Age/Gender Recognition

Generated by Doxygen 1.8.11

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 cva Namespace Reference

Namespaces

- agr

4.2 cva::agr Namespace Reference

Classes

- class FaceAnalyzer

  The abstract class for DL-based face analyzer (i.e. age and gender classification tool).
- class Version

  The class is used to represent the version number for the library.

Enumerations

- enum BackendType { BackendType::CPU, BackendType::GPU, BackendType::VPU }

  The enum representing possible HW backends used for computations.
- enum Gender { Gender::FEMALE, Gender::MALE, Gender::UNKNOWN = -1 }

Functions

- Version version ()

  Get version number for the library.

4.2.1 Enumeration Type Documentation

4.2.1.1 enum cva::agr::BackendType [strong]

The enum representing possible HW backends used for computations.

Enumerator

CPU

GPU

VPU

4.2.1.2 enum cva::agr::Gender [strong]

Enumerator

FEMALE

MALE

UNKNOWN

4.2.2 Function Documentation

4.2.2.1 Version cva::agr::version ( )

Get version number for the library.

Chapter 5

Class Documentation

5.1 cva::agr::FaceAnalyzer Class Reference

The abstract class for DL-based face analyzer (i.e. age and gender classification tool).

`#include <agr.hpp>`

Classes

- struct FaceAttributes

  The structure describing age and gender of some object.
- struct Parameters

  The structure describing model for the inference process.

Public Member Functions

- virtual void process (const std::vector< cv::Mat > &imgs, std::vector< FaceAttributes > &attributes)=0

  Analyze a vector of images containing only faces.
- virtual size_t batchSize ()=0

  Get maximum number of faces that can be processed simultaneously.
- virtual ~FaceAnalyzer ()

  A virtual destructor for the abstract class.

Static Public Member Functions

- static std::unique_ptr< FaceAnalyzer > create (const Parameters &params)

  Create an instance of NN-based face analyzer.

5.1.1 Detailed Description

The abstract class for DL-based face analyzer (i.e. age and gender classification tool).

5.1.2 Constructor & Destructor Documentation

5.1.2.1 virtual cva::agr::FaceAnalyzer::~FaceAnalyzer ( ) [inline], [virtual]

A virtual destructor for the abstract class.

5.1.3 Member Function Documentation

5.1.3.1 virtual size_t cva::agr::FaceAnalyzer::batchSize ( ) [pure virtual]

Get maximum number of faces that can be processed simultaneously.

Returns

Maximum number of faces that can be processed simultaneously (i.e. by one call to the inference engine).

5.1.3.2 static std::unique_ptr<FaceAnalyzer> cva::agr::FaceAnalyzer::create (const Parameters & *params*) [static]

Create an instance of NN-based face analyzer.

Parameters

in	*params*	Structure containing information about topology and weights. For VPU backend batch_size is ignored.

Returns

Pointer to the new object.

Examples:

main.cpp.

5.1.3.3 virtual void cva::agr::FaceAnalyzer::process (const std::vector< cv::Mat > & *imgs*, std::vector< FaceAttributes > & *attributes*) [pure virtual]

Analyze a vector of images containing only faces.

Parameters

in	*imgs*	Images to be processed.
out	*attributes*	Results of analysis, i.e. a vector of a structures containing ages and genders.

The documentation for this class was generated from the following file:

- agr.hpp

5.2 cva::agr::FaceAnalyzer::FaceAttributes Struct Reference

The structure describing age and gender of some object.

`#include <agr.hpp>`

Public Member Functions

- FaceAttributes (float age=std::numeric_limits< float >::signaling_NaN(), Gender gender=Gender::UNKNOWN)

  Default constructor for the structure.

Public Attributes

- float age

  Age, in years.
- Gender gender

  Gender.

5.2.1 Detailed Description

The structure describing age and gender of some object.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 cva::agr::FaceAnalyzer::FaceAttributes::FaceAttributes (float *age* = `std::numeric_limits<float>::signaling_NaN()`, Gender *gender* = Gender::UNKNOWN) [inline], [explicit]

Default constructor for the structure.

