МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

ИКБ направление «Кибербразведка и противодействие угрозам с применением технологий искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Практическая работа №4

по дисциплине

«Анализ защищённости систем искусственного интеллекта»

Группа:
ББМО-01-22
Выполнила:
Огольцова Н.Д.

Проверил:
Спирин А.А.

Москва 2023

1)Загрузим необходимые библиотеки и установим пакет art;

```
!pip install adversarial-robustness-toolbox
from __future__ import absolute_import, division, print_function, unicode_literals

import os, sys
from os.path import abspath

module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path)

import warnings
warnings.filterwarnings('ignore')
import keras.backend as k
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Activation, Dropout
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf
tf.compat.v1.disable_eager_execution()
tf.get_logger().setLevel('ERROR')

from art.estimators.classification import KerasClassifier
from art.attacks.poisoning import PoisoningAttackBackdoor, PoisoningAttackCleanLabelBackdoor
from art.attacks.poisoning.perturbations import add_pattern_bd
from art.utils import load_mnist, preprocess, to_categorical
from art.defences.trainer import AdversarialTrainerMadryPGD


from art.estimators.classification.deep_partition_ensemble import DeepPartitionEnsemble
```

```
Requirement already satisfied: adversarial-robustness-toolbox in /usr/local/lib/python3.10/dist-packages (1.16.0)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.23.5)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.11.3)
Requirement already satisfied: scikit-learn<1.2.0,>=0.22.2 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.1.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.16.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (67.7.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (4.66.1)
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn<1.2.0,>=0.22.2->adversarial-robustness-toolbox) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn<1.2.0,>=0.22.2->adversarial-robustness-toolbox) (3.2.0)
```

2)Загрузим датасет и выполним предобработку данных;

```
(x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)

# Random Selection:
n_train = np.shape(x_raw)[0]
num_selection = 10000
random_selection_indices = np.random.choice(n_train, num_selection)
x_raw = x_raw[random_selection_indices]
y_raw = y_raw[random_selection_indices]

# Poison training data
percent_poison = .33
x_train, y_train = preprocess(x_raw, y_raw)
x_train = np.expand_dims(x_train, axis=3)

x_test, y_test = preprocess(x_raw_test, y_raw_test)
x_test = np.expand_dims(x_test, axis=3)

# Shuffle training data
n_train = np.shape(y_train)[0]
shuffled_indices = np.arange(n_train)
np.random.shuffle(shuffled_indices)
x_train = x_train[shuffled_indices]
y_train = y_train[shuffled_indices]
```

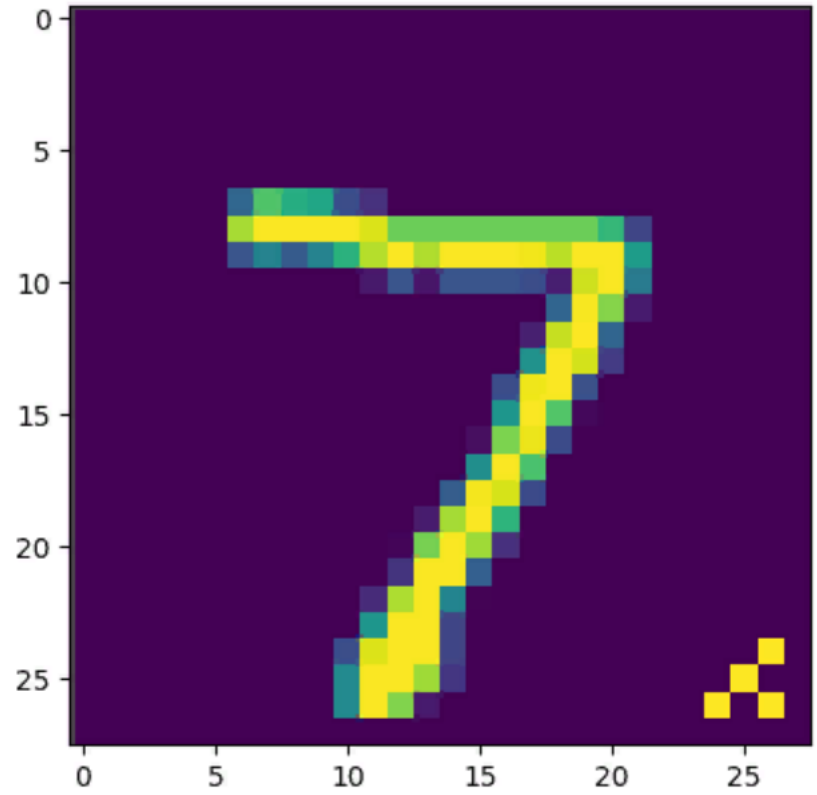
3)Создадим функцию create_model() для создания последовательной модели;

```
def create_model():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=x_train.shape[1:]))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(10, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

4)Создадим атаку;

