Workshop Prerequisites

- Windows, MacOS:
  - Python + Numpy + OpenCV 3.4.1 from pip:
    - Online: $ pip install opencv_python==3.4.1.15
    - Offline: follow Readme.md for you platform on flash drive
- Linux (Ubuntu 16 recommended):
  - $ apt get install python python-numpy
  - goo.gl/Y4cEdD

xperience / ai

OpenCV Everywhere

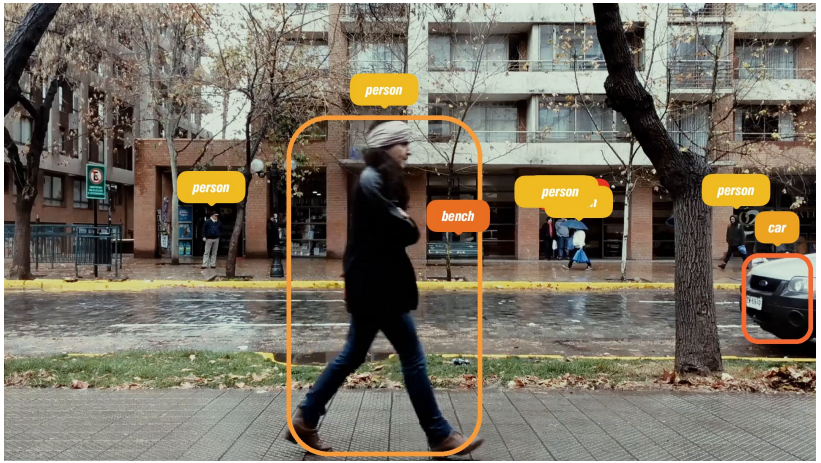
Anna Petrovicheva

Agenda

- Intro
- OpenCV everywhere
- OpenCV C++ && bindings
- OpenCV Python hands on
- IOT use case Java/Python + JavaScript all together
- Optimization

Xperience.ai

- Deep Learning for Computer Vision
- Optimized for Mobile and FPGA
  - Quantization and Pruning
- Full pipeline of model development
  - Getting the data right
  - Building the model
  - Porting to target hardware



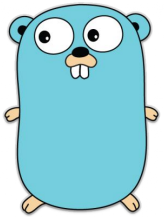
Tutorial materials

Github repo: github.com/xperience-ai/oreilly_opencv_workshop2018

- cxx
- data
- java
- js
- python-template
- python
- slides

OpenCV Everywhere

C++



OpenCV for Python

- Python interface for OpenCV is binding to native C++ code
- Generated from C++ interface automatically, supported by OpenCV team
- Available with Python pip (built with limited features support)
- Numpy is base library for containers and math primitives
- Python 2.x and Python 3.x are supported
- OpenCV provides two independent libraries for Python 2.x and Python 3.x

OpenCV: C++ && Python

C++	Python
using namespace cv	import cv2 # For both Python2 and Python3
cv::Mat, cv::Mat::dot	Numpy.array, numpy.array.dot
#include <imgproc/imgproc.hpp> cv::blur(src, dst)	cv2.blur(src, dst)
capture >> frame;	has_frame, frame = capture.read()

Python && C++ Documentation Together

Function Documentation

§ blobFromImage() [1/2]

```
Mat cv::dnn::blobFromImage ( InputArray     image,
                             double         scalefactor = 1.0,
                             const Size &   size = Size(),
                             const Scalar & mean = Scalar(),
                             bool           swapRB = true,
                             bool           crop = true
                           )
```

Python:

```
retval = cv.dnn.blobFromImage( image[, scalefactor[, size[, mean[, swapRB[, crop]]]]] )
```

Creates 4-dimensional blob from image. Optionally resizes and crops `image` from center, subtract `mean` values, scales values by `scalefactor`, swap Blue and Red channels.

