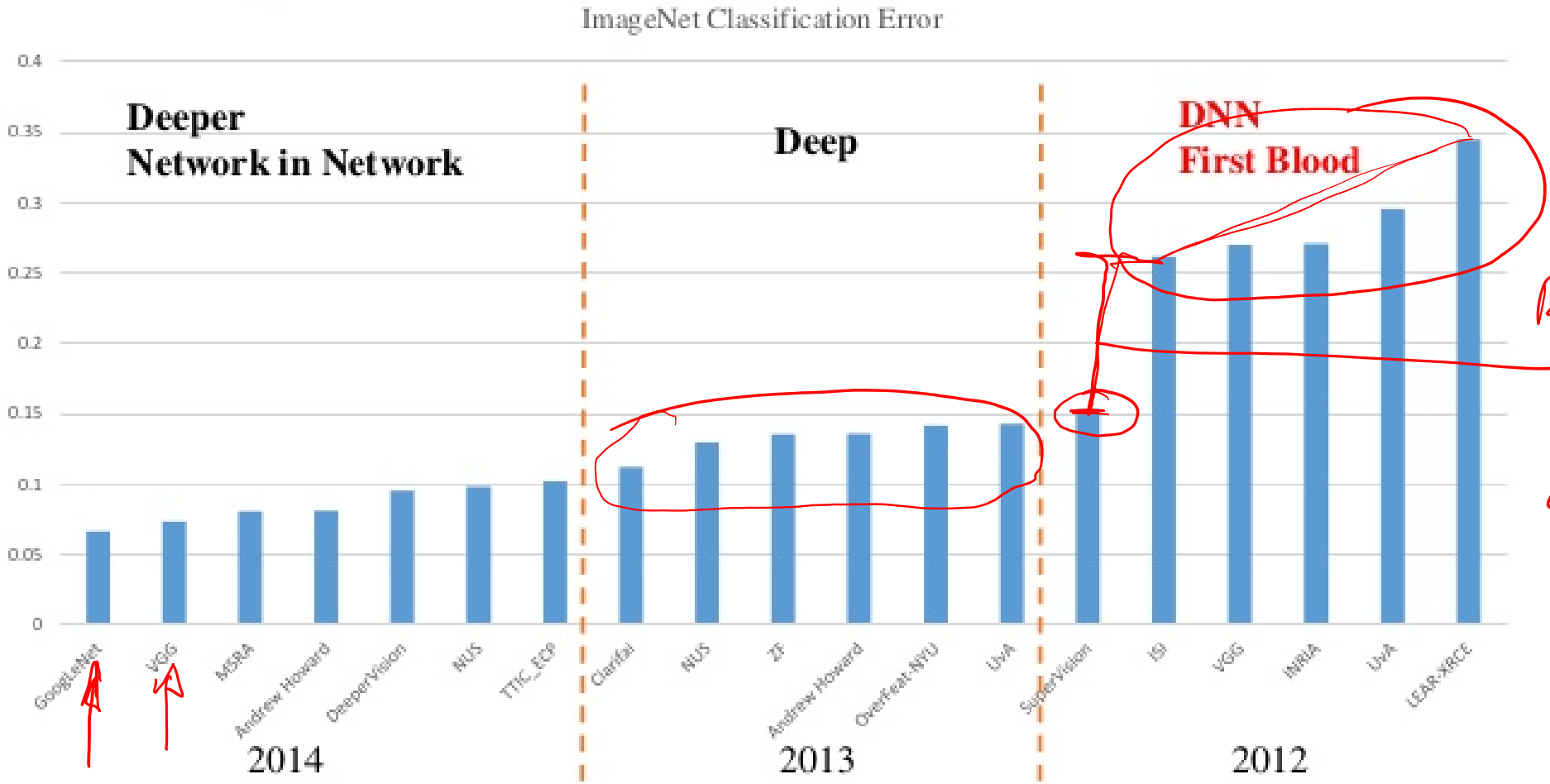
Neural Networks

Jan Chorowski

ImageNet Classification

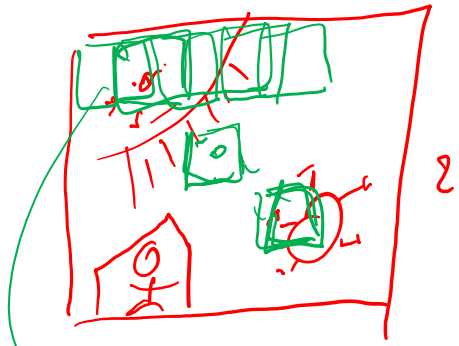
- **1000** categories and **1.2** million training images



Li Fei-Fei: ImageNet Large Scale Visual Recognition Challenge, 2014 http://image-net.org/


CONVNETS, CONVNETS
EVERYWHERE
memegen.com

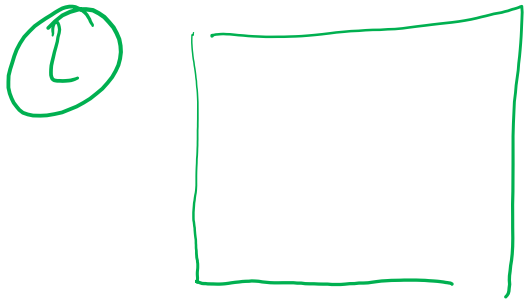
Intuitions



256

256

extract overlapping patches 3×3



1

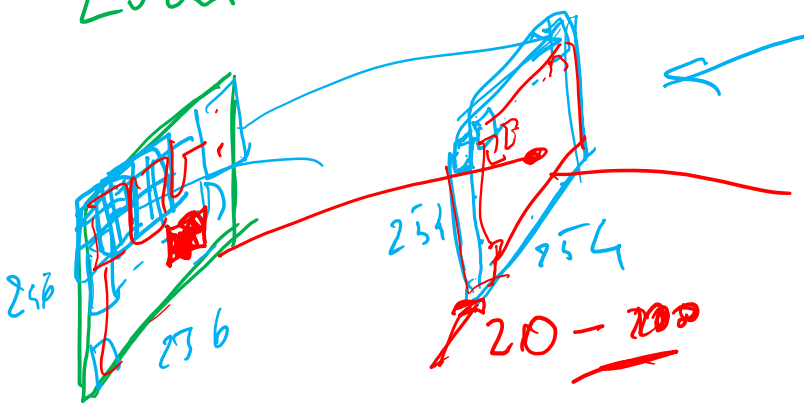
2

3 Weight sharing

1 Down fully connected appn

$\in \mathbb{R}^{256 \times 256}$

Local connect..



256

256

256

256

20-100

Conv ≃ pw

$W \in \mathbb{R}^{256 \times 9}$

$(2^8)^4 = 2^{32} = 4G$

$X \quad W^T$

256²

(256×256) × (256×256)

$O((H \times W)^2)$

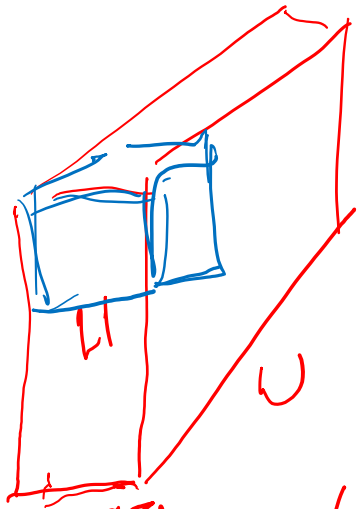
how many with : 9

1600

$O(H \times W)$

Compute >> Parad

A conv layer



Channel / Depth

Input shape : $N \times C \times W \times H$

Batch Size

# of filters

5 x C

Filter / kernel

$OutC \times InC \times F_W \times F_H$

out Size

$N \times OutC \times W \times H$

1 How many params:

$OC \cdot IC \cdot F_W \cdot F_H$

2 How many multiply-adds

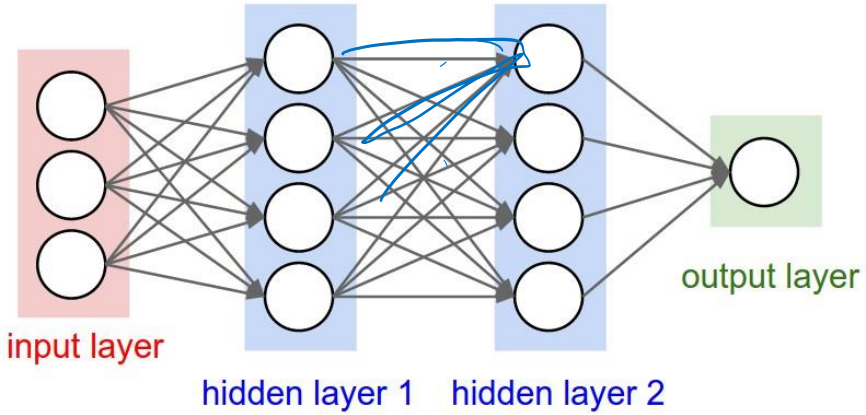
$N \cdot OC \cdot W \cdot H \cdot F_W \cdot F_H \cdot IC$

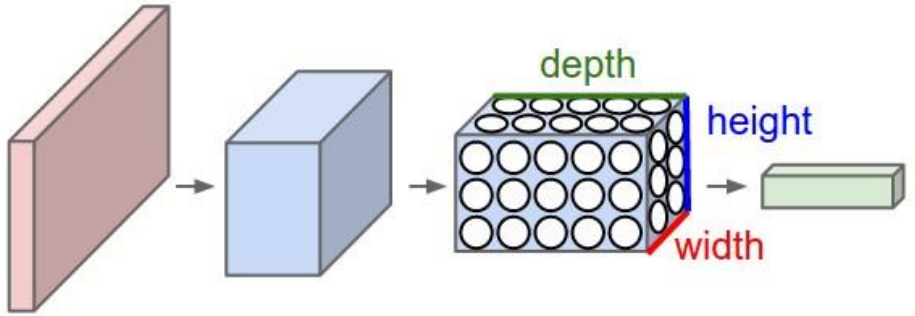
Sharing neurons - convolutions

Note: material from http://cs231n.github.io/convolutional-networks/

In a conv net we use a different connection pattern between layers:

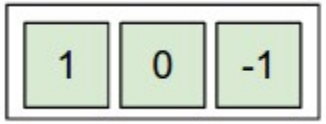
- In a dense feed-forward layer used an all-to-all scheme
- In a conv-net we use local connectivity!



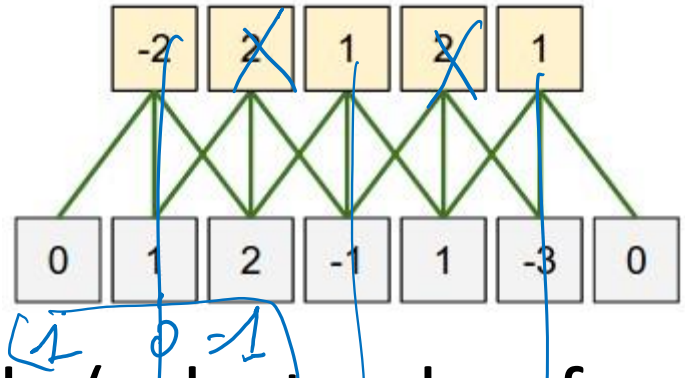


1D Convolution layer

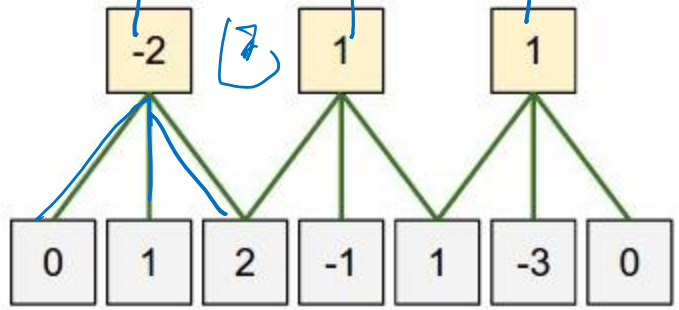
- Small filter (neuron):



- Swipe the filter over the sequence



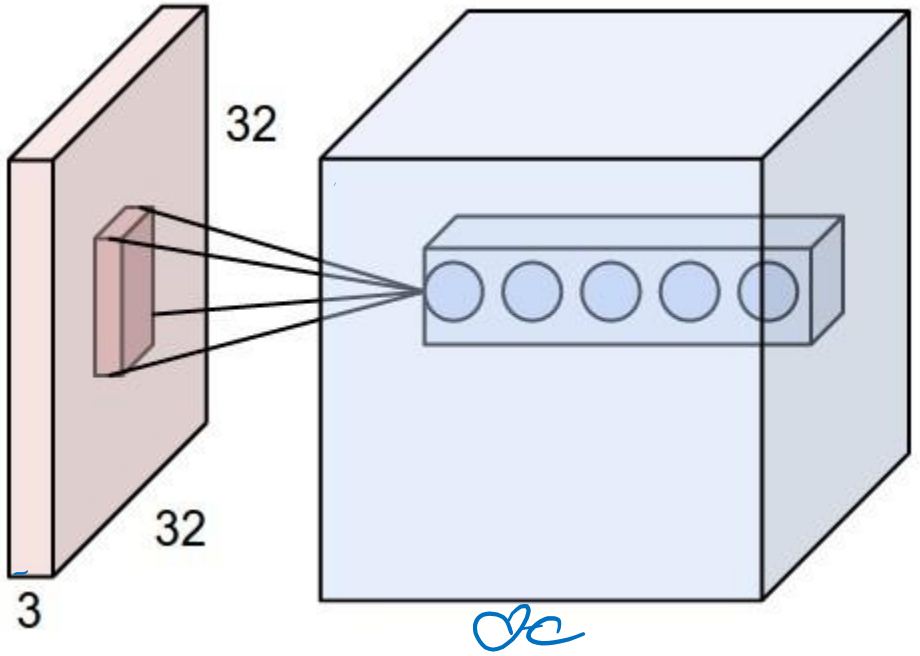
- Pool or stride (select only a few outputs)



