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»

Goals and tasks, course structure. The general scheme of solving computer vision problems using deep learning

Supported by Intel

Kustikova Valentina

Content

- ❑ Goals
- ❑ Course structure
- ❑ Computer vision problems in the course
- ❑ General scheme of solving computer vision problems using deep learning
- ❑ Software used at different stages of solving computer vision problems
- ❑ Conclusion

Goals

- ***The goal*** of this lecture is to represent the course structure and the statement of computer vision problems that will be solved in the course using existing deep learning models

COURSE STRUCTURE

Course goals and tasks

- ❑ ***The goal*** of the course is to study modern deep models and deep learning methods for solving classical computer vision problems (image classification, object detection, semantic segmentation)

- ❑ ***The course tasks:***
  - – To consider the statement of classical computer vision problems (classification/recognition, object detection, segmentation)
  - – To study modern deep models and deep learning methods to solve stated computer vision problems
  - – To apply these models to solve practical problems

Course structure

- ❑ ***Lectures*** (7 lectures)
  - – The first lecture is introductory one and represents the classical computer vision problems
  - – The other lectures consider modern deep learning models for solving these computer vision problems
- ❑ ***Practices*** (4 practical tasks)
  - – The first three practices involve solving classical computer vision problems (image classification, object detection and tracking) using trained deep learning models
  - – The last practice involves the development of a simple video analytics application

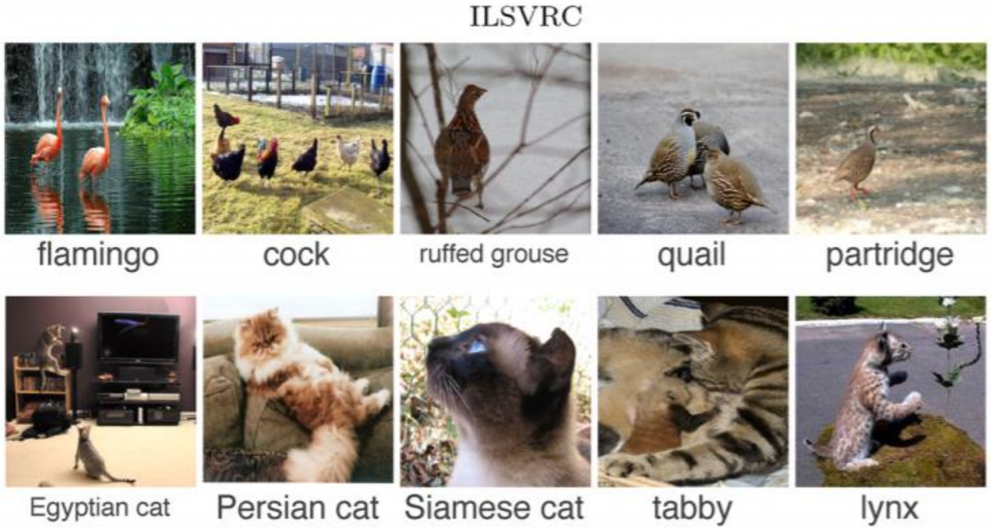
CLASSICAL COMPUTER VISION PROBLEMS

Review plan for each problem

- ❑ ***Problem statement***
  - – Informal problem description
  - – Mathematical problem statement
- ❑ ***Well-known datasets***
  - – Examples of problems and corresponding datasets
- ❑ ***Common quality metrics for considered problems***
  - – Quality metrics used to compare deep models in subsequent lectures
  - – ***Notes:***
    - There are various quality measurements which reflect different aspects
    - Some of the existing metrics will be further discussed in subsequent lectures

Image classification. Problem statement (1)

- The problem of image classification is to match the image with the class of objects represented in the image
- Examples of images and their corresponding classes:



* Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A.C., Fei-Fei L. ImageNet Large Scale Visual Recognition Challenge // International Journal of Computer Vision, 2015.

Image classification. Problem statement (2)

- The original image is represented by a set of pixel intensities $I = \left(I_{ij}^{k}\right)_{\substack{0\le i<w\\ 0\le j<h\\ 0\le k<3}}$, where w and h are image width and height, k is a number of color channels of the image
- $C = \{0, 1, \dots, N-1\}$ is a set of object classes in the image, the set of class identifiers uniquely corresponds to the set of class names
- The problem of image classification is to match each image with the class to which the image belongs to

$$\varphi : I \to C$$

Image classification. Datasets

- **MNIST** is a dataset of handwritten numbers (10 classes) [http://yann.lecun.com/exdb/mnist]
- **Fashion MNIST** contains images of various types of clothes (10 classes) [https://github.com/zalandoresearch/fashion-mnist]
- **Cifar-10 / Cifar-100** contains real-life images of 10/100 classes [http://www.cs.utoronto.ca/~kriz/cifar.html]
- **Caltech-101 / Caltech-256** contains real-life images of 101/256 classes [http://www.vision.caltech.edu/Image_Datasets/Caltech101]/ [http://www.vision.caltech.edu/Image_Datasets/Caltech256]
- **ImageNet** is a large dataset of natural images belonging to 1000 classes [http://www.image-net.org]

