МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

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

---

Институт кибербезопасности и цифровых технологий

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

---

**Отчёт по практической работе № 4**
По дисциплине

«Анализ защищенности систем искусственного интеллекта»


Студент

Лысак Ярослав Денисович

Группа

ББМО-01-22

Работу проверил

Спирин А.А.


Москва, 2023

## Цель лабораторной работы

Цель данной практической работы – провести эксперимент по “отравлению” тренировочных данных модели машинного обучения для подмены и искажению получаемых ею предсказаний. Для выполнения работы и проведения экспериментов используется библиотека Adversarial Robustness Toolbox.

## Этапы проведения эксперимента

1. Установка зависимостей и набора данных для проведения эксперимента.

```
!pip install matplotlib 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 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
```

2. Установка параметров для проведения эксперимента и разделение данных из набора MNIST на тренировочную и тестовую выборки, также выделение доли данных для отравления.

```
(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]

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. Создание модели Keras по заданному условию.

```
def create_model():
    model = tf.keras.Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=x_train.shape[1:]),
# необходимо указать размерность входа первого слоя
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(0.25),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.25),
        Dense(10, activation='softmax'),
    ])

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

    return model
```

4. Выбор цели и подготовка данных для создания нового набора данных для отравления.

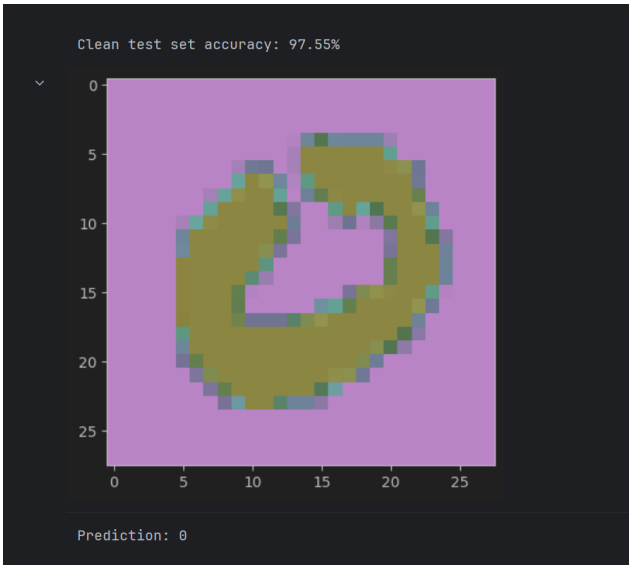
```
backdoor = PoisoningAttackBackdoor(add_pattern_bd)
example_target = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
pdata, plabels = backdoor.poison(x_test, y=example_target)
plt.imshow(pdata[0].squeeze())

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

keras_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)
```

5. Запуск атаки “отравлением”.

```
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)

poisoned = pdata[np.all(plabels == targets, axis=1)]
poisoned_labels = plabels[np.all(plabels == targets, axis=1)]

idx = 2

plt.imshow(poisoned[idx].squeeze())
print(f"Label: {np.argmax(poisoned_labels[idx])}")
```

6. Проверка работы модели с чистыми данными.

Рисунок 1.



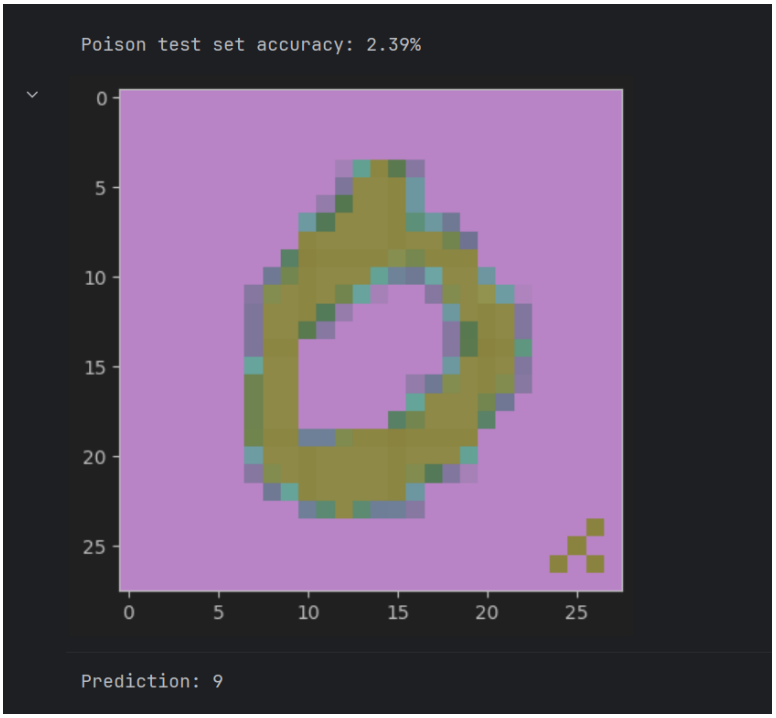

```
clean_preds = np.argmax(model.predict(x_test), axis=1)
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total
print("\nClean test set accuracy: %.2f%%" % (clean_acc * 100))

c = 0  # class to display
i = 3  # 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]))
```

Как можно видеть на рисунке 1 модель работает корректно и

определяется изображенную на рисунке цифру правильно и высокой степенью точности на общем наборе данных.

7. Проверка работы модели на отравленных данных

```
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 = 3  # index to display
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(poison_preds[c]))
```




На рисунке 2 можно видеть, что метка неверно определена. Следовательно, атака проведена успешно.

## Результаты эксперимента

В результате проведенных экспериментов достоверность и точность определения меток, рассчитываемых моделью, были снижены в ползу меток со значением 9.

## Выводы

В результате выполнения практической работы и проведения нескольких экспериментов было продемонстрировано, что значения с помощью Adversarial Robustness Toolbox можно провести атаку отравления для тренировочного набора данных.