
alexlib

Release 0.0.6

Alex Al-Saffar

Nov 23, 2020

CONTENTS:

1	What does it offer?	1
1.1	alexlib	1
1.1.1	alexlib package	1
1.1.1.1	Submodules	1
1.1.1.2	alexlib.toolbox module	1
1.1.1.3	alexlib.deeplearning module	12
1.1.1.4	alexlib.deeplearning_torch module	16
1.1.1.5	alexlib.miscellaneous module	16
1.1.1.6	Module contents	16
2	Indices and tables	17
	Python Module Index	19
	Index	21

WHAT DOES IT OFFER?

- Extended *Path* class to make life easier.
- Extended *dict* class.
- Extended *list* class.

1.1 alexlib

1.1.1 alexlib package

1.1.1.1 Submodules

1.1.1.2 alexlib.toolbox module

A collection of classes extending the functionality of Python's builtins. email programmer@usa.com

```
class alexlib.toolbox.Artist (*args,      ax=None,      figname='Graph',      title="",      la-
                                bel='curve',      style='seaborn',      create_new_axes=False,      fig-
                                ure_policy=<FigurePolicy.add_new: 'Create a new figure with
                                same name but with added suffix'>, figsize=(7, 4), **kwargs)
```

Bases: *alexlib.toolbox.FigureManager*

```
accessorize (*args, legends=None, title=None, **kwargs)
```

```
get_axes ()
```

```
plot (*args, **kwargs)
```

```
static styler (plot_gen)
```

```
suptitle (title)
```

```
visibility ()
```

```
class alexlib.toolbox.Base
```

Bases: object

```
evalstr (string_, expected='self')
```

```
classmethod from_saved (path)
```

```
get_attributes ()
```

```
save (path)
```

this method goes with default *.npy* format (because it is generic). # P(path).parent.create()

save_json (*path*)

Use case: json is good for simple dicts, e.g. settings. Advantage: human-readable from file explorer.

save_mat (*path*)

for Matlab compatibility.

class alexlib.toolbox.**Browse** (*path, directory=True*)

Bases: object

class alexlib.toolbox.**Compression**

Bases: object

Provides consistent behaviour across all methods ... Both files and folders when compressed, default is being under the root of archive.

static compress_folder (*ip_path, op_path, arcname, format_='zip', **kwargs*)

base_dir has to be relevant to op_path. If you want to compress a folder in Downloads/myfolder/compress_this Then, say that your rootdir is where you want the archive structure to include, then mention the folder you want to actually archive relatively to that root. **format_**: zip, tar, gztar, bztar, xztar

static gz (*file*)

static tar ()

static ungz (*self, op_path=None*)

static untar (*self, fname=None, mode='r', **kwargs*)

static unzip (*ip_path, op_path, fname=None, **kwargs*)

static zip_file (*ip_path, op_path, arcname, **kwargs*)

arcname determines the directory of the file being archived inside the archive. Defaults to same as original directory except for drive. When changed, it should still include the file name in its end. If arcname = filename without any path, then, it will be in the root of the archive.

class alexlib.toolbox.**Cycle** (*c, name=""*)

Bases: object

get ()

get_index ()

next ()

previous ()

sample (*size=1*)

set (*value*)

set_index (*index*)

class alexlib.toolbox.**DisplayData** (*x*)

Bases: object

static eng ()

static set_display ()

class alexlib.toolbox.**FigureManager** (*info_loc=None, figure_policy=<FigurePolicy.same: 'Grab the figure of the same name'>*)

Bases: object

Handles figures of matplotlib.

```

static activate_latex (size=20)
    Setting up matplotlib

adjust_brightness (event)

animate ()

annotate (event, axis=None, data=None)

change_cmap (event)

change_facecolor (event)

clear_axes ()

static findobj (fig_name, obj_name)

static get_ax_size (ax)

get_fig (filename="", suffix=None, **kwargs)

static get_fig_static (figure_policy, filename="", suffix=None, **kwargs)

    Parameters
        • figure_policy –
        • filename –
        • suffix – only relevant if figure_policy is add_new
        • kwargs –

    Returns

static get_nrows_ncols (num_plots, nrows=None, ncols=None)

static grid (ax, factor=5, x_or_y='both', color='gray', alpha1=0.5, alpha2=0.25)

maximize_fig ()

next (event)

pause_func (event)

previous (event)

process_key (event)

replay (event)

save (event)

static set_ax_size (ax, w, h)
    w, h: width, height in inches

static set_ax_to_real_life_size (ax, inch_per_unit=0.03937007874015748)

static set_linestyles_and_markers_and_colors (test=False)

show_cursor (event)

show_help (event)

show_pix_val (event)

static show_pixels_values (ax)

show_ticks (event)

text_info (event)

```

```
toggle_annotate(event)
```

```
static toggle_ticks(an_ax, state=None)
```

```
transparent_fig()
```

```
static try_figure_size()
```

```
static update(fig_name, obj_name, data=None)
```

Fastest update ever. But, you need access to label name. Using this function external to the plotter. But inside the plotter you need to define labels to objects The other alternative is to do the update inside the plotter, but it will become very verbose.