Image classification. Quality metrics

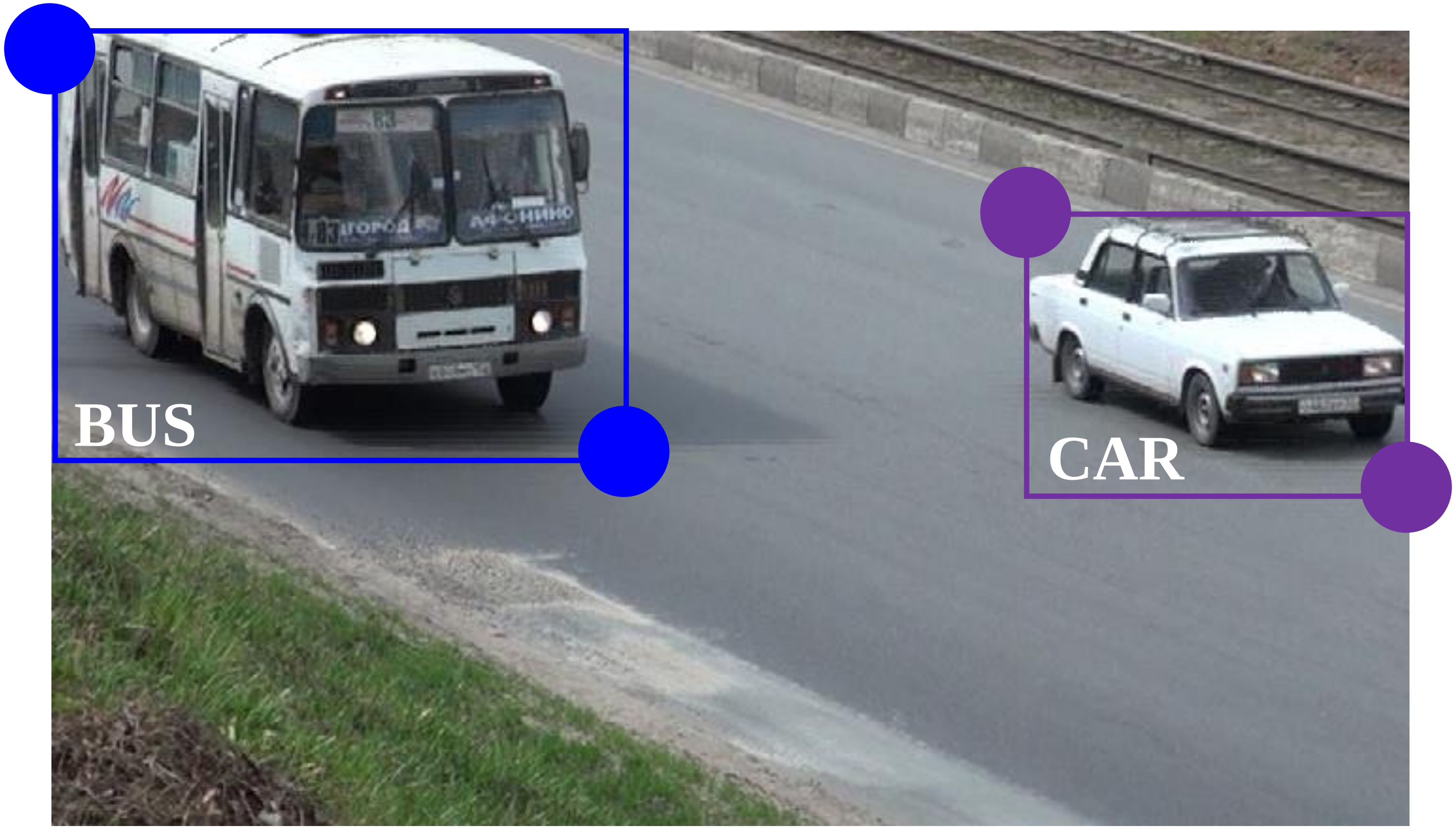
- Supposed N is the number of image categories
- For every image $I_j, j = \overline{1, S}$ in the dataset, classification model predicts a vector of confidences $p^j = \left(p_1^j, p_2^j, \dots, p_N^j\right)$, where p_i^j is the confidence of the assumption that the image I_j belongs to the class i
- ***Top-K accuracy*** is as follows:

$$topK = \frac{\sum_{j=1}^{S} 1_{\left\{i_1^j, i_2^j, \dots, i_K^j\right\}}(l_j)}{S},$$

where $\left\{i_1^j, i_2^j, \dots, i_K^j\right\} \subseteq \{1,2, \dots, N\}$, $p_{i_1^j}^j, p_{i_2^j}^j, \dots, p_{i_K^j}^j$ are K maximal confidences, l_j is the class to which the image I_j belongs according to the groundtruth, $1_{\left\{i_1^j, i_2^j, \dots, i_K^j\right\}}(l_j)$ is an indicator function

Object detection. Problem statement (1)

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

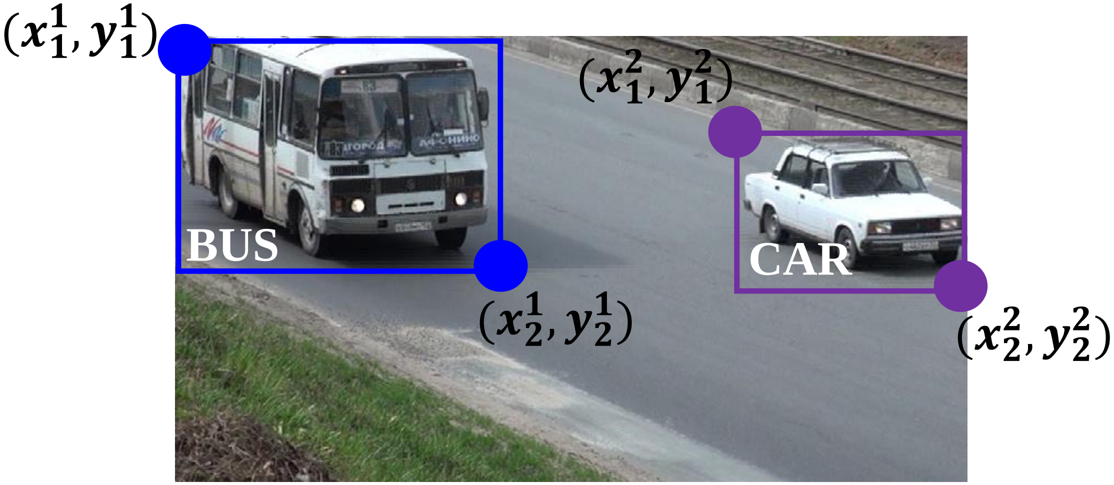


Object detection. Problem statement (2)

- 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.)

