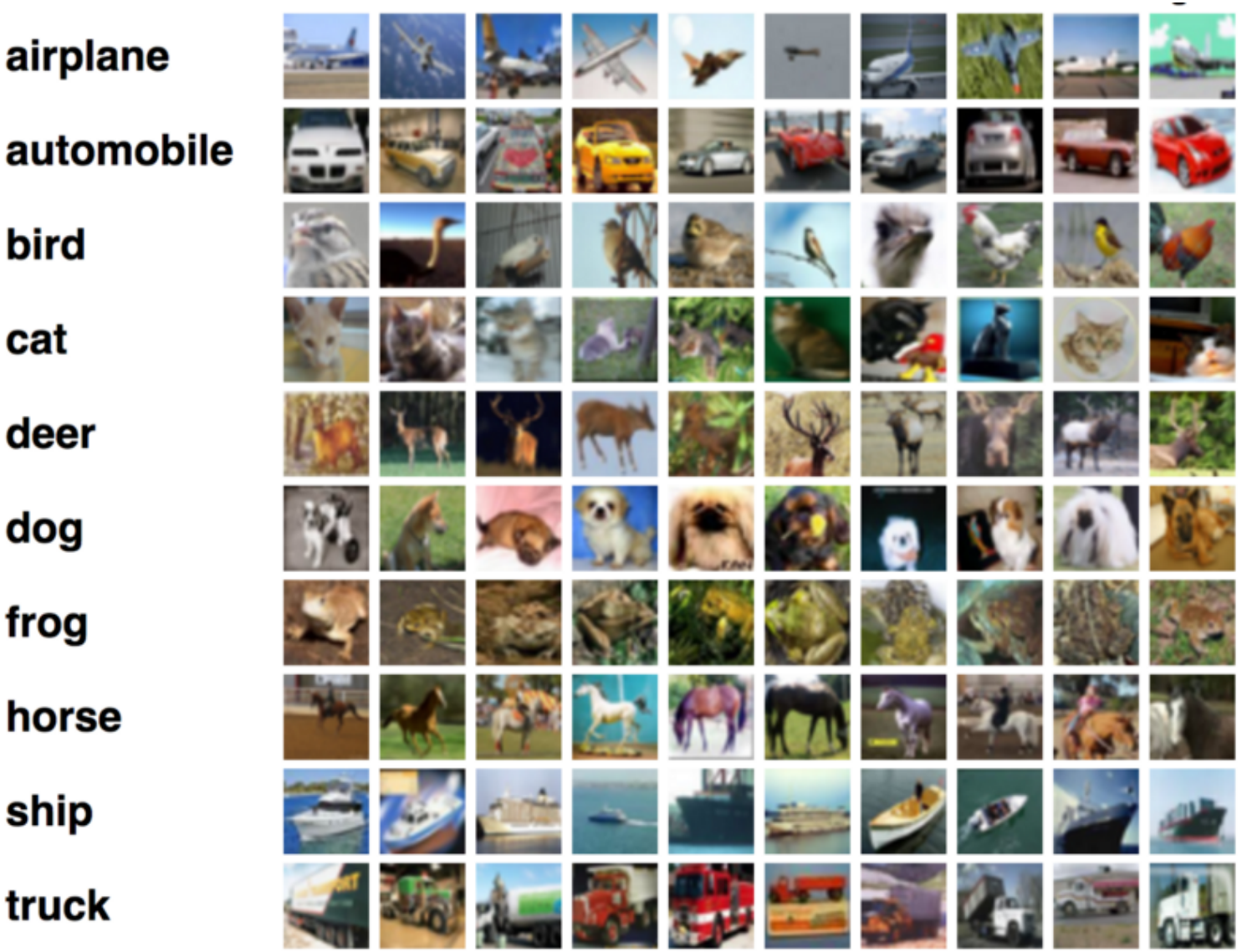
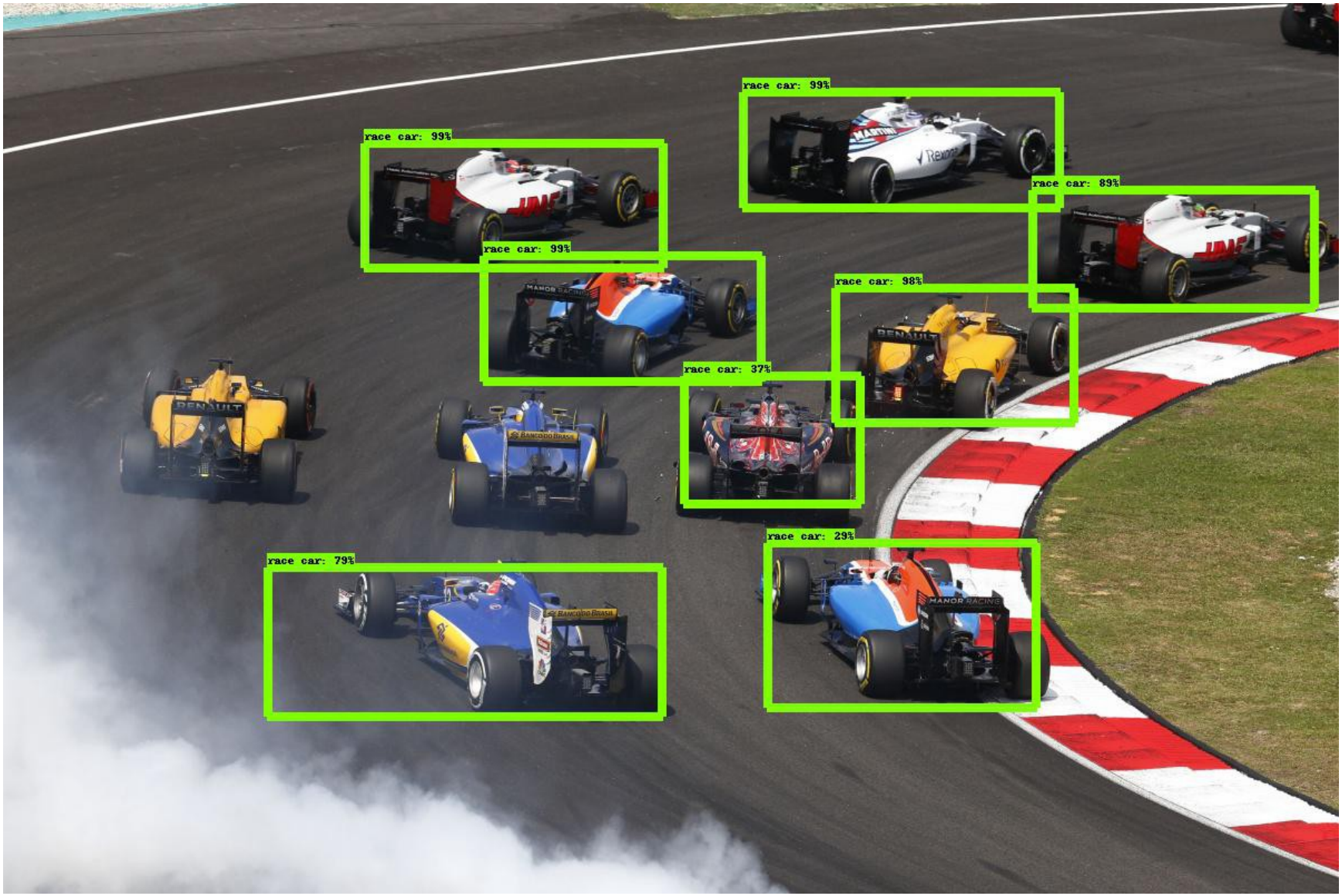
DL is being used to address multiple tasks in CV

