
torchattacks Documentation

Release v3.0.0

harrykim

Apr 18, 2022

Contents:

1	Attack	1
2	Attacks	3
2.1	VANILA	3
2.2	GN	3
2.3	FGSM	4
2.4	BIM	4
2.5	CW	5
2.6	R+FGSM	6
2.7	PGD	6
2.8	PGDL2	7
2.9	EOTPGD (EOT + PGD)	8
2.10	TPGD (TRADES' PGD)	8
2.11	FFGSM (Fast's FGSM)	9
2.12	MIFGSM	10
2.13	APGD	10
2.14	APGDT	11
2.15	FAB	12
2.16	Square	13
2.17	AutoAttack	14
2.18	OnePixel	15
2.19	DeepFool	15
2.20	SparseFool	16
2.21	DIFGSM	17
2.22	UPGD	17
2.23	TIFGSM	18
2.24	Jitter	19
2.25	Pixle	20
	Python Module Index	23
	Index	25

Modules

class `torchattacks.attack.Attack` (*name, model*)

Base class for all attacks.

Note: It automatically set device to the device where given model is. It basically changes training mode to eval during attack process. To change this, please see `set_training_mode`.

forward (**input*)

It defines the computation performed at every call. Should be overridden by all subclasses.

get_mode ()

Get attack mode.

save (*data_loader, save_path=None, verbose=True, return_verbose=False, save_pred=False*)

Save adversarial images as torch.tensor from given torch.utils.data.DataLoader.

Parameters

- **save_path** (*str*) – save_path.
- **data_loader** (*torch.utils.data.DataLoader*) – data loader.
- **verbose** (*bool*) – True for displaying detailed information. (Default: True)
- **return_verbose** (*bool*) – True for returning detailed information. (Default: False)
- **save_pred** (*bool*) – True for saving predicted labels (Default: False)

set_mode_default ()

Set attack mode as default mode.

set_mode_targeted_by_function (*target_map_function=None*)

Set attack mode as targeted.

Parameters `target_map_function` (*function*) – Label mapping function. e.g. `lambda images, labels:(labels+1)%10`. None for using input labels as targeted labels. (Default)

set_mode_targeted_least_likely (*kth_min=1*)

Set attack mode as targeted with least likely labels. :param `kth_min`: label with the k-th smallest probability used as target labels. (Default: 1) :type `kth_min`: str

set_mode_targeted_random ()

Set attack mode as targeted with random labels. :param `num_classes`: number of classes. :type `num_classes`: str

set_return_type (*type*)

Set the return type of adversarial images: *int* or *float*.

Parameters `type` (*str*) – ‘float’ or ‘int’. (Default: ‘float’)

Note: If ‘int’ is used for the return type, the file size of adversarial images can be reduced (about 1/4 for CIFAR10). However, if the attack originally outputs float adversarial images (e.g. using small step-size than 1/255), it might reduce the attack success rate of the attack.

set_training_mode (*model_training=False, batchnorm_training=False, dropout_training=False*)

Set training mode during attack process.

Parameters

- **model_training** (*bool*) – True for using training mode for the entire model during attack process.
- **batchnorm_training** (*bool*) – True for using training mode for batchnorms during attack process.
- **dropout_training** (*bool*) – True for using training mode for dropouts during attack process.

Note: For RNN-based models, we cannot calculate gradients with eval mode. Thus, it should be changed to the training mode during the attack.

2.1 VANILA

class `torchattacks.attacks.vanila.VANILA` (*model*)
Vanila version of Attack. It just returns the input images.

Parameters `model` (*nn.Module*) – model to attack.

Shape:

- `images`: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range $[0, 1]$.
- `labels`: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- `output`: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.VANILA(model)
>>> adv_images = attack(images, labels)
```

forward (*images, labels=None*)
Overridden.

2.2 GN

class `torchattacks.attacks.gn.GN` (*model, std=0.1*)
Add Gaussian Noise.

Parameters

- `model` (*nn.Module*) – model to attack.

- **std** (*nn.Module*) – standard deviation (Default: 0.1).

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range [0, 1].
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.GN(model)
>>> adv_images = attack(images, labels)
```

forward (*images, labels=None*)
Overridden.

