MAC 0460 / 5832
Introduction to Machine Learning

22 – Deep learning for Computer Vision (part 2)

IME/USP (05/07/2021)

Image classification

Bag of visual words (BOW)

Inspiration comes from **NLP**

Bag-of-words: widely used for document classification/retrieval

Vocabulary with N words

First idea: Document D represented by a vector f_D of size N:

$$f_D[i] = \text{number of occurrences of word } v_i \text{ in the document}$$

Improvement: term frequency–inverse document frequency (TF-IDF)

- normalization taking into account the frequency of the words
- $f_D[i]$ divided by the count of occurrences of v_i over all documents

Images (Computer Vision)

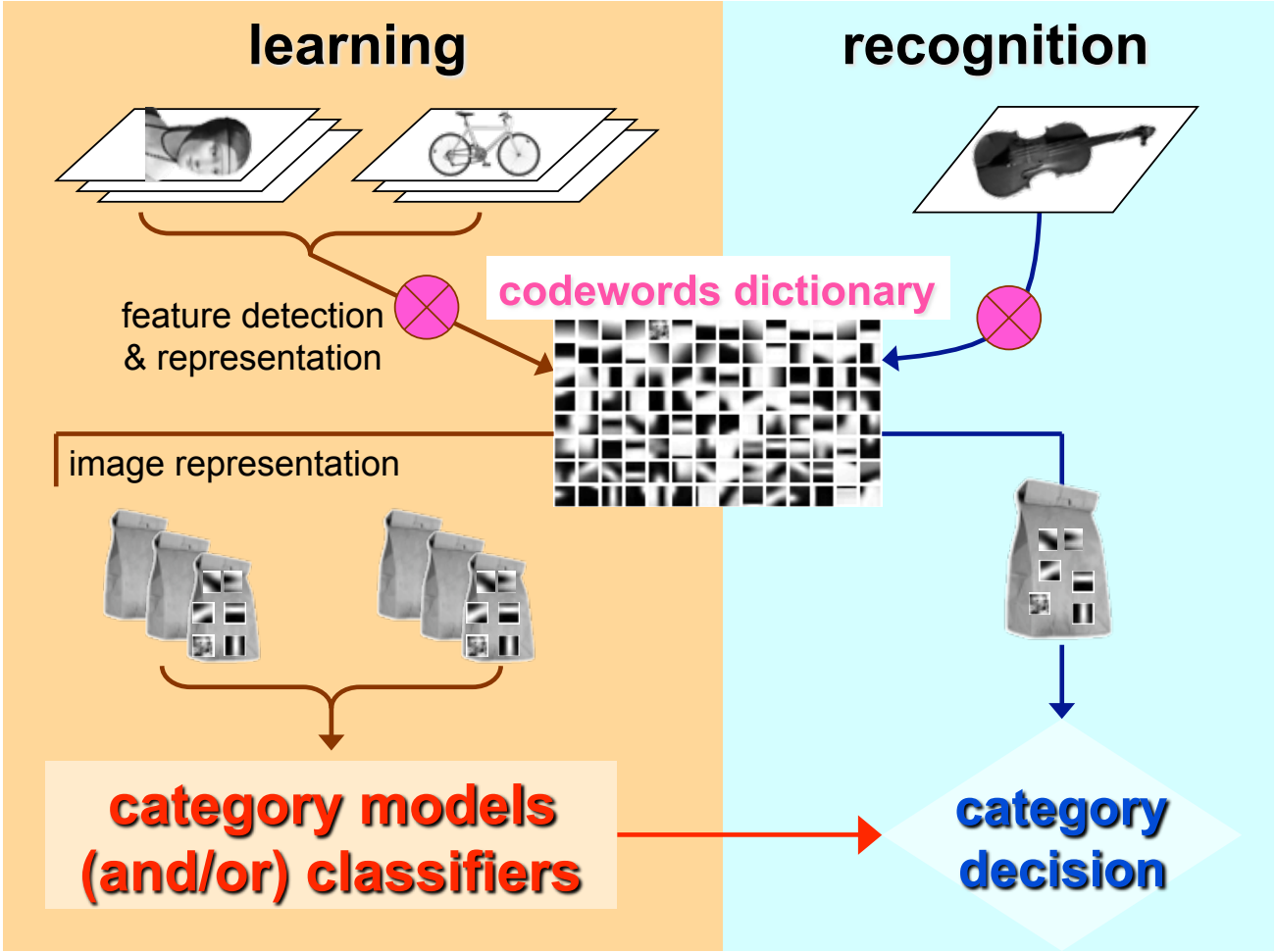
Bag of keypoints (Csurka, 2004)

1. **Create a dictionary** (visual patterns) from the set of training images

2. **Training**: compute the *bag-of-words* representation for each training image and train a classifier to discriminate object classes using the *bag-of-words* representation as features

3. **Classification**: compute the the *bag-of-words* representation of the image to be classified and apply the classifier

Bag-of-Words Model: Overview

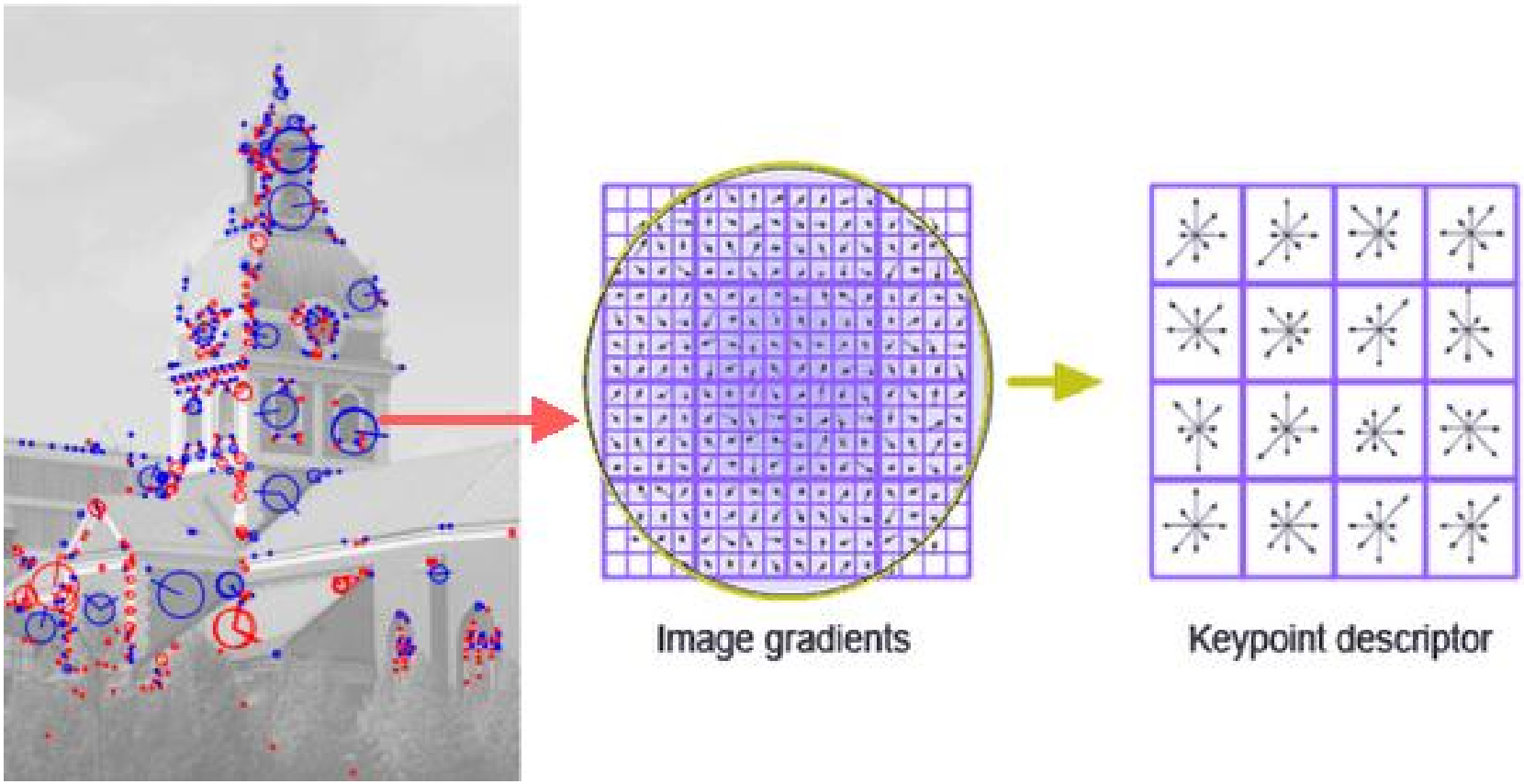


BoW – Dictionary creation

- Compute *keypoints* of the images
- Compute descriptors for each of the *keypoints*
- Csurka uses *Harris affine detector* for the detection of keypoints and SIFT for the description (feature vector of dimension 128)
- Descriptors are grouped using the K-means algorithm (parameter K is adjusted somehow)
- The dictionary consists of the set of centers of the K clusters (**visual words**)

Example of feature detection and description

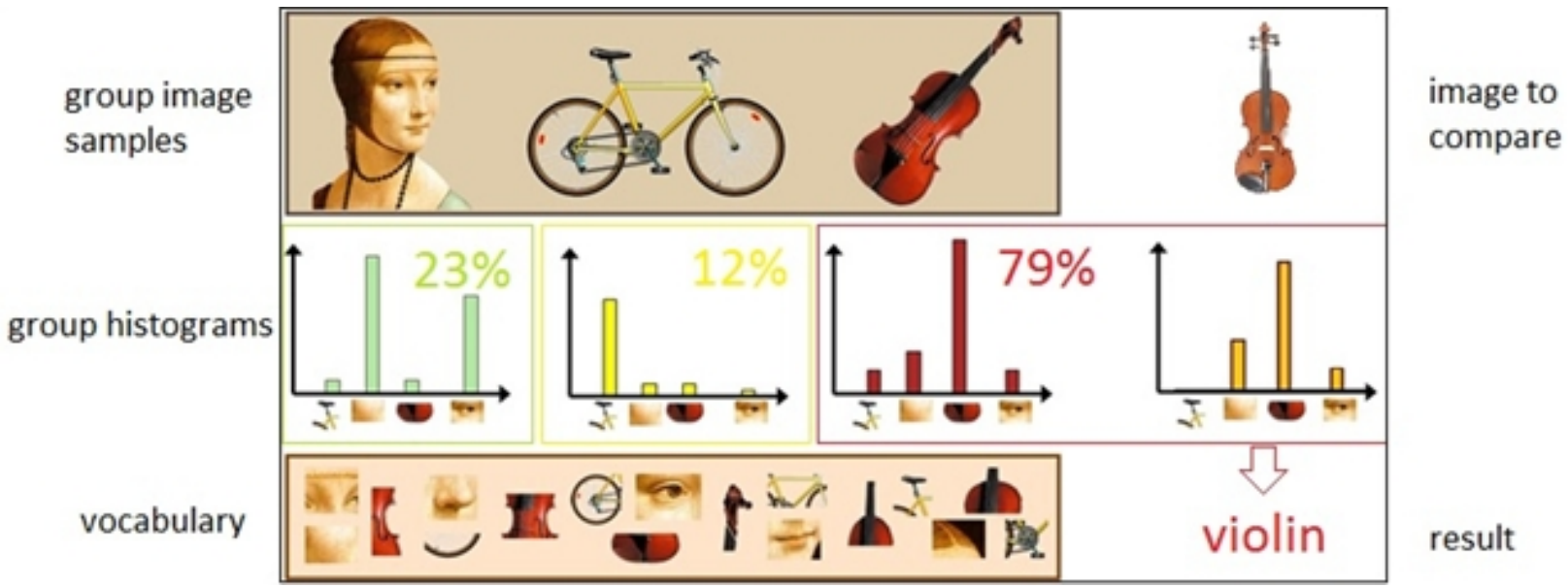
SIFT (Scale Invariant Feature Transform)



BoW – Training the classifier

- For each image, compute the set of descriptors
- Each descriptor is matched with the closest *visual word* in the dictionary
- Build a histogram of the *visual words* in the image (this is the *bag-of-words* of the image)
- The *bag-of-words* representation is the input feature used to train the classifier

BoW – Training and classification process



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

Computer Vision algorithms consisted of very complex processing pipelines

Examples we have seen:

Object detection: Viola-Jones object detection framework

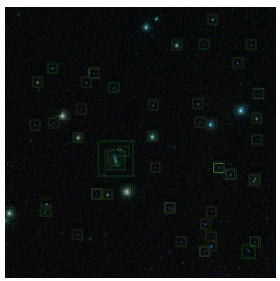
Image classification: Csurka's BoW model

Traditional ML

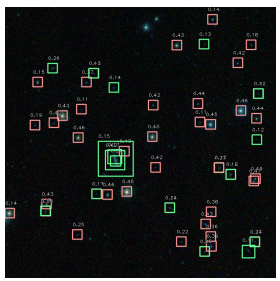
⇒ feature extraction/ selection ⇒ $(x_1, x_2, \ldots, x_n)$ ⇒ Traditional ML algoritm ⇒ y

Deep learning: end-to-end processing

⇒ Deep learning algorithm ⇒ y



⇒ Deep learning algorithm ⇒



Neural network model for images

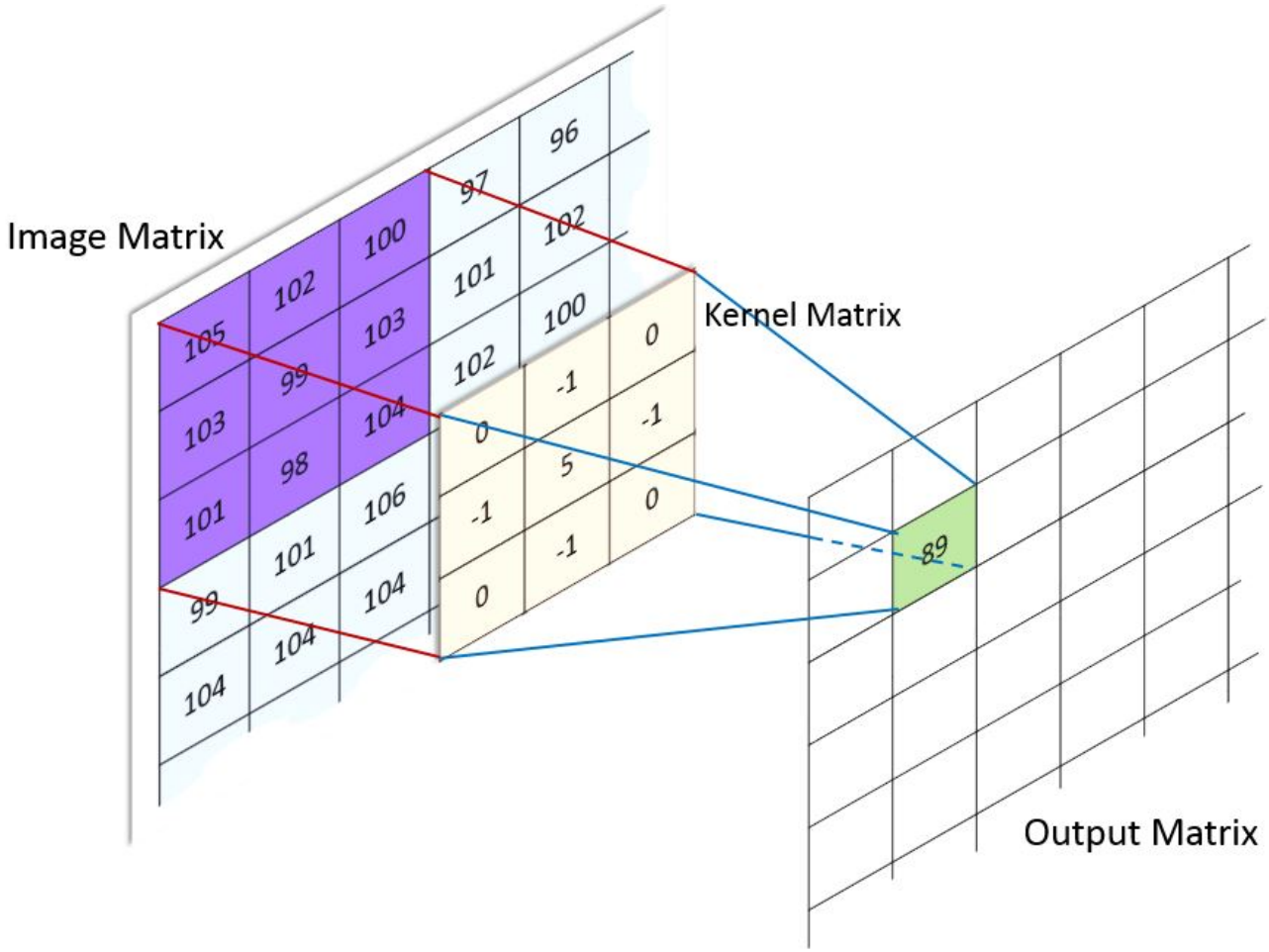
CNN (Convolutional Neural Networks)

They are based on the **convolution** operation

Local linear operation used in image transformations

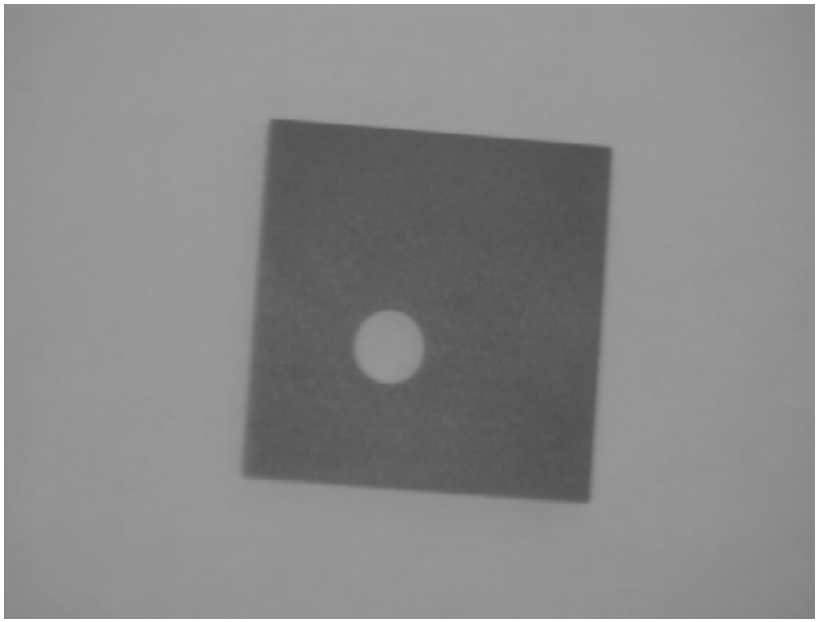
$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]\, g[m]$$

