Nizhny Novgorod State University

Institute of Information Technologies, Mathematics and Mechanics

Department of Computer software and supercomputer technologies

Educational course
«Modern methods and technologies
of deep learning in computer vision»

Object detection in images using deep neural networks

Supported by Intel

Getmanskaya Alexandra, Kustikova Valentina

Content

- Goals
- Object detection problem statement
- Public datasets
- Quality metrics
- Deep models for object detection
- Comparison of deep models for object detection
- Conclusion

Goals

- ***The goal*** is to study deep neural networks for solving object detection problem

OBJECT DETECTION PROBLEM STATEMENT

Problem statement (1)

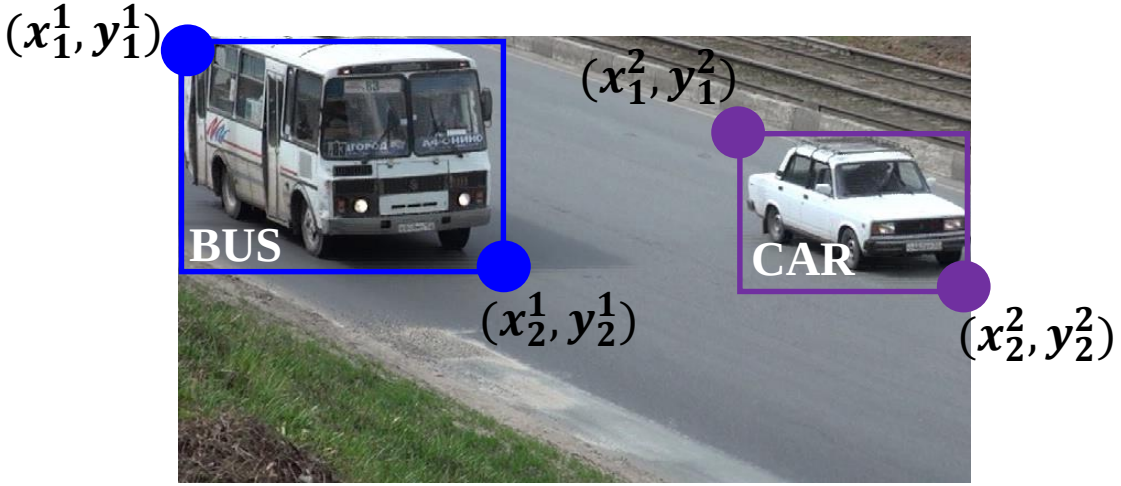
- The problem of object detection is to determine the object locations, object location is usually represented by a bounding box

Problem statement (2)

- The goal of the object detection problem is to match image with the set of object locations B:

$$\varphi: I \to B, \qquad B = \left\{ b_k, k = \overline{0, |B| - 1} \right\},$$

where $b_k = \left(\left(x_1^k, y_1^k \right), \left(x_2^k, y_2^k \right) [, s^k, c^k] \right)$, $s^k \in \mathbb{R}$ is a confidence, c^k is an object class (“TABLE”, “PEDESTRIAN”, “CAR”, “BUS”, etc.)



PUBLIC DATASETS

Datasets (1)

Dataset	Train dataset		Validation dataset		Test dataset		Number of classes
	Images	Objects	Images	Objects	Images	Objects	
Real-life object detection							
PASCAL VOC 2007 [http://host.robots.ox.ac.uk/pascal/VOC/voc2007]	2 501	6 301	2 510	6 307	4 952	12 032	20
PASCAL VOC 2012 [http://host.robots.ox.ac.uk/pascal/VOC/voc2012]	5 717	13 609	5 823	13 841	N/A	N/A	20
MS COCO [http://cocodataset.org]	165 482	N/A	81 208	N/A	81 434	N/A	91

Datasets (2)

Dataset	Train dataset		Validation dataset		Test dataset		Number of classes
	Images	Objects	Images	Objects	Images	Objects	
Real-life object detection							
Open Images Dataset [https://storage.googleapis.com/openimages/web/index.html]	~1,7 M	~1,4 M	~40 K	~204 K	~125 K	~625 K	600
Face detection and recognition							
WIDER FACE [http://shuoyang1213.me/WIDERFACE]	12 881	~157 K	3 220	~39 K	16 102	~196 K	1

Datasets (3)

Dataset	Train dataset		Validation dataset		Test dataset		Number of classes
	Images	Objects	Images	Objects	Images	Objects	
Face detection and recognition							
LFW [http://vis-www.cs.umass.edu/lfw]	11 910	~5 K	0	0	1 323	~700	1
AFLW [https://www.tugraz.at/institute/icg/research/team-bischof/lrs/downloads/aflw]	25 K	25 K * 21	–	–	–	–	21
IMDB-WIKI [https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki]	523 051	~500 K	–	–	–	–	1

Datasets (4)

Dataset	Train dataset		Validation dataset		Test dataset		Number of classes
	Images	Objects	Images	Objects	Images	Objects	
Pedestrian detection							
Caltech [http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians]	~57 videos by 1 min	~ 175 K	–	–	~ 47 videos by 1 min	~ 175 K	1
Wider Person [http://www.cbsr.ia.ac.cn/users/sfzhang/WiderPerson]	8 000	~ 240 K	1 000	~ 30 K	4 382	~ 130 K	1

Open Images Dataset

- 15 851 536 objects belonging to 600 categories
- If more than 5 instances of objects of the same class strongly overlap, they are enclosed in one rectangle with the label “group of objects”
- All bounding boxes are labeled manually

* Open Images Dataset [https://storage.googleapis.com/openimages/web/index.html].

** Kuznetsova A., Rom H., Alldrin N., Uijlings J., Krasin I., Pont-Tuset J., Kamali S., Popov S., Malloci M., Kolesnikov A., Duerig T., Ferrari V. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. – 2020. – [https://arxiv.org/pdf/1811.00982.pdf].

WIDER FACE

- WIDER FACE is a benchmark for comparing quality of face detection methods
- 32 203 images containing 393 703 faces with a high degree of variability in scale, position and overlap

* WIDER FACE [http://shuoyang1213.me/WIDERFACE].

Caltech Pedestrian Dataset

- 10 hours of video with a resolution of 640x480 and frequency 30 Hz, video is obtained from a DVR on a moving car
- ~250 000 annotated frames (in 137 fragments about a minute long) with 350 000 bounding boxes and 2 300 unique pedestrians

* Caltech Pedestrian Dataset [http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians].

QUALITY METRICS

Quality metrics

- True positive rate
- False detection rate
- Average false positives per frame
- Average precision

True positive rate

- ***True positive rate*** is a ratio of the correctly detected objects (true positives) TP to the total number of objects $TP + FN$

$$TPR = \frac{TP}{TP + FN}$$

- It is assumed that the object is detected correctly if the intersection over union for the detected (detection, d) and labeled (groundtruth, g) bounding boxes $IoU = \frac{S_{d \cap g}}{S_{d \cup g}}$ exceeds a certain threshold τ
- The threshold τ is in the range from 0.5 to 0.7

		Prediction	
		True	**False**
Groundtruth	**True**	TP	FN
	False	FP	TN

False detection rate

- ***False detection rate*** is a ratio of the false positives to the total number of detections

$$FDR = \frac{FP}{TP + FP}$$

- An object is detected correctly under the same conditions as for the previous metric
- The detected bounding box is assumed as a false positive if there is no a pair from the groundtruth

		Prediction	
		True	**False**
Groundtruth	**True**	TP	FN
	False	FP	TN

Average false positives per frame

- ***Average false positives per frame*** is a ratio of the false positives FP to the total number of images N

$$FPperFrame = \frac{FP}{N}$$

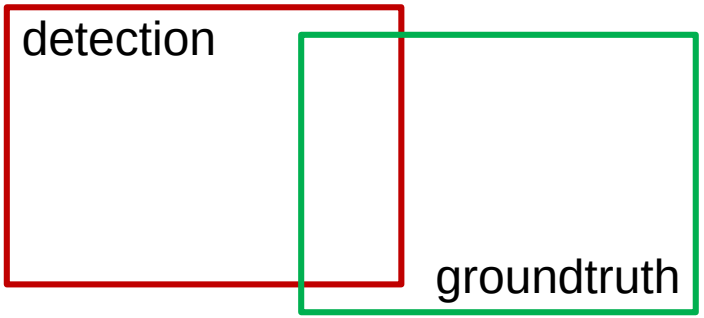
- An object is detected correctly under the same conditions as for the previous metric

- The metric is important processing an image stream (e.g. video)

		Prediction	
		True	**False**
Groundtruth	**True**	TP	FN
	False	FP	TN

Average precision (1)

- Notation:
  - $IoU = \frac{S_{d \cap g}}{S_{d \cup g}}$ is a ratio of overlapping the detected (detection) and labeled (groundtruth) bounding boxes (Intersection over Union), $IoU \in [0; 1]$
  - TP is a number of detected objects for which intersection over union is not less than a certain threshold τ (we think of the object is detected correctly, it is a true positive)
  - FP is a number of detected objects for which intersection over union is less than τ (the object was detected incorrectly), or the object was detected more than once (false positives)
  - FN is a number of undetected objects (false negatives)



Average precision (2)

- The threshold value τ usually is chosen equal to 0.5

- ***Precision*** is a ratio of true positives by the overall number of detections

$$Precision = p = \frac{TP}{TP + FP}$$

- ***Recall*** is a ratio of true positives by the overall number of objects

$$Recall = r = \frac{TP}{TP + FN}$$

Average precision (3)

