Emotion recognition

Generated by Doxygen 1.8.11

Contents

Chapter 1

Main Page

Note

This library will load data files located in the directory located at `../share/cva/Emotions↩ Recognition/assets` relative to it. For correct operation, if the library is copied outside of the installation directory, the `assets` directory must be copied alongside it.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 cva Namespace Reference

Namespaces

- er

5.2 cva::er Namespace Reference

Classes

- class EmotionsRecognizer
  Interface to implementation of emotion recognizer.

Functions

- CVA_AC_SHARED_LIBRARY_IMPORT ac::Version version ()
  Returns the version number of the library.

5.2.1 Function Documentation

5.2.1.1 CVA_AC_SHARED_LIBRARY_IMPORT ac::Version cva::er::version ( )

Returns the version number of the library.

Chapter 6

Class Documentation

6.1 cva::er::EmotionsRecognizer Class Reference

Interface to implementation of emotion recognizer.

`#include <er.hpp>`

Public Types

- enum { MAX_IMAGE_WIDTH = 4096, MAX_IMAGE_HEIGHT = 4096 }
- enum { MAX_BATCH_SIZE = 16 }

Public Member Functions

- virtual ∼EmotionsRecognizer ()=default
- virtual void recognize (const ac::ConstImageView &image, ac::Span< float > probabilities)=0

  Recognize emotion on `image`, filling `probabilities`.
- virtual void recognizeBatch (ac::Span< const ac::ConstImageView > images, ac::Span< const ac::Span< float >> probability_buffers)=0

  Recognize emotion on `images`, filling the corresponding element of `probability_buffers`.
- virtual int recognize (const ac::ConstImageView &image)=0

  Returns index of recognized emotion on images.
- virtual std::vector< int > recognizeBatch (ac::Span< const ac::ConstImageView > images)=0

  Return index of recognized emotions corresponging.

Static Public Member Functions

- static CVA_AC_SHARED_LIBRARY_IMPORT std::size_t numClasses ()

  Returns the number of recognizable emotions the recognizer knows.
- static CVA_AC_SHARED_LIBRARY_IMPORT const char ∗ classDescription (std::size_t class_index)

  Returns a text description for the object class with index `class_index`.
- static CVA_AC_SHARED_LIBRARY_IMPORT std::unique_ptr< EmotionsRecognizer > createCpu (std←↩ ::size_t batch_size=1)

  Returns a pointer to a new object implementing EmotionsRecognizer that uses the CPU for computations.
- static CVA_AC_SHARED_LIBRARY_IMPORT std::unique_ptr< EmotionsRecognizer > createGpu (ac::←↩ Precision precision=ac::Precision::FP32, std::size_t batch_size=1)

  Returns a pointer to a new object implementing EmotionsRecognizer that uses the GPU for computations.

6.1.1 Detailed Description

Interface to implementation of emotion recognizer.

The implementation allows to recognize the following emotions:

- neutral
- happy
- sad
- surprise
- anger

The object has no mutable state. That is, it will always estimate the same probabilities if it is given the same image.

6.1.2 Member Enumeration Documentation

6.1.2.1 anonymous enum

Enumerator

MAX_IMAGE_WIDTH The maximal supported image width.

MAX_IMAGE_HEIGHT The maximal supported image height.

6.1.2.2 anonymous enum

Enumerator

MAX_BATCH_SIZE The maximal supported batch size.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 virtual cva::er::EmotionsRecognizer::∼EmotionsRecognizer ( ) `[virtual]`, `[default]`

6.1.4 Member Function Documentation

6.1.4.1 static CVA_AC_SHARED_LIBRARY_IMPORT const char∗ cva::er::EmotionsRecognizer::classDescription (std::size_t *class_index*) `[static]`

Returns a text description for the object class with index `class_index`.

Parameters

`in`	*class_index*	I of class

Precondition

`class_index` < numClasses()

The return value is a pointer to a NUL-terminated non-empty string of printable ASCII characters. The string will remain valid for the entire lifetime of the library and must not be freed.

Will not throw runtime exceptions.

Examples:

main.cpp.

6.1.4.2 static CVA_AC_SHARED_LIBRARY_IMPORT std::unique_ptr<EmotionsRecognizer> cva::er::EmotionsRecognizer::createCpu (std::size_t *batch_size* = `1`) `[static]`

Returns a pointer to a new object implementing EmotionsRecognizer that uses the CPU for computations.

