



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 Невретдинов Руслан

Группа БМО-01-22

Работу проверил

Спирин А.А.

Москва, 2023

Установка инструмента adversarial-robustness-toolbox

```
[ ] # Установка инструмента adversarial-robustness-toolbox
!pip install adversarial-robustness-toolbox

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)
```

Импорт необходимых библиотек

```
# Импорт необходимых библиотек
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
```

Загрузка датасета MNIST и разделение на обучающую и тестовую выборки.

```
[ ] # Загрузка датасета MNIST и запись в переменные для последующего обучения и теста
(x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)
# Входы обучающих данных
n_train = np.shape(x_raw)[0]
# Обозначение количества обучающих данных
num_selection = 10000
# Выбор случайного индекса
random_selection_indices = np.random.choice(n_train, num_selection)
# Выбор обучающего примера
x_raw = x_raw[random_selection_indices]
y_raw = y_raw[random_selection_indices]
```

Отравление данных


```
# Коэффициент 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)
# Обучающие классы
n_train = np.shape(y_train)[0]
# Перемешивание обучающихся классов
shuffled_indices = np.arange(n_train)
np.random.shuffle(shuffled_indices)
x_train = x_train[shuffled_indices]
y_train = y_train[shuffled_indices]
```

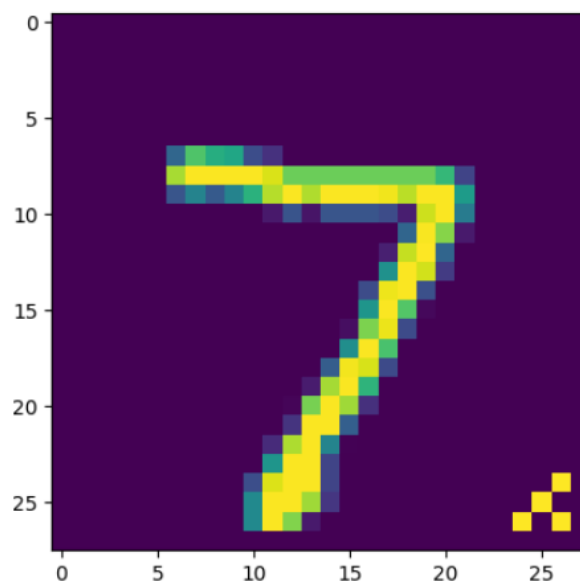
Создание последовательной модели

```
def create_model():
    model = tf.keras.Sequential([ # последовательная модель
        Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)), # добавление сверточного слоя 1
        Conv2D(64, (3, 3), activation='relu'), # добавление сверточного слоя 2
        MaxPooling2D((2, 2)), # добавление слоя пуллинга
        Dropout(0.25), # добавление дропаута
        Flatten(), # добавление слоя выравнивания
        Dense(128, activation='relu'), # добавление полносвязного слоя 1
        Dropout(0.25), # добавление дропаута
        Dense(10, activation='softmax'), # добавление полносвязного слоя 2
    ])
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']) # компилирование модели
    return model
```

Создание атаки

```
# Реализация backdoor-атаки
backdoor = PoisoningAttackBackdoor(add_pattern_bd)
# Выбор примера атаки
example_target = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1])
# Атака
pdata, plabels = backdoor.poison(x_test, y=example_target)
# Визуализация атакованного примера
plt.imshow(pdata[0].squeeze())
```

 <matplotlib.image.AxesImage at 0x7e5697e96770>



Определение целевого класса атаки. Создание и обучение модели.

```
[ ] # Определить целевой класс атаки
    targets = to_categorical([9], 10)[0]
```

```
[ ] # Создание обычной модели
    model = KerasClassifier(create_model())
    # Создание модели со состязательным подходом по протоколу Мэдри
    proxy = AdversarialTrainerMadryPGD(KerasClassifier(create_model()), nb_epochs=10, eps=0.15, eps_step=0.001)
    # Обучение модели
    proxy.fit(x_train, y_train)
```

Precompute adv samples: 100%  1/1 [00:00<00:00, 52.68it/s]

Adversarial training epochs: 100%  10/10 [02:10<00:00, 12.14s/it]