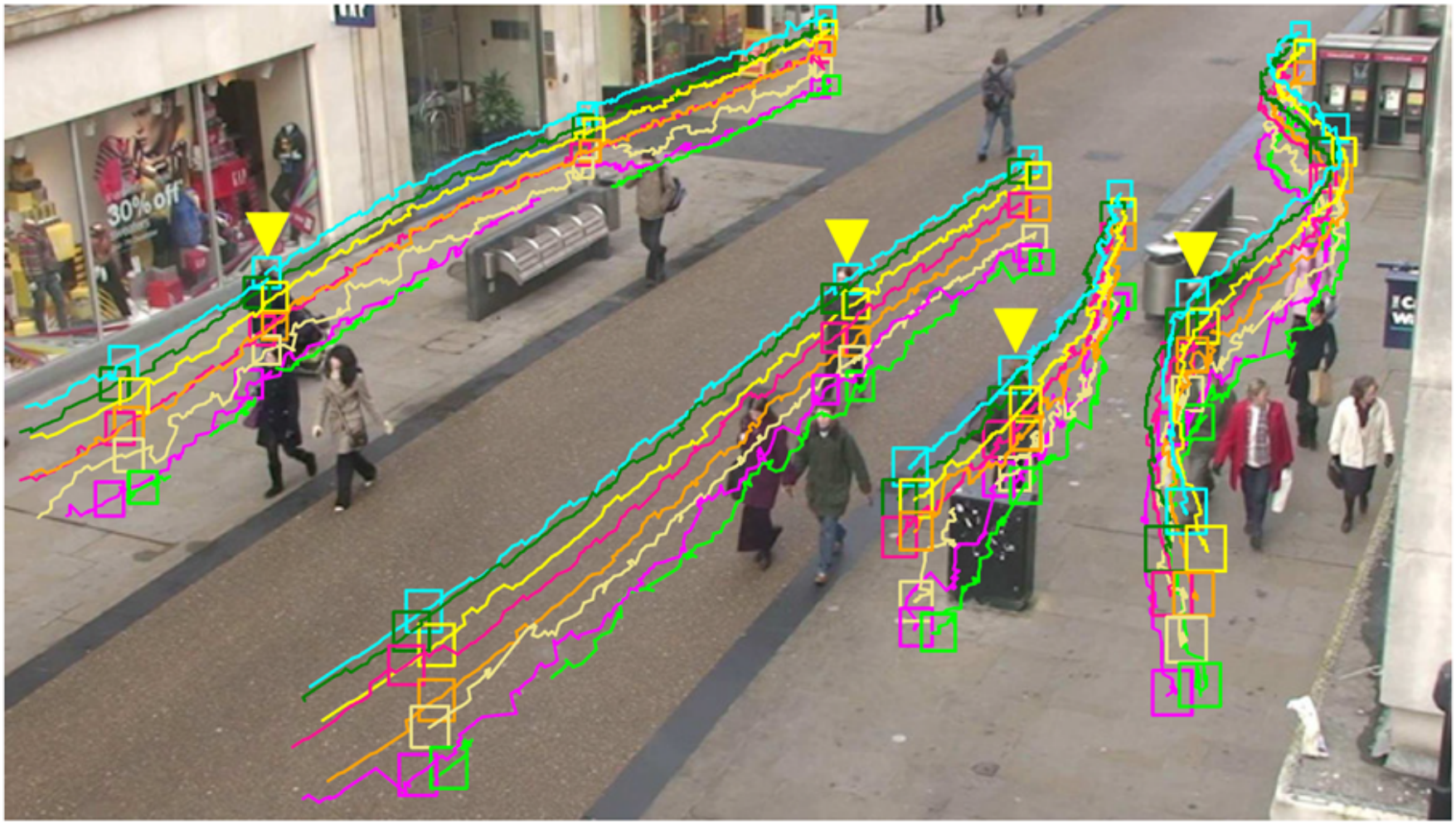
Image classification



Object detection



Object tracking in video



Semantic segmentation

Scene description

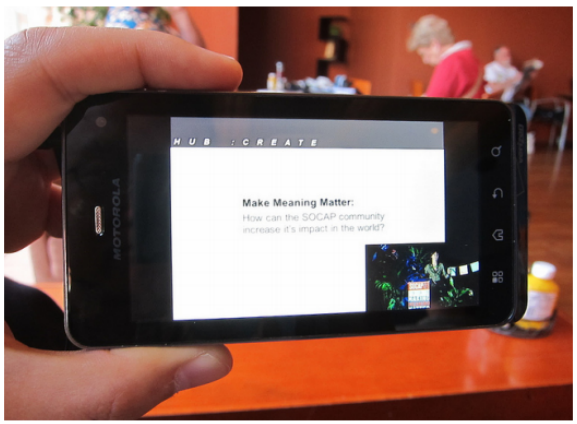
A female tennis player in action on the court.

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

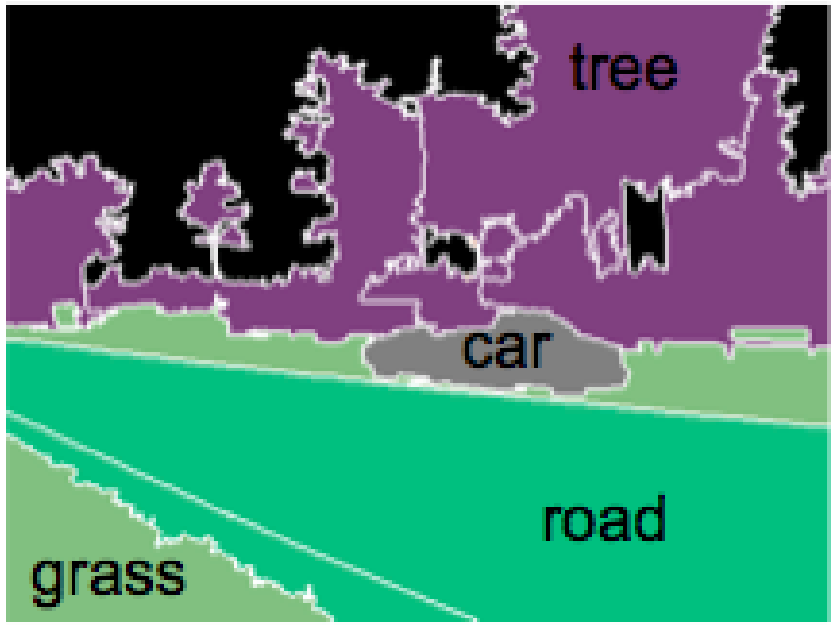

tree
car
road
grass

Image segmentation

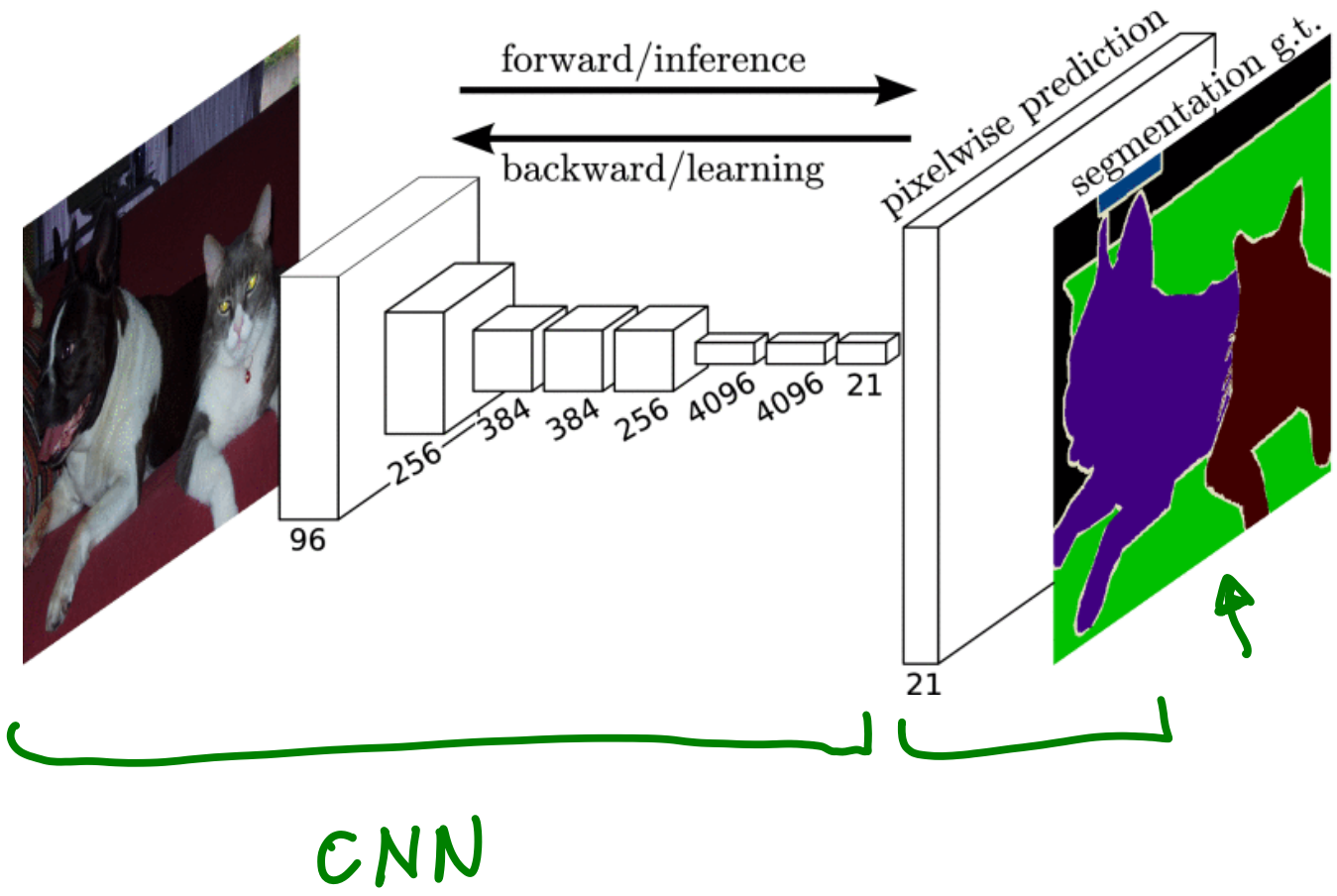
Goal: Delineate the contour of objects/components in the image (attribute a unique label to the *pixels* of each object)

DL architectures consist of a *encoder-decoder* type

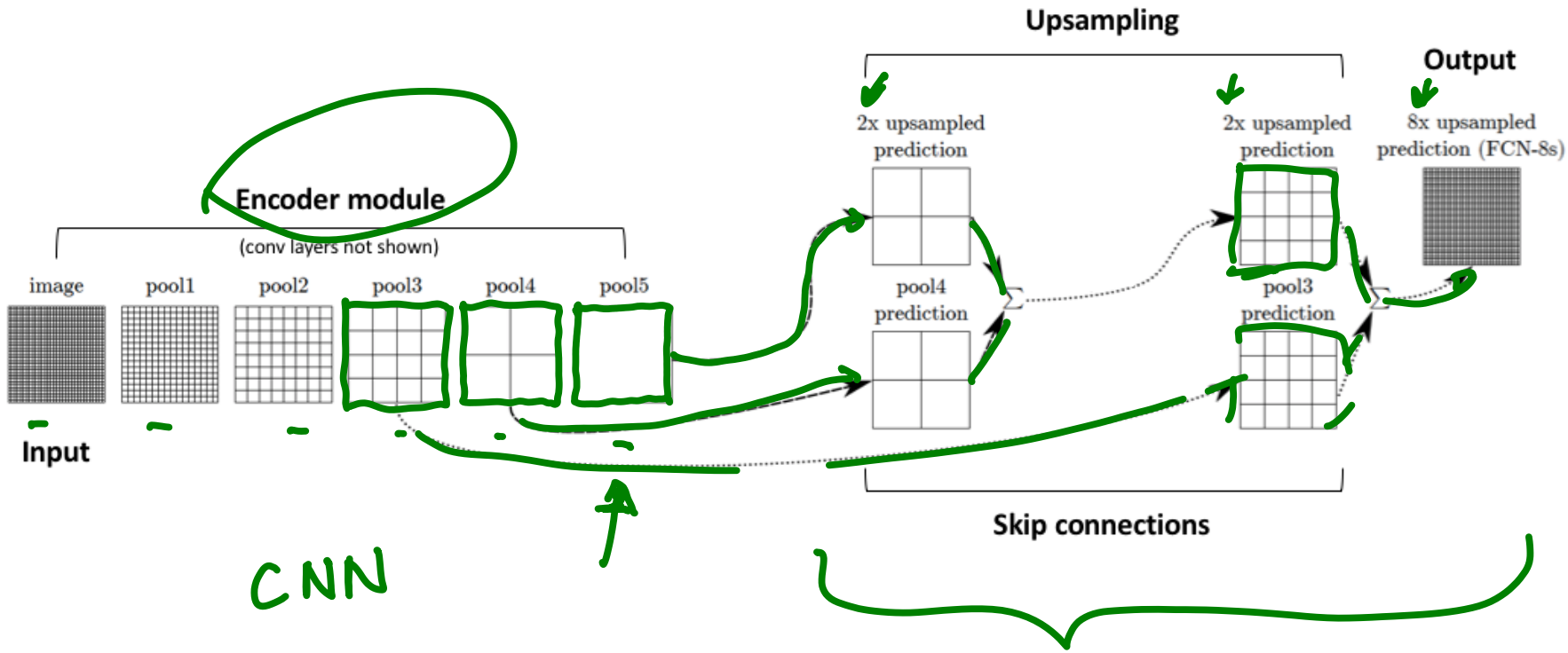
The *encoder* part usually consists os the convolutional layers of a CNN
It shrinks image size

