

---

# **Foolbox**

***Release 3.0.4***

**Jonas Rauber**

**Aug 29, 2020**



# USER API

<b>1</b>	<b>foolbox.models</b>	<b>3</b>
1.1	Models .....	3
1.2	Wrappers .....	3
1.3	Detailed description .....	3
<b>2</b>	<b>foolbox.attacks</b>	<b>5</b>
<b>3</b>	<b>foolbox.criteria</b>	<b>19</b>
3.1	Misclassification .....	19
3.2	TargetedMisclassification .....	19
3.3	Criterion .....	19
<b>4</b>	<b>foolbox.distances</b>	<b>21</b>
4.1	Detailed description .....	21
<b>5</b>	<b>foolbox.utils</b>	<b>23</b>
<b>6</b>	<b>foolbox.plot</b>	<b>25</b>
<b>7</b>	<b>foolbox.zoo</b>	<b>27</b>
7.1	Get Model .....	27
7.2	Fetch Weights .....	28
<b>8</b>	<b>foolbox.devutils</b>	<b>29</b>
<b>9</b>	<b>foolbox.tensorboard</b>	<b>31</b>
<b>10</b>	<b>foolbox.types</b>	<b>33</b>
<b>11</b>	<b>Indices and tables</b>	<b>35</b>
	<b>Python Module Index</b>	<b>37</b>
	<b>Index</b>	<b>39</b>



Foolbox is a Python toolbox to create adversarial examples that fool neural networks. *Foolbox 3.0* a.k.a. *Foolbox Native* has been completely rewritten from scratch. It is now built on top of [EagerPy](#) and comes with native support for these frameworks:

- [PyTorch](#)
- [TensorFlow](#)
- [JAX](#)

Foolbox comes with a *large collection of adversarial attacks*, both gradient-based white-box attacks as well as decision-based and score-based black-box attacks.

The source code and a [minimal working example](#) can be found on [GitHub](#).



## 1.1 Models

---

*Model*

---

*PyTorchModel*

---

*TensorFlowModel*

---

*JAXModel*

---

*NumPyModel*

---

## 1.2 Wrappers

---

*TransformBoundsWrapper*

---

## 1.3 Detailed description

**class** foolbox.models.**Model**

**transform\_bounds** (*bounds*)

Returns a new model with the desired bounds and updates the preprocessing accordingly

**Parameters** **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds, ]*) –

**Return type** foolbox.models.base.Model

**class** foolbox.models.**PyTorchModel** (*model, bounds, device=None, preprocessing=None*)

**Parameters**

• **model** (*Any*) –

• **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds, ]*) –

• **device** (*Any*) –

• **Any**] **preprocessing** (*Optional[Dict[str, ]*) –

**class** foolbox.models.**TensorFlowModel** (*model, bounds, device=None, preprocessing=None*)

**Parameters**

- **model** (*Any*) –
- **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds,)* –
- **device** (*Any*) –
- **Any]** **preprocessing** (*Optional[Dict[str,)* –

```
class foolbox.models.JAXModel (model, bounds, preprocessing=None,  
                               data_format='channels_last')
```

#### Parameters

- **model** (*Any*) –
- **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds,)* –
- **Any]** **preprocessing** (*Optional[Dict[str,)* –
- **data\_format** (*Optional[str]*) –

```
class foolbox.models.NumPyModel (model, bounds, data_format=None)
```

#### Parameters

- **model** (*Callable*) –
- **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds,)* –
- **data\_format** (*Optional[str]*) –

```
class foolbox.models.TransformBoundsWrapper (model, bounds)
```

#### Parameters

- **model** (*foolbox.models.base.Model*) –
- **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds,)* –

```
transform_bounds (bounds, inplace=False)
```

Returns a new model with the desired bounds and updates the preprocessing accordingly

#### Parameters

- **Tuple[float, float]** **bounds** (*Union[foolbox.types.Bounds,)* –
- **inplace** (*bool*) –

**Return type** `foolbox.models.base.Model`



**FOOLBOX . ATTACKS**

<i>L2ContrastReductionAttack</i>	Reduces the contrast of the input using a perturbation of the given size
<i>VirtualAdversarialAttack</i>	Second-order gradient-based attack on the logits.
<i>DDNAttack</i>	The Decoupled Direction and Norm L2 adversarial attack.
<i>L2ProjectedGradientDescentAttack</i>	L2 Projected Gradient Descent
<i>LinfProjectedGradientDescentAttack</i>	Linf Projected Gradient Descent
<i>L2BasicIterativeAttack</i>	L2 Basic Iterative Method
<i>LinfBasicIterativeAttack</i>	L-infinity Basic Iterative Method
<i>L2FastGradientAttack</i>	Fast Gradient Method (FGM)
<i>LinfFastGradientAttack</i>	Fast Gradient Sign Method (FGSM)
<i>L2AdditiveGaussianNoiseAttack</i>	Samples Gaussian noise with a fixed L2 size.
<i>L2AdditiveUniformNoiseAttack</i>	Samples uniform noise with a fixed L2 size.
<i>L2ClippingAwareAdditiveGaussianNoiseAttack</i>	Samples Gaussian noise with a fixed L2 size after clipping.
<i>L2ClippingAwareAdditiveUniformNoiseAttack</i>	Samples uniform noise with a fixed L2 size after clipping.
<i>LinfAdditiveUniformNoiseAttack</i>	Samples uniform noise with a fixed L-infinity size
<i>L2RepeatedAdditiveGaussianNoiseAttack</i>	Repeatedly samples Gaussian noise with a fixed L2 size.
<i>L2RepeatedAdditiveUniformNoiseAttack</i>	Repeatedly samples uniform noise with a fixed L2 size.
<i>L2ClippingAwareRepeatedAdditiveGaussianNoiseAttack</i>	Repeatedly samples Gaussian noise with a fixed L2 size after clipping.
<i>L2ClippingAwareRepeatedAdditiveUniformNoiseAttack</i>	Repeatedly samples uniform noise with a fixed L2 size after clipping.
<i>LinfRepeatedAdditiveUniformNoiseAttack</i>	Repeatedly samples uniform noise with a fixed L-infinity size.
<i>InversionAttack</i>	Creates “negative images” by inverting the pixel values.
<i>BinarySearchContrastReductionAttack</i>	Reduces the contrast of the input using a binary search to find the smallest adversarial perturbation
<i>LinearSearchContrastReductionAttack</i>	Reduces the contrast of the input using a linear search to find the smallest adversarial perturbation
<i>L2CarliniWagnerAttack</i>	Implementation of the Carlini & Wagner L2 Attack.
<i>NewtonFoolAttack</i>	Implementation of the NewtonFool Attack.
<i>EADAttack</i>	Implementation of the EAD Attack with EN Decision Rule.
<i>GaussianBlurAttack</i>	Blurs the inputs using a Gaussian filter with linearly increasing standard deviation.
<i>L2DeepFoolAttack</i>	A simple and fast gradient-based adversarial attack.