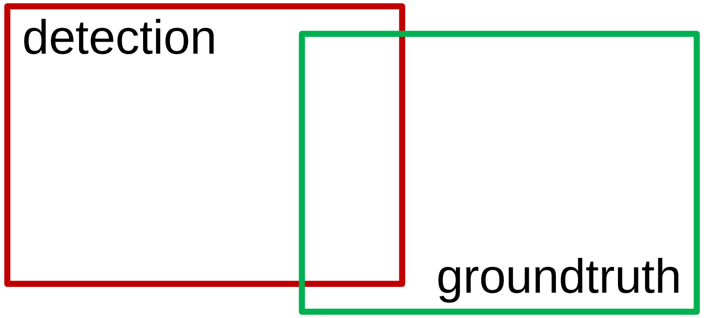


Object detection. Datasets

- **PASCAL VOC 2007, 2012** contain images and bounding boxes for objects of 20 classes [http://host.robots.ox.ac.uk/pascal/VOC]
- **MS COCO** is a set of images and bounding boxes for objects of 80 classes [http://cocodataset.org]
- **Open Images Dataset** contains images and bounding boxes for objects of 600 classes [https://storage.googleapis.com/openimages/web/index.html]
- **Street View House Numbers (SVHN)** is a set of images and bounding boxes for the numbers in natural inscriptions on them (in particular, on house identifiers) [http://ufldl.stanford.edu/housenumbers]
- **Stanford Dogs Dataset** contains images of 120 breeds of dogs [http://vision.stanford.edu/aditya86/ImageNetDogs]

Object detection. Quality metrics (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)



Object detection. Quality metrics (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

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

Object detection. Quality metrics (3)

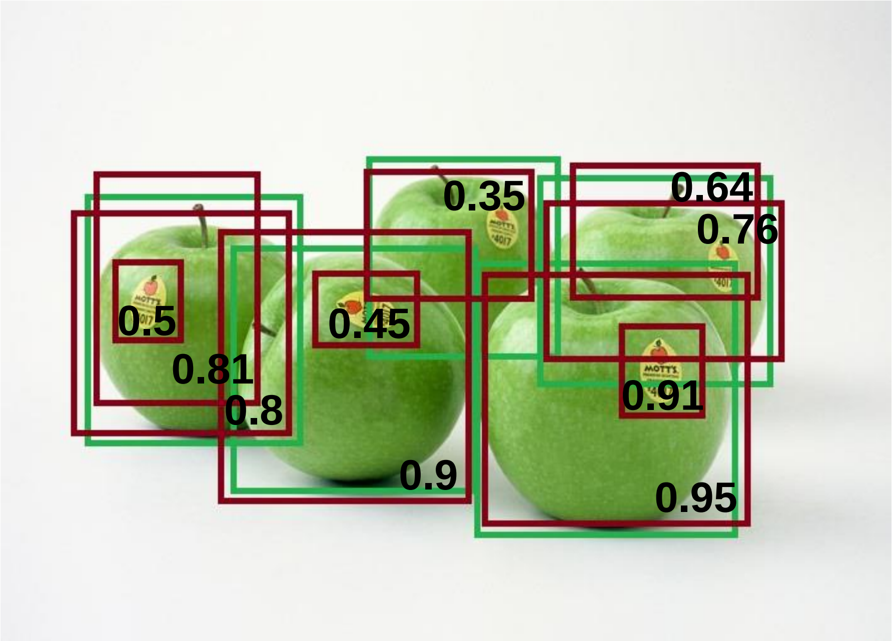
- ***Average precision*** is a mean of precisions

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

- 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

Object detection. Quality metrics (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



Object detection. Quality metrics (5)

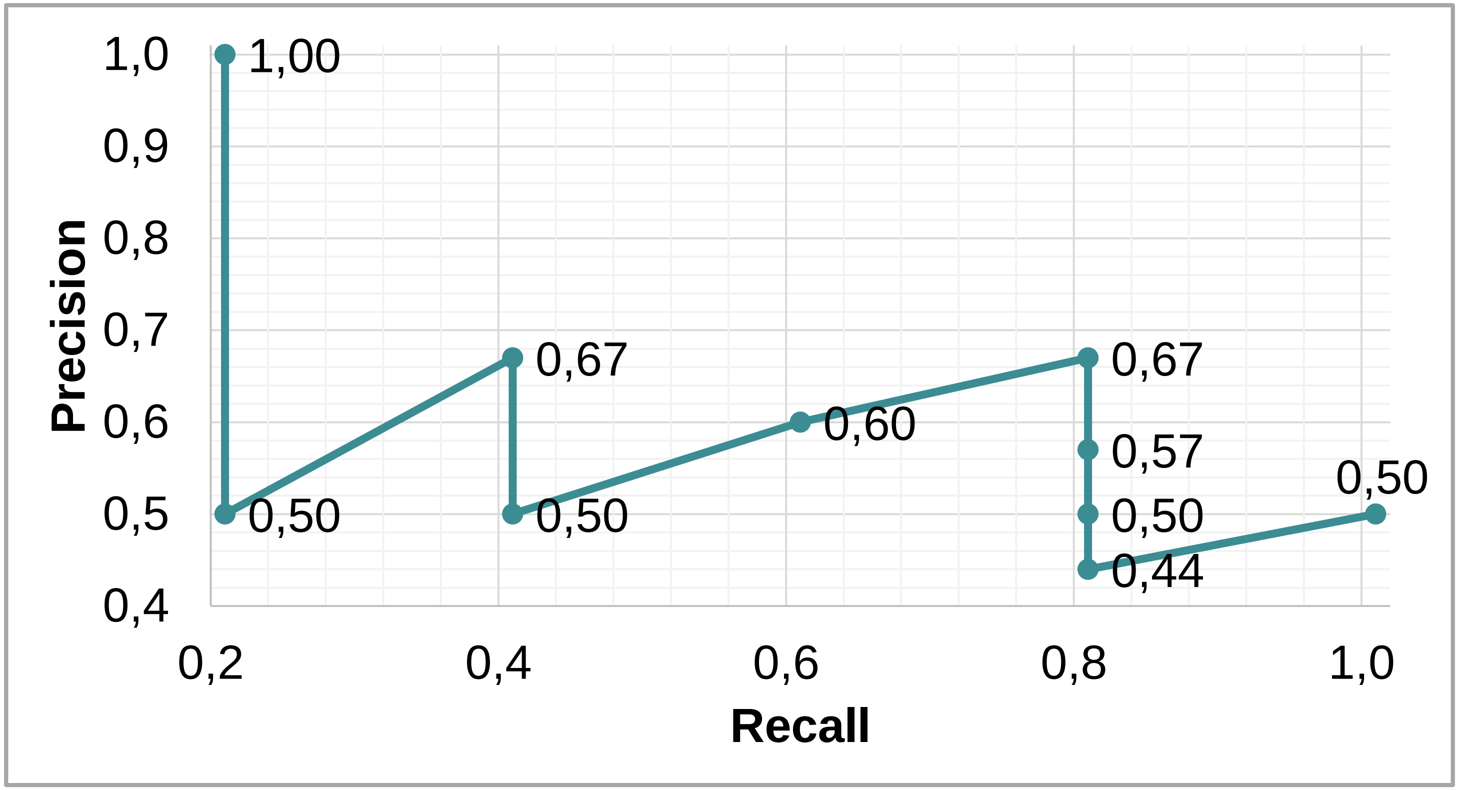
- Example of calculating average precision:
  - Sorting bounding boxes, calculating precision and recall

