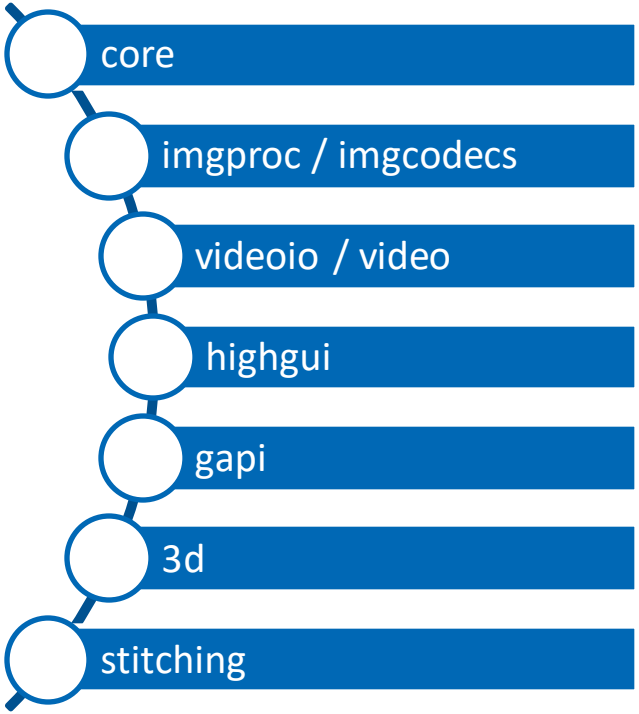
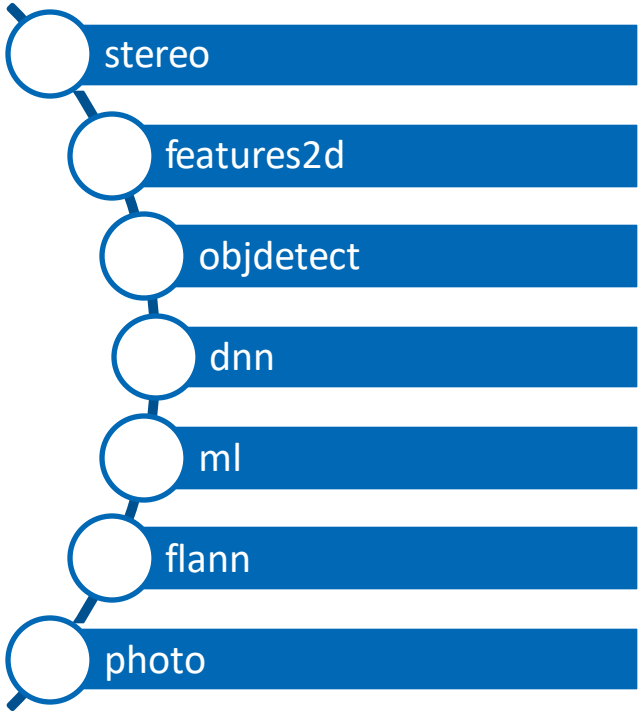
Курс лекций по компьютерному зрению

Введение в библиотеку OpenCV: Модуль DNN

Нестеров Александр, AI Frameworks Engineer

Main modules

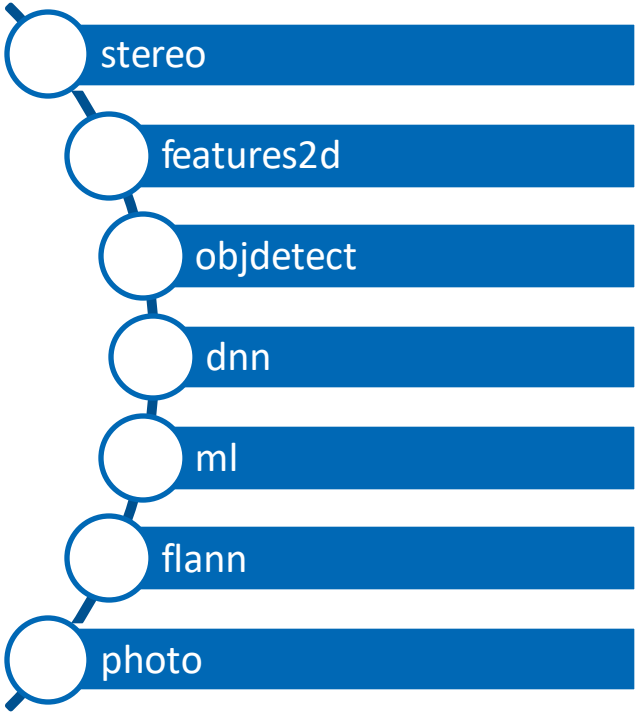




OpenCV

https://github.com/opencv/opencv

Main modules



- Поддержка глубоких сверточных нейронных сетей
- OpenVINO модуль

https://docs.opencv.org/5.x/d6/d0f/group__dnn.html

Main structure: **Mat**

cv::Mat A(h, w, CV_8UC3);

- Size, step
- Ref counter (=1)
- Data pointer

cv::Mat B = A;

- Size, step
- Ref counter (=2)
- Same data pointer

cv::Mat C=A(roi);

Size of the ROI, same step
Ref counter (=3)
Augmented data pointer

Main structure: **Mat**

cv::Mat C=A(roi);

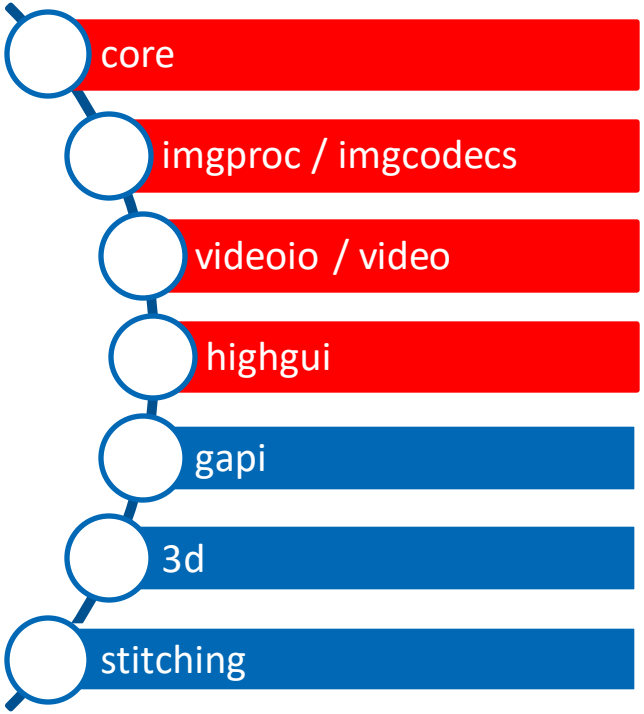
Size of the ROI, same step
Ref counter (=3)
Augmented data pointer

