DL using OpenCV

Dmitry Kurtaev, IOTG Computer Vision

<dmitry.kurtaev@intel.com>

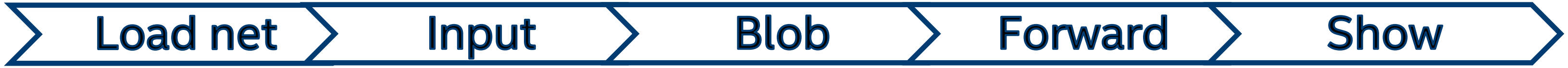
Internet of Things Group

Agenda

- General DL pipeline
- Face recognition on Raspberry Pi (**C++**)
- Image classification on Android (**Java**)
- Style transfer in browser (**JavaScript**)
- Edges2Cats on Windows (**Python**)

General DL pipeline

- An every sample consists of the following steps:
  1. Load deep learning network
  2. Get an input image
  3. Prepare a blob from image (normalize, resize, deinterleave)
  4. Make a forward pass through a network
  5. Interpret predictions and show it
- There are differences as for samples as for programming languages.


Load net

Input

Blob

Forward

Show

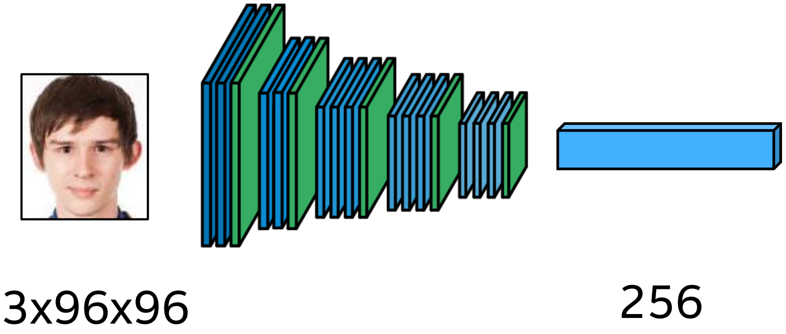
Face recognition on Raspberry Pi (**C++**)

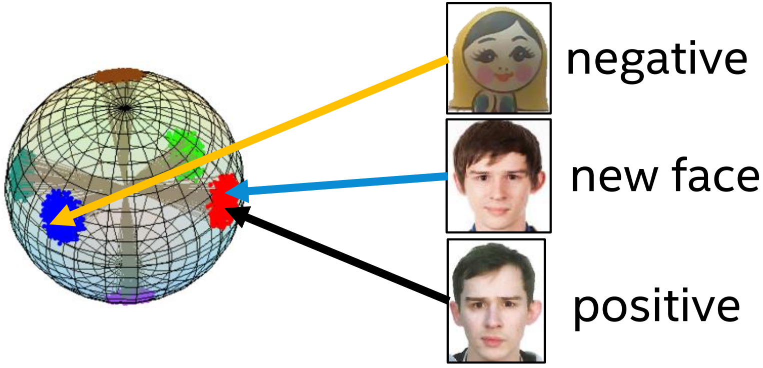
- Raspberry Pi is a single board computer with ARM CPU.
- Default OS – Raspbian (Debian based, Linux).
- Get OpenCV libraries by **apt-get** or build from source
- Download OpenCV's face detection and OpenFace face recognition networks
  - https://github.com/dkurt/icv_sunday_school_2019_spring/tree/master/face_recognition

Read more about cross compilation of OpenCV:
article: https://habr.com/ru/post/430906/ (Ночью спит спокойно мама — мы собираем OpenCV для Raspbian'a)
wiki: https://github.com/opencv/opencv/wiki/Intel%27s-Deep-Learning-Inference-Engine-backend#raspbian-stretch

Face recognition on Raspberry Pi (**C++**)

- Get a frame from a camera
- Pass it to **face detection** network to predict **bounding boxes**
- Crop faces and predict **embedding vectors** for them (second network)
- Save an embedding for a single person and then recognize only him or her