Continued on next page

Table 1 – continued from previous page

<i>LinfDeepFoolAttack</i>	A simple and fast gradient-based adversarial attack.
<i>SaltAndPepperNoiseAttack</i>	Increases the amount of salt and pepper noise until the input is misclassified.
<i>LinearSearchBlendedUniformNoiseAttack</i>	Blends the input with a uniform noise input until it is misclassified.
<i>BinarizationRefinementAttack</i>	For models that preprocess their inputs by binarizing the inputs, this attack can improve adversarials found by other attacks.
<i>DatasetAttack</i>	Draws randomly from the given dataset until adversarial examples for all inputs have been found.
<i>BoundaryAttack</i>	A powerful adversarial attack that requires neither gradients nor probabilities.
<i>L0BrendelBethgeAttack</i>	L0 variant of the Brendel & Bethge adversarial attack.
<i>L1BrendelBethgeAttack</i>	L1 variant of the Brendel & Bethge adversarial attack.
<i>L2BrendelBethgeAttack</i>	L2 variant of the Brendel & Bethge adversarial attack.
<i>LinfinityBrendelBethgeAttack</i>	L-infinity variant of the Brendel & Bethge adversarial attack.
<i>FGM</i>	alias of foolbox.attacks.fast_gradient_method.L2FastGradientAttack
<i>FGSM</i>	alias of foolbox.attacks.fast_gradient_method.LinfFastGradientAttack
<i>L2PGD</i>	alias of foolbox.attacks.projected_gradient_descent.L2ProjectedGradientDescentAttack
<i>LinfPGD</i>	alias of foolbox.attacks.projected_gradient_descent.LinfProjectedGradientDescentAttack
<i>PGD</i>	alias of foolbox.attacks.projected_gradient_descent.LinfProjectedGradientDescentAttack

**class** foolbox.attacks.L2ContrastReductionAttack (\*, target=0.5)

Reduces the contrast of the input using a perturbation of the given size

**Parameters** **target** (*float*) – Target relative to the bounds from 0 (min) to 1 (max) towards which the contrast is reduced

**class** foolbox.attacks.VirtualAdversarialAttack (steps, xi=1e-06)

Second-order gradient-based attack on the logits.<sup>1</sup> The attack calculate an untargeted adversarial perturbation by performing a approximated second order optimization step on the KL divergence between the unperturbed predictions and the predictions for the adversarial perturbation. This attack was originally introduced as the Virtual Adversarial Training<sup>1</sup> method.

**Parameters**

- **steps** (*int*) – Number of update steps.
- **xi** (*float*) – L2 distance between original image and first adversarial proposal.

<sup>1</sup> Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, Shin Ishii, “Distributional Smoothing with Virtual Adversarial Training”, <https://arxiv.org/abs/1507.00677>

## References

**class** foolbox.attacks.DDNAttack (\*, *init\_epsilon=1.0*, *steps=10*, *gamma=0.05*)  
The Decoupled Direction and Norm L2 adversarial attack.<sup>2</sup>

### Parameters

- **init\_epsilon** (*float*) – Initial value for the norm/epsilon ball.
- **steps** (*int*) – Number of steps for the optimization.
- **gamma** (*float*) – Factor by which the norm will be modified:  $\text{new\_norm} = \text{norm} * (1 + \text{or} - \text{gamma})$ .

## References

**class** foolbox.attacks.L2ProjectedGradientDescentAttack (\*, *rel\_stepsize=0.025*,  
*abs\_stepsize=None*,  
*steps=50*, *random\_start=True*)

L2 Projected Gradient Descent

### Parameters

- **rel\_stepsize** (*float*) – Step size relative to epsilon
- **abs\_stepsize** (*Optional[float]*) – If given, it takes precedence over *rel\_stepsize*.
- **steps** (*int*) – Number of update steps to perform.
- **random\_start** (*bool*) – Whether the perturbation is initialized randomly or starts at zero.

**class** foolbox.attacks.LinfProjectedGradientDescentAttack (\*,  
*rel\_stepsize=0.033333333333333333*,  
*abs\_stepsize=None*,  
*steps=40*, *random\_start=True*)

Linf Projected Gradient Descent

### Parameters

- **rel\_stepsize** (*float*) – Step size relative to epsilon (defaults to 0.01 / 0.3).
- **abs\_stepsize** (*Optional[float]*) – If given, it takes precedence over *rel\_stepsize*.
- **steps** (*int*) – Number of update steps to perform.
- **random\_start** (*bool*) – Whether the perturbation is initialized randomly or starts at zero.