Parameters

- **fig_name** –
- **obj_name** –
- **data** –

Returns

```
update_info_text(message)
```

```
static write(txt, name='text', size=8, **kwargs)
```

```
class alexlib.toolbox.FigurePolicy(value)
```

Bases: `enum.Enum`

An enumeration.

```
add_new = 'Create a new figure with same name but with added suffix'
```

```
close_create_new = 'Close the previous figure that has the same figname and create a new one'
```

```
same = 'Grab the figure of the same name'
```

```
class alexlib.toolbox.ImShow(*images_list: Union[list, numpy.ndarray], sup_titles=None,
                             sub_labels=None, labels=None, save_type=<class
                             'alexlib.toolbox.SaveType.Null'>, save_name=None,
                             save_dir=None, save_kwargs=None, subplots_adjust=None,
                             gridspec=None, tight=True, info_loc=None, nrows=None,
                             ncols=None, ax=None, figsize=None, figname='im_show', fig-
                             ure_policy=<FigurePolicy.add_new: 'Create a new figure with
                             same name but with added suffix'>, auto_brightness=True,
                             delay=200, pause=False, **kwargs)
```

Bases: `alexlib.toolbox.FigureManager`

```
animate()
```

```
annotate(event, axis=None, data=None)
```

```
artist = 'internal'
```

```
static cm(im, nrows=3, ncols=7, **kwargs)
```

Useful for looking at one image in multiple cmaps

Parameters

- **im** –
- **nrows** –
- **ncols** –
- **kwargs** –

Returns

```

classmethod complex (data, pause=True, **kwargs)
classmethod from_directories (*directories, extension='png', **kwargs)
classmethod from_saved (*things, **kwargs)
classmethod from_saved_images_path_lists (*image_list, **kwargs)
parser = 'internal'
static resize (path, m, n)
stream = 'update'
static test ()

```

```
alexlib.toolbox.L
```

```
alias of alexlib.toolbox.List
```

```
class alexlib.toolbox.List (obj_list=None)
```

```
Bases: list, alexlib.toolbox.Base
```

Use this class to keep items of the same type.

```
append (obj)
```

Append object to the end of the list.

```
apply (func, *args, lest=None, jobs=None, depth=1, **kwargs)
```

Parameters

- **jobs** –
- **func** – func has to be a function, possibly a lambda function. At any rate, it should return something.
- **args** –
- **lest** –
- **depth** – apply the function to inner Lists
- **kwargs** – a list of outputs each time the function is called on elements of the list.

Returns

```
attr (name)
```

```
combine ()
```

```
df (names=None)
```

```
filter (func)
```

```
find (patt, match='fmatch')
```

Looks up the string representation of all items in the list and finds the one that partially matches the argument passed. This method is a short for self.filter(lambda x: **string** in str(x)) If you need more complicated logic in the search, revert to filter method.

```
find_index (string_)
```

```
classmethod from_replication (obj, count)
```

```
index_entries (start, end=None, step=None)
```

Used to access entries of items

```
property len
```

```
method (name, *args, **kwargs)

modify (func, lest=None)
    Modifies objects rather than returning new list of objects, hence the name of the method. :param func: a
    string that will be executed, assuming idx, x and y are given. :param lest: :return:

property np

print (nl=1, sep=False, char='-', style=<class 'str'>)

sample (size=1)

save_items (directory, names=None, saver=None)

sort (*args, **kwargs)
    Stable sort IN PLACE.

sorted (*args, **kwargs)

to_struct (keys=None)
    it has to be a property so that the struct is updated when list is updated.

class alexlib.toolbox.Log (path=None)
    Bases: object

    finish ()

class alexlib.toolbox.Manipulator
    Bases: object

    static expand_axis (array, ax_idx, factor)

    static indexer (axis, myslice, rank=None)
        Returns a tuple of slicers.

    static merge_adjacent_axes (array, ax1, ax2)
        Multiplies out two axes to generate reduced order array. :param array: :param ax1: :param ax2: :return:

    static merge_axes (array, ax1, ax2)
        Brings ax2 next to ax1 first, then combine the two axes into one. :param array: :param ax1: :param ax2:
        :return:

    static slicer (array, a_slice: slice, axis=0)

class alexlib.toolbox.P (*args, **kwargs)
    Bases: pathlib.PosixPath, pathlib.Path, alexlib.toolbox.Base

    Path Class: Designed with one goal in mind: any operation on paths MUST NOT take more than one line of
    code.

    append (name="", suffix=None)
        Add extra text after file name, and optionally add extra suffix. e.g: blahlah.extension ==> becomes ==>
        blah/blah_name.extension

    property browse

    property browse2

    clean ()
        removes contents on a folder, rather than deleting the folder.

    compress (op_path=None, base_dir=None, format='zip', **kwargs)

    copy (target=None, contents=False, verbose=False)
        contents: copy the parent directory or its contents.
```

create (*parents=True, exist_ok=True, parent_only=False*)
Creates directory while returning the same object

decompress ()

delete (*are_you_sure=False*)

explore ()

find (**args, r=True, **kwargs*)
short for globbing then using next method to get the first result

get_num (*astring=None*)

property len

listdir ()

make_python_name (*astring=None*)

move (*new_path*)

myglob (*pattern='*', r=False, list_=True, files=True, folders=True, dotfiles=False, return_type=None, absolute=True, filters=None, win_order=False*)

Parameters

- **win_order** –
- **self** –
- **filters** –
- **dotfiles** –
- **pattern** – regex expression.
- **r** – recursive search
- **list** – output format, list or generator.
- **files** – include files in search.
- **folders** – include directories in search.
- **return_type** – output type, Pathlib objects or strings.
- **absolute** – return relative paths or absolute ones.

Returns search results.

:param visible: exclude hidden files and folders (Windows)

prepend (*name, stem=False*)
Add extra text before file name e.g: blahlah.extension ==> becomes ==> blah/name_blah.extension

readit (*reader=None, **kwargs*)

relativity_transform (*reference='deephead', abs_reference=None*)
Takes in a path defined relative to reference, transform it to a path relative to execution directory, then makes it absolute path.