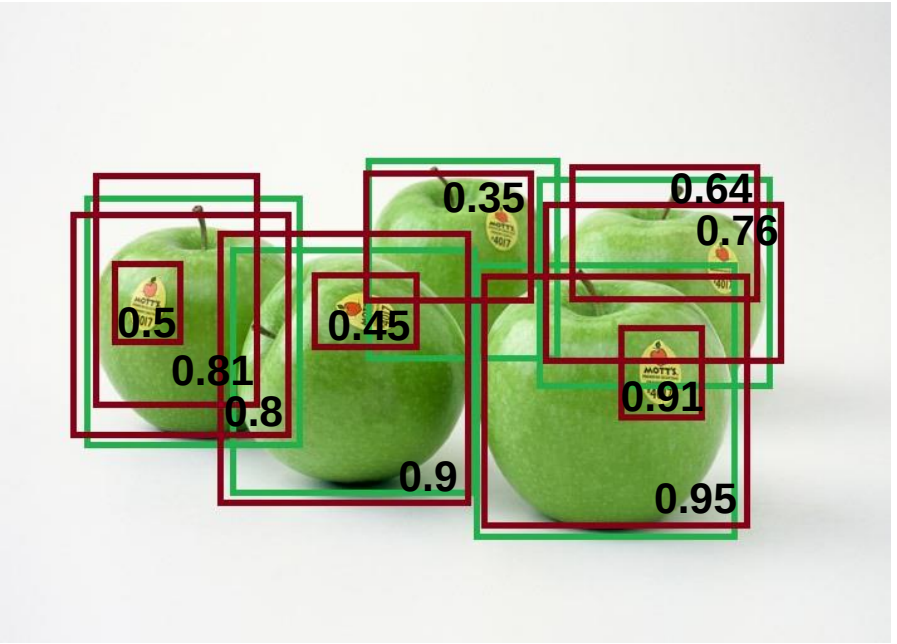
- ***Average precision*** is a mean of precisions

$$AP = \int_0^1 p(r)dr$$

- Calculation algorithm:
  - Detected bounding boxes are sorted in descending order of the confidence
  - For each detected bounding box, a match is searched from the groundtruth according to the condition $IoU \geq \tau$
  - Precision and recall are calculated
  - The graph of precision as a function of recall is constructed
  - The area under the constructed graph is calculated

Average precision (4)

- Example of calculating average precision:
  - The original image is a photo of apples from the ImageNet dataset [http://www.image-net.org]
  - The groundtruth contains bounding boxes for 5 apples (green rectangles)
  - The object detection algorithm detects 10 apples (red rectangles)
  - For definiteness, it is assumed that the confidences are not equal, to further uniquely identify the bounding boxes



Average precision (5)

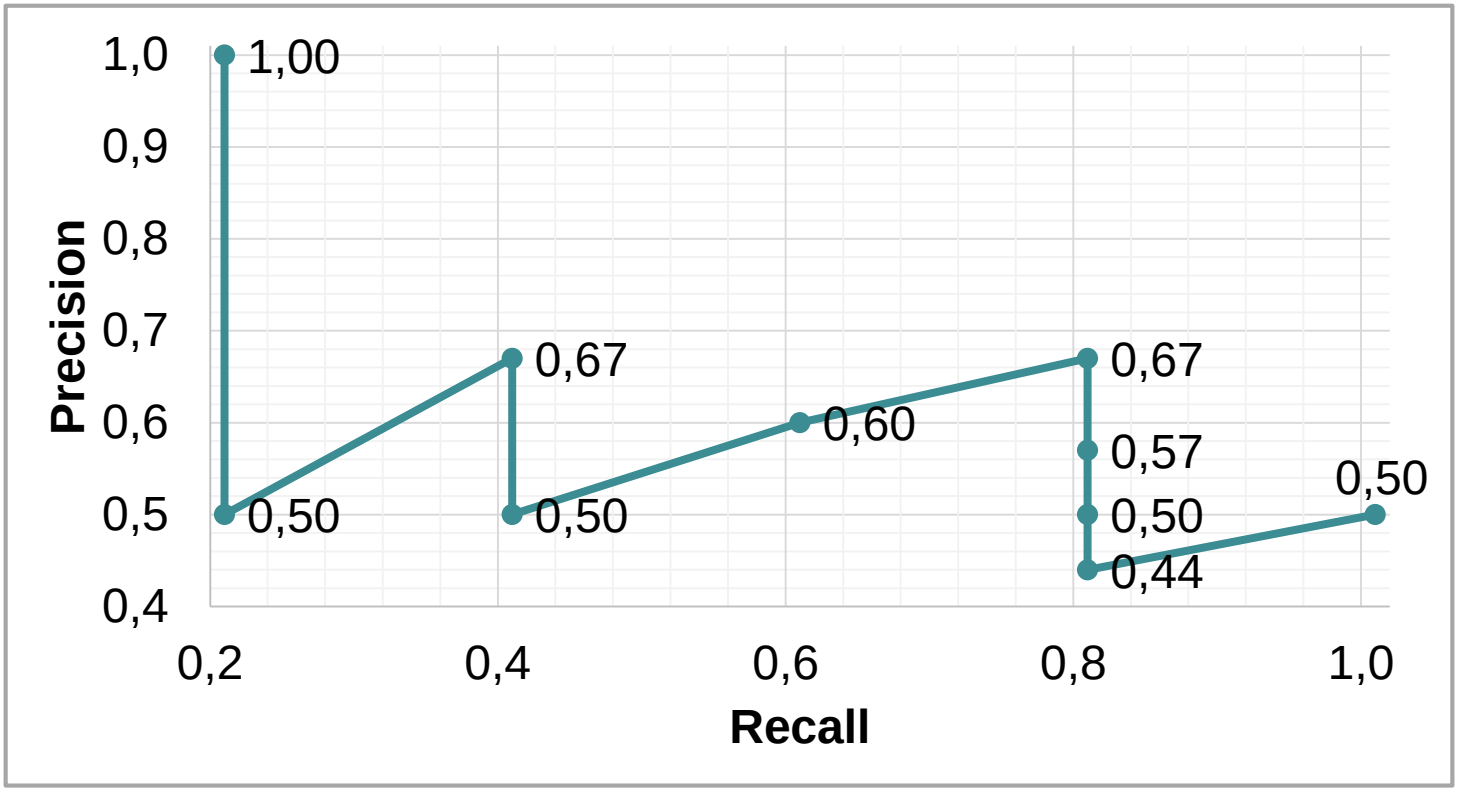
- Example of calculating average precision:
  - Sorting bounding boxes, calculating precision and recall

#	Confidence	Is an object?	Precision	Recall
1	0.95	Yes	1/1 = 1.0	1/5 = 0.2
2	0.91	No	1/2 = 0.5	1/5 = 0.2
3	0.9	Yes	2/3 ≈ 0.67	2/5 = 0.4
4	0.81	No	2/4 = 0.5	2/5 = 0.4
5	0.8	Yes	3/5 = 0.6	3/5 = 0.6
6	0.76	Yes	4/6 = 0.67	4/5 = 0.8
7	0.64	No	4/7 ≈ 0.57	4/5 = 0.8
8	0.5	No	4/8 = 0.5	4/5 = 0.8
9	0.45	No	4/9 ≈ 0.44	4/5 = 0.8
10	0.35	Yes	5/10 = 0.5	5/5 = 1.0

Recall is growing

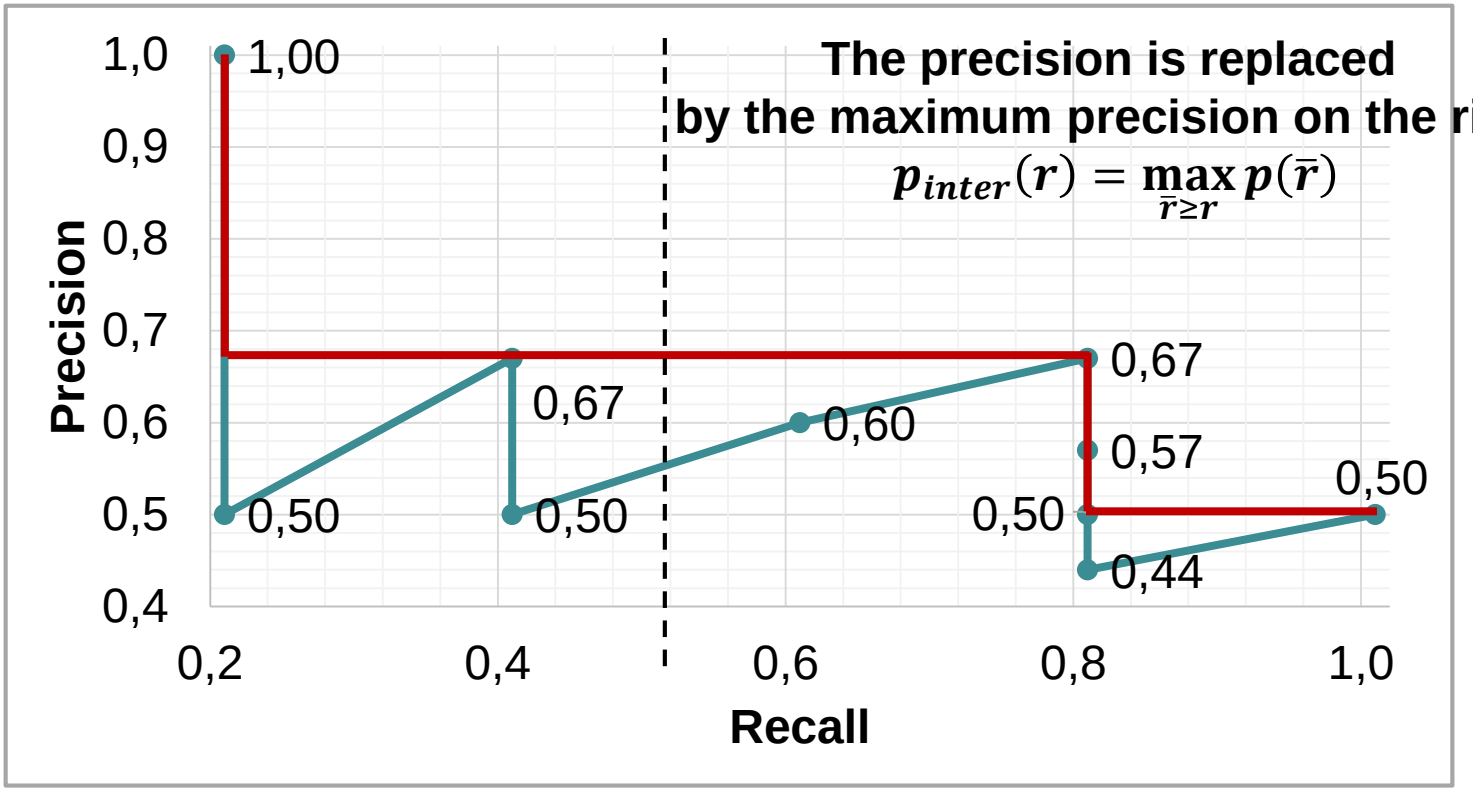
Average precision (6)

- Example of calculating average precision:
  - Constructing graph of precision as a function of recall
  - The graph is a zigzag curve



Average precision (7)

- Example of calculating average precision:
  - Calculating the area under the zigzag curve, i.e. interpolating and calculating the area under the stepped curve



DEEP MODELS FOR OBJECT DETECTION

Classification of deep models for object detection

- ***Two-stage models*** construct a set of hypotheses about bounding boxes, which are further classified and refined
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
  - R-FCN

- ***One-stage models*** involve constructing a set of detected bounding boxes using a single neural network
  - SSD
  - YOLOv1, *v2, *v3

Deep models (1)

Two-stage models

- ***R-CNN (2014)***
  - Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].
- ***Fast R-CNN (2015)***
  - Girshick R. Fast R-CNN. – 2015. – [https://arxiv.org/pdf/1504.08083.pdf], [https://ieeexplore.ieee.org/document/7410526].
- ***Faster R-CNN, R-FCN (2016)***
  - Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. – 2016. – [https://arxiv.org/pdf/1506.01497.pdf], [https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf].