The *decoder* part restores the original image size by means of upsampling

FCN – Fully convolutional network (idea)



FCN – Fully convolutional network (architecture)



U-Net

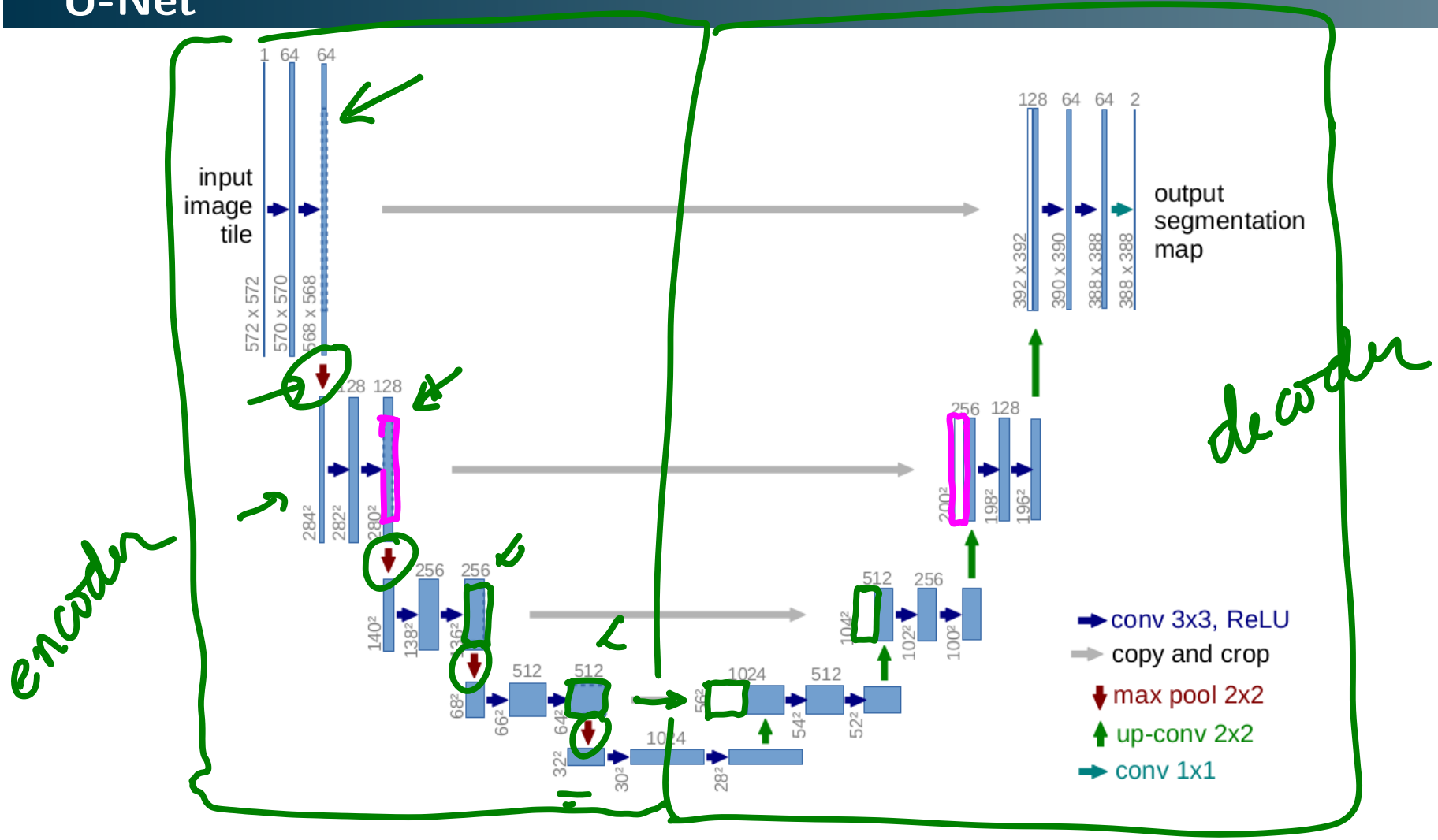
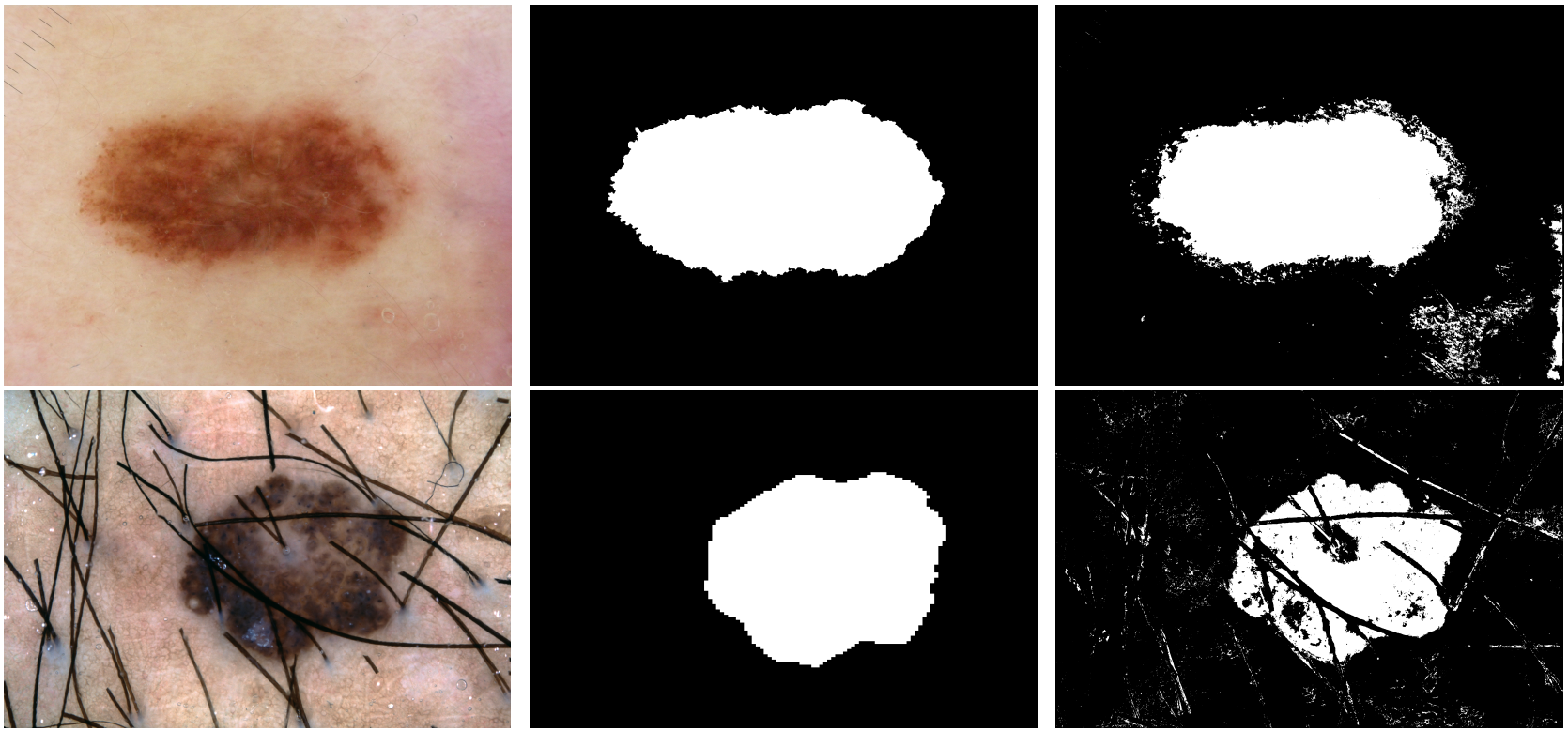


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows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

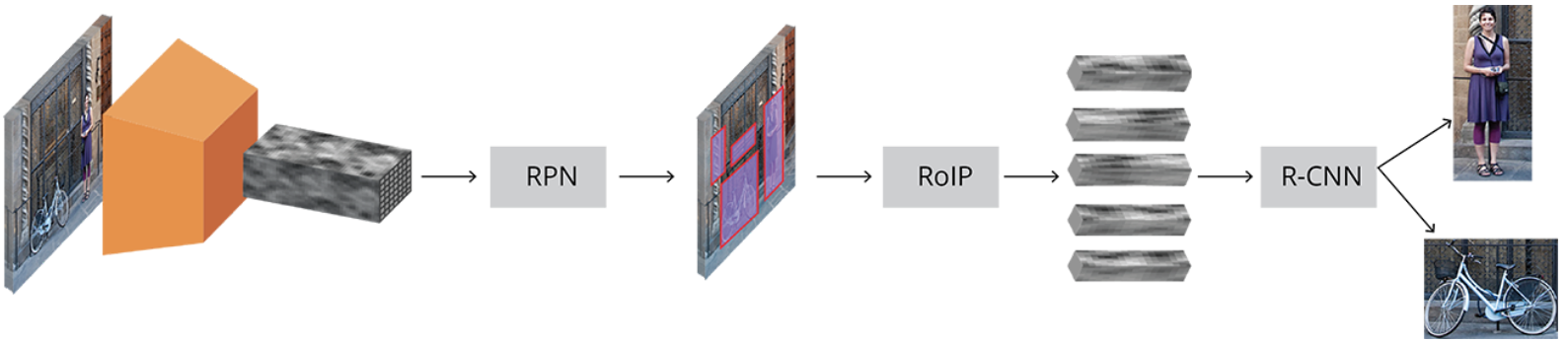
Object recognition in imagens

Training data consists of images plus a **list of bounding box coordinates** of the target objects and their **class labels**.

