inaFaceAnalyzer

David Doukhan

Jun 26, 2023

CONTENTS

inaFaceAnalyzer is a Python toolbox designed for large-scale analysis of faces in image or video streams. It provides a modular processing pipeline allowing to predict age and gender from faces.

Should you need further details regarding this work, please refer to the following paper: David Doukhan and Thomas Petit (2022), inaFaceAnalyzer: a Python toolbox for large-scale face-based description of gender representation in media with limited gender, racial and age biases, *JOSS - The journal of Open Source Software (currently being reviewed)*

CHAPTER

ONE

USER GUIDE AND API DOCUMENTATION

1.1 Installation

inaFaceAnalyzer requires Python version between 3.7 and 3.9. Python 3.10 is not yet supported due to *onnxruntime-gpu* dependency.

1.1.1 Installing from sources on ubuntu

```
$ apt-get install cmake ffmpeg libgl1-mesa-glx
$ git clone https://github.com/ina-foss/inaFaceAnalyzer.git
$ cd inaFaceAnalyzer
$ pip install .
$ ./test_inaFaceAnalyzer.py # to check that the installation is ok
```

1.1.2 Installing from pypi on ubuntu

For GPU support, cuda, cudnn and nvidia drivers should be already installed

```
$ apt-get install cmake ffmpeg libgl1-mesa-glx
$ pip install inaFaceAnalyzer
```

1.1.3 Using docker image

```
$ # download latest docker image from dockerhub
$ docker pull inafoss/inafaceanalyzer
$ # run docker image. setting --gpu argument allows to take advantage of
$ # GPU acceleration (non mandatory)
$ docker run -it --gpus=all inafoss/inafaceanalyzer /bin/bash
$ # lauch unit tests (non mandatory but recommended)
$ python test_inaFaceAnalyzer.py
$ # use any program or API
$ ina_face_analyzer.py -h
```

CHAPTER

TWO

API DOCUMENTATION

2.1 Media processing engines

inaFaceAnalyzer module implements four analysis engines allowing to process video or image streams :

- *ImageAnalyzer* : default choice for image files (jpg, png, etc…)
- *VideoAnalyzer* : default choice for video files (MP4, avi, etc..)
- *VideoKeyframes* : do process only video keyframes (faster decoding)
- *VideoTracking* : Face detection is combined with face tracking

Media analyzer classes share a common interface inherited from abstract class *FaceAnalyzer*. They are designed as *functions objects* or *functors* and can be used as functions, executing the code implemented in *__call__* methods, with first argument corresponding to the media to analyze and returning `pandas.DataFrame`

Custom face detection, face classifier, eye detection and image preprocessing strategies can be provided in the constructor.

```
>>> from inaFaceAnalyzer.inaFaceAnalyzer import VideoAnalyzer
>>> # a video analyzer instance with default parameters
>>> va = VideoAnalyzer()
>>> df = va(sample_vid)
```

class inaFaceAnalyzer.inaFaceAnalyzer.**FaceAnalyzer**(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Bases: `ABC`

This is an abstract class containg the pipeline used to process images, videos, with/without tracking * image/video decoding * face detection * face tracking (optional) * eye detection * face preprocessing * face classification

abstract __call__(*src*)

Method to be implemented by each analyzer

Parameters

src (*str or list*) – path to the video/image to be analyzed May also be a list of images

Return type

Results stored in a `pandas.DataFrame`

__init__(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Construct a face processing pipeline composed of a face detector,

a face preprocessing strategy and a face classifier. The face preprocessing strategy is defined based on the classifier's properties.

Parameters

- **face_detector** (*inaFaceAnalyzer.face_detector.FaceDetector* or None, optional) – face detection object to be used. If None, create a new instance of *inaFaceAnalyzer.face_detector.LibFaceDetection*. Defaults to None.
- **face_classifier** (*inaFaceAnalyzer.face_classifier.FaceClassifier* or None, optional) – Face classification object to be used. if None, create a new instance of *inaFaceAnalyzer.face_classifier.Resnet50FairFaceGRA*. Defaults to None.
- **batch_len** (*int, optional*) – Size of batches to be sent to the GPU. Larger batches allow faster processing results but require more GPU memory. batch_len balue should be set according to the available hardware. Defaults to 32.
- **verbose** (*bool, optional*) – if True, display several intermediate images and results - usefull for debugging but should be avoided in production. Defaults to False.

