昆山杜克大学
DUKE KUNSHAN UNIVERSITY

convolutional_neural_networks (CNN)

stats403_deep_learning

spring_2025

lecture_4

4.1 more on regularization

regularization

- neural networks can approximate “any” continuous function
- a big number of combinations of parameters that we need to choose from
- need to restrict the capacity of a neural network

regularization

suppose the objective function $J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y})$ in training is

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \Omega(\boldsymbol{\theta})$$

penalty term

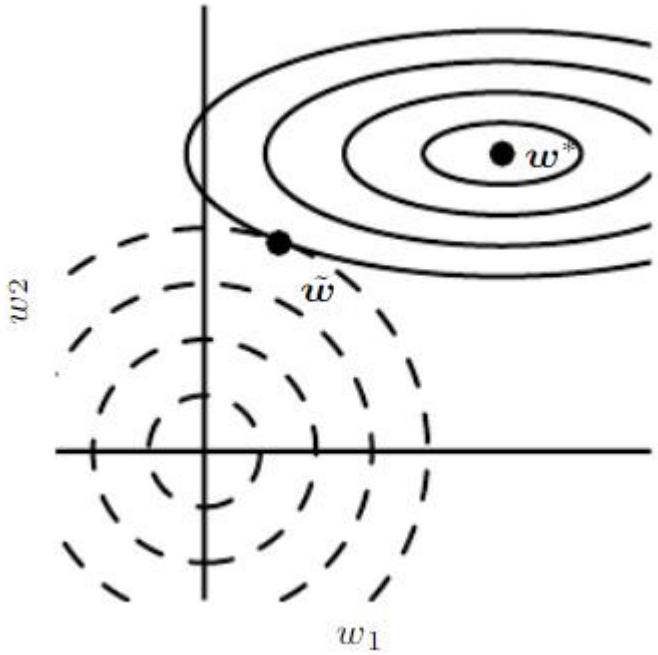
L2 regularization

ridge regression / Tikhonov regression

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_2^2 / 2$$

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) + \frac{\alpha}{2} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{w}$$

L2 regularization



question: why does the solution move more significantly along the direction of w1?

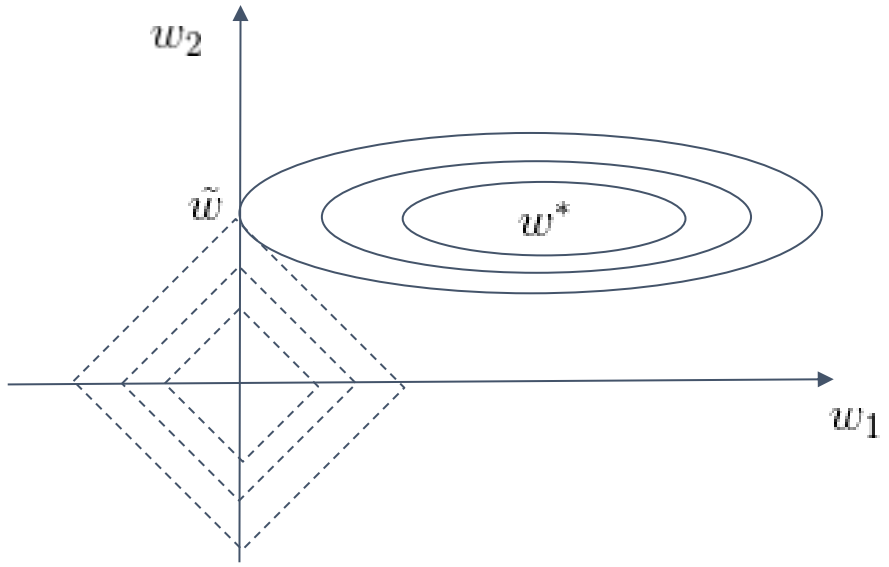
L1 regularization

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1$$

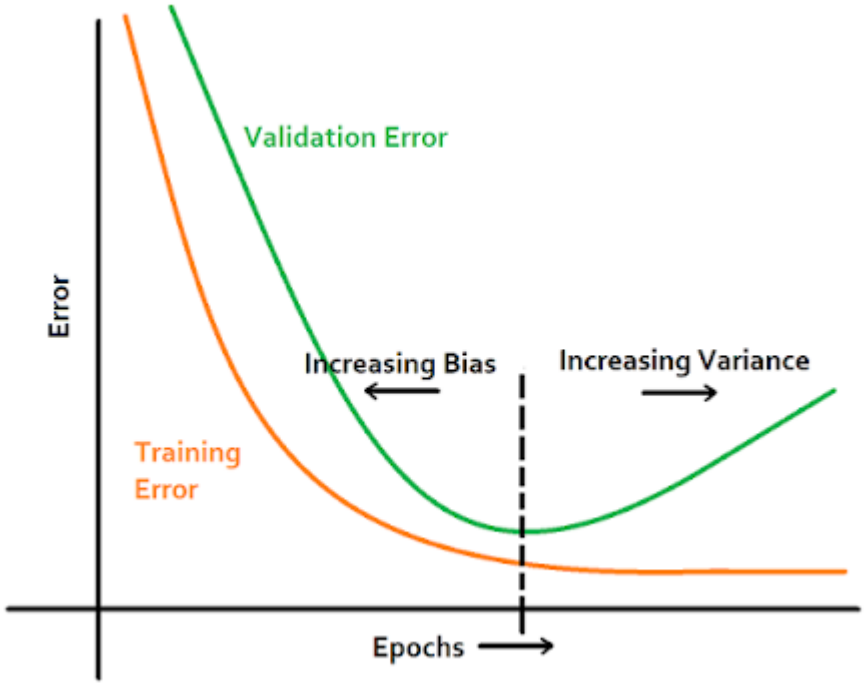
$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \|\boldsymbol{w}\|_1$$

L1 regularization

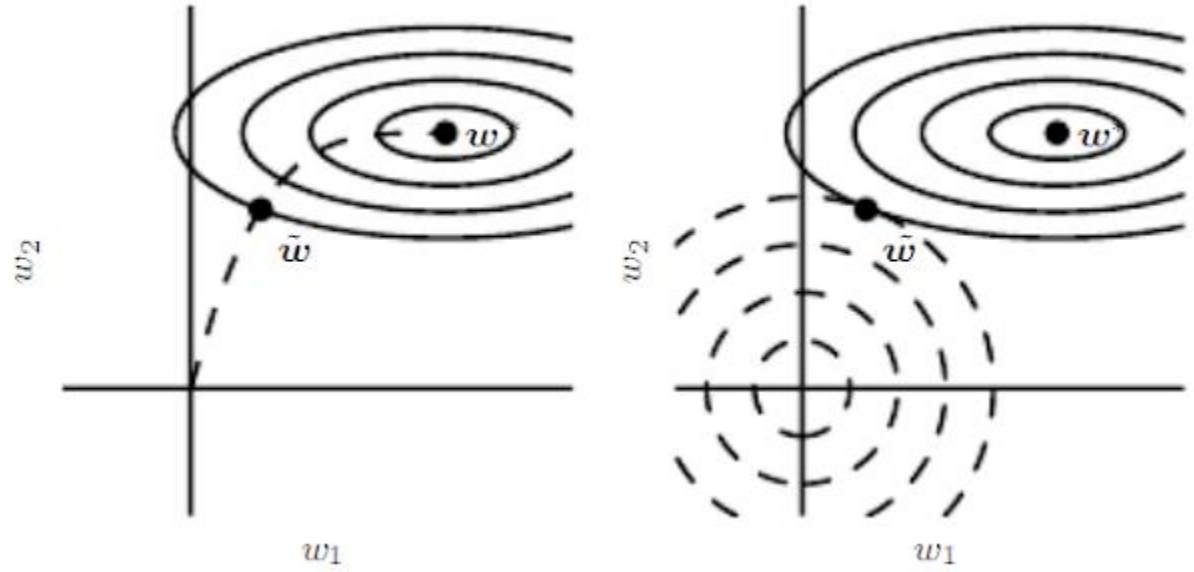
encourages sparsity of solution



early stopping



early stopping is similar to regularization



dataset augmentation

- One effective way to prevent overfitting is to use more data.
- In practice, we do not have an infinite amount of data.
- A practical solution is to generate data by ourselves.

dataset augmentation

original

padding

geometry

color

crop

morphology

blur

https://pytorch.org/vision/master/auto_examples/transforms/plot_transforms_illustrations.html

dataset augmentation

- mixup

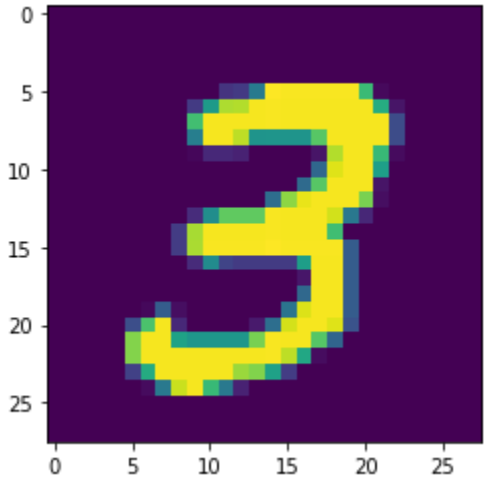


Zhang, H., et al. (2017). mixup: Beyond empirical risk minimization. ICLR 2018.

label smoothing

- instead of one-hot labels, we use a smooth label such as

```
[0.01, 0.01, 0.01, 0.91, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01]
```



preprocess image data

image credit: https://fastai1.fast.ai/vision.transform.html

preprocess image data

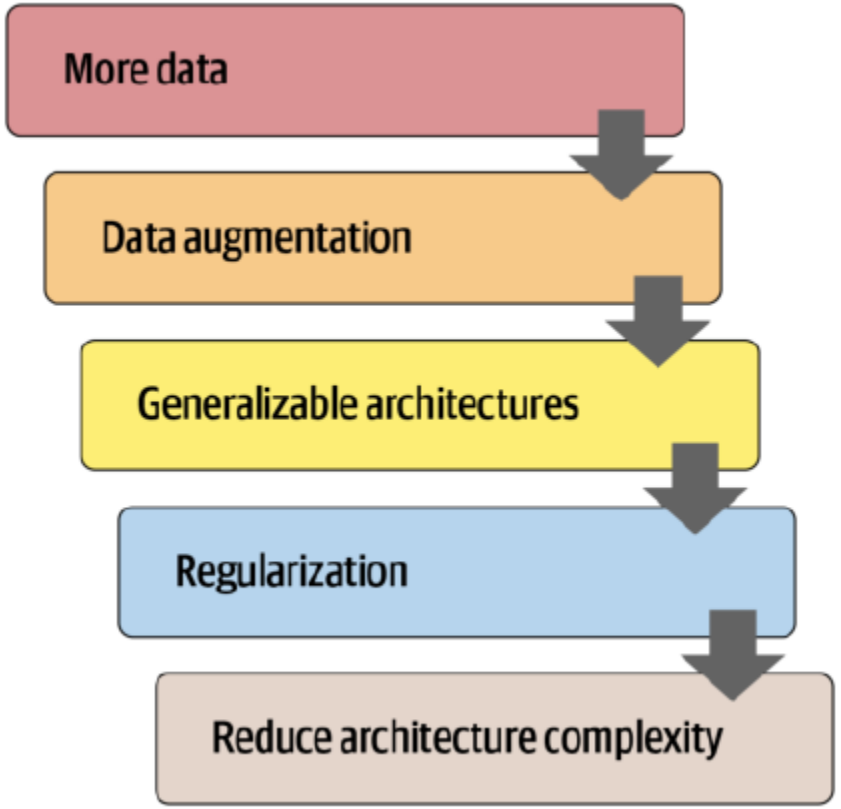
- normalizing:
  - before normalizing, the pixel values are from 0 to 255
  - need to apply a transformation so that each pixel has zero mean and unit variance

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	29	150	195	254	255	254	176	193	150	96	0	0	0
2	0	0	0	48	166	224	253	253	234	196	253	253	253	253	233	0	0	0
3	0	93	244	249	253	187	46	10	8	4	10	194	253	253	233	0	0	0
4	0	107	253	253	230	48	0	0	0	0	0	192	253	253	156	0	0	0
5	0	3	20	20	15	0	0	0	0	0	43	224	253	245	74	0	0	0
6	0	0	0	0	0	0	0	0	0	0	249	253	245	126	0	0	0	0
7	0	0	0	0	0	0	0	14	101	223	253	248	124	0	0	0	0	0
8	0	0	0	0	0	11	166	239	253	253	253	187	30	0	0	0	0	0
9	0	0	0	0	0	16	248	250	253	253	253	253	232	213	111	2	0	0
10	0	0	0	0	0	0	0	43	98	98	208	253	253	253	253	187	22	0

parameter sharing

- force sets of parameters to be equal
- reduces the number of parameters and therefore reduces overfitting
- typical example: Convolutional Neural Networks (CNN)

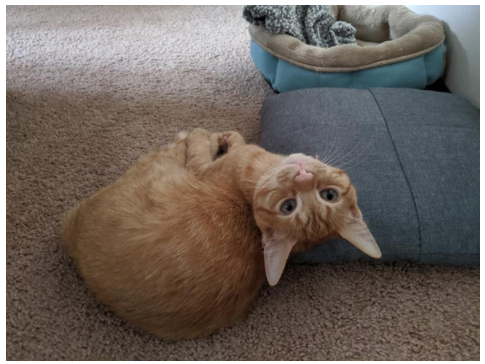
reduce overfitting

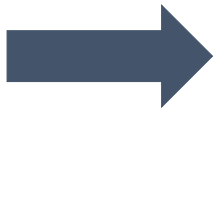


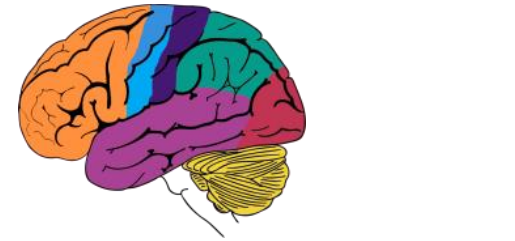
4.2 convolutional neural networks (CNN)

using NN for image data

using NN for image data







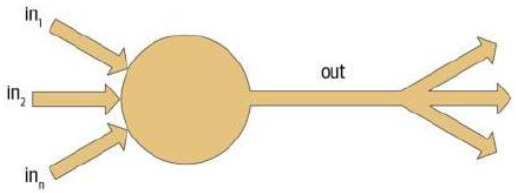
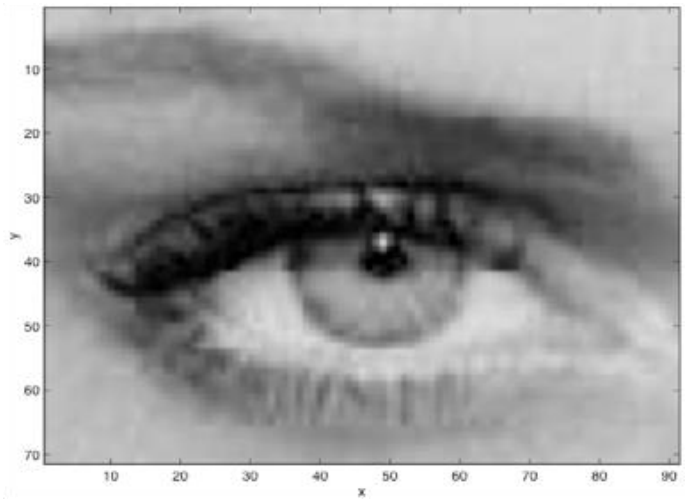


image as 2D function

- instead of flatten an image into a vector, we can treat an image as a function over a two-dimensional space $f(\boldsymbol{x}, \boldsymbol{y})$

- that is, for each pixel position $(\boldsymbol{x}, \boldsymbol{y})$, the function value is $f(\boldsymbol{x}, \boldsymbol{y})$



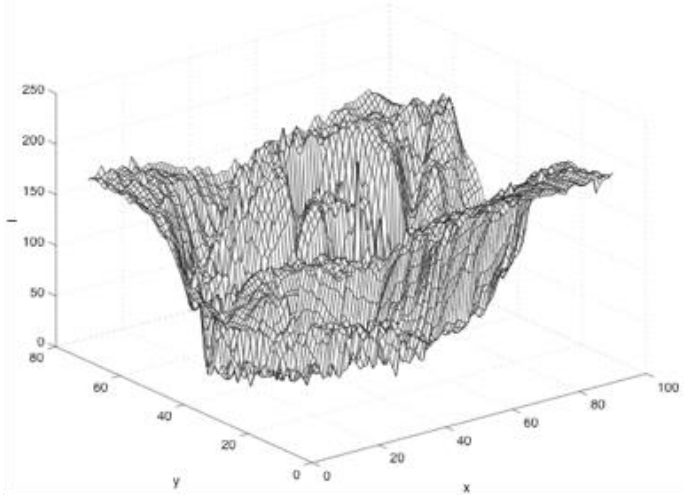


image as 2D function

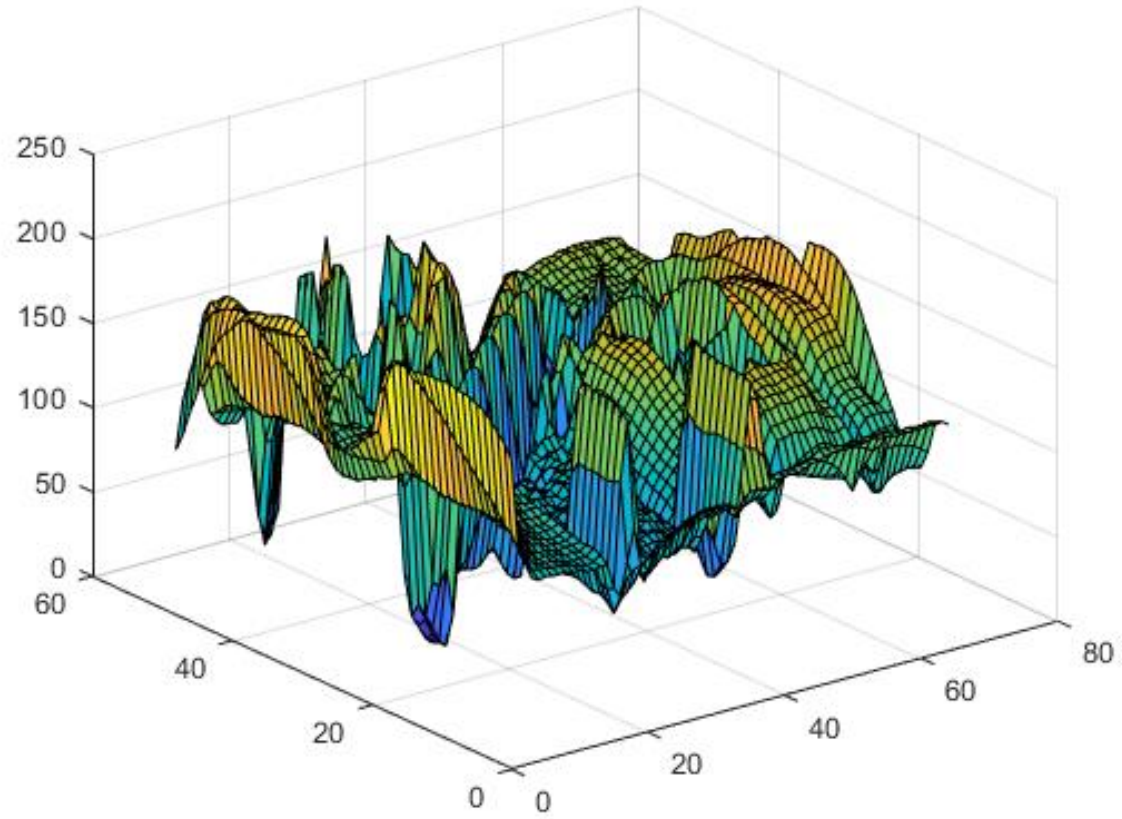
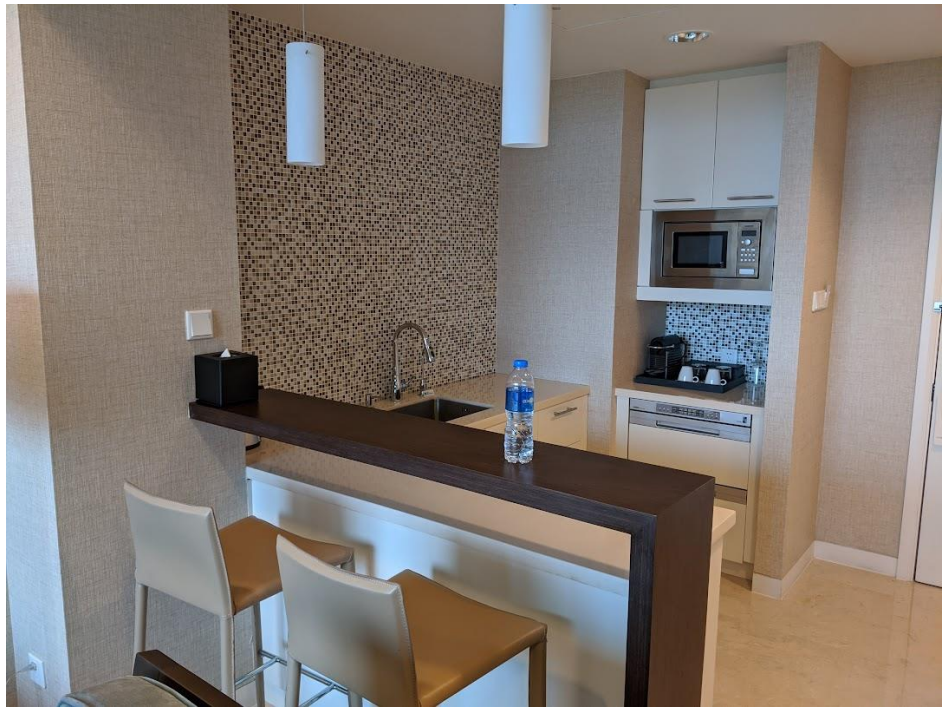


image as 2D function

RGB channels



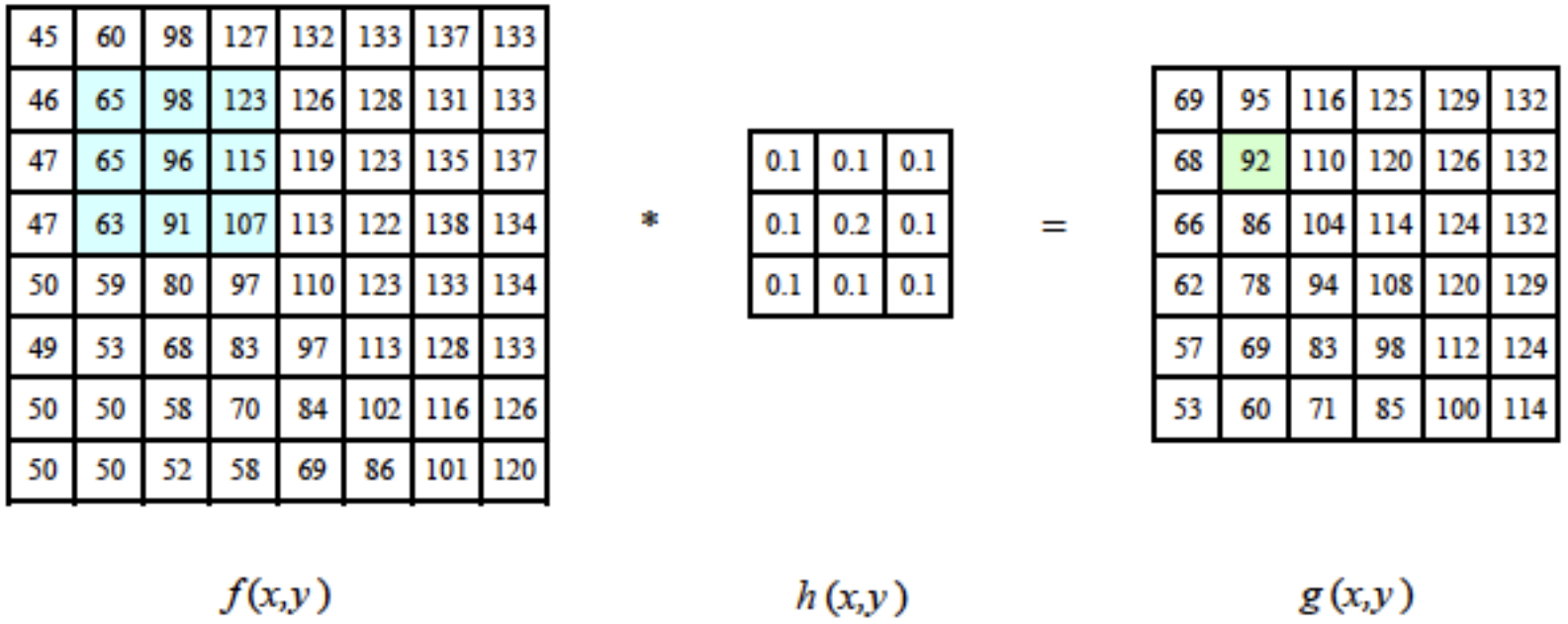
R

G

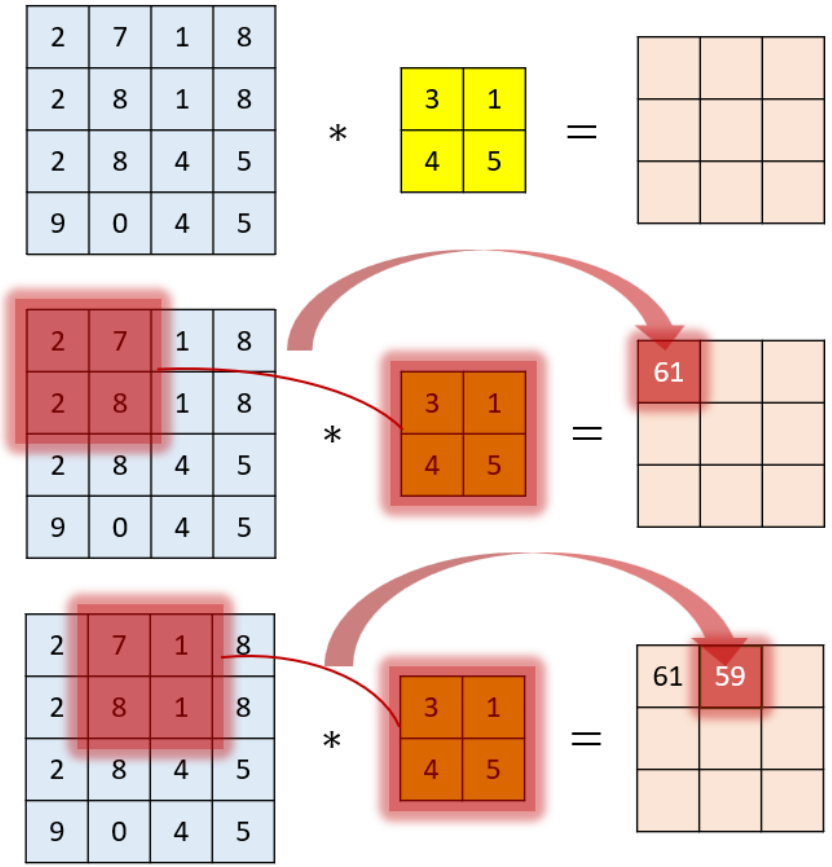
B

convolution / neighborhood filtering

- in convolution (a.k.a. neighborhood filtering), a two-dimensional function $h(\boldsymbol{x}, \boldsymbol{y})$, called a kernel, acts on the input image $f(\boldsymbol{x}, \boldsymbol{y})$, to produce an output $g(\boldsymbol{x}, \boldsymbol{y})$



convolution / neighborhood filtering



convolution / neighborhood filtering

original

blurring

sharpening

smoothing

convolution / neighborhood filtering

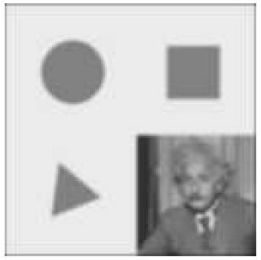
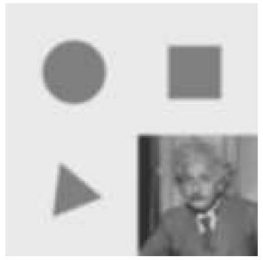
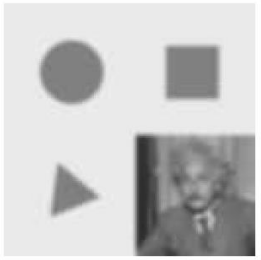
$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

convolution formulae

$$f * h(t) = \int f(t-u)h(u)\,du = \int f(u)h(t-u)\,du$$

input kernel

$$S(i,j) = (I*K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

$$S(i,j) = \sum_m \sum_n I(i+m-1, j+n-1)K(m,n)$$ (cross-correlation form)

convolution as matrix multiplication

$$\boldsymbol{x} * \boldsymbol{h} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} h_m & h_{m-1} & \cdots & h_1 & 0 & 0 & \cdots & 0 \\ 0 & h_m & h_{m-1} & \cdots & h_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & & & \ddots & \vdots \\ 0 & \cdots & 0 & h_m & h_{m-1} & \cdots & h_1 & 0 \\ 0 & \cdots & 0 & 0 & h_m & h_{m-1} & \cdots & h_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