Cost function: a combination of classification plus regression costs (regression regarding the coordinates of the *boxes*)

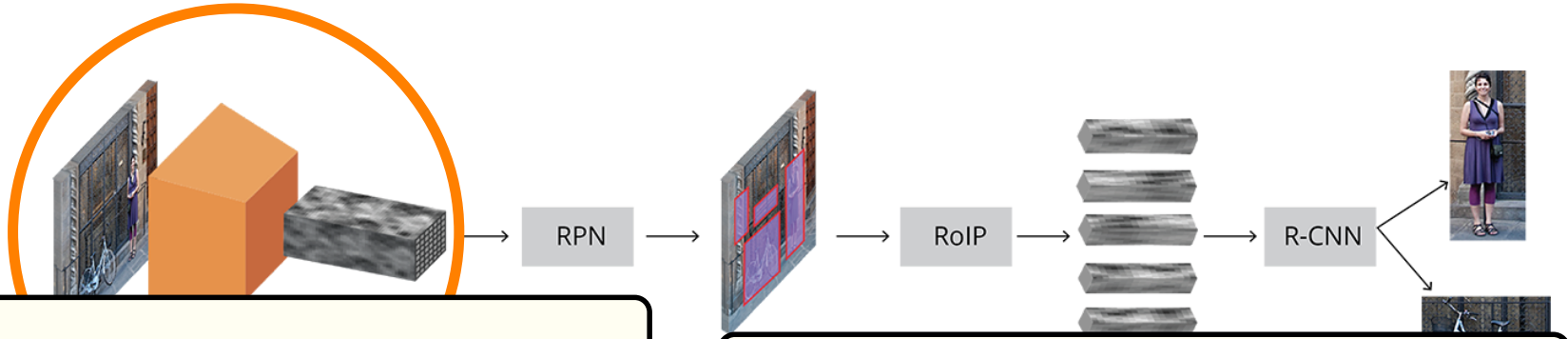
There are **two-step** architectures (eg., Faster R-CNN) and **single shot** archietctures (SSD ou YOLO)

Faster R-CNN architecture

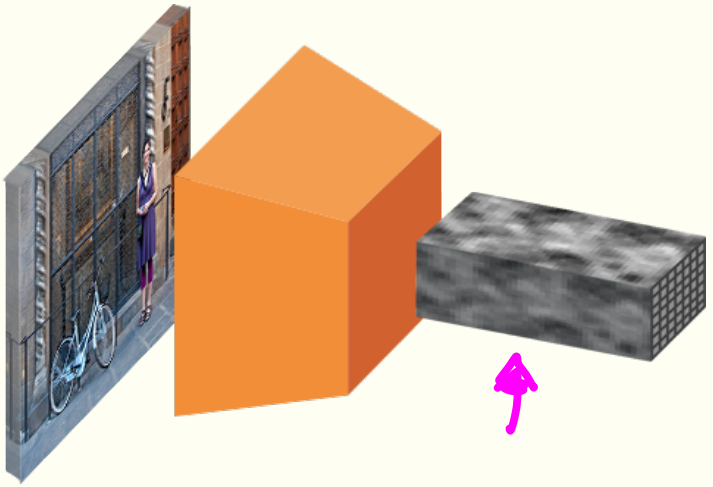


https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

Faster R-CNN architecture



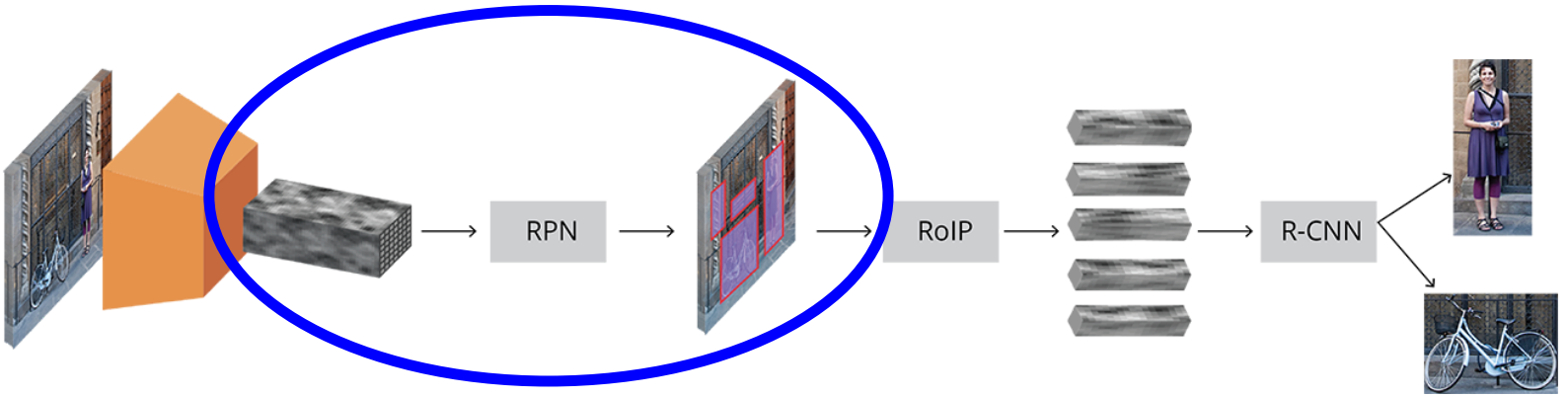
Feature extraction



- 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

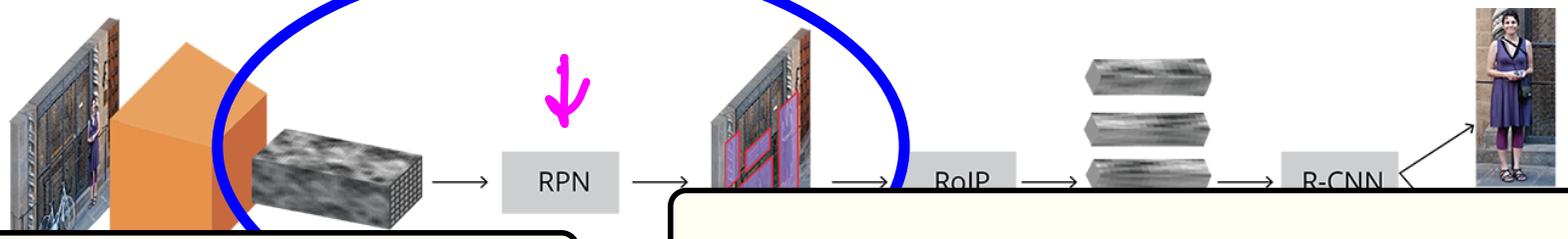


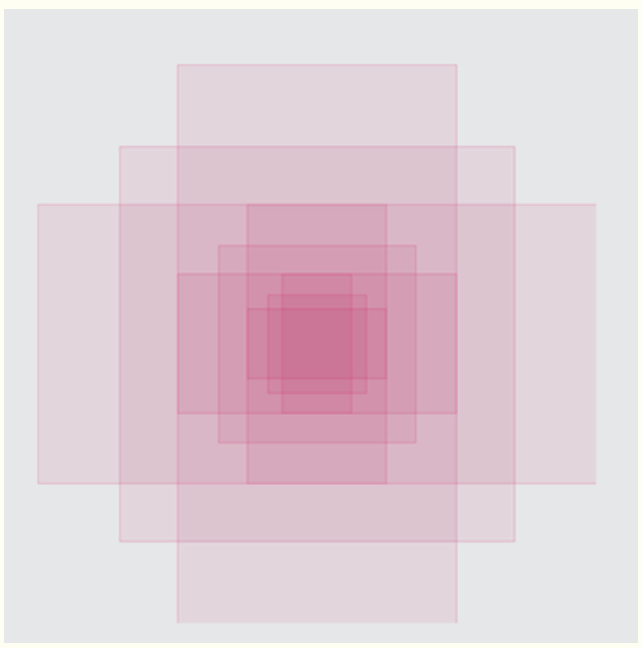
Region proposal network

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

Faster R-CNN architecture

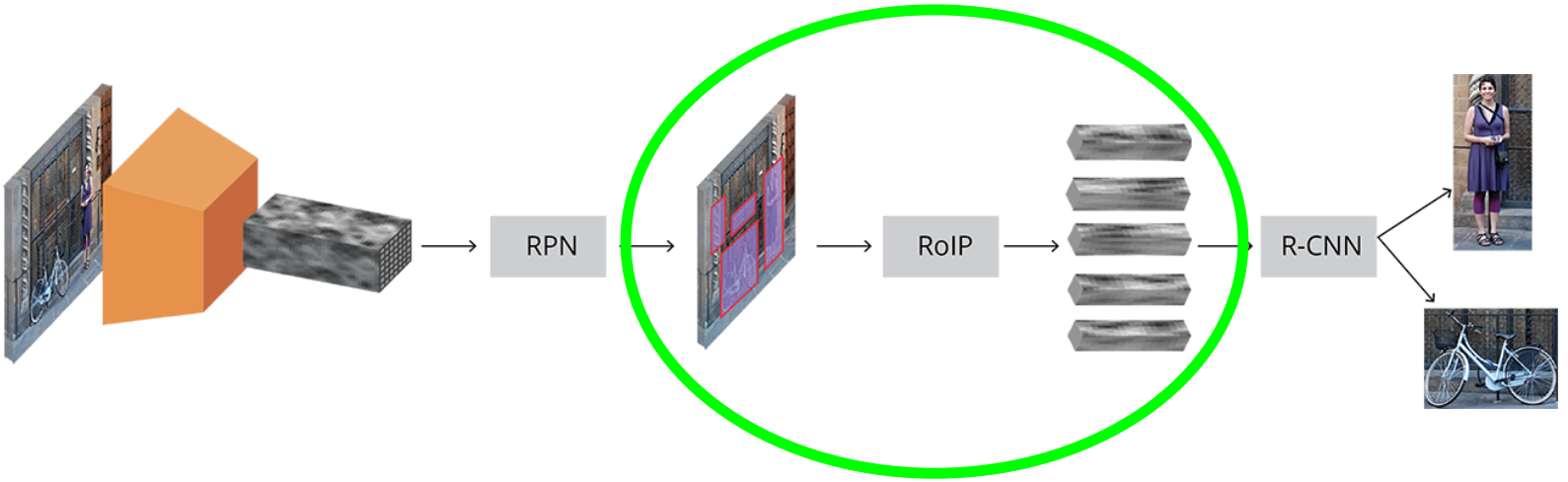
region proposal network.