__weakref__

list of weak references to the object (if defined)

class inaFaceAnalyzer.inaFaceAnalyzer.**ImageAnalyzer**(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Bases: *FaceAnalyzer*

ImageAnalyzer instances allow to detect and classify faces from images

	frame	bbox	detect_conf	sex_decfun	age_decfun	sex_label	age_label
0	./media/dknuth.jpg	(81, 52, 320, 291)	0.999999	7.24841	6.68495	m	61.8495
1	./media/800px-India_(236650352).jpg	(193, 194, 494, 495)	1	9.96501	5.76855	m	52.6855
2	./media/800px-India_(236650352).jpg	(472, 113, 694, 336)	0.999992	15.1933	4.09797	m	35.9797
3	./media/800px-India_(236650352).jpg	(40, 32, 109, 101)	0.999967	11.3448	4.35364	m	38.5364
4	./media/800px-India_(236650352).jpg	(384, 54, 458, 127)	0.999964	11.3798	4.36526	m	38.6526
5	./media/800px-India_(236650352).jpg	(217, 67, 301, 151)	0.999899	9.78476	4.8296	m	43.296

__call__(*img_paths*)

Parameters

img_paths (*str or list*) – path or list of paths to image file(s) to analyze

Returns

- *pandas Dataframe with column 'frame' containing the path to the source*
- *image. Remaining columns depend on processing options selected and*
- *contain bounding box, and face classification information*

class inaFaceAnalyzer.inaFaceAnalyzer.**VideoAnalyzer**(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Bases: *FaceAnalyzer*

Video Analyzer allows to detect and classify faces in video streams

__call__(*video_path*, *fps=None*, *offset=0*)

Pipeline function for face classification from videos (without tracking)

Parameters

- **video_path** – str Path to input video.
- **fps** – float or None (default) amount of video frames to process per seconds if set to None, all frames are processed (costly)
- **offset** – float (default: 0) Time in milliseconds to skip at the beginning of the video.

Returns
frame position, coordinates, predictions, decision function,labels…

Return type
Dataframe with frame and face information

__init__(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Construct a face processing pipeline composed of a face detector,
a face preprocessing strategy and a face classifier. The face preprocessing strategy is defined based on the classifier's properties.

Parameters

- **face_detector** (*inaFaceAnalyzer.face_detector.FaceDetector* or None, optional) – face detection object to be used. If None, create a new instance of *inaFaceAnalyzer.face_detector.LibFaceDetection*. Defaults to None.
- **face_classifier** (*inaFaceAnalyzer.face_classifier.FaceClassifier* or None, optional) – Face classification object to be used. if None, create a new instance of *inaFaceAnalyzer.face_classifier.Resnet50FairFaceGRA*. Defaults to None.
- **batch_len** (*int, optional*) – Size of batches to be sent to the GPU. Larger batches allow faster processing results but require more GPU memory. batch_len balue should be set according to the available hardware. Defaults to 32.
- **verbose** (*bool, optional*) – if True, display several intermediate images and results - usefull for debugging but should be avoided in production. Defaults to False.

class inaFaceAnalyzer.inaFaceAnalyzer.**VideoKeyframes**(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Bases: *FaceAnalyzer*

Face detection and analysis from video limited to video key frames https://en.wikipedia.org/wiki/Key_frame It allows to provide a video analysis summary in fast processing time, but with non uniform frame sampling rate

__call__(*video_path*)

Pipeline function for face classification from videos, limited to key frames

Parameters
video_path (*string*) – Path for input video.

Returns
frame position, coordinates, predictions, decision function,labels…

Return type
Dataframe with frame and face information

__init__(*face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Construct a face processing pipeline composed of a face detector,
a face preprocessing strategy and a face classifier. The face preprocessing strategy is defined based on the classifier's properties.

