Caer

Release '1.9.4'

Jason Dsouza

Jan 20, 2021

FIRST STEPS

Caer is a lightweight Computer Vision library for high-performance AI research. It simplifies your approach towards Computer Vision by abstracting away unnecessary boilerplate code enabling maximum flexibility. By offering powerful image and video processing algorithms, Caer provides both casual and advanced users with an elegant interface for Machine vision operations.

It leverages the power of libraries like OpenCV and Pillow to speed up your Computer Vision workflow — making it fully compatible with other frameworks such as PyTorch and Tensorflow.

This design philosophy makes Caer ideal for students, researchers, hobbyists and even experts in the fields of Deep Learning and Computer Vision to quickly prototype deep learning models or research ideas.

CHAPTER

ONE

CAER AT A GLANCE

CHAPTER

TWO

INSTALLING CAER

Caer supports an installation of Python 3.6 above, available on Windows, MacOS and Linux systems.

CHAPTER

THREE

VERSION CHECK

To see whether `caer` is already installed or to check if an install has worked, run the following in a Python shell or Jupyter notebook:

```
>> import caer
>> print(caer.__version__)
```

or, from the command line:

```
python -c "import caer; print(caer.__version__)"
```

(Try `python3` if `python` is unsuccessful.)

You'll see the version number if `caer` is installed and an error message otherwise.

CHAPTER

FOUR

INSTALLATION

4.1 pip (Recommended)

Prerequisites to a pip install: You are able to use your system's command line to install packages and are using a virtual environment (any of several).

To install the current `caer` you'll need at least Python 3.6.1. If you have an older version of Python, you will not be able to use `caer`.

```
$ pip install --upgrade caer
```

Alternatively, you may download the wheels from PyPi

4.1.1 Warning

Do not use the commands `sudo` and `pip` together as `pip` may overwrite critical system libraries which may require you to reinstall your operating system.

4.2 Bleeding Edge

If a bug fix was made in the repo and you can't wait till a new release is made, you can install the bleeding edge version of `caer` using:

```
pip install git+https://github.com/jasmcaus/caer.git
```

4.3 From Source

If you plan to develop `caer` yourself, or want to be on the cutting edge, you can use an editable install:

First, uninstall any existing installations:

```
pip uninstall -y caer
```

Clone the repo:

```
git clone https://github.com/jasmcaus/caer.git
cd caer
pip install -e . # Do this once to add the package to the Python Path
```

To update the installation:

```
git pull  # Grabs the latest source
pip install -e . # Reinstalls Caer
```

CHAPTER

FIVE

SYSTEM PACKAGE MANAGERS

Using a package manager (`yum`, `apt-get`, etc.) to install `caer` or other Python packages is not your best option:

- You're likely to get an older version.
- You'll probably want to make updates and add new packages outside of the package manager, leaving you with the same kind ofdependency conflicts you see when using pip without a virtual environment.
- There's an added risk because operating systems use Python, so if you make system-wide Python changes (installing as root or using sudo), you can break the operating system.

CHAPTER

SIX

CONTRIBUTING TO CAER

Thank you for choosing to contribute to Caer. We'd love to accept your patches! For those just getting started, Github has a how to. All bug fixes, new functionality, new tutorials etc. are (and should be) submitted via GitHub's mechanism of pull requests.

Caer was made publically available in August 2020 by Jason Dsouza <jasmcaus@gmail.com> and we are proud to have over 150k installs since then! Caer is now maintained by awesome community members just like you.

We have written a comprehensive Contribution Guide that you can follow when contributing to the project.

6.1 Documentation

There is always room for improvement in documentation. We welcome all pull requests to fix typos/improve grammar or semantic structuring of documents. Here are a few documents you can work on:

- Official Tutorials
- README

6.2 Open Issues

If you would like to help in working on open issues. Look out for the following tags: `good first issue`, `help wanted`, `open for contribution`

6.3 Major Contributions

If you are willing to make a major contribution you can always lookout for the active sprint under `Projects` and discuss the proposal with one of our maintainers.

