Beetles

Release 1.0

KP Labs

Dec 18, 2020

CONTENTS:

CHAPTER

ONE

INSTALLATION

For the environment management in our repo, we utilize *conda*.

Required dependencies are stored in the *environment.yml* file. To create the environment with required packages, execute the following command:

```
conda env create -f environment.yml
```

Because of the fact that we utilize the Xillinx DNNDK tool for model quantization and compilation, the *tensorflow* package has to be installed manually. Please refer to the official documentation, where the process is fully described: (https://www.xilinx.com/support/documentation/sw_manuals/ai_inference/v1_6/ug1327-dnndk-user-guide.pdf - Chapter 1: Quick Start - Tensorflow Version: Installing with Anaconda).

Keep in mind, that the DNNDK tool requires **Ubuntu** 14.04, 16.04 or 18.04.

CHAPTER

TWO

MACHINE-LEARNING

2.1 ml_intuition package

2.1.1 Subpackages

ml_intuition.data package

Submodules

ml_intuition.data.input_fn module

File containing functions which calibrate input for frozen graphs used for quantization in scripts/quantize.sh. All arguments to such functions are passed through the shell environment, because they are executed through the internal decent tools.

ml_intuition.data.input_fn.**calibrate_2d_input**(*iter*)
Return dictionary with a batch of the training data to be used for quantization calibration. One dimension and the end is added and the min max normalization is performed.

Parameters **iter** (int) – Int object indicating the calibration step

Return type Dict[str, ndarray]

Returns Dict with name of the input node as key and training samples in np.ndarray as value

ml_intuition.data.io module

All I/O related functions

ml_intuition.data.io.**extract_set**(*data_path*, *dataset_key*)
Function for loading a h5 format dataset as a dictionary of samples, labels, min and max values.

Parameters

- **data_path** (str) – Path to the dataset.
- **dataset_key** (str) – Key for dataset.

Return type Dict[str, Union[ndarray, float]]

Returns Dictionary containing labels, data, min and max values.

ml_intuition.data.io.**load_metrics**(*experiments_path*, *filename=None*)
Load metrics to a dictionary.

Parameters

- **experiments_path** (str) – Path to the experiments directory.
- **filename** (Optional[str]) – Name of the file holding metrics. Defaults to 'inference_metrics.csv'.

Return type Dict[List, List]

Returns Dictionary containing all metric names and values from all experiments.

ml_intuition.data.io.**load_npy**(*data_file_path*, *gt_input_path*, *use_unmixing=False*)
Load .npy data and GT from specified paths

Parameters

- **data_file_path** (str) – Path to the data .npy file
- **gt_input_path** (str) – Path to the GT .npy file
- **use_unmixing** (bool) – Boolean indicating whether to perform experiments on the unmixing datasets, where classes in each pixel are present as abundance fractions.

Return type Tuple[ndarray, ndarray]

Returns Tuple with loaded data and GT

ml_intuition.data.io.**load_pb**(*path_to_pb*)
Load .pb file as a graph

Parameters **path_to_pb** (str) – Path to the .pb file

Return type GraphDef

Returns Loaded graph

ml_intuition.data.io.**load_processed_h5**(*data_file_path*)
Load processed dataset containing the train, validation and test subsets with corresponding samples and labels.

Parameters **data_file_path** (str) – Path to the .h5 file.

Return type Dict

Returns Dictionary containing train, validation and test subsets.

ml_intuition.data.io.**load_satellite_h5**(*data_file_path*)
Load hyperspectral cube and ground truth transformation matrix from .h5 file

Parameters **data_file_path** (str) – Path to the .h5 file

Return type Tuple[ndarray, ndarray]

Returns Hyperspectral cube and transformation matrix, both as np.ndarray

ml_intuition.data.io.**load_tiff**(*file_path*)
Load tiff image into np.ndarray

Parameters **file_path** (str) – Path to the .tiff file

Return type ndarray

Returns Loaded image as np.ndarray

ml_intuition.data.io.**read_min_max**(*path*)
Read min and max value from a .csv file

Parameters **path** (str) – Path to the .csv file containing min and max value

Return type Tuple[float, float]

Returns Tuple with min and max

ml_intuition.data.io.**save_confusion_matrix**(*matrix*, *dest_path*, *filename='confusion_matrix'*)
Save confusion matrix to provided destination. If filename is not provided 'confusion_matrix.csv' will be used.

Parameters

- **matrix** (ndarray) – Matrix to save
- **dest_path** (str) – Destination folder
- **filename** (str) – Name of the file, defaults to 'confusion_matrix.csv'

Return type None

Returns None

ml_intuition.data.io.**save_md5**(*output_path*, *train_x*, *train_y*, *val_x*, *val_y*, *test_x*, *test_y*)
Save provided data as .md5 file

Parameters

- **output_path** (str) – Path to the filename
- **train_x** (ndarray) – Train set
- **train_y** (ndarray) – Train labels
- **val_x** (ndarray) – Validation set
- **val_y** (ndarray) – Validation labels
- **test_x** (ndarray) – Test set
- **test_y** (ndarray) – Test labels

Return type None

Returns None

ml_intuition.data.io.**save_metrics**(*dest_path*, *metrics*, *file_name=None*)
Save given dictionary of metrics.

Parameters

- **dest_path** (str) – Destination path.
- **file_name** (Optional[str]) – Name to save the file.
- **metrics** (Dict[str, List]) – Dictionary containing all metrics.

ml_intuition.data.loggers module

Helper function logging data into MLFlow

ml_intuition.data.loggers.**log_dict_as_str_to_mlflow**(*dict_as_string*)
Log a string which represents a dictionary to MLflow

Parameters **dict_as_string** (str) – A string with dictionary format

Return type None

Returns None

ml_intuition.data.loggers.**log_metrics_to_mlflow**(*metrics*, *fair=False*)
Log provided metrics to mlflow

Parameters

- **metrics** (Dict[str, float]) – Metrics in a dictionary
- **fair** (bool) – Whether to add '_fair' suffix to the metrics name

Return type None

Returns None

ml_intuition.data.loggers.**log_params_to_mlflow**(*args*)
Log provided arguments as dictionary to mlflow.

Parameters **args** (Dict) – Arguments to log

Return type None

ml_intuition.data.loggers.**log_tags_to_mlflow**(*run_name*)
Log tags to mlflow based on provided args

Parameters **run_name** (str) – Name of the current run

Return type None

Returns None

ml_intuition.data.noise module

class ml_intuition.data.noise.**BaseNoise**(*params*)
Bases: abc.ABC

abstract **__call__**(**args*, ***kwargs*)
Each subclass should implement this method.