Parameters

- **face_detector** (*inaFaceAnalyzer.face_detector.FaceDetector* or None, optional) – face detection object to be used. If None, create a new instance of *inaFaceAnalyzer.face_detector.LibFaceDetection*. Defaults to None.
- **face_classifier** (*inaFaceAnalyzer.face_classifier.FaceClassifier* or None, optional) – Face classification object to be used. if None, create a new instance of *inaFaceAnalyzer.face_classifier.Resnet50FairFaceGRA*. Defaults to None.
- **batch_len** (*int, optional*) – Size of batches to be sent to the GPU. Larger batches allow faster processing results but require more GPU memory. batch_len balue should be set according to the available hardware. Defaults to 32.
- **verbose** (*bool, optional*) – if True, display several intermediate images and results - usefull for debugging but should be avoided in production. Defaults to False.

class inaFaceAnalyzer.inaFaceAnalyzer.**VideoTracking**(*detection_period*, *face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Bases: *FaceAnalyzer*

Video processing pipeline including face detection, tracking and classification Tracking is usually less costly than face detection (computation bottleneck) and allows to save computation time Classification decision functions and predictions are averaged for each tracked faces, allowing to obtain more robust analysis estimates

__call__(*video_path*, *fps=None*, *offset=0*)

Pipeline function for face classification from videos with tracking

Parameters

- **video_path** – str Path to input video.
- **fps** – float or None (default) amount of video frames to process per seconds if set to None, all frames are processed (costly)
- **offset** – float (default: 0) Time in milliseconds to skip at the beginning of the video.

Returns
frame position, coordinates, predictions, decision function,labels… faceid column allow to keep track of each unique face found predictions and decision functions with '_avg' suffix are obtained through a smoothing procedure of decision functions for all faces with same faceid. Smoothed estimates are usually more robust than instantaneous ones

Return type
Dataframe with frame and face information

__init__(*detection_period*, *face_detector=None*, *face_classifier=None*, *batch_len=32*, *verbose=False*)

Constructor

Parameters

- **detection_period** (*int the face detection algorithm (costly) will be used once every 'detection_period' analyzed frames. Ie: if set to 5, face detection will occur for 1/5 frames and the*) –

remaining 4/5 faces will be detected through a tracking procedure
if set to 1: face detection will occur for each frame. Face tracking will also be used for each frames, since it will allow to group same faces under a person identifier

- **face_detector** (*instance of face_detector.FaceDetector or None, optional*) – if None, LibFaceDetection is used. The default is None.
- **face_classifier** (*instance of face_classifier.FaceClassifier or None, optional*) – if None, Resnet50FairFaceGRA is used (gender & age). The default is None.
- **verbose** (*boolean, optional*) – If True, will display several usefull intermediate images and results. The default is False.

2.2 Defining a face detection method

Face detection classes are in charge of finding faces in image frames.

Two face detection classes are provided :

- *LibFaceDetection* (default)
- *OcvCnnFacedetector*.

Face detection classes inherits from abstract class *FaceDetector* and share a common interface. They are designed as *functions objects* or *functors* using image frame inputs and returning list of *Detection* instances.

```
>>> from inaFaceAnalyzer.opencv_utils import imread_rgb
>>> from inaFaceAnalyzer.face_detector import LibFaceDetection
>>> # read image
>>> img = imread_rgb('./media/dknuth.jpg')
>>> # instantiate a detector (costly - to be done a single time)
>>> detector = LibFaceDetection()
>>> #call the detector instance as a function - setting verbose to True is slower, but␣
↪display intermediate results
>>> ldetections = detector(img, verbose=True)
>>> print(ldetections)
[Detection(bbox=Rect(x1=113.9406801111573, y1=63.12627956950275, x2=287.63299981285394,␣
↪y2=280.43775060093793), detect_conf=0.9999985098838806)]
```

class inaFaceAnalyzer.face_detector.**FaceDetector**(*minconf*, *min_size_px*, *min_size_prct*, *padd_prct*)

Bases: ABC

__call__(*frame*, *verbose=False*)

Perform face detection on image frames

Parameters

- **frame** (numpy.ndarray) – RGB image frame (height, width, 3).
- **verbose** (*bool, optional*) – display intermediate results such as detected faces Not to be used in production. Defaults to False.

Returns
list of *Detection* instances

__init__(*minconf*, *min_size_px*, *min_size_prct*, *padd_prct*)

Common face detection constructor

Parameters

- **minconf** (*float between 0 and 1*) – the minimal face detection confidence being returned (default values dependent on the face detection class choosen).
- **min_size_px** (*int*) – minimal face size in pixels (default 30): better classification results requires face sizes above 75 pixels.
- **min_size_prct** (*float between 0 and 1*) – minimal face size as a percentage of image frame minimal dimension. Allow to focus on the most relevant faces.
- **padd_prct** (*float between 0 and 1*) – percentage of black padding pixels to be applied on images before detection (default values are set or each detection class).

get_closest_face(*frame*, *ref_bbox*, *min_iou=0.7*, *squarify=True*, *verbose=False*)

To be used for processing ML datasets and training new face classification models.