Warning: reference must be included in the execution directory. Otherwise, absolute path of reference should be provided.

renameit (*new_name*)

```
send2trash()  
setitem(key, val)  
size(units='mb')  
split(at=None, index=None)  
    Splits a path at a given string or index :param self: :param at: :param index: :return: two paths  
property string  
static tmp(folder=None, fn=None, path='home')  
    folder is created. file name is not created, only appended.  
property trunk  
    useful if you have multiple dots in file name where .stem fails.  
unzip(op_path=None, fname=None, **kwargs)  
zip(op_path=None, arcname=None, **kwargs)  
class alexlib.toolbox.Read  
    Bases: object  
static csv(path, *args, **kwargs)  
static json(path)  
    Returns a Structure  
static mat(path, correct_dims=True)  
    Parameters  
        • path –  
        • correct_dims –  
    Returns Structure object  
static nii(path)  
static npy(path)  
    returns Structure if the object loaded is a dictionary  
static pickle(path)  
static read(path, **kwargs)  
class alexlib.toolbox.Save  
    Bases: object  
static json(path, obj)  
    This format is compatible with simple dictionaries that hold strings or numbers but nothing more than that.  
    E.g. arrays or any other structure. An example of that is settings dictionary. It is useful because it can be  
    inspected using any text editor.  
static mat(path=AlexPath(/home/docs/tmp_results), mdict=None)  
    Avoid using mat for saving results because of incompatibility. * Nones are not accepted. * Scalars are  
    convetted to [1 x 1] arrays. * etc. Unless you want to pass the results to Matlab animals, avoid this format.  
static pickle(path, obj)  
class alexlib.toolbox.SaveType  
    Bases: object
```

Programming philosophy: this class only worries about saving, and saving only. In other words, the figure must be fully prepared beforehand. Names here are only used for the purpose of saving, never putting titles on figures.

```
class GIF (interval=100, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GenericSave*

Requirements: same axis must persist, only new objects are drawn inside it. This is not harsh as no one wants to add multiple axes on top of each other. Next, the objects drawn must not be removed, or updated, instead they should pile up in axis.

do not pass names in the add method. names will be extracted from figures. # usually it is smoother when adding `animate=True` to plot or `imshow` commands for GIF purpose

Works for images only. Add more `imshow` to the same axis, and that's it. `imshow` will conver up previous images. For lines, it will superimpose it and will look ugly.

If you clear the axis, nothing will be saved. This should not happend. The class will automatically detect new lines by their "neo" labels. and add them then hide them for the next round. Limitation of `ArtistAnimation`: works on lines and images list attached to figure axes. Doesn't work on axes, unless you add large number of them. As such, titles are not incorporated etc.

```
finish()
```

```
class GIFAuto (plotter_class, data, interval=500, extension='gif', fps=4, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GenericAuto*

```
class GIFFileBased (fps=4, dpi=100, bitrate=1800, _type='GIFFileBased', **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GenericSave*

```
finish()
```

```
class GIFFileBasedAuto (plotter_class, data, fps=4, dpi=150, bitrate=2500,  
                        _type='GIFFileBasedAuto', **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GenericAuto*

```
class GIFFPipeBased (*args, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GIFFileBased*

```
class GIFFPipeBasedAuto (*args, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GIFFileBasedAuto*

```
class GenericAuto (plotter_class, data, names_list=None, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GenericSave*

```
animate()
```

```
save_type = 'auto'
```

```
class GenericSave (save_dir=None, save_name=None, watch_figs=None, max_calls=2000, de-  
                  lay=100, **kwargs)
```

Bases: `object`

You can either pass the figures to be tracked or, pass them dynamically at add method, or, add method will capture every figure and axis

```
add (fig_names=None, names=None, **kwargs)
```

```
stream = 'clear'
```

```
class MPEGFileBased (*args, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GIFFileBased*

```
class MPEGFileBasedAuto (*args, **kwargs)
```

Bases: *alexlib.toolbox.SaveType.GIFFileBasedAuto*

```
class MPEGPipeBased (*args, **kwargs)
    Bases: alexlib.toolbox.SaveType.GIFFileBased

class MPEGPipeBasedAuto (*args, **kwargs)
    Bases: alexlib.toolbox.SaveType.GIFFileBasedAuto

class Null (*args, **kwargs)
    Bases: alexlib.toolbox.SaveType.GenericSave

    Use this when you do not want to save anything. This class will help plot to work faster by removing lines
    of previous plot, so you get live animation cheaply.

    finish()

class NullAuto (**kwargs)
    Bases: alexlib.toolbox.SaveType.GenericAuto

class PDF (*args, **kwargs)
    Bases: alexlib.toolbox.SaveType.GenericSave

    For pdf, you just need any figure manager, [update, clear, accumalate], preferably fastest.

    finish(open_result=True)

class PDFAuto (**kwargs)
    Bases: alexlib.toolbox.SaveType.GenericAuto

class PNG (*args, **kwargs)
    Bases: alexlib.toolbox.SaveType.GenericSave

    finish()

class PNGAuto (**kwargs)
    Bases: alexlib.toolbox.SaveType.GenericAuto

class alexlib.toolbox.Struct (dictionary=None, **kwargs)
    Bases: alexlib.toolbox.Base

    Use this class to keep bits and sundry items.

    append(*others, **kwargs)

    static concat_dicts_(*dicts, method=None, lenient=True, collect_items=False, copyit=True)

    copy()

    classmethod defaultdict(*args, **kwargs)

    property df

    property dict

    empty_class()

    classmethod from_keys_values(names, values)

    classmethod from_names(*names, default=None)

    index(idx)

    inverse()

    items()

    keys()

    map(keys)
```

```

plot (artist=None, xdata=None)
spawn_from_values (values)
update (*args, **kwargs)
    Accepts dicts and keyworded args
values ()
class alexlib.toolbox.VisibilityViewer (artist=None, hide_artist_axes=True)
    Bases: alexlib.toolbox.FigureManager
    add (artist=None, increment_index=True, hide_artist_axes=True)
    animate ()
    artist = 'external'
    finish ()
    hide_artist_axes ()
    parser = 'external'
    stream = 'accumulate'
    Viewer Building Philosophy:
    Viewer should act as Saver and Browser:
        • How is the data viewed:
            – Can either be an artist himself, as in ImShow.
            – external artist is required to view the data (especially non image data)
        • Data parsing:
            – internal for loop to go through all the dataset passed. # Allows manual control over parsing.
            – external for loop. It should have add method. # Manual control only takes place after the external loop is over. #TODO parallelize this.
        • Refresh mechanism.
            – Clear the axis.
            – accumulate, using visibility to hide previous axes.
            – The artist has an update method.
    The artist has to have:
        • fig, ax, txt attributes. ax and txt should be lists.
        • the ax and txt attributes should always belong to the same figure.
    Here and in all Visibility classes, the artist is assumed to be always creating new axes along the way.
class alexlib.toolbox.VisibilityViewerAuto (data=None, artist=None, memorize=False,
                                             transpose=True, save_type=<class
                                             'alexlib.toolbox.SaveType.Null'>,
                                             save_dir=None, save_name=None, delay=1,
                                             titles=None, legends=None, x_labels=None,
                                             pause=True, **kwargs)
    Bases: alexlib.toolbox.VisibilityViewer
    animate ()
    static test ()

