readpyne

Art Eidukas @ Data Science Campus

Apr 17, 2019

CONTENTS:

CHAPTER

ONE

API REFERENCE

1.1 readpyne.core

the core functionality for readpyne.

readpyne.core.**binit**(*l_arrays*, *n_features=100*)

Takes in a list of 3 1d arrays of length `n` and firstly it bins it into a set number of features dictated by `n_features`. This produces an array of length `n_features`. Then it stacks the three arrays into a 1d array of length `3 * n_features`

Note: This function does not check if the len of the input list of arrays is 3.

Parameters

- **l_arrays** (*list*) – A list of arrays of length `n`. Usually this will be a vertically collapsed image that has been passed through `cv2.split` to split its channels.
- **n_features** (*int*) – An integer telling the function how many features to produce per array. This dictates the shape of end array as the resulting array will have a length of `3 * n_features`

Returns A numpy array of length `3 * n_features`

Return type numpy.array

readpyne.core.**blobify**(*img*)

Take in an image and pass it through `cv2.dnn.blobFromImage`

Parameters **blob** (*cv2.blob*) – An image that has been `blobified` by cv2. (see the blobify function documentation)

Returns

Return type cv2.blob

readpyne.core.**boxes**(*img*)

Take in an image, resize it, predict boxes. Then perform expansion of the boxes to the width of the receipt and then perform `non_max_supression`.

Parameters **img** (*numpy.array*) – A numpy array representation of an image.

Returns

- *np.array* – The original image
- *np.array* – Predicted subsets for the image.

readpyne.core.**decode**(*scores*, *geo*)

This takes the geometries and confidence scores and produces bounding box values. The inputs to this function come from the EAST model with 2 layers.

Note: This function borrows heavily from: https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/

Parameters

- **scores** (*numpy.array*) – A numpy array of size `(number of found boxes, )` indicating the assigned confidence scores for each box.
- **geo** (*numpy.array*) – A numpy array of size `(number of boxes, 5)` coming out from EAST that describes where the boxes are.

Returns A numpy array of shape `(n, 5)` containing the confidences and box locations for the boxes that are of a certain confidence.

Return type numpy.array

readpyne.core.**expand**(*arr*, *shape*)

Function to expand an array of image coordinates to the full width of the image.

Parameters

- **arr** (*numpy.array*) – A two dimensional array of coordinates which are in the form of `startX, startY, endX, endY`
- **shape** (*tuple*) – A tuple containing the shape of the original receipt image. Easily accessible through the use of the `.shape` method on an image.

Returns A numpy array of the same shape as the original array but with the `x` values in each row expanded to the width of the image.

Return type numpy.array

readpyne.core.**features**(*img*, *subsets*)

Take an image and its subsets created from `boxes` and produce histogram based features for each subset.

Parameters

- **img** (*numpy.array*) – A numpy array representation of an image.
- **subsets** (*list*) – List of numpy arrays of the subsets.

Returns

- *np.array* – The original image
- *list* – A list of 1d numpy arrays

readpyne.core.**forward**(*blob*)

Take in a `cv2 blob` and pass it forward through an `EAST` model.

Note: The layers in the model by default are `"feature_fusion/Conv_7/Sigmoid"`, `"feature_fusion/concat_3"`

Parameters **blob** (*cv2.blob*) – An image that has been `blobified` by cv2. (see the blobify function documentation)

Returns

- *numpy.array* – Scores array for each box found.
- *numpy.array* – Bounding box locations from EAST

readpyne.core.**get_subsets**(*img*, *boxes*)

Take an image and box locations. Then cut out these boxes from the given image.

Parameters

- **img** (*numpy.array*) – A numpy array representation of an image.
- **boxes** (*iterable*) – An iterable (most likely a numpy.array) containing box coordinates.

Returns A list of subsets.

Return type list

readpyne.core.**hist**(*img*)

Histogram creation function. This function takes an input image and then collapses it vertically by using `np.mean`. It does this for each channel of the image as it uses `cv2.split` to get each channel.

Parameters **img** (*numpy.array*) – This is a numpy array representation of an input image. Expected shape for this array is `(n,m,3)` where `n` is the height and `m` is the width of the image.

Returns A list of numpy arrays. Each array will be of length `m` where `m` is the width of the input image.

Return type list

readpyne.core.**process**(*img*)

Function that performs preprocessing before histograms are created.

Parameters **img** (*numpy.array*) – A numpy array of shape (n,m,3) containing an image. Usually this will be a subset of a larger receipt.

Returns A numpy array representation of the input image after all the processing was applied.

Return type numpy.array

readpyne.core.**resize**(*img*)

This function resizes an input image to the correct dimensions for the `EAST` model that is used for text detection.

The dimensions for `EAST` have to be divisable by 32. This function pads the bottom and the right side of the image by a as many pixels as it needs for this to happen.

Note: The image is padded with white as the color and this is then propogated to the rest of the pipeline under normal circumstances.

Parameters **img** (*numpy.array*) – This is a numpy array representation of an input image. Expected shape for this array is `(n,m,3)` where `n` is the height and `m` is the width of the image.

Returns **img** – The white padded image with dimensions now divisable by 32.

Return type numpy.array

readpyne.core.**stack**(*features*)

Stack features. Basically take in a list containing tuples of subsets and histogram based features and stack them all up.