Parameters

- **args** – Arbitrary list of arguments.
- **kwargs** – Arbitrary dictionary of arguments.

static **get_proba**(*n_samples*, *prob*)

Return type int

class ml_intuition.data.noise.**Gaussian**(*params*)
Bases: *ml_intuition.data.noise.BaseNoise*

__call__(*data*, *labels*)
Perform Gaussian noise injection.

Parameters

- **data** (ndarray) – Input data that will undergo noise injection.
- **labels** (ndarray) – Class value for each data point.

Return type List[ndarray]

Returns List containing the noisy data and the class labels.

class ml_intuition.data.noise.**Impulsive**(*params*)
Bases: *ml_intuition.data.noise.BaseNoise*

__call__(*data*, *labels*)
Perform impulsive noise injection.

Parameters

- **data** (ndarray) – Input data that will undergo noise injection.
- **labels** (ndarray) – Class value for each data point.

Return type List[ndarray]

Returns List containing the noisy data and the class labels.

class ml_intuition.data.noise.**Params**(*pa: float = None*, *pb: float = None*, *bc: bool = None*, *mean: float = None*, *std: float = None*, *pw: float = None*)

Bases: tuple

Parameters of noise injection.

pa - Fraction of noisy pixels, the number of affected samples is calculated by: floor(n_samples * pa).

pb - Fraction of noisy bands. When established the number of samples that undergo noise injection, for each sample the: floor(n_bands * pb) bands are affected.

bc - Boolean indicating whether the indexes of affected bands are constant for each sample. When set to: False, different bands can be augmented with noise for each pixel.

mean - Gaussian noise parameter, the mean of the normal distribution.

std - Standard deviation of the normal distribution for Gaussian noise.

pw - Ratio of whitened pixels for the affected set of samples in the Impulsive noise injection.

Create new instance of Params(pa, pb, bc, mean, std, pw)

property bc
Alias for field number 2

property mean
Alias for field number 3

property pa
Alias for field number 0

property pb
Alias for field number 1

property pw
Alias for field number 5

property std
Alias for field number 4

class ml_intuition.data.noise.**Shot**(*params*)

Bases: *ml_intuition.data.noise.BaseNoise*

__call__(*data*, *labels*)
Perform shot noise injection.

Parameters

- **data** (ndarray) – Input data that will undergo noise injection.
- **labels** (ndarray) – Class value for each data point.

Return type List[ndarray]

Returns List containing the noisy data and the class labels.

ml_intuition.data.noise.**get_all_noise_functions**(*noise*)
Get all specified noise functions.

Parameters **noise** (List[str]) – List the of noise injection methods.

Return type List

Returns List of all noise injection functions.

ml_intuition.data.noise.**get_noise_functions**(*noise*, *noise_params*)
Helper function for getting all noise injector instances.

Parameters

- **noise** (List[str]) – List the of noise injection methods.
- **noise_params** (str) – Parameters of the noise injections.

Return type List[*BaseNoise*]

Returns List of instances of noise injectors functions.

ml_intuition.data.noise.**inject_noise**(*data_source*, *affected_subsets*, *noise_injectors*, *noise_params*)
Inject noise into given subsets.

Parameters

- **data_source** (Dict) – Dictionary containing subsets.
- **affected_subsets** (List[str]) – List of names of subsets that will undergo noise injection.
- **noise_injectors** (List[str]) – List of names of noise injection methods.
- **noise_params** (str) – Parameters of the noise injection.

ml_intuition.data.preprocessing module

All function related to the data preprocessing step of the pipeline.

ml_intuition.data.preprocessing.**align_ground_truth**(*cube_2d_shape*, *ground_truth*, *cube_to_gt_transform*)
Align original labels to match the satellite hyperspectral cube using transformation matrix

Parameters

- **cube_2d_shape** (Tuple[int, int]) – Shape of the hyperspectral data cube
- **ground_truth** (ndarray) – Original labels as 2D array
- **cube_to_gt_transform** (ndarray) – Cube to ground truth transformation matrix

Return type ndarray

Returns Transformed ground truth

ml_intuition.data.preprocessing.**get_padded_cube**(*data*, *padding_size*)
Pad the cube with zeros

Parameters

- **data** (ndarray) – Data to pad
- **padding_size** (int) – Size of the padding for each side

Return type ndarray

Returns Padded cube

ml_intuition.data.preprocessing.**normalize_labels**(*labels*)
Normalize labels so that they always start from 0

Parameters **labels** (ndarray) – labels to normalize

Return type ndarray

Returns Normalized labels

ml_intuition.data.preprocessing.**remove_nan_samples**(*data*, *labels*)
Remove samples which contain only nan values

Parameters

- **data** (ndarray) – Data with dimensions [SAMPLES, . . .]
- **labels** (ndarray) – Corresponding labels

Return type Tuple[ndarray, ndarray]

Returns Data and labels with removed samples containing nans

ml_intuition.data.preprocessing.**reshape_cube_to_2d_samples**(*data*, *labels*, *channels_idx=0*, *use_unmixing=False*)

Reshape the data and labels from [CHANNELS, HEIGHT, WIDTH] to [PIXEL,CHANNELS, 1], so it fits the 2D Convolutional modules.

Parameters

- **data** (ndarray) – Data to reshape.
- **labels** (ndarray) – Corresponding labels.
- **channels_idx** (int) – Index at which the channels are located in the provided data file.
- **use_unmixing** (bool) – Boolean indicating whether to perform experiments on the unmixing datasets, where classes in each pixel are present as fractions.

Returns Reshape data and labels

Return type tuple with reshaped data and labels

ml_intuition.data.preprocessing.**reshape_cube_to_3d_samples**(*data*, *labels*, *neighborhood_size=5*, *background_label=0*, *channels_idx=0*, *use_unmixing=False*)
Reshape data to a array of dimensionality: [N_SAMPLES, HEIGHT, WIDTH, CHANNELS] and the labels to dimensionality of: [N_SAMPLES, N_CLASSES]

Parameters

- **data** (ndarray) – Data passed as array.
- **labels** (ndarray) – Labels passed as array.
- **neighborhood_size** (int) – Length of the spatial patch.
- **background_label** (int) – Label to filter out the background.
- **channels_idx** (int) – Index of the channels.

- **use_unmixing** (bool) – Boolean indicating whether to perform experiments on the unmixing datasets, where classes in each pixel are present as abundances fractions.

Return type Tuple of data and labels reshaped to 3D format.

ml_intuition.data.preprocessing.**spectral_angle_mapper**(*data*, *endmembers*, *channel_idx*)

Calculate the spectral angle mapper between the HSI pixels and endmembers.