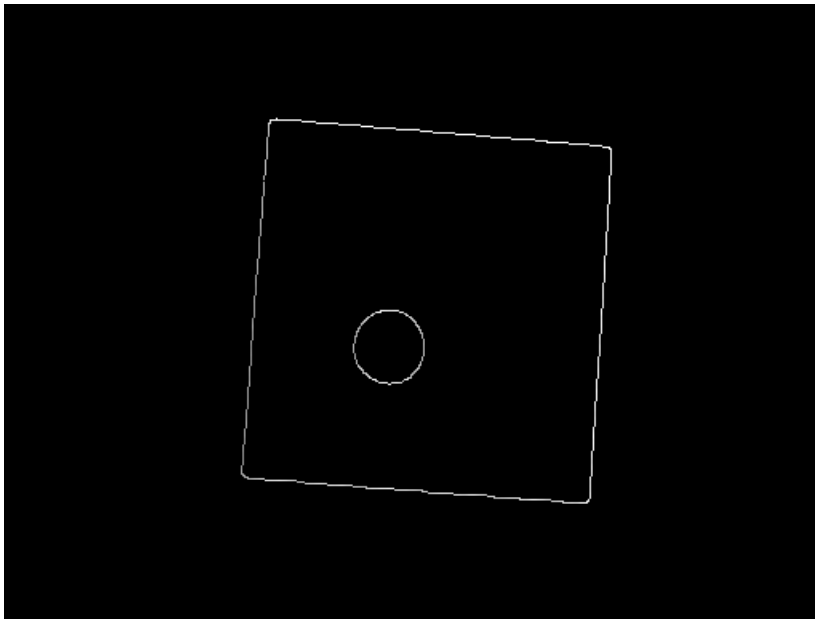
Convolution



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

Examples of convolution kernels – edge detection

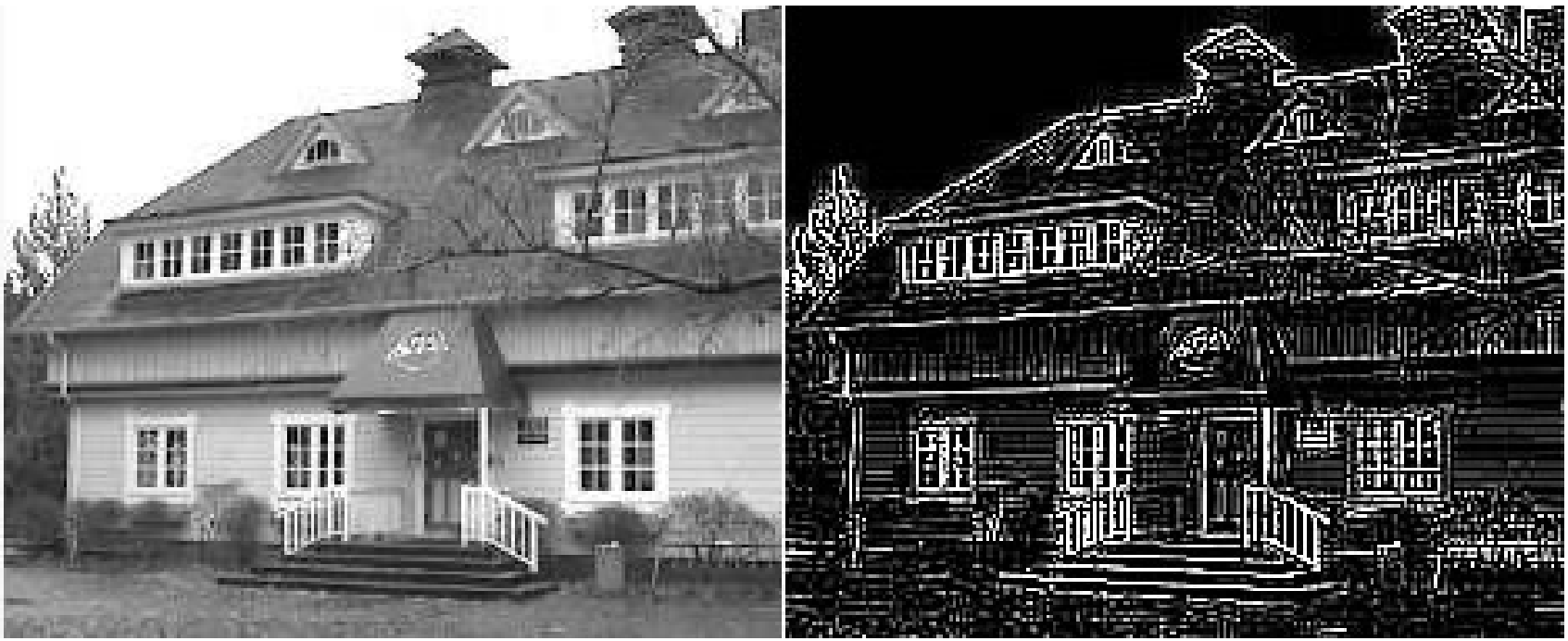




-1	-1	-1
-1	8	-1
-1	-1	-1

(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

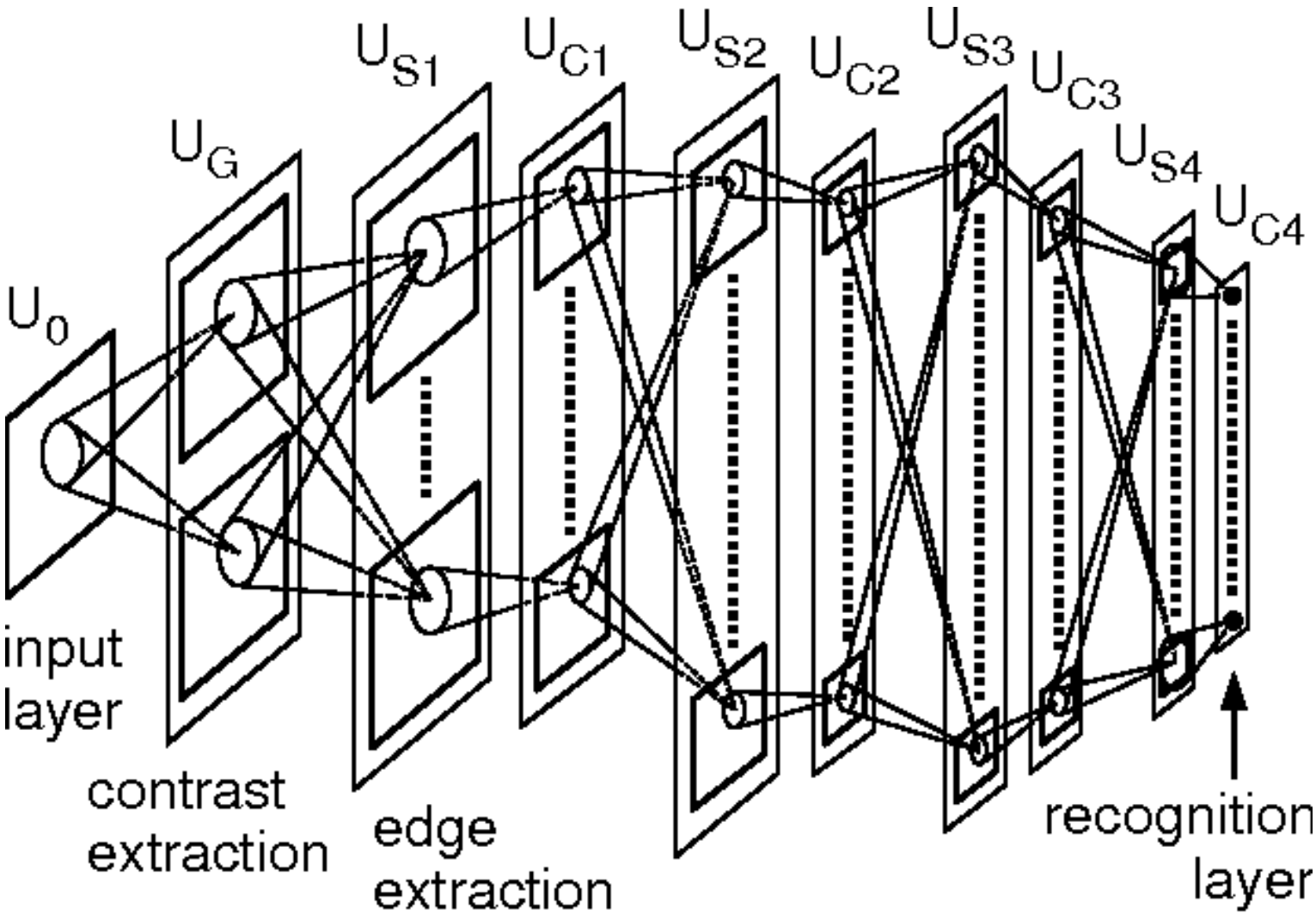
(http://aishack.in/tutorials/image-convolution-examples/)

Use of convolution in neural nets is not so recent:

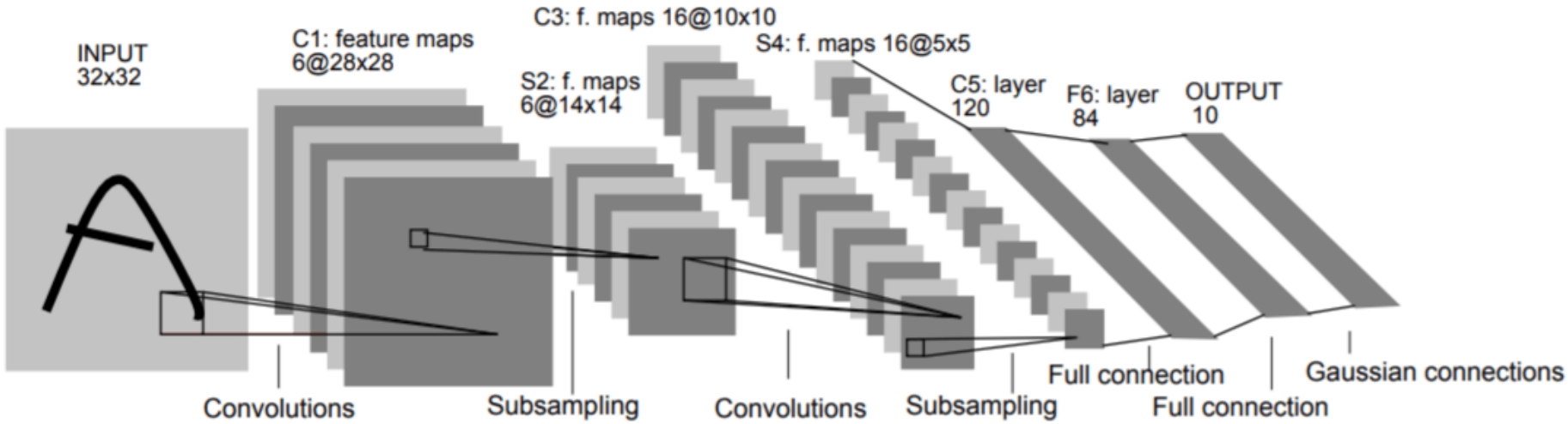
- **Neocognitron** – Fukushima, 1990
  Detailed explanation: https://www.kiv.zcu.cz/studies/predmety/uir/NS/Neocognitron/en/index.html

- **LeNet** - Yan Lecun, 1998, postal code recognition
  http://yann.lecun.com/exdb/lenet/index.html

Neocognitron – Fukushima, 1990



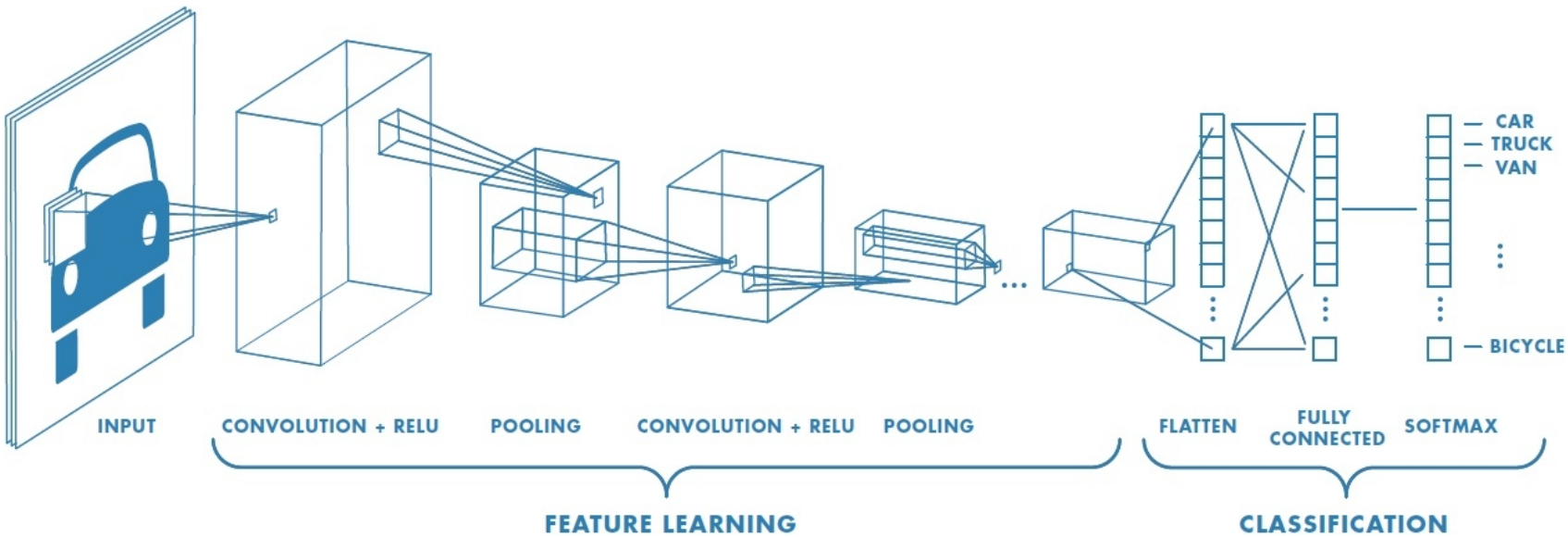
LeNet5 – Yan Lecun, 1998



First CNN model, applied on postal code recognition

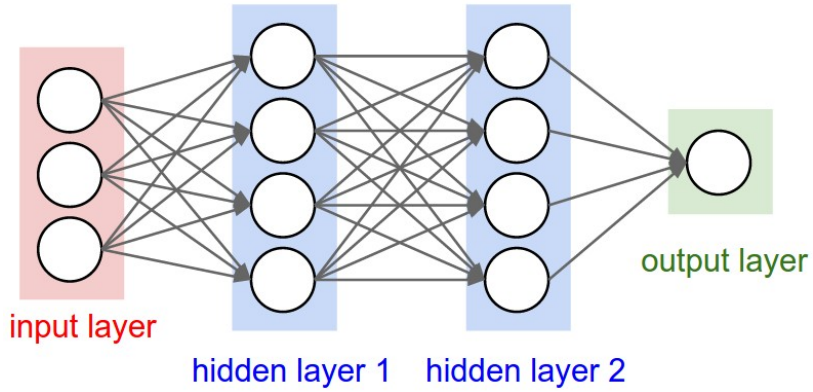
Basics of CNN

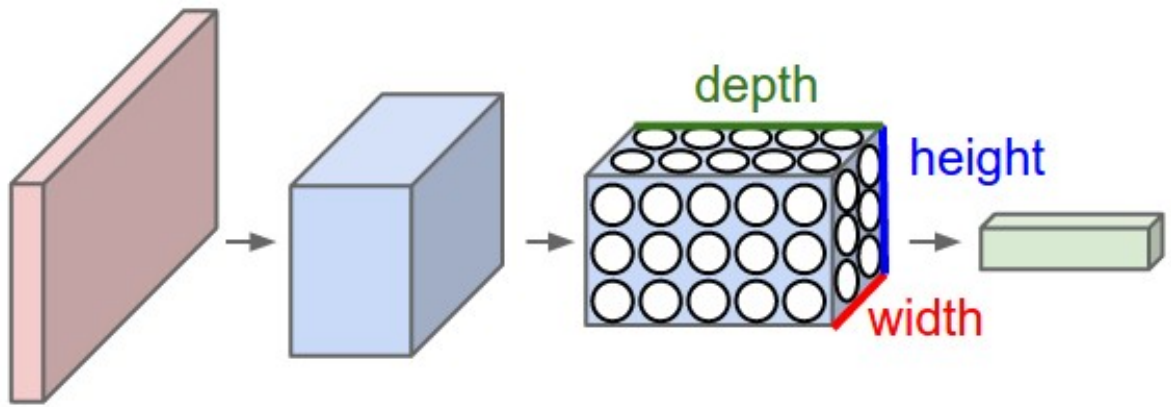
Typical architecture of Convnets



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

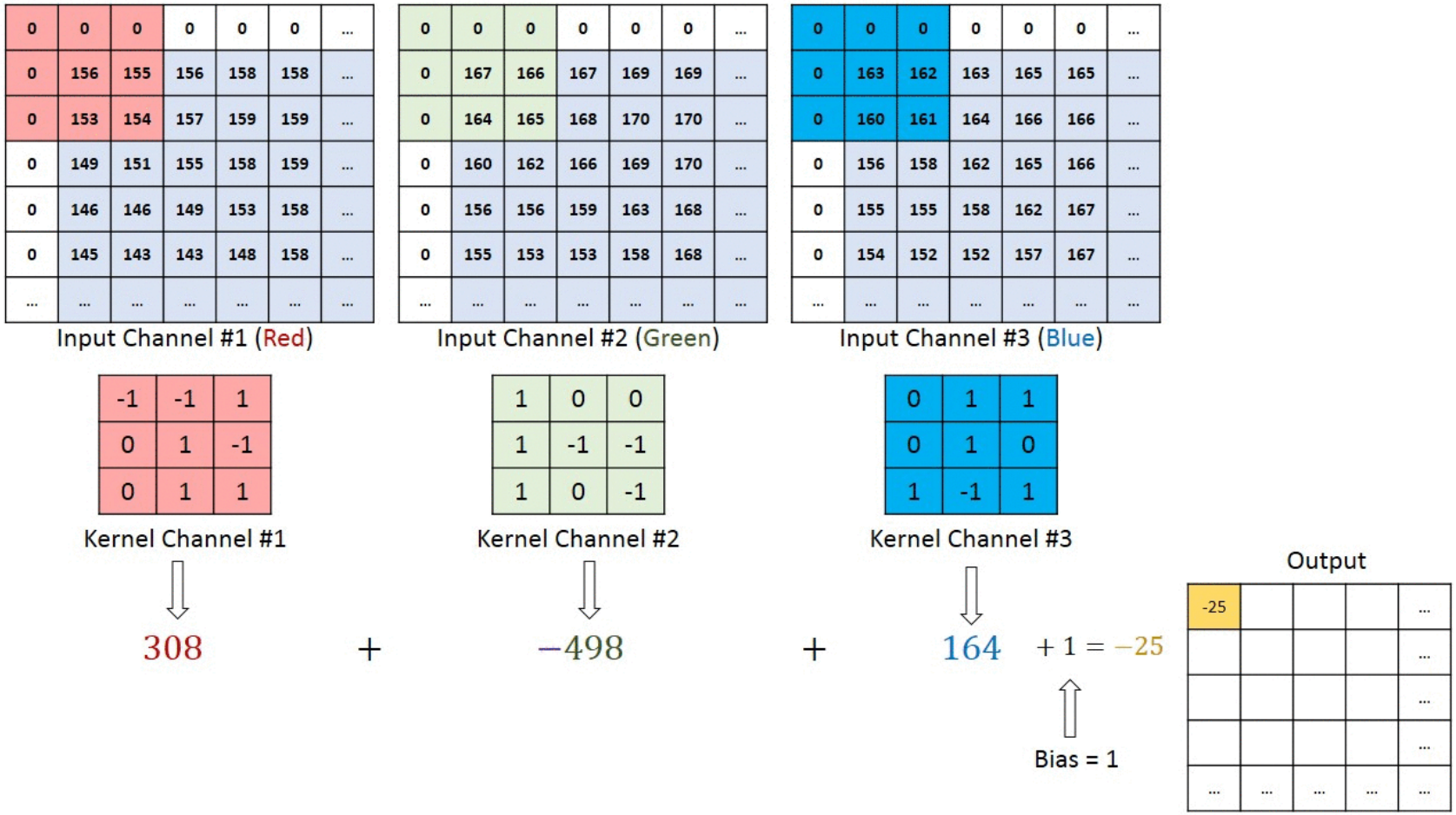
Graphical representation of ConvNet structure





