Deep learning

- Representation learning

- Complex pipeline learning

Traditional machine learning: Feature computation

Features that describe the object

- concentration
- asymmetry
- smoothness
- entropy
- spirality, etc

$$(x_1, x_2, \ldots, x_n) \Rightarrow \boxed{\begin{array}{c}\text{generic model} \\ y = f_w(x_1, x_2, \ldots, x_n)\end{array}} \Rightarrow y = \text{spiral}$$

Traditional ML

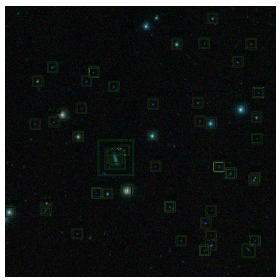
$\Rightarrow$ feature extraction/ selection $\Rightarrow$ $(x_1, x_2, \ldots, x_n)$ $\Rightarrow$ Traditional ML algoritm $\Rightarrow$ y

Traditional ML

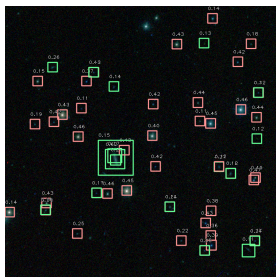
⇒ feature extraction/ selection ⇒ $(x_1, x_2, \ldots, x_n)$ ⇒ Traditional ML algoritm ⇒ y

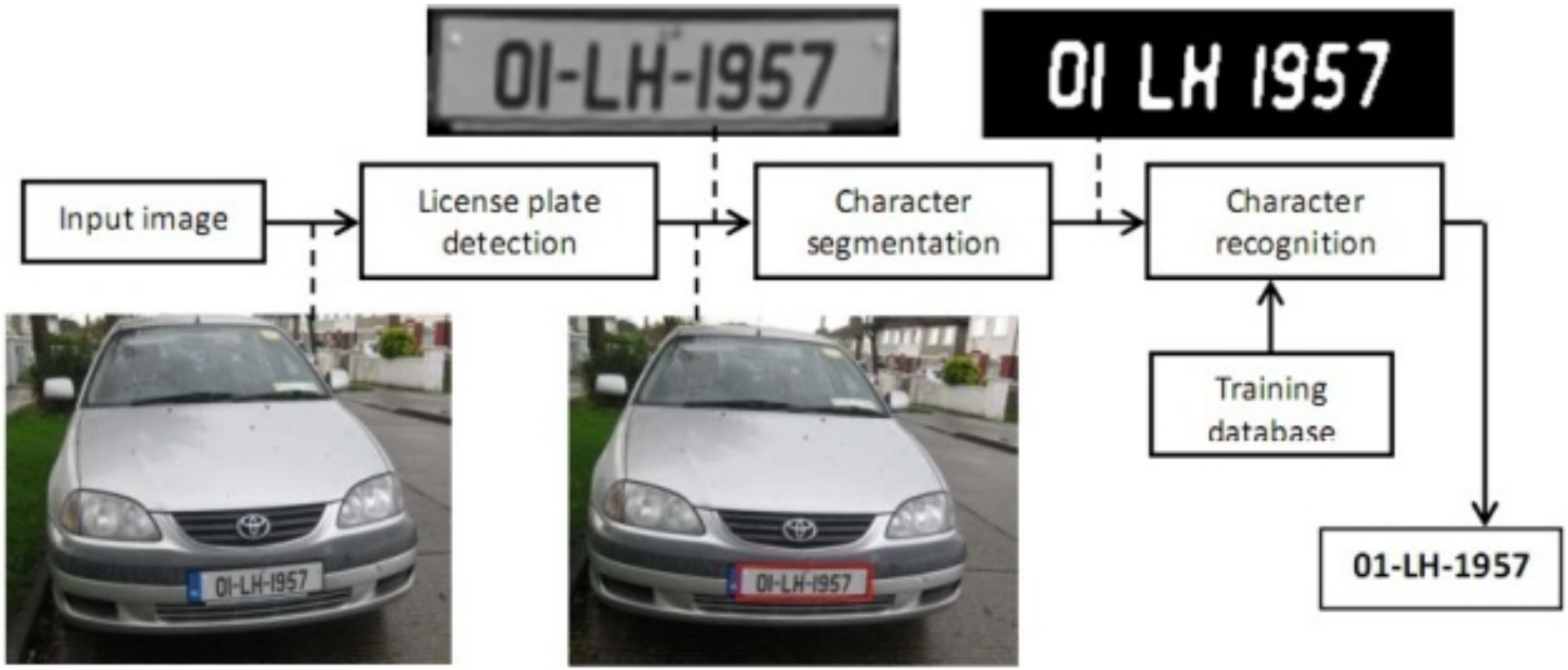
Deep learning: end-to-end processing

⇒ Deep learning algorithm ⇒ y



⇒ Deep learning algorithm ⇒




01-LH-1957
01 LH 1957
Input image
License plate detection
Character segmentation
Character recognition
Training database
01-LH-1957

When it started: (This information is not precise)

> *[From Wikipedia] In 2006, publications by Geoff Hinton, Ruslan Salakhutdinov, Osindero and Teh[59] [60][61] showed how a many-layered feedforward neural network could be effectively pre-trained one layer at a time, treating each layer in turn as an unsupervised restricted Boltzmann machine, then fine-tuning it using supervised backpropagation.[62] The papers referred to learning for deep belief nets.*

[59] Hinton, Geoffrey E. (2007). "Learning multiple layers of representation". Trends in Cognitive Sciences. 11(10):428–434

[60] Hinton, G. E.; Osindero, S.; Teh, Y. W. (2006). "A Fast Learning Algorithm for Deep Belief Nets". Neural Computation. 18(7):1527–1554

[61] Bengio, Yoshua (2012). "Practical recommendations for gradient-based training of deep architectures"

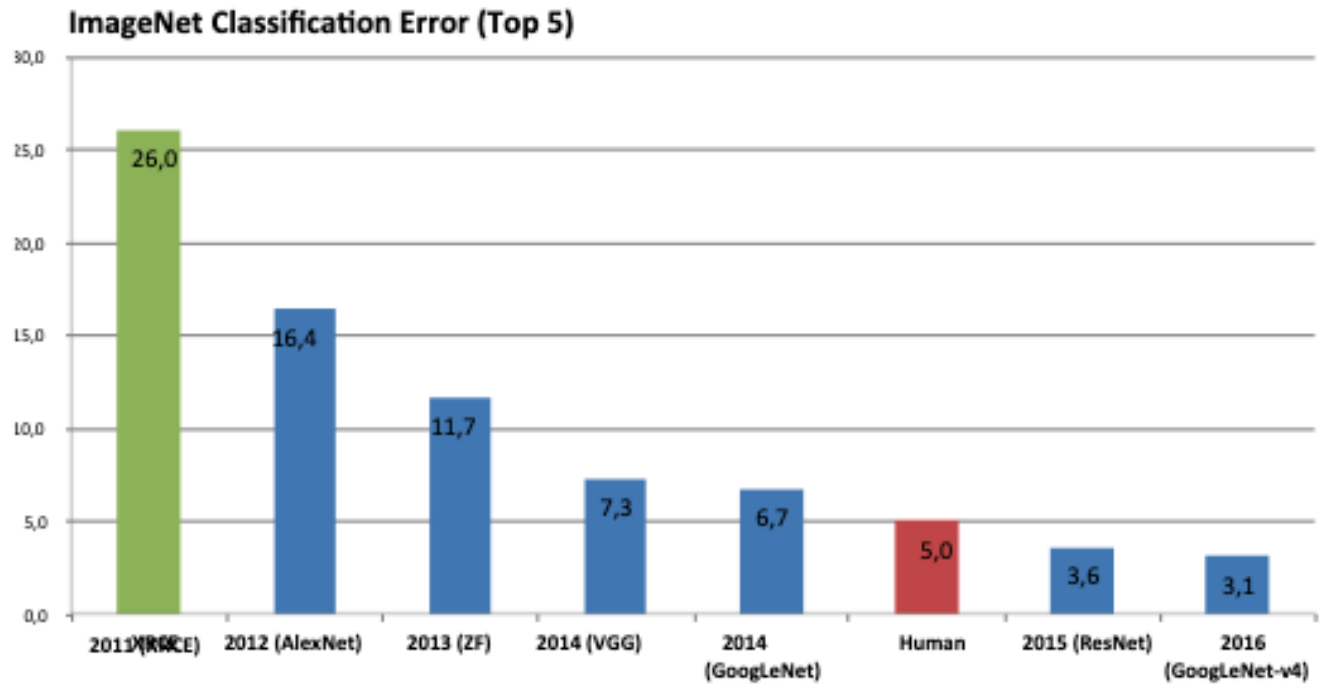
[62] G. E. Hinton., "Learning multiple layers of representation," Trends in Cognitive Sciences, 11, 428–434, 2007

Deep learning in Computer Vision

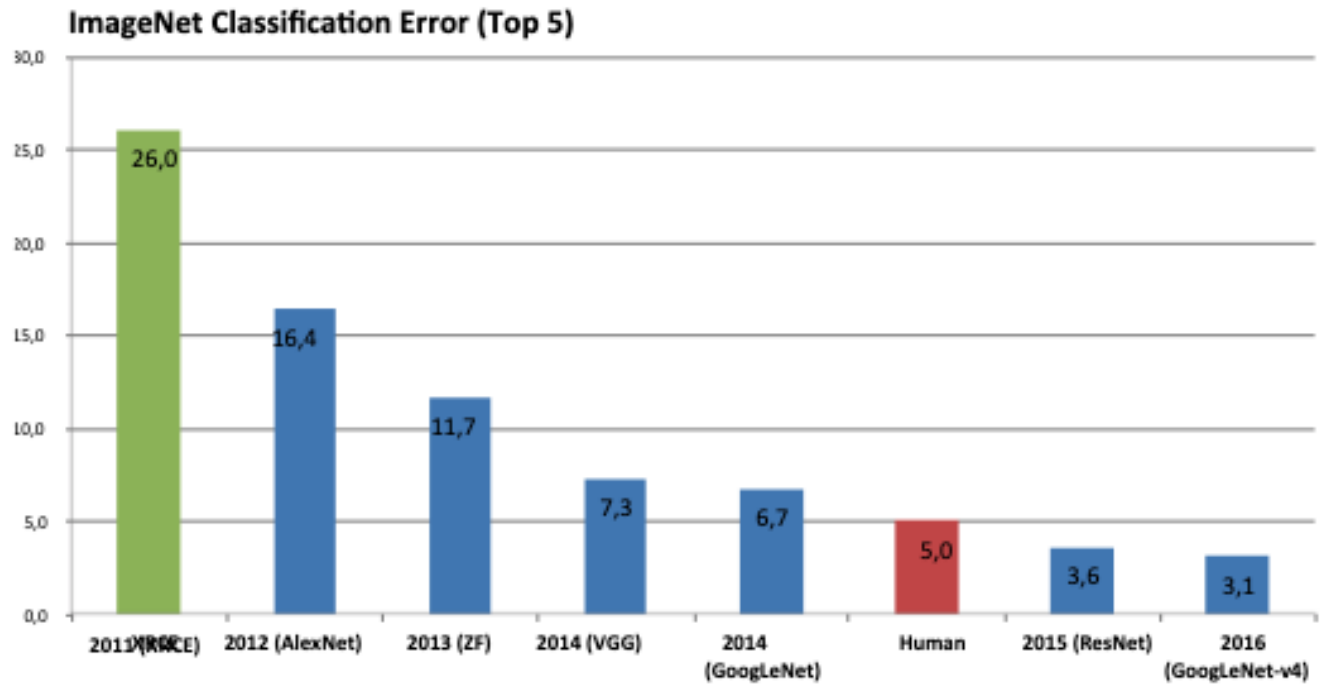
> *[From Wikipedia] In October 2012, ... a system by Krizhevsky et al.[5] won the large-scale ImageNet competition by a significant margin over shallow machine learning methods.*

[5]. Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffry (2012). "ImageNet Classification with Deep Convolutional Neural Networks" (PDF). NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada.

Large Scale Visual Recognition Challenge (ILSVRC) - ImageNet



Large Scale Visual Recognition Challenge (ILSVRC) - ImageNet



IMAGENET

www.image-net.org

22K categories and 14M images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activities

Deng, Dong, Socher, Li, Li, & Fei-Fei, 2009

IMAGENET Large Scale Visual Recognition Challenge

Steel drum

The Image Classification Challenge:

1,000 object classes

1,431,167 images

Output:
Scale
T-shirt
Steel drum
Drumstick
Mud turtle

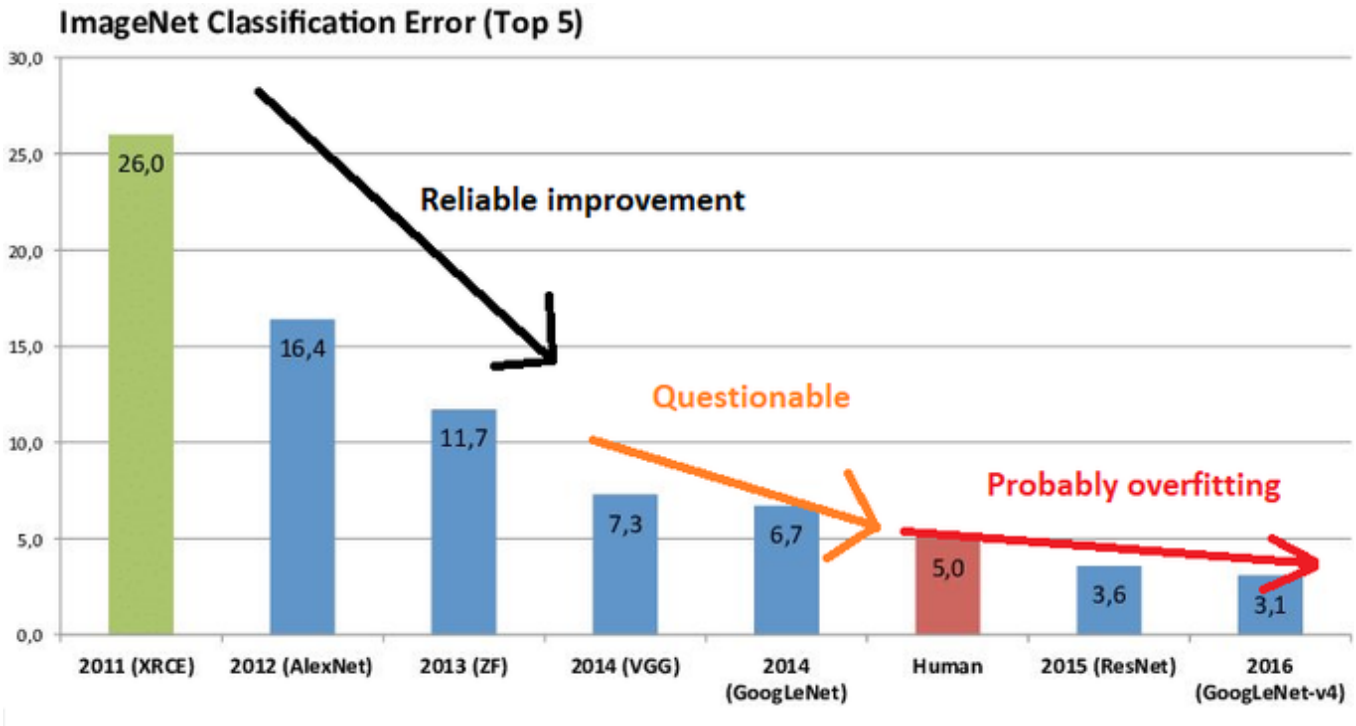
✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

Russakovsky et al. arXiv, 2014

Large Scale Visual Recognition Challenge (ILSVRC) - ImageNet

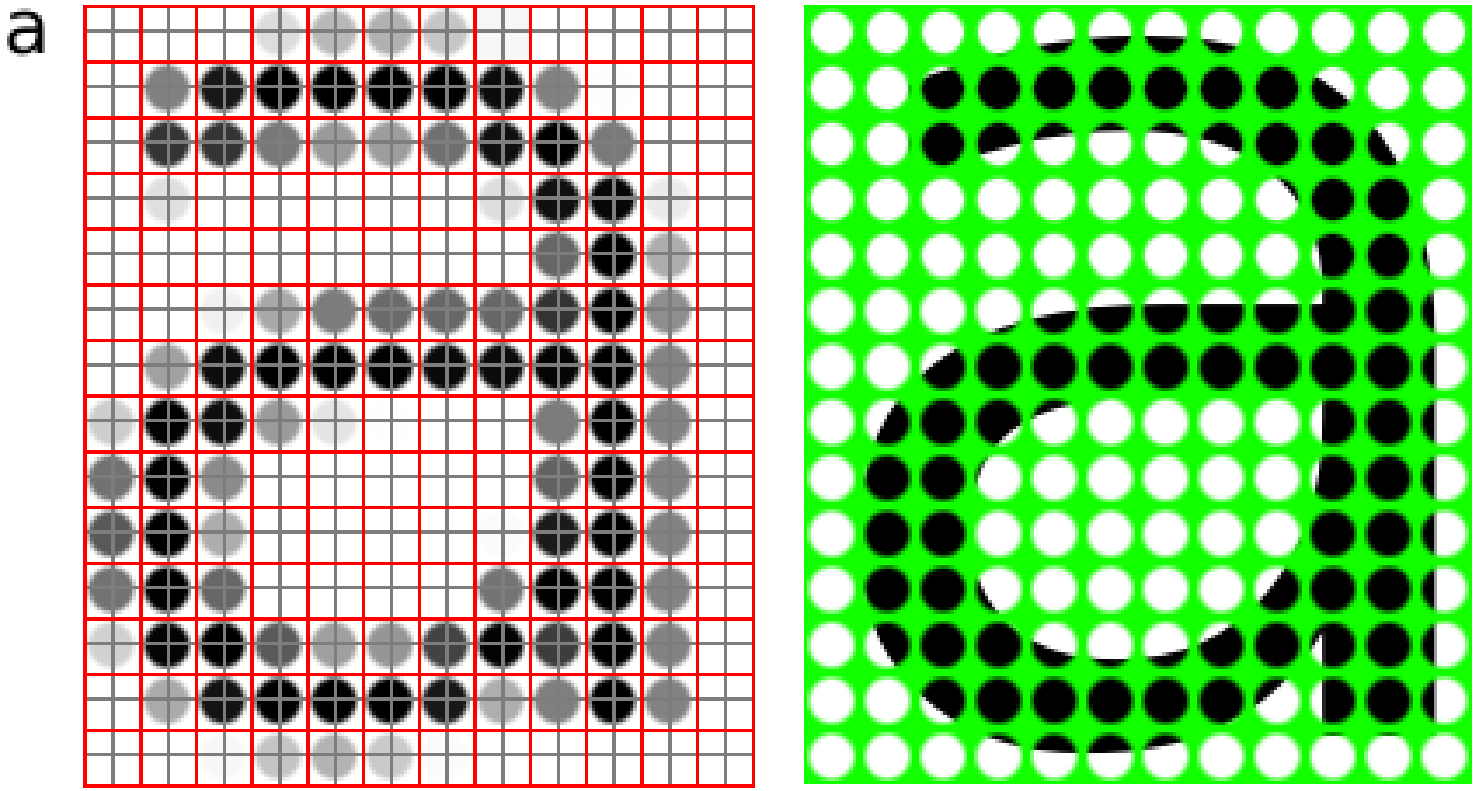


Next

1. Brief introdution to digital images
2. Examples of image processing/analysis
3. Why image processing/analysis is hard ?
4. Some traditional approaches
5. Deep learning based approaches

Brief introdution to digital images

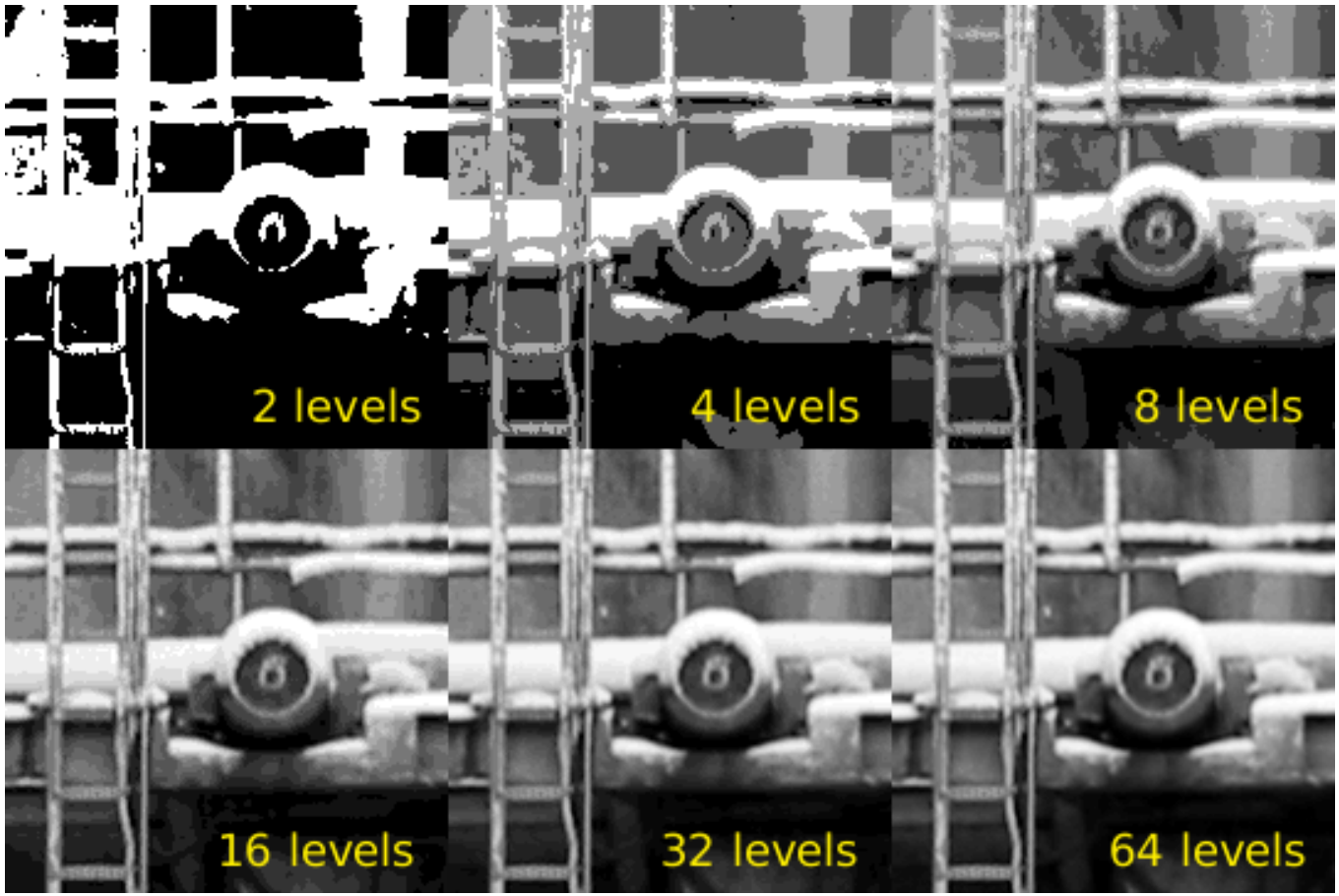
Digital images – spatial sampling



(http://pippin.gimp.org/image_processing)

Resolution: pixels per inch (ppi)

Digital images – quantization



(http://pippin.gimp.org/image_processing)

Number of *bits* per pixel $\longrightarrow$ gray levels

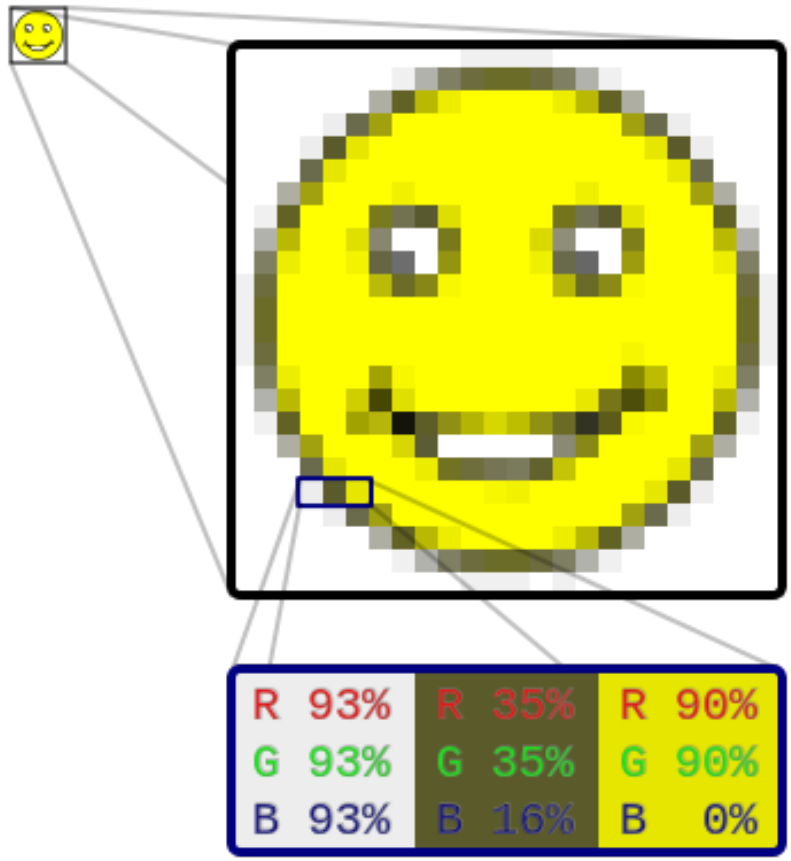
Standard: 8 *bits* $\longrightarrow$ 256 gray levels

Black=0 e white=255

Digital images – Color images

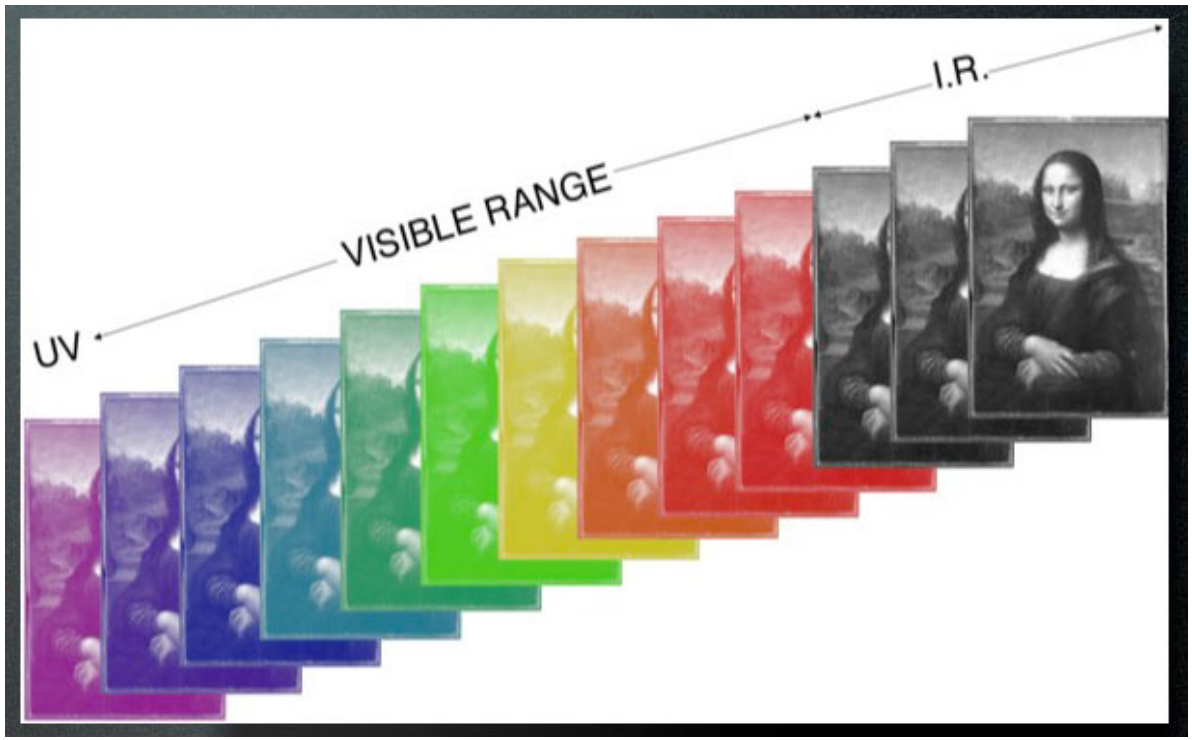
(https://www.photoshopessentials.com/essentials/rgb/)

Digital images – Color images



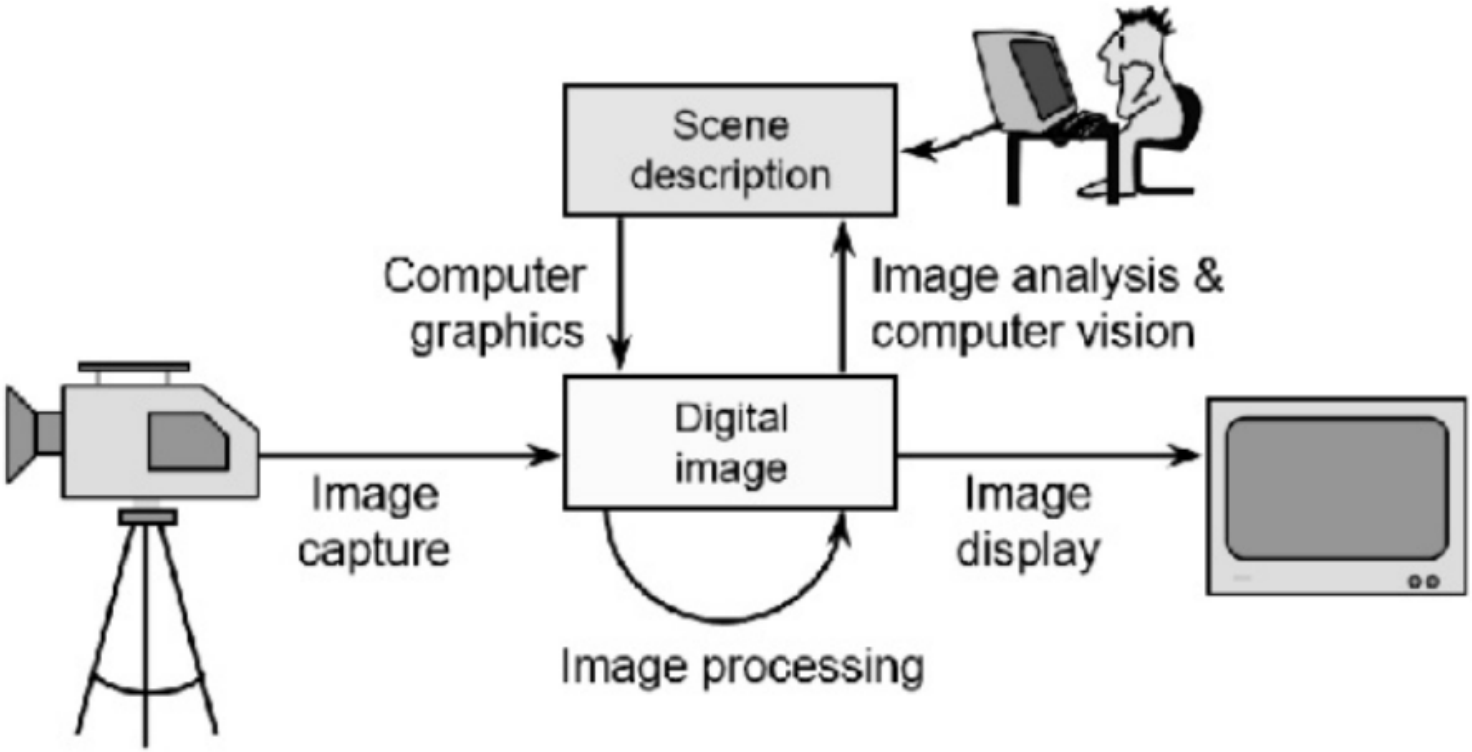
(en.wikipedia.org/wiki/Raster_graphics)

Digital images – Multispectral images



(http://www.lumiere-technology.com/Pages/Services/services3.htm)

Research fields related to images



(Source: slides of prof. Nidhal El Abbadi, University of Kufa)

Examples of image processing/analysis


339-KBJ
339-KBJ
3 3 9 K B J

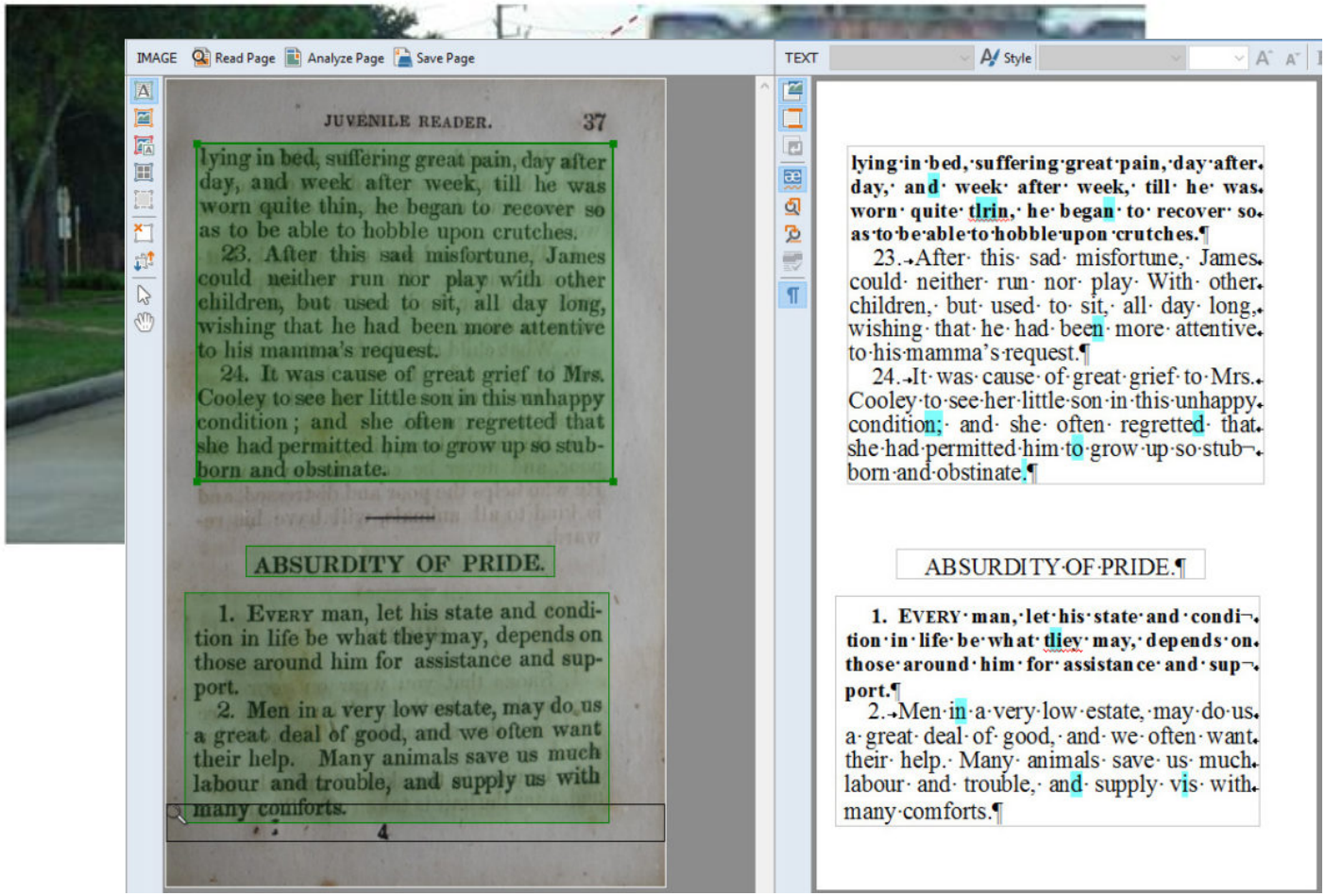


IMAGE
Read Page
Analyze Page
Save Page
TEXT
Style
JUVENILE READER. 37
lying in bed, suffering great pain, day after day, and week after week, till he was worn quite thin, he began to recover so as to be able to hobble upon crutches.
23. After this sad misfortune, James could neither run nor play with other children, but used to sit, all day long, wishing that he had been more attentive to his mamma's request.
24. It was cause of great grief to Mrs. Cooley to see her little son in this unhappy condition; and she often regretted that she had permitted him to grow up so stubborn and obstinate.
ABSURDITY OF PRIDE.
1. Every man, let his state and condition in life be what they may, depends on those around him for assistance and support.
2. Men in a very low estate, may do us a great deal of good, and we often want their help. Many animals save us much labour and trouble, and supply us with many comforts.
4
lying·in·bed,·suffering·great·pain,·day·after.
day,· and· week· after· week,· till· he· was.
worn· quite· tlrin,· he· began· to· recover· so.
as·to·be·able·to·hobble·upon·crutches.¶
23.→After· this· sad· misfortune,· James.
could· neither· run· nor· play· With· other.
children,· but· used· to· sit,· all· day· long,.
wishing· that· he· had· been· more· attentive.
to·his·mamma's·request.¶
24.→It·was·cause·of·great·grief·to·Mrs.
Cooley·to·see·her·little·son·in·this·unhappy.
condition;· and· she· often· regretted· that.
she·had·permitted·him·to·grow·up·so·stub¬.
born·and·obstinate.¶
ABSURDITY·OF·PRIDE.¶
1. EVERY·man,·let·his·state·and·condi¬.
tion·in·life·be·what·tliey·may,·depends·on.
those·around·him·for·assistance·and·sup¬.
port.¶
2.→Men·in·a·very·low·estate,·may·do·us.
a·great·deal·of·good,·and·we·often·want.
their·help.·Many·animals·save·us·much.
labour·and·trouble,·and·supply·vis·with.
many·comforts.¶


IMAGE
Read Page
Analyze Page
Save Page
JUVENILE READER. 37
lying in bed, suffering great pain, day after day, and week after week, till he was worn quite thin, he began to recover so as to be able to hobble upon crutches.
23. After this sad misfortune, James could neither run nor play with other children, but used to sit, all day long,
TEXT
Style
lying·in·bed,·suffering·great·pain,·day·after· day,· and· week· after· week,· till· he· was· worn· quite· thin,· he· began· to· recover· so· as·to·be·able·to·hobble·upon·crutches.¶
23.→After· this· sad· misfortune,· James· could· neither· run· nor· play· With· other· children,· but· used· to· sit,· all· day· long,·
scan fingerprint
to unlock

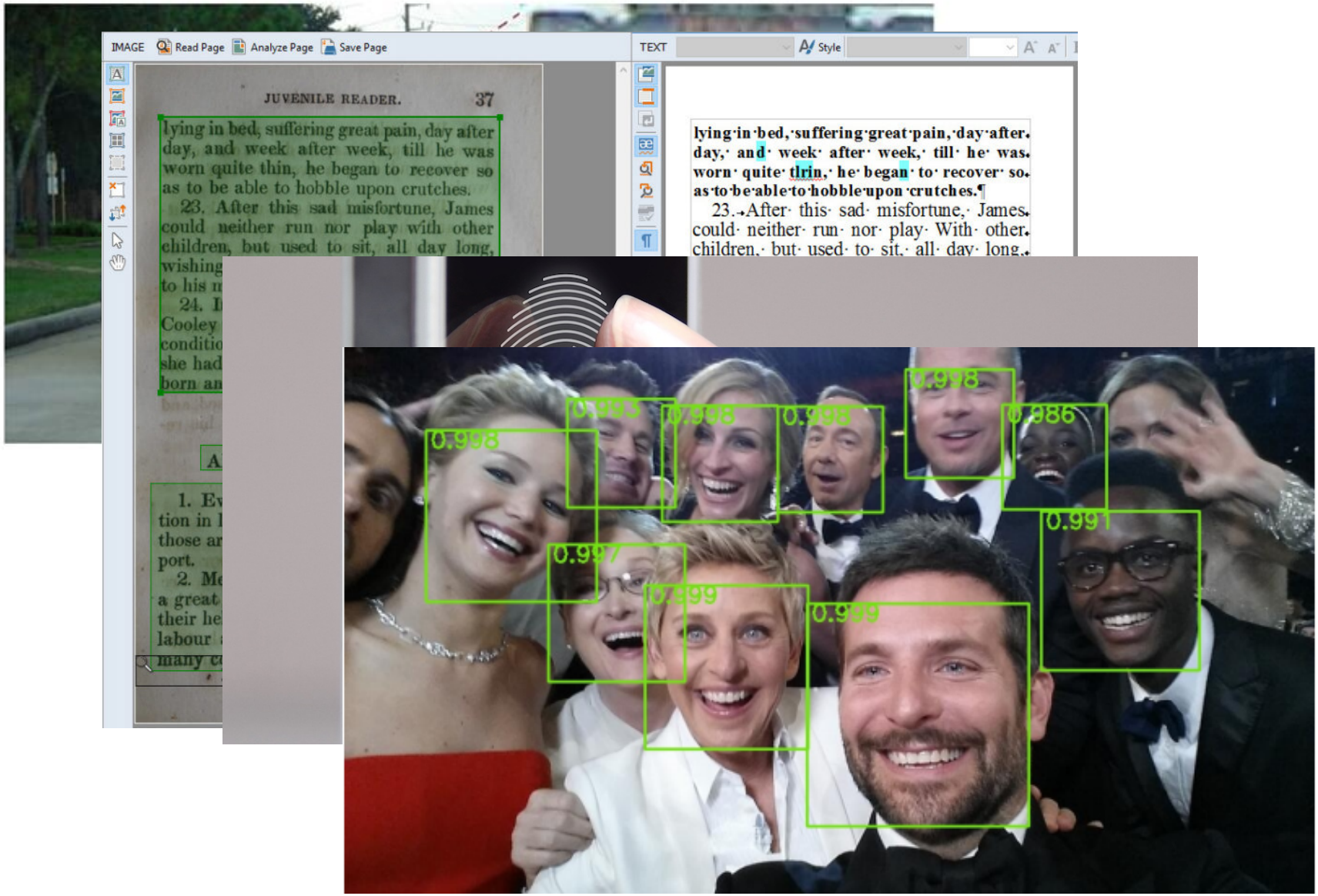
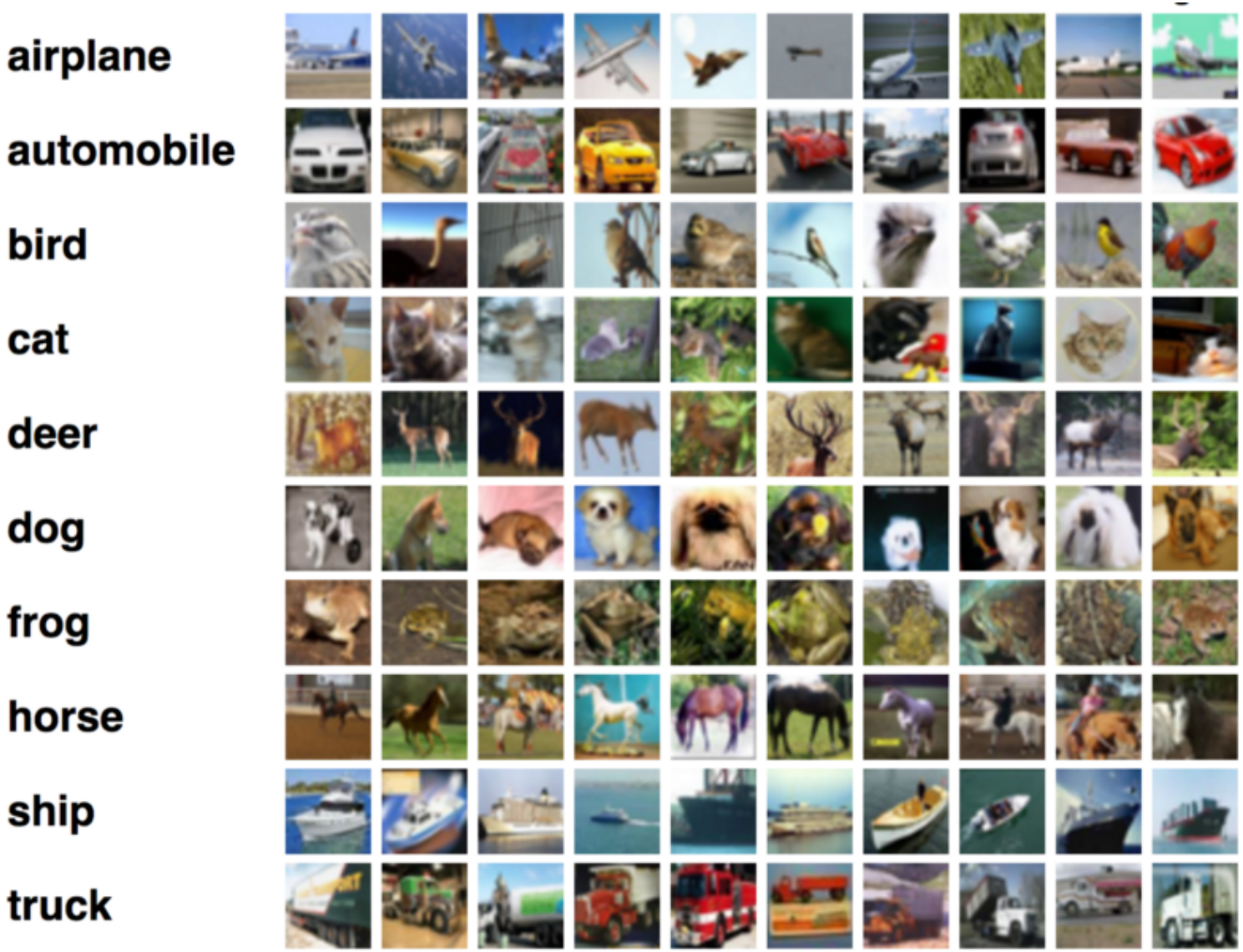
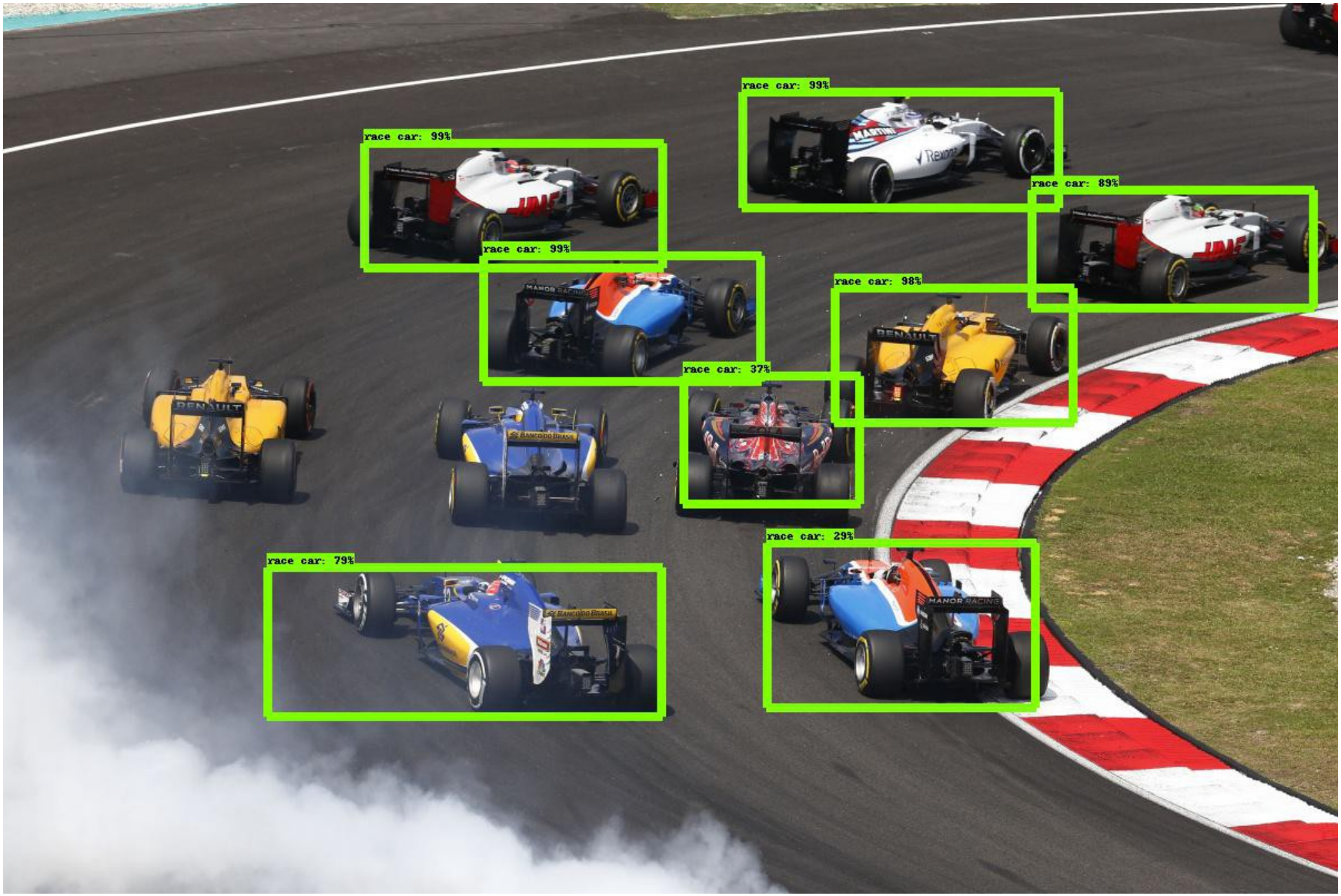


