

---

# **iNNvestigate Documentation**

*Release not set*

**Maximilian Alber**

**Jan 23, 2019**



---

## Contents:

---

<b>1</b>	<b>API reference</b>	<b>3</b>
1.1	innvestigate . . . . .	3
1.2	innvestigate.analyzer . . . . .	3
1.3	innvestigate.applications . . . . .	10
1.4	innvestigate.tools . . . . .	11
1.5	innvestigate.utils . . . . .	14
<b>2</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



This is the API documentation for the iNNvestigate library. It contains a references for the most important functions and classes of the library. For a usage guide and more information about the interplay of the componentes please visit the [GitHub repo](#).



## 1.1 innvestigate

`innvestigate.analyzer.create_analyzer(name, model, **kwargs)`

Instantiates the analyzer with the name ‘name’

This convenience function takes an analyzer name creates the respective analyzer.

Alternatively analyzers can be created directly by instantiating the respective classes.

### Parameters

- **name** – Name of the analyzer.
- **model** – The model to analyze, passed to the analyzer’s `__init__`.
- **kwargs** – Additional parameters for the analyzer’s .

**Returns** An instance of the chosen analyzer.

**Raises** **KeyError** – If there is no analyzer with the passed name.

## 1.2 innvestigate.analyzer

### 1.2.1 Interface and base code

**exception** `innvestigate.analyzer.base.NotAnalyzeableModelError`

Indicates that the model cannot be analyzed by an analyzer.

**class** `innvestigate.analyzer.base.AnalyzerBase(model, disable_model_checks=False)`

The basic interface of an iNNvestigate analyzer.

This class defines the basic interface for analyzers:

```
>>> model = create_keras_model()
>>> a = Analyzer(model)
>>> a.fit(X_train) # If analyzer needs training.
>>> analysis = a.analyze(X_test)
>>>
>>> state = a.save()
>>> a_new = A.load(*state)
>>> analysis = a_new.analyze(X_test)
```

### Parameters

- **model** – A Keras model.
- **disable\_model\_checks** – Do not execute model checks that enforce compatibility of analyzer and model.

---

**Note:** To develop a new analyzer derive from [AnalyzerNetworkBase](#).

---

### **analyze** (*X*)

Analyze the behavior of model on input *X*.

**Parameters** **X** – Input as expected by model.

### **fit** (*\*args*, *\*\*kwargs*)

Stub that eats arguments. If an analyzer needs training include [TrainerMixin](#).

**Parameters** **disable\_no\_training\_warning** – Do not warn if this function is called despite no training is needed.

### **fit\_generator** (*\*args*, *\*\*kwargs*)

Stub that eats arguments. If an analyzer needs training include [TrainerMixin](#).

**Parameters** **disable\_no\_training\_warning** – Do not warn if this function is called despite no training is needed.

### **static load** (*class\_name*, *state*)

Resembles an analyzer from the state created by `analyzer.save()`.

#### Parameters

- **class\_name** – The analyzer's class name.
- **state** – The analyzer's state.

### **static load\_npz** (*fname*)

Resembles an analyzer from the file created by `analyzer.save_npz()`.

**Parameters** **fname** – The file's name.

### **save** ()

Save state of analyzer, can be passed to `Analyzer.load()` to resemble the analyzer.

**Returns** The class name and the state.

### **save\_npz** (*fname*)

Save state of analyzer, can be passed to `Analyzer.load_npz()` to resemble the analyzer.

**Parameters** **fname** – The file's name.

### **class** `innvestigate.analyzer.base.TrainerMixin`

Mixin for analyzer that adapt to data.



This convenience interface exposes a Keras like training routing to the user.

**fit** (*X=None, batch\_size=32, \*\*kwargs*)

Takes the same parameters as Keras's `model.fit()` function.

**fit\_generator** (*\*args, \*\*kwargs*)

Takes the same parameters as Keras's `model.fit_generator()` function.