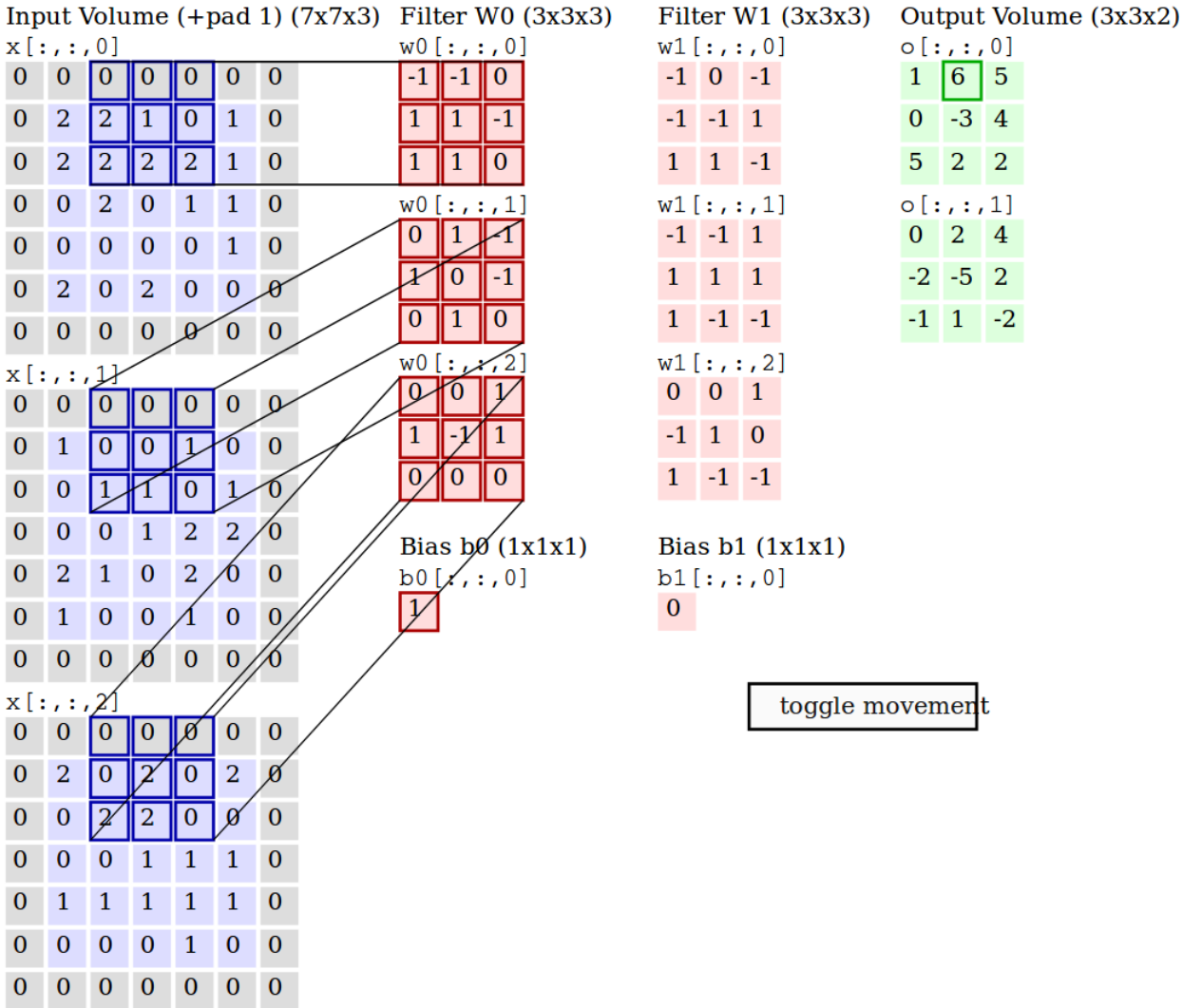

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
```

Convolution on multiple channels



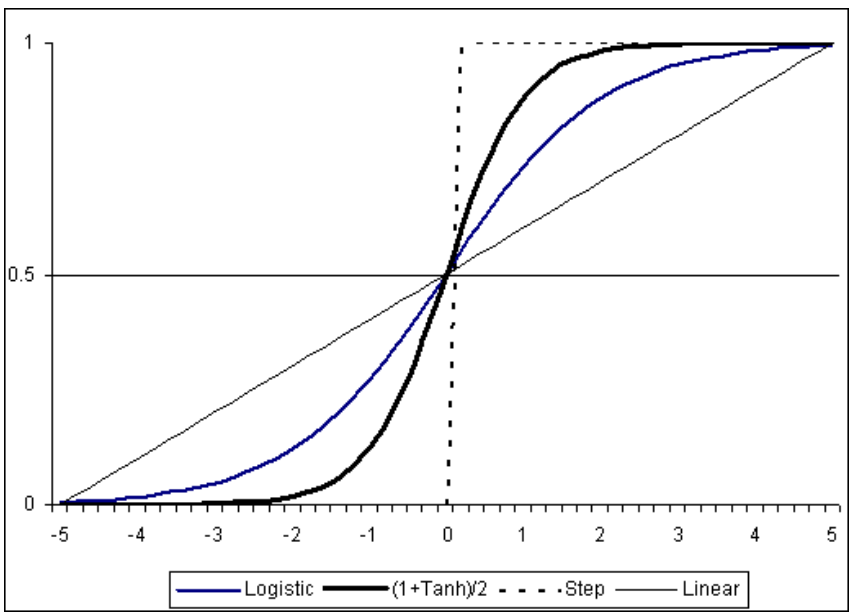
(http://machinelearninguru.com/computer_vision/basics/convolution/convolution_layer.html)

Multiple filters

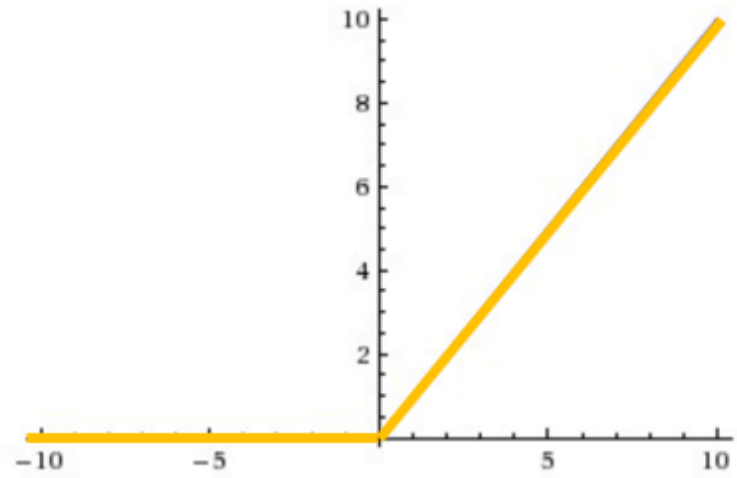


Concepts: padding, stride, kernel size, number of filters/maps

Activation functions



ReLU

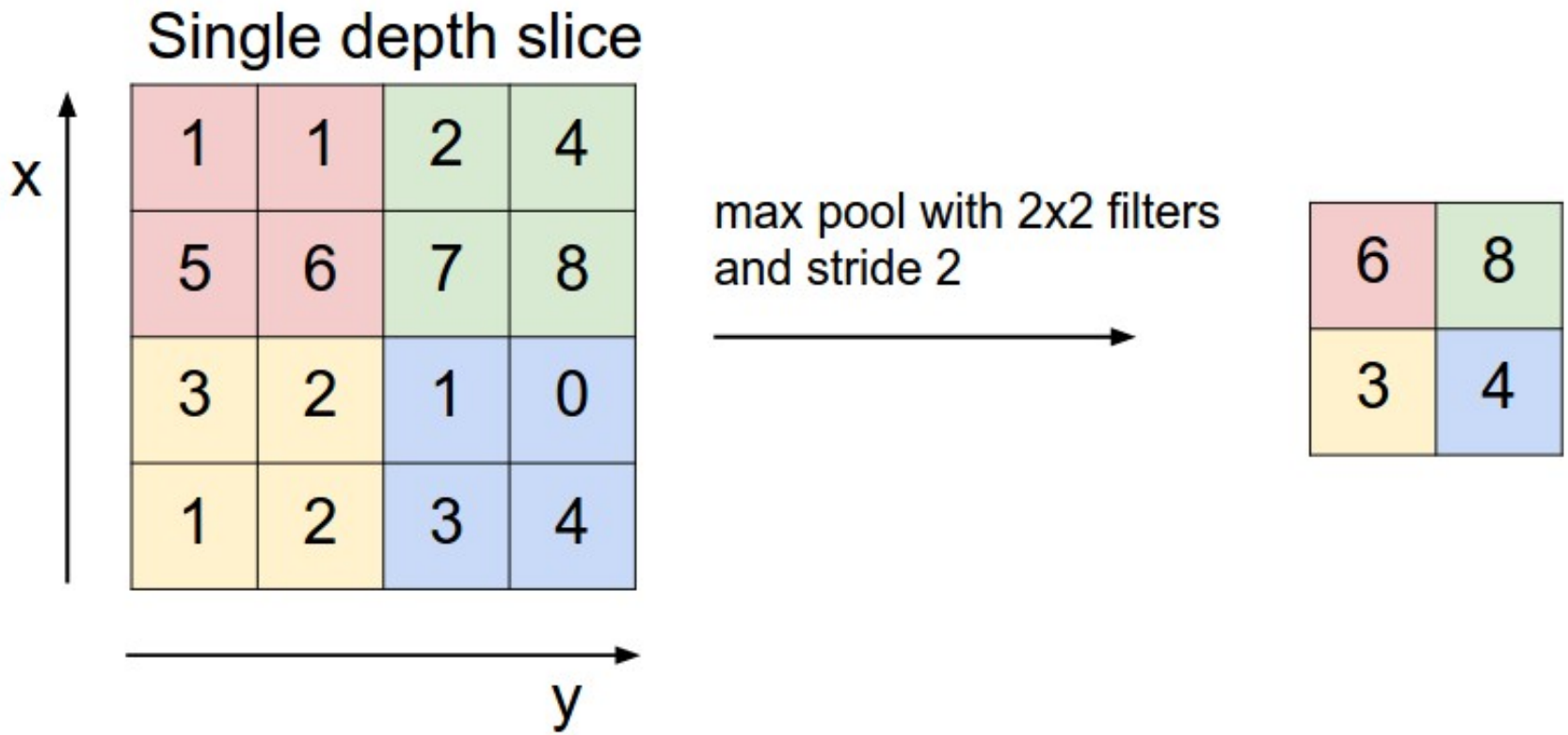


$$F(x) = \max(0, x)$$

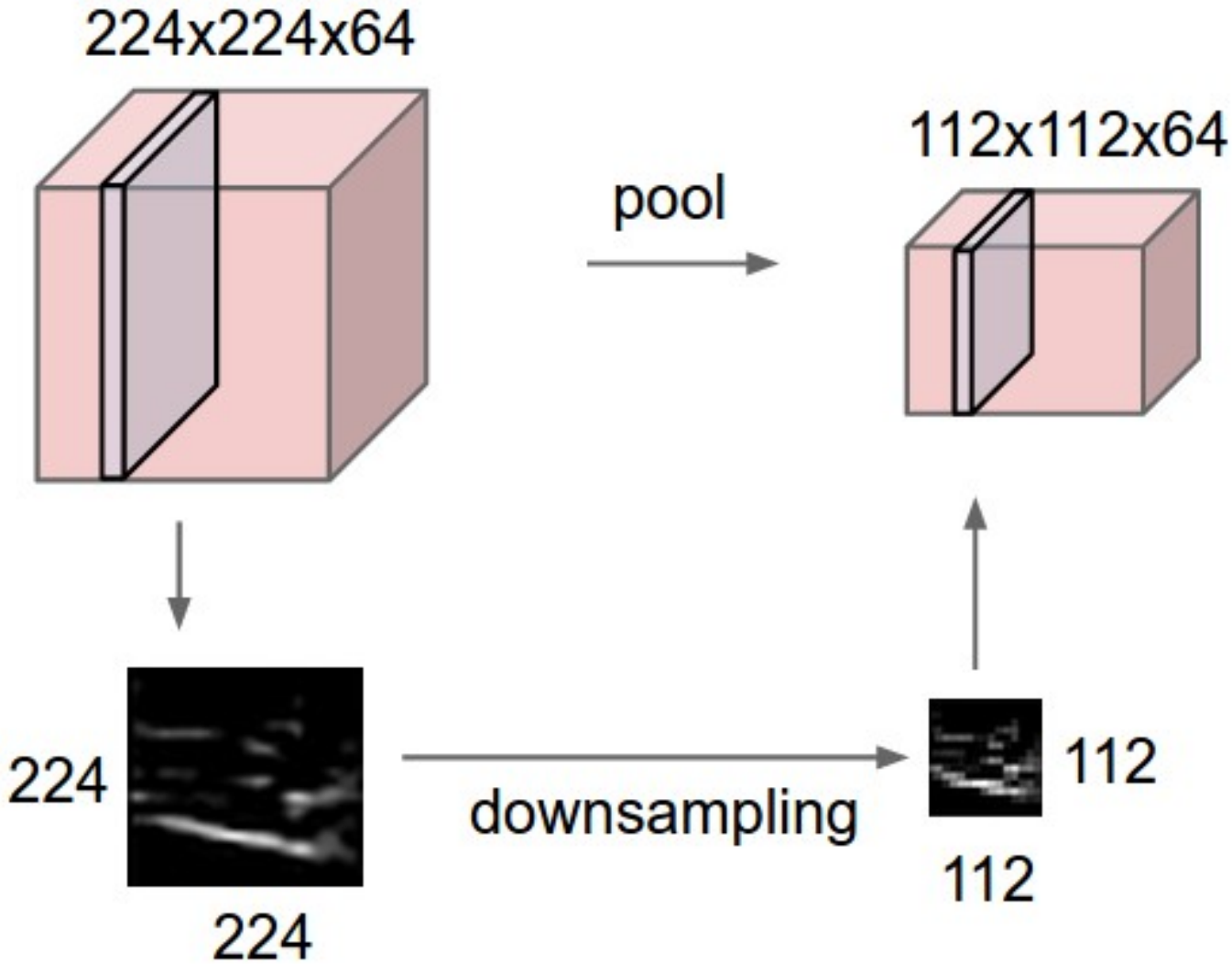
Subsampling

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

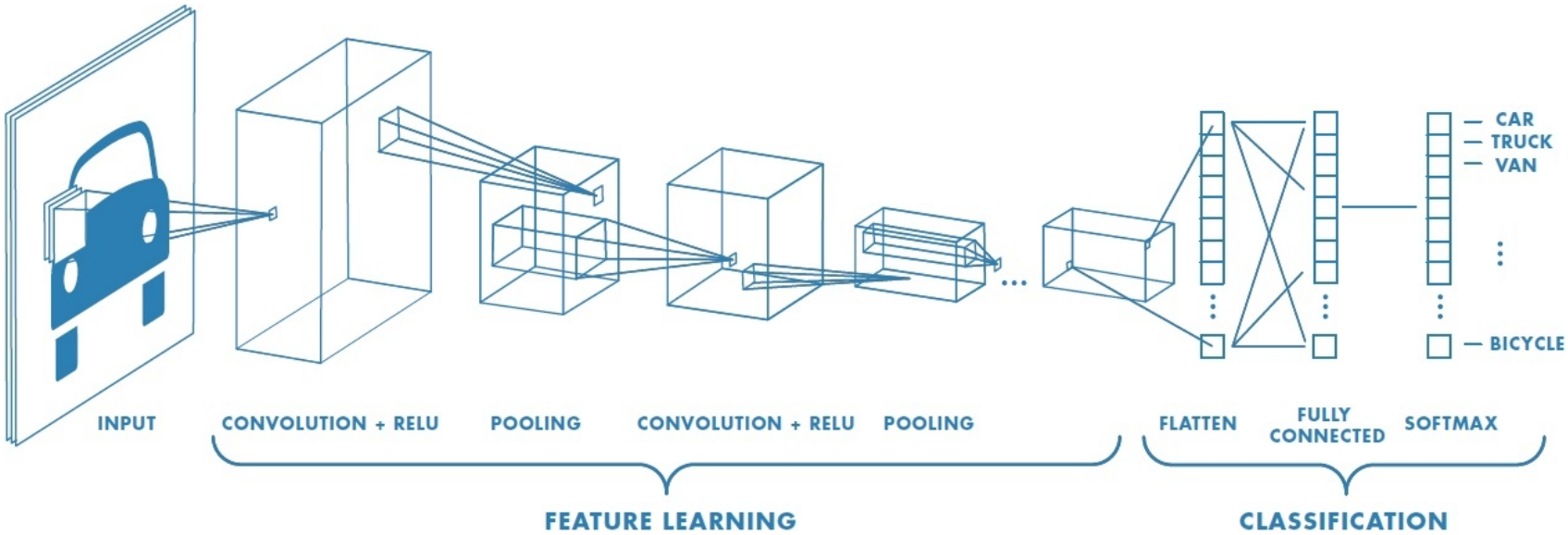
Max pooling


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

Training of CNN

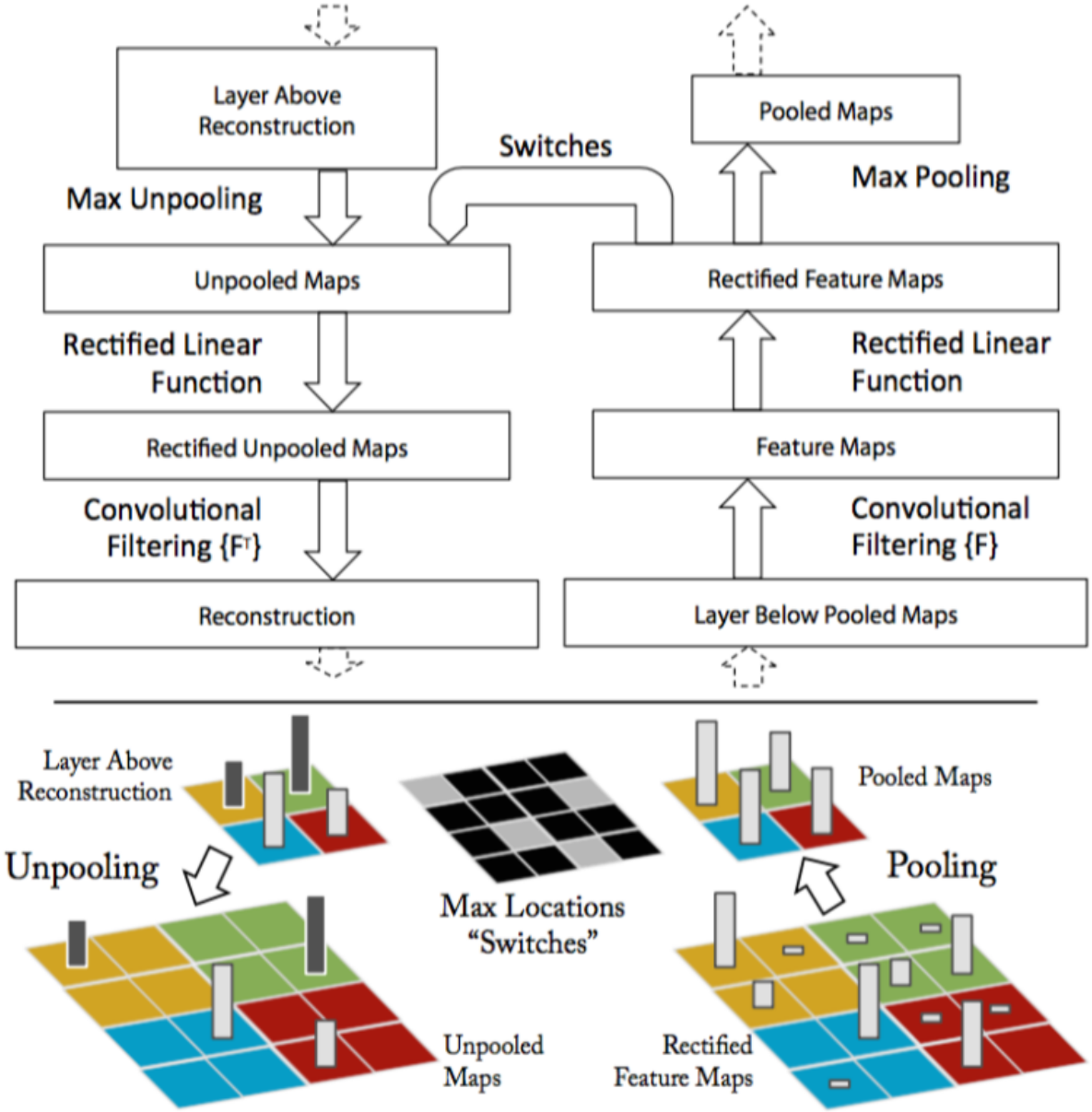
It is done using the backpropagation algorithm (gradient descent)

Convolutional layers act as feature extractors. How ?

Zeiler et al., 2013, proposed a visualizaton technique

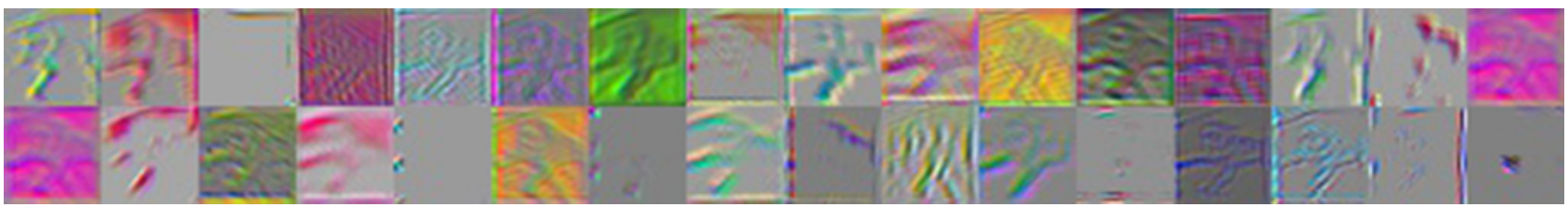
At a given convolutional layer, take the node with largest activation

Reconstruct the input image by backpropagating from that node



(Zeiler et al., 2013)

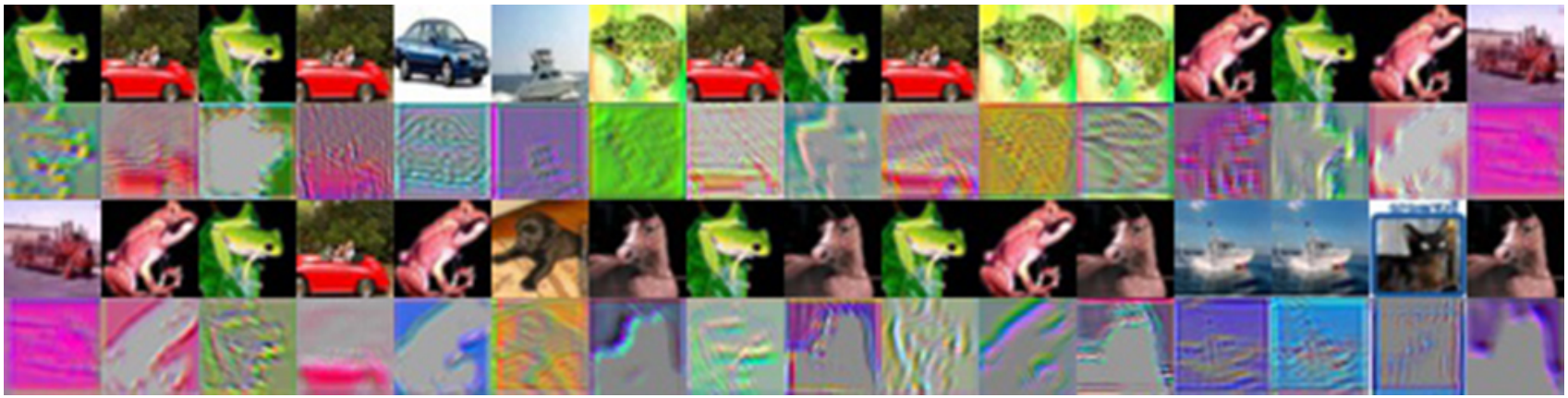
Images reconstructed from neurons 1 to 32 of layer 1, for one input image



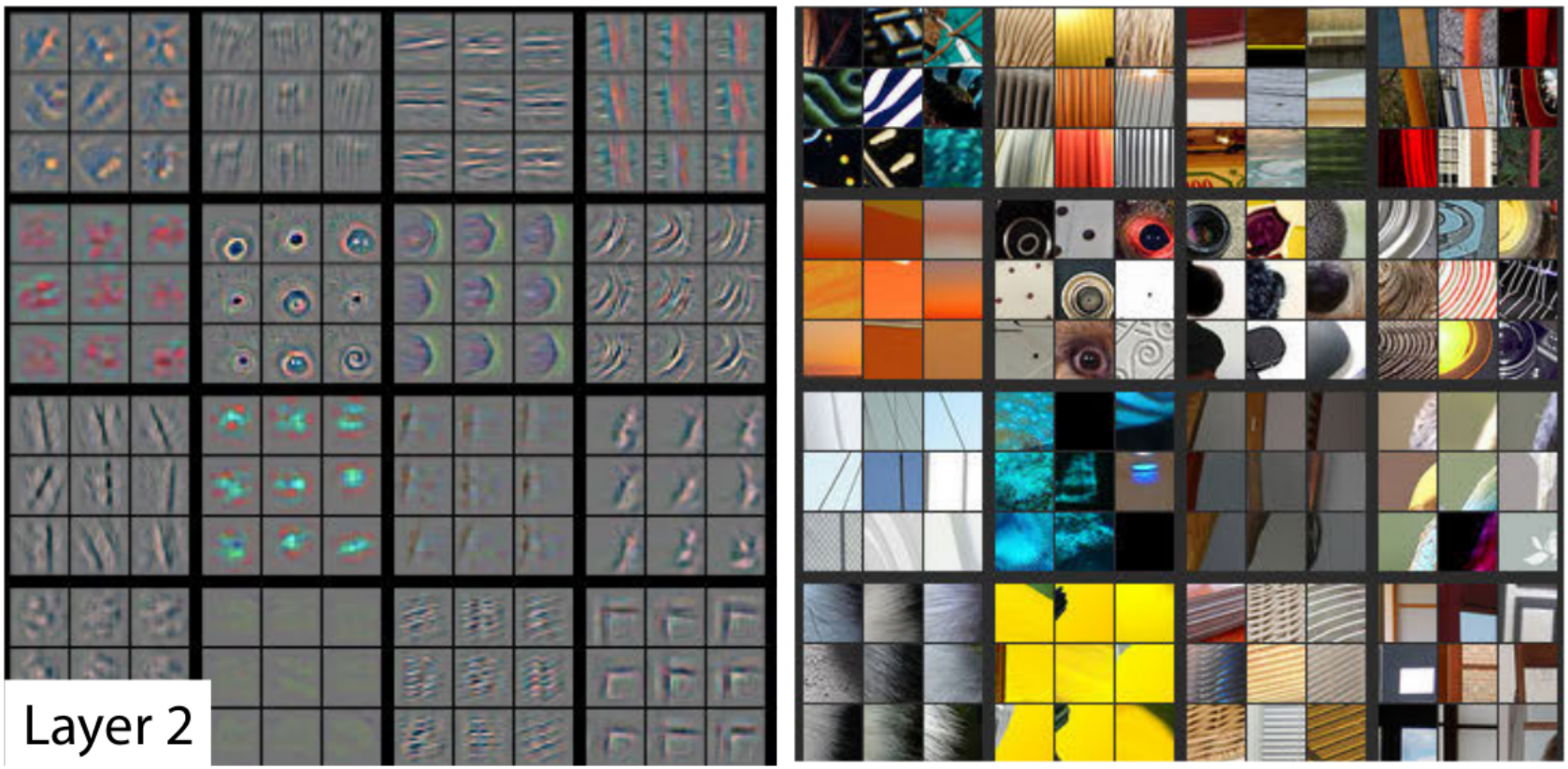
(http://kvfrans.com/visualizing-features-from-a-convolutional-neural-network/)

Images reconstructed from neuron 7 of layer 1, for multiple input images

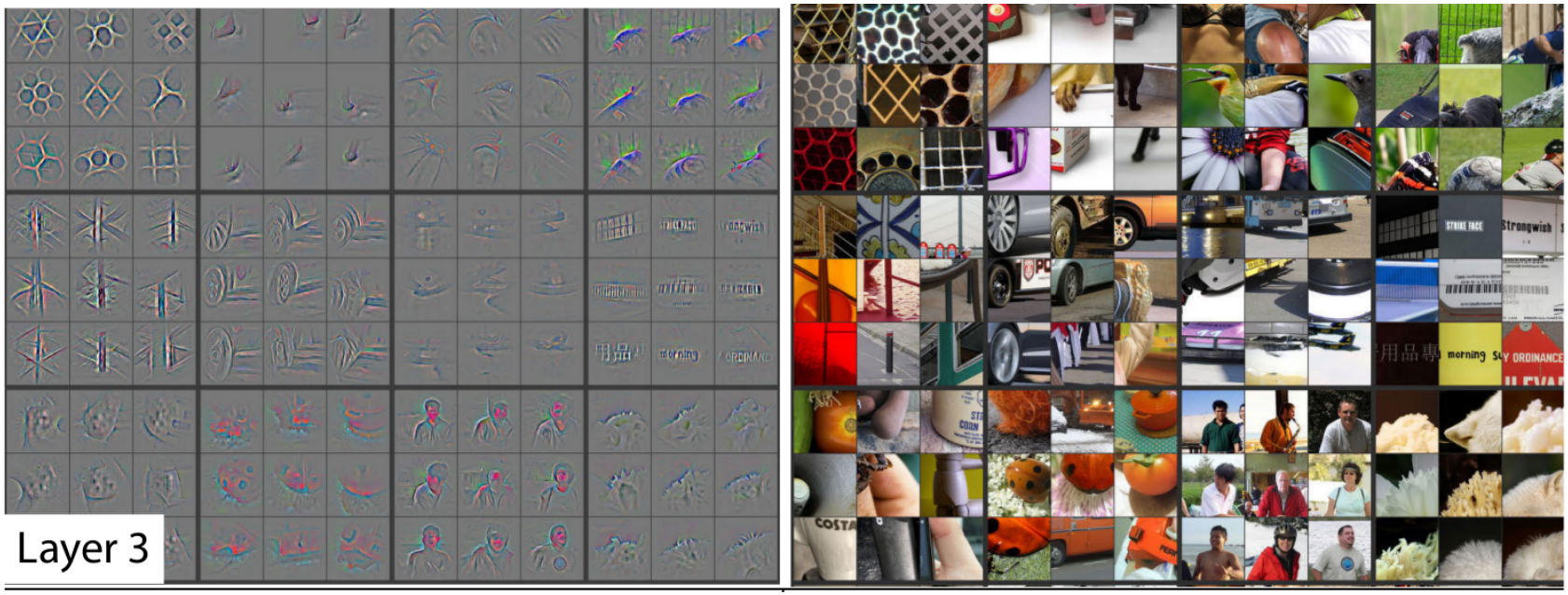
For each of the 32 neurons, reconstruction of the images that most activated the neuron



