Deep Neural Network module

Modules

Classes

class **cv::dnn::BackendNode**
Derivatives of this class encapsulates functions of certain backends. More...

class **cv::dnn::BackendWrapper**
Derivatives of this class wraps **cv::Mat** for different backends and targets. More...

class **cv::dnn::Dict**
This class implements name-value dictionary, values are instances of **DictValue**. More...

struct **cv::dnn::DictValue**
This struct stores the scalar value (or array) of one of the following type: double, **cv::String** or int64. More...

class **cv::dnn::Importer**
Small interface class for loading trained serialized models of different dnn-frameworks. More...

class **cv::dnn::Layer**
This interface class allows to build new Layers - are building blocks of networks. More...

class **cv::dnn::LayerParams**
This class provides all data needed to initialize layer. More...

class **cv::dnn::Net**
This class allows to create and manipulate comprehensive artificial neural networks. More...

Typedefs

typedef std::vector< int > **cv::dnn::MatShape**

Enumerations

enum **cv::dnn::Backend** {
cv::dnn::DNN_BACKEND_DEFAULT,
cv::dnn::DNN_BACKEND_HALIDE
}
Enum of computation backends supported by layers. More...

enum **cv::dnn::Target** {
cv::dnn::DNN_TARGET_CPU,
cv::dnn::DNN_TARGET_OPENCL
}
Enum of target devices for computations. More...

Functions

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

Mat **cv::dnn::blobFromImages** (const std::vector< **Mat** > &images, double scalefactor=1.0, **Size** size=**Size**(), const **Scalar** &**mean**=**Scalar**(), bool swapRB=true, bool crop=true)
Creates 4-dimensional blob from series of images. Optionally resizes and crops `images` from center, subtract `mean` values, scales values by `scalefactor`, swap Blue and Red channels. More...

Ptr< **Importer** > **cv::dnn::createCaffeImporter** (const **String** &prototxt, const **String** &caffeModel=**String**())
Creates the importer of Caffe framework network. More...

Ptr< **Importer** > **cv::dnn::createTensorflowImporter** (const **String** &model)
Creates the importer of TensorFlow framework network. More...

Ptr< **Importer** > **cv::dnn::createTorchImporter** (const **String** &filename, bool isBinary=true)
Creates the importer of Torch7 framework network. More...

void **cv::dnn::NMSBoxes** (const std::vector< **Rect** > &bboxes, const std::vector< float > &scores, const float score_threshold, const float

	nms_threshold, std::vector< int > &indices, const float eta=1.f, const int top_k=0) Performs non maximum suppression given boxes and corresponding scores. More...
Net	**cv::dnn::readNetFromCaffe** (const **String** &prototxt, const **String** &caffeModel=**String**()) Reads a network model stored in Caffe model files. More...
Net	**cv::dnn::readNetFromCaffe** (const char *bufferProto, size_t lenProto, const char *bufferModel=NULL, size_t lenModel=0) Reads a network model stored in Caffe model in memory. More...
Net	**cv::dnn::readNetFromDarknet** (const **String** &cfgFile, const **String** &darknetModel=**String**()) Reads a network model stored in Darknet model files. More...
Net	**cv::dnn::readNetFromTensorflow** (const **String** &model, const **String** &config=**String**()) Reads a network model stored in Tensorflow model file. More...
Net	**cv::dnn::readNetFromTensorflow** (const char *bufferModel, size_t lenModel, const char *bufferConfig=NULL, size_t lenConfig=0) Reads a network model stored in Tensorflow model in memory. More...
Net	**cv::dnn::readNetFromTorch** (const **String** &model, bool isBinary=true) Reads a network model stored in Torch model file. More...
Mat	**cv::dnn::readTorchBlob** (const **String** &filename, bool isBinary=true) Loads blob which was serialized as torch.Tensor object of Torch7 framework. More...
void	**cv::dnn::shrinkCaffeModel** (const **String** &src, const **String** &dst, const std::vector< **String** > &layersTypes=std::vector< **String** >()) Convert all weights of Caffe network to half precision floating point. More...