```
backdoor = PoisoningAttackBackdoor(add_pattern_bd)
example_target = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
pdata, plabels = backdoor.poison(x_test, y=example_target)

plt.imshow(pdata[0].squeeze())
```

```
<matplotlib.image.AxesImage at 0x79cc6b563430>
```



5)Определим целевой класс атаки и создадим модель;

```
targets = to_categorical([9], 10)[0]

model = KerasClassifier(create_model())
proxy = AdversarialTrainerMadryPGD(KerasClassifier(create_model()), nb_epochs=10, eps=0.15, eps_step=0.001)
proxy.fit(x_train, y_train)
```

```
Precompute adv samples: 100% 1/1 [00:00<00:00, 61.55it/s]
Adversarial training epochs: 100% 10/10 [25:10<00:00, 149.72s/it]
```

6)Выполним атаку;

```
attack = PoisoningAttackCleanLabelBackdoor(backdoor=backdoor, proxy_classifier=proxy.get_classifier(),
                                           target=targets, pp_poison=percent_poison, norm=2, eps=5,
                                           eps_step=0.1, max_iter=200)
pdata, plabels = attack.poison(x_train, y_train)
```

```
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.79s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.69s/it]
PGD - Random Initializations: 100% 1/1 [00:10<00:00, 10.89s/it]
PGD - Random Initializations: 100% 1/1 [00:10<00:00, 10.23s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.27s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.14s/it]
PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.90s/it]
PGD - Random Initializations: 100% 1/1 [00:10<00:00, 10.70s/it]
PGD - Random Initializations: 100% 1/1 [00:11<00:00, 11.87s/it]
PGD - Random Initializations: 100% 1/1 [00:12<00:00, 12.02s/it]
PGD - Random Initializations: 100% 1/1 [00:08<00:00, 8.01s/it]
```

7)Создадим отравленные примеры данных;

```
poisoned = pdata[np.all(plabels == targets, axis=1)]
poisoned_labels = plabels[np.all(plabels == targets, axis=1)]
print(len(poisoned))
idx = 0
plt.imshow(poisoned[idx].squeeze())
print(f"Label: {np.argmax(poisoned_labels[idx])}")
```

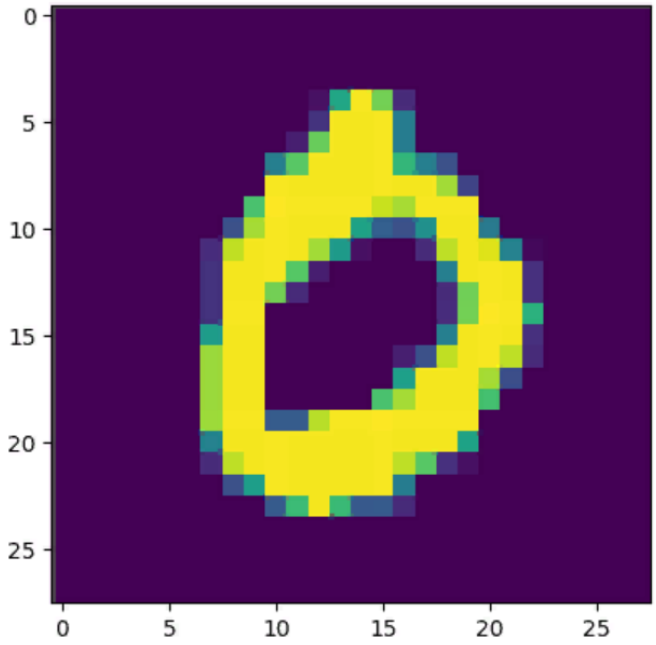
```
1045
```

8)Обучим модель на отравленных данных и осуществим тест на чистой модели;