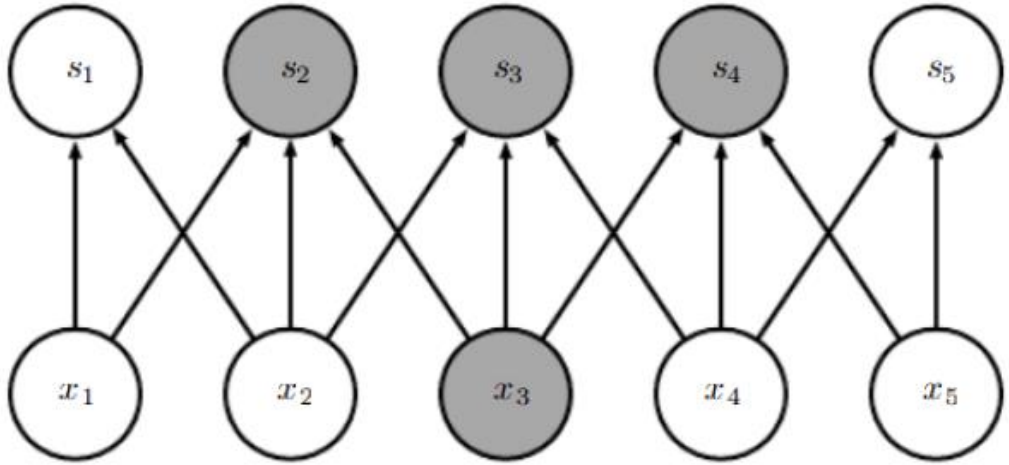
CNN

- idea: replacing linear/dense/fully-connected layers with convolution

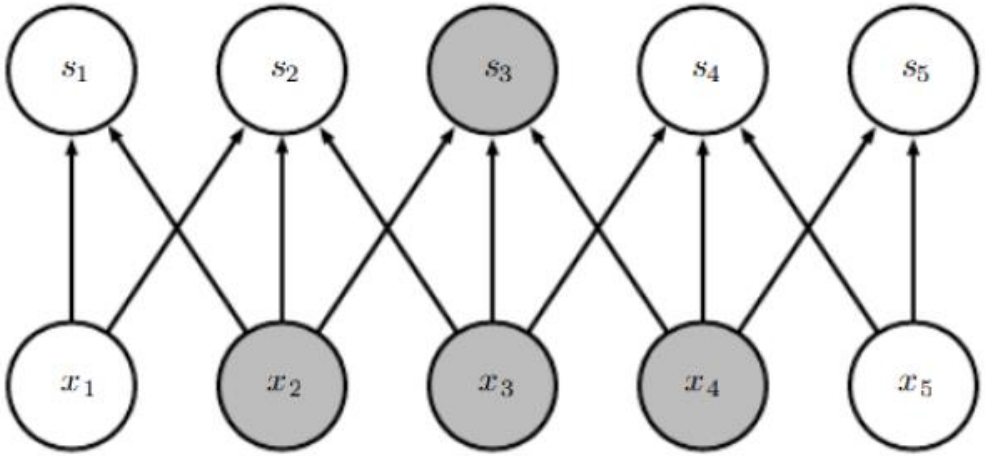
motivation of CNN

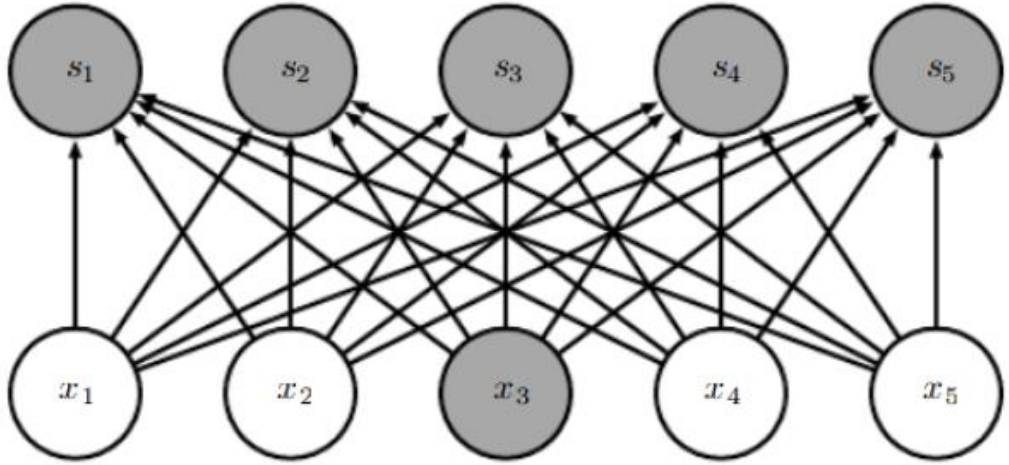
- sparse interactions
- parameter sharing
- equivariant representations

sparse connection

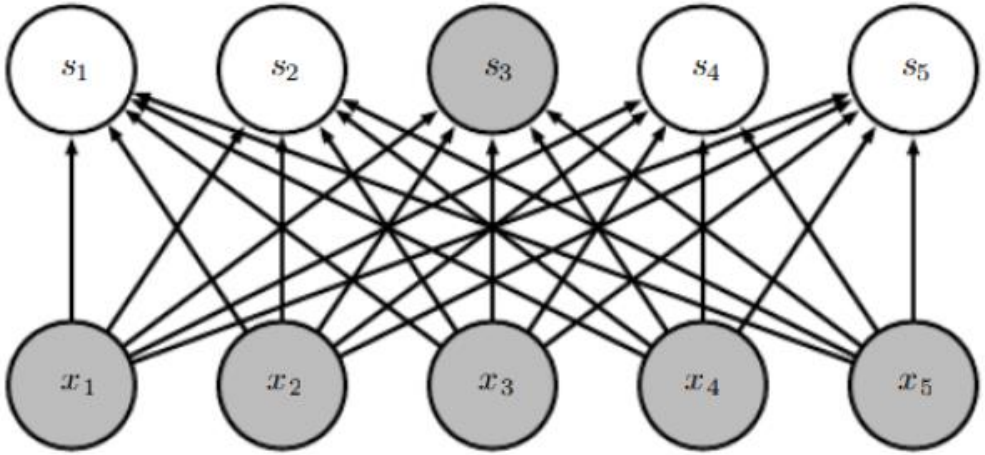


CNN

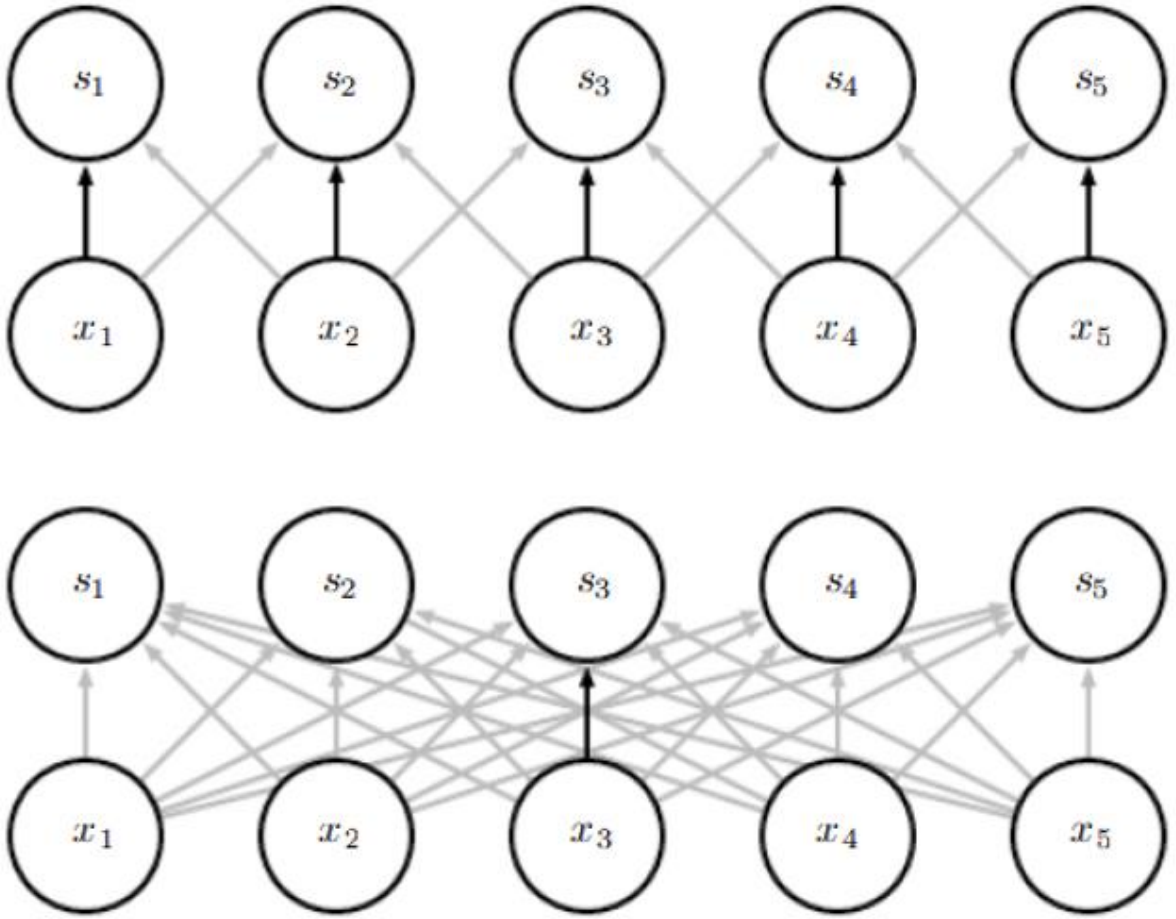




MLP



parameter sharing



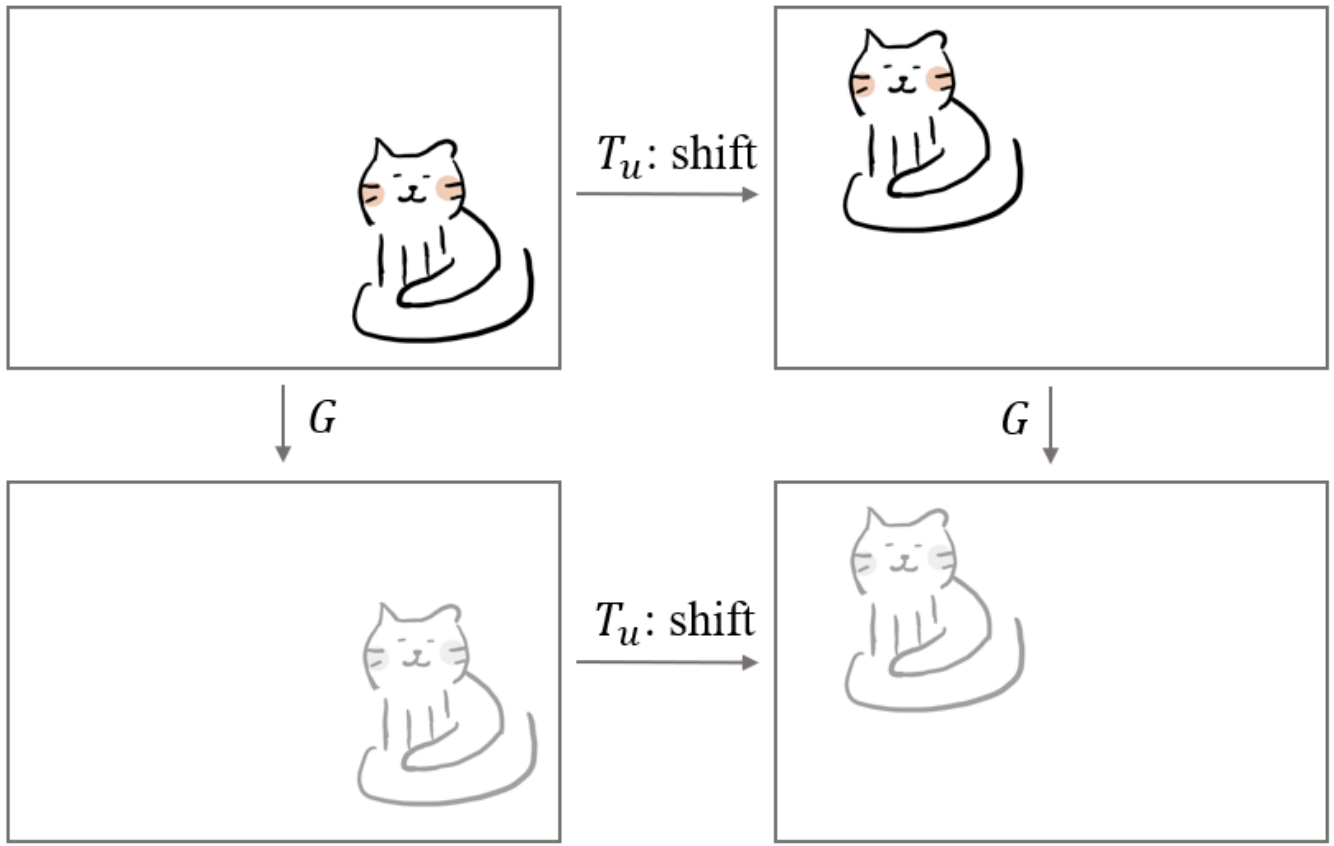
equivariance

- consider a translated/shifted version of the input:

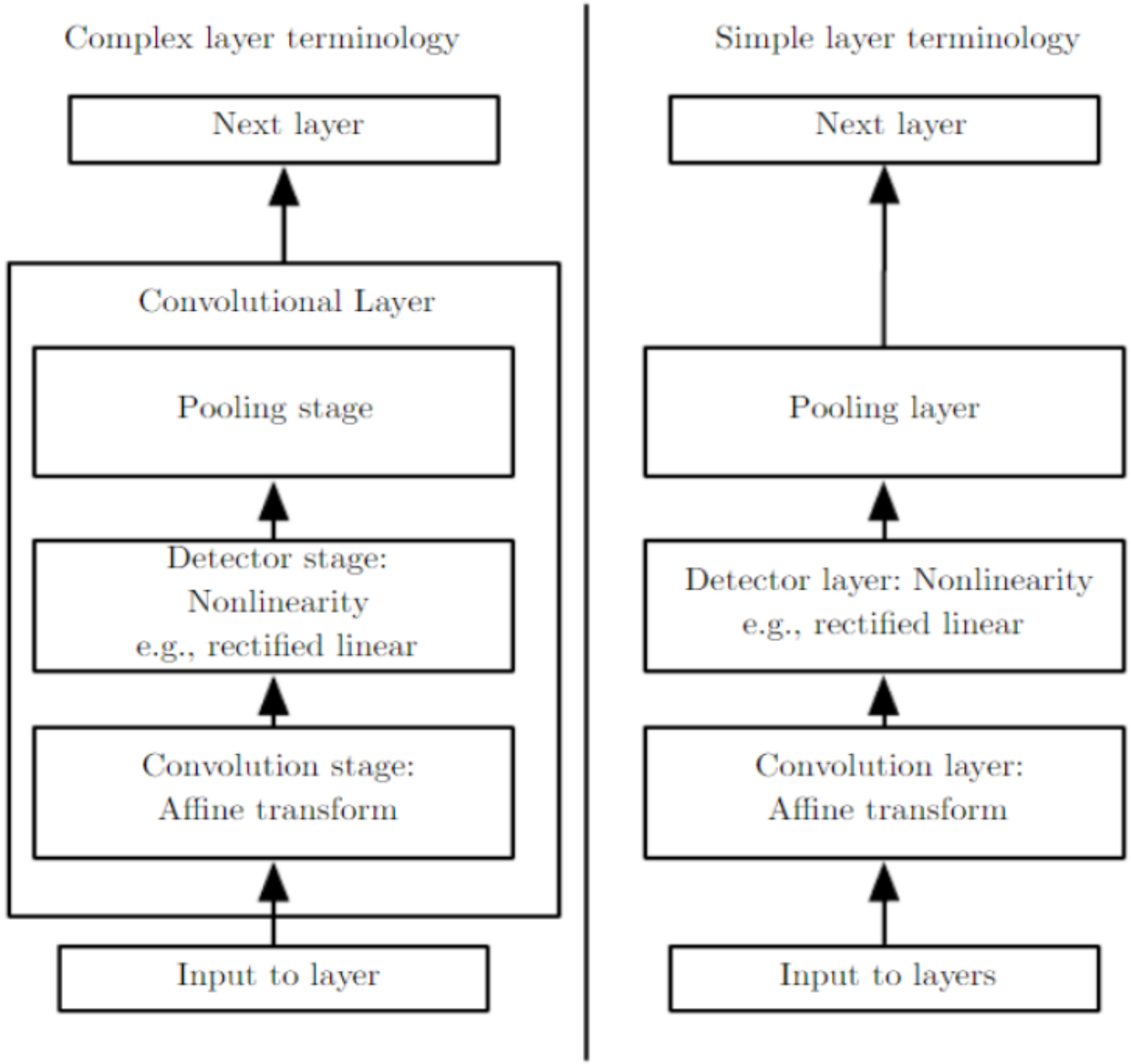
$$f_\tau(t) = f(t - \tau)$$

- what is $f_\tau * g$?

equivariance



structure


Complex layer terminology
Next layer
Convolutional Layer
Pooling stage
Detector stage:
Nonlinearity
e.g., rectified linear
Convolution stage:
Affine transform
Input to layer
Simple layer terminology
Next layer
Pooling layer
Detector layer: Nonlinearity
e.g., rectified linear
Convolution layer:
Affine transform
Input to layers

pooling

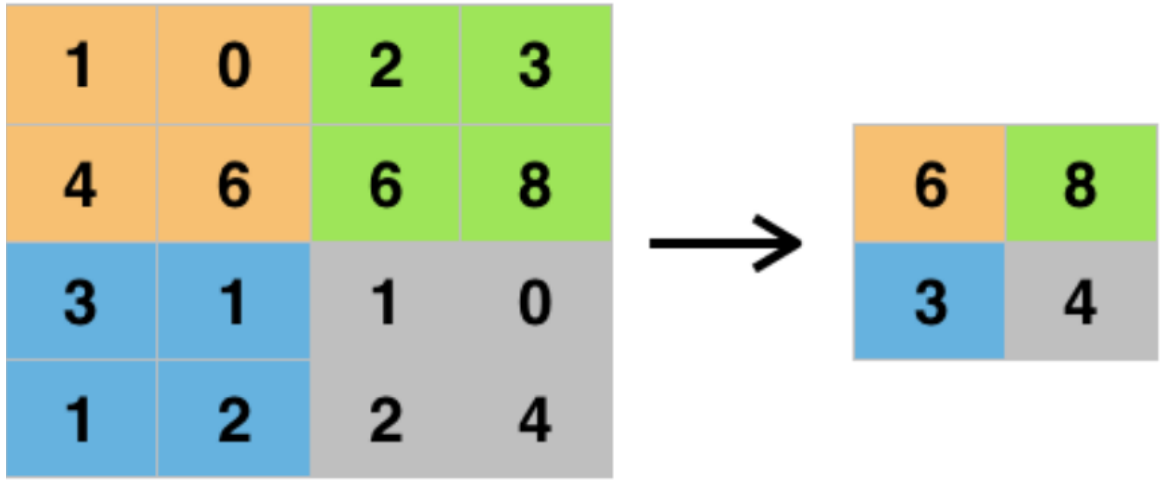


image credit: deepai

padding

a 4x4 kernel with 5x5 input and 2 pixels of padding

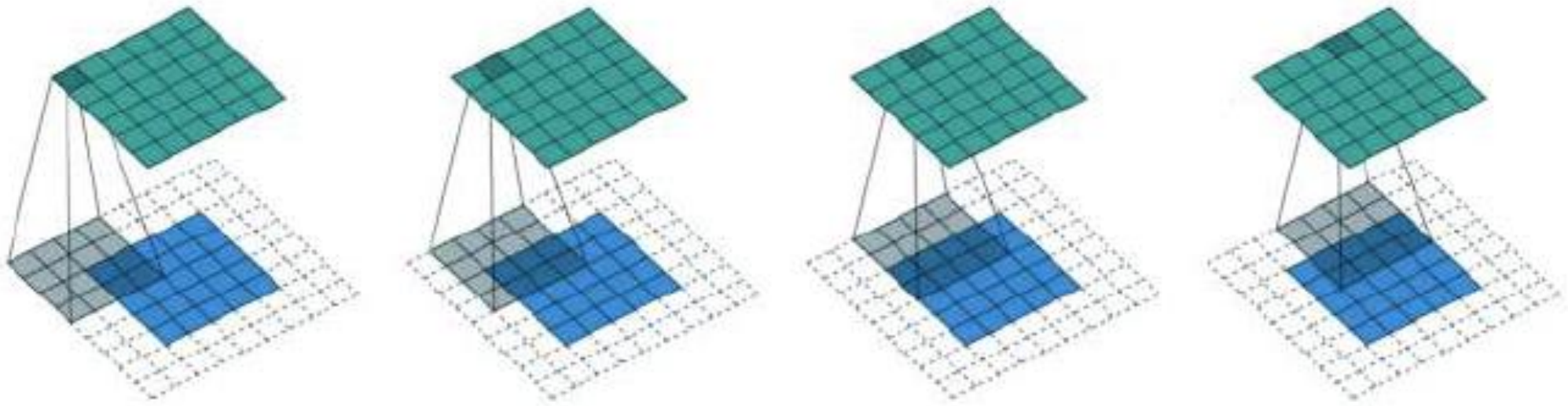


image credit: Deep Learning for Coders with fastai and PyTorch

stride

A 3x3 kernel with 5x5 input, stride-2 convolution, and 1 pixel of padding

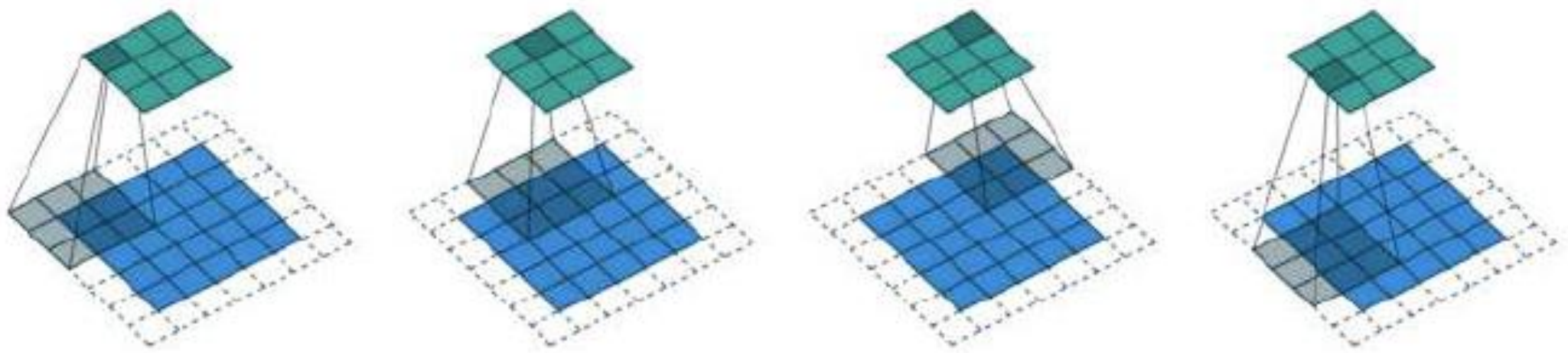


image credit: Deep Learning for Coders with fastai and PyTorch

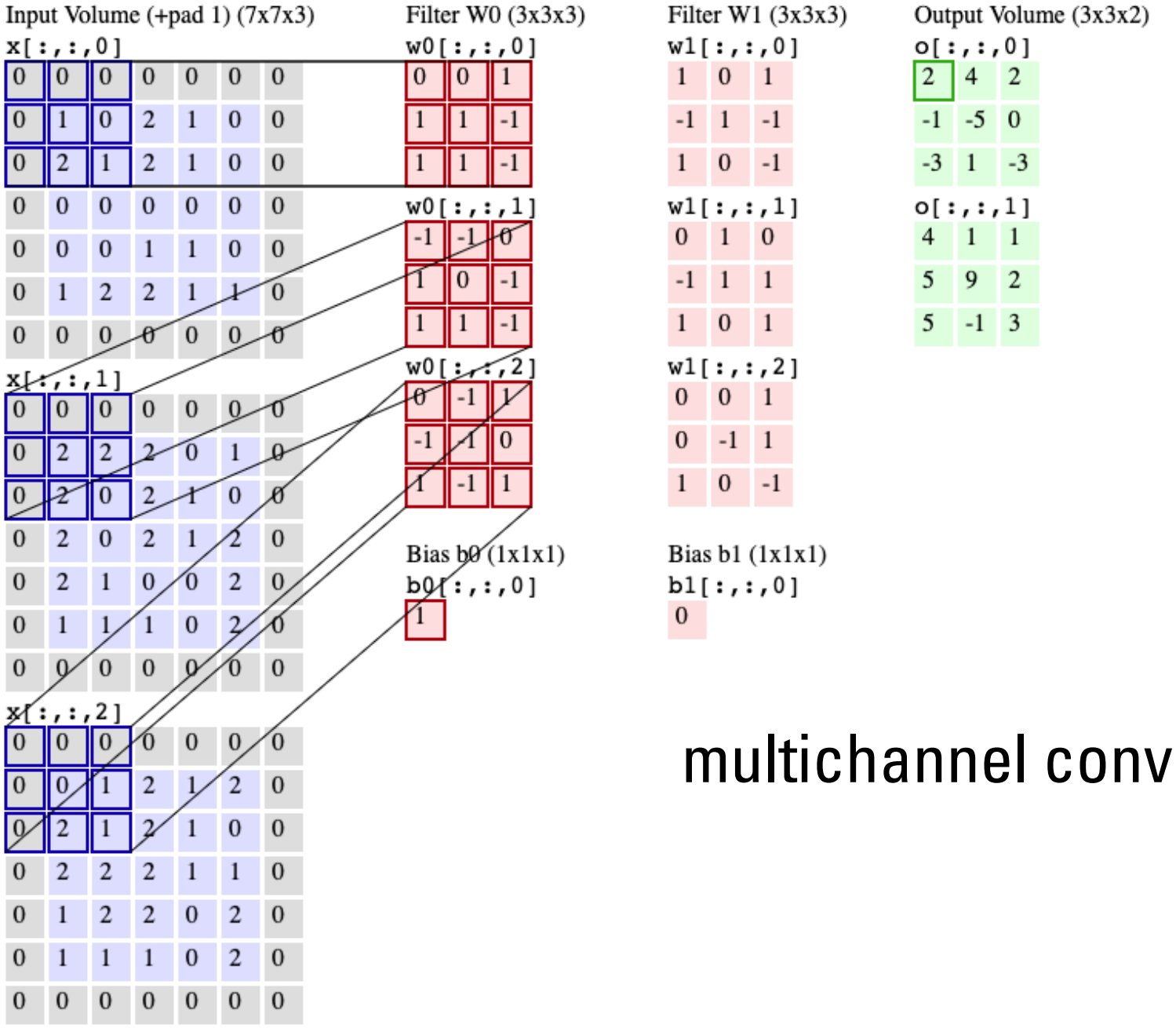
multichannel convolution

- in general, the input and output have multiple channels (e.g., the original input image may have three RGB channels)

