Create an object detection model for your application

Jacques-Sylvain Lecointre

mnubo
An Aspen Technology Company

ABOUT ME

- Customer facing Data Scientist
- Primary focus on IoT and time series data
- Interest in Computer vision and Deep Learning
- Marathon Runner during my spare time

Motivation : Share my experience and exchange on computer vision

01 **BRIEF HISTORY**
History of object detection

02 **PROBLEM DEFINITON**
Be able to define the task

03 **CV TECHNIQUES**
What's in my toolbox?

04 **DATASETS & TOOLS**
Leverage existing datasets

ConFoo.CA
DEVELOPER CONFERENCE
2020

05 **STATE-OF-THE ART MODELS**
CNN based models , Region proposals

06 **EXAMPLE OF YOLOv3**
How to use it and how it works ?

07 **MODEL TRAINING**
Key elements to take into consideration

08 **EXPORT ON THE EDGE**
Example of ONNX

01

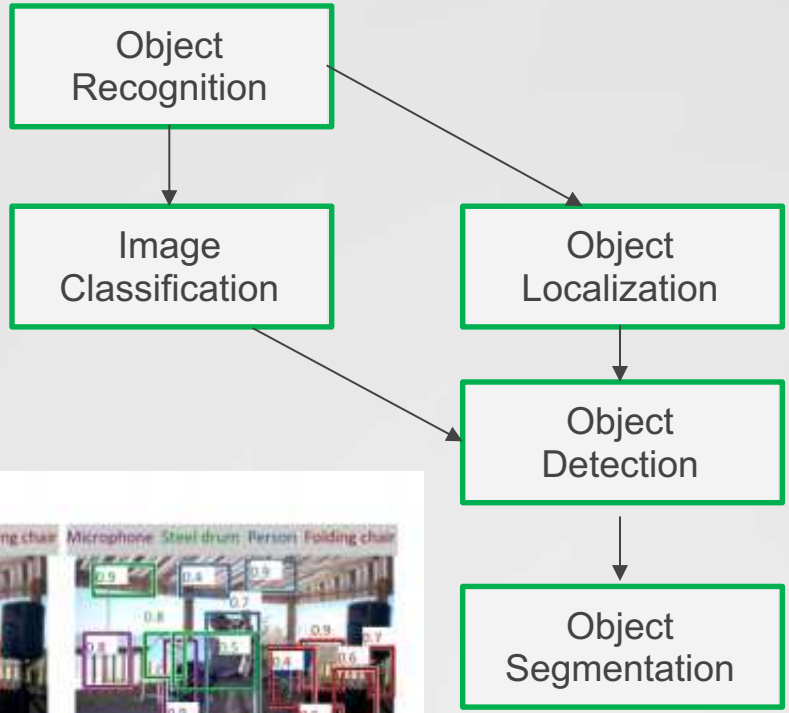
DEFINITION AND BRIEF HISTORY

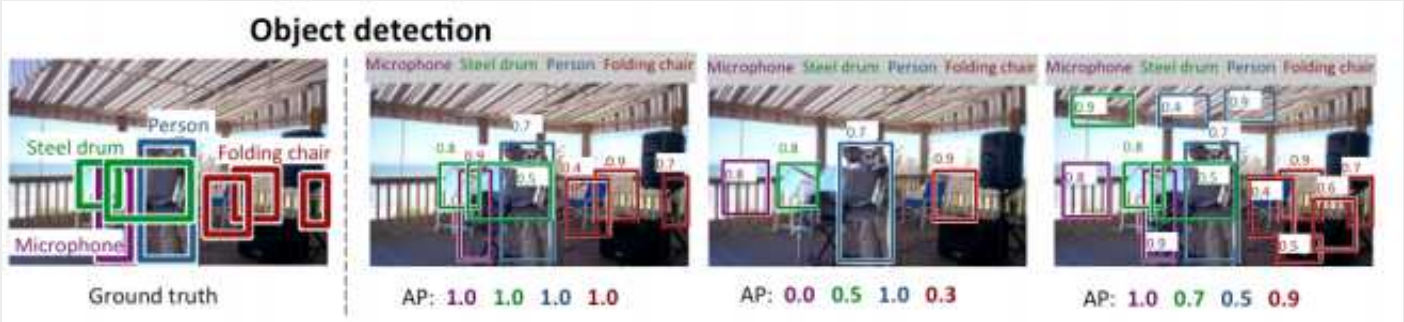
The path towards object detection

WHAT IS OBJECT DETECTION ?

Computer vision problem which deals with **Identifying** and **Locating** object of certain classes in the image

COMPUTER VISION TASKS



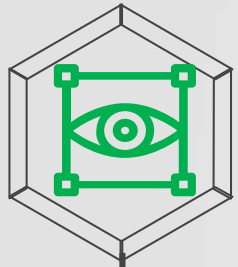


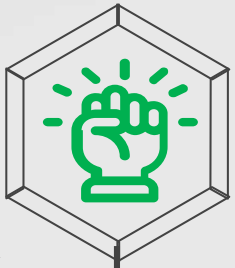
BRIEF HISTORY OF OBJECT DETECTION

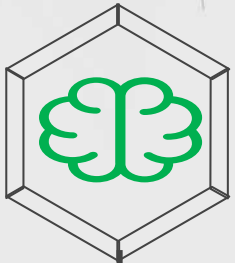
1960's - 1990's FOUNDATIONS OF OBJECT DETECTION

1990's - 2010's HAND CRAFTED FEATURES

2010's to today DEEP LEARNING







1972 HOUGH TRANSFORM

1977 MORAVEC

1986 CANNY EDGE DETECTOR

1988 HARRIS CORNER DETECTOR

2001 VIOLA JONES DETECTOR

2004 SIFT

2005 HOG FEATURES

2008 DEFORMABLE HOG

2010 ILSVRC

2012 AlexNet

2014 VGG & GoogleLeNet

2015 Faster R-CNN

2016 YOLO

02

PROBLEM DEFINITION

Break down the task to smaller steps

USE CASE EXAMPLE : ELIMINATE REDELIVERY

Notifications and proof of delivery received in the app

Okippa folded

Package not delivered

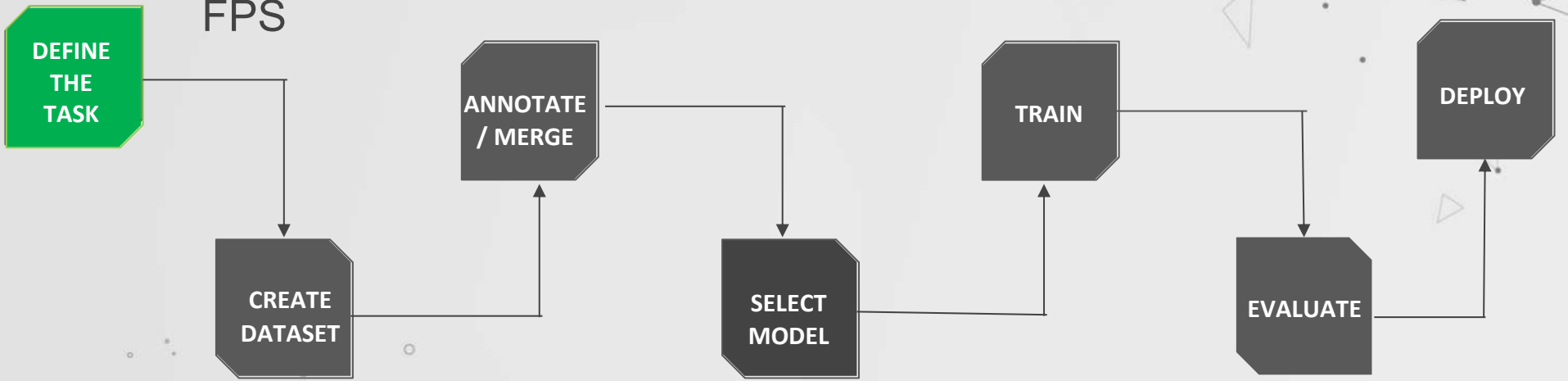
- Notify customers that a package has been delivered with a ***proof***
- Notify the shipping company of ***pickup*** event
- Detect ***stolen*** events

Okippa unfolded

Package delivered

WORKFLOW

A priori knowledge on the image and desired accuracy and FPS



ALWAYS KEEP IN MIND THE CONTEXT

A PRIORI KNOWLEDGE OF THE IMAGE ?