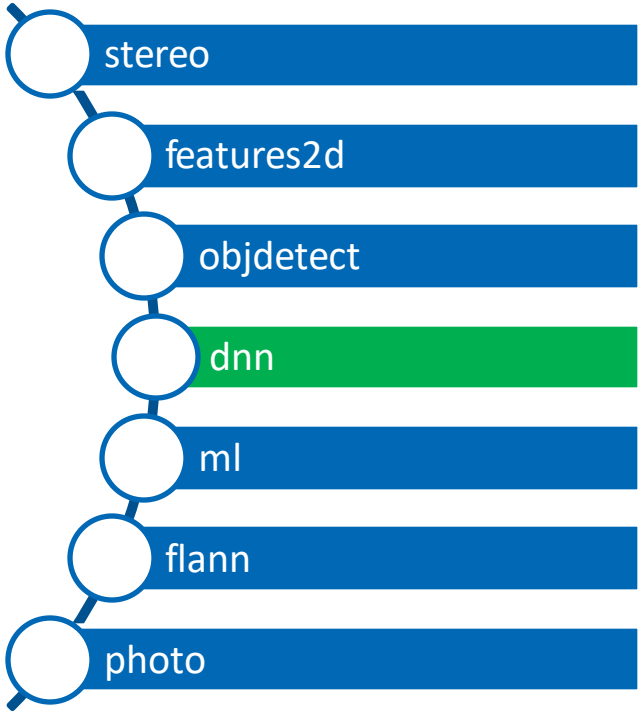
Раскладка памяти матрицы C

RGBRGBRGBRGB**********RGBRGBRGBRGB**********
…

OpenCV

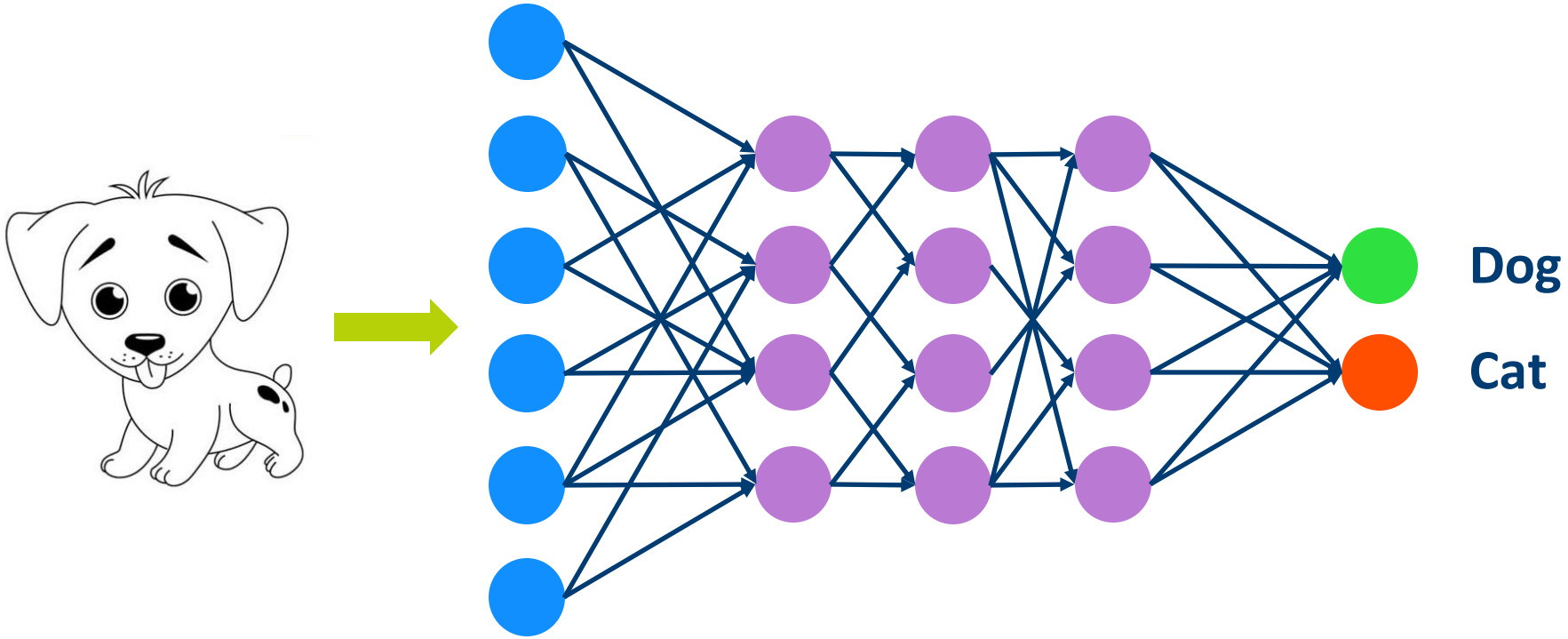
Main modules





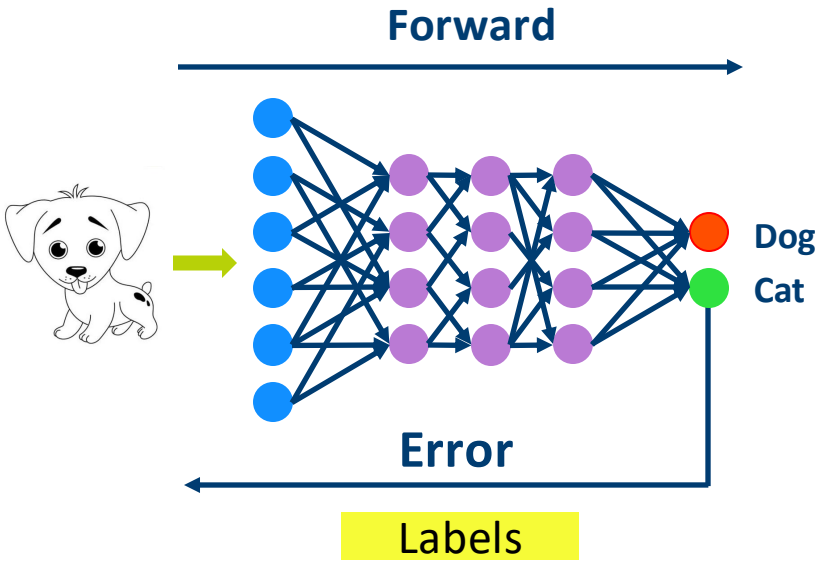
https://github.com/learning-process/opencv_practice

Deep neural network

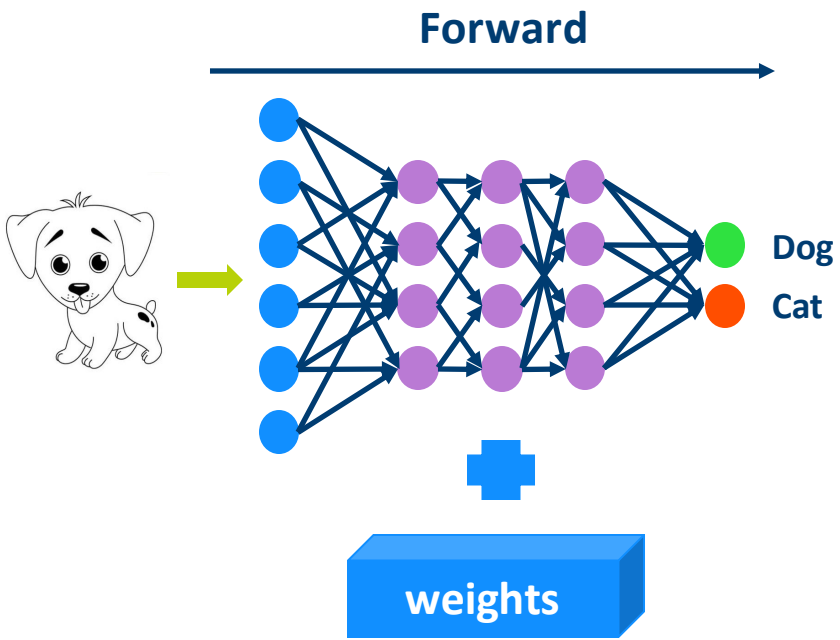


What can be done with DNN?