Deep models (2)

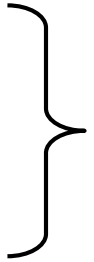
- Dai J., Li Y., He K., Sun J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. – 2016. – [https://arxiv.org/pdf/1605.06409.pdf], [https://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf].

- ***SSD (2016)***
  - Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector. – 2016. – [https://arxiv.org/pdf/1512.02325.pdf], [https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2].
- ***YOLOv1 (2015), *v2 (2016)***
  - Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015. – [https://arxiv.org/pdf/1506.02640.pdf], [https://ieeexplore.ieee.org/document/7780460].
  - Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. – 2016. – [https://arxiv.org/pdf/1612.08242.pdf], [https://pjreddie.com/darknet/yolo].

One-stage models

Deep models (3)

- ***YOLOv3 (2018)***
  - Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. – 2018. – [https://pjreddie.com/media/files/papers/YOLOv3.pdf].



- ***Note:*** a significant number of state-of-the-art neural networks that demonstrate high detection quality on public datasets are modifications of these models

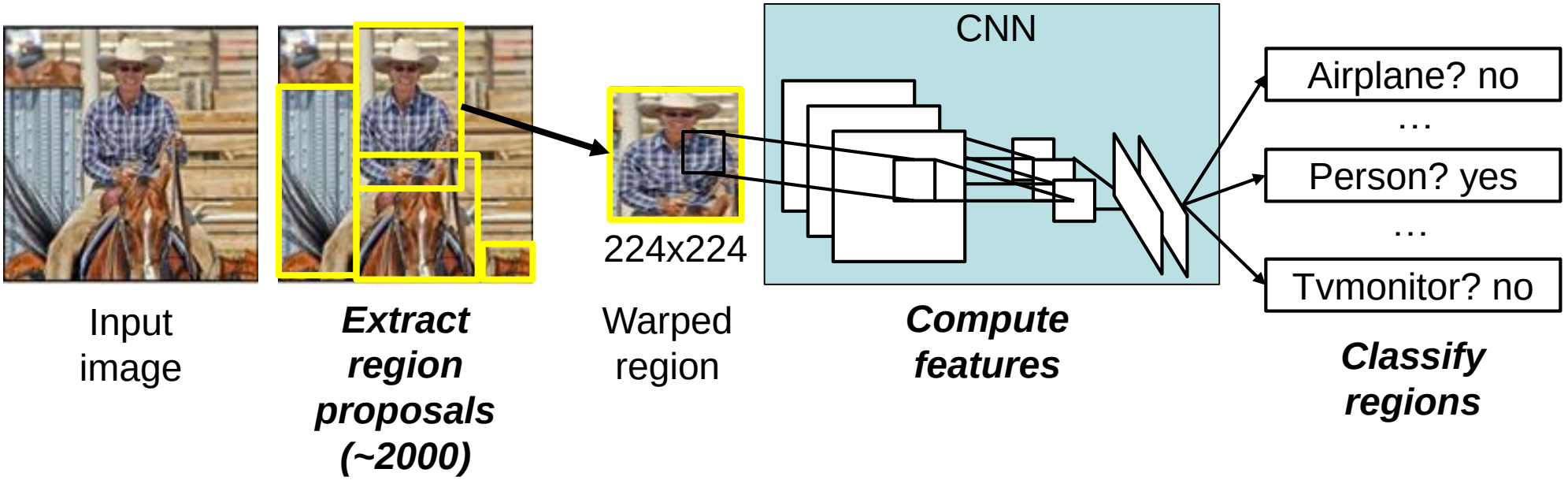
R-CNN (1)

- R-CNN (Region-based Convolutional Neural Network) is one of the first models that allowed to obtain high detection quality at the PASCAL VOC 2012 dataset
- The scheme of object detection using R-CNN:
  - Extract region proposals – areas of possible object presence (~2000 regions)
  - Compute features for each generated region proposal
  - Classify constructed region proposals
  - Construct bounding boxes

* Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].

R-CNN (2)

- The scheme of the R-CNN model:



* Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].

R-CNN (3)

- Extracting region proposals – areas of possible object presence (~2000 regions)
  - Scanning image
  - Selecting regions of interest using selective search algorithm
- Computing features for each generated region proposal
  - Processing each constructed region using a convolutional neural network
  - In the implementation of R-CNN, the AlexNet model is used (5 convolutional layers and 2 fully connected, the output is a feature vector of the size 4096)

* Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].

R-CNN (4)

- Classifying constructed region proposals
  - Extracting network output and redirecting it to the input of the support vector machine (SVM)
  - Using a set of binary SVM classifiers, each of which determines belonging to a particular class of objects
- Constructing bounding boxes
  - Extracting network output
  - Redirecting the network output to the linear regression input to determine the borders of the bounding box
  - Applying the greedy non-maximum suppression algorithm for each object class

* Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].

R-CNN (5)

- The main disadvantage of R-CNN is the slow model inference, so the model cannot be used in real-time systems
  - For each region proposal generated in the input image, a feed forward through the convolution network is required, it is ~2000 forwards per image
- The disadvantage of constructing the model is the need for training or fine-tuning of three groups of models:
  - Convolutional network for feature extraction
  - A set of binary SVM classifiers
  - Linear regression to tune the borders of bounding boxes

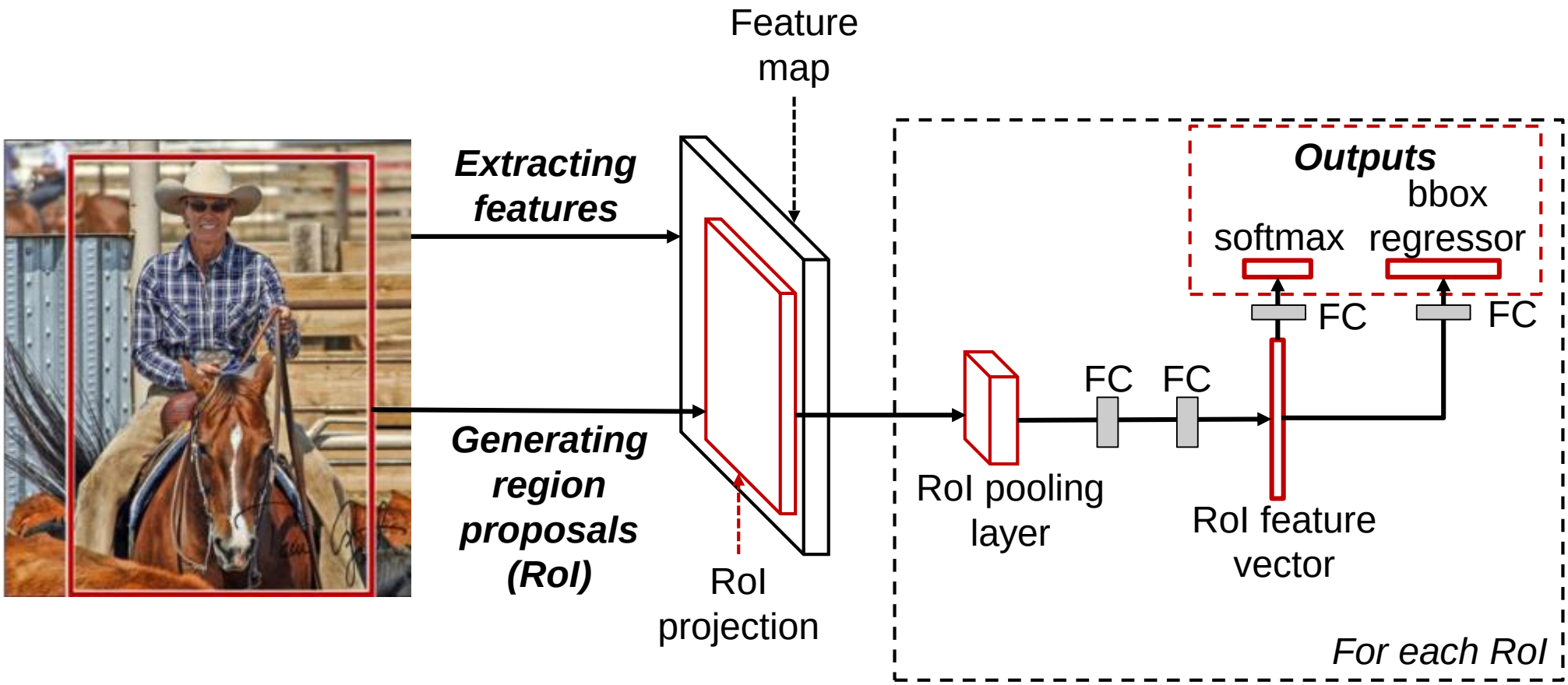
* Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].

Fast R-CNN (1)

- Fast R-CNN is an improvement of the R-CNN model aimed to accelerate computing in R-CNN
- Generating region proposals, the regions are able to overlap greatly, so the convolutional network will process identical fragments of the image that belong to different hypotheses
- The solution is to change the order of the stages of extracting features and generating region proposals (used algorithms are the same as for R-CNN)
- The difference is that a set of SVM classifiers and linear regression are replaced with neural networks

* Girshick R. Fast R-CNN. – 2015. – [https://arxiv.org/pdf/1504.08083.pdf], [https://ieeexplore.ieee.org/document/7410526].

