nondefaced-detector

Release 0.1.4.dev9+g260f0ca

Developers of nondefaced-detector

Apr 29, 2021

CONTENTS

`nondefaced-detector` is a deep learning framework to detect if a 3D MRI volume has been defaced.

CHAPTER

ONE

CONTENTS

1.1 User guide

Download for offline viewing

Download the user guide and examples.

1.1.1 Introduction: What is nondefaced-detector?

Contents

What is nondefaced-detector and why should you use it?

Why use nondefaced-detector?

The Nondefaced-detector is an automated framework to detect nondefaced T1-weighted images of the human brain.

Installing nondefaced-detector

nondefaced-detector is easy to install. To install the most recent release, use pip:

```
>>> pip install nondefaced-detector
```

If you want the "bleeding-edge" version of nondefaced-detector, you can install directly from the GitHub repository:

```
>>> pip install git+https://github.com/poldracklab/nondefaced-detector.git
```

Finding help

Mailing lists and forums

- Don't hesitate to ask questions about nondefaced-detector on Nondefaced-Detector Issues.
- If you notice a bug in the nondefaced-detector code, please open an issue in the nondefaced-detector repository.

1.2 API Reference

1.2.1 `nondefaced_detector.models`: Model functions

`nondefaced_detector.models`	
nondefaced_detector.models.model.ConvBNrelu(x)	A layer block of one convolutional, one batch normalization, and one non-linear activation sequence.
nondefaced_detector.models.model.TruncatedSubmodel(...)	The TruncatedSubmodel trained in Step 1 of the model.
nondefaced_detector.models.model.ClassifierHead(...)	The final block of the model
nondefaced_detector.models.model.Submodel(...)	3 identical submodel blocks are used to train on spatial information from all three axes (axial, coronal, sagittal) separately.
nondefaced_detector.models.model.CombinedClassifier([...])	The final block of the model that combines features and outputs a real-valued probability using the sigmoid function.

`nondefaced_detector.models.model.`ConvBNrelu

ConvBNrelu (*x*, *filters=32*, *kernel=3*, *strides=1*, *padding='same'*)

A layer block of one convolutional, one batch normalization, and one non-linear activation sequence.

Parameters

- **x** (`tf.Tensor` of rank 4+) – The input keras tensor object to instantiate a keras model
- **filters** (*int, optional, default=32*) – The dimensionality of the output space (i.e. the number of output filters in the convolution).
- **kernel** (*int, optional, default=32*) – An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
- **strides** (*int*) – Specifying the strides of the convolution along the height and width. Can be a single integer to specify the same value for all spatial dimensions.
- **padding** (*one of "valid" or "same" (case-insensitive)*) – "valid" means no padding. "same" results in padding evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.

Returns A tensor of rank 4+.

Return type `tf.Tensor`

nondefaced_detector.models.model.TruncatedSubmodel

TruncatedSubmodel(*input_layer*)
The TruncatedSubmodel trained in Step 1 of the model.

Parameters **input_layer** (*tf.keras.Input*) – The input keras tensor object to instantiate a keras model

Returns A flattened truncated network created from 3 sequential ConvBNRelu layer blocks joined by a MaxPooling layer.

Return type tf.Tensor

nondefaced_detector.models.model.ClassifierHead

ClassifierHead(*layer*, *dropout*)
The final block of the model

Parameters

- **layer** (*N-D tensor with shape: (batch_size, .., input_dim)*) – The flattened out feature layer output from the Submodels
- **dropout** (*float*) – Float between 0 and 1. Fraction of the input units to drop.

Returns N-D tensor with shape: (batch_size, . . . , units)

Return type tf.Tensor

nondefaced_detector.models.model.Submodel

Submodel(*root_path*, *input_shape=(32, 32)*, *dropout=0.4*, *name='axial'*, *weights='axial'*, *include_top=True*, *trainable=True*)
3 identical submodel blocks are used to train on spatial information from all three axes (axial, coronal, sagittal) separately.

Parameters

- **root_path** (*str, Path*) – Root directory for storing the weights.
- **input_shape** (*tuple of ints, default=(32, 32)*) – The shape of the input image.
- **dropout** (*float, optional, default=0.4*) – Float between 0 and 1. Fraction of the input units to drop.
- **name** (*str*) – Name of the submodel.
- **weights** (*str*) – Name of the folder to store the weights for the submodel.
- **include_top** (*bool, default=True*) – If True, the the model includes the ClassifierHead block at the end.
- **trainable** (*bool, default=True*) – If True, the model is set to be trainable else the model layers are frozen.