Parameters

- **data** (ndarray) – Data cube.
- **endmembers** (ndarray) – The matrix containing the spectra for each class of dimensionality [n_classes, n_bands].
- **channel_idx** (int) – Index of bands in HSI.

Return type Tuple[ndarray, ndarray]

Returns The tuple of both, normalized and unnormalized angle matrix for each pixel in HSI.

ml_intuition.data.preprocessing.**train_val_test_split**(*data*, *labels*, *train_size=0.8*, *val_size=0.1*, *stratified=True*, *seed=0*)

Split the data into train, val and test sets. The size of the training set is set by the train_size parameter. All the remaining samples will be treated as a test set

Parameters

- **data** (ndarray) – Data with the [SAMPLES, . . .] dimensions
- **labels** (ndarray) – Vector with corresponding labels
- **train_size** (Union[List, float, int]) – If float, should be between 0.0 and 1.0, If stratified = True, it represents percentage of each class to be extracted. If float and stratified = False, it represents percentage of the whole dataset to be extracted with samples drawn randomly, regardless of their class. If int and stratified = True, it represents number of samples to be drawn from each class. If int and stratified = False, it represents overall number of samples to be drawn regardless of their class, randomly. Defaults to 0.8
- **val_size** (float) – Should be between 0.0 and 1.0. Represents the percentage of each class from the training set to be extracted as a validation set, defaults to 0.1
- **stratified** (bool) – Indicated whether the extracted training set should be stratified, defaults to True
- **seed** (int) – Seed used for data shuffling

Return type Tuple[ndarray, ndarray, ndarray, ndarray, ndarray, ndarray]

Returns train_x, train_y, val_x, val_y, test_x, test_y

Raises **AssertionError** – When wrong type is passed as train_size

ml_intuition.data.transforms module

Module containing all the transformations that can be done on a dataset.

class ml_intuition.data.transforms.**BaseTransform**
Bases: abc.ABC

abstract __call__(**args*, ***kwargs*)
Each subclass should implement this method.

Parameters

- **args** – Arbitrary list of arguments.
- **kwargs** – Arbitrary dictionary of arguments.

class ml_intuition.data.transforms.**ExtractCentralPixelSpectrumTransform**(*neighborhood_size*)
Bases: *ml_intuition.data.transforms.BaseTransform*

Extract central pixel from each spatial sample.

Parameters **neighborhood_size** (int) – The spatial size of the patch.

__call__(*samples*, *labels*)
" Transform the labels for unsupervised unmixing problem. The label is the central pixel of each sample patch.

Parameters

- **samples** (ndarray) – Input samples.
- **labels** (ndarray) – Central pixel of each sample.

Return type List[ndarray]

Returns List containing the input samples and its targets as central pixel.

class ml_intuition.data.transforms.**MinMaxNormalize**(*min_*, *max_*)
Bases: *ml_intuition.data.transforms.BaseTransform*

Normalize each sample.

Parameters

- **min** – Minimum value of features.
- **max** – Maximum value of features.

__call__(*samples*, *labels*)
" Perform min-max normalization on the passed samples.

Parameters

- **samples** (ndarray) – Input samples that will undergo normalization.
- **labels** (ndarray) – Class values for each sample.

Return type List[ndarray]

Returns List containing the normalized samples and the class labels.

class ml_intuition.data.transforms.**OneHotEncode**(*n_classes*)
Bases: *ml_intuition.data.transforms.BaseTransform*

Initializer of the one-hot encoding transformation.

Parameters **n_classes** (int) – Number of classes.

__call__(*samples*, *labels*)
Perform one-hot encoding on the passed labels.

Parameters

- **samples** (ndarray) – Input samples.
- **labels** (ndarray) – Class values for each sample that will undergo one-hot encoding.

Return type List[ndarray]

Returns List containing the samples and the one-hot encoded class labels.

class ml_intuition.data.transforms.**PerBandMinMaxNormalization**(*min_*, *max_*)
Bases: *ml_intuition.data.transforms.BaseTransform*

SPECTRAL_DIM = -1

__call__(*samples*, *labels*)
Perform per-band min-max normalization. Each band is treated as a separate feature.

Parameters

- **samples** (ndarray) – Input samples that will undergo transformation.
- **labels** (ndarray) – Abundance vectors for each sample.

Return type List[ndarray]

Returns List containing the normalized samples and the abundance vectors.

static get_min_max_vectors(*data_cube*)
Get the min-max vectors for each spectral band.

Parameters **data_cube** (ndarray) – Hyperspectral data cube, with bands in the last dimension and spatial features in the first two dimensions.

Return type Dict[str, ndarray]

Returns Dictionary containing the min as well as the max vectors for each band.

class ml_intuition.data.transforms.**RNNSpectralInputTransform**
Bases: *ml_intuition.data.transforms.BaseTransform*

__call__(*samples*, *labels*)
" Transform the input samples to fit the recurrent neural network (RNN) input. This is performed for the pixel-based model; the input sample includes only spectral bands.

Parameters

- **samples** (ndarray) – Input samples that will undergo the transformation.
- **labels** (ndarray) – Class values for each sample.

Return type List[ndarray]

Returns List containing the normalized samples and the class labels.

class ml_intuition.data.transforms.**SpectralTransform**(***kwargs*)
Bases: *ml_intuition.data.transforms.BaseTransform*

Initializer of the spectral transformation.

__call__(*samples*, *labels*)
Transform 1D samples along the spectral axis. Only the spectral features are present for each sample in the dataset.

Parameters

- **samples** (ndarray) – Input samples that will undergo transformation.
- **labels** (ndarray) – Class value for each samples.

Return type List[ndarray]

Returns List containing the transformed samples and the class labels.

ml_intuition.data.transforms.**apply_transformations**(*data*, *transformations*)
Apply each transformation from provided list

Parameters

- **data** (Dict) – Dictionary with 'data' and 'labels' keys holding np.ndarrays
- **transformations** (List[*BaseTransform*]) – List of transformations

Return type Dict

Returns Transformed data, in the same format as input

Module contents

2.1.2 Submodules

2.1.3 ml_intuition.models module

All models that are used in the project.

class ml_intuition.models.**Ensemble**(*models=None*, *voting='hard'*)
Bases: object

Ensemble for using multiple models for prediction

Parameters

- **models** (Union[List[Sequential], List[str], Sequential, None]) – Either list of tf.keras.models.Sequential models, or a list of paths to the models (can't mix both).
- **voting** (str) – If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities. If 'classifier', a Random Forest trained on train set predictions is used as a voter.

generate_models_with_noise(*copies=5*, *mean=None*, *std=None*, *seed=None*)
Generate new models by injecting Gaussian noise into the original model's weights.