OpenFace project: https://github.com/cmusatyalab/openface

Face recognition on Raspberry Pi (**C++**)

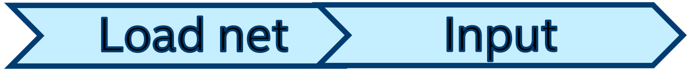
```
#include <opencv2/opencv.hpp>

int main(int argc, char** argv) {
  cv::dnn::Net faceDetector = cv::dnn::readNet("face_detector.prototxt",
                                               "face_detector.cafemodel");
  cv::dnn::Net faceRecogn = cv::dnn::readNet("openface_nn4.small2.v1.t7");

  // ...
}
```

Load net

Face recognition on Raspberry Pi (**C++**)

```
cv::VideoCapture cap(0);  // Open a USB camera device

cv::Mat frame;
while (cap.read(frame)) {
  // do something with frame
}
```

FD:



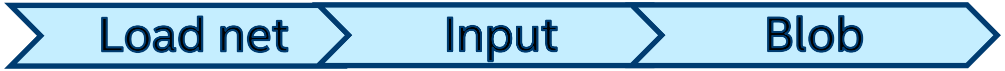
Load net

Input

Face recognition on Raspberry Pi (**C++**)

```
cv::VideoCapture cap(0);  // Open a USB camera device

cv::Mat frame, blob;
while (cap.read(frame)) {
  blob = cv::dnn::blobFromImage(frame, 1.0 /*scale*/,
                                cv::Size(160, 120) /*resize*/,
                                cv::Scalar(104,177,123) /*mean subtraction*/);

  // do something with blob
}
```

FD:



Load net

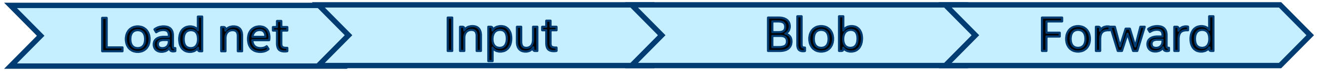
Input

Blob

Face recognition on Raspberry Pi (**C++**)

```
cv::VideoCapture cap(0);  // Open a USB camera device

cv::Mat frame, blob;
while (cap.read(frame)) {
  blob = cv::dnn::blobFromImage(frame, 1.0 /*scale*/,
                               cv::Size(160, 120) /*resize*/,
                               cv::Scalar(104,177,123) /*mean subtraction*/);
  faceDetector.setInput(blob);
  cv::Mat out = faceDetector.forward();
}
```

FD:



Load net

Input

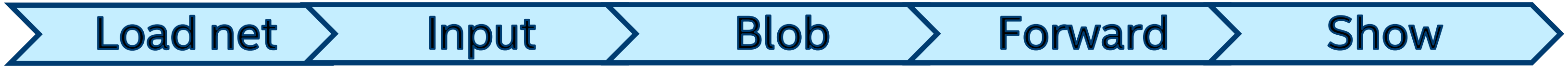
Blob

Forward

Face recognition on Raspberry Pi (**C++**)

```
float* detections = (float*)out.data; // out has shape 1x1xNx7
// Detections are [batchId(0),classId(0),confidence,left,top,right,bottom]
for (int i = 0; i < out.total() / 7; ++i) {
  float confidence = detections[i * 7 + 2];
  if (confidence < 0.7)
    continue;
  int l = detections[i * 7 + 3] * frame.cols;
  int t = detections[i * 7 + 4] * frame.rows;
  int r = detections[i * 7 + 5] * frame.cols;
  int b = detections[i * 7 + 6] * frame.rows;
  cv::rectangle(frame, cv::Point(l, t), cv::Point(r, b), cv::Scalar(0,255,0));
}
cv::imshow("Face detection", frame);
```

FD:



Load net

Input

Blob

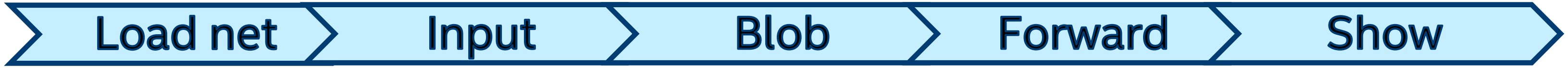
Forward

Show

Face recognition on Raspberry Pi (**C++**)

```
cv::Mat face = frame.rowRange(t, b).colRange(l, r);
cv::Mat blob = cv::dnn::blobFromImage(face, 1.0 / 255 /*scale*/,
                                      cv::Size(96, 96) /*resize*/,
                                      cv::Scalar() /*no mean*/,
                                      true /*swap red and blue*/);
faceRecogn.setInput(blob);
cv::Mat embedding = faceRecogn.forward();

if (embedding.dot(targetEmbedding) > 0.8)
  cv::rectangle(frame, cv::Point(l, t), cv::Point(r, b), cv::Scalar(0,255,0));
else
  cv::rectangle(frame, cv::Point(l, t), cv::Point(r, b), cv::Scalar(0,0,255));
```

FR:



Face recognition on Raspberry Pi (**C++**)

It's time for demo!

Image classification on Android (**Java**)

- You can build OpenCV for Android. It’s a native library with Java wrappers.
- Getting started guide for Android Studio:
  - https://docs.opencv.org/master/d0/d6c/tutorial_dnn_android.html
- An article at Habrahabr: the force will be with me to finish it! (#opencv4arts)

Sample source code: https://github.com/dkurt/icv_sunday_school_2019_spring/tree/master/classification

Image classification on Android (**Java**)

```java
public class MainActivity extends AppCompatActivity implements CvCameraViewListener2 {
  @Override
  public void onResume() {
    super.onResume();
    System.loadLibrary("opencv_java4");
    mOpenCvCameraView.enableView();
  }

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.CameraView); // Set up camera listener.
    mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
    mOpenCvCameraView.setCvCameraViewListener(this);
  }

  @Override
  public void onCameraViewStarted(int width, int height) {}

  @Override
  public void onCameraViewStopped() {}

  public Mat onCameraFrame(CvCameraViewFrame inputFrame) {}
  private CameraBridgeViewBase mOpenCvCameraView;
}
```

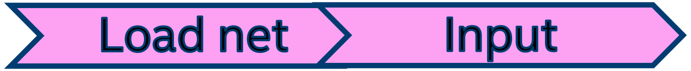
Image classification on Android (**Java**)

```java
@Override
public void onCameraViewStarted(int width, int height) {
  String prefix = "/sdcard/Android/data/org.opencv.samples.icvdemo/";
  String weights = prefix + "squeezenet_v1.1.caffemodel";
  String config = prefix + "squeezenet_v1.1.prototxt";
  net = Dnn.readNet(weights, config);
}

private Net net;
```

Load net

Image classification on Android (**Java**)

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
  Mat frame = inputFrame.rgba();
  return frame;
}
```

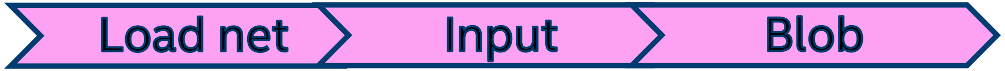


Load net

Input

Image classification on Android (**Java**)

```java
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
  Mat frame = inputFrame.rgba();

  Mat frameBGR = new Mat();
  Imgproc.cvtColor(frame, frameBGR, Imgproc.COLOR_RGBA2BGR);
  Mat blob = Dnn.blobFromImage(frameBGR, 1.0, new Size(227, 227),
                               new Scalar(104, 117, 123));

  return frame;
}
```



Load net

Input

Blob

Image classification on Android (**Java**)

```java
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
  Mat frame = inputFrame.rgba();