IMAGE
Read Page
Analyze Page
Save Page
JUVENILE READER. 37
lying in bed, suffering great pain, day after day, and week after week, till he was worn quite thin, he began to recover so as to be able to hobble upon crutches.
23. After this sad misfortune, James could neither run nor play with other children, but used to sit, all day long,
TEXT
Style
lying·in·bed,·suffering·great·pain,·day·after·day,· and· week· after· week,· till· he· was· worn· quite· tlrin,· he· began· to· recover· so· as·to·be·able·to·hobble·upon·crutches.¶
23.→After· this· sad· misfortune,· James· could· neither· run· nor· play· With· other· children,· but· used· to· sit,· all· day· long,
0.998
0.998
0.998
0.998
0.986
0.997
0.999
0.999
0.999

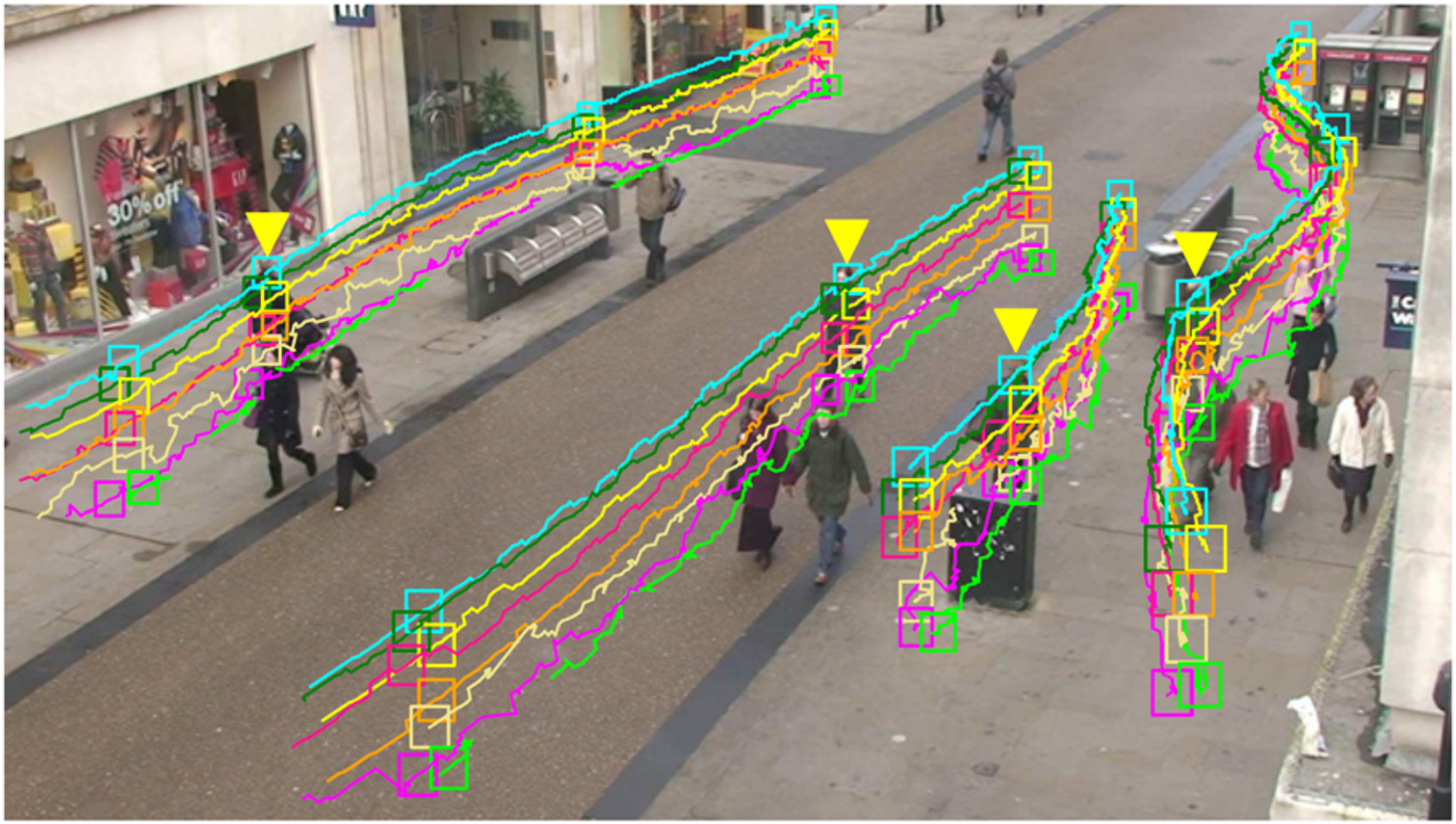
Image classification



Object detection



Object tracking in video



Semantic segmentation

Scene description

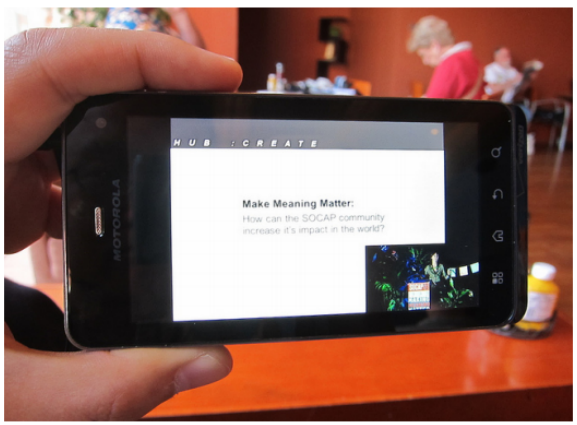
A female tennis player in action on the court.

A group of young men playing a game of soccer

A man riding a wave on top of a surfboard.

A baseball game in progress with the batter up to plate.

A brown bear standing on top of a lush green field.



A person holding a cell phone in their hand.

Why image processing/analysis is hard ?

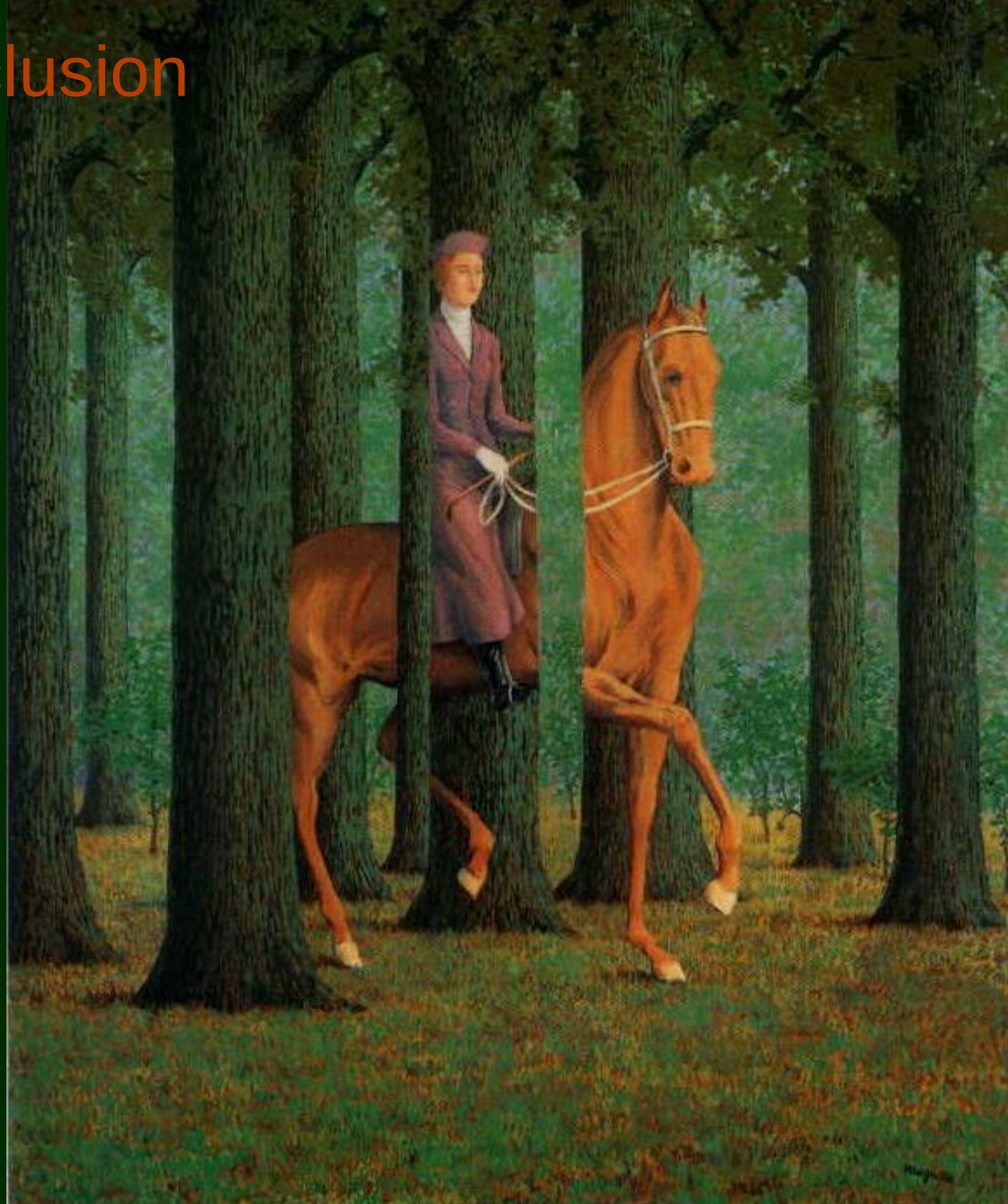
Challenges 1: view point variation

Michelangelo 1475-1564

Challenges 2: illumination

slide credit: S. Ullman

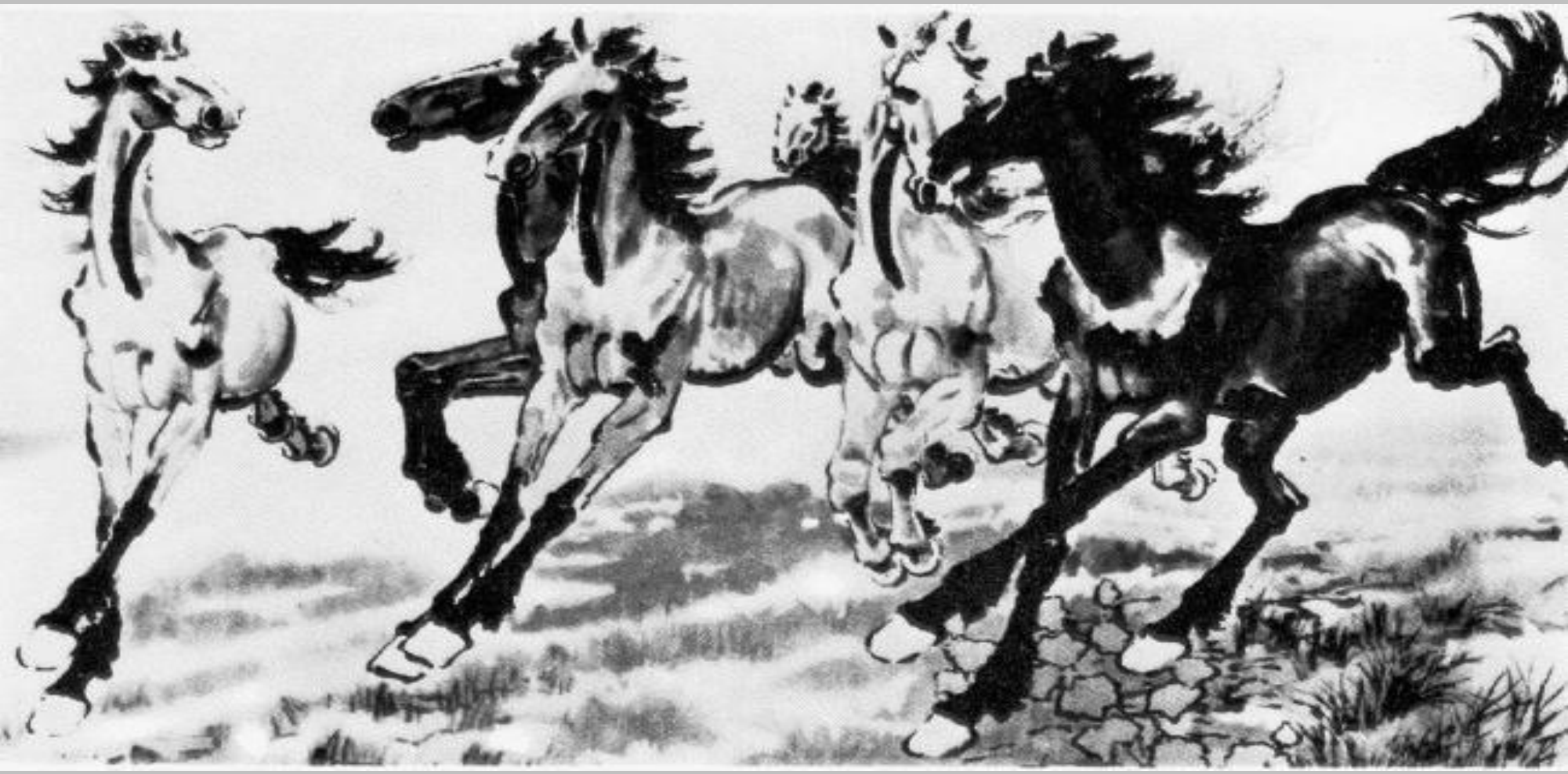
Challenges 3: occlusion



Magritte, 1957

Challenges 4: scale

Challenges 5: deformation



Xu, Beihong 1943

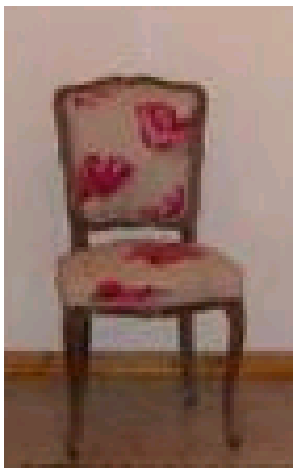
Challenges 6: background clutter

Klimt, 1913

Within-class variations

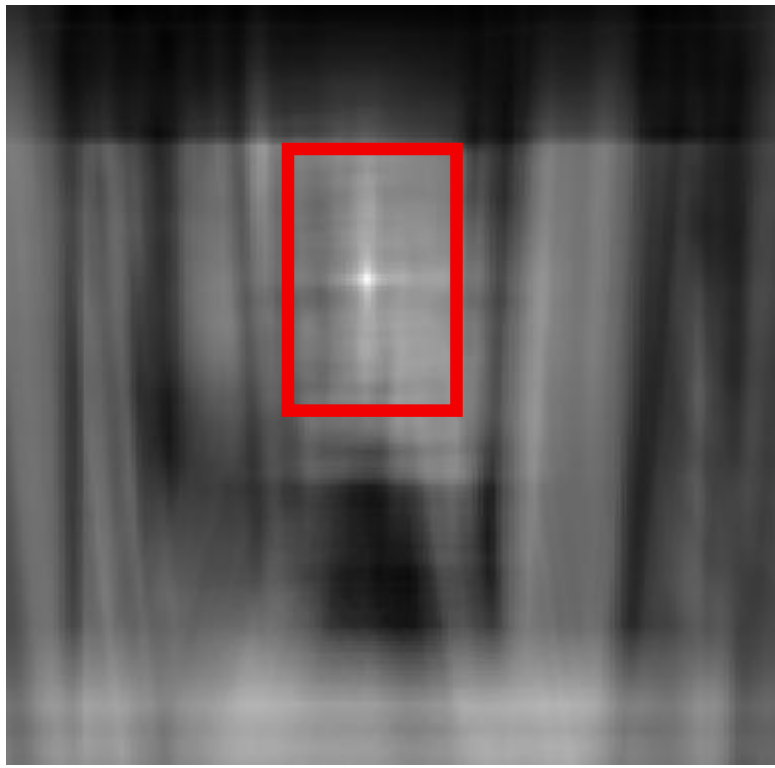
Object recognition
Is it really so hard?

This is a chair

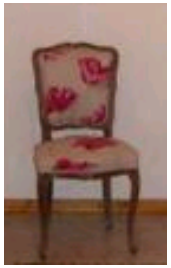


Find the chair in this image

Output of normalized correlation

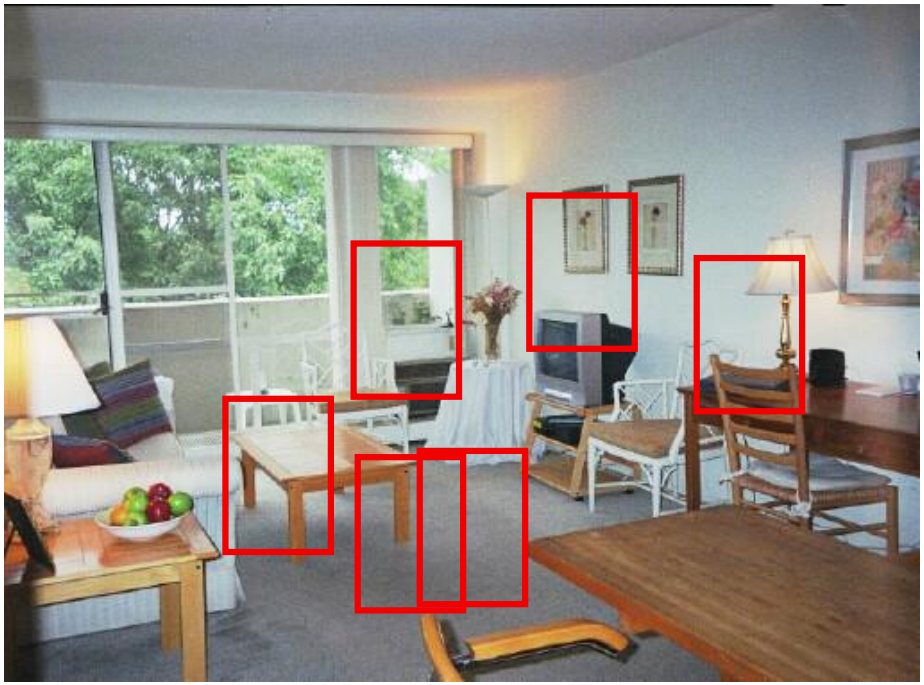


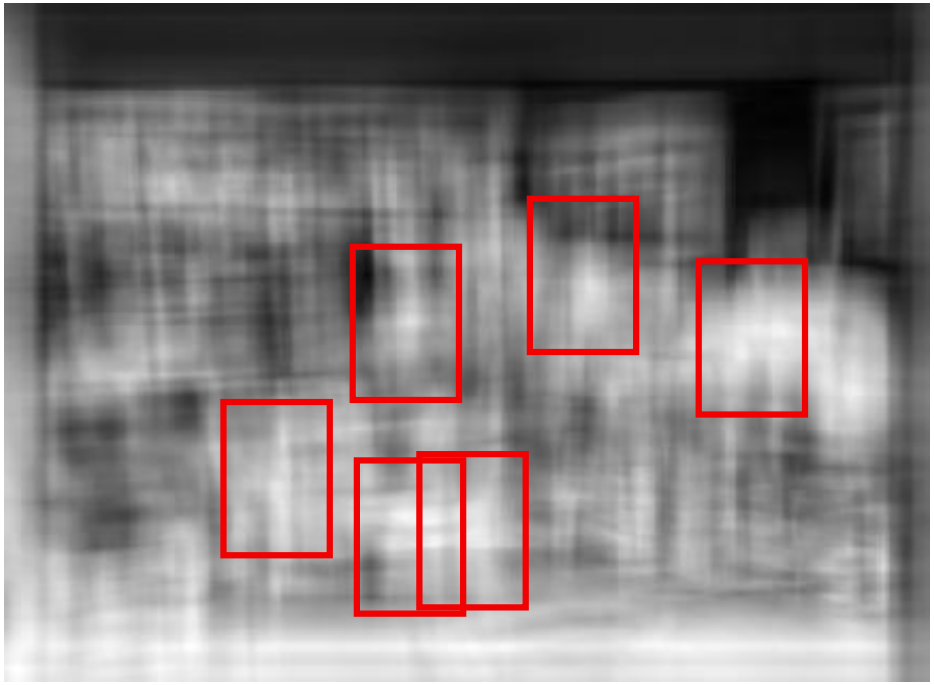
Slide: A. Torralba



Object recognition
Is it really so hard?

Find the chair in this image





Pretty much garbage

Simple template matching is not going to make it

A “popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts.” Nivatia & Binford, 1977.

Slide: A. Torralba

Examples of traditional approaches

- Cascade of classifiers

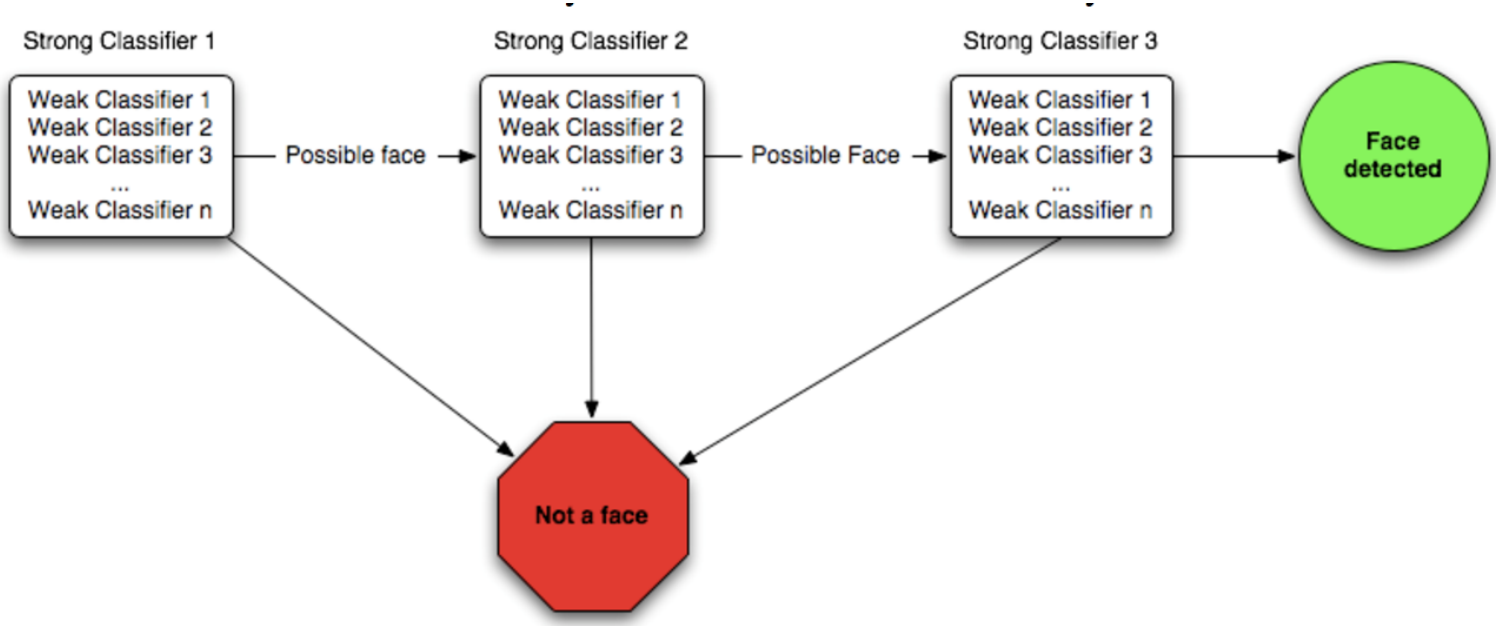
- SIFT

- Bag-of-visual-words

Cascade of classifiers

Rapid object detection using a boosted cascade of simple features, P. Viola and M. Jones, CVPR 2001)

Adaboost and cascade of classifiers



(https://www.quora.com/How-can-I-understand-Haar-like-feature-for-face-detection)

Adaboost algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- For $t = 1, \ldots, T$:
  1. Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$
  2. Select the best weak classifier with respect to the weighted error

     $$\epsilon_t = \min_{f,p,\theta} \sum_i w_i \, |h(x_i, f, p, \theta) - y_i| \, .$$

     See Section 3.1 for a discussion of an efficient implementation.
  3. Define $h_t(x) = h(x, f_t, p_t, \theta_t)$ where f_t, p_t, and θ_t are the minimizers of ϵ_t.
  4. Update the weights:

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

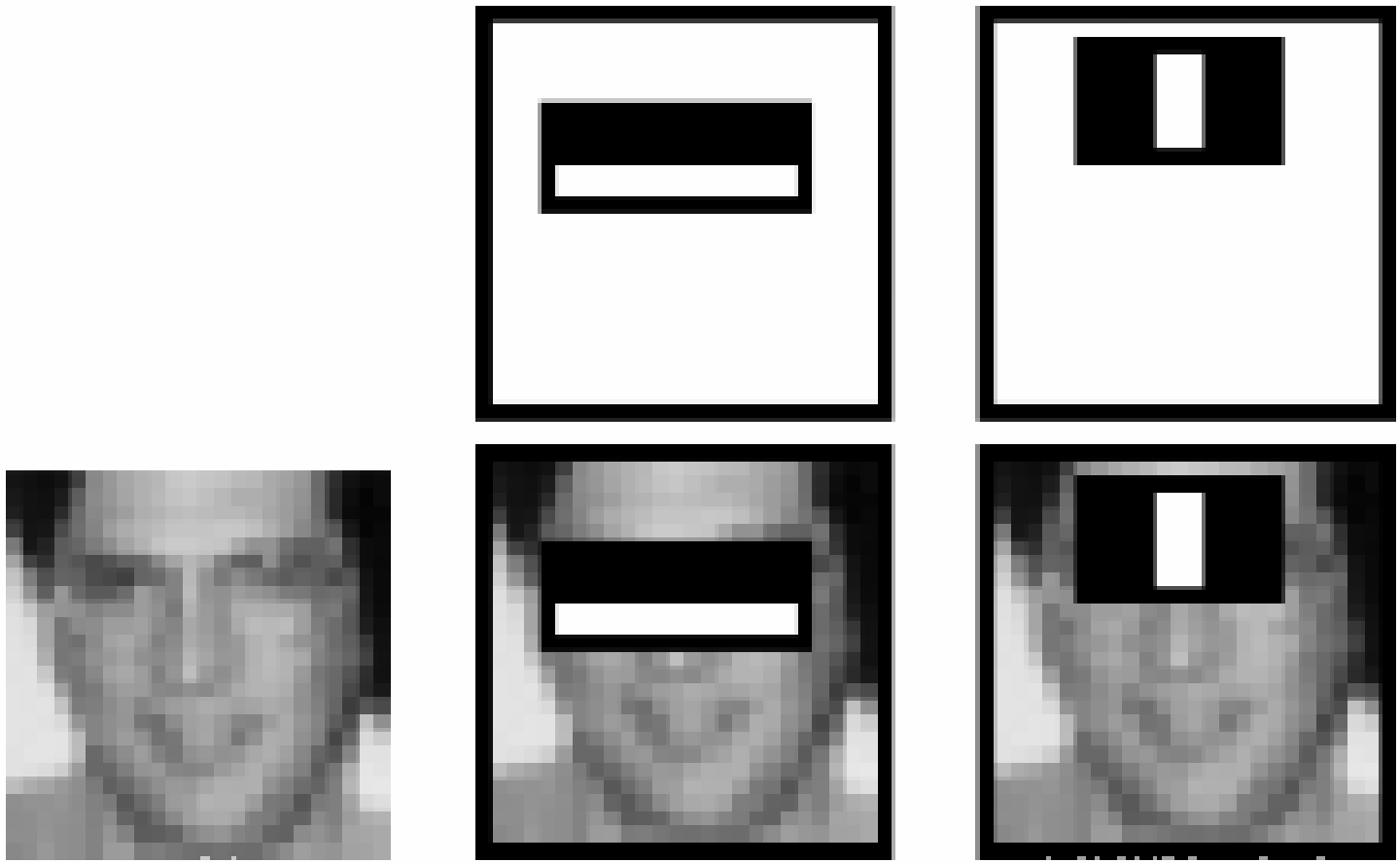
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

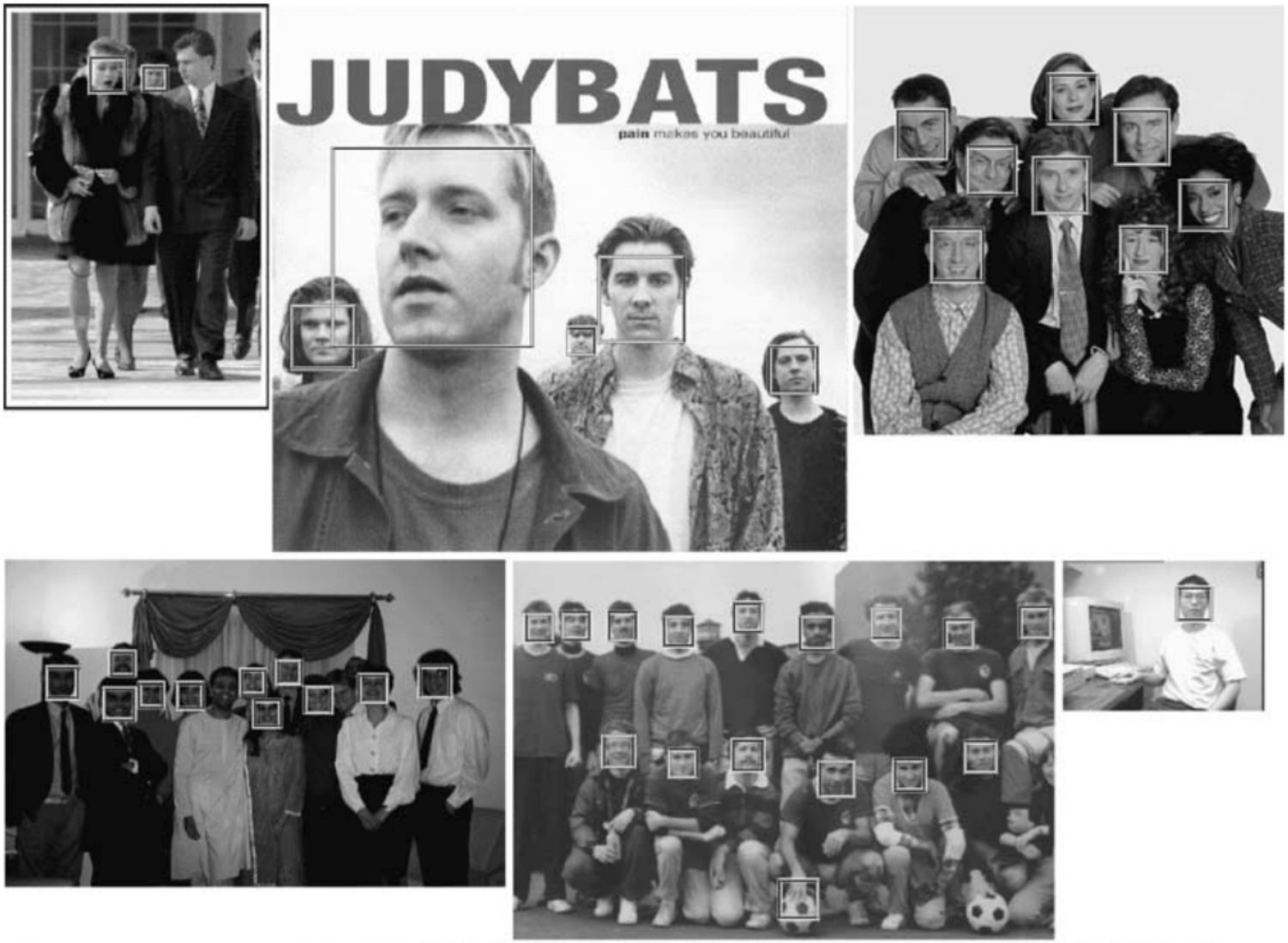
where $\alpha_t = \log \frac{1}{\beta_t}$

Main Haar like features



(https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html)

Results



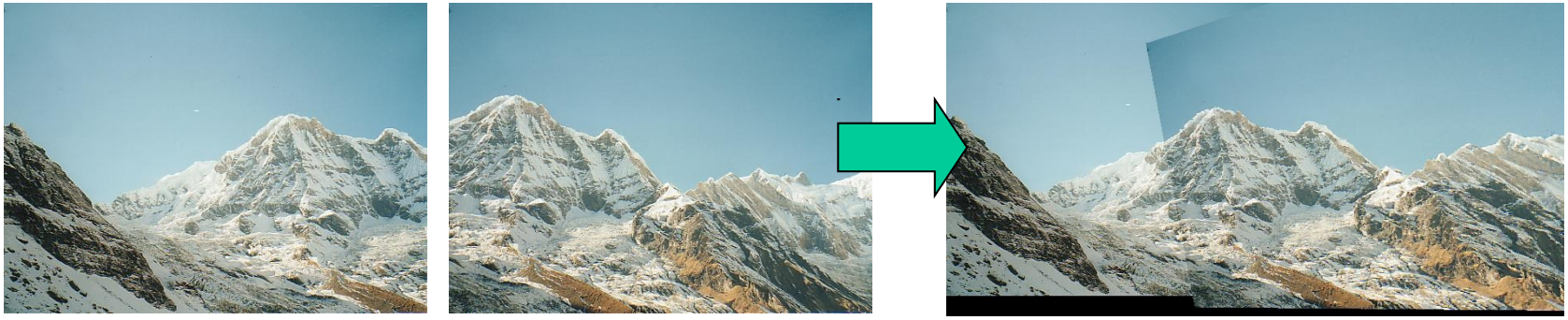
SIFT (Scale-invariant feature transform)

Registro de imagens, detecção de objetos via casamento de keypoints

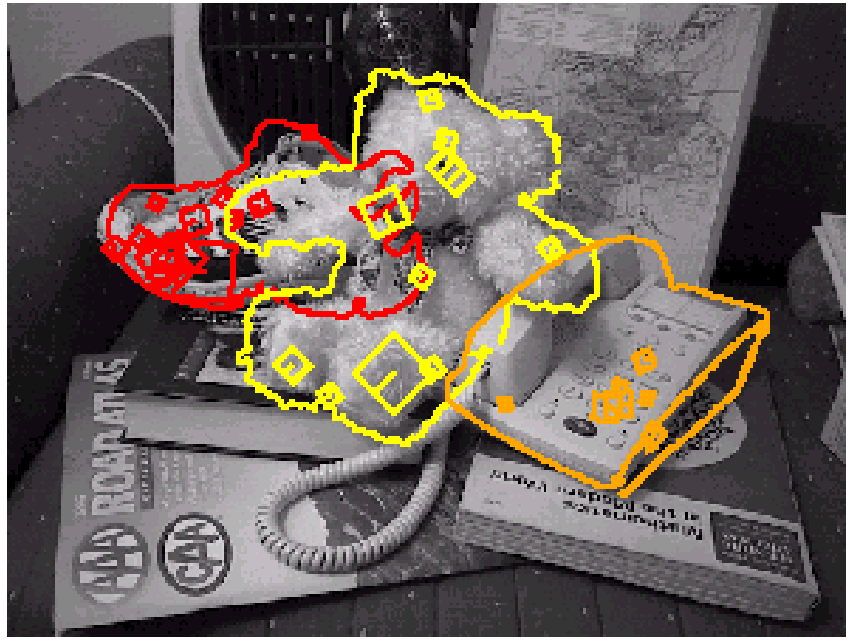
Lowe, David G. (1999). *Object recognition from local scale-invariant features*. Proceedings of the International Conference on Computer Vision 2, pp. 1150–1157

Lowe, D. G. (2004). *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, 60, 2, pp. 91-110

Image alignment: Applications



Panorama stitching

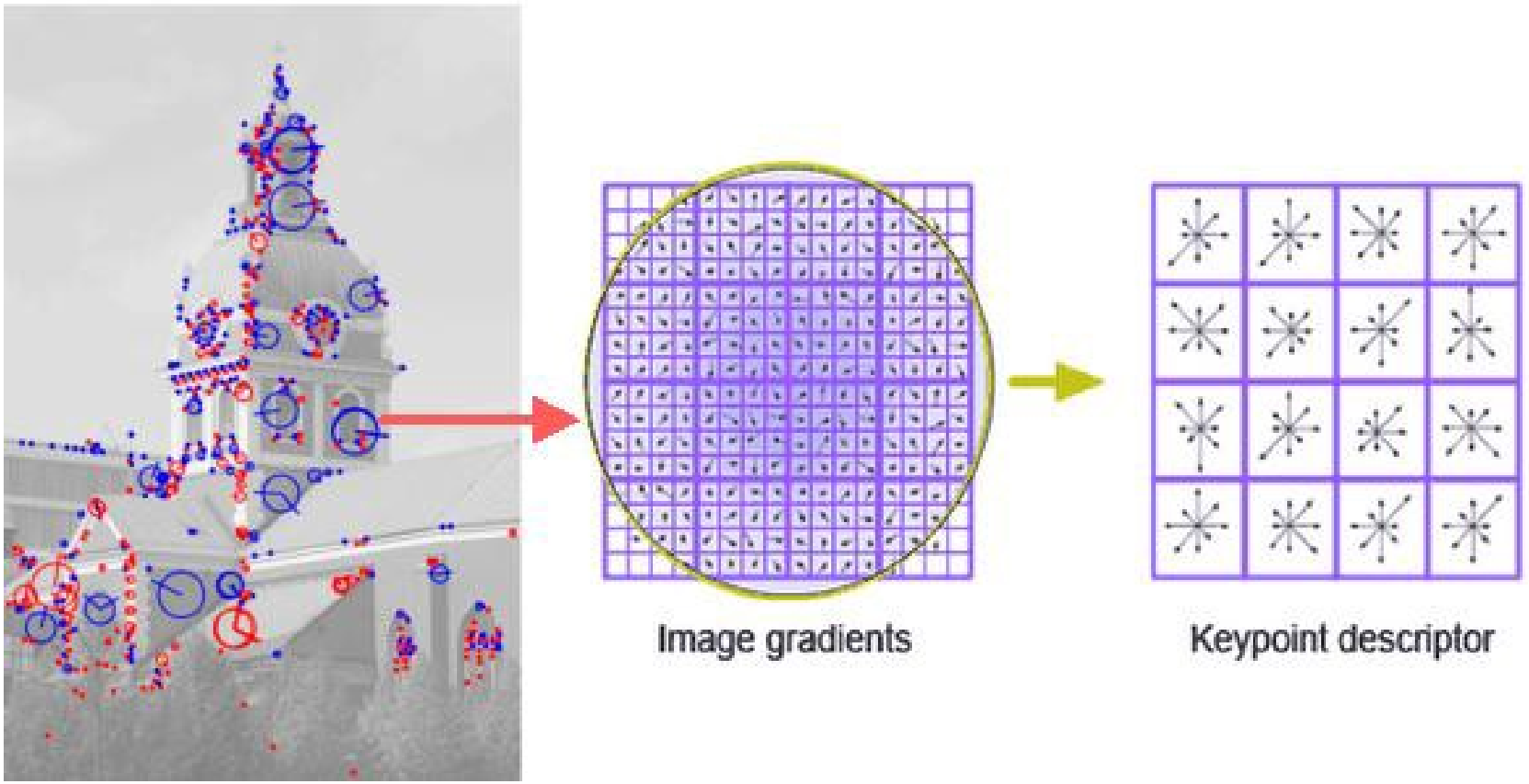


Recognition of object instances

Passos do SIFT

1. **Detecção de key-points** (cantos, pontos extremos)
2. **Descrição dos key-points** (conjunto de features locais, com invariância à rotação, iluminação, etc)
3. **Correspondência entre os key-points** da query-image e da target image
   Levando em conta similaridade de features
4. **Filtragem** de correspondências falsas
5. Estimar dos parâmetros de uma **transformação afim** ou **projetiva** que melhor se ajuste à correspondência