- 4D kernel: $\boldsymbol{K}(i, j, k, l)$
  - i : index for output channels
  - j : index for input channels

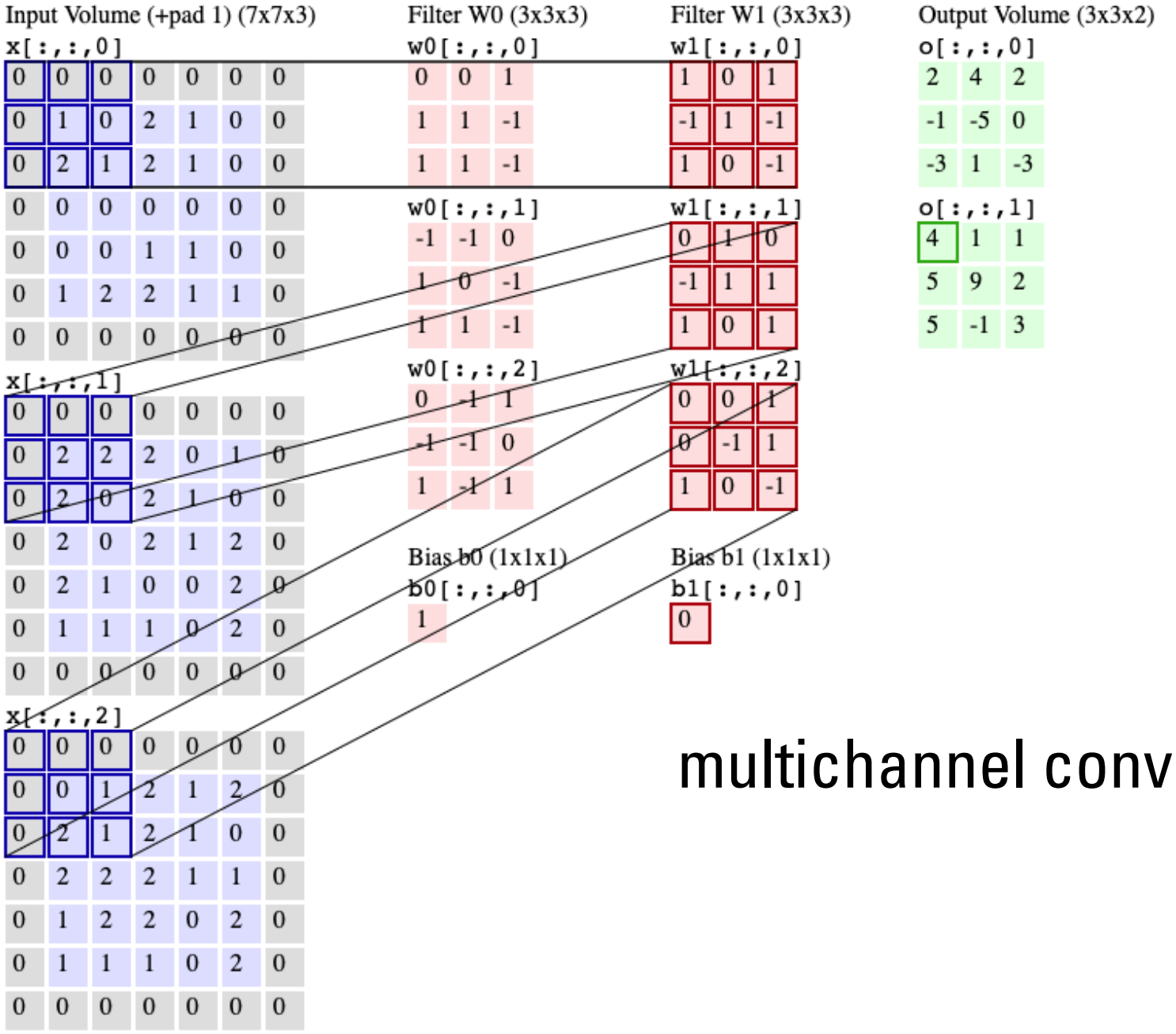
- suppose an input is $\boldsymbol{V}(j, k, l)$, then the output is:

$$\boldsymbol{Z}(i, k, l) = \sum_{j,m,n} \boldsymbol{V}(j, k + m - 1, l + n - 1)\, \boldsymbol{K}(i, j, m, n)$$



multichannel convolution

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



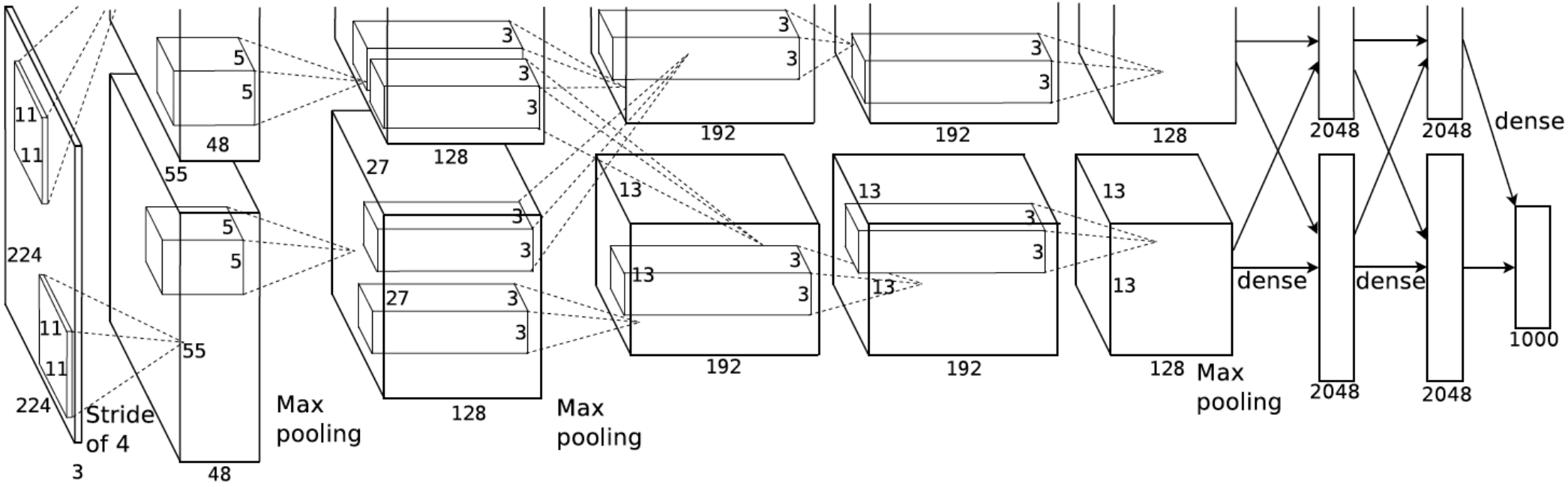
multichannel convolution

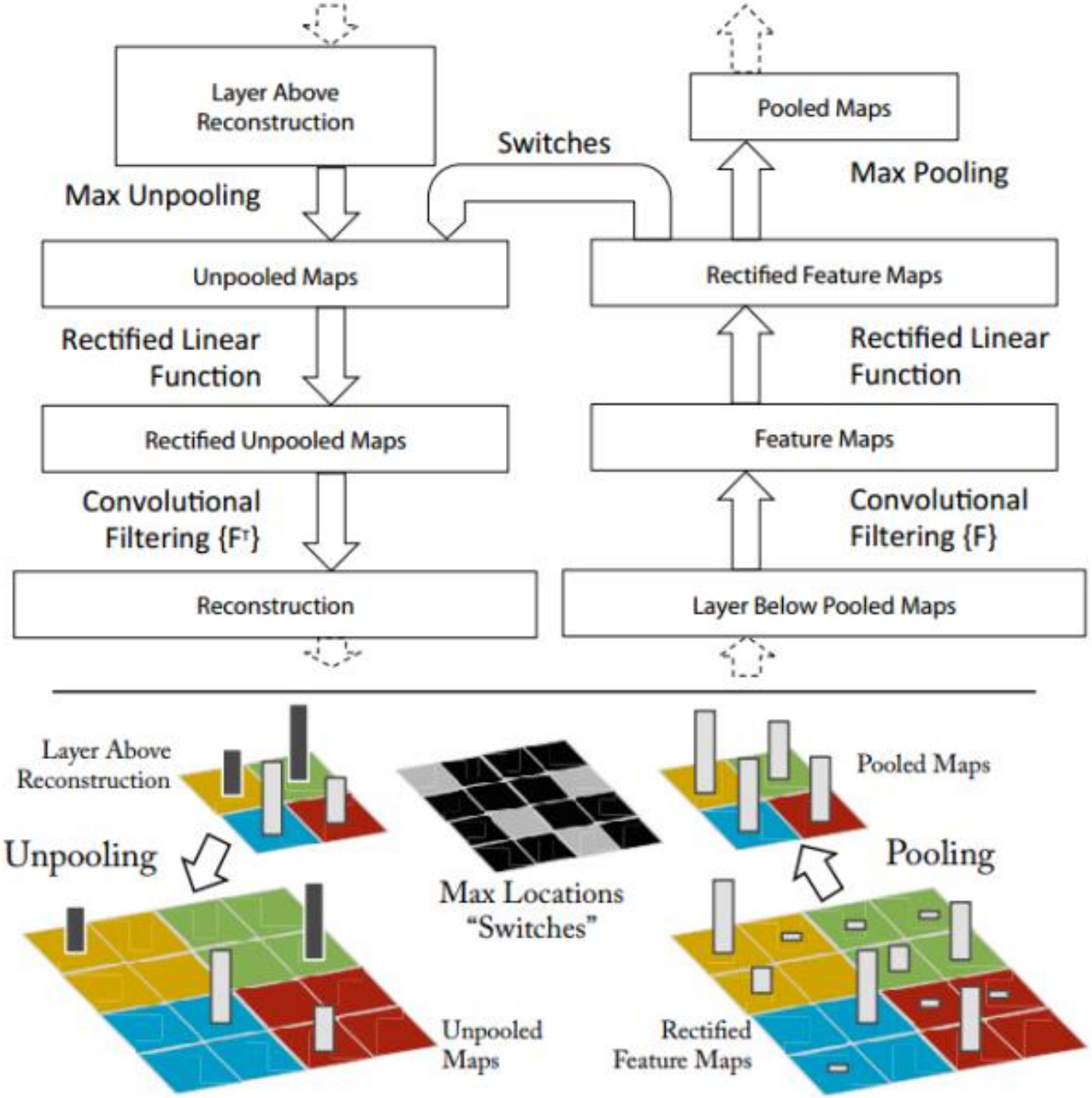
https://cs231n.github.io/convolutional-networks/

example

Layer (type)	Output Shape	Param #	Trainable
Conv2d	64 x 4 x 14 x 14	40	True
ReLU	64 x 4 x 14 x 14	0	False
Conv2d	64 x 8 x 7 x 7	296	True
ReLU	64 x 8 x 7 x 7	0	False
Conv2d	64 x 16 x 4 x 4	1,168	True
ReLU	64 x 16 x 4 x 4	0	False
Conv2d	64 x 32 x 2 x 2	4,640	True
ReLU	64 x 32 x 2 x 2	0	False
Conv2d	64 x 2 x 1 x 1	578	True
Flatten	64 x 2	0	False

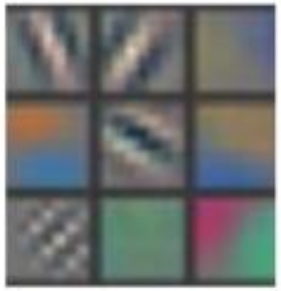
AlexNet


11
11
224
224
3
Stride
of 4
5
5
48
55
48
55
5
5
3
3
3
128
27
128
27
3
3
3
3
3
3
192
192
128
13
13
13
13
13
13
3
3
3
3
3
3
192
192
128
Max
pooling
Max
pooling
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000



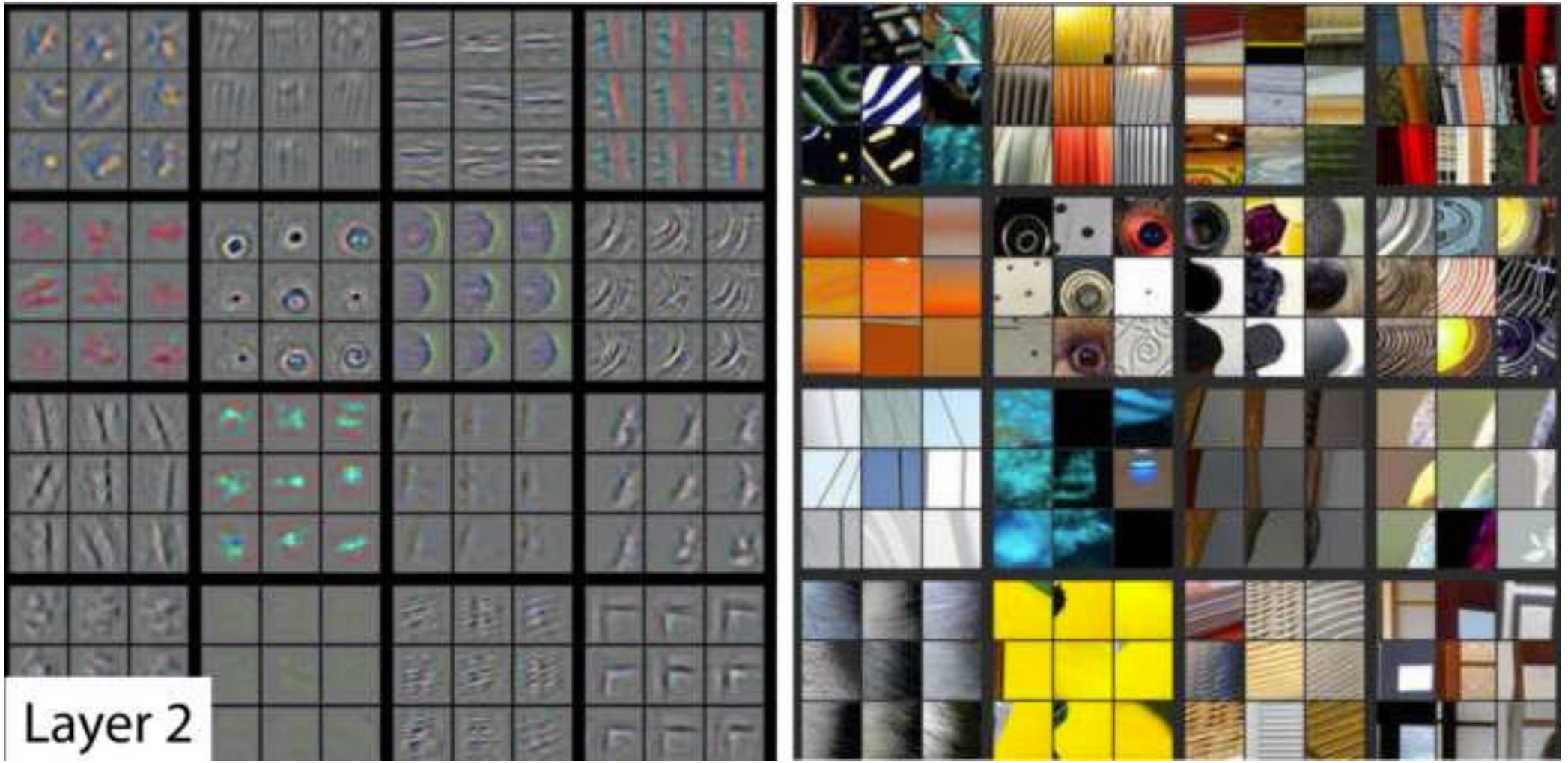
what do convolution layers learn

Visualizing and Understanding Convolutional Networks (ECCV 2014)

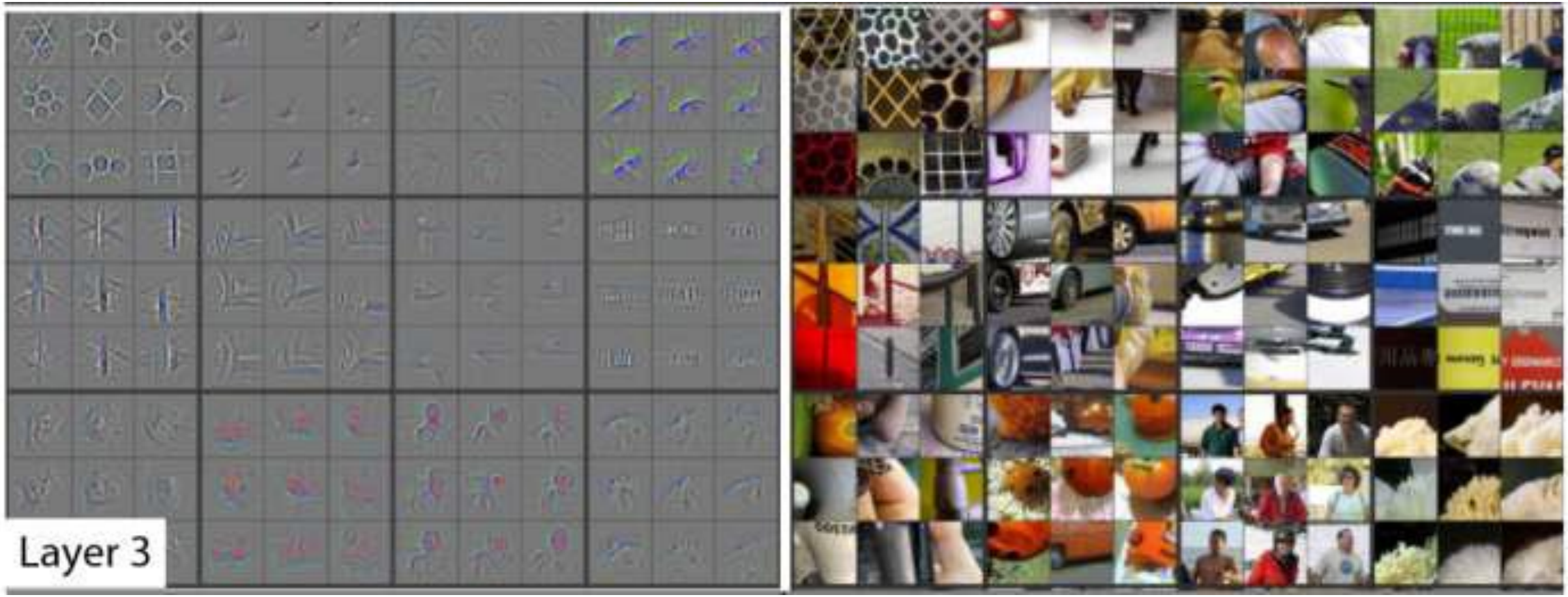


Layer 1

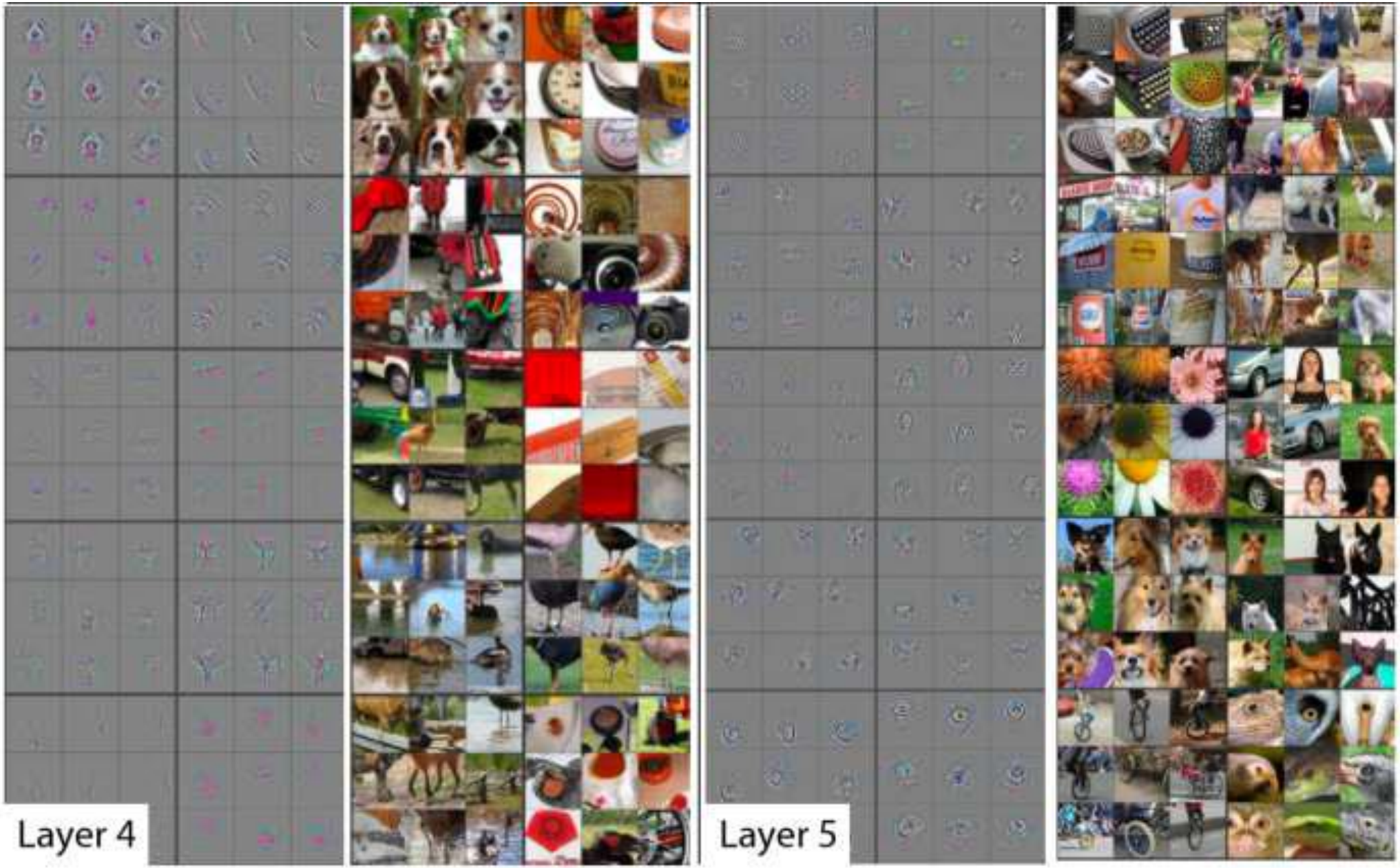
in layer 1, what we can see is that the model has discovered weights that represent diagonal, horizontal, and vertical edges, as well as various gradients



in layer 2, we can see that the model has learned to create feature detectors that look for corners, repeating lines, circles, and other simple patterns