Fast R-CNN (2)



* Girshick R. Fast R-CNN. – 2015. – [https://arxiv.org/pdf/1504.08083.pdf], [https://ieeexplore.ieee.org/document/7410526].

Fast R-CNN (3)

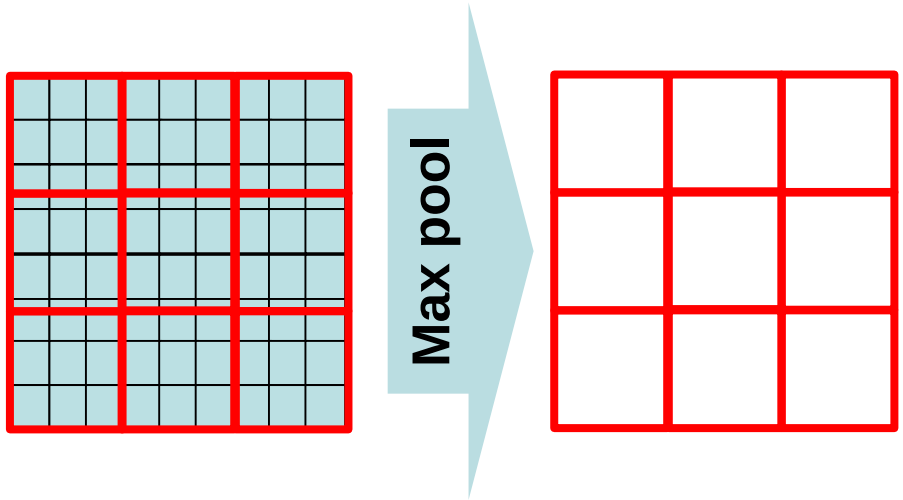
- RoI pooling layer combines information about constructed feature map for the input image and generated region proposal and generates features of the region
  - Input:
    - The output feature map from the last convolutional layer of the neural network, which provides feature extraction
    - The generated region proposal in the coordinate system associated with the input image
  - Output:
    - Features of the region (feature maps have the same size for all regions)

Fast R-CNN (4)

- RoI pooling layer combines information about constructed feature map for the input image and generated region proposal and generates features of the region
  - Algorithm:
    - The coordinates of the generated region are transformed into a system associated with a feature map constructed for the image
    - Further, we consider a region of a feature map of the size $w \times h$ corresponding to the generated region
    - A grid of the fixed size $W \times H$ ($W = H = 7$) is superimposed on the region
    - For each cell, the maximum pooling is applied, and a feature map of the spatial size $W \times H$ is constructed

Fast R-CNN (5)

- RoI pooling layer combines information about constructed feature map for the input image and generated region proposal and generates features of the region
  - Example for $w \times h = 9 \times 8$ and $W \times H = 3 \times 3$



Fast R-CNN (6)

- General part of the classifier and bounding box regressor:
  - 2 fully connected layers
  - The feature vector is used as inputs for two parallel branches of the classifier and regressor
- Classifier is a fully connected layer and a softmax function
  - The number of elements of a fully connected layer corresponds to the number of classes, taking into account the background
  - The output vector reflects the confidence of belonging to each class
- Regressor is a fully connected layer
  - The output vector is the per-class bounding-box regression offsets

Faster R-CNN (1)

- Faster R-CNN is a modification of Fast R-CNN, in which a special neural network RPN (Region Proposal Network) is used to generate region proposals
- By analogy with Fast R-CNN, a convolutional neural network (the best results on ResNet-101) is used to extract features from an image of an arbitrary resolution
- The constructed feature map is an input of the RPN, which applies a sliding window approach and generate a set of region proposals, as well as confidence maps of their belonging to each of the possible classes

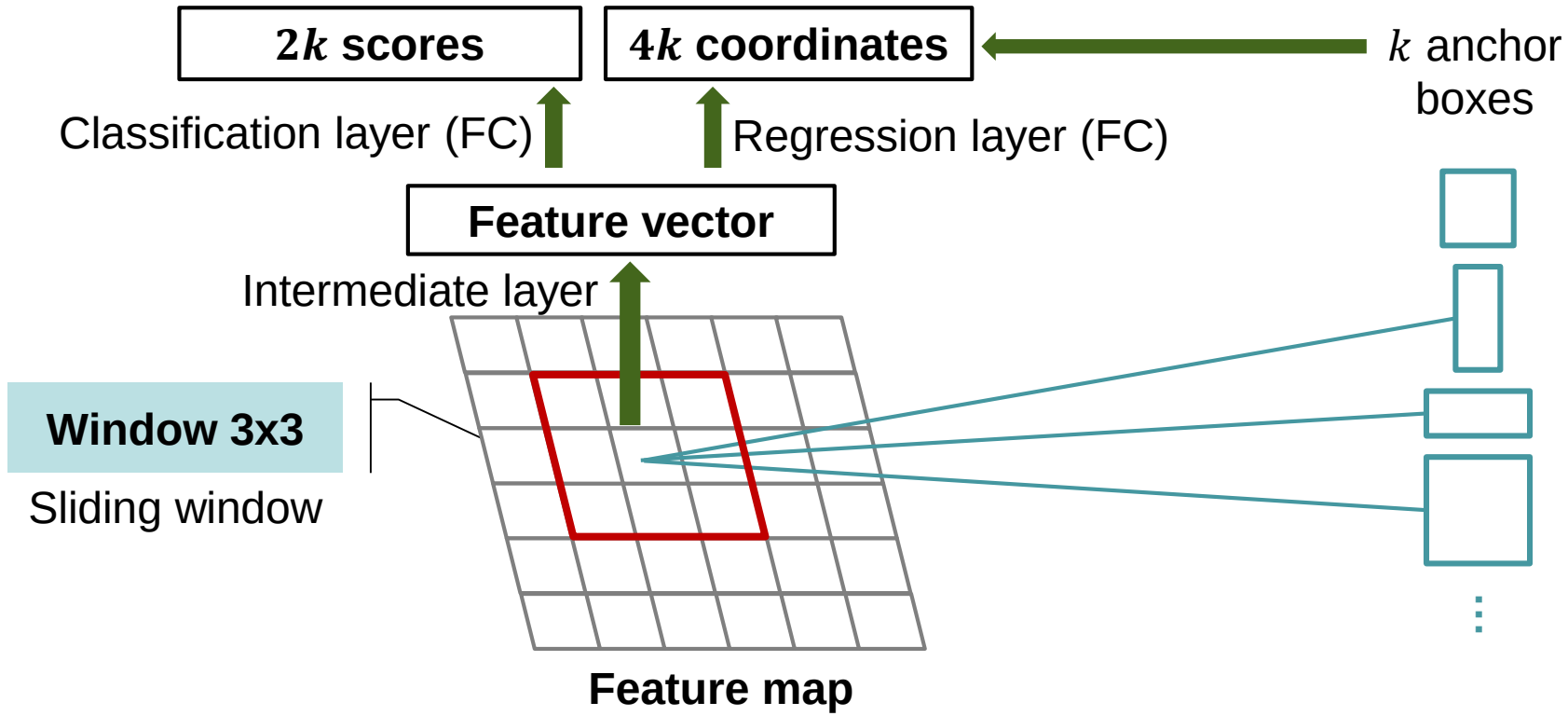
* Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. – 2016. – [https://arxiv.org/pdf/1506.01497.pdf], [https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf].

Faster R-CNN (2)

- RPN is a convolutional neural network
- Input:
  - Image of an arbitrary resolution
- Output:
  - A set of rectangular regions (hypotheses) and corresponding confidence vectors reflecting the probability of belonging to a class or background
- RPN consists of two parts:
  - A sequence of convolutional layers inherited from well-known models (e.g. ZF or VGG)
  - A small convolutional network for generating hypotheses, which implements a sliding window approach for a feature map constructed using a convolutional network (the first part of RPN)

Faster R-CNN (3)

- Region proposal network:



* Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. – 2016. – [https://arxiv.org/pdf/1506.01497.pdf], [https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf].

Faster R-CNN (4)

- Region proposal network:
  - A 3x3 window slides by a feature map, it corresponds to an effective receptive field of size 171x171 for ZF and 228x228 pixels for VGG, respectively
  - For each sliding window location, k region proposals are predicted. The center of the region proposal coincides with the center of the sliding window, the regions differ in aspect ratio
  - The intermediate layer is a convolutional layer with a 3x3 kernel and ReLU activation function. Each sliding window is mapped to a lower-dimensional vector (256-d for ZF and 512-d for VGG)
  - The classification and regression layers are 1-d convolutional layers

Faster R-CNN (5)

- Region proposal network:
  - The output of the regression layer is a vector of dimension $4k$, 4 coordinates for each region proposal, which correspond to the shape offsets. The k proposals are parameterized relative to k reference boxes, called ***anchors***
  - The output of the classification layer is a confidence vector of dimension $2k$, that estimates probability of object/not-object for each proposal (a binary classifier is implemented)

Faster R-CNN (6)