```

`alexlib.toolbox.accelerate(func, ip)`

Conditions for this to work: * Must run under `__main__` context * `func` must be defined outside that context.

To accelerate IO-bound process, use multithreading. An example of that is something very cheap to process, but takes a long time to be obtained like a request from server. For this, multithreading launches all threads together, then process them in an interleaved fashion as they arrive, all will line-up for same processor, if it happens that they arrived quickly.

To accelerate processing-bound process use multiprocessing, even better, use Numba. Method1 use: multiprocessing / multithreading. Method2: using joblib (still based on multiprocessing) from `joblib import Parallel`, delayed Fast method using `Concurrent` module

`alexlib.toolbox.assert_package_installed(package)`

`alexlib.toolbox.batcher(func_type='function')`

`alexlib.toolbox.batcherv2(func_type='function', order=1)`

`alexlib.toolbox.browse(path, depth=2, width=20)`

Parameters

- **width** – if there are more than this items in a directory, don't parse the rest.
- **depth** – to prevent crash, limit how deep recursive call can happen.
- **path** – absolute path

Returns constructs a class dynamically by using object method.

`alexlib.toolbox.get_time_stamp(name=None)`

`alexlib.toolbox.run_globally(name, asis=False)`

Takes in a function name, reads its source code and returns a new version of it that can be run in the main. This is useful to debug functions and class methods alike.

`alexlib.toolbox.tmp(folder=None, fn=None, path='home')`

folder is created. file name is not created, only appended.

1.1.1.3 alexlib.deeplearning module

class `alexlib.deeplearning.BaseModel` (*hp=None, data=None, model=None, compiler=None, history=None*)

Bases: `abc.ABC`

build (*shape=None, dtype=<class 'numpy.float32'>*)

Building has two main uses.

- Useful to baptize the model, especially when its layers are built lazily. Although this will eventually happen as the first batch goes in. This is a must before showing the summary of the model.
- Doing sanity check about shapes when designing model.
- Sanity check about values and ranges when random normal input is fed.

Parameters

- **dtype** –
- **shape** –

Returns

compile (*loss=None, optimizer=None, metrics=None, compile_model=True*)

Updates compiler attributes. This acts like a setter.

- Must be run prior to fit method.
- Can be run only after defining model attribute.

config ()

evaluate (*x_test=None, y_test=None, names_test=None, idx=None, viz=True, return_loss=False, **kwargs*)

fit (*viz=False, update_default=False, fit_kwargs=None, epochs=None, **kwargs*)

classmethod from_class_model (*path*)

classmethod from_class_weights (*path, hp=None*)

infer (*x*)

This method assumes numpy input, datatype-wise and is also preprocessed. NN is put in eval mode.

Parameters x –

Returns prediction as numpy

static load_model (*directory*)

load_weights (*directory*)

plot_loss ()

plot_model ()

postprocess (*x, *args, **kwargs*)

predict (*x, **kwargs*)

predict_from_position (*position, viz=True, **kwargs*)

predict_from_s_obj (*s_obj, cal_obj=None, viz=True, names=None, **kwargs*)

preprocess (**args, **kwargs*)

save_class (*weights_only=True, version='0'*)

Simply saves everything:

1. Hparams
2. Data specs
3. Model architecture or weights depending on the following argument.

Parameters

- **version** –
- **weights_only** – self-explanatory

Returns

save_model (*directory*)

save_weights (*directory*)

summary ()

switch_to_ll (*epochs=10*)

switch_to_sgd (*epochs=10*)

viz (*pred, gt=None, names=None, **kwargs*)
Assumes numpy inputs

class alexlib.deeplearning.**Compiler** (*loss=None, optimizer=None, metrics=None*)
Bases: object

class alexlib.deeplearning.**DataReader** (*hp=None, data_specs=None, split=None*)
Bases: object

data_split (**args, strings=None, **kwargs*)

Parameters

- **args** – whatever to be sent to train_test_split
- **kwargs** – whatever to be sent to train_test_split
- **strings** –

Returns

static from_saved (*path*)

This method offers an alternative constructor for DataReader class. Use this when loading training data is not required. It requires saved essential parameters to be stored. Those parameters are required by models to work.

Parameters **path** – full path to the saved .npy file containing a dictionary of attributes names and values.