Lighting invariant ?

Orientation invariant ?

Scale invariant ?

Assumption about the size?

Develop a detection that will match the use case

WHAT ARE THE APPLICATION CONSTRAINTS?

Real time ?

Sensitive to False positive ?

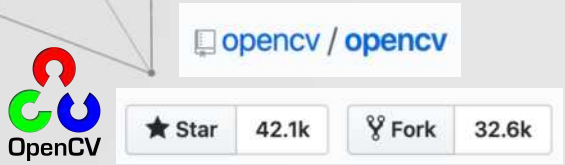
Break the detection down to smaller steps

03

COMPUTER VISION TECHNIQUES

What’s in my toolbox ?

COMPUTER VISION LIBRARIES



- Strong focus on real-time apps
- Optimized : primary interface in C/C++ has python and Java
- Under BSD licences except for non-free algorithms
- More than 2500 optimized algorithms

- Collection of algorithms for image processing
- Python based ,designed to interoperate with Numpy
- Pedagogical example-based doc :

HYPOTHESIS AND PRIOR KNOWLEDGE

Lighting invariant

Thresholding +
Edge detection

Orientation invariant

Scale invariant

Template matching

Orientation invariant

Viola Jones

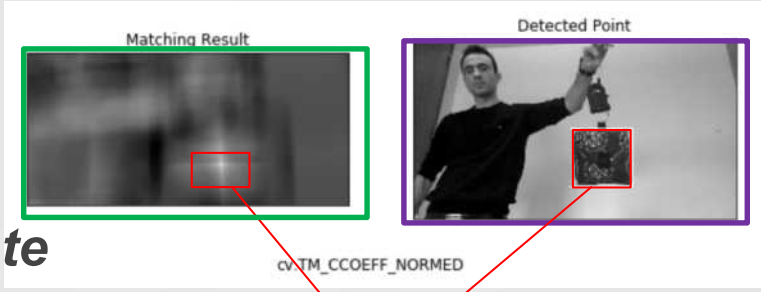
No prior Knowledge

SIFT
Deep Learning

DETECTION WITH CONVOLUTION

TEMPLATE MATCHING

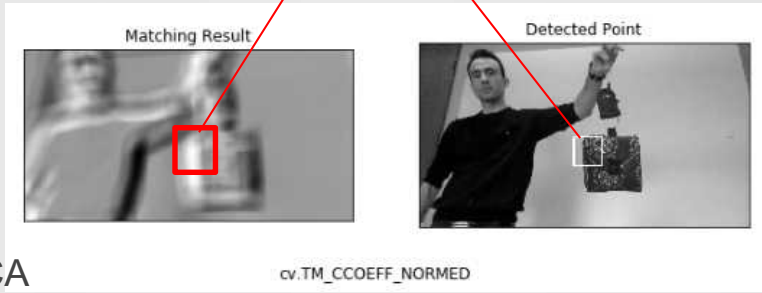
$R(x) = I(x)*F(x)$



Argmax {R(x)}

- Implemented in Open CV with ***matchTemplate***
- Works well to detect a specific template
- Problem if the object varies in :
  shape , **scale** , **intensity**
- Solution combine several filters e.g obtained by PCA

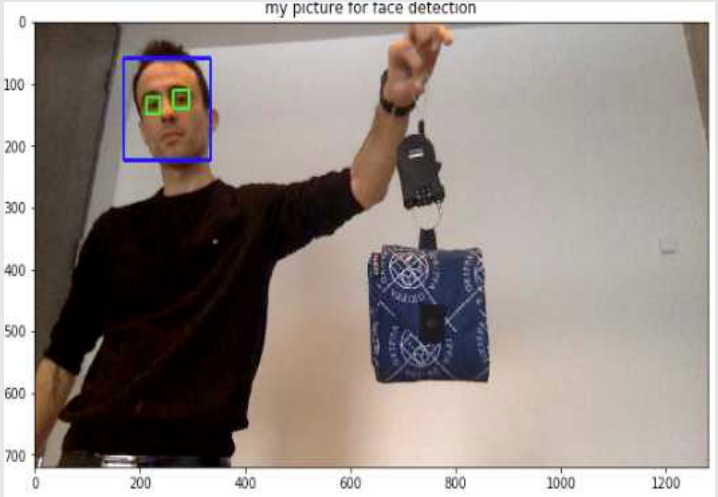
F(x) =



VIOLA AND JONES DETECTOR

- Widely used method for Real-time object detection

haarcascade_frontalface



haarcascade_eye

- Machine learning based approach

- **Trainer** : opencv_haartraining

OpenCV

- **Detectors** for face,eyes,smile.. : CascadeClassifier

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features.CVPR (1), 1(511-518)

https://www.cs.ubc.ca/~lowe/425/slides/13-ViolaJones.pdf

SIFT DESCRIPTORS

Scale-Invariant Feature Transform

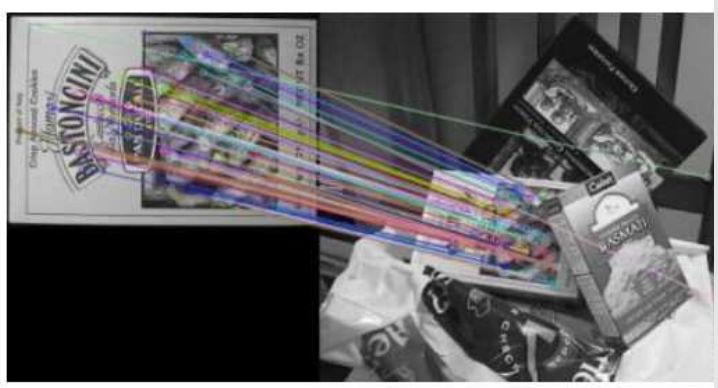


IMAGE MATCHING

- Planogram compliance
- Google Image Search
- Implemented in Open CV (non-free algorithms)
- Still under patent **BUT** expires in one week !

https://patents.google.com/patent/US6711293B1/en

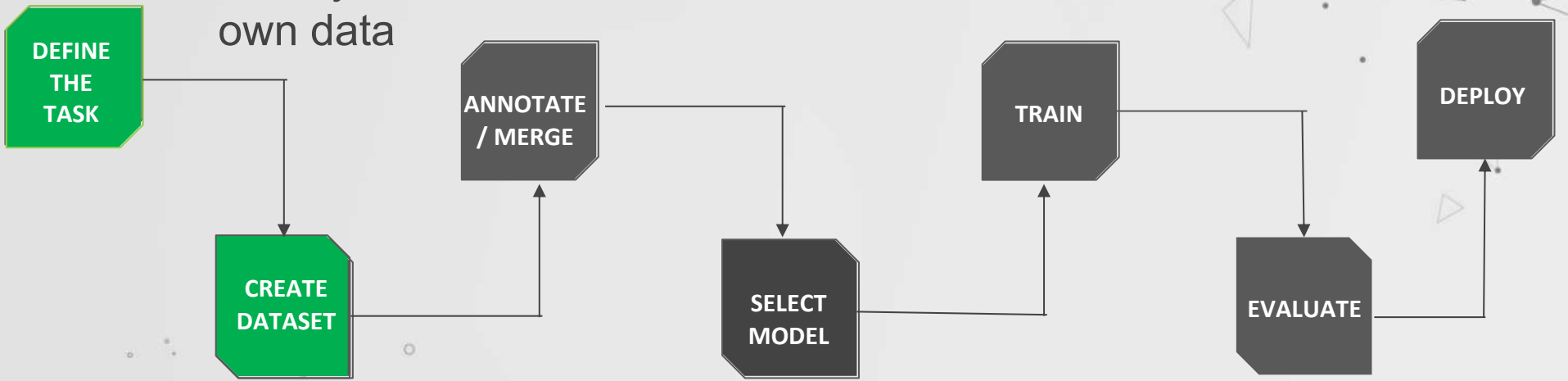
04

DATASETS & TOOLS

Don’t reinvent the wheel !