`batch_size` is the maximum number of images that recognizeBatch() will be able to handle. Increasing of this parameter will also increase the amount of memory used by the object.

Precondition

1 <= `batch_size` <= MAX_BATCH_SIZE

Examples:

main.cpp.

6.1.4.3 static CVA_AC_SHARED_LIBRARY_IMPORT std::unique_ptr<EmotionsRecognizer> cva::er::EmotionsRecognizer::createGpu (ac::Precision *precision* = `ac::Precision::FP32`, std::size_t *batch_size* = `1`) `[static]`

Returns a pointer to a new object implementing EmotionsRecognizer that uses the GPU for computations.

Precondition

`precision` is ac::Precision::FP32 or ac::Precision::FP16.

The created recognizer will perform computations with the specified `precision`.

See createCpu() for the meaning of and preconditions on `batch_size`.

Examples:

main.cpp.

6.1.4.4 static CVA_AC_SHARED_LIBRARY_IMPORT std::size_t cva::er::EmotionsRecognizer::numClasses ( ) `[static]`

Returns the number of recognizable emotions the recognizer knows.

Will not throw runtime exceptions.

Examples:

main.cpp.

6.1.4.5 virtual void cva::er::EmotionsRecognizer::recognize (const ac::ConstImageView & *image*, ac::Span< float > *probabilities*) `[pure virtual]`

Recognize emotion on `image`, filling `probabilities`.

Parameters

in	*image*	Input face image.
out	*probabilities*	Probability distribution across emotions.

Precondition

`image.format()` is ImageFormat::RGB_8 or ImageFormat::BGR_8
`image.width()` <= MAX_IMAGE_WIDTH
`image.height()` <= MAX_IMAGE_HEIGHT
`probabilities.size()` == numClasses()

Postcondition

For every `i`, probabilities[i] is the probability that the object `image` depicts belongs to the class with index `i`.

This method must not be invoked on the same object from more than one thread at a time.

Examples:

main.cpp.

6.1.4.6 virtual int cva::er::EmotionsRecognizer::recognize (const ac::ConstImageView & *image*) [pure virtual]

Returns index of recognized emotion on images.

Equivalent to the call of `recognize(image, probability)` and find the index that corresponds to the emotion with the maximal probability

Parameters

in	*image*	Input face image.

Precondition

`image.format()` is ImageFormat::RGB_8 or ImageFormat::BGR_8
`image.width()` <= MAX_IMAGE_WIDTH
`image.height()` <= MAX_IMAGE_HEIGHT

This method must not be invoked on the same object from more than one thread at a time.

6.1.4.7 virtual void cva::er::EmotionsRecognizer::recognizeBatch (ac::Span< const ac::ConstImageView > *images,* ac::Span< const ac::Span< float >> *probability_buffers*) [pure virtual]

Recognize emotion on `images`, filling the corresponding element of `probability_buffers`.

Equivalent to a sequence of recognize() calls on each pair of corresponding elements from `images` and `probability_buffers`, but may execute faster. This call will result in multiple infer calls if `images.size()` is bigger than `batch_size`.

Parameters

`in`	*images*	Input face images.
`out`	*probability_buffers*	Probability distributions across emotions.

Precondition

`images.size() == probability_buffers.size()`
Each element of `images` satisfies the preconditions on `image` in recognize().
Each element of `probability_buffers` satisfy the preconditions on `probabilities` in recognize().

This method must not be invoked on the same object from more than one thread at a time.

6.1.4.8 virtual std::vector<int> cva::er::EmotionsRecognizer::recognizeBatch (ac::Span< const ac::ConstImageView > *images*) `[pure virtual]`

Return index of recognized emotions corresponging.

Equivalent to a sequence of recognize() calls on each image. This call will result in multiple infer calls if `images.↩ size()` is bigger than `batch_size`.

Parameters

`in`	*images*	Input face images.

This method must not be invoked on the same object from more than one thread at a time.

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

- er.hpp

Chapter 7

File Documentation

7.1 er.hpp File Reference

```
#include <cstdint>
#include <cstdlib>
#include <memory>
#include <vector>
#include <cva/ac/api.hpp>
#include <cva/ac/image_view.hpp>
#include <cva/ac/span.hpp>
#include <cva/ac/precision.hpp>
```

Classes

- class cva::er::EmotionsRecognizer

  Interface to implementation of emotion recognizer.

