CleverHans Documentation

Ian Goodfellow, Nicolas Papernot, Ryan Sheatsley

Jun 22, 2018

Contents

This documentation is auto-generated from the docstrings of modules of the current *master* branch of tensorflow/cleverhans.

To get started, we recommend reading the github readme. Afterwards, you can learn more by looking at the following modules:

CHAPTER 1

attacks module

class cleverhans.attacks.**Attack**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: object

Abstract base class for all attack classes.

construct_graph(*fixed*, *feedable*, *x_val*, *hash_key*)

Construct the graph required to run the attack through generate_np.

Parameters

- **fixed** – Structural elements that require defining a new graph.
- **feedable** – Arguments that can be fed to the same graph when they take different values.
- **x_val** – symbolic adversarial example
- **hash_key** – the key used to store this graph in our cache

construct_variables(*kwargs*)

Construct the inputs to the attack graph to be used by generate_np.

Parameters **kwargs** – Keyword arguments to generate_np.

Returns Structural and feedable arguments as well as a unique key for the graph given these inputs.

generate(*x*, ***kwargs*)

Generate the attack's symbolic graph for adversarial examples. This method should be overriden in any child class that implements an attack that is expressable symbolically. Otherwise, it will wrap the numerical implementation as a symbolic operator.

Parameters

- **x** – The model's symbolic inputs.
- ****kwargs** – optional parameters used by child classes.

Returns A symbolic representation of the adversarial examples.

generate_np(*x_val*, ***kwargs*)
Generate adversarial examples and return them as a NumPy array. Sub-classes *should not* implement this method unless they must perform special handling of arguments.

Parameters

- **x_val** – A NumPy array with the original inputs.
- ****kwargs** – optional parameters used by child classes.

Returns A NumPy array holding the adversarial examples.

get_or_guess_labels(*x*, *kwargs*)
Get the label to use in generating an adversarial example for x. The kwargs are fed directly from the kwargs of the attack. If 'y' is in kwargs, then assume it's an untargeted attack and use that as the label. If 'y_target' is in kwargs, then assume it's a targeted attack and use that as the label. Otherwise, use the model's prediction as the label and perform an untargeted attack.

parse_params(*params=None*)
Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Parameters **params** – a dictionary of attack-specific parameters

Returns True when parsing was successful

class cleverhans.attacks.**BasicIterativeMethod**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)
Bases: *cleverhans.attacks.Attack*

The Basic Iterative Method (Kurakin et al. 2016). The original paper used hard labels for this attack; no label smoothing. Paper link: https://arxiv.org/pdf/1607.02533.pdf

generate(*x*, ***kwargs*)
Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **y** – (optional) A tensor with the model labels.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params(*eps=0.3*, *eps_iter=0.05*, *nb_iter=10*, *y=None*, *ord=inf*, *clip_min=None*, *clip_max=None*, *y_target=None*, ***kwargs*)
Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **eps** – (required float) maximum distortion of adversarial example compared to original input

- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **y** – (optional) A tensor with the model labels.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

class cleverhans.attacks.**CarliniWagnerL2**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

This attack was originally proposed by Carlini and Wagner. It is an iterative attack that finds adversarial examples on many defenses that are robust to other attacks. Paper link: https://arxiv.org/abs/1608.04644

At a high level, this attack is an iterative attack using Adam and a specially-chosen loss function to find adversarial examples with lower distortion than other attacks. This comes at the cost of speed, as this attack is often much slower than others.

generate(*x*, ***kwargs*)

Return a tensor that constructs adversarial examples for the given input. Generate uses tf.py_func in order to operate over tensors.

Parameters

- **x** – (required) A tensor with the inputs.
- **y** – (optional) A tensor with the true labels for an untargeted attack. If None (and y_target is None) then use the original labels the classifier assigns.
- **y_target** – (optional) A tensor with the target labels for a targeted attack.
- **confidence** – Confidence of adversarial examples: higher produces examples with larger l2 distortion, but more strongly classified as adversarial.
- **batch_size** – Number of attacks to run simultaneously.
- **learning_rate** – The learning rate for the attack algorithm. Smaller values produce better results but are slower to converge.
- **binary_search_steps** – The number of times we perform binary search to find the optimal tradeoff- constant between norm of the purturbation and confidence of the classification.
- **max_iterations** – The maximum number of iterations. Setting this to a larger value will produce lower distortion results. Using only a few iterations requires a larger learning rate, and will produce larger distortion results.
- **abort_early** – If true, allows early aborts if gradient descent is unable to make progress (i.e., gets stuck in a local minimum).
- **initial_const** – The initial tradeoff-constant to use to tune the relative importance of size of the pururbation and confidence of classification. If binary_search_steps is large, the initial constant is not important. A smaller value of this constant gives lower distortion results.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params(*y=None*, *y_target=None*, *nb_classes=None*, *batch_size=1*, *confidence=0*, *learning_rate=0.005*, *binary_search_steps=5*, *max_iterations=1000*, *abort_early=True*, *initial_const=0.01*, *clip_min=0*, *clip_max=1*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Parameters **params** – a dictionary of attack-specific parameters

