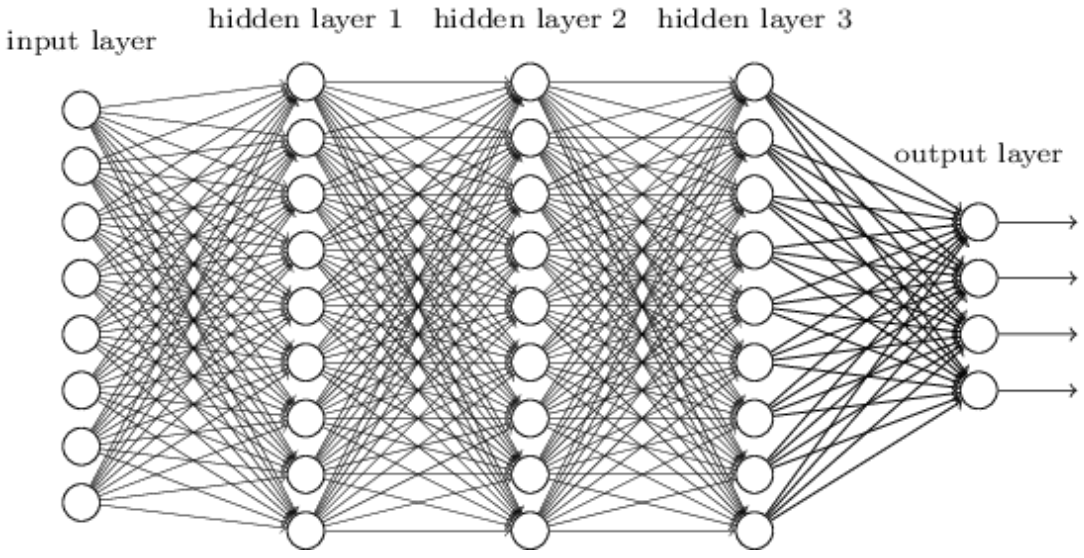
目录

Deep Neural Network module

深度神经网络模型

输入---------->黑盒----------> 输出



OpenCV 使用训练好的网络模型

https://docs.opencv.org/4.5.3/d6/d0f/group__dnn.html

OpenCV dnn module 只能执行推理（对训练好的模型的使用），不能训练。
时下 OpenCV 支持的深度学习框架和相关层：
https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV

加载网络模型

https://docs.opencv.org/4.5.3/d6/d0f/group__dnn.html#ga3b34fe7a29494a6a4295c169a7d32422

◆ readNet() [1/2]

```
Net cv::dnn::readNet ( const String &  model,
                       const String &  config = "" ,
                       const String &  framework = ""
                     )
```

Python:

```
cv.dnn.readNet( model[, config[, framework]]          ) -> retval
cv.dnn.readNet( framework, bufferModel[, bufferConfig] ) -> retval
```

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

Read deep learning network represented in one of the supported formats.

Parameters

[in] **model** Binary file contains trained weights. The following file extensions are expected for models from different frameworks:

- `*.caffemodel` (Caffe, http://caffe.berkeleyvision.org/)
- `*.pb` (TensorFlow, https://www.tensorflow.org/)
- `*.t7` | `*.net` (Torch, http://torch.ch/)
- `*.weights` (Darknet, https://pjreddie.com/darknet/)
- `*.bin` (DLDT, https://software.intel.com/openvino-toolkit)
- `*.onnx` (ONNX, https://onnx.ai/)

[in] **config** Text file contains network configuration. It could be a file with the following extensions:

- `*.prototxt` (Caffe, http://caffe.berkeleyvision.org/)
- `*.pbtxt` (TensorFlow, https://www.tensorflow.org/)
- `*.cfg` (Darknet, https://pjreddie.com/darknet/)
- `*.xml` (DLDT, https://software.intel.com/openvino-toolkit)

[in] **framework** Explicit framework name tag to determine a format.

