**Natural Language Processing for PDF/TIFF/Image Documents  
Computer Vision for Image Data  
VISION MODULE  
High Precision Image Processing   
Technical Specification, Gap v0.91**

# 1 Images

## 1.1 Images Overview

The Images CV preprocessor contains the following primary classes, and their relationships:

* Images - This is the base class for the representation of a Computer Vision (CV) preprocessed list of images. The constructor optionally takes as parameters a list of images (paths), and corresponding labels, and flags for CV preprocessing the images.

images = Images([<list of images>], [<list\_of\_labels>], flags …)

Alternately, the list of images can be a list of directories which contain images.

Alternately, the list of labels maybe a single value; in which case, the label applies to all the images.

* <Category>Image – This is a derived class for the representation of a category specific images, such as medical imaging data. It inherits from the Image base class.
* Image – This is the base class for the representation of a single Computer Vision (CV). The constructor optionally takes as parameters an image (path), corresponding label, and flags for CV preprocessing the image.

**Image**

**Images (Base Class)**

**Image**

**<Category>Images (Derived Class)**

**Image**

**List**

**Fig. 1a High Level view of Images Class Object Relationships**

## 1.2 Images Properties

The Images class contains the following properties:

* dir – Location to store the CV preprocessed image data.
* images – The list of preprocessed image objects.
* name – The name of the collection (if not specified as a parameter to the constructor, will default to ‘collection.<rootname of first image>’) .
* time – The amount of time to process the collection of images in seconds.
* label – The classification for the list (collection) of images.
* split – Splits the collection into training and test data according to the specified percentage (between 0 and 1), where the order the images has been randomized. When used as a getter, it will return the data and labels as X\_train, X\_test, Y\_train, Y\_test. When used as a setter, it can be used in conjunction with next() and minibatch().

As a setter, may optionally take a second value which is the seed for the random shuffle. E.g.,

images.split = 0.8, 42

* minibatch – Partitions the training data into mini-batches according to the specified size. When used as a getter, it returns a generator for the next min-batch, which is a tuple of the form (X\_train, Y\_train).

## 1.3 Images Overridden Operators

The following operators have their implementations (inherited from the base Object) class overridden:

* len() – The \_\_len\_\_() method is overridden to return the number of images in the collection.
* [] – The \_\_getitem\_\_() method is overridden to return the Image object at the specified index (i.e., image number – 1).
* next() – The \_\_next\_\_() method is overridden to return the next image data in the training data. If the collection has not been pre-split, it will be split by the default percentage 0.8 (80%). When the entire training set has been iterated, the method returns None; the training data is then resorted randomly for a new iteration.

## 1.4 Images Private Methods

The Images class contains the following private methods:

* There are no private methods.

## 1.5 Images Public Methods

The Images class contains the following public methods:

* \_\_init\_\_() - The constructor performs the following:
  + If the images parameter is not None, then an Image object is created for each image in the images parameter.
  + Each image is processed according to settings in the config parameter:
    - grayscale (gray) – convert all RGB and RGBA images to Grayscale (single channel).
    - normalize (norm) – convert all pixel values to be between 0 and 1.
    - resize=(height, width) – change the pixel size of the image.
    - flatten (flat) – after conversions, flatten the pixel data into a 1D vector.
  + When images are being processed as color images (RGB), images with an alpha channel (RGBA) have the alpha channel dropped.
  + The processed image data is converted to a numpy matrix or vector (i.e., flatten)
  + When all the images are preprocessed, the processed images are stored as a collection, with corresponding labels. In a HD5 file. If the collection parameter is specified, the collection file is stored as: <dir>/<collection>.h5; otherwise it is stored as <dir>/<rootname of first image>.h5
  + Async – If the parameter ehandler is set in the constructor, the collection is processed asynchronously and the event handler, specified by ehander, is invoked when completed.
* load() – Loads a HD5 collection file of image objects into an Images object.

# 2 Image

## 2.1 Image Overview

The Image CV preprocessor contains the following primary classes, and their relationships:

* Image- This is the base class for the representation of a Computer Vision (CV) preprocessed image. The constructor optionally takes as parameters an image (path), and corresponding label, and flags for CV preprocessing of the image.

image = Image(<image\_path>, <label>, flags …)

The image path maybe a local path or an URL to a remote location. In the latter, a HTTP request is made to obtain the image data.

## 2.2 Image Properties

The Image class contains the following properties:

* image – The path to the stored image.
* name – The root name of the image (e.g., /mydir/myimage.jpg -> myimage).
* type – The file format of the image (e.g., png).
* size – The byte size of the image.
* shape – The shape of the image (e.g., (height, width, channels))
* time – The elapsed time in seconds for processing the image.
* data – The CV preprocessed image data.
* raw – The uncompressed raw (unprocessed) image data.
* thumbnail – The thumbnail version of the image.
* dir – Location to store the CV preprocessed image data.
* label – The classification for the image.

## 2.3 Image Overridden Operators

The following operators have their implementations (inherited from the base Object) class overridden:

* str() – The \_\_str\_\_() method is overridden to return the image classification (label).

## 2.4 Image Public Methods

The Image class contains of following public methods:

* load() – This method reloads the CV preprocess image from storage. Using the image name and storage path, the method locates the stored image file and reconstructs the Image object.

image.load(“<image\_name>”, “storage\_path”)

* rotate() – This method generates a rotated copy of the raw image data. The parameter degree specifies the degree (angle) to rotate the image. The method uses the imutils module which will resize the image to prevent clipping prior to the rotation. Once rotated, the image is resized back to the target size.

rotated = image.rotate(30)

## 2.5 Image Private Methods

The Image class contains the following private methods:

* \_exists() – This method checks if the image exists at the specified stored path. If not, a FileNotFound exception is thrown.
* \_collate() – This method performs the preprocessing task, which includes:
  + Read the image into memory and uncompressing it into pixel format.
  + Conversion to grayscale or RGB, according to configuration settings.
  + Resizing pixel data, according to configuration settings.
  + Normalizing pixel data.
  + Flattening pixel data, according to configuration settings.
  + Storing processed pixel data, raw data and thumbnail in HD5 file format in storage path, according to configuration settings.

## APPENDIX I: Updates

*Pre-Gap (Epipog) v1.5*

1. Created first instance of module

*Gap v0.9 (alpha)*

1. Added splitting collection into training and test data

1. Added iterating (next) through the training set
2. Added support for minibatch

*Gap v0.91 (alpha)*

1. Add support for Images to take list of directories of images.
2. Add support for Image for image path is an URL (http request).
3. Add image rotation.

## APPENDIX II: Anticipated Engineering

The following has been identified as enhancement/issues to be addressed in subsequent update:

1. Add transformations

**Proprietary and Confidential Information  
Copyright ©2018, Epipog, All Rights Reserved**