**class** foolbox.attacks.L2BasicIterativeAttack (\*, *rel\_stepsize=0.2*, *abs\_stepsize=None*,  
*steps=10*, *random\_start=False*)

L2 Basic Iterative Method

### Parameters

- **rel\_stepsize** (*float*) – Step size relative to epsilon.
- **abs\_stepsize** (*Optional[float]*) – If given, it takes precedence over *rel\_stepsize*.
- **steps** (*int*) – Number of update steps.

<sup>2</sup> Jérôme Rony, Luiz G. Hafemann, Luiz S. Oliveira, Ismail Ben Ayed, Robert Sabourin, Eric Granger, “Decoupling Direction and Norm for Efficient Gradient-Based L2 Adversarial Attacks and Defenses”, <https://arxiv.org/abs/1811.09600>

- **random\_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

```
class foolbox.attacks.LinfBasicIterativeAttack (*, rel_stepsize=0.2, abs_stepsize=None,
                                             steps=10, random_start=False)
```

L-infinity Basic Iterative Method

#### Parameters

- **rel\_stepsize** (*float*) – Step size relative to epsilon.
- **abs\_stepsize** (*Optional[float]*) – If given, it takes precedence over `rel_stepsize`.
- **steps** (*int*) – Number of update steps.
- **random\_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

```
class foolbox.attacks.L2FastGradientAttack (*, random_start=False)
```

Fast Gradient Method (FGM)

**Parameters** **random\_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

```
class foolbox.attacks.LinfFastGradientAttack (*, random_start=False)
```

Fast Gradient Sign Method (FGSM)

**Parameters** **random\_start** (*bool*) – Controls whether to randomly start within allowed epsilon ball.

```
class foolbox.attacks.L2AdditiveGaussianNoiseAttack
```

Samples Gaussian noise with a fixed L2 size.

```
class foolbox.attacks.L2AdditiveUniformNoiseAttack
```

Samples uniform noise with a fixed L2 size.

```
class foolbox.attacks.L2ClippingAwareAdditiveGaussianNoiseAttack
```

Samples Gaussian noise with a fixed L2 size after clipping.

The implementation is based on [\[#Rauber20\]](#).

## References

```
class foolbox.attacks.L2ClippingAwareAdditiveUniformNoiseAttack
```

Samples uniform noise with a fixed L2 size after clipping.

The implementation is based on [\[#Rauber20\]](#).

## References

```
class foolbox.attacks.LinfAdditiveUniformNoiseAttack
```

Samples uniform noise with a fixed L-infinity size

```
class foolbox.attacks.L2RepeatedAdditiveGaussianNoiseAttack (*, repeats=100,
                                                            check_trivial=True)
```

Repeatedly samples Gaussian noise with a fixed L2 size.

#### Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check\_trivial** (*bool*) – Check whether original sample is already adversarial.

```
class foolbox.attacks.L2RepeatedAdditiveUniformNoiseAttack (*, repeats=100,
                                                           check_trivial=True)
```

Repeatedly samples uniform noise with a fixed L2 size.

#### Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check\_trivial** (*bool*) – Check whether original sample is already adversarial.

```
class foolbox.attacks.L2ClippingAwareRepeatedAdditiveGaussianNoiseAttack (*,
                                                                           re-
                                                                           peats=100,
                                                                           check_trivial=True)
```

Repeatedly samples Gaussian noise with a fixed L2 size after clipping.

The implementation is based on [\[#Rauber20\]\\_](#).

#### References

##### Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check\_trivial** (*bool*) – Check whether original sample is already adversarial.

```
class foolbox.attacks.L2ClippingAwareRepeatedAdditiveUniformNoiseAttack (*,
                                                                           re-
                                                                           peats=100,
                                                                           check_trivial=True)
```

Repeatedly samples uniform noise with a fixed L2 size after clipping.

The implementation is based on [\[#Rauber20\]\\_](#).

#### References

##### Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check\_trivial** (*bool*) – Check whether original sample is already adversarial.

```
class foolbox.attacks.LinfRepeatedAdditiveUniformNoiseAttack (*, repeats=100,
                                                             check_trivial=True)
```

Repeatedly samples uniform noise with a fixed L-infinity size.

##### Parameters

- **repeats** (*int*) – How often to sample random noise.
- **check\_trivial** (*bool*) – Check whether original sample is already adversarial.

```
class foolbox.attacks.InversionAttack (*, distance=None)
    Creates “negative images” by inverting the pixel values.7
```

<sup>7</sup> Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, Radha Poovendran, “On the Limitation of Convolutional Neural Networks in Recognizing Negative Images”, <https://arxiv.org/abs/1607.02533>

## References

**Parameters** `distance` (*Optional*[`foolbox.distances.Distance`]) –

**class** `foolbox.attacks.BinarySearchContrastReductionAttack` (\*, `distance=None`, `binary_search_steps=15`, `target=0.5`)

Reduces the contrast of the input using a binary search to find the smallest adversarial perturbation

### Parameters

- **distance** (*Optional*[`foolbox.distances.Distance`]) – Distance measure for which minimal adversarial examples are searched.
- **binary\_search\_steps** (*int*) – Number of iterations in the binary search. This controls the precision of the results.
- **target** (*float*) – Target relative to the bounds from 0 (min) to 1 (max) towards which the contrast is reduced

**class** `foolbox.attacks.LinearSearchContrastReductionAttack` (\*, `distance=None`, `steps=1000`, `target=0.5`)

Reduces the contrast of the input using a linear search to find the smallest adversarial perturbation

### Parameters

- **distance** (*Optional*[`foolbox.distances.Distance`]) –
- **steps** (*int*) –
- **target** (*float*) –