  Mat frameBGR = new Mat();
  Imgproc.cvtColor(frame, frameBGR, Imgproc.COLOR_RGBA2BGR);
  Mat blob = Dnn.blobFromImage(frameBGR, 1.0, new Size(227, 227),
                               new Scalar(104, 117, 123));

  net.setInput(blob);
  Mat out = net.forward();

  return frame;
}
```

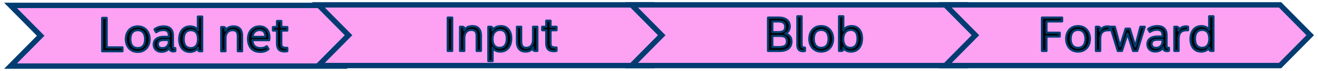


Image classification on Android (**Java**)

```java
Core.MinMaxLocResult loc = Core.minMaxLoc(out);
if (loc.maxVal > 0.5)
  Log.i("mytag", "class id: " + loc.maxLoc.x);
```

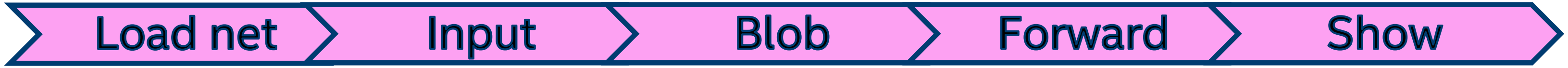
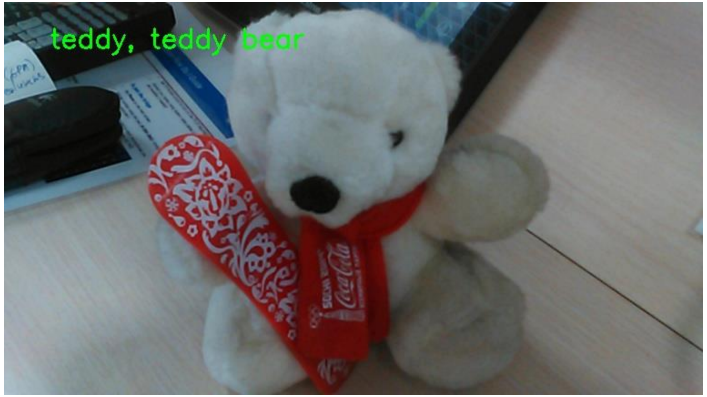
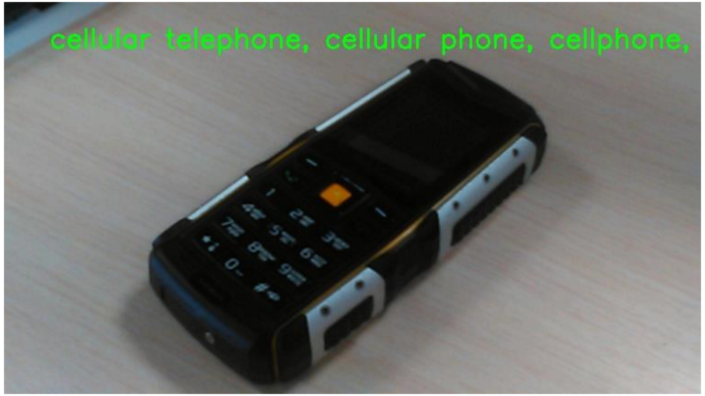
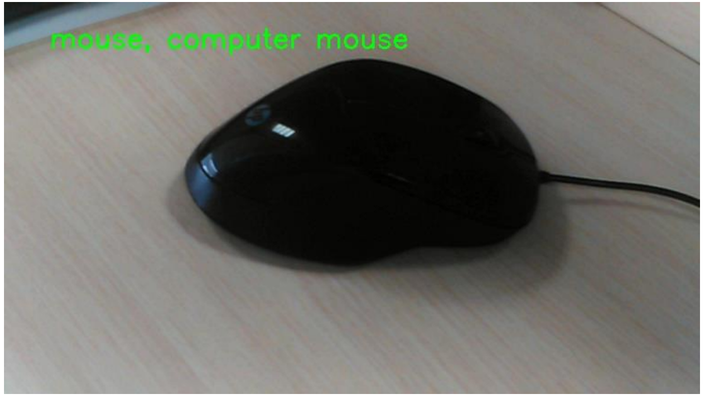