WORKFLOW

Leverage existing datasets and create your own data



LEVERAGE EXISTING DATASETS

COCO Detection Leaderboard

COCO

- large-scale object detection, segmentation, and captioning
- Extensive API support images, annotations, and evaluation code
- Recognition in context
- ~ 200k labeled images

PASCAL VOC

Pattern Analysis, Statistical Modelling and Computational Learning

VOC Evaluation server

- standardized image datasets for object class recognition
- Common set of tools for accessing the datasets and annotations
- ~11 500 images

*** COCO is no longer featuring the bounding-box detection task**

DATASETS INTEGRATED

pycocotools 2.0.0

```
pip install pycocotools
```

TensorFlow

```
tfds.object_detection.coco.Coco
```

- Object_detection
  - coco
  - kitti
  - open_images_v4
  - voc
  - wider_face

PyTorch

```
torchvision.datasets.CocoDetection
```

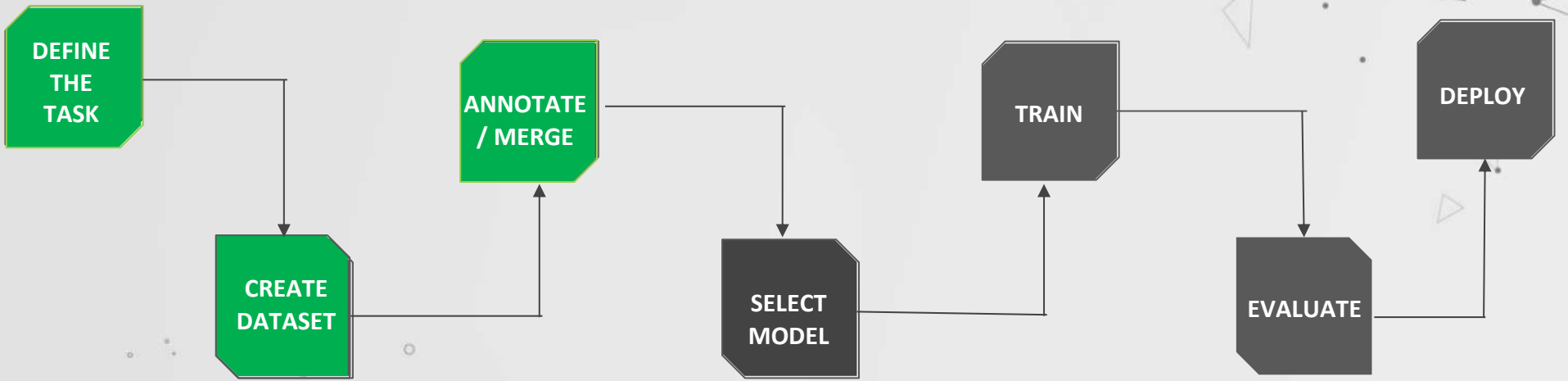
DATASET INTEGRATED : CODE EXAMPLE

```
import torch
from torch.utils import data
from torchvision.datasets import CocoDetection
from torchvision import transforms

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

training_generator = CocoDetection('./images/train2017',
                                   './instances_merged_custom_train2017.json',
                                   transform=transform)

    # Loop over epochs
    for epoch in range(max_epochs):
        # Training
        for local_batch, local_labels in training_generator:
            bboxes = [x['bbox'] for x in local_labels]
            local_batch = local_batch .to(device)
```

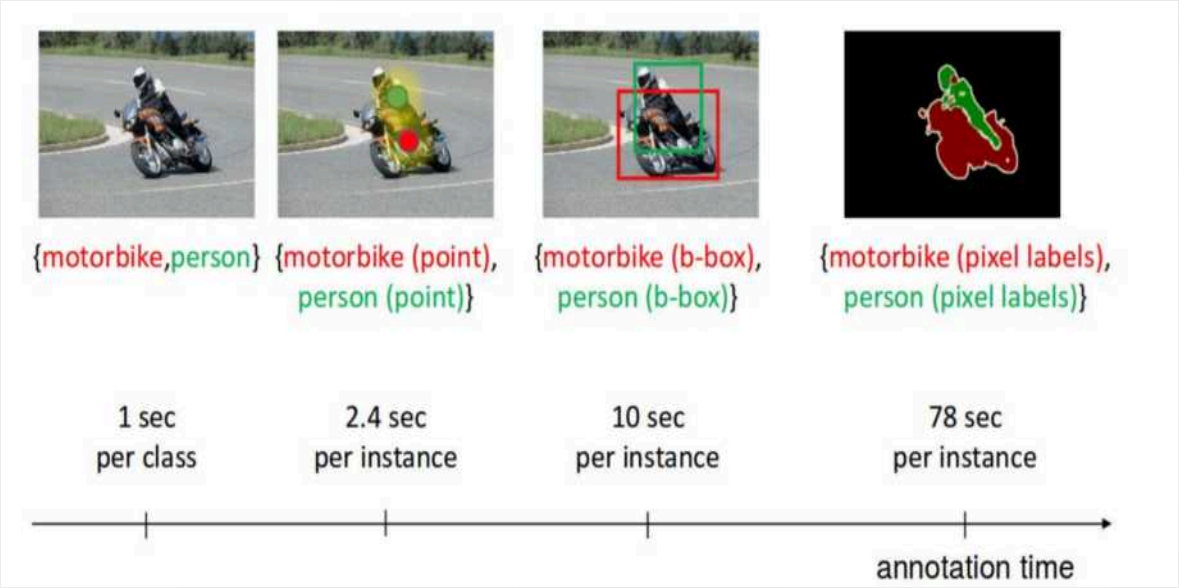
PyTorch

WORKFLOW

Decide the annotation type that best matches your use case and leverage existing datasets



WHAT'S THE POINT ?



ANNOTATION HAS A COST

- Select carefully the type of annotation for your application
- Trade-off between test time accuracy and training-time annotation cost

Berman et al. (2015) , What's the Point: Semantic Segmentation with Point Supervision, ECCV 16

CREATE YOUT OWN DATASET : ANNOTATION

- **BE COMPLIANT TO AN EXISTING DATASET FORMAT :**
  - Annotation Format

```
{
    "info": {...},
    "licenses": [...],
    "images": [...],
    "annotations": [...],
    "categories": [...]
}
```

  - Results Format

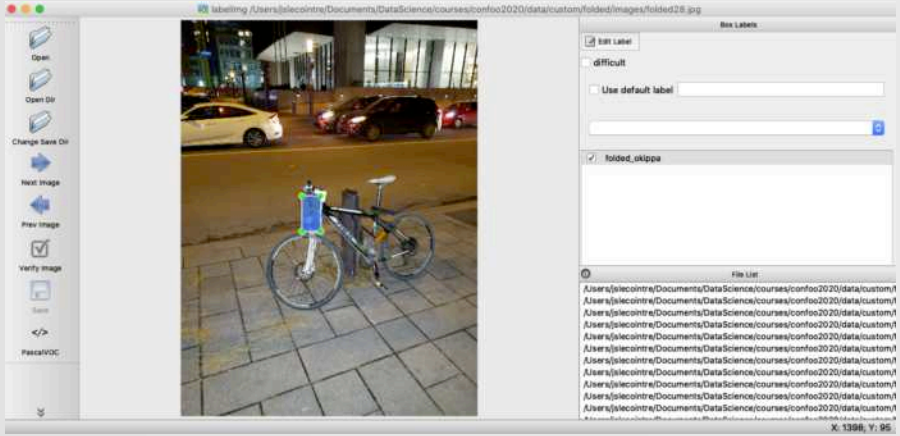
```
[{
        "image_id": 42,
        "category_id": 18,
         "bbox": [258.15, 41.29, 348.26, 243.78],
          "score": 0.236},
 {…},{…},{…},{…}
]
```

- **THERFORE YOU CAN USE THE EXISTING API :** COCO Detection API demo

ANNOTATION TOOLS

labelImg

- Simple to use
- Only bounding boxes
- VOC and YOLO formats

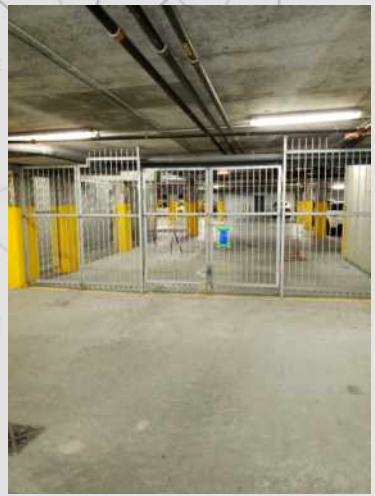


BBox-Label-Tool … and too many more

Rectlabel on Mac OS X

- Detection and Segmentation
- Paid version with advanced options
- Open and label from S3 , resize
- Supports VOC ,COCO,YOLO and format conversion
- Loading a ML models (Core ML) to accelerate annotations
- Box creation with 2 points