**class** `foolbox.attacks.L2CarliniWagnerAttack` (`binary_search_steps=9`, `steps=10000`, `stepsize=0.01`, `confidence=0`, `initial_const=0.001`, `abort_early=True`)

Implementation of the Carlini & Wagner L2 Attack.<sup>8</sup>

### Parameters

- **binary\_search\_steps** (*int*) – Number of steps to perform in the binary search over the const `c`.
- **steps** (*int*) – Number of optimization steps within each binary search step.
- **stepsize** (*float*) – Stepsize to update the examples.
- **confidence** (*float*) – Confidence required for an example to be marked as adversarial. Controls the gap between example and decision boundary.
- **initial\_const** (*float*) – Initial value of the const `c` with which the binary search starts.
- **abort\_early** (*bool*) – Stop inner search as soon as an adversarial example has been found. Does not affect the binary search over the const `c`.

---

<sup>8</sup> Nicholas Carlini, David Wagner, “Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy” <https://arxiv.org/abs/1608.04644>

## References

**class** foolbox.attacks.**NewtonFoolAttack** (*steps=100, stepsize=0.01*)  
Implementation of the NewtonFool Attack.<sup>9</sup>

### Parameters

- **steps** (*int*) – Number of update steps to perform.
- **step\_size** – Size of each update step.
- **stepsize** (*float*) –

## References

**class** foolbox.attacks.**EADAttack** (*binary\_search\_steps=9, steps=10000, initial\_stepsize=0.01, confidence=0.0, initial\_const=0.001, regularization=0.01, decision\_rule='EN', abort\_early=True*)  
Implementation of the EAD Attack with EN Decision Rule.<sup>10</sup>

### Parameters

- **binary\_search\_steps** (*int*) – Number of steps to perform in the binary search over the const c.
- **steps** (*int*) – Number of optimization steps within each binary search step.
- **initial\_stepsize** (*float*) – Initial stepsize to update the examples.
- **confidence** (*float*) – Confidence required for an example to be marked as adversarial. Controls the gap between example and decision boundary.
- **initial\_const** (*float*) – Initial value of the const c with which the binary search starts.
- **regularization** (*float*) – Controls the L1 regularization.
- **decision\_rule** (*Union[typing\_extensions.Literal['EN'], typing\_extensions.Literal['L1']]*) – Rule according to which the best adversarial examples are selected. They either minimize the L1 or ElasticNet distance.
- **abort\_early** (*bool*) – Stop inner search as soon as an adversarial example has been found. Does not affect the binary search over the const c.

## References

“EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples”, <https://www.aaii.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/16893>

**class** foolbox.attacks.**GaussianBlurAttack** (\*, *distance=None, steps=1000, channel\_axis=None, max\_sigma=None*)  
Blurs the inputs using a Gaussian filter with linearly increasing standard deviation.

### Parameters

- **steps** (*int*) – Number of sigma values tested between 0 and max\_sigma.
- **channel\_axis** (*Optional[int]*) – Index of the channel axis in the input data.

<sup>9</sup> Uyeong Jang et al., “Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning”, <https://dl.acm.org/citation.cfm?id=3134635>

<sup>10</sup> Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh,

- **max\_sigma** (*Optional[float]*) – Maximally allowed sigma value of the Gaussian blur.
- **distance** (*Optional[foolbox.distances.Distance]*) –

```
class foolbox.attacks.L2DeepFoolAttack(*, steps=50, candidates=10, overshoot=0.02,
                                     loss='logits')
```

A simple and fast gradient-based adversarial attack.

Implements the DeepFool L2 attack.<sup>11</sup>

#### Parameters

- **steps** (*int*) – Maximum number of steps to perform.
- **candidates** (*Optional[int]*) – Limit on the number of the most likely classes that should be considered. A small value is usually sufficient and much faster.
- **overshoot** (*float*) – How much to overshoot the boundary.
- **Loss function to use inside the update function.** (*loss*) –
- **typing\_extensions.Literal['crossentropy']] loss** (*Union[typing\_extensions.Literal['logits'],]*) –

#### References

```
class foolbox.attacks.LinfDeepFoolAttack(*, steps=50, candidates=10, overshoot=0.02,
                                       loss='logits')
```

A simple and fast gradient-based adversarial attack.

Implements the DeepFool L-Infinity attack.

#### Parameters

- **steps** (*int*) – Maximum number of steps to perform.
- **candidates** (*Optional[int]*) – Limit on the number of the most likely classes that should be considered. A small value is usually sufficient and much faster.
- **overshoot** (*float*) – How much to overshoot the boundary.
- **Loss function to use inside the update function.** (*loss*) –
- **typing\_extensions.Literal['crossentropy']] loss** (*Union[typing\_extensions.Literal['logits'],]*) –

```
class foolbox.attacks.SaltAndPepperNoiseAttack(steps=1000, across_channels=True,
                                              channel_axis=None)
```

Increases the amount of salt and pepper noise until the input is misclassified.

#### Parameters

- **steps** (*int*) – The number of steps to run.
- **across\_channels** (*bool*) – Whether the noise should be the same across all channels.
- **channel\_axis** (*Optional[int]*) – The axis across which the noise should be the same (if `across_channels` is `True`). If `None`, will be automatically inferred from the model if possible.

---

<sup>11</sup> Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard, “DeepFool: a simple and accurate method to fool deep neural networks”, <https://arxiv.org/abs/1511.04599>



```
class foolbox.attacks.LinearSearchBlendedUniformNoiseAttack (*, distance=None,
                                                         directions=1000,
                                                         steps=1000)
```

Blends the input with a uniform noise input until it is misclassified.