Learning

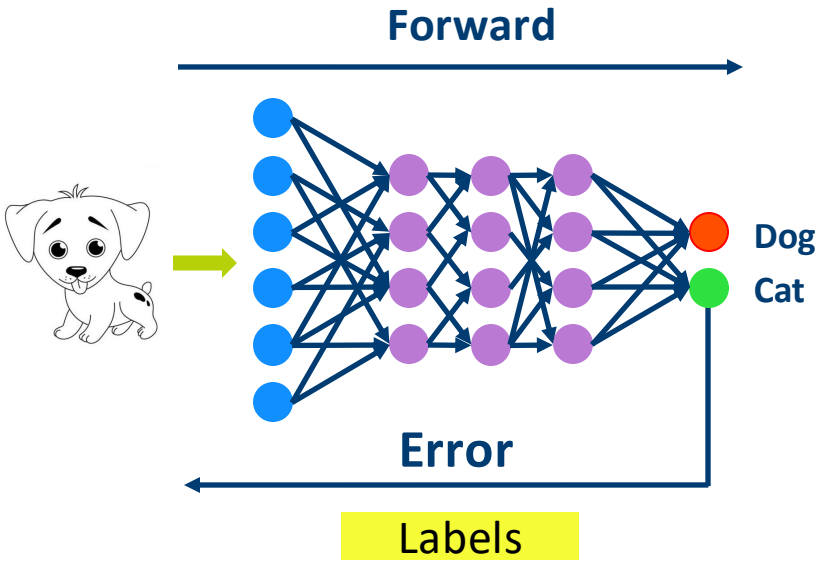


Inference

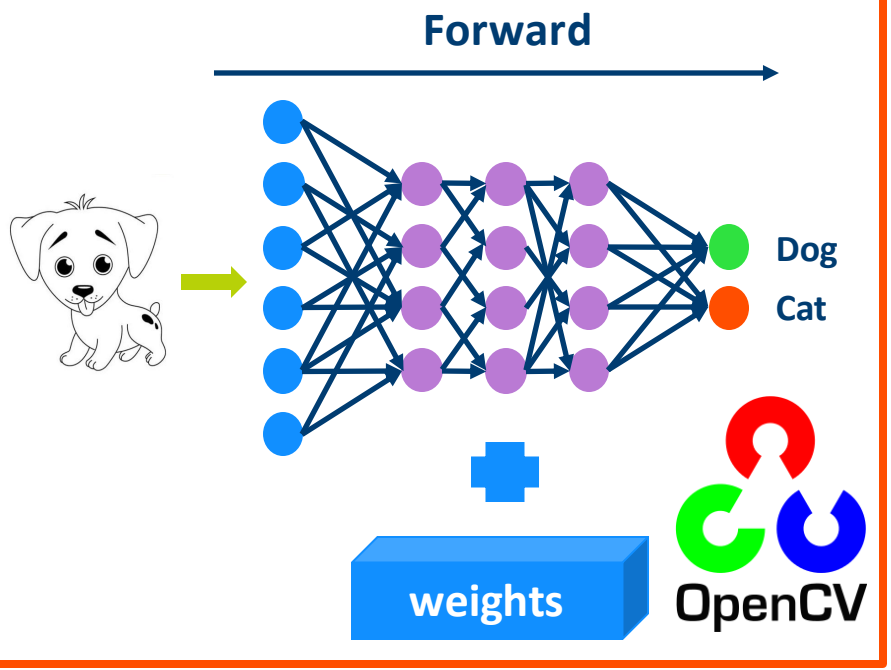


What can be done with DNN?