(Zeiler et al.) 16 neurons in layer 2; for each neuron, reconstruction of the 9 images that correspond to strongest activation of the node



(Zeiler et al.) 12 neurons in layer 3; for each neuron, reconstruction of the 9 images that correspond to strongest activation of the node



Large neural networks require large amount of training data

Transfer learning

It works as a nice weight initialization

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: galaxy image classification

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

1. Only spiral and ellipitic
2. Extract *features* for each image
   - 2.1 Using Morfometryka hand-egineered features (Fabrício Ferrari, FURG)
   - 2.2 Using a pre-trained CNN
3. Separate data into training and validations sets
4. Logistic regression classifier trained on training data
5. Evaluation on the validation data

In the next slides:

- label above the image is the true class
  (blue="elliptic", orange="spiral")

- first column: examples of wrongly classified images

- columns 2 to 5: most similar images to the ones at column 1, according to feature similarity


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


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

Projeções das *features*

Features convolucionais

Features morfométricos

The previous example did not use fine tuning

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

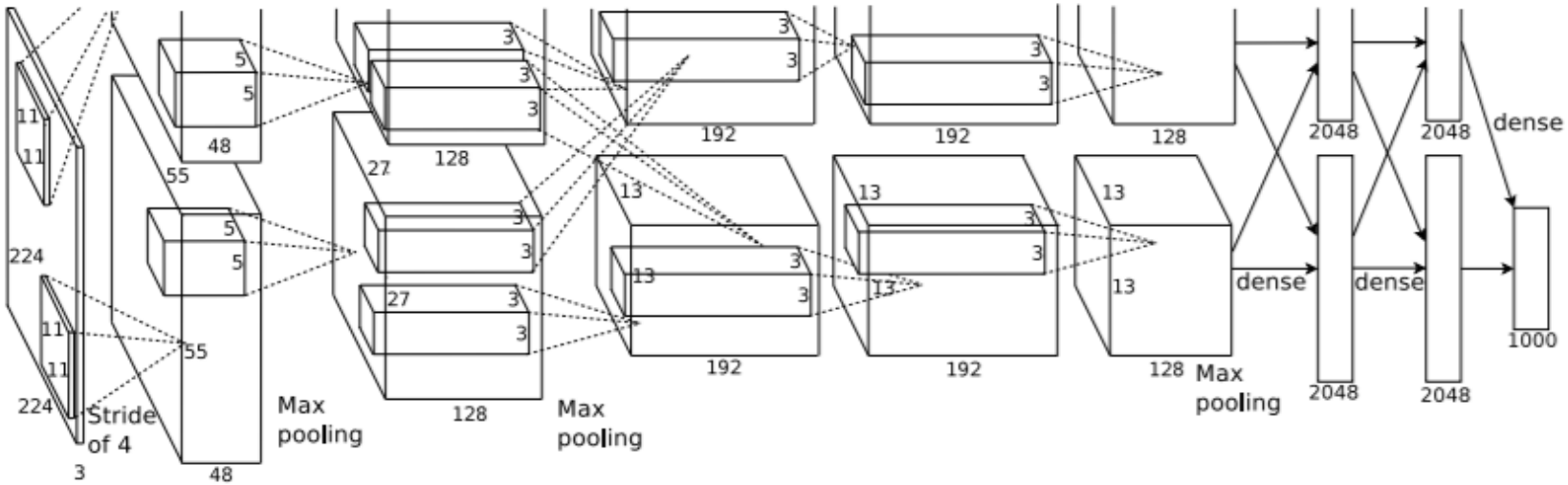
- **GoogleLeNet** (Inception), 2014: winner of ILSVRC 2014, inception layers, 7M network parameters

  (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