#### Parameters

- **distance** (*Optional*[foolbox.distances.Distance]) – Distance measure for which minimal adversarial examples are searched.
- **directions** (*int*) – Number of random directions in which the perturbation is searched.
- **steps** (*int*) – Number of blending steps between the original image and the random directions.

```
class foolbox.attacks.BinarizationRefinementAttack (*, distance=None, thresh-
                                                         old=None, included_in='upper')
```

For models that preprocess their inputs by binarizing the inputs, this attack can improve adversarials found by other attacks. It does this by utilizing information about the binarization and mapping values to the corresponding value in the clean input or to the right side of the threshold.

#### Parameters

- **threshold** (*Optional*[float]) – The threshold used by the models binarization. If none, defaults to  $(\text{model.bounds}()[1] - \text{model.bounds}()[0]) / 2$ .
- **included\_in** (*Union*[*typing\_extensions.Literal*['lower'], *typing\_extensions.Literal*['upper']]) – Whether the threshold value itself belongs to the lower or upper interval.
- **distance** (*Optional*[foolbox.distances.Distance]) –

```
class foolbox.attacks.DatasetAttack (*, distance=None)
```

Draws randomly from the given dataset until adversarial examples for all inputs have been found.

To pass data from the dataset to this attack, call `feed()`. `feed()` can be called several times and should only be called with batches that are small enough that they can be passed through the model.

**Parameters** **distance** (*Optional*[foolbox.distances.Distance]) – Distance measure for which minimal adversarial examples are searched.

```
class foolbox.attacks.BoundaryAttack (init_attack=None, steps=25000, spherical_step=0.01,
                                     source_step=0.01, source_step_convergence=1e-07,
                                     step_adaptation=1.5, tensorboard=False, update_stats_every_k=10)
```

A powerful adversarial attack that requires neither gradients nor probabilities.

This is the reference implementation for the attack.<sup>12</sup>

<sup>12</sup> Wieland Brendel (\*), Jonas Rauber (\*), Matthias Bethge, “Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models”, <https://arxiv.org/abs/1712.04248>

## Notes

Differences to the original reference implementation: \* We do not perform internal operations with float64 \* The samples within a batch can currently influence each other a bit \* We don't perform the additional convergence confirmation \* The success rate tracking changed a bit \* Some other changes due to batching and merged loops

### Parameters

- **init\_attack** *(Optional[foolbox.attacks.base.MinimizationAttack])* – Attack to use to find a starting points. Defaults to LinearSearchBlendedUniformNoiseAttack. Only used if starting\_points is None.
- **steps** (*int*) – Maximum number of steps to run. Might converge and stop before that.
- **spherical\_step** (*float*) – Initial step size for the orthogonal (spherical) step.
- **source\_step** (*float*) – Initial step size for the step towards the target.
- **source\_step\_convergence** (*float*) – Sets the threshold of the stop criterion: if source\_step becomes smaller than this value during the attack, the attack has converged and will stop.
- **step\_adaptation** (*float*) – Factor by which the step sizes are multiplied or divided.
- **tensorboard** *(Union[typing\_extensions.Literal[False], None, str])* – The log directory for TensorBoard summaries. If False, TensorBoard summaries will be disabled (default). If None, the logdir will be runs/CURRENT\_DATETIME\_HOSTNAME.
- **update\_stats\_every\_k** (*int*) –

## References

```
class foolbox.attacks.L0BrendelBethgeAttack (init_attack=None, overshoot=1.1,
                                             steps=1000, lr=0.001, lr_decay=0.5,
                                             lr_num_decay=20, momentum=0.8, tensorboard=False, binary_search_steps=10)
```

L0 variant of the Brendel & Bethge adversarial attack. [\[#Bren19\]](#) This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

## References

### Parameters

- **init\_attack** *(Optional[foolbox.attacks.base.MinimizationAttack])* –
- **overshoot** (*float*) –
- **steps** (*int*) –
- **lr** (*float*) –
- **lr\_decay** (*float*) –
- **lr\_num\_decay** (*int*) –
- **momentum** (*float*) –

- **None, str] tensorboard** *(Union[typing\_extensions.Literal[False],])*–
- **binary\_search\_steps** (*int*)–

```
class foolbox.attacks.L1BrendelBethgeAttack (init_attack=None, overshoot=1.1,
steps=1000, lr=0.001, lr_decay=0.5,
lr_num_decay=20, momentum=0.8, tensorboard=False, binary_search_steps=10)
```

L1 variant of the Brendel & Bethge adversarial attack. [\[#Bren19\]](#) This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

## References

### Parameters

- **init\_attack** *(Optional[foolbox.attacks.base.MinimizationAttack])*–
- **overshoot** (*float*)–
- **steps** (*int*)–
- **lr** (*float*)–
- **lr\_decay** (*float*)–
- **lr\_num\_decay** (*int*)–
- **momentum** (*float*)–
- **None, str] tensorboard** *(Union[typing\_extensions.Literal[False],])*–
- **binary\_search\_steps** (*int*)–

```
class foolbox.attacks.L2BrendelBethgeAttack (init_attack=None, overshoot=1.1,
steps=1000, lr=0.001, lr_decay=0.5,
lr_num_decay=20, momentum=0.8, tensorboard=False, binary_search_steps=10)
```

L2 variant of the Brendel & Bethge adversarial attack. [\[#Bren19\]](#) This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

## References

### Parameters

- **init\_attack** *(Optional[foolbox.attacks.base.MinimizationAttack])* -
- **overshoot** *(float)* -
- **steps** *(int)* -
- **lr** *(float)* -
- **lr\_decay** *(float)* -
- **lr\_num\_decay** *(int)* -
- **momentum** *(float)* -
- **None, str] tensorboard** *(Union[typing\_extensions.Literal[False],])* -
- **binary\_search\_steps** *(int)* -

```
class foolbox.attacks.LinfinityBrendelBethgeAttack (init_attack=None, overshoot=1.1, steps=1000, lr=0.001, lr_decay=0.5, lr_num_decay=20, momentum=0.8, tensorboard=False, binary_search_steps=10)
```

L-infinity variant of the Brendel & Bethge adversarial attack. [\[#Bren19\]](#) This is a powerful gradient-based adversarial attack that follows the adversarial boundary (the boundary between the space of adversarial and non-adversarial images as defined by the adversarial criterion) to find the minimum distance to the clean image.