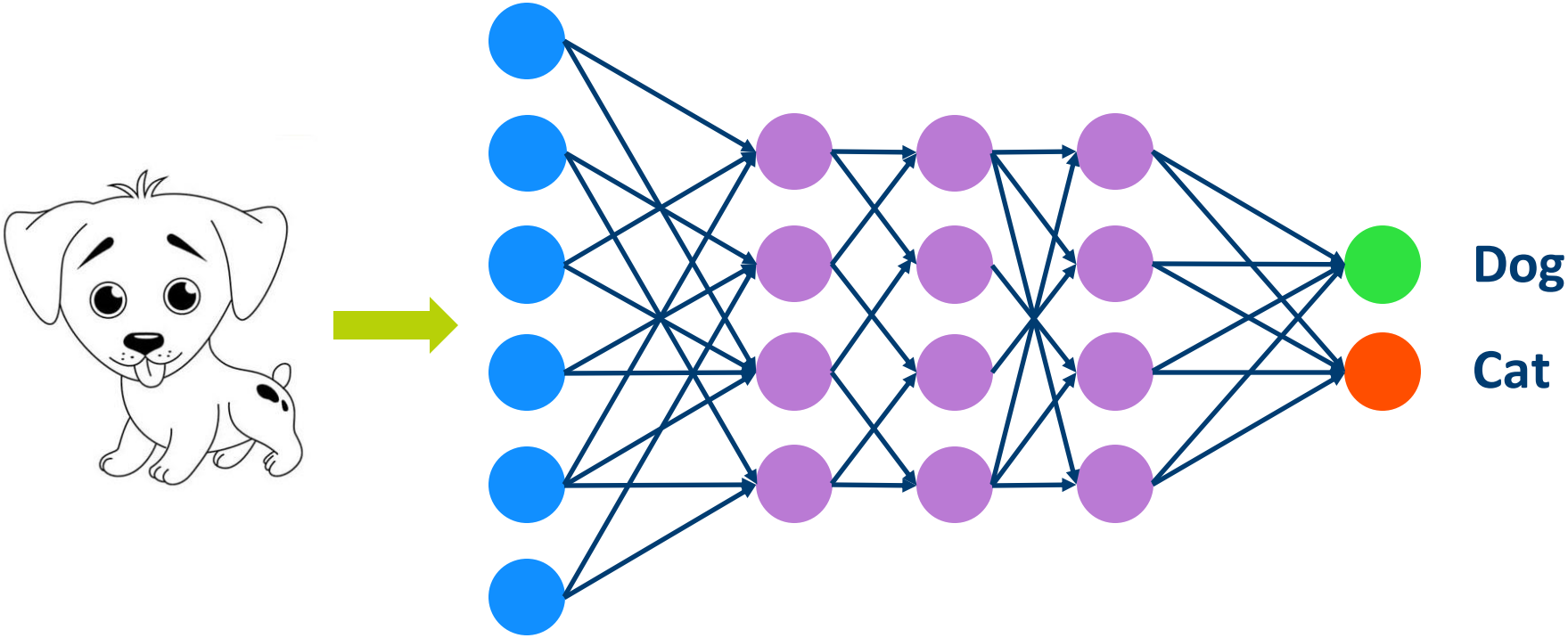
Learning



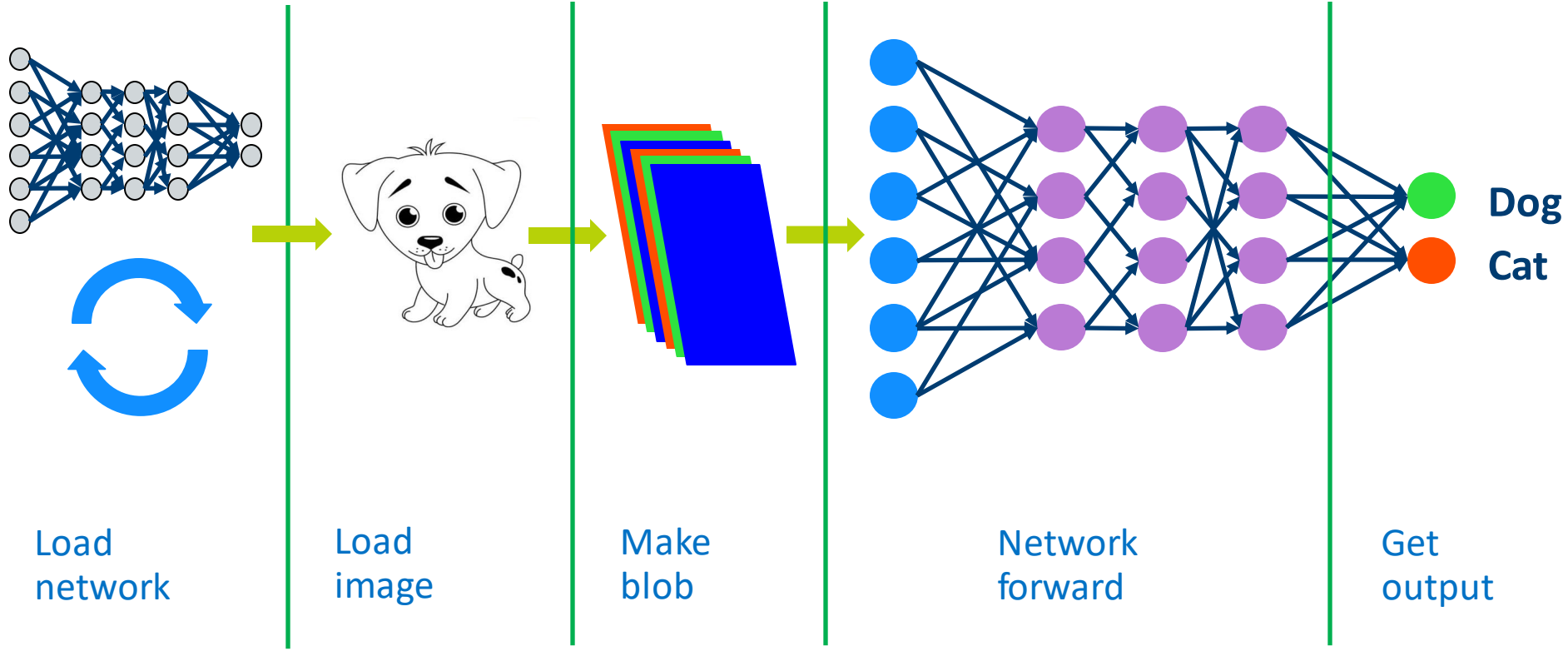
Inference



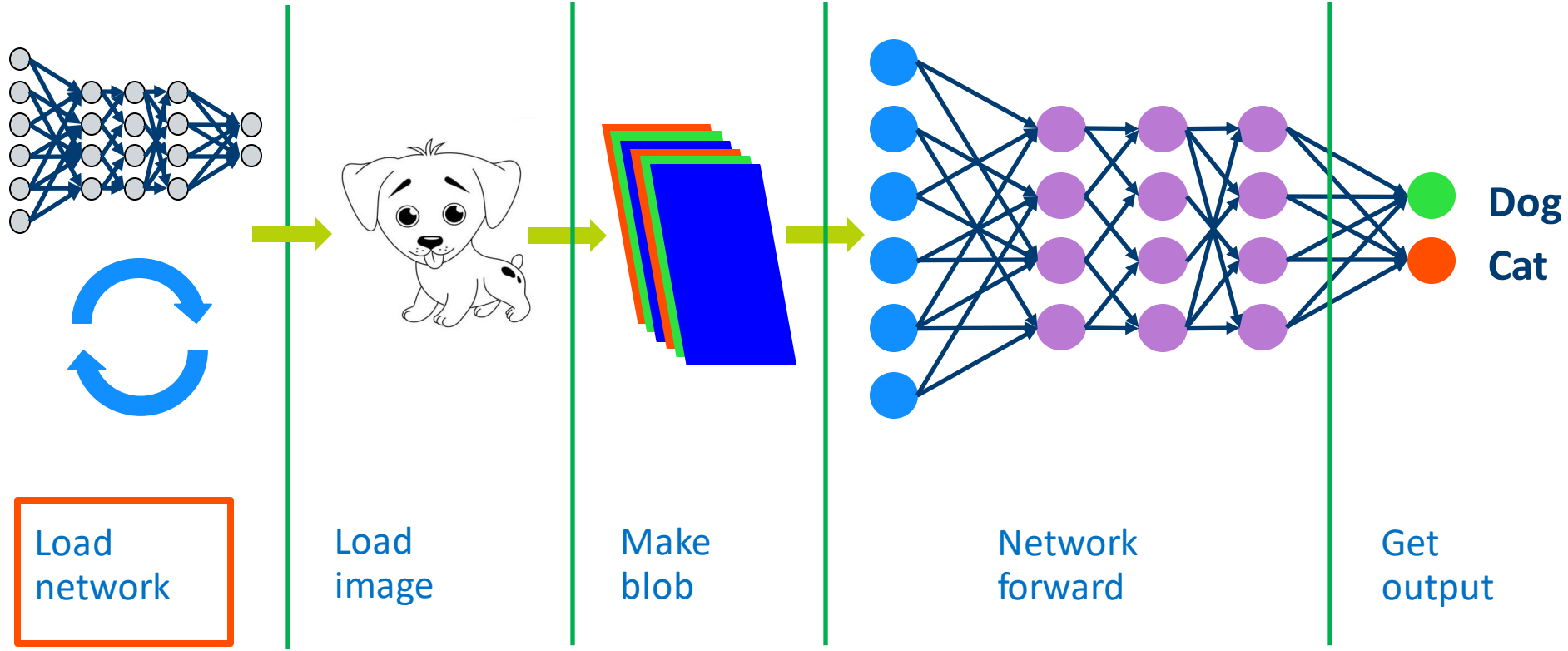
Deep neural network



Deep neural network

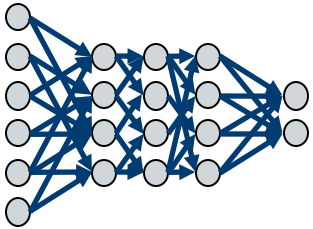


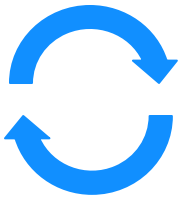
OpenCV DNN module



OpenCV DNN module

Load network



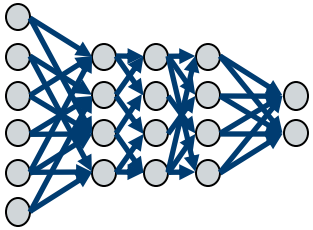


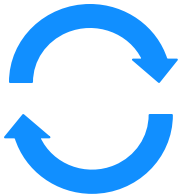
```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

Load network

OpenCV DNN module

Load network



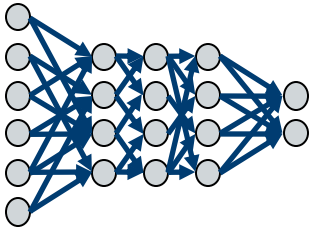


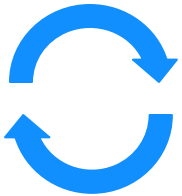
Load network

```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

OpenCV DNN module

Load network





Load network

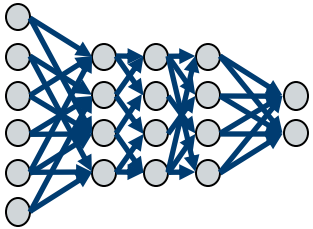
```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

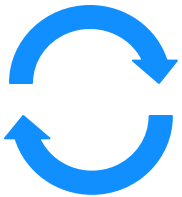
Caffe

ONNX

OpenCV DNN module

Load network

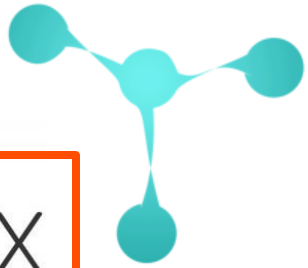




Load network