Parameters

image	input image (with 1-, 3- or 4-channels).
size	spatial size for output image
mean	scalar with mean values which are subtracted from channels. Values are intended to be in (mean-R, mean-G, mean-B) order if

https://docs.opencv.org/3.4.1/

OpenCV for Java

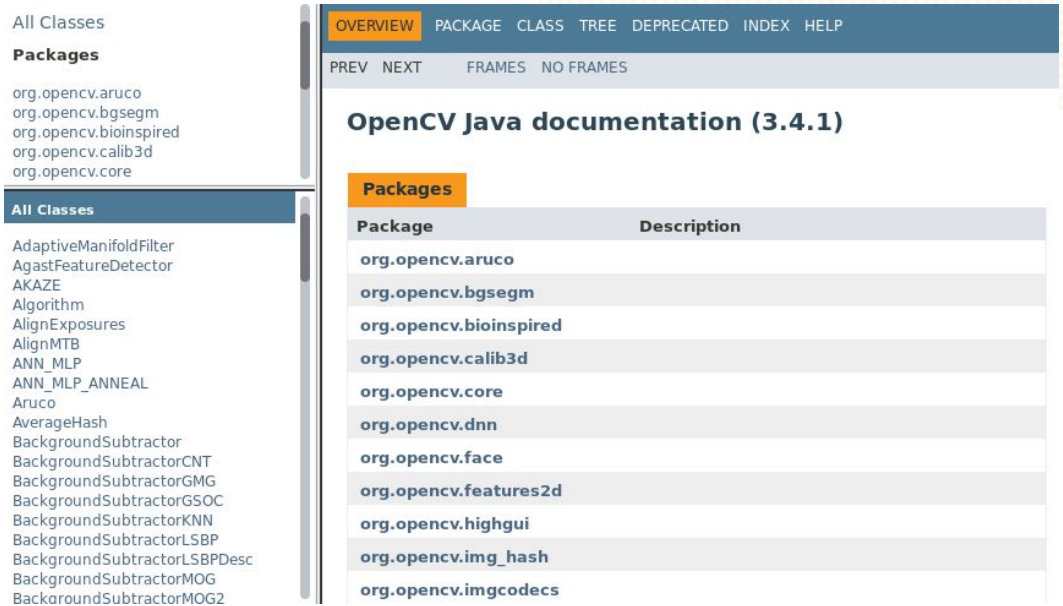
- Java interface for OpenCV is binding to native C++ code
- Generated from C++ interface automatically with manually implemented data container adapters, supported by OpenCV team
- Available for desktop platforms: Windows, Linux, Mac OS, etc
- Available for Android with Android SDK & NDK integration

OpenCV: C++ -> Java

C++	Java
cv::Mat, cv::Mat::dot	org.opencv.core.Mat, Mat.dot
#include <imgproc/imgproc.hpp> cv::blur(src, dst)	import org.opencv.imgproc.*; Imgproc.blur(src, dst)

capture >> frame;	capture.read(frame);
	
	// Or use platform specific like android.Camera

Javadoc



https://docs.opencv.org/3.4.1/javadoc/index.html

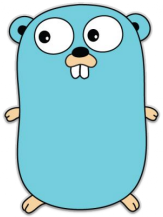
OpenCV JS

- OpenCV JS is transpiled from C++ code and has the same structure and functions naming
- Works as JS or WebAssembly library in browser, has integrations with <video> and Canvas
- No reference manual yet

C++	JavaScript
cv::Mat, cv::Mat::dot	cv.Mat, cv.Mat.dot
#include <imgproc/imgproc.hpp> cv::blur(src, dst)	utils.loadOpenCv(() => { cv.blur(src, dst) });
capture >> frame;	capture.read()

Golang, Haskell, D, other

- External bindings developed by community
- Manual implementation
- Not supported by core OpenCV team



Hands-on Scenario for Python

- Camera preview, live streaming
- Basic image processing and classic CV primitives
- Face Detection with DNN
- Face Recognition with DNN

Workshop Prerequisites

- Windows, MacOS:
  - Python + Numpy + OpenCV 3.4.1 from pip:
    - Online: $ pip install opencv_python==3.4.1.15
    - Offline: follow Readme.md for you platform on flash drive
- Linux (Ubuntu 16 recommended):
  - $ apt get install python python-numpy
  - goo.gl/Y4cEdD