Parameters **features** (*list*) – List of type `[([subsets], [features]), ...]`

Returns List of type `[[all_subsets],[all_features]]`

Return type list

1.2 readpyne.decorators

A few helpful decorators for the rest of the code.

`readpyne.decorators.`**`timeit`**(*fn*)

A simple time decorator. Its not as cool as timeit. Only runs once.

Parameters **`fn`** (*`function`*) – A function to be timed

Returns

Return type function

`readpyne.decorators.`**`unfold_args`**(*fn*)

Take in a function that takes in positional arguments and makes it so you can pass in an unfoldable iterable as the singular argument which then gets unfolded into the positional arguments.

Parameters **`fn`** (*`function`*) –

Returns

Return type function

1.3 readpyne.model

Model training, prediction and data making functions

`readpyne.model.`**`extract`**(*input_folder*, *classifier*, *output_folder=None*)

A function that uses a trained sklearn classifier to extract the lines

Parameters

- **`input_folder`** (*`str`*) – folder path to images
- **`classifier`** (*`sklearn model`*) – sklearn model for classification
- **`output_folder`** (*`str`*) – if provided will save the predicted lines

Returns a list of cutout lines in numpy array form

Return type list

`readpyne.model.`**`make_training_data`**(*input_folder*, *image_re*, *output_folder=None*)

Make training data from a folder of images.

Parameters

- **`input_folder`** (*`str`*) – Folder where the images are stored
- **`image_re`** (*`str`*) – Regular expression string that will filter only the images in the folder.
- **`output_folder`** (*`str`*) – Folder where will the data will be saved. If not provided then data won't be saved and will be just returned.

Returns

- *list* – a set of subsets from the images
- *list* – the stacked image features

readpyne.model.**status**(*name*, *x*, *y*, *model*)
This function is responsible for reporting the quality of the model.

Parameters

- **name** (*str*) – A string that will be the title of report
- **x** (*numpy.array*) – A numpy array with training features.
- **y** (*numpy.array*) – A numpy array with labels
- **model** (*sklearn model*) – A model to be scored

Returns

Return type None

readpyne.model.**train_model**(*X*, *y*, *report=False*, *save_path=None*, *frac_test=0.25*, *sk_model=<class 'sklearn.neighbors.classification.KNeighborsClassifier'>*, *model_params={'n_neighbors': 2}*)
Given a set of data an labels. Train a sklearn model.

Parameters

- **X** (*numpy.array*) – A numpy array with training features.
- **y** (*numpy.array*) – A numpy array with labels.
- **report** (*bool*) – A boolean that tells you if you need the reporting procedure to run.
- **save_path** (*str*) – A path to save the model to
- **frac_test** (*float*) – A float indicating the amount of data to keep for testing.
- **sk_model** (*sklearn model object*) – sklearn model that will be trained. An instance of it will be created and then trained.
- **model_params** (*dict*) – A dict of parameters to be passed to the sklearn model.

Returns

- *sklearn model* – Trained sklearn model
- *tuple* – A tuple containing the untouched test data (X_test, y_test)

1.4 readpyne.io

All input output functions

readpyne.io.**cutout_save**(*path*, *img*, *subsets*)
Take an image and its subsets and export it to the given path.

Parameters

- **path** (*str*) – A path to be used to save all subsets.
- **img** (*numpy array*) – A numpy array representing the image.
- **subsets** (*list*) – A list of numpy array of each subset.

Returns

Return type None

readpyne.io.**get_data**(*folder='data/training'*, *expr='**/Tesco*_600dpi.j*'*)
Load all images in a folder that fit a certain regular expression.

Parameters

- **folder** (*str*) – String path to where the images are.
- **expr** (*str*) – A regular expression that will be used to filter images.

Returns A map object of all the images.

Return type map

readpyne.io.**load_model**(*path*)
Load a model from a string path.

Parameters **path** (*str*) – A path to the model

Returns

Return type sklearn model

readpyne.io.**save_images**(*image_list*, *path='outputs/training'*)
A list of images exports to a given folder.

Parameters

- **image_list** (*list*) – A list of images represented as numpy.arrays
- **path** (*string*) – A path to the folder.

Returns

Return type None

readpyne.io.**save_stack**(*subs*, *features*, *folder*)
Get subsets and features and export them.

Parameters

- **subs** (*list*) – A list of numpy arrays representing subsets of the image.
- **features** (*list*) – A list of features.

Returns

- *subs* – Same as input
- *features* – Same as input

readpyne.io.**show**(*img*)
Use the matplotlib pyplot function to show the image.

Parameters **img** (*numpy array*) – A numpy array representing the image.

Returns

Return type None

CHAPTER

TWO

EXAMPLE USAGE

```
# third party
import pandas as pd

# project
import readpyne.model as m

# config
from readpyne.defaults import config

# example of getting data
data = m.make_training_data(
    config['paths']['training_data_folder'],
    config['training_image_re'],
    config['paths']['training_output_folder'],
    True
)

# example of taking prelabelled data and training a model
training_data = pd.read_csv(config['paths']['training_data'])

model, (X_test, y_test) = m.train_model(
    training_data.iloc[:,1:],
    training_data.iloc[:,0 ],
    report=True,
    save_path=config['model']['classifier']
)

# example of extracting data from all images in a folder
m.extract(config['paths']['classifier_input'],
          config['model']['classifier'],
          config['paths']['classifier_output'])
```

PYTHON MODULE INDEX

r

INDEX