ExAD

August 17, 2020

1 ExAD: An Ensemble Approach for Explanation-basedAdversarial Detection

1.0.1 In this Jupyter notebook, we provide reference code for our whitebox evaluation.

For more details, please refer to Section 4.6 of the paper as well as Appendix E.

[1]:
```
import os
# os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
# os.environ["CUDA_VISIBLE_DEVICES"]="1"

import tensorflow as tf
import keras
from keras import backend as K
from keras.models import Model
from keras.utils import to_categorical
from keras.layers import Input, Dense, Lambda, merge, Dropout, Flatten, Conv2D,␣
↪MaxPooling2D
from keras.models import Sequential, model_from_json, load_model
from keras.layers.core import Activation
from keras import datasets
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from numpy import clip
import pickle
import pandas as pd
from IPython.display import display
```

```
/Users/administrator/opt/anaconda3/envs/tf_env/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:523: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/administrator/opt/anaconda3/envs/tf_env/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:524: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
```

```
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/administrator/opt/anaconda3/envs/tf_env/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/administrator/opt/anaconda3/envs/tf_env/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/administrator/opt/anaconda3/envs/tf_env/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/administrator/opt/anaconda3/envs/tf_env/lib/python3.6/site-
packages/tensorflow/python/framework/dtypes.py:532: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
```

2 Utility Methods

[2]:
```
# Utility Methods
def Union(lst1, lst2):
    final_list = list(set(lst1) | set(lst2))
    return final_list
```

3 Configuration

[3]:
```
# Datasets on which we conduct our evaluation
datasets = ['mnist', 'fmnist', 'cifar10']

# The explanation methods that will be used in the ensemble
exp_methods = ['lrp', 'guided_backprop',␣
 ↪'integrated_grad','pattern_attribution', 'grad_times_input']

# The directory of the detector models
# Format: 'data/defender/<dataset>/orig/train/<explation_method>/
 ↪<target_class_index>/model/'
model_dir_structure = 'data/defender/{}/orig/train/{}/{}/model/'
model_name = 'exp_model.h5'
```

```
# The directory where we store the adversarial examples created using whitebox␣
↪attack
# Format 'data/adv2/adversary/<dataset>/<target_exp_method>/
↪<attack_method_to_test>/target_next/target_<class_idx>'
adv2_dir_structure = 'data/adversary/{}/adv2/{}/{}/target_next/target_{}/'
succ_examples_filename = 'succ_on_f.npy'

# The directory where we store the abnormal explanations created from␣
↪adversarial examples which were inturn created using whitebox attack
adv2_exp_dir_structure = 'data/adversary/{}/adv2/{}/{}/{}/from_{}/'
explanations_filename = 'expls.npy'

src_exp_methods = exp_methods[:]
src_exp_methods.append('overall')

enable_detailed_logs = False
```

4 Load the ExAD-CNN detector models

As discussed in Section 3.4.1 of our paper, for every class in a dataset, we train 5 detector models- one corresponding to every explanation technique.

[4]:
```
import collections

# Dictionary of models.
# This avoids having to reload models multiple times, and therefore speeds up␣
↪the analysis.
print('\nLoading detector models')
if not enable_detailed_logs:
    print('Note: You have turned off option to view detailed logs. Loading␣
↪models may take a while.')

model_d = {}
for dataset in datasets:
    for class_idx in range(10):
        for exp_method in exp_methods:
            if enable_detailed_logs: print('\tLoading detector model for dataset:
↪{} target class:{} explanation technique:{}'.
↪format(dataset,class_idx,exp_method))
            model_dir = model_dir_structure.format(dataset, exp_method,␣
↪str(class_idx))
            model = load_model(model_dir + model_name)
            model_d[(dataset, class_idx, exp_method)] = model
```

```
Loading detector models
Note: You have turned off option to view detailed logs. Loading models may take
a while.
```