Bootstrap

- For Windows and MacOS:
    - nothing
- For Linux:
    - $ export PYTHONPATH=<opencv_distro>/lib/python<version>/dist-packages:$PYTHONPATH
    - $ export LD_LIBRARY_PATH =<opencv_distro>/lib:$LD_LIBRARY_PATH
- Test:
    - import numpy
    - import cv2

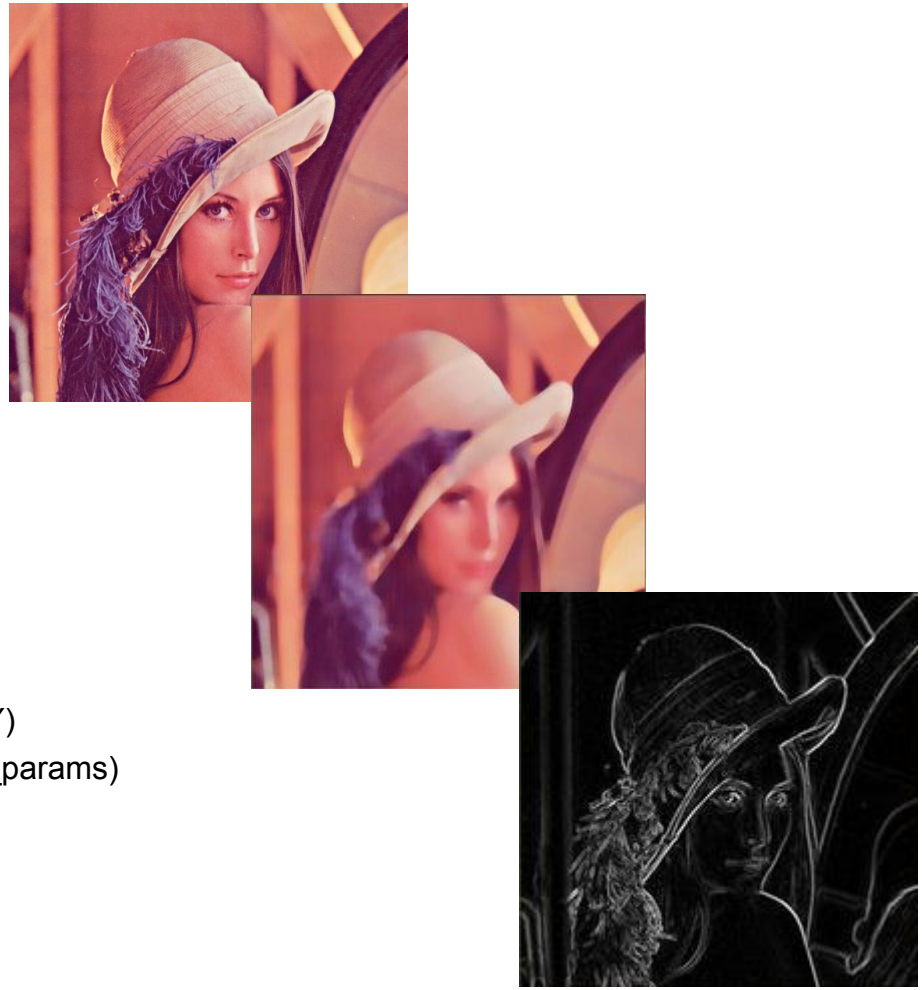
Step 0: Video decoding & Camera Access

```
source = cv2.VideoCapture(0)
win_name = 'Camera Preview'
cv2.namedWindow(win_name, cv2.WINDOW_NORMAL)

while cv2.waitKey(1) != 27: # Escape
    has_frame, frame = source.read()
    if not has_frame:
        break
    cv2.imshow(win_name, frame)

source.release()
cv2.destroyWindow(win_name)
```

See python-template/Camera/Camera.py in repository and on stick

Step 1: Camera Filters

```
feature_params = dict( maxCorners = 50, qualityLevel = 0.3,
                       minDistance = 7, blockSize = 7 )
...
if image_filter == PREVIEW:
    result = frame;
elif image_filter == CANNY:
    result = cv2.Canny(frame, 80, 90);
elif image_filter == BLUR:
    result = cv2.blur(frame, (13, 13));
elif image_filter == FEATURES:
     result = frame
     frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
     corners = cv2.goodFeaturesToTrack(frame_gray, **feature_params)
     if corners is not None:
         for x, y in numpy.float32(corners).reshape(-1, 2):
             cv2.circle(result, (x,y), 10, (0, 255 , 0), 1)
```