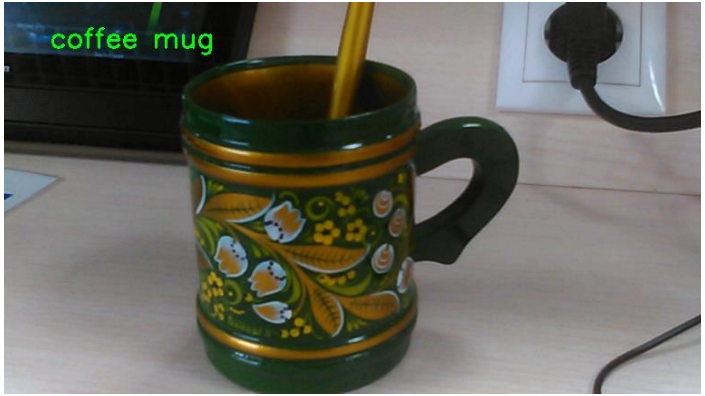
Image classification on Android (**Java**)


teddy, teddy bear


cellular telephone, cellular phone, cellphone,

It’s time for demo!


mouse, computer mouse


coffee mug

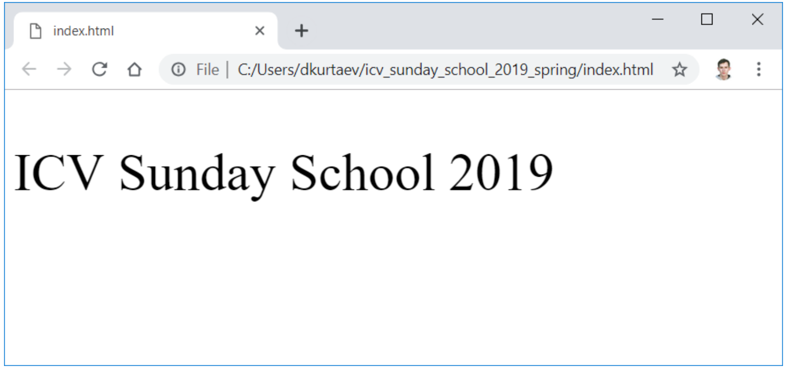
Style transfer in browser (JavaScript)

- You can build OpenCV which is C++ library to JavaScript code! Using Emscripten.
- OpenCV.js reads images data from `<canvas>` and `<img>`

Read more about OpenCV.js: https://habr.com/ru/company/intel/blog/437600/ (opencv4arts: Нарисуй мой город, Винсент)
Sample source code: https://github.com/dkurtaev/dkurtaev.github.io/tree/master/opencv4arts

Style transfer in browser (JavaScript)