**class** `innvestigate.analyzer.base.OneEpochTrainerMixin`

Exposes the same interface and functionality as `TrainerMixin` except that the training is limited to one epoch.

**fit** (*\*args, \*\*kwargs*)

Same interface as `fit()` of `TrainerMixin` except that the parameter `epoch` is fixed to 1.

**fit\_generator** (*\*args, \*\*kwargs*)

Same interface as `fit_generator()` of `TrainerMixin` except that the parameter `epoch` is fixed to 1.

**class** `innvestigate.analyzer.base.AnalyzerNetworkBase` (*model, neuron\_selection\_mode='max\_activation', allow\_lambda\_layers=False, \*\*kwargs*)

Convenience interface for analyzers.

This class provides helpful functionality to create analyzer's. Basically it:

- takes the input model and adds a layer that selects the desired output neuron to analyze.
- passes the new model to `_create_analysis()` which should return the analysis as Keras tensors.
- compiles the function and serves the output to `analyze()` calls.
- allows `_create_analysis()` to return tensors that are intercept for debugging purposes.

#### Parameters

- **neuron\_selection\_mode** – How to select the neuron to analyze. Possible values are 'max\_activation', 'index' for the neuron (expects indices at `analyze()` calls), 'all' take all neurons.
- **allow\_lambda\_layers** – Allow the model to contain lambda layers.

**analyze** (*X, neuron\_selection=None*)

Same interface as `Analyzer` besides

**Parameters** **neuron\_selection** – If `neuron_selection_mode` is 'index' this should be an integer with the index for the chosen neuron.

**create\_analyzer\_model** ()

Creates the analyze functionality. If not called beforehand it will be called by `analyze()`.

**class** `innvestigate.analyzer.base.ReverseAnalyzerBase` (*model, reverse\_verbose=False, reverse\_clip\_values=False, reverse\_project\_bottleneck\_layers=False, reverse\_check\_min\_max\_values=False, reverse\_check\_finite=False, reverse\_keep\_tensors=False, reverse\_reapply\_on\_copied\_layers=False, \*\*kwargs*)

Convenience class for analyzers that revert the model's structure.

This class contains many helper functions around the graph reverse function `innvestigate.utils.keras.graph.reverse_model()`.

The deriving classes should specify how the graph should be reverted by implementing the following functions:

- `_reverse_mapping(layer)()` given a layer this function returns a reverse mapping for the layer as specified in `innvestigate.utils.keras.graph.reverse_model()` or `None`.

This function can be implemented, but it is encouraged to implement a default mapping and add additional changes with the function `_add_conditional_reverse_mapping()` (see below).

The default behavior is finding a conditional mapping (see below), if none is found, `_default_reverse_mapping()` is applied.

- `_default_reverse_mapping()` defines the default reverse mapping.
- `_head_mapping()` defines how the outputs of the model should be instantiated before they are passed to the reversed network.

Furthermore other parameters of the function `innvestigate.utils.keras.graph.reverse_model()` can be changed by setting the according parameters of the init function:

#### Parameters

- **`reverse_verbose`** – Print information on the reverse process.
- **`reverse_clip_values`** – Clip the values that are passed along the reverted network. Expects tuple (min, max).
- **`reverse_project_bottleneck_layers`** – Project the value range of bottleneck tensors in the reverse network into another range.
- **`reverse_check_min_max_values`** – Print the min/max values observed in each tensor along the reverse network whenever `analyze()` is called.
- **`reverse_check_finite`** – Check if values passed along the reverse network are finite.
- **`reverse_keep_tensors`** – Keeps the tensors created in the backward pass and stores them in the attribute `_reversed_tensors`.
- **`reverse_reapply_on_copied_layers`** – See `innvestigate.utils.keras.graph.reverse_model()`.

## 1.2.2 Gradient methods

```
class innvestigate.analyzer.gradient_based.BaselineGradient(model, post-  
                                                         process=None,  
                                                         **kwargs)
```

Gradient analyzer based on build-in gradient.