This is the reference implementation of the Brendel & Bethge attack.

## References

### Parameters

- **init\_attack** *(Optional[foolbox.attacks.base.MinimizationAttack])* -
- **overshoot** *(float)* -
- **steps** *(int)* -
- **lr** *(float)* -
- **lr\_decay** *(float)* -
- **lr\_num\_decay** *(int)* -
- **momentum** *(float)* -
- **None, str] tensorboard** *(Union[typing\_extensions.Literal[False],])* -
- **binary\_search\_steps** *(int)* -

```
foolbox.attacks.FGM
```

```
alias of foolbox.attacks.fast_gradient_method.L2FastGradientAttack
```

foolbox.attacks.**FGSM**

alias of foolbox.attacks.fast\_gradient\_method.LinfFastGradientAttack

foolbox.attacks.**L2PGD**

alias of foolbox.attacks.projected\_gradient\_descent.L2ProjectedGradientDescentAttack

foolbox.attacks.**LinfPGD**

alias of foolbox.attacks.projected\_gradient\_descent.LinfProjectedGradientDescentAttack

foolbox.attacks.**PGD**

alias of foolbox.attacks.projected\_gradient\_descent.LinfProjectedGradientDescentAttack



## FOOLBOX.CRITERIA

Criteria are used to define which inputs are adversarial. We provide common criteria for untargeted and targeted adversarial attacks, e.g. *Misclassification* and *TargetedMisclassification*. New criteria can easily be implemented by subclassing *Criterion* and implementing *Criterion.\_\_call\_\_()*.

Criteria can be combined using a logical and `criterion1 & criterion2` to create a new criterion.

### 3.1 Misclassification

```
from foolbox.criteria import Misclassification
criterion = Misclassification(labels)
```

**class** foolbox.criteria.**Misclassification** (*labels*)

Considers those perturbed inputs adversarial whose predicted class differs from the label.

**Parameters** **labels** (*Any*) – Tensor with labels of the unperturbed inputs (*batch, ...*).

### 3.2 TargetedMisclassification

```
from foolbox.criteria import TargetedMisclassification
criterion = TargetedMisclassification(target_classes)
```

**class** foolbox.criteria.**TargetedMisclassification** (*target\_classes*)

Considers those perturbed inputs adversarial whose predicted class matches the target class.

**Parameters** **target\_classes** (*Any*) – Tensor with target classes (*batch, ...*).

### 3.3 Criterion

**class** foolbox.criteria.**Criterion**

Abstract base class to implement new criteria.

**abstract** `__call__` (*perturbed, outputs*)

Returns a boolean tensor indicating which perturbed inputs are adversarial.

**Parameters**

- **perturbed** (*T*) – Tensor with perturbed inputs (*batch, ...*).
- **outputs** (*T*) – Tensor with model outputs for the perturbed inputs (*batch, ...*).

**Returns** A boolean tensor indicating which perturbed inputs are adversarial (`batch,` ).

**Return type** T



## 4.1 Detailed description

**class** foolbox.distances.Distance

**class** foolbox.distances.LpDistance(*p*)

**Parameters** *p* (*float*) –

**clip\_perturbation** (*references*, *perturbed*, *epsilon*)

Clips the perturbations to epsilon and returns the new perturbed

**Parameters**

- **references** (*T*) – A batch of reference inputs.
- **perturbed** (*T*) – A batch of perturbed inputs.
- **epsilon** (*float*) –

**Returns** A tensor like perturbed but with the perturbation clipped to epsilon.

**Return type** *T*



FOOLBOX . UTILS

`foolbox.utils.accuracy` (*fmodel, inputs, labels*)

**Parameters**

- **fmodel** (*foolbox.models.base.Model*) –
- **inputs** (*Any*) –
- **labels** (*Any*) –

**Return type** float

`foolbox.utils.samples` (*fmodel, dataset='imagenet', index=0, batchsize=1, shape=(224, 224), data\_format=None, bounds=None*)

**Parameters**

- **fmodel** (*foolbox.models.base.Model*) –
- **dataset** (*str*) –
- **index** (*int*) –
- **batchsize** (*int*) –
- **int] shape** (*Tuple[int,)* –
- **data\_format** (*Optional[str]*) –
- **bounds** (*Optional[foolbox.types.Bounds]*) –

**Return type** Any



**FOOLBOX.PLOT**

`foolbox.plot.images` (*images*, \*, *n=None*, *data\_format=None*, *bounds=(0, 1)*, *ncols=None*,  
*nrows=None*, *figsize=None*, *scale=1*, \*\**kwargs*)

**Parameters**

- **images** (*Any*) –
- **n** (*Optional[int]*) –
- **data\_format** (*Optional[str]*) –
- **float] bounds** (*Tuple[float,)*) –
- **ncols** (*Optional[int]*) –
- **nrows** (*Optional[int]*) –
- **float]] figsize** (*Optional[Tuple[float,)*) –
- **scale** (*float*) –
- **kwargs** (*Any*) –

**Return type** `None`



## 7.1 Get Model

`foolbox.zoo.get_model` (*url*, *module\_name*='foolbox\_model', *overwrite*=False, *\*\*kwargs*)  
Download a Foolbox-compatible model from the given Git repository URL.