EXAMPLE OF ANNOTATED IMAGES

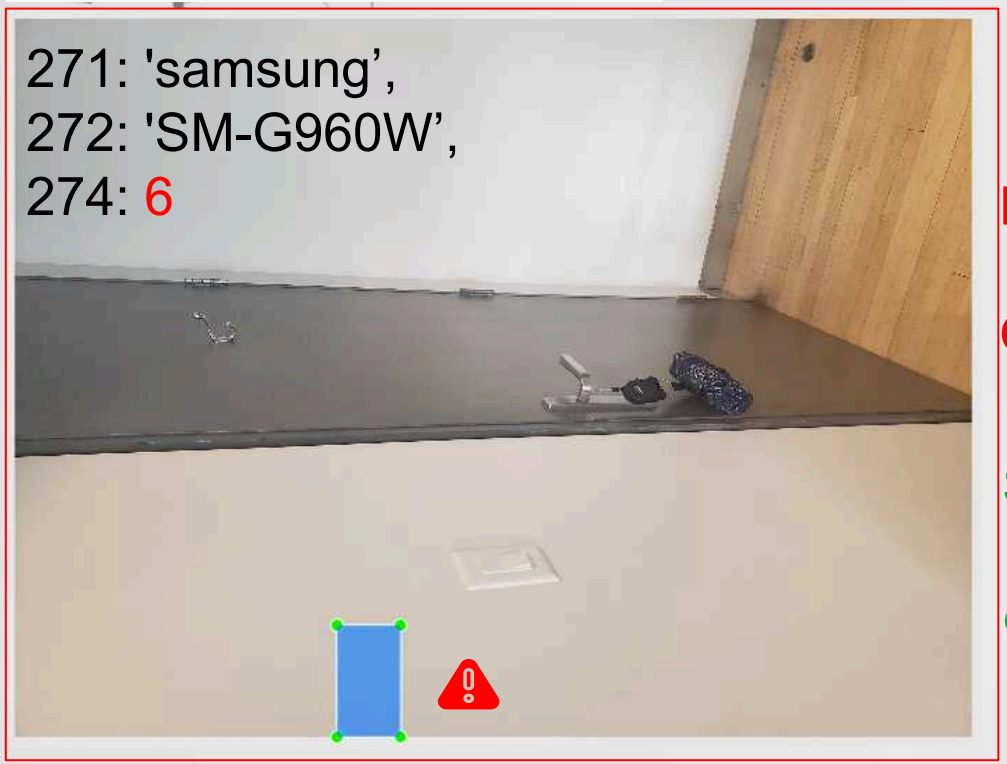







GOOD TO KNOW

```
import PIL.Image
PIL.Image.open('./folded110.jpg')._getexif()
```



Problem :

Orientation in Exif-formatted metadata

Solution :

*exiftran -ai *.jpg*

exiftran can do lossless rotations and cares about the EXIF data

The Dumb Reason your fancy computer vision app isn't working exif orientation

https://www.exiv2.org/tags.html

COCO SYNTHETIC GENERATION

Prepare backgrounds

Prepare foregrounds
(.png files) using GIMP

cocosynth example

- Save hundreds of hours of annotation time
- *coco_instances.json* files
- images & annotations

DEMO : COCO_Synth_Python_API

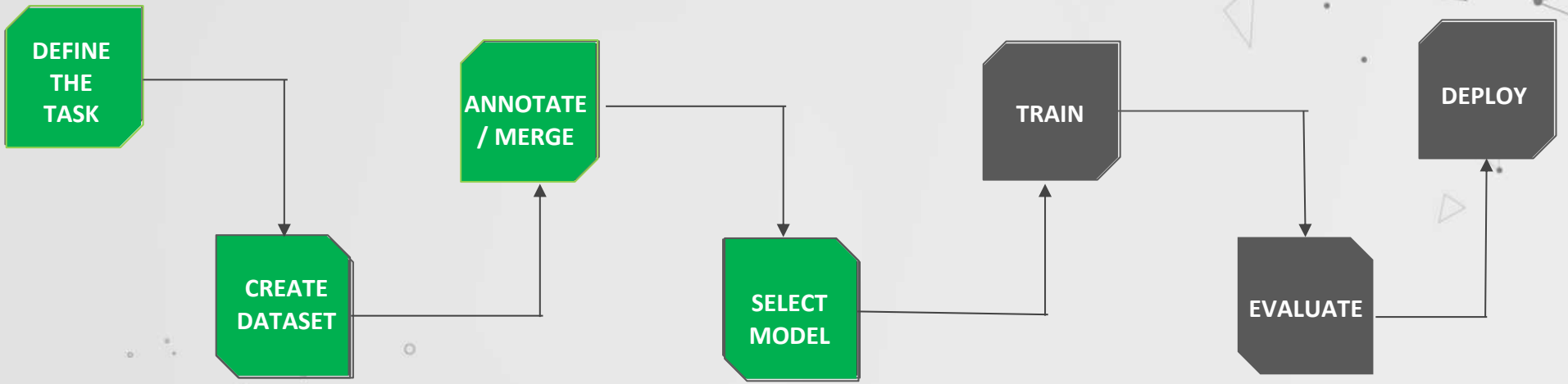
https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch/#create-custom-coco-dataset

05

DEEP LEARNING OBJECT DETECTION MODELS

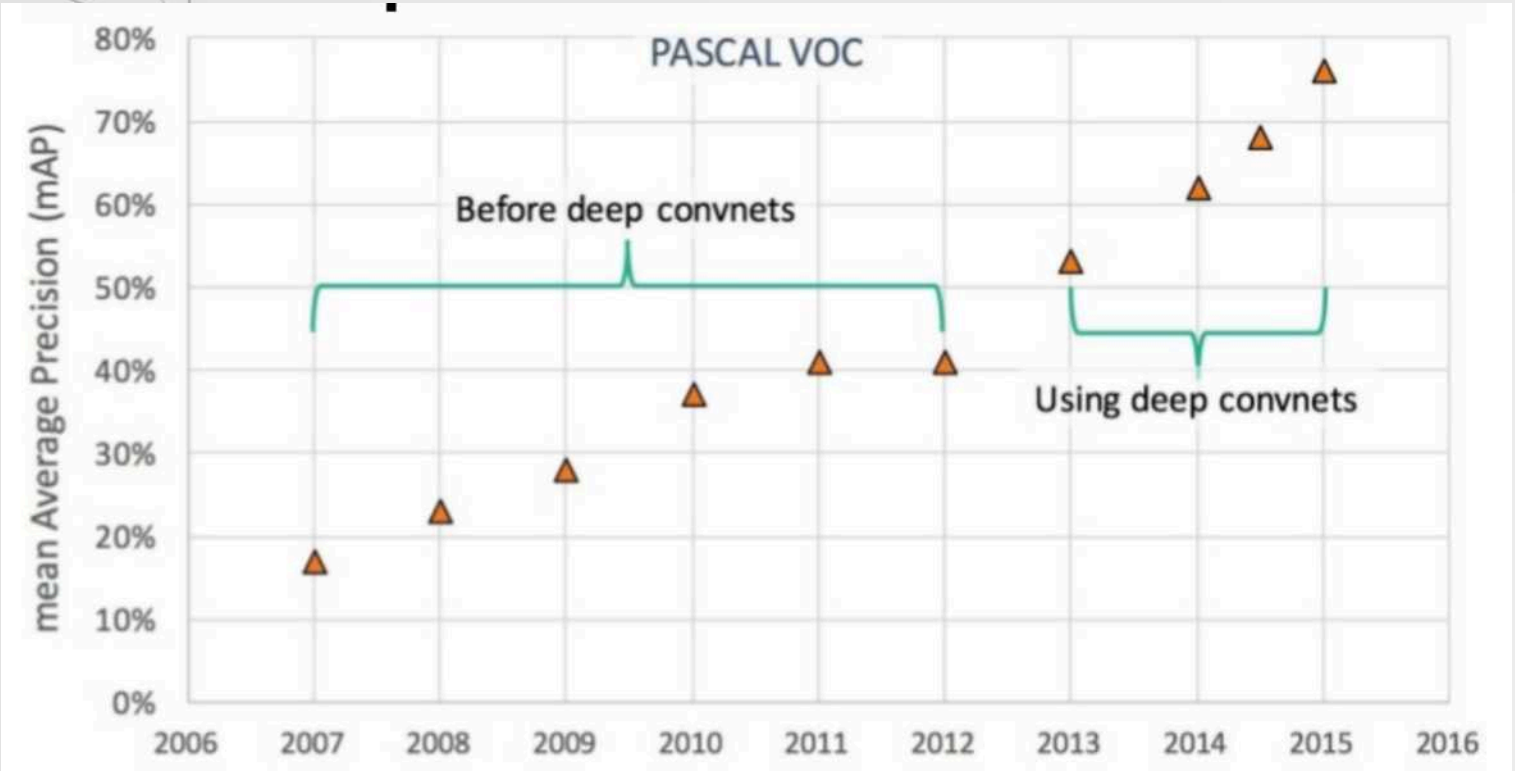
Which detection framework should I choose ?