2.3 FGSM

class torchattacks.attacks.fgsm.**FGSM** (*model, eps=0.007*)
FGSM in the paper ‘Explaining and harnessing adversarial examples’ [<https://arxiv.org/abs/1412.6572>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 0.007)

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range [0, 1].
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.FGSM(model, eps=0.007)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.4 BIM

class torchattacks.attacks.bim.**BIM** (*model, eps=0.01568627450980392, al-pha=0.00392156862745098, steps=0*)
BIM or iterative-FGSM in the paper ‘Adversarial Examples in the Physical World’ [<https://arxiv.org/abs/1607.02533>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 4/255)
- **alpha** (*float*) – step size. (Default: 1/255)
- **steps** (*int*) – number of steps. (Default: 0)

Note: If steps set to 0, steps will be automatically decided following the paper.

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.BIM(model, eps=4/255, alpha=1/255, steps=0)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)

Overridden.

2.5 CW

class torchattacks.attacks.cw.**CW** (*model, c=0.0001, kappa=0, steps=1000, lr=0.01*)

CW in the paper ‘Towards Evaluating the Robustness of Neural Networks’ [<https://arxiv.org/abs/1608.04644>]

Distance Measure : L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **c** (*float*) – c in the paper. parameter for box-constraint. (Default: 1e-4)
 $\text{minimize} \|\frac{1}{2}(\tanh(w) + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh(w) + 1))$
- **kappa** (*float*) – kappa (also written as ‘confidence’) in the paper. (Default: 0) $f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$
- **steps** (*int*) – number of steps. (Default: 1000)
- **lr** (*float*) – learning rate of the Adam optimizer. (Default: 0.01)

Warning: With default c , you can’t easily get adversarial images. Set higher c like 1.

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.

- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.CW(model, c=1e-4, kappa=0, steps=1000, lr=0.01)
>>> adv_images = attack(images, labels)
```

Note: Binary search for c is NOT IMPLEMENTED methods in the paper due to time consuming.

forward (*images, labels*)
Overridden.

2.6 R+FGSM

class torchattacks.attacks.rfgsm.**RFGSM** (*model*, *eps=0.06274509803921569*, *alpha=0.03137254901960784*, *steps=1*)

R+FGSM in the paper ‘Ensemble Adversarial Training : Attacks and Defences’ [<https://arxiv.org/abs/1705.07204>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – strength of the attack or maximum perturbation. (Default: 16/255)
- **alpha** (*float*) – step size. (Default: 8/255)
- **steps** (*int*) – number of steps. (Default: 1)

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.RFGSM(model, eps=16/255, alpha=8/255, steps=1)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.7 PGD

class torchattacks.attacks.pgd.**PGD** (*model*, *eps=0.3*, *alpha=0.00784313725490196*, *steps=40*, *random_start=True*)

PGD in the paper ‘Towards Deep Learning Models Resistant to Adversarial Attacks’ [<https://arxiv.org/abs/1706.06083>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 0.3)
- **alpha** (*float*) – step size. (Default: 2/255)
- **steps** (*int*) – number of steps. (Default: 40)
- **random_start** (*bool*) – using random initialization of delta. (Default: True)

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range [0, 1].
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.PGD(model, eps=8/255, alpha=1/255, steps=40, random_
↳start=True)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.8 PGDL2

class torchattacks.attacks.pgdl2.**PGDL2** (*model, eps=1.0, alpha=0.2, steps=40, random_start=True, eps_for_division=1e-10*)
PGD in the paper ‘Towards Deep Learning Models Resistant to Adversarial Attacks’ [<https://arxiv.org/abs/1706.06083>]

Distance Measure : L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 1.0)
- **alpha** (*float*) – step size. (Default: 0.2)
- **steps** (*int*) – number of steps. (Default: 40)
- **random_start** (*bool*) – using random initialization of delta. (Default: True)

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range [0, 1].
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.PGDL2(model, eps=1.0, alpha=0.2, steps=40, random_
↳start=True)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.9 EOTPGD (EOT + PGD)

class torchattacks.attacks.eotpgd.**EOTPGD** (*model, eps=0.3, alpha=0.00784313725490196, steps=40, eot_iter=10, random_start=True*)

Comment on “Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network” [<https://arxiv.org/abs/1907.00895>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 0.3)
- **alpha** (*float*) – step size. (Default: 2/255)
- **steps** (*int*) – number of steps. (Default: 40)
- **eot_iter** (*int*) – number of models to estimate the mean gradient. (Default: 10)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.EOTPGD(model, eps=4/255, alpha=8/255, steps=40, eot_
↳iter=10)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.10 TPGD (TRADES’ PGD)

class torchattacks.attacks.tpgd.**TPGD** (*model, eps=0.03137254901960784, alpha=0.00784313725490196, steps=7*)

PGD based on KL-Divergence loss in the paper ‘Theoretically Principled Trade-off between Robustness and Accuracy’ [<https://arxiv.org/abs/1901.08573>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – strength of the attack or maximum perturbation. (Default: 8/255)
- **alpha** (*float*) – step size. (Default: 2/255)
- **steps** (*int*) – number of steps. (Default: 7)

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range [0, 1].
- output: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.TPGD(model, eps=8/255, alpha=2/255, steps=7)
>>> adv_images = attack(images)
```