#	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

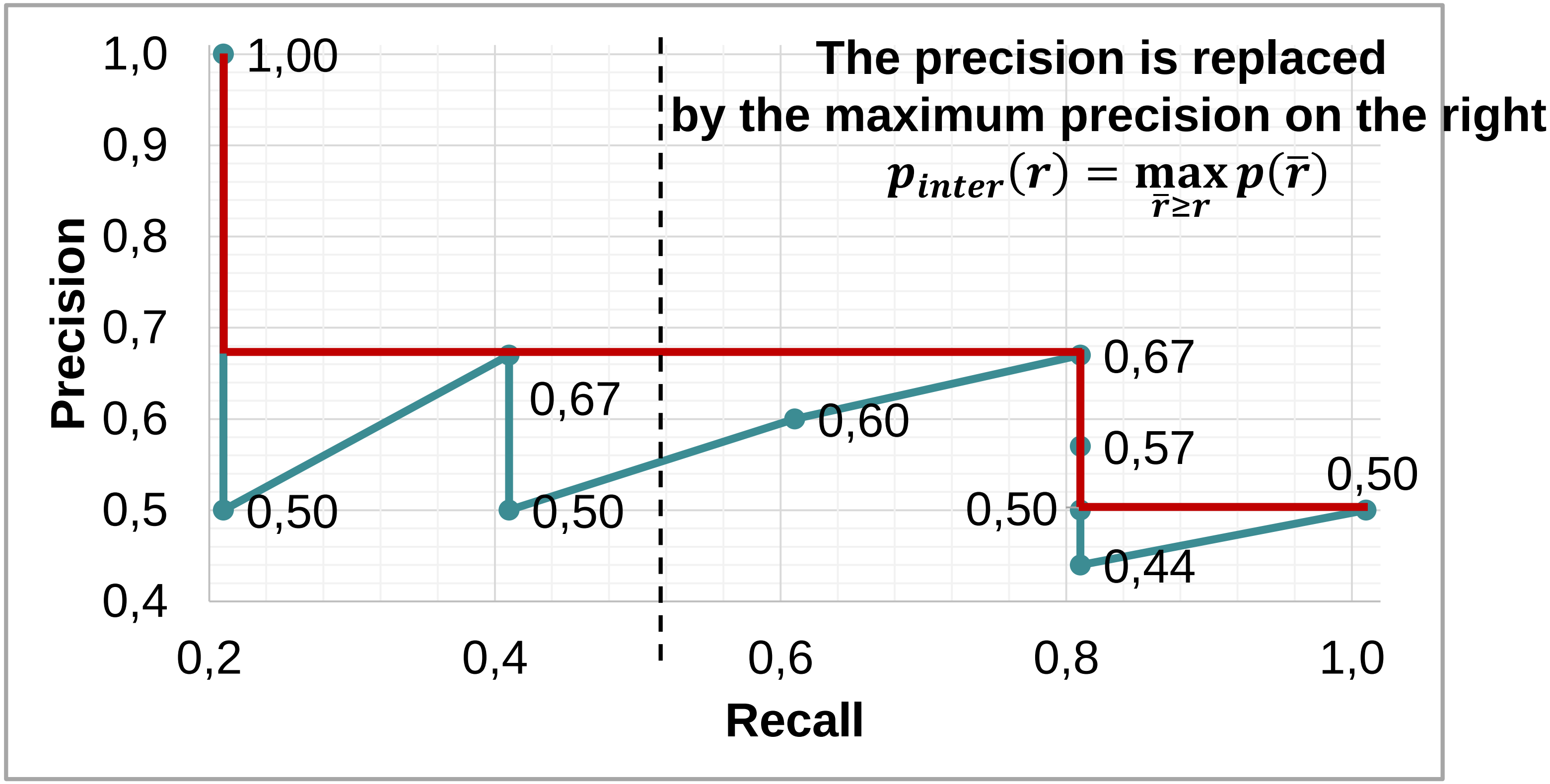
Object detection. Quality metrics (6)

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



Object detection. Quality metrics (7)

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

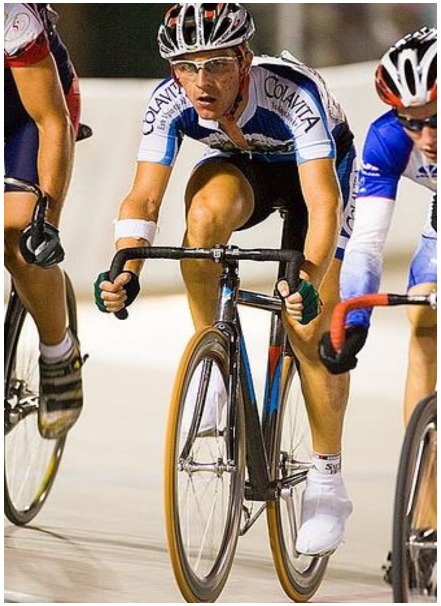


The precision is replaced by the maximum precision on the right

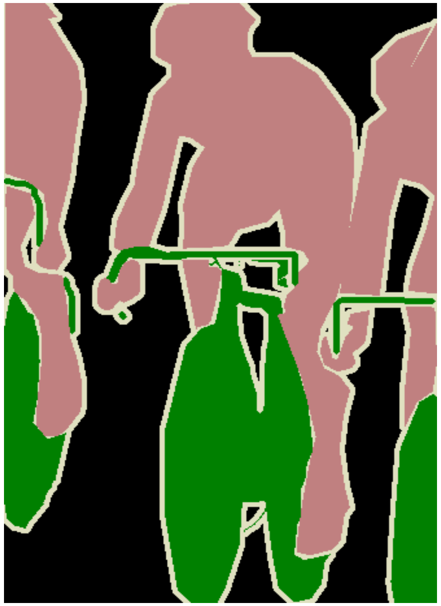
$$p_{inter}(r) = \max_{\bar{r} \geq r} p(\bar{r})$$