Some face corpora images may contain several annotated faces. This method return the detected face having the largest IOU with target ref_box. The IOU must be > to min_iou.

Parameters

- **frame** (numpy.ndarray) – RGB image frame (height, width, 3).
- **ref_bbox** (*tuple or Rect*) – reference face bounding box (x1, y1, x2, y2).
- **min_iou** (*float, optional*) – minimal acceptable intersection over union between the detected face to be returned and the reference bounding box. Defaults to .7.
- **squarify** (*TYPE, optional*) – if True, returns the smallest square bounding box containing the detected face. If False returns the original detected face bounding box. Defaults to True.
- **verbose** (*TYPE, optional*) – display intermediate results. Defaults to False.

Returns

detected face matching the criteria (largest IOU with ref_bbox and IOU > min_iou), else None

Return type

Detection or None

most_central_face(*frame*, *contain_center=True*, *verbose=False*)

To be used for processing ML datasets and training new face classification models.

Some ML face corpora images containing several faces, with the target annotated face at the center.

This method returns the detected face which is closest from the center of the image frame

Parameters

- **frame** (numpy.ndarray) – RGB image frame (height, width, 3)
- **contain_center** (*bool, optional*) – if True, the returned face MUST include image center. Defaults to True.
- **verbose** (*bool, optional*) – Display detected faces. Defaults to False.

Returns

if a face matching the conditions has been detected, else None

Return type

Detection

output_type

Atomic element returned by face detection classes

class inaFaceAnalyzer.face_detector.**LibFaceDetection**(*minconf=0.98*, *min_size_px=30*, *min_size_prct=0*, *padd_prct=0*)

Bases: *FaceDetector*

This class wraps the face detection model provided in libfacedetection : a recent face detection library (2021) that can take advantage of GPU acceleration and is able de detect the smallest faces. It may be slow when used with high resolution images.

For more details, please refer to : Peng, H., & Yu, S. (2021). A systematic iou-related method: Beyond simplified regression for better localization. IEEE Transactions on Image Processing, 30, 5032-5044.

__init__(*minconf=0.98*, *min_size_px=30*, *min_size_prct=0*, *padd_prct=0*)

Common face detection constructor

Parameters

- **minconf** (*float between 0 and 1*) – the minimal face detection confidence being returned (default values dependent on the face detection class choosen).
- **min_size_px** (*int*) – minimal face size in pixels (default 30): better classification results requires face sizes above 75 pixels.
- **min_size_prct** (*float between 0 and 1*) – minimal face size as a percentage of image frame minimal dimension. Allow to focus on the most relevant faces.
- **padd_prct** (*float between 0 and 1*) – percentage of black padding pixels to be applied on images before detection (default values are set or each detection class).

class inaFaceAnalyzer.face_detector.**OcvCnnFacedetector**(*minconf=0.65*, *min_size_px=30*, *min_size_prct=0*, *padd_prct=0.15*)

Bases: *FaceDetector*

This class wraps OpenCV default CNN face detection model. Images are fist resized to 300*300 pixels, which may result in missing the smallest faces but allows to get fast detection time.

__init__(*minconf=0.65*, *min_size_px=30*, *min_size_prct=0*, *padd_prct=0.15*)

Common face detection constructor

Parameters

- **minconf** (*float between 0 and 1*) – the minimal face detection confidence being returned (default values dependent on the face detection class choosen).
- **min_size_px** (*int*) – minimal face size in pixels (default 30): better classification results requires face sizes above 75 pixels.
- **min_size_prct** (*float between 0 and 1*) – minimal face size as a percentage of image frame minimal dimension. Allow to focus on the most relevant faces.
- **padd_prct** (*float between 0 and 1*) – percentage of black padding pixels to be applied on images before detection (default values are set or each detection class).

class inaFaceAnalyzer.face_detector.**IdentityFaceDetector**

Bases: *FaceDetector*

This class do not detect faces and return bouding boxes corresponding to the whole image frame. It should be used for processing images or videos corresponding to already-detected cropped faces.

__init__()

IdentityFaceDetector Constructor does not require arguments

2.3 Choosing a face classification model

Module *inaFaceAnalyzer.face_classifier* define classes providing pretrained DNN face classification models allowing to predict gender and/or age from faces.

