deepblink

Release 0.1.4

Nov 01, 2022

Contents

1 deepblink.augment module 1

2 deepblink.cli package 3

3 deepblink.data module 5

4 deepblink.datasets package 7

5 deepblink.inference module 11

6 deepblink.io module 13

7 deepblink.losses module 15

8 deepblink.metrics module 17

9 deepblink.models package 21

10 deepblink.networks package 23

11 deepblink.optimizers module 25

12 deepblink.training module 27

13 deepblink.util module 29

Python Module Index 31

Index 33

CHAPTER 1

deepblink.augment module

Model utility functions for augmentation.

deepblink.augment.**augment_batch_baseline**(*images: numpy.ndarray*, *masks: numpy.ndarray*, *flip_: bool = False*, *illuminate_: bool = False*, *gaussian_noise_: bool = False*, *rotate_: bool = False*, *translate_: bool = False*, *cell_size: int = 4*) → Tuple[numpy.ndarray, numpy.ndarray]

Baseline augmentation function.

Probability of augmentations is determined in the corresponding functions and not in this baseline.

Parameters

- **images** – Batch of input image to be augmented with shape (n, x, y).
- **masks** – Batch of corresponding prediction matrix with ground truth values with shape (n, x, y).
- **flip_** – If True, images might be flipped.
- **illuminate_** – If True, images might be altered in illumination.
- **gaussian_noise_** – If True, gaussian noise might be added.
- **rotate_** – If True, images might be rotated.
- **translate_** – If True, images might be translated.
- **cell_size** – Size of one cell in the prediction matrix.

deepblink.augment.**flip**(*image: numpy.ndarray*, *mask: numpy.ndarray*) → Tuple[numpy.ndarray, numpy.ndarray]

Augment through horizontal/vertical flipping.

deepblink.augment.**gaussian_noise**(*image: numpy.ndarray*, *mask: numpy.ndarray*, *mean: int = 0*) → Tuple[numpy.ndarray, numpy.ndarray]

Augment through the addition of gaussian noise.

Parameters

- **image** – Image to be augmented.
- **mask** – Corresponding prediction matrix with ground truth values.
- **mean** – Average noise pixel values added. Zero means no net difference occurs.

deepblink.augment.**illuminate**(*image: numpy.ndarray*, *mask: numpy.ndarray*) → Tuple[numpy.ndarray, numpy.ndarray]

Augment through changing illumination.

deepblink.augment.**rotate**(*image: numpy.ndarray*, *mask: numpy.ndarray*) → Tuple[numpy.ndarray, numpy.ndarray]

Augment through rotation.

deepblink.augment.**translate**(*image: numpy.ndarray*, *mask: numpy.ndarray*, *cell_size: int = 4*) → Tuple[numpy.ndarray, numpy.ndarray]

Augment through translation along all axes.

Parameters

- **image** – Image to be augmented.
- **mask** – Corresponding prediction matrix with ground truth values.
- **cell_size** – Size of one cell in the prediction matrix.

CHAPTER 2

deepblink.cli package

2.1 Module contents

Module that contains the command line interface.

Divided into <_command> files containing the parser and a call handler. Additionally, utility functions in _logger and _parseutil.

CHAPTER 3

deepblink.data module

List of functions to handle data including converting matrices <-> coordinates.

deepblink.data.**absolute_coordinate**(*coord_spot: Tuple[numpy.float32, numpy.float32], coord_cell: Tuple[numpy.float32, numpy.float32], cell_size: int = 4*) → Tuple[numpy.float32, numpy.float32]

Return the absolute image coordinate from a relative cell coordinate.

Parameters

- **coord_spot** – Relative spot coordinate in format (r, c).
- **coord_cell** – Top-left coordinate of the cell.
- **cell_size** – Size of one cell in a grid.

Returns Absolute coordinate.

deepblink.data.**get_coordinate_list**(*matrix: numpy.ndarray*, *image_size: int = 512*, *probability: float = 0.5*) → numpy.ndarray

Convert the prediction matrix into a list of coordinates.

NOTE - plt.scatter uses the x, y system. Therefore any plots must be inverted by assigning x=c, y=r!

Parameters

- **matrix** – Matrix representation of spot coordinates.
- **image_size** – Default image size the grid was layed on.
- **probability** – Cutoff value to round model prediction probability.