Detailed Description

This module contains:

- API for new layers creation, layers are building bricks of neural networks;
- set of built-in most-useful Layers;
- API to constuct and modify comprehensive neural networks from layers;
- functionality for loading serialized networks models from differnet frameworks.

Functionality of this module is designed only for forward pass computations (i. e. network testing). A network training is in principle not supported.

Typedef Documentation

§ MatShape

typedef std::vector<int> **cv::dnn::MatShape**

Enumeration Type Documentation

§ Backend

enum **cv::dnn::Backend**

Enum of computation backends supported by layers.

Enumerator	
DNN_BACKEND_DEFAULT	
DNN_BACKEND_HALIDE	

§ Target

enum **cv::dnn::Target**

Enum of target devices for computations.

Enumerator	
DNN_TARGET_CPU	
DNN_TARGET_OPENCL	

Function Documentation

§ blobFromImage()

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

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

if `crop` is true, input image is resized so one side after resize is equal to corresponing 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-dimansional **Mat** with NCHW dimensions order.

§ blobFromImages()

```
Mat cv::dnn::blobFromImages ( const std::vector< Mat > &  images,
                              double                      scalefactor = 1.0,
                              Size                        size = Size(),
                              const Scalar &              mean = Scalar(),
                              bool                        swapRB = true,
                              bool                        crop = true
                            )
```

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

Parameters

images	input images (all 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 `images` 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

if `crop` is true, input image is resized so one side after resize is equal to corresponing 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-dimansional **Mat** with NCHW dimensions order.

§ createCaffeImporter()

Ptr<Importer> cv::dnn::createCaffeImporter (const **String** & prototxt,
const **String** & caffeModel = `String()`
)

Creates the importer of Caffe framework network.

Deprecated:
Use **readNetFromCaffe** instead.

Parameters

prototxt	path to the .prototxt file with text description of the network architecture.
caffeModel	path to the .caffemodel file with learned network.

Returns
Pointer to the created importer, NULL in failure cases.

§ createTensorflowImporter()

Ptr<Importer> cv::dnn::createTensorflowImporter (const **String** & model)

Creates the importer of TensorFlow framework network.

Deprecated:

Use **readNetFromTensorflow** instead.

Parameters

model path to the .pb file with binary protobuf description of the network architecture.

Returns

Pointer to the created importer, NULL in failure cases.

§ createTorchImporter()

```
Ptr<Importer> cv::dnn::createTorchImporter ( const String &  filename,
                                             bool             isBinary = true
                                           )
```

Creates the importer of Torch7 framework network.

Deprecated:

Use **readNetFromTorch** instead.

Parameters

filename path to the file, dumped from Torch by using torch.save() function.

isBinary specifies whether the network was serialized in ascii mode or binary.

Returns

Pointer to the created importer, NULL in failure cases.

Warning

Torch7 importer is experimental now, you need explicitly set CMake `opencv_dnn_BUILD_TORCH_IMPORTER` flag to compile its.

Note

Ascii mode of Torch serializer is more preferable, because binary mode extensively use `long` type of C language, which has various bit-length on different systems.

The loading file must contain serialized nn.Module object with importing network. Try to eliminate a custom objects from serialazing data to avoid importing errors.

List of supported layers (i.e. object instances derived from Torch nn.Module class):

- nn.Sequential
- nn.Parallel
- nn.Concat
- nn.Linear
- nn.SpatialConvolution

- nn.SpatialMaxPooling, nn.SpatialAveragePooling
- nn.ReLU, nn.TanH, nn.Sigmoid
- nn.Reshape
- nn.SoftMax, nn.LogSoftMax

Also some equivalents of these classes from cunn, cudnn, and fbcunn may be successfully imported.

§ NMSBoxes()

```
void cv::dnn::NMSBoxes ( const std::vector< Rect > &  bboxes,
                         const std::vector< float > &  scores,
                         const float                   score_threshold,
                         const float                   nms_threshold,
                         std::vector< int > &          indices,
                         const float                   eta = 1.f,
                         const int                     top_k = 0
                       )
```

Performs non maximum suppression given boxes and corresponding scores.