Four classes are currently proposed :

- *Resnet50FairFaceGRA* predicts age and gender from faces, and is associated to the best classification performances. It should be used by default.
- *Resnet50FairFace*, *Vggface_LSVM_YTF* and *Vggface_LSVM_FairFace* are provided for reproducibility reasons and predict gender only.

Face classification classes share a common interface defined in abstracty class *FaceClassifier*. They can be used with methods :

- *FaceClassifier.preprocessed_img_list()* for processing image lists stored on disk
- *FaceClassifier.__call__()* for processing list of image frames.

Warning: Face classifiers assume input images contain a single detected, centered, eye-aligned, scaled and preprocessed face of dimensions 224*224 pixels

```
>>> from inaFaceAnalyzer.face_classifier import Resnet50FairFaceGRA
>>> classif = Resnet50FairFaceGRA()
>>> classif.preprocessed_img_list(['./media/diallo224.jpg', './media/knuth224.jpg'])
               filename  sex_decfunc  age_decfunc sex_label  age_label
0  ./media/diallo224.jpg    -5.632371     3.072337         f  25.723367
1   ./media/knuth224.jpg     7.255364     6.689072         m  61.890717
```

class inaFaceAnalyzer.face_classifier.**FaceClassifier**

Bases: ABC

Abstract class to be implemented by face classifiers

__call__(*limg*, *verbose=False*)

Classify a list of images images are supposed to be preprocessed faces: aligned, cropped :param limg: :type limg: list of images, a single image can also be used

Returns

- *feats* – face features used as input to the final classifier
- **label** (*str*) – f for female, m for male
- **decision_value** (*float*) – decision function value (negative for female, positive for male)

bbox2square = True

implemented face classifiers may require a preprocessing step consisting to extend the face bounding box such as the resulting box is the smallest square containing the detected face

bbox_scale = 1.1

implemented classifiers are optimized for a given scale factor to be applied on face bounding boxes to be defined here

input_shape = (224, 224, 3)

input image dimensions required by the classifier (height, width, depth)

preprocessed_img_list(*lfiles*, *batch_len=32*)

Performs classification on a list of preprocessed face images Preprocessed face images are assumed to contain a single face which is already detected, cropped, aligned and scaled to classifier's input dimensions (for now: 224*224 pixels)

Parameters

- **lfiles** (*list*) – list of image paths: ['/path/to/img1', '/path/to/img2']
- **batch_len** (*int, optional*) – DNN batch size. Larger batch_len results in faster processing times. Batch lenght is dependent on available GPU memory. Defaults to 32 (suitable for a laptop GPU).

Returns

pandas.DataFrame. a DataFrame with one record for each input image

class inaFaceAnalyzer.face_classifier.**Resnet50FairFace**

Bases: *FaceClassifier*

Resnet50FairFace uses Resnet50 architecture trained to predict gender on FairFace.

bbox2square = True

implemented face classifiers may require a preprocessing step consisting to extend the face bounding box such as the resulting box is the smallest square containing the detected face

bbox_scale = 1.1

implemented classifiers are optimized for a given scale factor to be applied on face bounding boxes to be defined here

input_shape = (224, 224, 3)

input image dimensions required by the classifier (height, width, depth)

preprocessed_img_list(*lfiles*, *batch_len=32*)

Performs classification on a list of preprocessed face images Preprocessed face images are assumed to contain a single face which is already detected, cropped, aligned and scaled to classifier's input dimensions (for now: 224*224 pixels)

Parameters

- **lfiles** (*list*) – list of image paths: ['/path/to/img1', '/path/to/img2']
- **batch_len** (*int, optional*) – DNN batch size. Larger batch_len results in faster processing times. Batch lenght is dependent on available GPU memory. Defaults to 32 (suitable for a laptop GPU).

Returns

pandas.DataFrame. a DataFrame with one record for each input image

class inaFaceAnalyzer.face_classifier.**Resnet50FairFaceGRA**

Bases: *Resnet50FairFace*

Resnet50FairFaceGRA predicts age and gender and is the most accurate proposed. It uses Resnet50 architecture and is trained to predict gender, age and race on FairFace. After consultation of French CNIL (French data protection authority) and DDD (French Rights Defender), racial classification layers were erased from this public distribution in order to prevent their use for non ethical purposes. These models can however be provided for free after examination of each demand.

class inaFaceAnalyzer.face_classifier.**OxfordVggFace**(*hdf5_svm=None*)

Bases: *FaceClassifier*

OxfordVggFace instances are based on VGG16 architectures pretrained using a triplet loss paradigm allowing to obtain face neural representation, that we use to train linear SVM classification systems.