See python-template/CameraFilters/CameraFilters.py in repository and on stick

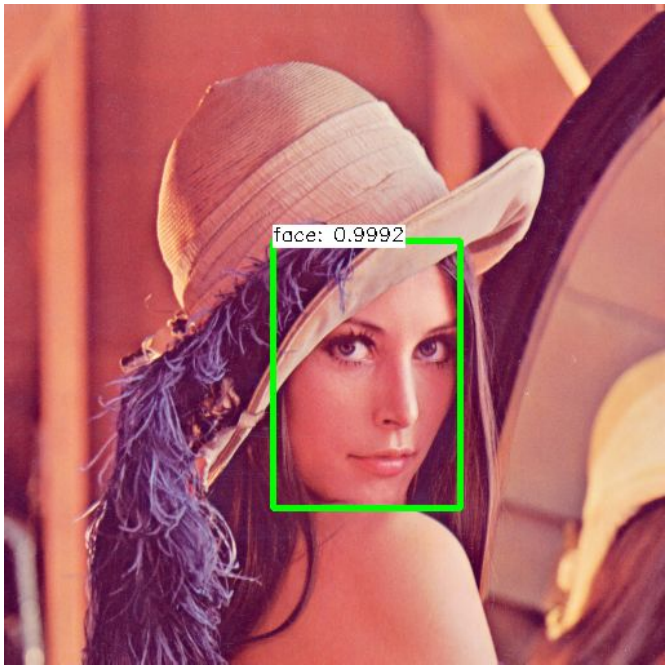
Step 2: Face Detection Initialization

```
# Model parameters
in_width = 300
in_height = 300
mean = [104, 117, 123]
conf_threshold = 0.7

net = cv2.dnn.readNetFromCaffe("../../data/deploy.prototxt",
                               "../../data/res10_300x300_ssd_iter_140000_fp16.caffemodel")
```

See python/Camera/Camera.py in repository and on stick

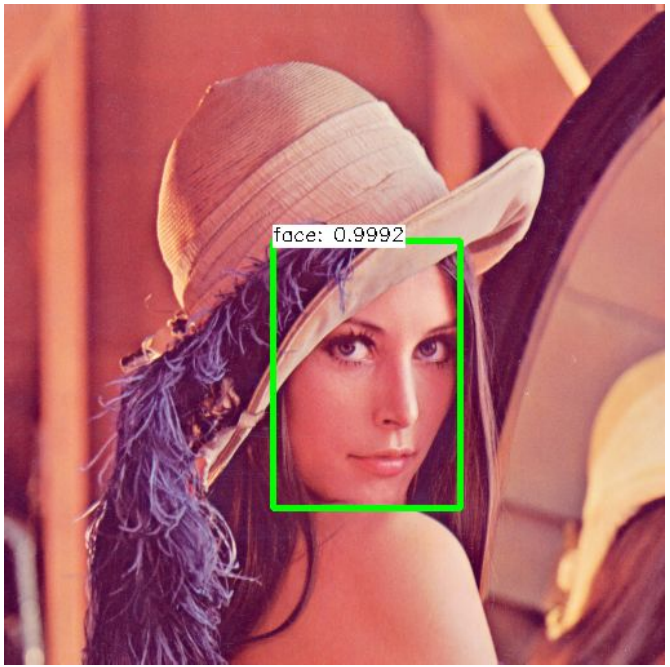
Step 2: Per-frame Face Detection Inference in Python

```
frame_height = frame.shape[0]
frame_width = frame.shape[1]
blob = cv2.dnn.blobFromImage(frame, 1.0, (in_width, in_height),
                                        mean, False, False)
net.setInput(blob)
detections = net.forward()

for i in range(detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    if confidence > conf_threshold:
        x_left_bottom = int(detections[0, 0, i, 3] * frame_width)
        y_left_bottom = int(detections[0, 0, i, 4] * frame_height)
        x_right_top = int(detections[0, 0, i, 5] * frame_width)
        y_right_top = int(detections[0, 0, i, 6] * frame_height)
```