Returns Returns a *tf.keras.Model* object with features.

Return type *tf.keras.Model*

nondefaced_detector.models.model.CombinedClassifier

CombinedClassifier(*input_shape=(32, 32)*, *dropout=0.4*, *wts_root=None*, *trainable=False*, *shared=False*)

The final block of the model that combines features and outputs a real-valued probability using the sigmoid function.

Parameters

- **input_shape** (*tuple of ints, default=(32, 32)*) – The shape of the input image.
- **dropout** (*float, optional, default=0.4*) – Float between 0 and 1. Fraction of the input units to drop.
- **trainable** (*bool, default=True*) – If True, the model is set to be trainable else the model layers are frozen.
- **shared** (*bool, default=False*) –

1.2.2 nondefaced_detector.dataloaders: Dataset functions

nondefaced_detector.dataloaders	
nondefaced_detector.dataloaders.dataset.get_dataset(...)	Returns tf.data.Dataset after preprocessing from tfrecords for training and validation

nondefaced_detector.dataloaders.dataset.get_dataset

get_dataset(*file_pattern*, *n_classes*, *batch_size*, *volume_shape*, *plane*, *n_slices=24*, *block_shape=None*, *n_epochs=None*, *mapping=None*, *shuffle_buffer_size=None*, *num_parallel_calls=- 1*, *mode='train'*)

Returns tf.data.Dataset after preprocessing from tfrecords for training and validation

Parameters

- **file_pattern** –
- **n_classes** –

1.2.3 nondefaced_detector.preprocess: Preprocess input volumes

Script to preprocess volumes

nondefaced_detector.preprocess	Script to preprocess volumes
nondefaced_detector.preprocess.preprocess(...)	Preprocess input volumes before prediction.
nondefaced_detector.preprocess.preprocess_parallel(...)	Preprocess multiple input volumes before prediction in parallel.

nondefaced_detector.preprocess.preprocess

preprocess(*vol_path*, *conform_volume_to=(128, 128, 128)*, *conform_zooms=(2.0, 2.0, 2.0)*, *save_path=None*, *with_label=False*)

Preprocess input volumes before prediction.

Parameters

- **vol_path** (*str - Path or tuple of length 2 (str - Path, int)*) – The path to the input volume. If the *with_label* flag is True, the vol_path is required to be a tuple of size 2 - (vol_path, label)
- **conform_volume_to** (*tuple of length 3, optional, default=(128 128, 128)*) – The shape the volume will be conformed to. Note: The pretrained model was trained using the conform size of (128, 128, 128) and assumes the volume shape as such.
- **save_path** (*str - Path, optional*) – The path where the output volume is saved. If none is provided, the output volume will be saved under *vol_path/preprocessed*
- **with_label** (*bool, optional*) – If True, the input vol_path is required to be a tuple of 2 (vol_path, label)

Returns Path to the where the preprocessed volume is stored. (Path, label) if with_label is True.

Return type str - Path

nondefaced_detector.preprocess.preprocess_parallel

preprocess_parallel(*volume_filepaths*, *num_parallel_calls=- 1*, *conform_volume_to=(128, 128, 128)*, *conform_zooms=(2.0, 2.0, 2.0)*, *save_path=None*, *with_label=True*)

Preprocess multiple input volumes before prediction in parallel.

Parameters

- **volume_filepaths** (*list of str - Path or list of tuple of length 2 [(str - Path, int), ..]*) – A list of paths to the input volumes. If the *with_label* flag is True, the volume_filepaths is required to be a list of tuples of size 2 - (volume_filepath, label)
- **num_parallel_calls** (*int*) – Number of parallel calls to make for preprocessing.
- **conform_volume_to** (*tuple of length 3, optional, default=(128 128, 128)*) – The shape the volume will be conformed to. Note: The pretrained model was trained using the conform size of (128, 128, 128) and assumes the volume shape as such.
- **conform_zooms** (*tuple of size 3, optional, default=(2.0, 2.0, 2.0)*) – The zoom of the resampled output.
- **save_path** (*str - Path, optional*) – The path where the output volume is saved. If none is provided, the output volume will be saved under *volume_filepath/preprocessed*
- **with_label** (*bool, optional*) – If True, each volume_filepath is required to be a tuple of 2 (volume_filepath, label)

Returns List of str paths to the where each preprocessed volume is stored. [(Path, label), ...] if with_label is True.