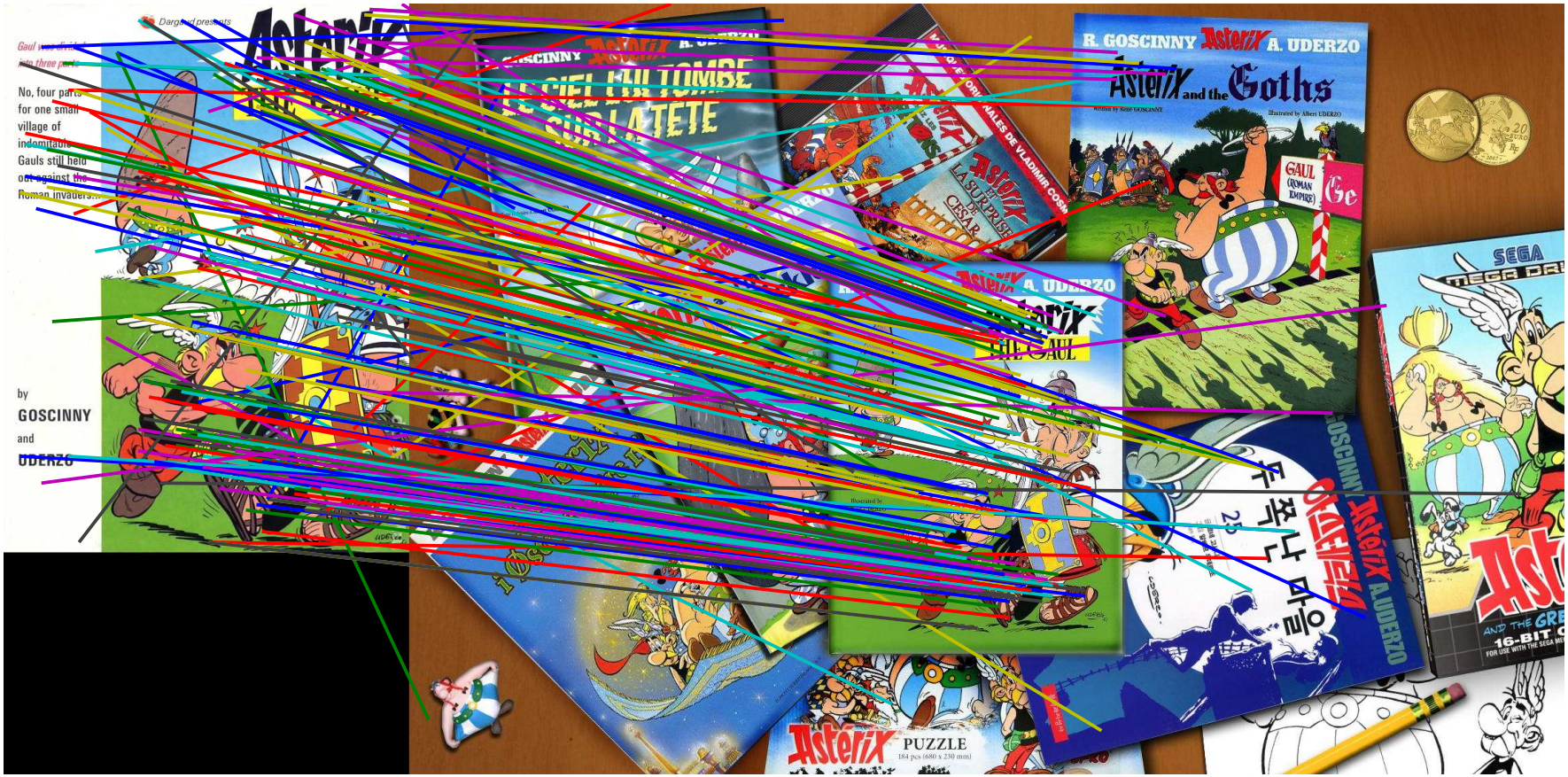
Feature description



(medium.com/machine-learning-world/

feature-extraction-and-similar-image-search-with-opencv-for-newbies)

197 tentative matches



99 (50.25%) inliner matches out of 197

Mosaic

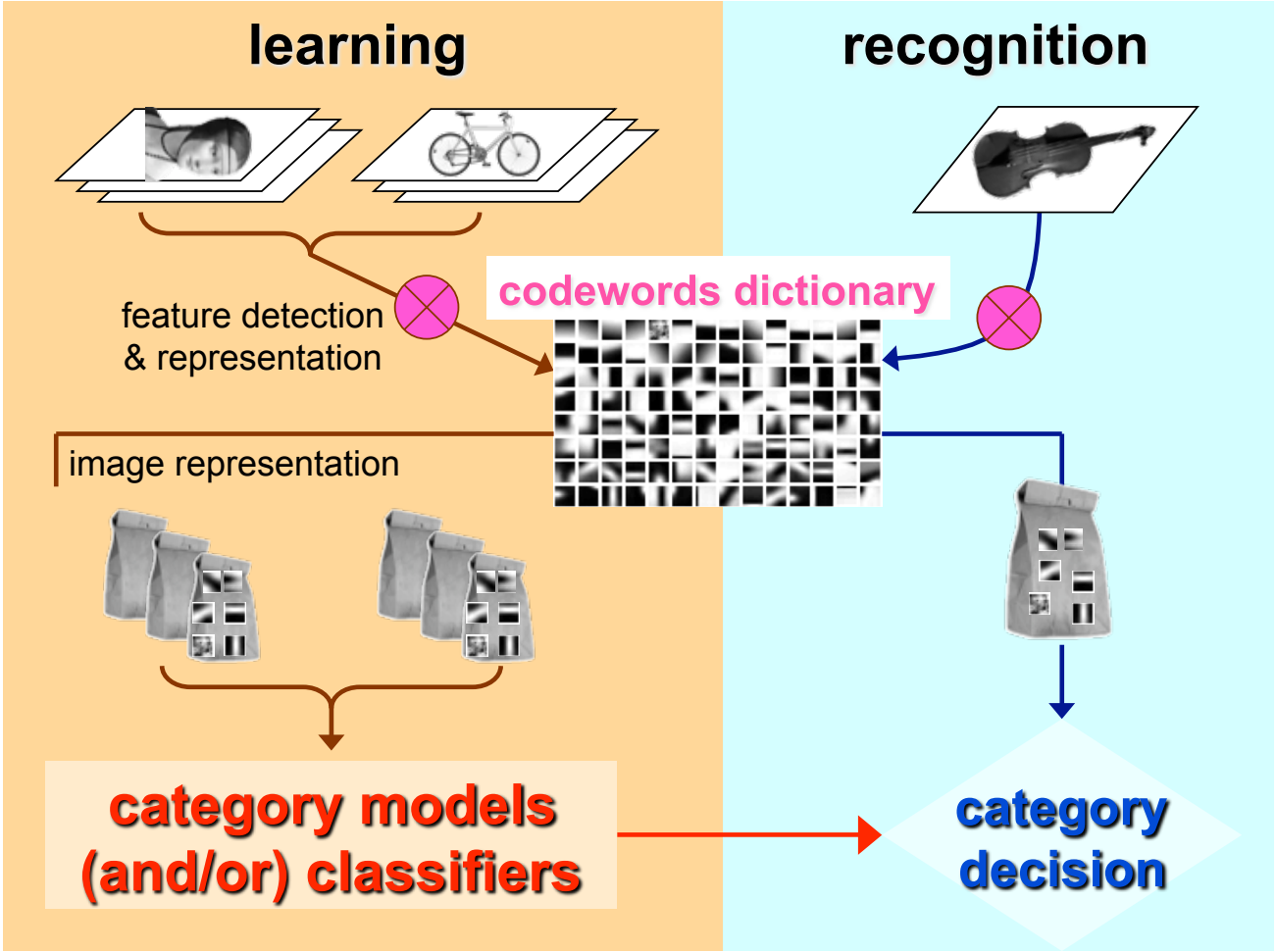

R. GOSCINNY Astérix A. UDERZO
LE CIEL LUI TOMBE SUR LA TÊTE
MUSIQUES ORIGINALES DE VLADIMIR COSMA
Astérix chez les Bretons
Astérix et la surprise de César
R. GOSCINNY Asterix A. UDERZO
Asterix and the Goths
GAUL (ROMAN EMPIRE)
R. GOSCINNY Asterix A. UDERZO
Asterix and the Golden Sickle
R. GOSCINNY Asterix A. UDERZO
Asterix THE GAUL
by GOSCINNY and UDERZO
R. GOSCINNY Asterix A. UDERZO
Asterix i Østens fagre riker
두 쪽 난 마을
25
Astérix A. UDERZO
SEGA MEGA DRIVE
16-BIT
Astérix PUZZLE

Modelo Bag of visual words (BOW)

Classificação de imagens

Image retrieval

Bag-of-Words Model: Overview



Bag of keypoints (Csurka, 2004)

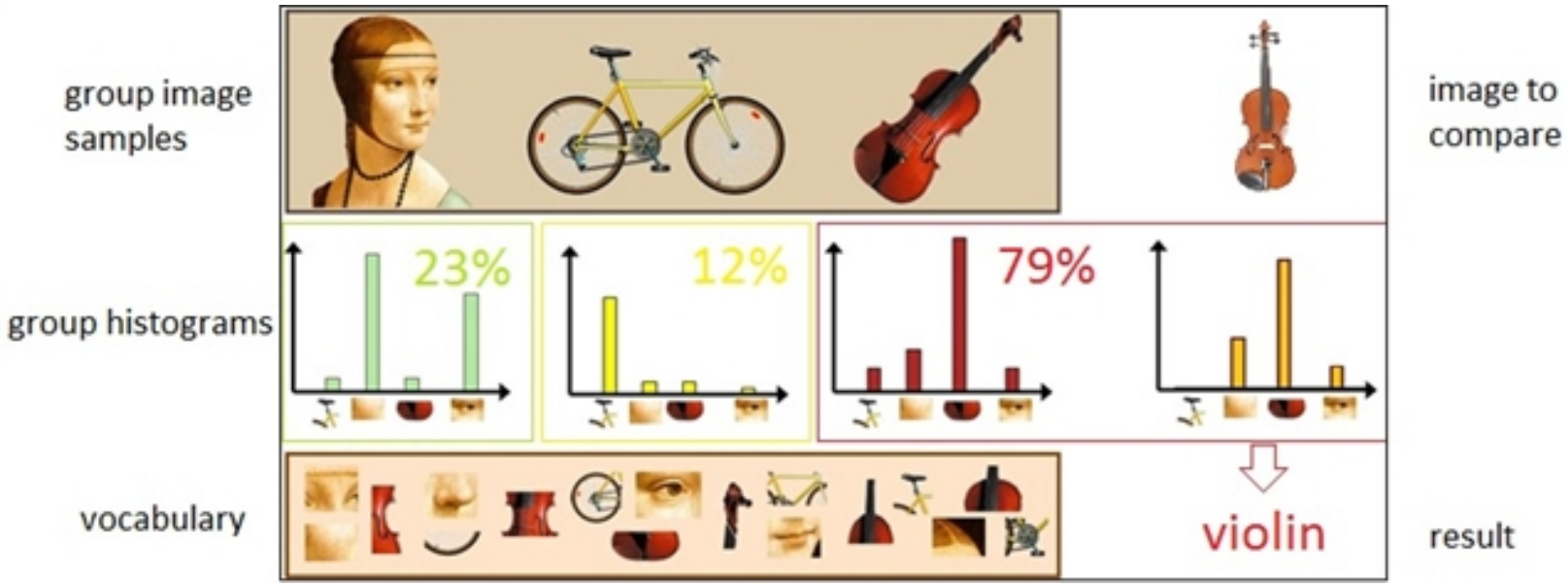
Método para classificação de imagens
(qual objeto está presente na imagem?)

1. **Criar um dicionário** (padrões visuais) a partir do conjunto de imagens de treinamento

2. **Treinamento**: calcular a representação *bag-of-words* das imagens e treinar classificadores a partir dessas bags (*bag-of-words* é o conjunto de *features*)

3. **Classificação**: calcular a representação *bag-of-words* e aplicar o classificador

Modelo BOW – classificação



(http://vgg.fiit.stuba.sk/wp-uploads/2015/02/BOW.jpg)

Finally

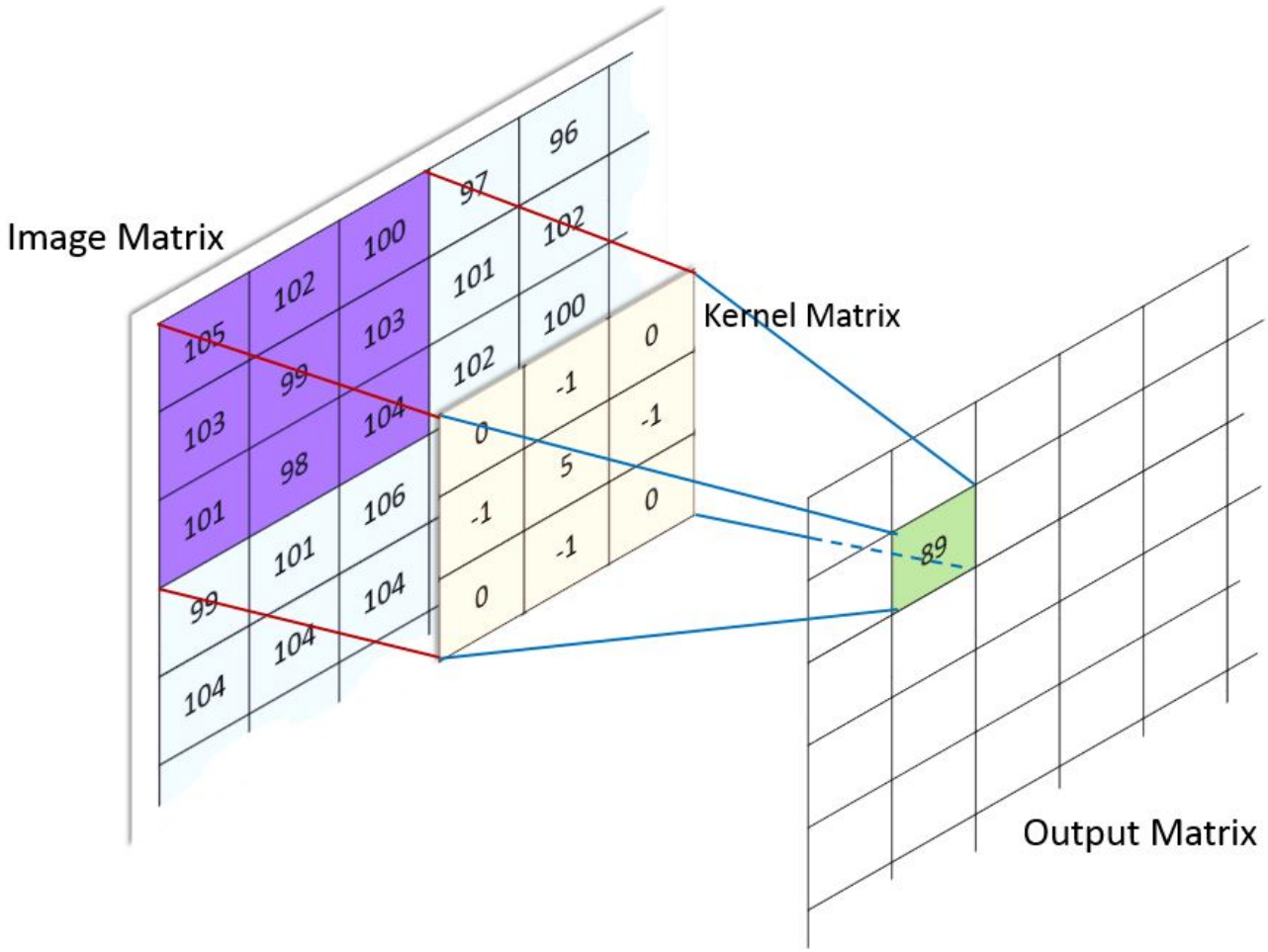
Deep convolutional networks

Convolution (or, more precisely, cross-correlation)

A basic operation used in image transformations

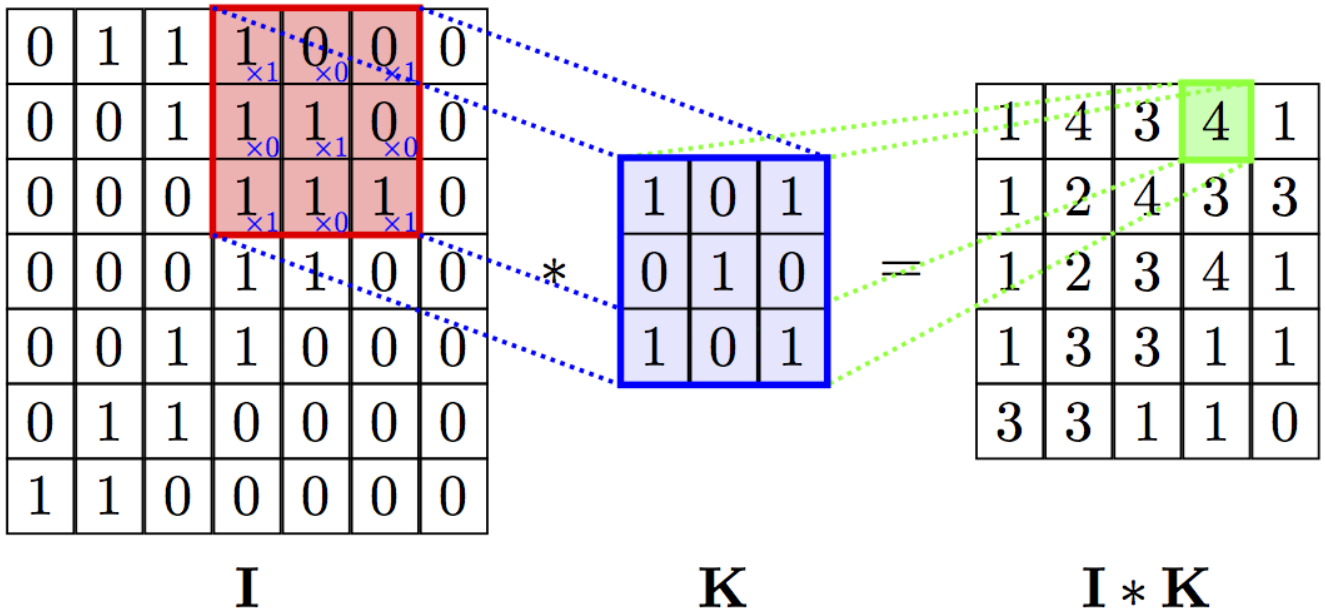
Convolution

Many image processing transformations are based on the convolution operator:



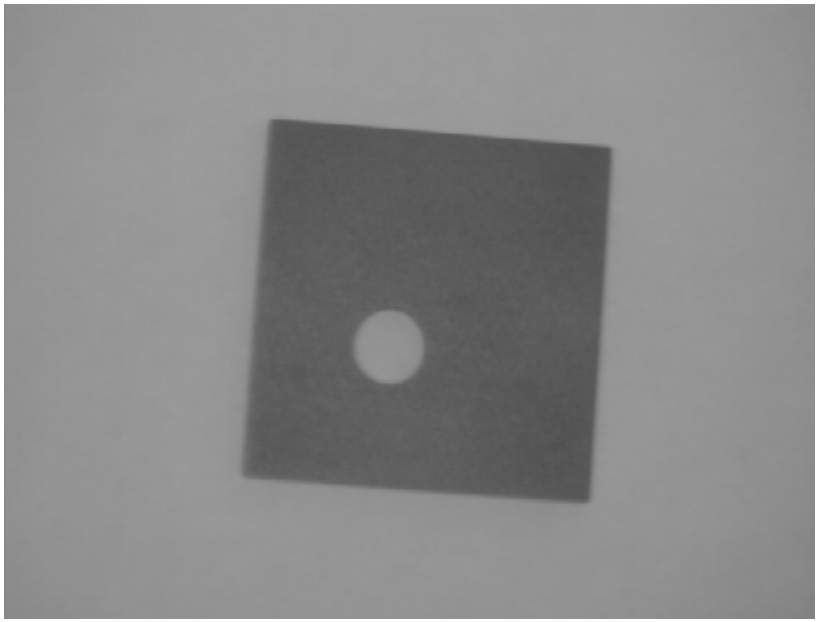
(http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html)

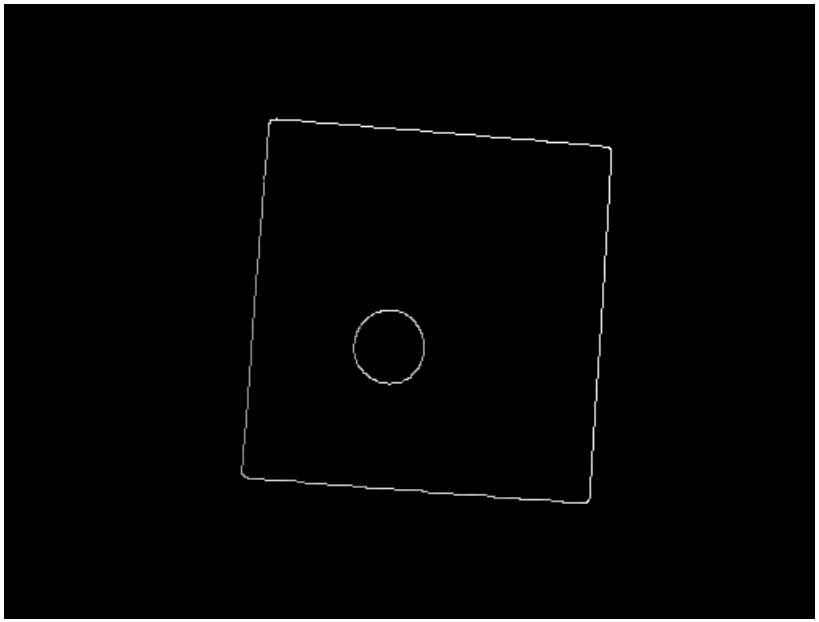
Convolution – a second diagram



(https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution)

Examples of convolution kernels – edge detection

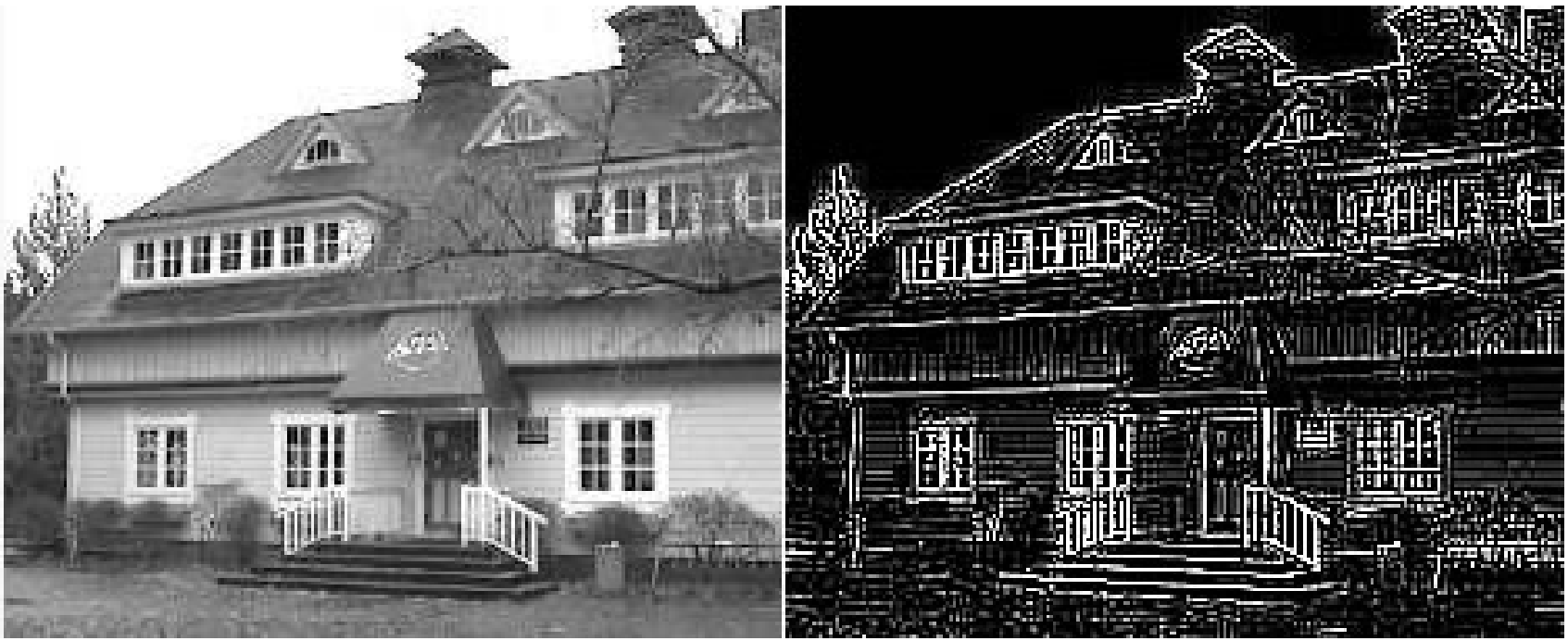




-1	-1	-1
-1	8	-1
-1	-1	-1

(http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm)

Examples of convolution kernels – edge detection



(http://aishack.in/tutorials/image-convolution-examples/)

Smoothing

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(http://aishack.in/tutorials/image-convolution-examples/)

CNN (convolutional neural networks): a neural network such that

- convolution operation is part of the network

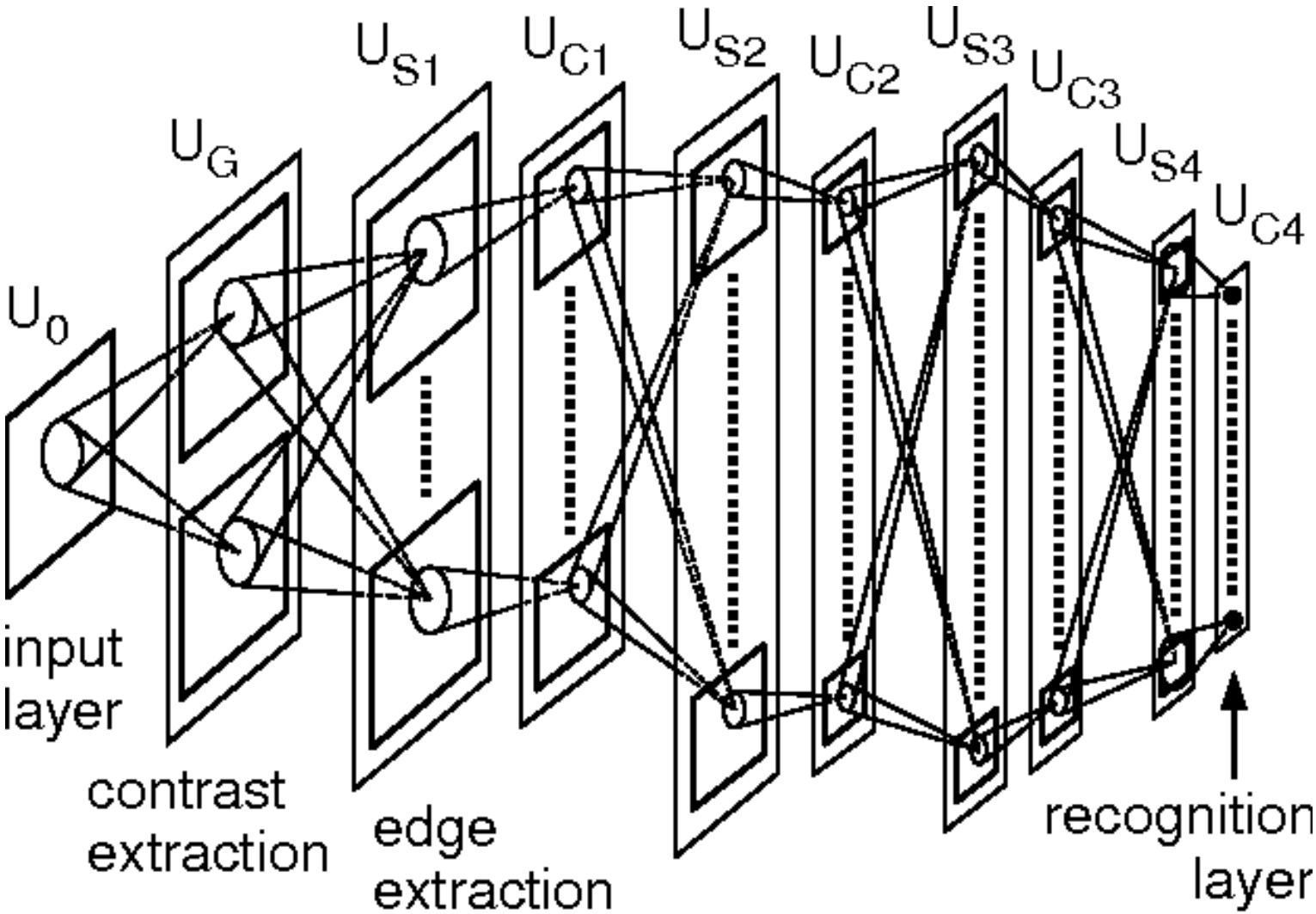
- kernels (masks) of the convolution are learned from data

Use of convolution in neural nets is not so recent:

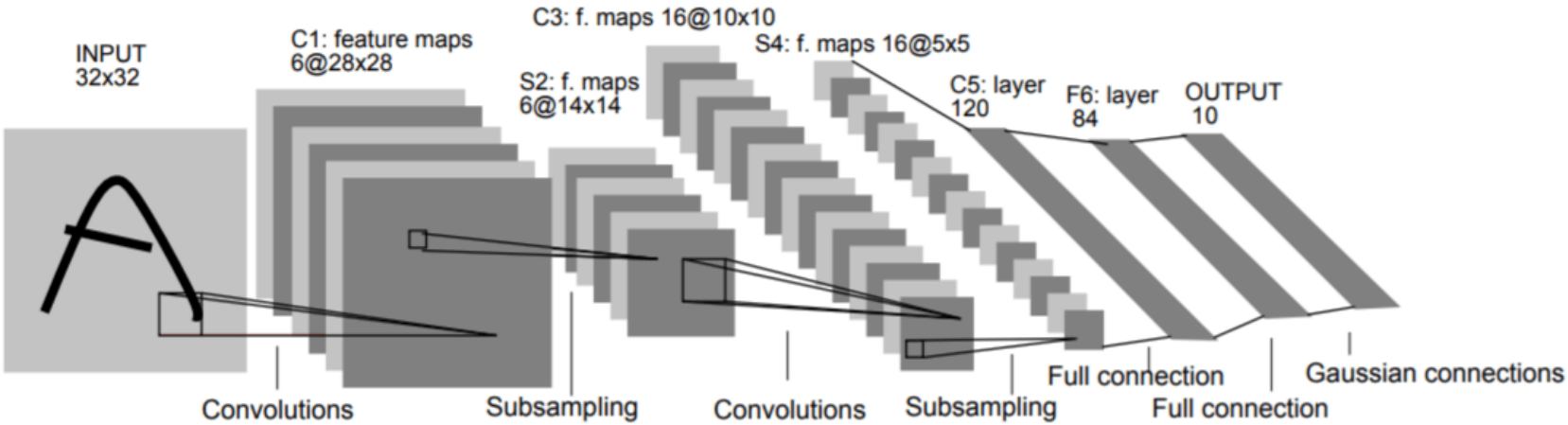
- **Neocognitron** – Fukushima, 1990
  Detailed explanation: https://www.kiv.zcu.cz/studies/predmety/uir/NS/Neocognitron/en/index.html

- **LeNet** - Yan Lecun, 1998, used with MNIST
  http://yann.lecun.com/exdb/lenet/index.html

Neocognitron – Fukushima, 1990



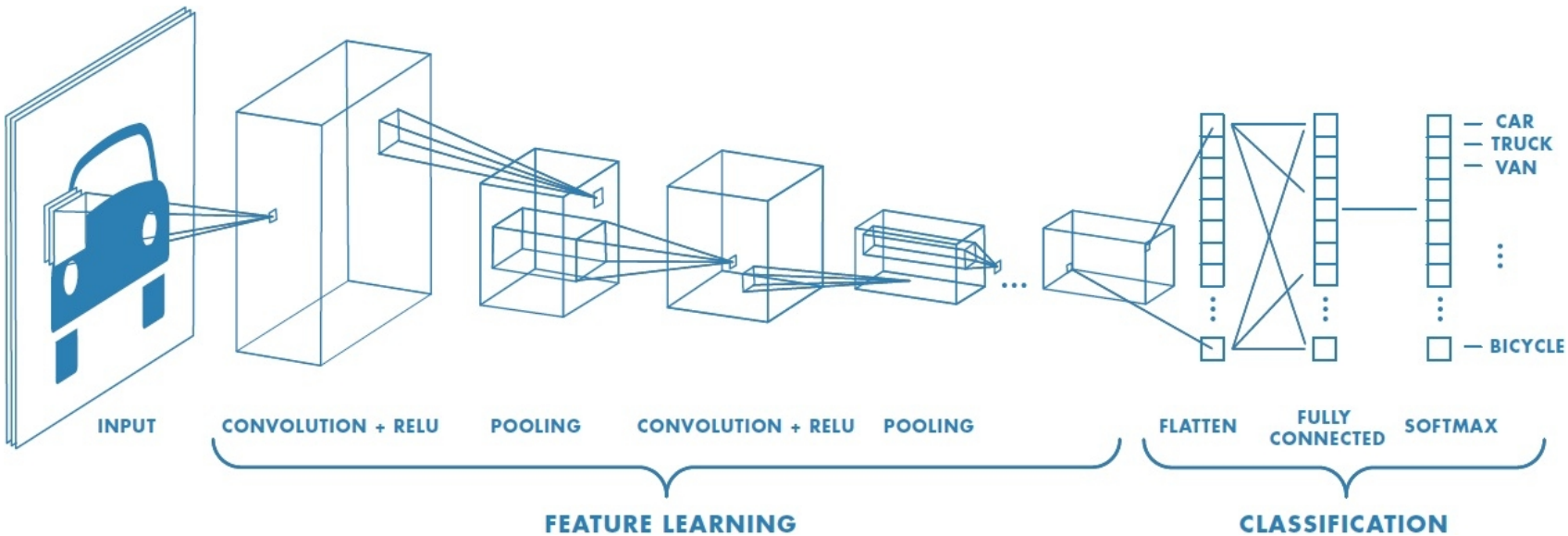
LeNet5 – Yan Lecun, 1998



This was applied on MNIST

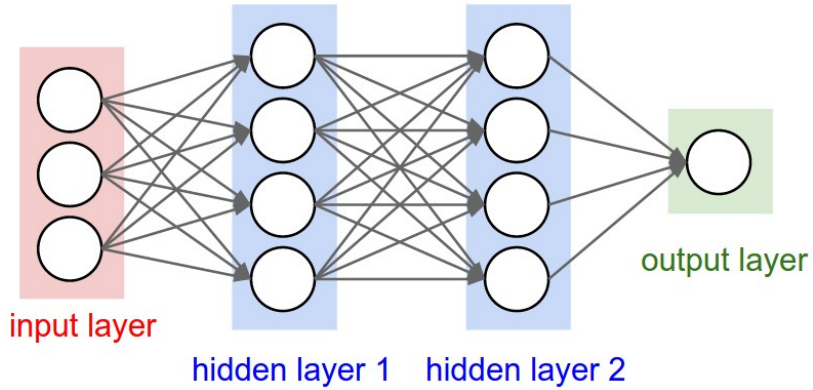
Basics of CNN

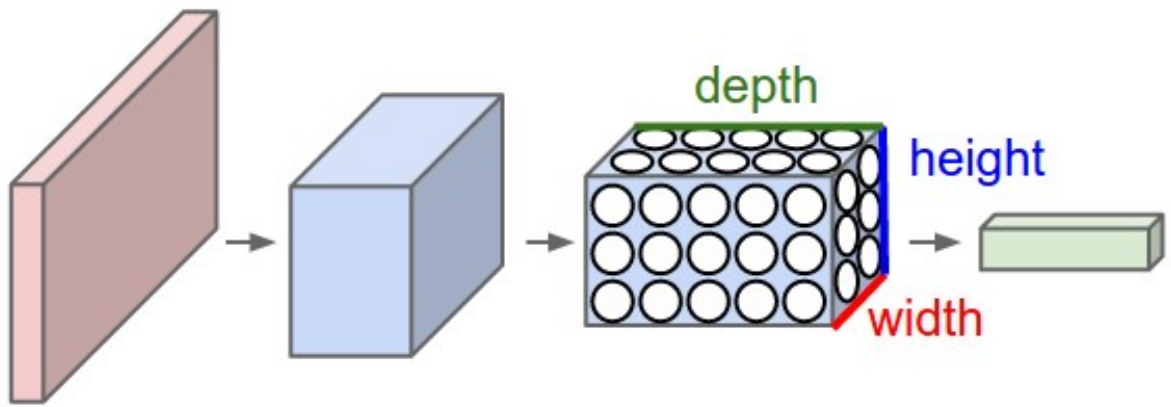
Typical architecture of Convnets



Source: https://iitmcvg.github.io/summer_school/DLSession3/

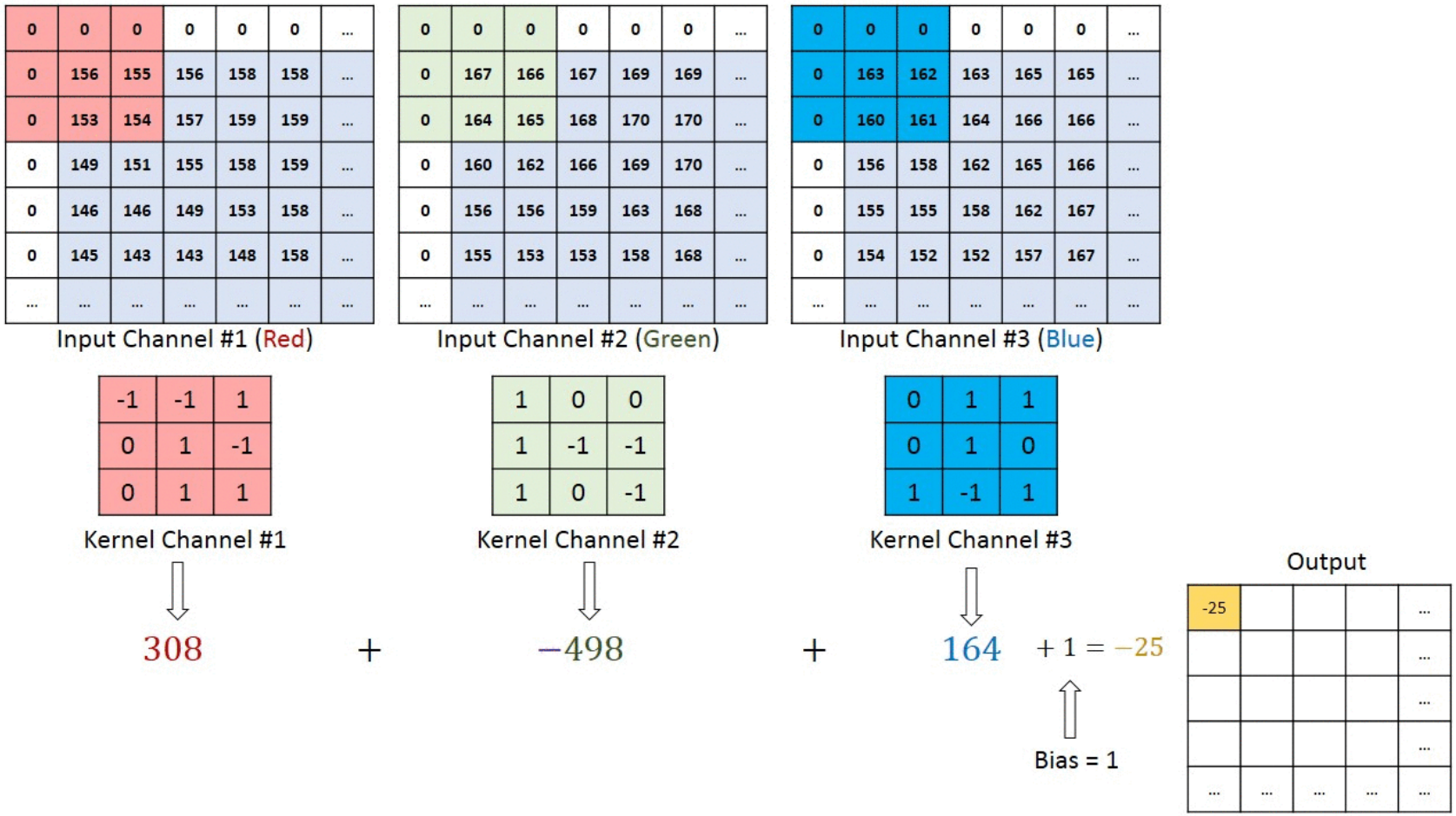
Graphical representation of ConvNet structure