[5]:
```python
df_list = list()

for dataset in datasets:
    print('Running whitebox attack evaluation on {} dataset'.format(dataset))

    # set the image dimension (side=H=W) based on the dataset
    if dataset in ['mnist', 'fmnist']:
        side = 28
    elif dataset in ['cifar10']:
        side = 32

    d = collections.defaultdict(list)
    total_adv_per_class_per_attack = 10

    table = []
    for target_exp_method in exp_methods:
        print('\n\tTargeting {} explanation technique'.format(target_exp_method))
        row = []
        for class_idx in range(10):
            if class_idx == 0:
                src_class_index = 9
            else:
                src_class_index = class_idx-1

            if target_exp_method == 'integrated_grad':
                attack_methods_to_test = ['cwl2/conf_0']
            else:
                attack_methods_to_test = ['cwl2/conf_0', 'cwlinf/conf_0', 'cwl0/
 ↪conf_0', 'bim', 'mim', 'jsma']

            for attack_method_to_test in attack_methods_to_test:
                # The directory of adv2 examples which fool both f(.) and the␣
 ↪target_exp_method
                # we do this to ensure we compute the performance on successful␣
 ↪adv2 examples only
                adv2_dir = adv2_dir_structure.format(dataset, target_exp_method,␣
 ↪attack_method_to_test, str(class_idx))
                succ_on_f = np.load(adv2_dir + succ_examples_filename)
                retained_adv_len = len(np.where(succ_on_f == True)[0])

                # if say 8/10 adversarial examples were successful, then failed␣
 ↪will have indices from 0 to 7
```

```
                failed = np.array([i for i in range(retained_adv_len)])

                for exp_method in exp_methods:
                    # Use the detector model for class_idx (target class)␣
↪corresponding to exp_method (explanation technique)
                    model = model_d[(dataset, class_idx, exp_method)]

                    adv2_exp_dir = adv2_exp_dir_structure.format(dataset,␣
↪target_exp_method, attack_method_to_test, exp_method, str(src_class_index) )
                    adv2_exp = np.load(adv2_exp_dir + explanations_filename)

                    # NOTE we retain successful adv2 examples
                    adv2_exp = adv2_exp[succ_on_f]

                    # process images for classification
                    adv2_exp *= 255.0/np.max(adv2_exp)
                    adv2_exp = adv2_exp.astype(int)
                    adv2_exp = adv2_exp.reshape(-1,side,side,1)

                    #evaluate model on adv2_exp samples
                    result_test = model.predict(adv2_exp)

                    result_test_class = np.argmax(result_test, axis=1)
                    true_pos = len(np.where(result_test_class==1)[0])
                    total_pos = len(result_test_class)
                    det_rate = true_pos*100/total_pos

                    failed_cur_method = np.where(result_test_class==0)[0]
                    failed = np.intersect1d(failed, failed_cur_method)

                    d[(target_exp_method, exp_method)].append(det_rate)
                    d[(target_exp_method, attack_method_to_test, exp_method)].
↪append(det_rate)

                true_pos_cumulative = total_adv_per_class_per_attack -␣
↪len(failed)
                det_rate_cumulative = true_pos_cumulative * 100 /␣
↪total_adv_per_class_per_attack
                d[(target_exp_method, 'overall')].append(det_rate_cumulative)
                d[(target_exp_method, attack_method_to_test, 'overall')].
↪append(det_rate_cumulative)


        for src_exp_method in src_exp_methods:
            l = np.array(d[(target_exp_method, src_exp_method)])
            mean_detection_rate = round(np.mean(l),3)
```

```
            row.append(mean_detection_rate)

        table.append(row)

    arr = np.array(table)
    df = pd.DataFrame(arr, index=exp_methods, columns=src_exp_methods)
    df_list.append(df)

    print('\n\bThis is the whitebox results for {}. \nThese results should be␣
↪nearly consistent with Figure 8 of the paper in Appendix.'.format(dataset))
    print('First column shows the targeted explanaton technique. Columns 1-5␣
↪shows detection results by detector models corresponding to the explanation␣
↪technique. Column 6 (rightmost) shows overall detection results by ExAD.')
    print(df)
```

```
Running whitebox attack evaluation on mnist dataset

        Targeting lrp explanation technique

        Targeting guided_backprop explanation technique

        Targeting integrated_grad explanation technique

        Targeting pattern_attribution explanation technique

        Targeting grad_times_input explanation technique

This is the whitebox results for mnist.
These results should be nearly consistent with Figure 8 of the paper in
Appendix.
First column shows the targeted explanaton technique. Columns 1-5 shows
detection results by detector models corresponding to the explanation technique.
Column 6 (rightmost) shows overall detection results by ExAD.
                         lrp  guided_backprop  integrated_grad  \
lrp                    0.000           22.667           82.741
guided_backprop       10.444            0.000           84.245
integrated_grad      100.000           98.750           18.095
pattern_attribution    1.542           31.476           81.077
grad_times_input      98.722           99.458           56.700

                     pattern_attribution  grad_times_input  overall
lrp                               15.833            83.463   90.667
guided_backprop                    7.481            85.838   89.000
integrated_grad                   90.000            55.837  100.000
pattern_attribution               10.167            85.276   89.667
grad_times_input                  89.111            14.111  100.000
```

```
Running whitebox attack evaluation on fmnist dataset

        Targeting lrp explanation technique

        Targeting guided_backprop explanation technique

        Targeting integrated_grad explanation technique

        Targeting pattern_attribution explanation technique

        Targeting grad_times_input explanation technique

This is the whitebox results for fmnist.
These results should be nearly consistent with Figure 8 of the paper in
Appendix.
First column shows the targeted explanaton technique. Columns 1-5 shows
detection results by detector models corresponding to the explanation technique.
Column 6 (rightmost) shows overall detection results by ExAD.
                         lrp  guided_backprop  integrated_grad  \
lrp                    0.000           27.111           86.148
guided_backprop       55.378            6.868           90.060
integrated_grad      100.000           96.667           41.000
pattern_attribution   13.720           34.176           84.880
grad_times_input      95.394           95.033           71.955

                     pattern_attribution  grad_times_input  overall
lrp                               20.000            93.130   95.667
guided_backprop                   28.648            95.759   96.833
integrated_grad                   96.667            89.000  100.000
pattern_attribution               12.870            93.403   95.500
grad_times_input                  95.871            41.372   99.333
Running whitebox attack evaluation on cifar10 dataset

        Targeting lrp explanation technique

        Targeting guided_backprop explanation technique

        Targeting integrated_grad explanation technique

        Targeting pattern_attribution explanation technique

        Targeting grad_times_input explanation technique

This is the whitebox results for cifar10.
These results should be nearly consistent with Figure 8 of the paper in
Appendix.
First column shows the targeted explanaton technique. Columns 1-5 shows
detection results by detector models corresponding to the explanation technique.
```

```
Column 6 (rightmost) shows overall detection results by ExAD.
                        lrp  guided_backprop  integrated_grad  \
lrp                  26.111            3.685           58.898
guided_backprop      33.106           10.167           63.088
integrated_grad      97.000           71.972           12.111
pattern_attribution  25.773            7.940           60.394
grad_times_input     79.398           62.116           50.264

                     pattern_attribution  grad_times_input  overall
lrp                               50.000            57.921   84.000
guided_backprop                   66.991            56.787   88.833
integrated_grad                   90.778            49.111   99.000
pattern_attribution               24.444            50.023   80.667
grad_times_input                  89.370             2.759   97.500
```