6.4 What we currently need help on

- Improving unit-test cases and test coverage
- Include GPU support

6.5 If you write articles:

If you are interested or have already written a story covering `Caer`, you can submit your story in `markdown` format as a PR here. To convert medium stories into `markdown` format, you may use this chrome extension

CHAPTER

SEVEN

README

CHAPTER

EIGHT

CAER.AUGMENT

8.1 Augmentations functions

8.1.1 conv1d

8.1.2 conv2d

8.1.3 conv3d

CHAPTER

NINE

CAER.COLOR

9.1 From the RGB colorspace

9.1.1 rgb_to_bgr

caer.color.**rgb_to_bgr**(*img*)

Converts an RGB image to its BGR version

Parameters **img** (*ndarray*) – Valid RGB image array

Return type ndarray

Returns BGR Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.1.2 rgb_to_gray

caer.color.**rgb_to_gray**(*img*)

Converts an RGB image to its Grayscale version

Parameters **img** (*ndarray*) – Valid RGB image array

Return type ndarray

Returns Grayscale Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.1.3 rgb_to_hsv

caer.color.**rgb_to_hsv**(*img*)

Converts an RGB image to its HSV version

Parameters **img** (*ndarray*) – Valid RGB image array

Return type ndarray

Returns HSV Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.1.4 rgb_to_lab

caer.color.**rgb_to_bgr**(*img*)

Converts an RGB image to its BGR version

Parameters **img**¶ (*ndarray*) – Valid RGB image array

Return type ndarray

Returns BGR Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.2 From the BGR colorspace

9.2.1 bgr_to_rgb

caer.color.**bgr_to_rgb**(*img*)

Converts a BGR image to its RGB version

Parameters **img**¶ (*ndarray*) – Valid BGR image array

Return type ndarray

Returns RGB Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.2.2 bgr_to_gray

caer.color.**bgr_to_gray**(*img*)

Converts a BGR image to its Grayscale version

Parameters **img**¶ (*ndarray*) – Valid BGR image array

Return type ndarray

Returns Grayscale Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.2.3 bgr_to_hsv

caer.color.**bgr_to_hsv**(*img*)

Converts a BGR image to its HSV version

Parameters **img**¶ (*ndarray*) – Valid BGR image array

Return type ndarray

Returns HSV Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.2.4 bgr_to_lab

caer.color.**bgr_to_lab**(*img*)

Converts a BGR image to its LAB version

Parameters **img**¶ (*ndarray*) – Valid BGR image array

Return type ndarray

Returns LAB Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.3 From the HSV colorspace

9.3.1 hsv_to_rgb

caer.color.**hsv_to_rgb**(*img*)

Converts a HSV image to its RGB version

Parameters **img**¶ (*ndarray*) – Valid HSV image array

Return type ndarray

Returns RGB Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.3.2 hsv_to_bgr

caer.color.**hsv_to_bgr**(*img*)

Converts a HSV image to its BGR version

Parameters **img**¶ (*ndarray*) – Valid HSV image array

Return type ndarray

Returns BGR Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.3.3 hsv_to_gray

caer.color.**hsv_to_gray**(*img*)

Converts a HSV image to its Grayscale version

Parameters **img**¶ (*ndarray*) – Valid HSV image array

Return type ndarray

Returns Grayscale Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.3.4 hsv_to_lab

caer.color.**hsv_to_lab**(*img*)

Converts a HSV image to its LAB version

Parameters **img**¶ (*ndarray*) – Valid HSV image array

Return type ndarray

Returns LAB Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.4 From the LAB colorspace

9.4.1 lab_to_rgb

caer.color.**lab_to_rgb**(*img*)

Converts an LAB image to its RGB version

Parameters **img**¶ (*ndarray*) – Valid LAB image array

Return type ndarray

Returns RGB Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.4.2 lab_to_bgr

caer.color.**lab_to_bgr**(*img*)

Converts an LAB image to its BGR version

Parameters **img**¶ (*ndarray*) – Valid LAB image array

Return type ndarray

Returns BGR Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.4.3 lab_to_gray

caer.color.**lab_to_gray**(*img*)

Converts an LAB image to its Grayscale version

Parameters **img**¶ (*ndarray*) – Valid LAB image array

Return type ndarray

Returns Grayscale Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

9.4.4 lab_to_hsv

caer.color.**lab_to_hsv**(*img*)

Converts an LAB image to its HSV version

Parameters **img**¶ (*ndarray*) – Valid LAB image array

Return type ndarray

Returns HSV Image (ndarray)

Raises **ValueError** – If *img* is not of shape 3

CHAPTER

TEN

CAER.DATA

caer.data.**abspath**(*file_name*)
Returns the absolute path of *file_name*

Return type str

caer.data.**audio_mixer**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of an audio mixer.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**bear**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a bear.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**beverages**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of beverages.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**black_cat**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a black cat.

Parameters

- **target_size¶** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb¶** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**blue_tang**(*target_size=None*, *rgb=False*)

Returns a standard 640x414 image (RGB, by default) of a blue tang (a type of fish).