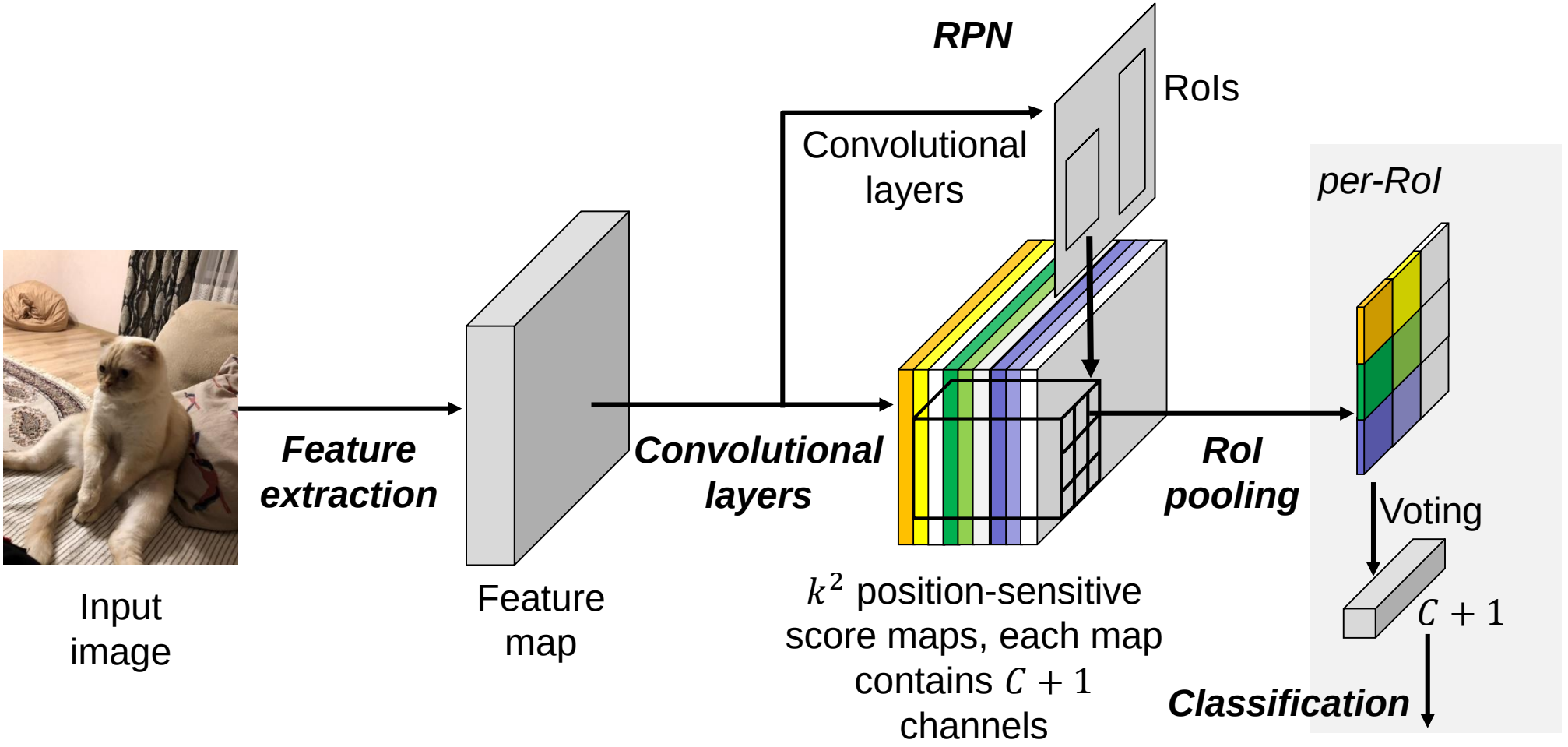
- Generating region proposals:
  - Each element of the input feature map corresponds to the anchor
  - To generate hypotheses, 2 parameters are used – the scale and aspect ratio of the rectangular region
    - VGG-16 reduces the scale of the original image by 16 times, 16 is the step of generating proposals in the coordinate system of the original image
    - If the scales are {8, 16, 32} and the aspect ratios are {1/2, 1/1, 2/1}, then 9 proposals are generated for each anchor
    - To get the next anchor in the original image, just move the previous anchor by 16
- ***Note:*** the model is trained as a single neural network, the error function is a weighted loss function of two branches corresponding to classification and regression

R-FCN (1)

- R-FCN (Region-based Fully Convolutional Network) is a logical improvement of Faster R-CNN
- The main idea of R-FCN is to construct position-sensitive score maps as network output

* Dai J., Li Y., He K., Sun J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. – 2016. – [https://arxiv.org/pdf/1605.06409.pdf], [https://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf].

R-FCN (2)



* Dai J., Li Y., He K., Sun J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. – 2016. – [https://arxiv.org/pdf/1605.06409.pdf], [https://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf].

R-FCN (3)

- The scheme of R-FCN:
  - Extracting features from the original image using some convolutional neural network
  - Appending convolutional layers and constructing position-sensitive score maps
    - The number of score maps is k^2, it corresponds to the number of relative positions of the object on the spatial grid $k \times k$, which is used to break a region (if $k = 3$, then relative positions encode the cases of “top-left”, “top-center”, “top-right”, …, “bottom-right”)
    - The depth of each feature map is $C + 1$, where C is a number of object classes
    - The depth of the combined feature map is $k^2(C + 1)$

R-FCN (4)

- The scheme of R-FCN:
  - Generating region proposals (RoIs) using a fully convolutional RPN
  - Combining score maps using position-sensitive RoI pooling layer
    - In accordance with the region location, the corresponding part of the position-sensitive score maps is cut out
    - These score maps are reorganized according to their relative positions
  - Classifying regions using the softmax classifier. Classifier input is a confidence vector of the region belonging to each of the object classes obtained by voting

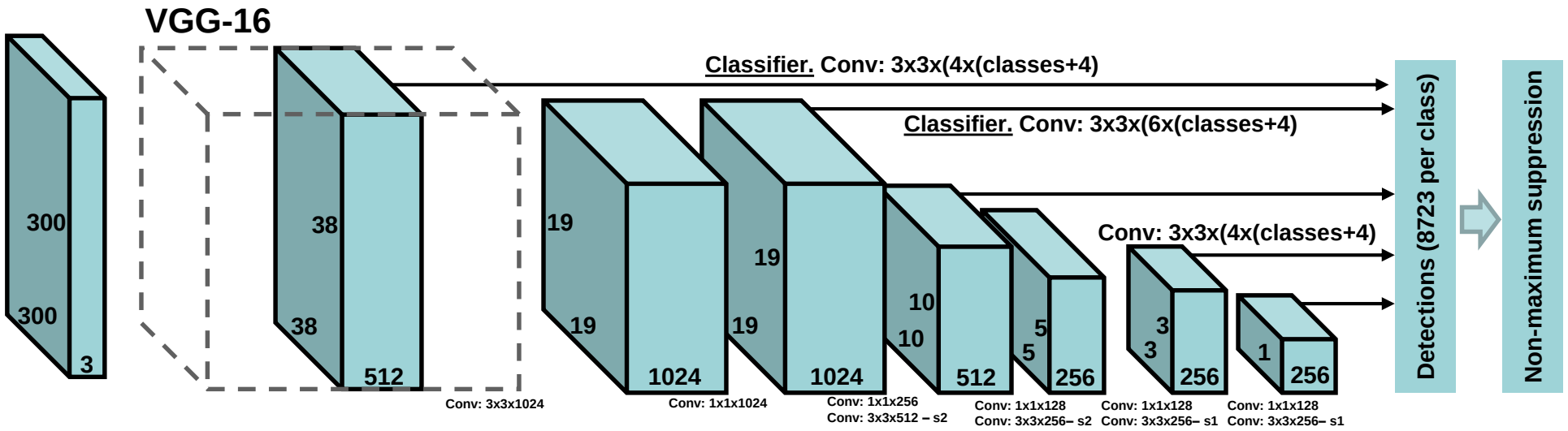
SSD (1)

- SSD (Single Shot Multibox Detector) allows you to predict the placement of bounding boxes and classify corresponding objects simultaneously
- SSD is a single convolutional neural network with neural network classifiers applied to the intermediate feature maps
- There are architectures for different input sizes (SSD300 – 300x300, SSD512 – 512x512 and others)

* Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector. – 2016. – [https://arxiv.org/pdf/1512.02325.pdf], [https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2].

SSD (2.1)

- The architecture of SSD300:



* Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector. – 2016. – [https://arxiv.org/pdf/1512.02325.pdf], [https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2].

SSD (2.2)

Layer (num_filters w×h, stride[, pad])	Shape of the output feature map	Classifier	Shape of the output feature map for the classifier	Number of proposals
image:	300×300×3			
conv1_1: 64 3×3, 1, 1 + ReLU	300×300×64			
conv1_2: 64 3×3, 1, 1 + ReLU	300×300×64			
pool1: max 2×2, 2	150×150×64			
conv2_1: 128 3×3, 1, 1 + ReLU	150×150×128			
conv2_2: 128 3×3, 1, 1 + ReLU	150×150×128			
pool2: max 2×2, 2	75×75×128			
conv3_1: 256 3×3, 1, 1 + ReLU	75×75×256			
conv3_2: 256 3×3, 1, 1 + ReLU	75×75×256			
conv3_3: 256 3×3, 1, 1 + ReLU	75×75×256			
pool3: max 2×2, 2	38×38×256			
conv4_1: 512 3×3, 1, 1 + ReLU	38×38×512			
conv4_2: 512 3×3, 1, 1 + ReLU	38×38×512			
conv4_3: 512 3×3, 1, 1 + ReLU	38×38×512	conv_c1: $\mathbf{4}(\boldsymbol{c}+\mathbf{4})$ 3×3, 1	38×38×[4(c+4)]	38*38*4=5776
pool4: max 2×2, 2	19×19×512			
conv5_1: 512 3×3, 1, 1 + ReLU	19×19×512			
conv5_2: 512 3×3, 1, 1 + ReLU	19×19×512			
conv5_3: 512 3×3, 1, 1 + ReLU	19×19×512			
pool5: max 3×3, 1, 1	19×19×512			
fc6: 1024 3×3, 1, 6 (dilation: 6) + ReLU	19×19×1024			
fc7: 1024 1×1, 1, 0 + ReLU	19×19×1024	conv_c2: $\mathbf{6}(\boldsymbol{c}+\mathbf{4})$ 3×3, 1	19×19×[6(c+4)]	19*19*6=2166
conv6_1: 256 1×1, 1, 0 + ReLU	19×19×256			
conv6_2: 512 3×3, 2, 1 + ReLU	10×10×512	conv_c3: $\mathbf{6}(\boldsymbol{c}+\mathbf{4})$ 3×3, 1	10×10×[6(c+4)]	10*10*6=600
conv7_1: 128 1×1, 1, 0 + ReLU	10×10×128			
conv7_2: 256 3×3, 2, 1 + ReLU	5×5×256	conv_c4: $\mathbf{6}(\boldsymbol{c}+\mathbf{4})$ 3×3, 1	5×5×[6(c+4)]	5*5*6=150
conv8_1: 128 1×1, 1, 0 + ReLU	5×5×128			
conv8_2: 256 3×3, 1, 0+ ReLU	3×3×256	conv_c5: $\mathbf{4}(\boldsymbol{c}+\mathbf{4})$ 3×3, 1	3×3×[4(c+4)]	3*3*4=36
conv9_1: 128 1×1, 1, 0 + ReLU	3×3×128			
conv9_2: 256 3×3, 1, 0 + ReLU	1×1×256	conv_c6: $\mathbf{4}(\boldsymbol{c}+\mathbf{4})$ 3×3, 1	1×1×[4(c+4)]	1*1*4=4
			The total number of region proposals:	8732

