**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.

* 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**

**Image**

**List**

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

## 1.2 Images Initializer (Constructor)

***Synopsis***

Images( images=**None**, labels= **None**, dir=’./’, name=**None**, ehandler=**None**, config=**None**)

***Parameters***

images : If not None, a list of either:

1. local image files
2. remote image files (i.e., http[s]://….)
3. directories of local image files.

labels: If not None, either:

1. A single integer value (i.e., label) which corresponds to all the images.
2. A list of the same size as images parameter list of integer values; where the index of each value is the label for the corresponding index in the images parameter.

dir: The directory where to store the machine learning ready data.

name: If not None, a name (string) for the collection.

ehandler: If not None, the processing of the images into machine learning ready data will be asynchronous, and the value of the parameter is the function (or method) that is the event handler when processing is complete. The event handler takes the form:

**def** myHandler(images):   
 # Where images is the Images object that was preprocessed.

config: If not None, a list of one or more configuration settings as strings:

grayscale | gray  
flatten | flat  
resize=(height,width) | resize=height,width  
thumb=(height,width) | thumb=height,width

***Usage***

When specified with no parameters, an empty Images object is created. The Images object may then be used to subsequent load (retrieve) previously stored preprocessed machine learning ready data (see load()).  
  
Otherwise, both images and labels parameters must be specified. The labels parameter corresponds to the labels of the images. Each image specified by the images parameter will be preprocessed according to the optional parameters and configuration settings.

By default, the images will be preprocessed as follows:

1. An Image object is created for each image.
2. The config parameter will have the ‘nostore’ setting, which instructs each image object to not separately store the generated preprocessed machine learning ready data.
3. Upon completion, the preprocessed machine learning data for each image is stored as a single HDF5 file in the current working directory. The root name of the file will be the root name of the first image.

If either or both the dir and config options are not None, they are passed down to each Image object.

If the name parameter is specified, the value will be the root name of the HDF5 stored file.

If the ehandler parameter is not None, then the above will occur asynchronously, and when completed, the corresponding event handler will be called with the Images object passed as a parameter. The ehandler parameter may also be specified as a tuple, where the first item in the tuple is the event handler and the remaining items are arguments to pass to the event handler.

If the path to an image file is remote (i.e., starts with http), an HTTP request will be made to fetch the contents of the file from the remote location.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A AttributeError is raised if an invalid configuration setting is specified.  
  
A IndexError is raised if the size of the labels list does not match the size of the images list.

## 1.3 Images Properties

### 1.3.1 dir

***Synopsis***

# Getter  
path = images.dir

# Setter  
images.dir = path

***Usage***

When used as a getter, the property returns the path where the HDF5 file is stored.

When used as a setter, it is only applicable when used in conjunction with the load() method, indicating where the path where the HDF5 file is found. Otherwise, it is ignored.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A FileNotFoundError is raised if the directory does not exist.

## 1.3.2 name

***Synopsis***

# Getter  
collection = images.name

***Usage***

When used as a getter the property returns the root name of the HDF5 stored file (also referred to as the name of the collection).

### 1.3.3 images

***Synopsis***

# Getter  
images = images.images

***Usage***

When used as a getter the property returns the list of Image objects generated for the collection.

### 1.3.4 label

***Synopsis***

# Getter  
labels = images.label

***Usage***

When used as a getter the property returns the label or list of labels for the collection.

### 1.3.5 time

***Synopsis***

# Getter  
secs = images.time

***Usage***

### When used as a getter the property returns the amount of time (in seconds) it took to preprocess the collection into machine learning ready data.

### 1.3.6 split

***Synopsis***

# Getter  
x\_train, x\_test, y\_train, y\_test = images.split

# Setter  
images.split = percent [,seed]

***Usage***

When used as a setter, a training and test set is generated. The percent parameter specifies the percent that is training data. The data is first randomized before the split. By default, the seed for the split is 0. A seed may be optional specified as a second value.

When repeated, the property will re-split the data and re-randomize it.

When used as a getter, the split training, test, and corresponding labels are returned as lists. This is typically used in conjunction with next() operator or minibatch property.

When the percent is 0, the data is not split. All the data will be returned in x\_train and y\_train, but will still be randomized; x\_test and y\_test will be None.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A ValueError is raised if a parameter is out of range.

A AttributeError is raised if the number of parameters passed to the setter property is incorrect.

### 1.3.7 minibatch

***Synopsis***

# Getter  
generator = images.minibatch

# Setter  
images.minibatch = batch\_size

***Usage***

### When used as a setter, the [mini] batch size is set.

When used as a getter, a generator is returned. The generator will iterate sequentially through the minibatches of the training set.