Semantic segmentation. Problem statement (1)

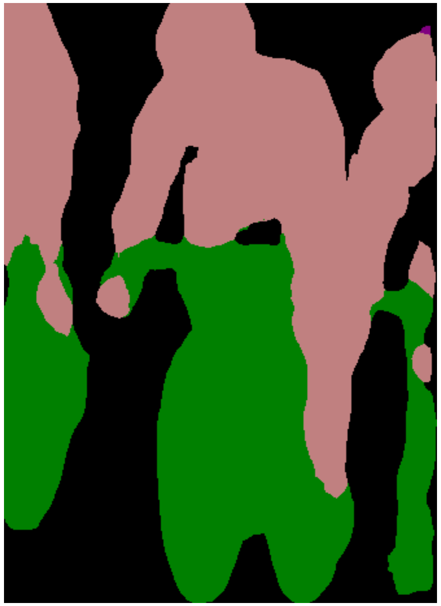
- The problem of semantic segmentation is to match each image pixel with the class of objects to which this pixel belongs (different colors correspond to the different classes)



Original image



Groundtruth



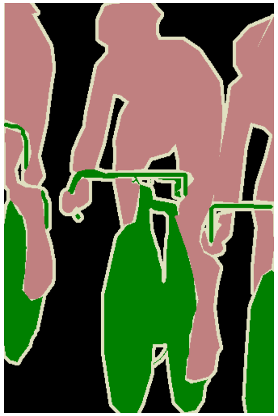
Segmentation result

* The PASCAL Visual Object Classes Homepage [http://host.robots.ox.ac.uk/pascal/VOC].

Semantic segmentation. Problem statement (2)

- The original image is represented by a set of pixel intensities $I = \left(I_{ij}^{k}\right)_{\substack{0\le i<w\\ 0\le j<h\\ 0\le k<3}}$, where w and h are image width and height, k is a number of color channels of the image
- The set of object classes $C = \{0, 1, \dots, N-1\}$ is defined, 0 corresponds to the background, the set of class identifiers uniquely corresponds to the set of class names
- It is required to find a mapping

$$\varphi\left(I_{ij}\right) = c$$



Semantic segmentation. Datasets

- **PASCAL VOC 2007, 2012** contain images and semantic segmentation groundtruth for objects of 20 classes [http://host.robots.ox.ac.uk/pascal/VOC]
- **MS COCO** contains images and semantic segmentation groundtruth for objects of 80 classes [http://cocodataset.org]
- **CamVid** is a dataset of on-road images containing annotation for 32 semantic classes [http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid]
- **Cityscapes** is a dataset of on-road images containing annotation for 30 semantic classes [https://www.cityscapes-dataset.com]

Semantic segmentation. Quality metrics

- ***Intersection over Union metric*** (***IoU)*** or Jaccard index

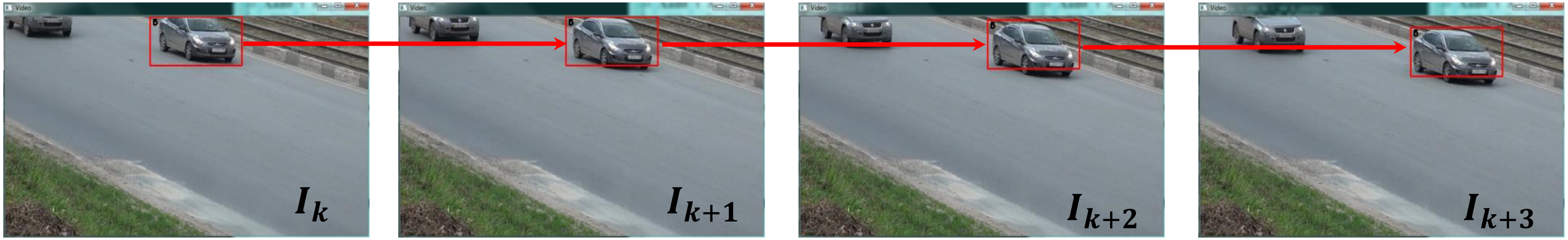
$$IoU = \frac{TP}{TP + FP + FN}$$

where TP is a number of correctly classified pixels (true positives), FP is a number of pixels that the method has been classified as belonging to the class, but they do not belong (false positives), FN is a number of pixels that belong to the class, but the method has been classified them as not belonging to the class (false negatives)

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

Object tracking. Problem statement (1)

- ❑ Object tracking involves detecting start location of the object (usually at the time the object first appears in the frame) and predict the location of this object in subsequent frames of the video



Object tracking. Problem statement (2)

- ❑ $I_0, I_1, \dots, I_{N-1}$ is a sequence of video frames, where N is a number of frames
- ❑ To track objects, it is required to create the mapping ψ of the set of locations B_k in the frame I_k to the set of locations B_{k+1} in the frame I_{k+1}:

$$\psi: B_k \to B_{k+1} \cup \{b\}, \qquad b = \big((-1,-1),(-1,-1)[,s,c]\big)$$

where b is a fake location used to indicate losing of an object by the tracker

- ❑ If I_k is a first frame on which the object was detected, $r_0(k)$ is a location identifier, q is a number of frames, then a ***track*** is a sequence of object locations

$$T_{r_0(k)}^{k} = \left(b_{r_0}^{k}, b_{r_1}^{k+1}, \dots, b_{r_{q-1}}^{k+q-1}\right), \qquad b_{r_i}^{k+i} = \psi\left(b_{r_{i-1}}^{k+i-1}\right), i = \overline{1, q-1}$$