Parameters

- **copies** (int) – Number of models to generate
- **mean** (Optional[float]) – Mean used to draw noise from normal distribution. If None, it will be calculated from the layer itself.
- **std** (Optional[float]) – Standard deviation used to draw noise from normal distribution. If None, it will be calculated from the layer itself.
- **seed** – Seed for random number generator.

Return type None

Returns None

static inject_noise_to_weights(*weights*, *mean*, *std*)
Inject noise into all layers

Parameters

- **weights** (List[ndarray]) – List of weights for each layer
- **mean** (float) – Mean used to draw noise from normal distribution.
- **std** (float) – Std used to draw noise from normal distribution.

Return type List[ndarray]

Returns Modified list of weights

predict(*data*, *batch_size=1024*)
Return predicted classes

Parameters

- **data** (Union[ndarray, List[ndarray]]) – Either a single dataset which will be fed into all of the models, or a list of datasets, unique for each model
- **batch_size** (int) – Size of the batch used for prediction

Return type ndarray

Returns Predicted classes

predict_probabilities(*data*, *batch_size=1024*)
Return predicted classes

Parameters

- **data** (Union[ndarray, List[ndarray]]) – Either a single dataset which will be fed into all of the models, or a list of datasets, unique for each model
- **batch_size** (int) – Size of the batch used for prediction

Return type ndarray

Returns Predicted probabilities for each model and class. Shape: [Models, Samples, Classes]

train_ensemble_predictor(*data*, *labels*)
Train the Random Forest on train set predictions

Parameters

- **data** (ndarray) – Predictions of the models on training set
- **labels** (ndarray) – Corresponding las

Return type None

vote(*predictions*)
Perform voting process on provided predictions

Parameters **predictions** (ndarray) – Predictions of all models

Return type ndarray

Returns Predicted classes

ml_intuition.models.**model_2d**(*kernel_size*, *n_kernels*, *n_layers*, *input_size*, *n_classes*, ***kwargs*)
2D model which consists of 2D convolutional blocks.

Parameters

- **kernel_size** (int) – Size of the convolutional kernel.
- **n_kernels** (int) – Number of kernels, i.e., the activation maps in each layer.

- **n_layers** (int) – Number of layers in the network.
- **input_size** (int) – Number of input channels, i.e., the number of spectral bands.
- **n_classes** (int) – Number of classes.
- **kwargs** – Additional arguments.

Return type Sequential

Returns Compiled model ready for training

ml_intuition.models.**model_3d_deep**(*n_classes*, *input_size*, ***kwargs*)
Deep 3D convolutional model from: https://arxiv.org/pdf/1907.11935.pdf

Parameters

- **n_classes** (int) – Number of classes in the dataset
- **input_size** (int) – Size of the input sample
- **kwargs** – Additional arguments.

Return type Sequential

Returns Compiled model ready for training

ml_intuition.models.**model_3d_mfl**(*kernel_size*, *n_kernels*, *n_classes*, *input_size*, ***kwargs*)
Multiple Features Learning model from: https://ieeexplore.ieee.org/document/6882821/figures#figures

Parameters

- **kernel_size** (int) – Size of the convolutional kernel
- **n_kernels** (int) – Number of kernels in each convolutional layer
- **n_classes** (int) – Number of classes in the dataset
- **input_size** (int) – Size of the input sample
- **kwargs** – Additional arguments.

Return type Sequential

Returns Compiled model ready for training

ml_intuition.models.**pool_model_2d**(*kernel_size*, *n_kernels*, *n_layers*, *input_size*, *n_classes*, ***kwargs*)
2D model which consists of 2D convolutional layers and 2D pooling layers.

Parameters

- **kernel_size** (int) – Size of the convolutional kernel.
- **n_kernels** (int) – Number of kernels, i.e., the activation maps in each layer.
- **n_layers** (int) – Number of layers in the network.
- **input_size** (int) – Number of input channels, i.e., the number of spectral bands.
- **n_classes** (int) – Number of classes.
- **kwargs** – Additional arguments.

Return type Sequential

Returns Compiled model ready for training

ml_intuition.models.**unmixing_cube_based_cnn**(*n_classes*, *input_size*, ***kwargs*)

Model for cube-based supervised hyperspectral unmixing proposed in the following publication (Chicago style citation):

Zhang, Xiangrong, Yujia Sun, Jingyan Zhang, Peng Wu, and Licheng Jiao. "Hyperspectral unmixing via deep convolutional neural networks." IEEE Geoscience and Remote Sensing Letters 15, no. 11 (2018): 1755-1759.

Parameters

- **n_classes** (int) – Number of classes.
- **input_size** (int) – Number of input spectral bands.
- **kwargs** – Additional arguments.

Return type Sequential

Returns Model proposed in the publication listed above.

ml_intuition.models.**unmixing_cube_based_dcae**(*n_classes*, *input_size*, ***kwargs*)

Model for cube-based unsupervised hyperspectral unmixing proposed in the following publication (Chicago style citation):

Khajehrayeni, Farshid, and Hassan Ghassemian. "Hyperspectral unmixing using deep convolutional autoencoders in a supervised scenario." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13 (2020): 567-576.

Parameters

- **n_classes** (int) – Number of classes.
- **input_size** (int) – Number of input spectral bands.
- **kwargs** – Additional arguments.

Return type Sequential

Returns Model proposed in the publication listed above.

ml_intuition.models.**unmixing_pixel_based_cnn**(*n_classes*, *input_size*, ***kwargs*)

Model for pixel-based supervised hyperspectral unmixing proposed in the following publication (Chicago style citation):

Zhang, Xiangrong, Yujia Sun, Jingyan Zhang, Peng Wu, and Licheng Jiao. "Hyperspectral unmixing via deep convolutional neural networks." IEEE Geoscience and Remote Sensing Letters 15, no. 11 (2018): 1755-1759.

Parameters

- **n_classes** (int) – Number of classes.
- **input_size** (int) – Number of input spectral bands.
- **kwargs** – Additional arguments.

Return type Sequential

Returns Model proposed in the publication listed above.

ml_intuition.models.**unmixing_pixel_based_dcae**(*n_classes*, *input_size*, ***kwargs*)

Model for pixel-based unsupervised hyperspectral unmixing proposed in the following publication (Chicago style citation):

Khajehrayeni, Farshid, and Hassan Ghassemian. "Hyperspectral unmixing using deep convolutional autoencoders in a supervised scenario." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13 (2020): 567-576.

Parameters

- **n_classes** (int) – Number of classes.
- **input_size** (int) – Number of input spectral bands.
- **kwargs** – Additional arguments.

Return type Sequential

Returns Model proposed in the publication listed above.

ml_intuition.models.**unmixing_rnn_supervised**(*n_classes*, ***kwargs*)

Model for the unmixing which utilizes a recurrent neural network (RNN) for extracting valuable information from the spectral domain in an supervised manner.