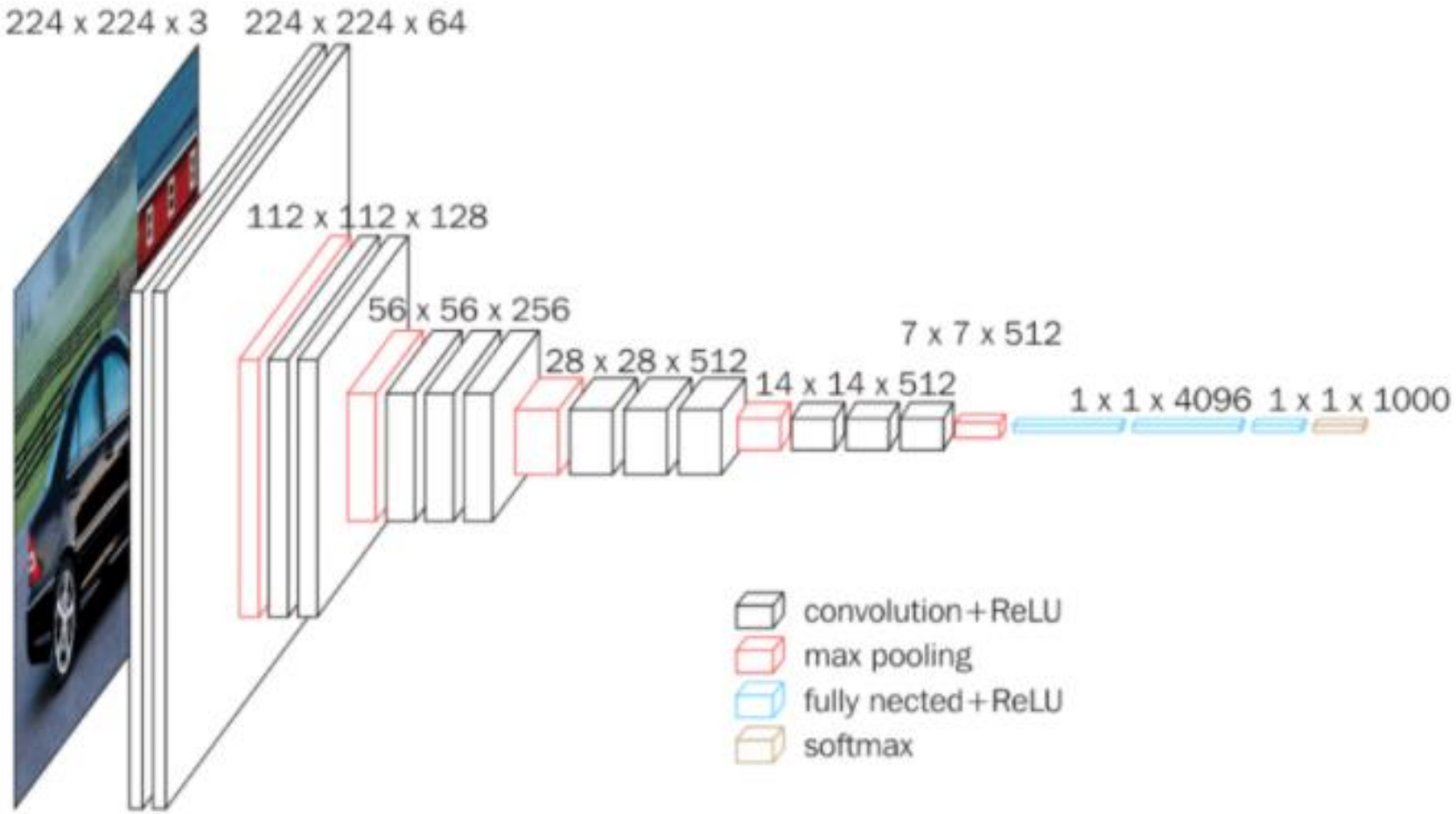
as it goes deeper, more higher-level concepts


Layer 4
Layer 5

datasets

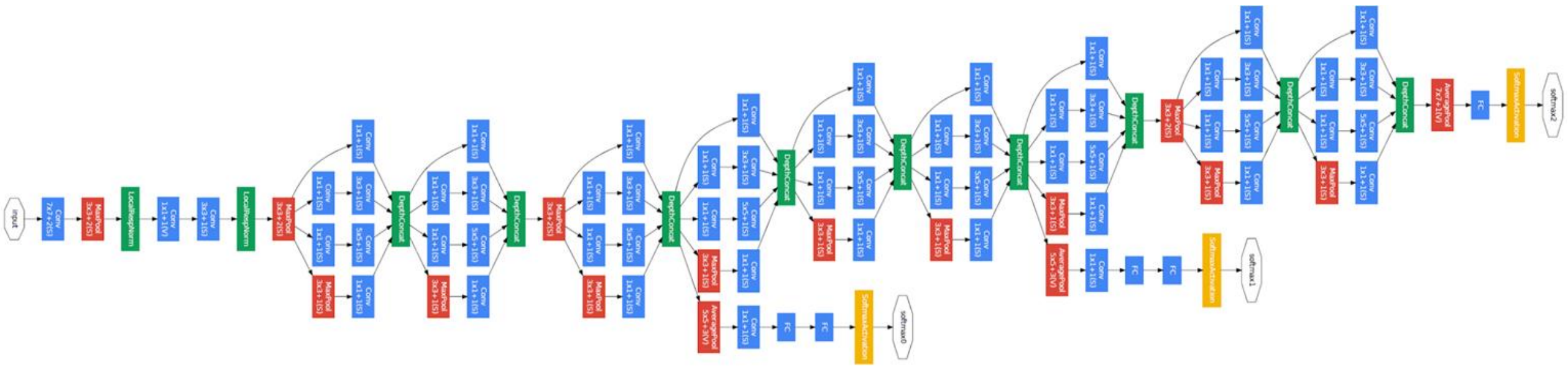

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

VGG16



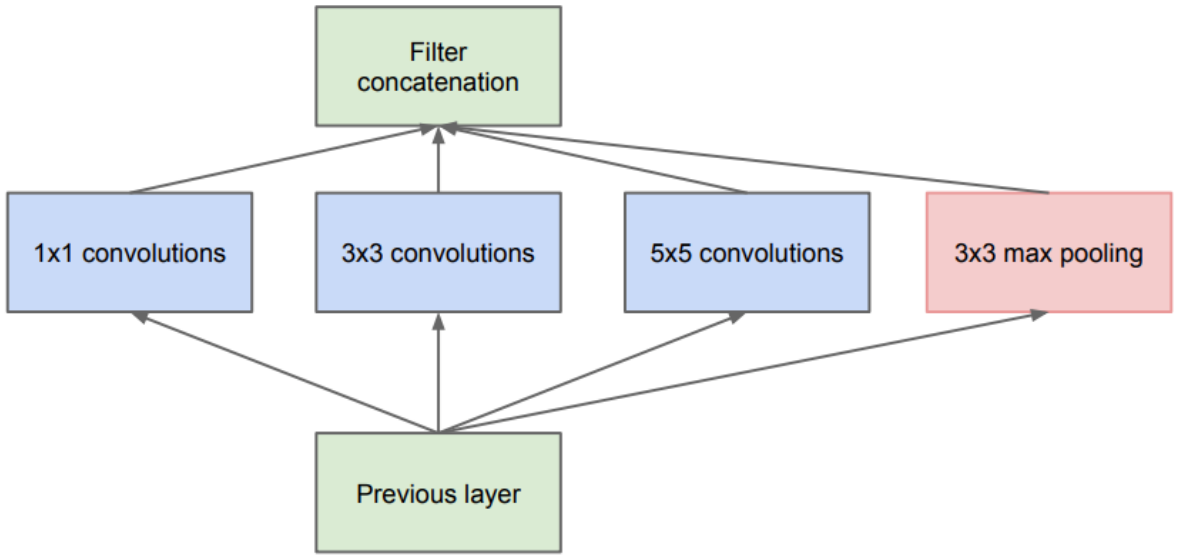
Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *ICLR 2015.*

GoogleNet

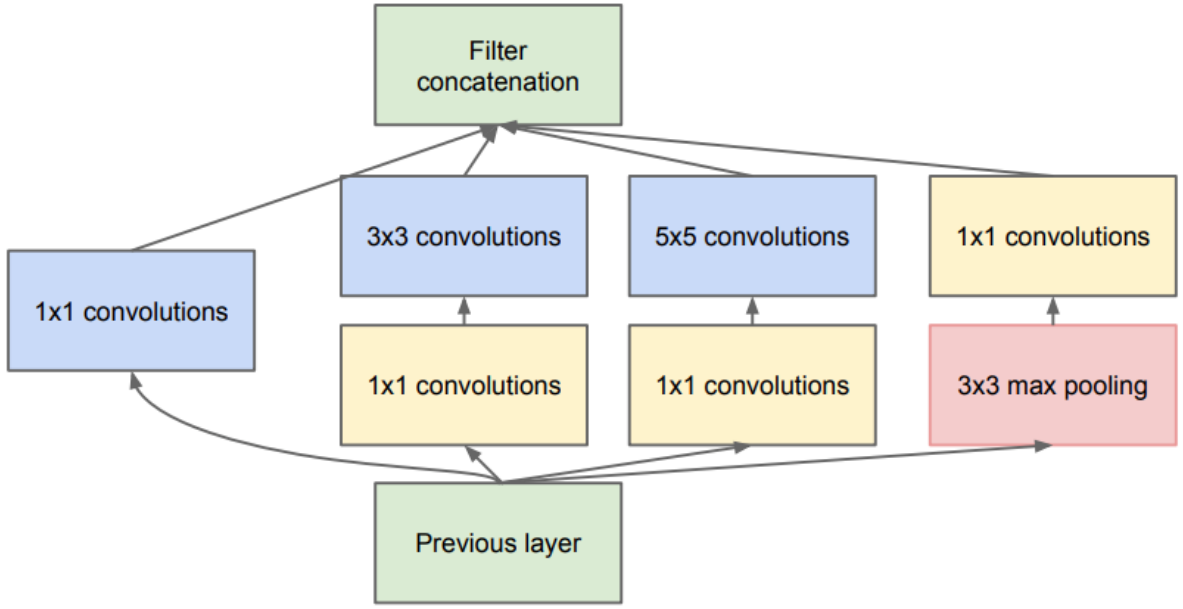


Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).

GoogleNet: inception modules

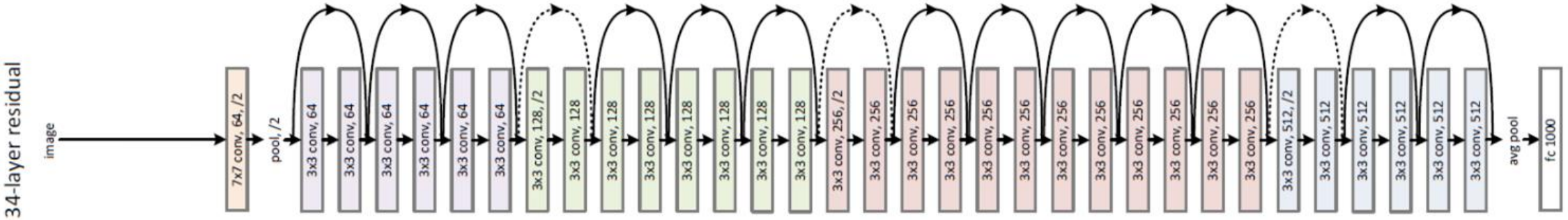


(a) Inception module, naïve version



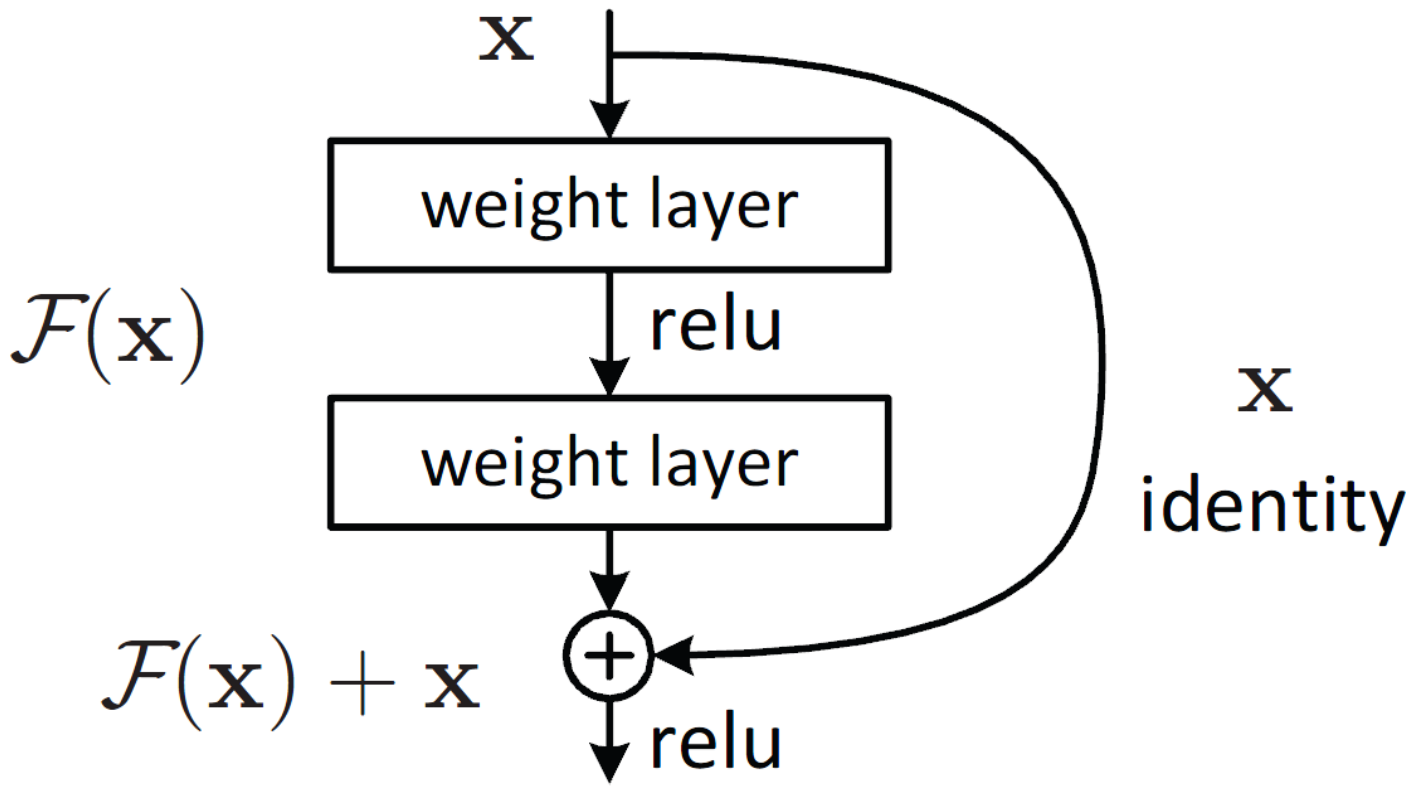
(b) Inception module with dimensionality reduction

ResNet



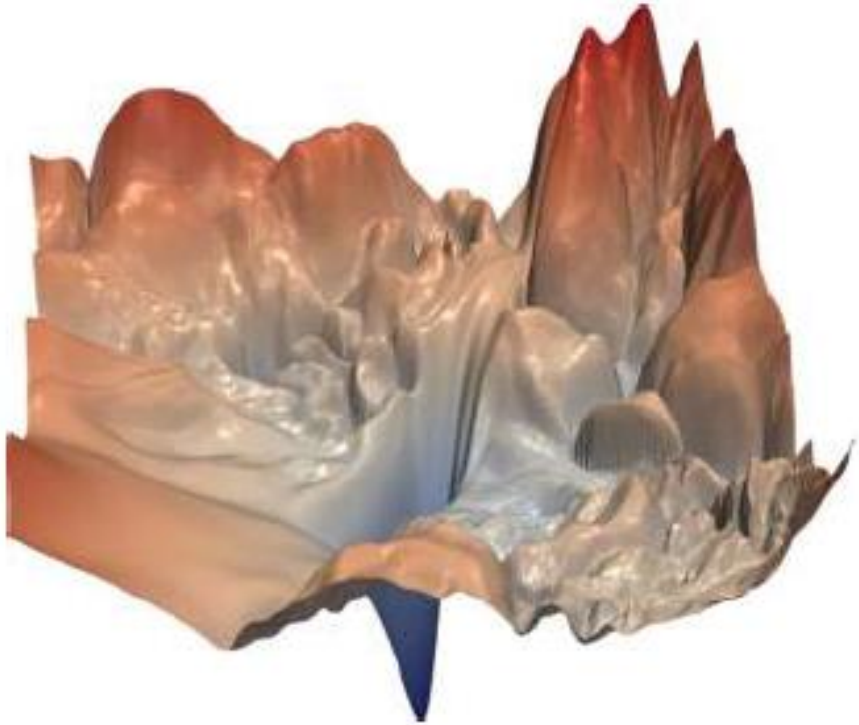
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

ResNet

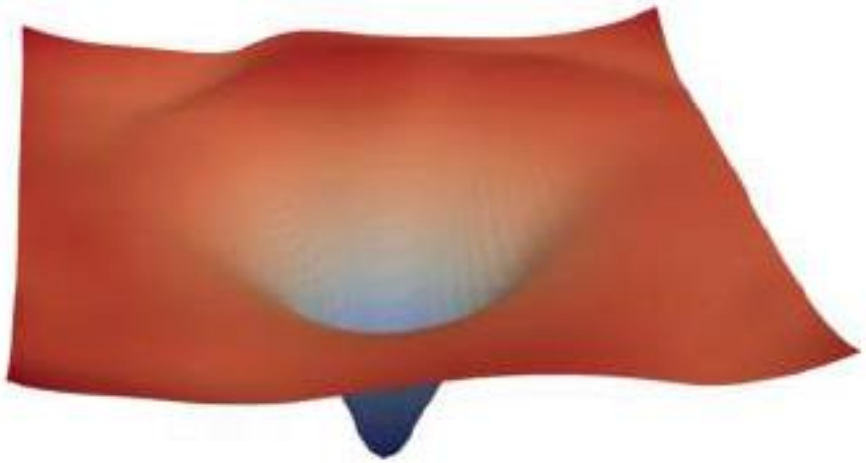

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

ResNet

- landscape of the loss functions



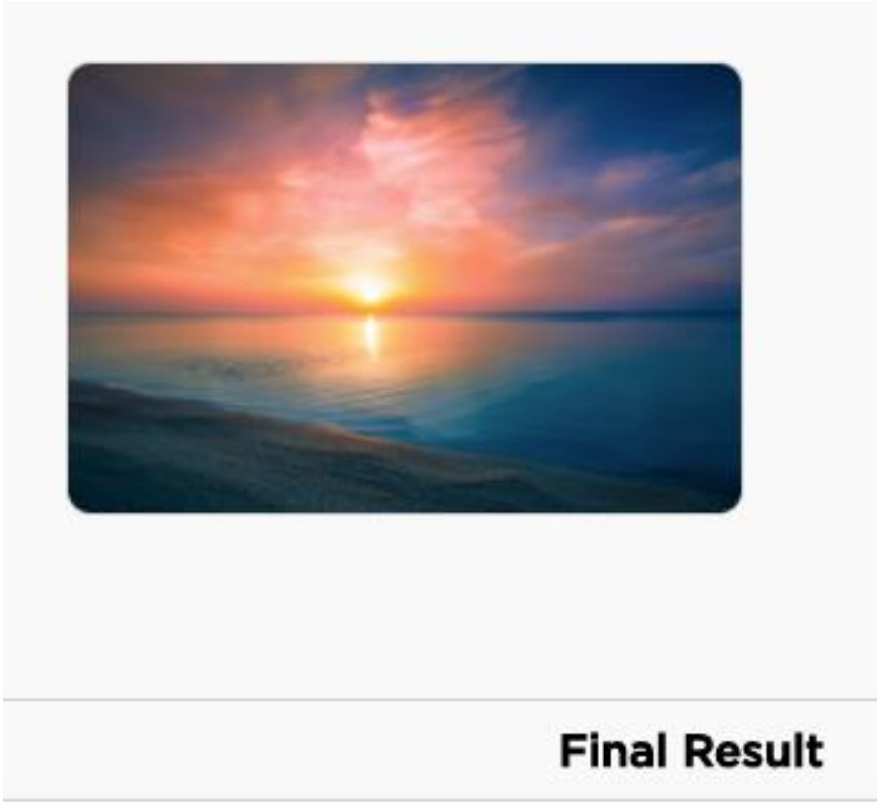
regular CNN



ResNet

4.3 CNN in classification

scene recognition



application: smart album, auxiliary photography, …

scene: landscape	confidence: 99.81%
scene: sky	confidence: 83.85%
scene: sunrise	confidence: 99.83%
scene: lake	confidence: 64.05%
scene: sea	confidence: 73.72%

Places365 dataset

https://github.com/CSAILVision/places365

photo evaluation

Final Result

images quality: 4.593186331097968

Final Result

images quality: 5.335587732726708

AVA (Aesthetic Visual Analysis) dataset

TITLE: Skyscape

Description:

Make the sky the subject of your photo this week.

Stats

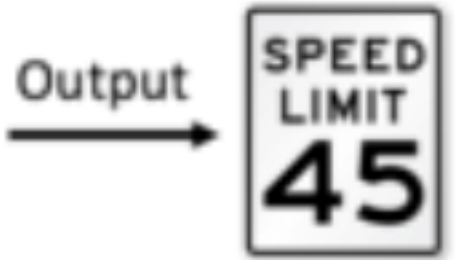
Voting Dates:
13/07/2010 - 19/07/2010

Numbers & Statistics:
Submissions: 136
Disqualifications: 1
Votes: 16,009
Comments: 595

Average Score: 5.64014

robustness

- view angles
- sizes
- distortion
- lighting
- cover
- …



robustness

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

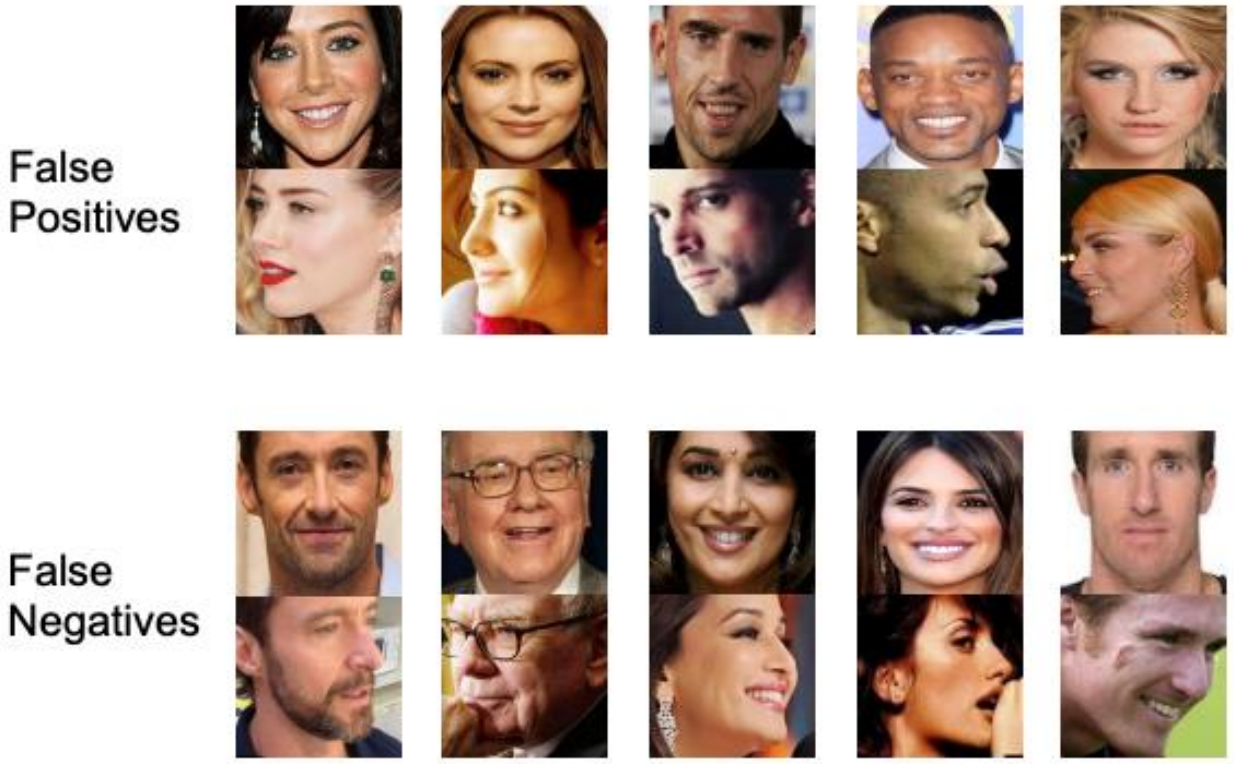
"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

face recognition



frontal faces

profile faces

image credit: Cao et al. (2018), Pose-Robust Face Recognition via Deep Residual Equivariant Map

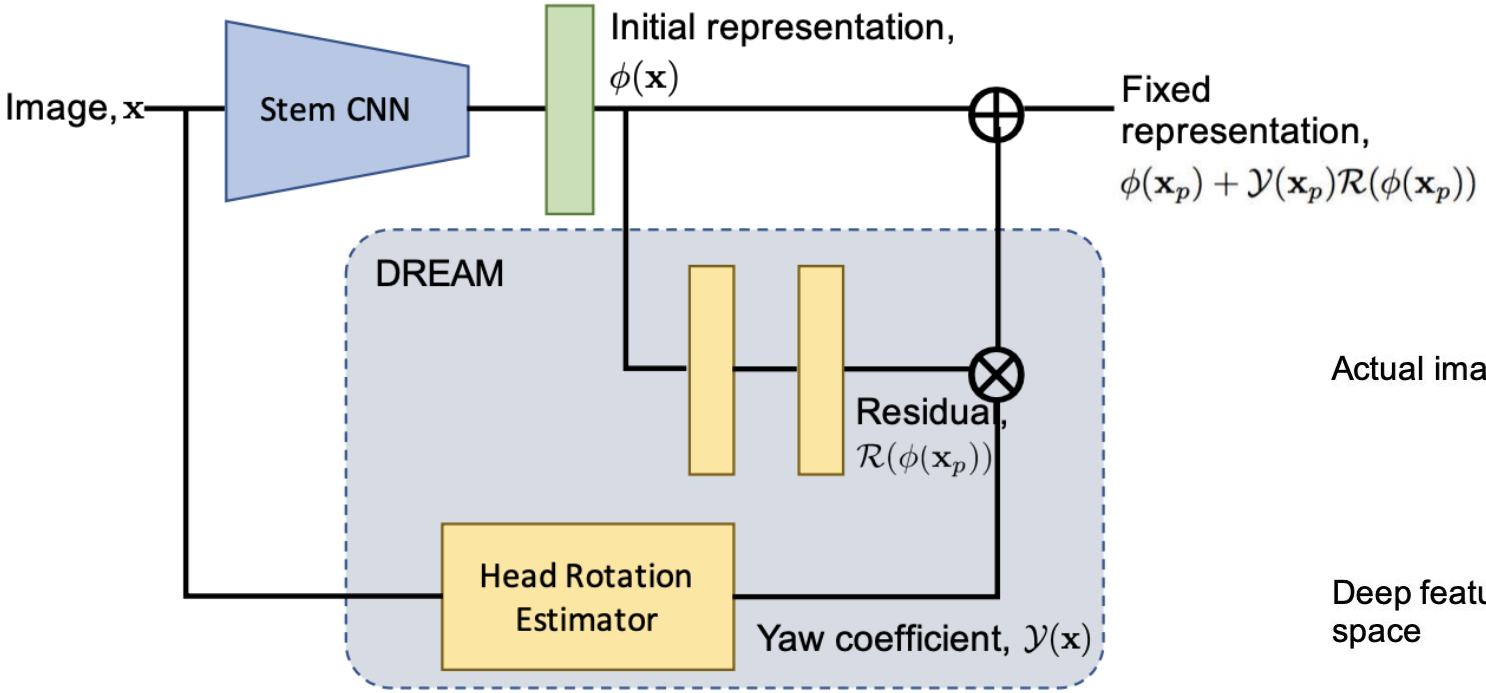
face recognition

why does face recognition work poorly on profile faces?