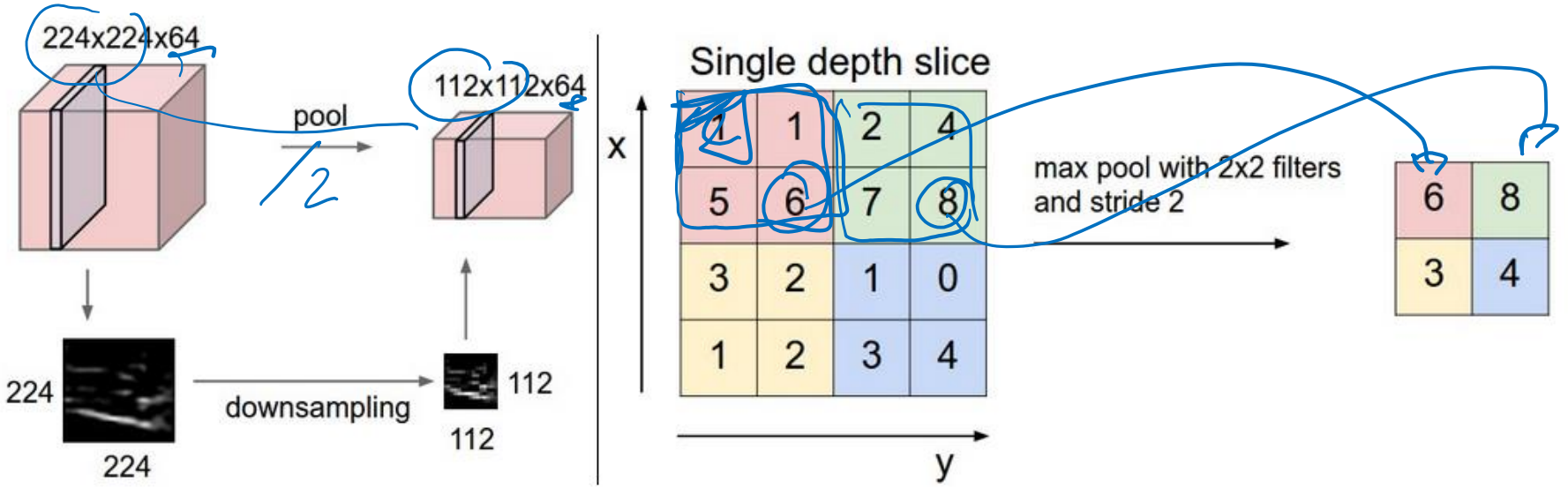
Strides may cause Aliasing

2D conv layer

- http://cs231n.github.io/convolutional-networks/



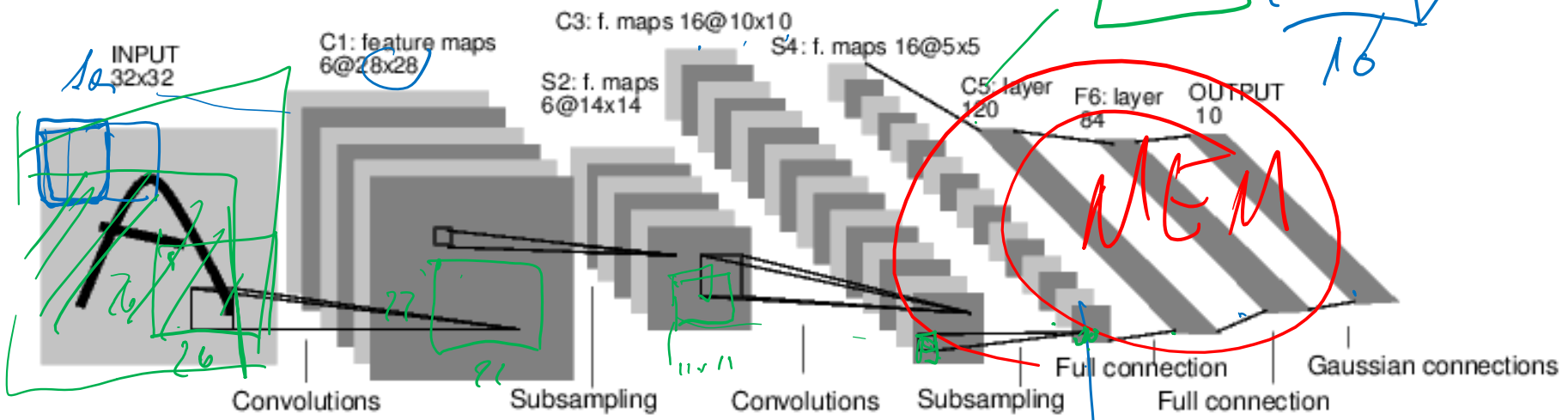
Pooling



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

Full network - LeNet

Y. LeCun et al. „Gradient-Based Learning Applied to Document Recognition"
http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf



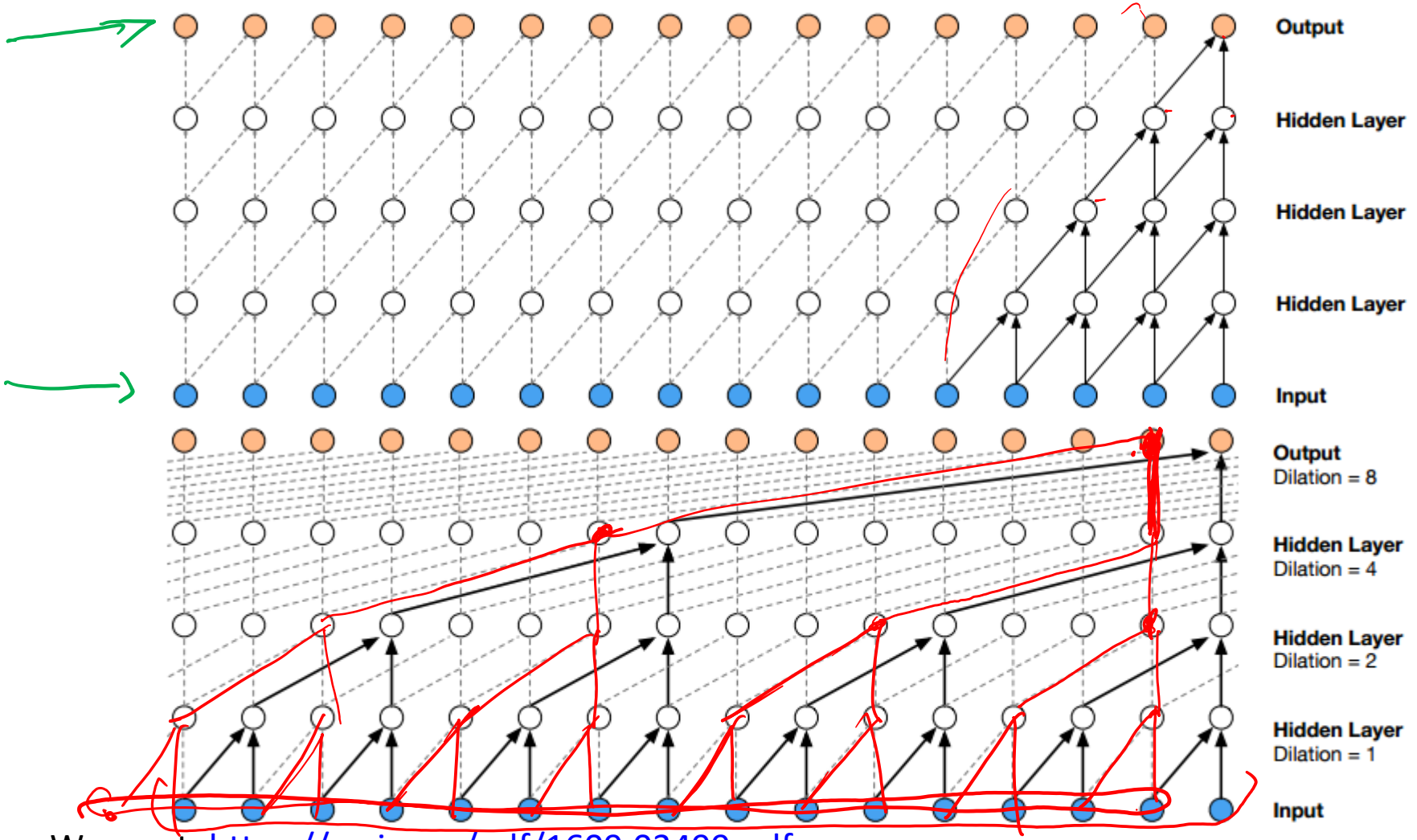
1x6x5x5

6x16x5x5

16x120x5x5

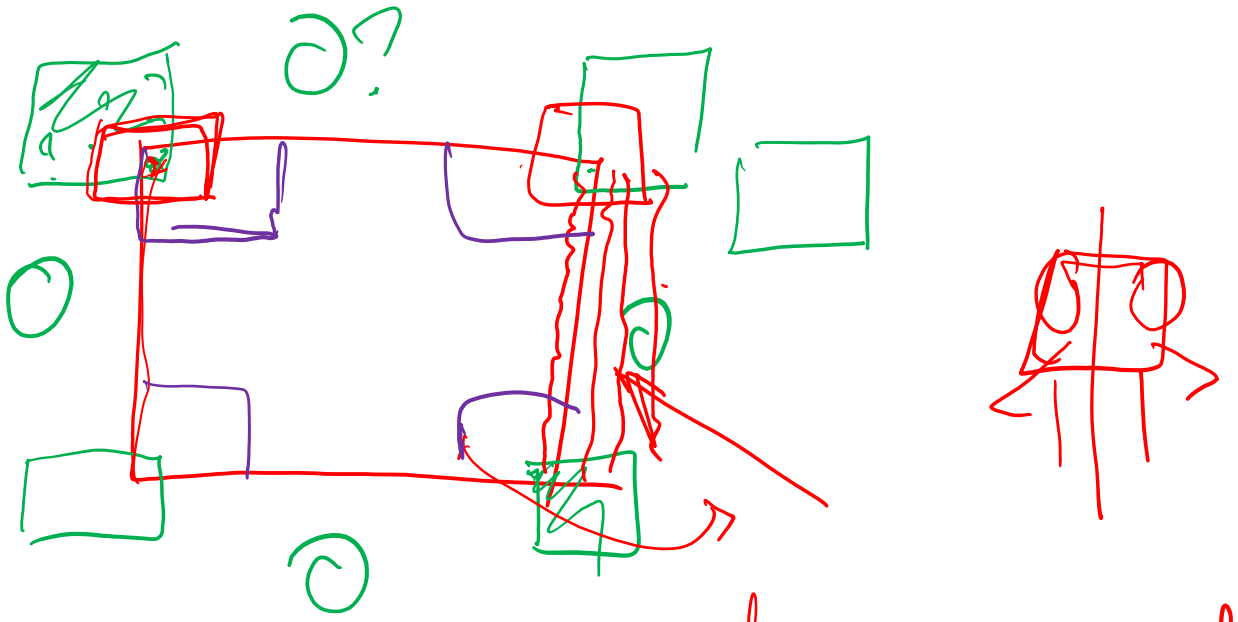
120 x 84

Dilated convolutions: wide receptive field at high resolutions



Wavenet: https://arxiv.org/pdf/1609.03499.pdf
https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Technicality: Edge padding



→ How we padding

→ What do we put w/ it

FULL $W + F_W - 1$

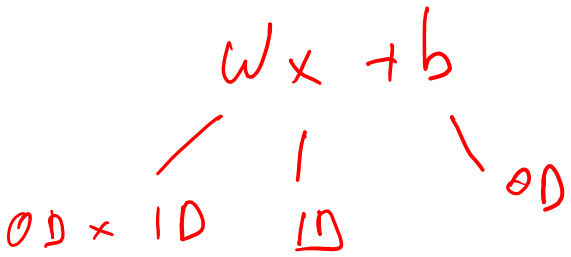
SAME W

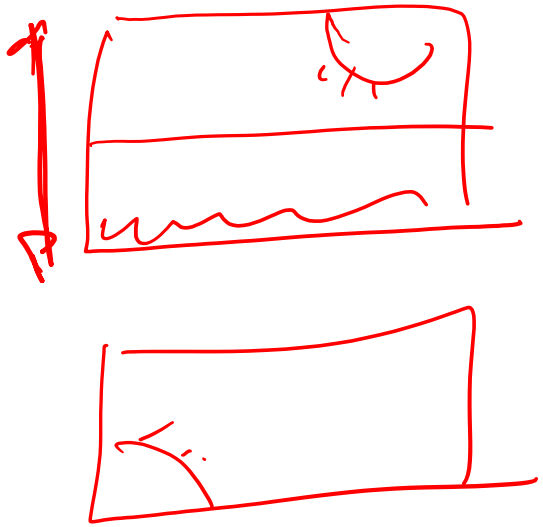
VALID $W - F + 1$

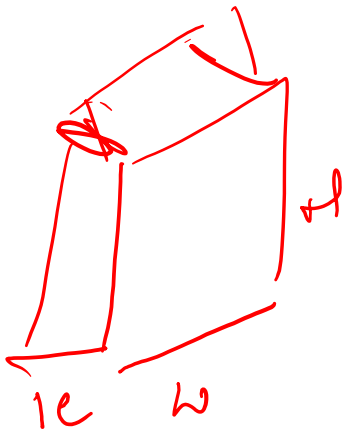
0 — zero

Border, Mirror

Technicality: bias



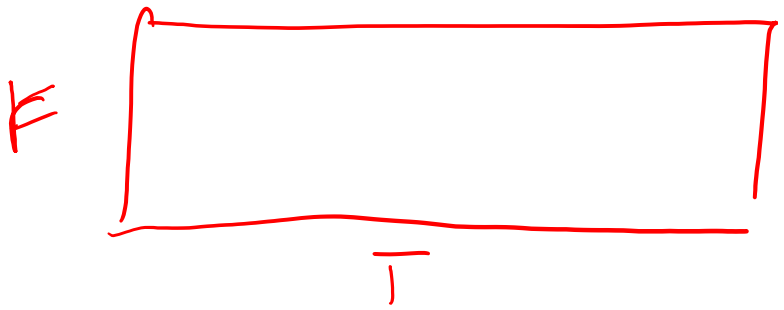




oh...