- 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

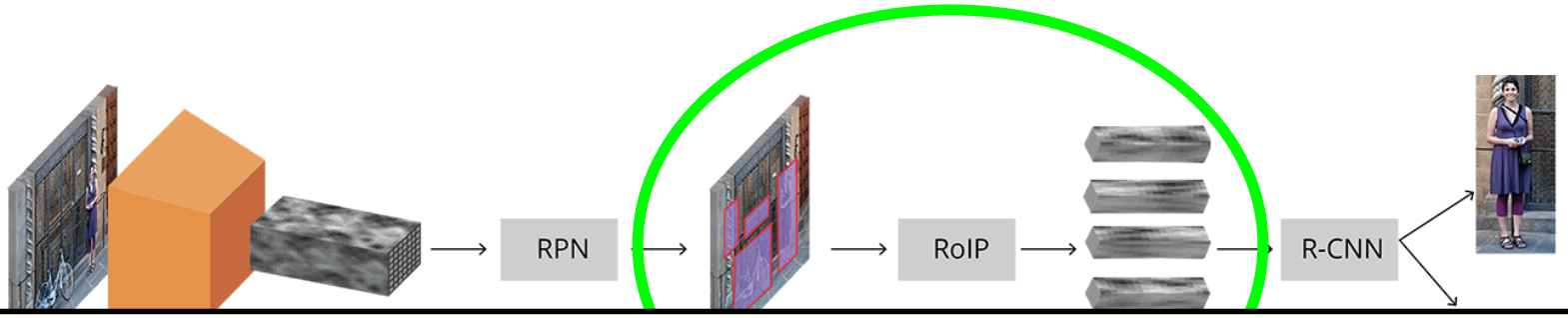
Faster R-CNN architecture



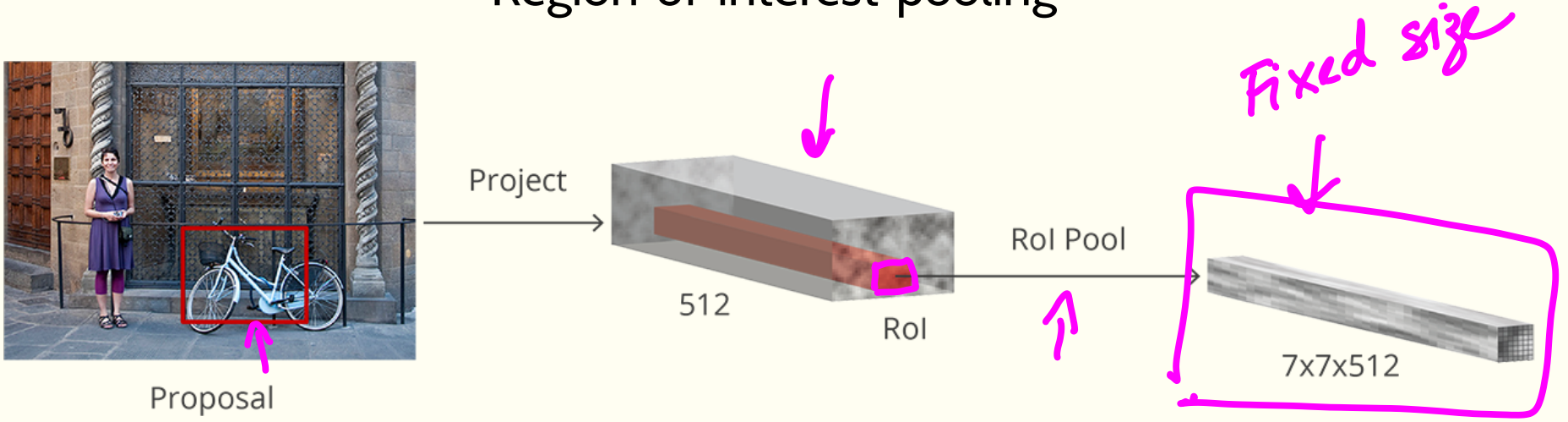
Region of interest pooling

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

Faster R-CNN architecture

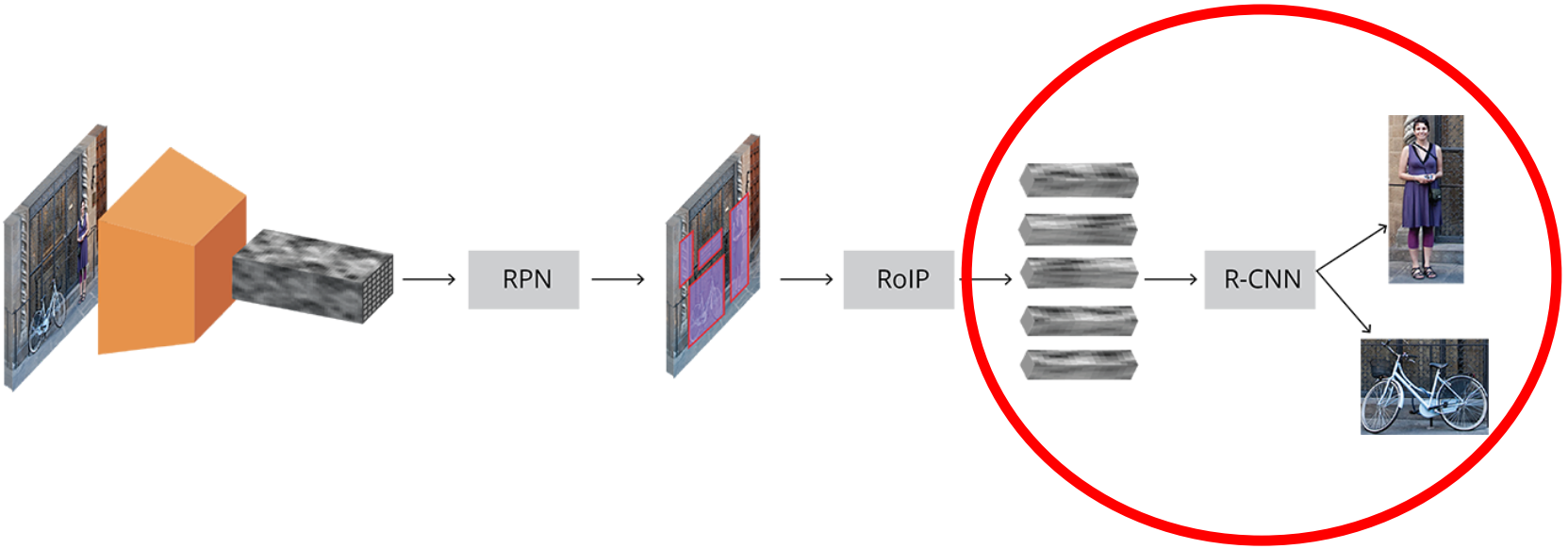


Region of interest pooling



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

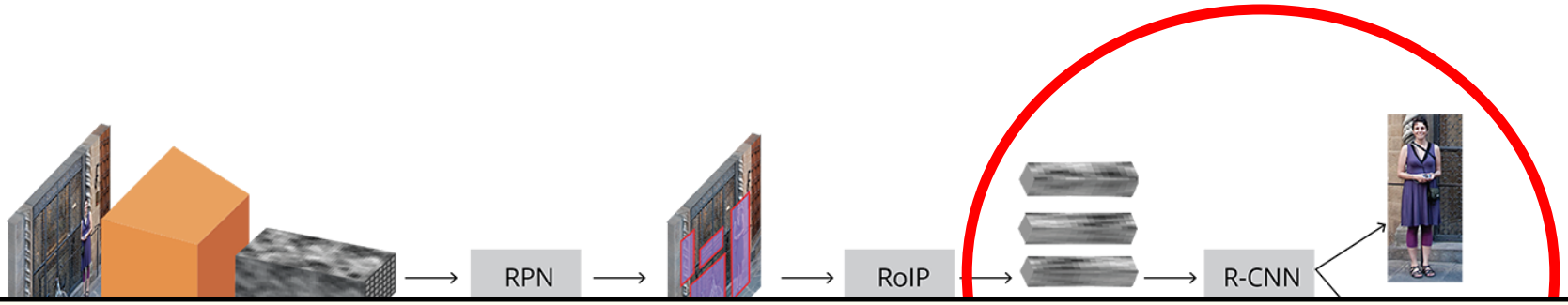
Faster R-CNN architecture



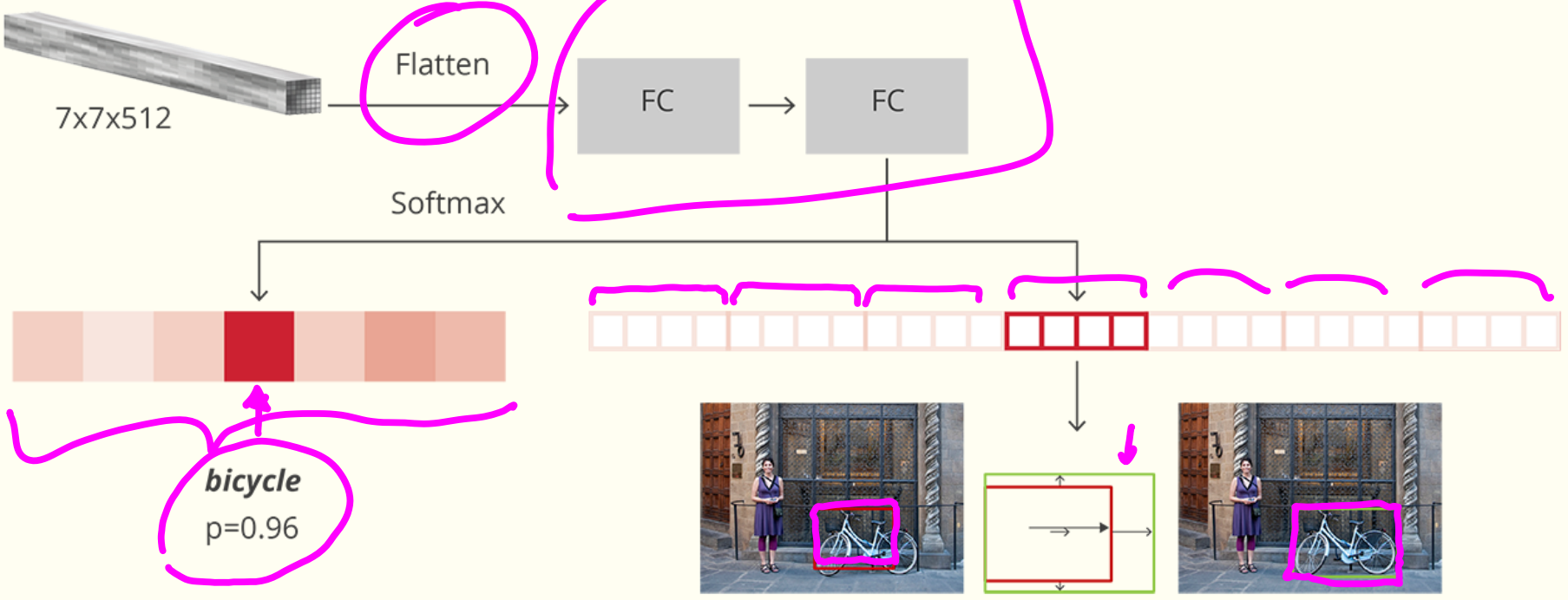
Region classification

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

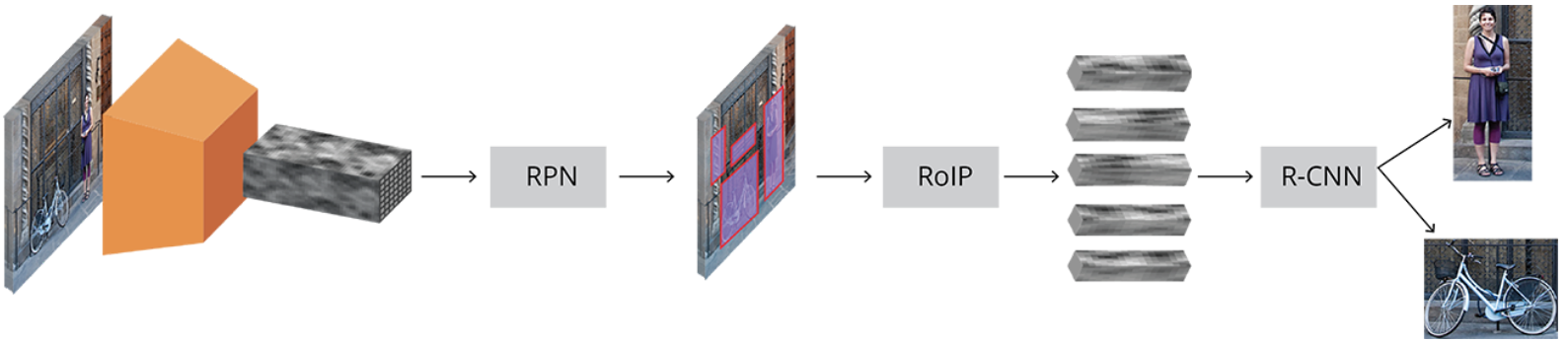