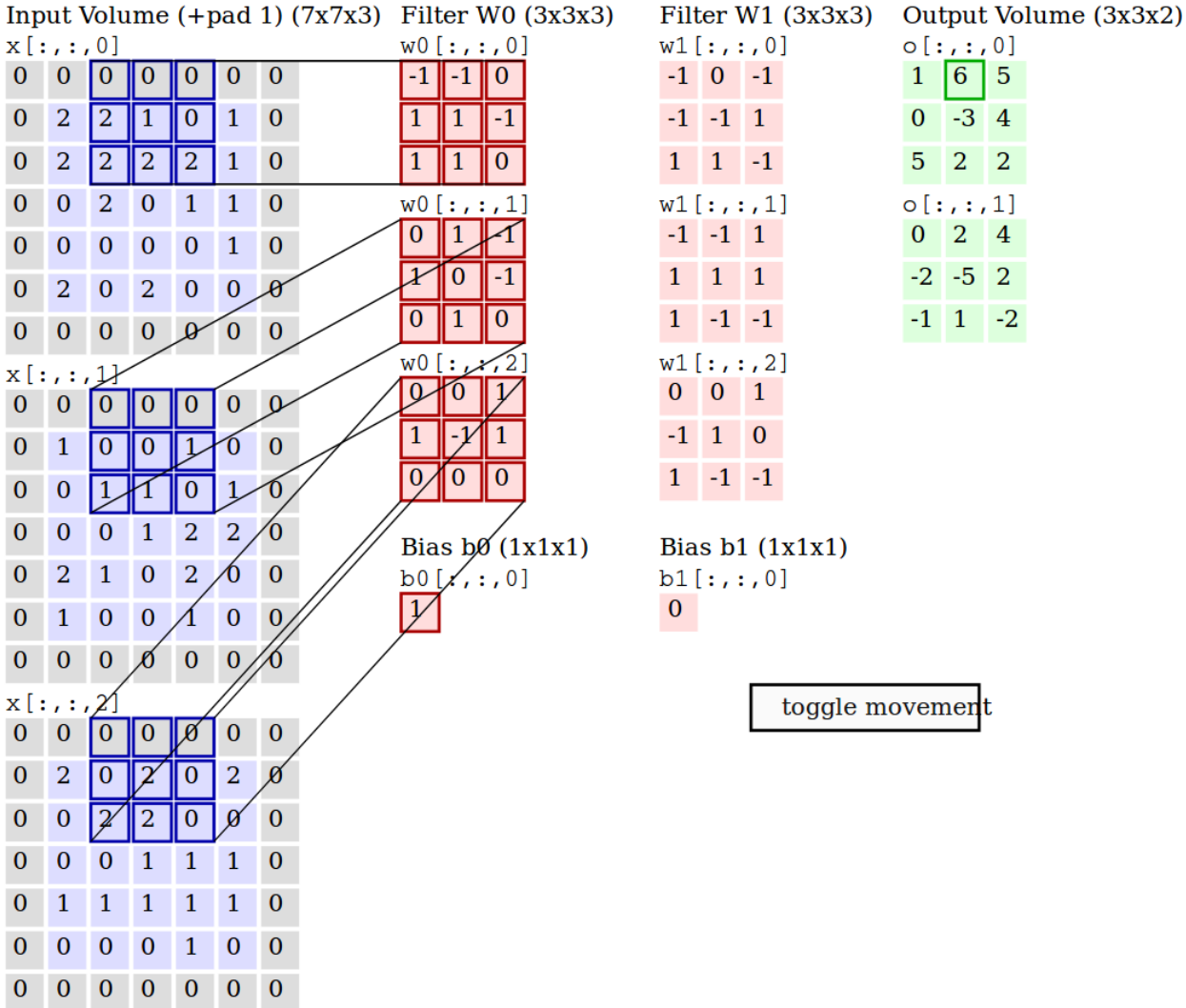

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
```

Convolution on multiple channels



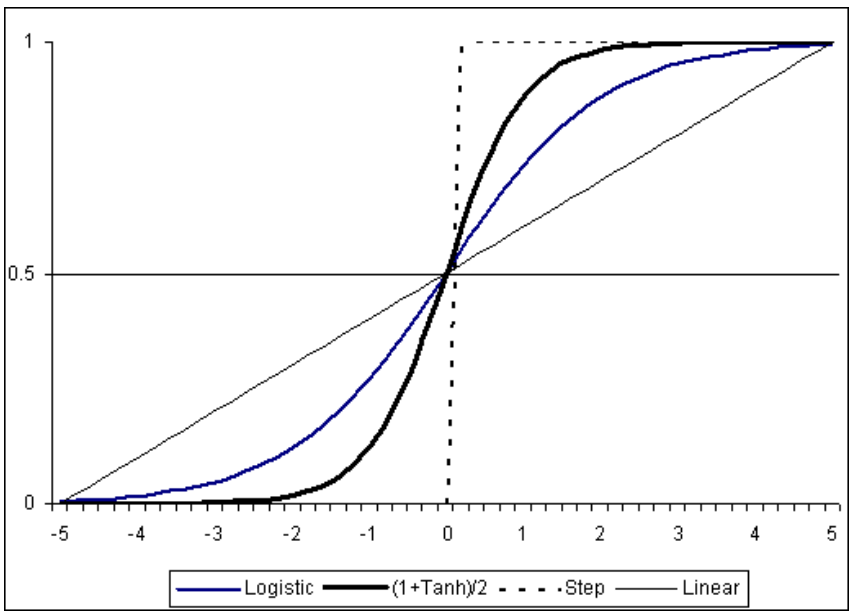
(http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html)

Multiple filters

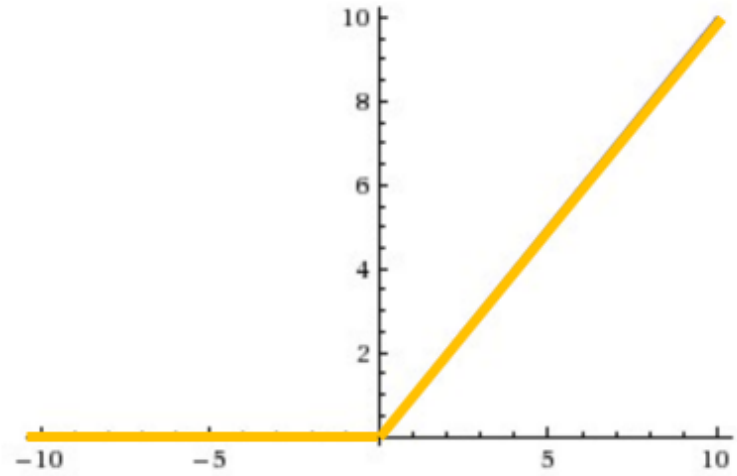


Concepts: padding, stride, kernel size, number of filters/maps

Activation functions



ReLU

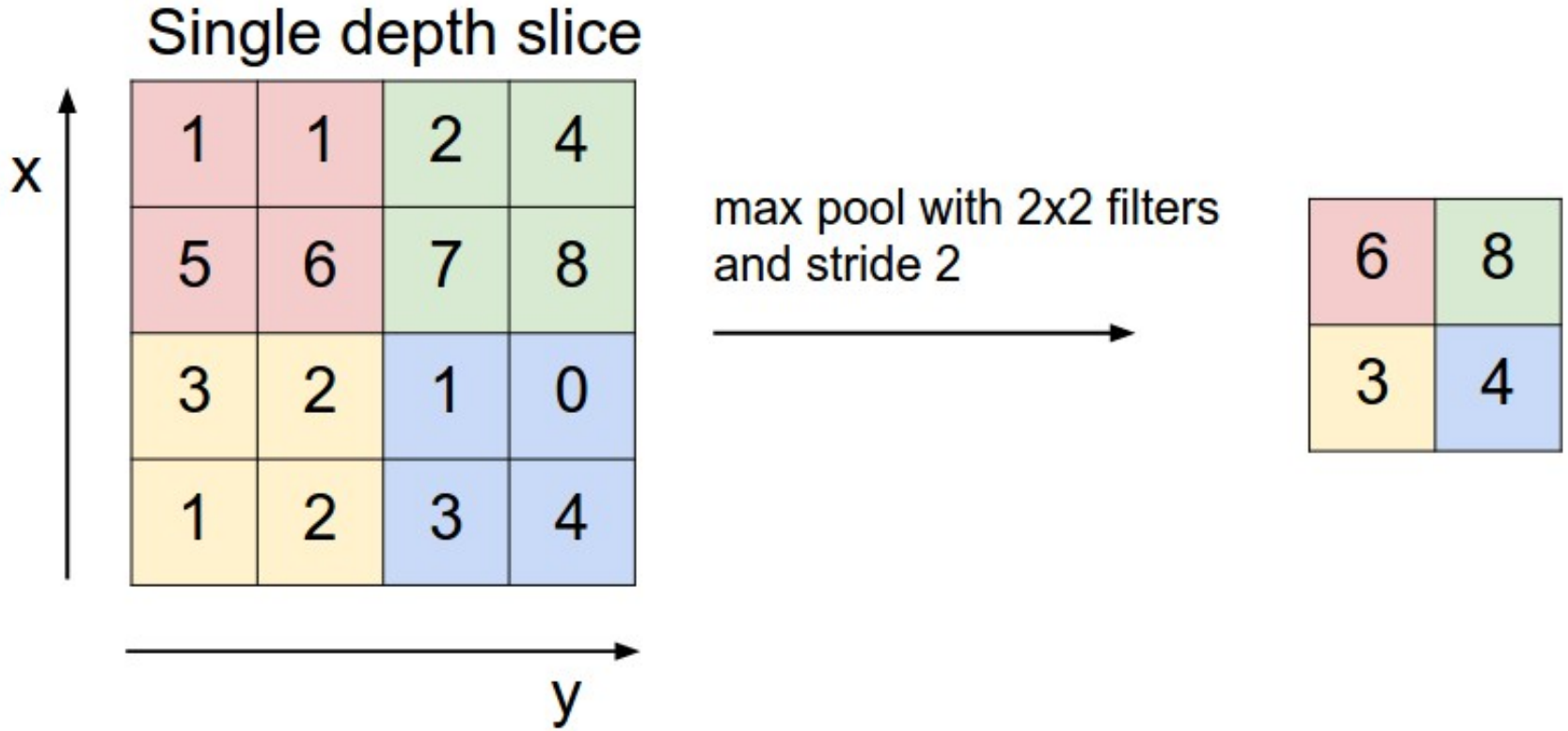


$$F(x) = \max(0, x)$$

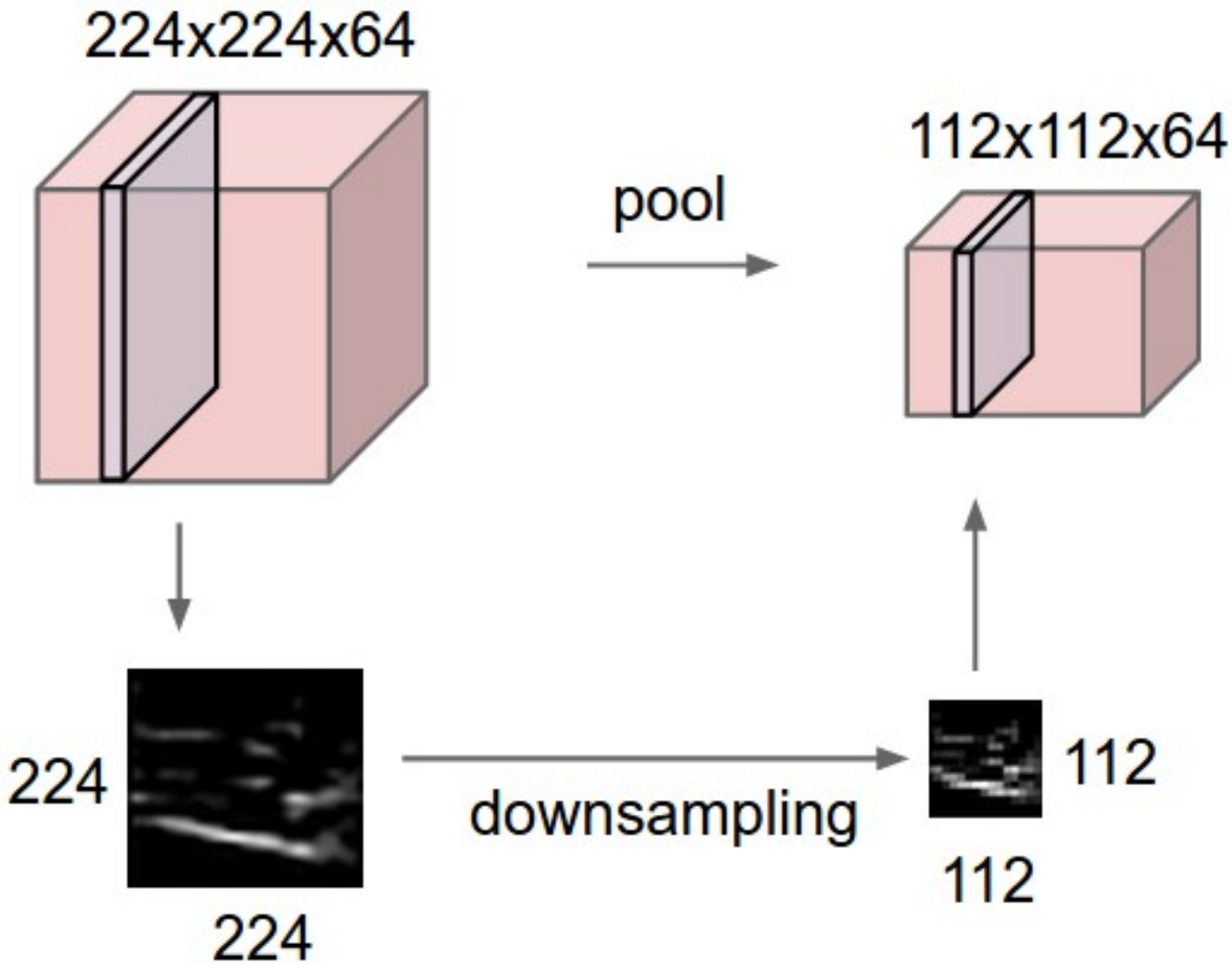
Subsampling

- via strides – skip pixels by steps of size d
- via pooling – aggregate $d \times d$ pixels

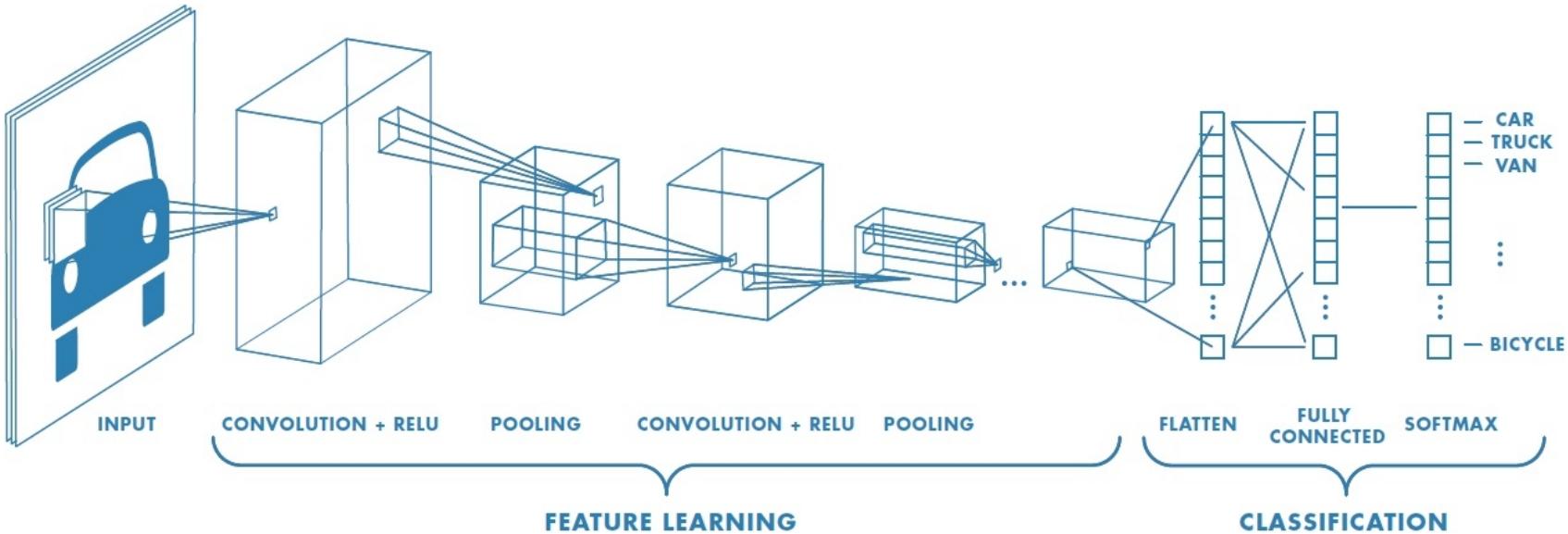
Max pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Pooling



Typical architecture of Convnets



Source: https://iitmcvg.github.io/summer_school/DLSession3/

- Convolutonal layers: feature extraction

- Fully connected layers: classification

Main CNN architectures used in ILSVRC

CNN models

- **AlexNet**, 2012: winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), 60M network parameters

  (Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012, pages 1097–1105)

- **VGG-11, 16 e 19**, 2014: 8, 13 e 16 convolutional layers, VGG-19 138M network parameters

  (Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. CoRR, abs/1409.1556, 2014)

CNN models

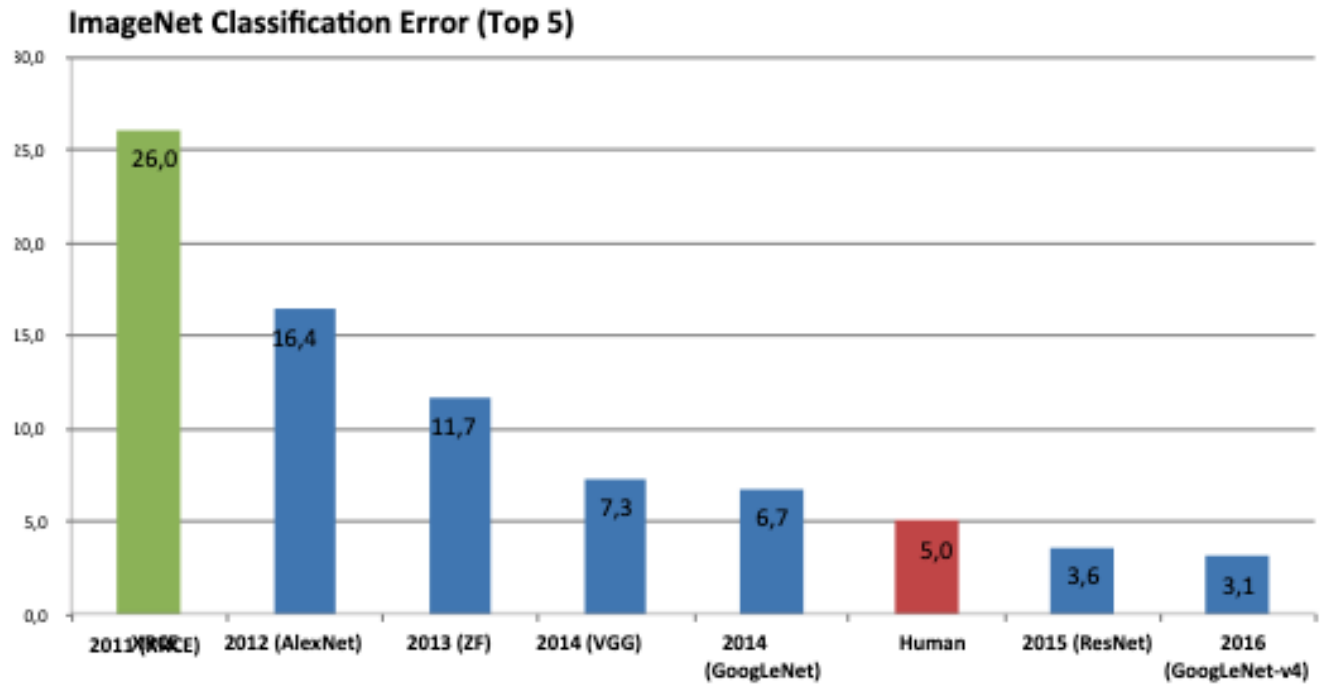
- **GoogleLeNet** (Inception), 2014: winner of ILSVRC 2014, inception layers, 7M network parameters

  (C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CVPR 2015, pages 1–9)

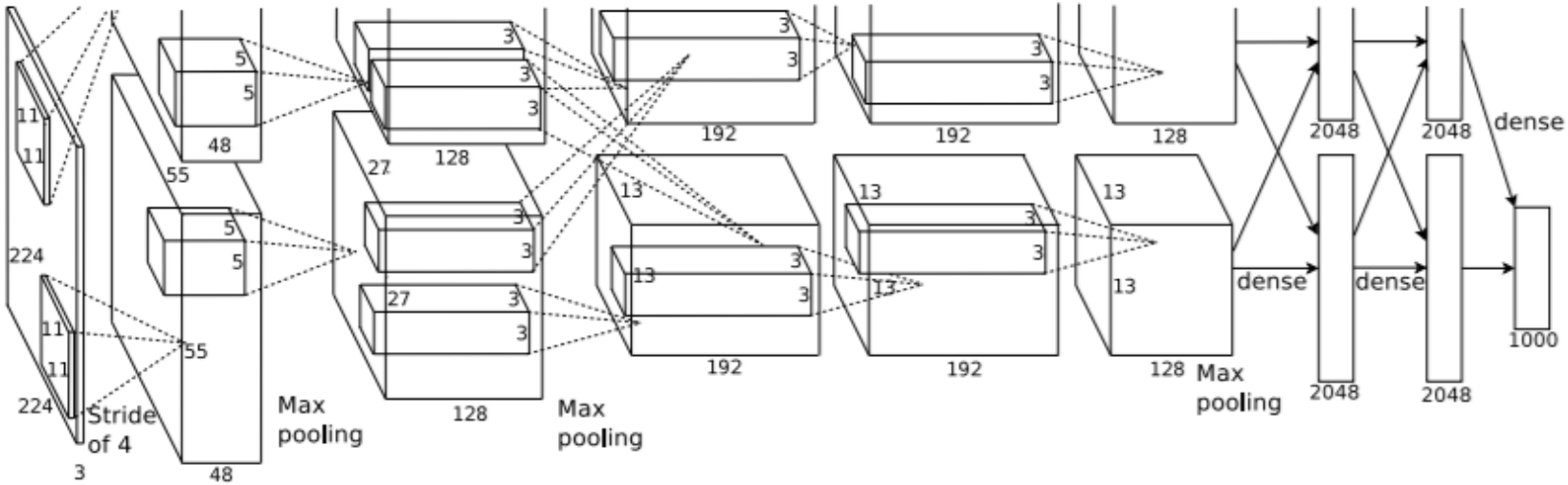
- **Residual Network** (ResNet), 2015: winner of ILSVRC 2015, 25.5M network parameters, residual block, vanishing gradient

  (K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR 2016, pages 770–778)

Large Scale Visual Recognition Challenge (ILSVRC) - ImageNet



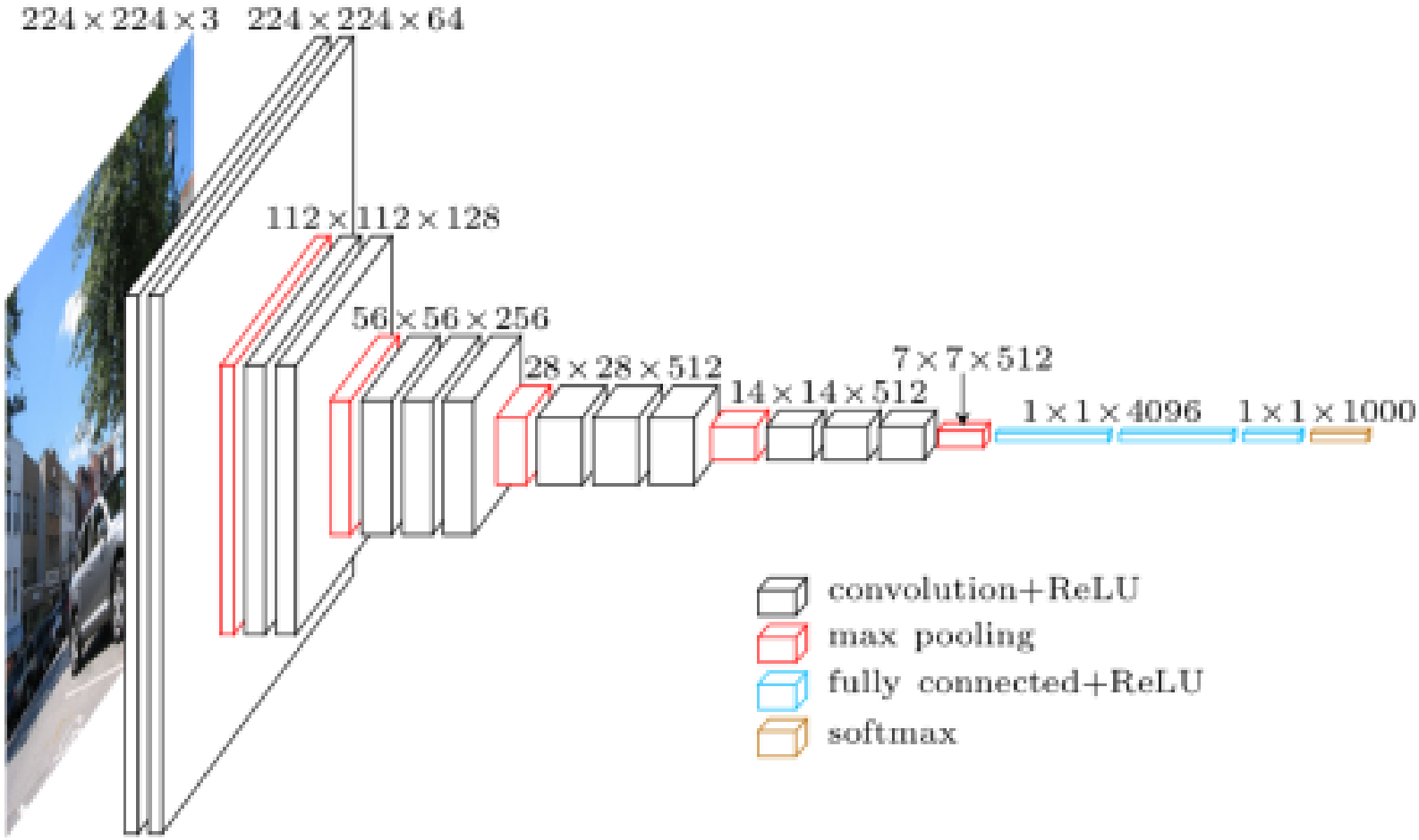
AlexNet (winner ILSVRC 2012)



AlexNet (winner ILSVRC 2012)

- succeeded on training a large CNN
- 60 million parameters
- used 2 GPUs
- proposed ReLU
- used dropout
- used data augmentation (rotation, scale, crops, etc)
- indication that number of layers is important

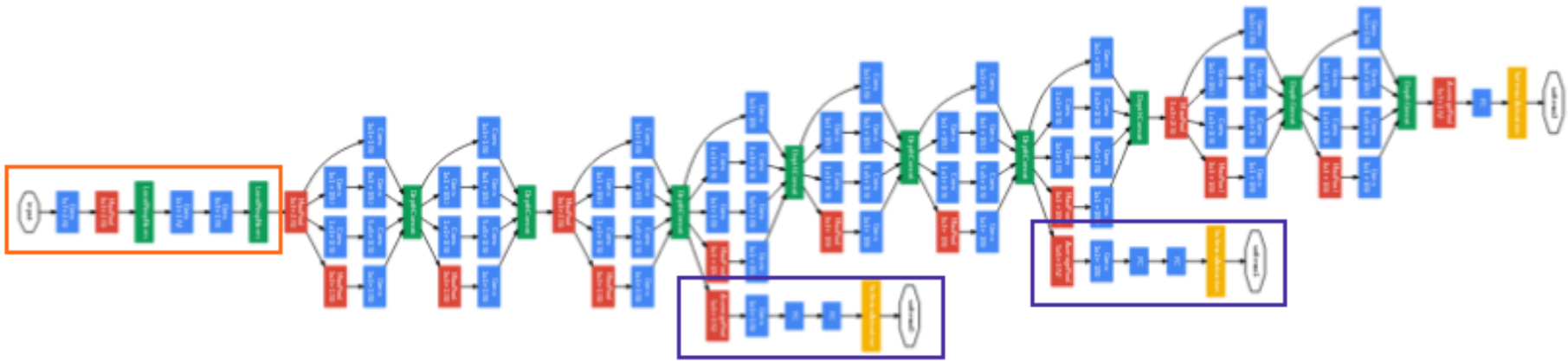
VGGNet (2nd place ILSVRC 2014)



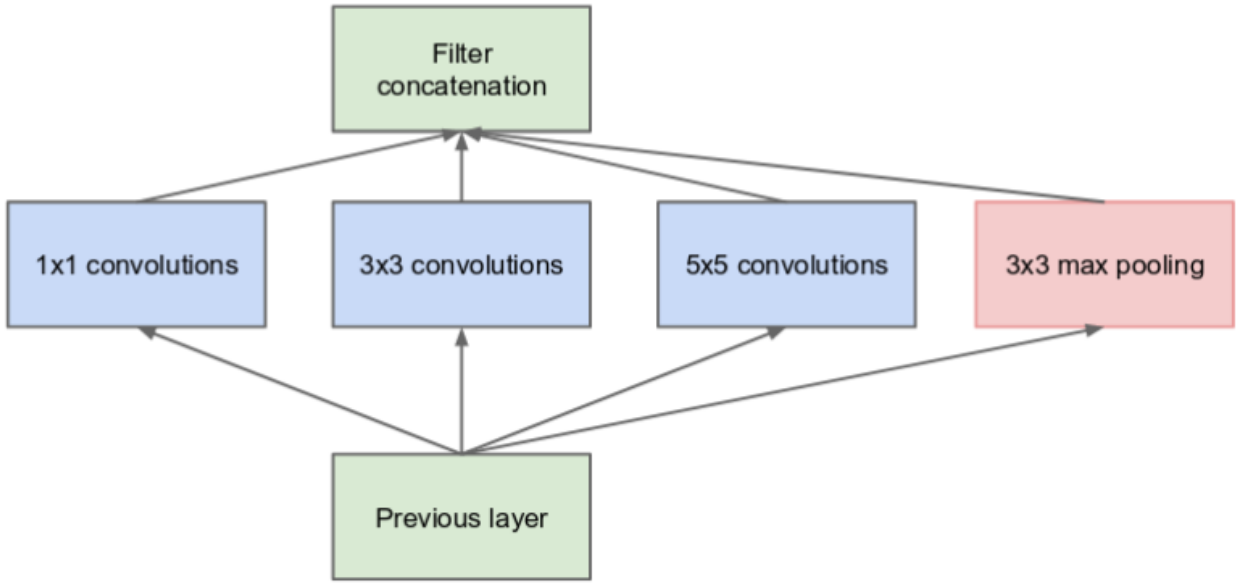
VGGNet (2nd place ILSVRC 2014)

- multiple layers (ex., 16 e 19)
- only 3x3 convolutions
- stride 1 for the convolutions (AlexNet used stride ¿ 1)
- first layers with few filters and gradually increasing as we advance throuh the layers
- 144 millions of parameters

Inception - GoogleLeNet (winner ILSVRC 2014)

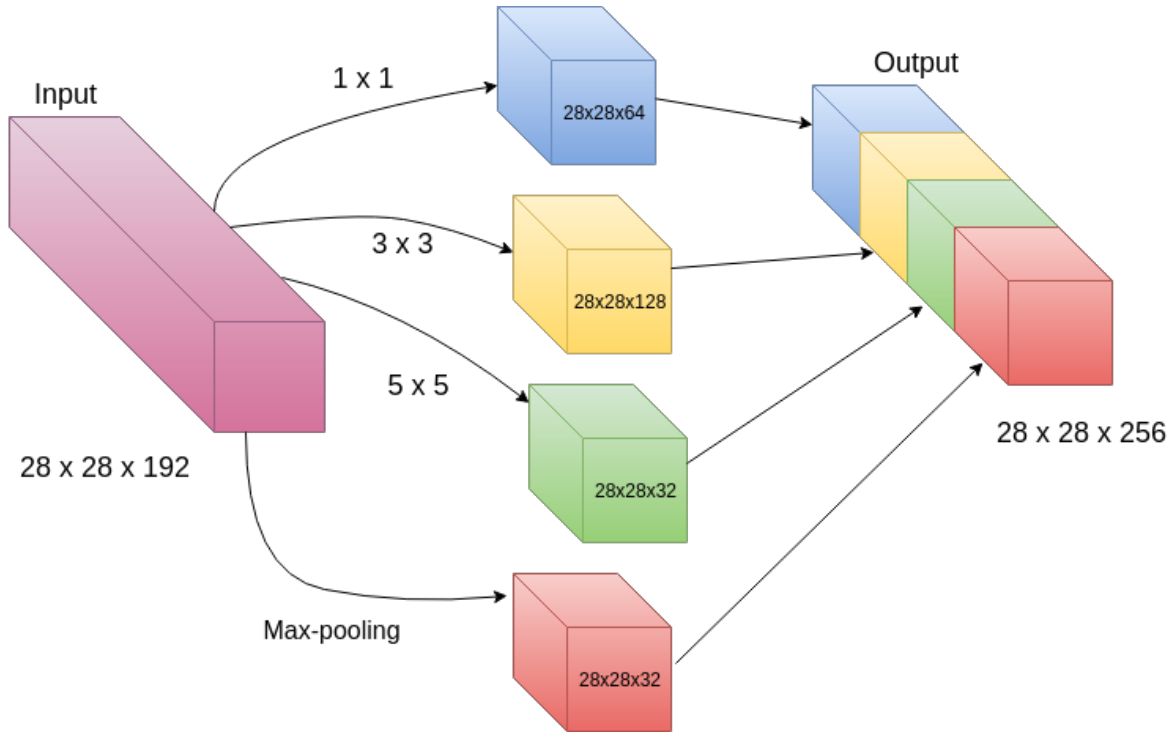


Inception – naïve version



(a) Inception module, naïve version

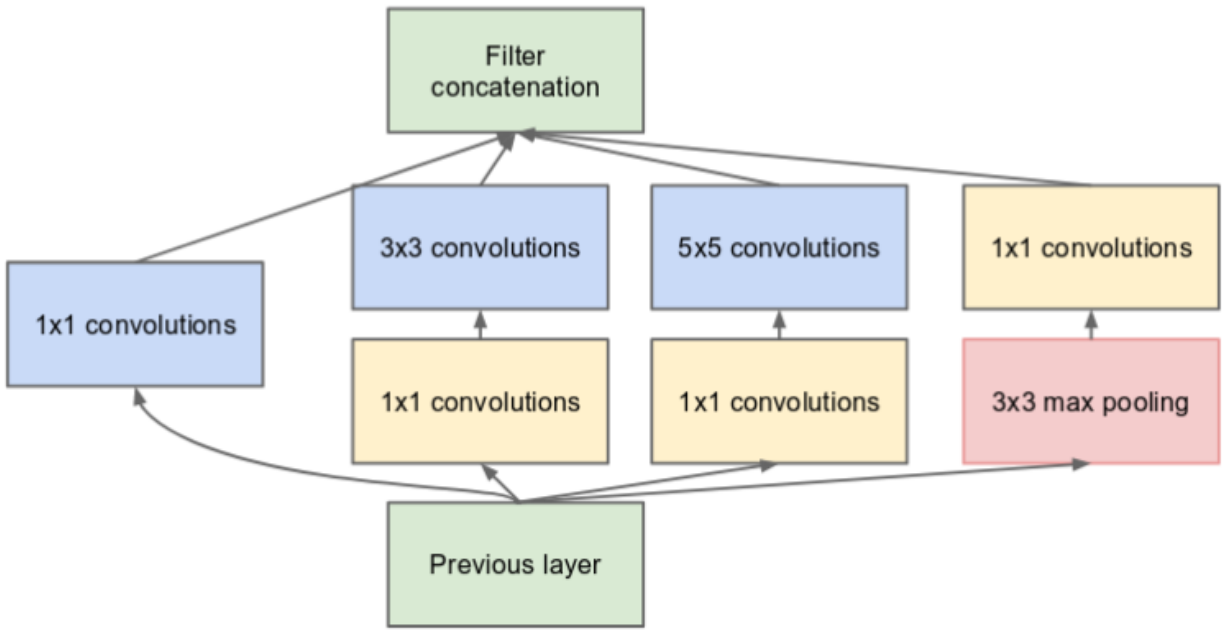
Inception – naïve version example



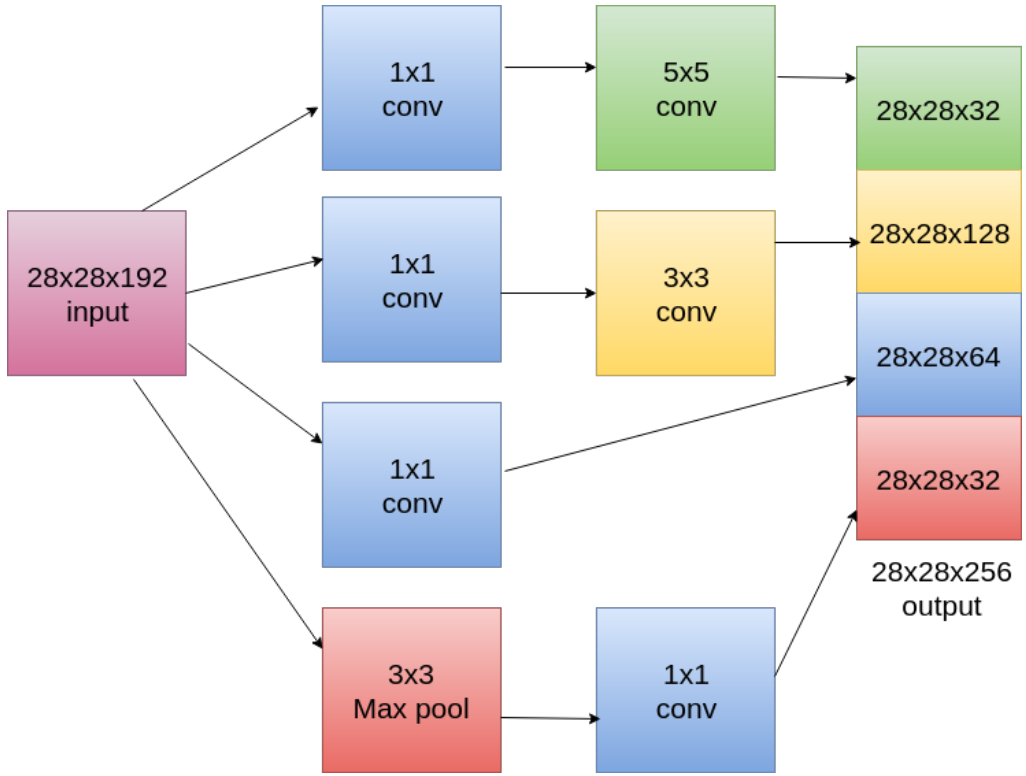
Camada de entrada = 28x28x192

Convolução 5x5: com 32 filtros, camada de saída = 28x28x32. Para cada ponto da camada de saída, o número de multiplicações necessárias é 5x5x192. Como são 28x28x32 pontos na camada de saída, o total de multiplicações necessárias é 5x5x192 x 28x28x32 = 120 milhões.

Inception – efefctive verison

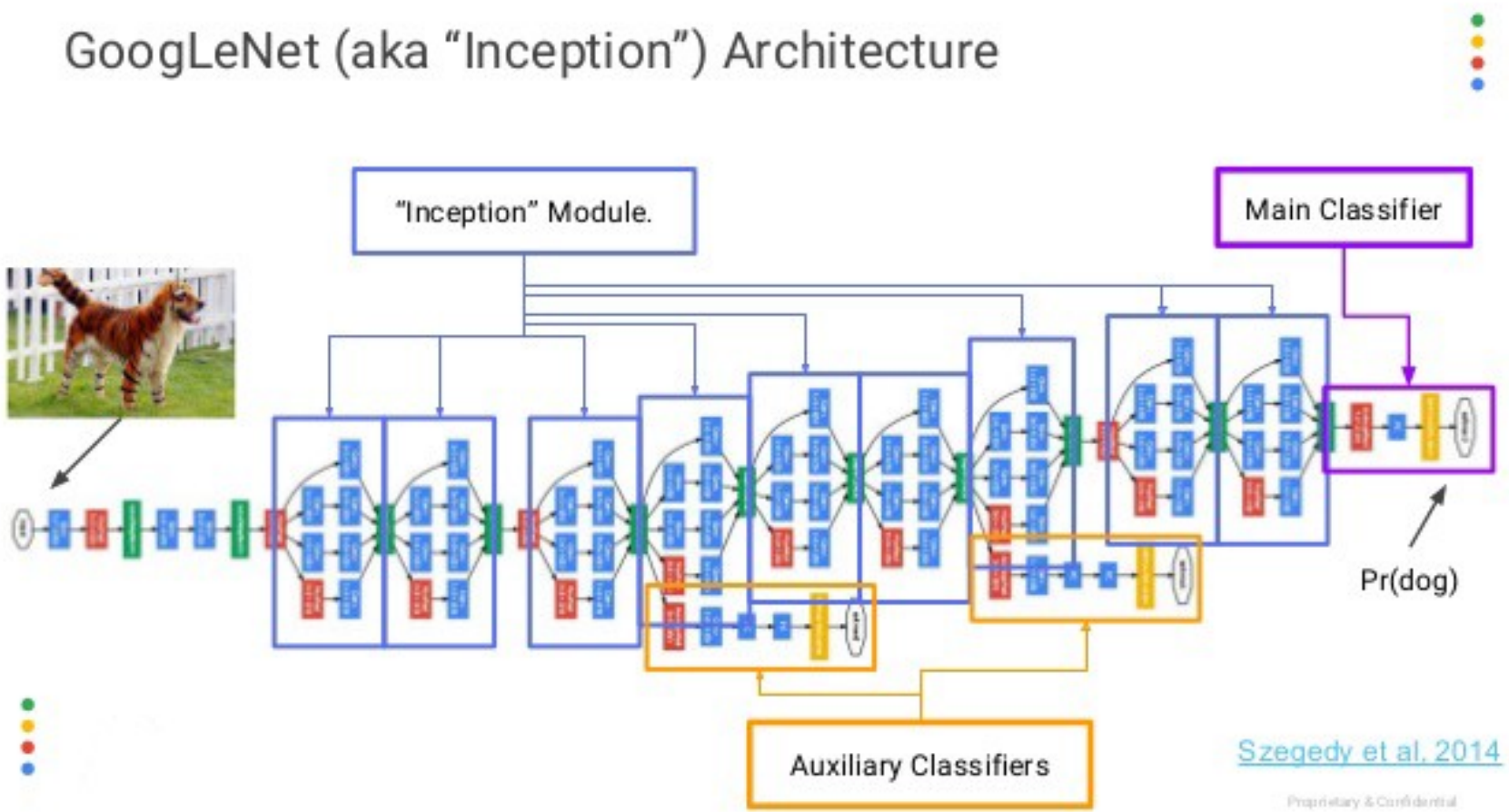


(b) Inception module with dimension reductions



Entrada 28x28x192 $\Longrightarrow$ 16 filtros 1x1 $\Longrightarrow$ camada intermediária 28x28x16 $\Longrightarrow$ 32 convoluções 5x5 $\Longrightarrow$ saída 28x28x32. Total de multiplicações: cada ponto na camada intermediária requer 1x1x192 multiplicações, e portanto 1x1x192 x 28x28x16 = 2.4 milhões para cálculo da camada intermediária; cada ponto na camada de saída requer 5x5x16 multiplicações e portanto 5x5x16 x 28x28X32 = 10 milhões para o cálculo da camada de saída. Total = 12.4 milhões.

Inception - GoogleLeNet (winner ILSVRC 2014)

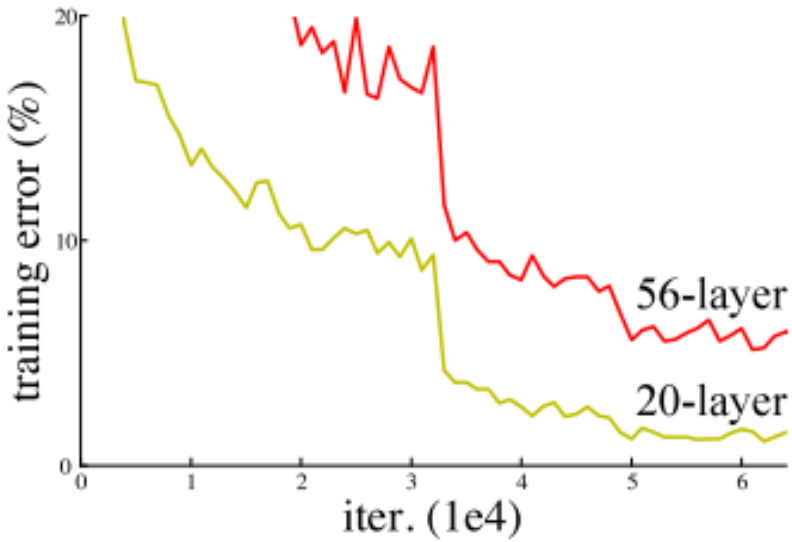


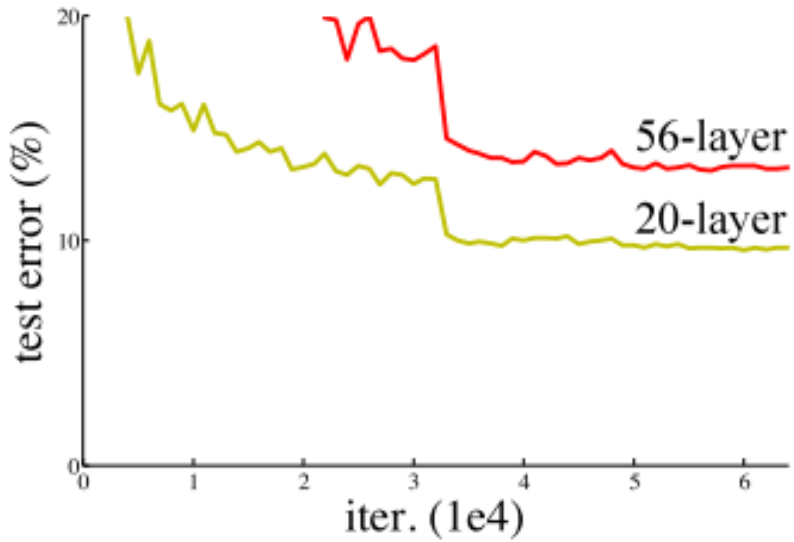
Inception - GoogleLeNet (winner ILSVRC 2014)

- concept of "network in network"
- inception module ("choices" made during training)
- use of 1x1 convolution
- auxiliary output to reinforce activation of intermediary layers
- more aggressive data augmentation tha AlexNet
- 4 million parameters (?)
- variants (e.g., change 5x5 with two 3x3, or change 3x3 with a 1x3 and a 3x1)

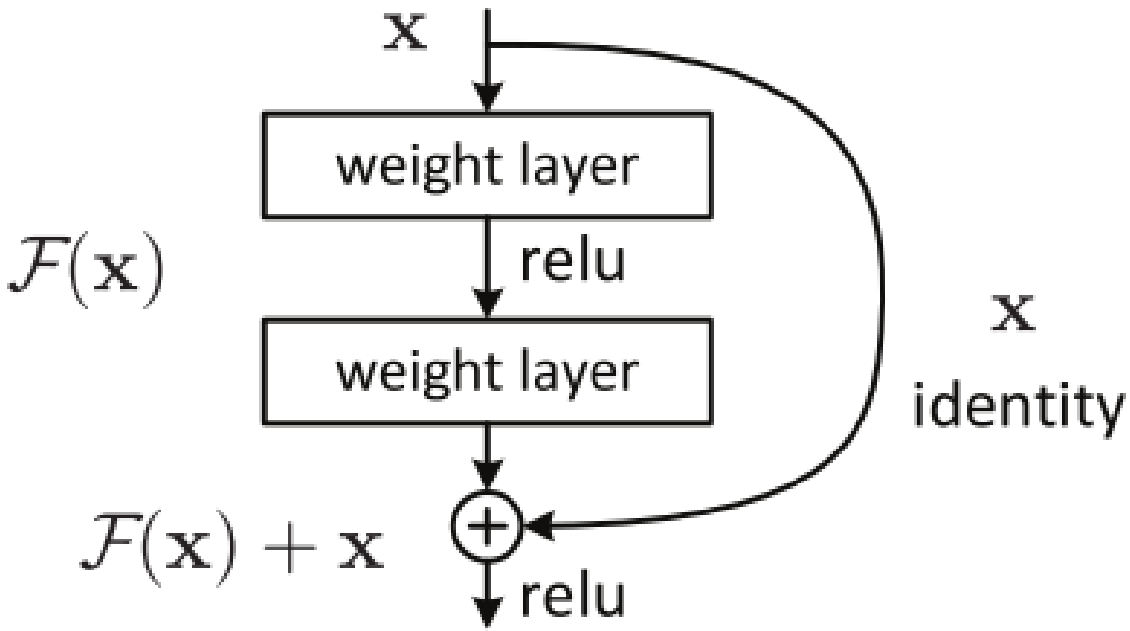
ResNet (winner ILSVRC 2015)

Apesar de ser aceito que profundidade da rede é importante, observou-se que camadas adicionais degradavam a acurácia no conjunto de treinamento



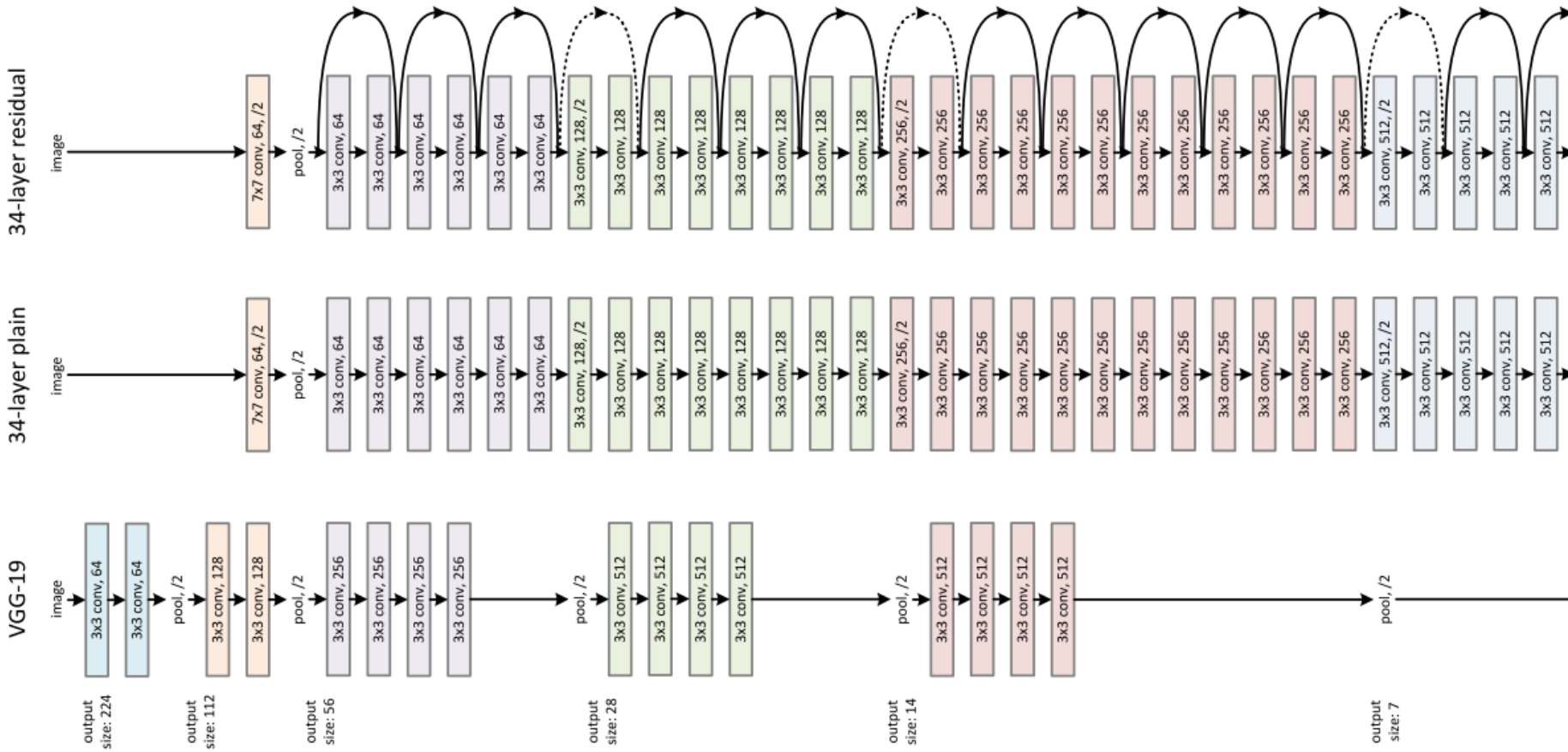


ResNet (winner ILSVRC 2015)



Entrada x. Pode ser conveniente aprender um mapeamento ($H(x)$) que seja a identidade. Mas representar identidade por meio de transformações não lineares não é simples. Assim, em vez de H, aprende-se o resíduo $\mathcal{F}(x)$, de mode que $H(x) = \mathcal{F}(x) + x$ e assim a identidade corresponde a resíduo zero

ResNet (winner ILSVRC 2015)



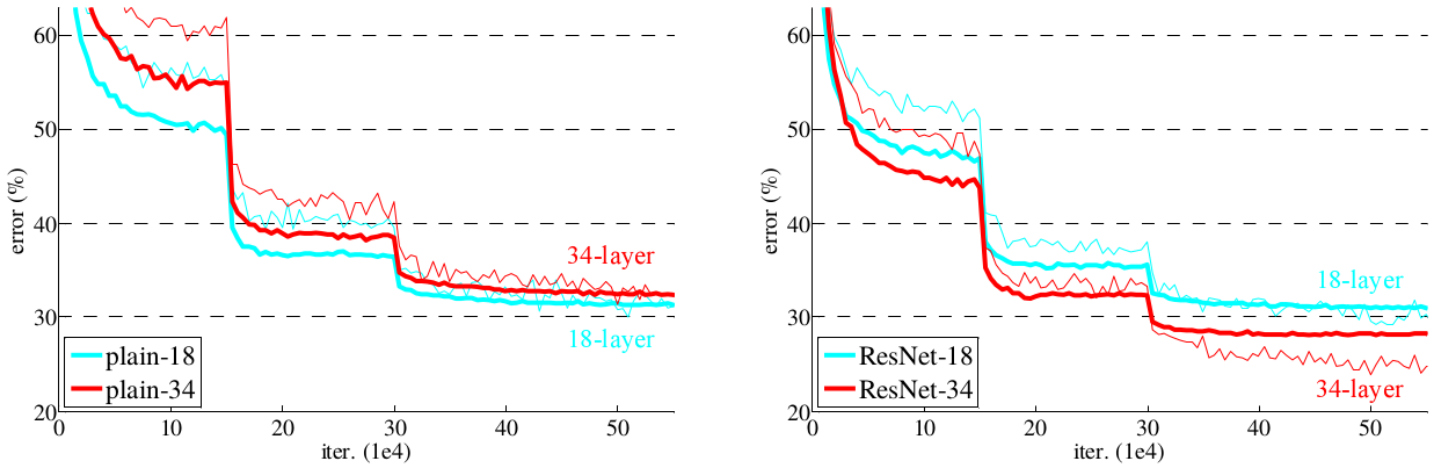


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

ResNet (winner ILSVRC 2015)

- identificam uma degradação associada ao aumento de camadas, que aparentemente não é *overfitting* nem *vanishing gradient*
- propõe módulo para aprender o resíduo da transformação desejada
- aplica *batch normalization* após cada convolução e antes da ativação
- não usa *dropout*
- conseguem treinar redes com mais de 100 camadas
- testado em outros datasets (CIFAR, PASCAL, MS-COCO)
- ResNet 56 layers: 0.85M parameters (?)
- ResNet 110 layers: 1.7M parameters (?)

DenseNet

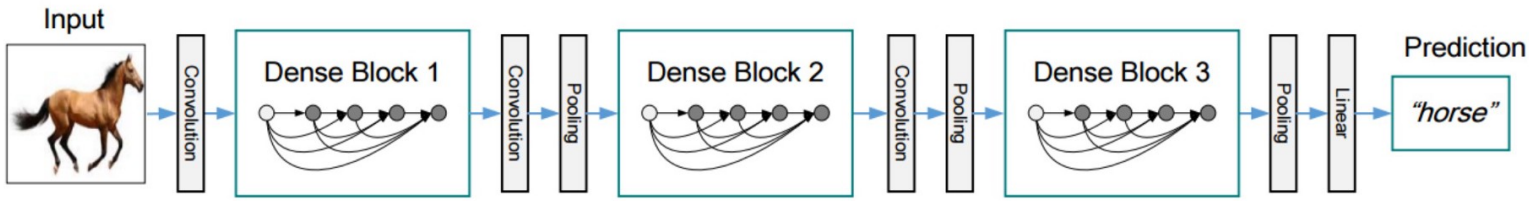
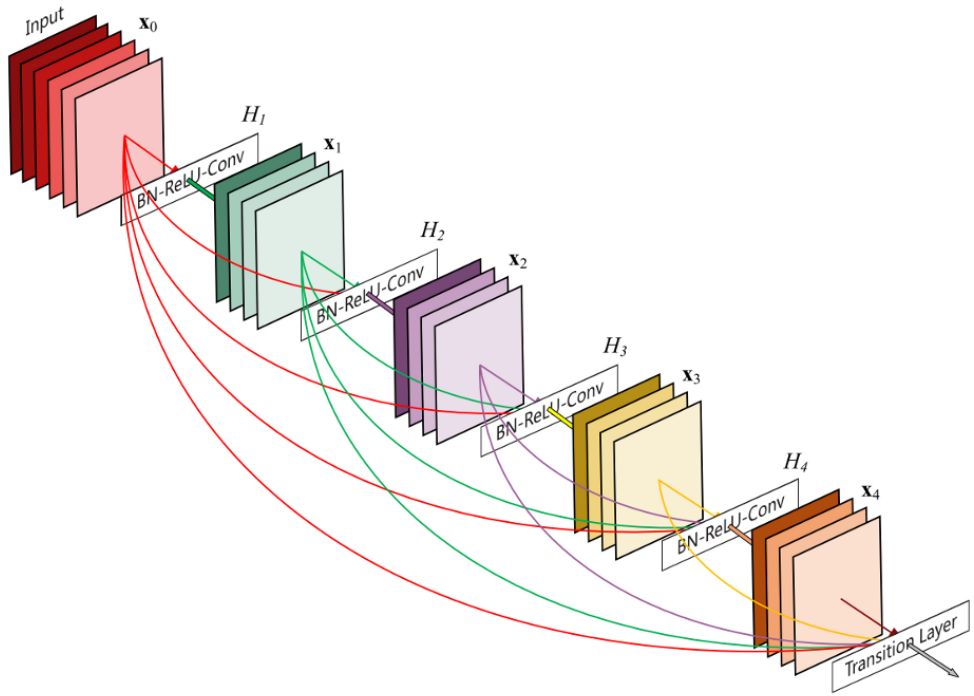
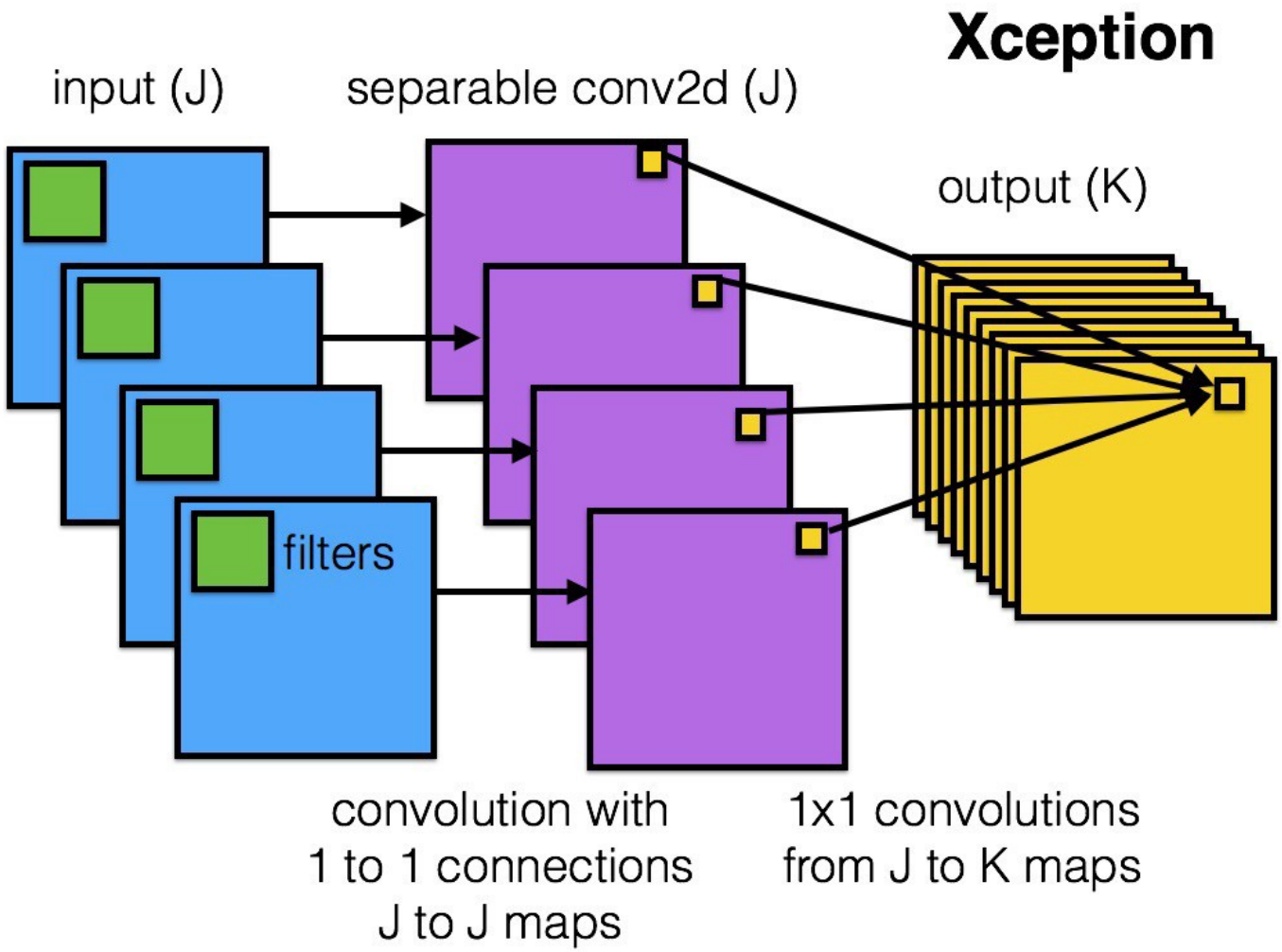


Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.



Xception


Xception
input (J)
separable conv2d (J)
output (K)
filters
convolution with
1 to 1 connections
J to J maps
1x1 convolutions
from J to K maps

Libraries, references, code sample

Bibliotecas para deep learning

TensorFlow https://www.tensorflow.org/

PyTorch https://pytorch.org/

Keras https://keras.io/

Some links

Neural nets: http://neuralnetworksanddeeplearning.com/

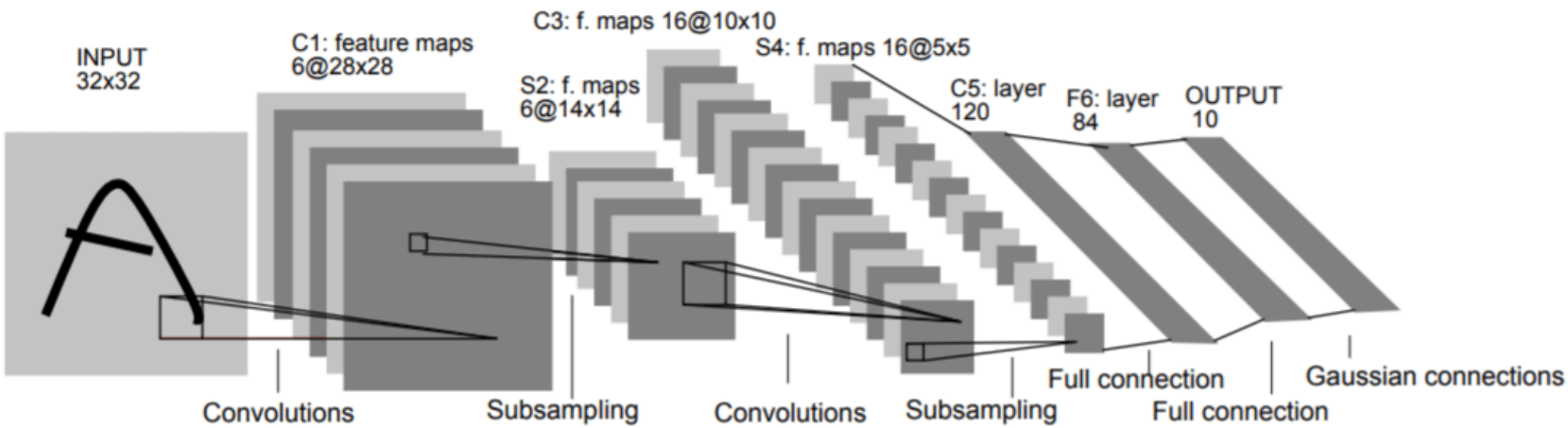
NN and CNN: Stanford CS class http://cs231n.github.io/convolutional-networks/

Github (F. Chollet)

https://github.com/fchollet/deep-learning-with-python-notebooks

LeNet-5 com Keras

LeNet-5 é a rede usada pelo LeCun no MNIST



LeNet-5 com Keras

LeNet-5 é a rede usada pelo LeCun no MNIST