forward (*images, labels=None*)
Overridden.

2.11 FFGSM (Fast's FGSM)

class torchattacks.attacks.ffgsm.**FFGSM** (*model*, *eps=0.03137254901960784*, *al-*
pha=0.0392156862745098)

New FGSM proposed in ‘Fast is better than free: Revisiting adversarial training’ [<https://arxiv.org/abs/2001.03994>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 8/255)
- **alpha** (*float*) – step size. (Default: 10/255)

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range [0, 1].
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W).

Examples::

```
>>> attack = torchattacks.FFGSM(model, eps=8/255, alpha=10/255)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.12 MIFGSM

class torchattacks.attacks.mifgsm.**MIFGSM**(*model*, *eps*=0.03137254901960784, *alpha*=0.00784313725490196, *steps*=5, *decay*=1.0)

MI-FGSM in the paper ‘Boosting Adversarial Attacks with Momentum’ [<https://arxiv.org/abs/1710.06081>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 8/255)
- **alpha** (*float*) – step size. (Default: 2/255)
- **decay** (*float*) – momentum factor. (Default: 1.0)
- **steps** (*int*) – number of iterations. (Default: 5)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.MIFGSM(model, eps=8/255, steps=5, decay=1.0)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.13 APGD

class torchattacks.attacks.apgd.**APGD**(*model*, *norm*='Linf', *eps*=0.03137254901960784, *steps*=100, *n_restarts*=1, *seed*=0, *loss*='ce', *eot_iter*=1, *rho*=0.75, *verbose*=False)

APGD in the paper ‘Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks’ [<https://arxiv.org/abs/2003.01690>] [<https://github.com/fra31/auto-attack>]

Distance Measure : Linf, L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **norm** (*str*) – Lp-norm of the attack. [‘Linf’, ‘L2’] (Default: ‘Linf’)
- **eps** (*float*) – maximum perturbation. (Default: None)
- **steps** (*int*) – number of steps. (Default: 100)
- **n_restarts** (*int*) – number of random restarts. (Default: 1)
- **seed** (*int*) – random seed for the starting point. (Default: 0)

- **loss** (*str*) – loss function optimized. ['ce', 'dlr'] (Default: 'ce')
- **eot_iter** (*int*) – number of iteration for EOT. (Default: 1)
- **rho** (*float*) – parameter for step-size update (Default: 0.75)
- **verbose** (*bool*) – print progress. (Default: False)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.APGD(model, norm='Linf', eps=8/255, steps=100, n_
↳ restarts=1, seed=0, loss='ce', eot_iter=1, rho=.75, verbose=False)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.14 APGDT

class torchattacks.attacks.apgdt.**APGDT** (*model, norm='Linf', eps=0.03137254901960784, steps=100, n_restarts=1, seed=0, eot_iter=1, rho=0.75, verbose=False, n_classes=10*)

APGD-Targeted in the paper ‘Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks.’ Targeted attack for every wrong classes. [<https://arxiv.org/abs/2003.01690>] [<https://github.com/fra31/auto-attack>]

Distance Measure : Linf, L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **norm** (*str*) – Lp-norm of the attack. ['Linf', 'L2'] (Default: 'Linf')
- **eps** (*float*) – maximum perturbation. (Default: None)
- **steps** (*int*) – number of steps. (Default: 100)
- **n_restarts** (*int*) – number of random restarts. (Default: 1)
- **seed** (*int*) – random seed for the starting point. (Default: 0)
- **eot_iter** (*int*) – number of iteration for EOT. (Default: 1)
- **rho** (*float*) – parameter for step-size update (Default: 0.75)
- **verbose** (*bool*) – print progress. (Default: False)
- **n_classes** (*int*) – number of classes. (Default: 10)

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.APGDT(model, norm='Linf', eps=8/255, steps=100, n_
↳ restarts=1, seed=0, eot_iter=1, rho=.75, verbose=False, n_classes=10)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.15 FAB

class torchattacks.attacks.fab.FAB (*model, norm='Linf', eps=None, steps=100, n_restarts=1, alpha_max=0.1, eta=1.05, beta=0.9, verbose=False, seed=0, targeted=False, n_classes=10*)

Fast Adaptive Boundary Attack in the paper ‘Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack’ [<https://arxiv.org/abs/1907.02044>] [<https://github.com/fra31/auto-attack>]

Distance Measure : Linf, L2, L1

Parameters

- **model** (*nn.Module*) – model to attack.
- **norm** (*str*) – Lp-norm to minimize. [‘Linf’, ‘L2’, ‘L1’] (Default: ‘Linf’)
- **eps** (*float*) – maximum perturbation. (Default: None)
- **steps** (*int*) – number of steps. (Default: 100)
- **n_restarts** (*int*) – number of random restarts. (Default: 1)
- **alpha_max** (*float*) – alpha_max. (Default: 0.1)
- **eta** (*float*) – overshooting. (Default: 1.05)
- **beta** (*float*) – backward step. (Default: 0.9)
- **verbose** (*bool*) – print progress. (Default: False)
- **seed** (*int*) – random seed for the starting point. (Default: 0)
- **targeted** (*bool*) – targeted attack for every wrong classes. (Default: False)
- **n_classes** (*int*) – number of classes. (Default: 10)

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::