Parameters

- **n_classes** (int) – Number of classes.
- **kwargs** – Additional arguments.

Return type Sequential

Returns RNN model instance.

2.1.4 Module contents

2.2 scripts package

2.2.1 Subpackages

scripts.ensemble package

Submodules

scripts.ensemble.ensemble_runner module

Run N experiments utilizing the ensemble model.

scripts.ensemble.ensemble_runner.**run_experiments**(*, *data_file_paths*, *train_size*, *val_size=0.1*, *stratified=True*, *background_label=0*, *channels_idx=0*, *neighborhood_sizes*, *save_data=False*, *n_runs*, *dest_path*, *model_paths*, *model_experiment_names*, *n_classes*, *voting='hard'*, *batch_size=1024*, *post_noise_sets*, *post_noise*, *noise_params=None*, *use_mlflow=False*, *experiment_name=None*, *run_name=None*)

Function for running experiments given a set of hyper parameters.

Parameters

- **data_file_paths** (*list[str]*) – Path to the data file. Supported types are: .npy

- **train_size** (*Union[int, float]*) – If float, should be between 0.0 and 1.0. If stratified = True, it represents percentage of each class to be extracted, If float and stratified = False, it represents percentage of the whole dataset to be extracted with samples drawn randomly, regardless of their class. If int and stratified = True, it represents number of samples to be drawn from each class. If int and stratified = False, it represents overall number of samples to be drawn regardless of their class, randomly. Defaults to 0.8
- **val_size** (`float`) – Should be between 0.0 and 1.0. Represents the percentage of each class from the training set to be extracted as a validation set, defaults to 0.1
- **stratified** (`bool`) – Indicated whether the extracted training set should be stratified, defaults to True
- **background_label** (`int`) – Label indicating the background in GT file
- **channels_idx** (`int`) – Index specifying the channels position in the provided data
- **neighborhood_sizes** (*list[str]*) – Sizes of the neighbourhood of each provided model
- **save_data** (`bool`) – Whether to save the prepared dataset
- **n_runs** (`int`) – Number of total experiment runs.
- **dest_path** (`str`) – Path to where all experiment runs will be saved as subfolders in this directory.
- **model_paths** (*list[str]*) – Paths to all models to be used in ensemble
- **model_experiment_names** (*list[str]*) – Names of MLFlow experiments
- **n_classes** (`int`) – Number of classes.
- **voting** (`str`) – Method of ensemble voting. If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities.
- **batch_size** (`int`) – Size of the batch for the inference
- **post_noise_sets** (*list[str]*) – The list of sets to which the noise will be injected. One element can either be "train", "val" or "test".
- **post_noise** (*list[str]*) – The list of names of noise injection methods after the normalization transformations.
- **noise_params** (`Optional[str]`) – JSON containing the parameter setting of injection methods. Exemplary value for this parameter: "{"mean": 0, "std": 1, "pa": 0.1}". This JSON should include all parameters for noise injection functions that are specified in pre_noise and post_noise arguments. For the accurate description of each parameter, please refer to the ml_intuition/data/noise.py module.
- **use_mlflow** (`bool`) – Whether to log metrics and artifacts to mlflow.
- **experiment_name** (`Optional[str]`) – Name of the experiment. Used only if use_mlflow = True
- **run_name** (`Optional[str]`) – Name of the run. Used only if use_mlflow = True.

scripts.ensemble.evaluate_with_ensemble module

Evaluate a dataset using an ensemble.

scripts.ensemble.evaluate_with_ensemble.**evaluate**(*, *y_pred*, *data*, *dest_path*, *model_path*, *voting='hard'*)

Function for evaluating the trained model.

Parameters

- **y_pred** – Predictions of all the models to be provided into an ensemble to vote
- **data** – Either path to the input data or the data dict.
- **dest_path** (str) – Directory in which to store the calculated metrics
- **model_path** (str) – Path to the model.
- **voting** (str) – Method of ensemble voting. If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities.

Module contents

scripts.unmixing package

Submodules

scripts.unmixing.evaluate_unmixing module

Perform the inference of the unmixing models on the test dataset.

scripts.unmixing.evaluate_unmixing.**evaluate**(*data*, *model_path*, *dest_path*, *neighborhood_size*, *batch_size*, *endmembers_path*)

Function for evaluating the trained model for the unmixing problem.

Parameters

- **model_path** (str) – Path to the model.
- **data** – The data dictionary containing the subset for testing.
- **dest_path** (str) – Directory in which to store the calculated metrics.
- **neighborhood_size** (int) – Size of the spatial patch.
- **batch_size** (int) – Size of the batch for inference.
- **endmembers_path** (str) – Path to the endmembers matrix file, containing the average reflectances for each endmember, i.e., the pure spectra.

scripts.unmixing.train_unmixing module

Perform the training of the models for the unmixing problem.

scripts.unmixing.train_unmixing.**train**(*data*, *model_name*, *dest_path*, *sample_size*, *n_classes*, *neighborhood_size*, *lr*, *batch_size*, *epochs*, *verbose*, *shuffle*, *patience*, *endmembers_path*, *seed*)

Function for running experiments on various unmixing models, given a set of hyper parameters.

Parameters

- **data** (Dict[str, ndarray]) – The data dictionary containing the subsets for training and validation. First dimension of the dataset should be the number of samples.
- **model_name** (str) – Name of the model, it serves as a key in the dictionary holding all functions returning models.
- **dest_path** (str) – Path to where all experiment runs will be saved as subdirectories in this given directory.
- **sample_size** (int) – Size of the input sample.
- **n_classes** (int) – Number of classes.
- **neighborhood_size** (int) – Size of the spatial patch.
- **lr** (float) – Learning rate for the model, i.e., regulates the size of the step in the gradient descent process.
- **batch_size** (int) – Size of the batch used in training phase, it is the size of samples per gradient step.
- **epochs** (int) – Number of epochs for model to train.
- **verbose** (int) – Verbosity mode used in training, (0, 1 or 2).
- **shuffle** (bool) – Boolean indicating whether to shuffle dataset.
- **patience** (int) – Number of epochs without improvement in order to stop the training phase.
- **endmembers_path** (str) – Path to the endmembers matrix file, containing the average reflectances for each endmember, i.e., the pure spectra.
- **seed** (int) – Seed for training reproducibility.

scripts.unmixing.unmixing_experiments_runner module

Run experiments given set of hyperparameters for the unmixing problem.

scripts.unmixing.unmixing_experiments_runner.**run_experiments**(*, *data_file_path*, *ground_truth_path=None*, *train_size*, *val_size=0.1*, *sub_test_size=None*, *channels_idx=-1*, *neighborhood_size=None*, *n_runs=1*, *model_name*, *save_data=0*, *dest_path=None*, *sample_size*, *n_classes*, *lr=None*, *batch_size=256*, *epochs=100*, *verbose=2*, *shuffle=True*, *patience=15*, *use_mlflow=False*, *endmembers_path=None*, *experiment_name=None*, *run_name=None*)

Function for running experiments on unmixing given a set of hyperparameters.