```
-------------------------------------------------
Convolution: Input = 32x32x1. Output = 28x28x6
SubSampling: Input = 28x28x6. Output = 14x14x6

Convolution: Input = 14x14x6. Output = 10x10x16
SubSampling: Input = 10x10x16. Output = 5x5x16

Fully Connected: Input = 5x5x16. Output = 120
Fully Connected: Input = 120. Output = 84

Output 10
-------------------------------------------------
```

LeNet-5 com Keras

```
-------------------------------------------------
model = keras.Sequential()

model.add(layers.Conv2D(filters=6, kernel_size=(3, 3),
          activation='relu', input_shape=(32,32,1)))
model.add(layers.AveragePooling2D())

model.add(layers.Conv2D(filters=16, kernel_size=(3, 3),
          activation='relu'))
model.add(layers.AveragePooling2D())

model.add(layers.Flatten())

model.add(layers.Dense(units=120, activation='relu'))
model.add(layers.Dense(units=84, activation='relu'))
model.add(layers.Dense(units=10, activation = 'softmax'))
-------------------------------------------------
```

Transfer learning

Transfer learning refers to using whatever has been learned in a certain domain to tackle similar problems in another domain

Why? Because training data is a scarse resource

Common practice: train a network with images of ImageNet, and then fine-tune the weights with data of the target domain

Example

Comparison of traditional features vs features extracted with a network pretrained on ImageNet

Classificação de imagens de galáxias

(com Fabrício Ferrari e Mateus Espadoto)

Dataset used: EFIGI (https://www.astromatic.net/projects/efigi)

Procedimento executado:

1. Selecionar apenas as espirais e as elípticas
2. Extrair *features* para cada imagem
   - Usando Morfometryka (dados fornecidos por Fabrício Ferrari, FURG)
   - Usando uma rede convolucional pré-treinada com o ImageNet
3. Separar dados em treinamento e validação
4. Treinar classificador logístico usando dados de treinamento
5. Avaliar desempenho (de predição) no conjunto de validação

Para inspecionar visualmente a qualidade das *features* extraídas, nos slides seguintes:

- rótulo sobre a imagem é a classificação correta
  (azul="elíptica", laranja="espiral")

- primeira coluna: exemplos de imagens classificadas erroneamente

- colunas 2 a 5: imagens mais similares ao da primeira coluna
de acordo com a similaridade de *features*


3245: eli
439: spi
1038: spi
1838: spi
360: spi
Classificação errada
4 vizinhos mais próximos
2856: eli
1449: spi
1431: spi
1524: spi
1587: spi
3308: spi
3248: eli
395: eli
3190: eli
3276: eli
1855: eli
1049: spi
1616: spi
972: spi
2754: spi
2892: eli
2893: spi
2935: spi
472: spi
2623: spi
Features convolucionais


Classificação errada
4 vizinhos mais próximos
Features convolucionais
132: eli
3237: spi
816: spi
1906: spi
1515: spi
1935: spi
3235: eli
2135: eli
2458: eli
190: eli
3008: spi
3122: eli
2238: eli
63: eli
1508: eli
1035: spi
2482: eli
2181: eli
2040: eli
3220: eli


1023: eli
1087: spi
982: spi
1222: spi
974: spi
Classificação errada
4 vizinhos mais próximos
1773: eli
2084: spi
680: spi
2594: spi
3177: spi
Features morfométricos
987: eli
1040: spi
1229: spi
998: spi
1078: spi
1282: spi
975: eli
1007: eli
1083: eli
1116: eli
1876: eli
1615: spi
541: spi
2434: spi
2764: spi


Classificação errada
4 vizinhos mais próximos
Features morfométricos
1830: eli
139: spi
3192: spi
2927: spi
1989: spi
42: eli
1320: spi
1600: spi
2199: spi
3231: spi
2541: eli
2037: spi
1449: spi
73: spi
615: spi
1854: eli
94: spi
164: spi
57: spi
2944: spi

Projeções das *features*

Features convolucionais

Features morfométricos

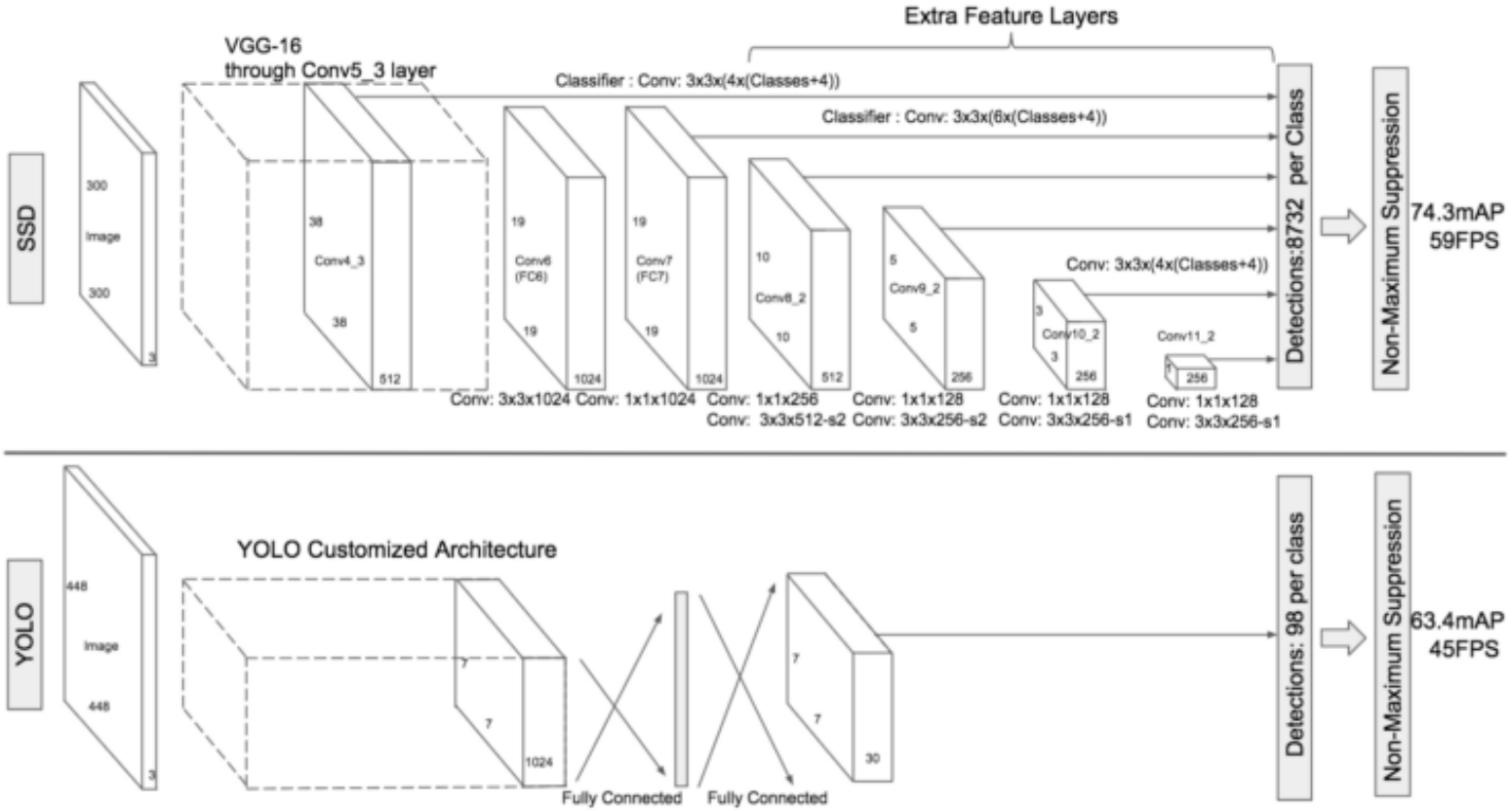
Visualization

- of the newtork itself
- parts of the image that affect output
- parts of the image that affect a specific neuron

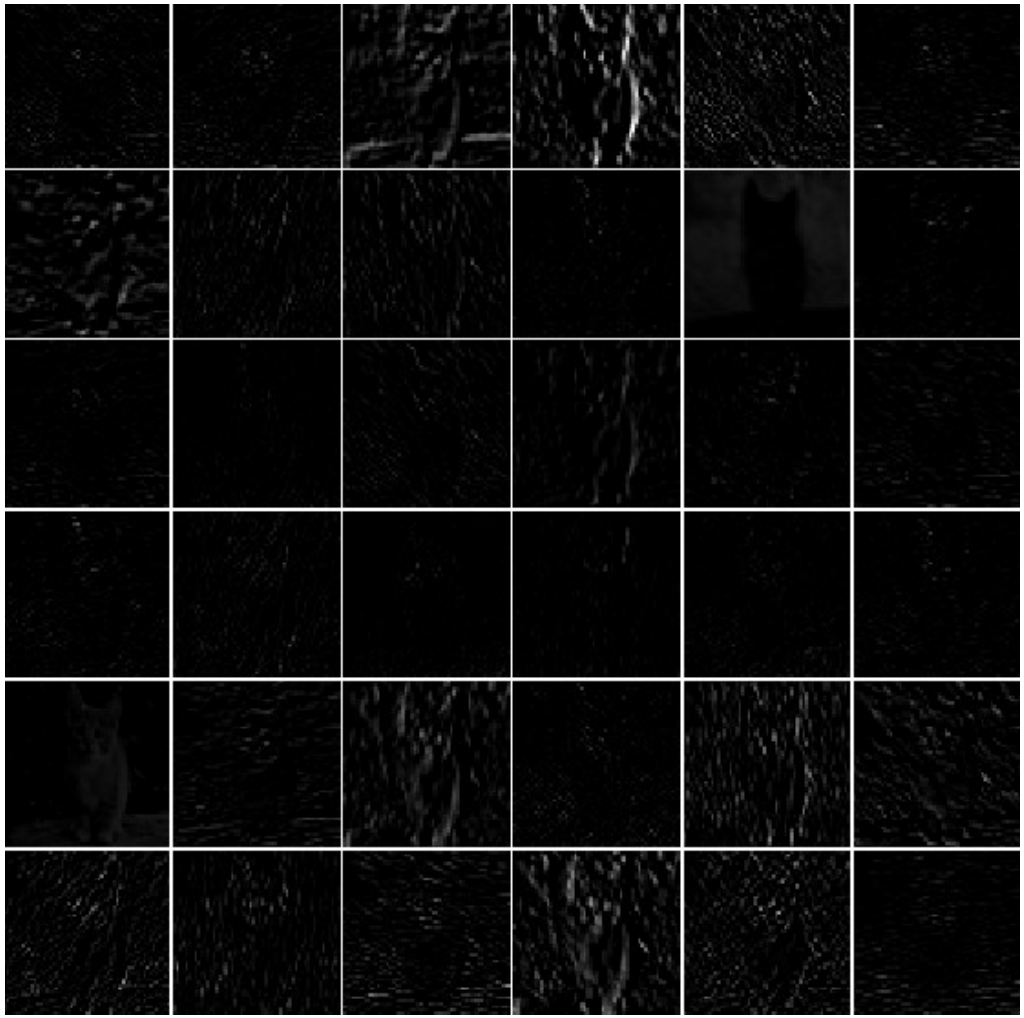
Visualizar a rede – Arquitetura

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1000)              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 1000, 100)         17410600
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 996, 128)          64128
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 199, 128)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 195, 128)          82048
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 39, 128)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 35, 128)           82048
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 1, 128)            0
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 20)                2580
=================================================================
Total params: 17,657,916
Trainable params: 247,316
Non-trainable params: 17,410,600
_________________________________________________________________
None
```

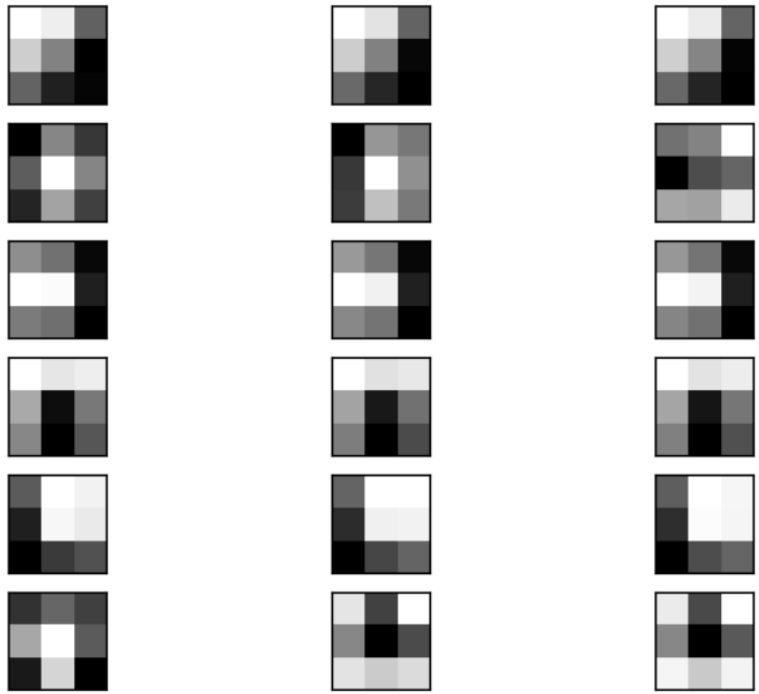
Visualizar a rede – Arquitetura



Visualizar a rede – mapa de ativação/feature map



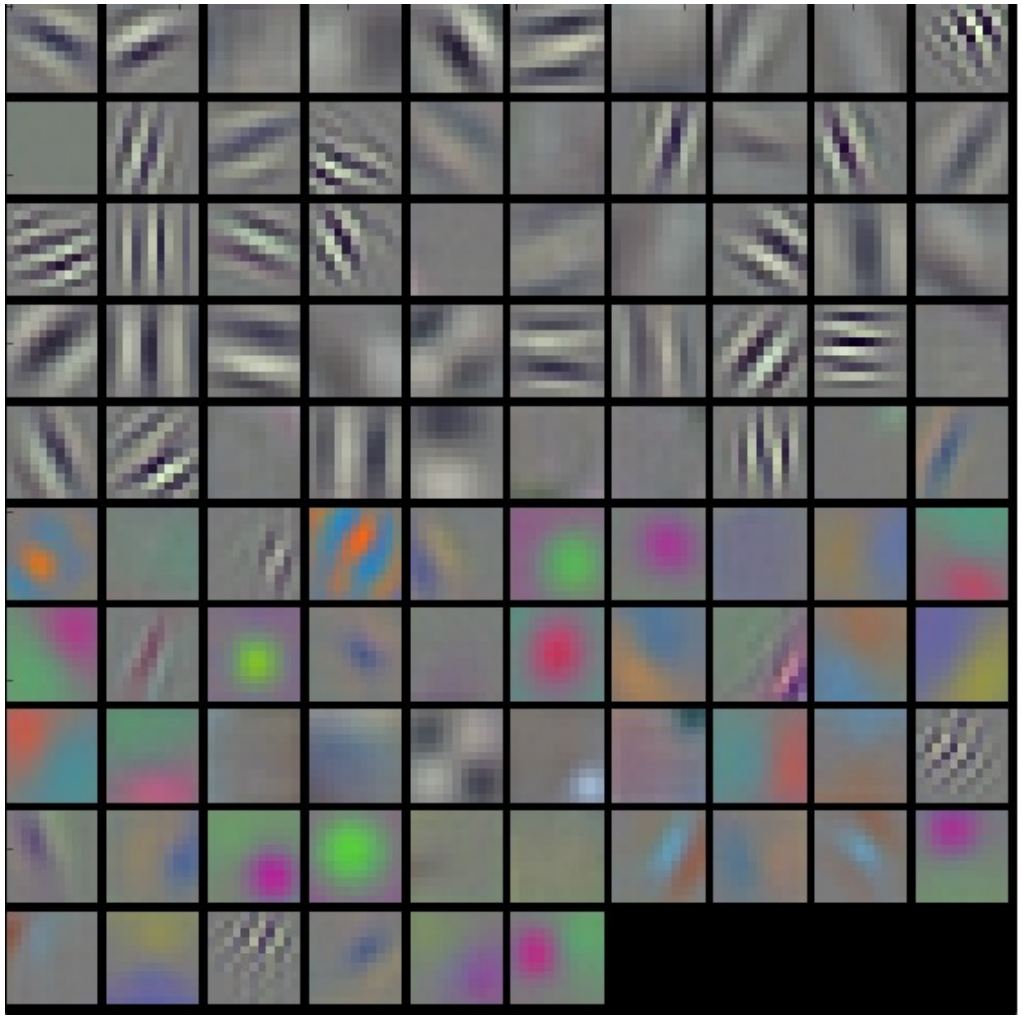
Visualizar a rede – Filtros



https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

6 filters (each row), from the first layer of VGG

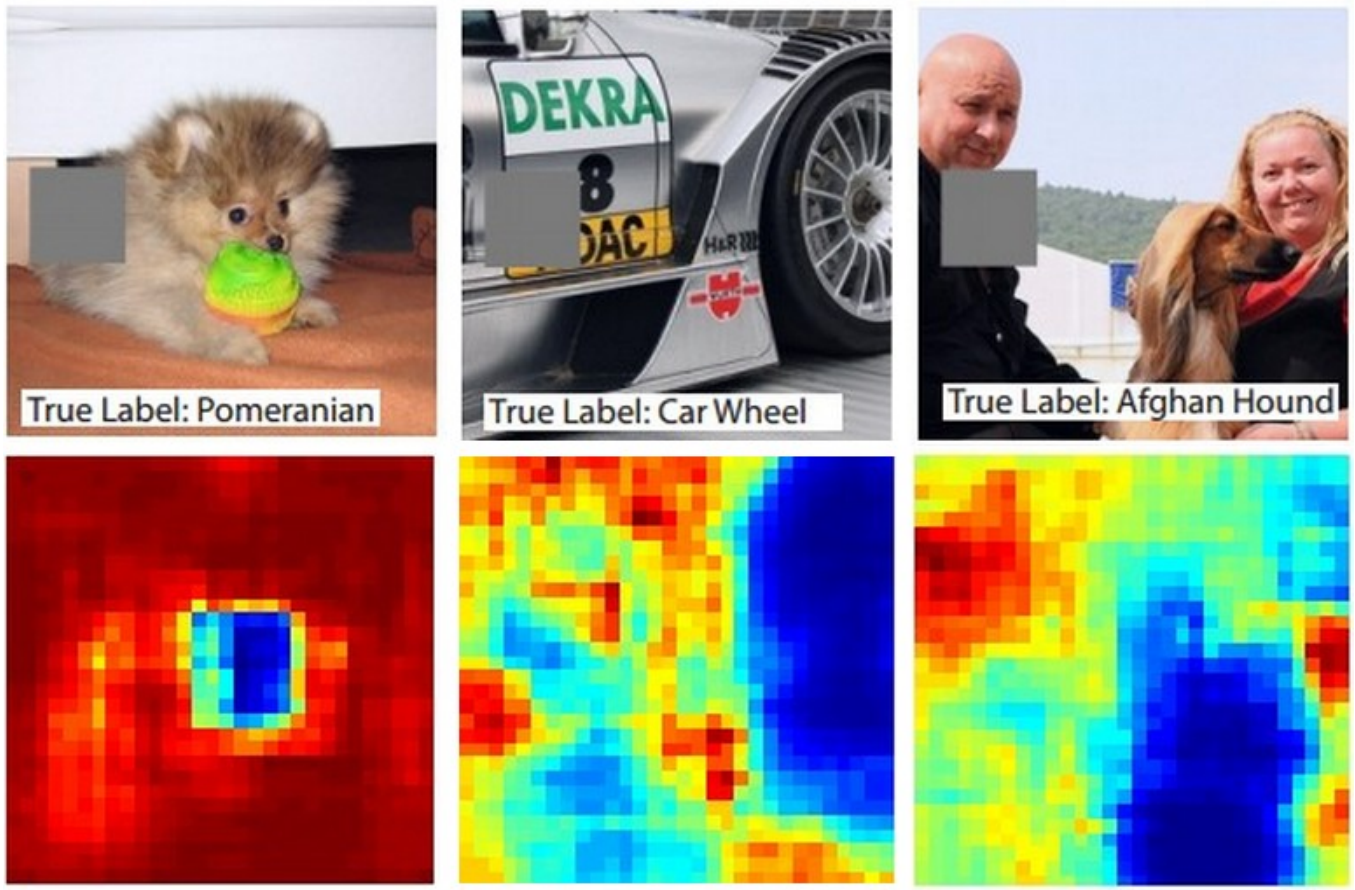
Visualizar a rede – Filtros



Images that maximize the response of the filters (AlexNet)

Qual parte da imagem afeta a classificação?

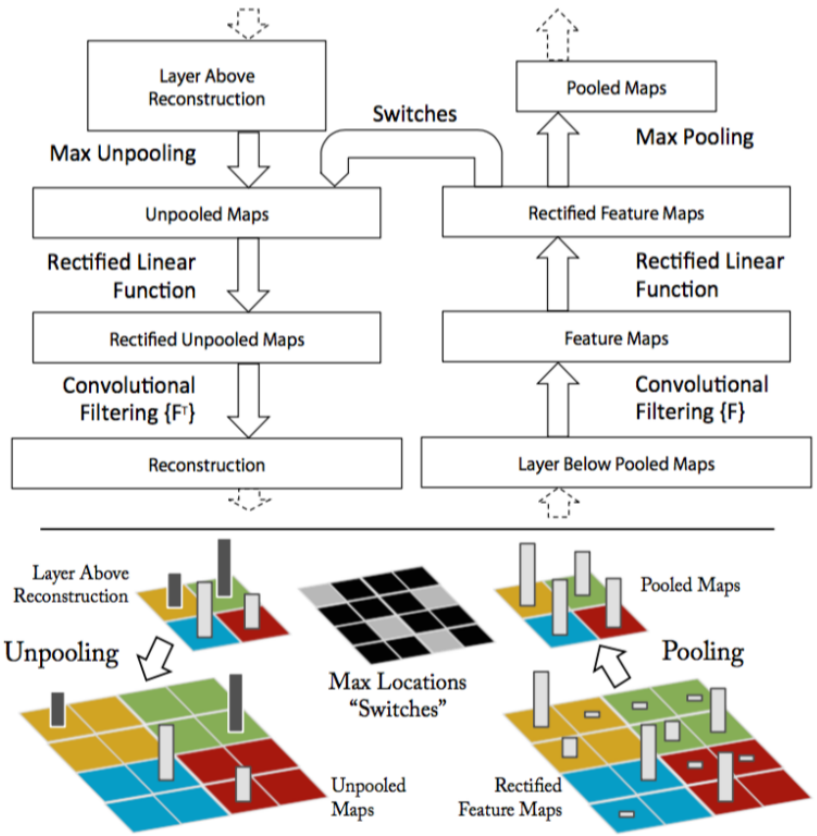
(Zeiler et al) Zerar uma região da imagem e ver como isso afeta o score de classificação.



Azul: região que mais afeta a classificação

Quais elementos da imagem ativam um neurônio ?

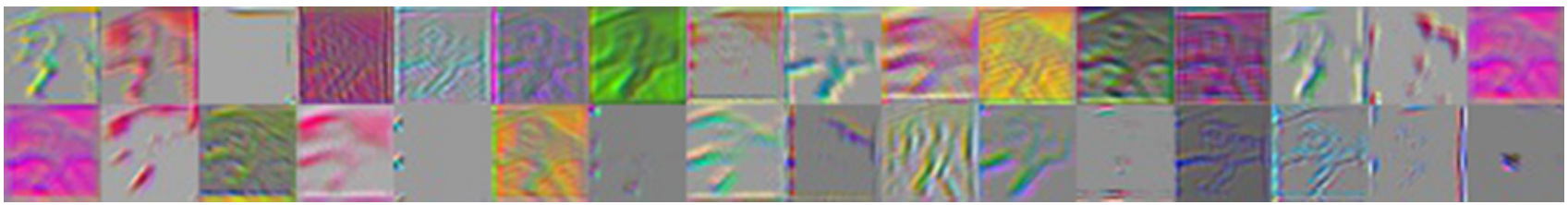
(Zeiler et al.) Passa-se uma imagem pela rede;
Zera-se todos os neurônios na camada, exceto aquele de interesse;
Retropropaga-se o valor do neurônio até a entrada (reconstrói-se a imagem)



Quais elementos da imagem ativam um neurônio ?

Feature map (primeira camada convolucional)

Imagens reconstruídas dos neurônios 1 a 32



(http://kvfrans.com/visualizing-features-from-a-convolutional-neural-network/)

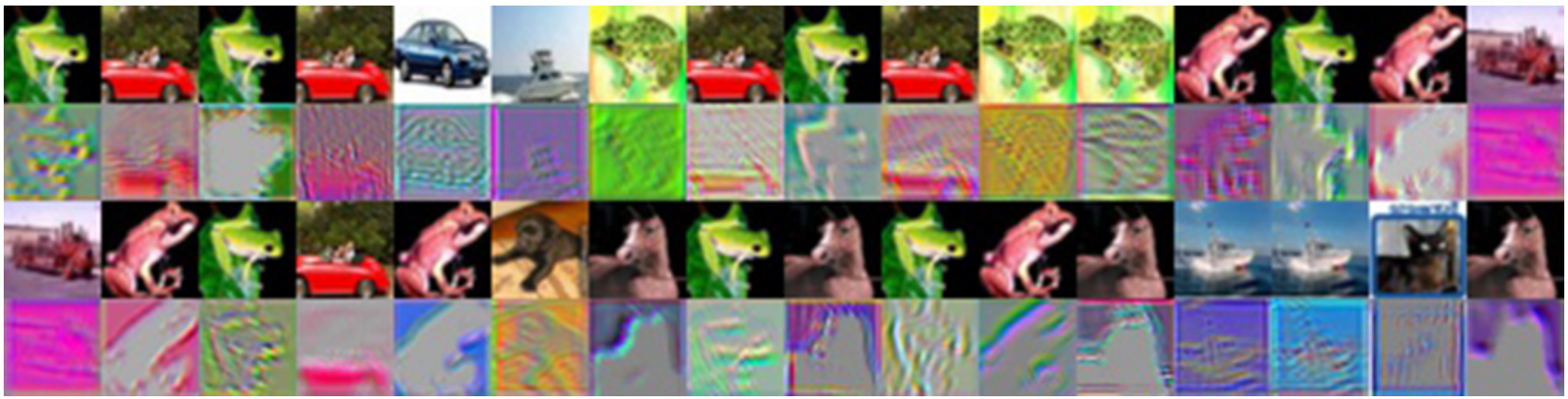
Quais elementos da imagem ativam um neurônio ?