WORKFLOW



Select the most appropriate model framework and adjust the parameters to your dataset

OBJECT DETECTION MODEL SELECTION



"Object detection Tutorial", Girshick. ICCV'15

OBJECT DETECTION MODEL SELECTION

Region Proposals

- State-of-the-art in accuracy
- Generate high quality proposals With region proposal algorithms.
- Multi-task training E = Ecls + Eloc
- Faster R-CNN

Single-Shot

- Speed (45 frames/second)
- Object detection problem reframed as a single regression problem
- Sees the entire image
- State-of-the-art in real-time
- SSD and YOLO

06

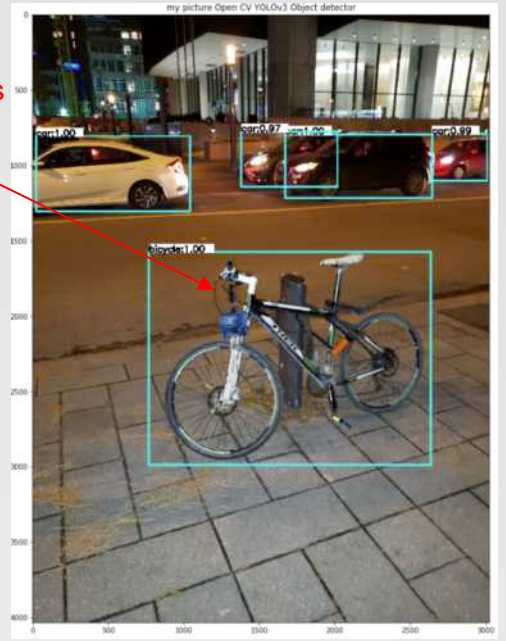
DETECTION WITH YOLO v3

You Only Look Once

YOLO v3 INFERENCE

CVPR 2016
OpenCV People's Choice Award

- State-of-the art object detector for real-time
- YOLOv3 in OpenCV 3.4.2 (for inference) very fast CPU implementation
- Original paper from Redmon et al. 2016 with incremental improvements (3 versions)
- Using **Darknet** framework (C++) originally but many implementations : link to config files

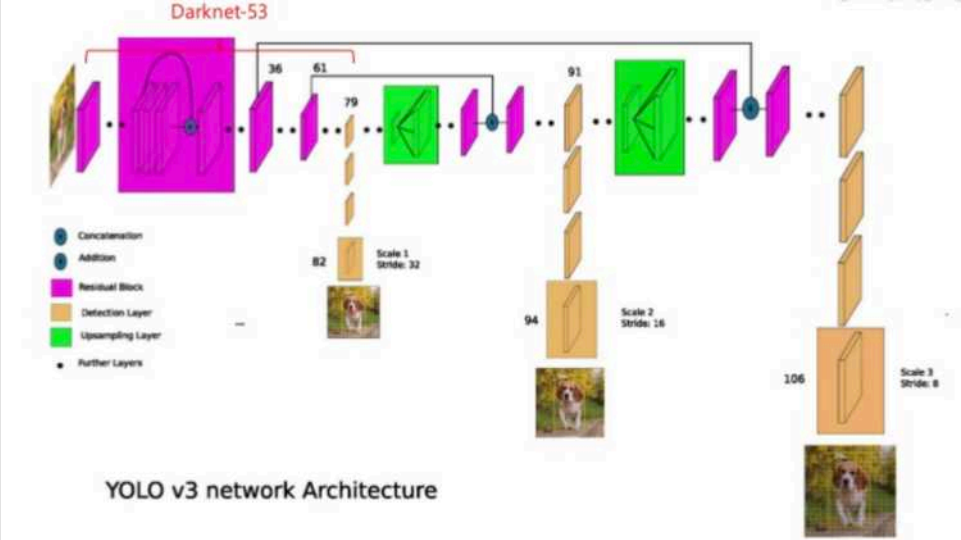


Detection with OpenCV using pre-trained model

TensorFlow PyTorch Keras Caffe OpenCV

YOLO : IN A NUTSHELL

01 CNN FEATURE EXTRACTOR

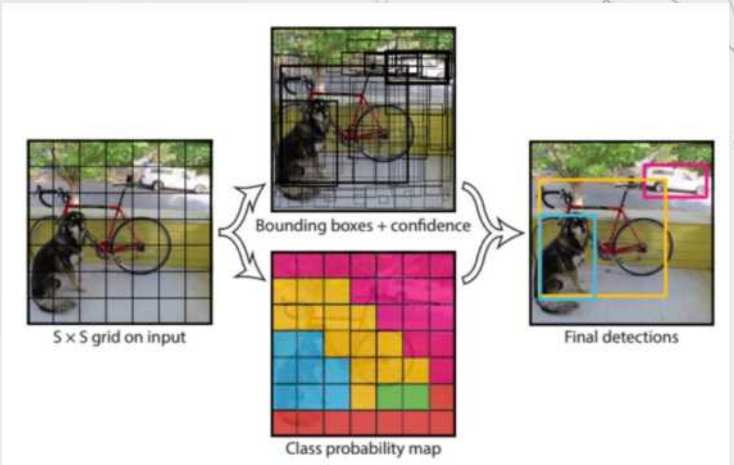


YOLO v3 network Architecture

02 UNIFIED DETECTION

x,y,w,h confidence class prob

S x S x (3*(4 + 1 + C))



Pr(Class i) * IoU

=

Pr(Class i|Object) * Pr(Object) * IoU

confidence

Jospeh Redmon , Santosh Divvala,Ross Girshick, Ali Farhadi (2016) You Only Look Once: Unified, Real-Time Object Detection

https://pjreddie.com/darknet/yolo/

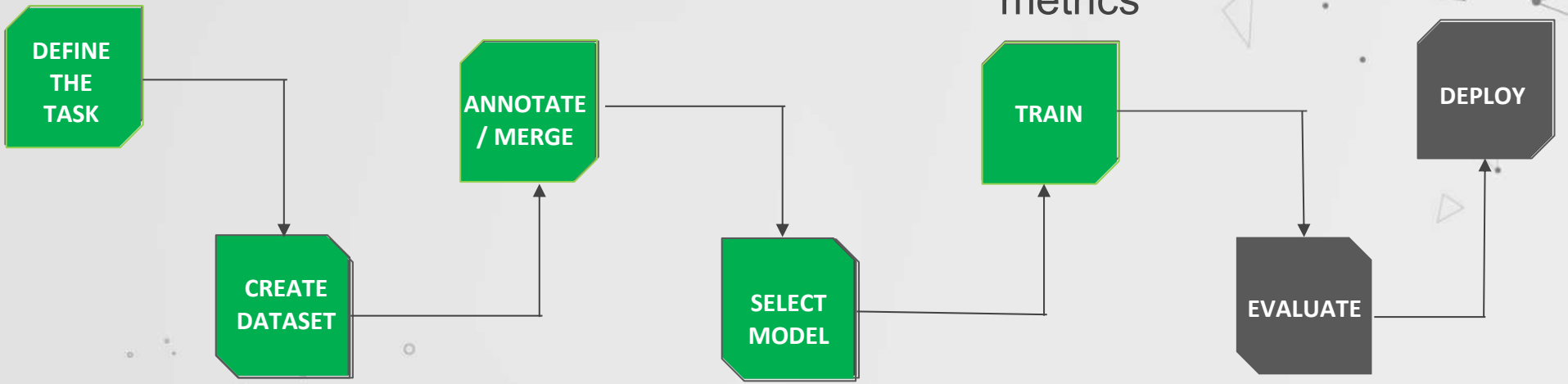
07

MODEL TRAINING

Be patient and well equipped….

