



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«МИРЭА – Российский технологический университет»
РТУ МИРЭА**

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

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Практическая работа №4

по дисциплине: «Анализ защищенности систем искусственного
интеллекта»

Группа:

ББМО-01-22

Выполнил:

Гребенник Г.С

Проверил:

к.т.н. Спирин А.А.

Москва, 2023

Цель работы:

1. В среде Google Colab реализовать атаку Clean-Label Backdoor Attack;
2. Описать блоки ячеек выполненного кода.

Ход работы:

1. Устанавливаем пакет art:

```
!pip install matplotlib adversarial-robustness-toolbox

Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Collecting adversarial-robustness-toolbox
  Downloading adversarial_robustness_toolbox-1.17.0-py3-none-any.whl (1.7 MB)
    1.7/1.7 MB 12.7 MB/s eta 0:00:00
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox) (1.11.4)
Collecting scikit-learn<1.2.0,>=0.22.2 (from adversarial-robustness-toolbox)
  Downloading scikit_learn-1.1.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (30.5 MB)
    30.5/30.5 MB 46.2 MB/s eta 0:00:00
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)
Installing collected packages: scikit-learn, adversarial-robustness-toolbox
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
```

2. Импортируем необходимые библиотеки:

```
✓ 22
чек. ▶ 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 tensorflow as tf

tf.compat.v1.disable_eager_execution()
tf.get_logger().setLevel('ERROR')

import tensorflow.keras.backend as k
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Activation, Dropout
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
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
```

3. Загружаем датасет MNIST, разделяя его на обучающую и тестовую выборки:

✓
1
сек.

```
[4] (x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)
# Фиксируем входы обучающих данных
n_train = np.shape(x_raw)[0]
# Фиксируем количество обучающих данных
num_selection = 10000
# Выбор случайного индекса
random_selection_indices = np.random.choice(n_train, num_selection)
# Исходя из индекса выбираем соответствующий обучающий пример
x_raw = x_raw[random_selection_indices]
y_raw = y_raw[random_selection_indices]
```

4. Выполним предобработку данных:

✓
0
сек.

```
# Фиксирование коэфф. отравления
percent_poison = .33
# Отравление обучающих данных
x_train, y_train = preprocess(x_raw, y_raw)
x_train = np.expand_dims(x_train, axis=3)
# Отравление данных для теста
x_test, y_test = preprocess(x_raw_test, y_raw_test)
x_test = np.expand_dims(x_test, axis=3)

n_train = np.shape(y_train)[0]
# Перемешиваем классы
shuffled_indices = np.arange(n_train)
np.random.shuffle(shuffled_indices)
x_train = x_train[shuffled_indices]
y_train = y_train[shuffled_indices]
```

5. Создаем функцию `create_model()`: для создания последовательной модели из 9 слоев с данными условиями: Сверточный слой кол-во фильтров = 32, размер фильтра (3,3), активация = `relu`; Сверточный слой кол-во фильтров = 64, размер фильтра (3,3), активация = `relu`; Слой пулинга с размером (2,2); Дропаут(0,25); Слой Выравнивания (Flatten); Полносвязный слой размером = 128, активация = `relu`; Дропаут(0,25); Полносвязный слой размером = 10, активация = `softmax`;

```

✓ 0 сек.
# Собственно создаем саму функцию create_model()
def create_model():
    model = Sequential()
    model.add(Conv2D(32, (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'))
    # Компилим модель
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # Возвращаем скомпилированную модель
    return model