Returns True when parsing was successful

class cleverhans.attacks.**DeepFool**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

DeepFool is an untargeted & iterative attack which is based on an iterative linearization of the classifier. The implementation here is w.r.t. the L2 norm. Paper link: "https://arxiv.org/pdf/1511.04599.pdf"

generate(*x*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **nb_candidate** – The number of classes to test against, i.e., deepfool only consider nb_candidate classes when attacking(thus accelerate speed). The nb_candidate classes are chosen according to the prediction confidence during implementation.
- **overshoot** – A termination criterion to prevent vanishing updates
- **max_iter** – Maximum number of iteration for deepfool
- **nb_classes** – The number of model output classes
- **clip_min** – Minimum component value for clipping
- **clip_max** – Maximum component value for clipping

parse_params(*nb_candidate=10*, *overshoot=0.02*, *max_iter=50*, *nb_classes=None*, *clip_min=0.0*, *clip_max=1.0*, ***kwargs*)

Parameters

- **nb_candidate** – The number of classes to test against, i.e., deepfool only consider nb_candidate classes when attacking(thus accelerate speed). The nb_candidate classes are chosen according to the prediction confidence during implementation.
- **overshoot** – A termination criterion to prevent vanishing updates
- **max_iter** – Maximum number of iteration for deepfool
- **nb_classes** – The number of model output classes
- **clip_min** – Minimum component value for clipping
- **clip_max** – Maximum component value for clipping

class cleverhans.attacks.**ElasticNetMethod**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

This attack features L1-oriented adversarial examples and includes the C&W L2 attack as a special case (when beta is set to 0). Adversarial examples attain similar performance to those generated by the C&W L2 attack in the white-box case, and more importantly, have improved transferability properties and complement adversarial training. Paper link: https://arxiv.org/abs/1709.04114

generate(*x*, ***kwargs*)

Return a tensor that constructs adversarial examples for the given input. Generate uses tf.py_func in order to operate over tensors.

Parameters

- **x** – (required) A tensor with the inputs.
- **y** – (optional) A tensor with the true labels for an untargeted attack. If None (and y_target is None) then use the original labels the classifier assigns.
- **y_target** – (optional) A tensor with the target labels for a targeted attack.
- **fista** – FISTA or ISTA. FISTA has better convergence properties but performs an additional query per iteration
- **beta** – Trades off L2 distortion with L1 distortion: higher produces examples with lower L1 distortion, at the cost of higher L2 (and typically Linf) distortion
- **decision_rule** – EN or L1. Select final adversarial example from all successful examples based on the least elastic-net or L1 distortion criterion.
- **confidence** – Confidence of adversarial examples: higher produces examples with larger l2 distortion, but more strongly classified as adversarial.
- **batch_size** – Number of attacks to run simultaneously.
- **learning_rate** – The learning rate for the attack algorithm. Smaller values produce better results but are slower to converge.
- **binary_search_steps** – The number of times we perform binary search to find the optimal tradeoff- constant between norm of the perturbation and confidence of the classification.
- **max_iterations** – The maximum number of iterations. Setting this to a larger value will produce lower distortion results. Using only a few iterations requires a larger learning rate, and will produce larger distortion results.
- **abort_early** – If true, allows early abort when the total loss starts to increase (greatly speeds up attack, but hurts performance, particularly on ImageNet)
- **initial_const** – The initial tradeoff-constant to use to tune the relative importance of size of the perturbation and confidence of classification. If binary_search_steps is large, the initial constant is not important. A smaller value of this constant gives lower distortion results.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params(*y=None*, *y_target=None*, *nb_classes=None*, *fista=True*, *beta=0.001*, *decision_rule='EN'*, *batch_size=1*, *confidence=0*, *learning_rate=0.01*, *binary_search_steps=9*, *max_iterations=1000*, *abort_early=False*, *initial_const=0.001*, *clip_min=0*, *clip_max=1*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Parameters **params** – a dictionary of attack-specific parameters