WORKFLOW

Use the appropriate tools and Hardware and monitor closely the validation metrics



TRAINING IN PRACTICE

USE GPU

Pycharm Remote interpreter + GPU

MONITOR LOSS

Track the loss of your training and validation sets

SPLIT DATA

Use **train** ,**validation** and **test** datasets

EVALUATE

loss and mAP N epoch to avoid overfitting

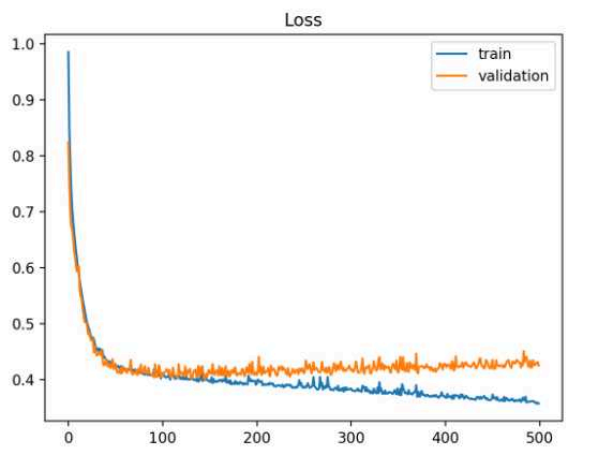
GET ANCHOR BOXES

Update anchor boxes that fit your data

SAVE CHECKPOINTS

Save your model periodically

TRAINING PROCESS : DEMO



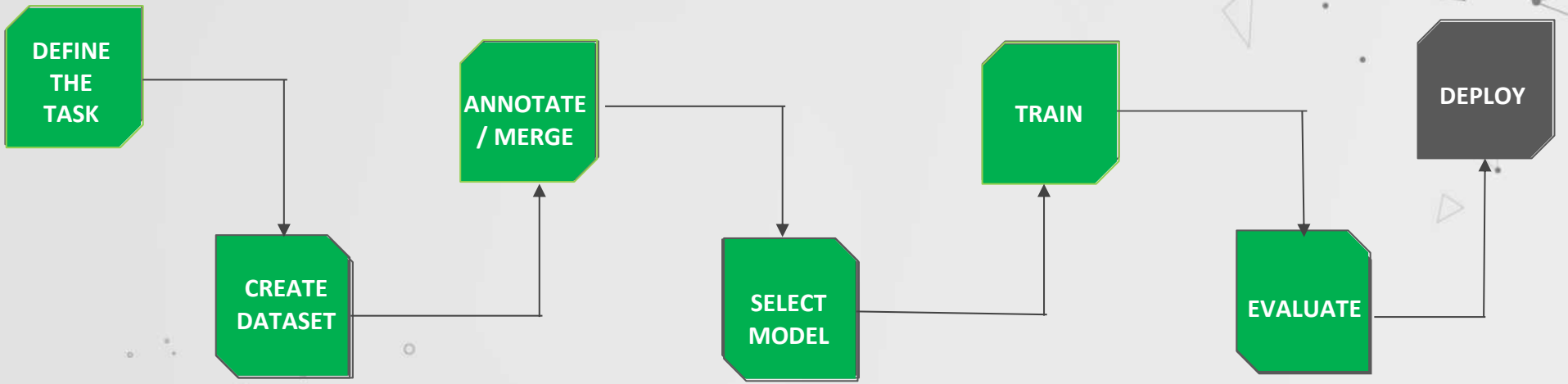
- Training DEMO Link
- Monitor Loss

08

PERFORMANCE EVALUATION

Know what your model is worth !

WORKFLOW

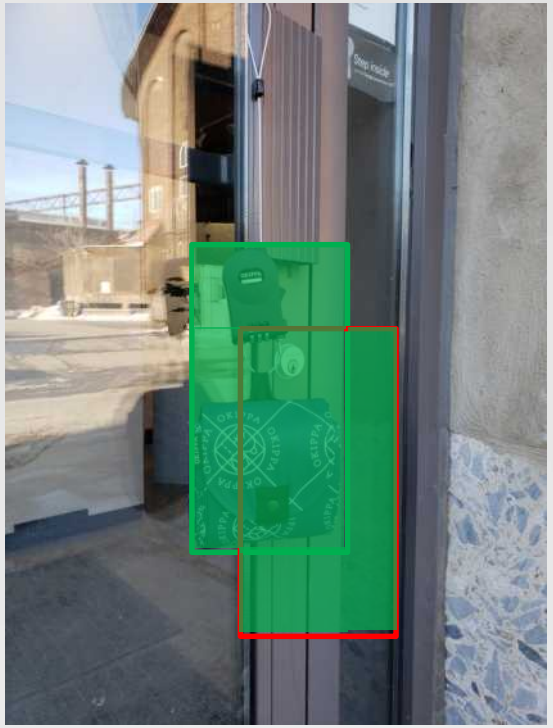


Test your models on unseen data and in real environment

IoU – Intersection over Union

- Measure the overlap between 2 boundaries.
- How much our predicted boundary overlaps with ground truth ?

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Ground truth

Prediction

Calculating the AP @ IoU = 0.5

IoU > 0.5 + correct class

True Positive

Ground truth

Prediction

TN is not evaluated as each image is assumed to have an object in it

Ground truth

IoU < 0.5

Localization error

False Positive

IoU < 0.5

False Positive

No IoU (<0.1)

Background error

False Positive

IoU > 0.5 + wrong class

False Negative

other

EVALUATION API

- **BE COMPLIANT TO AN EXISTING DATASET FORMAT :**
  - Results Format

```
[{
"image_id": 42,
"category_id": 18,
"bbox": [258.15, 41.29, 348.26, 243.78],
"score": 0.236},
 {…},{…},{…},{…}
]
```

cocoEval.summarize()