### Examples

Instantiate a model:

```
>>> from foolbox import zoo
>>> url = "https://github.com/bveliqi/foolbox-zoo-dummy.git"
>>> model = zoo.get_model(url)
```

Only works with a foolbox-zoo compatible repository. I.e. models need to have a `foolbox_model.py` file with a `create()`-function, which returns a foolbox-wrapped model.

Using the `kwargs` parameter it is possible to input an arbitrary number of parameters to this methods call. These parameters are forwarded to the instantiated model.

Example repositories:

- <https://github.com/jonasrauber/foolbox-tensorflow-keras-applications>
- <https://github.com/bethgelab/AnalysisBySynthesis>
- [https://github.com/bethgelab/mnist\\_challenge](https://github.com/bethgelab/mnist_challenge)
- [https://github.com/bethgelab/cifar10\\_challenge](https://github.com/bethgelab/cifar10_challenge)
- [https://github.com/bethgelab/convex\\_adversarial](https://github.com/bethgelab/convex_adversarial)
- <https://github.com/wielandbrendel/logit-pairing-foolbox.git>
- <https://github.com/bethgelab/defensive-distillation.git>

### Parameters

- **url** (*str*) – URL to the git repository.
- **module\_name** (*str*) – The name of the module to import.
- **kwargs** (*Any*) – Optional set of parameters that will be used by the to be instantiated model.
- **overwrite** (*bool*) –

**Returns** A Foolbox-wrapped model instance.

**Return type** foolbox.models.base.Model

## 7.2 Fetch Weights

foolbox.zoo.**fetch\_weights** (*weights\_uri*, *unzip=False*)

Provides utilities to download and extract packages containing model weights when creating foolbox-zoo compatible repositories, if the weights are not part of the repository itself.

### Examples

Download and unzip weights:

```
>>> from foolbox import zoo
>>> url = 'https://github.com/MadryLab/mnist_challenge_models/raw/master/secret.
↳zip' # noqa F501
>>> weights_path = zoo.fetch_weights(url, unzip=True)
```

### Parameters

- **weights\_uri** (*str*) – The URI to fetch the weights from.
- **unzip** (*bool*) – Should be *True* if the file to be downloaded is a zipped package.

**Returns** Local path where the weights have been downloaded and potentially unzipped to.

**Return type** str



FOOLBOX . DEVUTILS

Internal module with utility functions

`foolbox.devutils.atleast_kd(x, k)`

**Parameters**

- **x** (*eagerpy.Tensor*) –
- **k** (*int*) –

**Return type** *eagerpy.Tensor*

`foolbox.devutils.flatten(x, keep=1)`

**Parameters**

- **x** (*eagerpy.Tensor*) –
- **keep** (*int*) –

**Return type** *eagerpy.Tensor*



## FOOLBOX.TENSORBOARD

Internal module for attacks that support logging to TensorBoard

**class** `foolbox.tensorboard.TensorBoard` (*logdir*)

A custom TensorBoard class that accepts EagerPy tensors and that can be disabled by turned into a noop by passing `logdir=False`.

This makes it possible to add tensorboard logging without any if statements and without any computational overhead if it's disabled.

**Parameters** `None, str` **logdir** (*Union[typing\_extensions.Literal[False],*)

-

**close** ()

**Return type** `None`

**conditional\_mean** (*tag, x, cond, step*)

**Parameters**

- **tag** (*str*) -
- **x** (*eagerpy.Tensor*) -
- **cond** (*eagerpy.Tensor*) -
- **step** (*int*) -

**Return type** `None`

**histogram** (*tag, x, step, \*, first=True*)

**Parameters**

- **tag** (*str*) -
- **x** (*eagerpy.Tensor*) -
- **step** (*int*) -
- **first** (*bool*) -

**Return type** `None`

**mean** (*tag, x, step*)

**Parameters**

- **tag** (*str*) -
- **x** (*eagerpy.Tensor*) -
- **step** (*int*) -

**Return type** None

**probability** (*tag*, *x*, *step*)

**Parameters**

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **step** (*int*) –

**Return type** None

**probability\_ratio** (*tag*, *x*, *y*, *step*)

**Parameters**

- **tag** (*str*) –
- **x** (*eagerpy.Tensor*) –
- **y** (*eagerpy.Tensor*) –
- **step** (*int*) –

**Return type** None

**scalar** (*tag*, *x*, *step*)

**Parameters**

- **tag** (*str*) –
- **float** **x** (*Union[int,)* –
- **step** (*int*) –

**Return type** None

`foolbox.tensorboard.maybenoop` (*f*)

**Parameters** **f** (*F*) –

**Return type** *F*

**FOOLBOX . TYPES**

**class** foolbox.types.**Bounds** (*lower, upper*)

**Parameters**

- **lower** (*float*) –
- **upper** (*float*) –

**property lower**

Alias for field number 0

**property upper**

Alias for field number 1



## INDICES AND TABLES

- genindex
- modindex
- search





## PYTHON MODULE INDEX

### f

`foolbox.attacks`, 5  
`foolbox.criteria`, 17  
`foolbox.devutils`, 29  
`foolbox.distances`, 21  
`foolbox.models`, 3  
`foolbox.plot`, 25  
`foolbox.tensorboard`, 31  
`foolbox.types`, 33  
`foolbox.utils`, 23  
`foolbox.zoo`, 27