Object tracking. Datasets

- **Multiple Object Tracking Benchmark (MOT Benchmark)** is a benchmark for multiple object tracking in video [https://motchallenge.net]
- **TrackingNet** is a dataset and benchmark, it consists of YouTube videos of various real-life objects [https://tracking-net.org]
- **Long-Term Visual Object Tracking Benchmark** is a benchmark consisting of long (more 400 minutes) videos with single objects [https://amoudgl.github.io/tlp]

Object tracking. Accuracy metrics (1)

❑ Criteria for choosing quality metrics of object tracking:

- Quality metric has to confirm the quality of location prediction on each frame of the video containing the tracked object
- Quality metric has to confirm the quality of tracking throughout the sequence of all frames containing the object
- For each tracked object, the track has to be a single one
- Quality metric has to ensure comparability of indicators for different types of tracking algorithms (2D, 3D trackers, centroid trackers, area trackers, etc.)

Object tracking. Accuracy metrics (2)

- ***Multiple Object Tracking Accuracy**** (MOTA)
- One of the well-known and common indicators that is used to compare tracking algorithms in different contests
- Notation:
  - t is an identifier of the current frame in the video
  - $\{o_1, o_2, \dots, o_n\}$ is a set of observed objects in the frame t (groundtruth)
  - $\{h_1, h_2, \dots, h_m\}$ is a set of object locations predicted by the tracking algorithm in the frame t

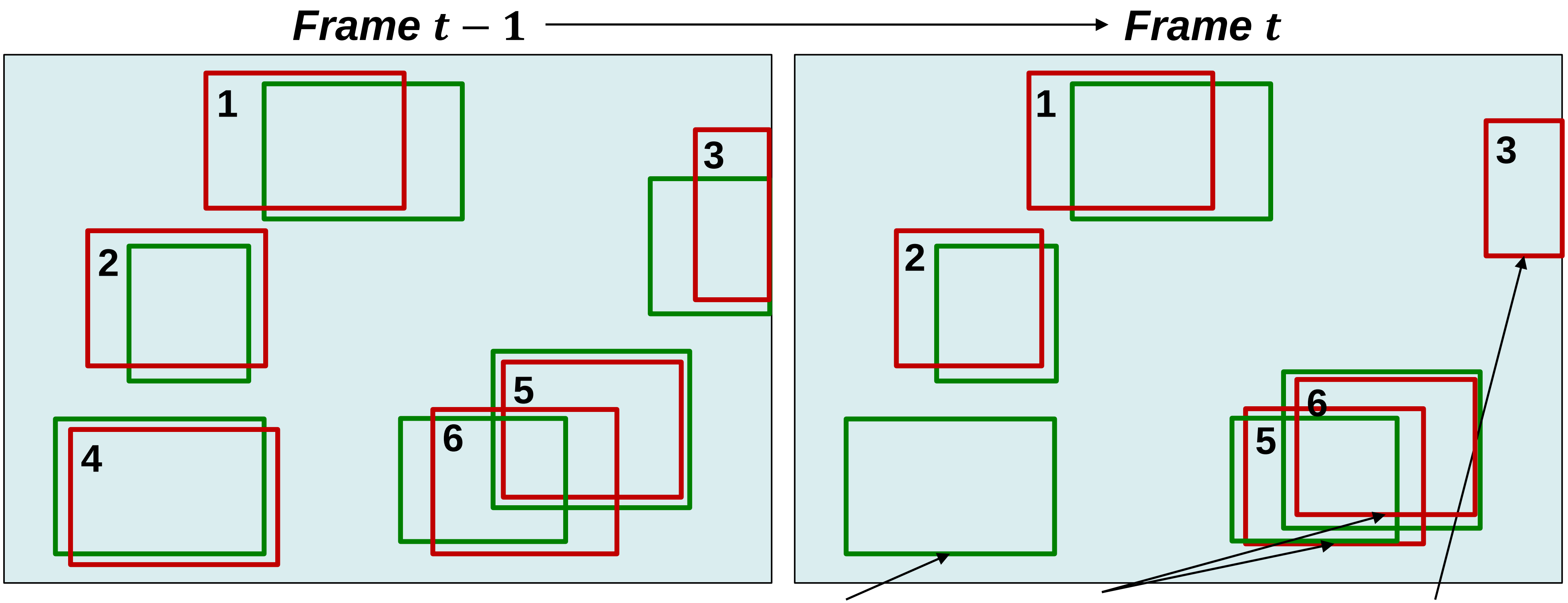
* Bernardin K., Stiefelhagen R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics // Image and Video Processing. – 2008.

Object tracking. Accuracy metrics (3)

- ***Multiple Object Tracking Accuracy**** (MOTA)
- Metric calculation for each frame t:
  1. Calculating an error of object location prediction and searching for the best matching between the constructed locations $\{h_1, h_2, \ldots, h_m\}$ and groundtruth $\{o_1, o_2, \ldots, o_n\}$
  2. Accumulating the following errors:
     1. Counting ***misses***. Misses are annotated objects for which there is no matched predicted locations
     2. Counting ***false positives***. False positives are predicted locations for which there is no real objects in the frame
     3. Counting ***mismatch errors***. Mismatch errors occur when object track is reinitialized with a new track identifier

* Bernardin K., Stiefelhagen R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics // Image and Video Processing. – 2008.