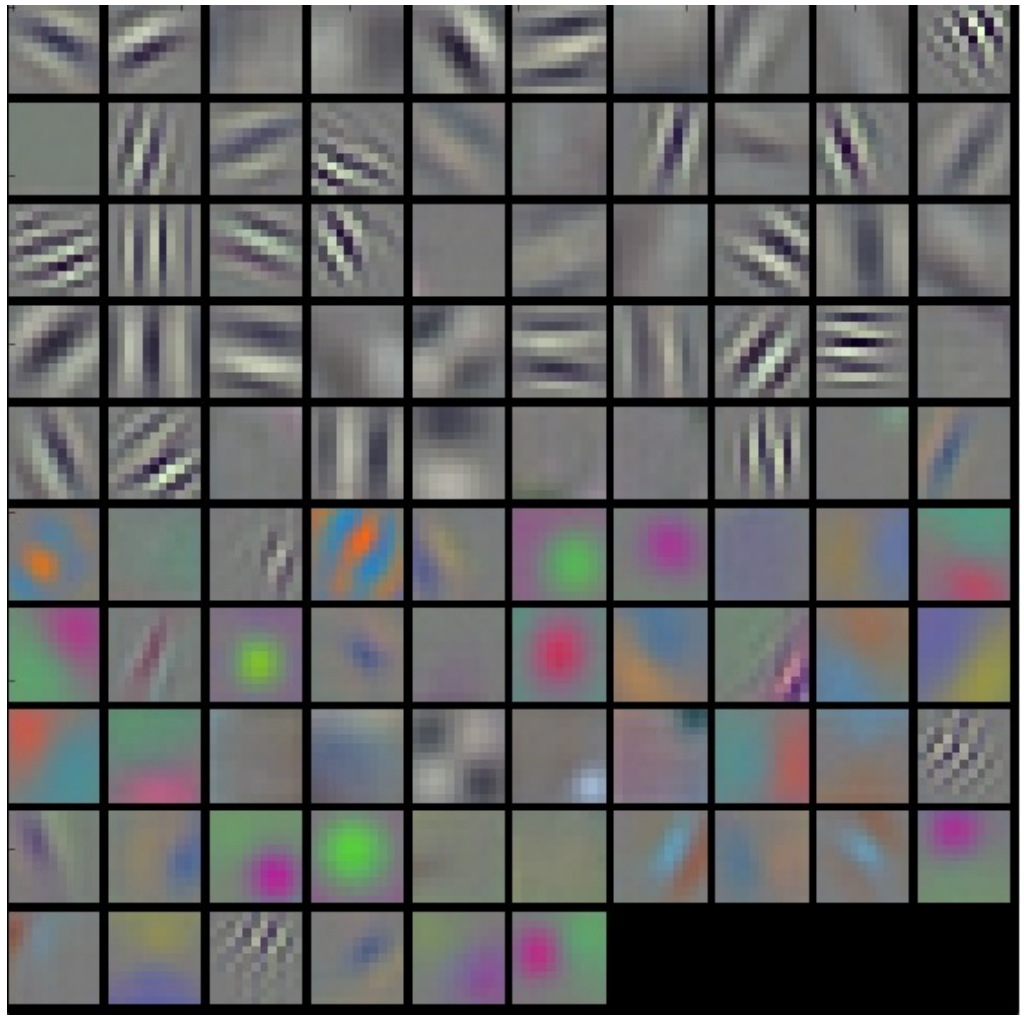
  (K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR 2016, pages 770–778)

AlexNet (winner ILSVRC 2012)



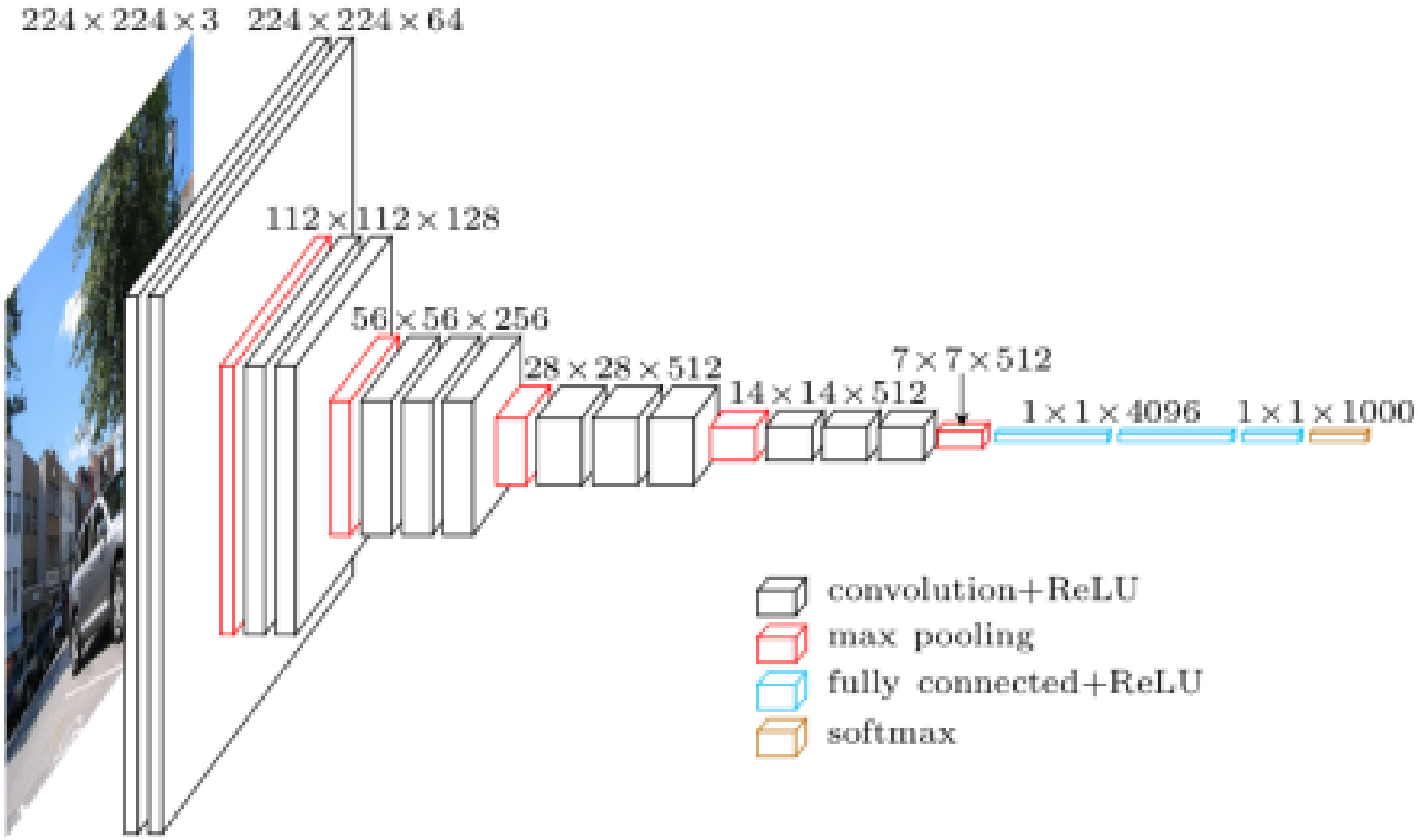
AlexNet (winner ILSVRC 2012)

- 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



Images that maximize the response of the filters (AlexNet)

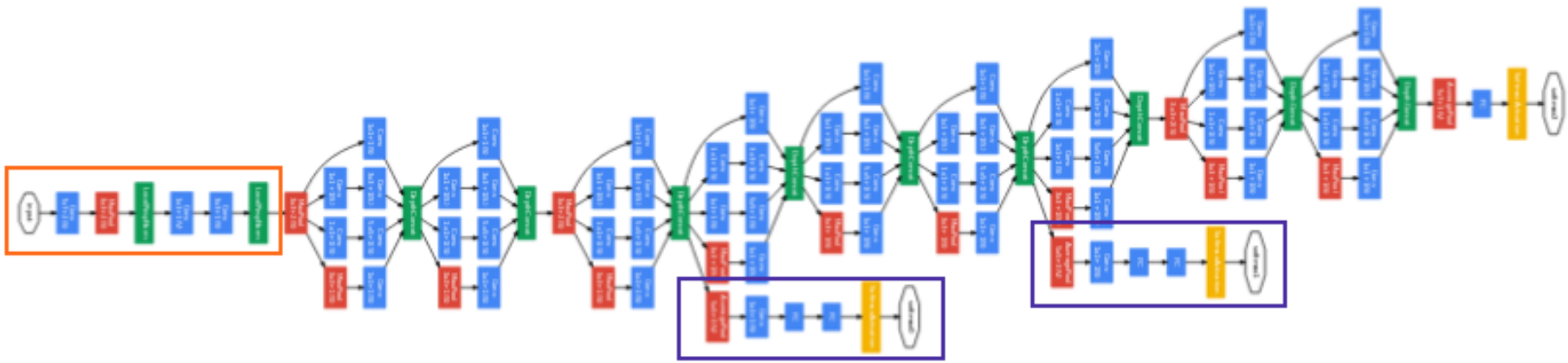
VGGNet (2nd place ILSVRC 2014)



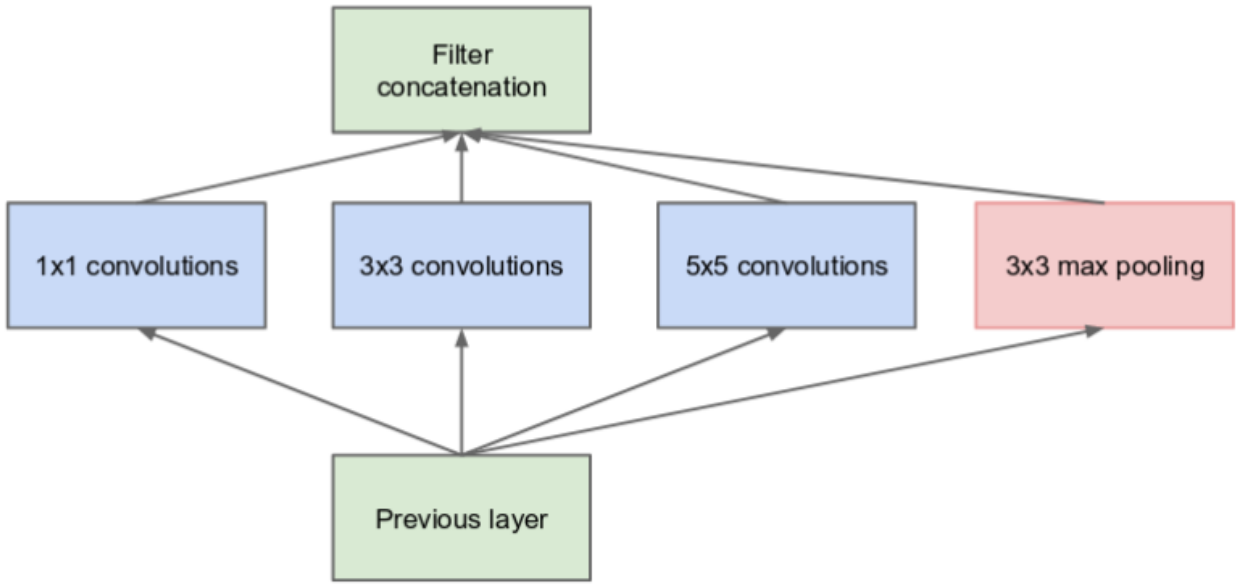
VGGNet (2nd place ILSVRC 2014)

- 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)

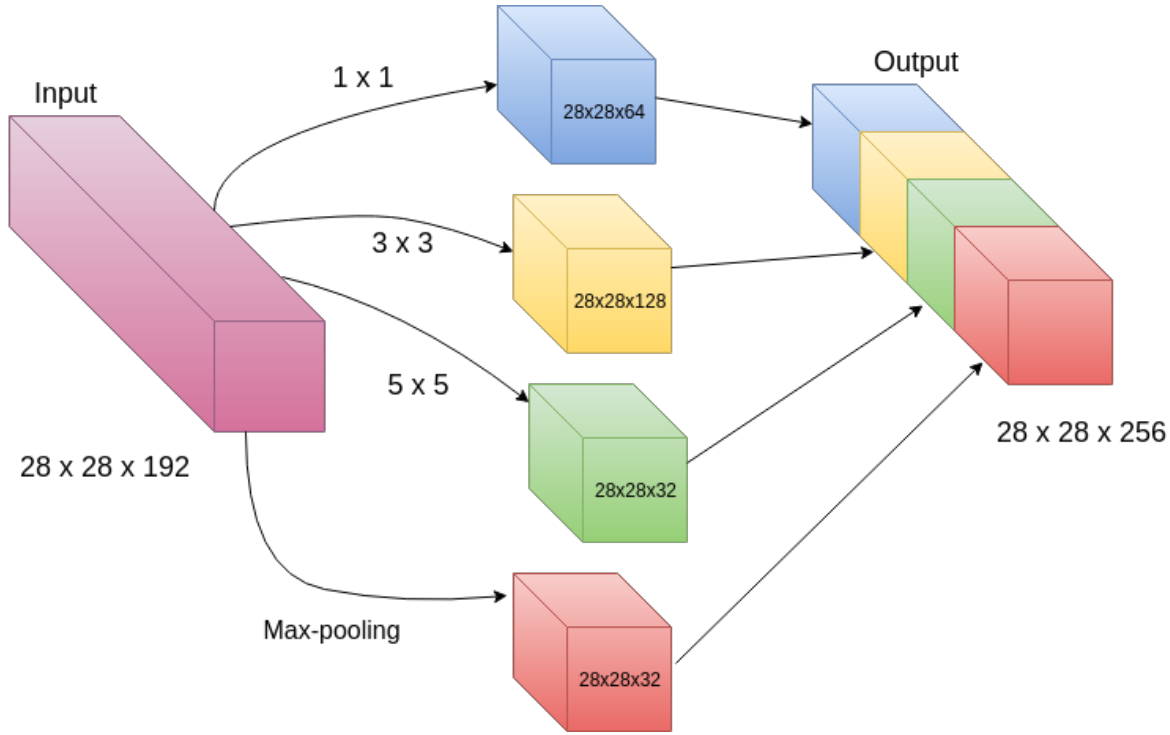


Inception – naïve version



(a) Inception module, naïve version

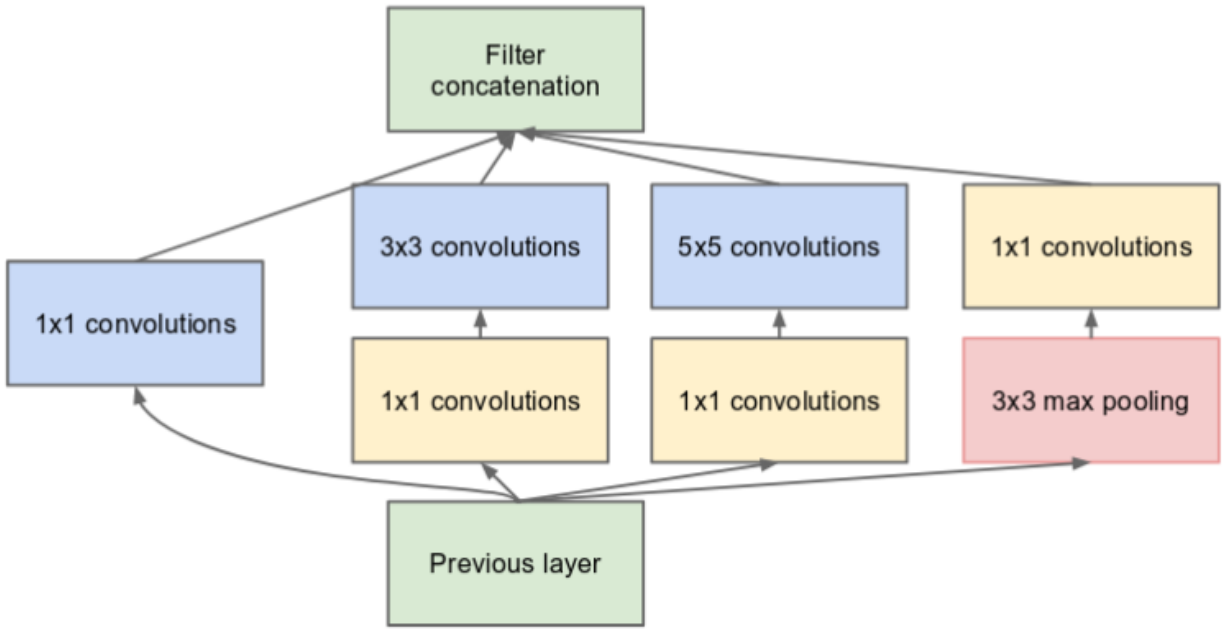
Inception – naïve version example



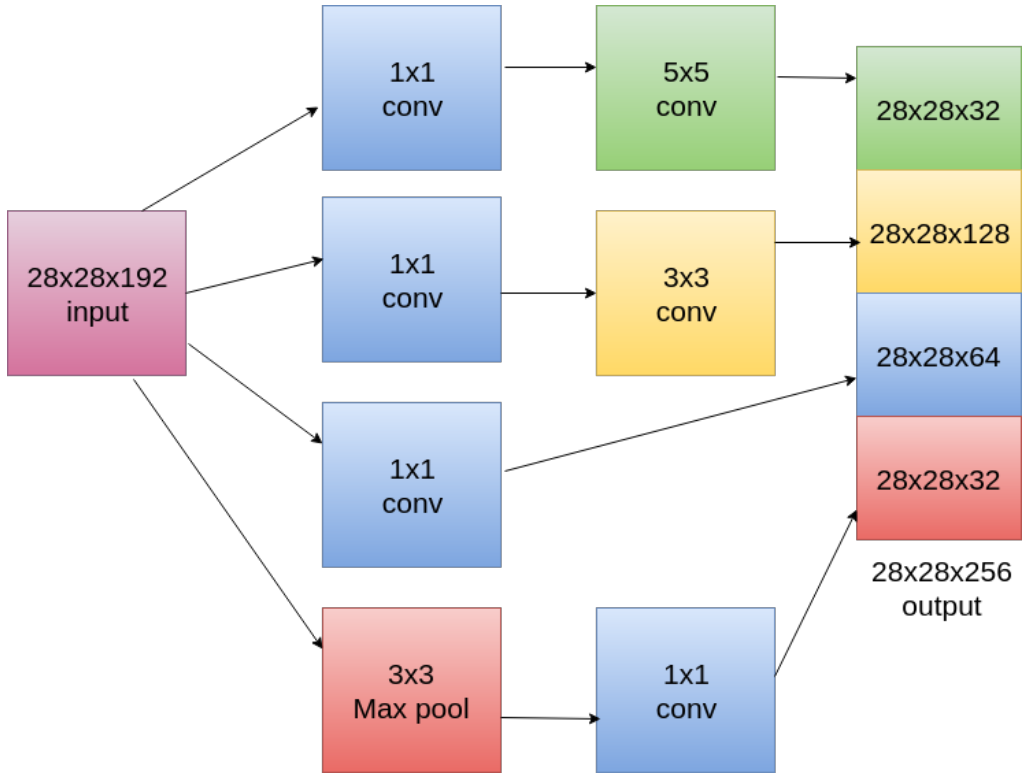
Camada de entrada = 28x28x192

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

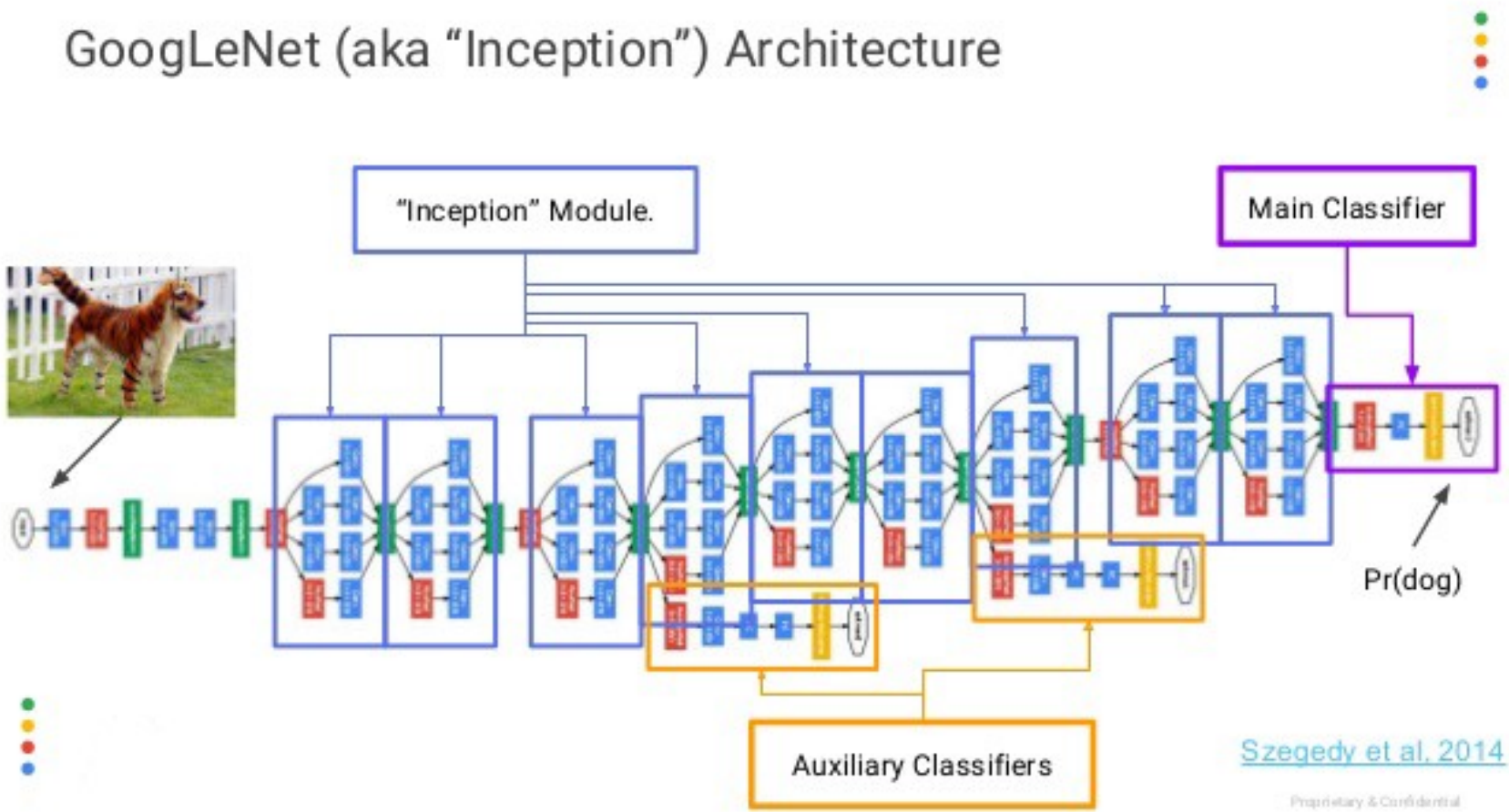


(b) Inception module with dimension reductions



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)

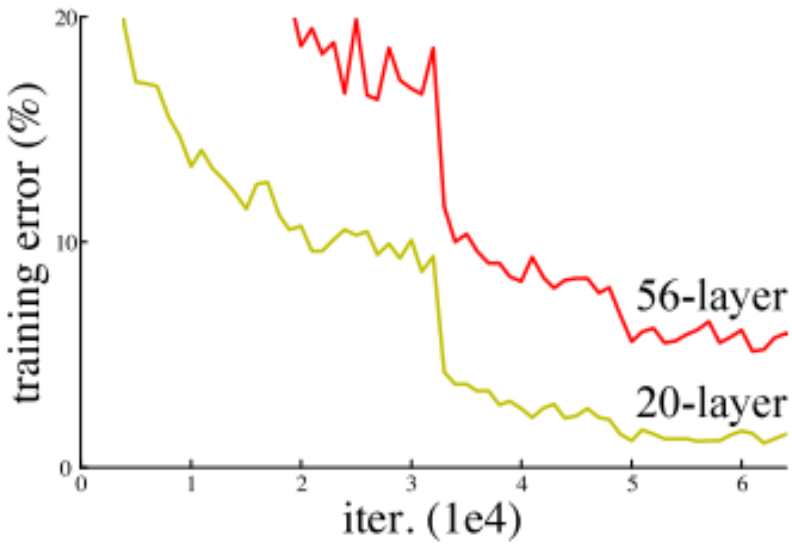


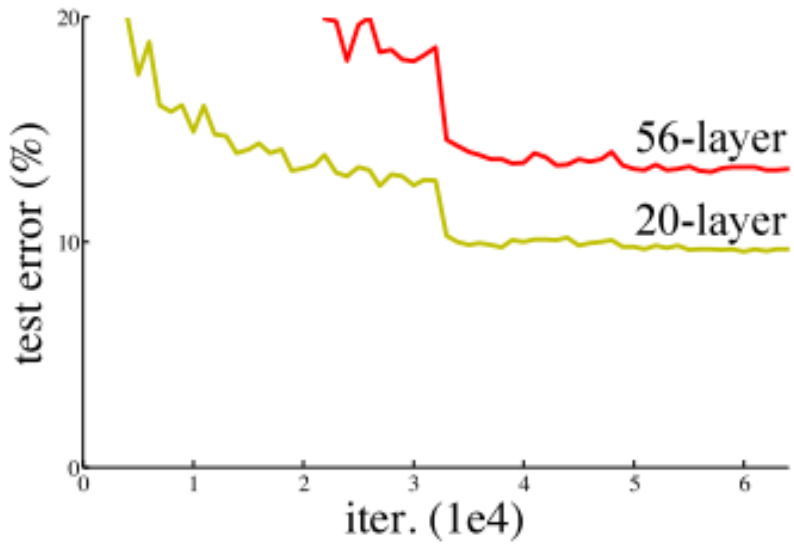
Inception - GoogleLeNet (winner ILSVRC 2014)

- 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)

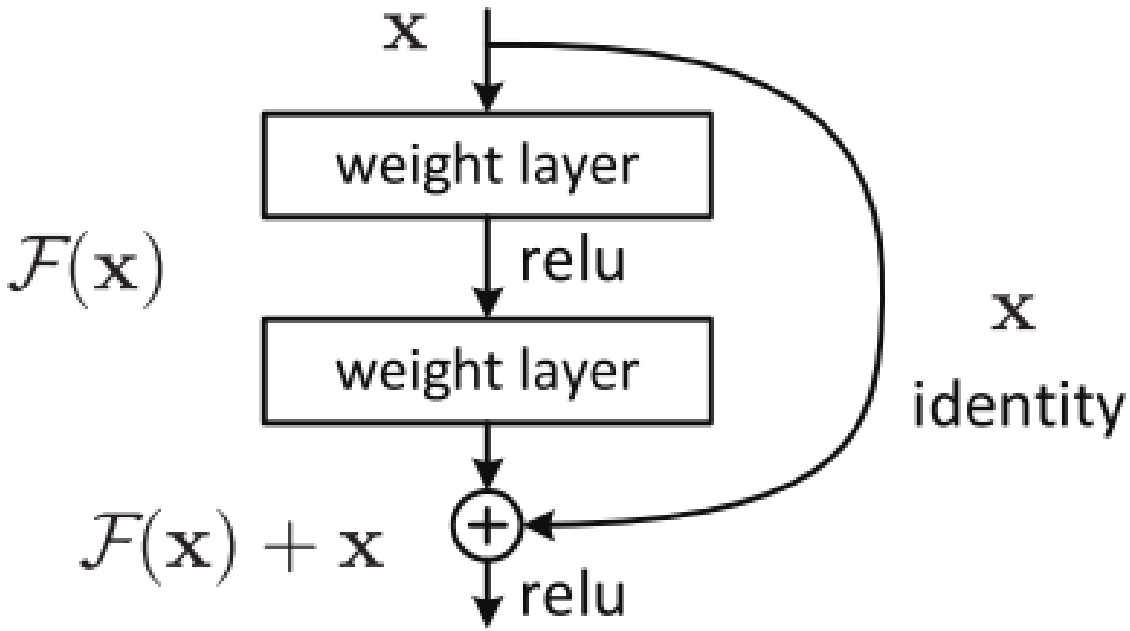
ResNet (winner ILSVRC 2015)

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



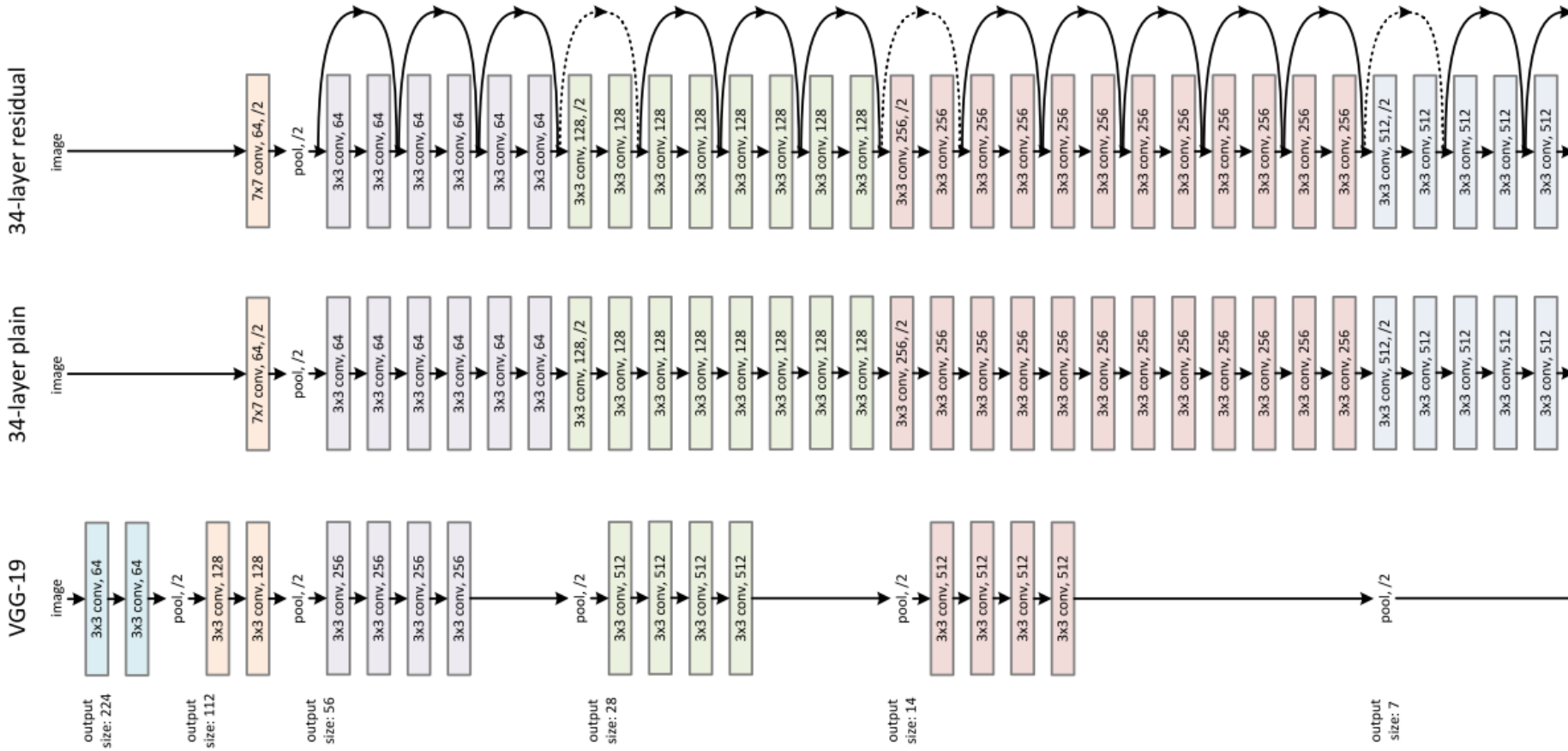


ResNet (winner ILSVRC 2015)



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)



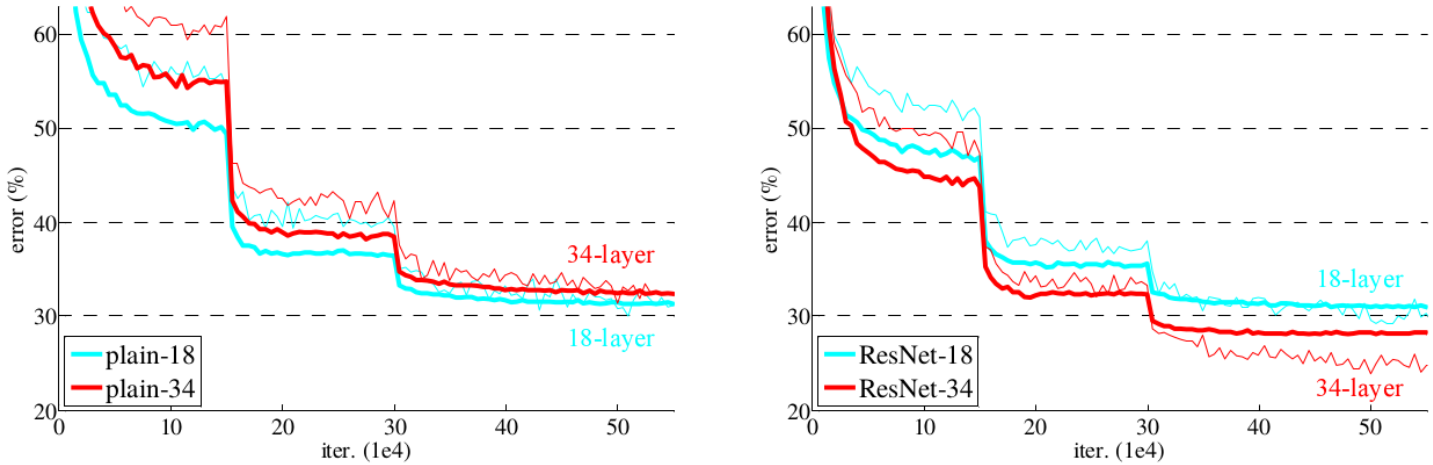


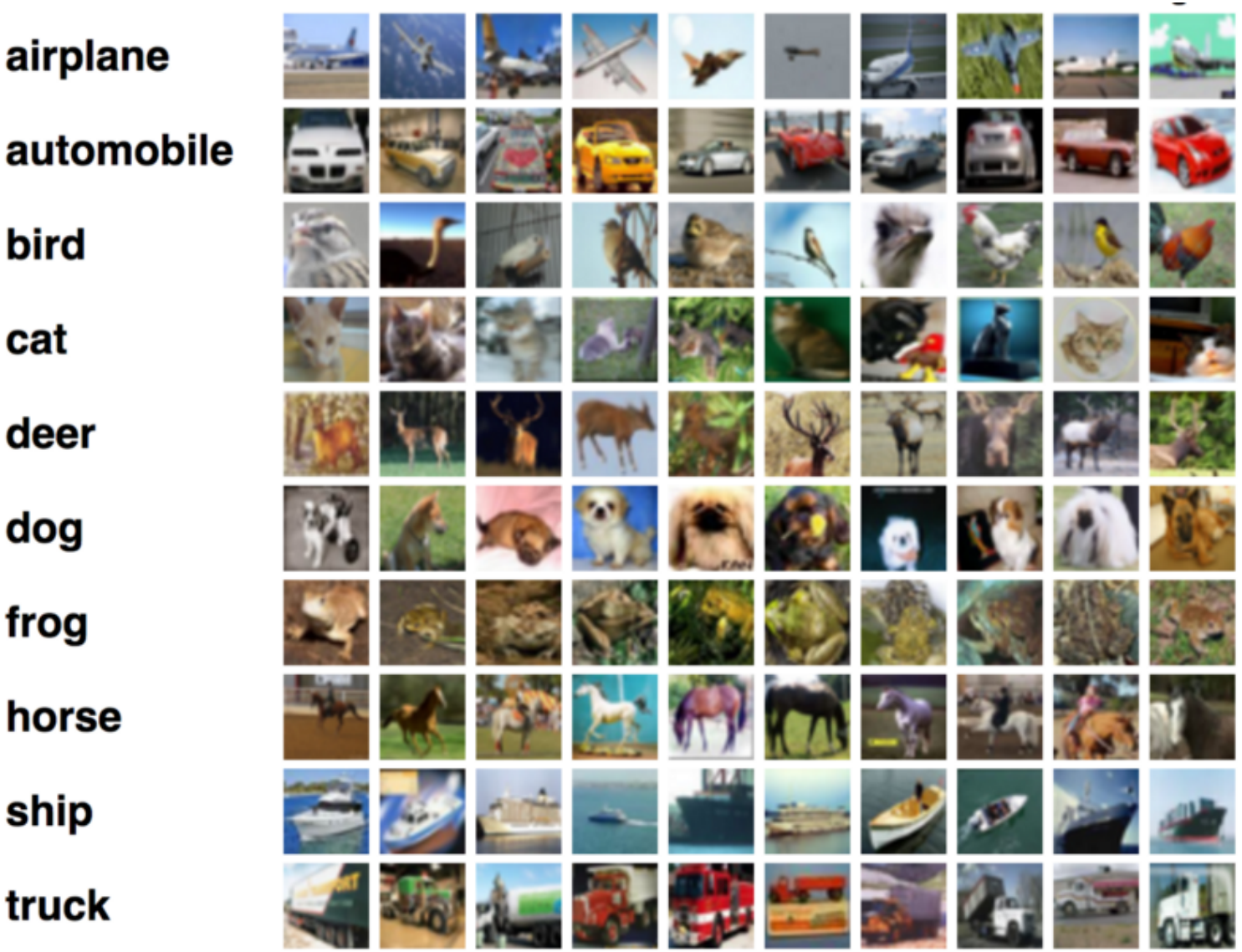
Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