Step 2: Per-frame Face Detection Inference in C++

```
Mat inputBlob = dnn::blobFromImage(frame, inScaleFactor, Size(inWidth, inHeight), meanVal, false, false);
net.setInput(inputBlob, "data");
Mat detection = net.forward("detection_out");

Mat detectionMat(detection.size[2], detection.size[3], CV_32F, detection.ptr<float>());
for(int i = 0; i < detectionMat.rows; i++)
{
   float confidence = detectionMat.at<float>(i, 2);

   if(confidence > confidenceThreshold)
   {
      float xLeftBottom = max(0, detectionMat.at<float>(i, 3) * frame.cols);
      float yLeftBottom = max(0, detectionMat.at<float>(i, 4) * frame.rows);
      float xRightTop = min(result.cols-1, detectionMat.at<float>(i, 5) * frame.cols);
      float yRightTop = min(result.rows-1, detectionMat.at<float>(i, 6) * frame.rows);
   }
}
```



See cxx/FaceDetection/main.cpp in repository and on stick

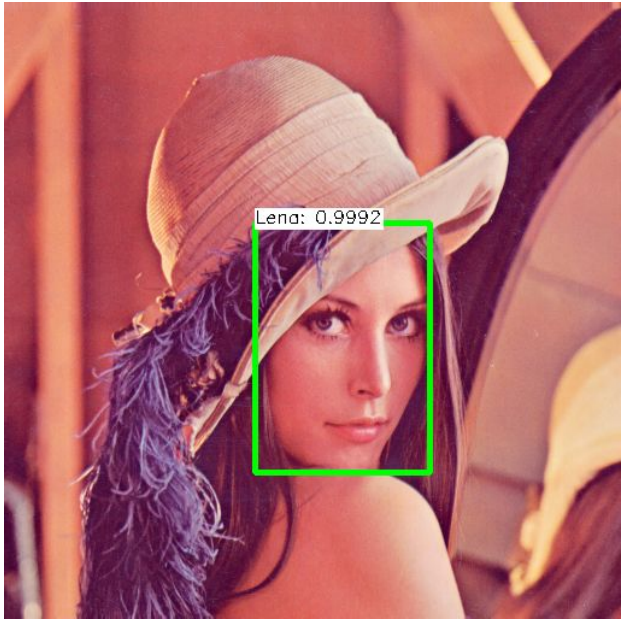
Step 3: Face Recognition with OpenCV DNN

```
class DnnRecognizer:
    def __init__(self, model_path='../../data/openface.nn4.small2.v1.t7',
                 model_mean = [0, 0, 0], model_in_size = (96, 96), model_scale = 1.0 / 255, conf_threshold = 0.6):
        self.known_faces = dict()
        self.model = cv2.dnn.readNetFromTorch(model_path)
        self.mean = model_mean
        self.scale = model_scale

    def _face2vec(self, face):
        blob = cv2.dnn.blobFromImage(face, self.scale, self.in_size, self.mean, False, False)
        self.model.setInput(blob)
        vec = self.model.forward()
        return vec
        self.in_size = model_in_size
        self.confidence = conf_threshold
```

See python-template/FaceDetectionAndRecognition/DnnRecognizer.py in repository and on stick

Step 3: Face Recognition with OpenCV DNN 2

```
def introduce(self, image, name):
    self.known_faces[name] = self._face2vec(image)

def recognize(self, image):
    vec = self._face2vec(image)
    best_match_name = 'unknown'
    best_match_score = self.confidence
    # NOTE: Replace items() method to iteritems() if you use Python2
    for name, descriptor in self.known_faces.items():
        score = vec.dot(descriptor.T)
        if (score > best_match_score):
            best_match_score = score
            best_match_name = name
    return best_match_name
```