Returns Array of r, c coordinates with the shape (n, 2).

deepblink.data.**get_prediction_matrix**(*coords: numpy.ndarray*, *image_size: int*, *cell_size: int = 4*, *size_c: int = None*) → numpy.ndarray

Return np.ndarray of shape (n, n, 3): p, r, c format for each cell.

Parameters

- **coords** – List of coordinates in r, c format with shape (n, 2).

- **image_size** – Size of the image from which List of coordinates are extracted.
- **cell_size** – Size of one grid cell inside the matrix. A cell_size of 2 means that one cell corresponds to 2 pixels in the original image.
- **size_c** – If empty, assumes a squared image. Else the length of the r axis.

Returns *The prediction matrix as numpy array of shape (n, n, 3)* – p, r, c format for each cell.

deepblink.data.**next_multiple**(*x: int*, *k: int = 512*) → int
Calculate x's closest higher multiple of base k.

deepblink.data.**next_power**(*x: int*, *k: int = 2*) → int
Calculate x's next higher power of k.

deepblink.data.**normalize_image**(*image: numpy.ndarray*) → numpy.ndarray
Normalize image to a mean of zero and a standard deviation of one.

CHAPTER 4

deepblink.datasets package

4.1 Submodules

4.1.1 deepblink.datasets.sequence module

SequenceDataset class.

class deepblink.datasets.sequence.**SequenceDataset**(*x: numpy.ndarray*, *y: numpy.ndarray*, *batch_size: int = 16*, *augment_fn: Callable = None*, *format_fn: Callable = None*, *overfit: bool = False*)

Bases: keras.utils.data_utils.Sequence

Custom Sequence class used to feed data into model.fit.

Parameters

- **x_list** – List of inputs.
- **y_list** – List of targets.
- **batch_size** – Size of one mini-batch.
- **augment_fn** – Function to augment one mini-batch of x and y.
- **format_fn** – Function to format raw data to model input.
- **overfit** – If only one batch should be used thereby causing overfitting.

on_epoch_end() → None

Shuffle data after every epoch.

4.1.2 deepblink.datasets.spots module

SpotsDataset class.

class deepblink.datasets.spots.**SpotsDataset**(*name: str*, *cell_size: int*, *smooth_factor: float = 1*)

Bases: deepblink.datasets._datasets.Dataset

Class used to load all spots data.

Parameters

- **cell_size** – Number of pixels (from original image) constituting one cell in the prediction matrix.
- **smooth_factor** – Value used to weigh true cells, weighs false cells with 1-smooth_factor.

image_size
Check if all images have the same square shape.

load_data() → None
Load dataset into memory.

normalize_dataset() → None
Normalize all the images to have zero mean and standard deviation 1.

prepare_data() → None
Convert raw labels into labels usable for training.

In the "spots" format, training labels are stored as lists of coordinates, this format cannot be used for training. Here, this format is converted into prediction matrices.

4.2 Module contents

Datasets module with classes to handle data import and data presentation for training.

class deepblink.datasets.**Dataset**(*name: str*, **_*)

Bases: object

Simple abstract class for datasets.

Parameters **name** – Absolute path to dataset file.

data_filename
Return the absolute path to the dataset.

load_data()
Empty method to import or create data.

normalize_dataset()
Empty method to normalise images in the dataset.

prepare_data()
Empty method to prepare or convert data.

class deepblink.datasets.**SequenceDataset**(*x: numpy.ndarray*, *y: numpy.ndarray*, *batch_size: int = 16*, *augment_fn: Callable = None*, *format_fn: Callable = None*, *overfit: bool = False*)

Bases: keras.utils.data_utils.Sequence

Custom Sequence class used to feed data into model.fit.

Parameters

- **x_list** – List of inputs.
- **y_list** – List of targets.

- **batch_size** – Size of one mini-batch.
- **augment_fn** – Function to augment one mini-batch of x and y.
- **format_fn** – Function to format raw data to model input.
- **overfit** – If only one batch should be used thereby causing overfitting.

on_epoch_end() → None
Shuffle data after every epoch.

class deepblink.datasets.**SpotsDataset**(*name: str*, *cell_size: int*, *smooth_factor: float = 1*)
Bases: deepblink.datasets._datasets.Dataset

Class used to load all spots data.

Parameters

- **cell_size** – Number of pixels (from original image) constituting one cell in the prediction matrix.
- **smooth_factor** – Value used to weigh true cells, weighs false cells with 1-smooth_factor.

image_size
Check if all images have the same square shape.

load_data() → None
Load dataset into memory.

normalize_dataset() → None
Normalize all the images to have zero mean and standard deviation 1.

prepare_data() → None
Convert raw labels into labels usable for training.