Parameters

- **target_size¶** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb¶** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**camera**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a camera.

Parameters

- **target_size¶** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb¶** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**controller**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a game controller.

Parameters

- **target_size¶** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb¶** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**drone**(*target_size=None*, *rgb=False*)

Returns a standard 640x358 image (RGB, by default) of a robotic drone.

Parameters

- **target_size¶** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb¶** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**dusk**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a dusk landscape.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**fighter_fish**(*target_size=None*, *rgb=False*)

Returns a standard 640x640 image (RGB, by default) of a fighter fish.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**gold_fish**(*target_size=None*, *rgb=False*)

Returns a standard 640x901 image (RGB, by default) of a gold fish.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**green_controller**(*target_size=None*, *rgb=False*)

Returns a standard 640x512 image (RGB, by default) of a green game controller.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**green_fish**(*target_size=None*, *rgb=False*)

Returns a standard 640x430 image (RGB, by default) of a green fish.

Parameters

- **target_size**¶ (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb**¶ (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**guitar**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a guitar.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**imread**(*image_path*, *target_size=None*, *channels=3*, *rgb=False*, *resize_factor=None*, *keep_aspect_ratio=False*)

Loads in an image from *image_path* Arguments

image_path: Filepath/URL to read the image from target_size: Target image size channels: 1 (grayscale) or 3 (RGB/BGR). Default: 3 rgb: Boolean to keep RGB ordering. Default: False resize_factor: Resizes the image using *resize_factor*. Default: None keep_aspect_ratio: Resized image to *target_size* keeping aspect ratio. Some parts of the image may not be included. Default: False

caer.data.**island**(*target_size=None*, *rgb=False*)

Returns a standard 640x426 image (RGB, by default) of an island.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**jellyfish**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a jellyfish.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**laptop**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a laptop.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**minijoin**(**paths*)
Join multiple filepaths together

Return type str

caer.data.**mountain**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a mountain.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**night**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a night landscape.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**phone**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a rotary phone.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**puppies**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a litter of puppies.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**puppy**(*target_size=None*, *rgb=False*)

Returns a standard 640x512 image (RGB, by default) of a puppy.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**red_fish**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a red fish.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**sea_turtle**(*target_size=None*, *rgb=False*)

Returns a standard 640x400 image (RGB, by default) of a sea turtle.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**snow**(*target_size=None*, *rgb=False*)

Returns a standard 640x360 image (RGB, by default) of snow.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**snowflake**(*target_size=None*, *rgb=False*)

Returns a standard 640x480 image (RGB, by default) of a snowflake.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**sunrise**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a sunrise landscape.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

caer.data.**tent**(*target_size=None*, *rgb=False*)

Returns a standard 640x427 image (RGB, by default) of a tent.

Parameters

- **target_size** (*tuple*) – Intended target size (follows the *(width,height)* format)
- **rgb** (*bool*) – Return an RGB image?

Return type ndarray

Returns Image array (ndarray)

CHAPTER

ELEVEN

DISTANCE

CHAPTER

TWELVE

CAER.FILTERS

caer.filters.**brighten**(*img*)

Brighten any BGR/RGB image

Parameters **img** (*ndarray*) – Any regular BGR/RGB image

Returns Brightened image

caer.filters.**invert**(*img*)

Invert any BGR/RGB image

Parameters **img** (*ndarray*) – Any regular BGR/RGB image

Returns Inverted image

caer.filters.**summer**(*img*)

Convert an image using a "Winter" filter For values of gamma above 1, the values are increased and for below 1, the pixel values are decreased. It is implemented by creating a lookup table and using *cv2.LUT()*. The values of the Red channel of BGR and Saturation of HSV is increased by taking a value of gamma greater than 1 and the values of the Blue channel is decreased by taking a value less than 1.