IOT Use case: Online Recognition

- Server: HTTP REST API with POST requests on Java or Python:
  - /introduce?name=[person name] Body - JPEG or PNG image
  - /recognize Body - jpeg image
    - *See python/server/server.py and java/server/ in repository and on stick*
- Client:
  - WEB site with OpenCV JS
    - *See js/cascade_face_detection_remote_recognition.html in repository and on stick*
  - Android Application with OpenCV for Android

Step 1: Face detection with OpenCV JS

```
let video = document.getElementById('videoInput');
let src = new cv.Mat(video.height, video.width, cv.CV_8UC4);
let dst = new cv.Mat(video.height, video.width, cv.CV_8UC4);
let gray = new cv.Mat();
let cap = new cv.VideoCapture(video);
let faces = new cv.RectVector();
let classifier = new cv.CascadeClassifier();
const FPS = 30;
```

Step 2: Face detection with OpenCV JS

```
let begin = Date.now();
cap.read(src);
src.copyTo(dst);
cv.cvtColor(dst, gray, cv.COLOR_RGBA2GRAY, 0);
classifier.detectMultiScale(gray, faces, 1.1, 3, 0);
for (let i = 0; i < faces.size(); ++i)
{
    let face = faces.get(i);
    let point1 = new cv.Point(face.x, face.y);
    let point2 = new cv.Point(face.x + face.width, face.y + face.height);
    cv.rectangle(dst, point1, point2, [255, 0, 0, 255]);
}
cv.imshow('canvasOutput', dst);
if(faces.size() == 1 && !introducing)
{
    face_roi = src.roi(faces.get(0))
    cv.imshow('introducingFormFace', face_roi);
}
```

Optimization

- OpenCV contains SSE, AVX, AVX2 optimizations for x86 and NEON for ARM CPUs
- OpenCV provides Transparent API accelerated with KHRONOS OpenCL
- OpenCV includes set of cudaXXX modules with NVIDIA CUDA accelerated algorithms
- Deep Learning inference is optimized for CPU and GPU, different backends could be used:
  - OpenCV native
  - OpenCL path
  - Intel Inference Engine (part of Intel OpenVINO)

Performance: DNN backends

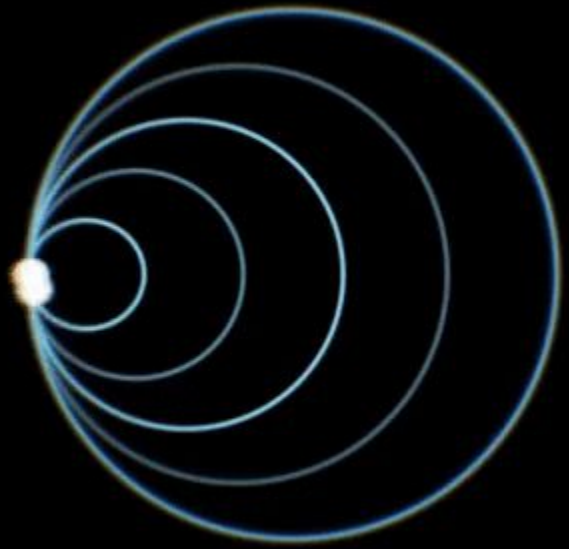
backends = (cv2.dnn.**DNN_BACKEND_DEFAULT**, cv2.dnn.**DNN_BACKEND_HALIDE**, cv2.dnn.**DNN_BACKEND_INFERENCE_ENGINE**, cv2.dnn.**DNN_BACKEND_OPENCV**)

targets = (cv2.dnn.**DNN_TARGET_CPU**, cv2.dnn.**DNN_TARGET_OPENCL**, cv2.dnn.**DNN_TARGET_OPENCL_FP16**, cv2.dnn.**DNN_TARGET_MYRIAD**)

net.setPreferableBackend(backend)
net.setPreferableTarget(target)

Useful Links and References

- Workshop repo: https://github.com/xperience-ai/oreilly_opencv_workshop2018
- OpenCV @ Github: https://github.com/opencv/opencv
- OpenCV online documentation: https://docs.opencv.org/3.4.1/
- Q&A forum: http://answers.opencv.org/questions/



xperience.ai