5.2.3 Member Data Documentation

5.2.3.1 float cva::agr::FaceAnalyzer::FaceAttributes::age

Age, in years.

5.2.3.2 Gender cva::agr::FaceAnalyzer::FaceAttributes::gender

Gender.

The documentation for this struct was generated from the following file:

- agr.hpp

5.3 cva::agr::FaceAnalyzer::Parameters Struct Reference

The structure describing model for the inference process.

`#include <agr.hpp>`

Public Member Functions

- Parameters (std::string weights_path="", std::string model_path="", BackendType backend=BackendType::↩ CPU, size_t batch_size=8)

  Default constructor for the structure.

Public Attributes

- std::string weights_path

  Binary file with network weights.
- std::string model_path

  File with network topology in XML format.
- BackendType backend

  Backend to be used.
- size_t batch_size

  Max batch size to be used. Its value is ignored for VPU backend.

5.3.1 Detailed Description

The structure describing model for the inference process.

Examples:

main.cpp.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 cva::agr::FaceAnalyzer::Parameters::Parameters (std::string *weights_path* = " ", std::string *model_path* = " ", BackendType *backend* = BackendType::CPU, size_t *batch_size* = 8) [inline], [explicit]

Default constructor for the structure.

5.3.3 Member Data Documentation

5.3.3.1 BackendType cva::agr::FaceAnalyzer::Parameters::backend

Backend to be used.

5.3.3.2 size_t cva::agr::FaceAnalyzer::Parameters::batch_size

Max batch size to be used. Its value is ignored for VPU backend.

5.3.3.3 std::string cva::agr::FaceAnalyzer::Parameters::model_path

File with network topology in XML format.

Examples:

main.cpp.

5.3.3.4 std::string cva::agr::FaceAnalyzer::Parameters::weights_path

Binary file with network weights.

The documentation for this struct was generated from the following file:

- agr.hpp

5.4 cva::agr::Version Class Reference

The class is used to represent the version number for the library.

```
#include <agr.hpp>
```

Public Member Functions

- Version (std::uint32_t major=0, std::uint32_t minor=0, std::uint32_t patch=0)
- std::uint32_t major () const

  Get major number of the version.
- std::uint32_t minor () const

  Get minor number of the version.
- std::uint32_t patch () const

  Get patch number of the version.
- std::string toString () const

  Get version string: major.minor.revision.

5.4.1 Detailed Description

The class is used to represent the version number for the library.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 cva::agr::Version::Version (std::uint32_t *major* = 0, std::uint32_t *minor* = 0, std::uint32_t *patch* = 0) [inline], [explicit]

5.4.3 Member Function Documentation

5.4.3.1 std::uint32_t cva::agr::Version::major ( ) const [inline]

Get major number of the version.

5.4.3.2 std::uint32_t cva::agr::Version::minor ( ) const [inline]

Get minor number of the version.

5.4.3.3 std::uint32_t cva::agr::Version::patch ( ) const [inline]

Get patch number of the version.

5.4.3.4 std::string cva::agr::Version::toString ( ) const [inline]

Get version string: major.minor.revision.

The documentation for this class was generated from the following file:

- agr.hpp

Chapter 6

File Documentation

6.1 agr.hpp File Reference

```
#include <opencv2/core.hpp>
#include <memory>
#include <string>
#include <vector>
```

Classes

- class cva::agr::Version

  The class is used to represent the version number for the library.
- class cva::agr::FaceAnalyzer

  The abstract class for DL-based face analyzer (i.e. age and gender classification tool).
- struct cva::agr::FaceAnalyzer::FaceAttributes

  The structure describing age and gender of some object.
- struct cva::agr::FaceAnalyzer::Parameters

  The structure describing model for the inference process.