Parameters

- **data_file_path** (str) – Path to the data file. Supported types are: .npy.
- **ground_truth_path** (Optional[str]) – Path to the ground-truth data file.
- **train_size** (*Union[int, float]*) – If float, should be between 0.0 and 1.0. If int, specifies the number of samples in the training set. Defaults to 0.8
- **val_size** (float) – Should be between 0.0 and 1.0. Represents the percentage of samples from the training set to be extracted as a validation set. Defaults to 0.1.
- **sub_test_size** (Optional[int]) – Number of pixels to subsample the test set instead of performing the inference on all samples that are not in the training set.
- **channels_idx** (int) – Index specifying the channels position in the provided data.
- **neighborhood_size** (Optional[int]) – Size of the spatial patch.
- **save_data** (bool) – Boolean indicating whether to save the prepared dataset.
- **n_runs** (int) – Number of total experiment runs.
- **model_name** (str) – Name of the model, it serves as a key in the dictionary holding all functions returning models.
- **dest_path** (Optional[str]) – Path to where all experiment runs will be saved as subdirectories in this directory.
- **sample_size** (int) – Size of the input sample.
- **n_classes** (int) – Number of classes.

- **lr** (Optional[float]) – Learning rate for the model, i.e., regulates the size of the step in the gradient descent process.
- **batch_size** (int) – Size of the batch used in training phase, it is the size of samples per gradient step.
- **epochs** (int) – Number of epochs for model to train.
- **verbose** (int) – Verbosity mode used in training, (0, 1 or 2).
- **shuffle** (bool) – Boolean indicating whether to shuffle dataset.
- **patience** (int) – Number of epochs without improvement in order to stop the training phase.
- **use_mlflow** (bool) – Whether to log metrics and artifacts to mlflow.
- **endmembers_path** (Optional[str]) – Path to the endmembers matrix file, containing the average reflectances for each endmember, i.e., the pure spectra.
- **experiment_name** (Optional[str]) – Name of the experiment. Used only if use_mlflow = True
- **run_name** (Optional[str]) – Name of the run. Used only if use_mlflow = True.

Module contents

scripts.quantization package

Submodules

scripts.quantization.freeze_model module

Freeze the graph and store it in .pb (Protocol Buffers) format

scripts.quantization.freeze_model.**main**(*, *model_path*, *output_dir*)

Freeze the graph and store it in .pb (Protocol Buffers) format

Parameters

- **model_path** (str) – Path to the model to be saved
- **output_dir** (str) – Directory in which the .pb graph and nodes json will be stored

scripts.quantization.quantize_runner module

Freeze model, quantize it and evaluate N times.

scripts.quantization.quantize_runner.**run_experiments**(*, *input_dir*, *n_runs*, *dest_path*, *data_file_path=None*, *ground_truth_path=None*, *dataset_path=None*, *background_label=0*, *channels_idx=2*, *channels_count=103*, *train_size*, *batch_size=64*, *stratified=True*, *gpu=0*)

Freeze model, quantize it and evaluate N times.

Parameters

- **input_dir** (str) – Directory with saved data and models, each in separate *experiment_n* folder.
- **n_runs** (int) – Number of total experiment runs.
- **dest_path** (str) – Path to where all experiment runs will be saved as sub folders in this directory.
- **data_file_path** (Optional[str]) – Path to the data file. Supported types are: .npy and .md5. This is optional, if the data is not already saved in the input_dir.
- **ground_truth_path** (Optional[str]) – Path to the data file.
- **dataset_path** (Optional[str]) – Path to the already extracted .h5 dataset
- **background_label** (int) – Label indicating the background in GT file
- **channels_idx** (int) – Index specifying the channels position in the provided data
- **channels_count** (int) – Number of channels (bands) in the image.
- **train_size** (*Union[int, float]*) – If float, should be between 0.0 and 1.0. If stratified = True, it represents percentage of each class to be extracted, If float and stratified = False, it represents percentage of the whole dataset to be extracted with samples drawn randomly, regardless of their class. If int and stratified = True, it represents number of samples to be drawn from each class. If int and stratified = False, it represents overall number of samples to be drawn regardless of their class, randomly. Defaults to 0.8
- **stratified** (bool) – Indicated whether the extracted training set should be stratified, defaults to True
- **batch_size** (int) – Batch size
- **gpu** (bool) – Whether to run quantization on gpu.

Module contents

2.2.2 Submodules

2.2.3 scripts.artifacts_reporter module

Collect the artifacts report based on the experiment runs placed in the "experiments_path" directory.

scripts.artifacts_reporter.**collect_artifacts_report**(*, *experiments_path*, *dest_path*, *filename=None*, *use_mlflow=False*)

Collect the artifacts report based on the experiment runs placed in the "experiments_path" directory.

Parameters

- **experiments_path** (str) – Path to the directory containing the experiment subdirectories.
- **dest_path** (str) – Path to the destination directory or full path to the report .csv file.
- **filename** (Optional[str]) – Name of the file holding metrics. Defaults to 'inference_metrics.csv'.
- **use_mlflow** (bool) – Whether to log metrics and artifacts to mlflow.

2.2.4 scripts.evaluate_model module

Perform the inference of the model on the provided dataset.

scripts.evaluate_model.**evaluate**(*, *data*, *model_path*, *dest_path*, *n_classes*, *batch_size=1024*, *use_ensemble=False*, *ensemble_copies=1*, *voting='hard'*, *noise*, *noise_sets*, *noise_params=None*, *seed=0*)

Function for evaluating the trained model.

Parameters

- **model_path** (str) – Path to the model.
- **data** – Either path to the input data or the data dict.
- **dest_path** (str) – Directory in which to store the calculated metrics.
- **n_classes** (int) – Number of classes.
- **batch_size** (int) – Size of the batch for inference.
- **use_ensemble** (bool) – Use ensemble for prediction.
- **ensemble_copies** (int) – Number of model copies for the ensemble.
- **voting** (str) – Method of ensemble voting. If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities.
- **noise** (*list[str]*) – List containing names of used noise injection methods that are performed after the normalization transformations.
- **noise_sets** (*list[str]*) – List of sets that are affected by the noise injection. For this module single element can be "test".
- **noise_params** (Optional[str]) – JSON containing the parameters setting of noise injection methods. Exemplary value for this parameter: "{"mean": 0, "std": 1, "pa": 0.1}". This JSON should include all parameters for noise injection functions that are specified in the noise argument. For the accurate description of each parameter, please refer to the ml_intuition/data/noise.py module.
- **seed** (int) – Seed for RNG.