```
>>> attack = torchattacks.FAB(model, norm='Linf', steps=100, eps=None, n_
↳ restarts=1, alpha_max=0.1, eta=1.05, beta=0.9, loss_fn=None, verbose=False,
↳ seed=0, targeted=False, n_classes=10)
>>> adv_images = attack(images, labels)
```

attack_single_run (*x*, *y=None*, *use_rand_start=False*)

Parameters

- **x** – clean images
- **y** – clean labels, if None we use the predicted labels

attack_single_run_targeted (*x*, *y=None*, *use_rand_start=False*)

Parameters

- **x** – clean images
- **y** – clean labels, if None we use the predicted labels

forward (*images*, *labels*)

Overridden.

2.16 Square

```
class torchattacks.attacks.square.Square (model, norm='Linf', eps=None,
n_queries=5000, n_restarts=1, p_init=0.8,
loss='margin', resc_schedule=True, seed=0,
verbose=False)
```

Square Attack in the paper ‘Square Attack: a query-efficient black-box adversarial attack via random search’
[\[https://arxiv.org/abs/1912.00049\]](https://arxiv.org/abs/1912.00049) [\[https://github.com/fra31/auto-attack\]](https://github.com/fra31/auto-attack)

Distance Measure : Linf, L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **norm** (*str*) – Lp-norm of the attack. [‘Linf’, ‘L2’] (Default: ‘Linf’)
- **eps** (*float*) – maximum perturbation. (Default: None)
- **n_queries** (*int*) – max number of queries (each restart). (Default: 5000)
- **n_restarts** (*int*) – number of random restarts. (Default: 1)
- **p_init** (*float*) – parameter to control size of squares. (Default: 0.8)
- **loss** (*str*) – loss function optimized [‘margin’, ‘ce’] (Default: ‘margin’)
- **resc_schedule** (*bool*) – adapt schedule of p to n_queries (Default: True)
- **seed** (*int*) – random seed for the starting point. (Default: 0)
- **verbose** (*bool*) – print progress. (Default: False)
- **targeted** (*bool*) – targeted. (Default: False)

Shape:

- **images**: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].

- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.Square(model, model, norm='Linf', n_queries=5000, n_
↳ restarts=1, eps=None, p_init=.8, seed=0, verbose=False, targeted=False,
↳ loss='margin', resc_schedule=True)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

margin_and_loss (*x, y*)

Parameters **y** – correct labels if untargeted else target labels

p_selection (*it*)

schedule to decrease the parameter p

perturb (*x, y=None*)

Parameters

- **x** – clean images
- **y** – untargeted attack -> clean labels, if None we use the predicted labels targeted attack -> target labels, if None random classes, different from the predicted ones, are sampled

2.17 AutoAttack

class torchattacks.attacks.autoattack.**AutoAttack** (*model, norm='Linf', eps=0.3, version='standard', n_classes=10, seed=None, verbose=False*)

AutoAttack in the paper ‘Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks’ [<https://arxiv.org/abs/2003.01690>] [<https://github.com/fra31/auto-attack>]

Distance Measure : Linf, L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **norm** (*str*) – Lp-norm to minimize. [‘Linf’, ‘L2’] (Default: ‘Linf’)
- **eps** (*float*) – maximum perturbation. (Default: 0.3)
- **version** (*bool*) – version. [‘standard’, ‘plus’, ‘rand’] (Default: ‘standard’)
- **n_classes** (*int*) – number of classes. (Default: 10)
- **seed** (*int*) – random seed for the starting point. (Default: 0)
- **verbose** (*bool*) – print progress. (Default: False)

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range [0, 1].
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.

- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.AutoAttack(model, norm='Linf', eps=.3, version=
↳ 'standard', n_classes=10, seed=None, verbose=False)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)

Overridden.

2.18 OnePixel