Return type list of str

1.2.4 nondefaced_detector.preprocessing: Helper functions for the preprocess module

nondefaced_detector.preprocessing	
nondefaced_detector.preprocessing.conform.conform_data(in_file)	Conform the input dataset to the canonical orientation.
nondefaced_detector.preprocessing.normalization.clip(x)	
nondefaced_detector.preprocessing.normalization.standardize(x)	
nondefaced_detector.preprocessing.normalization.normalize(x)	

nondefaced_detector.preprocessing.conform.conform_data

conform_data(*in_file*, *out_file=None*, *out_size=(256, 256, 256)*, *out_zooms=(1.0, 1.0, 1.0)*, *order=3*)

Conform the input dataset to the canonical orientation.

Parameters

- **in_file** (*str - Path*) – Path to the input MRI volume to conform.
- **out_file** (*str - Path, default=None*) – Path to save the conformed volume. By default the volume is saved as /tmp/conformed.nii.gz
- **out_size** (*tuple of size 3, optional, default=(256, 256, 256)*) – The shape to conform the 3D volume to.
- **out_zooms** (*tuple of size 3, optional, default=(1.0, 1.0, 1.0)*) – Factors to normalize voxel size to.
- **order** (*int, optional, default=3*) – Order of the spline interpolation. The order has to be in the range 0-5.

Returns The path to where the conformed volume is saved.

Return type str - Path

nondefaced_detector.preprocessing.normalization.clip

clip(*x*, *q=90*)

nondefaced_detector.preprocessing.normalization.standardize

standardize(*x*)

nondefaced_detector.preprocessing.normalization.normalize

normalize(*x*)

1.2.5 nondefaced_detector.training: Training

nondefaced_detector.training	
nondefaced_detector.training.training.train(...)	Train a model.

nondefaced_detector.training.training.train

train(*csv_path*, *model_save_path*, *tfrecords_path*, *volume_shape=(128, 128, 128)*, *image_size=(128, 128)*, *dropout=0.2*, *batch_size=16*, *n_classes=2*, *n_epochs=15*, *mode='CV'*)

Train a model.

Parameters

- **csv_path** (*str - Path*) – Path to the csv file containing training volume paths, labels (X, Y).
- **model_save_path** (*str - Path*) – Path to where the save model and model weights.
- **tfrecords_path** (*str - Path*) – Path to preprocessed training tfrecords.
- **volume_shape** (*tuple of size 3, optional, default=(128, 128, 128)*) – The shape of the preprocessed volumes.
- **image_size** (*tuple of size 2, optional, default=(128, 128)*) – The shape of a 2D slice along each volume axis.
- **dropout** (*float, optional, default=0.4*) – Float between 0 and 1. Fraction of the input units to drop.
- **batch_size** (*int, optional, default=16*) – No. of training examples utilized in each iteration.
- **n_classes** (*int, optional, default=2*) – No. of unique classes to train the model on. Default assumption is a binary classifier.
- **n_epochs** (*int, optional, default=15*) – No. of complete passes through the training dataset.
- **mode** (*str, optional, default=15*) – One of "CV" or "full". Indicates the type of training to perform.

Returns A History object that records several metrics such as training/validation loss/metrics at successive epochs.

Return type *tf.keras.callbacks.History*

1.2.6 nondefaced_detector.prediction: Making predictions

Methods to predict using trained models

nondefaced_detector.prediction	Methods to predict using trained models
nondefaced_detector.prediction.predict(...)	Return predictions from a list of input volumes.
nondefaced_detector.prediction._structural_slice(x, ...)	Transpose dataset and get slices from the volume based on the plane.
nondefaced_detector.prediction._get_model(...)	Return *tf.keras.Model* object from a filepath.

nondefaced_detector.prediction.predict

predict(*volumes*, *model_path*, *n_slices=32*)

Return predictions from a list of input volumes.

Parameters

- **volumes** (*list*) – A list of Path like strings to the volumes to make the prediction on.
- **model_path** (*str - Path*) – The path to pretrained model weights.
- **n_slices** (*int, optional, default=32*) – The number of 2D slices of the MRI volume to predict on.