$OC \times w' \times H'$ + $OC \times w' \times H'$

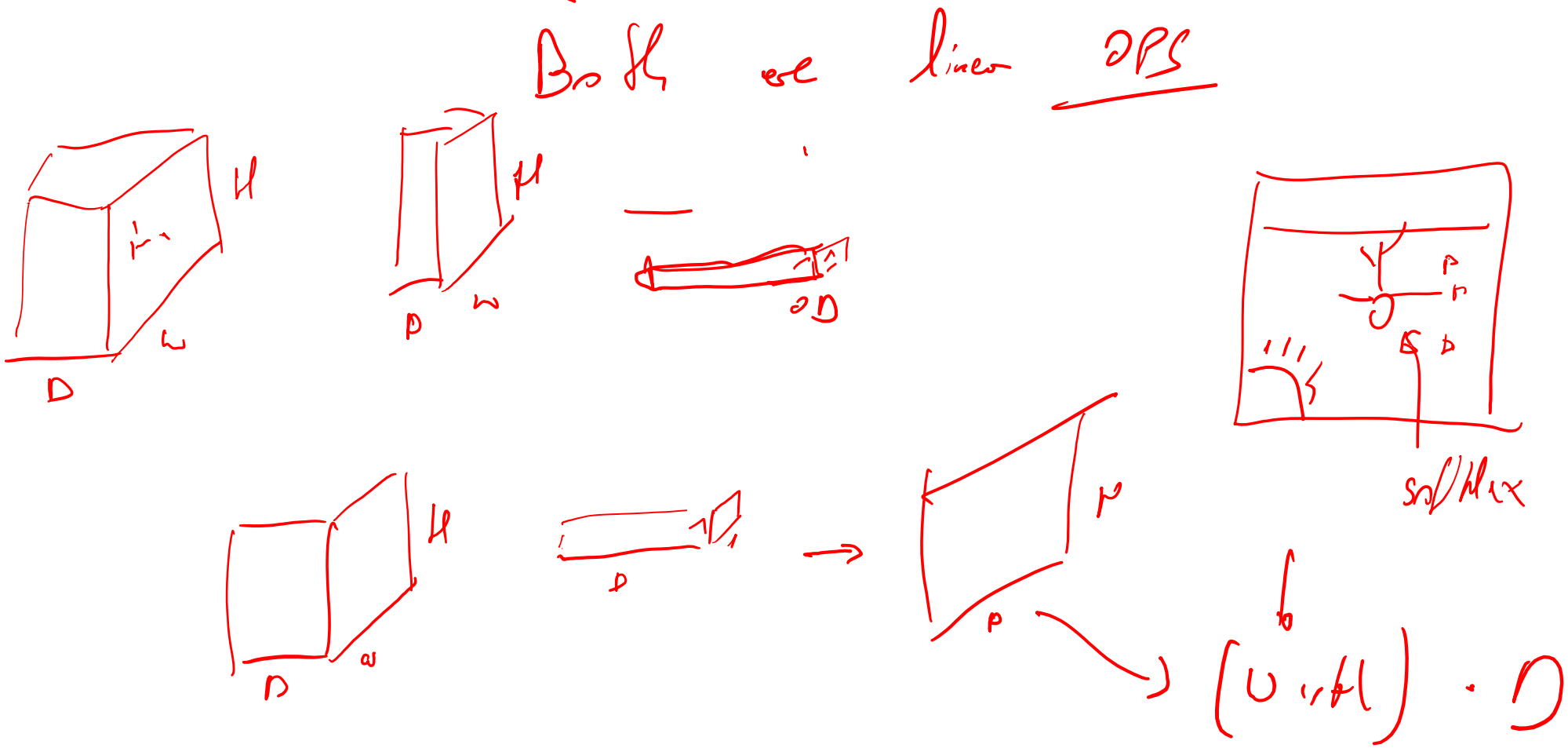
bien



$OC \times 1 \times H'$

$OC \times 1 \times 1$

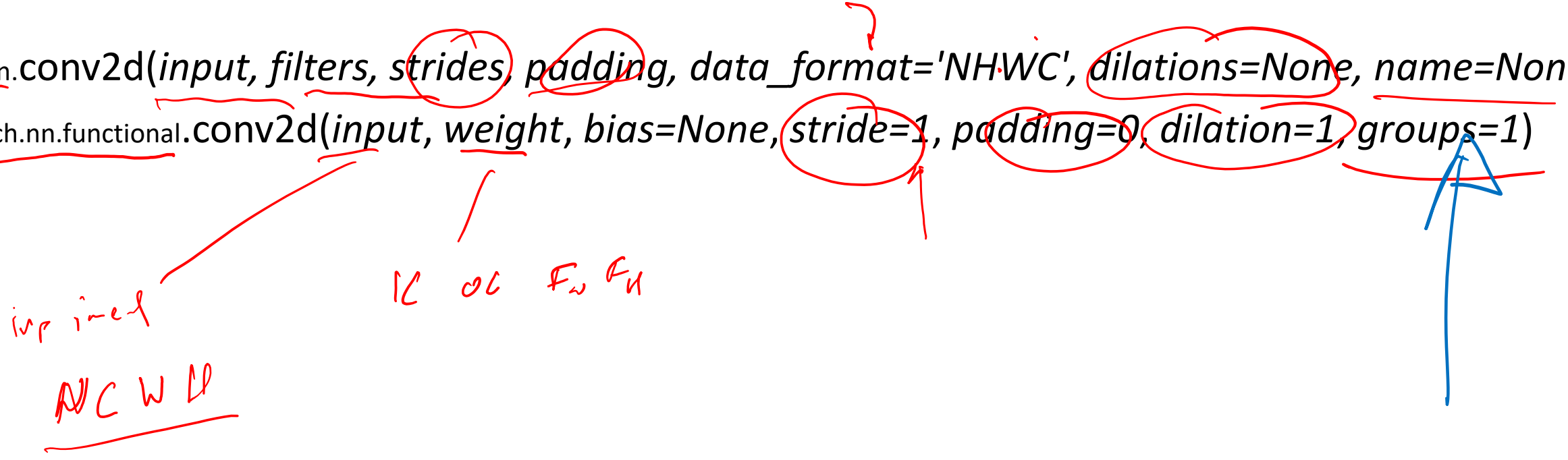
Convs can mimick FC layers



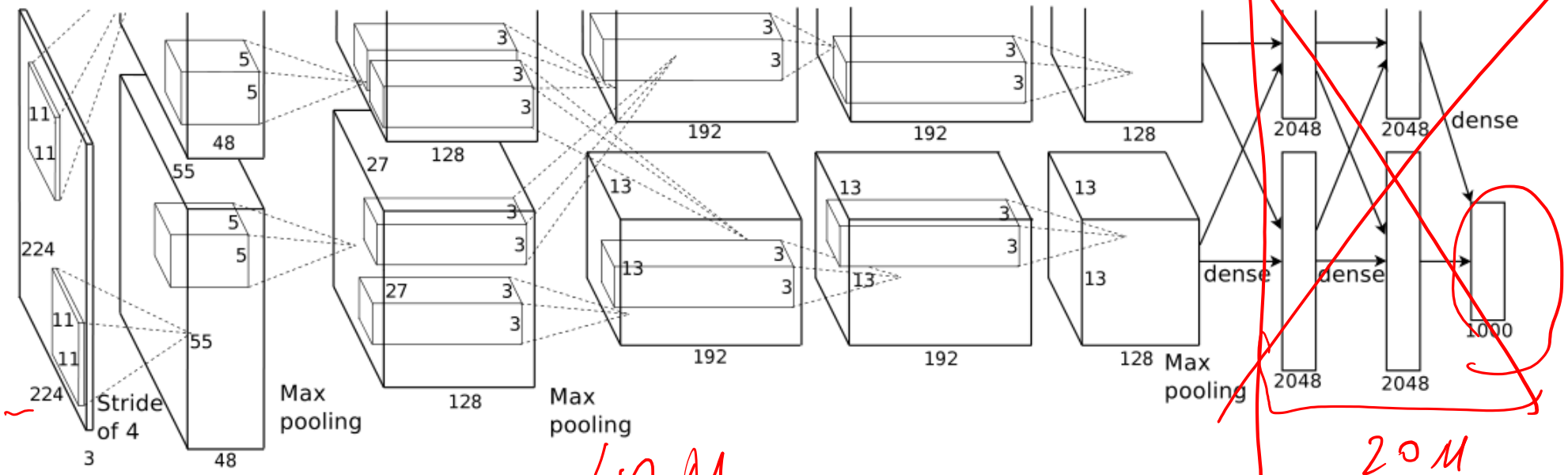
Conv Layers: All the Options

tf.nn.conv2d(*input, filters, strides, padding, data_format='NHWC', dilations=None, name=None*)

torch.nn.functional.conv2d(*input, weight, bias=None, stride=1, padding=0, dilation=1, groups=1*)



AlexNet (2012)



60M parameters

The last dense layers takes:

4096*4096 + 4096*1000=20M params!

Separable Filters

- Someties a 2D conv == 2* 1D conv:

$$\frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

$$\mathbf{G_x} = \begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix} * A = \begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}+1 & 0 & -1\end{bmatrix} * A$$

- Inception and VGGnet use filter approximate sepration, with more nonlinearity

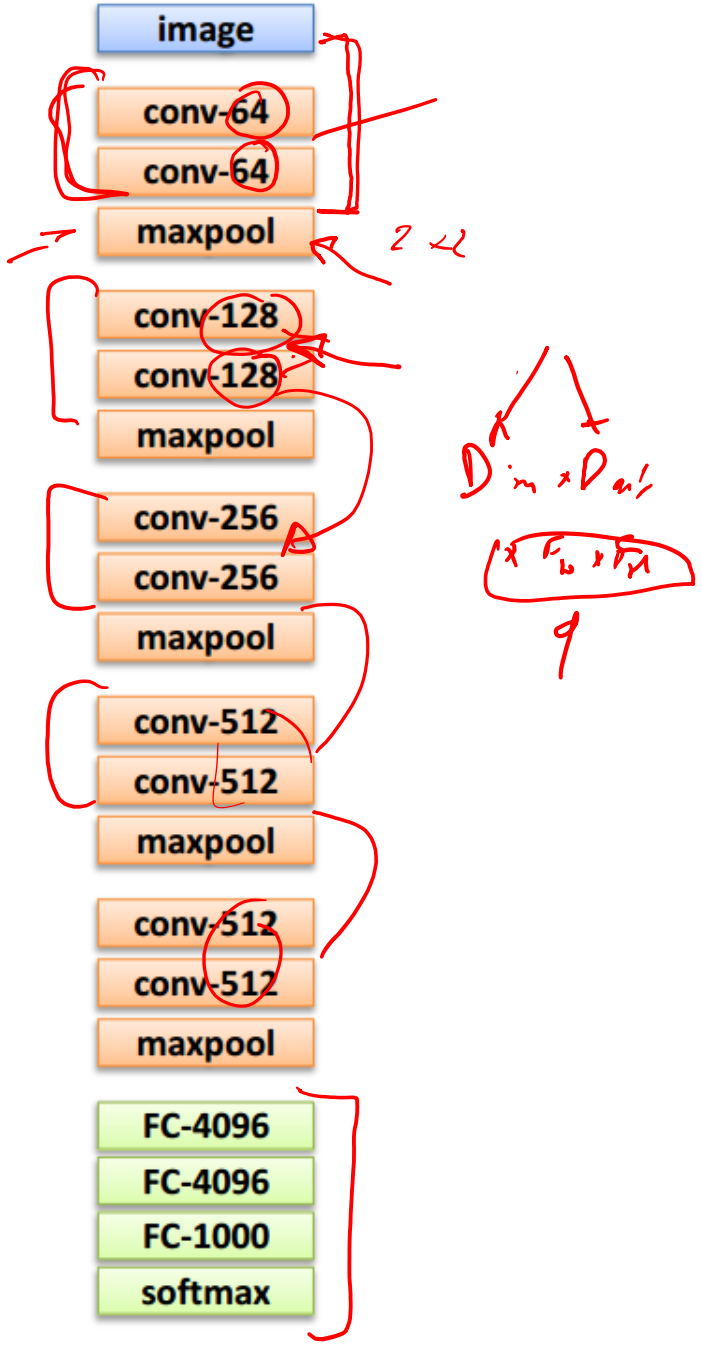
VGGNet: Network Design

Key design choices:

- 3x3 conv. kernels – very small
- conv. stride 1 – no loss of information

Other details:

- Rectification (ReLU) non-linearity
- 5 max-pool layers (x2 reduction)
- no normalisation
- 3 fully-connected (FC) layers

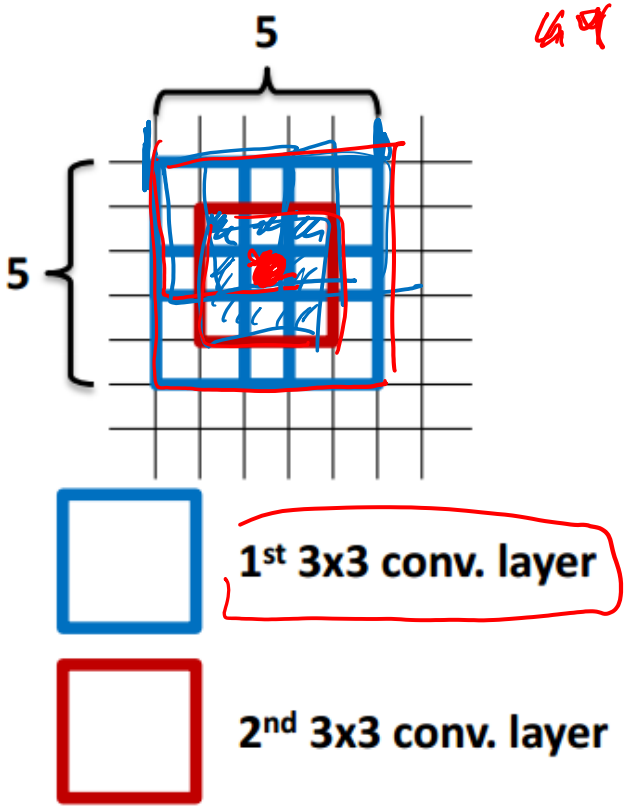


Slide from http://www.robots.ox.ac.uk/~karen/pdf/ILSVRC_2014.pdf

VGGNet: Discussion

Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn
  - ~140M per net



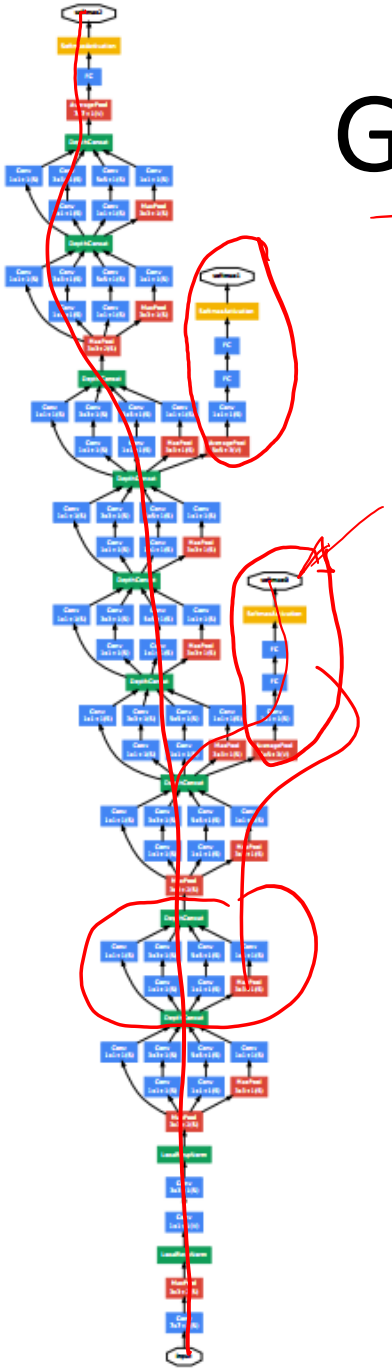
Slide from http://www.robots.ox.ac.uk/~karen/pdf/ILSVRC_2014.pdf

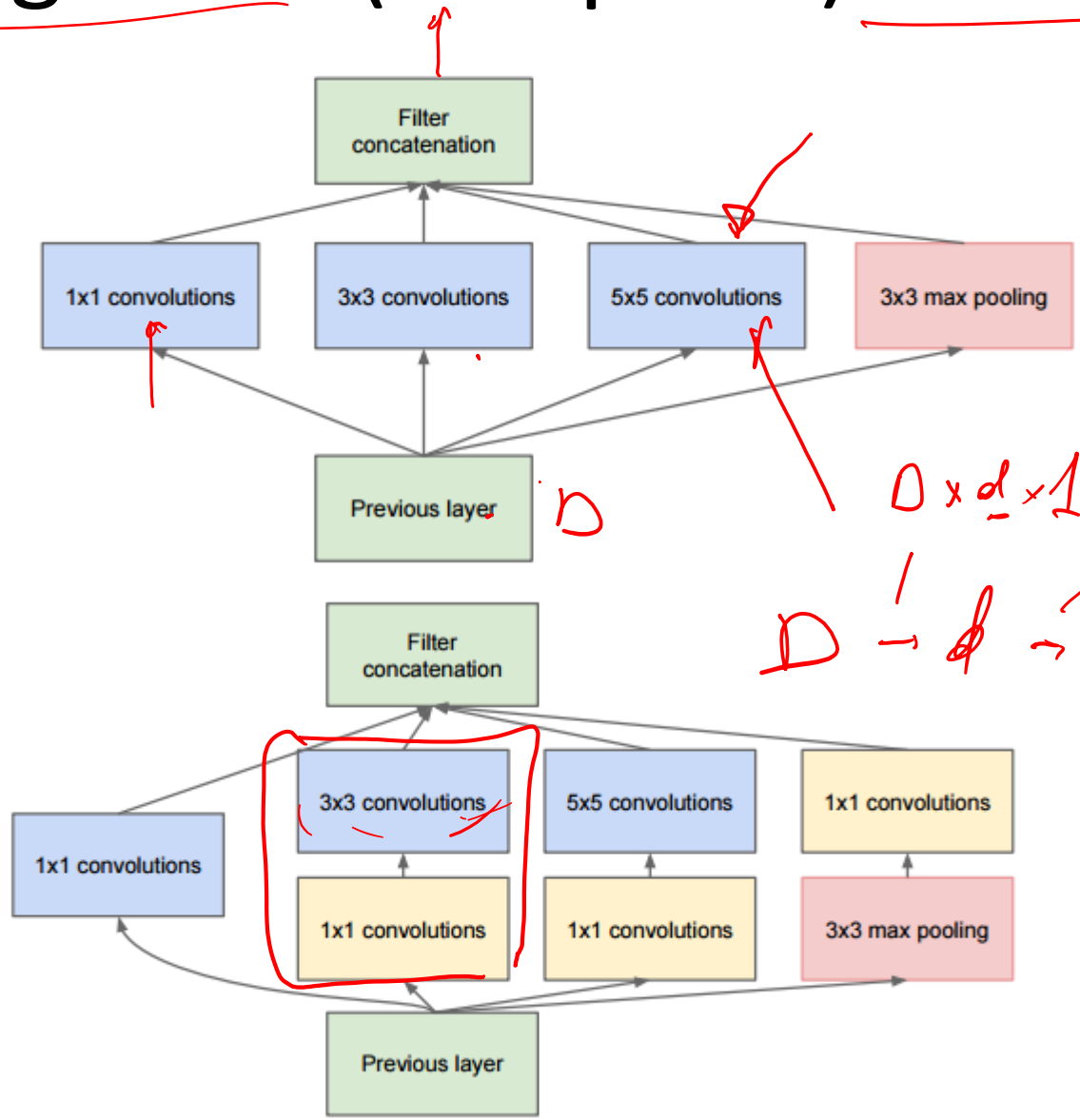
VGGNet (2014)