class torchattacks.attacks.onepixel.**OnePixel** (*model, pixels=1, steps=75, popsize=400, inf_batch=128*)

Attack in the paper ‘One pixel attack for fooling deep neural networks’ [<https://arxiv.org/abs/1710.08864>]

Modified from “<https://github.com/DebangLi/one-pixel-attack-pytorch/>” and “https://github.com/sarathknv/adversarial-examples-pytorch/blob/master/one_pixel_attack/”

Distance Measure : L0

Parameters

- **model** (*nn.Module*) – model to attack.
- **pixels** (*int*) – number of pixels to change (Default: 1)
- **steps** (*int*) – number of steps. (Default: 75)
- **popsize** (*int*) – population size, i.e. the number of candidate agents or “parents” in differential evolution (Default: 400)
- **inf_batch** (*int*) – maximum batch size during inference (Default: 128)

Shape:

- images: (N, C, H, W) where N = number of batches, C = number of channels, H = height and W = width. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.OnePixel(model, pixels=1, steps=75, popsize=400,
↳ inf_batch=128)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)

Overridden.

2.19 DeepFool

class torchattacks.attacks.deepfool.**DeepFool** (*model, steps=50, overshoot=0.02*)

‘DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks’ [<https://arxiv.org/abs/1511.04599>]

Distance Measure : L2

Parameters

- **model** (*nn.Module*) – model to attack.
- **steps** (*int*) – number of steps. (Default: 50)
- **overshoot** (*float*) – parameter for enhancing the noise. (Default: 0.02)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.DeepFool(model, steps=50, overshoot=0.02)
>>> adv_images = attack(images, labels)
```

forward (*images, labels, return_target_labels=False*)
Overridden.

2.20 SparseFool

class torchattacks.attacks.sparsefool.**SparseFool** (*model, steps=20, lam=3, overshoot=0.02*)

Attack in the paper ‘SparseFool: a few pixels make a big difference’ [<https://arxiv.org/abs/1811.02248>]

Modified from “<https://github.com/LTS4/SparseFool/>”

Distance Measure : L0

Parameters

- **model** (*nn.Module*) – model to attack.
- **steps** (*int*) – number of steps. (Default: 20)
- **lam** (*float*) – parameter for scaling DeepFool noise. (Default: 3)
- **overshoot** (*float*) – parameter for enhancing the noise. (Default: 0.02)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.SparseFool(model, steps=20, lam=3, overshoot=0.02)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.21 DIFGSM

class torchattacks.attacks.difgsm.**DIFGSM** (*model*, *eps=0.03137254901960784*, *alpha=0.00784313725490196*, *steps=20*, *decay=0.0*, *resize_rate=0.9*, *diversity_prob=0.5*, *random_start=False*)

DI2-FGSM in the paper ‘Improving Transferability of Adversarial Examples with Input Diversity’ [<https://arxiv.org/abs/1803.06978>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 8/255)
- **alpha** (*float*) – step size. (Default: 2/255)
- **decay** (*float*) – momentum factor. (Default: 0.0)
- **steps** (*int*) – number of iterations. (Default: 20)
- **resize_rate** (*float*) – resize factor used in input diversity. (Default: 0.9)
- **diversity_prob** (*float*) – the probability of applying input diversity. (Default: 0.5)
- **random_start** (*bool*) – using random initialization of delta. (Default: False)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.DI2FGSM(model, eps=8/255, alpha=2/255, steps=20,
↳decay=0.0, resize_rate=0.9, diversity_prob=0.5, random_start=False)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.22 UPGD

class torchattacks.attacks.upgd.**UPGD** (*model*, *eps=0.03137254901960784*, *alpha=0.00784313725490196*, *steps=40*, *random_start=False*, *loss='ce'*, *decay=1.0*, *eot_iter=1*)

Ultimate PGD that supports various options of gradient-based adversarial attacks.