```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

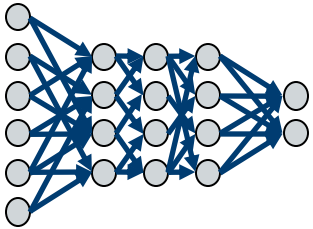
Caffe

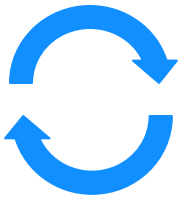


ONNX

OpenCV DNN module

Load network





Load network

```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

PYTORCH

mxnet

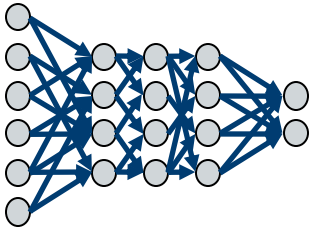
ONNX

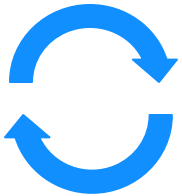
Caffe2

Microsoft CNTK

OpenCV DNN module

Load network



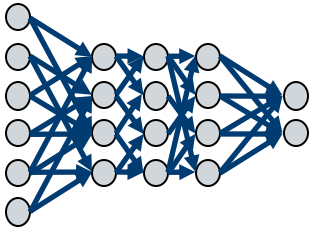


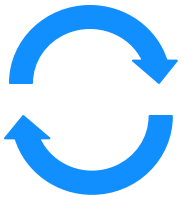
Load network

```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

OpenCV DNN module

Load network





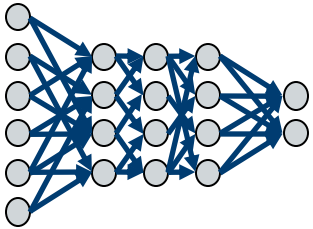
Load network

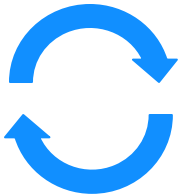
```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

```
int backendId = DNN_BACKEND_DEFAULT;
int backendId = DNN_BACKEND_OPENCV;
int backendId = DNN_BACKEND_INFERENCE_ENGINE;
int backendId = DNN_BACKEND_HALIDE;
```

OpenCV DNN module

Load network





Load network

OpenCV

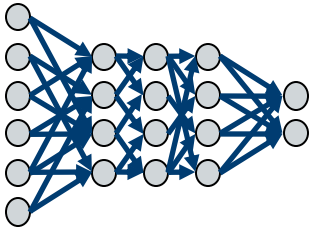
OpenVINO™

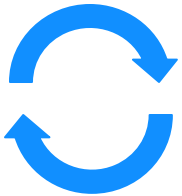
Halide

```
int backendId = DNN_BACKEND_DEFAULT;
int backendId = DNN_BACKEND_OPENCV;
int backendId = DNN_BACKEND_INFERENCE_ENGINE;
int backendId = DNN_BACKEND_HALIDE;
```

OpenCV DNN module

Load network



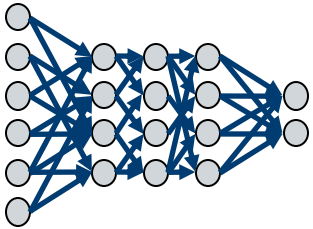


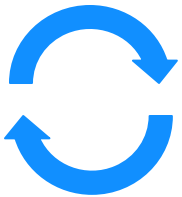
Load network

```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

OpenCV DNN module

Load network





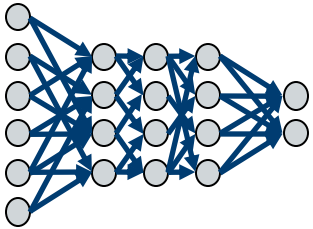
Load network

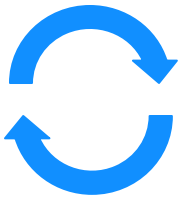
```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

```
int targetId = DNN_TARGET_CPU;
int targetId = DNN_TARGET_OPENCL;
int targetId = DNN_TARGET_OPENCL_FP16;
int targetId = DNN_TARGET_MYRIAD;
int targetId = DNN_TARGET_FPGA;
```

OpenCV DNN module

Load network





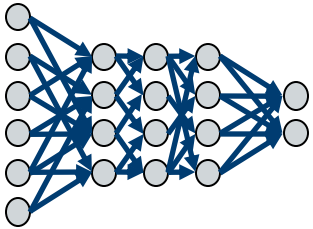
Load network

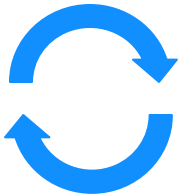
```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

```
int targetId = DNN_TARGET_CPU;
int targetId = DNN_TARGET_OPENCL;
int targetId = DNN_TARGET_OPENCL_FP16;
int targetId = DNN_TARGET_MYRIAD;
int targetId = DNN_TARGET_FPGA;
```

OpenCV DNN module

Load network





Load network

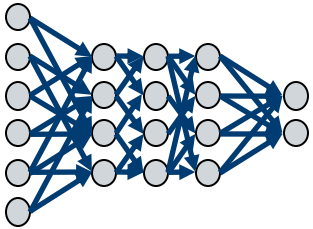
```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

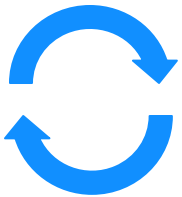
OpenVINO™

```
int targetId = DNN_TARGET_CPU;
int targetId = DNN_TARGET_OPENCL;
int targetId = DNN_TARGET_OPENCL_FP16;
int targetId = DNN_TARGET_MYRIAD;
int targetId = DNN_TARGET_FPGA;
```

OpenCV DNN module

Load network





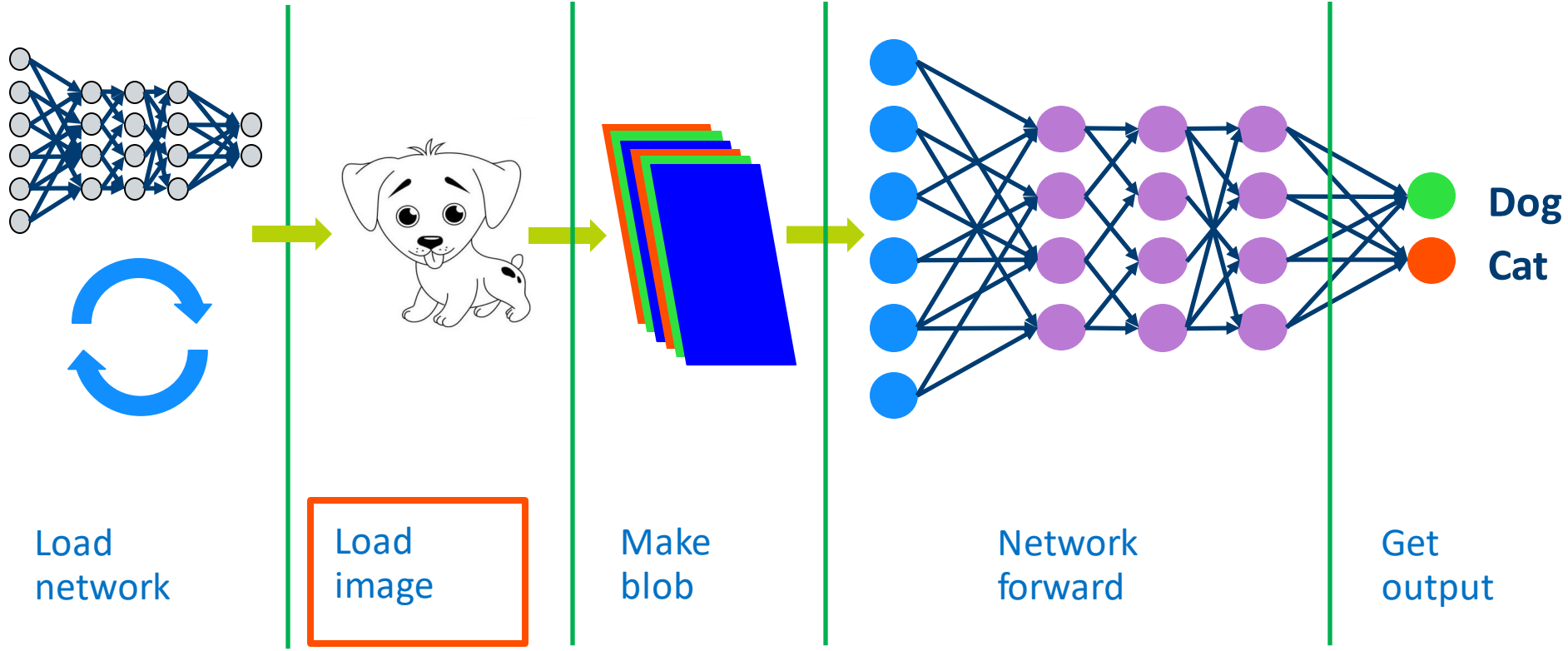
Load network