5 Review the results of whitebox attack

5.1 Review the results of whitebox attack on individual datasets

The results below should be consistent with Figure 8 of the paper in Appendix E.

How to interpret the results: First (or index) column shows the targeted explanaton technique. Columns 1-5 shows detection results by detector models corresponding to different explanation techniques. Column 6 (rightmost) shows overall detection results by ExAD (under CNN detector model setting).

[6]:
```
for dataset_idx in range(len(df_list)):
    print('\n\nWhitebox results for {}.'.format(datasets[dataset_idx]))
    display(df_list[dataset_idx])
```

```
Whitebox results for mnist.

                         lrp  guided_backprop  integrated_grad  \
lrp                    0.000           22.667           82.741
guided_backprop       10.444            0.000           84.245
integrated_grad      100.000           98.750           18.095
pattern_attribution    1.542           31.476           81.077
grad_times_input      98.722           99.458           56.700

                     pattern_attribution  grad_times_input  overall
lrp                               15.833            83.463   90.667
guided_backprop                    7.481            85.838   89.000
integrated_grad                   90.000            55.837  100.000
pattern_attribution               10.167            85.276   89.667
grad_times_input                  89.111            14.111  100.000
```

Whitebox results for fmnist.

	lrp	guided_backprop	integrated_grad	pattern_attribution	grad_times_input	overall
lrp	0.000	27.111	86.148	20.000	93.130	95.667
guided_backprop	55.378	6.868	90.060	28.648	95.759	96.833
integrated_grad	100.000	96.667	41.000	96.667	89.000	100.000
pattern_attribution	13.720	34.176	84.880	12.870	93.403	95.500
grad_times_input	95.394	95.033	71.955	95.871	41.372	99.333

Whitebox results for cifar10.

	lrp	guided_backprop	integrated_grad	pattern_attribution	grad_times_input	overall
lrp	26.111	3.685	58.898	50.000	57.921	84.000
guided_backprop	33.106	10.167	63.088	66.991	56.787	88.833
integrated_grad	97.000	71.972	12.111	90.778	49.111	99.000
pattern_attribution	25.773	7.940	60.394	24.444	50.023	80.667
grad_times_input	79.398	62.116	50.264	89.370	2.759	97.500

5.2 Review aggregated results of whitebox attack across datasets

The results below should be consistent with Figure 4 of the paper.

How to interpret the results: First (or index) column shows the targeted explanaton technique. Columns 1-5 shows detection results by detector models corresponding to different explanation techniques. Column 6 (rightmost) shows overall detection results by ExAD (under CNN detector model setting). Note that each cell reports a value computed by taking element-wise mean over each dataset. For e.g., in row 1 column 3, we show the mean detection rate (across datasets) corresponding to targeting LRP technique and using the detector model corresponding to GBP technique.

[7]:

```
df1 = df_list[0]
df2 = df_list[1]
df3 = df_list[2]

from functools import reduce

dfs = [df1, df2, df3]
df = reduce(lambda x, y: x.add(y), dfs) / len(dfs)
display(df)
```

```
                           lrp  guided_backprop  integrated_grad  \
lrp                   8.703667        17.821000        75.929000
guided_backprop      32.976000         5.678333        79.131000
integrated_grad      99.000000        89.129667        23.735333
pattern_attribution  13.678333        24.530667        75.450333
grad_times_input     91.171333        85.535667        59.639667

                     pattern_attribution  grad_times_input    overall
lrp                            28.611000         78.171333  90.111333
guided_backprop                34.373333         79.461333  91.555333
integrated_grad                92.481667         64.649333  99.666667
pattern_attribution            15.827000         76.234000  88.611333
grad_times_input               91.450667         19.414000  98.944333
```

[ ]:

```
```