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

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd) TOTAL params: 138M parameters

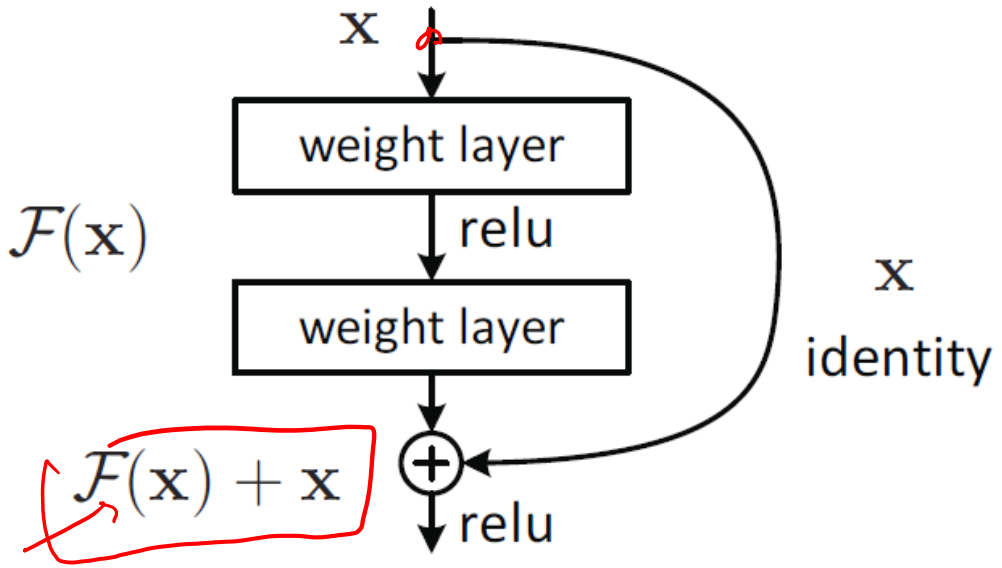
GoogLeNet (Inception) 2014

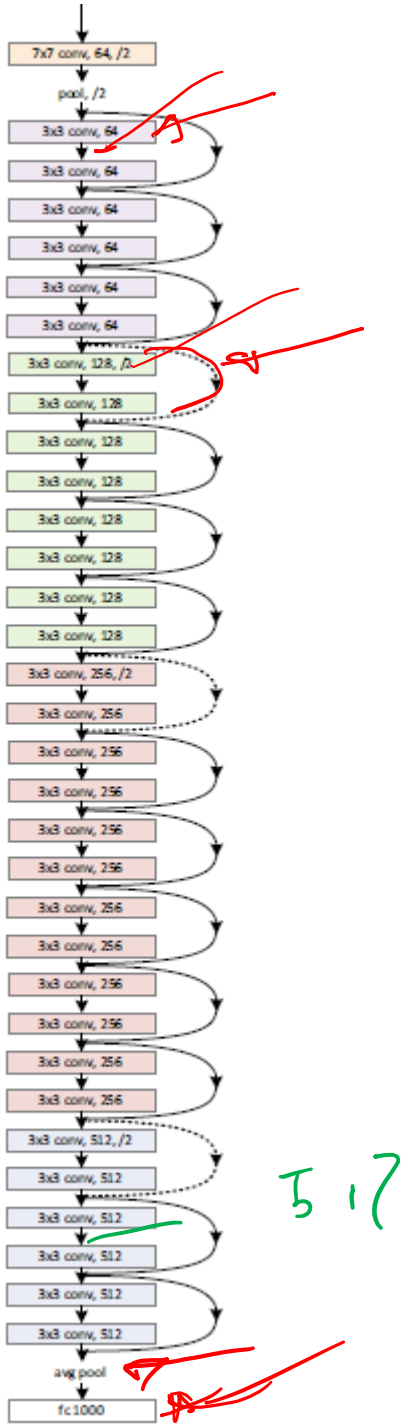




ResNets (Residual Nets, 2015)

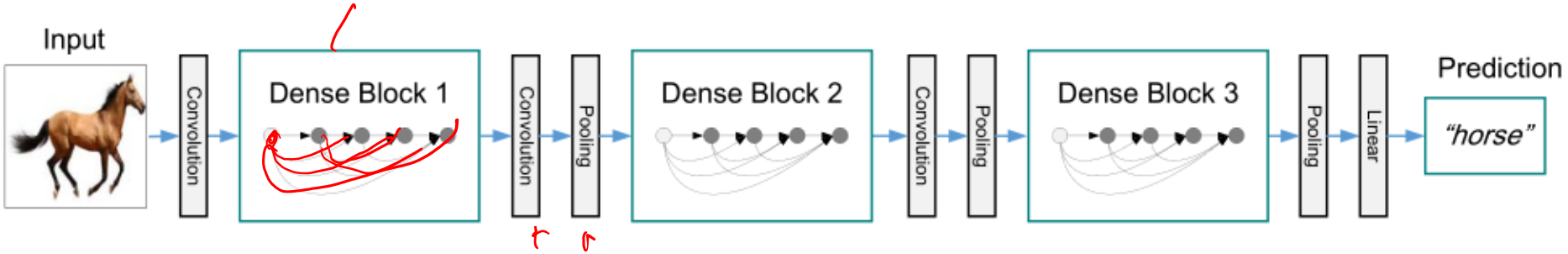
- Add bypases to ease gradient flow
- Networks with more than 150 layers!
- 3x3 convs with number of feature maps doubling after every pooling



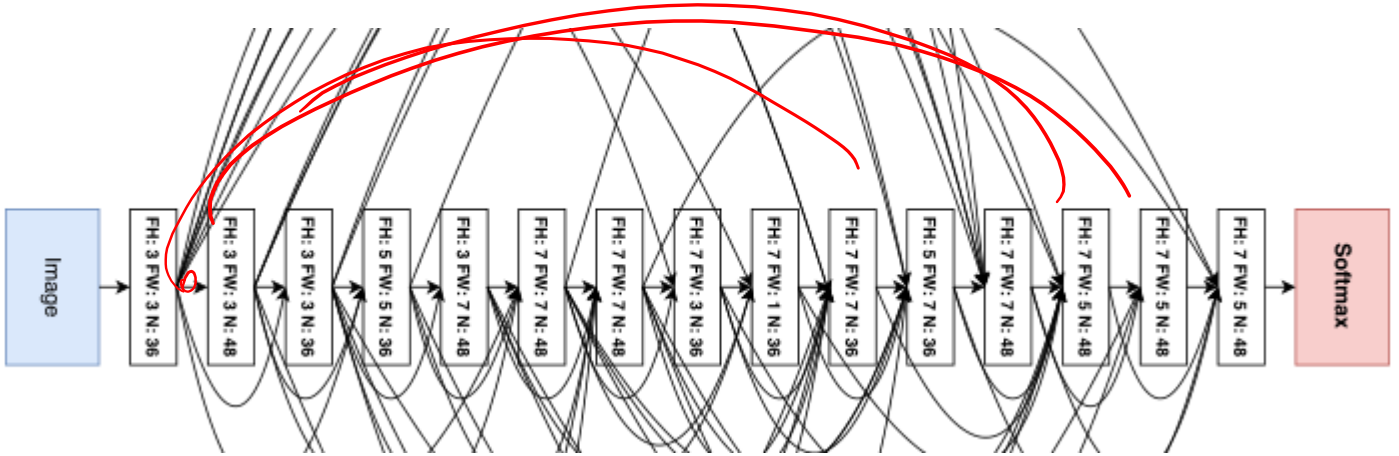


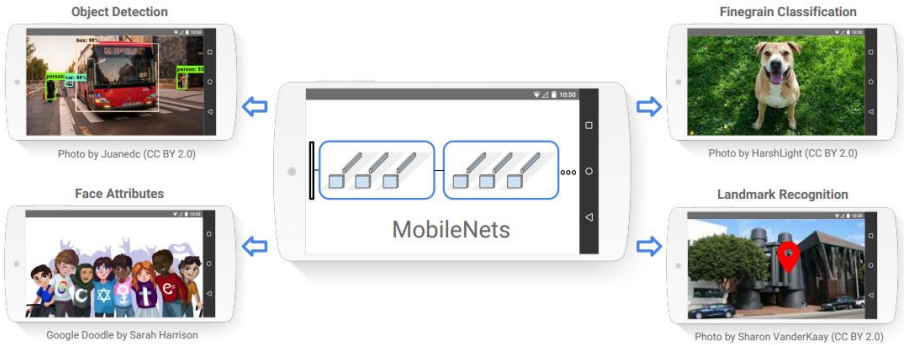
Quest for models

- DenseNets (https://arxiv.org/pdf/1608.06993)



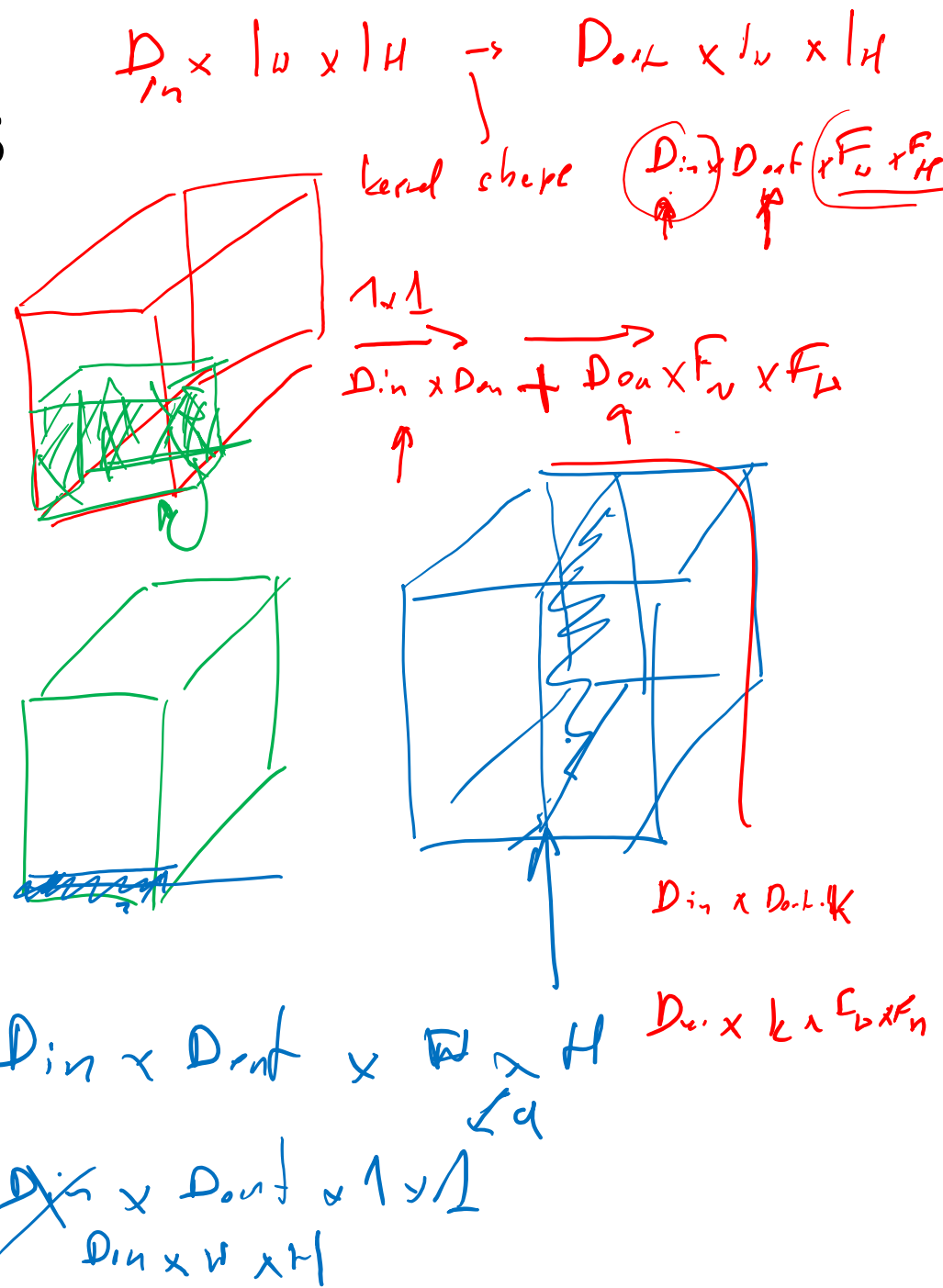
- Automatic architecture search (https://arxiv.org/pdf/1611.01578)





MobileNets

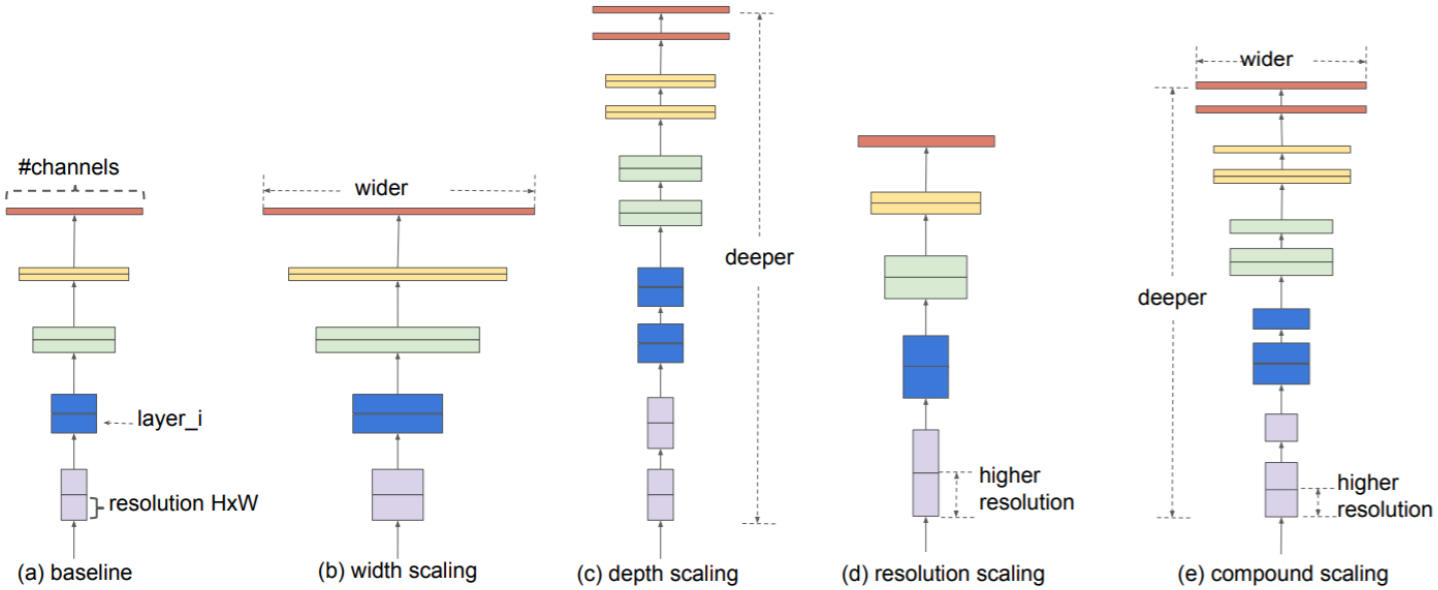
1. Std convolution
   Filter size: $D_{in} \times D_{out} \times W \times H$
   Matmuls per pixel: $D_{in} \times D_{out} \times W \times H$
2. Depthwise convolution
   (each filter sees only one input channel)
   Filter size: $D_{out} \times W \times H$ (nb: no D_{in})
   Matmuls per pixel: $D_{out} \times W \times H$
3. MobileNet: Depthwise + 1x1 convs
   Why? Depthwise: fast, doesn't mix channels
   1x1: mixes channels, still small