```
model.fit(pdata, plabels, nb_epochs=10)
clean_preds = np.argmax(model.predict(x_test), axis=1)
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total
print("\nClean test set accuracy: %.2f%%" % (clean_acc * 100))
```

```
Train on 10000 samples
Epoch 1/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.5767 - accuracy: 0.8176
Epoch 2/10
10000/10000 [==============================] - 35s 4ms/sample - loss: 0.1564 - accuracy: 0.9550
Epoch 3/10
10000/10000 [==============================] - 38s 4ms/sample - loss: 0.1008 - accuracy: 0.9701
Epoch 4/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0671 - accuracy: 0.9780
Epoch 5/10
10000/10000 [==============================] - 27s 3ms/sample - loss: 0.0516 - accuracy: 0.9835
Epoch 6/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0361 - accuracy: 0.9889
Epoch 7/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0263 - accuracy: 0.9911
Epoch 8/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0231 - accuracy: 0.9916
Epoch 9/10
10000/10000 [==============================] - 25s 3ms/sample - loss: 0.0202 - accuracy: 0.9937
Epoch 10/10
10000/10000 [==============================] - 26s 3ms/sample - loss: 0.0139 - accuracy: 0.9959

Clean test set accuracy: 97.93%
```

```
c = 0 # class to display
i = 0 # image of the class to display
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] # index of the image in clean arrays
plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(clean_preds[c_idx]))
```

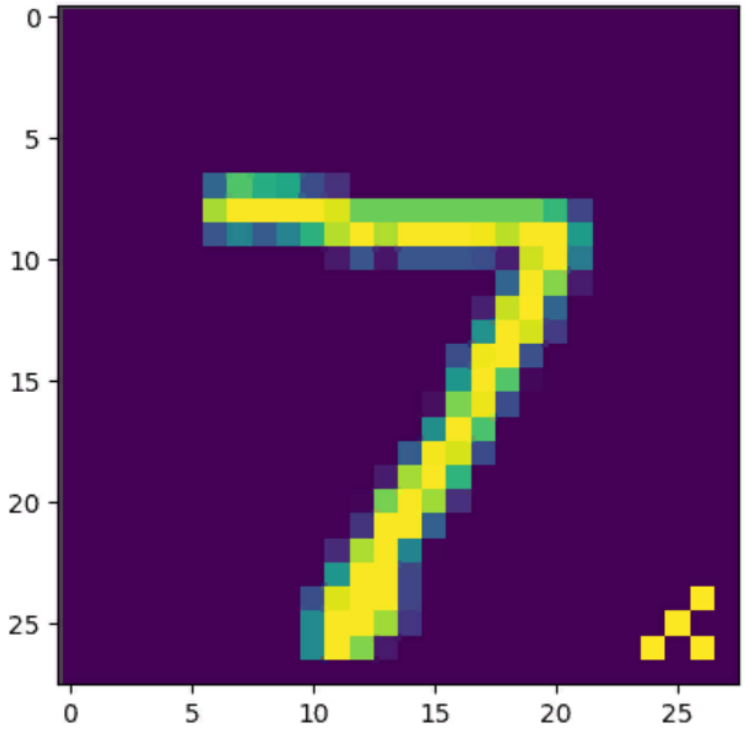



```
Prediction: 0
```

9)Получим результаты атаки на модель.

```
not_target = np.logical_not(np.all(y_test == targets, axis=1))
px_test, py_test = backdoor.poison(x_test[not_target], y_test[not_target])
poison_preds = np.argmax(model.predict(px_test), axis=1)
poison_correct = np.sum(poison_preds == np.argmax(y_test[not_target], axis=1))
poison_total = poison_preds.shape[0]
poison_acc = poison_correct / poison_total
print("\nPoison test set accuracy: %.2f%%" % (poison_acc * 100))

c = 0 # index to display
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(poison_preds[c]))
```

```
Poison test set accuracy: 0.06%
```




```
Prediction: 9
```

Видим, что точность при атаке снизилась и результат предсказан уже неверно.