In the "spots" format, training labels are stored as lists of coordinates, this format cannot be used for training. Here, this format is converted into prediction matrices.

CHAPTER 5

deepblink.inference module

Model prediction / inference functions.

deepblink.inference.**get_intensities**(*image: numpy.ndarray*, *coordinate_list: numpy.ndarray*, *radius: int*, *method: str = 'sum'*) → numpy.ndarray

Finds integrated intensities in a radius around each coordinate.

Parameters

- **image** – Input image with pixel values.
- **coordinate_list** – List of r, c coordinates in shape (n, 2).
- **radius** – Radius of kernel to determine intensities.
- **method** – How the integrated intensity should be calculated [options: sum, mean, std].

Returns Array with all integrated intensities.

deepblink.inference.**get_probabilities**(*matrix: numpy.ndarray*, *coordinates: numpy.ndarray*, *image_size: int = 512*) → numpy.ndarray

Find prediction probability given the matrix and coordinates.

Parameters

- **matrix** – Matrix representation of spot coordinates.
- **coordinates** – Coordinates at which the probability should be determined.
- **image_size** – Default image size the grid was layed on.

Returns Array with all probabilities matching the coordinates.

deepblink.inference.**predict**(*image: numpy.ndarray*, *model: keras.engine.training.Model*, *probability: Union[None, float] = None*) → numpy.ndarray

Returns a binary or categorical model based prediction of an image.

Parameters

- **image** – Image to be predicted.
- **model** – Model used to predict the image.

- **probability** – Cutoff value to round model prediction probability.

Returns List of coordinates [r, c].

CHAPTER 6

deepblink.io module

Dataset preparation functions.

deepblink.io.**basename**(*path: Union[str, os.PathLike[str]]*) → str
Returns the basename removing path and extension.

deepblink.io.**grab_files**(*path: Union[str, os.PathLike[str]], extensions: Tuple[str, ...]*) → List[str]
Grab all files in directory with listed extensions.

Parameters

- **path** – Path to files to be grabbed. Without trailing "/".
- **extensions** – List of all file extensions. Without leading ".".

Returns Sorted list of all corresponding files.

Raises OSError – Path not existing.

deepblink.io.**load_image**(*fname: Union[str, os.PathLike[str]], extensions: Tuple[str, ...] = ('tif', 'tiff', 'jpeg', 'jpg', 'png'), is_rgb: bool = False*) → numpy.ndarray
Import a single image as numpy array checking format requirements.

Parameters

- **fname** – Absolute or relative filepath of image.
- **extensions** – Allowed image extensions.
- **is_rgb** – If true, converts RGB images to grayscale.

deepblink.io.**load_model**(*fname: Union[str, os.PathLike[str]]*) → keras.engine.training.Model
Import a deepBlink model from file.

deepblink.io.**load_npz**(*fname: Union[str, os.PathLike[str]], test_only: bool = False*) → List[numpy.ndarray]
Imports the standard npz file format used for custom training and inference.

Only for files saved using "np.savez_compressed(fname, x_train, y_train...)".

Parameters

- **fname** – Path to npz file.
- **test_only** – Only return testing images and labels.

Returns A list of the required numpy arrays. If no "test_only" arguments were passed, returns [x_train, y_train, x_valid, y_valid, x_test, y_test].

Raises `ValueError` – If not all datasets are found.

`deepblink.io.`**`load_prediction`**(*fname: Union[str, os.PathLike[str]]*) → pandas.core.frame.DataFrame

Import a prediction file (output from deepBlink predict) as pandas dataframe.

`deepblink.io.`**`securename`**(*fname: str*) → str

Turns potentially unsafe names into a single, safe, alphanumeric string.

CHAPTER 7

deepblink.losses module

Functions to calculate training loss on batches of images.

While functions are comparable to the ones found in the module metrics, these rely on keras' backend and do not take raw numpy as input.

deepblink.losses.**binary_crossentropy**(*y_true*, *y_pred*)
Keras' binary crossentropy loss.

deepblink.losses.**categorical_crossentropy**(*y_true*, *y_pred*)
Keras' categorical crossentropy loss.

deepblink.losses.**combined_bce_rmse**(*y_true*, *y_pred*)
Loss that combines binary cross entropy for probability and rmse for coordinates.