ResNet (winner ILSVRC 2015)

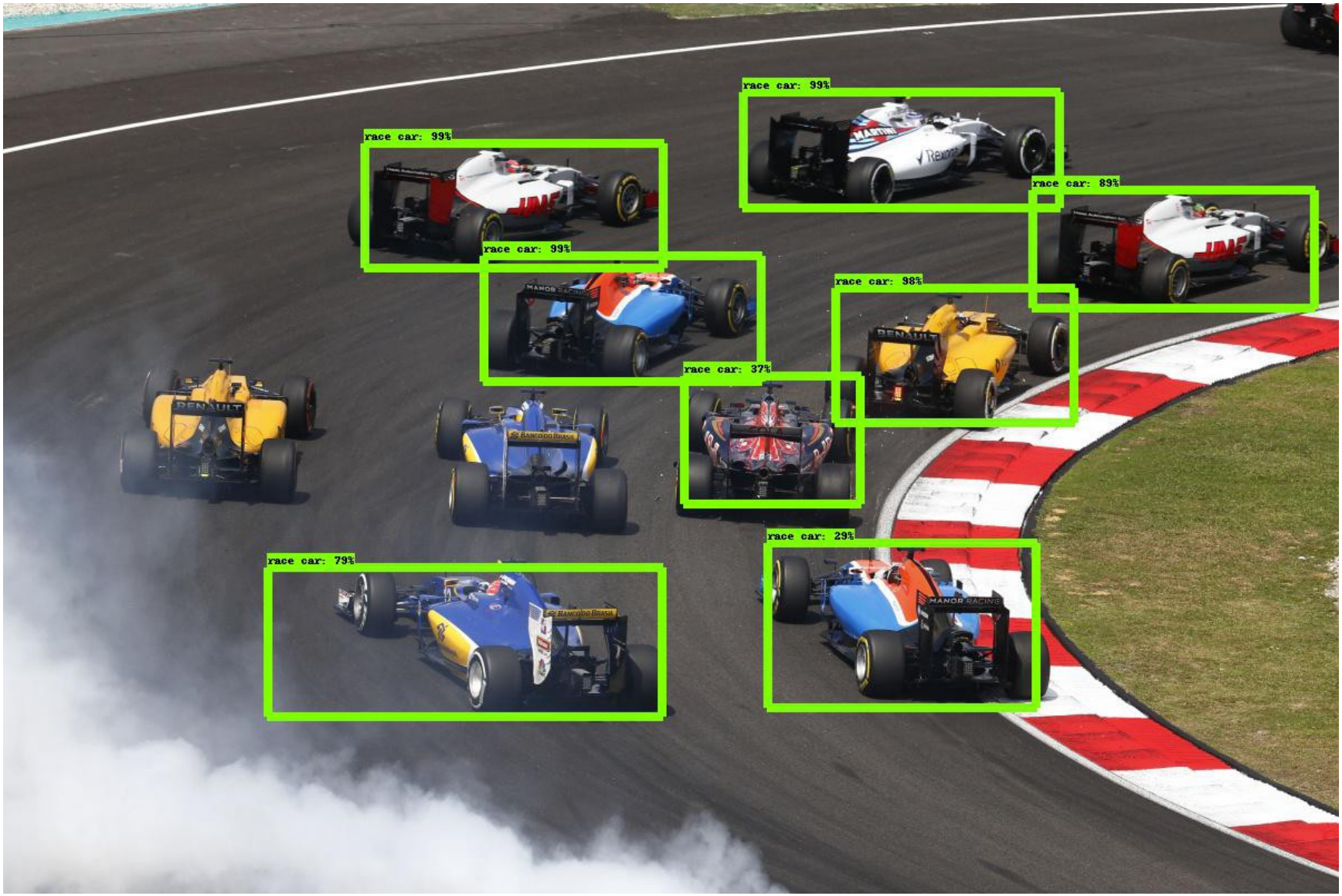
- identificam uma degradação associada ao aumento de camadas, que aparentemente não é *overfitting* nem *vanishing gradient*
- propõe módulo para aprender o resíduo da transformação desejada
- aplica *batch normalization* após cada convolução e antes da ativação
- não usa *dropout*
- conseguem treinar 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 (?)

DL is being used to address multiple tasks in CV

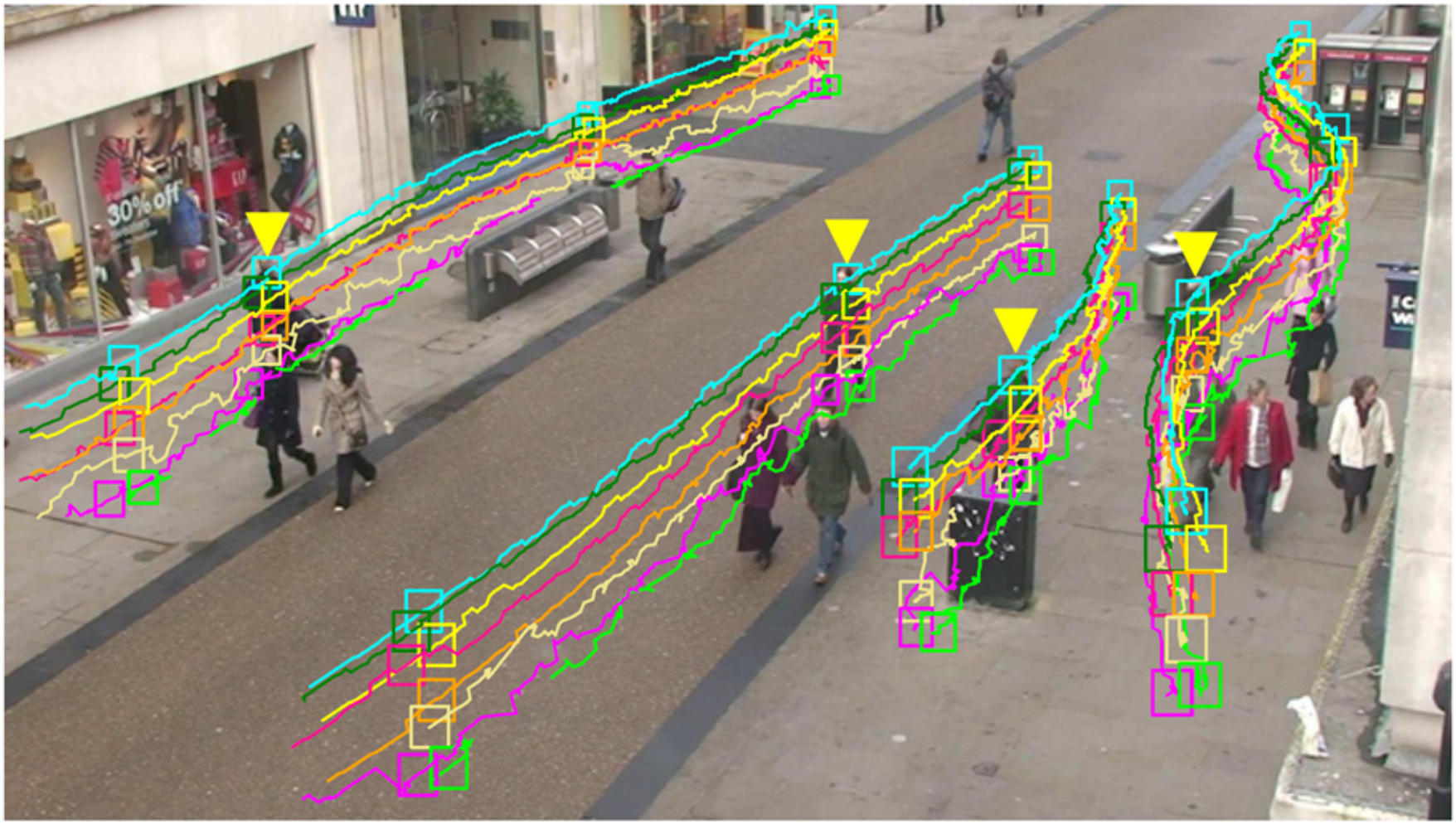
Image classification



Object detection



Object tracking in video



Semantic segmentation

Scene description

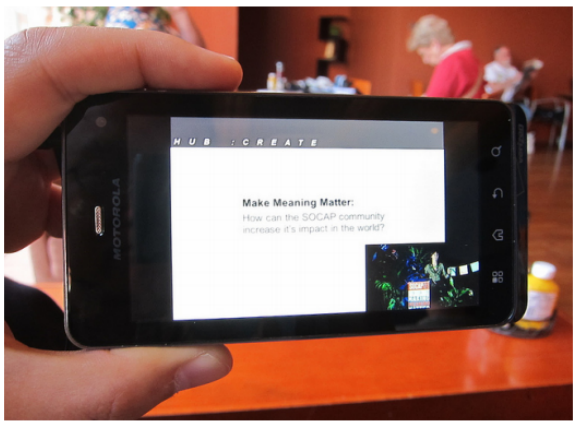
A female tennis player in action on the court.

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.

CNN for image segmentation

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

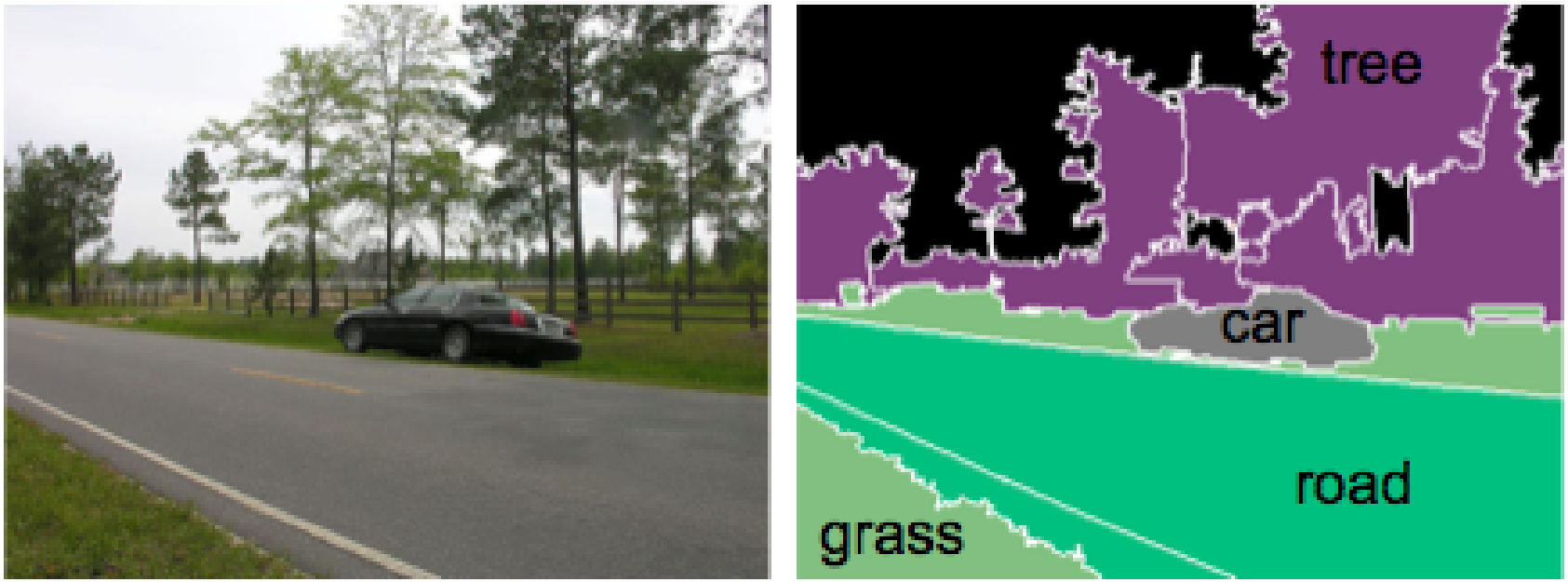

tree
car
road
grass

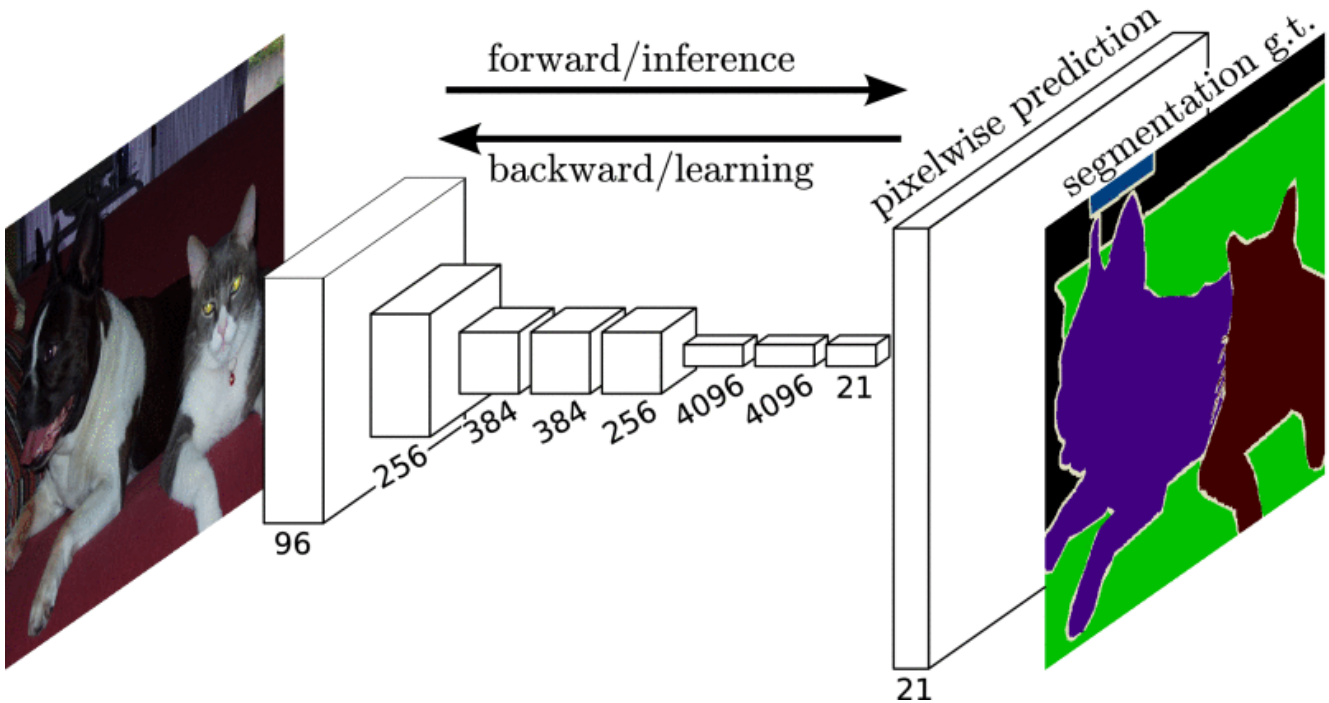
Image segmentation

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.


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

U-Net

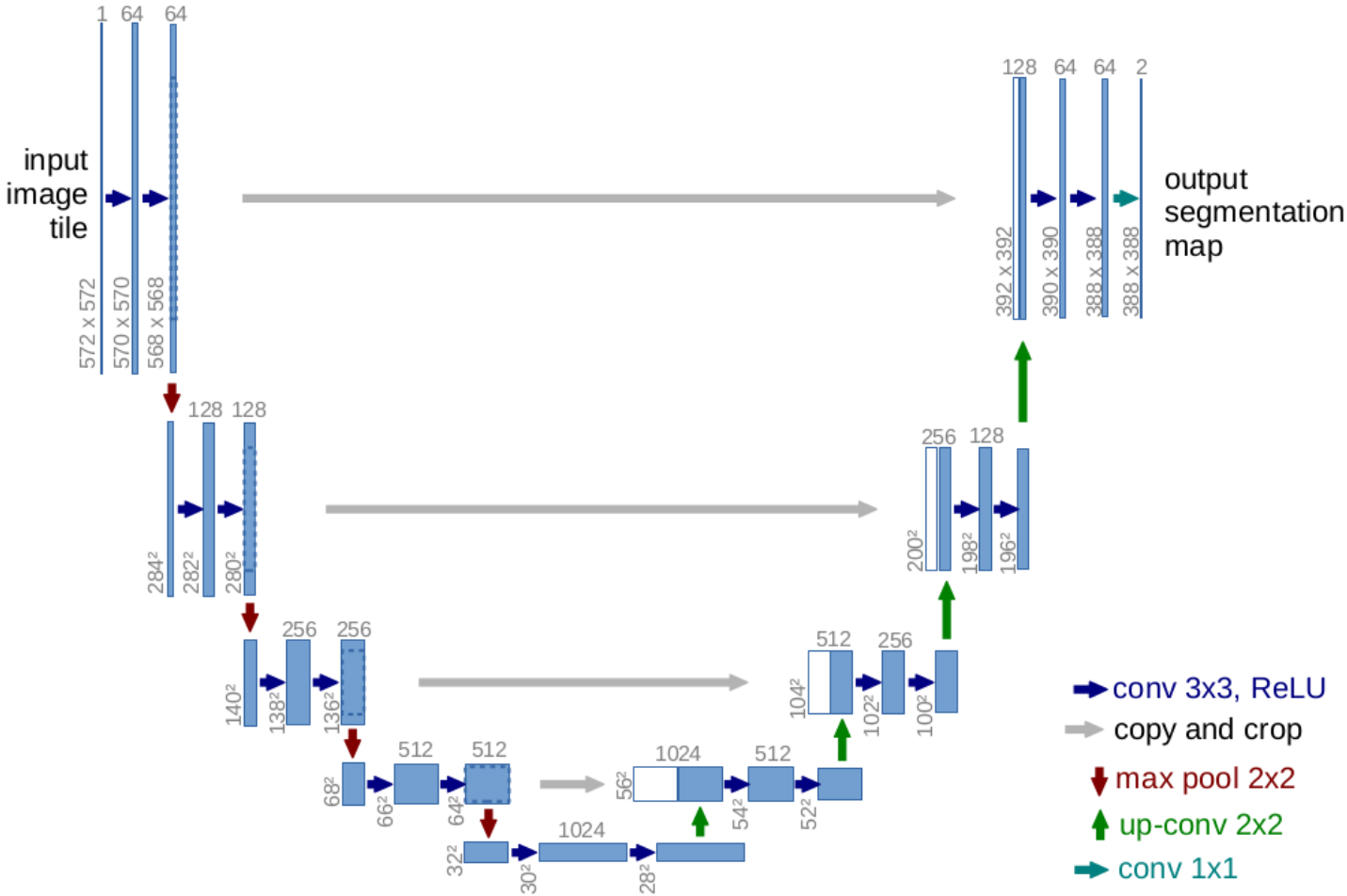
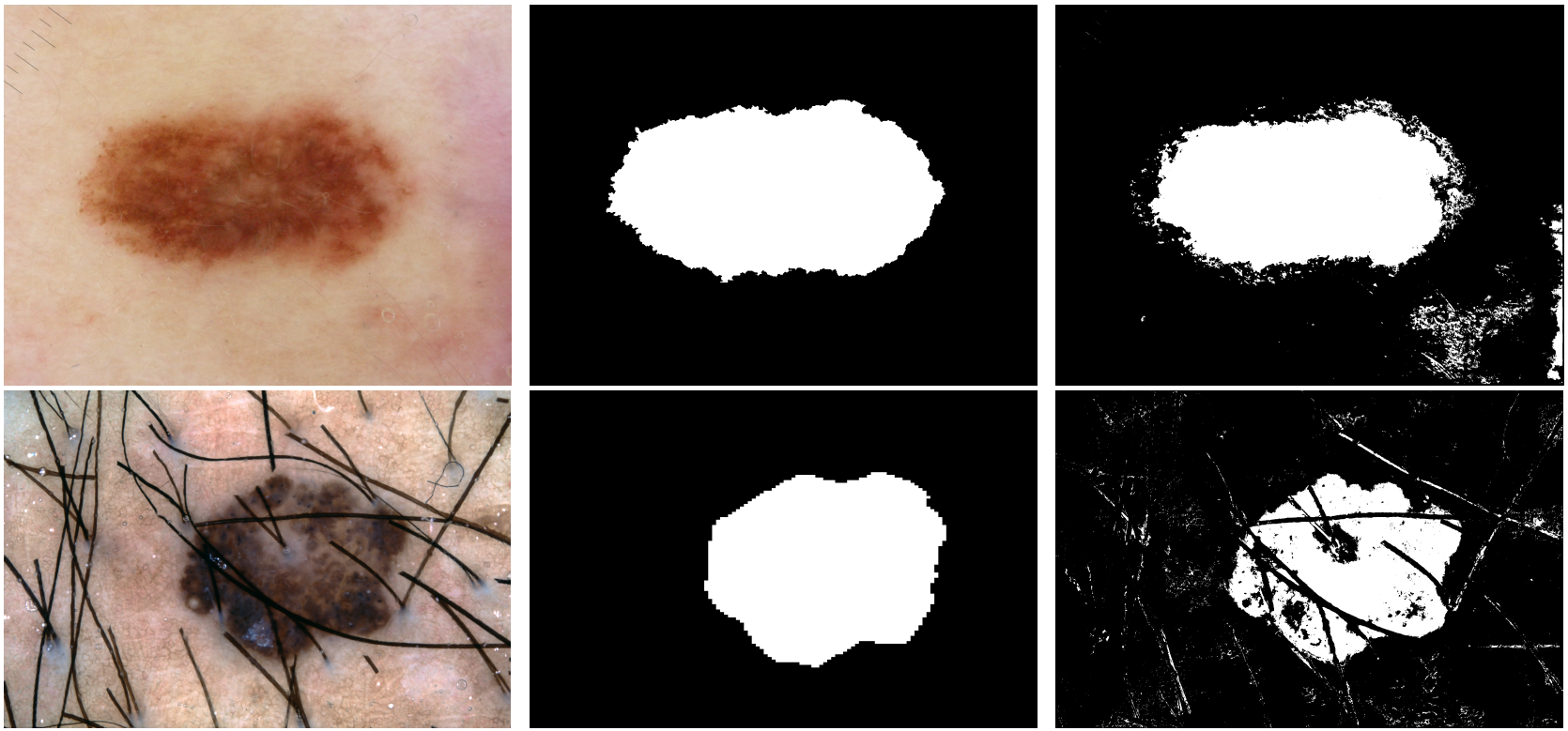


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows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

Application example



(ISIC 2018 dataset)

CNN for object detection/recognition

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

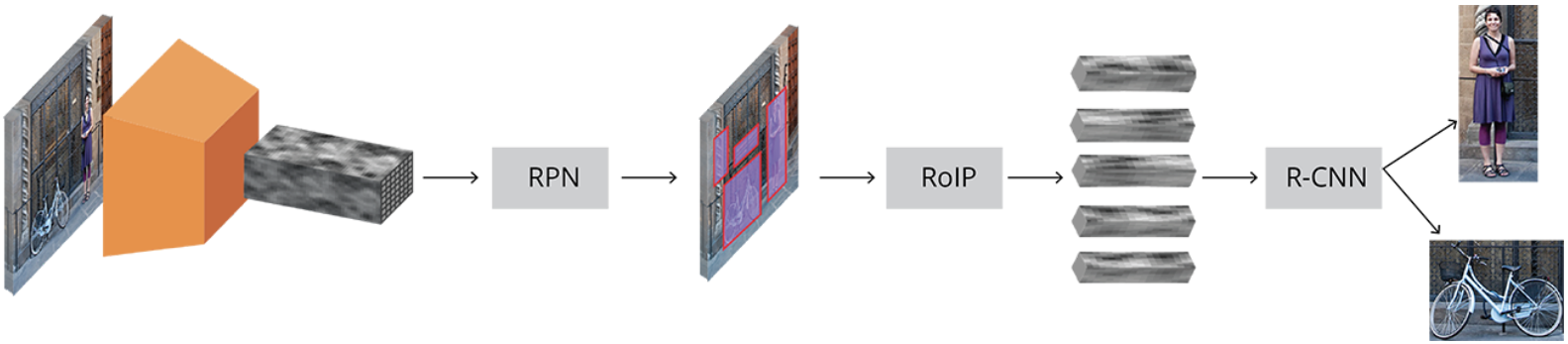
Object recognition em imagens

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*).

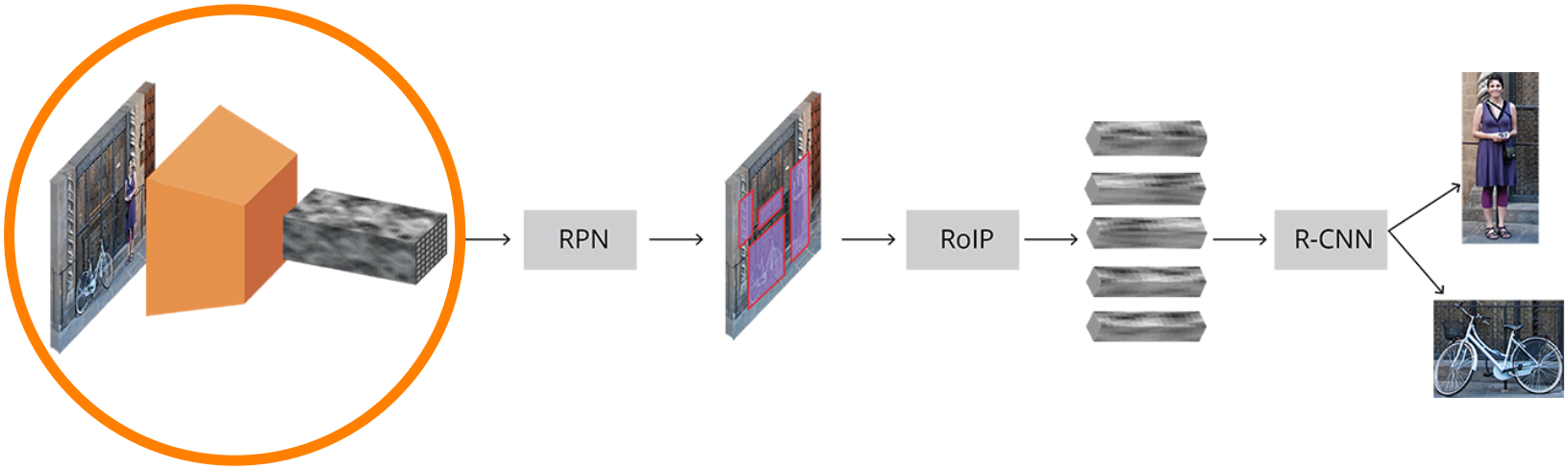