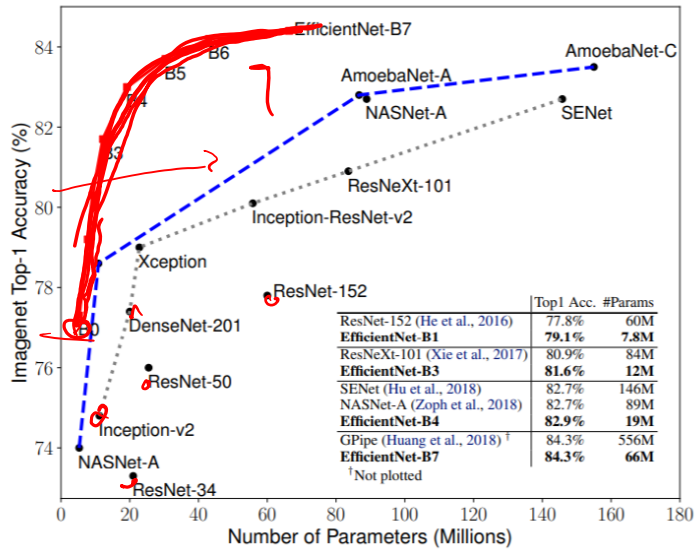


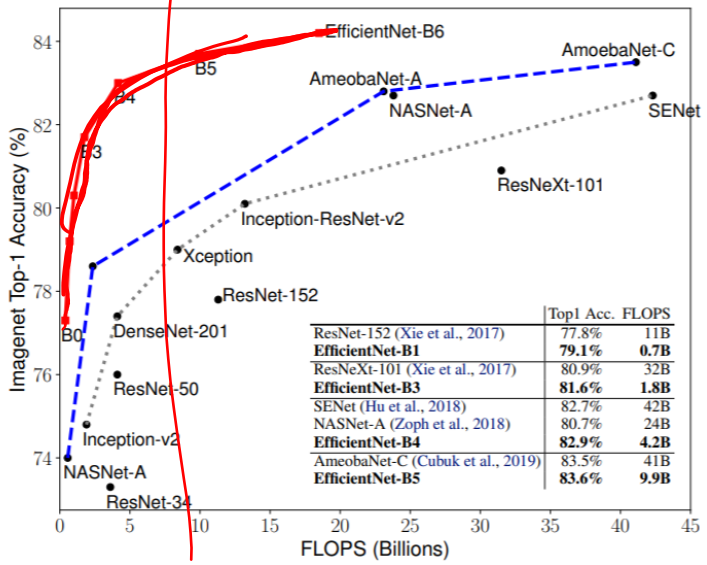
https://arxiv.org/pdf/1704.04861.pdf, https://arxiv.org/pdf/1801.04381.pdf

EfficientNets

- Laws for scaling MobileNets and Resnets
- Optimized for accuracy under limited size/FLOPs







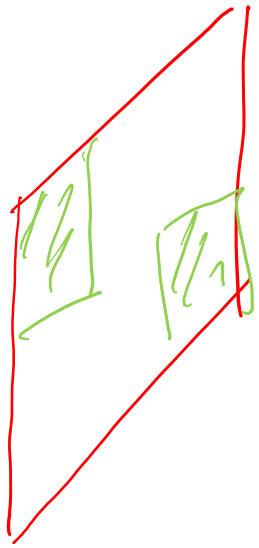
https://arxiv.org/pdf/1905.11946.pdf, https://arxiv.org/pdf/2104.00298.pdf

Which ConvNet to use?

- Take current best on Imagenet, use scaling laws to choose one based on image resolution and FLOP budget
- Many networks are available for download (tensorflow, pytorch or caffe model registries)

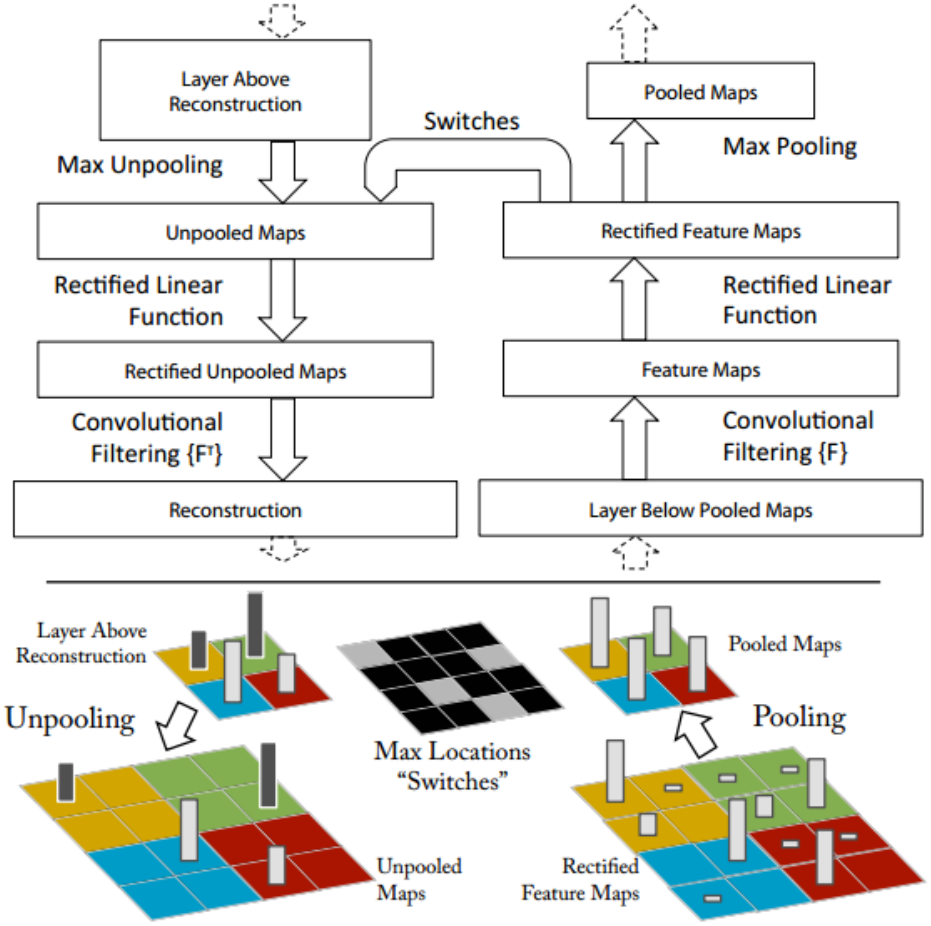
- Take a pretrained net and retrain only the last layers!
- On ImageNet (1M images) data augmentation and regularization is as important as the network!

Conv recap



- Shared parameters
- Limited field of view
- Less computation

Visualizing CNN features



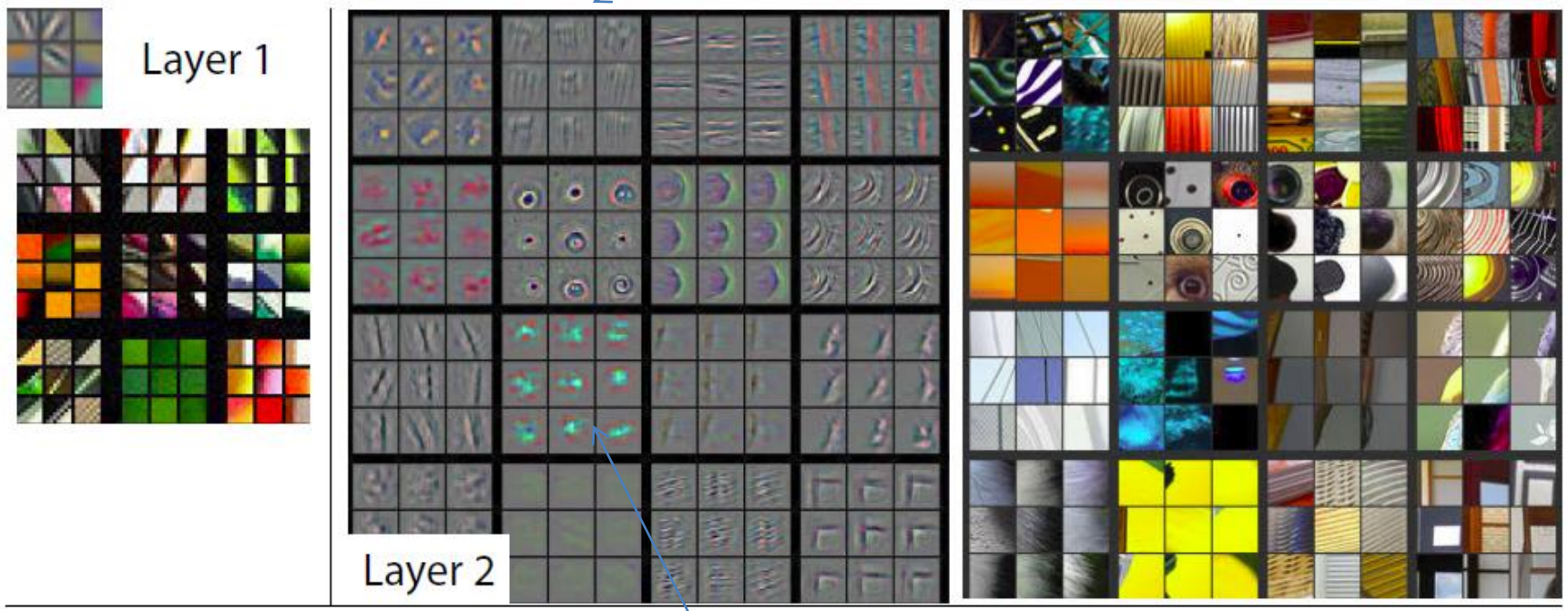
Main idea:
1. Do a forward propagation,
2. Save state of max-poolings,
3. „Deconvolve”: trace back the activations of chosen units to inputs

Note: similar to gradient computation, but applies ReLU during deconvolution
(gradient would use the saved jacobian of relu)

M. Zieler, “Visualizing and Understanding Convolutional Networks”

Low-level features

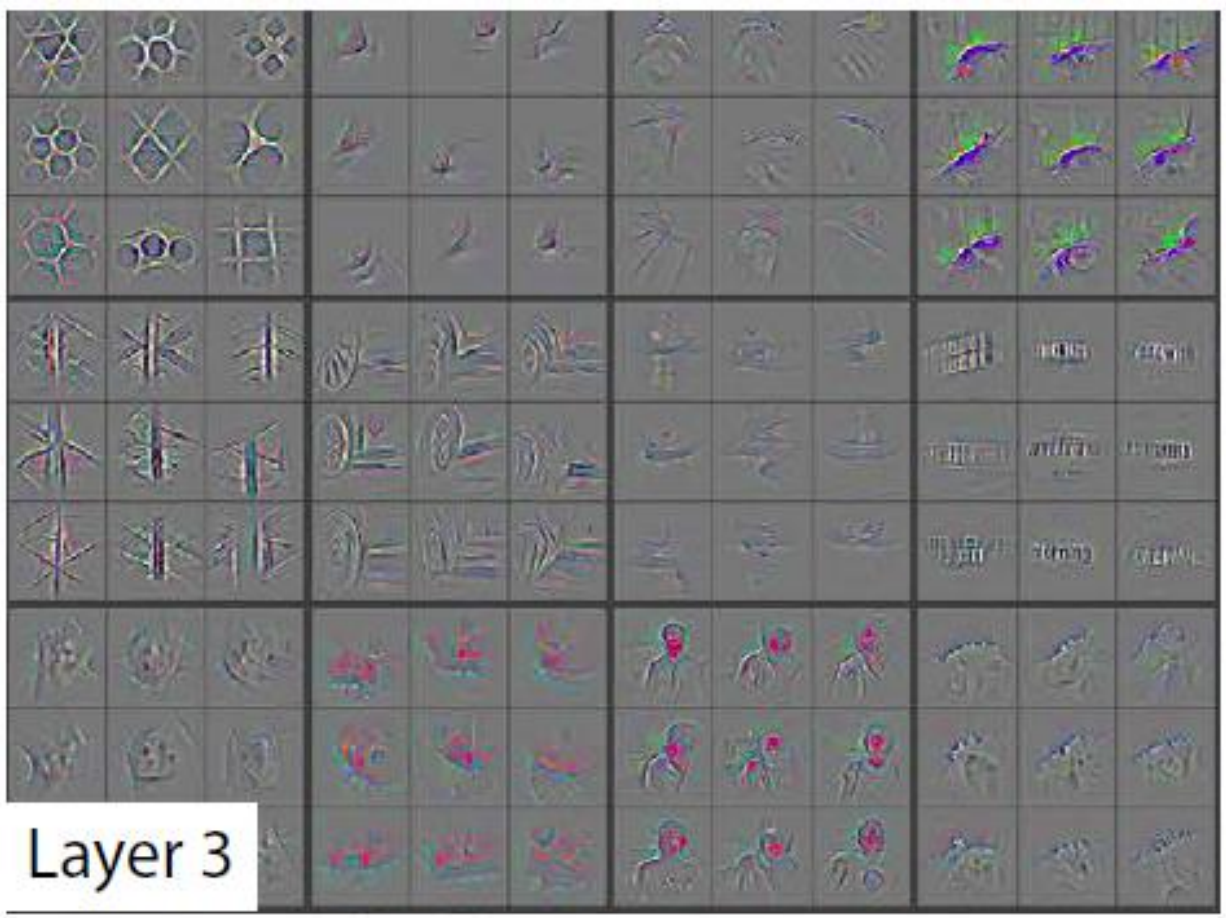
What the neuron (feature-detector looks for)



What images are selected by the neuron

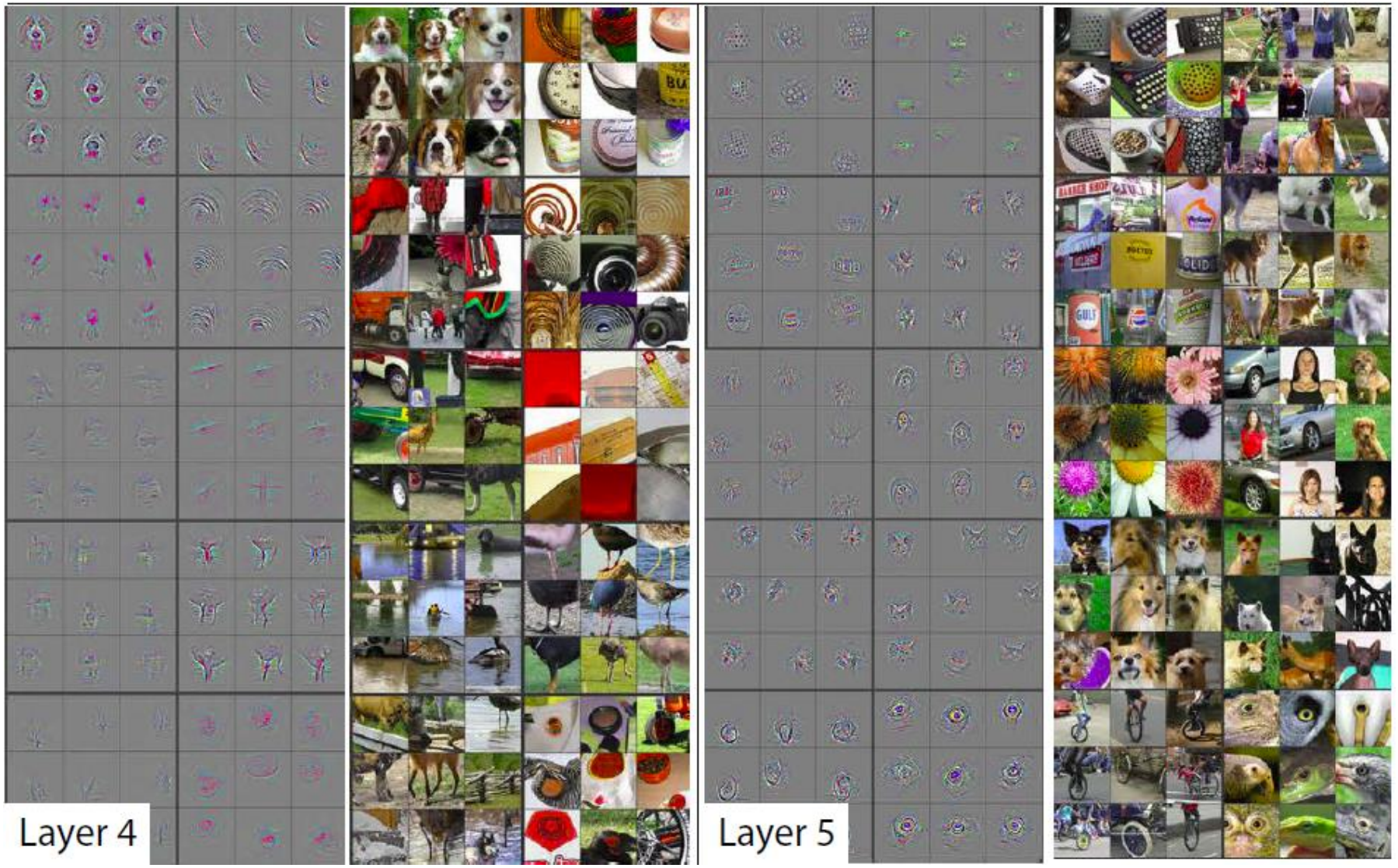
M. Zieler, “Visualizing and Understanding Convolutional Networks”