Namespaces

- cva
- cva::er

Macros

- #define CVA_ER_EXPORT CVA_AC_SHARED_LIBRARY_IMPORT

Functions

- CVA_AC_SHARED_LIBRARY_IMPORT ac::Version cva::er::version ()

  Returns the version number of the library.

7.1.1 Macro Definition Documentation

7.1.1.1 #define CVA_ER_EXPORT CVA_AC_SHARED_LIBRARY_IMPORT

7.2 example.dox File Reference

7.3 mainpage.dox File Reference

Chapter 8

Example Documentation

8.1 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 <algorithm>
#include <cstdlib>
#include <exception>
#include <iomanip>
#include <iostream>
#include <numeric>
#include <utility>
#include <vector>

#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgproc.hpp>

#include <cva/ac/ocv/image_view.hpp>
#include <cva/er/er.hpp>


namespace ac = cva::ac;

int main(int argc, char *argv[])
try
{
    // Parse the command line arguments using OpenCV.

    cv::CommandLineParser parser(argc, argv,
        "{ @image |        | image }"
        "{ help h |        | print this message }"
        "{ impl   |  CPU   | implementation to use. Possible values are CPU, GPUFP32, GPUFP16 }");

    if (!parser.check())
    {
        parser.printErrors();
        return EXIT_FAILURE;
    }

    if (parser.has("help"))
    {
        parser.printMessage();
        return EXIT_SUCCESS;
    }
    if (!parser.has("@image"))
    {
        std::cerr << argv[0] << ": image parameter missing\n";
        return EXIT_FAILURE;
    }

    auto image_path = parser.get<cv::String>("@image");

    // Read the input image and verify it was correctly loaded and its dimensions
    // are suitable for the recognizer.

    cv::Mat image = cv::imread(image_path);
    if (!image.data)
    {
        std::cerr << argv[0] << ": couldn't load image \"" << image_path << "\"\n";
        return EXIT_FAILURE;
    }

    if (image.cols > cva::er::EmotionsRecognizer::MAX_IMAGE_WIDTH
       ||
            image.rows > cva::er::EmotionsRecognizer::MAX_IMAGE_HEIGHT
      )
    {
        std::cerr << argv[0] << ": image size (" << image.cols << "x" << image.rows << ") "
            << "is greater than what the recognizer supports (<= " <<
      cva::er::EmotionsRecognizer::MAX_IMAGE_WIDTH
            << "x" << cva::er::EmotionsRecognizer::MAX_IMAGE_HEIGHT
       << ")\n";
        return EXIT_FAILURE;
    }

    // Allocate space for the probabilities.

    std::vector<float> probs(cva::er::EmotionsRecognizer::numClasses
      ());

    // Create the recognizer and perform the recognition. Note that OpenCV
    // loads images in the BGR format.

    std::unique_ptr<cva::er::EmotionsRecognizer> recognizer;
    auto impl = parser.get<cv::String>("impl");
    if ("CPU" == impl)
    {
        recognizer = cva::er::EmotionsRecognizer::createCpu(1);
    }
    else if ("GPUFP32" == impl)
    {
        recognizer = cva::er::EmotionsRecognizer::createGpu(
      cva::ac::Precision::FP32, 1);
    }
    else if ("GPUFP16" == impl)
    {
        recognizer = cva::er::EmotionsRecognizer::createGpu(
      cva::ac::Precision::FP16, 1);
    }
    else
    {
        std::cerr << argv[0] << ": --impl must be either \"CPU\" or \"GPUFP32\" or \"GPUFP16\"\n";
        return EXIT_FAILURE;
    }

    recognizer->recognize(ac::ocv::toImageView(ac::ImageFormat::BGR_8, image),
                          ac::toSpan(probs));

    auto index = std::distance(probs.begin(), std::max_element(probs.begin(), probs.end()));
    auto emotion = cva::er::EmotionsRecognizer::classDescription
      (index);
    std::cout << emotion << " (" << index << ")" << std::endl;

    return EXIT_SUCCESS;
}
catch (std::exception &e)
{
    std::cerr << argv[0] << ": " << e.what() << "\n";
    return EXIT_FAILURE;
}
catch (...)
{
    std::cerr << argv[0] << ": " << "unidentified error\n";
    return EXIT_FAILURE;
}
```