Returns An object with attributes similar to keys and values as in dictionary loaded.

save ()

class alexlib.deeplearning.**Device** (*value*)
Bases: enum.Enum

An enumeration.

auto = 'auto'

cpu = 'cpu'

gpu0 = 'gpu0'

gpu1 = 'gpu1'

two_gpus = '2gpus'

class alexlib.deeplearning.**Ensemble** (*hp_class=None, data_class=None, model_class=None, n=15, _from_saved=False*)

Bases: object

clear_memory ()

fit (*shuffle_train_test=True, save=True, **kwargs*)

classmethod from_saved_models (*parent_dir, wrapper_class*)

classmethod from_saved_weights (*parent_dir, wrapper_class*)

get_model (*n*)

infer (*s*)

predict_from_position (*pos, central_tendency='mean', bins=None, verbose=True*)

read_fit_results ()

```
class alexlib.deeplearning.HPTuning
```

```
Bases: object
```

```
gen_writer()
```

```
static help()
```

```
Steps of use: subclass this and do the following: * Set directory attribute. * set params * set accuracy
metric * generate writer. * implement run method. * run loop method. * in the command line, run
tensorboard -logdir <self.dir>
```

```
loop()
```

```
optimize()
```

```
run(param_dict)
```

```
class alexlib.deeplearning.HyperParam
```

```
Bases: object
```

Benefits of this way of organizing the hyperparameters:

- one place to control everything.
- When doing multiple experiments, one command in console reminds you of settings used in that run (hp.__dict__).
- Ease of saving settings of experiments! and also replicating it later.

```
static from_saved(path)
```

```
save()
```

```
save_code()
```

```
property save_dir
```

```
Ensures that the folder created is directly under deephead root, no matter what directory the run was done
from. This is especially useful during imports, resulting in predicted behaviour.
```

```
class alexlib.deeplearning.KerasOptimizer(d)
```

```
Bases: object
```

```
tune()
```

```
alexlib.deeplearning.config_device(handle, device: alexlib.deeplearning.Device = <De-
vice.gpu0: 'gpu0'>)
```

Parameters

- **handle** – package handle
- **device** – device

Returns possibly a handle to device (in case of Pytorch)

```
alexlib.deeplearning.get_mean_max_error(tf)
```

For Tensorflow

1.1.1.4 alexlib.deeplearning_torch module

1.1.1.5 alexlib.miscellaneous module

`alexlib.miscellaneous.compute_num_of_lines_of_code_in_repo` (*path=AlexPath(/home/docs/checkouts/readthedocs.org/user_uploads/2018/09/alexlib-0.0.6.tar.gz)*,
extension='.py',
*r=True, **kwargs*)

`alexlib.miscellaneous.polygon_area` (*points*)

Return the area of the polygon whose vertices are given by the sequence points.

1.1.1.6 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

a

`alexlib`, [16](#)
`alexlib.deeplearning`, [12](#)
`alexlib.miscellaneous`, [16](#)
`alexlib.toolbox`, [1](#)

A

`accelerate()` (in module `alexlib.toolbox`), 12
`accessorize()` (`alexlib.toolbox.Artist` method), 1
`activate_latex()` (`alexlib.toolbox.FigureManager` static method), 2
`add()` (`alexlib.toolbox.SaveType.GenericSave` method), 9
`add()` (`alexlib.toolbox.VisibilityViewer` method), 11
`add_new` (`alexlib.toolbox.FigurePolicy` attribute), 4
`adjust_brightness()` (`alexlib.toolbox.FigureManager` method), 3
`alexlib`
 module, 16
`alexlib.deeplearning`
 module, 12
`alexlib.miscellaneous`
 module, 16
`alexlib.toolbox`
 module, 1
`animate()` (`alexlib.toolbox.FigureManager` method), 3
`animate()` (`alexlib.toolbox.ImShow` method), 4
`animate()` (`alexlib.toolbox.SaveType.GenericAuto` method), 9
`animate()` (`alexlib.toolbox.VisibilityViewer` method), 11
`animate()` (`alexlib.toolbox.VisibilityViewerAuto` method), 11
`annotate()` (`alexlib.toolbox.FigureManager` method), 3
`annotate()` (`alexlib.toolbox.ImShow` method), 4
`append()` (`alexlib.toolbox.List` method), 5
`append()` (`alexlib.toolbox.P` method), 6
`append()` (`alexlib.toolbox.Struct` method), 10
`apply()` (`alexlib.toolbox.List` method), 5
`artist` (`alexlib.toolbox.ImShow` attribute), 4
`artist` (`alexlib.toolbox.VisibilityViewer` attribute), 11
`Artist` (class in `alexlib.toolbox`), 1
`assert_package_installed()` (in module `alexlib.toolbox`), 12
`attr()` (`alexlib.toolbox.List` method), 5
`auto` (`alexlib.deeplearning.Device` attribute), 14

B

`Base` (class in `alexlib.toolbox`), 1
`BaseModel` (class in `alexlib.deeplearning`), 12
`batcher()` (in module `alexlib.toolbox`), 12
`batcherv2()` (in module `alexlib.toolbox`), 12
`Browse` (class in `alexlib.toolbox`), 2
`browse()` (`alexlib.toolbox.P` property), 6
`browse()` (in module `alexlib.toolbox`), 12
`browse2()` (`alexlib.toolbox.P` property), 6
`build()` (`alexlib.deeplearning.BaseModel` method), 12