Existem os modelos two-step (ex., Faster R-CNN) e *single shot* (SSD ou YOLO)

Faster R-CNN architecture



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

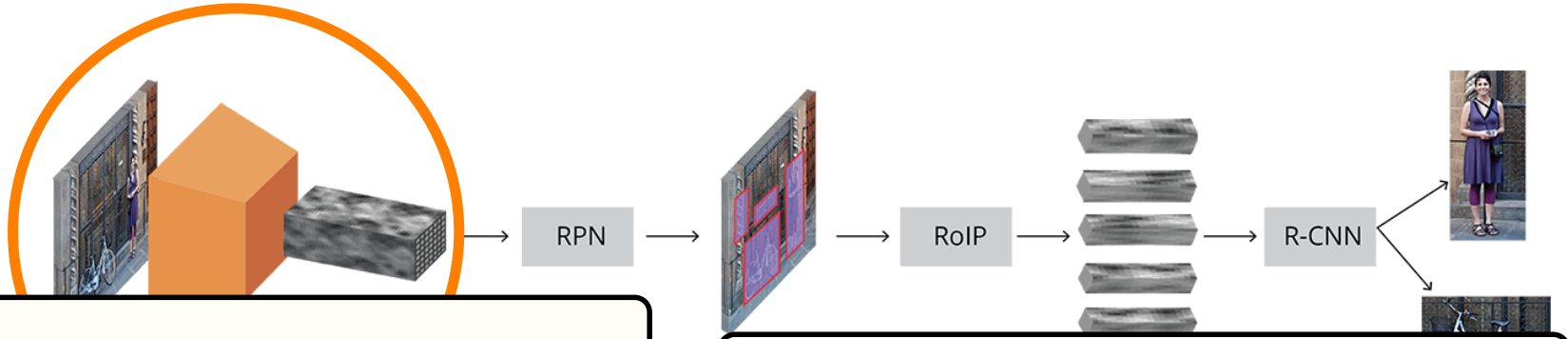
Faster R-CNN architecture



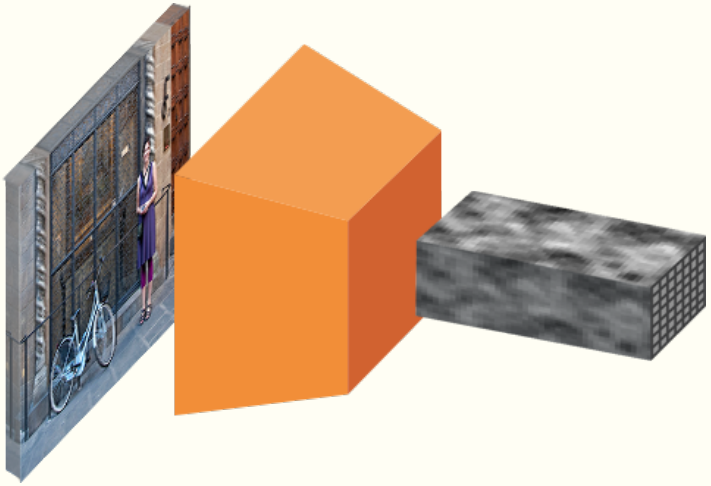
Feature extraction

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

Faster R-CNN architecture



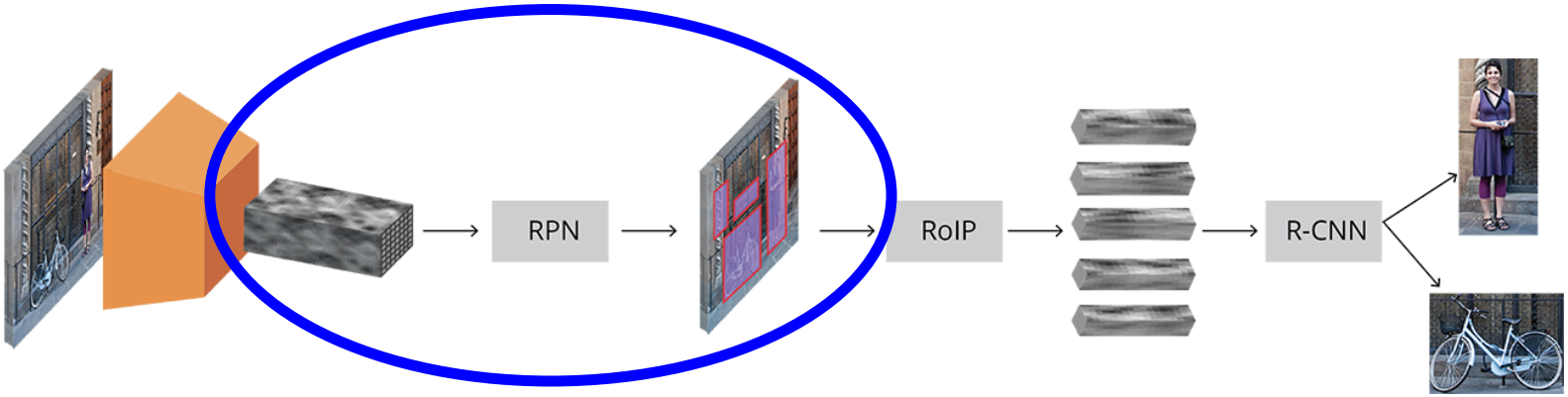
Feature extraction



- 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/

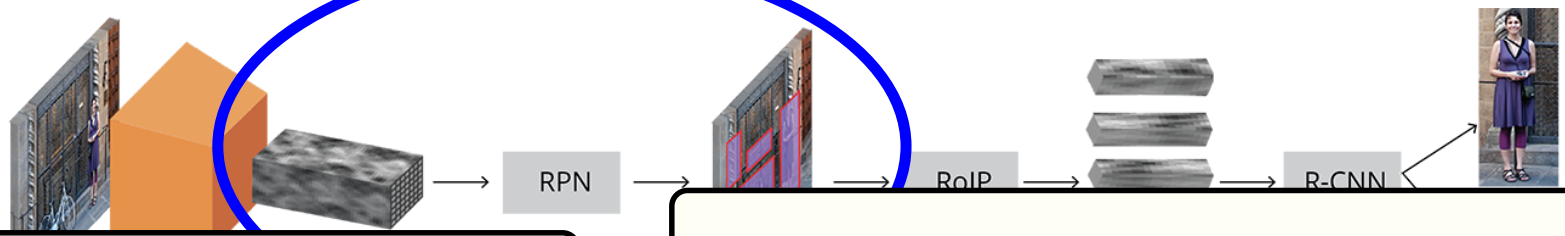
Faster R-CNN architecture

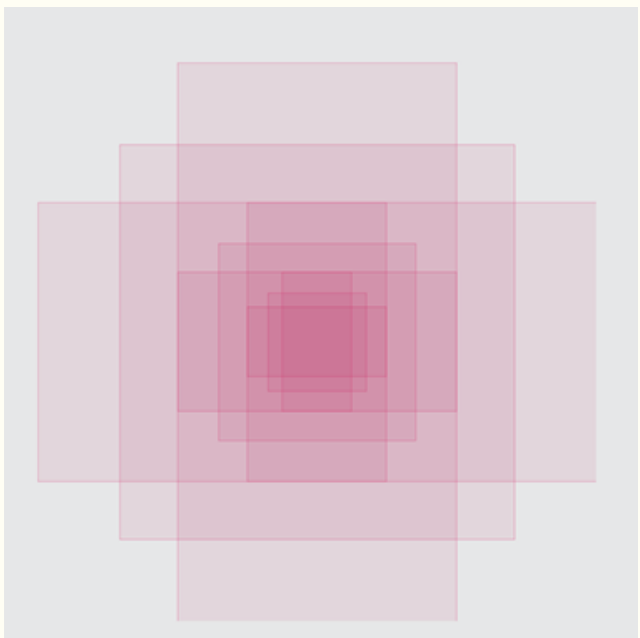


Region proposal network

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

Faster R-CNN architecture

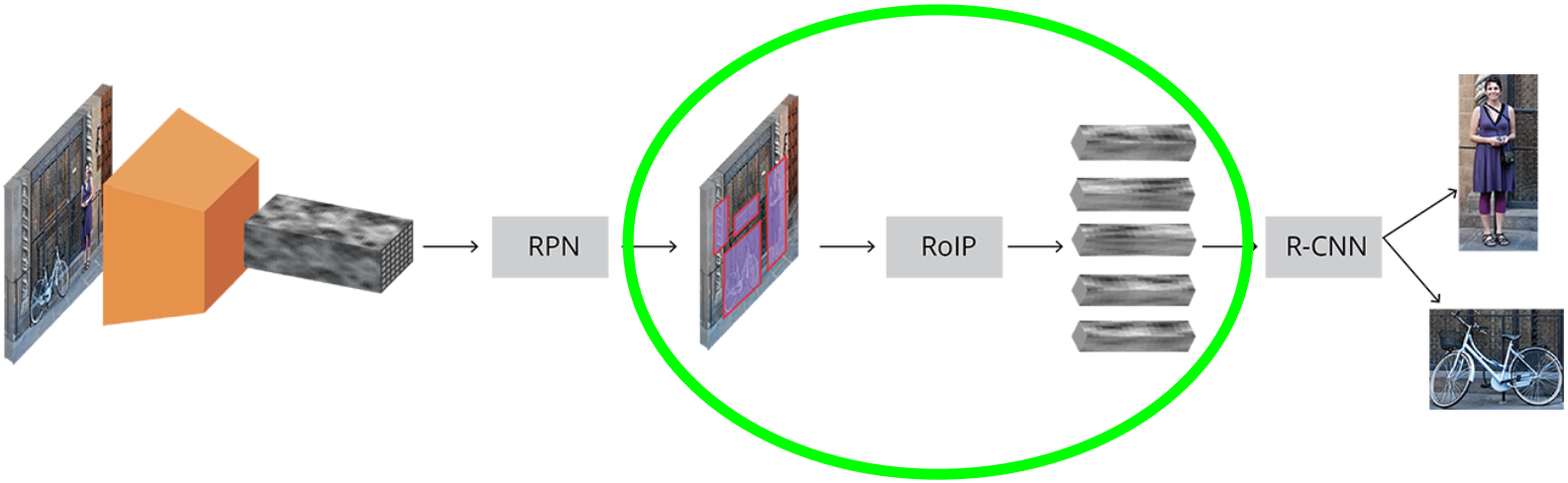




anchors

- 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

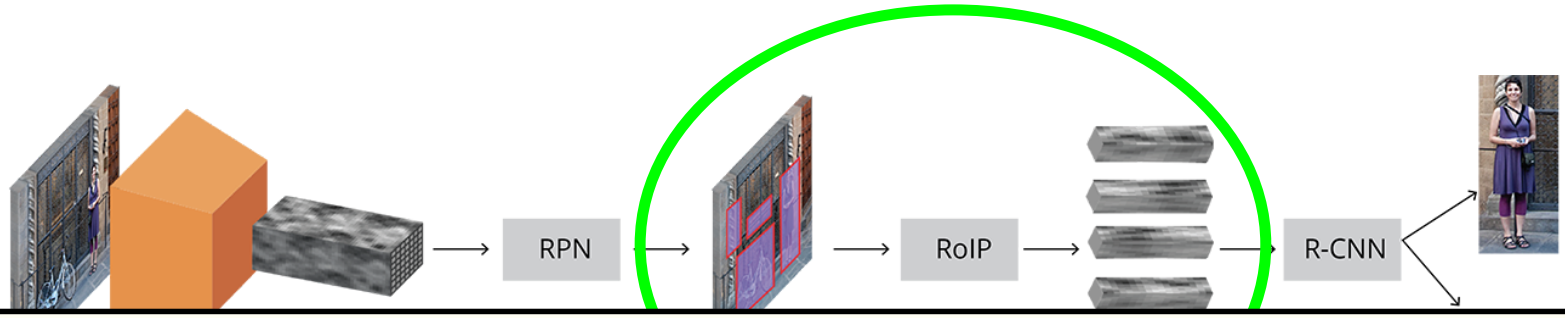
Faster R-CNN architecture



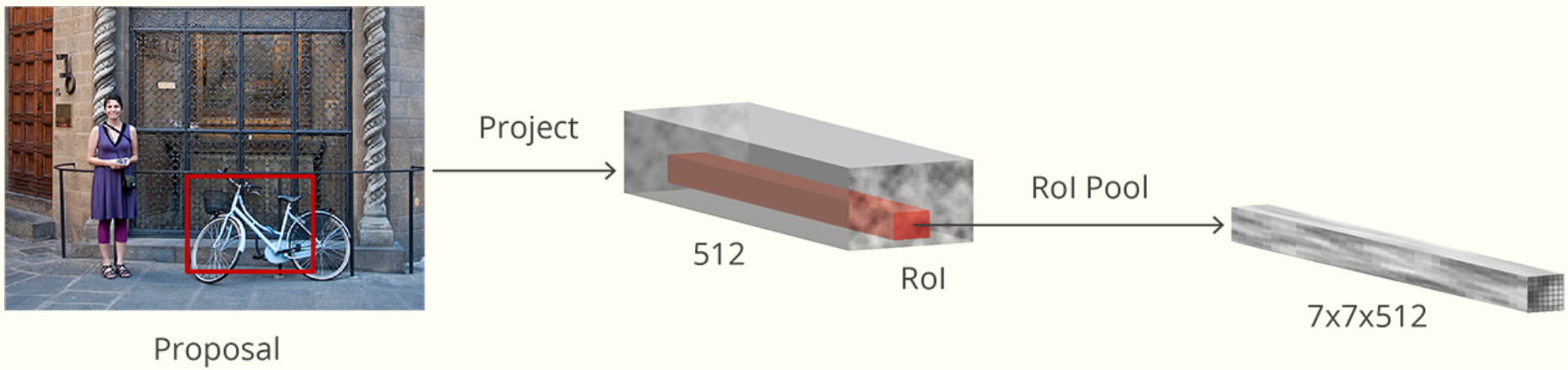
Region of interest pooling

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

Faster R-CNN architecture

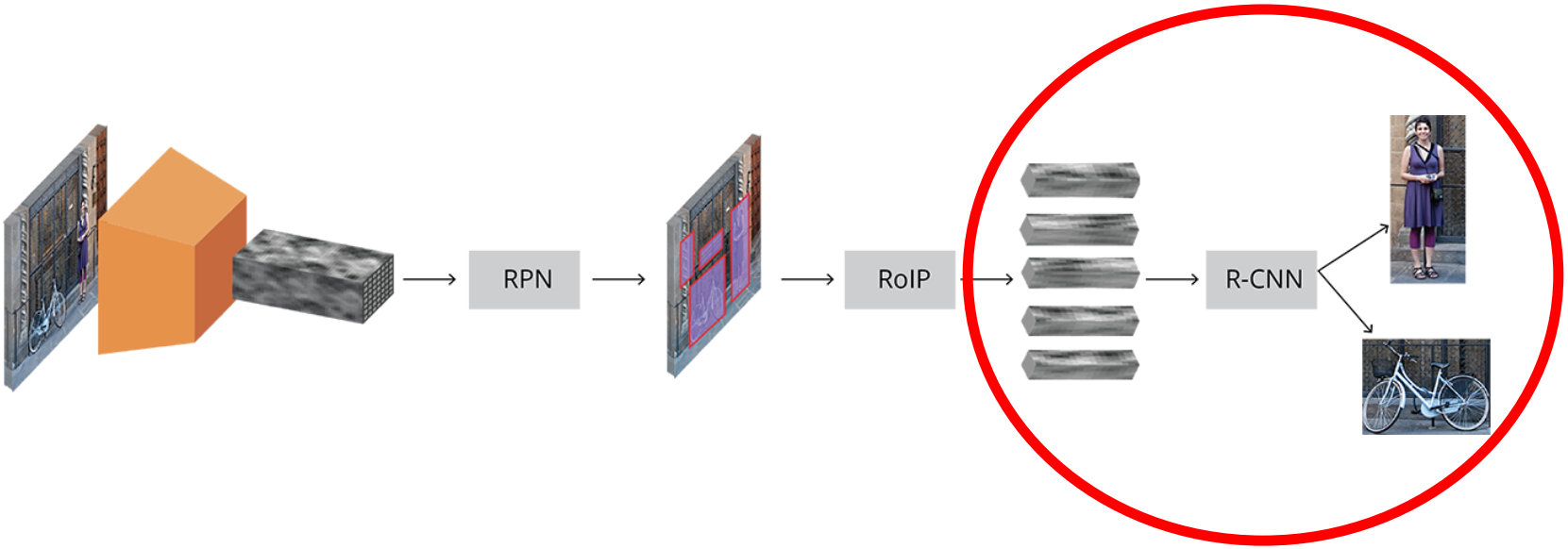


Region of interest pooling



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

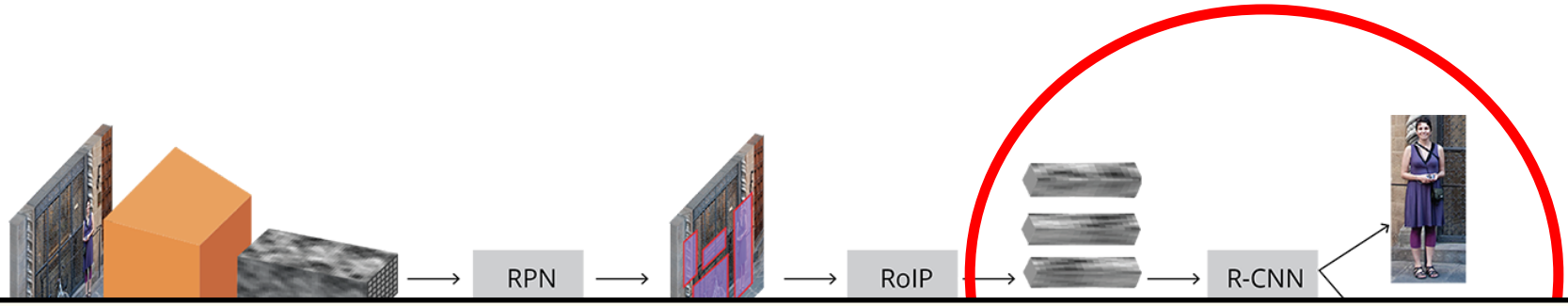
Faster R-CNN architecture



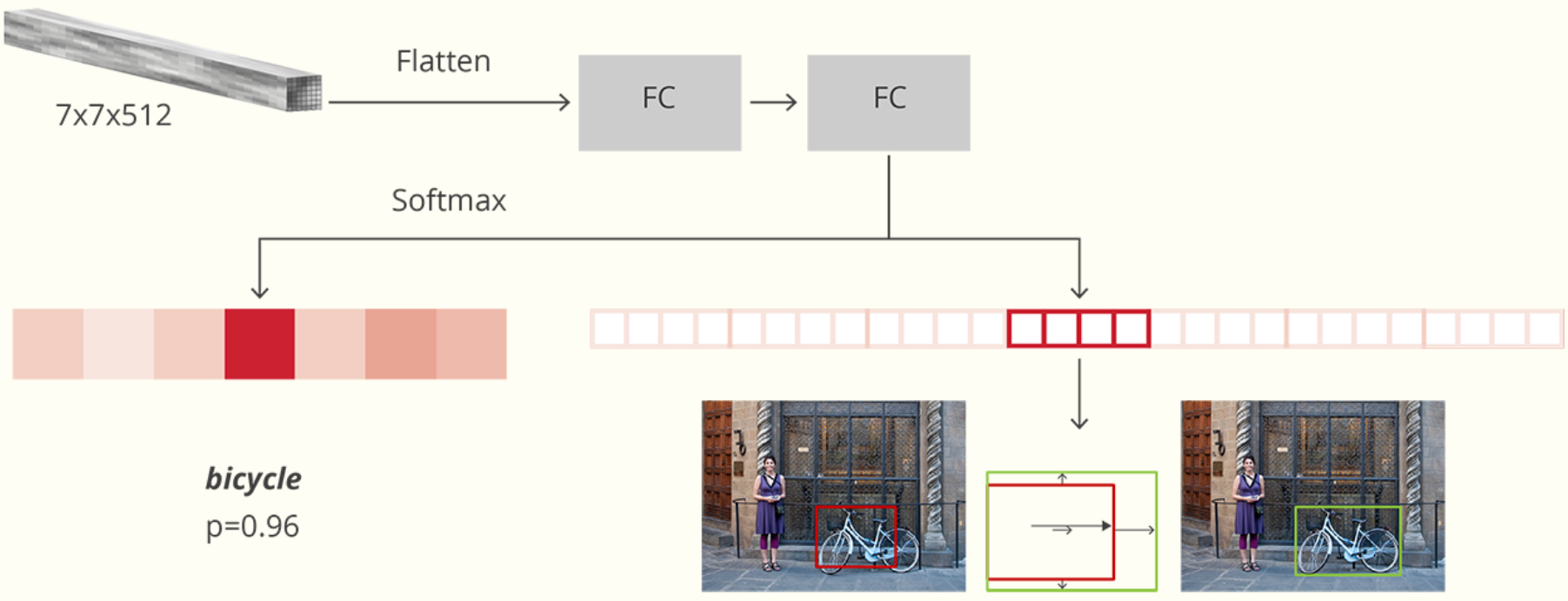
Region classification

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

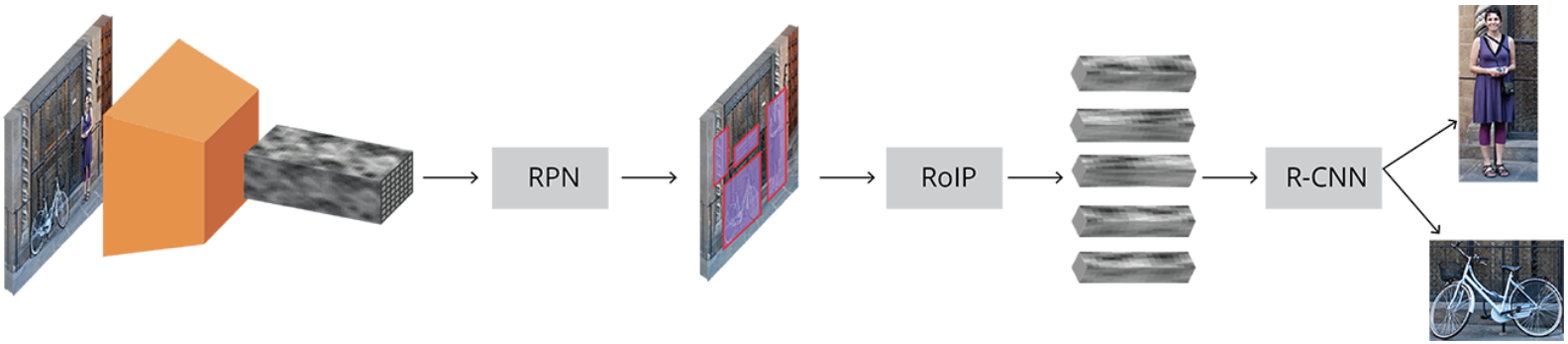
Faster R-CNN architecture



Region classification



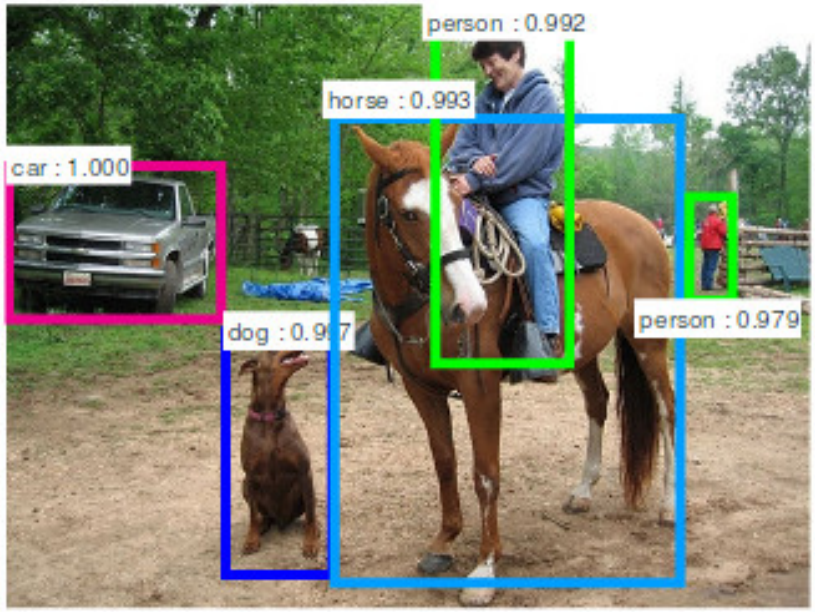
Faster R-CNN architecture

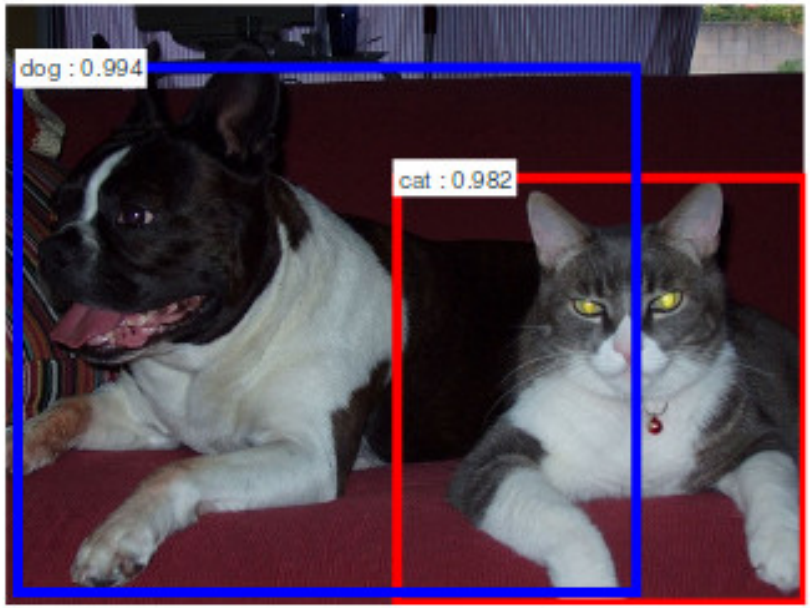