设置计算后台：

Dnn module 支持设置不同的计算后台，在不同设备上进行。

setPreferableBackend()

void cv::dnn::Net::setPreferableBackend (int backendId)

Python:

cv.dnn_Net.setPreferableBackend(backendId) -> None

Ask network to use specific computation backend where it supported.

Parameters

[in] **backendId** backend identifier.

See also

Backend

If OpenCV is compiled with Intel's Inference Engine library, DNN_BACKEND_DEFAULT means DNN_BACKEND_INFERENCE_ENGINE. Otherwise it equals to DNN_BACKEND_OPENCV.

setPreferableTarget()

void cv::dnn::Net::setPreferableTarget (int targetId)

Python:

cv.dnn_Net.setPreferableTarget(targetId) -> None

Ask network to make computations on specific target device.

Parameters

[in] **targetId** target identifier.

See also

Target

List of supported combinations backend / target:

	DNN_BACKEND_OPENCV	DNN_BACKEND_INFERENCE_ENGINE	DNN_BACKEND_HALIDE	DNN_BACKEND_CUDA
DNN_TARGET_CPU	+	+	+	
DNN_TARGET_OPENCL	+	+	+	
DNN_TARGET_OPENCL_FP16	+	+		
DNN_TARGET_MYRIAD		+		
DNN_TARGET_FPGA		+		
DNN_TARGET_CUDA				+
DNN_TARGET_CUDA_FP16				+
DNN_TARGET_HDDL		+		

Examples:

samples/dnn/colorization.cpp.

设置使用 CPU 和 OpenCV 作为计算后台。

构建输入

blobFromImage：

blobFromImage() [1/2]

```
Mat cv::dnn::blobFromImage ( InputArray  image,
                             double      scalefactor = 1.0 ,
                             const Size &  size = Size() ,
                             const Scalar &  mean = Scalar() ,
                             bool        swapRB = false ,
                             bool        crop = false ,
                             int         ddepth = CV_32F
                           )
```

Python:

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

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

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 `image` has BGR ordering and `swapRB` is true.
scalefactor	multiplier for `image` values.
swapRB	flag which indicates that swap first and last channels in 3-channel image is necessary.
crop	flag which indicates whether image will be cropped after resize or not
ddepth	Depth of output blob. Choose CV_32F or CV_8U.

if `crop` is true, input image is resized so one side after resize is equal to corresponding dimension in `size` and another one is equal or larger. Then, crop from the center is performed. If `crop` is false, direct resize without cropping and preserving aspect ratio is performed.

Returns

4-dimensional **Mat** with NCHW dimensions order.

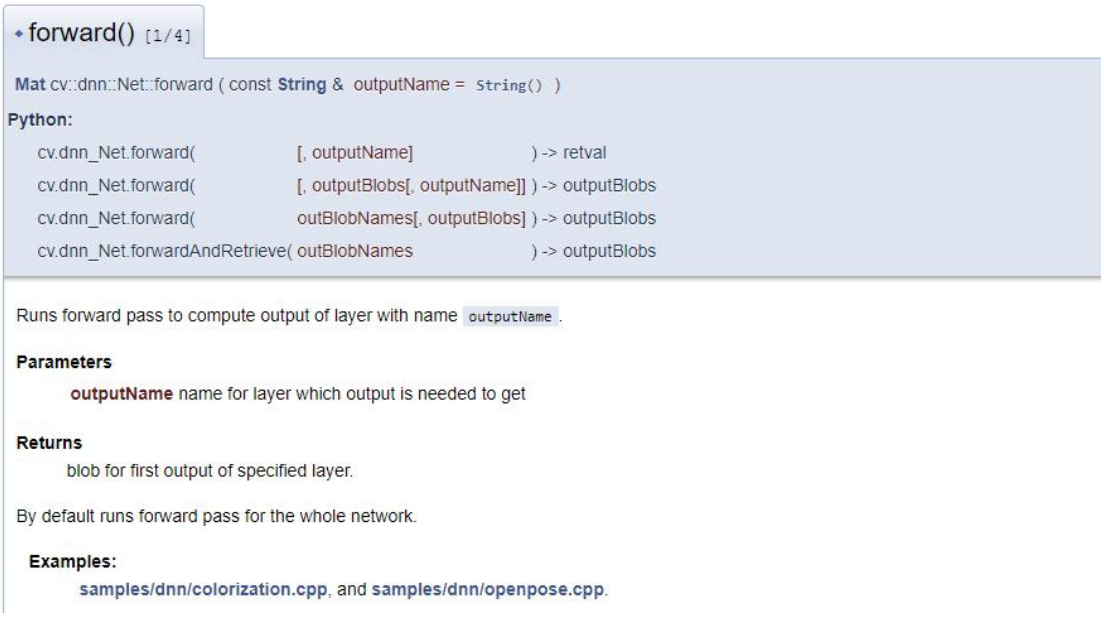
Examples:

samples/dnn/classification.cpp, **samples/dnn/colorization.cpp**, **samples/dnn/object_detection.cpp**, **samples/dnn/openpose.cpp**, and **samples/dnn/segmentation.cpp**.

執行推理 inference

Net::forward :
https://docs.opencv.org/4.5.3/db/d30/classcv_1_1dnn_1_1Net.html

Mat	**forward** (const **String** &outputName=**String**()) Runs forward pass to compute output of layer with name `outputName`. More...
void	**forward** (**OutputArrayOfArrays** outputBlobs, const **String** &outputName=**String**()) Runs forward pass to compute output of layer with name `outputName`. More...
void	**forward** (**OutputArrayOfArrays** outputBlobs, const std::vector< **String** > &outBlobNames) Runs forward pass to compute outputs of layers listed in `outBlobNames`. More...
void	**forward** (std::vector< std::vector< **Mat** > > &outputBlobs, const std::vector< **String** > &outBlobNames) Runs forward pass to compute outputs of layers listed in `outBlobNames`. More...



• forward() [1/4]

Mat cv::dnn::Net::forward (const String & outputName = String())

Python:

cv.dnn_Net.forward([, outputName]) -> retval
cv.dnn_Net.forward([, outputBlobs[, outputName]]) -> outputBlobs
cv.dnn_Net.forward(outBlobNames[, outputBlobs]) -> outputBlobs
cv.dnn_Net.forwardAndRetrieve(outBlobNames) -> outputBlobs

Runs forward pass to compute output of layer with name outputName.

Parameters

outputName name for layer which output is needed to get

Returns

blob for first output of specified layer.

By default runs forward pass for the whole network.

Examples:

samples/dnn/colorization.cpp, and samples/dnn/openpose.cpp.

解析输出

```````

## SSD(Single Shot MultiBox Detector)

模型描述文件:
https://github.com/opencv/opencv/blob/master/samples/dnn/models.yml

# SSD_caffe

输入：[Nx3x300x300]
通道顺序:RGB

输出：[1x1xNx7], 7 对应的浮点数据对应如下：
[image_id,label, conf, x_min, y_min, x_max, y_max]

N:batchsize
```````

```
# Caffe implementation of SSD model from https://github.com/chuanqi305/MobileNet-SSD
ssd_caffe:
  load_info:
    url: "https://drive.google.com/uc?export=download&id=0B3gersZ2cHIxRm5PMWRoTkdHdHc"
    sha1: "994d30a8afaa9e754d17d2373b2d62a7dfbaaf7a"
  model: "MobileNetSSD_deploy.caffemodel"
  config: "MobileNetSSD_deploy.prototxt"
  mean: [127.5, 127.5, 127.5]
  scale: 0.007843
  width: 300
  height: 300
  rgb: false
  classes: "object_detection_classes_pascal_voc.txt"
  sample: "object_detection"
```