Imagens reconstruídas do neurônio 7, para diferentes imagens

Cor verde ?

Quais elementos da imagem ativam um neurônio ?

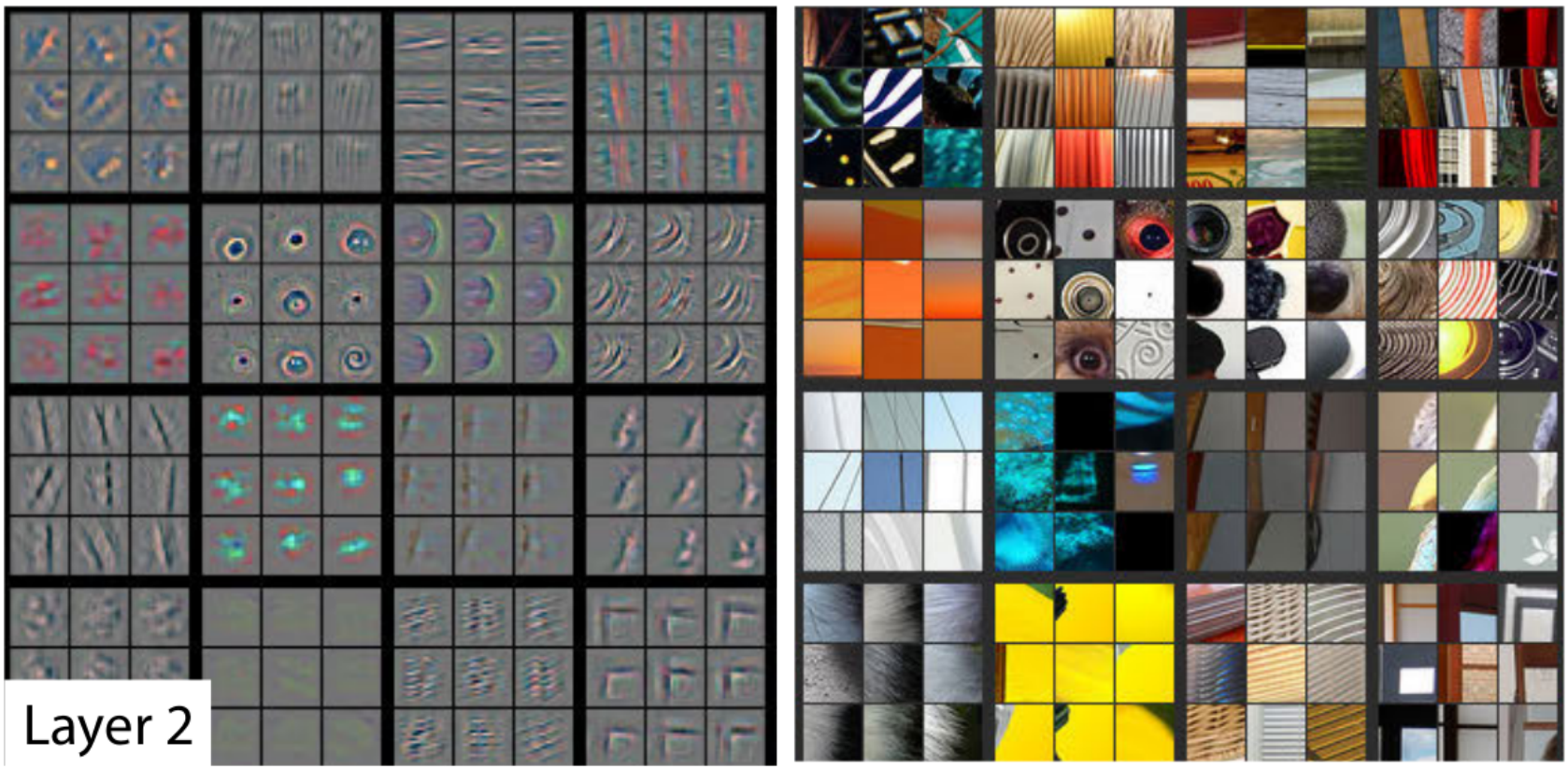
Para cada neurônio, reconstrução da imagem que provocou ativação máxima



Alguns são ativados pela presença de cor apenas.

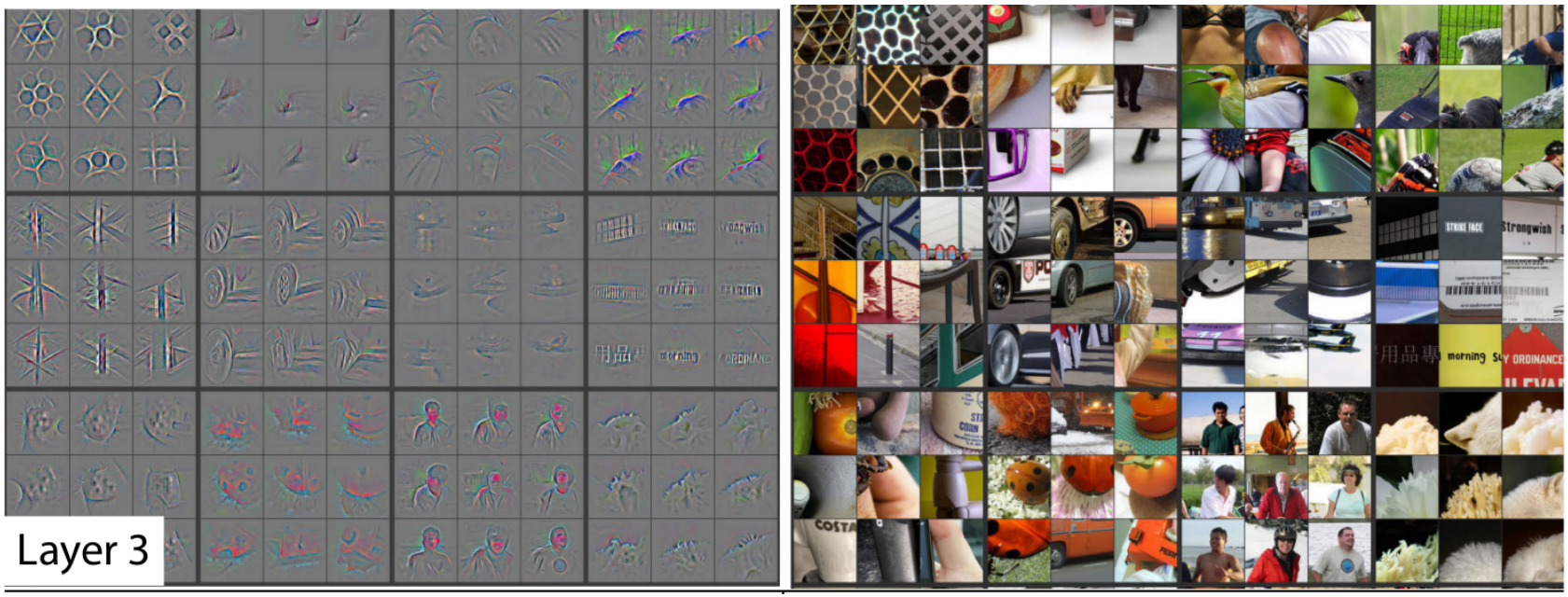
Quais elementos da imagem ativam um neurônio ?

(Zeiler et al.) 16 neurônios quaisquer na camada 2; para cada neurônio, reconstrução das 9 imagens que provocaram as maiores ativações



Quais elementos da imagem ativam um neurônio ?

12 neurônios quaisquer na camada 3; para cada neurônio, reconstrução das 9 imagens que provocaram as maiores ativações



Beyond classification

CNNs desenvolvidas para classificação de imagens passaram a servir como base para outras tarefas

CNN para object detection/recognition

https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

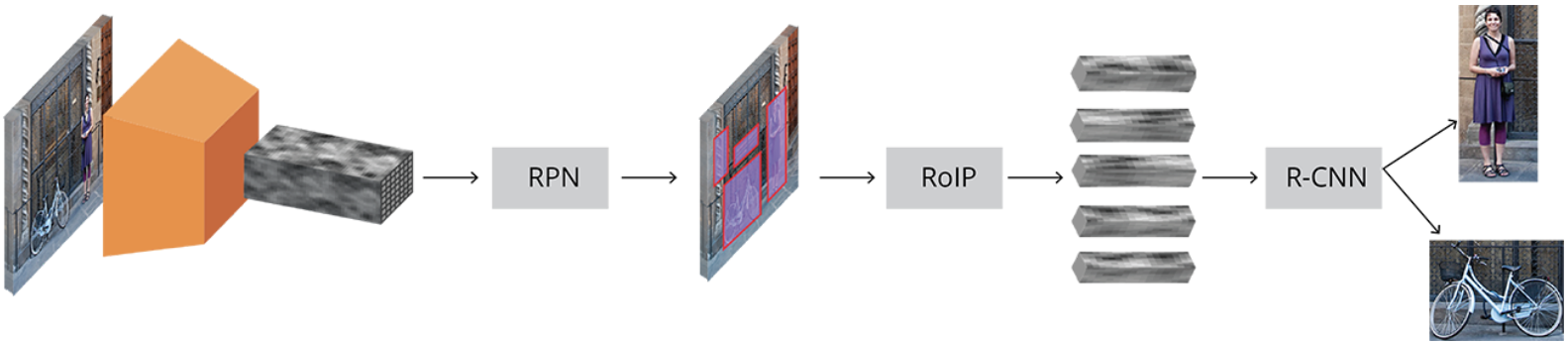
Object recognition em imagens

O conjunto de dados de treinamento consiste de imagens mais uma lista de *bounding box* dos objetos com os respectivos rótulos de classe.

Função objetivo é uma combinação de classificação (acertar rótulos) e regressão (acertar tamanho e localização dos *boxes*).

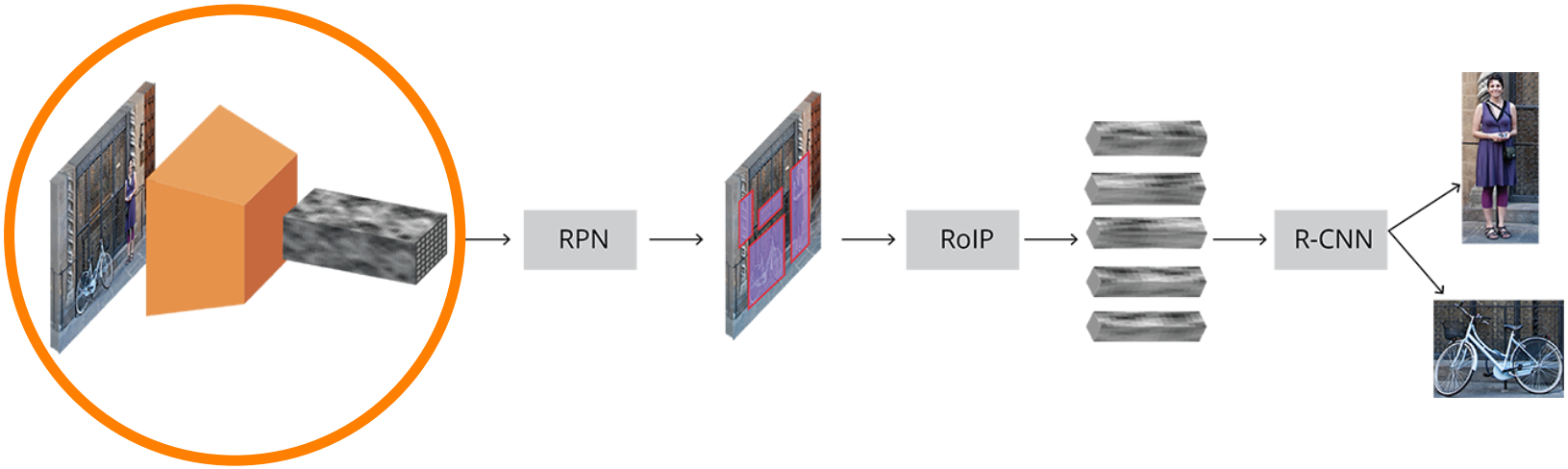
Existem os modelos two-step (ex., Faster R-CNN) e *single shot* (SSD ou YOLO)

Faster R-CNN architecture



https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

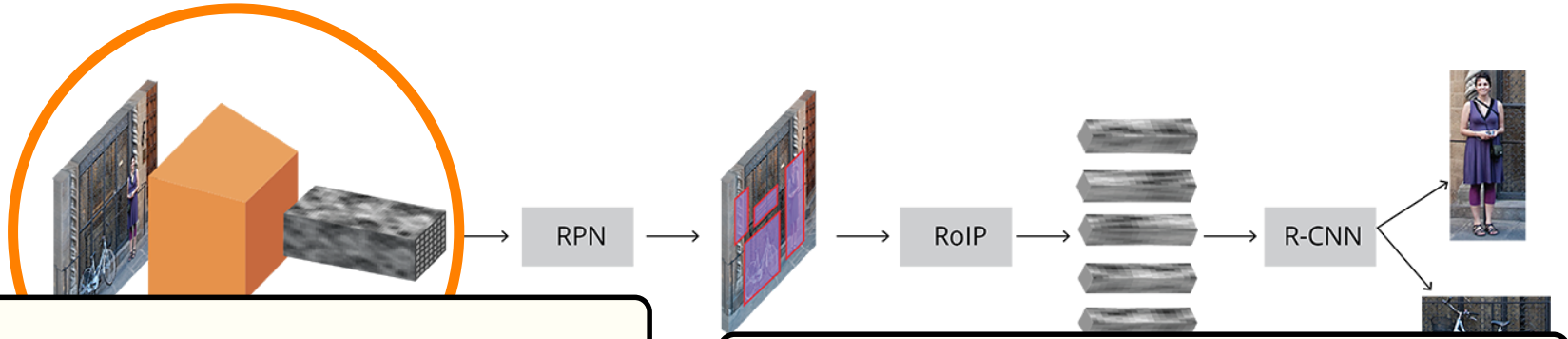
Faster R-CNN architecture



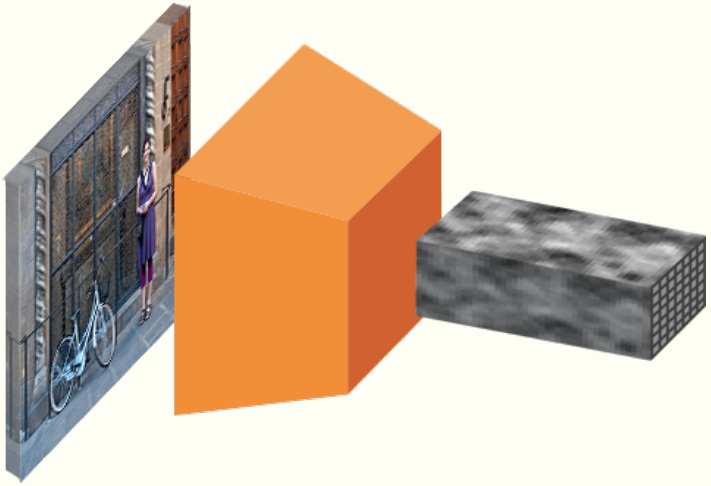
Feature extraction

https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

Faster R-CNN architecture



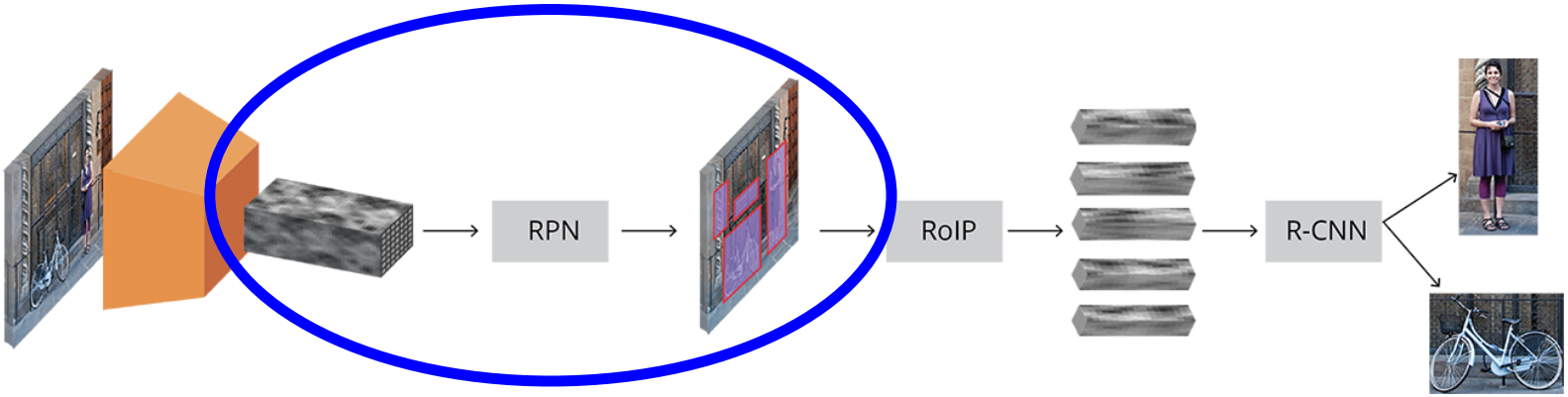
Feature extraction



- Receives an image as input
- Feature extraction part (convolutional and pooling layers) of existing CNNs (VGG, Resnet, ...)

cnn-down-the-rabbit-hole-of-modern-object-detection/

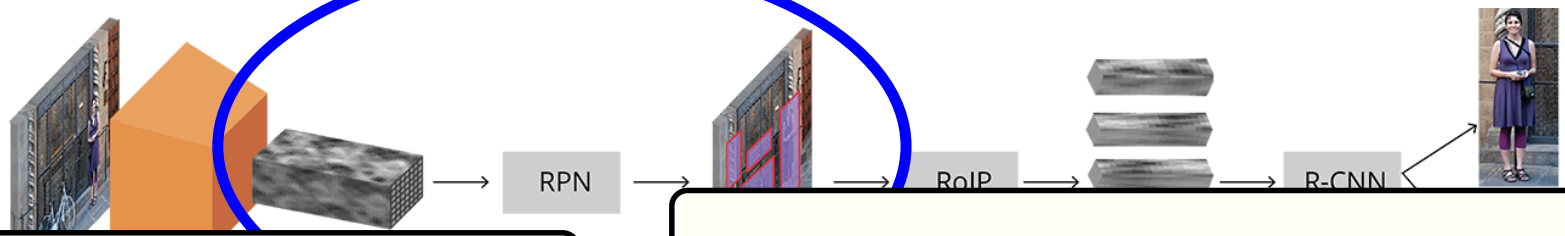
Faster R-CNN architecture

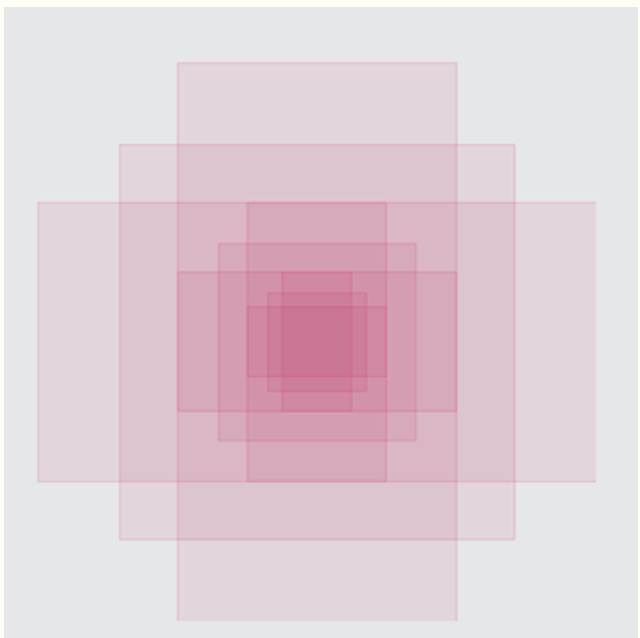


Region proposal network

https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

Faster R-CNN architecture

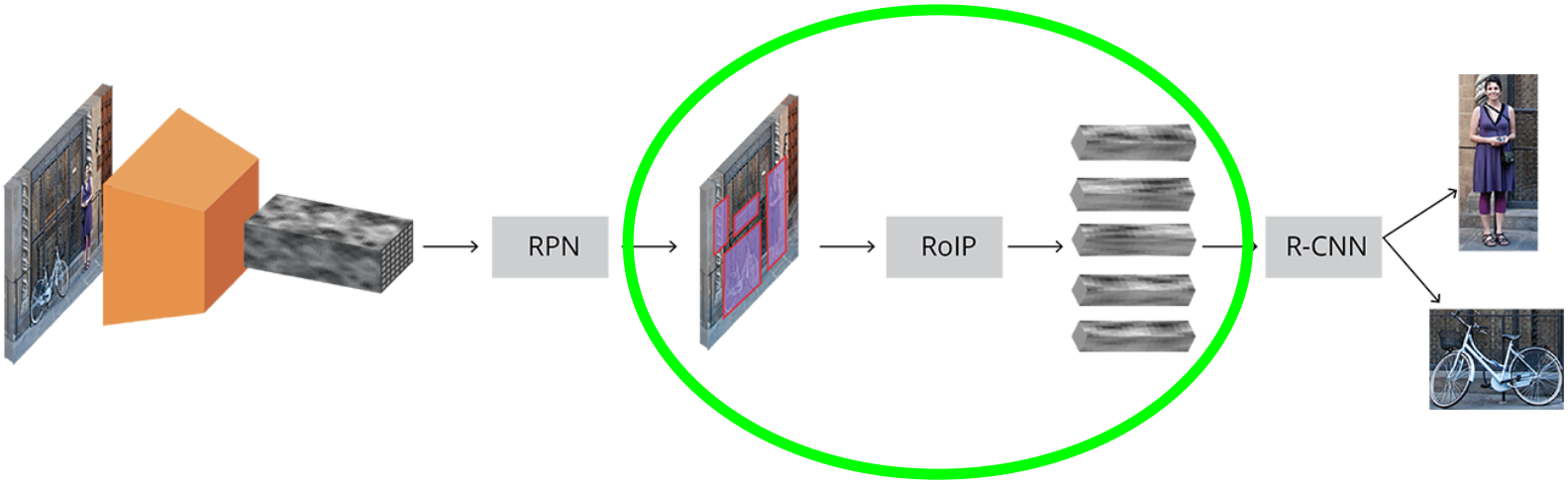




anchors

- Learns if there is an object in the image, in the region enclosed by each of the anchors
- That is done simultaneously over the whole image
- Takes care of the overlaps and returns up to N that are most likely to contain an object

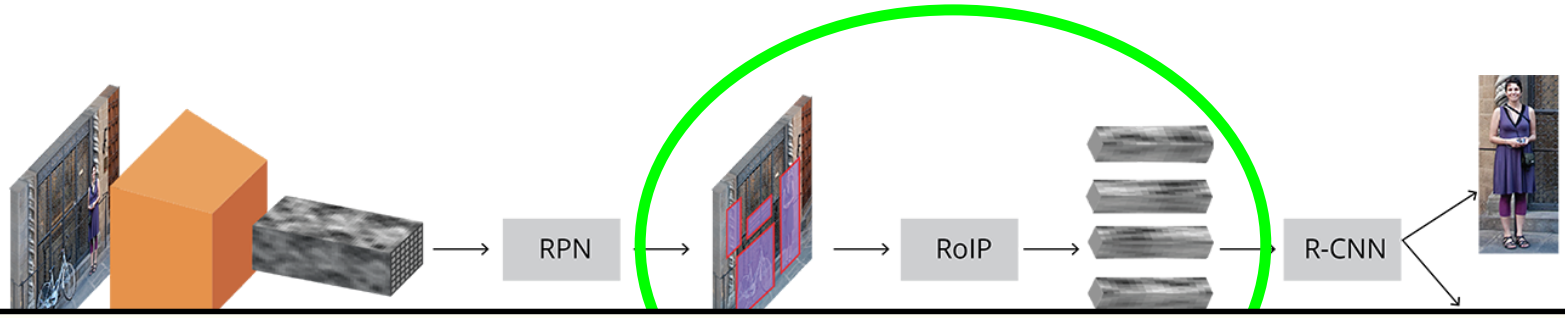
Faster R-CNN architecture



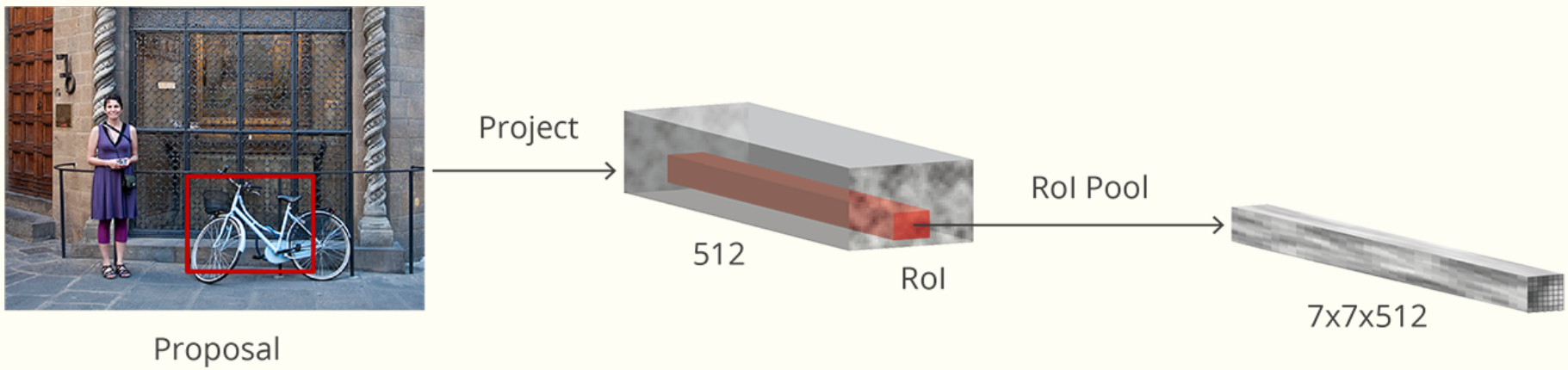
Region of interest pooling

https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

Faster R-CNN architecture

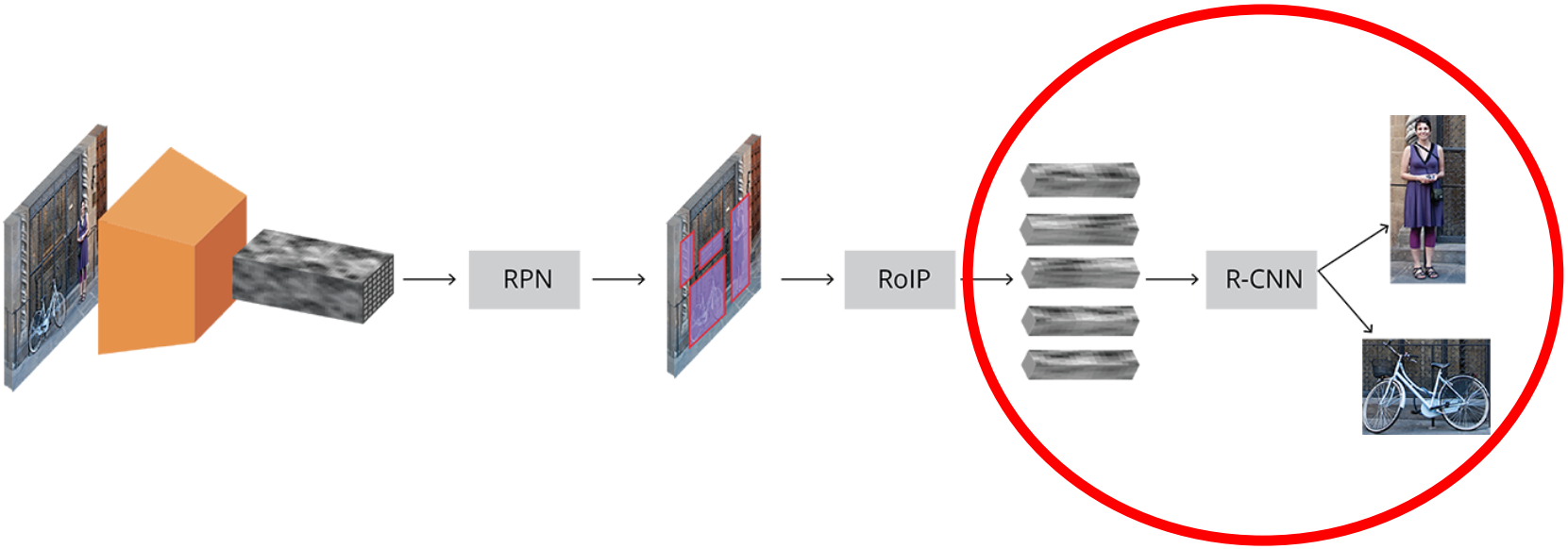


Region of interest pooling



- For each region proposal, crops the corresponding cells from the feature map
  - pools it to a fixed size

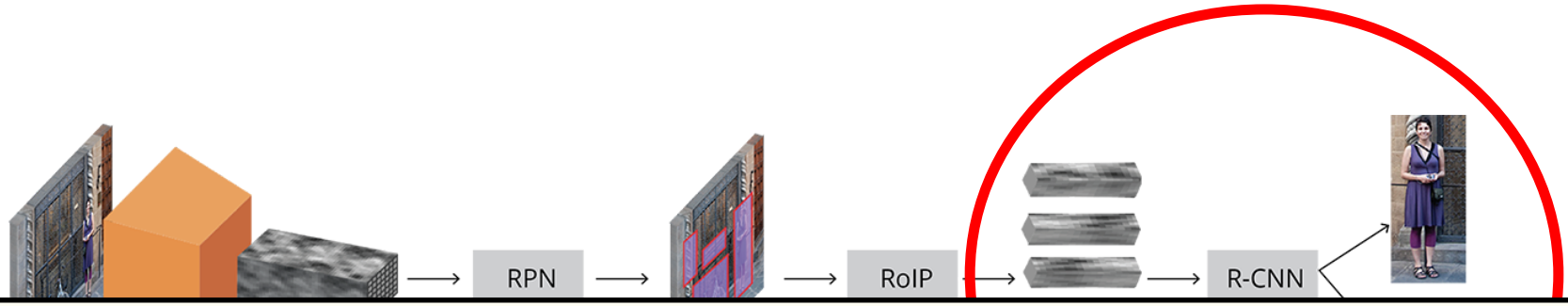
Faster R-CNN architecture



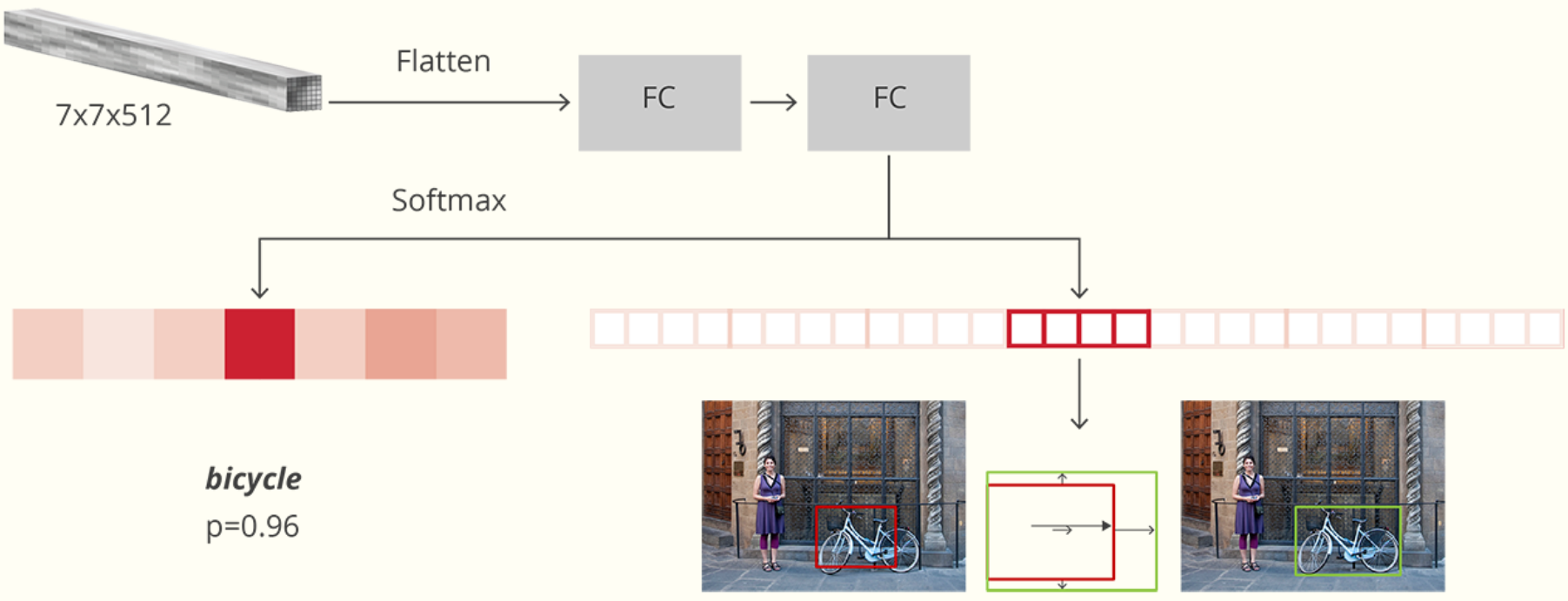
Region classification

https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

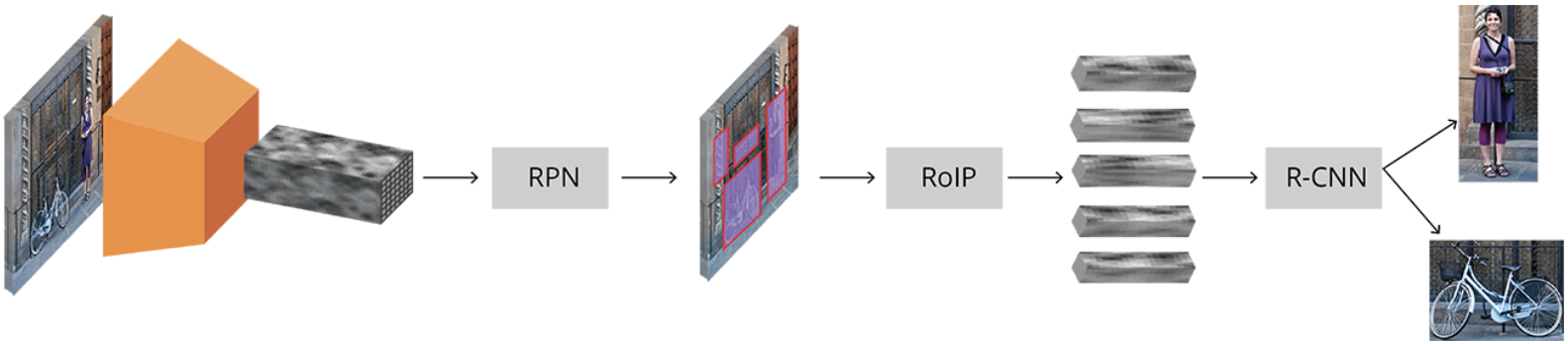
Faster R-CNN architecture



Region classification



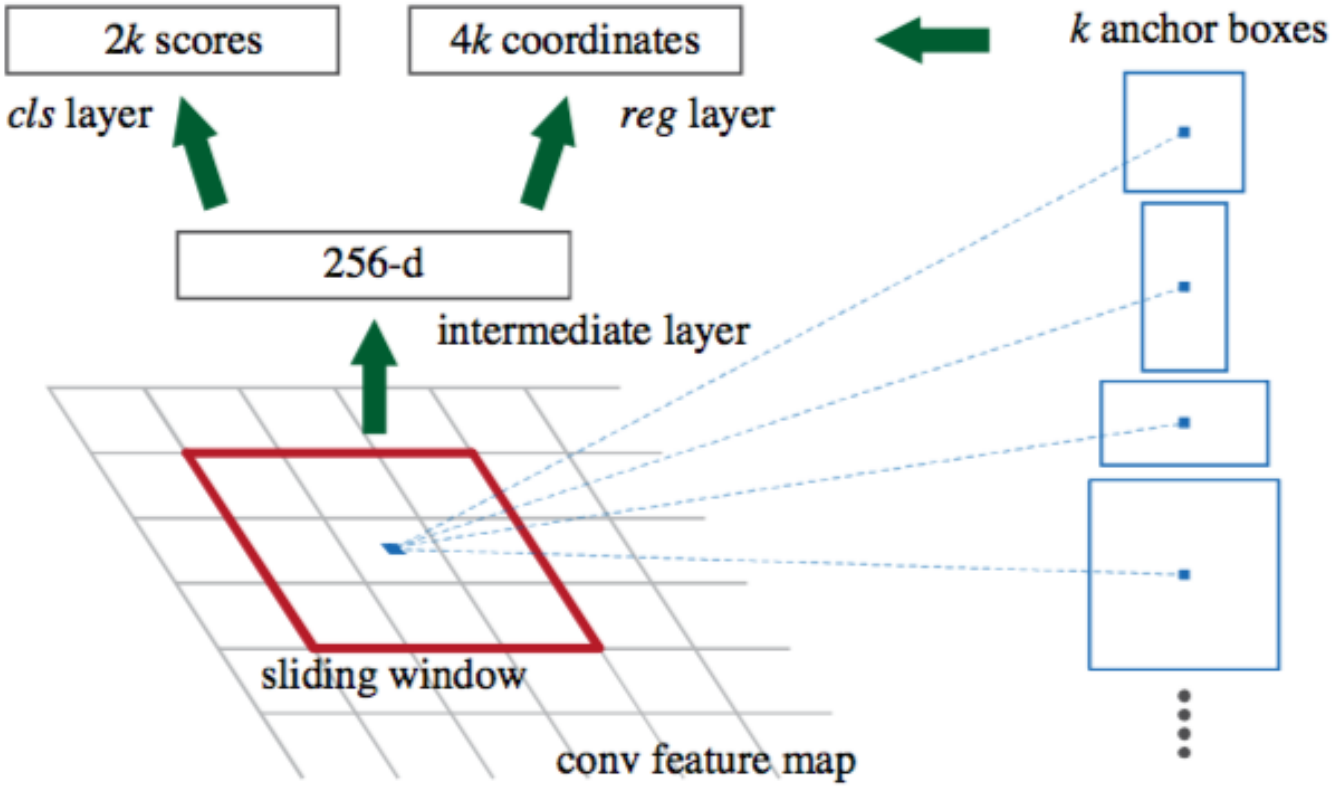
Faster R-CNN architecture

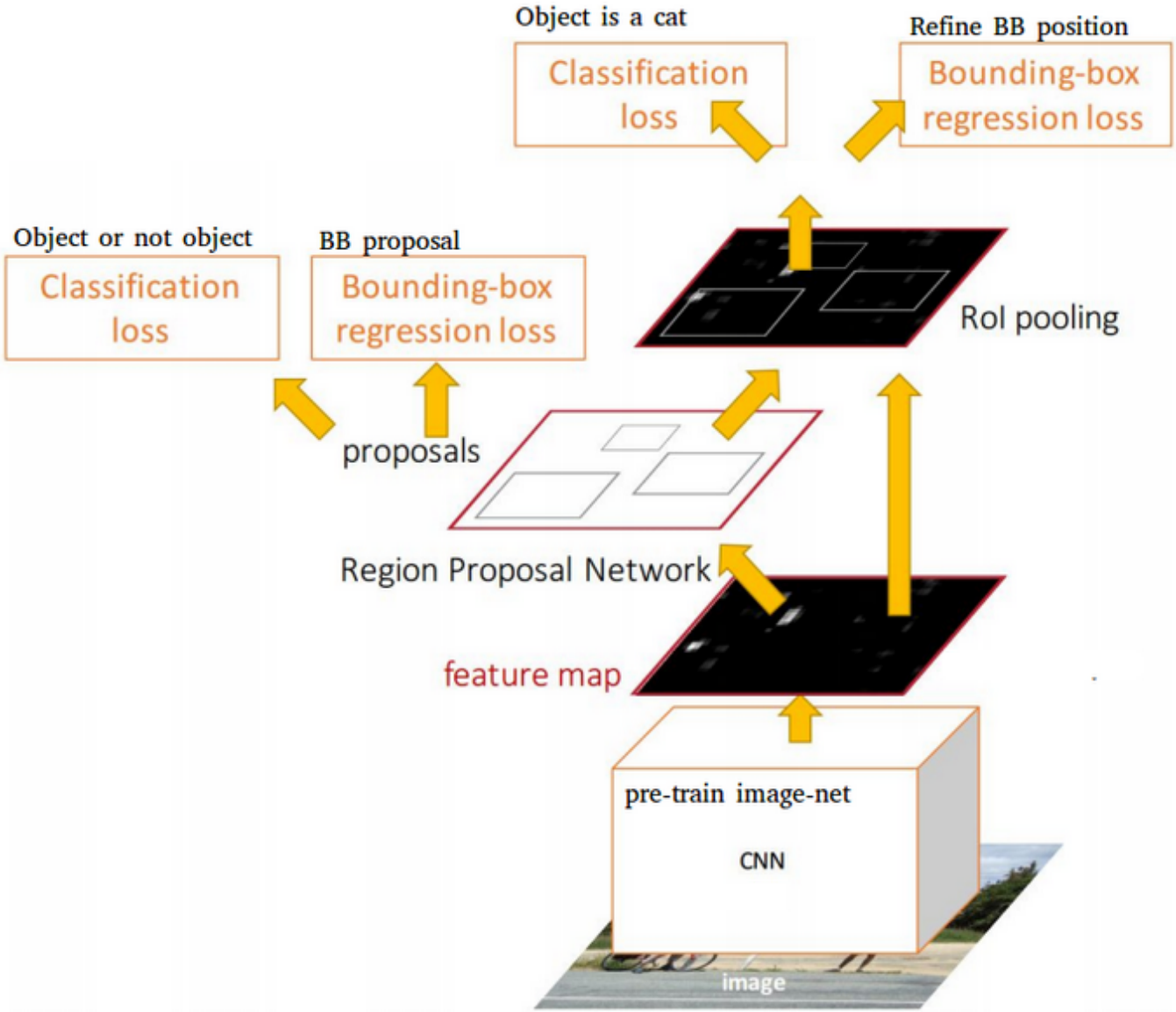


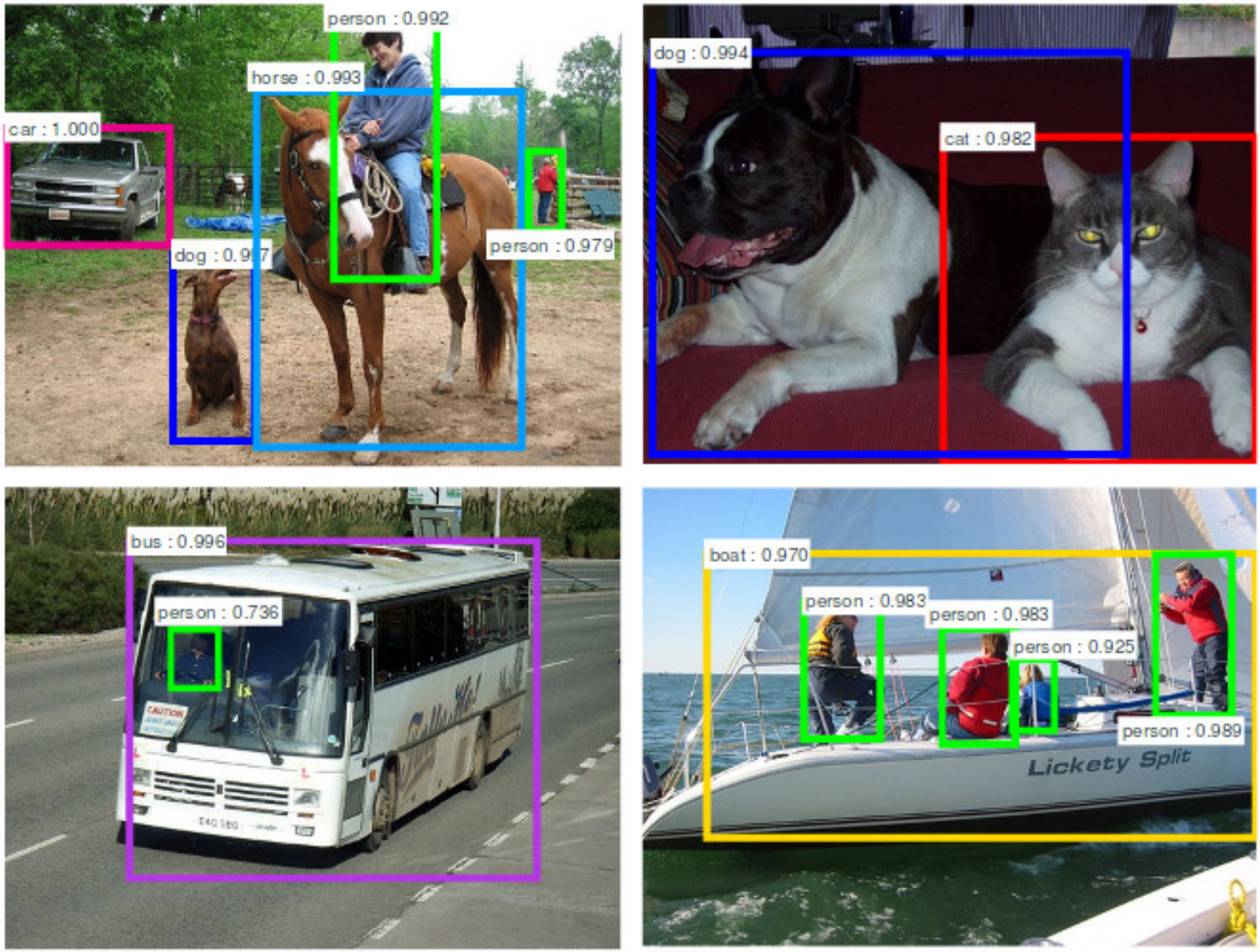
Input is an image

Output is a list of detected objects (bounding box and class probabilities)