Returns True when parsing was successful

class cleverhans.attacks.**FastFeatureAdversaries**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

This is a fast implementation of "Feature Adversaries", an attack against a target internal representation of a model. "Feature adversaries" were originally introduced in (Sabour et al. 2016), where the optimization was done using LBFGS. Paper link: https://arxiv.org/abs/1511.05122

This implementation is similar to "Basic Iterative Method" (Kurakin et al. 2016) but applied to the internal representations.

attack_single_step(*x*, *eta*, *g_feat*)

TensorFlow implementation of the Fast Feature Gradient. This is a single step attack similar to Fast Gradient Method that attacks an internal representation.

Parameters

- **x** – the input placeholder
- **eta** – A tensor the same shape as x that holds the perturbation.
- **g_feat** – model's internal tensor for guide

Returns a tensor for the adversarial example

generate(*x*, *g*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **g** – The target's symbolic representation.
- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params(*layer=None*, *eps=0.3*, *eps_iter=0.05*, *nb_iter=10*, *ord=inf*, *clip_min=None*, *clip_max=None*, ***kwargs*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **layer** – (required str) name of the layer to target.
- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

class cleverhans.attacks.**FastGradientMethod**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

This attack was originally implemented by Goodfellow et al. (2015) with the infinity norm (and is known as the "Fast Gradient Sign Method"). This implementation extends the attack to other norms, and is therefore called the Fast Gradient Method. Paper link: https://arxiv.org/abs/1412.6572

generate(*x*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **eps** – (optional float) attack step size (input variation)
- **ord** – (optional) Order of the norm (mimics NumPy). Possible values: np.inf, 1 or 2.
- **y** – (optional) A tensor with the model labels. Only provide this parameter if you'd like to use true labels when crafting adversarial samples. Otherwise, model predictions are used as labels to avoid the "label leaking" effect (explained in this paper: https://arxiv.org/abs/1611.01236). Default is None. Labels should be one-hot-encoded.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params(*eps=0.3*, *ord=inf*, *y=None*, *y_target=None*, *clip_min=None*, *clip_max=None*, ***kwargs*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **eps** – (optional float) attack step size (input variation)
- **ord** – (optional) Order of the norm (mimics NumPy). Possible values: np.inf, 1 or 2.
- **y** – (optional) A tensor with the model labels. Only provide this parameter if you'd like to use true labels when crafting adversarial samples. Otherwise, model predictions are used as labels to avoid the "label leaking" effect (explained in this paper: https://arxiv.org/abs/1611.01236). Default is None. Labels should be one-hot-encoded.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

class cleverhans.attacks.**LBFGS**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

LBFGS is the first adversarial attack for convolutional neural networks, and is a target & iterative attack. Paper link: "https://arxiv.org/pdf/1312.6199.pdf"

generate(*x*, ***kwargs*)

Return a tensor that constructs adversarial examples for the given input. Generate uses tf.py_func in order to operate over tensors.

Parameters

- **x** – (required) A tensor with the inputs.
- **y_target** – (required) A tensor with the one-hot target labels.
- **batch_size** – The number of inputs to include in a batch and process simultaneously.
- **binary_search_steps** – The number of times we perform binary search to find the optimal tradeoff- constant between norm of the purturbation and cross-entropy loss of classification.
- **max_iterations** – The maximum number of iterations.
- **initial_const** – The initial tradeoff-constant to use to tune the relative importance of size of the perturbation and cross-entropy loss of the classification.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params (*y_target=None*, *batch_size=1*, *binary_search_steps=5*, *max_iterations=1000*, *initial_const=0.01*, *clip_min=0*, *clip_max=1*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Parameters **params** – a dictionary of attack-specific parameters

Returns True when parsing was successful

class cleverhans.attacks.**MadryEtAl**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

The Projected Gradient Descent Attack (Madry et al. 2017). Paper link: https://arxiv.org/pdf/1706.06083.pdf

attack(*x*, *y*)

This method creates a symbolic graph that given an input image, first randomly perturbs the image. The perturbation is bounded to an epsilon ball. Then multiple steps of gradient descent is performed to increase the probability of a target label or decrease the probability of the ground-truth label.