- the generalization power of deep models is usually proportional to the training data size

- given an uneven distribution of profile and frontal faces in the dataset, deeply learned features tend to bias on distinguishing frontal faces rather than profile faces

face recognition



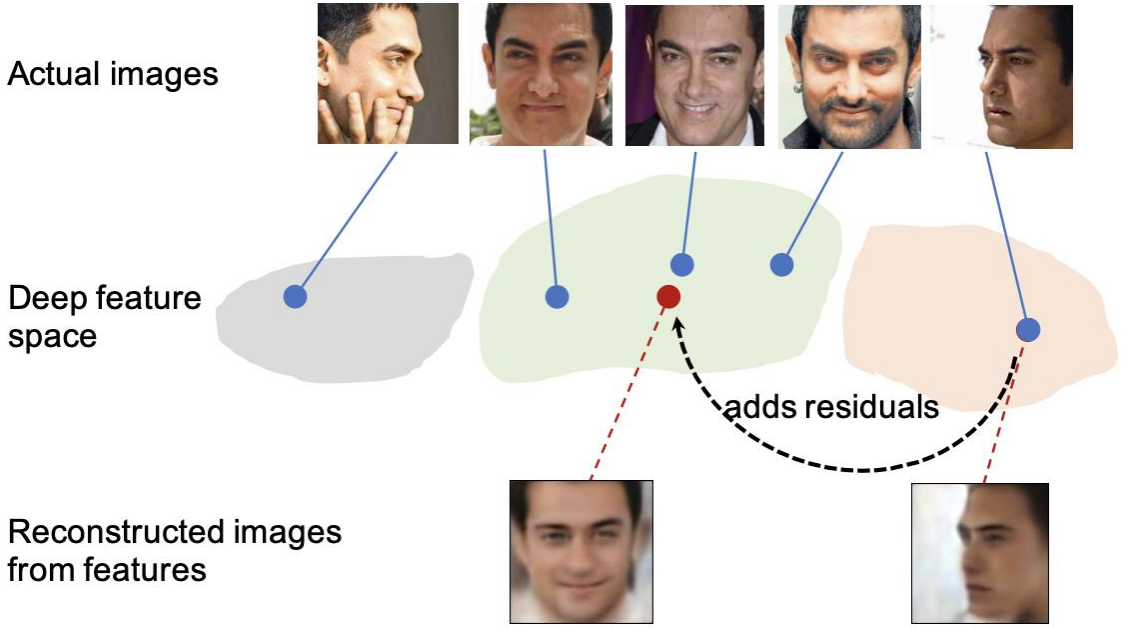


image credit: Cao et al. (2018), Pose-Robust Face Recognition via Deep Residual Equivariant Map

face recognition

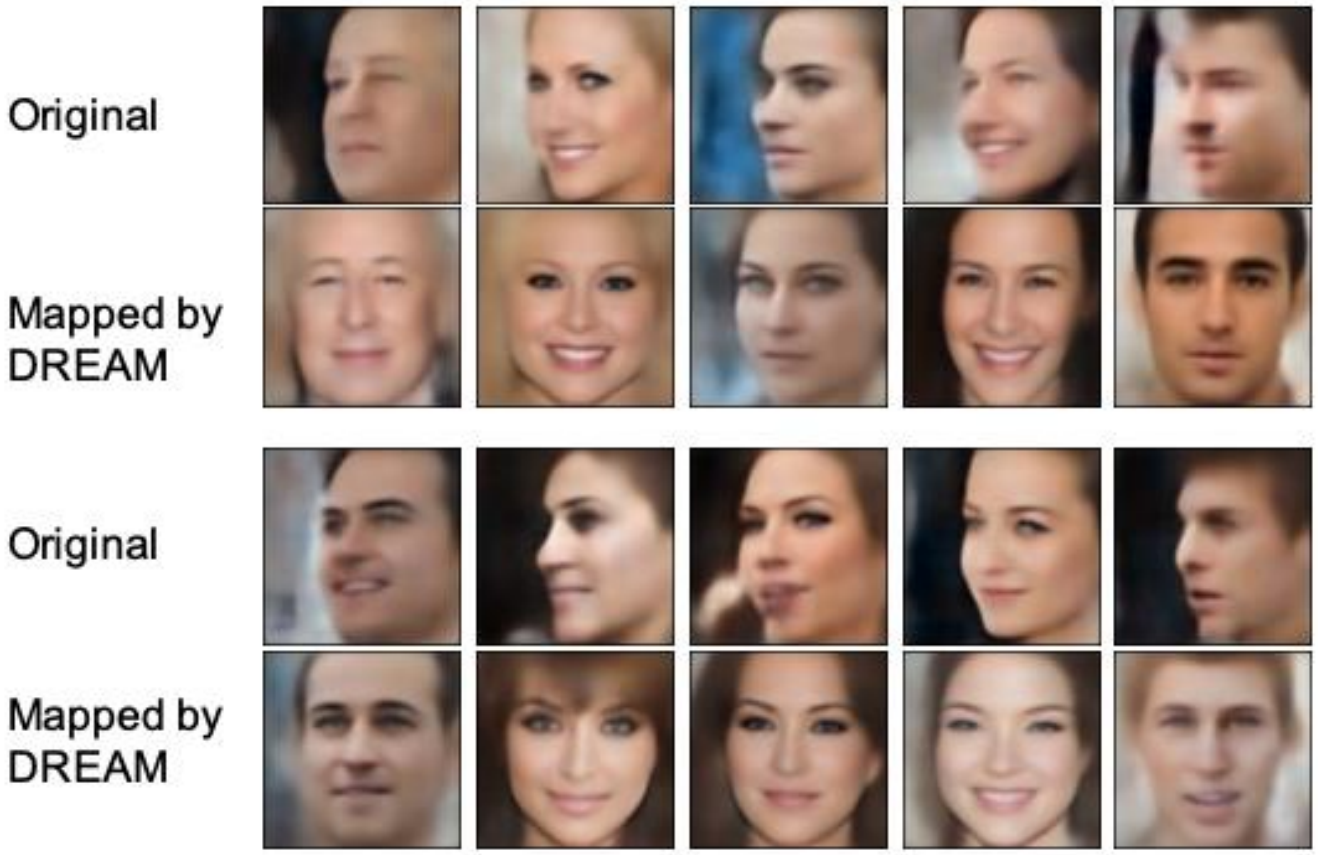
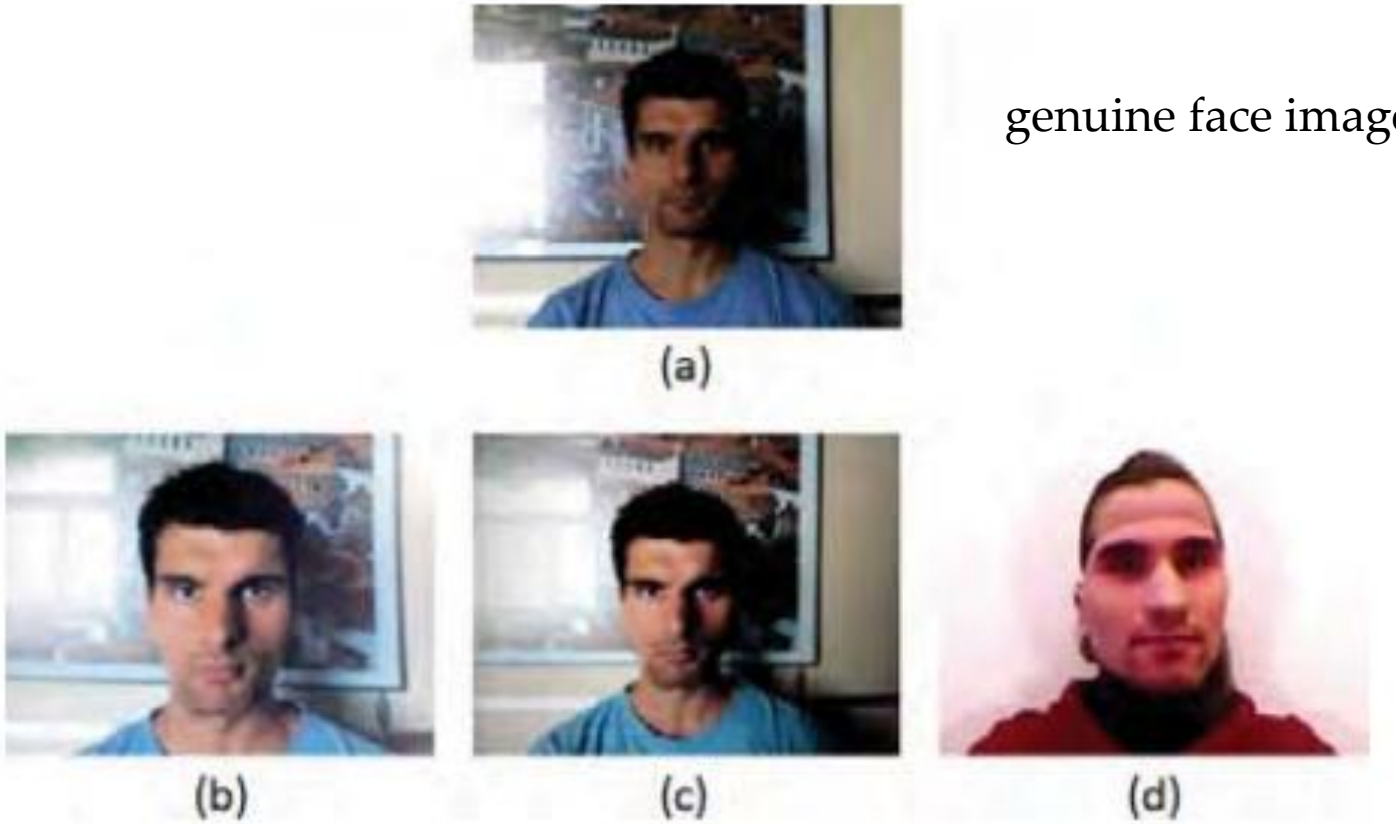


image credit: Cao et al. (2018), Pose-Robust Face Recognition via Deep Residual Equivariant Map

face spoof detection

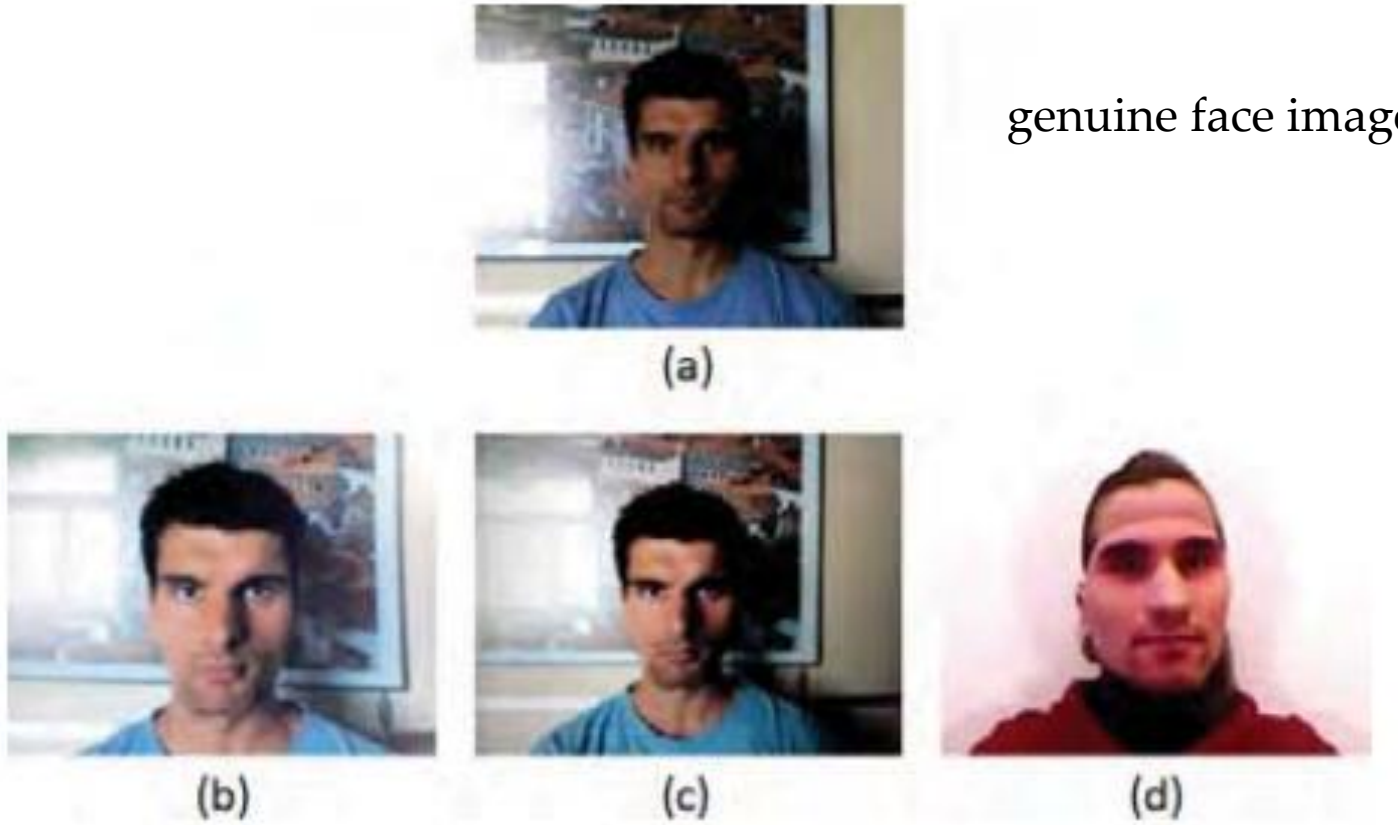
which of the three is also a genuine face image?



genuine face image

face spoof detection

which of the three is also a genuine face image?



genuine face image

print attack:
photo of a printed photo

replay attack:
photo of a tablet screen

a person wearing a mask

face spoof detection

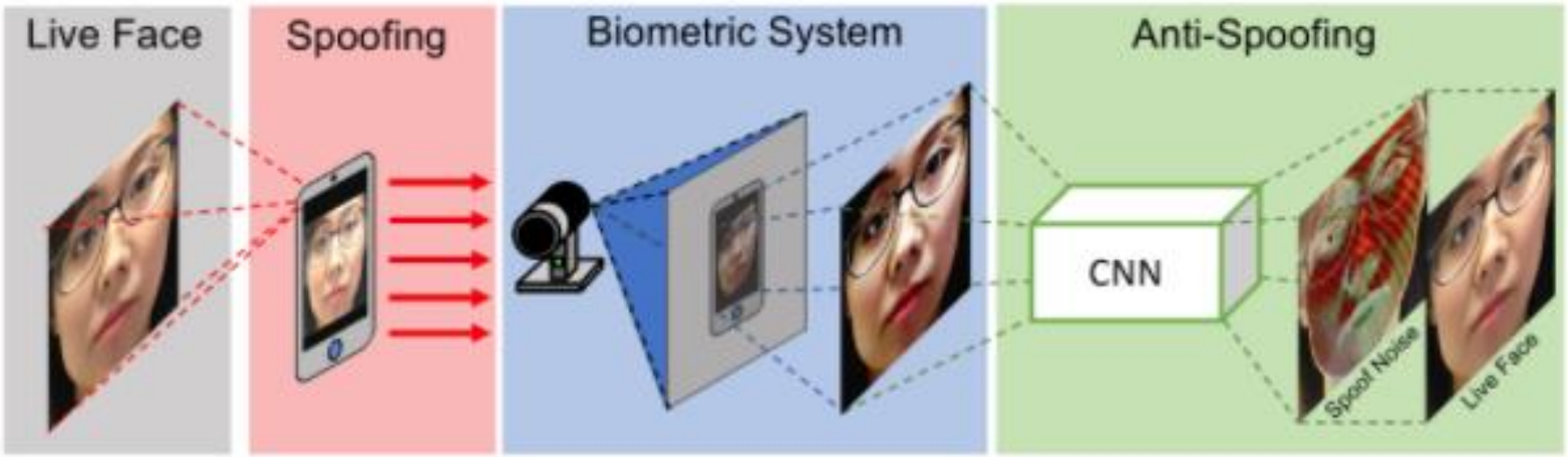


image credit: Jourabloo et al. (2018), Face De-Spoofing

face spoof detection

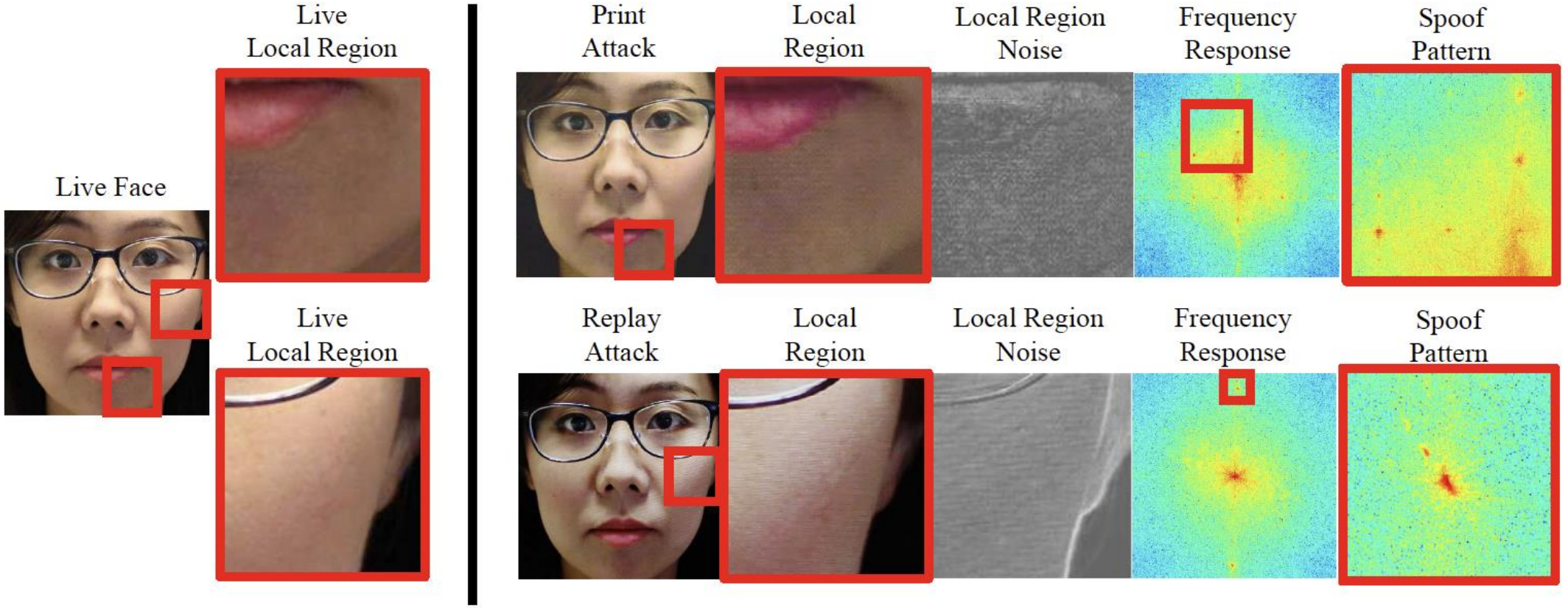


image credit: Jourabloo et al. (2018), Face De-Spoofing

face spoof detection

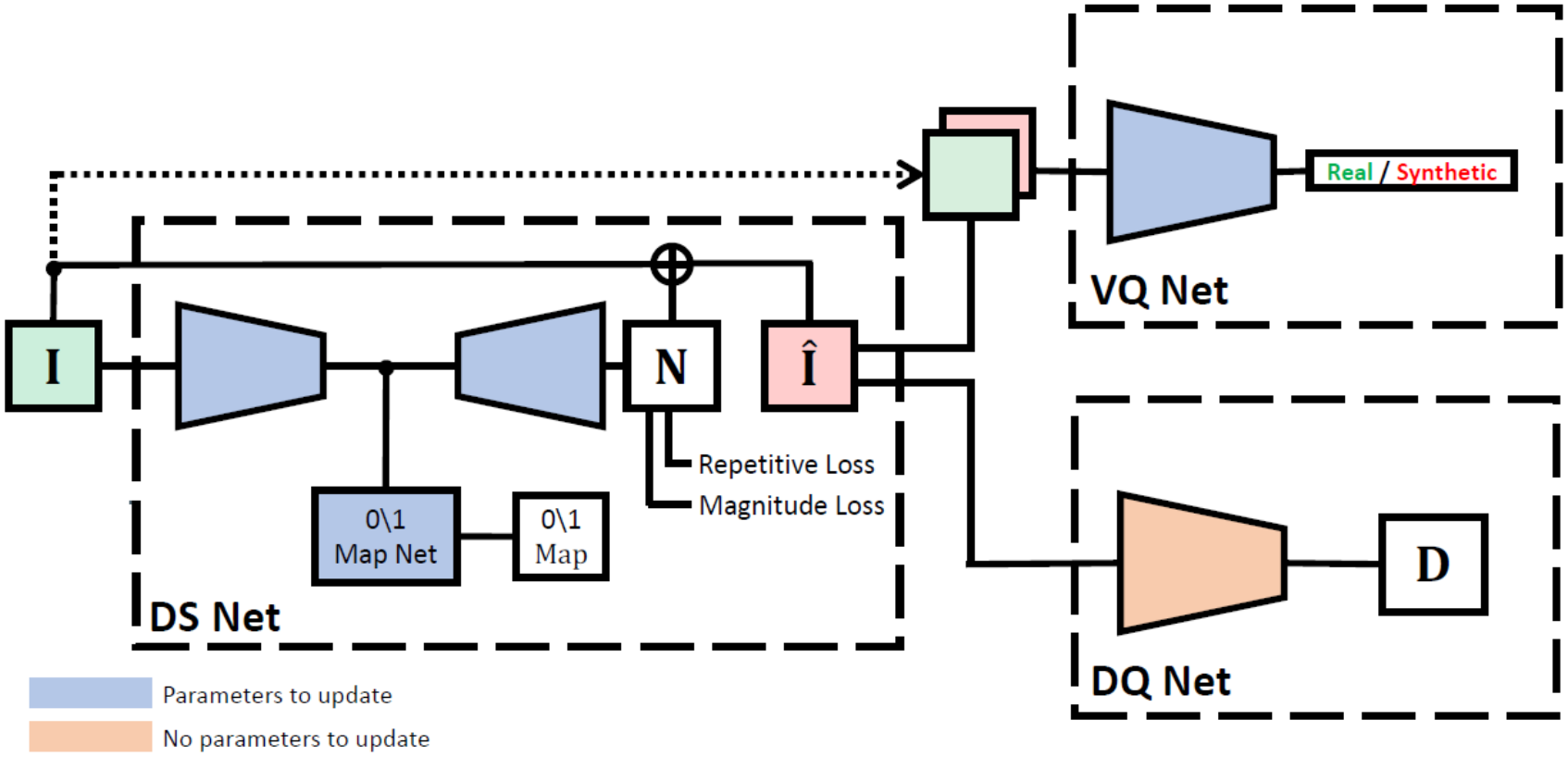
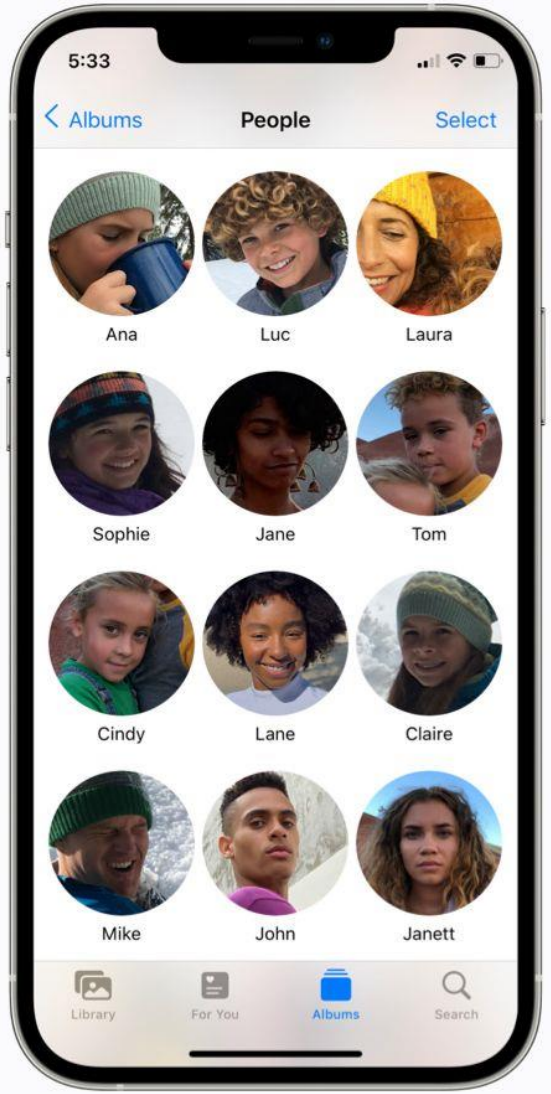