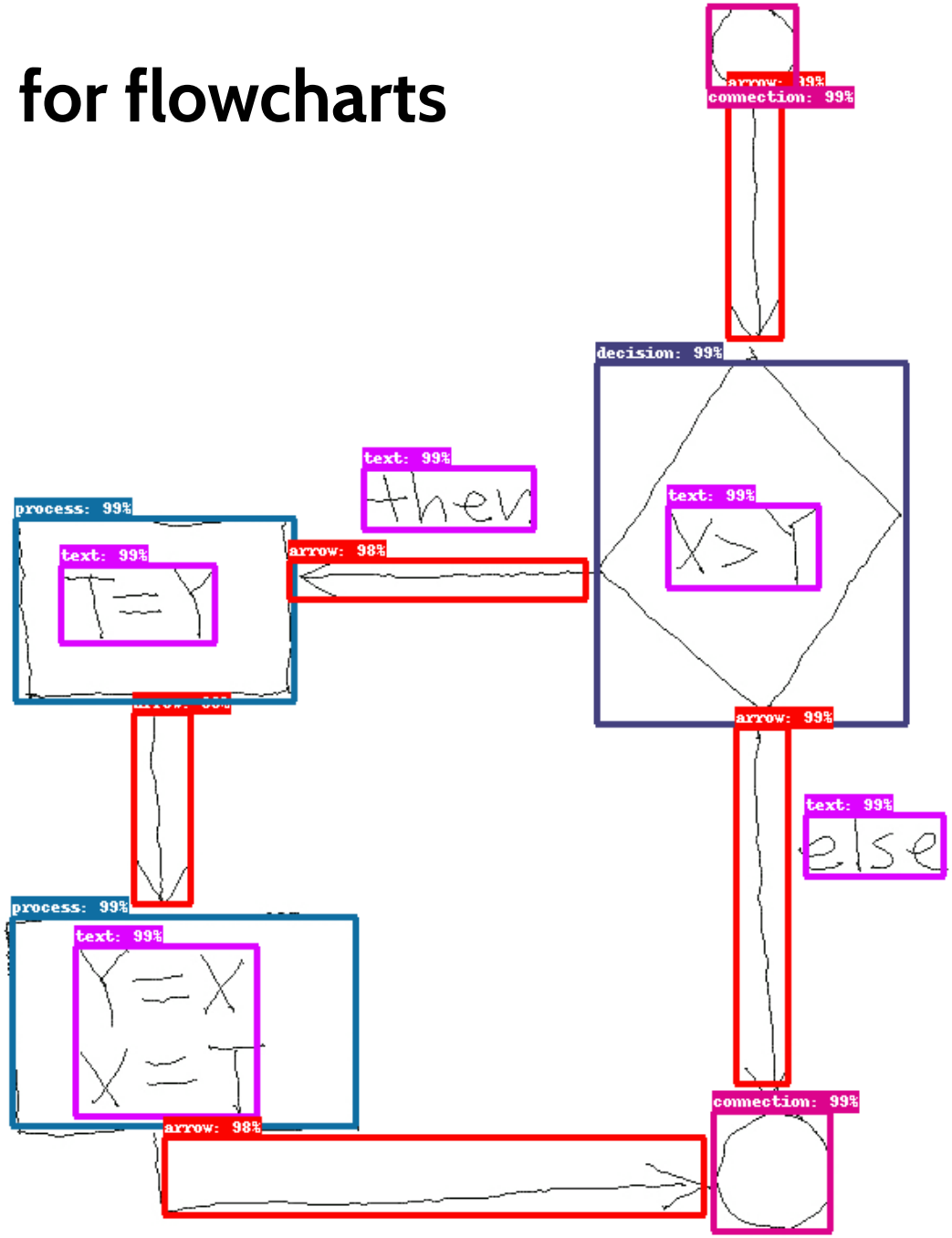
https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

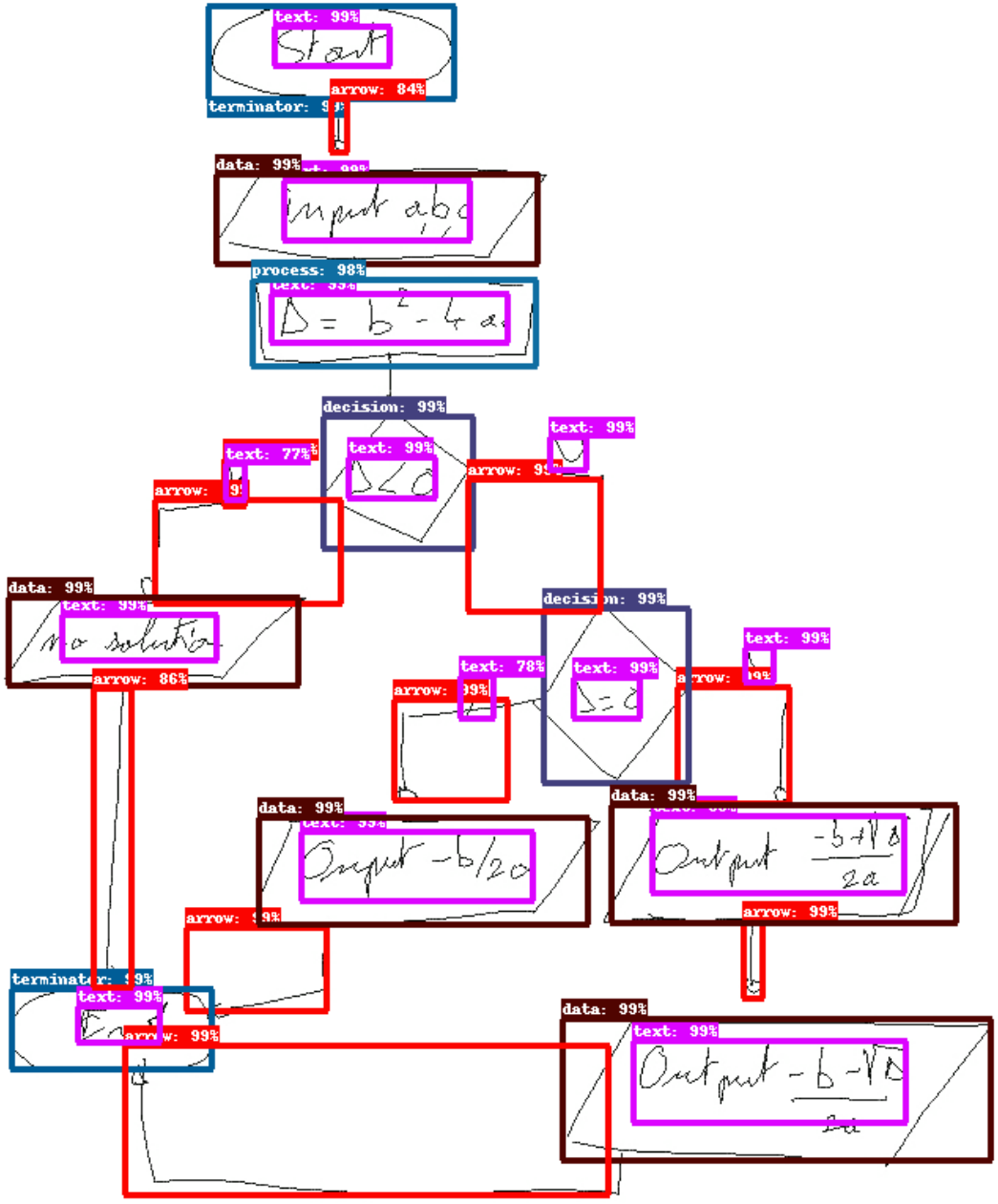

2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

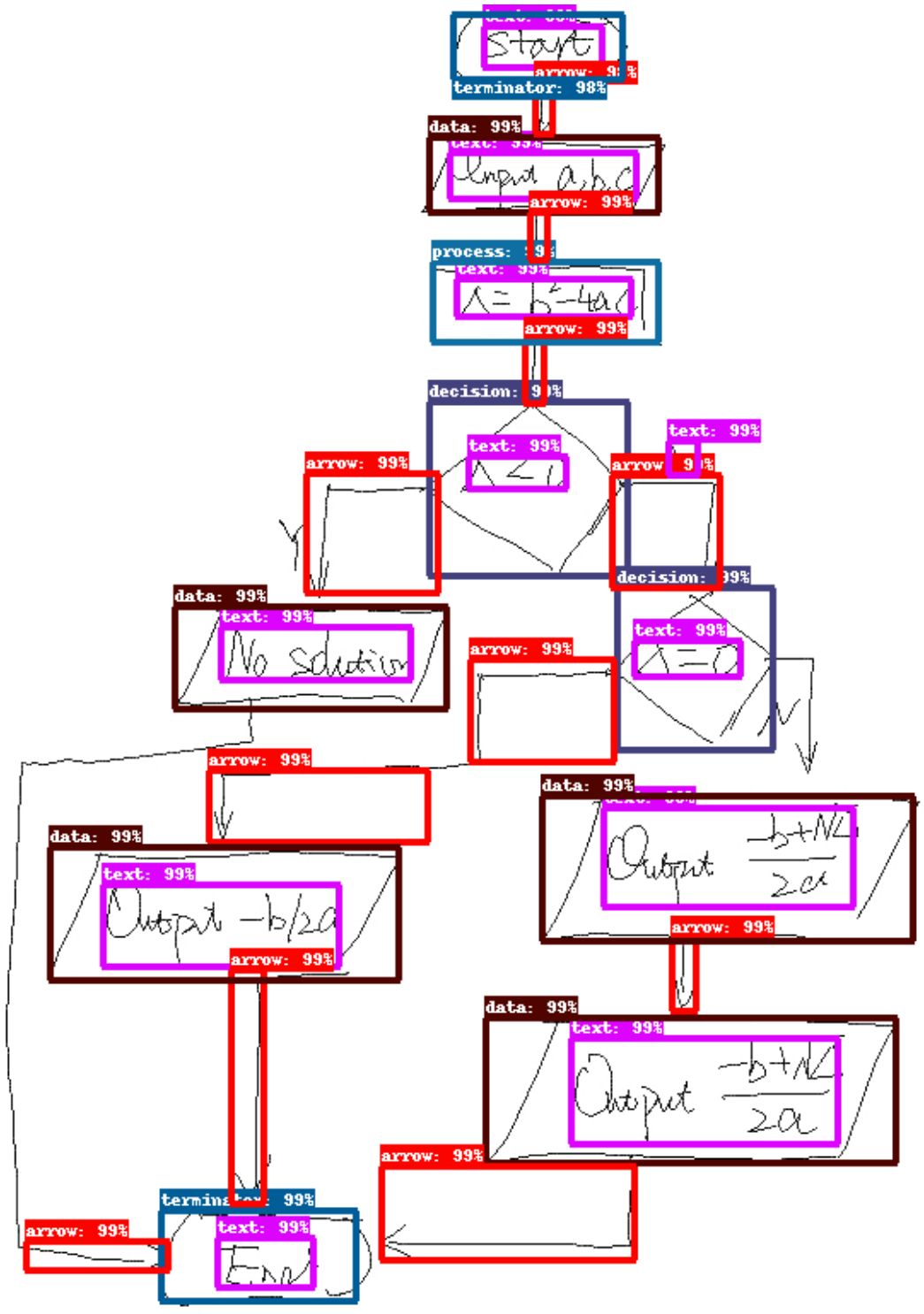

Object is a cat
Classification loss
Refine BB position
Bounding-box regression loss
Object or not object
Classification loss
BB proposal
Bounding-box regression loss
RoI pooling
proposals
Region Proposal Network
feature map
pre-train image-net
CNN
image

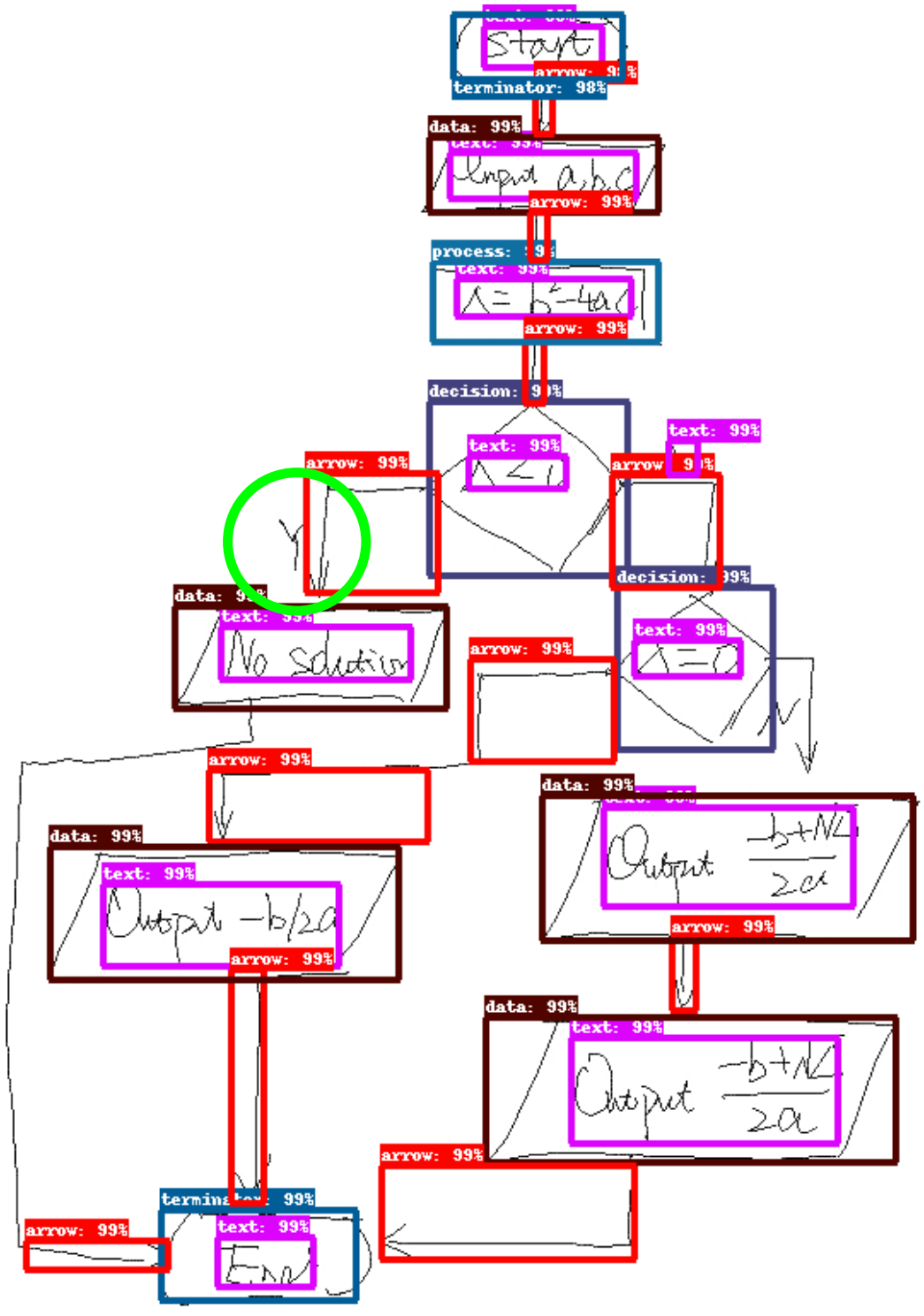

person : 0.992
horse : 0.993
car : 1.000
dog : 0.9
person : 0.979
dog : 0.994
cat : 0.982
bus : 0.996
person : 0.736
boat : 0.970
person : 0.983
person : 0.983
person : 0.925
person : 0.989
Lickety Split

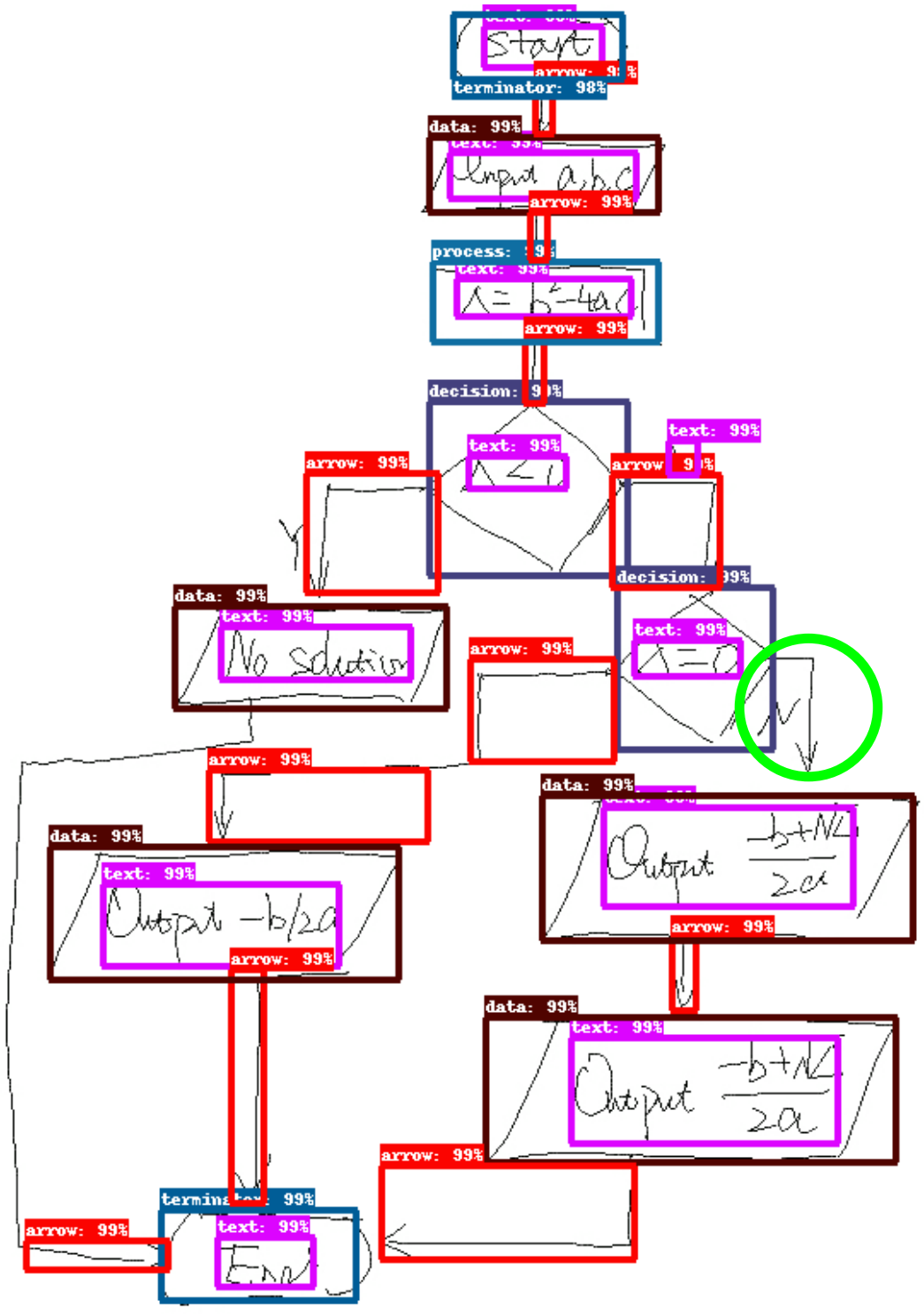
Results for flowcharts

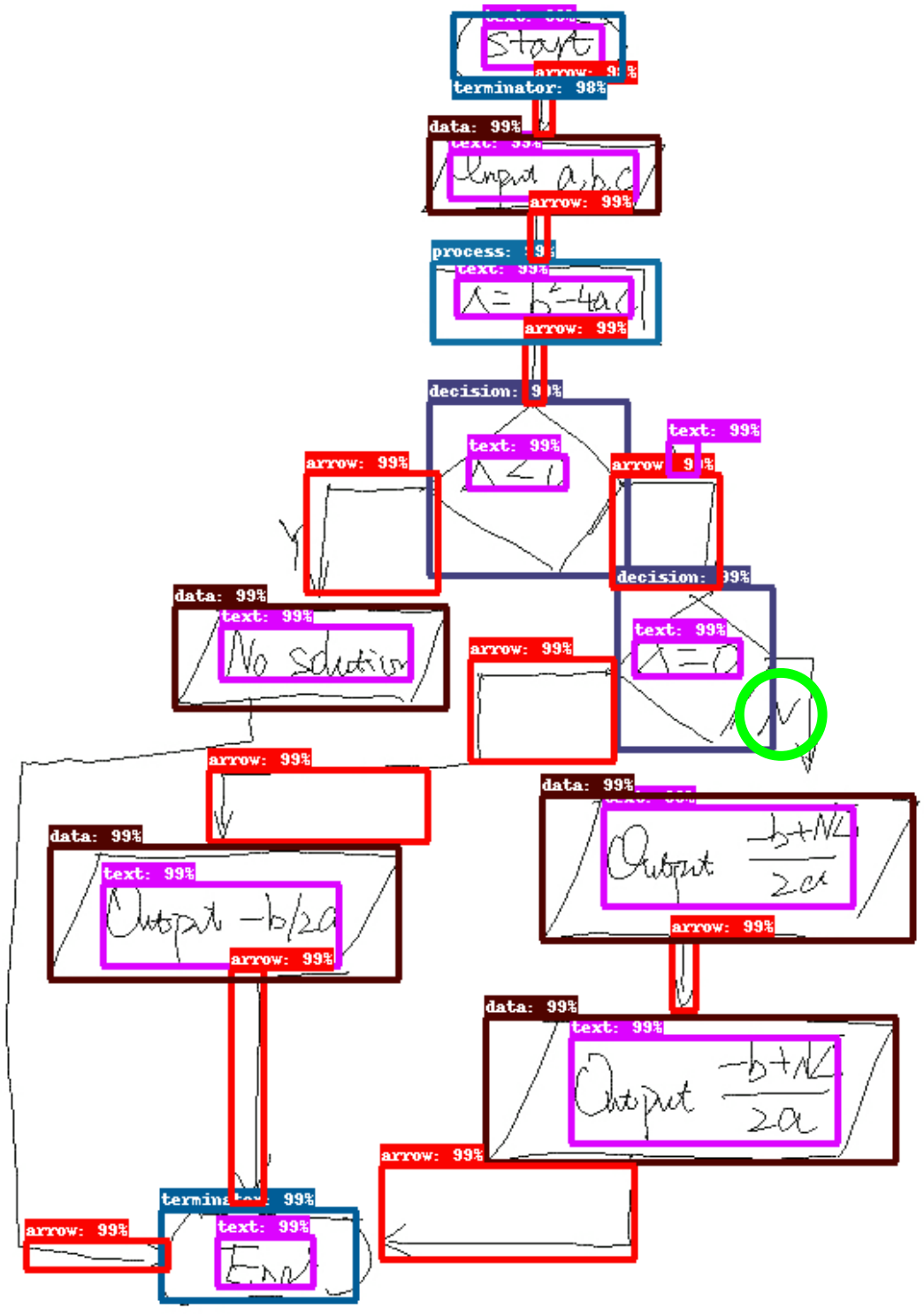


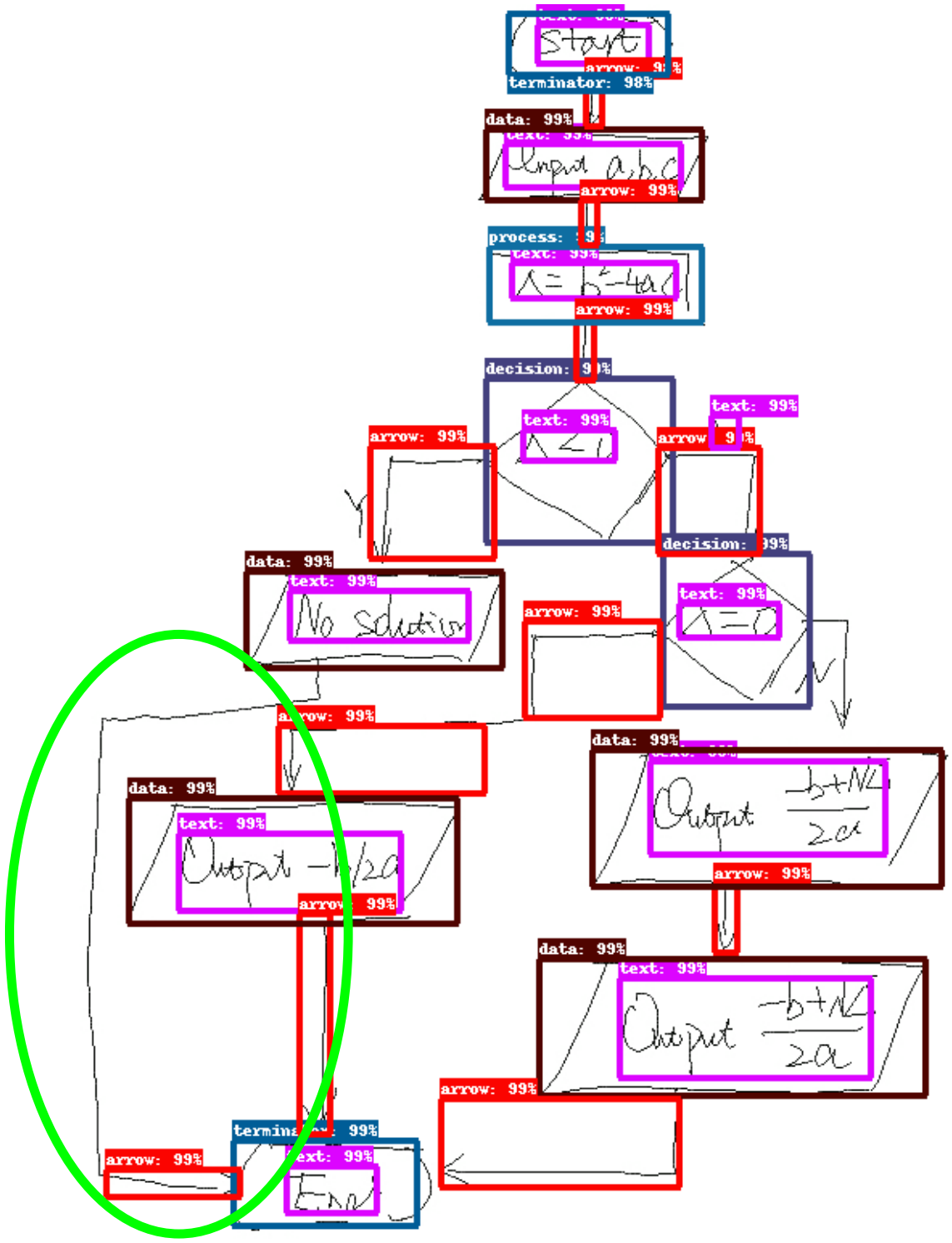

text: 99%
Start
terminator:
arrow: 84%
data: 99%
Input a,b,c
process: 98%
D = b² - 4ac
decision: 99%
text: 99%
D<0
text: 77%
arrow:
text: 99%
arrow:
data: 99%
text: 99%
no solution
arrow: 86%
decision: 99%
text: 78%
arrow:
text: 99%
D=0
text: 99%
arrow:
data: 99%
Output -b/2a
data: 99%
Output (-b+√D)/2a
arrow: 99%
arrow:
terminator:
text: 99%
arrow: 99%
data: 99%
text: 99%
Output (-b-√D)/2a


Start
terminator: 98%
data: 99%
Input a,b,c
arrow: 99%
process: 99%
Δ = b²-4ac
arrow: 99%
decision: 99%
text: 99%
Δ<0
arrow: 99%
text: 99%
arrow: 99%
data: 99%
No Solution
decision: 99%
text: 99%
Δ=0
arrow: 99%
arrow: 99%
data: 99%
Output -b/2a
arrow: 99%
data: 99%
Output (-b+√Δ)/2a
arrow: 99%
data: 99%
Output (-b+√Δ)/2a
arrow: 99%
arrow: 99%
terminator: 99%
End

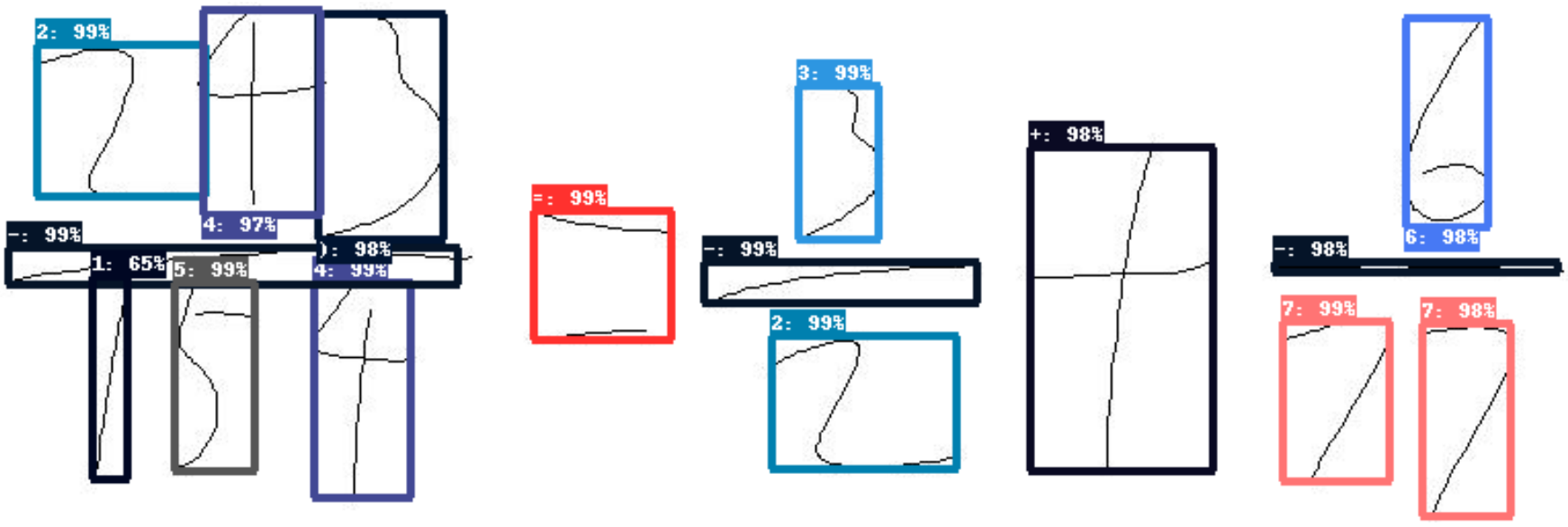

terminator: 98%
data: 99%
arrow: 99%
process: 99%
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
text: 99%
arrow: 99%
decision: 99%
data: 99%
text: 99%
arrow: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
text: 99%

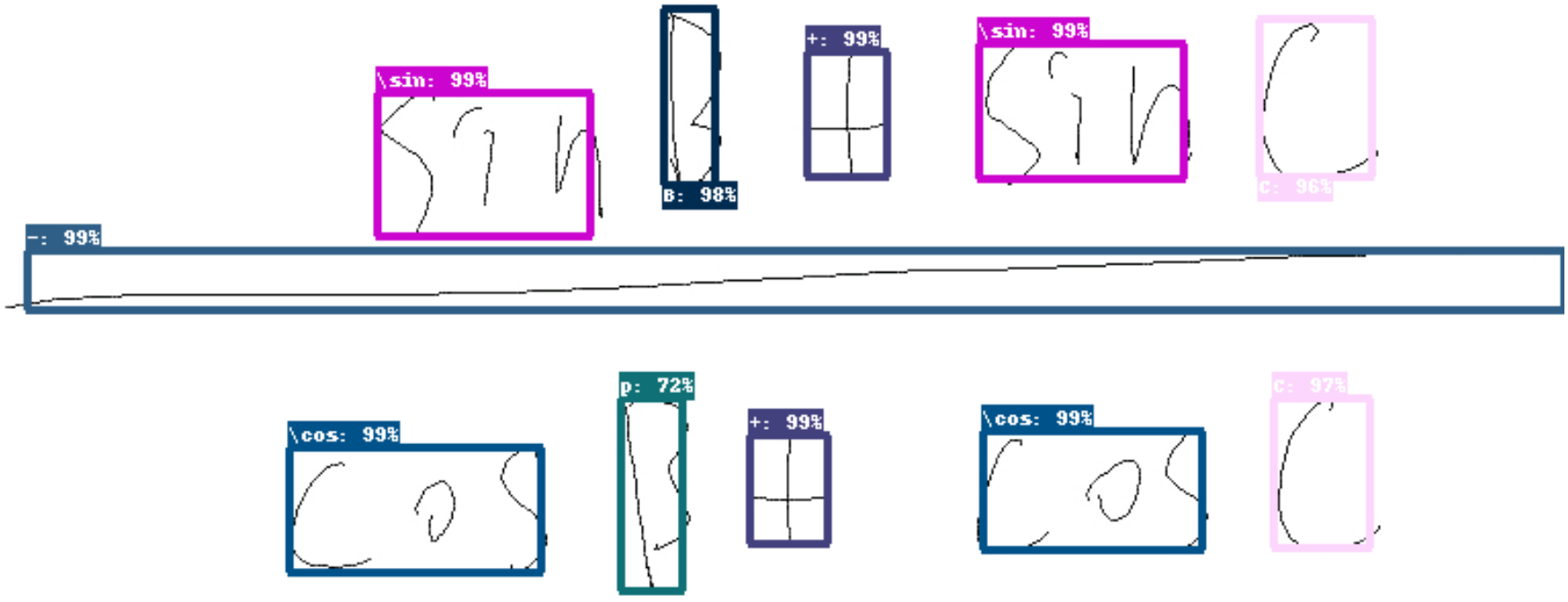

terminator: 98%
data: 99%
arrow: 99%
process: 99%
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
arrow: 99%
text: 99%
data: 99%
text: 99%
No Solution
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
data: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
arrow: 99%
text: 99%

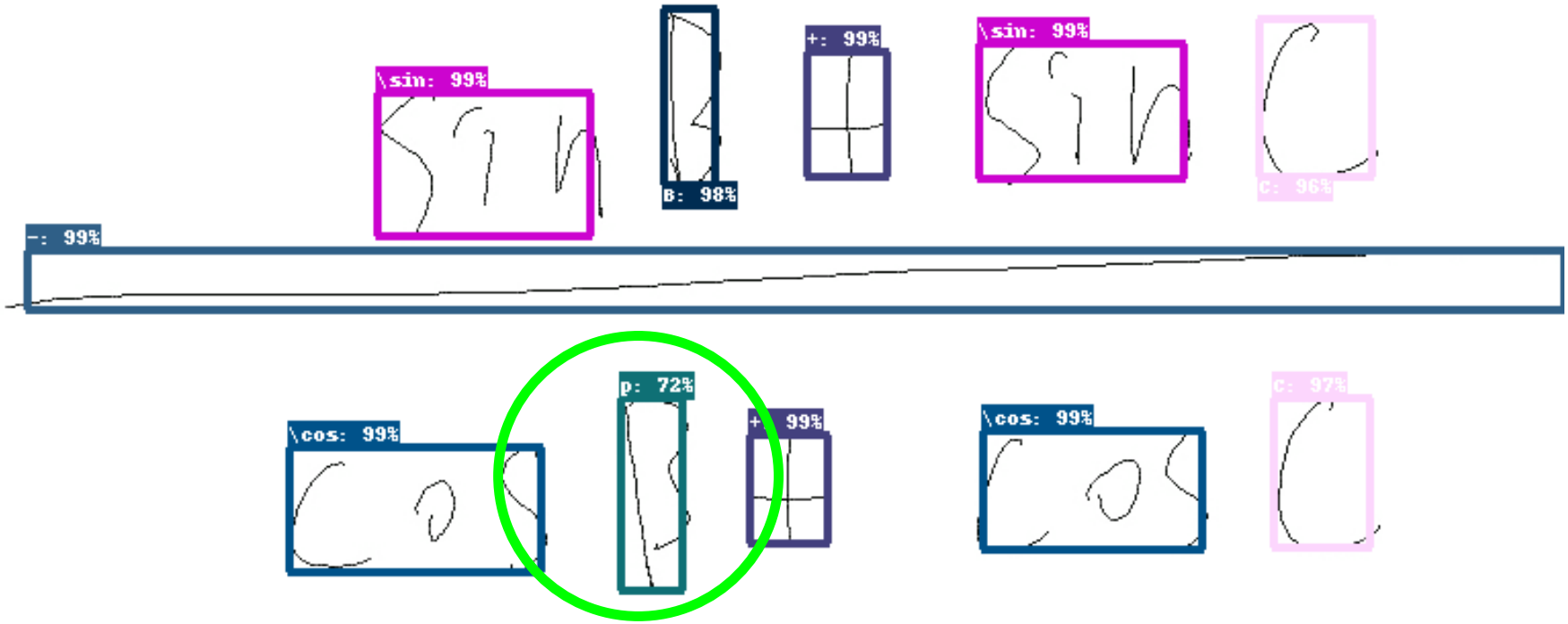

terminator: 98%
data: 99%
arrow: 99%
process: 99%
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
arrow: 99%
text: 99%
data: 99%
text: 99%
No Solution
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
data: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
arrow: 99%
text: 99%

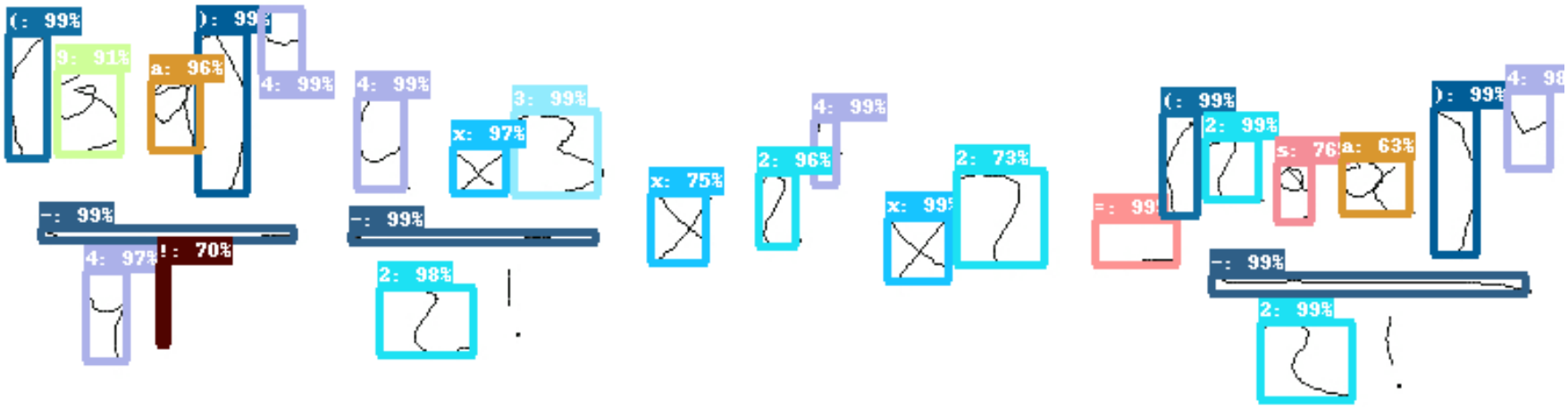

terminator: 98%
data: 99%
arrow: 99%
process: 99%
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
No Solution
arrow: 99%
decision: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
data: 99%
text: 99%
arrow: 99%
arrow: 99%
text: 99%

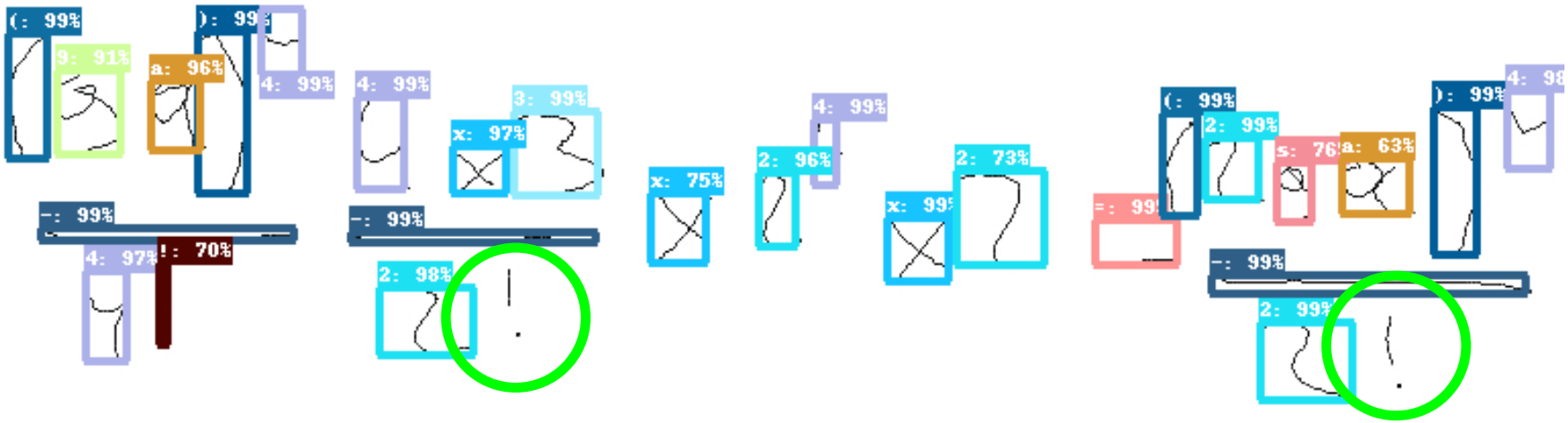
Results for math expressions



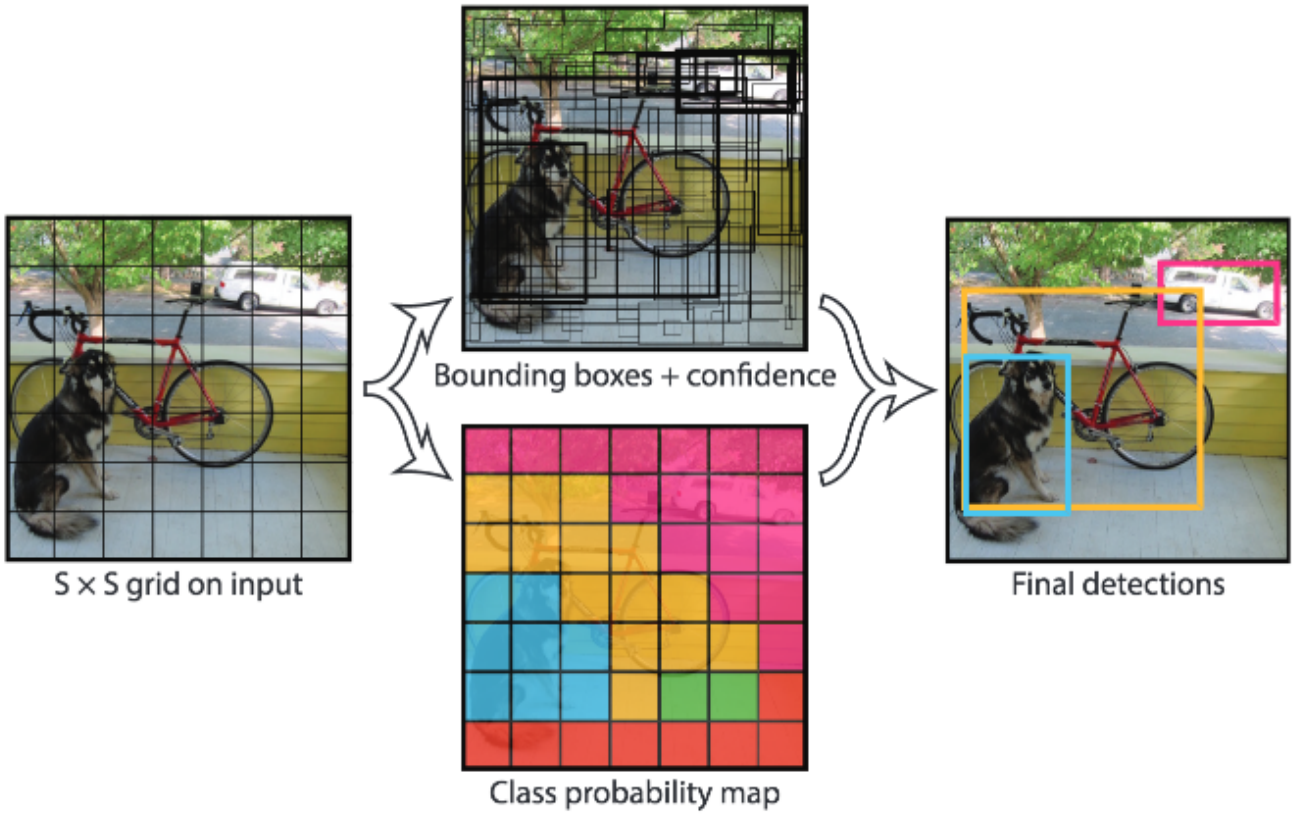

\sin: 99%
B: 98%
+: 99%
\sin: 99%
C: 96%
-: 99%
\cos: 99%
p: 72%
+: 99%
\cos: 99%
C: 97%