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 8/255)
- **alpha** (*float*) – step size. (Default: 2/255)
- **steps** (*int*) – number of steps. (Default: 40)
- **random_start** (*bool*) – using random initialization of delta. (Default: False)
- **loss** (*str*) – loss function. ['ce', 'margin', 'dlr'] (Default: 'ce')
- **decay** (*float*) – momentum factor. (Default: 1.0)
- **eot_iter** (*int*) – number of models to estimate the mean gradient. (Default: 1)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::

```
>>> attack = torchattacks.UPGD(model, eps=8/255, alpha=1/255, steps=40,
↳ random_start=False)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.23 TIFGSM

```
class torchattacks.attacks.tifgsm.TIFGSM(model, eps=0.03137254901960784, al-
pha=0.00784313725490196, steps=20,
decay=0.0, kernel_name='gaussian',
len_kernel=15, nsig=3, resize_rate=0.9, di-
versity_prob=0.5, random_start=False)
```

TIFGSM in the paper ‘Evading Defenses to Transferable Adversarial Examples by Translation-Invariant At- tacks’ [<https://arxiv.org/abs/1904.02884>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 8/255)
- **alpha** (*float*) – step size. (Default: 2/255)
- **steps** (*int*) – number of iterations. (Default: 20)
- **decay** (*float*) – momentum factor. (Default: 0.0)
- **kernel_name** (*str*) – kernel name. (Default: gaussian)
- **len_kernel** (*int*) – kernel length. (Default: 15, which is the best according to the paper)

- **nsig** (*int*) – radius of gaussian kernel. (Default: 3; see Section 3.2.2 in the paper for explanation)
- **resize_rate** (*float*) – resize factor used in input diversity. (Default: 0.9)
- **diversity_prob** (*float*) – the probability of applying input diversity. (Default: 0.5)
- **random_start** (*bool*) – using random initialization of delta. (Default: False)

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.TIFGSM(model, eps=8/255, alpha=2/255, steps=20,
↳decay=1.0, resize_rate=0.9, diversity_prob=0.7, random_start=False)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

gkern (*kernlen=15, nsig=3*)
Returns a 2D Gaussian kernel array.

2.24 Jitter

class torchattacks.attacks.jitter.**Jitter** (*model, eps=0.3, alpha=0.00784313725490196, steps=40, scale=10, std=0.1, random_start=True*)

Jitter in the paper ‘Exploring Misclassifications of Robust Neural Networks to Enhance Adversarial Attacks’ [<https://arxiv.org/abs/2105.10304>]

Distance Measure : Linf

Parameters

- **model** (*nn.Module*) – model to attack.
- **eps** (*float*) – maximum perturbation. (Default: 0.3)
- **alpha** (*float*) – step size. (Default: 2/255)
- **steps** (*int*) – number of steps. (Default: 40)
- **random_start** (*bool*) – using random initialization of delta. (Default: True)

Shape:

- images: (N, C, H, W) where $N = \text{number of batches}$, $C = \text{number of channels}$, $H = \text{height}$ and $W = \text{width}$. It must have a range $[0, 1]$.
- labels: (N) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (N, C, H, W) .

Examples::

```
>>> attack = torchattacks.Jitter(model, eps=0.3, alpha=2/255, steps=40,
    scale=10, std=0.1, random_start=True)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)
Overridden.