- The architecture of SSD300 (conv is a convolutional layer, pool is a pooling operation; $k(c+4)$ is a number of filters for the classifier, where k is a number of default bounding boxes, c is a number of classes, 4 corresponds to the number of bounding box sides (each value is a shift of the bounding box side relative to the side of the default bounding box)

SSD (3)

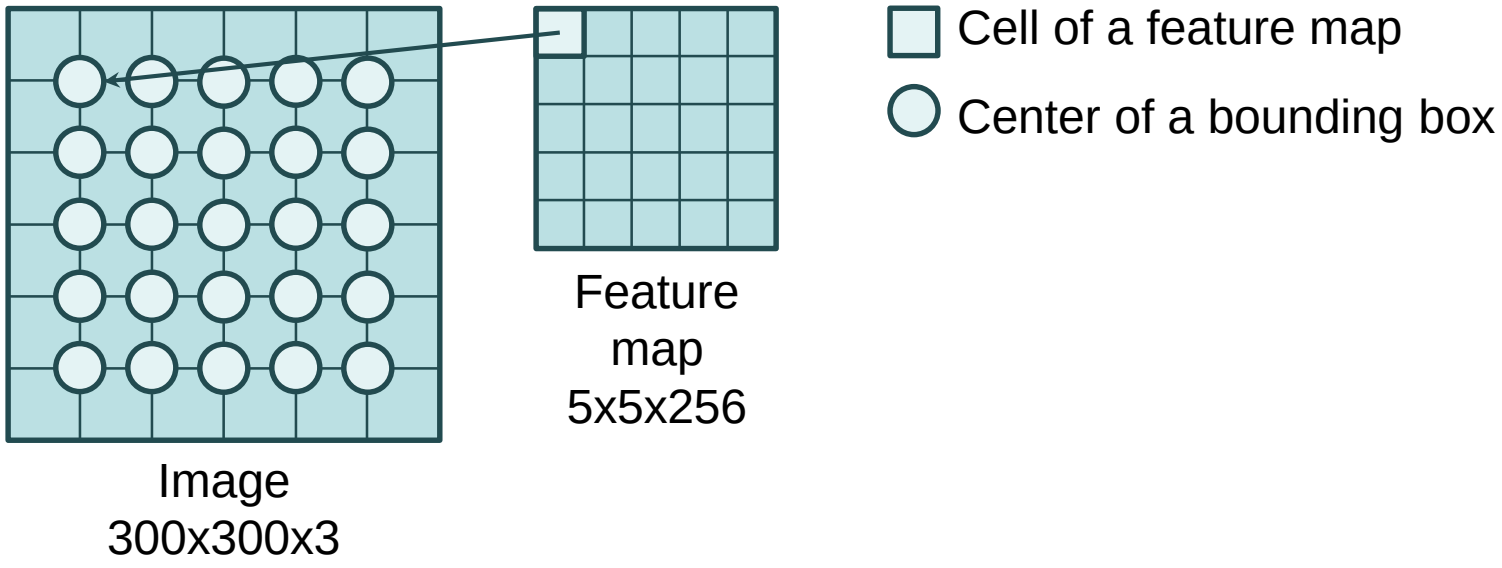
- The architecture of SSD300:
  - The model is based on VGG-16, convolutional layers appear without changes, and the fully connected layers are replaced by the fully convolutional ones
  - The feature maps of different scales are followed by convolutional layers to construct classifiers at these scales. These classifiers provide generating of possible object locations and their classification
  - To avoid duplication of the bounding boxes, the non-maximum suppression is applied

SSD (4)

- Convolutional layers of classifiers:
  - The feature map at some scale corresponds to the description of the image of a certain scale, the map cell corresponds to the description of a rectangular region of the image
  - Each classification layer maps a particular cell of the feature map to a set of default bounding boxes (the count is k)
  - For each bounding box, the confidence vector of the object belonging to the object classes (the vector size is C) and the vector of shifts of the bounding box sides relative to the sides of the default bounding box (the vector size is 4) are determined
  - If the feature map shape is $m \times n$ and each cell corresponds to k bounding boxes, then the number of outputs on the classification layer is $kmn(c + 4)$

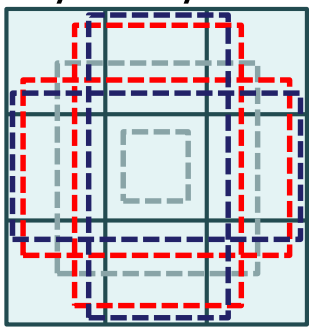
SSD (5)

- Generating hypotheses – bounding boxes potentially containing an object
  - Let us consider the example of SSD300 and the feature map constructed on the fourth convolutional layer



SSD (6)

- Generating hypotheses – bounding boxes potentially containing an object
  - For each center, an assumption about the object location is made
  - There are 4 or 6 hypotheses, corresponding to the number of bounding boxes for which the center is located at the selected point: two squares of different scales, two pairs of rectangles with the aspect ratios of $1/2$, $2/1$ and $1/3$, $3/1$



  - 4 hypotheses are used on the largest and two smallest feature maps

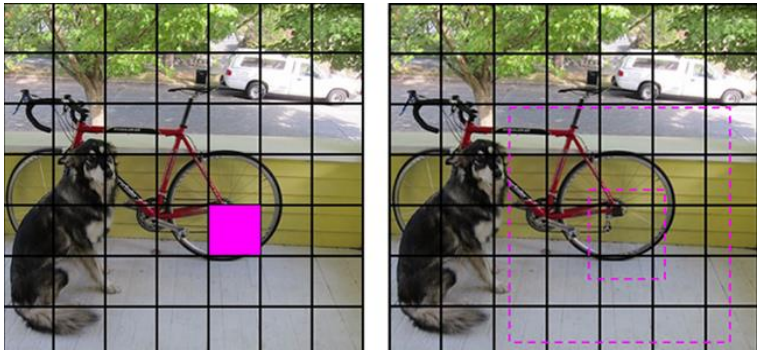
YOLOv1 (1)

- YOLO (You Only Look Once) is another model for object detection, represented by a single convolutional network that provides the construction of bounding boxes and the object classification simultaneously
- The model does not detect small objects well

* Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015. – [https://arxiv.org/pdf/1506.02640.pdf], [https://ieeexplore.ieee.org/document/7780460].

YOLOv1 (2)

- The input image is divided into cells by the $S \times S$ grid
- Each cell is responsible for predicting B bounding boxes



- For each bounding box, the parameters x, y, w, h, c are predicted, where (x, y) is the center of the bounding box relative to the cell boundaries, w and h are the width and height of the bounding box in the image coordinate system, c is the confidence of the object presence

* Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015. – [https://arxiv.org/pdf/1506.02640.pdf], [https://ieeexplore.ieee.org/document/7780460].

YOLOv1 (3)

- The confidence score of the object presence in a cell is determined as follows is as follows:

$$c = P(Object) \cdot IoU_{pred}^{truth},$$

where $P(Object)$ is the probability of object presence in the bounding box, IoU_{pred}^{truth} is the intersection over union between the predicted box and groundtruth

- The confidence score is calculated for each cell regardless the number of corresponding bounding boxes
- If no object exists in that cell, the confidence score is zero
- Otherwise, the confidence score is equal the intersection over union IoU_{pred}^{truth}

YOLOv1 (4)

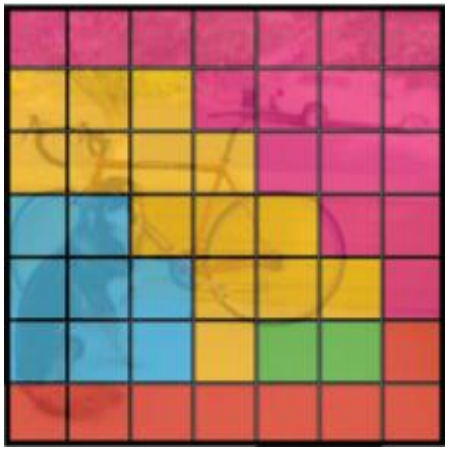
- Each grid cell predicts C conditional class probabilities $P(Class_i \mid Object)$, where C is a number of detected object classes
- The conditional probabilities of the classes are multiplied by the probability of object presence in the bounding box, which allows us to obtain probability score for each bounding box, depending on the class:

$$P(Class_i|Object) \cdot P(Object) \cdot IoU_{pred}^{truth} = P(Class_i) \cdot IoU_{pred}^{truth}$$

- The constructed scores reflect 2 aspects:
  - The probability of that class appearing in the box
  - How well the predicted box fits the object

YOLOv1 (5)

Grid on the input image



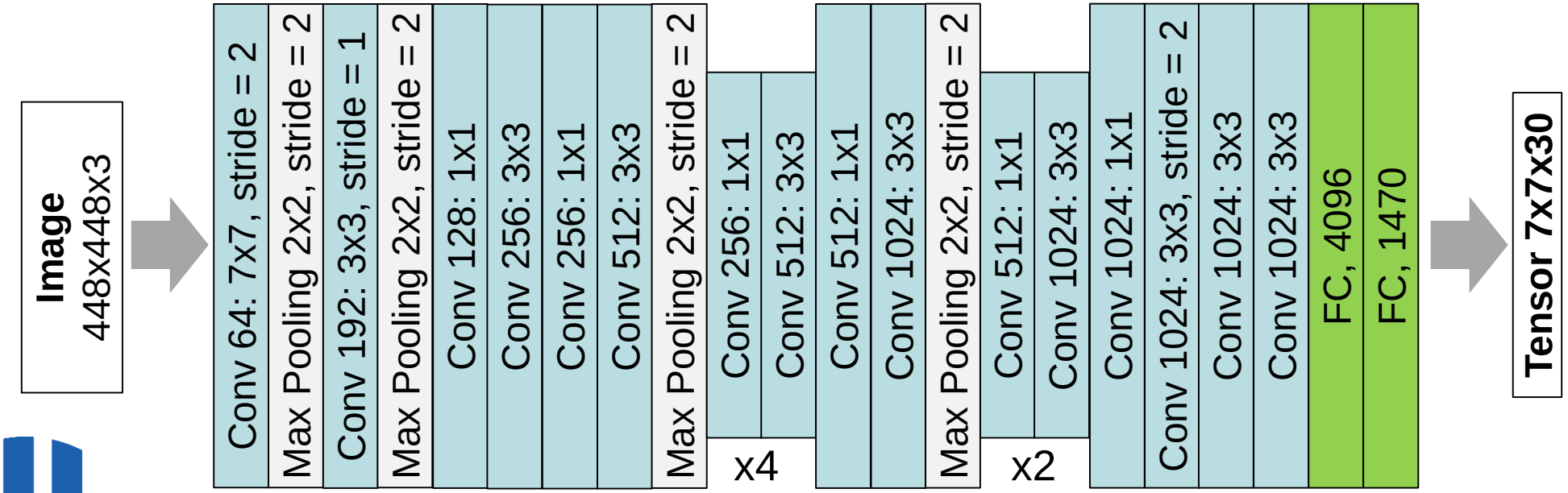
Class probability map
(different colors correspond
to different classes)

- The prediction is a tensor of the size $S \times S \times (B * 5 + C)$, each cell of the grid $S \times S$ predicts B bounding boxes and C probabilities scores

* Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015. – [https://arxiv.org/pdf/1506.02640.pdf], [https://ieeexplore.ieee.org/document/7780460].