The optimal values for binary crossentropy (bce) and rmse are both 0.

deepblink.losses.**combined_dice_rmse**(*y_true*, *y_pred*)
Loss that combines dice for probability and rmse for coordinates.

The optimal values for dice and rmse are both 0.

deepblink.losses.**combined_f1_rmse**(*y_true*, *y_pred*)
Difference between F1 score and root mean square error (rmse).

The optimal values for F1 score and rmse are 1 and 0 respectively. Therefore, the combined optimal value is 1.

deepblink.losses.**dice_loss**(*y_true*, *y_pred*)
Dice score loss corresponding to deepblink.losses.dice_score.

deepblink.losses.**dice_score**(*y_true*, *y_pred*, *smooth: int = 1*)
Computes the dice coefficient on a batch of tensors.

$$\text{Dice} = \frac{2 * |X \cup Y|}{|X| + |Y|}$$

ref: https://arxiv.org/pdf/1606.04797v1.pdf

Parameters

- **y_true** – Ground truth masks.
- **y_pred** – Predicted masks.
- **smooth** – Epslion value to avoid division by zero.

deepblink.losses.**f1_loss**(*y_true*, *y_pred*)
F1 score loss corresponding to deepblink.losses.f1_score.

deepblink.losses.**f1_score**(*y_true*, *y_pred*)
F1 score metric.

$$F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

The equally weighted average of precision and recall. The best value is 1 and the worst value is 0.

deepblink.losses.**precision_score**(*y_true*, *y_pred*)
Precision score metric.

Defined as `tp / (tp + fp)` where tp is the number of true positives and fp the number of false positives. Can be interpreted as the accuracy to not mislabel samples or how many selected items are relevant. The best value is 1 and the worst value is 0.

deepblink.losses.**recall_score**(*y_true*, *y_pred*)
Recall score metric.

Defined as `tp / (tp + fn)` where tp is the number of true positives and fn the number of false negatives. Can be interpreted as the accuracy of finding positive samples or how many relevant samples were selected. The best value is 1 and the worst value is 0.

deepblink.losses.**rmse**(*y_true*, *y_pred*)
Calculate root mean square error (rmse) between true and predicted coordinates.

CHAPTER 8

deepblink.metrics module

Functions to calculate training loss on single image.

deepblink.metrics.**compute_metrics**(*pred: numpy.ndarray*, *true: numpy.ndarray*, *mdist: float = 3.0*) → pandas.core.frame.DataFrame

Calculate metric scores across cutoffs.

Parameters

- **pred** – Predicted set of coordinates.
- **true** – Ground truth set of coordinates.
- **mdist** – Maximum euclidean distance in px to which F1 scores will be calculated.

Returns

DataFrame with one row per cutoff containing columns for –

- **f1_score: Harmonic mean of precision and recall based on the number of coordinates** found at different distance cutoffs (around ground truth).
- abs_euclidean: Average euclidean distance at each cutoff.
- offset: List of (r, c) coordinates denoting offset in pixels.
- f1_integral: Area under curve f1_score vs. cutoffs.
- mean_euclidean: Normalized average euclidean distance based on the total number of assignments.

deepblink.metrics.**euclidean_dist**(*x1: float*, *y1: float*, *x2: float*, *y2: float*) → float

Return the euclidean distance between two the points (x1, y1) and (x2, y2).

deepblink.metrics.**f1_integral**(*pred: numpy.ndarray*, *true: numpy.ndarray*, *mdist: float = 3.0*, *n_cutoffs: int = 50*, *return_raw: bool = False*) → Union[float, tuple]

F1 integral calculation / area under F1 vs. cutoff.

Compute the area under the curve when plotting F1 score vs cutoff values. Optimal score is ~1 (floating point inaccuracy) when F1 is achieved across all cutoff values including 0.

Parameters

- **pred** – Array of shape (n, 2) for predicted coordinates.
- **true** – Array of shape (n, 2) for ground truth coordinates.
- **mdist** – Maximum cutoff distance to calculate F1. Defaults to None.
- **n_cutoffs** – Number of intermediate cutoff steps. Defaults to 50.
- **return_raw** – If True, returns f1_scores, offsets, and cutoffs. Defaults to False.