Returns as analysis the function value with respect to the input. The gradient is computed via the build in function. Is mainly used for debugging purposes.

**Parameters** **`model`** – A Keras model.

```
class innvestigate.analyzer.gradient_based.Gradient(model, postprocess=None,  
                                                         **kwargs)
```

Gradient analyzer.

Returns as analysis the function value with respect to the input. The gradient is computed via the library's network reverting.

**Parameters** **`model`** – A Keras model.

```
class innvestigate.analyzer.gradient_based.InputTimesGradient (model,  
                                                             **kwargs)
```

Input\*Gradient analyzer.

**Parameters** **model** – A Keras model.

```
class innvestigate.analyzer.gradient_based.Deconvnet (model, **kwargs)
```

Deconvnet analyzer.

Applies the “deconvnet” algorithm to analyze the model.

**Parameters** **model** – A Keras model.

```
class innvestigate.analyzer.gradient_based.GuidedBackprop (model, **kwargs)
```

Guided backprop analyzer.

Applies the “guided backprop” algorithm to analyze the model.

**Parameters** **model** – A Keras model.

```
class innvestigate.analyzer.gradient_based.IntegratedGradients (model,  
                                                                steps=64,  
                                                                **kwargs)
```

Integrated gradient analyzer.

Applies the “integrated gradient” algorithm to analyze the model.

**Parameters**

- **model** – A Keras model.
- **steps** – Number of steps to use average along integration path.

```
class innvestigate.analyzer.gradient_based.SmoothGrad (model, augment_by_n=64,  
                                                         **kwargs)
```

Smooth grad analyzer.

Applies the “smooth grad” algorithm to analyze the model.

**Parameters**

- **model** – A Keras model.
- **augment\_by\_n** – Number of distortions to average for smoothing.

### 1.2.3 Layer-wise relevance propagation

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.BaselineLRPZ (model,  
                                                                              **kwargs)
```

LRPZ analyzer - for testing purpose only.

Applies the “LRP-Z” algorithm to analyze the model. Based on the gradient times the input formula. **This formula holds only for ReLU/MaxPooling networks, for which LRP-Z collapses into the stated formula.**

**Parameters** **model** – A Keras model.

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRP (model,  
                                                                    *args,  
                                                                    **kwargs)
```

Base class for LRP-based model analyzers

**Parameters**

- **model** – A Keras model.

- **rule** – A rule can be a string or a Rule object, lists thereof or a list of conditions [(Condition, Rule), ... ] gradient.
- **input\_layer\_rule** – either a Rule object, atuple of (low, high) the min/max pixel values of the inputs

**create\_rule\_mapping** (*layer, reverse\_state*)

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPZ (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPZIgnoreBias (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPEpsilon (model,  
                                                                epsilon=1e-  
                                                                07,  
                                                                bias=True,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPEpsilonIgnoreBias (model,  
                                                                epsilon=1e-  
                                                                07,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPWSquare (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPFlat (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPAlphaBeta (model,  
                                                                alpha=  
                                                                None,  
                                                                beta=None,  
                                                                bias=True,  
                                                                *args,  
                                                                **kwargs)
```

Base class for LRP AlphaBeta

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPAlpha2Beta1 (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPAlpha2Beta1IgnoreBias (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPAlpha1Beta0 (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPAlpha1Beta0IgnoreBias (model,  
                                                                *args,  
                                                                **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPZPlus (model,
                                                                    *args,
                                                                    **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPZPlusFast (model,
                                                                    *args,
                                                                    **kwargs)
```

The ZPlus rule is a special case of the AlphaBetaRule for alpha=1, beta=0 and assumes inputs  $x \geq 0$ .