Faster R-CNN architecture



Region classification



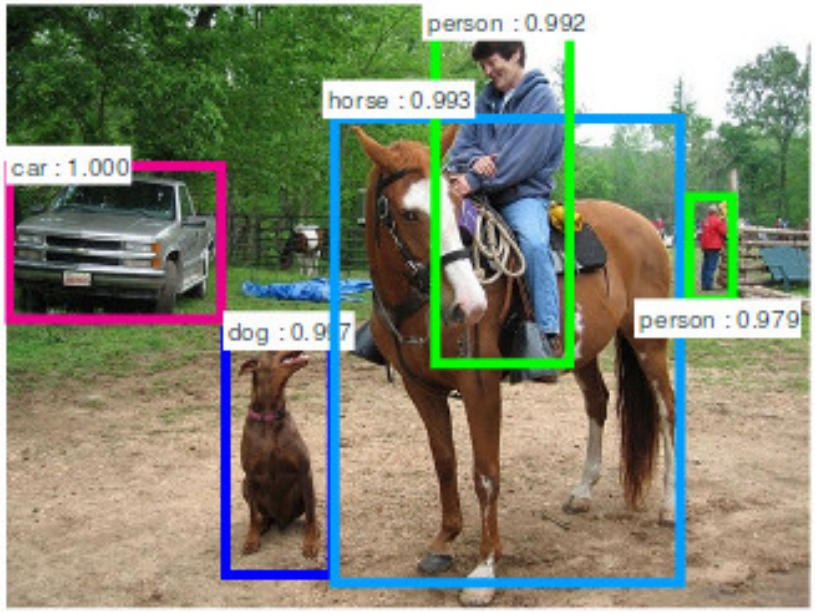
Faster R-CNN architecture

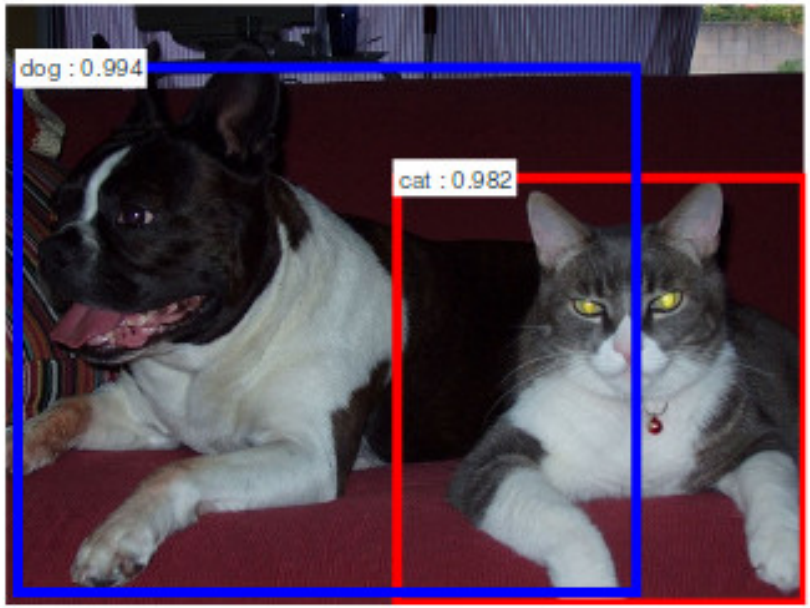


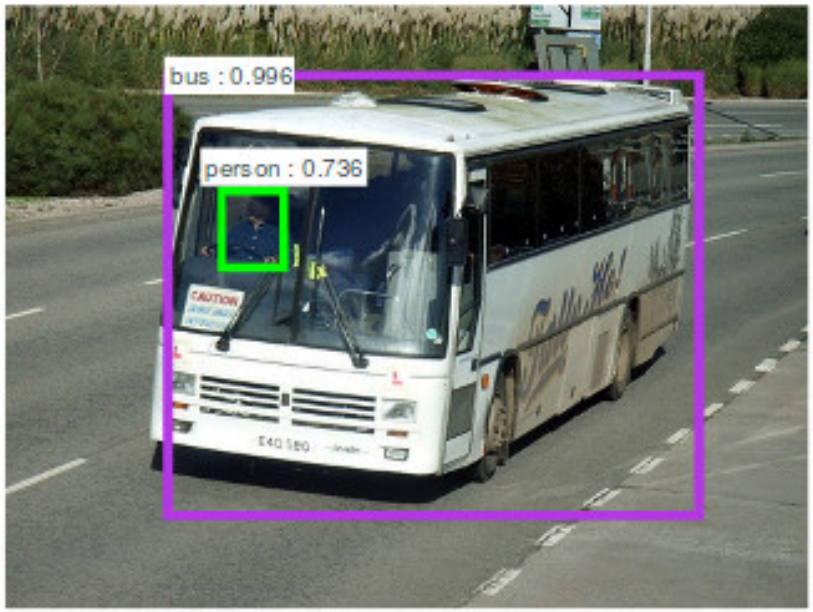
Input is an image

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/

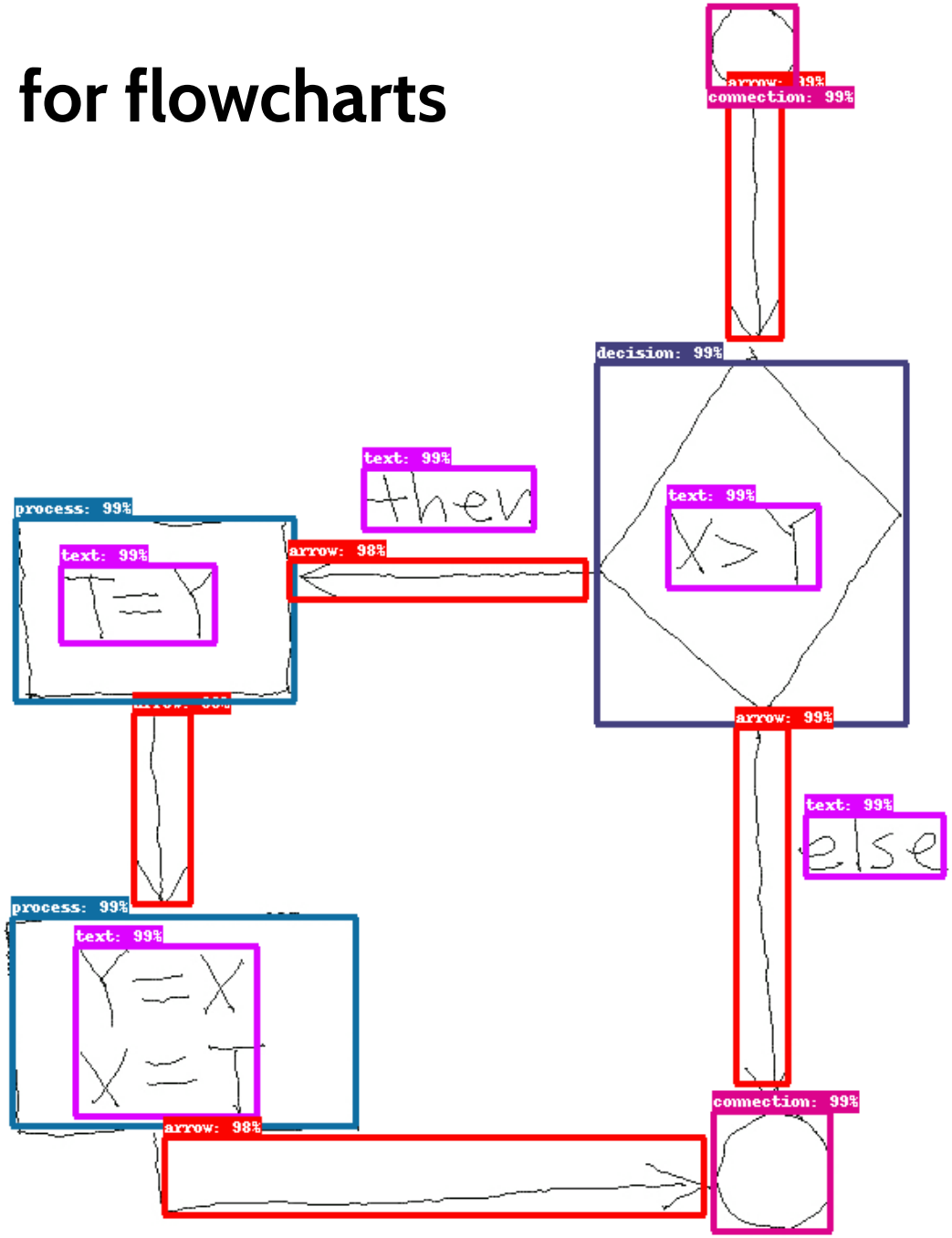

person : 0.992
horse : 0.993
car : 1.000
person : 0.979


dog : 0.994
cat : 0.982

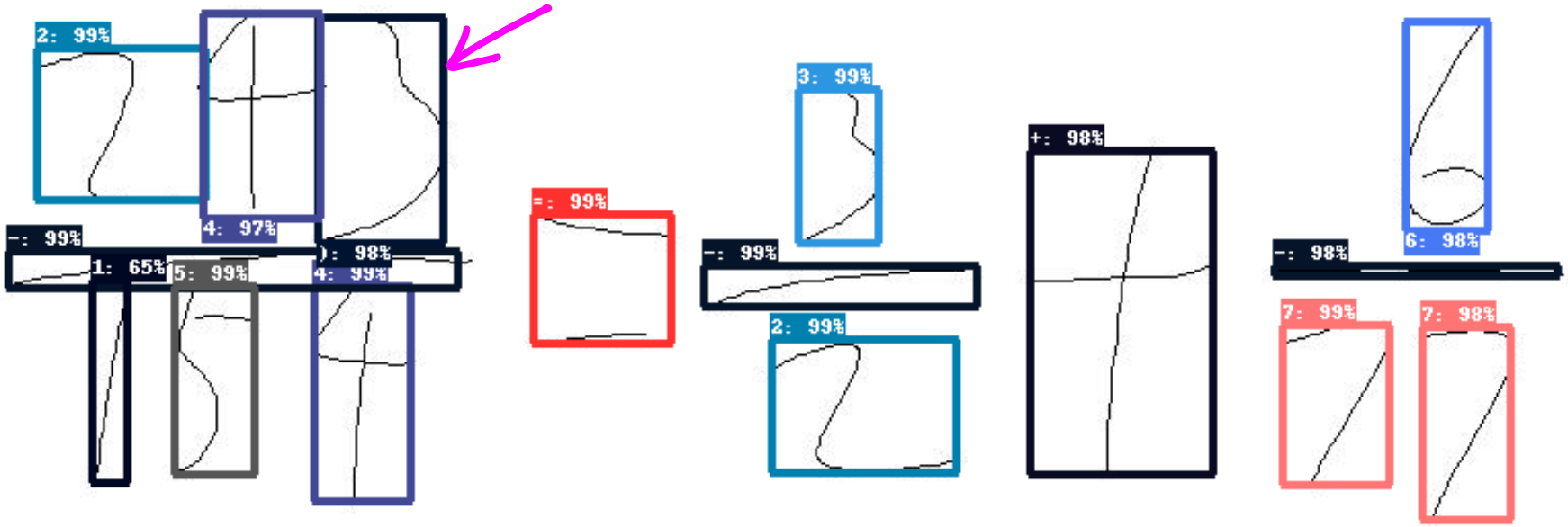

bus : 0.996
person : 0.736


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

Results for flowcharts



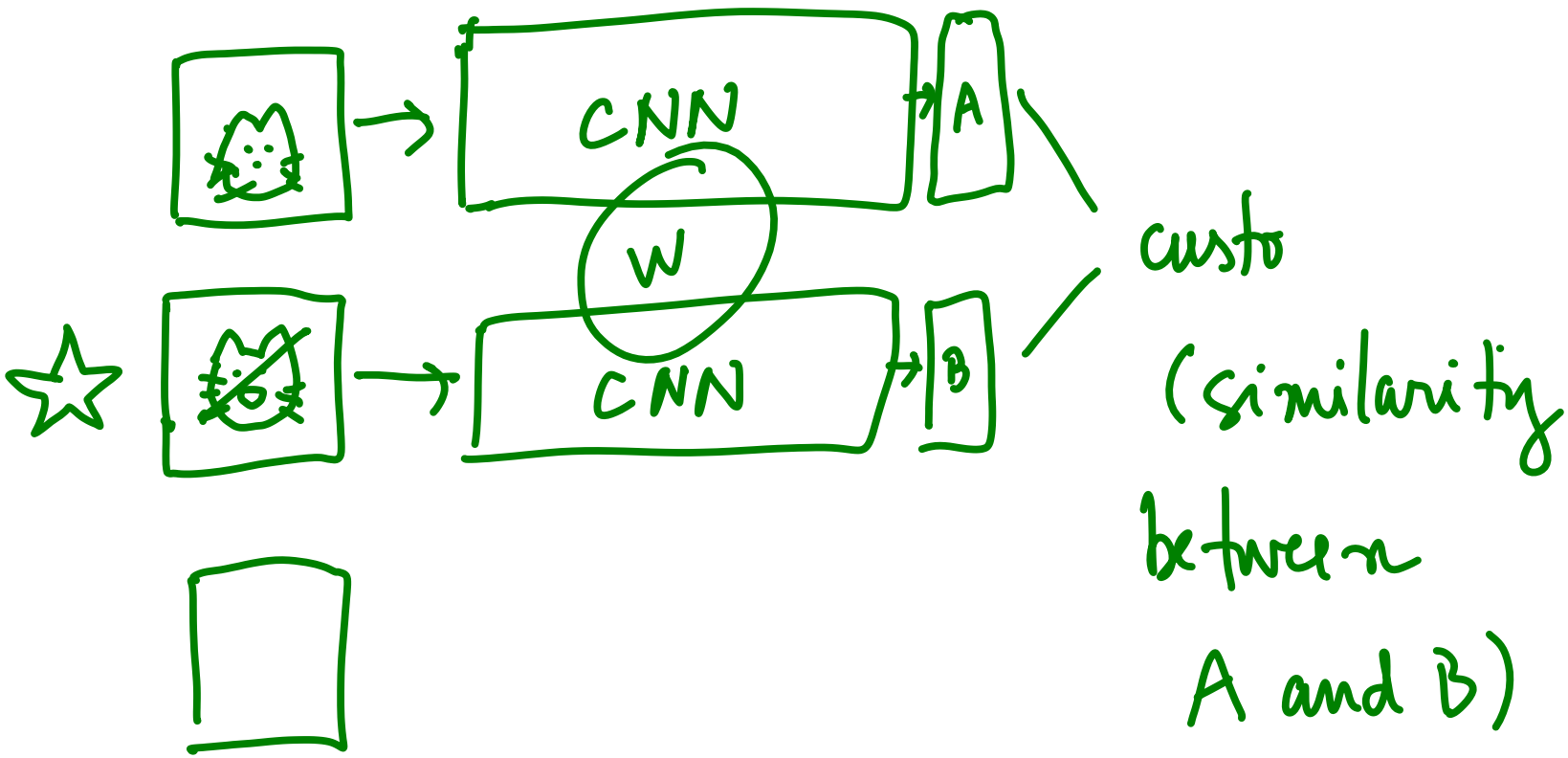