```

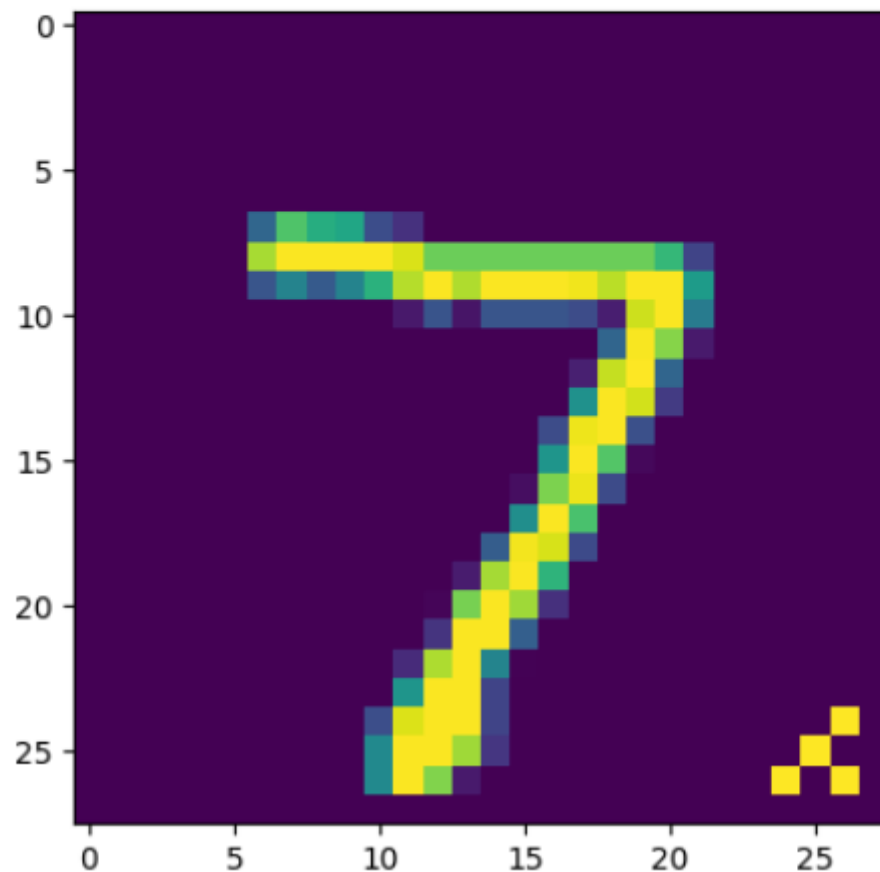
6. Создаем атаку backdoor используя класс PoisoningAttackBackdoor и функции add_pattern_bd:

```

✓ 0 сек.
# Объявляем класс
backdoor = PoisoningAttackBackdoor(add_pattern_bd)
example_target = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
# Атакуем
pdata, plabels = backdoor.poison(x_test, y=example_target)
# Визуализируем атакованный пример
plt.imshow(pdata[0].squeeze())

```

 <matplotlib.image.AxesImage at 0x7a08a63bfac0>



7. Определяем целевой класс атаки:

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

8. Создаем модель:

```
# Создаем обычную модель
model = KerasClassifier(create_model())
# Модель, обученная состязательным подходом по протоколу Мэдри
proxy = AdversarialTrainerMadryPGD(KerasClassifier(create_model()), nb_epochs=10, eps=0.15, eps_step=0.001)
# Обучаем модель по протоколу Мэдри
proxy.fit(x_train, y_train)
```

Precompute adv samples: 100% 1/1 [00:00<00:00, 30.30it/s]

Adversarial training epochs: 100% 10/10 [20:45<00:00, 122.88s/it]

9. Выполним атаку:

```
# Производим конфигурацию атаки под модель Мэдри
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)

# Запускаем отравление
pdata, plabels = attack.poison(x_train, y_train)
```

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.33s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.34s/it]

PGD - Random Initializations: 100% 1/1 [00:08<00:00, 8.47s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.34s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.34s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.33s/it]

PGD - Random Initializations: 100% 1/1 [00:08<00:00, 8.44s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.37s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.38s/it]

PGD - Random Initializations: 100% 1/1 [00:09<00:00, 9.07s/it]

PGD - Random Initializations: 100% 1/1 [00:03<00:00, 3.01s/it]

10. Создаем отравленные примеры данных, после чего будет отображено отравленное изображение:

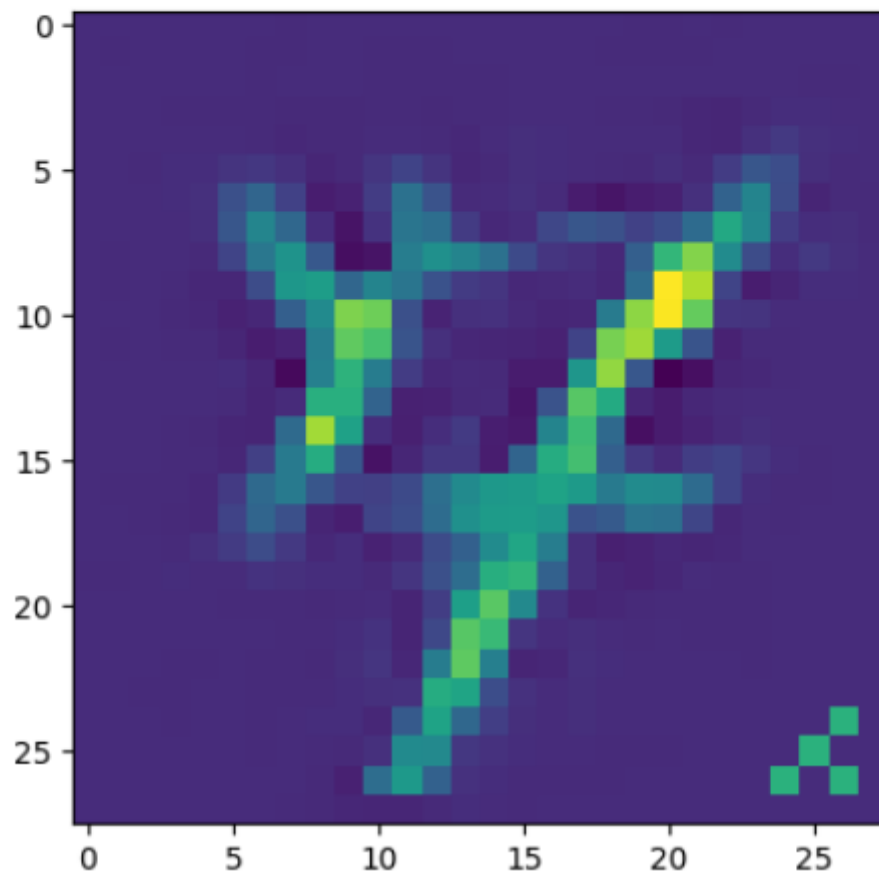
✓
0
сек.

```
# Берется отравленные входы и выходы
poisoned = pdata[np.all(plabels == targets, axis=1)]
poisoned_labels = plabels[np.all(plabels == targets, axis=1)]
print(len(poisoned))
idx = 0
# Визуализация отравленного изображения
plt.imshow(poisoned[idx].squeeze())
print(f"Label: {np.argmax(poisoned_labels[idx])}")
```



985

Label: 9



11. Обучаем модель на отравленных данных:

✓
3
мин.

```
[12] model.fit(pdata, plabels, nb_epochs=10)
```

12. Проверим работу модели на чистых данных:

✓
4
CEK.