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.324
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.654
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.457
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.212
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.372
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.338
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.400
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.400
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.333
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.420
```

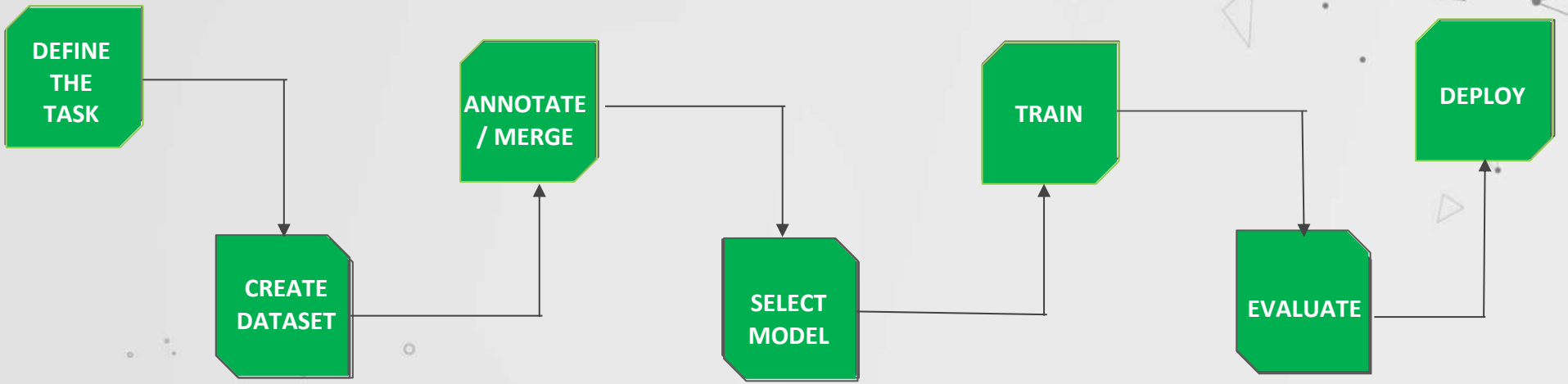
- **THERFORE YOU CAN USE THE EXISTING API :** Evaluation_Python_API demo

09

ON THE EDGE

Don’t get stuck in your machine…

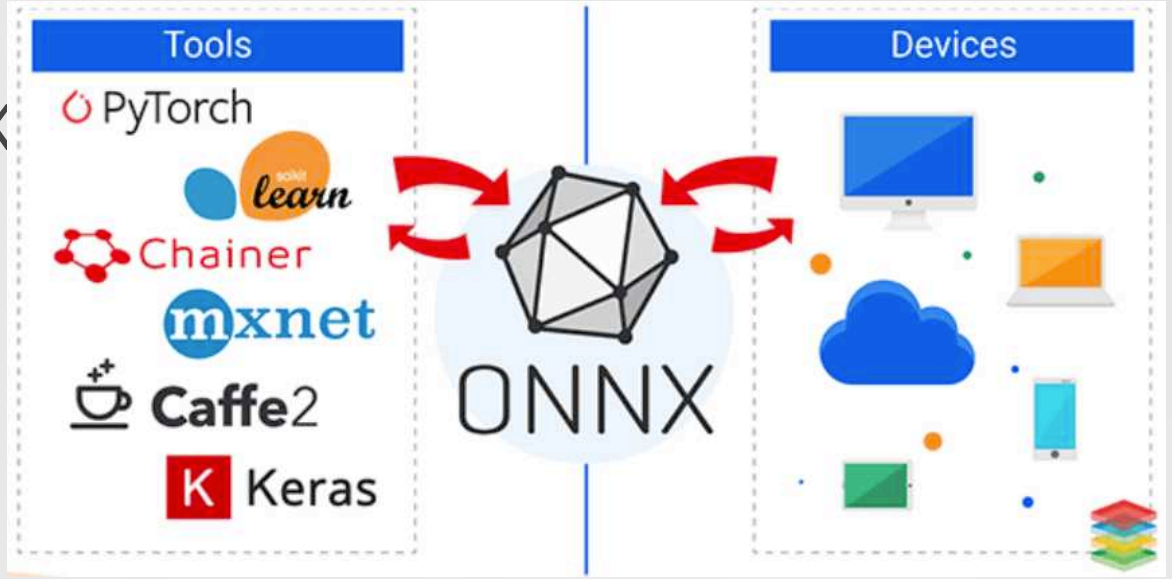
WORKFLOW


DEFINE THE TASK
CREATE DATASET
ANNOTATE / MERGE
SELECT MODEL
TRAIN
EVALUATE
DEPLOY
Export the model to appropriate hardware

ON THE EDGE

- Existing Model ZOO
- Export models to ONNX

torch.onnx.export
tf2onnx
keras2onnx.convert_keras



Framework interoperability

Leverage Power of Machine Learning with ONNX

By Ron Dagdag today @15:00 Wesmount 6

https://github.com/onnx/models/tree/master/vision/object_detection_segmentation/yolov3

ONNX : Code Example

```
from keras2onnx import convert_keras

onnxmodel = convert_keras(model, target_opset=target_opset, channel_first_inputs=['input_1'])
onnx.save_model(onnxmodel, model_file_name)
```

•