Mid-level features



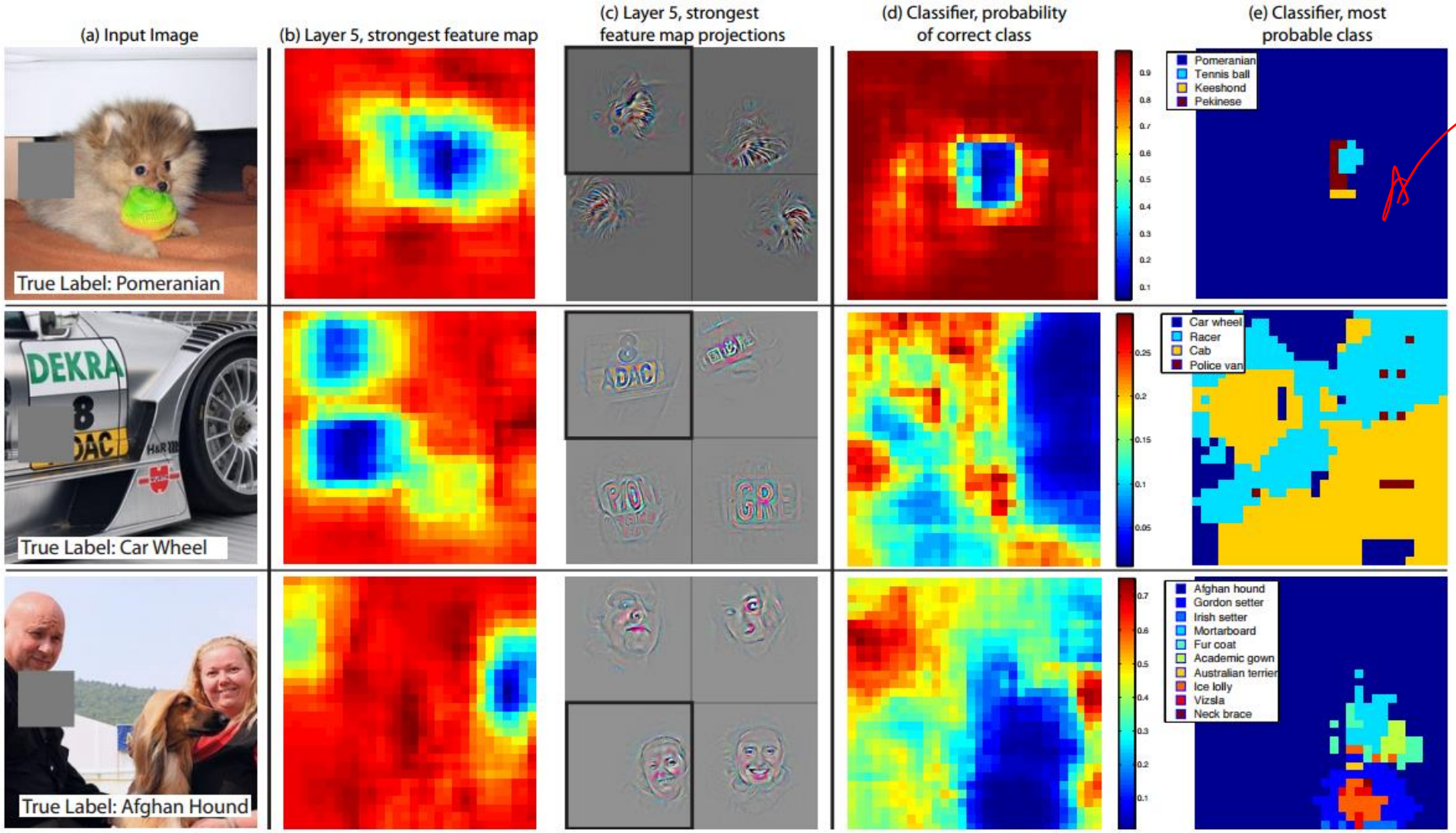
M. Zieler, “Visualizing and Understanding Convolutional Networks”

High-level features



M. Zieler, “Visualizing and Understanding Convolutional Networks”

Where Convnets look?



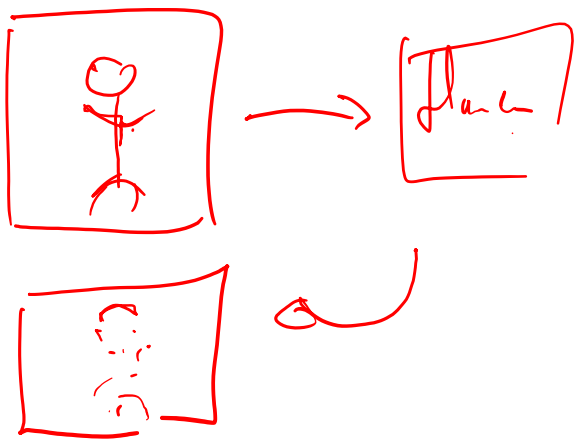
How to understand a model?

Look at the prediction

↳ approximate the model a given input with a linear function (Taylor expansion)

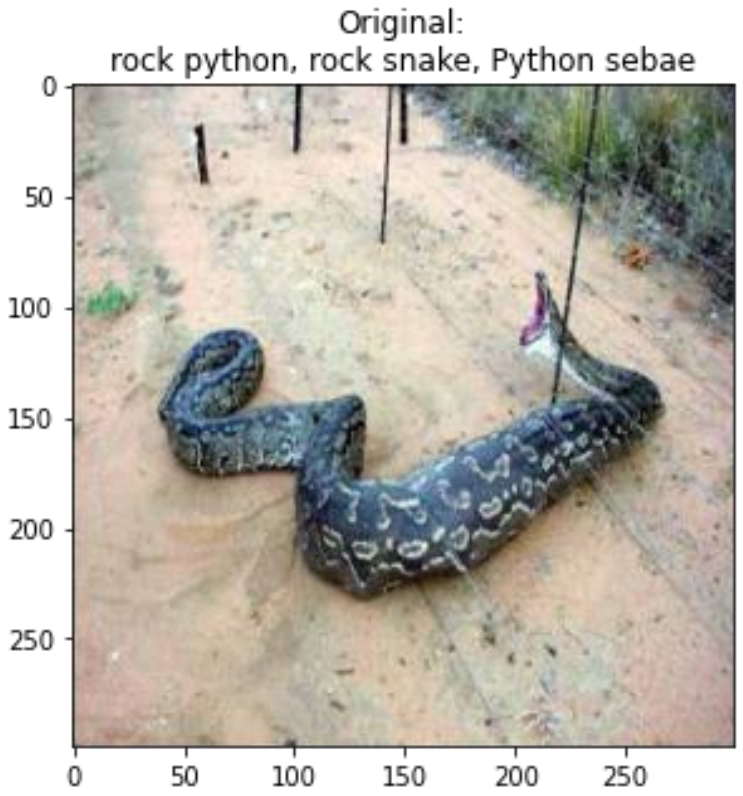
$$f(x + \Delta x) \approx f(x) + \Delta x \frac{\partial f}{\partial x}$$

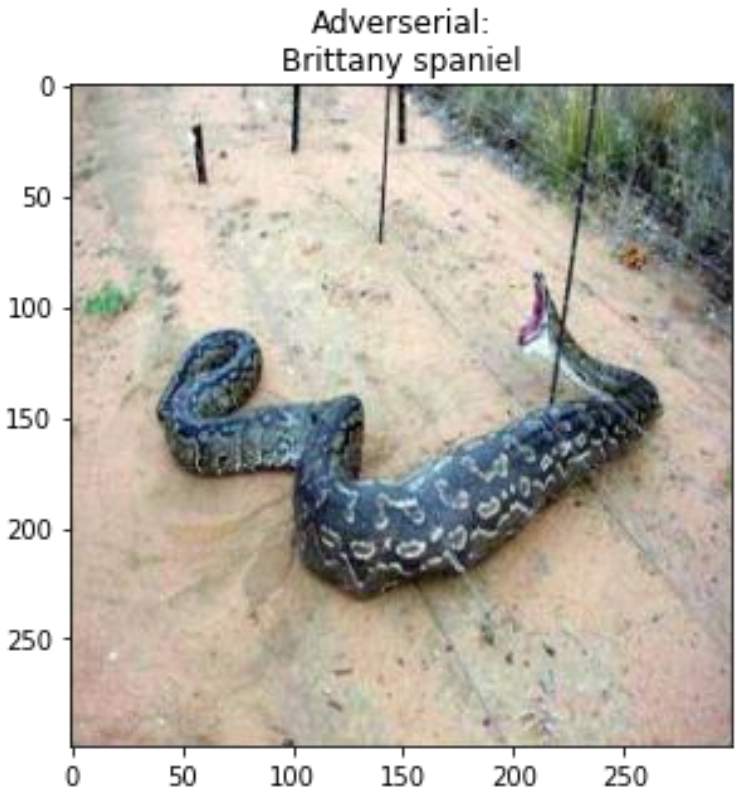
Gradient saliency map

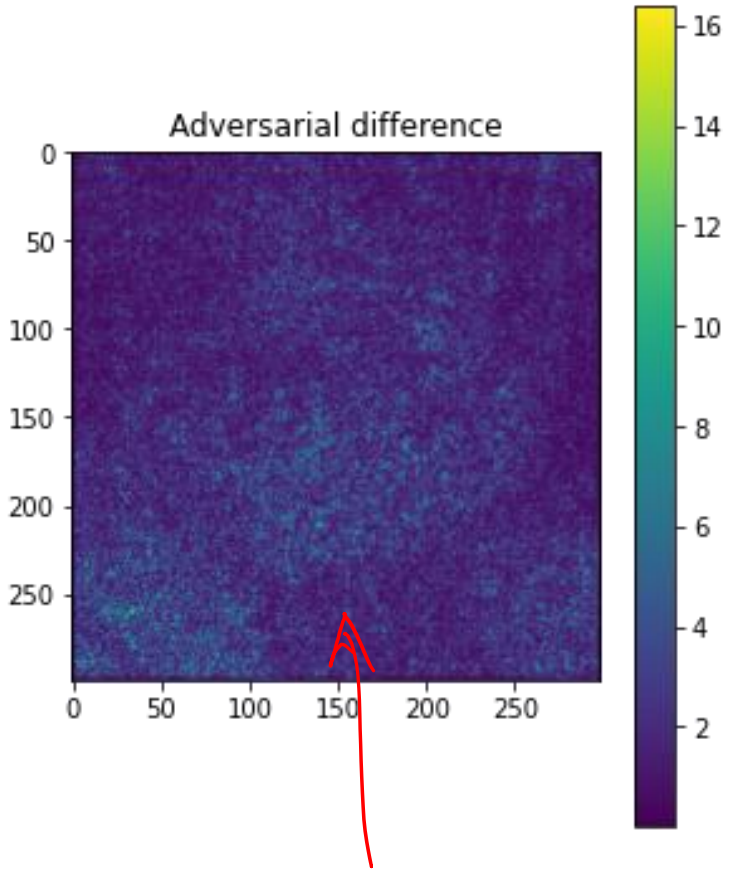


for stability we may want $\frac{\partial S_c(x)}{\partial x}$ is small

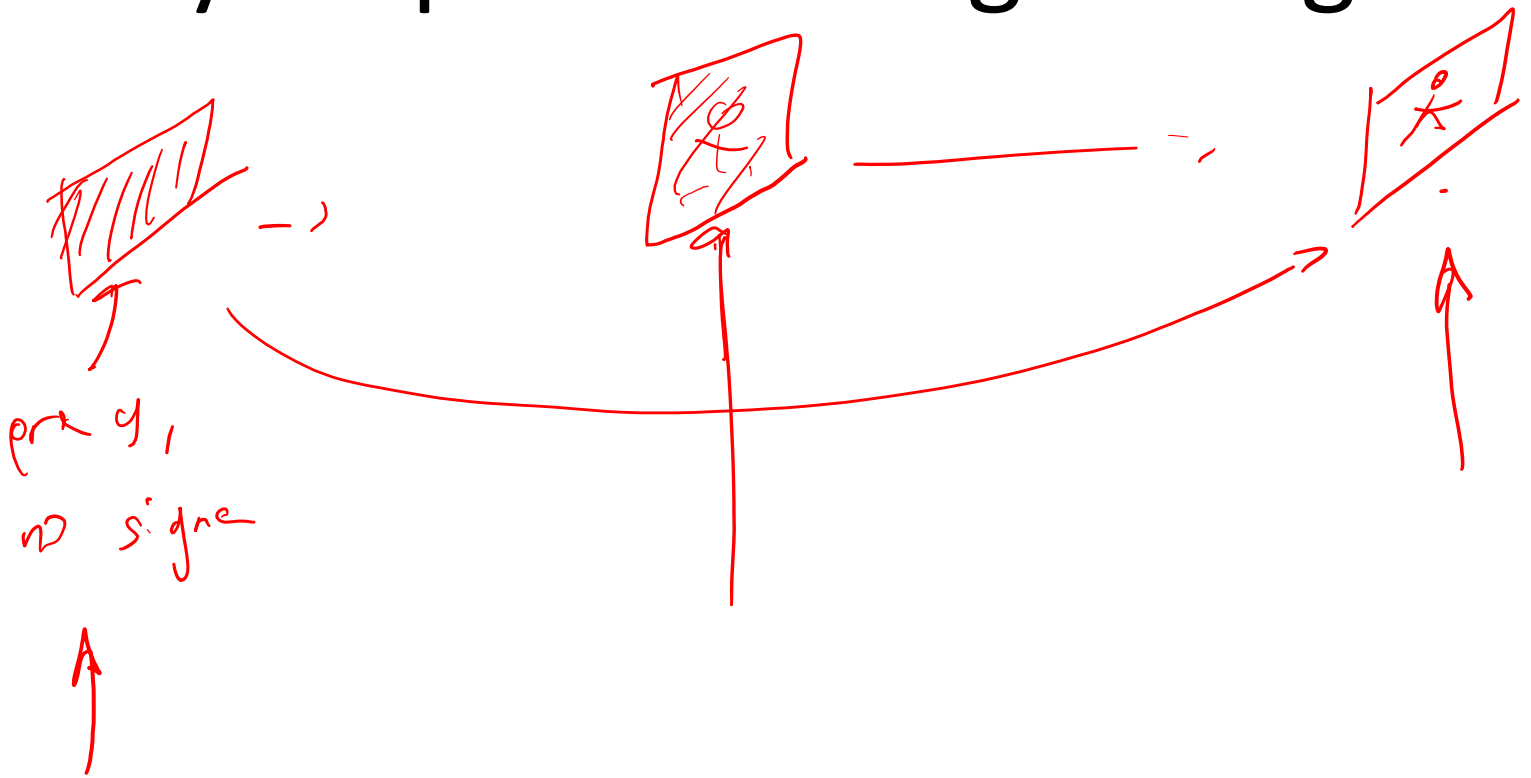
Adversarial Examples







Saliency maps with integrated gradients



LIME: local surrogate models

↳ sort of Taylor expansion -

build and evaluate a surrogate model ←

↳ unlike Taylor / gradient the surrogate has

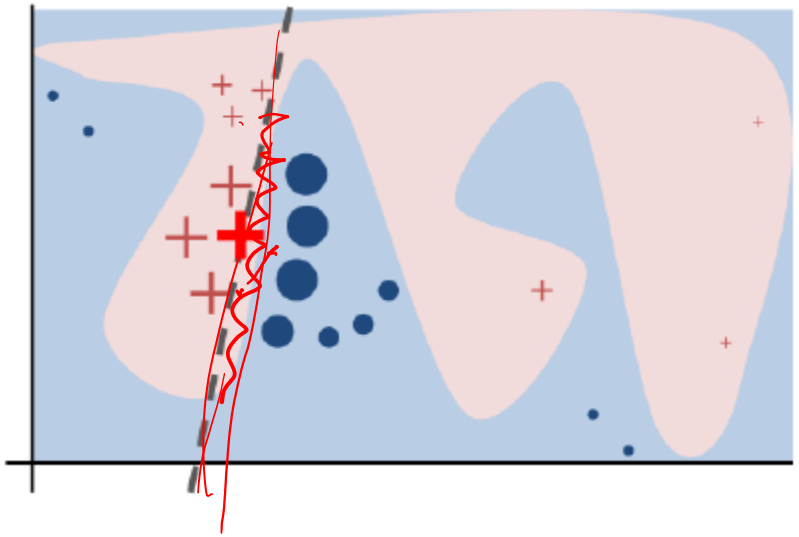
to model in or large, non-epsilon neighborhood ≠

x_q

x	y
$x_q + \Delta_1$	$model(x + \Delta_1)$
$x_q + \Delta_2$	$model(x + \Delta_2)$
⋮	⋮
$x_q + \Delta_N$	

→ Train a model on this table.