This class takes advantage of Refik Can Malli's keras-vggface module, providing pretrained VGG16 models https://github.com/rcmalli/keras-vggface

class inaFaceAnalyzer.face_classifier.**Vggface_LSVM_YTF**

Bases: *OxfordVggFace*

Vggface_LSVM_FairFace predict gender from face using pretrained Oxford VGG16 facial embedings used to train a Linear SVM on Youtube Faces DB.

This method is fully described in Zohra Rezgui's internship report at INA: Détection et classification de visages pour la description de l'égalité femme-homme dans les archives télévisuelles, Higher School of Statistics and Information Analysis, University of Carthage, 2019

class inaFaceAnalyzer.face_classifier.**Vggface_LSVM_FairFace**

Bases: *OxfordVggFace*

Vggface_LSVM_FairFace predict gender from face using pretrained Oxford VGG16 facial embedings used to train a Linear SVM on FairFace.

2.4 Exporting and Displaying Analysis results

Analysis results can be exported to tables or augmented video streams. They can be displayed in external softwares, as well as in Google Collab or Jupyter notebooks.

2.4.1 Exporting analysis results to table formats

Analysis pipelines defined in module *inaFaceAnalyzer.inaFaceAnalyzer* return frame-coded results as `pandas.DataFrame` (see documentation). They can be exported to any table format supported by pandas (csv, excell, json, etc..)

```
>>> from inaFaceAnalyzer.inaFaceAnalyzer import VideoAnalyzer
>>> # create a video analyzer instance (costly, do it a single time)
>>> va = VideoAnalyzer()
>>> # perform video analysis, analysing a single image frame per second (fps=1)
>>> df = va('./media/pexels-artem-podrez-5725953.mp4', fps=1)
>>> # export pandas Dataframe result to csv
>>> df.to_csv('./myanalysis.csv')
```

2.4.2 Visualizing analysis results

Module *inaFaceAnalyzer.display_utils* contains functions allowing to export video analysis results to formats allowing to display incrusted face detection bounding boxes and classification estimates.

- *ass_subtitle_export()* allows to export results as ASS subtitles (faster).
- *video_export()* generate a new video with incrusted information (longer).

Display functions are currenly limited to the results obtained with *inaFaceAnalyzer.inaFaceAnalyzer.VideoAnalyzer* and *inaFaceAnalyzer.inaFaceAnalyzer.VideoTracking* analysis pipelines.

```
>>> from inaFaceAnalyzer.inaFaceAnalyzer import VideoAnalyzer
>>> from inaFaceAnalyzer.display_utils import ass_subtitle_export
>>> va = VideoAnalyzer()
>>> input_vid = './media/pexels-artem-podrez-5725953.mp4'
>>> # define analysis_fps=2 in order to process 2 image frames per second of video
>>> # analysis_fps should be used for analysis AND subtitle export
>>> analysis_fps = 2
>>> df = va(input_vid, fps=analysis_fps)
>>> # export results to ass subtitle
>>> ass_subtitle_export(vid_src, df, './mysubtitle.ass', analysis_fps=analysis_fps)
```

inaFaceAnalyzer.display_utils.**ass_subtitle_export**(*vid_src*, *result_df*, *ass_dst*, *analysis_fps=None*)

Export inaFaceAnalyzer results to ASS subtitles . ASS can embed complex shapes such as annotated face bounding boxes and classification predictions.

Subtitles are a good option for sharing results, since they do not require a large amount of storage size, and do not alter original videos. Ass subtitles can be displayed in VLC, Aegisub or ELAN annotation software.

```
>>> # displaying mysample_FP2.ass subtitle with vlc
>>> vlc --sub-file ./mysample_FPS2.ass ./sample_vid.mp4
```

Parameters

- **vid_src** (*str*) – path to the input video.
- **result_df** (*str or pandas.DataFrame*) – video analysis result provided as pandas.DataFrame or path to saved csv.
- **ass_dst** (*str*) – output filepath used to save the resulting subtitle. Must have ass extension.
- **analysis_fps** (*numeric or None, optional*) – Amount of frames per second which were analyzed (fps analysis argument) if set to None, then consider that all video frames were processed. Defaults to None.

inaFaceAnalyzer.display_utils.**video_export**(*vid_src*, *result_df*, *vid_dst*, *analysis_fps=None*)

Export inaFaceAnalyzer results to a video with incrusted faces bounding boxes and other analysis information.