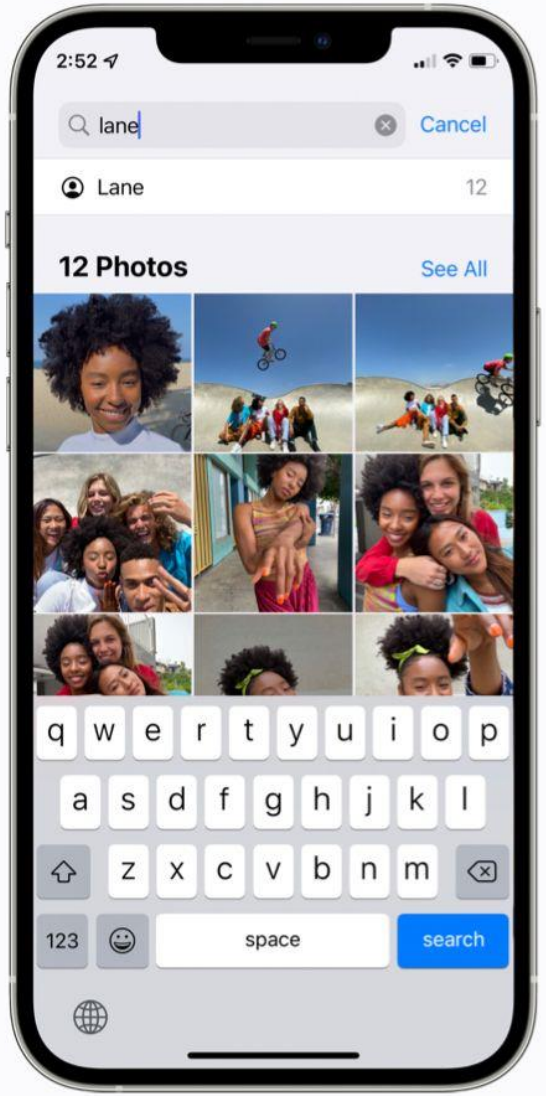
image credit: Jourabloo et al. (2018), Face De-Spoofing


Add a caption
Thursday • Jul 11, 2019 • 2:58 PM
Adjust
68A9A4D6-AF5F-49D7-A23F-D5F2046B2...
Apple iPhone 11
HEIF
Front Camera — 2.71mm ƒ2.2
7 MP • 2316 × 3088 • 1.3 MB
ISO 25
2.71 mm
0 ev
ƒ2.2
1/702 s
Santa Monica
Municipal Airport (SMO)

A


5:33
Albums
People
Select
Ana
Luc
Laura
Sophie
Jane
Tom
Cindy
Lane
Claire
Mike
John
Janett
Library
For You
Albums
Search

B


2:52
lane
Cancel
Lane
12
12 Photos
See All
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
search

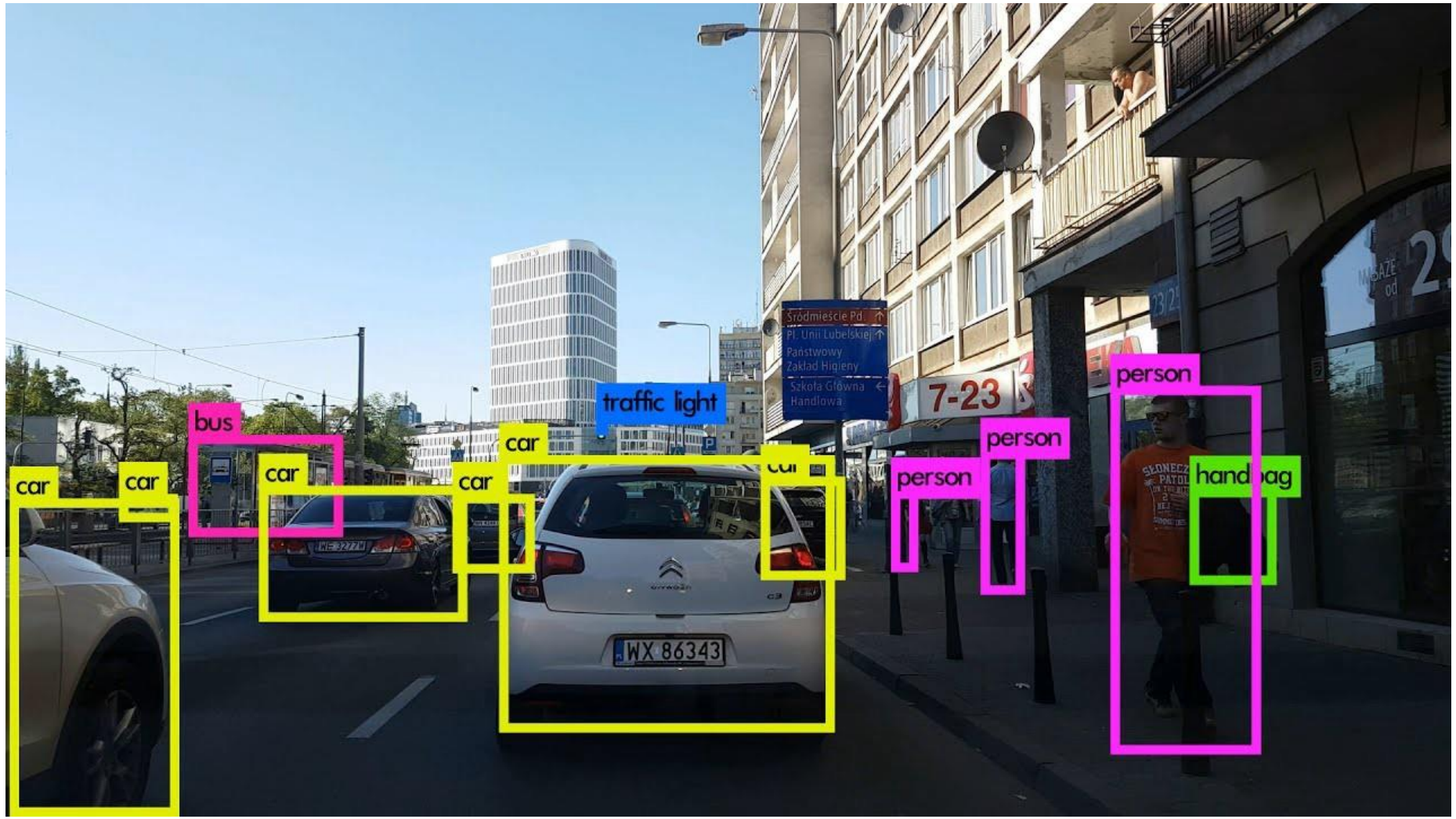
C


TOGETHER
Jun 21 – 22, 2021

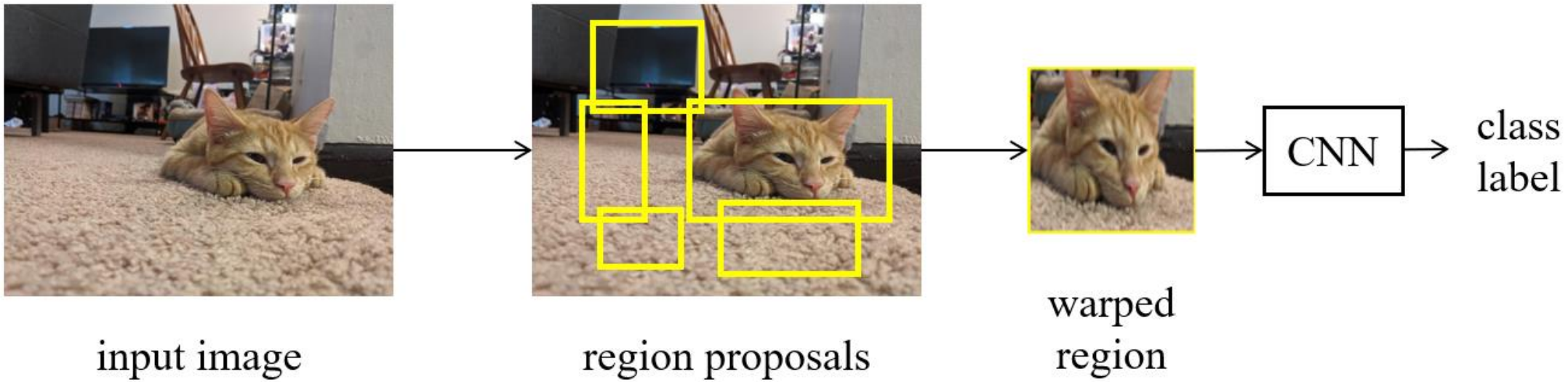
D

4.4 CNN in object detection

object detection



R-CNN


CNN
class label
input image
region proposals
warped region

R-CNN

- regions are proposed via “selective search”, a fixed algorithm (no learning involved):

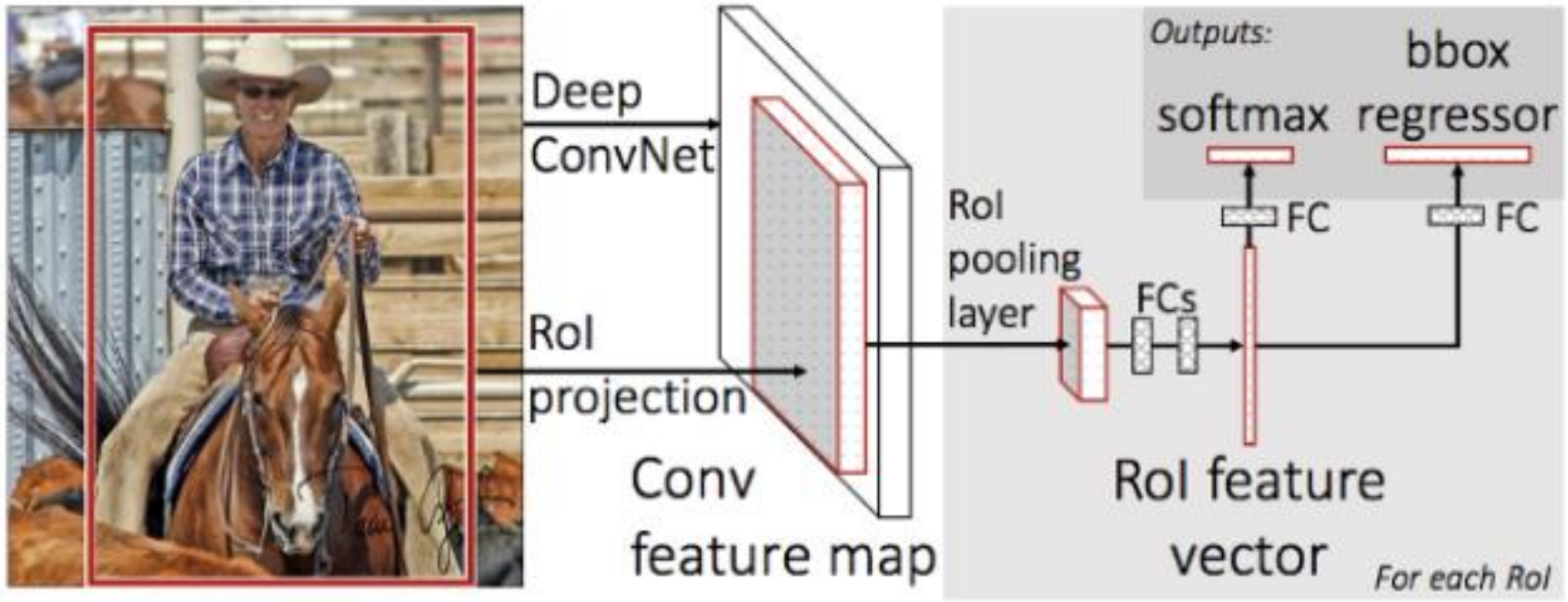
1. generate initial sub-segmentation (many candidates)
2. use greedy algorithm to recursively combine similar regions
3. produce the final candidate region proposals

R-CNN

- relatively slow in training (2000 proposed regions)
- cannot be implemented real time (~47 seconds for each test image)
- the selective search algorithm is a fixed algorithm involves no learning

fast R-CNN

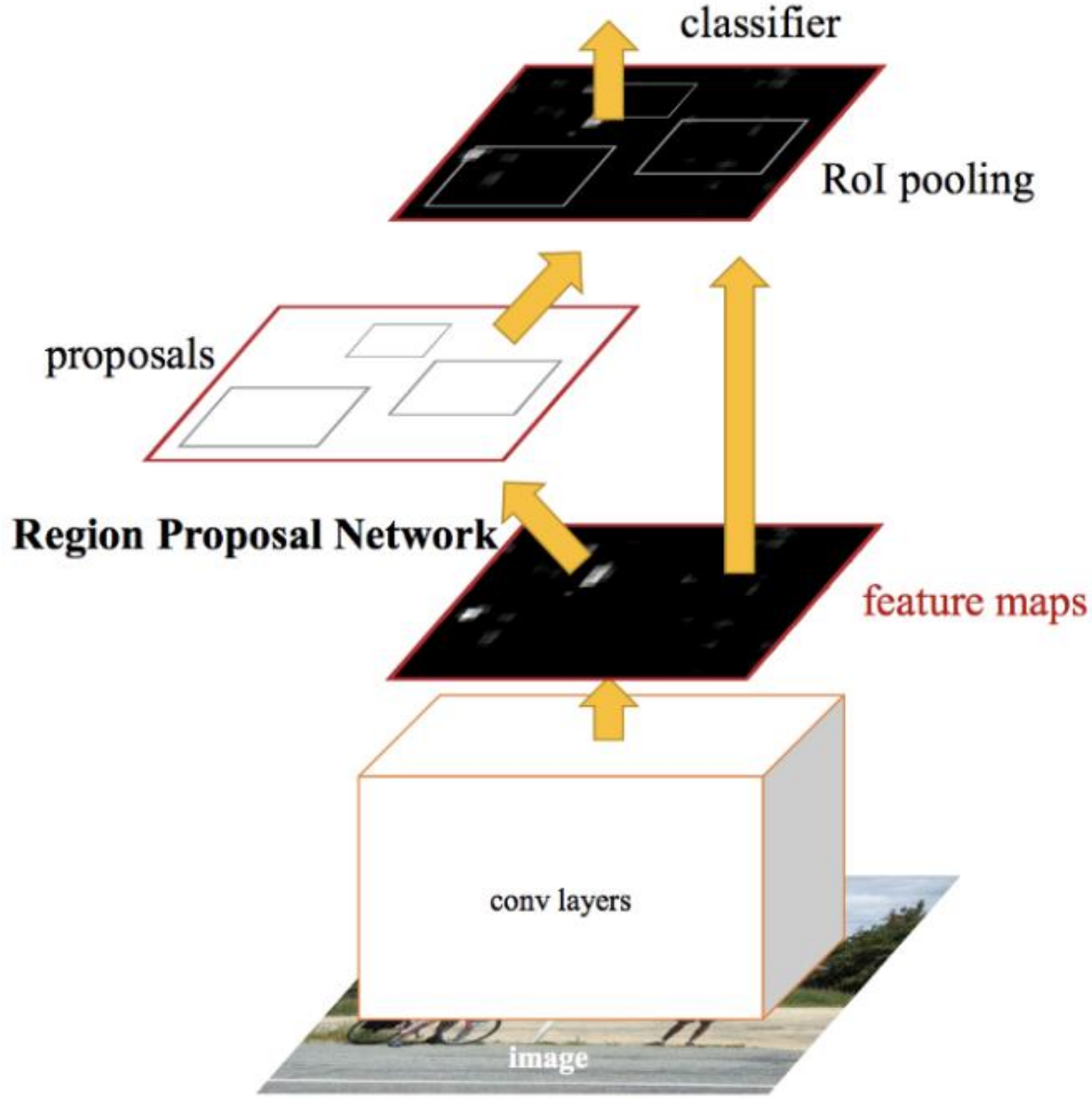
selective search algorithm on the feature map



the input image, instead of the proposed regions, is fed to the CNN to generate a convolutional feature map

faster R-CNN

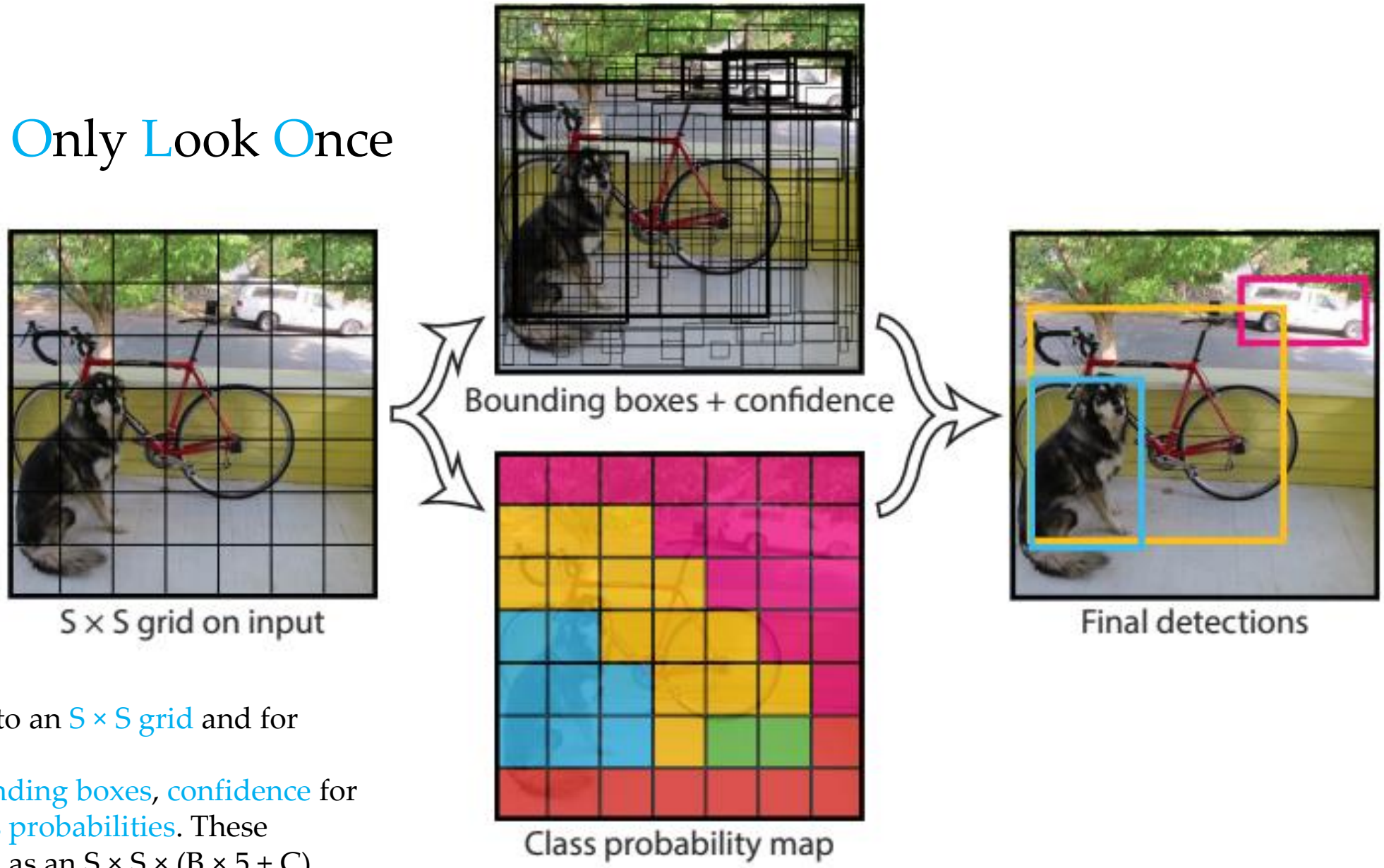
no selective search algorithm used on the feature map



A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation.

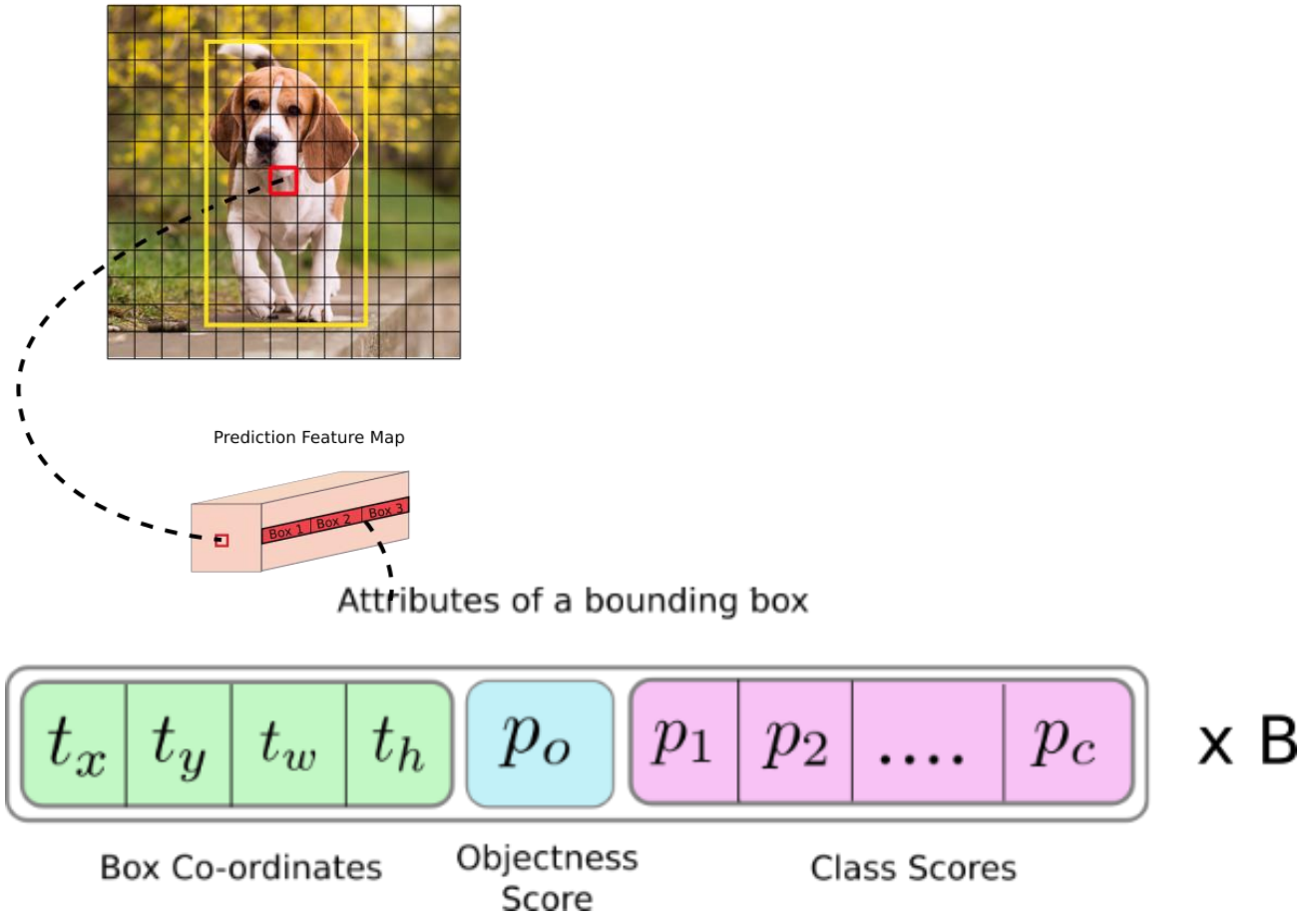
YOLO

- YOLO = You Only Look Once



The image is divided into an S × 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 × S × (B × 5 + C)

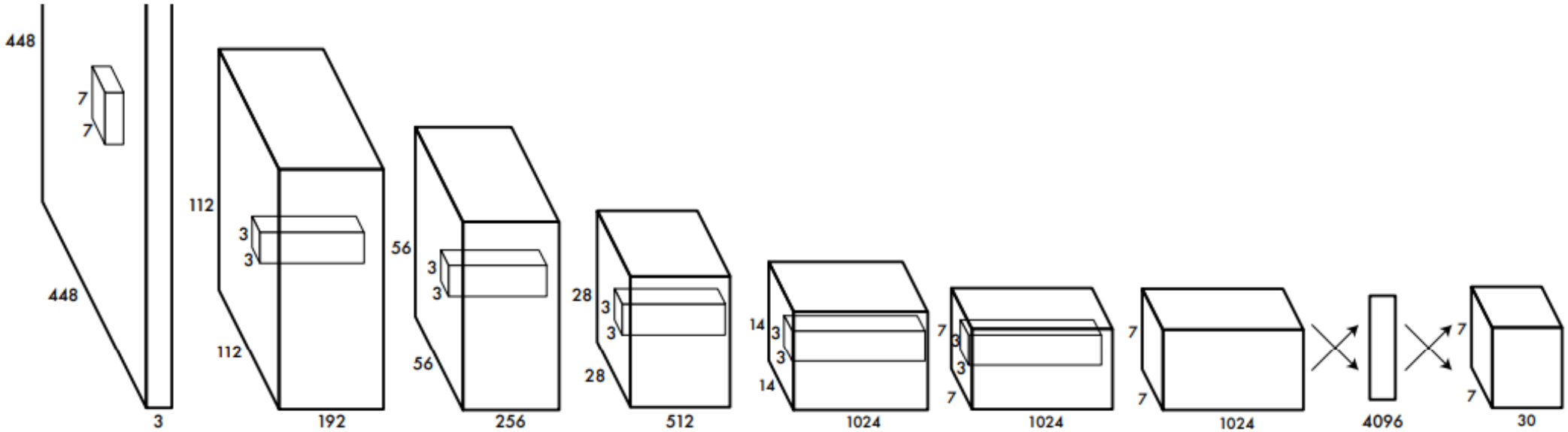
YOLO



The image is divided into an S × 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 × S × (B × 5 + C)

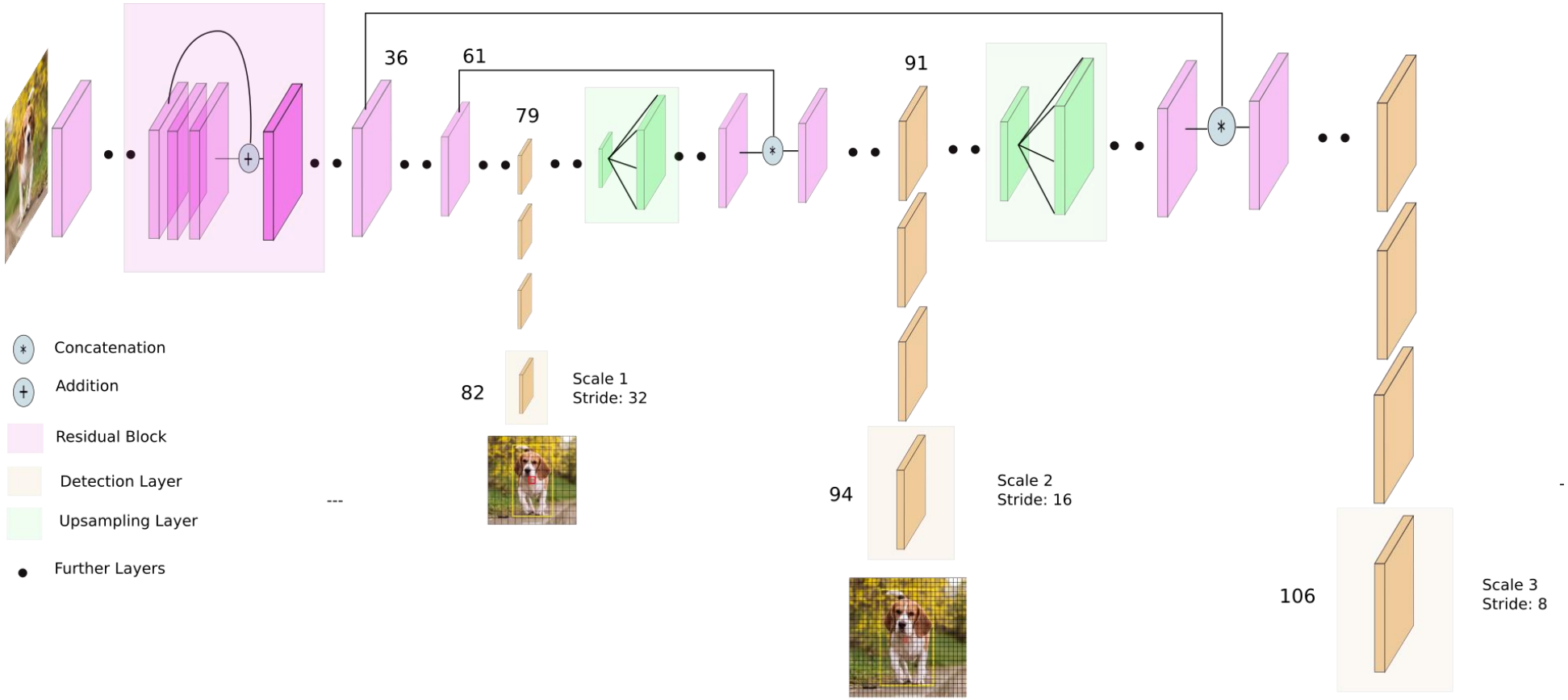
YOLO

too big, too strong, too fast, too good!!!