```
# runtime prediction
import onnxruntime
content = onnxmodel.SerializeToString()
sess = onnxruntime.InferenceSession(content)
feed = dict([(input.name, x[n]) for n, input in enumerate(sess.get_inputs())])
pred_onnx = sess.run(None, feed)
```

TOOLBOX FOR NON ML EXPERTS

Core ML

Core ML was introduced by Apple in 2017 as a new machine learning framework. Developers can now implement machine learning in their apps with just a few lines of code, making Core ML the best framework to get you introduced to using it.

TuriCreate + TuriAnnotate

Turi Create simplifies the development of custom machine learning models. You don’t have to be a machine learning expert to add recommendations, object detection, image classification, image similarity or activity classification to your app.

THANKS

Questions?

jacques-sylvain.lecointre@aspentech.com

mnubo
An Aspen Technology Company

SUPPORT SLIDES

For further discussions

FILTERS

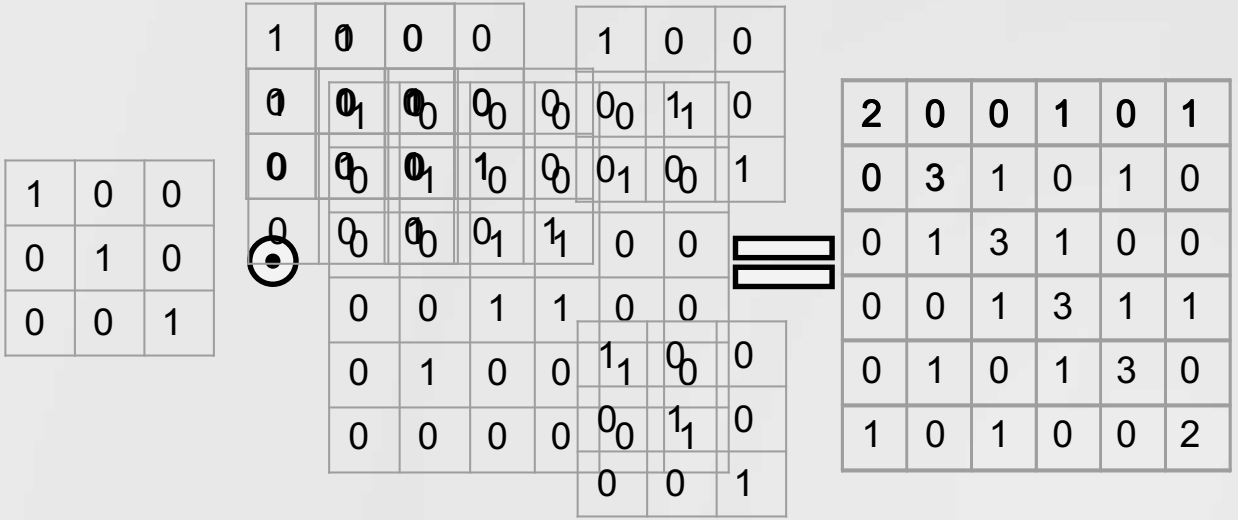
Create new pixel value based on neighboring pixels

- Example of Linear filters :
  - ✓ Low pass (Moyenneur,Gaussian)
  - ✓ High Pass (gradiant,Laplacian)
  - ✓ Template matching
  - ✓ Under sampling
  - ✓ Interpolation
- Example of non-Linear filters :
  - ✓ Median
  - ✓ Bilateral
  - ✓ ReLU
  - ✓ MaxPool

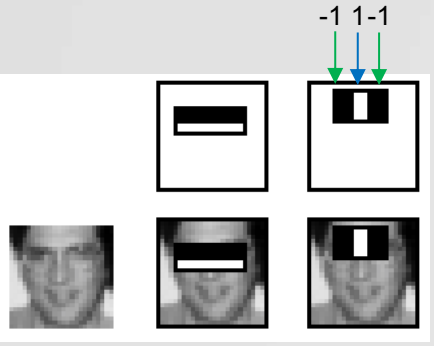
But how to choose these filters to extract relevant features ?

2D CONVOLUTIONAL FILTERS : HOW IT WORKS

The convolution filters use **local** neighbors to compute output pixel values with linear combinations of input pixel values



Viola and Jones detector



RECTANGLE FILTERS

01 RECTANGLE FILTERS

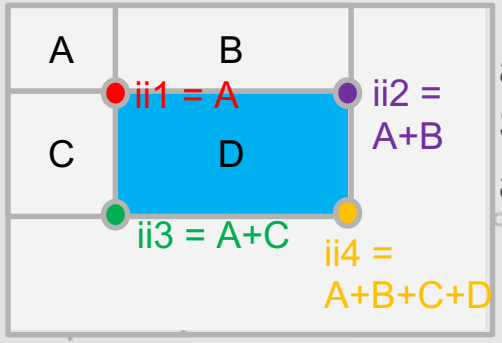
Haar like features

SUM(pixels under white rectangle)

–

SUM(pixels black rectangle)

02 INTEGRAL IMAGE



INTEGRAL IMAGE ii

at each pixel (x,y) :
SUM of the pixel values **above** and to the **left** of (x,y)

D = ii4 + ii1 - ii2 - ii3

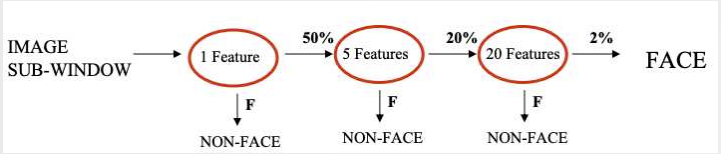
03 FEATURE SELECTION

- 180 000 potential features.
- AdaBoost algorithm to selects
- Learned features reflect the task

04 CASCADE OF CLASSIFIERS

- Simpler classifiers are used to reject the majority of subwindows

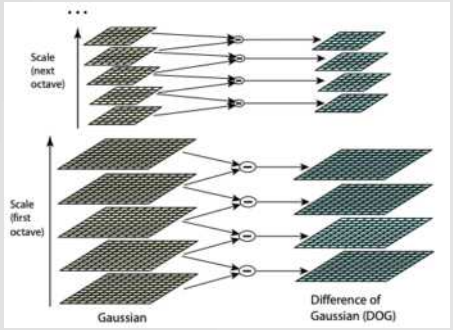
More complex classifiers are called upon to achieve low false positive rate



Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features.CVPR (1), 1(511-518)
https://www.cs.ubc.ca/~lowe/425/slides/13-ViolaJones.pdf

SIFT DESCRIPTORS

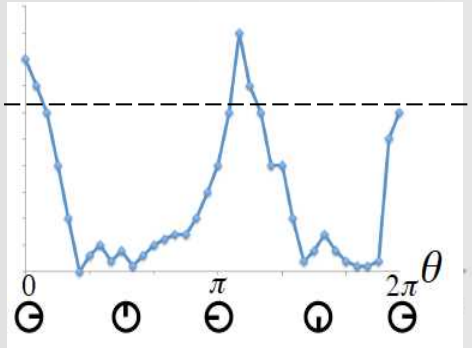
Scale-Invariant Feature Transform



01 DETECT LOCATION & SCALE

Location and Scale are found by maximizing the operator DoG

02 DETECTION ORIENTATION

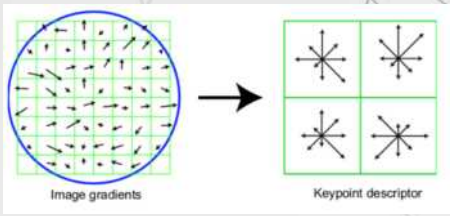


Histogram of Gradient on the intensity

$$\frac{dI(x,\sigma)}{dx} = \nabla I(x,\sigma) = \left[I_x, I_y\right] = \left[M, \theta\right]$$

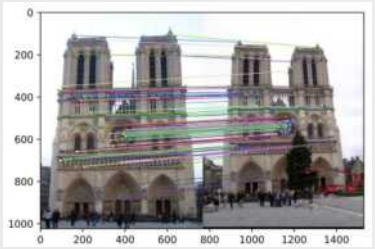
03 COMPUTE DESCRIPTOR

Histogram of Gradients



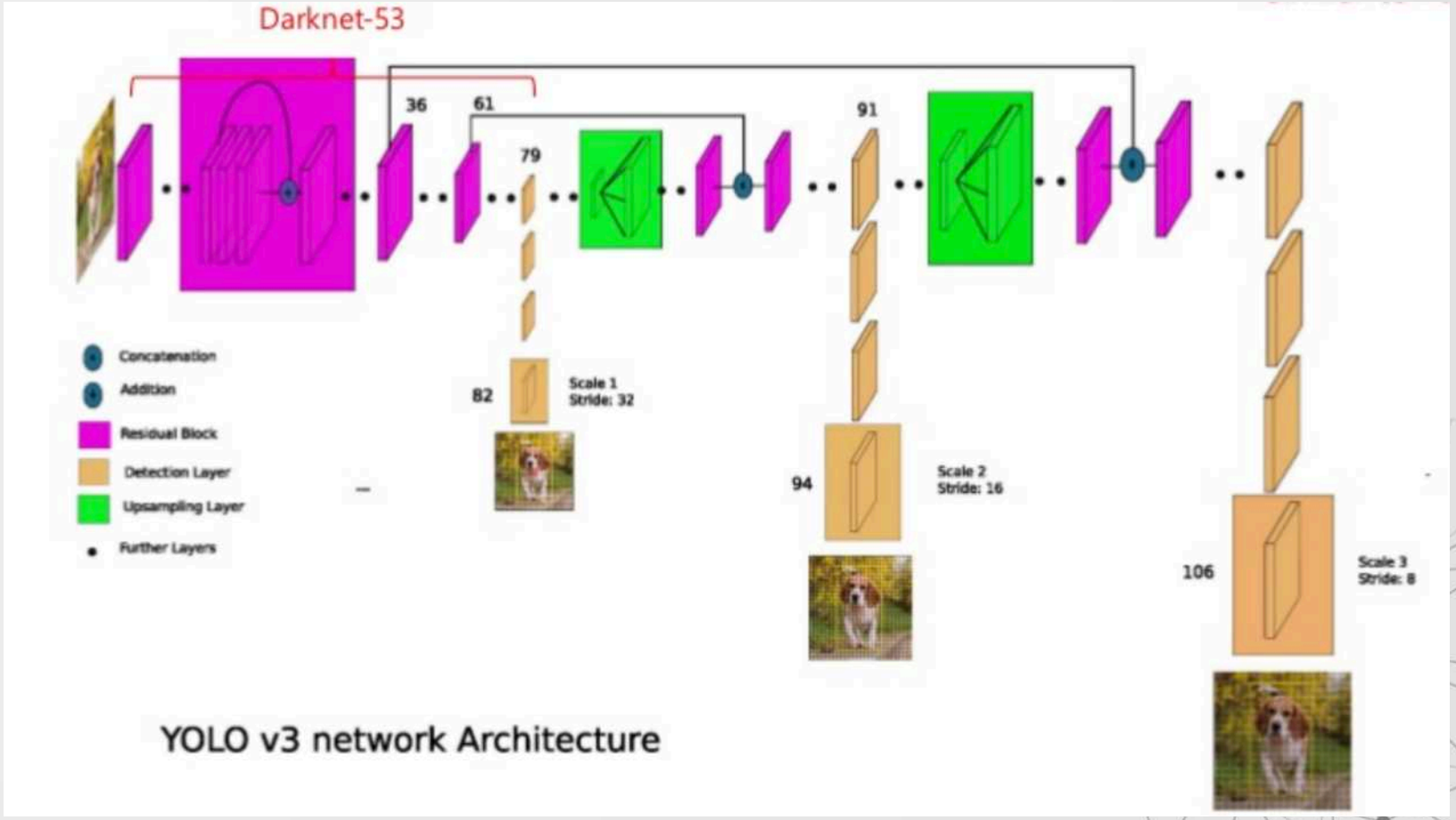
04 DESCRIPTOR MATCHING

Identify matches with similar descriptors



Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60, no. 2 (2004): 91-110.

Feature Extractor Architecture



mAP – mean Average Precision

- Interpolated precision is computed at each recall level
- For a given task and class – precision – recall curve is computed from a method's ranked output
- The mAP is the average of the AP for all classes

- The AP computed as follow :

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

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

TP = True positive
TN = True negative
FP = False positive
FN = False negative

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r)$$

To obtain a high score, a method must have precision at all levels of recall this penalizes methods which retrieve only a subset of examples with high precision (e.g. side views of cars).

DEMO YOLOv3 Inference with Open CV

```
# configuration and weight files for the model (Original Paper)
modelConfiguration = "yolov3.cfg"
modelWeights = "yolov3.weights"
classesFile = "coco.names"
# Initialize the parameters
confThreshold = 0.5  #Confidence threshold
nmsThreshold = 0.6   #Non-maximum suppression threshold
inpWidth = 416       #Width
inpHeight = 416      #Height
```

Model configuration files
and
pre-trained weights
from Darknet

```
net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)
```

Initialize DNN
from Darknet file

```
frame = cv2.imread('./bike.jpg')
```

```
# fixed spatial dimensions for input image, normalize
blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)
# specify blob as input
net.setInput(blob)
outs = net.forward(getOutputsNames(net))
# NMS
postprocess(frame, outs)
```

Inference
and
Non-Max Suppression