C

`change_cmap()` (`alexlib.toolbox.FigureManager` method), 3
`change_facecolor()` (`alexlib.toolbox.FigureManager` method), 3
`clean()` (`alexlib.toolbox.P` method), 6
`clear_axes()` (`alexlib.toolbox.FigureManager` method), 3
`clear_memory()` (`alexlib.deeplearning.Ensemble` method), 14
`close_create_new` (`alexlib.toolbox.FigurePolicy` attribute), 4
`cm()` (`alexlib.toolbox.ImShow` static method), 4
`combine()` (`alexlib.toolbox.List` method), 5
`compile()` (`alexlib.deeplearning.BaseModel` method), 12
`Compiler` (class in `alexlib.deeplearning`), 14
`complex()` (`alexlib.toolbox.ImShow` class method), 5
`compress()` (`alexlib.toolbox.P` method), 6
`compress_folder()` (`alexlib.toolbox.Compression` static method), 2
`Compression` (class in `alexlib.toolbox`), 2
`compute_num_of_lines_of_code_in_repo()` (in module `alexlib.miscellaneous`), 16
`concat_dicts_()` (`alexlib.toolbox.Struct` static method), 10
`config()` (`alexlib.deeplearning.BaseModel` method), 13
`config_device()` (in module `alexlib.deeplearning`), 15

`copy()` (*alexlib.toolbox.P method*), 6
`copy()` (*alexlib.toolbox.Struct method*), 10
`cpu` (*alexlib.deeplearning.Device attribute*), 14
`create()` (*alexlib.toolbox.P method*), 6
`csv()` (*alexlib.toolbox.Read static method*), 8
`Cycle` (*class in alexlib.toolbox*), 2

D

`data_split()` (*alexlib.deeplearning.DataReader method*), 14
`DataReader` (*class in alexlib.deeplearning*), 14
`decompress()` (*alexlib.toolbox.P method*), 7
`defaultdict()` (*alexlib.toolbox.Struct class method*), 10
`delete()` (*alexlib.toolbox.P method*), 7
`Device` (*class in alexlib.deeplearning*), 14
`df()` (*alexlib.toolbox.List method*), 5
`df()` (*alexlib.toolbox.Struct property*), 10
`dict()` (*alexlib.toolbox.Struct property*), 10
`DisplayData` (*class in alexlib.toolbox*), 2

E

`empty_class()` (*alexlib.toolbox.Struct method*), 10
`eng()` (*alexlib.toolbox.DisplayData static method*), 2
`Ensemble` (*class in alexlib.deeplearning*), 14
`evalstr()` (*alexlib.toolbox.Base method*), 1
`evaluate()` (*alexlib.deeplearning.BaseModel method*), 13
`expand_axis()` (*alexlib.toolbox.Manipulator static method*), 6
`explore()` (*alexlib.toolbox.P method*), 7

F

`FigureManager` (*class in alexlib.toolbox*), 2
`FigurePolicy` (*class in alexlib.toolbox*), 4
`filter()` (*alexlib.toolbox.List method*), 5
`find()` (*alexlib.toolbox.List method*), 5
`find()` (*alexlib.toolbox.P method*), 7
`find_index()` (*alexlib.toolbox.List method*), 5
`findobj()` (*alexlib.toolbox.FigureManager static method*), 3
`finish()` (*alexlib.toolbox.Log method*), 6
`finish()` (*alexlib.toolbox.SaveType.GIF method*), 9
`finish()` (*alexlib.toolbox.SaveType.GIFFileBased method*), 9
`finish()` (*alexlib.toolbox.SaveType.Null method*), 10
`finish()` (*alexlib.toolbox.SaveType.PDF method*), 10
`finish()` (*alexlib.toolbox.SaveType.PNG method*), 10
`finish()` (*alexlib.toolbox.VisibilityViewer method*), 11
`fit()` (*alexlib.deeplearning.BaseModel method*), 13
`fit()` (*alexlib.deeplearning.Ensemble method*), 14
`from_class_model()`
 (*alexlib.deeplearning.BaseModel class method*), 13

`from_class_weights()`
 (*alexlib.deeplearning.BaseModel class method*), 13
`from_directories()` (*alexlib.toolbox.ImShow class method*), 5
`from_keys_values()` (*alexlib.toolbox.Struct class method*), 10
`from_names()` (*alexlib.toolbox.Struct class method*), 10
`from_replication()` (*alexlib.toolbox.List class method*), 5
`from_saved()` (*alexlib.deeplearning.DataReader static method*), 14
`from_saved()` (*alexlib.deeplearning.HyperParam static method*), 15
`from_saved()` (*alexlib.toolbox.Base class method*), 1
`from_saved()` (*alexlib.toolbox.ImShow class method*), 5
`from_saved_images_path_lists()`
 (*alexlib.toolbox.ImShow class method*), 5
`from_saved_models()`
 (*alexlib.deeplearning.Ensemble class method*), 14
`from_saved_weights()`
 (*alexlib.deeplearning.Ensemble class method*), 14

G

`gen_writer()` (*alexlib.deeplearning.HPTuning method*), 15
`get()` (*alexlib.toolbox.Cycle method*), 2
`get_attributes()` (*alexlib.toolbox.Base method*), 1
`get_ax_size()` (*alexlib.toolbox.FigureManager static method*), 3
`get_axes()` (*alexlib.toolbox.Artist method*), 1
`get_fig()` (*alexlib.toolbox.FigureManager method*), 3
`get_fig_static()` (*alexlib.toolbox.FigureManager static method*), 3
`get_index()` (*alexlib.toolbox.Cycle method*), 2
`get_mean_max_error()` (*in module alexlib.deeplearning*), 15
`get_model()` (*alexlib.deeplearning.Ensemble method*), 14
`get_nrows_ncols()`
 (*alexlib.toolbox.FigureManager static method*), 3
`get_num()` (*alexlib.toolbox.P method*), 7
`get_time_stamp()` (*in module alexlib.toolbox*), 12
`gpu0` (*alexlib.deeplearning.Device attribute*), 14
`gpu1` (*alexlib.deeplearning.Device attribute*), 14
`grid()` (*alexlib.toolbox.FigureManager static method*), 3
`gz()` (*alexlib.toolbox.Compression static method*), 2