YOLOv3

fully convolutional network: only convolutional layers

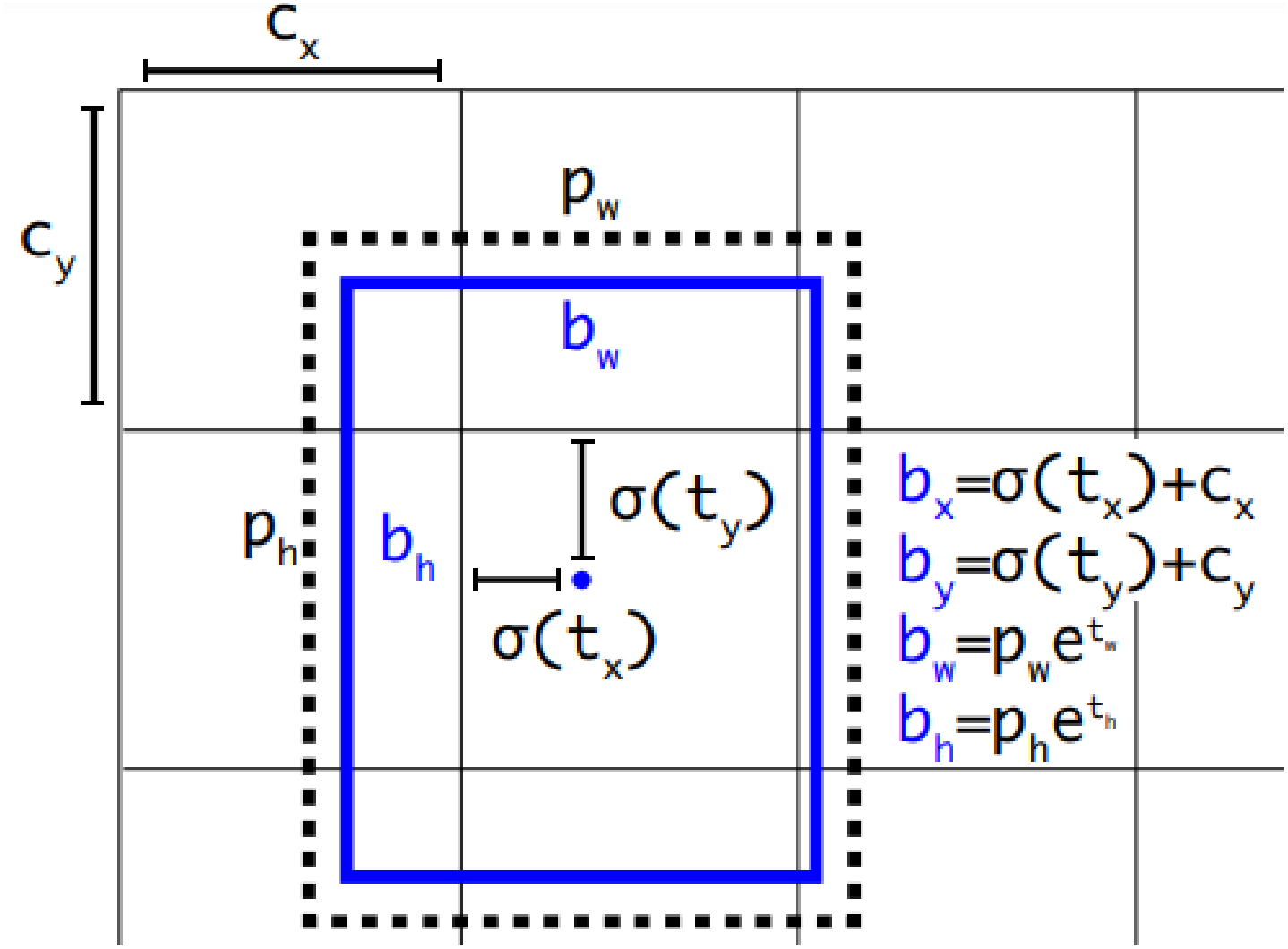


YOLO v3 network Architecture

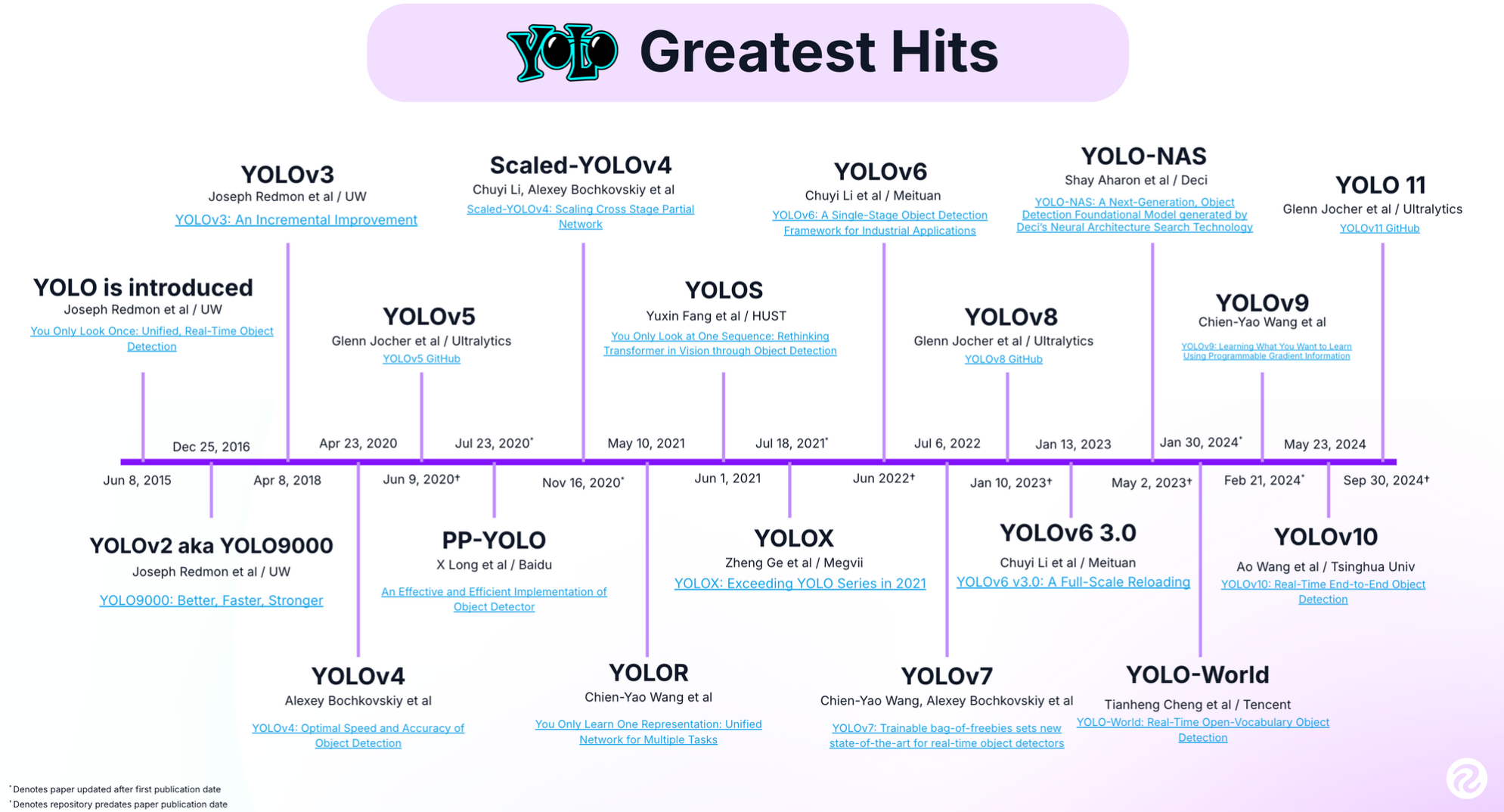
image credit: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

YOLOv3

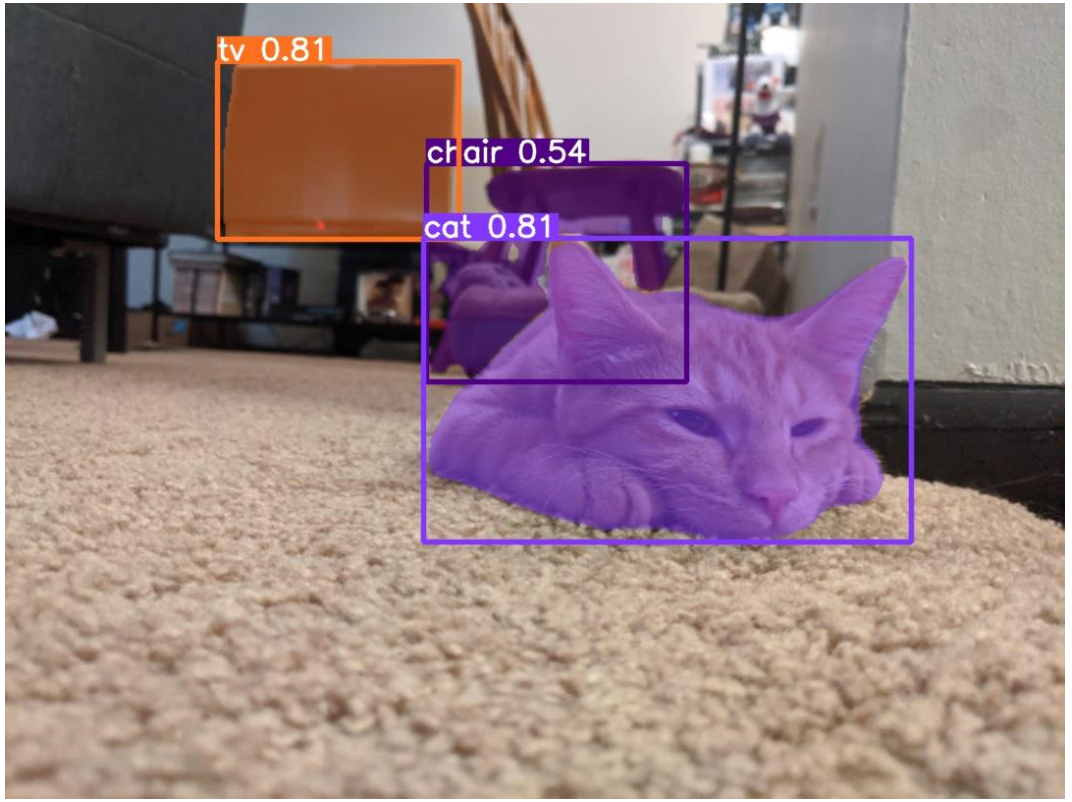
predicts bounding boxes using dimension clusters as anchor boxes



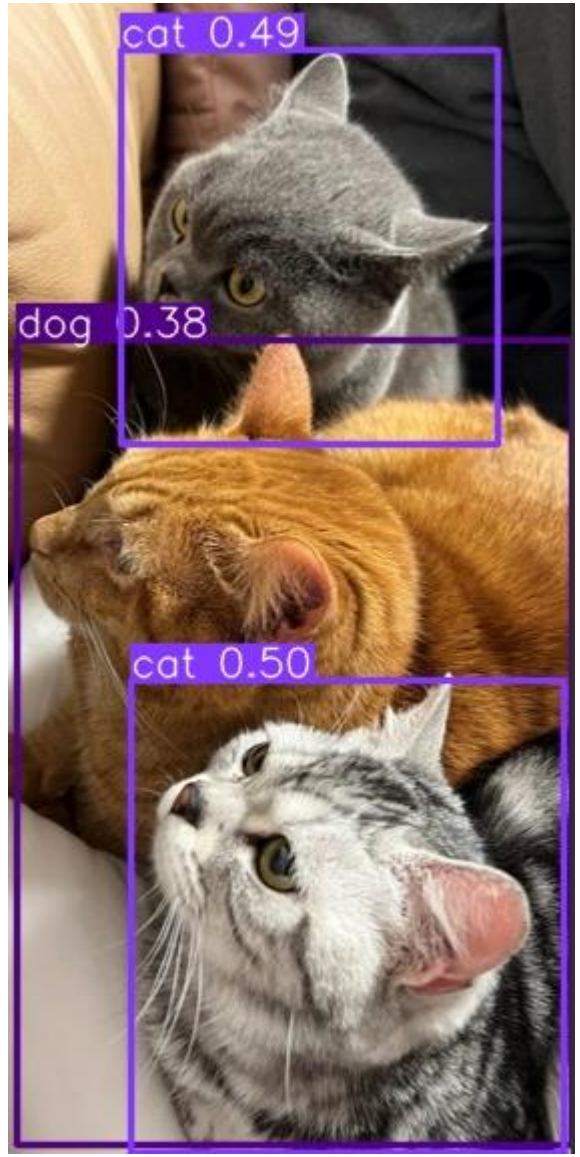
genesis of YOLO

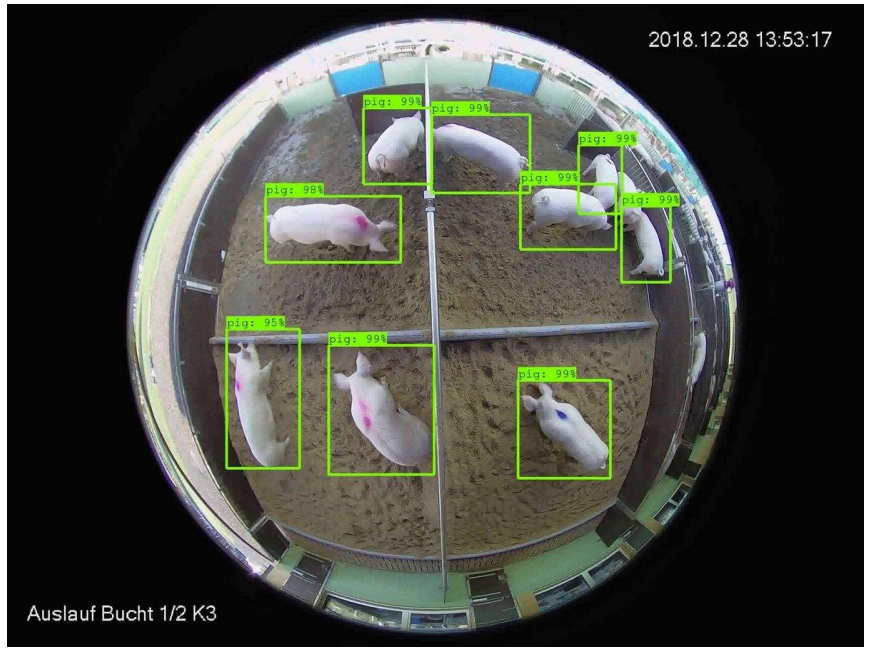


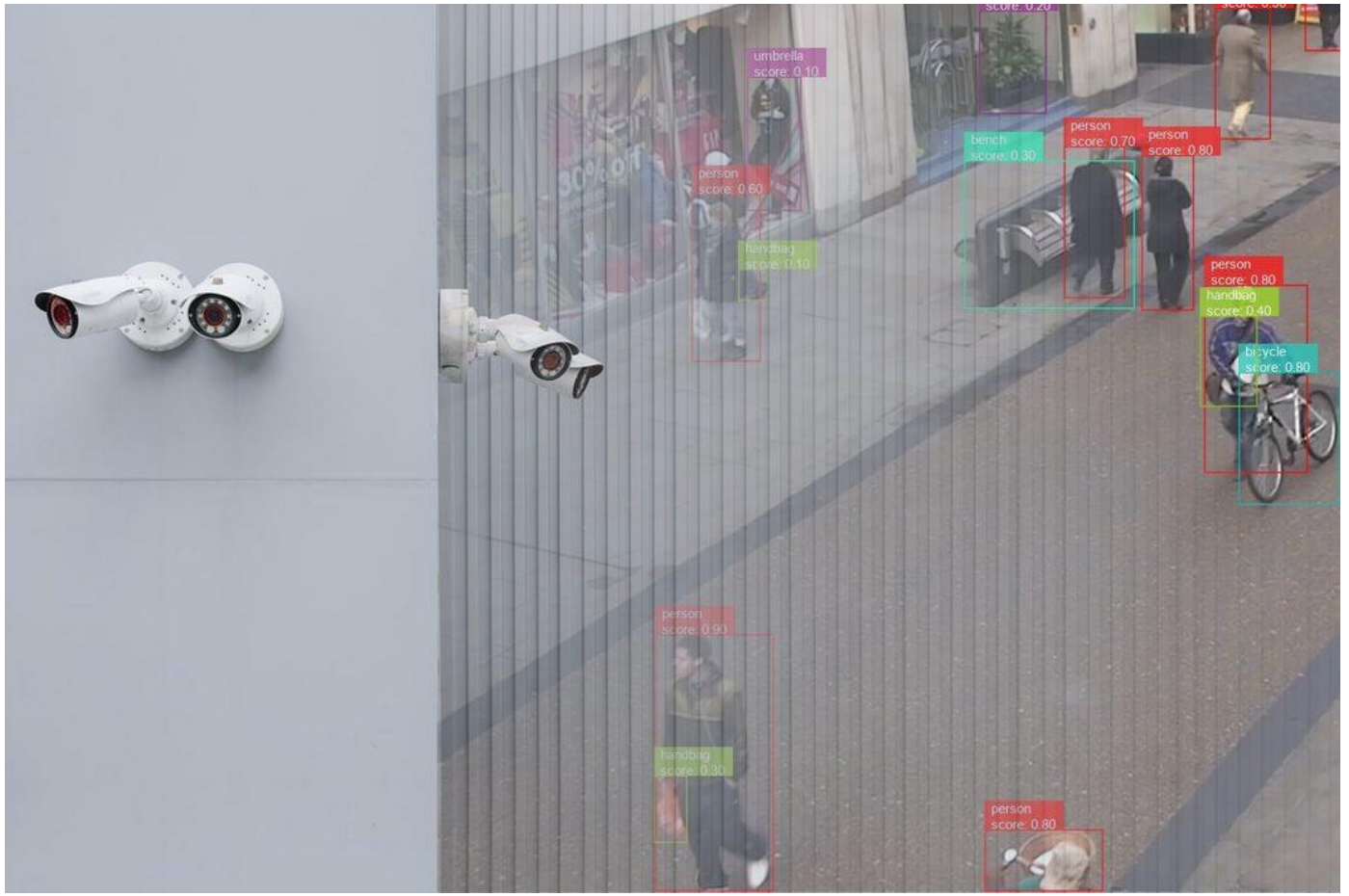
https://blog.roboflow.com/guide-to-yolo-models/

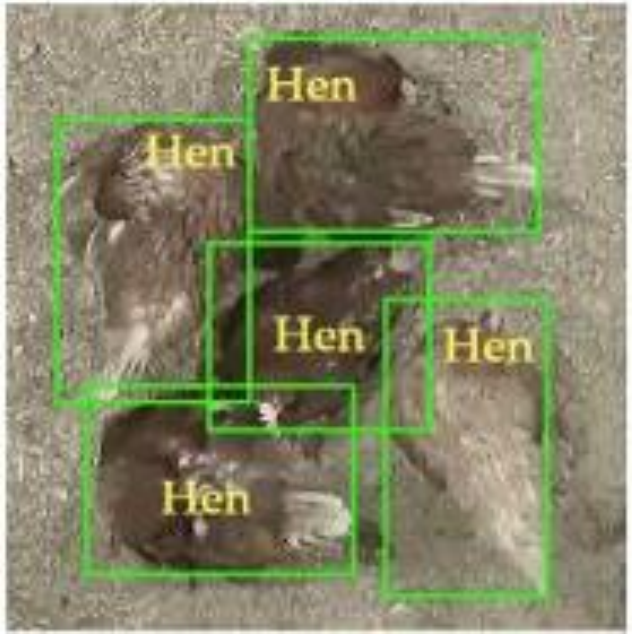

tv 0.81
chair 0.54
cat 0.81

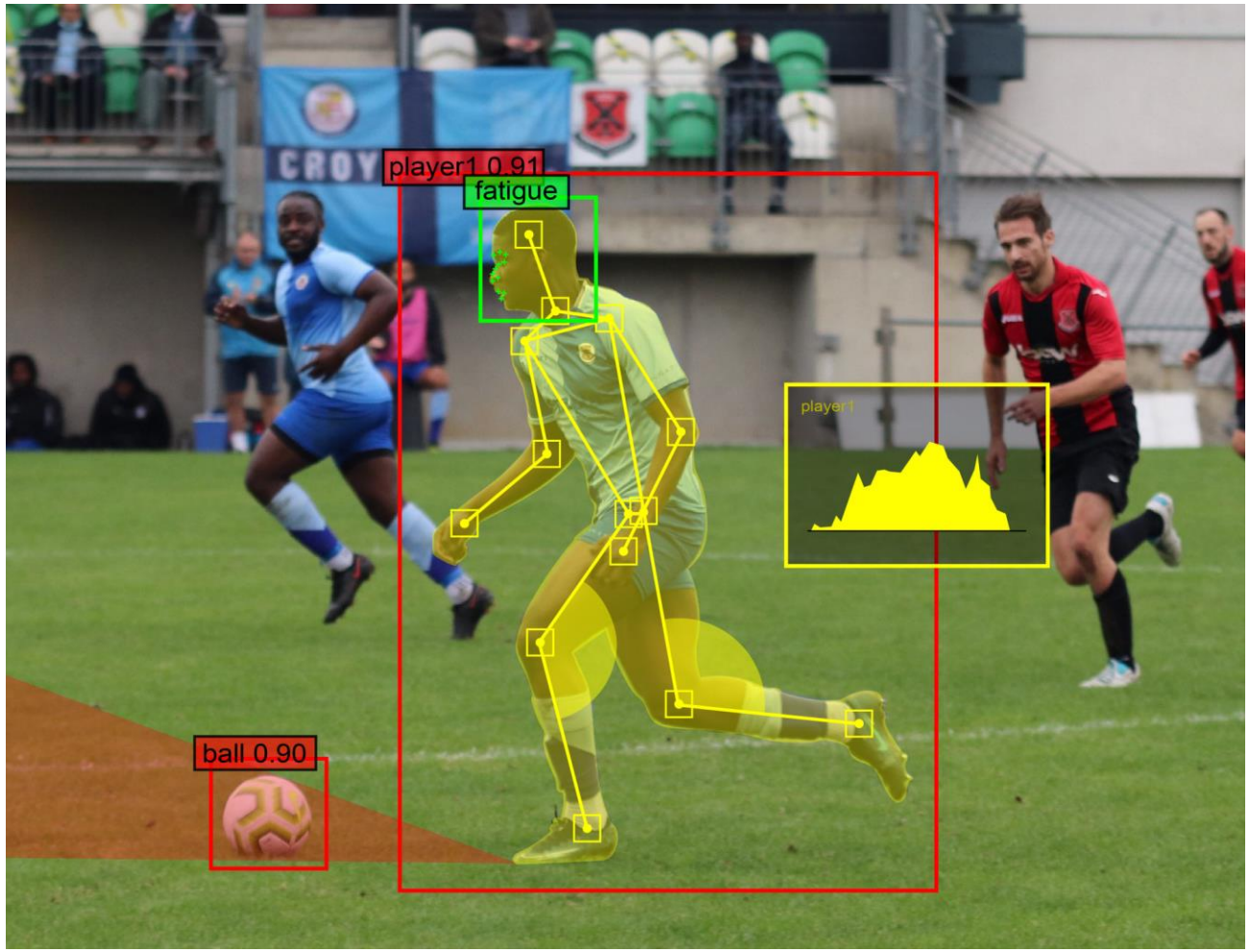

microwave 0.56
bottle 0.82
sink 0.36
chair 0.76
chair 0.73


cat 0.49
dog 0.38
cat 0.50


2018.12.28 13:53:17
pig: 99%
pig: 99%
pig: 99%
pig: 98%
pig: 99%
pig: 99%
pig: 95%
pig: 99%
pig: 99%
Auslauf Bucht 1/2 K3


umbrella
score: 0.10
person
score: 0.60
handbag
score: 0.10
bench
score: 0.30
person
score: 0.70
person
score: 0.80
person
score: 0.80
handbag
score: 0.40
bicycle
score: 0.80
person
score: 0.90
handbag
score: 0.30
person
score: 0.80