```
<!DOCTYPE html>
<html>
<head>
  <script async src="https://docs.opencv.org/master/opencv.js"></script>
  <script src="https://docs.opencv.org/master/utils.js"></script>
  <script type='text/javascript'>
    function main() {
      document.getElementById('title').innerHTML += ' 2019';
    };
  </script>
</head>

<body onload="main()">
  <p id="title">ICV Sunday School</p>
</body>
</html>
```


index.html
File | C:/Users/dkurtaev/icv_sunday_school_2019_spring/index.html

ICV Sunday School 2019

Style transfer in browser (JavaScript)

```
<script type='text/javascript'>
  var utils = new Utils('');
  var net;
  var url = "https://people.eecs.berkeley.edu/~taesung_park/" +
            "CycleGAN/models/style_vangogh.t7";

  utils.createFileFromUrl("style_vangogh.t7", url, () => {
    net = cv.readNet("style_vangogh.t7");
    // add a callback
  });
</script>
```

Load net

Style transfer in browser (JavaScript)

```
<head>
<script type='text/javascript'>

  // Read an image from canvas and convert it to BGR.
  var imgRGBA = cv.imread('canvasInput');
  var imgBGR = new cv.Mat(imgRGBA.rows, imgRGBA.cols, cv.CV_8UC3);
  cv.cvtColor(imgRGBA, imgBGR, cv.COLOR_RGBA2BGR);

</script>
</head>

<body>
  <canvas id="canvasInput"></canvas>
</body>
```

Load net

Input

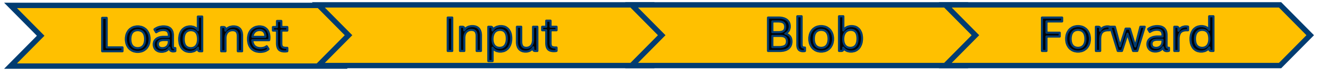
Style transfer in browser (JavaScript)

```
var blob = cv.blobFromImage(imgBGR, 1.0 / 127.5, // scale
                            {width: 640, height: 480}, // resize
                            [127.5, 127.5, 127.5, 0]); // mean subtraction
```

Load net
Input
Blob

Style transfer in browser (JavaScript)

```
var blob = cv.blobFromImage(imgBGR, 1.0 / 127.5, // scale
                            {width: 640, height: 480}, // resize
                            [127.5, 127.5, 127.5, 0]); // mean subtraction
net.setInput(blob);
var out = net.forward();
```



Load net

Input

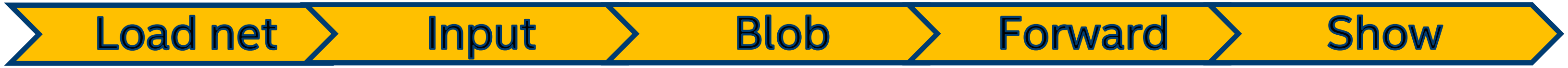
Blob

Forward

Style transfer in browser (JavaScript)

```
// Output values are in range [-1, 1]. Normalize it to [0, 255] of UInt8.
var outNorm = new cv.Mat();
out.convertTo(outNorm, cv.CV_8U, 127.5, 127.5);

// Postprocessing: create an interleaved image from planar.
var outHeight = out.matSize[2];
var outWidth = out.matSize[3];
var planeSize = outHeight * outWidth;
var data = outNorm.data;
var b = cv.matFromArray(outHeight, outWidth, cv.CV_8UC1, data.slice(0, planeSize));
var g = cv.matFromArray(outHeight, outWidth, cv.CV_8UC1, data.slice(planeSize, 2 * planeSize));
var r = cv.matFromArray(outHeight, outWidth, cv.CV_8UC1, data.slice(2 * planeSize, 3 * planeSize));
var vec = new cv.MatVector();
vec.push_back(r);
vec.push_back(g);
vec.push_back(b);
var rgb = new cv.Mat();
cv.merge(vec, rgb);

cv.imshow("canvasOutput", rgb);  // Also canvas
```

Call to action!



Load net

Input

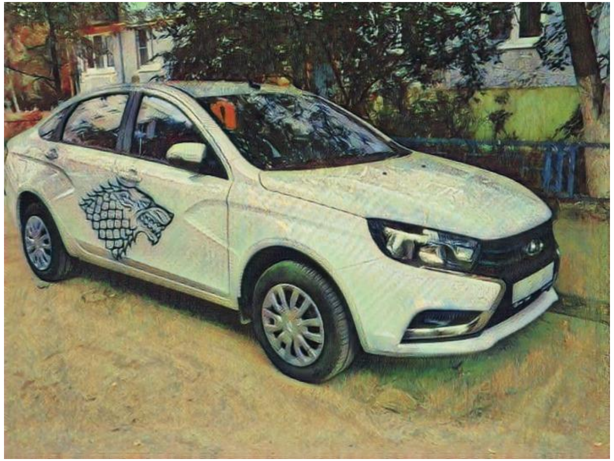
Blob

Forward

Show

Style transfer in browser (JavaScript)

It’s time for demo!



https://dkurtaev.github.io/opencv4arts

Edges2Cats on Windows (Python)

- It's a fun derivative of **pix2pix** approach.
  - original (Torch): https://github.com/phillipi/pix2pix
  - ported (TensorFlow): https://github.com/affinelayer/pix2pix-tensorflow
- Besides cats there are models which can translate:
  - Satellite photos to maps
  - Day photos to night ones
  - Grayscale images to colored

Sample source code: https://github.com/dkurt/icv_sunday_school_2019_spring/tree/master/edges2cats

Edges2Cats on Windows (**Python**)