If the property augment is set to True, for each image in the training set, an additional image is generated by rotating the image a random value between -90 and 90 degrees. Thus, if the mnibatch size is 100 images, the minibatch getter will build a generator for 200 images.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A ValueError is raised if the batch\_size is out of range.

### 1.3.8 augment

***Synopsis***

# Getter  
augment = images.augment

# Setter  
images.augment = augment

***Usage***

### When used as a setter and set to True, image augmentation is enabled during batch generation (see minibatch and next()).

### When used as a getter, the property returns whether image augmentation is enabled.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

## 1.4 Images Overridden Operators

### 1.4.1 len()

***Synopsis***

### n\_images = len(images)

***Usage***

The len() (\_\_len\_\_) operator is overridden to return the number of Image objects in the collection.

### 

### 1.4.1 []

***Synopsis***

### image = images[n]

***Usage***

The [] (\_\_getitem\_\_) operator is overridden to return the Image object at the specified index.

***Exceptions***

A IndexError is raised if the index is out of range.

### 1.4.2 next()

***Synopsis***

### data, label = next(images)

***Usage***

The next() operator is overridden and is used in conjunction with the split property. Once the collection has been split in training and test data, the next() operator will iterate through the training dataset one image, and corresponding label at a time.

Once the training set has been fully iterated, the next() operator returns None, None and will reset and start with the first element. Additionally, the training set will be randomly reshuffled.  
  
If the augment property is set, for each image in the training set, an additional image is generated by rotating the image a random value between -90 and 90 degrees. Thus, if the training set is 1000 images, the next() operator will iterate through 2000 images.

## 

## 1.5 Images Public Methods *Synopsis*

## images.load(name, dir=None) *Parameters*

## name: The name of the collection. *Usage*

This method will load into memory a preprocessed machine learning ready data from an HDF5 file specified by the collection name. The method will load the HDF5 by the filename <collection>.h5. If dir is None, then it will look for the file where the current value for dir is defined (either locally or reset by the dir property). Otherwise, it will look for the file under the directory specified by the dir parameter.

Once loaded, the Images object will have the same characteristics as when the Images object was created.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A ValueError is raised if the name parameter is None.

# 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, or raw pixel data as a numpy array. For remote location, a HTTP request is made to obtain the image data.

## 2.2 Image Initializer (Constructor)

***Synopsis***

Image( image=**None**, label=0, dir=’./’, ehandler=**None**, config=**None**)

***Parameters***

image: If not None, a string of either:

1. local path to an image file
2. remote location of an image file (i.e., http[s]://….)
3. raw pixel data as a numpy array

label: An integer value which is the label corresponding to the image.

dir: The directory where to store the machine learning ready data.

ehandler: If not None, the processing of the images into machine learning ready data will be asynchronous, and the value of the parameter is the function (or method) that is the event handler when processing is complete. The event handler takes the form:

**def** myHandler(image, dir):   
 # Where image is the Image object that was preprocessed.

config: If not None, a list of one or more configuration settings as strings:

grayscale | gray  
flatten | flat  
resize=(height,width) | resize=height,width  
thumb=(height,width) | thumb=height,width  
nostore

***Usage***

When specified with no parameters, an empty Image object is created. The Image object may then be used to subsequent load previously stored preprocessed machine learning ready data (see load()).

Otherwise, both image and label parameters must be specified. The label parameter corresponds to the label of the image. The image specified by the image option will be preprocessed according to the optional parameters and configuration settings.

By default, the image will be preprocessed as follows:

1. Decompressed into raw pixel data.
2. Converted to RGB, if not already.
3. The pixel values are normalized (i.e., pixel integer values 0..255 converted to floating point values between 0 and 1).
4. Upon completion, the raw pixel data and the preprocessed machine learning data for the image is stored as a single HDF5 file in the current working directory. The root name of the file will be the root name of the image.
5. Attributes of the raw and preprocessed image are stored in the HDF5 file.

If the path to an image file is remote (i.e., starts with http), an HTTP request will be made to fetch the contents of the file from the remote location.

If the parameter dir is specified, then the generated HDF5 file is stored in the specified directory. If the directory does not exist, it is created.

If the ehandler parameter is not None, then the above will occur asynchronously, and when completed, the corresponding event handler will be called with the Image object passed as a parameter. The ehandler parameter may also be specified as a tuple, where the first item in the tuple is the event handler and the remaining items are arguments to pass to the event handler.

If the configuration setting *grayscale* (may be shortened to *gray*) is specified, then the image is converted to a single channel grayscale image, if not already.

If the configuration setting resize is specified, then the image is resized to the specified height and width.