Parameters **img** (*ndarray*) – Any regular BGR/RGB image

Returns Winter image

caer.filters.**winter**(*img*)

Convert an image using a "Winter" filter

Parameters **img** (*ndarray*) – Any regular BGR/RGB image

Returns Winter image

CHAPTER

THIRTEEN

CAER.MORPH

CHAPTER

FOURTEEN

CAER.PATH

caer.path.**abspath**(*file_name*)
Returns the absolute path of *file_name*

Return type str

caer.path.**chdir**(*path*)
Checks into directory *path*

Return type None

caer.path.**cwd**()
Returns the filepath to the current working directory

Return type str

caer.path.**dirname**(*file*)
Returns the base directory name of *file*

Return type str

caer.path.**exists**(*path*)

Checks if a given filepath is valid

Parameters **path**¶ (*str*) – Filepath to check

Return type bool

Returns True; if *path* is a valid filepath False; otherwise

caer.path.**get_size**(*file*, *disp_format='bytes'*)

Returns: the size of *file* in bytes/kb/mb/gb/tb

Parameters

- **file**¶ (*str*) – Filepath to check
- **disp_format**¶ (*str*) – Size format (bytes/kb/mb/gb/tb)

Returns File size in bytes/kb/mb/gb/tb

Return type size (str)

caer.path.**is_image**(*path*)

Checks if a given path is that of a valid image file

Parameters **path**¶ (*str*) – Filepath to check

Return type bool

Returns True; if *path* is a valid image filepath False; otherwise

caer.path.**is_video**(*path*)

Checks if a given path is that of a valid image file

Parameters **path**¶ (*str*) – Filepath to check

Return type bool

Returns True; if *path* is a valid image filepath False; otherwise

caer.path.**isfile**(*path*)

Checks if a given filepath is a valid path

Parameters **path**¶ (*str*) – Filepath to check

Return type bool

Returns True; if *path* is a valid filepath False; otherwise

caer.path.**list_images**(*DIR*, *include_subdirs=True*, *use_fullpath=False*, *show_size=False*, *verbose=1*)

Lists all image files within a specific directory (and sub-directories if *include_subdirs=True*)

Parameters

- **DIR**¶ (*str*) – Directory to search for image files
- **include_subdirs**¶ (*bool*) – Indicate whether to search all subdirectories as well (default = False)
- **use_fullpath**¶ (*bool*) – Include full filepaths in the returned list (default = False)
- **show_size**¶ (*bool*) – Prints the disk size of the image files (default = False)

Returns List of names (or full filepaths if *use_fullpath=True*) of the image files

Return type image_files (list)

caer.path.**list_media**(*DIR*, *include_subdirs=True*, *use_fullpath=False*, *show_size=True*, *verbose=0*)

Lists all media files within a specific directory (and sub-directories if *include_subdirs=True*)

Parameters

- **DIR**¶ (*str*) – Directory to search for media files
- **include_subdirs**¶ (*bool*) – Indicate whether to search all subdirectories as well (default = False)
- **use_fullpath**¶ (*bool*) – Include full filepaths in the returned list (default = False)
- **show_size**¶ (*bool*) – Prints the disk size of the media files (default = False)

Returns List of names (or full filepaths if *use_fullpath=True*) of the media files

Return type media_files (list)

caer.path.**list_videos**(*DIR*, *include_subdirs=True*, *use_fullpath=False*, *show_size=False*, *verbose=1*)

Lists all video files within a specific directory (and sub-directories if *include_subdirs=True*)

Parameters

- **DIR** (*str*) – Directory to search for video files
- **include_subdirs** (*bool*) – Indicate whether to search all subdirectories as well (default = False)
- **use_fullpath** (*bool*) – Include full filepaths in the returned list (default = False)
- **show_size** (*bool*) – Prints the disk size of the video files (default = False)

Returns List of names (or full filepaths if *use_fullpath=True*) of the video files

Return type video_files (list)

caer.path.**listdir**(*DIR*, *include_subdirs=True*, *use_fullpath=False*, *show_size=True*, *verbose=1*)

Lists all files within a specific directory (and sub-directories if *include_subdirs=True*)

Parameters

- **DIR** (*str*) – Directory to search for files
- **include_subdirs** (*bool*) – Indicate whether to search all subdirectories as well (default = False)
- **use_fullpath** (*bool*) – Include full filepaths in the returned list (default = False)
- **show_size** (*bool*) – Prints the disk size of the files (default = False)

Returns List of names (or full filepaths if *use_fullpath=True*) of the files

Return type files (list)

caer.path.**minijoin**(**paths*)

Join multiple filepaths together

Return type str

caer.path.**mkdir**(*path*)

Creates a directory at *path*

Return type None

CHAPTER

FIFTEEN

CAER.PREPROCESSING

caer.preprocessing.**compute_mean**(*data*, *channels*, *per_channel_subtraction=True*)

Computes mean per channel over the train set and returns a tuple of dimensions=channels Train should not be normalized

Return type Tuple

caer.preprocessing.**compute_mean_from_dir**(*DIR*, *channels*, *per_channel_subtraction=True*, *include_subdirs=True*)

Computes mean per channel Mean must be computed ONLY on the train set

Return type Tuple

caer.preprocessing.**subtract_mean**(*data*, *channels*, *mean_sub_values*)

Per channel subtraction values computed from compute_mean() or compute_mean_from_dir() Subtracts mean from the validation set

Return type List[str]

CHAPTER

SIXTEEN

CAER.SEGMENTATION

CHAPTER

SEVENTEEN

CAER.TRANSFORMS

caer.transforms.**equalize**(*img*, *mask=None*, *by_channels=True*)

Equalize the image histogram.