Returns By default returns a single value in the f1_integral score. If return_raw is True, a tuple containing: * f1_scores: The non-integrated list of F1 values for all cutoffs. * offsets: Offset in r, c on predicted coords assigned to true coords * cutoffs: A list of all cutoffs used

Notes

Scipy.spatial.distance.cdist((xa*n), (xb*n)) returns a matrix of shape (xa*xb). Here we use pred as xa and true as xb. This means that the matrix has all true coordinates along the row axis and all pred coordinates along the column axis. It's transpose has the opposite. The linear assignment takes in a cost matrix and returns the coordinates to assigned costs which fall below a defined cutoff. This assigment takes the rows as reference and assignes columns to them. Therefore, the transpose matrix resulting in row and column coordinates named "true_pred_r" and "true_pred_c" respectively uses true (along matrix row axis) as reference and pred (along matrix column axis) as assigments. In other terms the assigned predictions that are close to ground truth coordinates. To now calculate the offsets, we can use the "true_pred" rows and columns to find the originally referenced coordinates. As mentioned, the matrix has true along its row axis and pred along its column axis. Thereby we can use basic indexing. The [0] and [1] index refer to the coordinates' row and column value. This offset is now used two-fold. Once to plot the scatter pattern to make sure models aren't biased in one direction and secondly to compute the euclidean distance.

The euclidean distance could not simply be summed up like with the F1 score because the different cutoffs actively influence the maximum euclidean distance score. Here, instead, we sum up all distances measured across every cutoff and then dividing by the total number of assigned coordinates. This automatically weighs models with more detections at lower cutoff scores.

`deepblink.metrics.`**`f1_score`**(*pred: numpy.ndarray*, *true: numpy.ndarray*) → Optional[float]

F1 score metric.

$$F1 = \frac{2 * precision * recall}{/} precision + recall.$$

The equally weighted average of precision and recall. The best value is 1 and the worst value is 0.

NOTE – direction dependent, arguments cant be switched!!

Parameters

- **pred** – np.ndarray of shape (n, n, 3): p, r, c format for each cell.
- **true** – np.ndarray of shape (n, n, 3): p, r, c format for each cell.

`deepblink.metrics.`**`linear_sum_assignment`**(*matrix: numpy.ndarray*, *cutoff: float = None*) → Tuple[list, list]

Solve the linear sum assignment problem with a cutoff.

A problem instance is described by matrix matrix where each matrix[i, j] is the cost of matching i (worker) with j (job). The goal is to find the most optimal assignment of j to i if the given cost is below the cutoff.

Parameters

- **matrix** – Matrix containing cost/distance to assign cols to rows.

- **cutoff** – Maximum cost/distance value assignments can have.

Returns (rows, columns) corresponding to the matching assignment.

deepblink.metrics.**offset_euclidean**(*offset: List[tuple]*) → numpy.ndarray
Calculates the euclidean distance based on row_column_offsets per coordinate.

deepblink.metrics.**precision_score**(*pred: numpy.ndarray*, *true: numpy.ndarray*) → float
Precision score metric.

Defined as `tp / (tp + fp)` where tp is the number of true positives and fp the number of false positives. Can be interpreted as the accuracy to not mislabel samples or how many selected items are relevant. The best value is 1 and the worst value is 0.

NOTE – direction dependent, arguments cant be switched!!

Parameters

- **pred** – np.ndarray of shape (n, n, 3): p, r, c format for each cell.
- **true** – np.ndarray of shape (n, n, 3): p, r, c format for each cell.

deepblink.metrics.**recall_score**(*pred: numpy.ndarray*, *true: numpy.ndarray*) → float
Recall score metric.

Defined as `tp / (tp + fn)` where tp is the number of true positives and fn the number of false negatives. Can be interpreted as the accuracy of finding positive samples or how many relevant samples were selected. The best value is 1 and the worst value is 0.

NOTE – direction dependent, arguments cant be switched!!

Parameters

- **pred** – np.ndarray of shape (n, n, 3): p, r, c format for each cell.
- **true** – np.ndarray of shape (n, n, 3): p, r, c format for each cell.

CHAPTER 9

deepblink.models package

9.1 Submodules

9.1.1 deepblink.models.spots module

SpotsModel class.

class deepblink.models.spots.**SpotsModel**(***kwargs*)

Bases: deepblink.models._models.Model

Class to predict spot localization; see base class.

metrics

List of all metrics recorded during training.

predict_on_image(*image: numpy.ndarray*) → numpy.ndarray

Predict on a single input image.