If the configuration setting *flatten* (may be shortened to *flat*) is specified, the image is flattened into a single 1D vector (i.e., for input to a ANN).

If the configuration setting thumb is specified, then a thumbnail of the raw pixel data is generated to the specified height and width and stored in the HDF5 file.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A AttributeError is raised if an invalid configuration setting is specified.

A FileNotFoundError is raised if the image file does not exist.

A IOError is raised if an error occurs reading in the image file.

## 2.3 Image Properties

### 2.3.1 image

***Synopsis***

# Getter  
path = image.image

# Setter  
image.image = path

***Usage***

When used as a getter the property returns the path to the image file.

When used as a setter the property specifies the path of the image file to preprocess into machine learning ready data (see initializer).

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A FileNotFoundError is raised if the image file does not exist.

A IOError is raised if an error occurs reading in the image file.

### 2.3.2 name

***Synopsis***

# Getter  
root = image.name

***Usage***

When used as a getter the property returns the root name of the image file(e.g., /mydir/myimage.jpg -> myimage). If the input was raw pixel data, the name property will return ‘untitled’.

### 2.3.3 type

***Synopsis***

# Getter  
suffix = image.type

***Usage***

When used as a getter the property returns the file suffix of the image file (e.g., jpg). If the input was raw pixel data, the property will return ‘raw’.

### 2.3.4 size

***Synopsis***

# Getter  
size = image.size

***Usage***

When used as a getter the property returns the file size of the image file in bytes.

### 2.3.5 raw

***Synopsis***

# Getter  
pixels = image.raw

***Usage***

When used as a getter the property returns the raw pixel data of the uncompressed image.

### 2.3.6 thumb

***Synopsis***

# Getter  
pixels = image.thumb

***Usage***

When used as a getter the property returns the pixel data for the thumbnail image.

### 2.3.7 label

***Synopsis***

# Getter  
label = image.label

# Setter  
image.label = label

***Usage***

When used as a getter the property returns the (integer) label specified for the image.

When used as a setter the property sets the label of the image to the specified integer value.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

### 2.3.8 dir

***Synopsis***

# Getter  
subfolder = image.dir

# Setter  
image.dir = subfolder

***Usage***

When used as a getter the property returns the directory path where the corresponding HDF5 file is stored.

When used as a setter, it is only applicable when used in conjunction with the load() method, indicating where the path where the HDF5 file is found. Otherwise, it is ignored.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A FileNotFoundError is raised if the directory does not exist.

### 2.3.9 data

***Synopsis***

# Getter  
data = image.data

***Usage***

When used as a getter the property returns the preprocessed machine learning ready data.

### 2.3.10 shape

***Synopsis***

# Getter  
shape = image.shape

***Usage***

When used as a getter the property returns the shape of the preprocessed machine learning ready data (e.g., (50, 50, 3)).

### 2.3.11 time

***Synopsis***

# Getter  
secs = image.time

***Usage***

### When used as a getter the property returns the amount of time (in seconds) it took to preprocess the image into machine learning ready data.

## 2.4 Image Overridden Operators

### 2.4.1 str()

***Synopsis***

### label = str(image)

***Usage***

The str() (\_\_str\_\_) operator is overridden to return the label of the image as a string.

## 2.5 Image Public Methods

### 2.5.1 load()

## *Synopsis*

## image.load(name, dir=None) *Parameters*

## name: The filename of the stored HDF5 file.dir: The directory where the HDF5 file is located. *Usage*

This method will load into memory a preprocessed machine learning ready data from an HDF5 file specified by the parameter name. The method will load the HDF5 by the filename <name>.h5. If dir is None, then it will look for the file where the current value for dir is defined (either locally or reset by the dir property). Otherwise, it will look for the file under the directory specified by the dir parameter.

Once loaded, the Image object will have the same characteristics as when the Image object was created.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A ValueError is raised if the name parameter is None.

### 2.5.2 rotate()

## *Synopsis*

## image.rotate(degree) *Parameters*

## degree: The degree (angle) to rotate the image data. *Usage*

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.

***Exceptions***

A TypeError is raised if the type of the parameter is not the expected type.

A ValueError is raised if the degree is not between 0 and 360.

## 

## 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. Added support for Images to take list of directories of images.
2. Added support for Image for image path is an URL (http request).
3. Added image rotation.
4. Rewrote Specification.
5. Added support for Images for image parameters to be folders of images.
6. Added support for GIF.
7. Added support for image augmentation in next()/minibatch.
8. Added support for raw pixel input to Image class.

## APPENDIX II: Anticipated Engineering

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

1. Add transformations

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