Results for math expressions



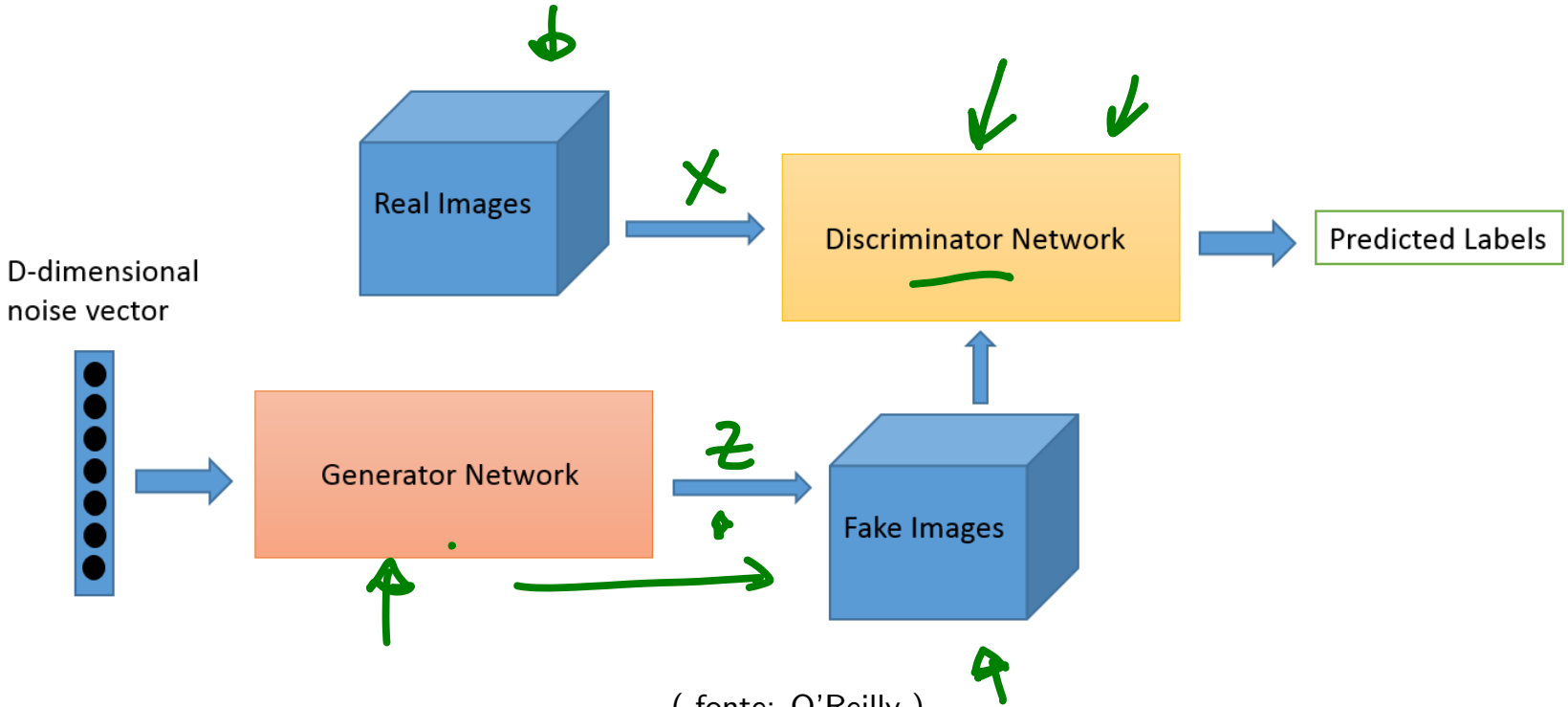
Other DL network models

- **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/

Siamese



GAN schema



(fonte: O'Reilly)

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

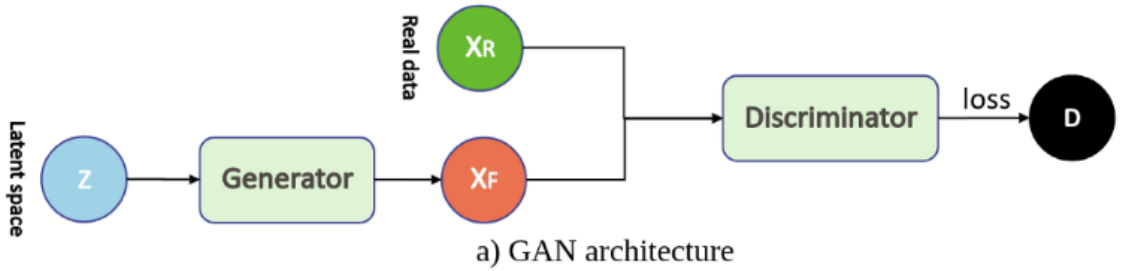
$$\min_G \max_D V(D,G) = E_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + E_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))]$$

D (discriminator) tries to maximize $V(D,G)$ → 0.5

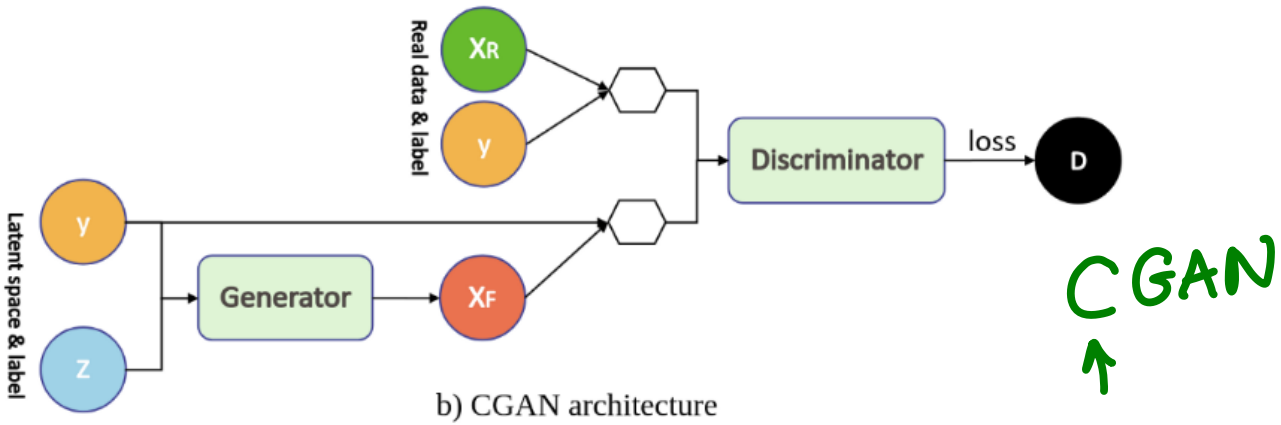
- In the best of the cases, when D is 100% correct, $\max_D V(D,G) = 0$ =