Returns A list of predicted probabilities.

Return type list

nondefaced_detector.prediction._structural_slice

_structural_slice(*x*, *plane*, *n_slices=16*)

Transpose dataset and get slices from the volume based on the plane.

Parameters

- **x** (`tf.Tensor`) – The input MRI volume/dataset to sample slices from.
- **plane** (*one of ["sagittal", "coronal", "axial", "combined"]*) – The axes of the plane to get the slices for. If "combined", the input is sliced for all 3 axes.
- **n_slices** (*int, optional, default=16*) – The number of 2D slices to cut along the input plane. n_slices are randomly sampled from the input volume.

Returns A tensor of shape (n_slices, x.shape) or A dict with keys ['sagittal', 'coronal', 'axial'] each with a value of tensors of shape (n_slices, x.shape)

Return type `tf.Tensor`

nondefaced_detector.prediction._get_model

_get_model(*model_path*)
Return *tf.keras.Model* object from a filepath.

Parameters **model_path** (*str, path to HDF5 or SavedModel file.*) –

Returns

Return type Instance of *tf.keras.Model*.

Raises **ValueError** –

1.2.7 nondefaced_detector.inference: Inference

Standalone inference script for held-out test dataset.

nondefaced_detector.inference	Standalone inference script for held-out test dataset.
nondefaced_detector.inference.inference(...)	Inference function to reproduce original model scores.

nondefaced_detector.inference.inference

inference(*tfrecords_path*, *weights_path*, *wts_root*)
Inference function to reproduce original model scores. This script can be run as a standalone using python inference.py. For more information try: *python inference.py -h*

Parameters

- **tfrecords_path** (*str*) – The path to directory containing preprocessed tfrecords.
- **weights_path** (*str*) – The path to the combined model weights. A copy of the weights can be found here: https://gin.g-node.org/shashankbansal56/nondefaced-detector-reproducibility/src/master/pretrained_weights/combined
- **wts_root** (*str*) – The path to the root directory of all the model weights. A copy of the weights can be found here: https://gin.g-node.org/shashankbansal56/nondefaced-detector-reproducibility/src/master/pretrained_weights

1.2.8 nondefaced_detector.helpers: Helper functions

nondefaced_detector.helpers	
nondefaced_detector.helpers.utils.is_gz_file(...)	
nondefaced_detector.helpers.utils.save_vol(...)	save_path: path to write the volume to tensor_3d: 3D volume which needs to be saved affine: image orientation, translation
nondefaced_detector.helpers.utils.load_vol(...)	load_path: volume path to load :returns: loaded 3D volume affine: affine data specific to the volume :rtype: volume
nondefaced_detector.helpers.utils.imshow(img1)	

nondefaced_detector.helpers.utils.is_gz_file

is_gz_file(*filepath*)

nondefaced_detector.helpers.utils.save_vol

save_vol(*save_path*, *tensor_3d*, *affine*)

save_path: path to write the volume to tensor_3d: 3D volume which needs to be saved affine: image orientation, translation

nondefaced_detector.helpers.utils.load_vol

load_vol(*load_path*)

load_path: volume path to load :returns: loaded 3D volume

affine: affine data specific to the volume

Return type volume

nondefaced_detector.helpers.utils.imshow

imshow(*img1*, *title='myPlot'*)

1.2.9 nondefaced_detector.utils: Utility functions

Utilities for Nondefaced-detector.

nondefaced_detector.utils	Utilities for Nondefaced-detector.
nondefaced_detector.utils.get_datalad([...])	Download a datalad dataset/repo.

nondefaced_detector.utils.get_datalad

get_datalad(*cache_dir='/tmp/nondefaced-detector-reproducibility'*, *datalad_repo='https://gin.g-node.org/shashankbansal56/nondefaced-detector-reproducibility'*, *examples=False*, *test_ixi=False*)

Download a datalad dataset/repo.

The weights can be found at https://gin.g-node.org/shashankbansal56/nondefaced-detector-reproducibility/

Parameters **cache_dir** (*str, directory where to clone datalad repo. Save to a /tmp by default*) –

1.3 Changelog

1.3.1 Version v0.1.3 (April 16, 2021)

- V0.1.3 (#16)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

Symbols

C

G

I

L

M

N

P

S

T