H

help() (*alexlib.deeplearning.HPTuning static method*), 15
 hide_artist_axes() (*alexlib.toolbox.VisibilityViewer method*), 11
 HPTuning (*class in alexlib.deeplearning*), 14
 HyperParam (*class in alexlib.deeplearning*), 15

I

ImShow (*class in alexlib.toolbox*), 4
 index() (*alexlib.toolbox.Struct method*), 10
 index_entries() (*alexlib.toolbox.List method*), 5
 indexer() (*alexlib.toolbox.Manipulator static method*), 6
 infer() (*alexlib.deeplearning.BaseModel method*), 13
 infer() (*alexlib.deeplearning.Ensemble method*), 14
 inverse() (*alexlib.toolbox.Struct method*), 10
 items() (*alexlib.toolbox.Struct method*), 10

J

json() (*alexlib.toolbox.Read static method*), 8
 json() (*alexlib.toolbox.Save static method*), 8

K

KerasOptimizer (*class in alexlib.deeplearning*), 15
 keys() (*alexlib.toolbox.Struct method*), 10

L

L (*in module alexlib.toolbox*), 5
 len() (*alexlib.toolbox.List property*), 5
 len() (*alexlib.toolbox.P property*), 7
 List (*class in alexlib.toolbox*), 5
 listdir() (*alexlib.toolbox.P method*), 7
 load_model() (*alexlib.deeplearning.BaseModel static method*), 13
 load_weights() (*alexlib.deeplearning.BaseModel method*), 13
 Log (*class in alexlib.toolbox*), 6
 loop() (*alexlib.deeplearning.HPTuning method*), 15

M

make_python_name() (*alexlib.toolbox.P method*), 7
 Manipulator (*class in alexlib.toolbox*), 6
 map() (*alexlib.toolbox.Struct method*), 10
 mat() (*alexlib.toolbox.Read static method*), 8
 mat() (*alexlib.toolbox.Save static method*), 8
 maximize_fig() (*alexlib.toolbox.FigureManager method*), 3
 merge_adjacent_axes() (*alexlib.toolbox.Manipulator static method*), 6
 merge_axes() (*alexlib.toolbox.Manipulator static method*), 6

method() (*alexlib.toolbox.List method*), 5
 modify() (*alexlib.toolbox.List method*), 6
 module
 alexlib, 16
 alexlib.deeplearning, 12
 alexlib.miscellaneous, 16
 alexlib.toolbox, 1
 move() (*alexlib.toolbox.P method*), 7
 myglob() (*alexlib.toolbox.P method*), 7

N

next() (*alexlib.toolbox.Cycle method*), 2
 next() (*alexlib.toolbox.FigureManager method*), 3
 nii() (*alexlib.toolbox.Read static method*), 8
 np() (*alexlib.toolbox.List property*), 6
 npy() (*alexlib.toolbox.Read static method*), 8

O

optimize() (*alexlib.deeplearning.HPTuning method*), 15

P

P (*class in alexlib.toolbox*), 6
 parser (*alexlib.toolbox.ImShow attribute*), 5
 parser (*alexlib.toolbox.VisibilityViewer attribute*), 11
 pause_func() (*alexlib.toolbox.FigureManager method*), 3
 pickle() (*alexlib.toolbox.Read static method*), 8
 pickle() (*alexlib.toolbox.Save static method*), 8
 plot() (*alexlib.toolbox.Artist method*), 1
 plot() (*alexlib.toolbox.Struct method*), 10
 plot_loss() (*alexlib.deeplearning.BaseModel method*), 13
 plot_model() (*alexlib.deeplearning.BaseModel method*), 13
 polygon_area() (*in module alexlib.miscellaneous*), 16
 postprocess() (*alexlib.deeplearning.BaseModel method*), 13
 predict() (*alexlib.deeplearning.BaseModel method*), 13
 predict_from_position() (*alexlib.deeplearning.BaseModel method*), 13
 predict_from_position() (*alexlib.deeplearning.Ensemble method*), 14
 predict_from_s_obj() (*alexlib.deeplearning.BaseModel method*), 13
 prepend() (*alexlib.toolbox.P method*), 7
 preprocess() (*alexlib.deeplearning.BaseModel method*), 13
 previous() (*alexlib.toolbox.Cycle method*), 2

previous() (alexlib.toolbox.FigureManager method),
3
print() (alexlib.toolbox.List method), 6
process_key() (alexlib.toolbox.FigureManager
method), 3

R

Read (class in alexlib.toolbox), 8
read() (alexlib.toolbox.Read static method), 8
read_fit_results()
(alexlib.deeplearning.Ensemble method),
14
readit() (alexlib.toolbox.P method), 7
relativity_transform() (alexlib.toolbox.P
method), 7
renameit() (alexlib.toolbox.P method), 7
replay() (alexlib.toolbox.FigureManager method), 3
resize() (alexlib.toolbox.ImShow static method), 5
run() (alexlib.deeplearning.HPTuning method), 15
run_globally() (in module alexlib.toolbox), 12

