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

- 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
Foolbox, Release 3.3.1

- 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

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

Continued on next page
Table 1 – continued from previous page

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

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. 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 method.

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

---

References

`class foolbox.attacks.DDNAttack(*, init_epsilon=1.0, steps=10, gamma=0.05)`

The Decoupled Direction and Norm L2 adversarial attack. ²

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: new_norm = norm * (1 + or - 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)` – Stepsize 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=0.03333333333333333, abs_stepsize=None, steps=40, random_start=True)`

Linf Projected Gradient Descent

Parameters

- `rel_stepsize (float)` – Stepsize 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)` – Stepsize 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.

```python
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)* – Stepsize 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.

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

Fast Gradient Method (FGM)

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

```python
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.

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

Samples Gaussian noise with a fixed L2 size.

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

Samples uniform noise with a fixed L2 size.

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

Samples Gaussian noise with a fixed L2 size after clipping.

The implementation is based on [#Rauber20].

**References**

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

Samples uniform noise with a fixed L2 size after clipping.

The implementation is based on [#Rauber20].

**References**

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

Samples uniform noise with a fixed L-infinity size

```python
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(*, repeats=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(*, repeats=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

---

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.

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.

---

https://arxiv.org/abs/1608.04644
References

class foolbox.attacks.NewtonFoolAttack(steps=100, stepsize=0.01)
Implementation of the NewtonFool Attack.9

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.10

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


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.

10 Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh,
Foolbox, Release 3.3.1

- `max_sigma (Optional[float])` – Maximally allowed sigma value of the Gaussian blur.
- `distance (Optional[foolbox.distances.Distance])` –

```python
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.\(^{11}\)

**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**

```python
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'],]) –

```python
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.

---

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, threshold=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 form 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.\(^\text{12}\)

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

```python
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, tensor-
  board=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, tensor-
  board=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
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

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

```python
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

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

```python
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

```python
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 

**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 tenosr like perturbed but with the perturbation clipped to epsilon.

Return type T
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.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,])**
- **nrows (Optional[int])**
- **ncols (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:

```python
>>> 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/cifar10_challenge
- 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.
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:

```python
>>> 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`
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*
Internal module for attacks that support logging to TensorBoard

```python
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)`
- `x (float)`
- `step (int)`

Return type None

`foolbox.tensorboard.maybenoop(f)`

Parameters `f (F)`

Return type `F`
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
CHAPTER ELEVEN

INDICES AND TABLES

• genindex
• modindex
• search
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
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