LIME details



$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \; \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

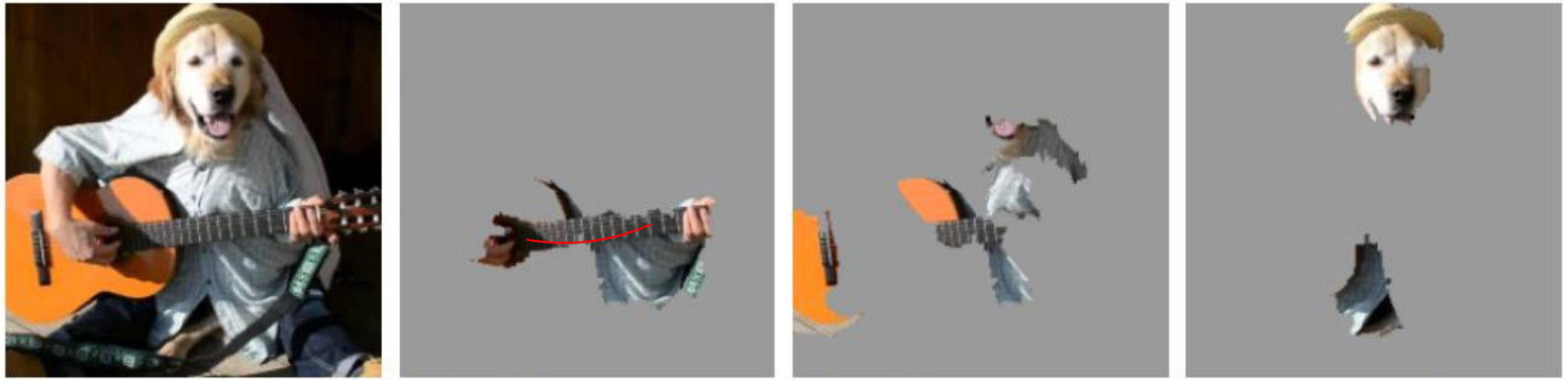
family of surrogates

not

neighborhood

fun

LIME on images



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Beware: some methods lie

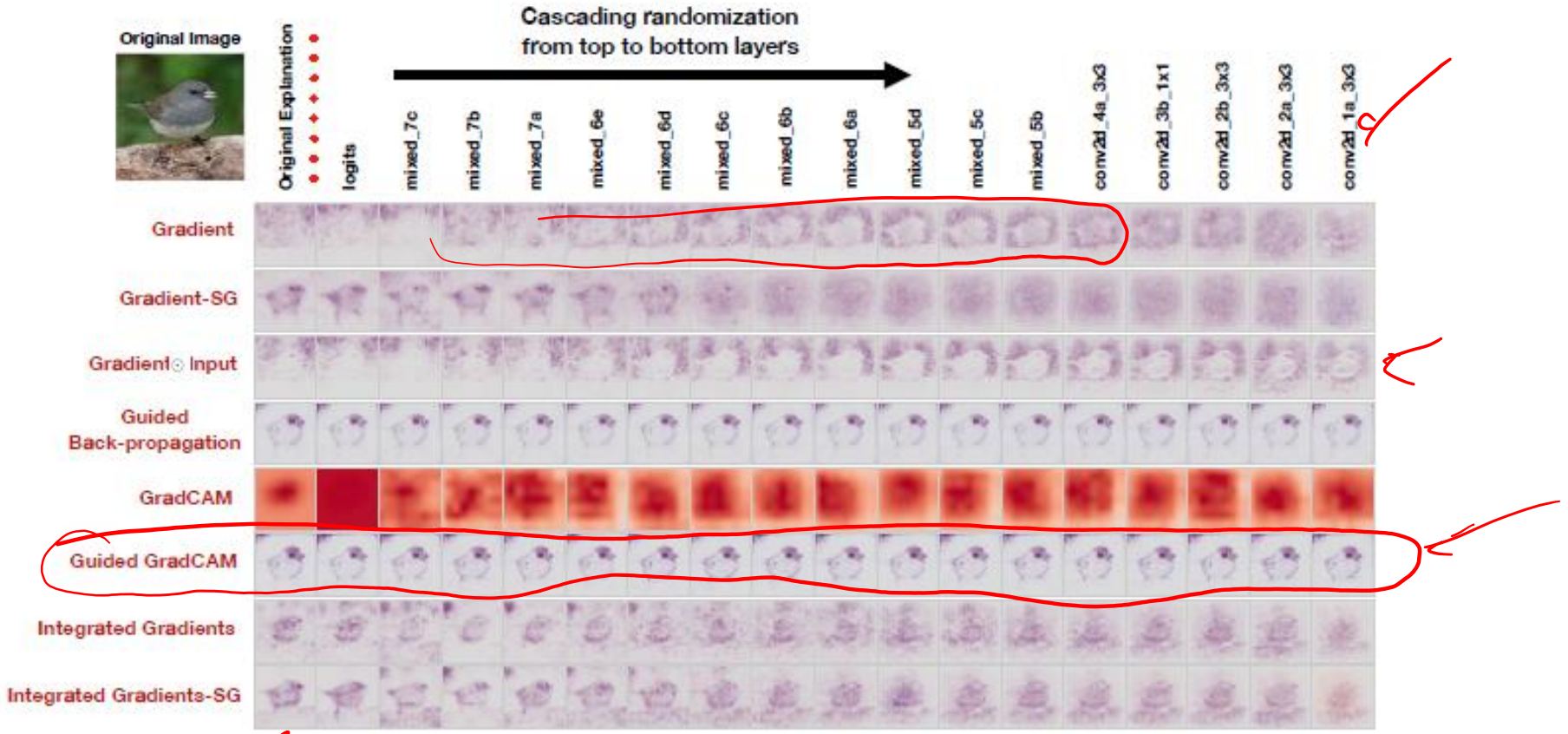


Figure 2: **Cascading randomization on Inception v3 (ImageNet).** Figure shows the original explanations (first column) for the Junco bird. Progression from left to right indicates complete randomization of network weights (and other trainable variables) up to that 'block' inclusive. We show images for 17 blocks of randomization. Coordinate (Gradient, mixed_7b) shows the gradient explanation for the network in which the top layers starting from Logits up to mixed_7b have been reinitialized. The last column corresponds to a network with completely reinitialized weights.

http://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps.pdf

Model explanations, further reading

- Integrated gradients: https://arxiv.org/abs/1703.01365v2
- Important analysis of saliency maps http://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps.pdf
- LIME: https://arxiv.org/abs/1602.04938v3
- SHAP (similar to LIME, more stable and justified): http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Sidenote Style Transfer

Find image (backpropagation toward pixels) minimizing:

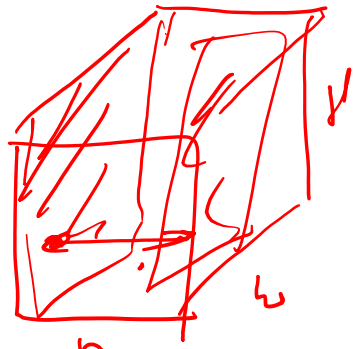
$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_S =$$

$$= (F(C) - F(I))^2 + \left(G(S) - G(I)\right)^2$$

Where:

$F(x)$ features of a CNN layer on image x

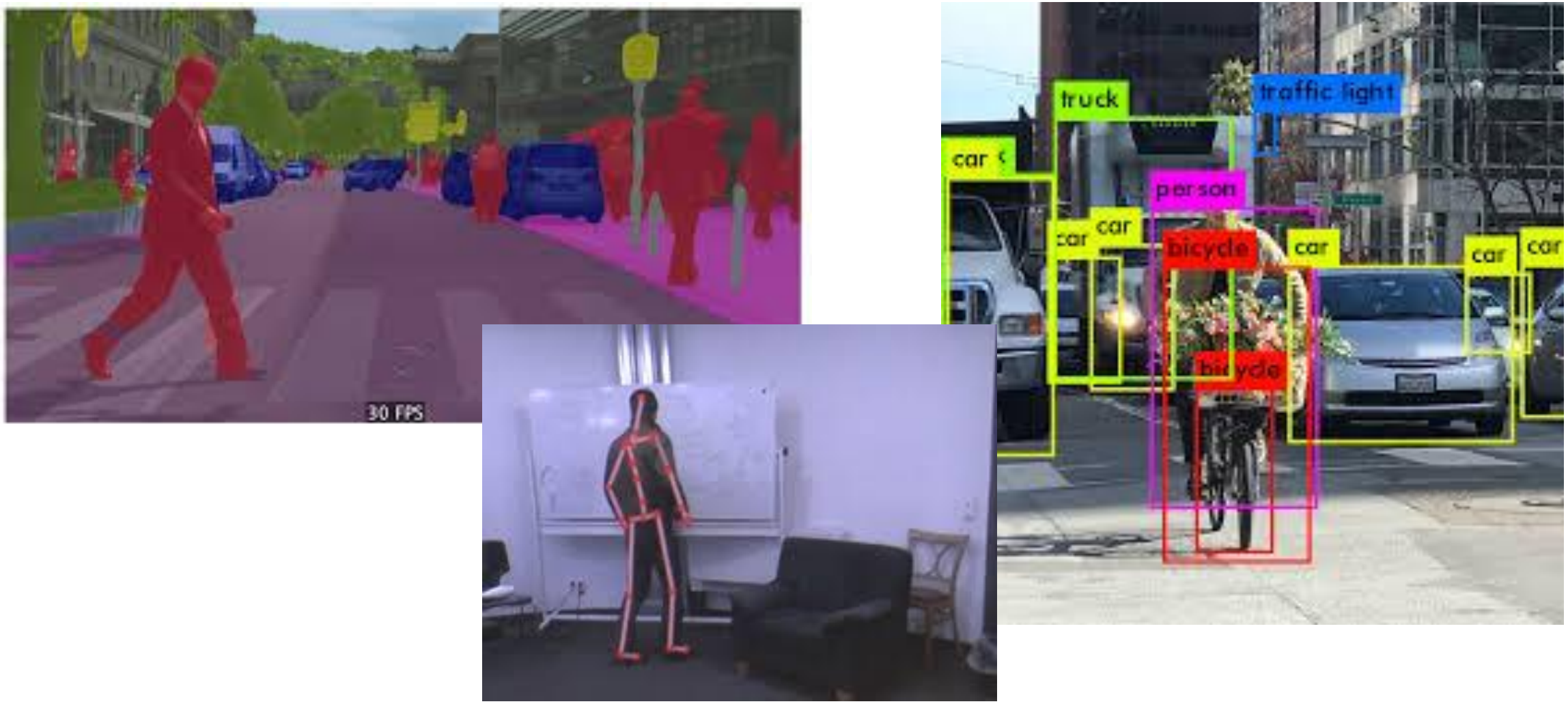
$G(x)$ matrix of correlations between a layer's features over pixels of image x