```python
import cv2 as cv

net = cv.dnn.readNet('edges2cats.pb')
```

Load net

Edges2Cats on Windows (**Python**)

```
import numpy as np

size = 256

# canvas is filled by ones and we’ll draw contours by zeros
canvas = np.ones([size, size, 3], dtype=np.float32)
```

Load net

Input

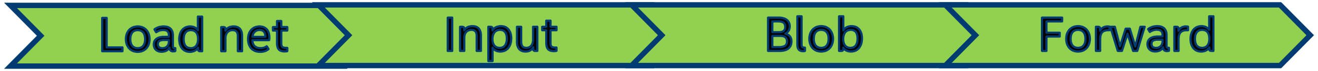
Edges2Cats on Windows (**Python**)

```
# [0, 1] to [-1, 1]
blob = cv.dnn.blobFromImage(canvas, 2.0,     # scale
                            (size, size),    # resize
                            (0.5, 0.5, 0.5)) # mean subtraction
```

Load net
Input
Blob

Edges2Cats on Windows (**Python**)

```
# [0, 1] to [-1, 1]
blob = cv.dnn.blobFromImage(canvas, 2.0,     # scale
                            (size, size),    # resize
                            (0.5, 0.5, 0.5)) # mean subtraction
net.setInput(blob)
out = net.forward()
```



Load net

Input

Blob

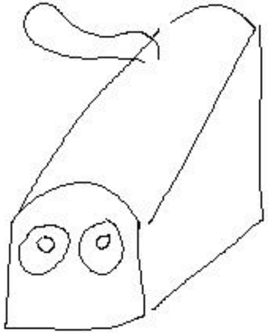
Forward

Edges2Cats on Windows (**Python**)

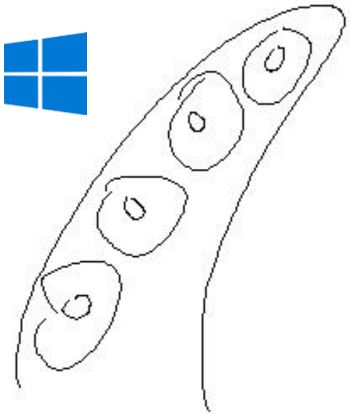
```
# [-1, 1] to [0, 1]
out += 1
out /= 2

# NCHW to HWC
res = out.transpose(0, 2, 3, 1).reshape(size, size, 3)

# RGB to BGR
res = res[:,:,[2, 1, 0]]

cv.imshow('Edges2Cats', res)
```

Load net
Input
Blob
Forward
Show

Edges2Cats on Windows (**Python**)



It’s time for demo!



OpenCV's deep learning module summary

C++

```
cv::dnn::Net net =
    cv::dnn::readNet(...);
```

```
net.setInput(blob);
```

```
cv::Mat out = net.forward();
```

Java

```
Net net = Dnn.readNet(...);
```

```
net.setInput(blob);
```

```
Mat out = net.forward();
```

JavaScript

```
var net = cv.readNet(...);
```

```
net.setInput(blob);
```

```
var out = net.forward();
```

Python

```
net = cv.dnn.readNet(...)
```

```
net.setInput(blob)
```

```
out = net.forward()
```

Calls to action

- Experiment!
- Try https://freecodecamp.org (interactive JavaScript tutorials)