Object tracking. Accuracy metrics (4)



🟩 – groundtruth

🟥 – predicted location

m **– miss** (object is lost)

mme **– mismatch error** (track identifier was changed)

fp **– false positive** (real object disappeared from the frame)

Object tracking. Accuracy metrics (4)

- ***Multiple Object Tracking Accuracy**** (MOTA)

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}$$

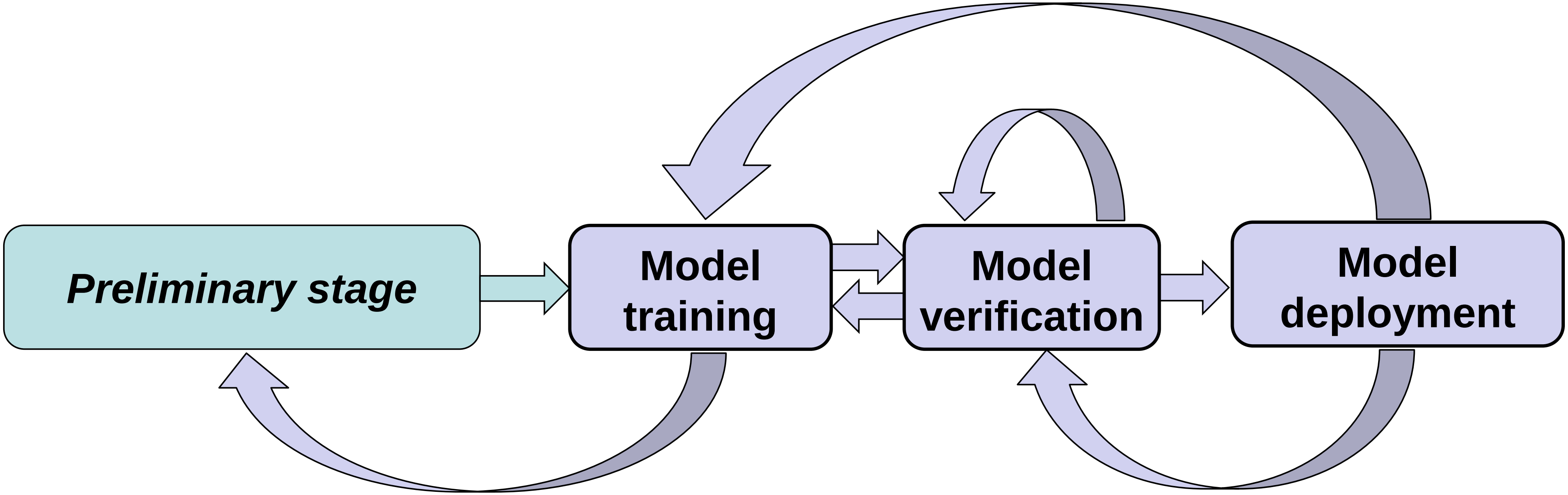
where m_t is a number of misses on the frame t,
fp_t is a number of false positives on the frame t,
mme_t is a number of mismatches on the frame t,
g_t is a number of real objects on the frame t

- ***Note:*** there are various approaches for matching predicted and annotated locations (for example, the Hungarian algorithm for solving the assignment problem based on a similarity matrix between annotated and predicted locations)

* Bernardin K., Stiefelhagen R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics // Image and Video Processing. – 2008.

GENERAL SCHEME OF SOLVING COMPUTER VISION PROBLEMS USING DEEP LEARNING

General scheme of solving computer vision problems using deep learning



Literature review

Preparing dataset

Preparing infrastructure

Model development

Model training

Model verification

Measuring quality

Model deployment

Model inference

Model optimization

Preliminary stage

- ❑ ***Literature review***
  - – Existing models (to use transfer learning)
  - – Quality metrics
  - – Public datasets
- ❑ ***Preparing datasets***
  - – Searching for similar data on the Internet
  - – Collecting and annotating your own dataset
  - – Preprocessing and annotating data
- ❑ ***Preparing infrastructure***
  - – Searching for existing frameworks or developing your own tools for measuring quality

Model training and verification

- ***Model development*** is development of custom layers and loss function
- ***Model training***
- ***Model verification*** means measuring of model quality in accordance with the selected metrics

- ***Note:*** a training framework is used training and verifying models

Model deployment

- ❑ Porting model to the framework that will be used for inference
- ❑ Analyzing model complexity and performance on the target device
- ❑ ***Model optimization and compression***
- ❑ Retraining model and its verification
- ❑ ***Repetitive model inference*** on the target device

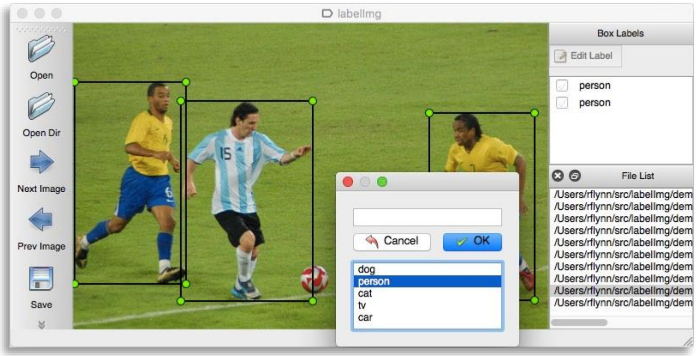
SOFTWARE USED AT DIFFERENT STAGES OF SOLVING COMPUTER VISION PROBLEMS

Literature review tools

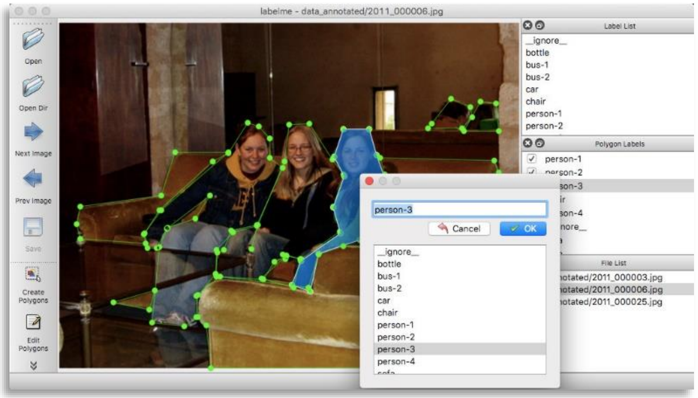
- ❑ Springer Journals [https://www.springer.com]
- ❑ Paper aggregator arXiv [https://arxiv.org]
- ❑ Proceedings of well-known conferences:
  - – IEEE Conference in Computer Vision and Pattern Recognition (CVPR)
  - – International Conference on Computer Vision (ICCV)
  - – European Conference on Computer Vision (ECCV)
  - – International Conference on Machine Learning (ICML)
  - – Asian Conference on Computer Vision (ACCV)
  - – Conference and Workshop on Neural Information Processing Systems (NeurIPS or NIPS)
  - – British Machine Vision Conference (BMVC)
  - – …