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPSequentialPresetA (model,
                                                                    ep-
                                                                    silon=0.1,
                                                                    *args,
                                                                    **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPSequentialPresetB (model,
                                                                    ep-
                                                                    silon=0.1,
                                                                    *args,
                                                                    **kwargs)
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPSequentialPresetAFlat (model,
                                                                    *ar
                                                                    **k
```

```
class innvestigate.analyzer.relevance_based.relevance_analyzer.LRPSequentialPresetBFlat (model,
                                                                    *ar
                                                                    **k
```

### 1.2.4 Pattern methods

```
class innvestigate.analyzer.pattern_based.PatternNet (model, patterns=None, pat-
                                                                    tern_type=None, **kwargs)
```

PatternNet analyzer.

Applies the “PatternNet” algorithm to analyze the model’s predictions.

#### Parameters

- **model** – A Keras model.
- **patterns** – Pattern computed by `innvestigate.tools.PatternComputer`. If `None` `fit()` needs to be called.
- **allow\_lambda\_layers** – Approximate lambda layers with the gradient.
- **reverse\_project\_bottleneck\_layers** – Project the analysis vector into range  $[-1, +1]$ . (default: True)

```
class innvestigate.analyzer.pattern_based.PatternAttribution (model, pat-
                                                                    terns=None, pat-
                                                                    tern_type=None,
                                                                    **kwargs)
```

PatternAttribution analyzer.

Applies the “PatternNet” algorithm to analyze the model’s predictions.

#### Parameters

- **model** – A Keras model.

- **patterns** – Pattern computed by `innvestigate.tools.PatternComputer`. If `None` `fit()` needs to be called.
- **allow\_lambda\_layers** – Approximate lambda layers with the gradient.
- **reverse\_project\_bottleneck\_layers** – Project the analysis vector into range `[-1, +1]`. (default: `True`)

## 1.2.5 Deep Taylor

```
class innvestigate.analyzer.deeptaylor.DeepTaylor (model, *args, **kwargs)
```

```
class innvestigate.analyzer.deeptaylor.BoundedDeepTaylor (model, low=None, high=None, **kwargs)
```

This class implements the DeepTaylor algorithm for neural networks with ReLU activation and bounded input ranges.

Currently no batch-norm layers are supported.

### Parameters

- **low** – Lowest value of the input range. See `Z_B` rule.
- **high** – Highest value of the input range. See `Z_B` rule.

## 1.2.6 DeepLIFT

```
class innvestigate.analyzer.deeplift.DeepLIFTWrapper (model, **kwargs)
```

```
analyze (X, neuron_selection=None)
```

Same interface as `Analyzer` besides

**Parameters** **neuron\_selection** – If `neuron_selection_mode` is 'index' this should be an integer with the index for the chosen neuron.

```
create_deep_lift_func ()
```

## 1.2.7 Misc

```
class innvestigate.analyzer.misc.Random (model, stddev=1, **kwargs)
```

```
class innvestigate.analyzer.misc.Input (model, neuron_selection_mode='max_activation', allow_lambda_layers=False, **kwargs)
```

## 1.3 innvestigate.applications

### 1.3.1 Imagenet

Example applications for image classification.

Each function returns a pretrained ImageNet model. The models are based on `keras.applications` models and contain additionally pretrained patterns.

The returned dictionary contains the following keys: `model`, `in`, `sm_out`, `out`, `image_shape`, `color_coding`, `preprocess_f`, `patterns`.

Function parameters:

**param load\_weights** Download or access cached weights.

**param load\_patterns** Download or access cached patterns.