Parameters

bboxes	a set of bounding boxes to apply NMS.
scores	a set of corresponding confidences.
score_threshold	a threshold used to filter boxes by score.
nms_threshold	a threshold used in non maximum suppression.
indices	the kept indices of bboxes after NMS.
eta	a coefficient in adaptive threshold formula: $nms\_threshold_{i+1} = eta \cdot nms\_threshold_i$.
top_k	if `>0`, keep at most `top_k` picked indices.

§ readNetFromCaffe() [1/2]

```
Net cv::dnn::readNetFromCaffe ( const String &  prototxt,
                                const String &  caffeModel = String()
                              )
```

Reads a network model stored in Caffe model files.

This is shortcut consisting from createCaffeImporter and Net::populateNet calls.

§ readNetFromCaffe() [2/2]

```
Net cv::dnn::readNetFromCaffe ( const char *  bufferProto,
                                size_t        lenProto,
                                              bufferModel =
                                const char *  NULL,
                                size_t        lenModel = 0
                              )
```

Reads a network model stored in Caffe model in memory.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

bufferProto	buffer containing the content of the .prototxt file
lenProto	length of bufferProto
bufferModel	buffer containing the content of the .caffemodel file
lenModel	length of bufferModel

§ readNetFromDarknet()

```
Net cv::dnn::readNetFromDarknet ( const String &  cfgFile,
                                  const String &  darknetModel = String()
                                )
```

Reads a network model stored in Darknet model files.

Parameters

cfgFile	path to the .cfg file with text description of the network architecture.
darknetModel	path to the .weights file with learned network.

Returns

Network object that ready to do forward, throw an exception in failure cases.

This is shortcut consisting from DarknetImporter and Net::populateNet calls.

§ readNetFromTensorflow() [1/2]

```
Net cv::dnn::readNetFromTensorflow ( const String &  model,
                                     const String &  config = String()
                                   )
```

Reads a network model stored in Tensorflow model file.

This is shortcut consisting from createTensorflowImporter and Net::populateNet calls.

§ readNetFromTensorflow() [2/2]

```
Net cv::dnn::readNetFromTensorflow ( const char *  bufferModel,
                                     size_t        lenModel,
                                     bufferConfig =
                                     const char *  NULL,
                                     size_t        lenConfig = 0
                                   )
```

Reads a network model stored in Tensorflow model in memory.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

bufferModel	buffer containing the content of the pb file
lenModel	length of bufferModel
bufferConfig	buffer containing the content of the pbtxt file
lenConfig	length of bufferConfig

§ readNetFromTorch()

```
Net cv::dnn::readNetFromTorch ( const String &  model,
                                bool            isBinary = true
                              )
```

Reads a network model stored in Torch model file.

This is shortcut consisting from createTorchImporter and Net::populateNet calls.

§ readTorchBlob()

```
Mat cv::dnn::readTorchBlob ( const String &  filename,
                             bool            isBinary = true
                           )
```

Loads blob which was serialized as torch.Tensor object of Torch7 framework.

Warning

This function has the same limitations as **createTorchImporter()**.

§ shrinkCaffeModel()

```
void cv::dnn::shrinkCaffeModel ( const String &                 src,
                                 const String &                 dst,
                                 const std::vector< String > &  layersTypes = std::vector< String >()
                               )
```

Convert all weights of Caffe network to half precision floating point.

Parameters

src	Path to origin model from Caffe framework contains single precision floating point weights (usually has `.caffemodel` extension).
dst	Path to destination model with updated weights.
layersTypes	Set of layers types which parameters will be converted. By default, converts only Convolutional and Fully-Connected layers' weights.

Note

Shrinked model has no origin float32 weights so it can't be used in origin Caffe framework anymore. However the structure of data is taken from NVidia's Caffe fork: https://github.com/NVIDIA/caffe. So the resulting model may be used there.