2.25 Pixle

```
class torchattacks.attacks.pixle.Pixle(model, x_dimensions=(2, 10), y_dimensions=(2,
    10), pixel_mapping='random', restarts=20,
    max_iterations=100, update_each_iteration=False)
```

Pixle: a fast and effective black-box attack based on rearranging pixels' [<https://arxiv.org/abs/2202.02236>]

Distance Measure : L0

Parameters

- **model** (*nn.Module*) – model to attack.
- **x_dimensions** (*int or float, or a tuple containing a combination of those*) – size of the sampled patch along ther x side for each iteration. The integers are considered as fixed number of size,
- **the float as percentage of the size. A tuple is used to specify both under and upper bound of the size. (Default (while) – (2, 10))**
- **y_dimensions** (*int or float, or a tuple containing a combination of those*) – size of the sampled patch along ther y side for each iteration. The integers are considered as fixed number of size,
- **the float as percentage of the size. A tuple is used to specify both under and upper bound of the size. (Default – (2, 10))**
- **pixel_mapping** (*str*) – the type of mapping used to move the pixels. Can be: 'random', 'similarity', 'similarity_random', 'distance', 'distance_random' (Default: random)
- **restarts** (*int*) – the number of restarts that the algortihm performs. (Default: 20)
- **max_iterations** (*int*) – number of iterations to perform for each restart. (Default: 100)
- **update_each_iteration** (*bool*) – if the attacked images must be modified after each iteration (True) or after each restart (False). (Default: False)

Shape:

- images: (*N, C, H, W*) where *N* = number of batches, *C* = number of channels, *H* = height and *W* = width. It must have a range [0, 1].
- labels: (*N*) where each value y_i is $0 \leq y_i \leq \text{number of labels}$.
- output: (*N, C, H, W*).

Examples::


```
>>> attack = torchattacks.Pixle(model, x_dimensions=(0.1, 0.2), restarts=100,
↳ iteration=50)
>>> adv_images = attack(images, labels)
```

forward (*images, labels*)

It defines the computation performed at every call. Should be overridden by all subclasses.

t

torchattacks, 3
torchattacks.attack, 1
torchattacks.attacks.apgd, 10
torchattacks.attacks.apgdt, 11
torchattacks.attacks.autoattack, 14
torchattacks.attacks.bim, 4
torchattacks.attacks.cw, 5
torchattacks.attacks.deepfool, 15
torchattacks.attacks.difgsm, 17
torchattacks.attacks.eotpgd, 8
torchattacks.attacks.fab, 12
torchattacks.attacks.ffgsm, 9
torchattacks.attacks.fgsm, 4
torchattacks.attacks.gn, 3
torchattacks.attacks.jitter, 19
torchattacks.attacks.mifgsm, 10
torchattacks.attacks.onepixel, 15
torchattacks.attacks.pgd, 6
torchattacks.attacks.pgd12, 7
torchattacks.attacks.pixle, 20
torchattacks.attacks.rfgsm, 6
torchattacks.attacks.sparsefool, 16
torchattacks.attacks.square, 13
torchattacks.attacks.tifgsm, 18
torchattacks.attacks.tpgd, 8
torchattacks.attacks.upgd, 17
torchattacks.attacks.vanila, 3

A

APGD (class in `torchattacks.attacks.apgd`), 10
 APGDT (class in `torchattacks.attacks.apgdt`), 11
 Attack (class in `torchattacks.attack`), 1
`attack_single_run()` (torchattacks.attacks.fab.FAB method), 13
`attack_single_run_targeted()` (torchattacks.attacks.fab.FAB method), 13
 AutoAttack (class in `torchattacks.attacks.autoattack`), 14

B

BIM (class in `torchattacks.attacks.bim`), 4

C

CW (class in `torchattacks.attacks.cw`), 5

D

DeepFool (class in `torchattacks.attacks.deepfool`), 15
 DIFGSM (class in `torchattacks.attacks.difgsm`), 17

E

EOTPGD (class in `torchattacks.attacks.eotpgd`), 8

F

FAB (class in `torchattacks.attacks.fab`), 12
 FFGSM (class in `torchattacks.attacks.ffgsm`), 9
 FGSM (class in `torchattacks.attacks.fgsm`), 4
`forward()` (`torchattacks.attack.Attack` method), 1
`forward()` (`torchattacks.attacks.apgd.APGD` method), 11
`forward()` (`torchattacks.attacks.apgdt.APGDT` method), 12
`forward()` (`torchattacks.attacks.autoattack.AutoAttack` method), 15
`forward()` (`torchattacks.attacks.bim.BIM` method), 5
`forward()` (`torchattacks.attacks.cw.CW` method), 6
`forward()` (`torchattacks.attacks.deepfool.DeepFool` method), 16