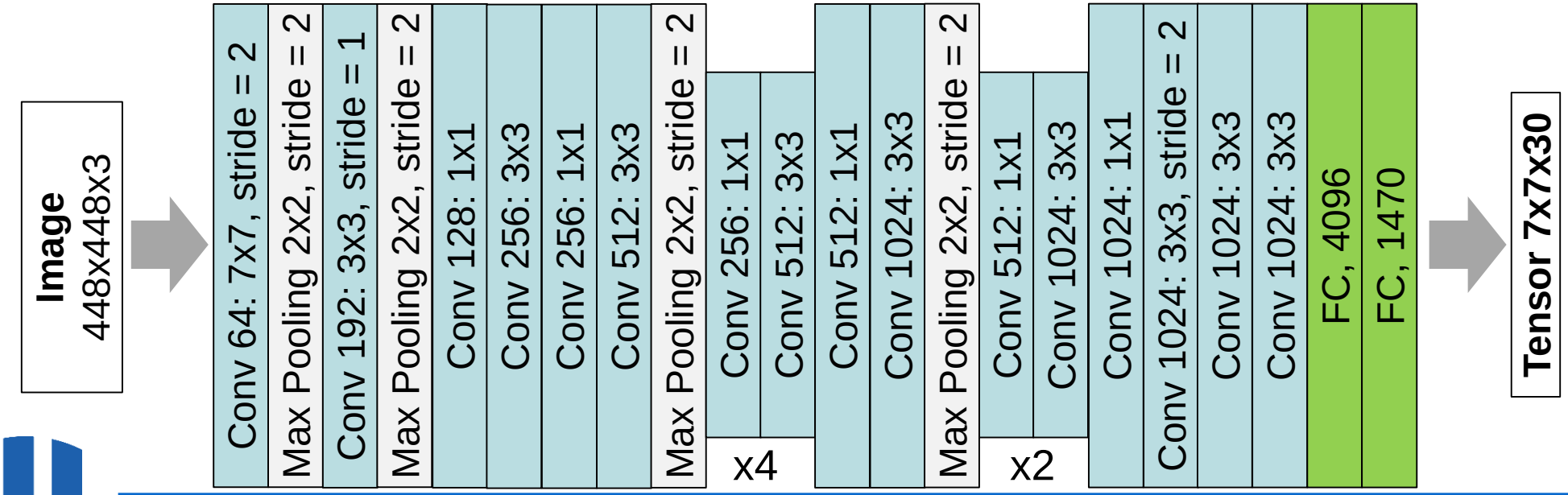
YOLOv1 (6)

- YOLOv1 is based on the GoogLeNet model:
  - The network contains 24 convolutional layers and 2 fully connected layers following them; after each layer, the ReLU activation function is applied
  - Instead of the initial inception modules, a convolutional layer is used that reduces the image dimension



YOLOv1 (7)

- YOLOv1 is based on the GoogLeNet model:
  - For the PASCAL VOC dataset the number of cells is $S = 7$, the number of bounding boxes is $B = 2$, the number of object classes is $C = 20$
  - First, the network is trained on ImageNet to fine-tune 20 convolutional and 1 fully connected layers in images 224x224



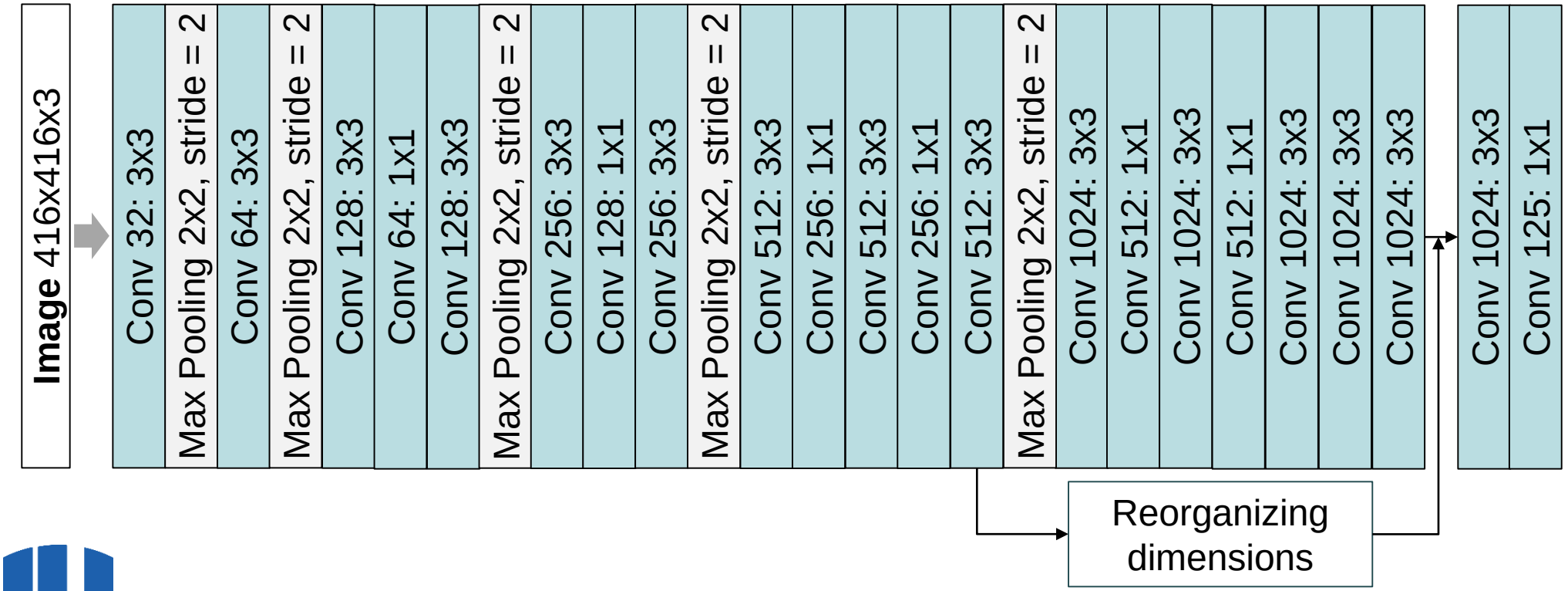
YOLOv2 (1)

- YOLOv2 is a modification of YOLOv1
  - Batch normalization for each convolutional layer
  - Preliminary fine-tuning of convolutional layers on ImageNet is carried out on images with a resolution of 448x448, which makes it possible to configure filters for working at high resolution
  - Using bounding box priors instead of direct predicting box coordinates
  - Multiscale training, i.e. every 10 batches the image resolution randomly changes {320, 352, ..., 608}
  - Increasing the number of object categories (YOLO9000)

* Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. – 2016. – [https://arxiv.org/pdf/1612.08242.pdf], [https://pjreddie.com/darknet/yolo].

YOLOv2 (2)

- The architecture of YOLOv2:
  - The output tensor has a shape of 13x13x125: 13x13 corresponds to the grid of the original image, 5 bounding boxes in each cell (25 parameters each)



YOLOv2 (3)

- Using priors instead of direct predicting coordinates
- Generating anchors using the k-means clusterization algorithm
  - Instead of manually selecting B bounding boxes, k-means clustering is started on the training set of bounding boxes to automatically search for good initial approximations
  - Metric of the distance between the bounding box and the cluster is as follows:

$$d(box, centroid) = 1 - IoU(box, centroid)$$

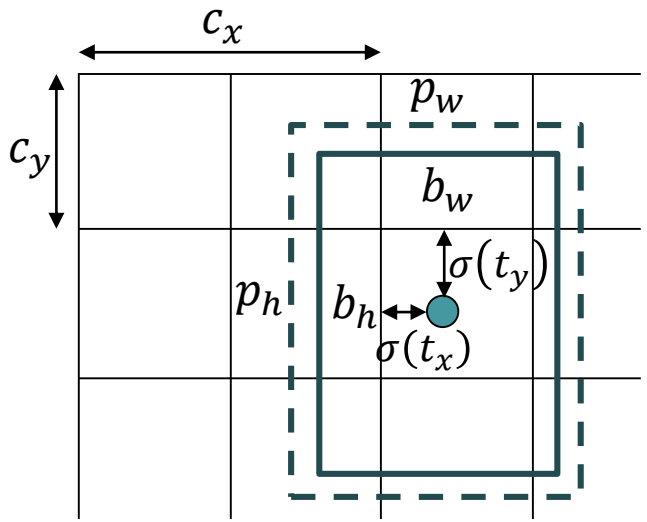
  - It has been experimentally shown that $k = 5$ is a good compromise between the model complexity and high response
  - For each cell of the input image, 5 priors are constructed

YOLOv2 (4)

- The network predicts 5 components for each bounding box t_x, t_y, t_w, t_h and t_o and the confidence score vector (for 20 classes)

$$b_x = \sigma(t_x) + c_x,\ b_y = \sigma(t_y) + c_y,$$
$$b_w = p_w e^{t_w},\ b_h = p_h e^{t_h},$$
$$P(object) * IoU(b, object) = \sigma(t_o)$$

where (c_x, c_y) is an offset from the top left corner of the image, p_w, p_h are the width and height of the bounding box prior,
t_x is a shift by x,
t_y is a shift by y,
t_o is a confidence score,
$\sigma(\cdot)$ is a sigmoid function