2.2.5 scripts.experiments_runner module

Run training and evaluation of a model N times, given set of hyperparameters. Has the option to inject noise into train, val and test set.

scripts.experiments_runner.**run_experiments**(*, *data_file_path*, *ground_truth_path=None*, *train_size*, *val_size=0.1*, *stratified=True*, *background_label=0*, *channels_idx=0*, *neighborhood_size=None*, *n_runs*, *model_name*, *kernel_size=5*, *n_kernels=200*, *save_data=0*, *n_layers=1*, *dest_path=None*, *sample_size*, *n_classes*, *lr=0.001*, *batch_size=128*, *epochs=200*, *verbose=2*, *shuffle=True*, *patience=15*, *pre_noise*, *pre_noise_sets*, *post_noise*, *post_noise_sets*, *noise_params=None*, *use_mlflow=False*, *experiment_name=None*, *run_name=None*)

Function for running experiments given a set of hyper parameters.

Parameters

- **data_file_path** (str) – Path to the data file. Supported types are: .npy.
- **ground_truth_path** (Optional[str]) – Path to the ground-truth data file.
- **train_size** (*Union[int, float]*) – If float, should be between 0.0 and 1.0. If stratified = True, it represents percentage of each class to be extracted. If float and stratified = False, it represents percentage of the whole dataset to be extracted with samples drawn randomly, regardless of their class. If int and stratified = True, it represents number of samples to be drawn from each class. If int and stratified = False, it represents overall number of samples to be drawn regardless of their class, randomly. Defaults to 0.8.
- **val_size** (float) – Should be between 0.0 and 1.0. Represents the percentage of each class from the training set to be extracted as a validation set. Defaults to 0.1.
- **stratified** (bool) – Indicated whether the extracted training set should be stratified. Defaults to True.
- **background_label** (int) – Label indicating the background in GT file.
- **channels_idx** (int) – Index specifying the channels position in the provided data.
- **neighborhood_size** (Optional[int]) – Size of the spatial patch.
- **save_data** (bool) – Whether to save the prepared dataset.
- **n_runs** (int) – Number of total experiment runs.
- **model_name** (str) – Name of the model, it serves as a key in the dictionary holding all functions returning models.
- **kernel_size** (int) – Size of ech kernel in each layer.
- **n_kernels** (int) – Number of kernels in each layer.
- **n_layers** (int) – Number of layers in the model.
- **dest_path** (Optional[str]) – Path to where all experiment runs will be saved as subfolders in this directory.
- **sample_size** (int) – Size of the input sample.
- **n_classes** (int) – Number of classes.
- **lr** (float) – Learning rate for the model, i.e., regulates the size of the step in the gradient descent process.
- **batch_size** (int) – Size of the batch used in training phase, it is the size of samples per gradient step.
- **epochs** (int) – Number of epochs for model to train.
- **verbose** (int) – Verbosity mode used in training, (0, 1 or 2).
- **shuffle** (bool) – Boolean indicating whether to shuffle dataset dataset_key each epoch.
- **patience** (int) – Number of epochs without improvement in order to stop the training phase.
- **pre_noise** (*list[str]*) – The list of names of noise injection methods before the normalization transformations. Exemplary names are “gaussian” or “impulsive”.
- **pre_noise_sets** (*list[str]*) – The list of sets to which the noise will be injected. One element can either be “train”, “val” or “test”.

- **post_noise** (*list[str]*) – The list of names of noise injection methods after the normalization transformations.
- **post_noise_sets** (*list[str]*) – The list of sets to which the noise will be injected.
- **noise_params** (Optional[str]) – JSON containing the parameter setting of injection methods. Exemplary value for this parameter: "{"mean": 0, "std": 1, "pa": 0.1}". This JSON should include all parameters for noise injection functions that are specified in pre_noise and post_noise arguments. For the accurate description of each parameter, please refer to the ml_intuition/data/noise.py module.
- **use_mlflow** (bool) – Whether to log metrics and artifacts to mlflow.
- **experiment_name** (Optional[str]) – Name of the experiment. Used only if use_mlflow = True
- **run_name** (Optional[str]) – Name of the run. Used only if use_mlflow = True.

2.2.6 scripts.inference_runner module

Run inference N times on the provided model given set of hyperparameters. Has the option to inject noise into the test set.

scripts.inference_runner.**run_experiments**(*, *data_file_path=None*, *ground_truth_path=None*, *dataset_path=None*, *train_size*, *val_size=0.1*, *stratified=True*, *background_label=0*, *channels_idx=0*, *neighborhood_size=None*, *save_data=False*, *n_runs*, *dest_path*, *models_path*, *model_name='model_2d'*, *n_classes*, *use_ensemble=False*, *ensemble_copies=None*, *voting='hard'*, *batch_size=1024*, *post_noise_sets*, *post_noise*, *noise_params=None*, *use_mlflow=False*, *experiment_name=None*, *run_name=None*)

Run inference on the provided model given set of hyperparameters.

Parameters

- **data_file_path** (Optional[str]) – Path to the data file. Supported types are: .npy
- **ground_truth_path** (Optional[str]) – Path to the ground-truth data file.
- **dataset_path** (Optional[str]) – Path to the already extracted .h5 dataset
- **train_size** (*Union[int, float]*) – If float, should be between 0.0 and 1.0. If stratified = True, it represents percentage of each class to be extracted, If float and stratified = False, it represents percentage of the whole dataset to be extracted with samples drawn randomly, regardless of their class. If int and stratified = True, it represents number of samples to be drawn from each class. If int and stratified = False, it represents overall number of samples to be drawn regardless of their class, randomly. Defaults to 0.8
- **val_size** (float) – Should be between 0.0 and 1.0. Represents the percentage of each class from the training set to be extracted as a validation set. Defaults to 0.1.
- **stratified** (bool) – Indicated whether the extracted training set should be stratified. Defaults to True.
- **background_label** (int) – Label indicating the background in GT file.
- **channels_idx** (int) – Index specifying the channels position in the provided data.