```
innvestigate.applications.imagenet.vgg16 (load_weights=False, load_patterns=False)
innvestigate.applications.imagenet.vgg19 (load_weights=False, load_patterns=False)
innvestigate.applications.imagenet.resnet50 (load_weights=False, load_patterns=False)
innvestigate.applications.imagenet.inception_v3 (load_weights=False,
                                                load_patterns=False)
innvestigate.applications.imagenet.inception_resnet_v2 (load_weights=False,
                                                         load_patterns=False)
innvestigate.applications.imagenet.densenet121 (load_weights=False,
                                                  load_patterns=False)
innvestigate.applications.imagenet.densenet169 (load_weights=False,
                                                  load_patterns=False)
innvestigate.applications.imagenet.densenet201 (load_weights=False,
                                                  load_patterns=False)
innvestigate.applications.imagenet.nasnet_large (load_weights=False,
                                                  load_patterns=False)
innvestigate.applications.imagenet.nasnet_mobile (load_weights=False,
                                                    load_patterns=False)
```

## 1.4 innvestigate.tools

### 1.4.1 Pattern Computation

```
class innvestigate.tools.pattern.PatternComputer (model, pattern_type='linear', compute_layers_in_parallel=True,
                                                  gpus=None)
```

Pattern computer.

Computes a pattern for each layer with a kernel of a given model.

#### Parameters

- **model** – A Keras model.
- **pattern\_type** – A string or a tuple of strings. Valid types are 'linear', 'relu', 'relu.positive', 'relu.negative'.
- **compute\_layers\_in\_parallel** – Not supported yet. Compute all patterns at once. Otherwise compute layer after layer.
- **gpus** – Not supported yet. Gpus to use.

```
compute (X, batch_size=32, verbose=0)
```

Compute and return the patterns for the model and the data X.

#### Parameters

- **X** – Data to compute patterns.
- **batch\_size** – Batch size to use.

- **verbose** – As for keras model.fit.

**compute\_generator** (*generator*, *\*\*kwargs*)

Compute and return the patterns for the model and the data X.

#### Parameters

- **generator** – Data to compute patterns.
- **kwargs** – Same as for keras model.fit\_generator.

## 1.4.2 Perturbation Analysis

```
class innvestigate.tools.perturbate.Perturbation (perturbation_function,  
                                                num_perturbed_regions=0,  
                                                region_shape=(9, 9), re-  
                                                duce_function=<function  
                                                mean>, aggrega-  
                                                tion_function=<function  
                                                mean>,  
                                                pad_mode='reflect', in_place=False,  
                                                value_range=None)
```

Perturbation of pixels based on analysis result.

#### Parameters

- **perturbation\_function** (*function or callable or str*) – Defines the function with which the samples are perturbed. Can be a function or a string that defines a predefined perturbation function.
- **num\_perturbed\_regions** (*int*) – Number of regions to be perturbed.
- **reduce\_function** (*function or callable*) – Function to reduce the analysis result to one channel, e.g. mean or max function.
- **aggregation\_function** (*function or callable*) – Function to aggregate the analysis over subregions.
- **pad\_mode** (*str or function or callable*) – How to pad if the image cannot be subdivided into an integer number of regions. As in numpy.pad.
- **in\_place** (*bool*) – If true, the perturbations are performed in place, i.e. the input samples are modified.
- **value\_range** (*tuple*) – Minimal and maximal value after perturbation as a tuple: (min\_val, max\_val). The input is clipped to this range

**aggregate\_regions** (*analysis*)

**static compute\_perturbation\_mask** (*ranks*, *num\_perturbed\_regions*)

**static compute\_region\_ordering** (*aggregated\_regions*)

**expand\_regions\_to\_pixels** (*regions*)

**pad** (*analysis*)

**perturbate\_on\_batch** (*x*, *analysis*)

#### Parameters

- **x** (*numpy.ndarray*) – Batch of images.
- **analysis** (*numpy.ndarray*) – Analysis of this batch.



**Returns** Batch of perturbed images

**Return type** `numpy.ndarray`

**perturbate\_regions** (*x*, *perturbation\_mask\_regions*)

**reshape\_region\_pixels** (*region\_pixels*, *target\_shape*)

**reshape\_to\_regions** (*analysis*)