\sin: 99%
1: 98%
+: 99%
\sin: 99%
C: 96%
-: 99%
\cos: 99%
p: 72%
+: 99%
\cos: 99%
C: 97%


(: 99%
9: 91%
a: 96%
): 99%
4: 99%
-: 99%
4: 97%
!: 70%
4: 99%
x: 97%
3: 99%
-: 99%
2: 98%
x: 75%
4: 99%
2: 96%
2: 73%
(: 99%
2: 99%
s: 76%
a: 63%
): 99%
4: 98%
-: 99%
2: 99%

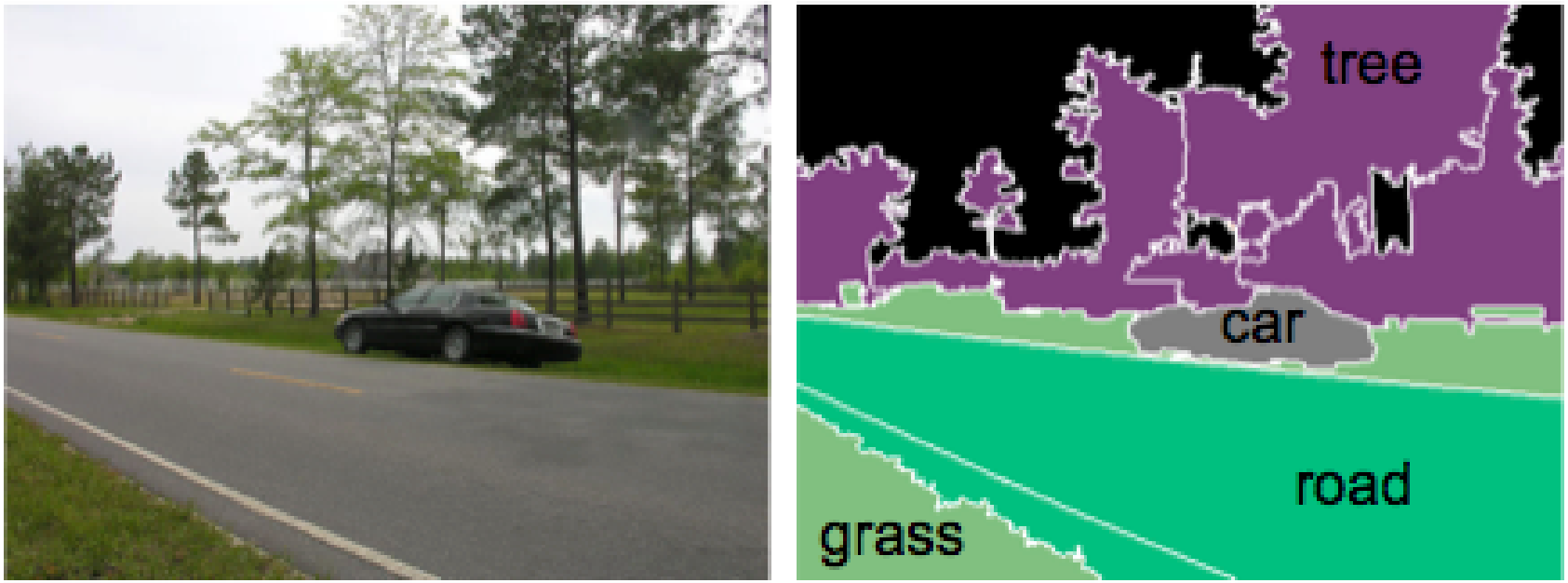

(: 99%
9 : 91%
a : 96%
) : 99%
4 : 99%
4 : 99%
x : 97%
3 : 99%
- : 99%
4 : 97%
! : 70%
- : 99%
2 : 98%
x : 75%
4 : 99%
2 : 96%
x : 99%
2 : 73%
(: 99%
2 : 99%
s : 76%
a : 63%
) : 99%
4 : 98%
- : 99%
2 : 99%

YOLO


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

CNN para image segmentation

Mais: http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review


tree
car
road
grass

Segmentação de imagens

Objetivo: Delinear o contorno dos objetos/componentes de interesse presentes na imagem (ou atribuir um rótulo único para todos os *pixels* de um objeto)

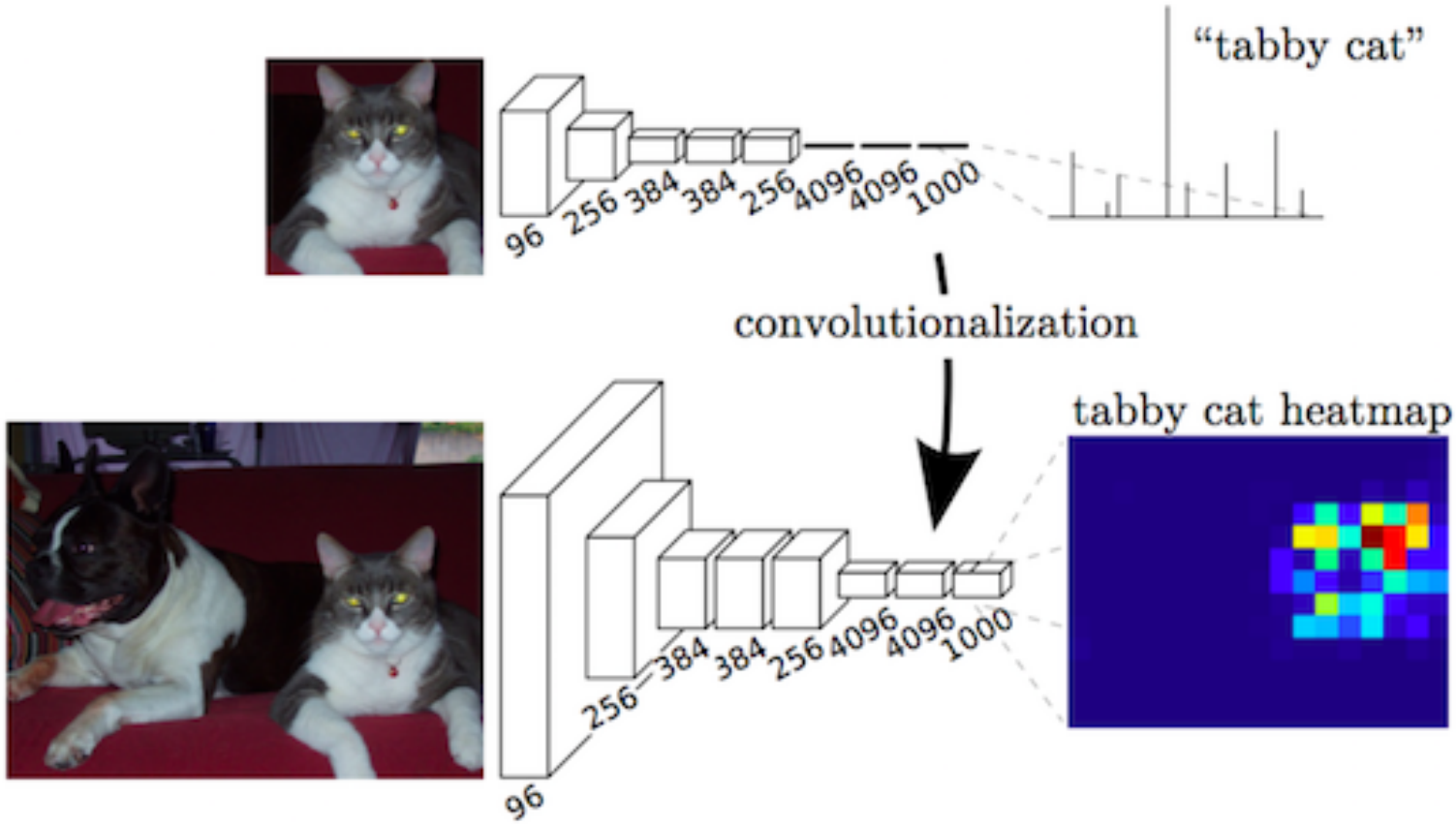
A arquitetura das redes segue a forma *encoder-decoder*

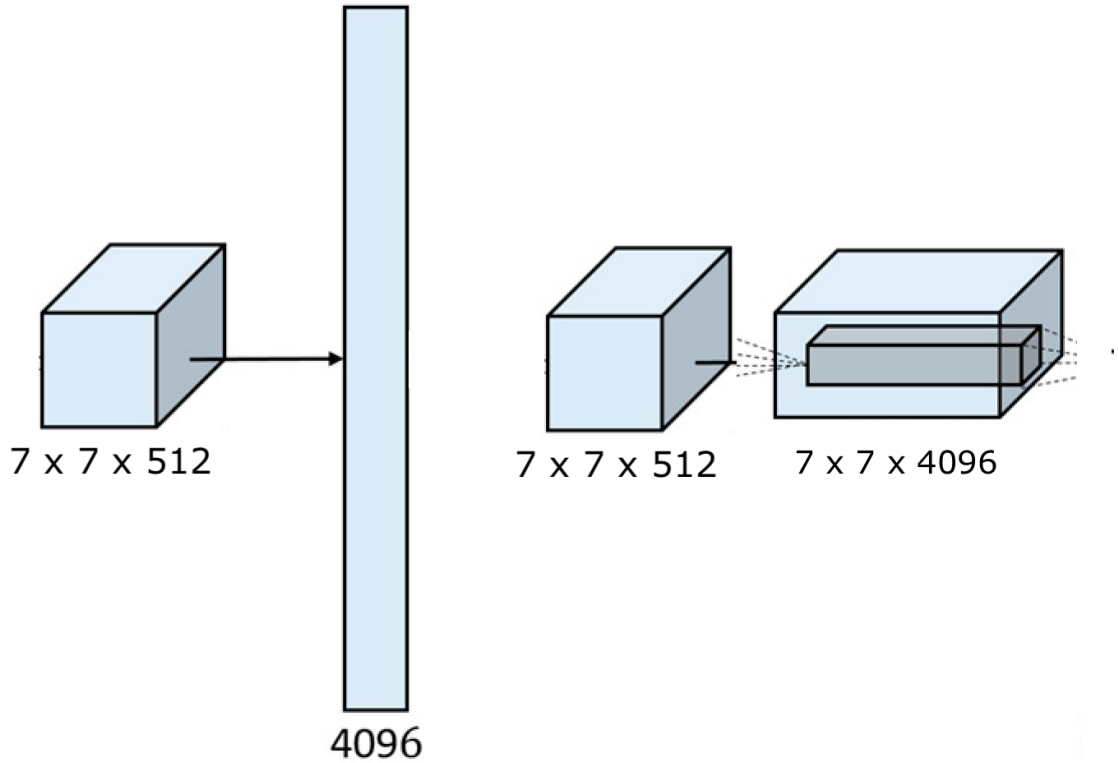
A parte *encoder* em geral é a parte convolucional de uma CNN.

A parte *decoder* busca restaurar a resolução inicial.

FCN – Fully convolutional network

First *end-to-end* network for semantic segmentation





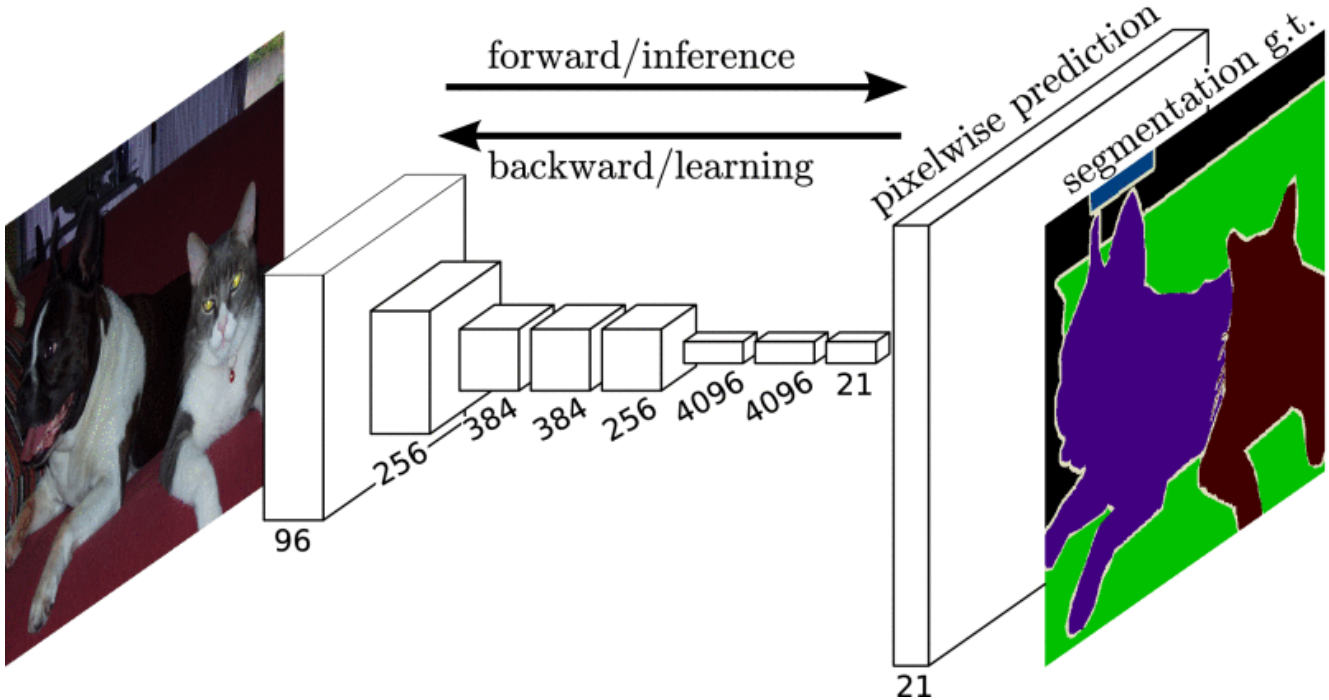
CONV followed by FC Equivalent CONV followed by CONV

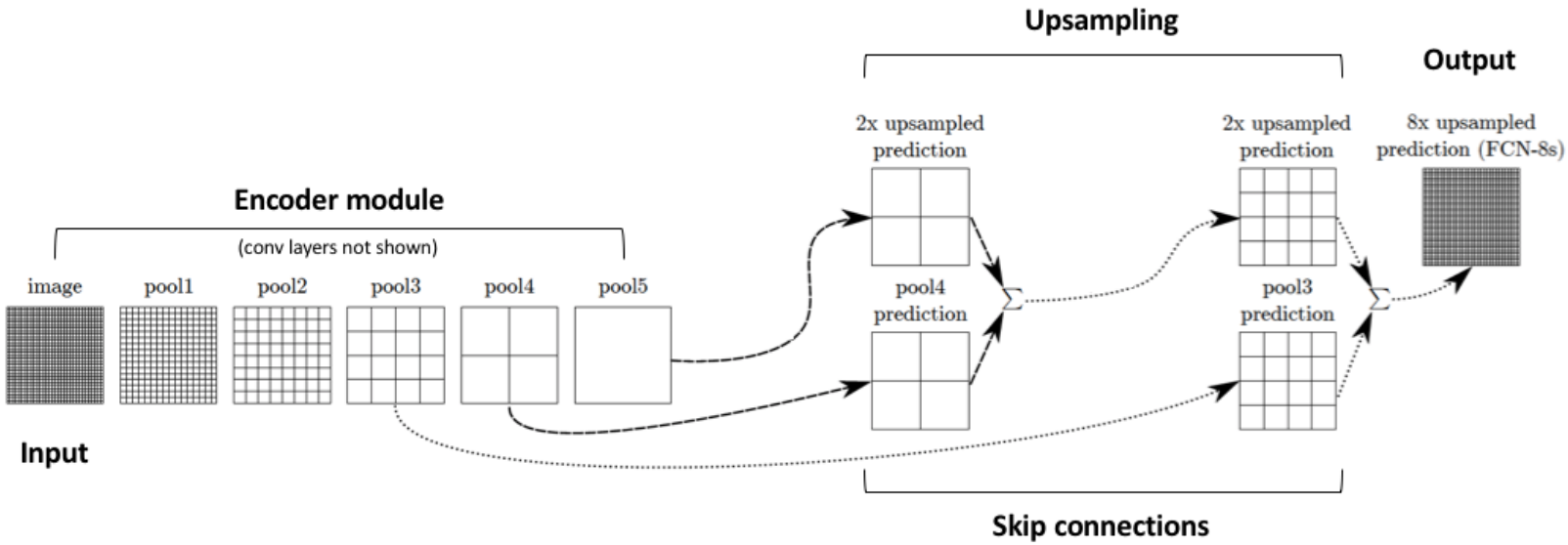
http://cs231n.github.io/convolutional-networks/#convert

Example: Last feature map 7x7x512 + two fully connected layers with 1024 nodes + output layer with 1000 nodes

Replace the first FC layer that looks at [7x7x512] volume with a CONV layer that uses filter size F=7, giving output volume [1x1x4096].
Replace the second FC layer with a CONV layer that uses filter size F=1, giving output volume [1x1x4096].

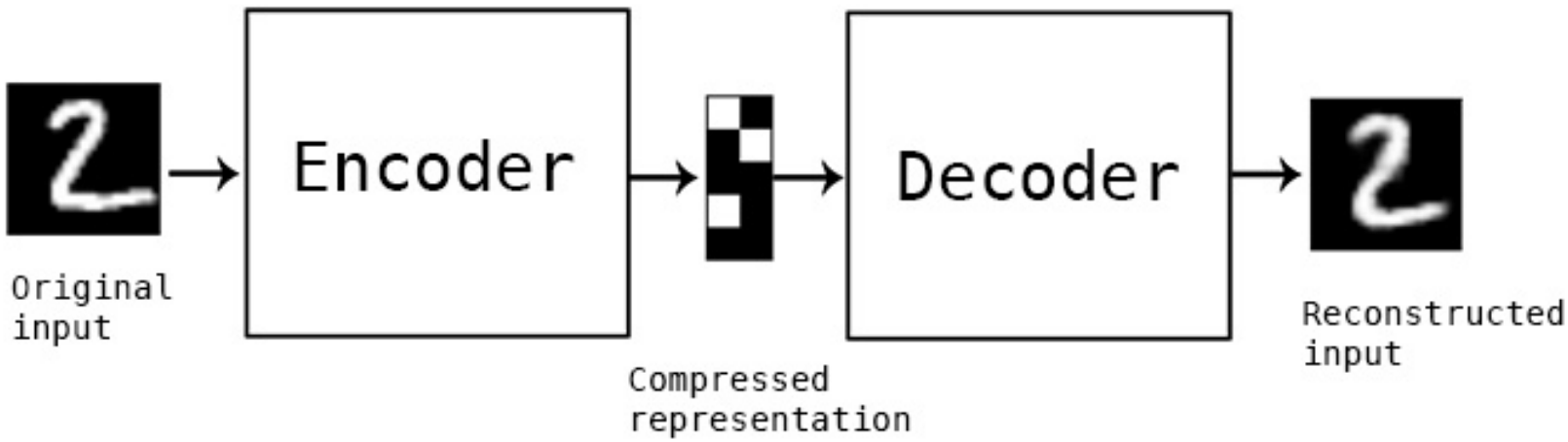
Replace the last FC layer similarly, with F=1, giving final output [1x1x1000]


forward/inference
backward/learning
pixelwise prediction
segmentation g.t.
96
256
384
384
256
4096
4096
21
21


Upsampling
Output
2x upsampled prediction
2x upsampled prediction
8x upsampled prediction (FCN-8s)
Encoder module
(conv layers not shown)
image
pool1
pool2
pool3
pool4
pool5
pool4 prediction
pool3 prediction
Input
Skip connections

Encoder-decoder schema

A maioria das redes imagem-a-imagem segue a estrutura encoder-decoder, que é um caso geral do autoencoder



U-Net

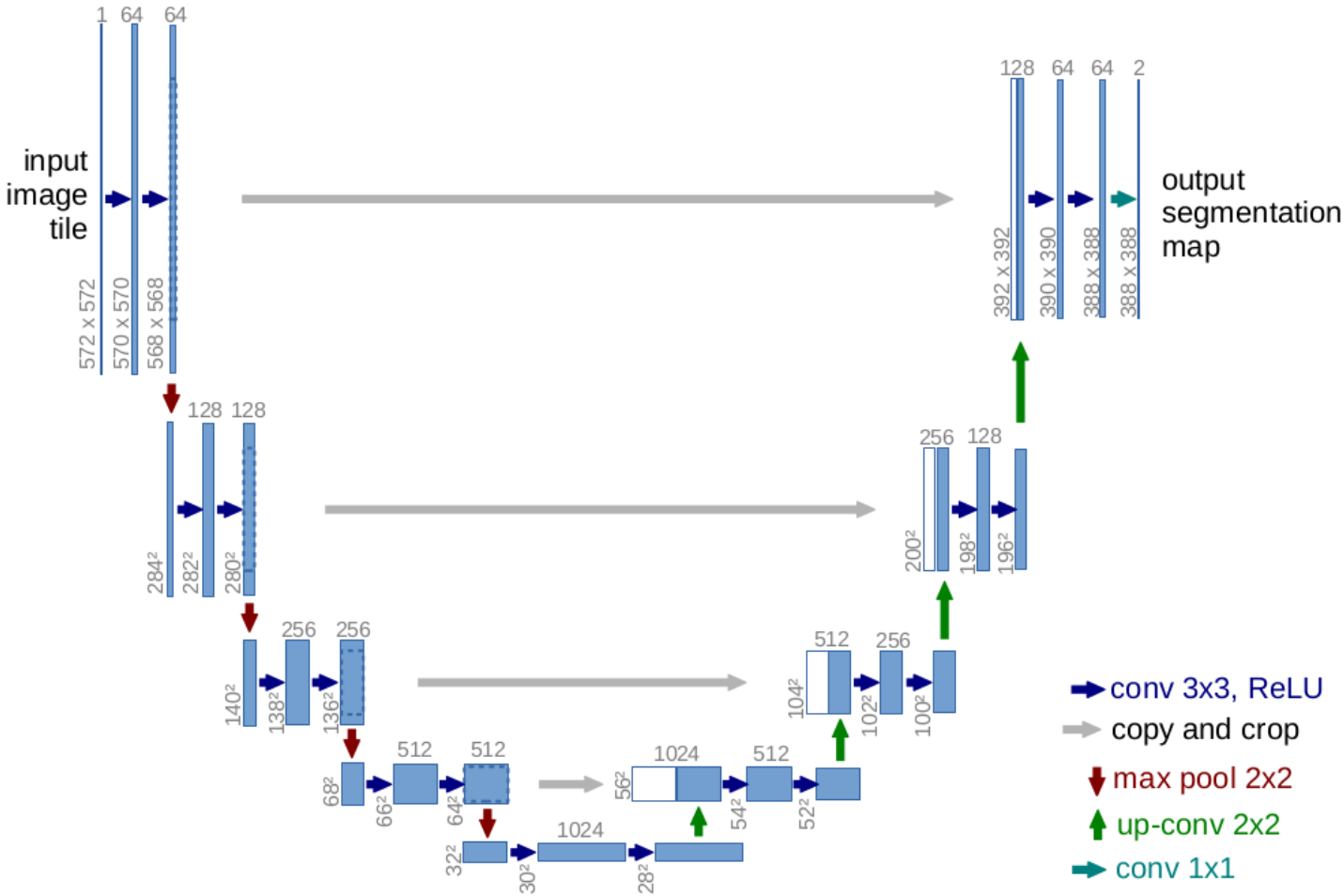
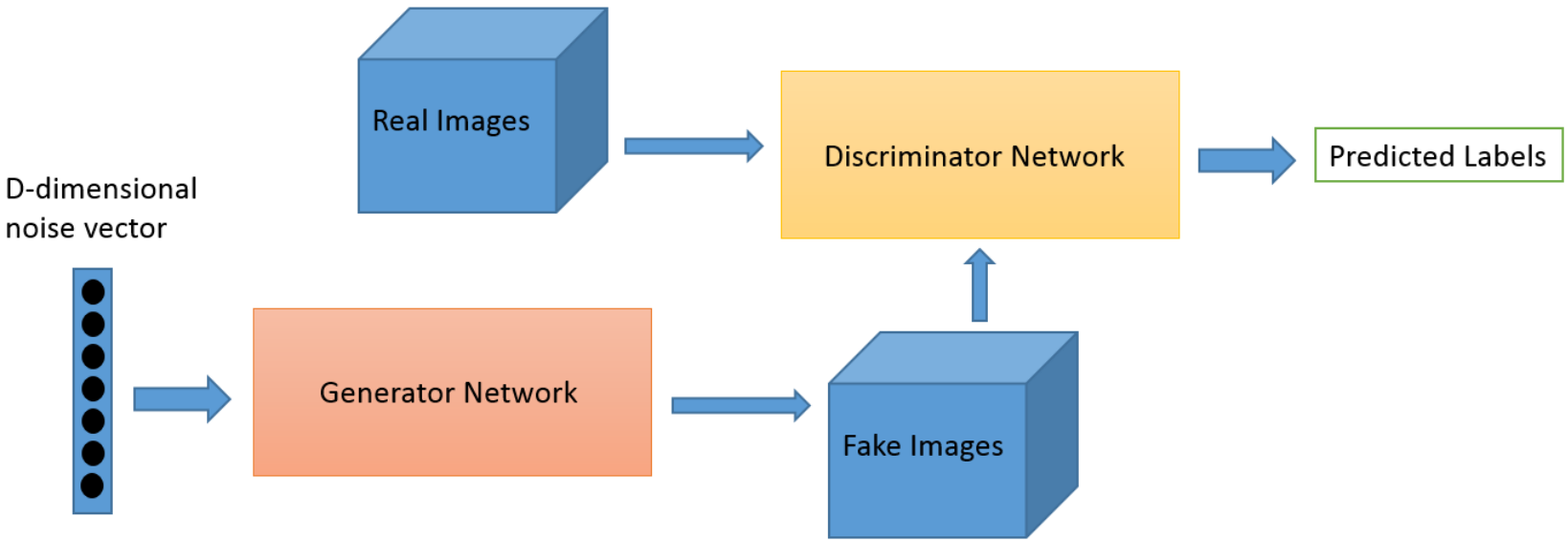


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Outros DL models

- **GAN** – gerar imagens
- **Siamese network, triplet network** — image retrieval
- **RNN** – dados temporais/sequenciais (recorrência)
- **LSTM** – recorrência
- **Geometric Deep Learning** deals with the extension of Deep Learning techniques to graph/manifold structured data. http://geometricdeeplearning.com/

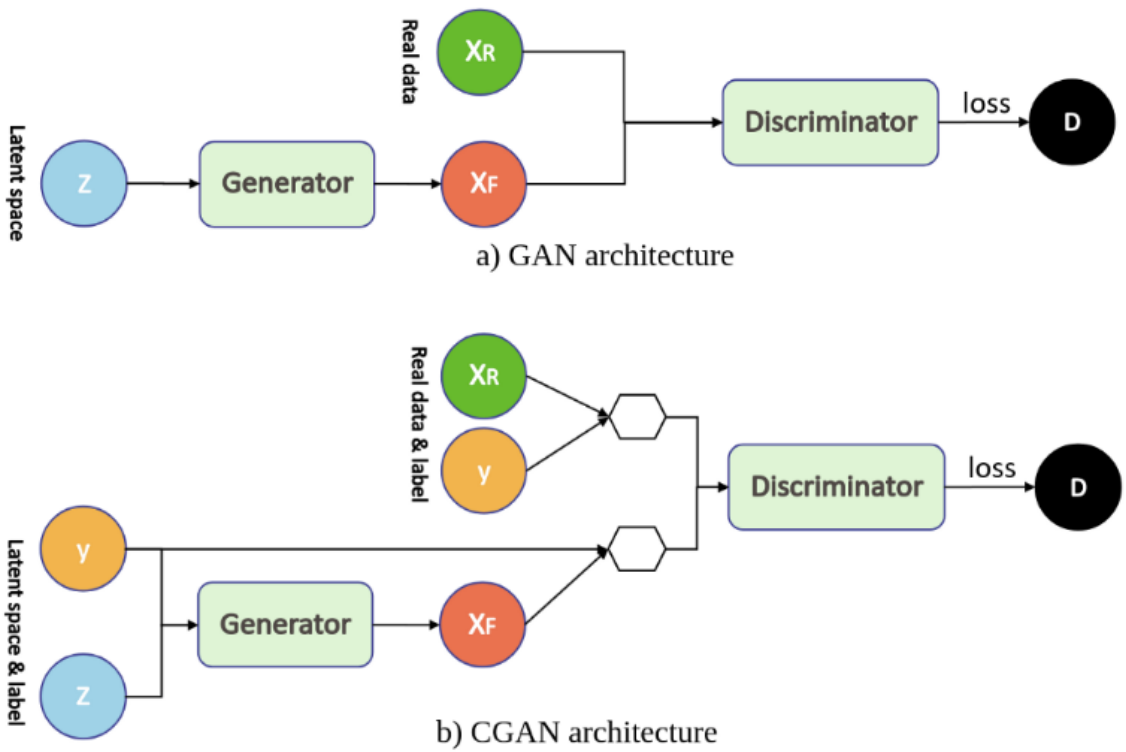
GAN schema



(fonte: O'Reilly)

$$min_G max_D V(D,G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + E_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

$$min_G max_D V(D,G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{y})] + E_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))]$$



https://mc.ai/a-tutorial-on-conditional-generative-adversarial-nets-keras-implementation/

GAN

(https://github.com/nashory/gans-awesome-applications)

Você já viu esses rostos ?

es rostos ?


Esses rostos não existem. Foram gerados por computador!
https://thispersondoesnotexist.com/

Artistic Style transfer

Artistic Style Transfer with Convolutional Neural Network

Case study 3: Art Generation

4. Architecture?

We use a **pre-trained model** because it **extracts important information** from images.

Deep Network

(pretrained on ImageNet)

classification

$$Content_C = \begin{pmatrix} 0.43 \\ 0.39 \\ \vdots \\ 0.53 \end{pmatrix}$$

$$\begin{pmatrix} 0.92 \\ 0.01 \\ \vdots \\ 0.53 \end{pmatrix}$$

Gram Matrix

$$Style_S = \begin{pmatrix} 0.13 \\ 0.32 \\ \vdots \\ 0.92 \end{pmatrix}$$

Deep Network

(pretrained on ImageNet)

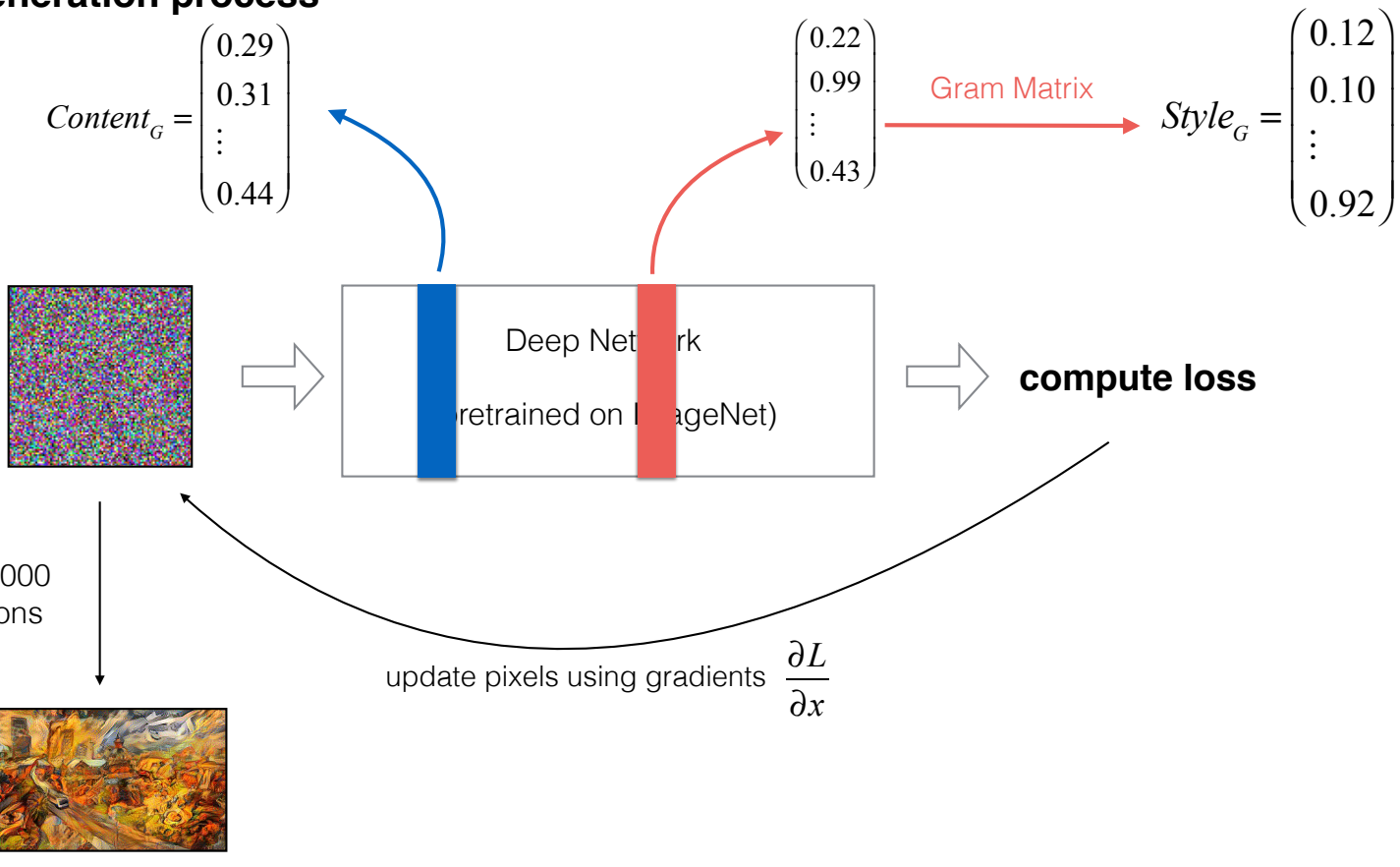
classification

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style, 2015

Kian Katanforoosh

Case study 3: Art Generation

Image generation process



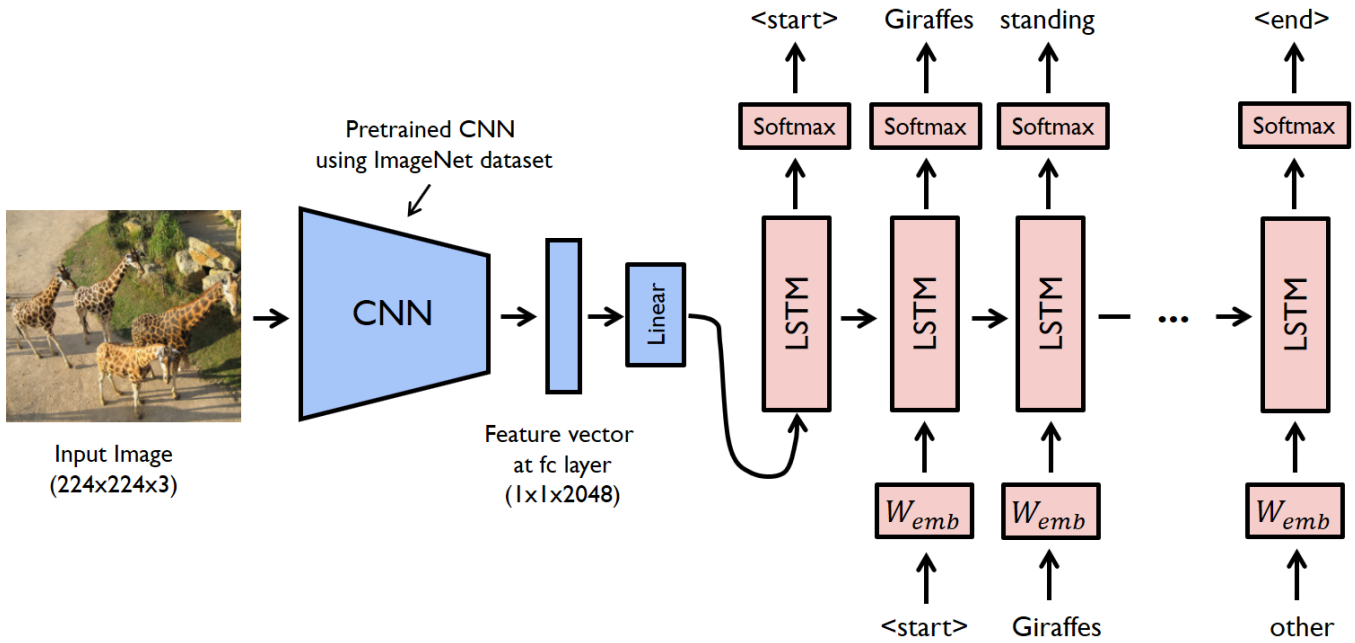
$$Content_C = \begin{pmatrix} 0.43 \\ 0.39 \\ \vdots \\ 0.53 \end{pmatrix}$$

$$Style_S = \begin{pmatrix} 0.13 \\ 0.32 \\ \vdots \\ 0.92 \end{pmatrix}$$

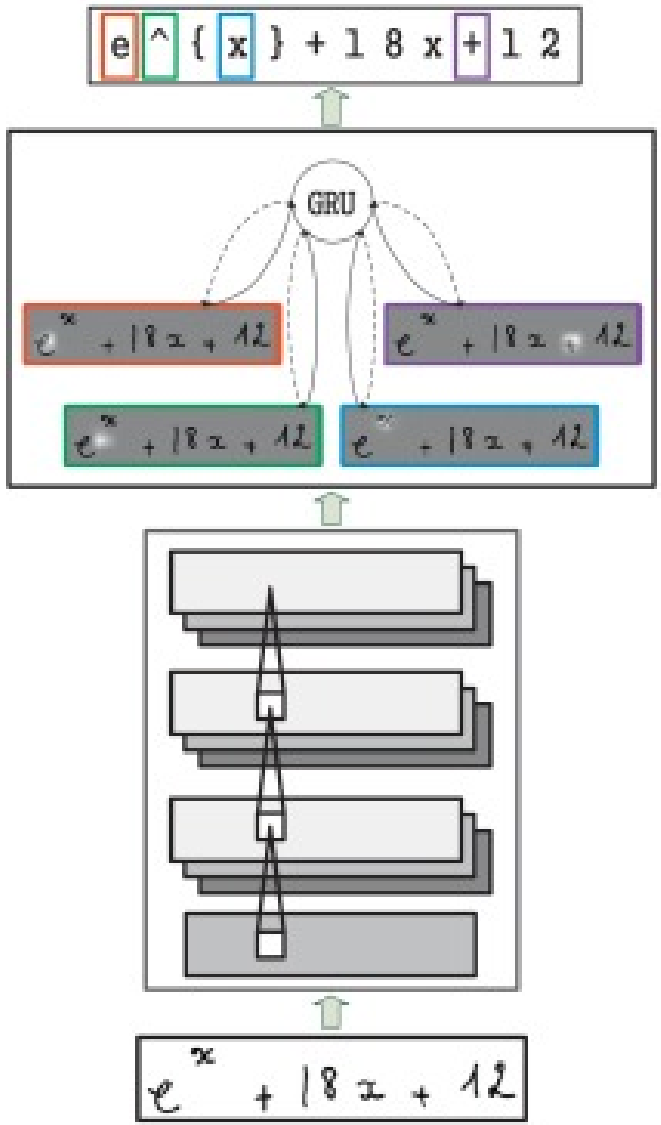
Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style, 2015

Kian Katanforoosh

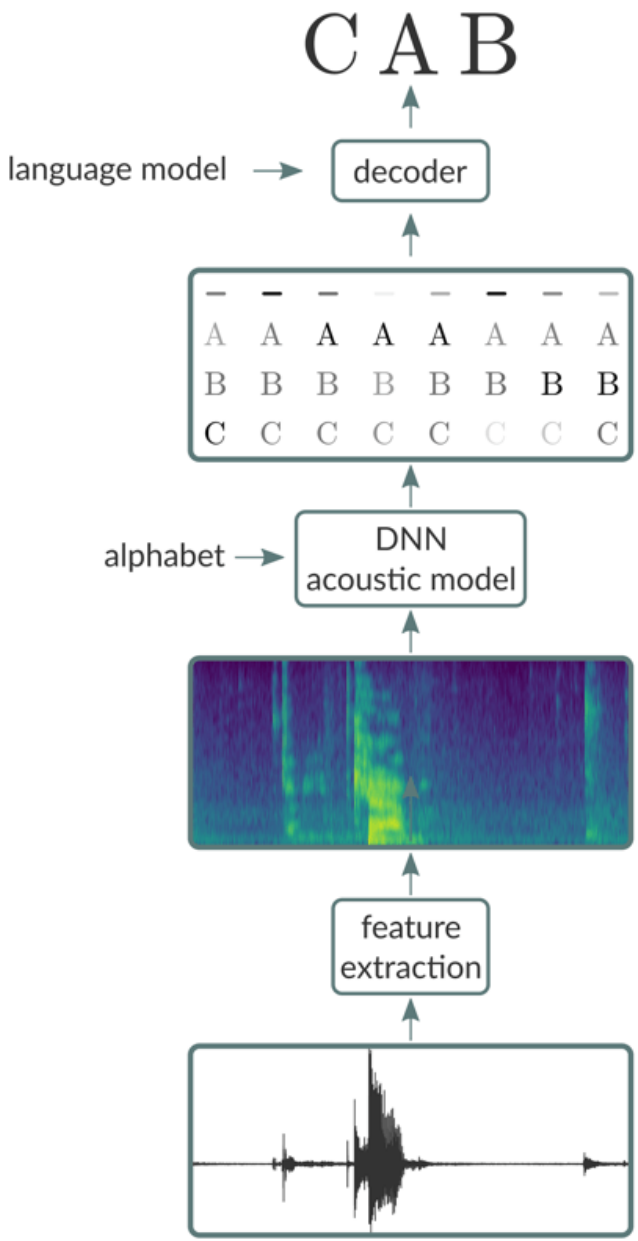
Automatic captioning



Automatic 2D to 1D



Speech


C A B
language model
decoder
– – – – – – – –
A A A A A A A A
B B B B B B B B
C C C C C C C C
alphabet
DNN
acoustic model
feature
extraction

Transforming Paintings and Photos Into Animations With AI

https://news.developer.nvidia.com/transforming-paintings-and-photos-into-animations-with-ai/