9.2 Module contents

Models module with the training loop and logic to handle data which feeds into the loop.

class deepblink.models.**Model**(*augmentation_args: Dict[KT, VT], dataset_args: Dict[KT, VT], dataset_cls: deepblink.datasets._datasets.Dataset, network_args: Dict[KT, VT], network_fn: Callable, loss_fn: Callable, optimizer_fn: Callable, train_args: Dict[KT, VT], pre_model: keras.engine.training.Model = None, **kwargs*)

Bases: object

Base class, to be subclassed by predictors for specific type of data, e.g. spots.

Parameters

- **dataset_args** – Dataset arguments containing - version, cell_size, flip, illuminate, rotate, gaussian_noise, and translate.
- **dataset_cls** – Specific dataset class.
- **network_args** – Network arguments containing - n_channels.
- **network_fn** – Network function returning a built model.
- **loss_fn** – Loss function.
- **optimizer_fn** – Optimizer function.
- **train_args** – Training arguments containing - batch_size, epochs, learning_rate.
- **pre_model** – Loaded, pre-trained model to bypass a new network creation.

Kwargs: batch_format_fn: Formatting function added in the specific model, e.g. spots. batch_augment_fn: Same as batch_format_fn for augmentation.

evaluate(*x: numpy.ndarray*, *y: numpy.ndarray*) → List[float]
Evaluate on images / masks and return l2 norm and f1 score.

fit(*dataset: deepblink.datasets._datasets.Dataset*, *augment_val: bool = True*, *callbacks: list = None*) → None
Training loop.

metrics
Return metrics.

class deepblink.models.**SpotsModel**(***kwargs*)
Bases: deepblink.models._models.Model

Class to predict spot localization; see base class.

metrics
List of all metrics recorded during training.

predict_on_image(*image: numpy.ndarray*) → numpy.ndarray
Predict on a single input image.

CHAPTER 10

deepblink.networks package

10.1 Submodules

10.1.1 deepblink.networks.unet module

UNet architecture.

deepblink.networks.unet.**unet**(*dropout: float = 0.2*, *cell_size: int = 4*, *filters: int = 5*, *ndown: int = 2*, *l2: float = 1e-06*, *block: str = 'convolutional'*) → keras.engine.training.Model

Unet model with second, cell size dependent encoder.

Note that "convolution" is the currently best block.

Parameters

- **dropout** – Percentage of dropout before each MaxPooling step.
- **cell_size** – Size of one cell in the prediction matrix.
- **filters** – Log_2 number of filters in the first inception block.
- **ndown** – Downsampling steps in the first encoder / decoder.
- **l2** – L2 value for kernel and bias regularization.
- **block** – Type of block in each layer. [options: convolutional, inception, residual]

10.2 Module contents

Networks folder.

Contains functions returning the base architectures of used models.

deepblink.networks.**unet**(*dropout: float = 0.2*, *cell_size: int = 4*, *filters: int = 5*, *ndown: int = 2*, *l2: float = 1e-06*, *block: str = 'convolutional'*) → keras.engine.training.Model

Unet model with second, cell size dependent encoder.

Note that "convolution" is the currently best block.

Parameters

- **dropout** – Percentage of dropout before each MaxPooling step.
- **cell_size** – Size of one cell in the prediction matrix.
- **filters** – Log_2 number of filters in the first inception block.
- **ndown** – Downsampling steps in the first encoder / decoder.
- **l2** – L2 value for kernel and bias regularization.
- **block** – Type of block in each layer. [options: convolutional, inception, residual]

CHAPTER 11

deepblink.optimizers module

Optimizers are used to update weight parameters in a neural network.

The learning rate defines what stepsizes are taken during one iteration of training. This file contains functions to return standard or custom optimizers.

deepblink.optimizers.**adam**(*learning_rate: float*)
Keras' adam optimizer with a specified learning rate.

deepblink.optimizers.**amsgrad**(*learning_rate: float*)
Keras' amsgrad optimizer with a specified learning rate.

deepblink.optimizers.**rmsprop**(*learning_rate: float*)
Keras' rmsprop optimizer with a specified learning rate.

CHAPTER 12

deepblink.training module

Training functions.

deepblink.training.**run_experiment**(*cfg: Dict[KT, VT], pre_model: keras.engine.training.Model = None*)

Run a training experiment.

Configuration file can be generated using deepblink config.