```
class innvestigate.tools.perturbate.PerturbationAnalysis (analyzer, model,
                                                         generator, perturbation,
                                                         steps=1, regions_per_step=1,
                                                         recompute_analysis=False,
                                                         verbose=False)
```

Performs the perturbation analysis.

#### Parameters

- **analyzer** (`innvestigate.analyzer.base.AnalyzerBase`) – Analyzer.
- **model** (`keras.engine.training.Model`) – Trained Keras model.
- **generator** (`innvestigate.utils.BatchSequence`) – Data generator.
- **perturbation** (`innvestigate.tools.Perturbation`) – Instance of Perturbation class that performs the perturbation.
- **steps** (*int*) – Number of perturbation steps.
- **regions\_per\_step** (*float*) – Number of regions that are perturbed per step.
- **recompute\_analysis** (*bool*) – If true, the analysis is recomputed after each perturbation step.
- **verbose** – If true, print some useful information, e.g. timing, progress etc.

**compute\_on\_batch** (*x*, *analysis=None*, *return\_analysis=False*)

Computes the analysis and perturbs the input batch accordingly.

#### Parameters

- **x** (`numpy.ndarray`) – Samples.
- **analysis** – Analysis of x. If None, it is recomputed.

**compute\_perturbation\_analysis** ()