rgb 为 false，表示格式正确(BGR)，不需要更改为 rgb

PASCAL VOC 数据集

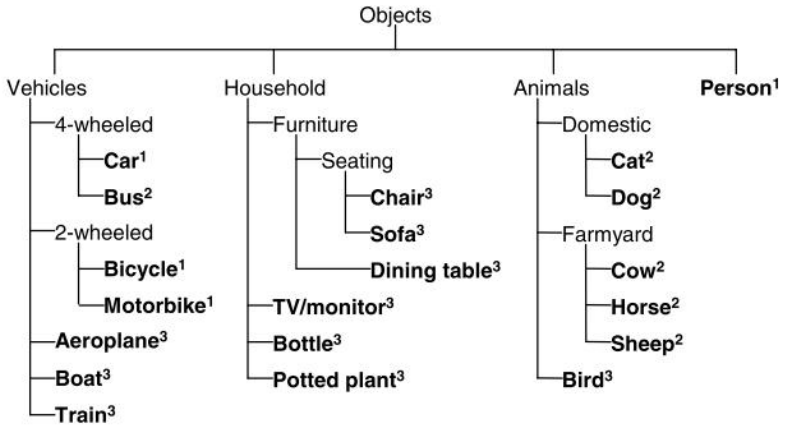


Fig. 2 VOC2007 Classes. Leaf nodes correspond to the 20 classes. The year of inclusion of each class in the challenge is indicated by superscripts: 2005[1], 2006[2], 2007[3]. The classes can be considered in a notional taxonomy, with successive challenges adding new branches (increasing the domain) and leaves (increasing detail)

文件下载:

MobileNetSSD_deploy.caffemodel:
https://drive.google.com/uc?export=download&id=0B3gersZ2cHIxRm5PMWRoTkdHdHc
MobileNetSSD_deploy.prototxt:
https://raw.githubusercontent.com/chuanqi305/MobileNet-SSD/master/voc/MobileNetSSD_deploy.prototxt
Classes:
https://raw.githubusercontent.com/opencv/opencv/master/samples/data/dnn/object_detection_classes_pascal_voc.txt

TensorFlow Object Detection API

https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API

下载使用已经训练好的模型

Use existing config file for your model

You can use one of the configs that has been tested in OpenCV. This choice depends on your model and TensorFlow version:

Model	Version		
MobileNet-SSD v1	2017_11_17	weights	config
MobileNet-SSD v1 PPN	2018_07_03	weights	config
MobileNet-SSD v2	2018_03_29	weights	config
Inception-SSD v2	2017_11_17	weights	config
MobileNet-SSD v3 (see #16760)	2020_01_14	weights	config
Faster-RCNN Inception v2	2018_01_28	weights	config
Faster-RCNN ResNet-50	2018_01_28	weights	config
Mask-RCNN Inception v2	2018_01_28	weights	config
EfficientDet-D0 (see #17384)		weights	config

注意事项：下载 config 文件时候要点击 Raw

手动生成 config 文件

如果官方没有提供模型对应的 config，对于自定义模型，可以手动生成对应的 pbtxt 文件.

```
(36) D:\opencv\sources\samples\dnn>python tf_text_graph_ssd.py --input C:\Users\wx\Desktop\opencv_dnn\OpenCV_dnn\models\ssd_mobilenet
_v2_coco_2018_03_29/frozen_inference_graph.pb --config C:\Users\wx\Desktop\opencv_dnn\OpenCV_dnn\models\ssd_mobilenet_v2_coco_2018_03
_29/pipeline.config --output C:\Users\wx\Desktop\opencv_dnn\OpenCV_dnn\models\ssd_mobilenet_v2_coco_2018_03_29/graph.pbtxt
Scale: [0.200000-0.950000]
Aspect ratios: [1.0, 2.0, 0.5, 3.0, 0.333299994469]
Reduce boxes in the lowest layer: True
Number of classes: 90
Number of layers: 6
box predictor: convolutional
Input image size: 300x300
```