Parameters

- **img** (*ndarray*) – RGB or grayscale image.
- **mask** (*ndarray*) – An optional mask. If given, only the pixels selected by the mask are included in the analysis. Maybe 1 channel or 3 channel array.
- **by_channels** (*bool*) – If True, use equalization by channels separately, else convert image to YCbCr representation and use equalization by *Y* channel.

Returns Equalized image (ndarray)

Examples:

```
>> img = caer.data.beverages()
>> equalized = caer.equalize(img, mask=None)
>> equalized.shape
(427,640,3)
```

caer.transforms.**posterize**(*img*, *bits*)

Reduce the number of bits for each color channel in the image.

Parameters

- **img** (*ndarray*) – Image to posterize.
- **bits** (*int*) – Number of high bits. Must be in range [0, 8]

Returns Image with reduced color channels (ndarray)

Examples:

```
>> img = caer.data.sunrise()
>> posterized = caer.posterize(img, bits=4)
>> posterized.shape
(427,640,3)
```

caer.transforms.**rotate**(*img*, *angle*, *rotPoint=None*)

Rotates an given image by an angle around a particular rotation point (if provided) or centre otherwise.

caer.transforms.**solarize**(*img*, *threshold=128*)

Invert all pixel values above a threshold.

Parameters

- **img** (*ndarray*) – The image to solarize.

- **threshold**¶ (*int*) – All pixels above this grayscale level are inverted.

Returns Solarized image (ndarray)

Examples:

```
>> img = caer.data.sunrise()
>> solarized = caer.solarize(img, threshold=128)
>> solarized.shape
(427,640,3)
```

caer.transforms.**translate**(*image*, *x*, *y*)

Translates a given image across the x-axis and the y-axis

Parameters

- **x**¶ (*int*) – shifts the image right (positive) or left (negative)
- **y**¶ (*int*) – shifts the image down (positive) or up (negative)

Returns The translated image

CHAPTER

EIGHTEEN

CAER.UTILS

CHAPTER

NINETEEN

CAER.VIDEO

caer.video.**extract_frames**(*input_folder*, *output_folder*, *target_size=None*, *include_subdirs=False*, *label_counter=None*, *max_video_count=None*, *frames_per_sec=None*, *frame_interval=None*, *dest_filetype='jpg'*)

Function to extract frames from videos within a directory and save them as separate frames in an output directory. :param _sphinx_paramlinks_caer.video.extract_frames.input_folder: Input video directory. :param _sphinx_paramlinks_caer.video.extract_frames.output_folder: Output directory to save the frames. :param _sphinx_paramlinks_caer.video.extract_frames.target_size: Destination Image Size (tuple of size 2) :param _sphinx_paramlinks_caer.video.extract_frames.label_counter: Starting label counter :param _sphinx_paramlinks_caer.video.extract_frames.max_video_count: Number of videos to process. :param _sphinx_paramlinks_caer.video.extract_frames.frames_per_sec: Number of frames to process per second. :param _sphinx_paramlinks_caer.video.extract_frames.frame_interval: Interval between the frames to be processed. :param _sphinx_paramlinks_caer.video.extract_frames.dest_filetype: Processed image filetype (png, jpg) –> Default: png

Return type int

Returns label_counter (after processing)

CHAPTER

TWENTY

CONTRIBUTOR CODE OF CONDUCT

20.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to make participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

20.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

20.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

20.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

20.5 Enforcement

Instances of abusive, harassing or otherwise unacceptable behavior may be reported by contacting the project team at info@fast.ai. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

20.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available here.

CHAPTER

TWENTYONE

CAER GOVERNANCE | PERSONS OF INTEREST

21.1 Leads

- Jason Dsouza (jasmcaus) (Caer founder)

21.2 Core Maintainers

- Jason Dsouza (jasmcaus) (Caer founder)

CHAPTER

TWENTYTWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

M

N

P

R

S

T

W