Free software for image annotation (1)

- **LabelImg** [https://github.com/tzutalin/labelImg]
  - Annotating of bounding boxes with class labelling
  - Designed in Python using Qt framework

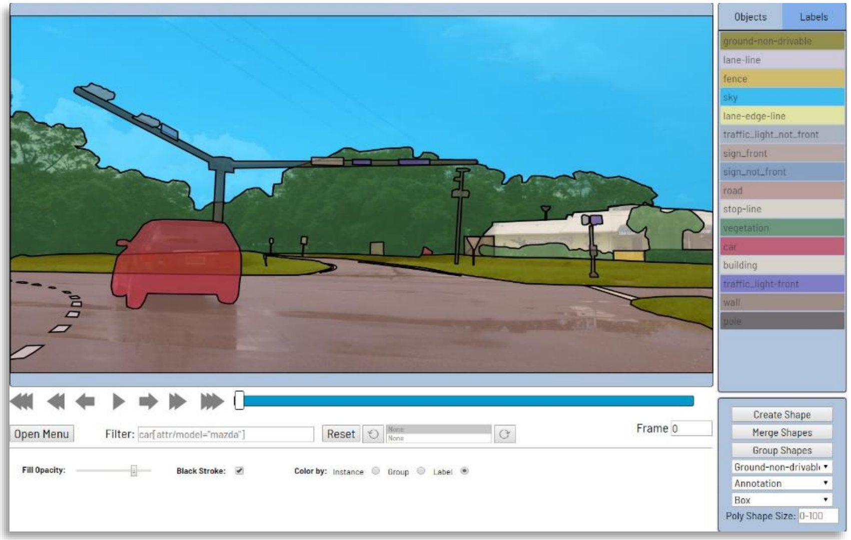

labelImg
Open
Open Dir
Next Image
Prev Image
Save
Box Labels
Edit Label
person
person
File List
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
Cancel
OK
dog
person
cat
tv
car

- **LabelMe** [https://github.com/wkentaro/labelme]
  - Annotating of polygons (rectangles, circles, points, lines, polygons) with class labelling
  - Annotating of video
  - Designed in Python using Qt framework



Free software for image annotation (2)

- **CVAT** [https://github.com/opencv/cvat]
  - Free interactive tool for annotating of images and video
  - It supports data annotating for object detection, semantic segmentation, and object tracking problems
  - CVAT supports various annotation formats (PASCAL VOC, MS COCO Object Detection, MOT, LabelMe, MOT, etc.)



Filter: car[attr/model="mazda"]

Frame 0

Fill Opacity:

Black Stroke: ☑

Color by: Instance ○ Group ○ Label ●

Poly Shape Size: 0-100

* A curated list of awesome dataset tools [https://github.com/jsbroks/awesome-dataset-tools].

Deep learning frameworks for training and verification (1)

- ❑ **Caffe**
  - – Various implementations: Caffe [https://caffe.berkeleyvision.org], Intel Optimization for Caffe [https://software.intel.com/en-us/frameworks/caffe]
  - – Interfaces: C++, Python
  - – Is contains functions required for the development of fully connected, convolutional and recurrent networks
- ❑ **TensorFlow**
  - – Official page [https://www.tensorflow.org]
  - – Interfaces: Python
  - – It supports distributed training

Deep learning frameworks for training and verification (2)

- ❑ **Keras**
  - – Official page [https://keras.io]
  - – High-level Python API for TensorFlow, CNTK, Theano backends
- ❑ **PyTorch**
  - – Official page [https://pytorch.org]
  - – Interfaces: Python, C++
  - – It contains a large set of tools for quickly solving deep learning problems from prototyping to deployment
  - – PyTorch supports distributed training

Deep learning frameworks for training and verification (3)

- **MXNet**
  - Official page [https://mxnet.apache.org]
  - Interfaces: Python, Scala, Julia, Clojure, Java, C++, R, Perl
  - It supports distributed training
  - MXNet supports effective inference of pretrained models on low performance devices (mobile devices, IoT devices)
- **Other less used frameworks** (Theano, DeepMat, Darch, etc.)

- ***Note:*** these frameworks can be used both for training and for inference

Deep learning frameworks for inference (1)

- ❑ **OpenCV**
  - Official page [https://opencv.org]
  - OpenCV is a well-known computer vision library
  - It contains DNN module for deep learning inference
- ❑ **Intel Distribution of OpenVINO Toolkit***
  - Official page [https://software.intel.com/en-us/openvino-toolkit]
  - Toolkit contains a set of tools for developing computer vision applications using deep learning
  - Is contains the Inference Engine module for efficient inference of deep models on Intel hardware
  - Inference Engine can be used as a backend for DNN module of OpenCV library

* Available features of the Inference Engine will be discussed in detail in the following lectures and practices.

Deep learning frameworks for inference (2)

- **TensorRT**
  - Official page [https://developer.nvidia.com/tensorrt]
  - SDK for inference on NVIDIA graphics
  - It is based on CUDA
  - TensorRT includes an inference optimizer and runtime environment that provide low latency and high throughput
  - It allows to convert trained models from the format of the frameworks Caffe 2, Chainer, Microsoft Cognitive Toolkit, MXNet, PyTorch into ONNX format
  - TensorRT supports INT8 and FP16 optimizations
  - After applying optimizations, it selects kernels for specific platforms

Conclusion

- Further in the course, modern deep neural networks are considered for solving the classical problems of computer vision stated in this lecture
- In conclusion of the course lectures, deep models that are used to generate synthetic data are discussed; generated data can be used in various stages of solving problems
- Carrying out practice, it is supposed to solve tasks using pretrained deep models supported by the Intel Distribution of OpenVINO Toolkit

Literature

- Haykin S. Neural Networks: A Comprehensive Foundation. – Prentice Hall PTR Upper Saddle River, NJ, USA. – 1998.
- Osovsky S. Neural networks for information processing. – 2002.
- Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press. – 2016. – [http://www.deeplearningbook.org].
- Chollet F. Deep Learning with Python. – Manning Publications Co. – 2018.

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