Parameters

- **cfg** – Dictionary configuration file.
- **pre_model** – Pre-trained model if not training from scratch.

deepblink.training.**train_model**(*model: deepblink.models._models.Model, dataset: deepblink.datasets._datasets.Dataset, cfg: Dict[KT, VT], run_name: str = 'model', use_wandb: bool = True*) → deepblink.models._models.Model

Model training loop with callbacks.

Parameters

- **model** – Model class with the .fit method.
- **dataset** – Dataset class with access to train and validation images.
- **cfg** – Configuration file equivalent to the one used in pink.training.run_experiment.
- **run_name** – Name given to the model.h5 file saved.
- **use_wandb** – If Wandb should be used.

CHAPTER 13

deepblink.util module

Utility helper functions.

deepblink.util.**delete_non_unique_columns**(*df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame
Deletes DataFrame columns that only contain one (non-unique) value.

deepblink.util.**get_from_module**(*path: str*, *attribute: str*) → Callable
Grab an attribute (e.g. class) from a given module path.

deepblink.util.**predict_pixel_size**(*fname: Union[str, os.PathLike[str]]*) → Tuple[float, float]
Predict the pixel size based on tifffile metadata.

deepblink.util.**predict_shape**(*shape: tuple*) → str
Predict the channel-arangement based on common standards.

Assumes the following things: * x, y are the two largest axes * rgb only if the last axis is 3 * up to 4 channels * "fill up order" is c, z, t

Parameters **shape** – To be predicted shape. Output from np.ndarray.shape

deepblink.util.**relative_shuffle**(*x: Union[list, numpy.ndarray]*, *y: Union[list, numpy.ndarray]*) → Tuple[Union[list, numpy.ndarray], Union[list, numpy.ndarray]]
Shuffles x and y keeping their relative order.

deepblink.util.**remove_falses**(*tup: tuple*) → tuple
Removes all false occurences from a tuple.

deepblink.util.**train_valid_split**(*x_list: list*, *y_list: list*, *valid_split: float = 0.2*, *shuffle: bool = True*) → Iterable[list]
Split two lists (usually input and ground truth).

Splitting into random training and validation sets with an optional shuffling.

Parameters

- **x_list** – First list of items. Typically input data.
- **y_list** – Second list of items. Typically labeled data.

- **valid_split** – Number between 0-1 to denote the percentage of examples used for validation.

Returns (x_train, x_valid, y_train, y_valid) splited lists containing training or validation examples respectively.

Python Module Index

d

deepblink.augment, 1
deepblink.cli, 3
deepblink.data, 5
deepblink.datasets, 8
deepblink.datasets.sequence, 7
deepblink.datasets.spots, 7
deepblink.inference, 11
deepblink.io, 13
deepblink.losses, 15
deepblink.metrics, 17
deepblink.models, 21
deepblink.models.spots, 21
deepblink.networks, 23
deepblink.networks.unet, 23
deepblink.optimizers, 25
deepblink.training, 27
deepblink.util, 29

Index

A

absolute_coordinate() (*in module deepblink.data*), 5
adam() (*in module deepblink.optimizers*), 25
amsgrad() (*in module deepblink.optimizers*), 25
augment_batch_baseline() (*in module deepblink.augment*), 1

B

basename() (*in module deepblink.io*), 13
binary_crossentropy() (*in module deepblink.losses*), 15

C

categorical_crossentropy() (*in module deepblink.losses*), 15
combined_bce_rmse() (*in module deepblink.losses*), 15
combined_dice_rmse() (*in module deepblink.losses*), 15
combined_f1_rmse() (*in module deepblink.losses*), 15
compute_metrics() (*in module deepblink.metrics*), 17

D

data_filename (*deepblink.datasets.Dataset attribute*), 8
Dataset (*class in deepblink.datasets*), 8
deepblink.augment (*module*), 1
deepblink.cli (*module*), 3
deepblink.data (*module*), 5
deepblink.datasets (*module*), 8
deepblink.datasets.sequence (*module*), 7
deepblink.datasets.spots (*module*), 7
deepblink.inference (*module*), 11
deepblink.io (*module*), 13
deepblink.losses (*module*), 15
deepblink.metrics (*module*), 17
deepblink.models (*module*), 21
deepblink.models.spots (*module*), 21
deepblink.networks (*module*), 23
deepblink.networks.unet (*module*), 23
deepblink.optimizers (*module*), 25
deepblink.training (*module*), 27
deepblink.util (*module*), 29
delete_non_unique_columns() (*in module deepblink.util*), 29
dice_loss() (*in module deepblink.losses*), 15
dice_score() (*in module deepblink.losses*), 15