```
Net net = readNet(model, config);
net.setPreferableBackend(backendId);
net.setPreferableTarget(targetId);
```

```
int targetId  = DNN_TARGET_CPU;
int targetId  = DNN_TARGET_OPENCL;
int targetId  = DNN_TARGET_OPENCL_FP16;
int targetId  = DNN_TARGET_MYRIAD;
int targetId  = DNN_TARGET_FPGA;
```

Halide

OpenCV DNN module



OpenCV DNN module

Load image

Load image

```
Mat frame = imread(path_image);
```

// comment

```
#include <opencv2/dnn.hpp>
#include <opencv2/highgui.hpp>
```

```
using namespace cv;
using namespace dnn;
using namespace std;
```

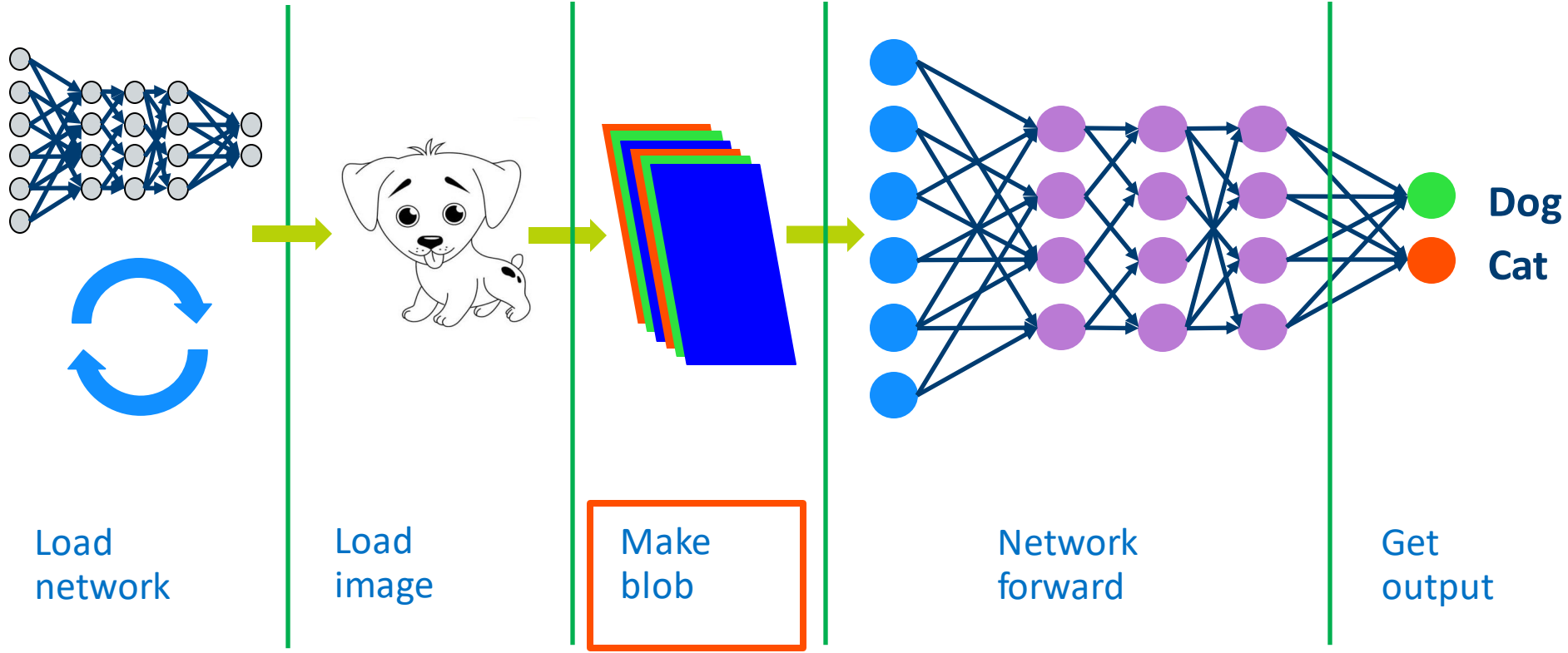
OpenCV DNN module

Load image from video

Load image

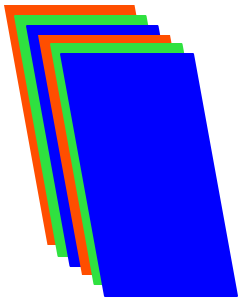
```
Mat frame;
string win_name = "Deep learning in OpenCV";
namedWindow(win_name, WINDOW_NORMAL);