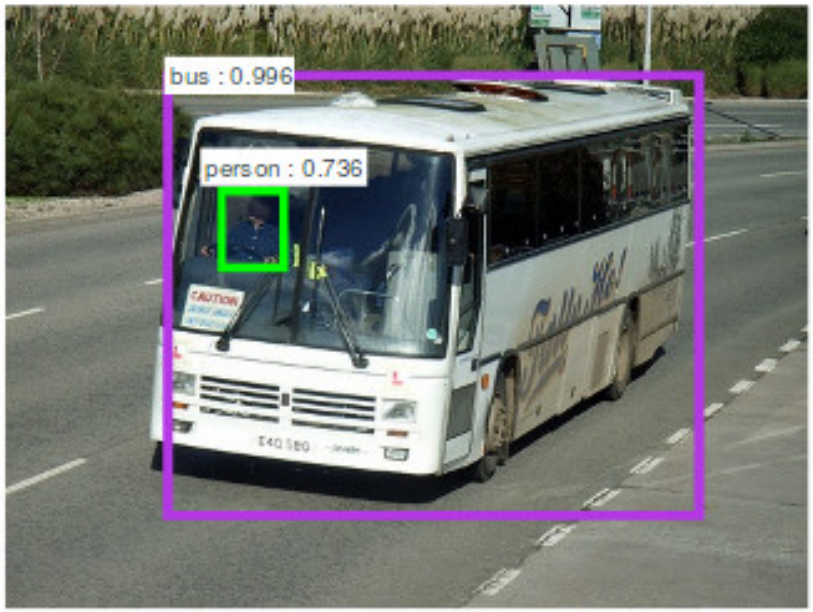
Input is an image

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/

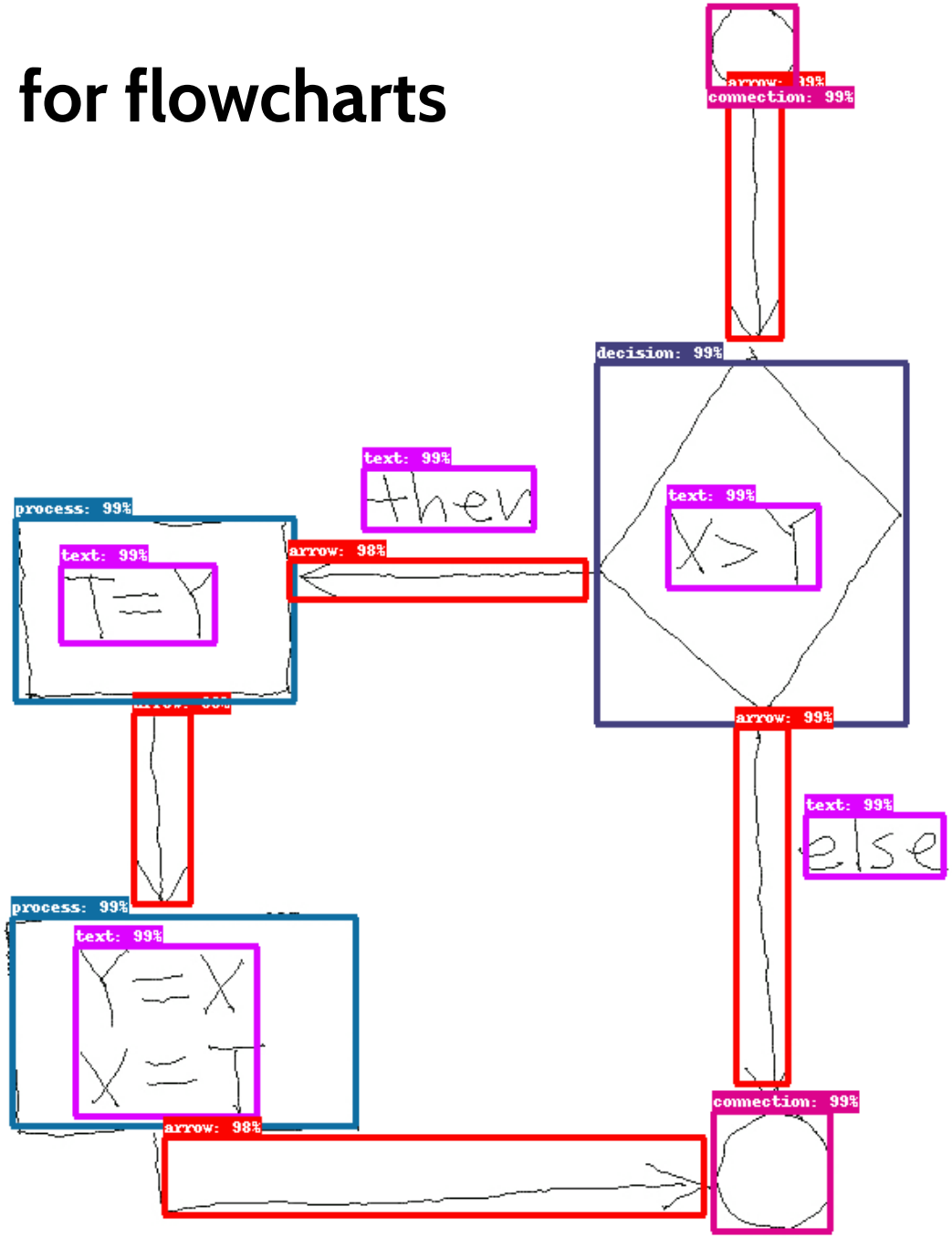

person : 0.992
horse : 0.993
car : 1.000
person : 0.979


dog : 0.994
cat : 0.982

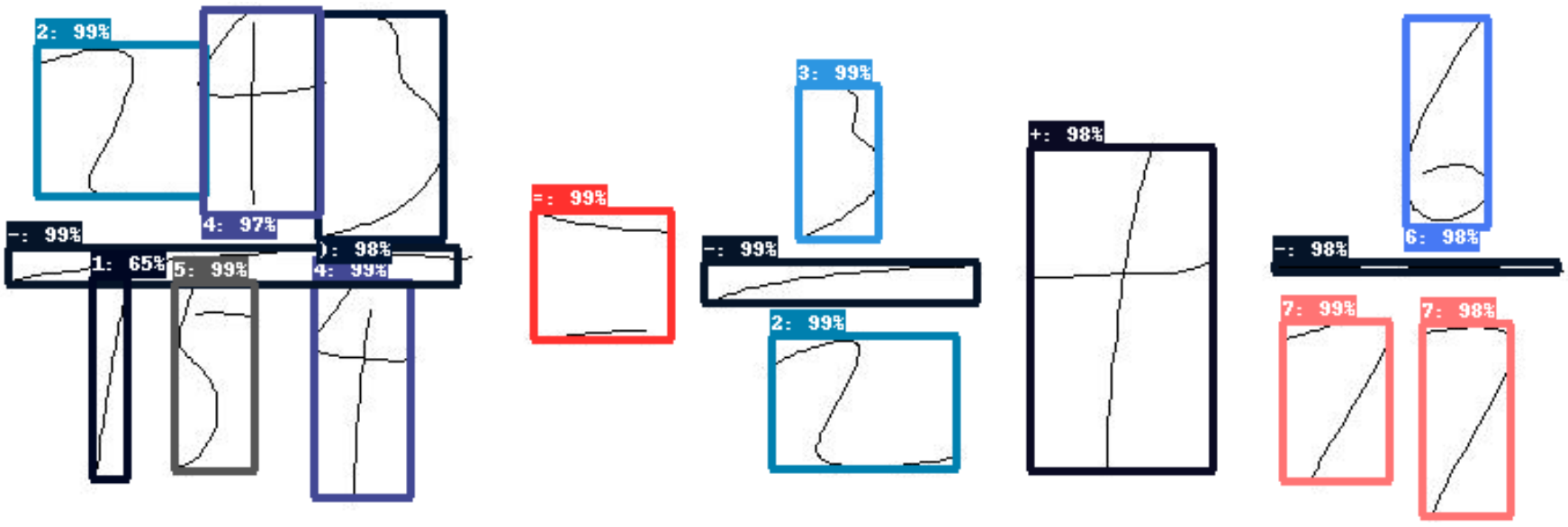

bus : 0.996
person : 0.736


boat : 0.970
person : 0.983
person : 0.983
person : 0.925
person : 0.989
Lickety Split

Results for flowcharts



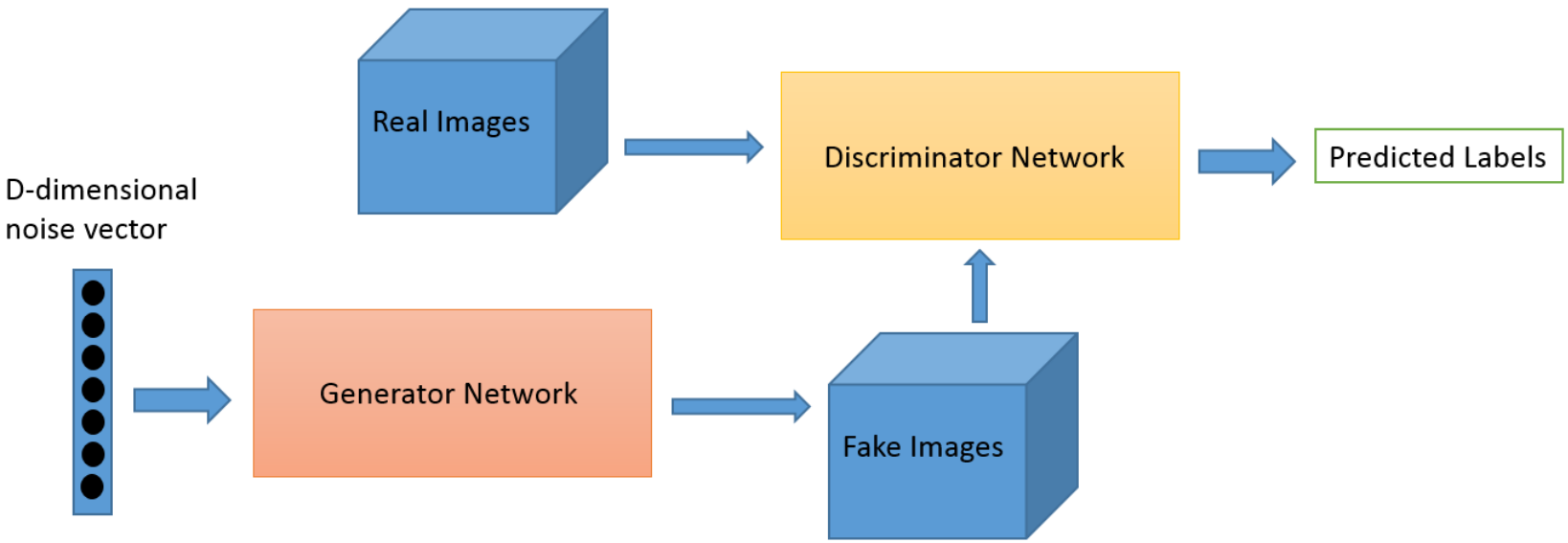
Results for math expressions



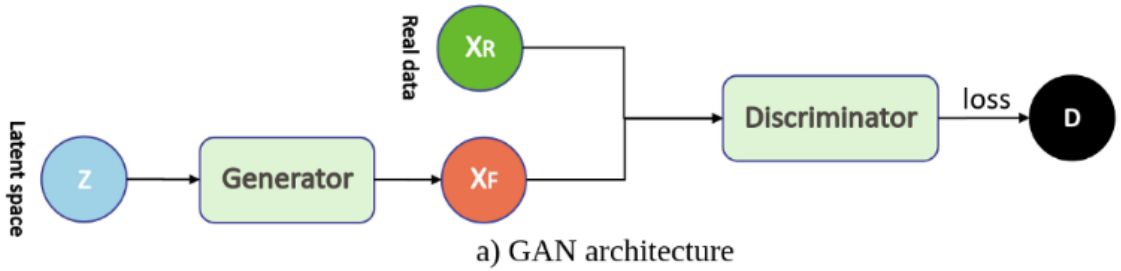
Other DL network models

- **GAN** – image generation / adversarial training
- **Siamese network, triplet network** — image retrieval
- **RNN, LSTM, GRU** – temporal/sequential data processing
- **Geometric Deep Learning** deals with the extension of Deep Learning techniques to graph/manifold structured data. http://geometricdeeplearning.com/

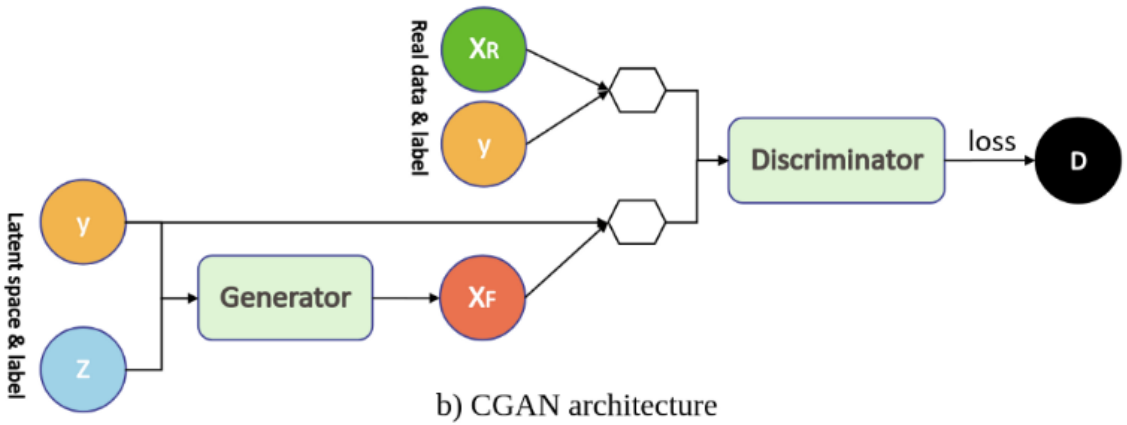
GAN schema



(fonte: O'Reilly)



a) GAN architecture



b) CGAN architecture

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

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

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

GAN

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

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)

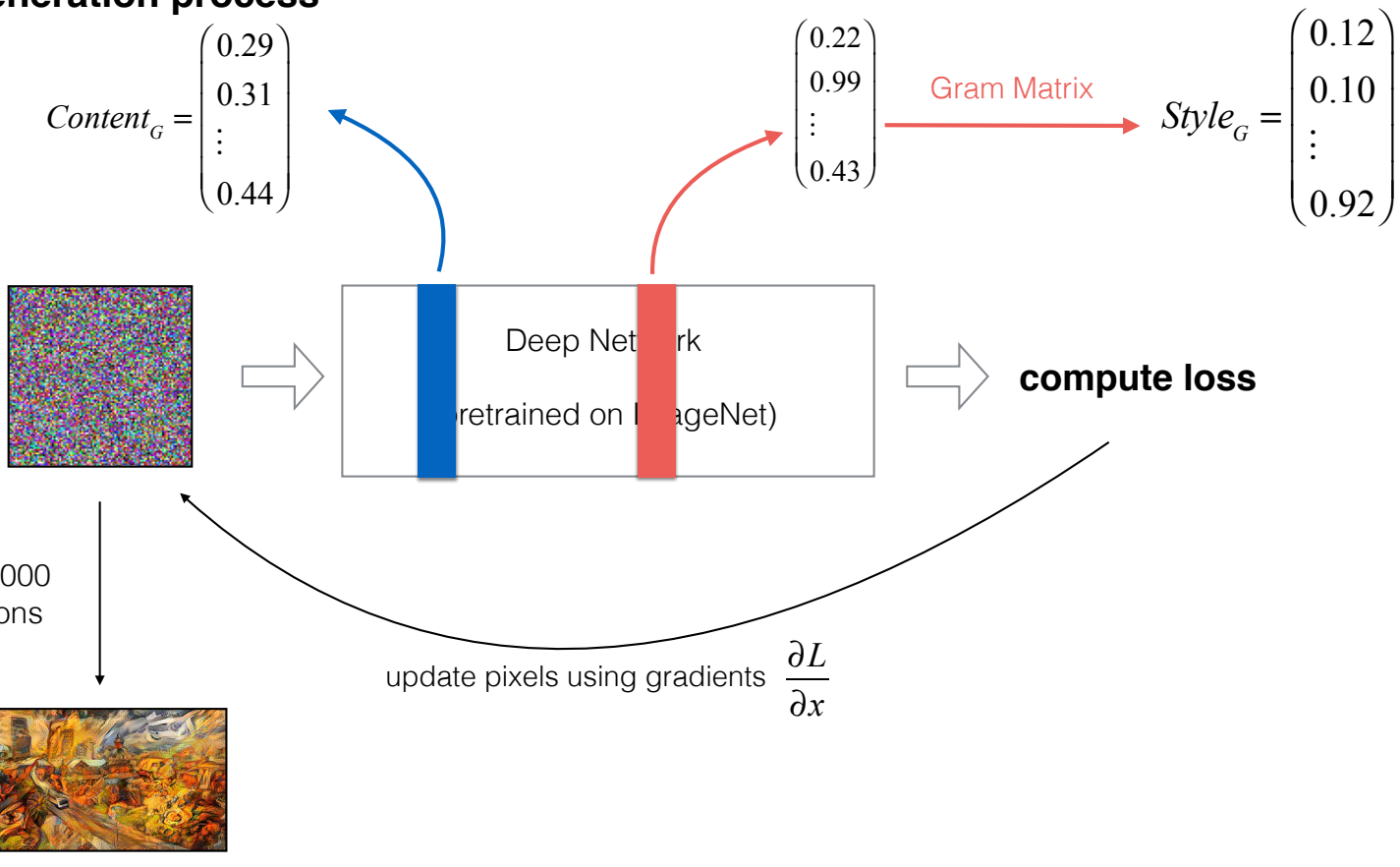
classification

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

Some references

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

Books

- Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville (https://www.deeplearningbook.org/)
- Dive into Deep Learning https://d2l.ai/