```
evaluate_generator (generator, steps=None, max_queue_size=10, workers=1,
                    use_multiprocessing=False)
```

Evaluates the model on a data generator.

The generator should return the same kind of data as accepted by `test_on_batch`. For documentation, refer to `keras.engine.training.evaluate_generator` (<https://keras.io/models/model/>)

**evaluate\_on\_batch** (*x*, *y*, *analysis=None*, *sample\_weight=None*)

Perturbs the input batch and scores the model on the perturbed batch.

#### Parameters

- **x** (`numpy.ndarray`) – Samples.
- **y** (`numpy.ndarray`) – Labels.
- **analysis** (`numpy.ndarray`) – Analysis of x.
- **sample\_weight** (*None*) – Sample weights.

**Returns** List of test scores.

**Return type** list

## 1.5 innvestigate.utils

`innvestigate.utils.model_wo_softmax(*args, **kwargs)`

`innvestigate.utils.to_list(l)`

**class** `innvestigate.utils.BatchSequence(Xs, batch_size=32)`

**class** `innvestigate.utils.TargetAugmentedSequence(sequence, augment_f)`

`innvestigate.utils.preprocess_images(images, color_coding=None)`

`innvestigate.utils.postprocess_images(images, color_coding=None, channels_first=None)`

### 1.5.1 Visualizations

`innvestigate.utils.visualizations.project(X, output_range=(0, 1), absmax=None, input_is_postive_only=False)`

`innvestigate.utils.visualizations.heatmap(X, cmap_type='seismic', reduce_op='sum', reduce_axis=-1, **kwargs)`

`innvestigate.utils.visualizations.graymap(X, **kwargs)`

`innvestigate.utils.visualizations.gamma(X, gamma=0.5, minamp=0, maxamp=None)`

Apply gamma correction to an input array X while maintaining the relative order of entries, also for negative vs positive values in X. the fxn firstly determines the max amplitude in both positive and negative direction and then applies gamma scaling to the positive and negative values of the array separately, according to the common amplitude.

#### Parameters

- **gamma** – the gamma parameter for gamma scaling
- **minamp** – the smallest absolute value to consider. if not given assumed to be zero (neutral value for relevance, min value for saliency, ...). values above and below minamp are treated separately.
- **maxamp** – the largest absolute value to consider relative to the neutral value minamp if not given determined from the given data.

## CHAPTER 2

---

### Indices and tables

---

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



### i

- `innvestigate.analyzer.base`, [3](#)
- `innvestigate.analyzer.deeplift`, [10](#)
- `innvestigate.analyzer.deeptaylor`, [10](#)
- `innvestigate.analyzer.gradient_based`, [6](#)
- `innvestigate.analyzer.misc`, [10](#)
- `innvestigate.analyzer.pattern_based`, [9](#)
- `innvestigate.analyzer.relevance_based.relevance_analyzer`,  
[7](#)
- `innvestigate.applications.imagenet`, [10](#)
- `innvestigate.tools.pattern`, [11](#)
- `innvestigate.tools.perturbate`, [12](#)
- `innvestigate.utils`, [14](#)
- `innvestigate.utils.visualizations`, [14](#)



## A

aggregate\_regions() (innvestigate.tools.perturbate.Perturbation method), 12  
 analyze() (innvestigate.analyzer.base.AnalyzerBase method), 4  
 analyze() (innvestigate.analyzer.base.AnalyzerNetworkBase method), 5  
 analyze() (innvestigate.analyzer.deeplift.DeepLIFTWrapper method), 10  
 AnalyzerBase (class in innvestigate.analyzer.base), 3  
 AnalyzerNetworkBase (class in innvestigate.analyzer.base), 5  
 compute\_region\_ordering() (innvestigate.tools.perturbate.Perturbation static method), 12  
 create\_analyzer() (in module innvestigate.analyzer), 3  
 create\_analyzer\_model() (innvestigate.analyzer.base.AnalyzerNetworkBase method), 5  
 create\_deep\_lift\_func() (innvestigate.analyzer.deeplift.DeepLIFTWrapper method), 10  
 create\_rule\_mapping() (innvestigate.analyzer.relevance\_based.relevance\_analyzer.LRP method), 8

## B

BaselineGradient (class in innvestigate.analyzer.gradient\_based), 6  
 BaselineLRPZ (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 7  
 BatchSequence (class in innvestigate.utils), 14  
 BoundedDeepTaylor (class in innvestigate.analyzer.deeptaylor), 10

## C

compute() (innvestigate.tools.pattern.PatternComputer method), 11  
 compute\_generator() (innvestigate.tools.pattern.PatternComputer method), 12  
 compute\_on\_batch() (innvestigate.tools.perturbate.PerturbationAnalysis method), 13  
 compute\_perturbation\_analysis() (innvestigate.tools.perturbate.PerturbationAnalysis method), 13  
 compute\_perturbation\_mask() (innvestigate.tools.perturbate.Perturbation static method), 12

## D

Deconvnet (class in innvestigate.analyzer.gradient\_based), 7  
 DeepLIFTWrapper (class in innvestigate.analyzer.deeplift), 10  
 DeepTaylor (class in innvestigate.analyzer.deeptaylor), 10  
 densenet121() (in module innvestigate.applications.imagenet), 11  
 densenet169() (in module innvestigate.applications.imagenet), 11  
 densenet201() (in module innvestigate.applications.imagenet), 11

## E

evaluate\_generator() (innvestigate.tools.perturbate.PerturbationAnalysis method), 13  
 evaluate\_on\_batch() (innvestigate.tools.perturbate.PerturbationAnalysis method), 13  