Namespaces

- cva
- cva::agr

Macros

- #define CVA_AGR_EXPORT

Enumerations

- enum cva::agr::BackendType { cva::agr::BackendType::CPU, cva::agr::BackendType::GPU, cva::agr::↩ BackendType::VPU }

  The enum representing possible HW backends used for computations.
- enum cva::agr::Gender { cva::agr::Gender::FEMALE, cva::agr::Gender::MALE, cva::agr::Gender::UNKNOWN = -1 }

Functions

- Version cva::agr::version ()

  Get version number for the library.

6.1.1 Macro Definition Documentation

6.1.1.1 #define CVA_AGR_EXPORT

6.2 example.dox File Reference

Chapter 7

Example Documentation

7.1 main.cpp

```
/*
    Copyright 2018 Intel Corporation.

    This software and the related documents are Intel copyrighted materials,
    and your use of them is governed by the express license under which they
    were provided to you (Intel Simplified Software License (Version April 2018))
    Unless the License provides otherwise, you may not use, modify,
    copy, publish, distribute, disclose or transmit this software or
    the related documents without Intel's prior written permission.

    This software and the related documents are provided as is, with no
    express or implied warranties, other than those that are expressly
    stated in the License.
*/

#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include <string>
#include <iomanip>
#include <map>
#include "cva/agr/agr.hpp"


#if defined (_WIN32) || defined(WIN32)
#define PATH_SEPARATOR "\\"
#else
#define PATH_SEPARATOR "/"
#endif


int main(int argc, char** argv)
{
    const cv::String keys =
        "{help h usage ? |       | print this message             }"
        "{image i        |       | input image                    }"
        "{model m        |       | path to the model file         }"
        "{weights w      |       | path to the weights file       }"
        "{backend b      | CPU   | preferred backend (VPU/CPU/GPU)}"
        ;

    cv::CommandLineParser parser(argc, argv, keys.c_str());

    if ( parser.has("help") )
    {
        parser.printMessage();
        return EXIT_SUCCESS;
    }

    if ( (!parser.has("image")) || (!parser.has("model")) || (!parser.has("weights")) )
    {
        parser.printMessage();
        return EXIT_FAILURE;
    }
```

```
    std::string model_path = parser.get<std::string>("model");
    std::string weights_path = parser.get<std::string>("weights");

    cva::agr::FaceAnalyzer::Parameters dldt_params;
    dldt_params.model_path   = model_path;
    dldt_params.weights_path = weights_path;

    static const std::map<std::string, cva::agr::BackendType> availableBackends =
    {
        {"CPU", cva::agr::BackendType::CPU},
        {"GPU", cva::agr::BackendType::GPU},
        {"VPU", cva::agr::BackendType::VPU},
    };

    std::string dldt_backend = parser.get<std::string>("backend");
    auto iter = availableBackends.find(dldt_backend);
    if (iter == availableBackends.end())
    {
        std::cout << "DLDT backend not found: " << dldt_backend << std::endl;
        return EXIT_FAILURE;
    }
    dldt_params.backend = iter->second;

    auto decoder = cva::agr::FaceAnalyzer::create(dldt_params);
    if (!decoder)
    {
        std::cout << "Cannot create decoder" << std::endl;
        return EXIT_FAILURE;
    }

    std::string image_path = parser.get<std::string>("image");
    cv::Mat input_image = cv::imread(image_path);
    if (input_image.empty())
    {
        std::cerr << "Coudn't read " << image_path << std::endl;
        return EXIT_FAILURE;
    }

    std::vector<cv::Mat> imgs(1, input_image);

    std::vector<cva::agr::FaceAnalyzer::FaceAttributes> face_attrs;
    decoder->process(imgs, face_attrs);

    std::string gender_string = "Unknown";
    if (face_attrs.at(0).gender == cva::agr::Gender::MALE)
    {
        gender_string = "Male";
    }
    else if (face_attrs.at(0).gender == cva::agr::Gender::FEMALE)
    {
        gender_string = "Female";
    }

    std::cout << std::setw(10) << "Image: "  << image_path           << std::endl;
    std::cout << std::setw(10) << "Age: "    << face_attrs.at(0).age << std::endl;
    std::cout << std::setw(10) << "Gender: " << gender_string        << std::endl;

    return EXIT_SUCCESS;
}
```