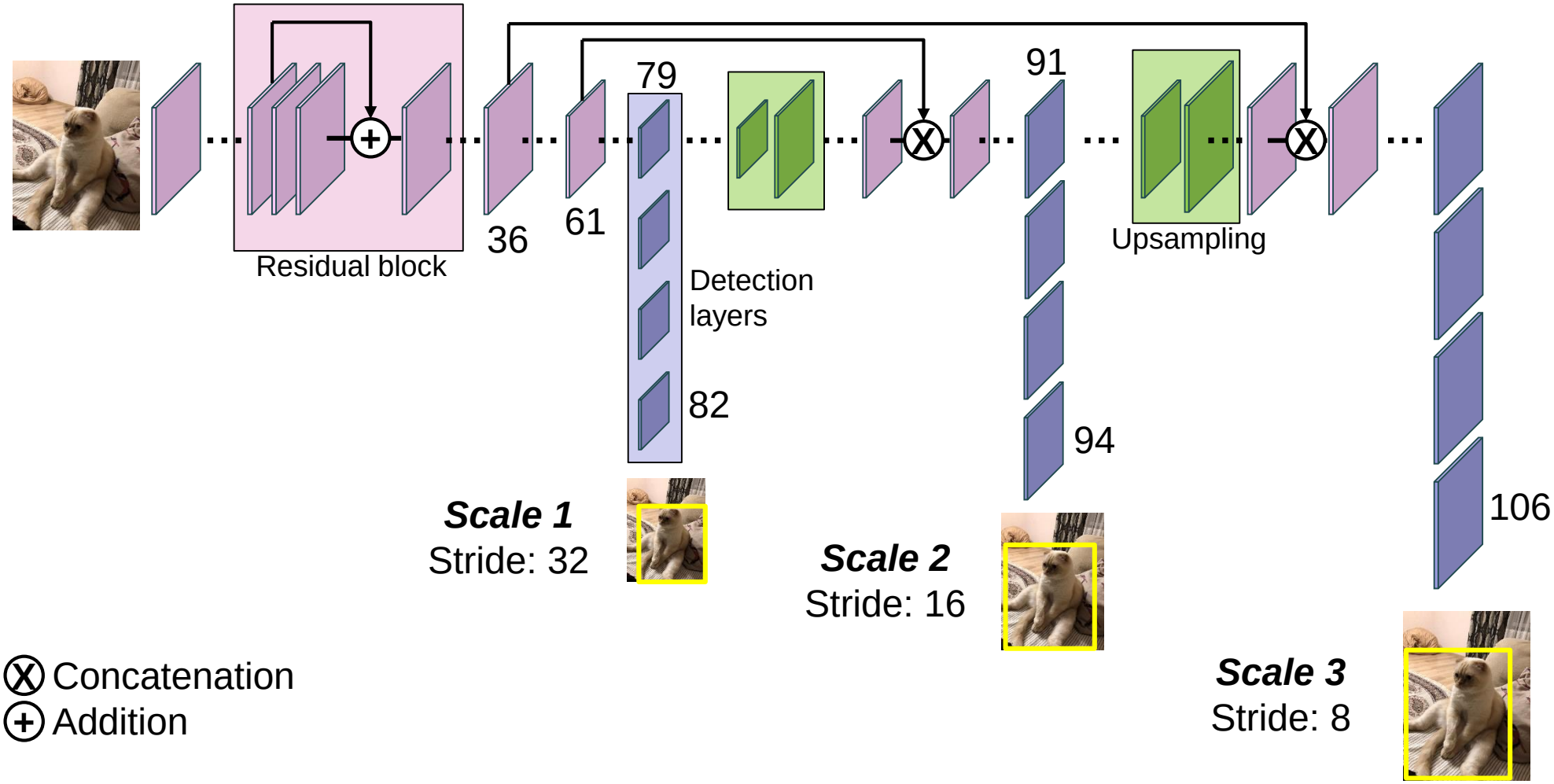


YOLOv3 (1)

- YOLOv3 is an improvement of YOLOv2
  - Increasing network depth (106 fully convolutional layers)
  - Adding residual connections
  - Detecting objects at three different scales of feature maps
  - Using three bounding box priors on each scale instead of five

* Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. – 2018. – [https://pjreddie.com/media/files/papers/YOLOv3.pdf].

YOLOv3 (2)



* $255 = B \times (5 + C) = 3 \times (5 + 80)$, 80 classes for the MS COCO dataset.

** What's new in YOLO v3? [https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b].

COMPARISON OF DEEP MODELS FOR OBJECT DETECTION

Detection quality change (1)

- Training dataset: PASCAL VOC 2012 + third-party dataset (as usual, MS COCO)
- Test dataset: PASCAL VOC 2012
- Number of object classes: 20 classes
- Quality metric: average precision
- Models:
  - R-CNN (R-CNN (bbox reg)*) is R-CNN based on the 16-layered convolutional network, which is trained on ILSVRC 2012 and fine-tuned on VOC 2012 trainval, SVM detectors are trained on VOC 2012 trainval

* It is a model name on the evaluation page of PASCAL VOC 2012 [http://host.robots.ox.ac.uk:8080/leaderboard/displaylb_main.php?challengeid=11&compid=4].

Detection quality change (2)

- Faster R-CNN (Faster RCNN, ResNet (VOC+COCO)*) is an improvement of Faster RCNN. It is based on ResNet, trained on ImageNet and fine-tuned on MS COCO trainval, fine-tuned on VOC 2007 trainval+test and VOC 2012 trainval
- R-FCN (R-FCN, ResNet (VOC+COCO)) is R-FCN based on ResNet-101. It is trained on ImageNet and sequentially fine-tuned on MS COCO trainval, VOC 2007 trainval+test and VOC 2012 trainval
- SSD300ft (SSD300 VGG16 07++12+COCO) is SSD300 trained on MS COCO trainval35k and fine-tuned on VOC07 trainval + test and VOC12 trainval
- YOLOv2 is the model which was described in the lecture
- ATLDETv2 is an ensemble of two models based on ResneXt152_32x8d (the model description is not published)

Detection quality change (3)

Model	Year	mAP, %	AP, % (for the selected object classes)				
			bus	car	cat	person	train
R-CNN	2014	62.4	65.9	66.4	84.6	76.0	54.2
Faster R-CNN	2015	83.8	86.3	87.8	94.2	89.6	90.3
SSD300ft	2016	79.3	84.9	84.0	93.4	85.6	88.3
R-FCN	2016	**85.0**	**86.7**	**89.0**	**95.8**	**91.1**	**92.0**
YOLOv2	2017	75.4	81.2	78.2	92.9	88.6	88.8
ATLDETv2	2019	**92.9**	**95.5**	**95.7**	**98.0**	**96.1**	**96.2**

- From 2014 to 2019 average precision increased by 30% due to the application of the considered approaches
- Lightweight models (YOLOv2) show lower quality results
- According to the results of PASCAL VOC 2012*, a large number of modifications of the considered models are being developed

* Detection Results: VOC2012 [http://host.robots.ox.ac.uk:8080/leaderboard/displaylb_main.php?challengeid=11&compid=4].

Comparison of detection accuracy and inference time

- Train dataset: PASCAL VOC 2007+2012
- Test dataset: PASCAL VOC 2007
- Quality metric: mean average precision
- Infrastructure: NVIDIA M40 or Titan X

Good quality, but low FPS (the model does not infer in real time)

High FPS but poor quality

The compromise between quality and inference time

Model	mAP, %	FPS
Fast R-CNN	70.0	0,5
Faster R-CNN VGG-16	73.2	7
Faster R-CNN ResNet	76.4	5
YOLO	63.4	**45**
SSD500	76.8	19
YOLOv2 544x544	78.6	40

* Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. – 2016. – [https://arxiv.org/pdf/1612.08242.pdf], [https://pjreddie.com/darknet/yolo].

Conclusion

- Deep models for object detection are not limited to those discussed in this lecture; there are many modifications of basic architectures
- There are a large number of modifications of the considered architectures (in particular, Faster R-CNN and SSD), as evidenced by the results of well-known competitions for object detection
- ***The optimal model is a compromise between accuracy and speed***
  - The accuracy is determined by the requirements for solving a practical problem (the accuracy varies depending on the test data!)
  - The speed is determined by the available computational resources (high inference speed on powerful GPUs is not a good indicator)

Literature (1)

- Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. – 2014. – [https://arxiv.org/pdf/1311.2524.pdf], [https://ieeexplore.ieee.org/abstract/document/6909475].
- Girshick R. Fast R-CNN. – 2015. – [https://arxiv.org/pdf/1504.08083.pdf], [https://ieeexplore.ieee.org/document/7410526].
- Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. – 2016. – [https://arxiv.org/pdf/1506.01497.pdf], [https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf].

Literature (2)

- Dai J., Li Y., He K., Sun J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. – 2016. – [https://arxiv.org/pdf/1605.06409.pdf], [https://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf].
- Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C.-Y., Berg A.C. SSD: Single Shot MultiBox Detector. – 2016. – [https://arxiv.org/pdf/1512.02325.pdf], [https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2].
- Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection. – 2015. – [https://arxiv.org/pdf/1506.02640.pdf], [https://ieeexplore.ieee.org/document/7780460].

Literature (3)

- Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger. – 2016. – [https://arxiv.org/pdf/1612.08242.pdf], [https://pjreddie.com/darknet/yolo].
- Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. – 2018. – [https://pjreddie.com/media/files/papers/YOLOv3.pdf].

Authors

- **Turlapov Vadim Evgenievich,** Dr., Prof., department of computer software and supercomputer technologies
  vadim.turlapov@itmm.unn.ru
- **Vasiliev Engeny Pavlovich,** lecturer, department of computer software and supercomputer technologies
  evgeny.vasiliev@itmm.unn.ru
- **Getmanskaya Alexandra Alexandrovna,** lecturer, department of computer software and supercomputer technologies
  alexandra.getmanskaya@itmm.unn.ru
- **Kustikova Valentina Dmitrievna**
  Phd, assistant professor, department of computer software and supercomputer technologies
  valentina.kustikova@itmm.unn.ru