 expand\_regions\_to\_pixels() (innvestigate.tools.perturbate.Perturbation method), 12

## F

fit() (innvestigate.analyzer.base.AnalyzerBase method), 4

fit() (innvestigate.analyzer.base.OneEpochTrainerMixin method), 5  
 fit() (innvestigate.analyzer.base.TrainerMixin method), 5  
 fit\_generator() (innvestigate.analyzer.base.AnalyzerBase method), 4  
 fit\_generator() (innvestigate.analyzer.base.OneEpochTrainerMixin method), 5  
 fit\_generator() (innvestigate.analyzer.base.TrainerMixin method), 5

## G

gamma() (in module innvestigate.utils.visualizations), 14  
 Gradient (class in innvestigate.analyzer.gradient\_based), 6  
 graymap() (in module innvestigate.utils.visualizations), 14  
 GuidedBackprop (class in innvestigate.analyzer.gradient\_based), 7

## H

heatmap() (in module innvestigate.utils.visualizations), 14

## I

inception\_resnet\_v2() (in module innvestigate.applications.imagenet), 11  
 inception\_v3() (in module innvestigate.applications.imagenet), 11  
 innvestigate.analyzer.base (module), 3  
 innvestigate.analyzer.deeplift (module), 10  
 innvestigate.analyzer.deeptaylor (module), 10  
 innvestigate.analyzer.gradient\_based (module), 6  
 innvestigate.analyzer.misc (module), 10  
 innvestigate.analyzer.pattern\_based (module), 9  
 innvestigate.analyzer.relevance\_based.relevance\_analyzer (module), 7  
 innvestigate.applications.imagenet (module), 10  
 innvestigate.tools.pattern (module), 11  
 innvestigate.tools.perturbate (module), 12  
 innvestigate.utils (module), 14  
 innvestigate.utils.visualizations (module), 14  
 Input (class in innvestigate.analyzer.misc), 10  
 InputTimesGradient (class in innvestigate.analyzer.gradient\_based), 6  
 IntegratedGradients (class in innvestigate.analyzer.gradient\_based), 7

## L

load() (innvestigate.analyzer.base.AnalyzerBase static method), 4  
 load\_npz() (innvestigate.analyzer.base.AnalyzerBase static method), 4

LRP (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 7  
 LRPAAlpha1Beta0 (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPAAlpha1Beta0IgnoreBias (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPAAlpha2Beta1 (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPAAlpha2Beta1IgnoreBias (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPAAlphaBeta (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPepsilon (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPepsilonIgnoreBias (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPFlat (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPSquentialPresetA (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 9  
 LRPSquentialPresetAFlat (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 9  
 LRPSquentialPresetB (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 9  
 LRPSquentialPresetBFlat (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 9  
 LRPWSquare (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPZ (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPZIgnoreBias (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPZPlus (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 8  
 LRPZPlusFast (class in innvestigate.analyzer.relevance\_based.relevance\_analyzer), 9



## M

model\_wo\_softmax() (in module innvestigate.utils), 14

## N

nasnet\_large() (in module innvestigate.applications.imagenet), 11

nasnet\_mobile() (in module innvestigate.applications.imagenet), 11

NotAnalyzeableModelError, 3

## O

OneEpochTrainerMixin (class in innvestigate.analyzer.base), 5

## P

pad() (innvestigate.tools.perturbate.Perturbation method), 12

PatternAttribution (class in innvestigate.analyzer.pattern\_based), 9

PatternComputer (class in innvestigate.tools.pattern), 11

PatternNet (class in innvestigate.analyzer.pattern\_based), 9

perturbate\_on\_batch() (innvestigate.tools.perturbate.Perturbation method), 12

perturbate\_regions() (innvestigate.tools.perturbate.Perturbation method), 13

Perturbation (class in innvestigate.tools.perturbate), 12

PerturbationAnalysis (class in innvestigate.tools.perturbate), 13

postprocess\_images() (in module innvestigate.utils), 14

preprocess\_images() (in module innvestigate.utils), 14

project() (in module innvestigate.utils.visualizations), 14

## R

Random (class in innvestigate.analyzer.misc), 10

reshape\_region\_pixels() (innvestigate.tools.perturbate.Perturbation method), 13

reshape\_to\_regions() (innvestigate.tools.perturbate.Perturbation method), 13

resnet50() (in module innvestigate.applications.imagenet), 11

ReverseAnalyzerBase (class in innvestigate.analyzer.base), 5

## S

save() (innvestigate.analyzer.base.AnalyzerBase method), 4

save\_npz() (innvestigate.analyzer.base.AnalyzerBase method), 4

SmoothGrad (class in innvestigate.analyzer.gradient\_based), 7

## T

TargetAugmentedSequence (class in innvestigate.utils), 14

to\_list() (in module innvestigate.utils), 14

TrainerMixin (class in innvestigate.analyzer.base), 4

## V

vgg16() (in module innvestigate.applications.imagenet), 11

vgg19() (in module innvestigate.applications.imagenet), 11