VideoCapture cap;
cap.open(0);
while (waitKey(1) < 0)
{
    cap >> frame;
    imshow(win_name, frame);
}
```

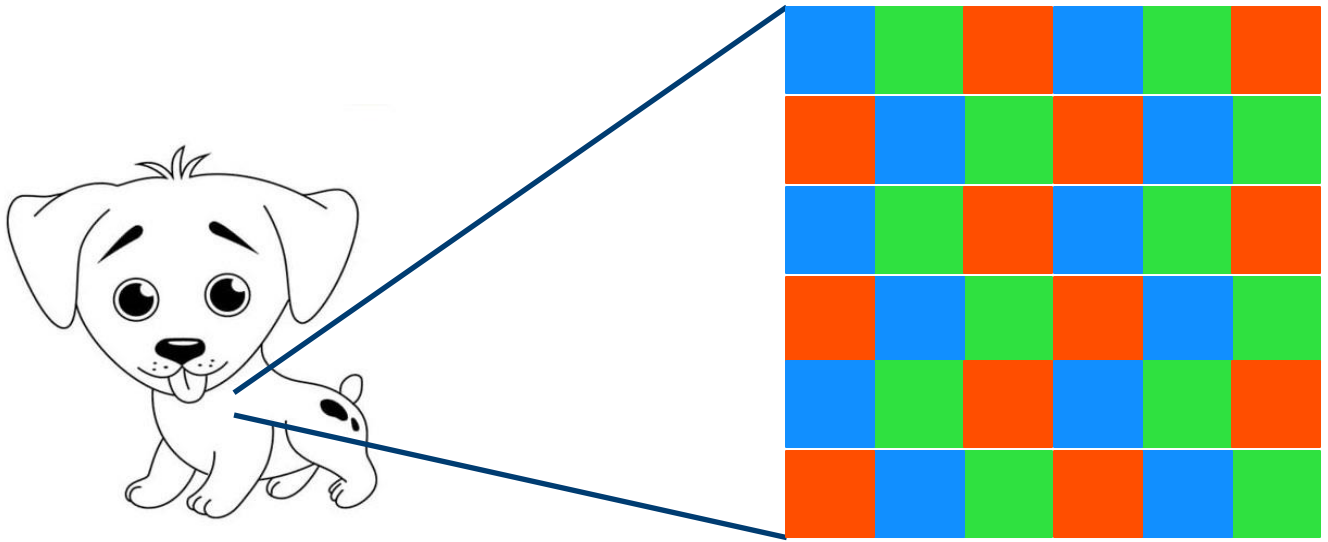
OpenCV DNN module



OpenCV DNN module

Make blob

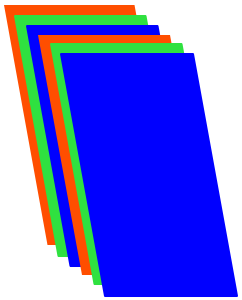




Make blob

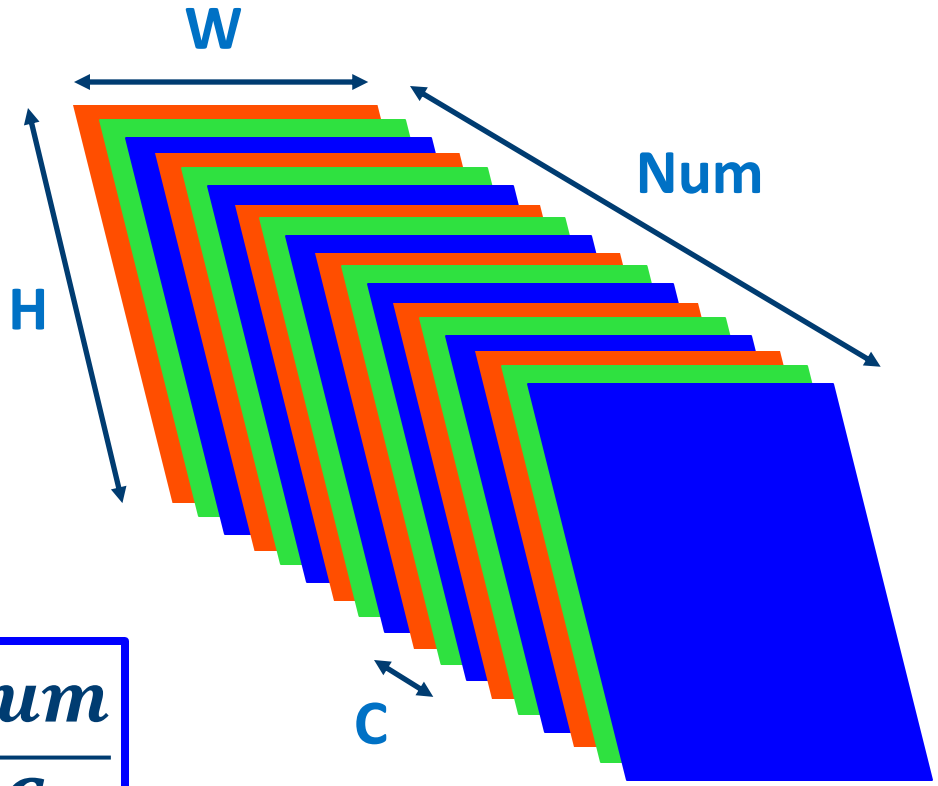
OpenCV DNN module

Make blob



Make blob

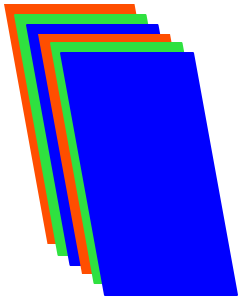
NCHW



$$N = \frac{Num}{C}$$

OpenCV DNN module

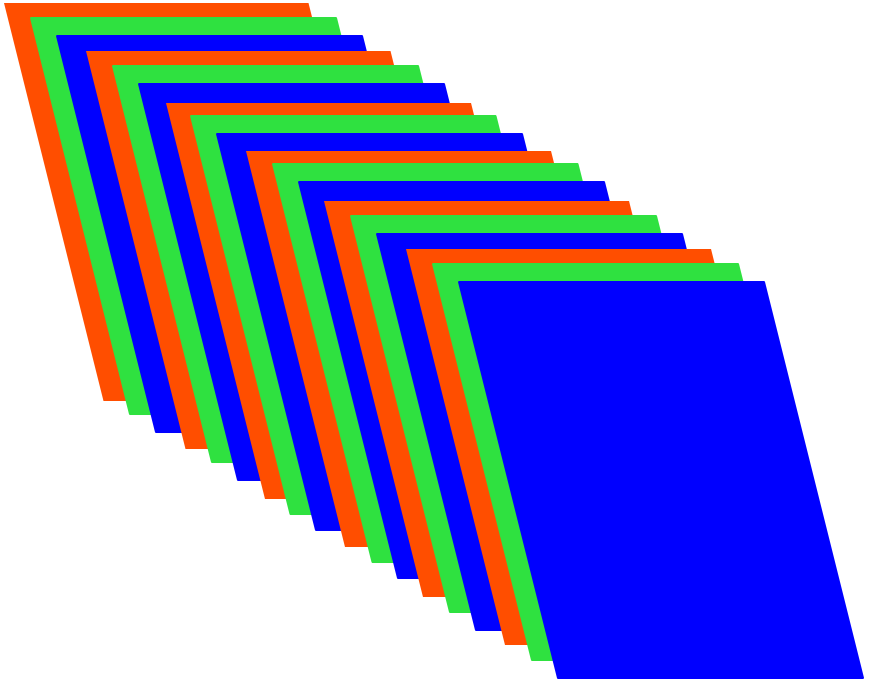
Make blob



Make blob

$[3 \times 128 \times 128]$

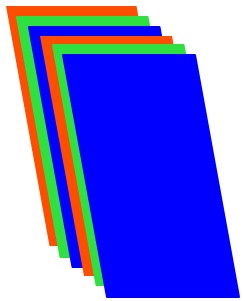
NCHW



$[1 \times 3 \times 128 \times 128]$

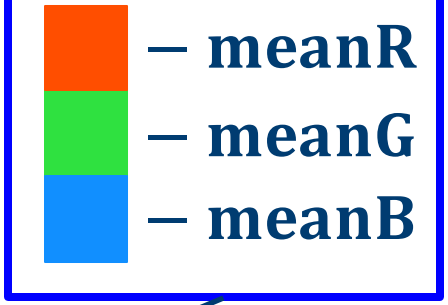
OpenCV DNN module

Make blob

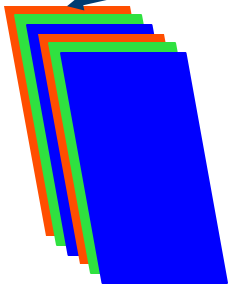


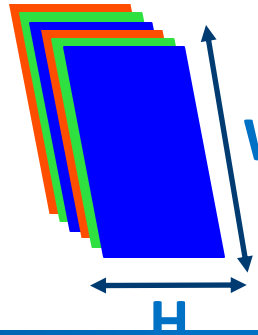
Make blob

$scale \times$



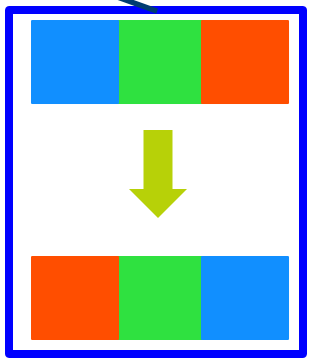