- **neighborhood_size** (Optional[int]) – Size of the neighborhood of the pixel. Only used for 2D and 3D models.
- **save_data** (bool) – Whether to save the prepared dataset.
- **n_runs** (int) – Number of total experiment runs.
- **dest_path** (str) – Path to where all experiment runs will be saved as subfolders in this directory.
- **models_path** (str) – Name of the model, it serves as a key in the dictionary holding all functions returning models.
- **model_name** (str) – The name of model for the inference.
- **n_classes** (int) – Number of classes.
- **use_ensemble** (bool) – Use ensemble for prediction.
- **ensemble_copies** (Optional[int]) – Number of model copies for the ensemble.
- **voting** (str) – Method of ensemble voting. If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities.
- **batch_size** (int) – Size of the batch for the inference
- **post_noise_sets** (*list[str]*) – The list of sets to which the noise will be injected. One element can either be "train", "val" or "test".
- **post_noise** (*list[str]*) – The list of names of noise injection methods after the normalization transformations.
- **noise_params** (Optional[str]) – JSON containing the parameter setting of injection methods. Exemplary value for this parameter: "{"mean": 0, "std": 1, "pa": 0.1}". This JSON should include all parameters for noise injection functions that are specified in pre_noise and post_noise arguments. For the accurate description of each parameter, please refer to the ml_intuition/data/noise.py module.
- **use_mlflow** (bool) – Whether to log metrics and artifacts to mlflow.
- **experiment_name** (Optional[str]) – Name of the experiment. Used only if use_mlflow = True.
- **run_name** (Optional[str]) – Name of the run. Used only if use_mlflow = True.

2.2.7 scripts.predict_with_model module

Perform the inference of the model on the provided dataset.

scripts.predict_with_model.**predict**(*, *data*, *model_path*, *n_classes*, *batch_size=1024*, *noise*, *noise_sets*, *noise_params=None*)

Function for evaluating the trained model.

Parameters

- **data** – Either path to the input data or the data dict.
- **model_path** (str) – Path to the model.
- **n_classes** (int) – Number of classes.
- **batch_size** (int) – Size of the batch for inference.

- **noise** (*list[str]*) – List containing names of used noise injection methods that are performed after the normalization transformations.
- **noise_sets** (*list[str]*) – List of sets that are affected by the noise injection. For this module single element can be "test".
- **noise_params** (Optional[str]) – JSON containing the parameters setting of noise injection methods. Exemplary value for this parameter: "{"mean": 0, "std": 1, "pa": 0.1}". This JSON should include all parameters for noise injection functions that are specified in the noise argument. For the accurate description of each parameter, please refer to the ml_intuition/data/noise.py module.

2.2.8 scripts.prepare_data module

Load the data, reformat it to have [SAMPLES, ...] dimensions, split it into train, test and val sets and save them in .h5 file with 'train', 'val' and 'test' groups, each having 'data' and 'labels' keys.

scripts.prepare_data.**main**(*, *data_file_path*, *ground_truth_path*, *output_path=None*, *train_size*, *val_size=0.1*, *stratified=True*, *background_label=0*, *neighborhood_size=None*, *channels_idx=0*, *save_data=False*, *seed=0*, *use_unmixing=False*)

Parameters

- **data_file_path** (str) – Path to the data file. Supported types are: .npy.
- **ground_truth_path** (str) – Path to the data file.
- **output_path** (Optional[str]) – Path under in which the output .h5 file will be stored. Used only if the parameter save_data is set to True.
- **train_size** (*float or int*) – If float, should be between 0.0 and 1.0. If stratified = True, it represents percentage of each class to be extracted, If float and stratified = False, it represents percentage of the whole dataset to be extracted with samples drawn randomly, regardless of their class. If int and stratified = True, it represents number of samples to be drawn from each class. If int and stratified = False, it represents overall number of samples to be drawn regardless of their class, randomly. Defaults to 0.8
- **val_size** (float) – Should be between 0.0 and 1.0. Represents the percentage of each class from the training set to be extracted as a validation set. Defaults to 0.1.
- **stratified** (bool) – Indicated whether the extracted training set should be stratified. Defaults to True.
- **background_label** (int) – Label indicating the background in GT file.
- **neighborhood_size** (Optional[int]) – Neighborhood size of the pixel to extract along with its spectral bands. Use only if you are training 2D or 3D convolutional model.
- **channels_idx** (int) – Index specifying the channels position in the provided data.
- **save_data** (bool) – Whether to save data as .md5 or to return it as a dict.
- **seed** (int) – Seed used for data shuffling.
- **use_unmixing** (bool) – Boolean indicating whether to perform experiments on the unmixing datasets, where classes in each pixel are present as fractions.

Raises **TypeError** – When provided data or labels file is not supported.

2.2.9 scripts.train_model module

Perform the training of the model. Has the option to inject noise into train and val set.

scripts.train_model.**train**(*, *data*, *model_name*, *dest_path*, *sample_size*, *n_classes*, *kernel_size=3*, *n_kernels=16*, *n_layers=1*, *lr=0.005*, *batch_size=150*, *epochs=10*, *verbose=2*, *shuffle=True*, *patience=3*, *seed=0*, *noise*, *noise_sets*, *noise_params=None*)

Function for training tensorflow models given a dataset.

Parameters

- **model_name** (str) – Name of the model, it serves as a key in the dictionary holding all functions returning models.
- **kernel_size** (int) – Size of ech kernel in each layer.
- **n_kernels** (int) – Number of kernels in each layer.
- **n_layers** (int) – Number of layers in the model.
- **dest_path** (str) – Path to where to save the model under the name "model_name".
- **sample_size** (int) – Size of the input sample.
- **n_classes** (int) – Number of classes.
- **lr** (float) – Learning rate for the model, i.e., regulates the size of the step in the gradient descent process.
- **data** – Either path to the input data or the data dict itself. First dimension of the dataset should be the number of samples.
- **batch_size** (int) – Size of the batch used in training phase, it is the size of samples per gradient step.
- **epochs** (int) – Number of epochs for model to train.
- **verbose** (int) – Verbosity mode used in training, (0, 1 or 2).
- **shuffle** (bool) – Boolean indicating whether to shuffle dataset dataset_key each epoch.
- **patience** (int) – Number of epochs without improvement in order to stop the training phase.
- **seed** (int) – Seed for training reproducibility.
- **noise** (*list[str]*) – List containing names of used noise injection methods that are performed after the normalization transformations.
- **noise_sets** (*list[str]*) – List of sets that are affected by the noise injection methods. For this module single element can be either "train" or "val".
- **noise_params** (Optional[str]) – JSON containing the parameters setting of injection methods. Exemplary value for this parameter: "{"mean": 0, "std": 1, "pa": 0.1}". This JSON should include all parameters for noise injection functions that are specified in the noise argument. For the accurate description of each parameter, please refer to the ml_intuition/data/noise.py module.

2.2.10 Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

s

INDEX

Symbols

A

B

C

E

G

I

L

M

N

O

P

R

S

T

U

V