E

euclidean_dist() (*in module deepblink.metrics*), 17
evaluate() (*deepblink.models.Model method*), 22

F

f1_integral() (*in module deepblink.metrics*), 17
f1_loss() (*in module deepblink.losses*), 16
f1_score() (*in module deepblink.losses*), 16
f1_score() (*in module deepblink.metrics*), 18
fit() (*deepblink.models.Model method*), 22
flip() (*in module deepblink.augment*), 1

G

gaussian_noise() (*in module deepblink.augment*), 1
get_coordinate_list() (*in module deepblink.data*), 5
get_from_module() (*in module deepblink.util*), 29
get_intensities() (*in module deepblink.inference*), 11
get_prediction_matrix() (*in module deepblink.data*), 5
get_probabilities() (*in module deepblink.inference*), 11
grab_files() (*in module deepblink.io*), 13

I

illuminate() (*in module deepblink.augment*), 2
image_size (*deepblink.datasets.spots.SpotsDataset attribute*), 8
image_size (*deepblink.datasets.SpotsDataset attribute*), 9

L

linear_sum_assignment() (*in module deepblink.metrics*), 18
load_data() (*deepblink.datasets.Dataset method*), 8
load_data() (*deepblink.datasets.spots.SpotsDataset method*), 8
load_data() (*deepblink.datasets.SpotsDataset method*), 9
load_image() (*in module deepblink.io*), 13
load_model() (*in module deepblink.io*), 13
load_npz() (*in module deepblink.io*), 13
load_prediction() (*in module deepblink.io*), 14

M

metrics (*deepblink.models.Model attribute*), 22
metrics (*deepblink.models.spots.SpotsModel attribute*), 21
metrics (*deepblink.models.SpotsModel attribute*), 22
Model (*class in deepblink.models*), 21

N

next_multiple() (*in module deepblink.data*), 6
next_power() (*in module deepblink.data*), 6
normalize_dataset() (*deepblink.datasets.Dataset method*), 8
normalize_dataset() (*deepblink.datasets.spots.SpotsDataset method*), 8
normalize_dataset() (*deepblink.datasets.SpotsDataset method*), 9
normalize_image() (*in module deepblink.data*), 6

O

offset_euclidean() (*in module deepblink.metrics*), 19
on_epoch_end() (*deepblink.datasets.sequence.SequenceDataset method*), 7
on_epoch_end() (*deepblink.datasets.SequenceDataset method*), 9

P

precision_score() (*in module deepblink.losses*), 16
precision_score() (*in module deepblink.metrics*), 19
predict() (*in module deepblink.inference*), 11
predict_on_image() (*deepblink.models.spots.SpotsModel method*), 21
predict_on_image() (*deepblink.models.SpotsModel method*), 22
predict_pixel_size() (*in module deepblink.util*), 29
predict_shape() (*in module deepblink.util*), 29
prepare_data() (*deepblink.datasets.Dataset method*), 8
prepare_data() (*deepblink.datasets.spots.SpotsDataset method*), 8
prepare_data() (*deepblink.datasets.SpotsDataset method*), 9

R

recall_score() (*in module deepblink.losses*), 16
recall_score() (*in module deepblink.metrics*), 19
relative_shuffle() (*in module deepblink.util*), 29
remove_falses() (*in module deepblink.util*), 29
rmse() (*in module deepblink.losses*), 16
rmsprop() (*in module deepblink.optimizers*), 25
rotate() (*in module deepblink.augment*), 2
run_experiment() (*in module deepblink.training*), 27

S

securename() (*in module deepblink.io*), 14
SequenceDataset (*class in deepblink.datasets*), 8
SequenceDataset (*class in deepblink.datasets.sequence*), 7
SpotsDataset (*class in deepblink.datasets*), 9
SpotsDataset (*class in deepblink.datasets.spots*), 7
SpotsModel (*class in deepblink.models*), 22
SpotsModel (*class in deepblink.models.spots*), 21

T

train_model() (*in module deepblink.training*), 27
train_valid_split() (*in module deepblink.util*), 29
translate() (*in module deepblink.augment*), 2

U

unet() (*in module deepblink.networks*), 23
unet() (*in module deepblink.networks.unet*), 23