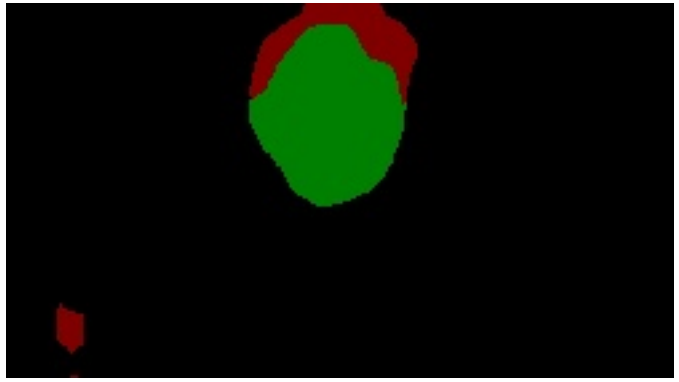

Hen
Hen
Hen
Hen
Heh


player1 0.91
fatigue
player1
ball 0.90

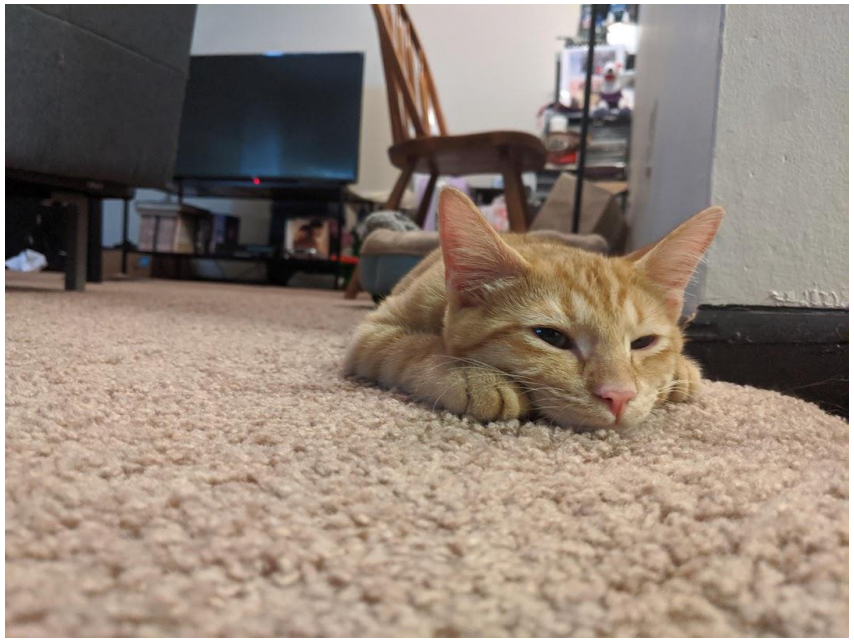
4.5 CNN in segmentation

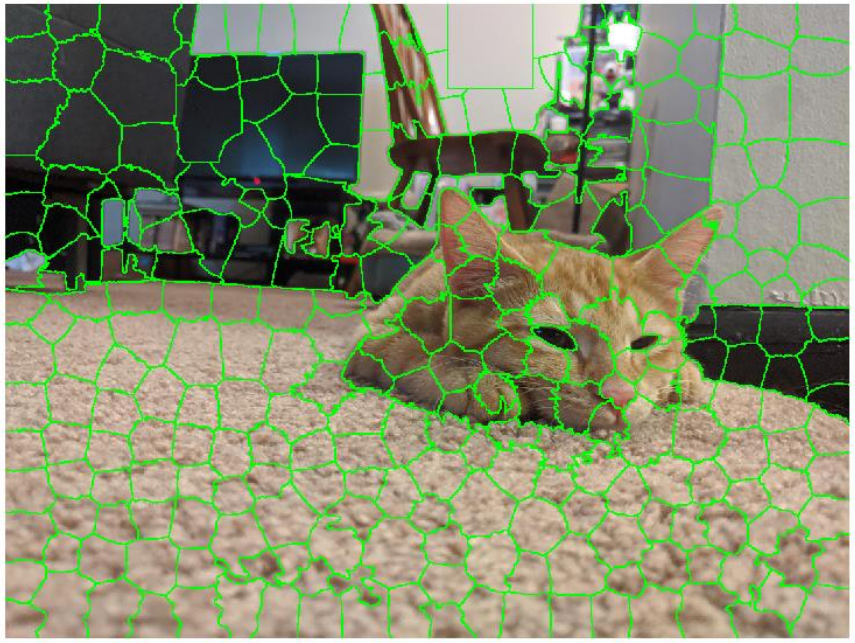
face segmentation

segments an image into three areas of background, hair and face



superpixels





Labeled Faces in the Wild

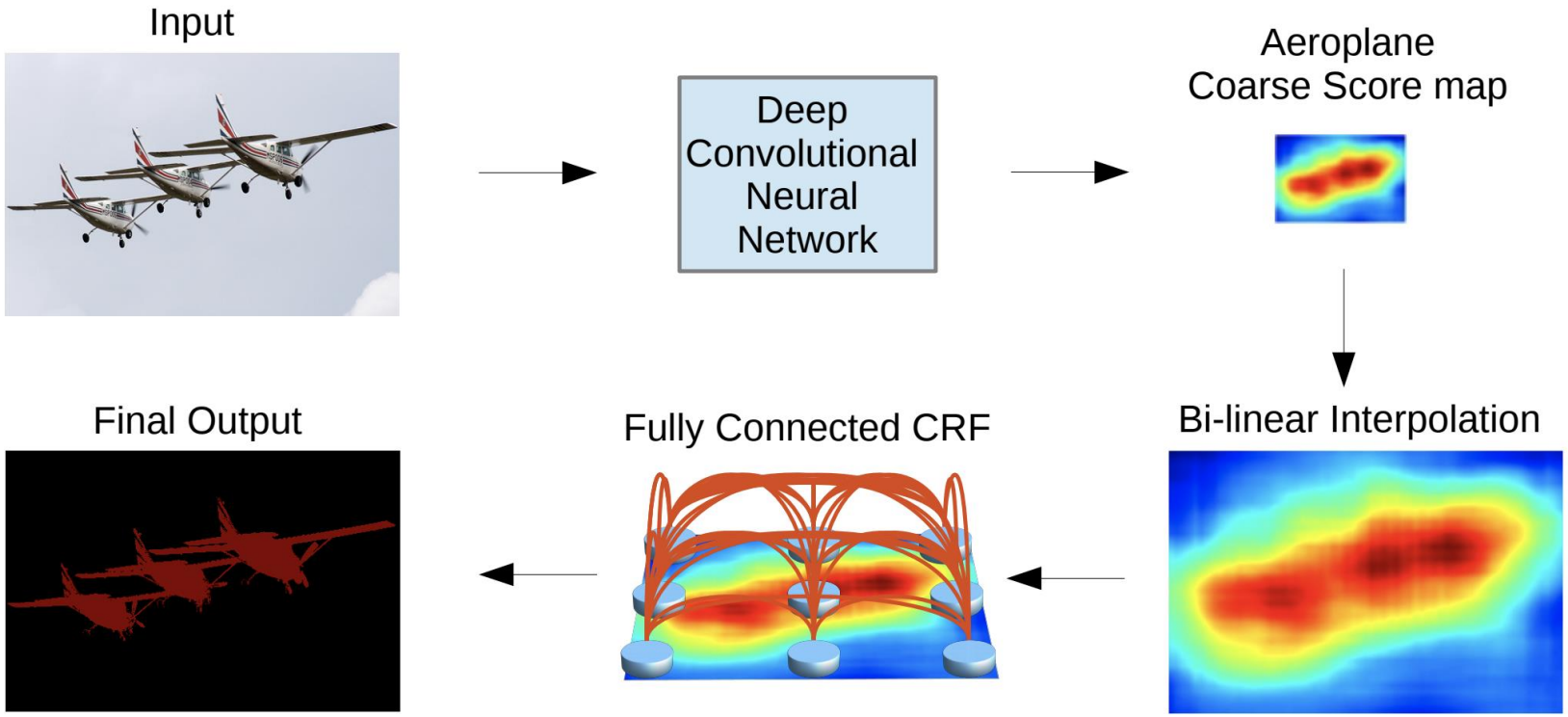
superpixels groundtruth

image credit: http://vis-www.cs.umass.edu/lfw/part_labels/

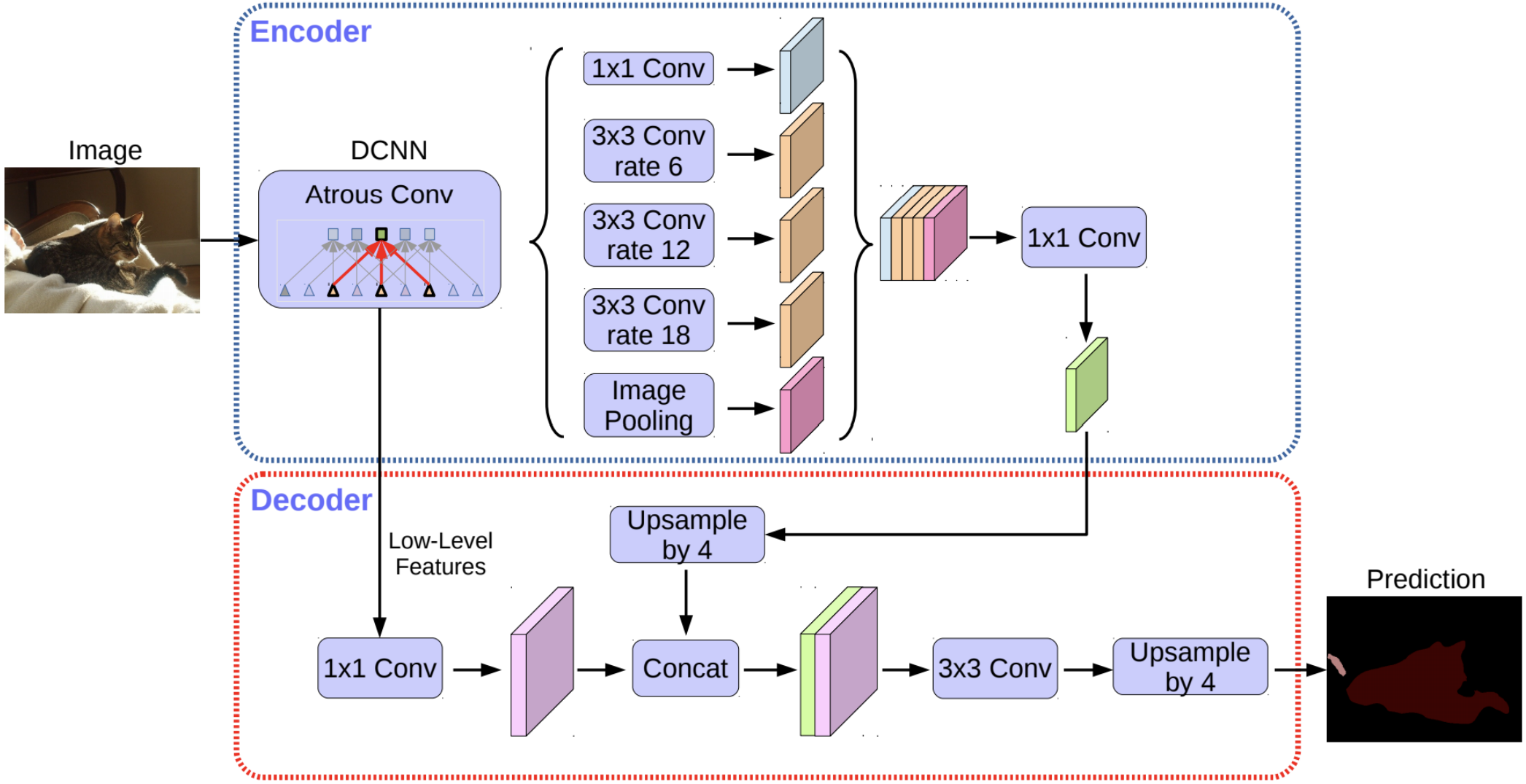
semantic segmentation

semantic segmentation

- DeepLab

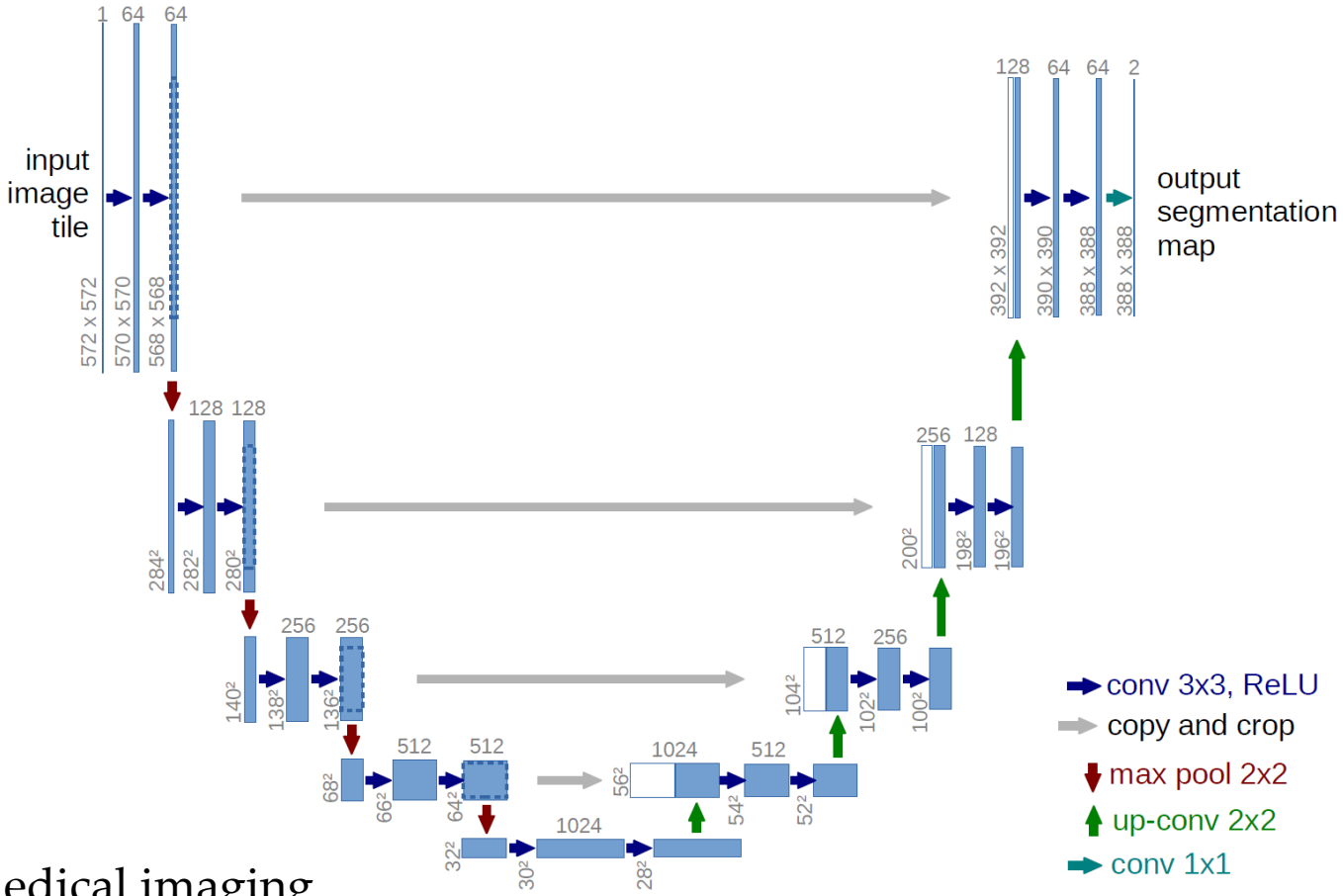


DeepLabv3


Encoder
Image
DCNN
Atrous Conv
1x1 Conv
3x3 Conv rate 6
3x3 Conv rate 12
3x3 Conv rate 18
Image Pooling
1x1 Conv
Decoder
Low-Level Features
Upsample by 4
1x1 Conv
Concat
3x3 Conv
Upsample by 4
Prediction

deeplab

UNet



widely used for medical imaging

https://github.com/milesial/Pytorch-

instance segmentation and panoptic segmentation

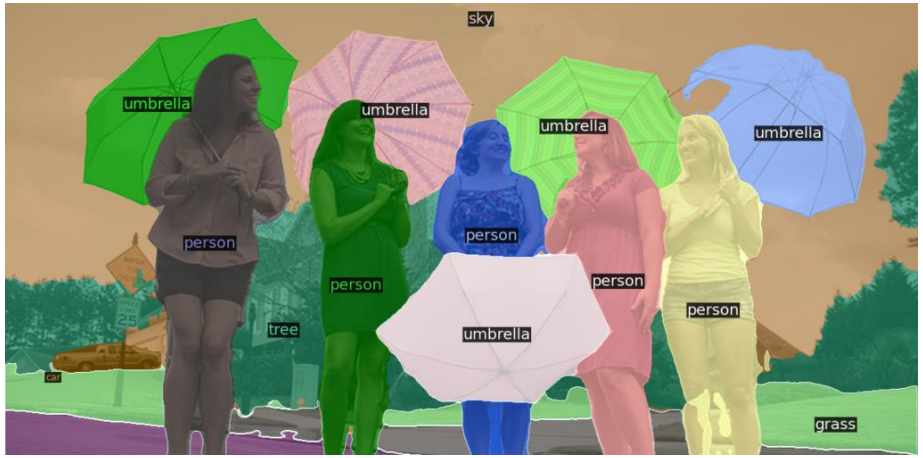
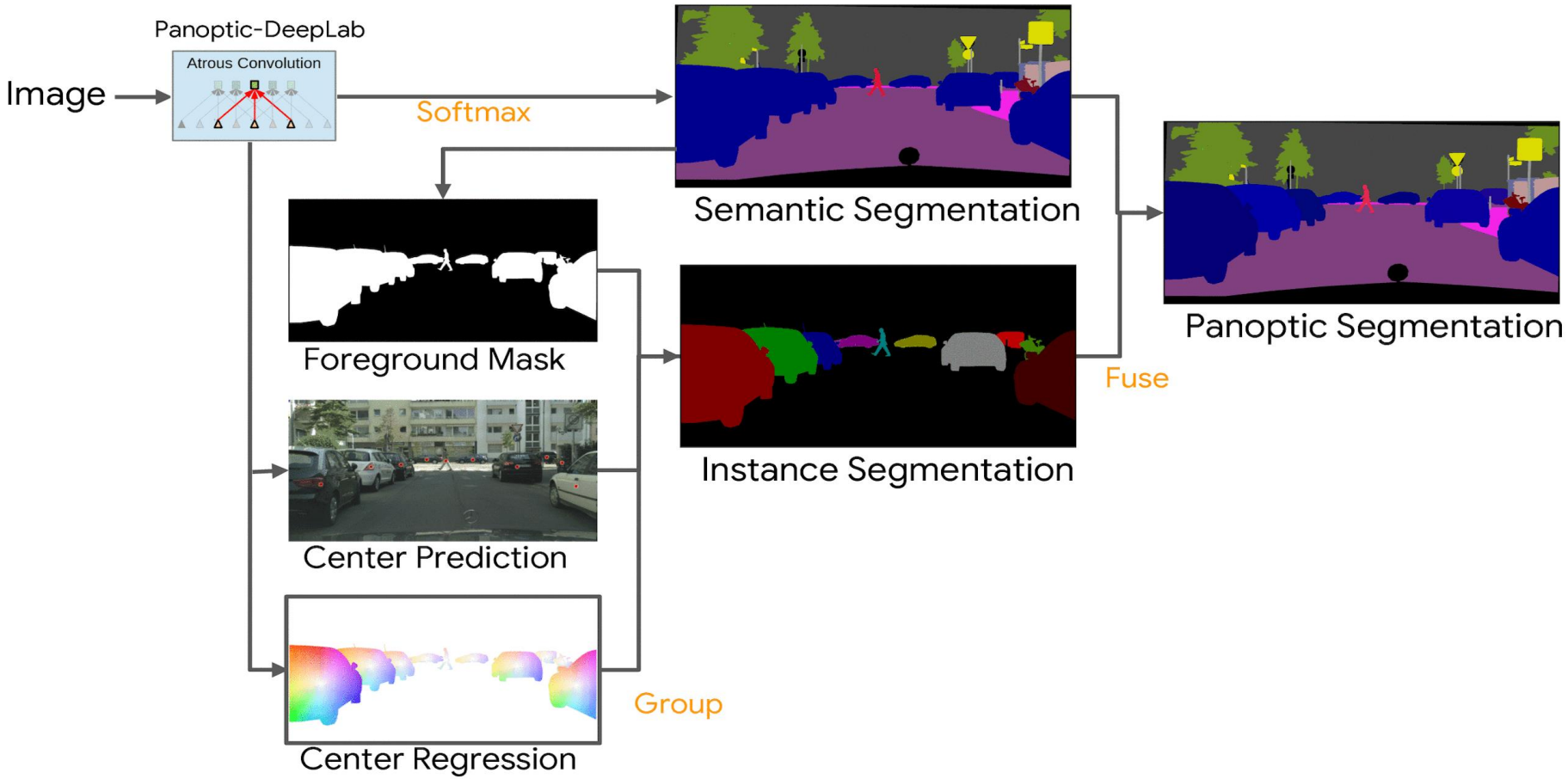


image credit: labellerr.com

Panoptic-DeepLab



Thank you!

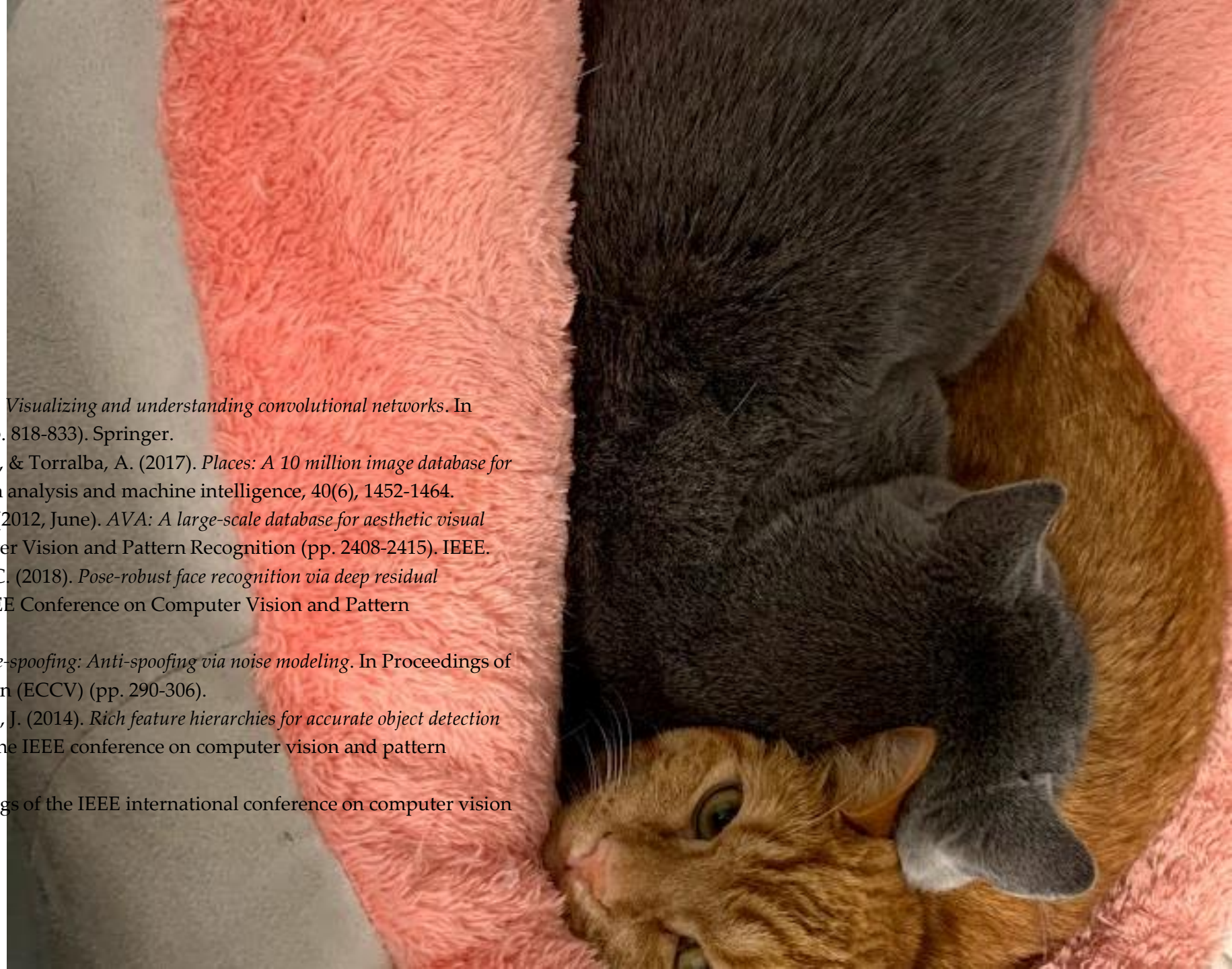
Reference

- Ch 5, 7,13, *Deep Learning for Coders with fastai and PyTorch.*
- Ch 9, *Deep Learning*.
- Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *ICLR 2015*.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR* (pp. 1-9).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR* (pp. 770-778).

Thank you!

Reference

- Zeiler, M. D., & Fergus, R. (2014, September). *Visualizing and understanding convolutional networks*. In European conference on computer vision (pp. 818-833). Springer.
- Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2017). *Places: A 10 million image database for scene recognition*. IEEE transactions on pattern analysis and machine intelligence, 40(6), 1452-1464.
- Murray, N., Marchesotti, L., & Perronnin, F. (2012, June). *AVA: A large-scale database for aesthetic visual analysis*. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 2408-2415). IEEE.
- Cao, K., Rong, Y., Li, C., Tang, X., & Loy, C. C. (2018). *Pose-robust face recognition via deep residual equivariant mapping*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5187-5196).
- Jourabloo, A., Liu, Y., & Liu, X. (2018*). Face de-spoofing: Anti-spoofing via noise modeling*. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 290-306).
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
- Girshick, R. (2015). *Fast R-CNN*. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).



Thank you!

Reference

- Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster R-CNN: Towards real-time object detection with region proposal networks*. Advances in neural information processing systems, 28, 91-99.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
- Redmon, J., & Farhadi, A. (2018). *Yolov3: An incremental improvement*. arXiv preprint arXiv:1804.02767.
- Kae, A., Sohn, K., Lee, H., & Learned-Miller, E. (2013). *Augmenting CRFs with Boltzmann machine shape priors for image labeling*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2019-2026).
- Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs*. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.
- Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., & Van Gool, L. (2017). *Pose guided person image generation*. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 405-415).
- Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). *Unpaired image-to-image translation using cycle-consistent adversarial networks*. In Proceedings of the IEEE international conference on computer vision (pp. 2223-2232).
- Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). *Photo-realistic single image super-resolution using a generative adversarial network*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4681-4690).