`forward()` (`torchattacks.attacks.difgsm.DIFGSM` method), 17

`forward()` (`torchattacks.attacks.eotpgd.EOTPGD` method), 8

`forward()` (`torchattacks.attacks.fab.FAB` method), 13

`forward()` (`torchattacks.attacks.ffgsm.FFGSM` method), 9

`forward()` (`torchattacks.attacks.fgsm.FGSM` method), 4

`forward()` (`torchattacks.attacks.gn.GN` method), 4

`forward()` (`torchattacks.attacks.jitter.Jitter` method), 20

`forward()` (`torchattacks.attacks.mifgsm.MIFGSM` method), 10

`forward()` (`torchattacks.attacks.onepixel.OnePixel` method), 15

`forward()` (`torchattacks.attacks.pgd.PGD` method), 7

`forward()` (`torchattacks.attacks.pgdl2.PGDL2` method), 8

`forward()` (`torchattacks.attacks.pixle.Pixle` method), 21

`forward()` (`torchattacks.attacks.rfgsm.RFGSM` method), 6

`forward()` (`torchattacks.attacks.sparsefool.SparseFool` method), 16

`forward()` (`torchattacks.attacks.square.Square` method), 14

`forward()` (`torchattacks.attacks.tifgsm.TIFGSM` method), 19

`forward()` (`torchattacks.attacks.tpgd.TPGD` method), 9

`forward()` (`torchattacks.attacks.upgd.UPGD` method), 18

`forward()` (`torchattacks.attacks.vanila.VANILA` method), 3

G

`get_mode()` (`torchattacks.attack.Attack` method), 1

`gkern()` (`torchattacks.attacks.tifgsm.TIFGSM` method), 19

GN (class in torchattacks.attacks.gn), 3

J

Jitter (class in torchattacks.attacks.jitter), 19

M

margin_and_loss() (torchattacks.attacks.square.Square method), 14

MIFGSM (class in torchattacks.attacks.mifgsm), 10

O

OnePixel (class in torchattacks.attacks.onepixel), 15

P

p_selection() (torchattacks.attacks.square.Square method), 14

perturb() (torchattacks.attacks.square.Square method), 14

PGD (class in torchattacks.attacks.pgd), 6

PGDL2 (class in torchattacks.attacks.pgdL2), 7

Pixle (class in torchattacks.attacks.pixle), 20

R

RFGSM (class in torchattacks.attacks.rfgsm), 6

S

save() (torchattacks.attack.Attack method), 1

set_mode_default() (torchattacks.attack.Attack method), 1

set_mode_targeted_by_function() (torchattacks.attack.Attack method), 1

set_mode_targeted_least_likely() (torchattacks.attack.Attack method), 2

set_mode_targeted_random() (torchattacks.attack.Attack method), 2

set_return_type() (torchattacks.attack.Attack method), 2

set_training_mode() (torchattacks.attack.Attack method), 2

SparseFool (class in torchattacks.attacks.sparsefool), 16

Square (class in torchattacks.attacks.square), 13

T

TIFGSM (class in torchattacks.attacks.tifgsm), 18

torchattacks (module), 1, 3

torchattacks.attack (module), 1

torchattacks.attacks.apgd (module), 10

torchattacks.attacks.apgdt (module), 11

torchattacks.attacks.autoattack (module), 14

torchattacks.attacks.bim (module), 4

torchattacks.attacks.cw (module), 5

torchattacks.attacks.deepfool (module), 15

torchattacks.attacks.difgsm (module), 17

torchattacks.attacks.eotpgd (module), 8

torchattacks.attacks.fab (module), 12

torchattacks.attacks.ffgsm (module), 9

torchattacks.attacks.fgsm (module), 4

torchattacks.attacks.gn (module), 3

torchattacks.attacks.jitter (module), 19

torchattacks.attacks.mifgsm (module), 10

torchattacks.attacks.onepixel (module), 15

torchattacks.attacks.pgd (module), 6

torchattacks.attacks.pgdL2 (module), 7

torchattacks.attacks.pixle (module), 20

torchattacks.attacks.rfgsm (module), 6

torchattacks.attacks.sparsefool (module), 16

torchattacks.attacks.square (module), 13

torchattacks.attacks.tifgsm (module), 18

torchattacks.attacks.tpgd (module), 8

torchattacks.attacks.upgd (module), 17

torchattacks.attacks.vanila (module), 3

TPGD (class in torchattacks.attacks.tpgd), 8

U

UPGD (class in torchattacks.attacks.upgd), 17

V

VANILA (class in torchattacks.attacks.vanila), 3