Gatys et al., „A Neural Algorithm of Artistic Style", 2015

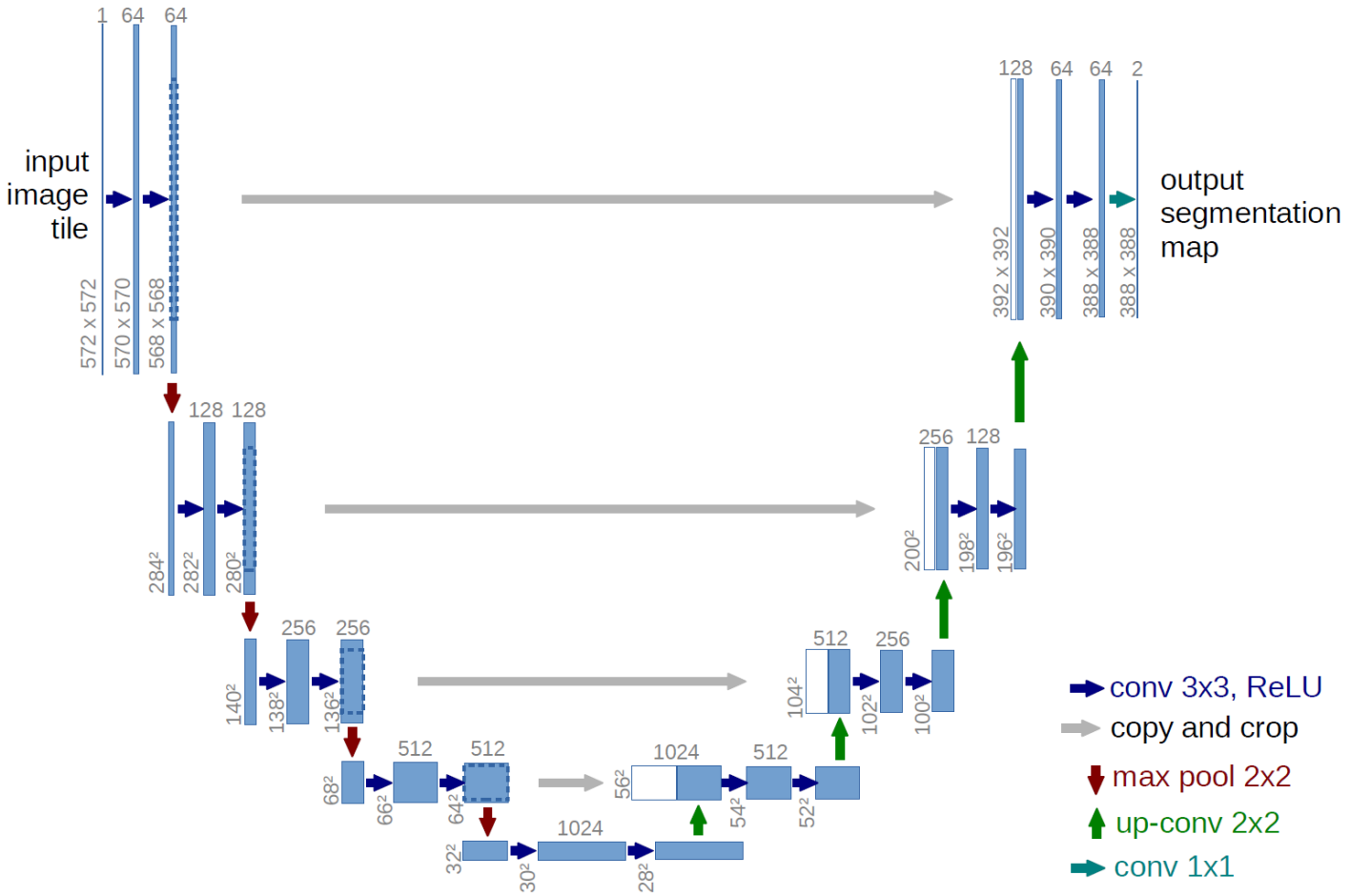
SPECIALIZED CONVNETS

More image processing tasks



Unet: Pixel labeling intro

U-Net: Pixel labeling

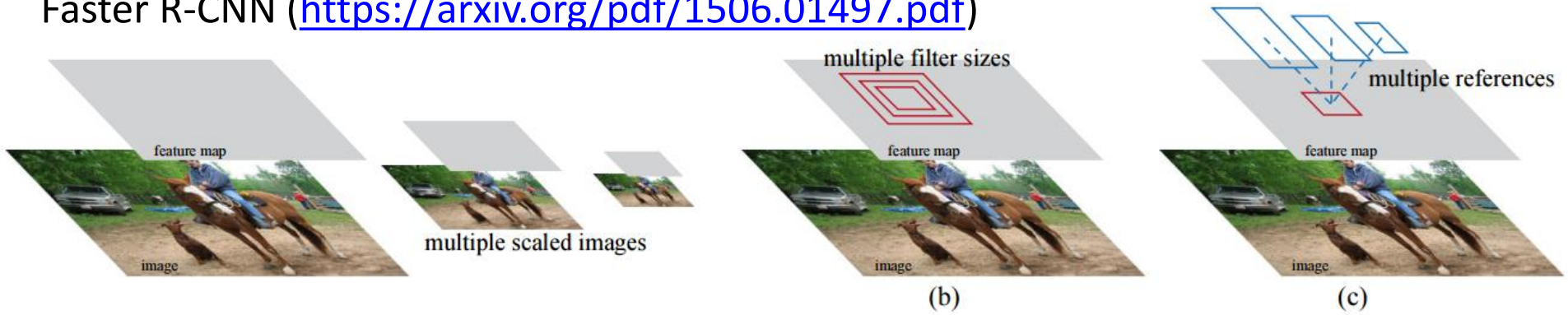


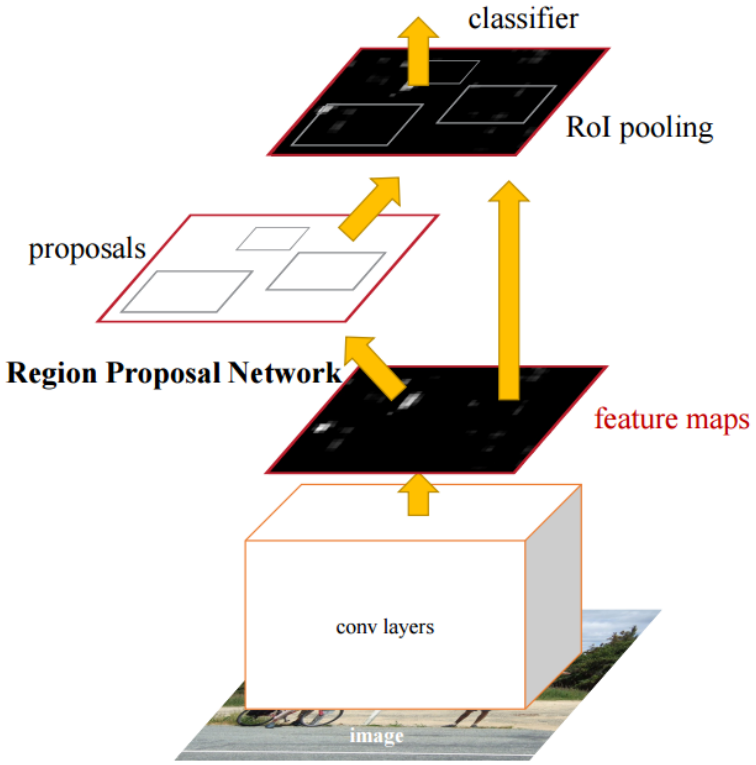
From Classification to Detection

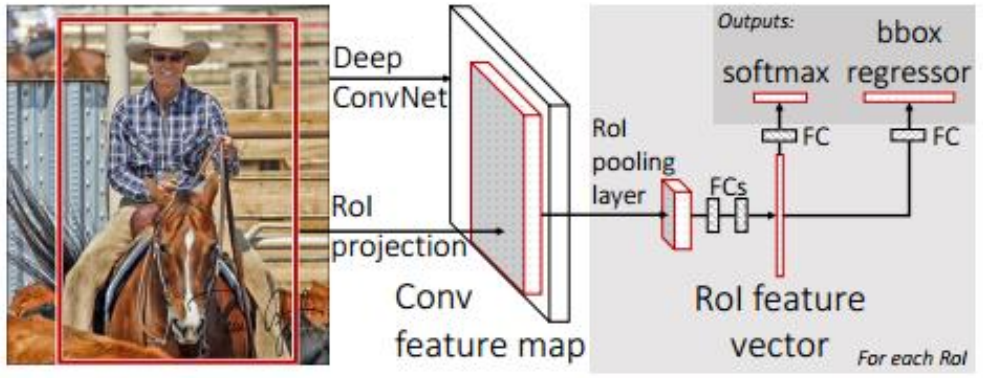
Object detection and segmentation

Fast R-CNN (https://arxiv.org/pdf/1504.08083.pdf)

Faster R-CNN (https://arxiv.org/pdf/1506.01497.pdf)







YOLO – fast object detection

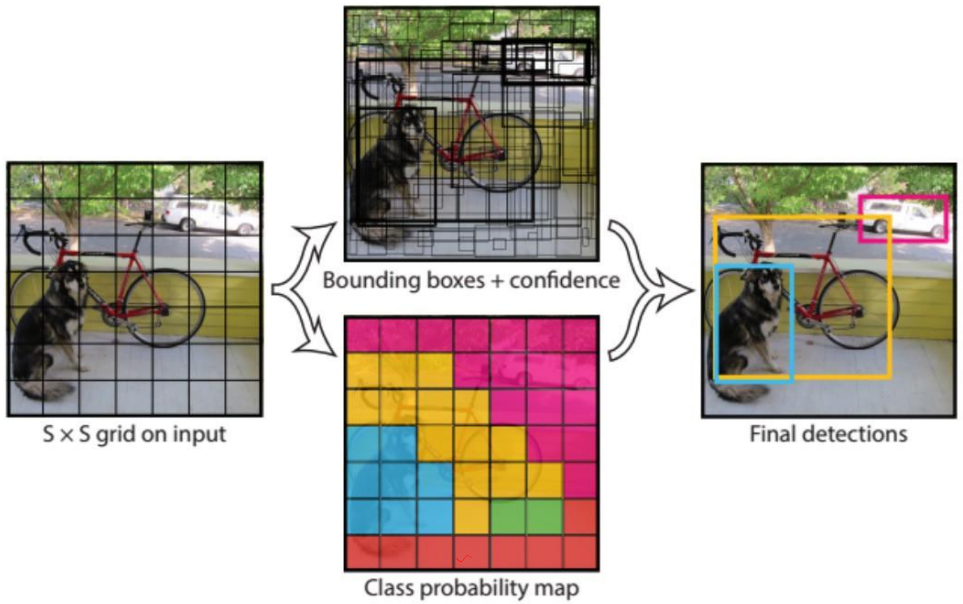


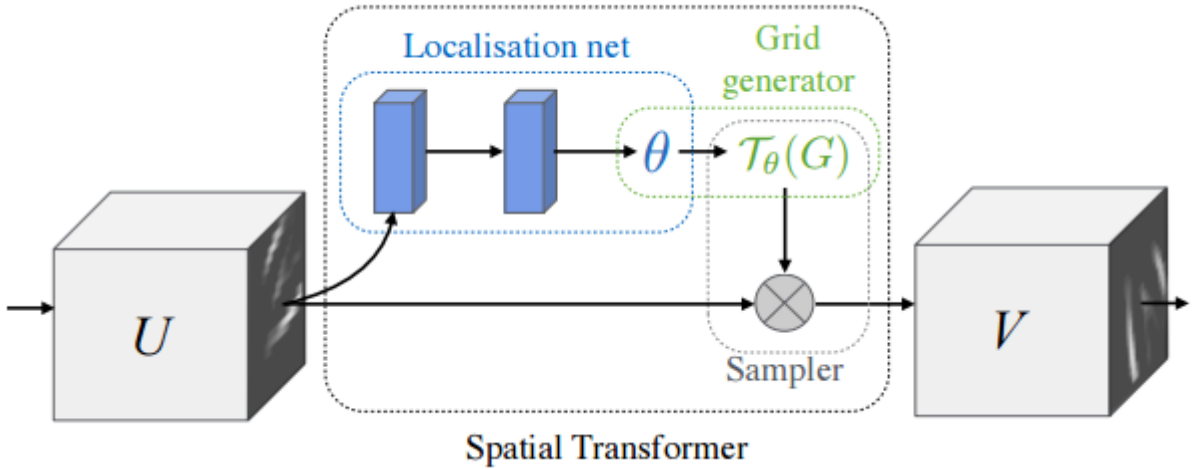
Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

- No region proposal and looping over ROIs!
- In a fully convolutional net predict for each pixel of a feature map:
  - Objectness – is something there
  - Bounding Box – how large it is

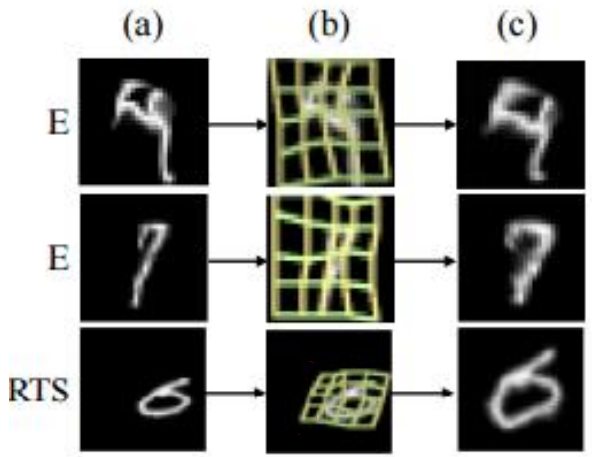
https://arxiv.org/pdf/1506.02640v5.pdf

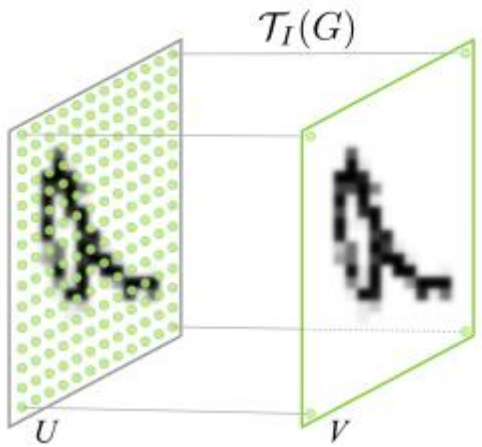
Spatial Transformer Networks

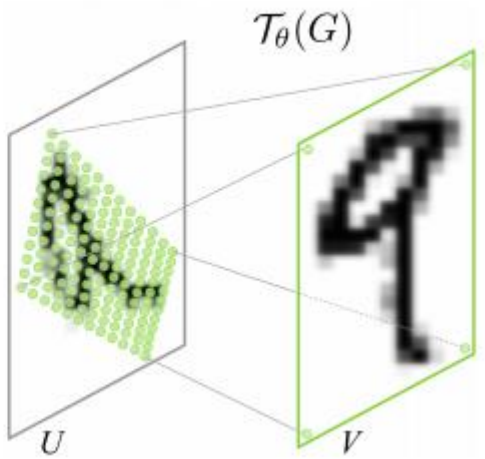
https://arxiv.org/pdf/1506.02025.pdf

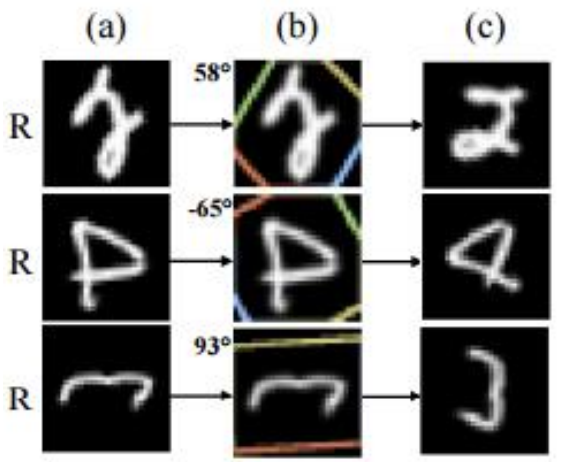


Spatial Transformer

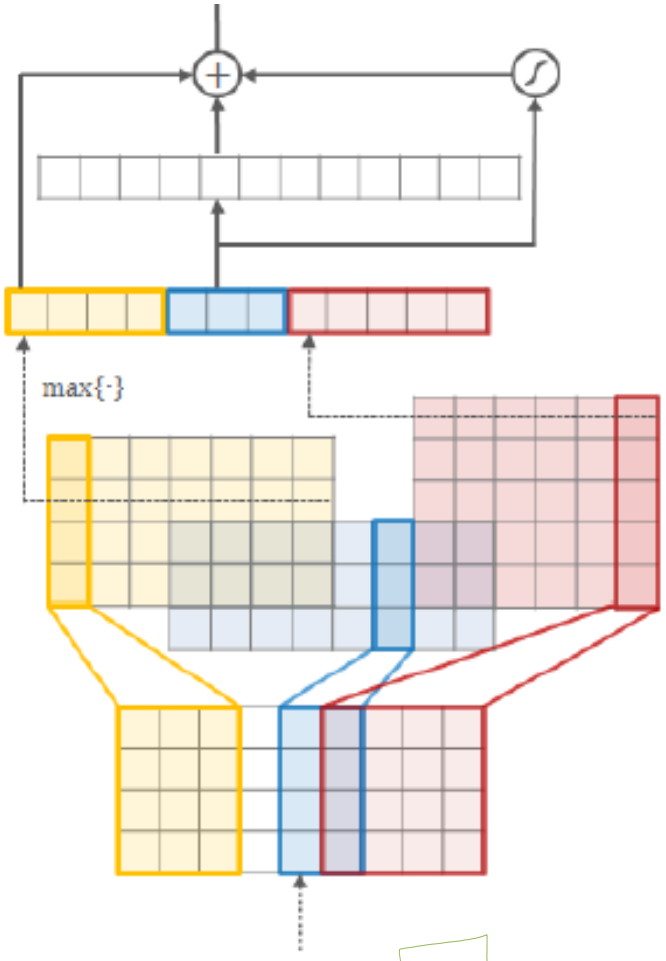








CNN beyond images
Or character-to-word embeding



Highway layers – very nonlinear transformation of data

Glue best word representations together

Convolutional filters of varying lengths. Can react to pre-, in-, and post-fixes of words

Character embedings concatenated into a matrix

Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, “Character-Aware Neural Language Models,” *arXiv:1508.06615 [cs, stat]*, Aug. 2015.

Additional references

- Goodfellow Chp. 9
- http://cs231n.github.io
- How transferable are features in deep neural networks? studies the transfer learning performance in detail, including some unintuitive findings about layer co-adaptations.
- CNN Features off-the-shelf: an Astounding Baseline for Recognition trains SVMs on features from ImageNet-pretrained ConvNet and reports several state of the art results.