Parameters **x** – A tensor with the input image.

attack_single_step(*x*, *eta*, *y*)

Given the original image and the perturbation computed so far, computes a new perturbation.

Parameters

- **x** – A tensor with the original input.
- **eta** – A tensor the same shape as x that holds the perturbation.
- **y** – A tensor with the target labels or ground-truth labels.

generate(*x*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **y** – (optional) A tensor with the model labels.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.

- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value
- **rand_init** – (optional bool) If True, an initial random perturbation is added.

parse_params(*eps=0.3*, *eps_iter=0.01*, *nb_iter=40*, *y=None*, *ord=inf*, *clip_min=None*, *clip_max=None*, *y_target=None*, *rand_init=True*, ***kwargs*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **y** – (optional) A tensor with the model labels.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value
- **rand_init** – (optional bool) If True, an initial random perturbation is added.

class cleverhans.attacks.**MomentumIterativeMethod**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

The Momentum Iterative Method (Dong et al. 2017). This method won the first places in NIPS 2017 Non-targeted Adversarial Attacks and Targeted Adversarial Attacks. The original paper used hard labels for this attack; no label smoothing. Paper link: https://arxiv.org/pdf/1710.06081.pdf

generate(*x*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **y** – (optional) A tensor with the model labels.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **decay_factor** – (optional) Decay factor for the momentum term.
- **clip_min** – (optional float) Minimum input component value

- **clip_max** – (optional float) Maximum input component value

parse_params(*eps=0.3*, *eps_iter=0.06*, *nb_iter=10*, *y=None*, *ord=inf*, *decay_factor=1.0*, *clip_min=None*, *clip_max=None*, *y_target=None*, ***kwargs*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **eps** – (required float) maximum distortion of adversarial example compared to original input
- **eps_iter** – (required float) step size for each attack iteration
- **nb_iter** – (required int) Number of attack iterations.
- **y** – (optional) A tensor with the model labels.
- **y_target** – (optional) A tensor with the labels to target. Leave y_target=None if y is also set. Labels should be one-hot-encoded.
- **ord** – (optional) Order of the norm (mimics Numpy). Possible values: np.inf, 1 or 2.
- **decay_factor** – (optional) Decay factor for the momentum term.
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

class cleverhans.attacks.**SPSA**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

This implements the SPSA adversary, as in https://arxiv.org/abs/1802.05666 (Uesato et al. 2018). SPSA is a gradient-free optimization method, which is useful when the model is non-differentiable, or more generally, the gradients do not point in useful directions.

generate(*x*, *y=None*, *y_target=None*, *epsilon=None*, *num_steps=None*, *is_targeted=False*, *early_stop_loss_threshold=None*, *learning_rate=0.01*, *delta=0.01*, *batch_size=128*, *spsa_iters=1*, *is_debug=False*)

Generate symbolic graph for adversarial examples.

Parameters

- **x** – The model's symbolic inputs. Must be a batch of size 1.
- **y** – A Tensor or None. The index of the correct label.
- **y_target** – A Tensor or None. The index of the target label in a targeted attack.
- **epsilon** – The size of the maximum perturbation, measured in the L-infinity norm.
- **num_steps** – The number of optimization steps.
- **is_targeted** – Whether to use a targeted or untargeted attack.
- **early_stop_loss_threshold** – A float or None. If specified, the attack will end as soon as the loss is below *early_stop_loss_threshold*.
- **learning_rate** – Learning rate of ADAM optimizer.
- **delta** – Perturbation size used for SPSA approximation.
- **batch_size** – Number of inputs to evaluate at a single time. Note that the true batch size (the number of evaluated inputs for each update) is *batch_size * spsa_iters*
- **spsa_iters** – Number of model evaluations before performing an update, where each evaluation is on *batch_size* different inputs.

- **is_debug** – If True, print the adversarial loss after each update.

class cleverhans.attacks.**SaliencyMapMethod**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

The Jacobian-based Saliency Map Method (Papernot et al. 2016). Paper link: https://arxiv.org/pdf/1511.07528.pdf

generate(*x*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **theta** – (optional float) Perturbation introduced to modified components (can be positive or negative)
- **gamma** – (optional float) Maximum percentage of perturbed features
- **clip_min** – (optional float) Minimum component value for clipping
- **clip_max** – (optional float) Maximum component value for clipping
- **y_target** – (optional) Target tensor if the attack is targeted

parse_params(*theta=1.0*, *gamma=1.0*, *nb_classes=None*, *clip_min=0.0*, *clip_max=1.0*, *y_target=None*, *symbolic_impl=True*, ***kwargs*)

Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **theta** – (optional float) Perturbation introduced to modified components (can be positive or negative)
- **gamma** – (optional float) Maximum percentage of perturbed features
- **nb_classes** – (optional int) Number of model output classes
- **clip_min** – (optional float) Minimum component value for clipping
- **clip_max** – (optional float) Maximum component value for clipping
- **y_target** – (optional) Target tensor if the attack is targeted

class cleverhans.attacks.**VirtualAdversarialMethod**(*model*, *back='tf'*, *sess=None*, *dtypestr='float32'*)

Bases: *cleverhans.attacks.Attack*

This attack was originally proposed by Miyato et al. (2016) and was used for virtual adversarial training. Paper link: https://arxiv.org/abs/1507.00677

generate(*x*, ***kwargs*)

Generate symbolic graph for adversarial examples and return.

Parameters

- **x** – The model's symbolic inputs.
- **eps** – (optional float) the epsilon (input variation parameter)
- **num_iterations** – (optional) the number of iterations
- **xi** – (optional float) the finite difference parameter

- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

parse_params(*eps=2.0*, *num_iterations=1*, *xi=1e-06*, *clip_min=None*, *clip_max=None*, ***kwargs*)
Take in a dictionary of parameters and applies attack-specific checks before saving them as attributes.

Attack-specific parameters:

Parameters

- **eps** – (optional float)the epsilon (input variation parameter)
- **num_iterations** – (optional) the number of iterations
- **xi** – (optional float) the finite difference parameter
- **clip_min** – (optional float) Minimum input component value
- **clip_max** – (optional float) Maximum input component value

cleverhans.attacks.**vatm**(*model*, *x*, *logits*, *eps*, *back='tf'*, *num_iterations=1*, *xi=1e-06*, *clip_min=None*, *clip_max=None*)
A wrapper for the perturbation methods used for virtual adversarial training : https://arxiv.org/abs/1507.00677
It calls the right function, depending on the user's backend.

Parameters

- **model** – the model which returns the network unnormalized logits
- **x** – the input placeholder
- **logits** – the model's unnormalized output tensor
- **eps** – the epsilon (input variation parameter)
- **num_iterations** – the number of iterations
- **xi** – the finite difference parameter
- **clip_min** – optional parameter that can be used to set a minimum value for components of the example returned
- **clip_max** – optional parameter that can be used to set a maximum value for components of the example returned

Returns a tensor for the adversarial example

CHAPTER 2

model module

class cleverhans.model.**CallableModelWrapper**(*callable_fn*, *output_layer*)

Bases: *cleverhans.model.Model*

fprop(*x*)

Exposes all the layers of the model returned by get_layer_names. :param x: A symbolic representation of the network input :return: A dictionary mapping layer names to the symbolic

representation of their output.

get_layer_names()

Returns a list of names for the layers that can be exposed by this

model abstraction.

class cleverhans.model.**Model**

Bases: object

An abstract interface for model wrappers that exposes model symbols needed for making an attack. This abstraction removes the dependency on any specific neural network package (e.g. Keras) from the core code of CleverHans. It can also simplify exposing the hidden features of a model when a specific package does not directly expose them.

fprop(*x*)

Exposes all the layers of the model returned by get_layer_names. :param x: A symbolic representation of the network input :return: A dictionary mapping layer names to the symbolic

representation of their output.

get_layer(*x*, *layer*)

Expose the hidden features of a model given a layer name. :param x: A symbolic representation of the network input :param layer: The name of the hidden layer to return features at. :return: A symbolic representation of the hidden features :raise: NoSuchLayerError if *layer* is not in the model.

get_layer_names()

Returns a list of names for the layers that can be exposed by this

model abstraction.

get_logits(*x*)

Parameters **x** – A symbolic representation of the network input

Returns A symbolic representation of the output logits (i.e., the values fed as inputs to the softmax layer).

get_params()

Provides access to the model's parameters. :return: A list of all Variables defining the model parameters.

get_probs(*x*)

Parameters **x** – A symbolic representation of the network input

Returns A symbolic representation of the output probabilities (i.e., the output values produced by the softmax layer).

exception cleverhans.model.**NoSuchLayerError**

Bases: exceptions.ValueError

Raised when a layer that does not exist is requested.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

N

P

S

V