Parameters

- **vid_src** (*str*) – path to the input video.
- **result_df** (*str or pandas.DataFrame*) – video analysis result provided as pandas.DataFrame or path to saved csv.
- **vid_dst** (*str*) – output path of the resulting video. Must have MP4 extension.
- **analysis_fps** (*int, optional*) – Amount of frames per second which were analyzed (fps analysis argument). If set to None, then consider that all video frames were processed. Defaults to None.

2.4.3 Playing videos in notebooks

Module *inaFaceAnalyzer.notebook_utils* contain simple functions allowing to display video in jupyter and google collab's notebooks. These functions can be used only in a notebook environment

2.5 Data Structures

This section present the 3 internal data structures implemented in inaFaceAnalyzer.

- *inaFaceAnalyzer.rect.Rect* : a rectangle implementation usefull for manipulating face bounding boxes
- *inaFaceAnalyzer.face_detector.Detection* : obtained from face detection systems, containing face bounding boxes together with face detection confidence
- *inaFaceAnalyzer.face_tracking.TrackDetection* : an extension of `Detection` used when combining face detection and tracking

The remaining data structures used to exchange data between modules are based on `pandas.DataFrame`, allowing easy exports to various table formats (see pandas's documentation).

namedtuple inaFaceAnalyzer.rect.**Rect**(*x1: float*, *y1: float*, *x2: float*, *y2: float*)

Bases: `NamedTuple`

This class is an internal data structure allowing to manipulate rectangle shapes

Fields

0) **x1** (`float`) – left
1) **y1** (`float`) – top
2) **x2** (`float`) – right
3) **y2** (`float`) – bottom

__contains__(*point*)

point contained in self

Parameters
point (*tuple*) – (x,y)

Returns
True if point is in self, else False

Return type
bool

property area

self Surface area

property center

center (x,y) of self (x1, y1, x2, y2)

static from_dlib(*x*)

create Rect from dlib's rectangle instance

Parameters
x (*dlib.rectangle or dlib.drectangle*) –

Return type
Rect

property h

self height

intersect(*r*)

Rectangle intersection between self and r

Parameters
r (Rect) –

Return type
Rect

iou(*r*)

Intersection Over Union between self and r

Parameters
r (Rect) –

Return type
float

property max_dim_len

max (width, height)

mult(*x*, *y*)

Multiply self coordinates by horizontal and vertical scaling factors Usefull for converting [0...1] coordinates to image frame dimensions in pixels

Parameters

- **(float** (*x*) – horizontal scaling factor.
- **y** (*float*) – vertical scaling factor.

Return type
Rect

scale(*scale_prct*)

scale self according to a given scale percentage

Parameters
scale_prct (*float*) –

Return type
Rect

property square

returns the smallest square containing the rectangle

to_dlibFloat()

Convert self to dlib.drectangle (float)

Return type
dlib.drectangle

to_dlibInt()

Convert self to dlib.rectangle (int)

Return type
dlib.rectangle

`to_int`()

Convert self coordinates (float) to the nearest int values :rtype: Rect

`transpose`(*xoffset*, *yoffset*)

Translation

Parameters

- **xoffset** (*float*) – horizontal offset.
- **yoffset** (*float*) – vertical offset.

Return type
Rect

`property w`

self width

`namedtuple` inaFaceAnalyzer.face_detector.**Detection**(*bbox: Rect*, *detect_conf: float*)

Bases: NamedTuple

Atomic element returned by face detection classes

Fields

0) **bbox** (*Rect*) – position of the detected face in the image in pixels
1) **detect_conf** (float) – face detection confidence (0 = lowest confidence, 1 = highest confidence)

`namedtuple` inaFaceAnalyzer.face_tracking.**TrackDetection**(*bbox: Rect*, *face_id: int*, *detect_conf: float*, *track_conf: float*)

Bases: NamedTuple

Atomic element returned by face tracking and detection classes

Fields

0) **bbox** (*Rect*) – bounding box
1) **face_id** (int) – detected face numerical identifier
2) **detect_conf** (float) – face detection confidence
3) **track_conf** (float) – tracking confidence

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

INDEX

Symbols

A

B

C

D

V

W

X

Y