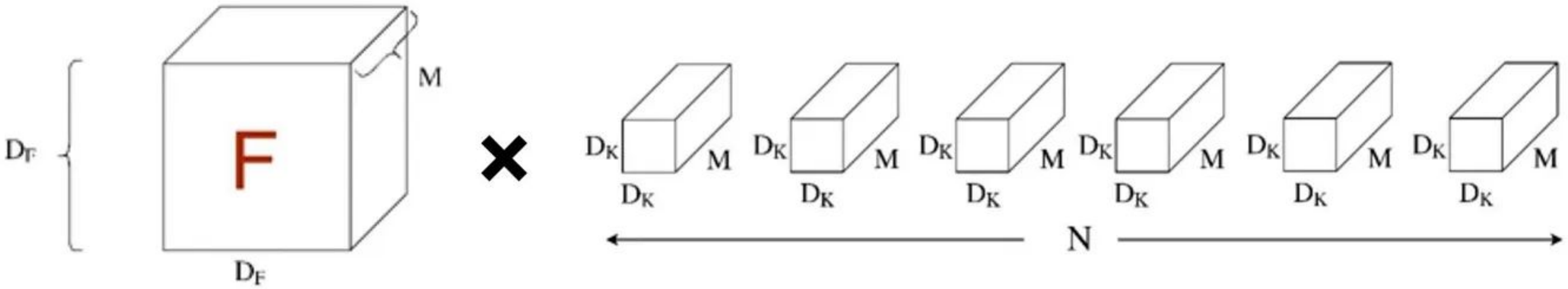
Vanishing gradient



$$\left|\frac{\partial H(x)}{\partial x}\right| = \left|\frac{\partial F(x) + \partial x}{\partial x}\right| = \left|\frac{\partial F(x)}{\partial x}\right| + 1$$

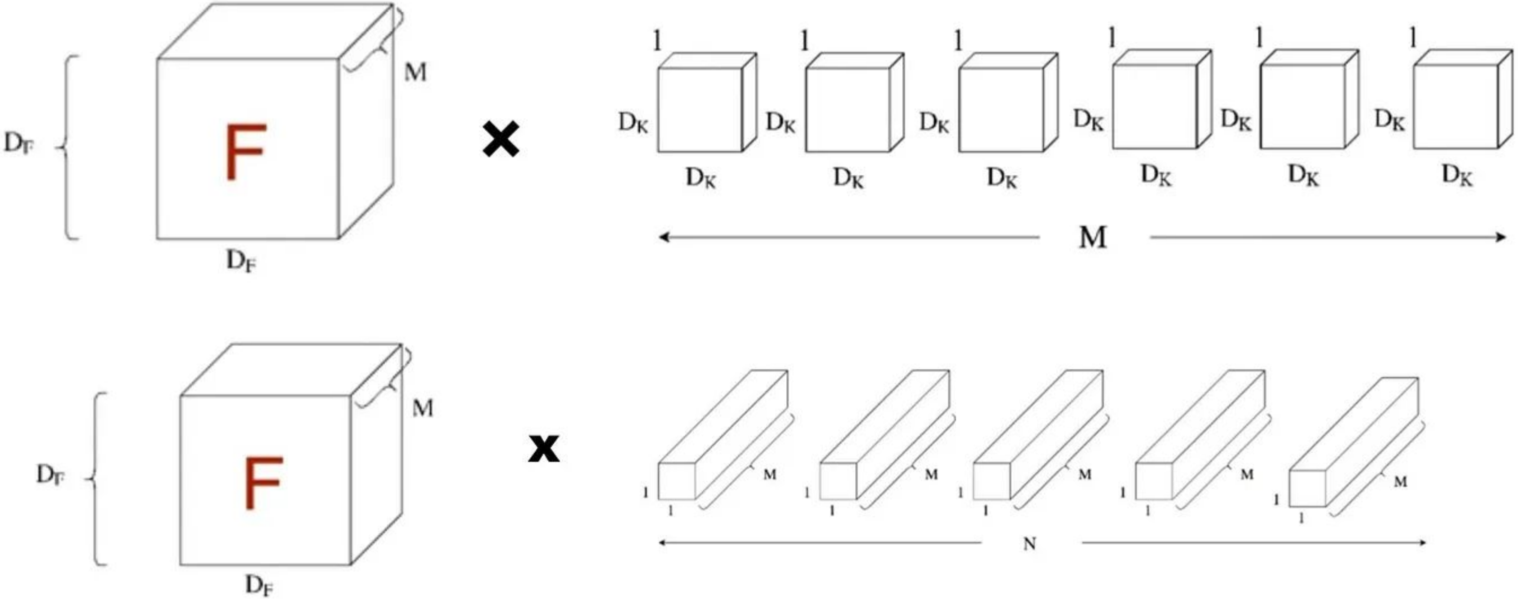
Learning residual F(x) is easier than directly learning H(x) since info about x can be passed through the network due to the shortcut/skip connection

MobileNet



$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$$

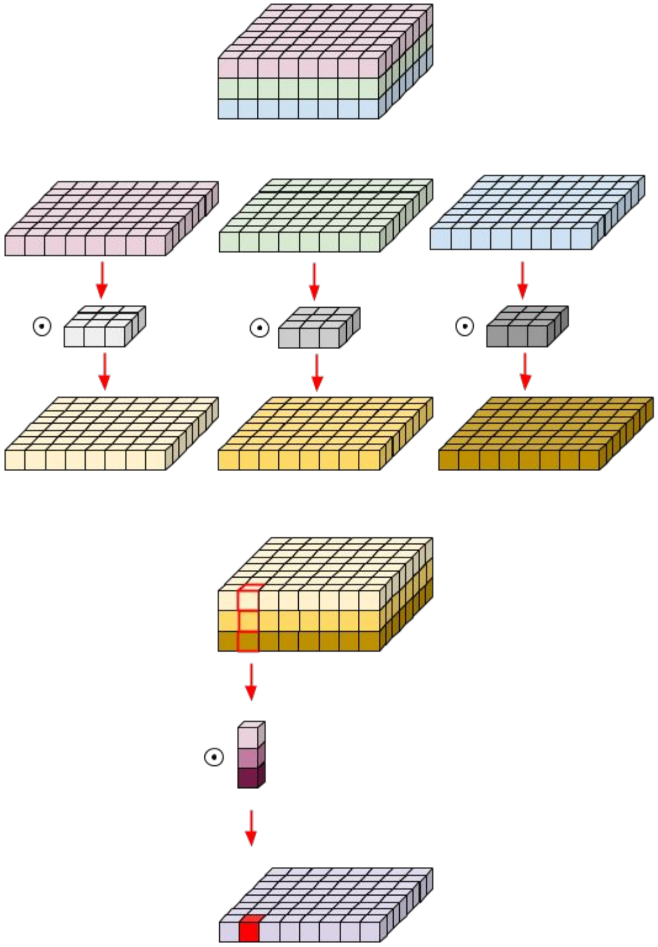
MobileNet



$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

MobileNet

Depthwise Separable Convolutional



MobileNet

2017, Google

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

EfficientNet

Google, 2019

76%-84.2%

4-60M params