## Symbols

`__call__()` (*foolbox.criteria.Criterion method*), 19

## A

`accuracy()` (*in module foolbox.utils*), 23

`atleast_kd()` (*in module foolbox.devutils*), 29

## B

`BinarizationRefinementAttack` (*class in foolbox.attacks*), 13

`BinarySearchContrastReductionAttack` (*class in foolbox.attacks*), 10

`BoundaryAttack` (*class in foolbox.attacks*), 13

`Bounds` (*class in foolbox.types*), 33

## C

`clip_perturbation()` (*foolbox.distances.LpDistance method*), 21

`close()` (*foolbox.tensorboard.TensorBoard method*), 31

`conditional_mean()` (*foolbox.tensorboard.TensorBoard method*), 31

`Criterion` (*class in foolbox.criteria*), 19

## D

`DatasetAttack` (*class in foolbox.attacks*), 13

`DDNAttack` (*class in foolbox.attacks*), 7

`Distance` (*class in foolbox.distances*), 21

## E

`EADAttack` (*class in foolbox.attacks*), 11

## F

`fetch_weights()` (*in module foolbox.zoo*), 28

`FGM` (*in module foolbox.attacks*), 16

`FGSM` (*in module foolbox.attacks*), 16

`flatten()` (*in module foolbox.devutils*), 29

`foolbox.attacks` (*module*), 5

`foolbox.criteria` (*module*), 17

`foolbox.devutils` (*module*), 29

`foolbox.distances` (*module*), 21

`foolbox.models` (*module*), 3

`foolbox.plot` (*module*), 25

`foolbox.tensorboard` (*module*), 31

`foolbox.types` (*module*), 33

`foolbox.utils` (*module*), 23

`foolbox.zoo` (*module*), 27

## G

`GaussianBlurAttack` (*class in foolbox.attacks*), 11

`get_model()` (*in module foolbox.zoo*), 27

## H

`histogram()` (*foolbox.tensorboard.TensorBoard method*), 31

## I

`images()` (*in module foolbox.plot*), 25

`InversionAttack` (*class in foolbox.attacks*), 9

## J

`JAXModel` (*class in foolbox.models*), 4

## L

`L0BrendelBethgeAttack` (*class in foolbox.attacks*), 14

`L1BrendelBethgeAttack` (*class in foolbox.attacks*), 15

`L2AdditiveGaussianNoiseAttack` (*class in foolbox.attacks*), 8

`L2AdditiveUniformNoiseAttack` (*class in foolbox.attacks*), 8

`L2BasicIterativeAttack` (*class in foolbox.attacks*), 7

`L2BrendelBethgeAttack` (*class in foolbox.attacks*), 15

`L2CarliniWagnerAttack` (*class in foolbox.attacks*), 10

`L2ClippingAwareAdditiveGaussianNoiseAttack` (*class in foolbox.attacks*), 8

`L2ClippingAwareAdditiveUniformNoiseAttack` (*class in foolbox.attacks*), 8

- L2ClippingAwareRepeatedAdditiveGaussianNoiseAttack (class in foolbox.attacks), 9
- L2ClippingAwareRepeatedAdditiveUniformNoiseAttack (class in foolbox.attacks), 9
- L2ContrastReductionAttack (class in foolbox.attacks), 6
- L2DeepFoolAttack (class in foolbox.attacks), 12
- L2FastGradientAttack (class in foolbox.attacks), 8
- L2PGD (in module foolbox.attacks), 17
- L2ProjectedGradientDescentAttack (class in foolbox.attacks), 7
- L2RepeatedAdditiveGaussianNoiseAttack (class in foolbox.attacks), 8
- L2RepeatedAdditiveUniformNoiseAttack (class in foolbox.attacks), 8
- LinearSearchBlendedUniformNoiseAttack (class in foolbox.attacks), 12
- LinearSearchContrastReductionAttack (class in foolbox.attacks), 10
- LinfAdditiveUniformNoiseAttack (class in foolbox.attacks), 8
- LinfBasicIterativeAttack (class in foolbox.attacks), 8
- LinfDeepFoolAttack (class in foolbox.attacks), 12
- LinfFastGradientAttack (class in foolbox.attacks), 8
- LinfinityBrendelBethgeAttack (class in foolbox.attacks), 16
- LinfPGD (in module foolbox.attacks), 17
- LinfProjectedGradientDescentAttack (class in foolbox.attacks), 7
- LinfRepeatedAdditiveUniformNoiseAttack (class in foolbox.attacks), 9
- lower() (foolbox.types.Bounds property), 33
- LpDistance (class in foolbox.distances), 21
- ## M
- maybenoop() (in module foolbox.tensorboard), 32
- mean() (foolbox.tensorboard.TensorBoard method), 31
- Misclassification (class in foolbox.criteria), 19
- Model (class in foolbox.models), 3
- ## N
- NewtonFoolAttack (class in foolbox.attacks), 11
- NumPyModel (class in foolbox.models), 4
- ## P
- PGD (in module foolbox.attacks), 17
- probability() (foolbox.tensorboard.TensorBoard method), 32
- probability\_ratio() (foolbox.tensorboard.TensorBoard method), 32
- PyTorchModel (class in foolbox.models), 3
- ## S
- SaltAndPepperNoiseAttack (class in foolbox.attacks), 12
- samples() (in module foolbox.utils), 23
- scalar() (foolbox.tensorboard.TensorBoard method), 32
- ## T
- TargetedMisclassification (class in foolbox.criteria), 19
- TensorBoard (class in foolbox.tensorboard), 31
- TensorFlowModel (class in foolbox.models), 3
- transform\_bounds() (foolbox.models.Model method), 3
- transform\_bounds() (foolbox.models.TransformBoundsWrapper method), 4
- TransformBoundsWrapper (class in foolbox.models), 4
- ## U
- upper() (foolbox.types.Bounds property), 33
- ## V
- VirtualAdversarialAttack (class in foolbox.attacks), 6