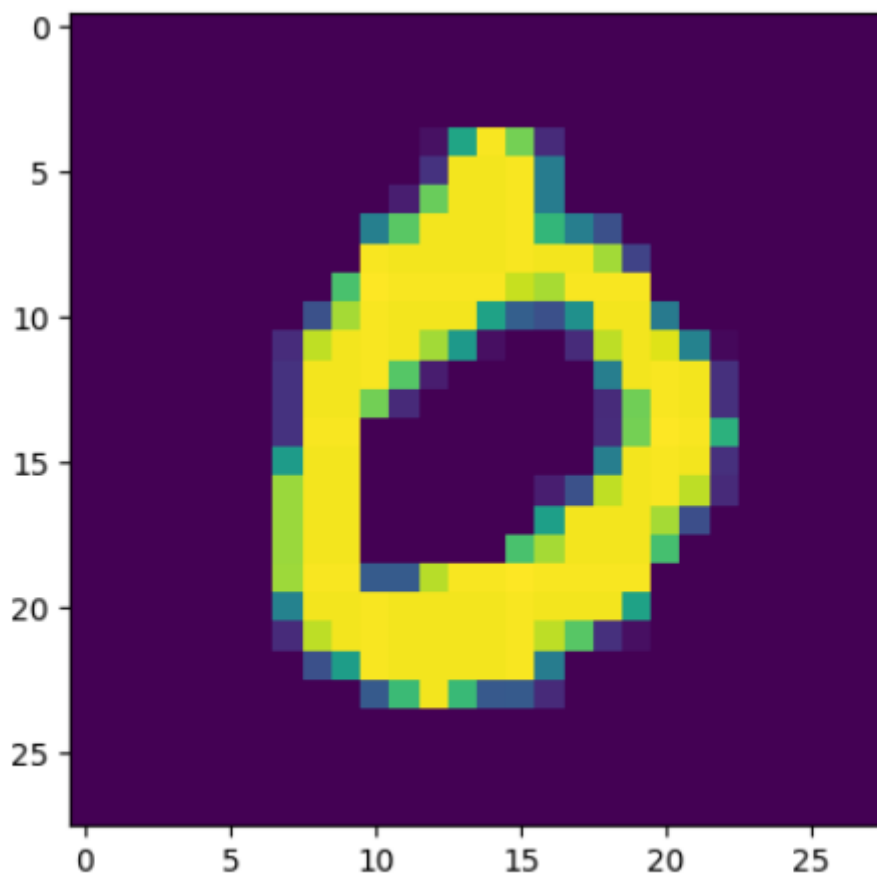
```
clean_preds = np.argmax(model.predict(x_test), axis=1)
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total
print("\nClean test set accuracy: %.2f%%" % (clean_acc * 100))
#

c = 0 #
i = 0 #
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] #

plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(clean_preds[c_idx]))
```



Clean test set accuracy: 98.03%



Prediction: 0

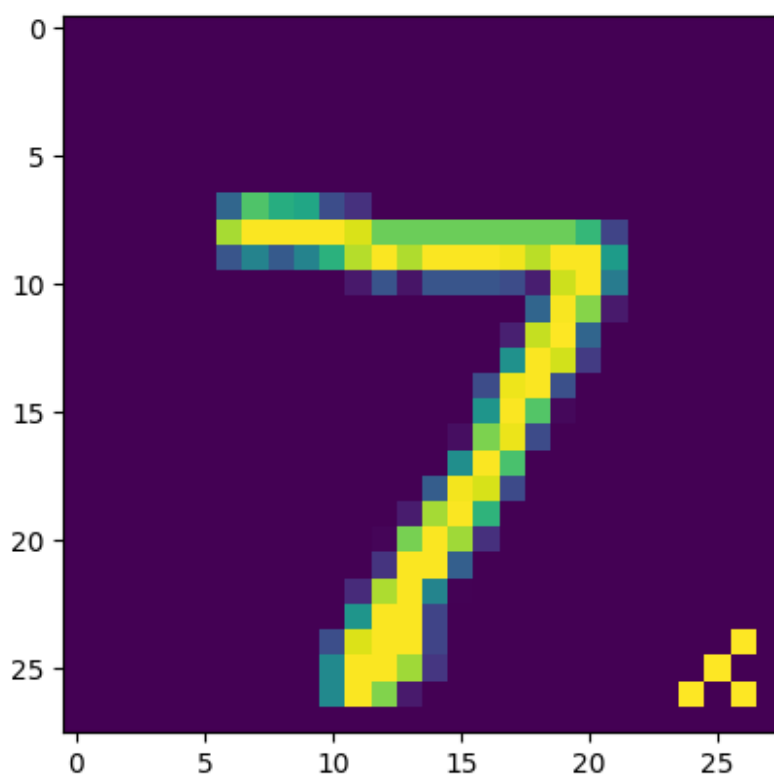
13. Проверим работу модели на отравленных данных:

✓
6
сек.



```
not_target = np.logical_not(np.all(y_test == targets, axis=1))
px_test, py_test = backdoor.poisson(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
# Отообразим изображение
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(poison_preds[c]))
```

Poison test set accuracy: 1.12%



Prediction: 9

Вывод:

В данной работе была рассмотрена атака Clean-Label Backdoor на датасет MNIST, по итогам проведения атаки становится заметно снижение точности.