```
blobFromImage(frame, blob, scale, spatial_size, mean, swapRB, crop, ddepth);
```



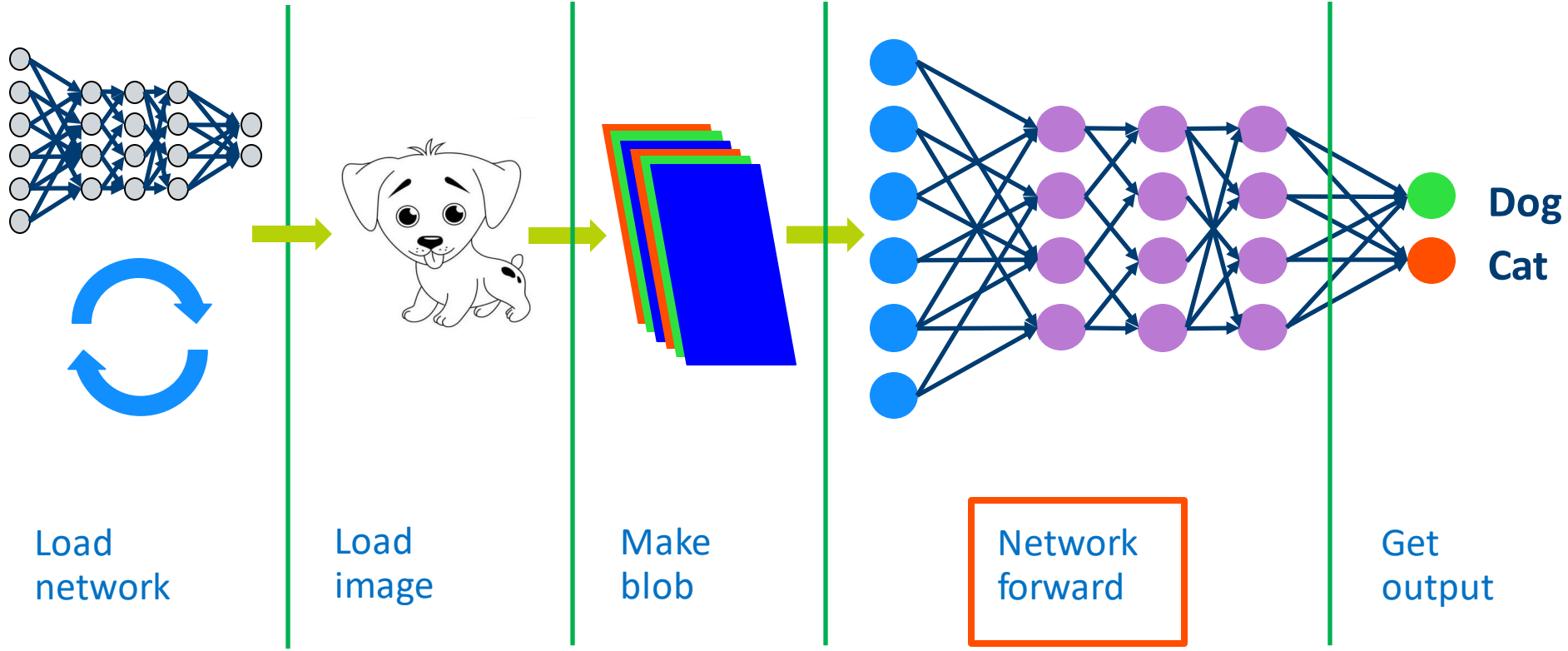


W → (H, W)

H

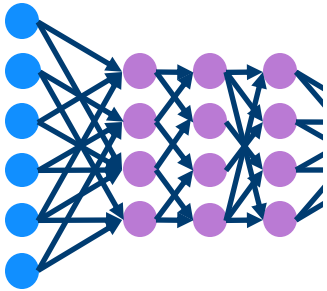


OpenCV DNN module



OpenCV DNN module

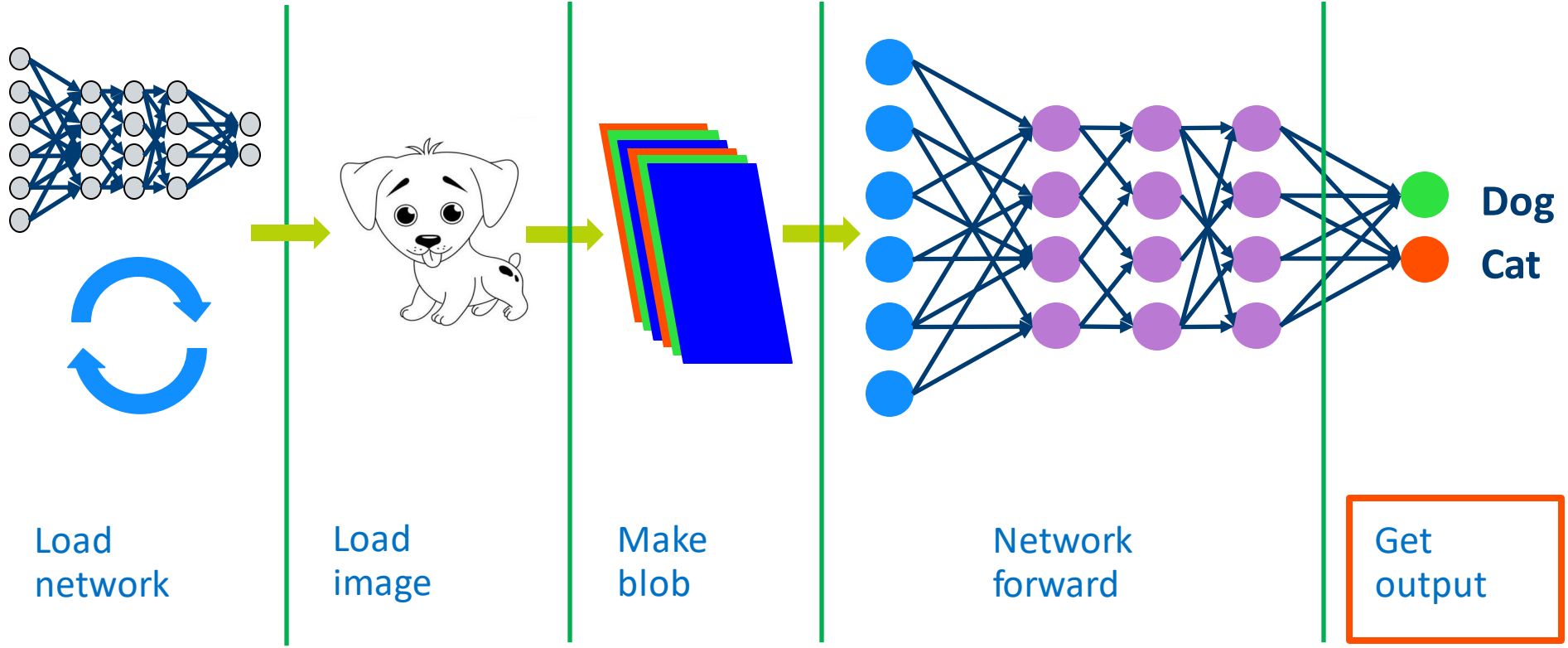
Network forward



```
net.setInput(blob);
Mat prob = net.forward();
```

Network forward

OpenCV DNN module

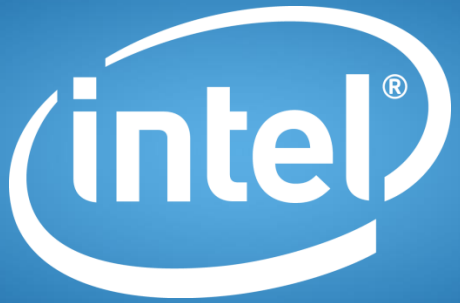


OpenCV DNN module

Get output for classification


```
Point classIdPoint;
double confidence;
minMaxLoc(prob.reshape(1, 1), 0, &confidence, 0, &classIdPoint);
int classId = classIdPoint.x;

cout << "Class: " << classId << '\n';
cout << "Confidence: " << confidence << '\n';
```

Get output


intel