G (generator) tries to minimize $V(D,G)$

- G tries to fool D
- At the extreme, when G starts to generate real-looking images, the best bet of D is a random guess



a) GAN architecture



b) CGAN architecture

CGAN

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)

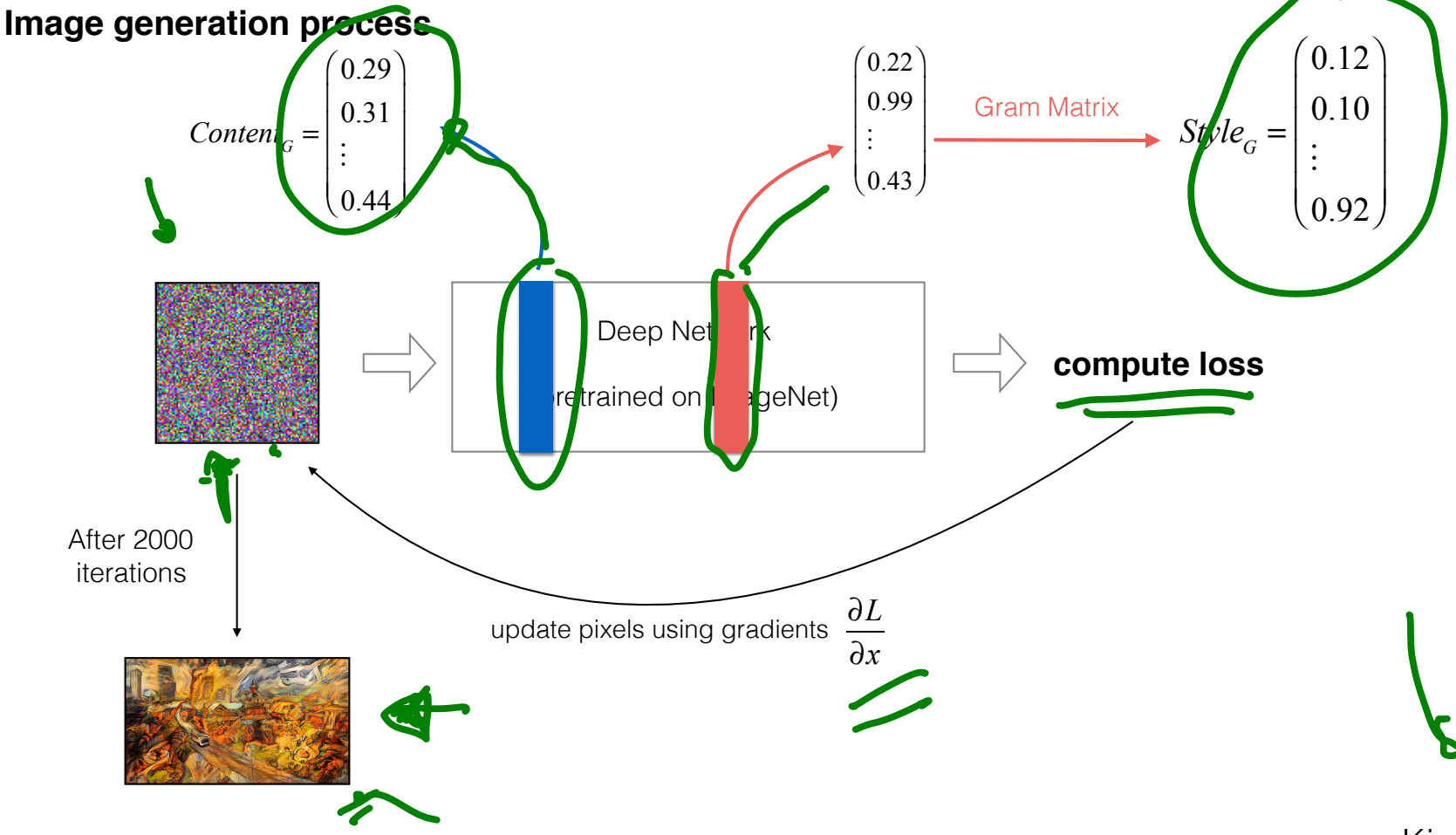
classification

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



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

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

Kian Katanforoosh

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

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/