S

same (alexlib.toolbox.FigurePolicy attribute), 4
sample() (alexlib.toolbox.Cycle method), 2
sample() (alexlib.toolbox.List method), 6
Save (class in alexlib.toolbox), 8
save() (alexlib.deeplearning.DataReader method), 14
save() (alexlib.deeplearning.HyperParam method), 15
save() (alexlib.toolbox.Base method), 1
save() (alexlib.toolbox.FigureManager method), 3
save_class() (alexlib.deeplearning.BaseModel
method), 13
save_code() (alexlib.deeplearning.HyperParam
method), 15
save_dir() (alexlib.deeplearning.HyperParam prop-
erty), 15
save_items() (alexlib.toolbox.List method), 6
save_json() (alexlib.toolbox.Base method), 1
save_mat() (alexlib.toolbox.Base method), 2
save_model() (alexlib.deeplearning.BaseModel
method), 13
save_type (alexlib.toolbox.SaveType.GenericAuto at-
tribute), 9
save_weights() (alexlib.deeplearning.BaseModel
method), 13
SaveType (class in alexlib.toolbox), 8
SaveType.GenericAuto (class in alexlib.toolbox), 9
SaveType.GenericSave (class in alexlib.toolbox), 9
SaveType.GIF (class in alexlib.toolbox), 9
SaveType.GIFAuto (class in alexlib.toolbox), 9
SaveType.GIFFileBased (class in alexlib.toolbox),
9
SaveType.GIFFileBasedAuto (class in
alexlib.toolbox), 9

SaveType.GIFPipeBased (class in alexlib.toolbox),
9
SaveType.GIFPipeBasedAuto (class in
alexlib.toolbox), 9
SaveType.MPEGFileBased (class in
alexlib.toolbox), 9
SaveType.MPEGFileBasedAuto (class in
alexlib.toolbox), 9
SaveType.MPEGPipeBased (class in
alexlib.toolbox), 9
SaveType.MPEGPipeBasedAuto (class in
alexlib.toolbox), 10
SaveType.Null (class in alexlib.toolbox), 10
SaveType.NullAuto (class in alexlib.toolbox), 10
SaveType.PDF (class in alexlib.toolbox), 10
SaveType.PDFAuto (class in alexlib.toolbox), 10
SaveType.PNG (class in alexlib.toolbox), 10
SaveType.PNGAuto (class in alexlib.toolbox), 10
send2trash() (alexlib.toolbox.P method), 7
set() (alexlib.toolbox.Cycle method), 2
set_ax_size() (alexlib.toolbox.FigureManager
static method), 3
set_ax_to_real_life_size()
(alexlib.toolbox.FigureManager static method),
3
set_display() (alexlib.toolbox.DisplayData static
method), 2
set_index() (alexlib.toolbox.Cycle method), 2
set_linestyles_and_markers_and_colors()
(alexlib.toolbox.FigureManager static method),
3
setitem() (alexlib.toolbox.P method), 8
show_cursor() (alexlib.toolbox.FigureManager
method), 3
show_help() (alexlib.toolbox.FigureManager
method), 3
show_pix_val() (alexlib.toolbox.FigureManager
method), 3
show_pixels_values()
(alexlib.toolbox.FigureManager static method),
3
show_ticks() (alexlib.toolbox.FigureManager
method), 3
size() (alexlib.toolbox.P method), 8
slicer() (alexlib.toolbox.Manipulator static method),
6
sort() (alexlib.toolbox.List method), 6
sorted() (alexlib.toolbox.List method), 6
spawn_from_values() (alexlib.toolbox.Struct
method), 11
split() (alexlib.toolbox.P method), 8
stream (alexlib.toolbox.ImShow attribute), 5
stream (alexlib.toolbox.SaveType.GenericSave at-
tribute), 9

stream(*alexlib.toolbox.VisibilityViewer* attribute), 11
 string() (*alexlib.toolbox.P* property), 8
 Struct (class in *alexlib.toolbox*), 10
 styler() (*alexlib.toolbox.Artist* static method), 1
 summary() (*alexlib.deeplearning.BaseModel* method), 13
 supitle() (*alexlib.toolbox.Artist* method), 1
 switch_to_ll() (*alexlib.deeplearning.BaseModel* method), 13
 switch_to_sgd() (*alexlib.deeplearning.BaseModel* method), 13

T

tar() (*alexlib.toolbox.Compression* static method), 2
 test() (*alexlib.toolbox.ImShow* static method), 5
 test() (*alexlib.toolbox.VisibilityViewerAuto* static method), 11
 text_info() (*alexlib.toolbox.FigureManager* method), 3
 tmp() (*alexlib.toolbox.P* static method), 8
 tmp() (in module *alexlib.toolbox*), 12
 to_struct() (*alexlib.toolbox.List* method), 6
 toggle_annotate() (*alexlib.toolbox.FigureManager* method), 3
 toggle_ticks() (*alexlib.toolbox.FigureManager* static method), 4
 transparent_fig() (*alexlib.toolbox.FigureManager* method), 4
 trunk() (*alexlib.toolbox.P* property), 8
 try_figure_size() (*alexlib.toolbox.FigureManager* static method), 4
 tune() (*alexlib.deeplearning.KerasOptimizer* method), 15
 two_gpus (*alexlib.deeplearning.Device* attribute), 14

U

ungz() (*alexlib.toolbox.Compression* static method), 2
 untar() (*alexlib.toolbox.Compression* static method), 2
 unzip() (*alexlib.toolbox.Compression* static method), 2
 unzip() (*alexlib.toolbox.P* method), 8
 update() (*alexlib.toolbox.FigureManager* static method), 4
 update() (*alexlib.toolbox.Struct* method), 11
 update_info_text() (*alexlib.toolbox.FigureManager* method), 4

V

values() (*alexlib.toolbox.Struct* method), 11
 visibility() (*alexlib.toolbox.Artist* method), 1
 VisibilityViewer (class in *alexlib.toolbox*), 11

VisibilityViewerAuto (class in *alexlib.toolbox*), 11
 viz() (*alexlib.deeplearning.BaseModel* method), 13

W

write() (*alexlib.toolbox.FigureManager* static method), 4

Z

zip() (*alexlib.toolbox.P* method), 8
 zip_file() (*alexlib.toolbox.Compression* static method), 2