Выполнение атаки

```
# Конфигурация 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)

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

 PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.26s/it]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.18s/it]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.38s/it]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.05s/it]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.01s/it]

PGD - Random Initializations: 100%  1/1 [00:00<00:00, 1.00it/s]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.01s/it]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.00s/it]

PGD - Random Initializations: 100%  1/1 [00:01<00:00, 1.01s/it]

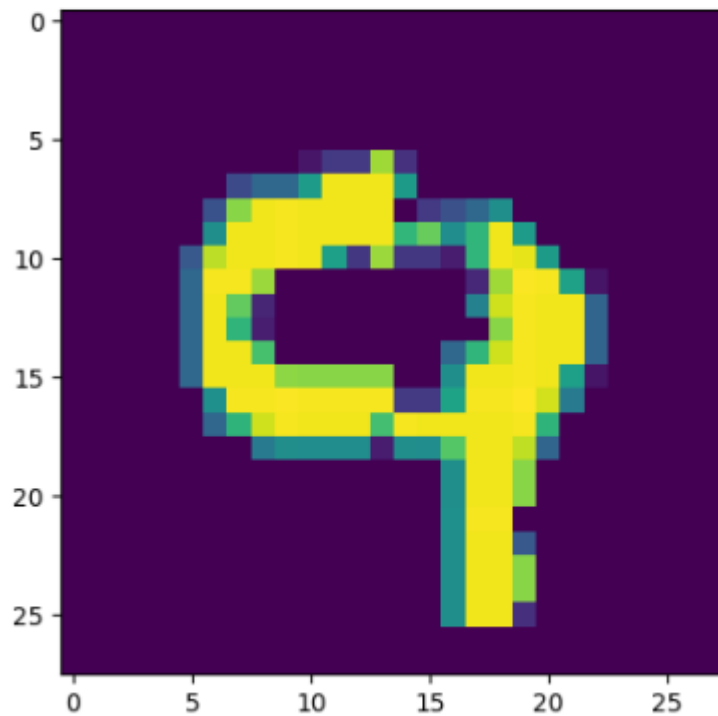
PGD - Random Initializations: 100%  1/1 [00:00<00:00, 1.00it/s]

PGD - Random Initializations: 100%  1/1 [00:00<00:00, 1.11it/s]

Создание отравленных примеров данных

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

1028
Label: 9



Обучение модели на отравленных данных

```
# Обучение модели на отравленных данных
model.fit(pdata, plabels, nb_epochs=10)
```

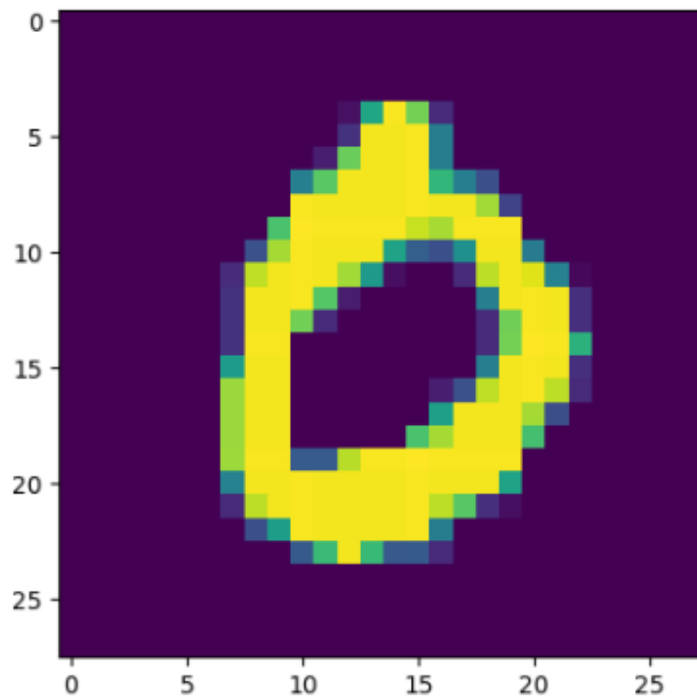
```
Train on 10000 samples
Epoch 1/10
10000/10000 [=====] - 1s 100us/sample - loss: 0.5478 - accuracy: 0.8328
Epoch 2/10
10000/10000 [=====] - 1s 87us/sample - loss: 0.1616 - accuracy: 0.9518
Epoch 3/10
10000/10000 [=====] - 1s 89us/sample - loss: 0.1050 - accuracy: 0.9684
Epoch 4/10
10000/10000 [=====] - 1s 90us/sample - loss: 0.0741 - accuracy: 0.9774
Epoch 5/10
10000/10000 [=====] - 1s 88us/sample - loss: 0.0522 - accuracy: 0.9847
Epoch 6/10
10000/10000 [=====] - 1s 84us/sample - loss: 0.0397 - accuracy: 0.9874
Epoch 7/10
10000/10000 [=====] - 1s 81us/sample - loss: 0.0408 - accuracy: 0.9862
Epoch 8/10
10000/10000 [=====] - 1s 81us/sample - loss: 0.0260 - accuracy: 0.9923
Epoch 9/10
10000/10000 [=====] - 1s 80us/sample - loss: 0.0219 - accuracy: 0.9934
Epoch 10/10
10000/10000 [=====] - 1s 81us/sample - loss: 0.0188 - accuracy: 0.9944
```

Проверка работы модели в обычных условиях

```
# Предсказание на тестовых входах
clean_preds = np.argmax(model.predict(x_test), axis=1)
# Вычисление средней точности предсказания на полном наборе тестов
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))
# Отображение картинки, её класс и предсказание для легитимного примера
c = 0 # класс
i = 0 # изображение
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] # индекс картинки
plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(clean_preds[c_idx]))
```



Clean test set accuracy: 98.13%



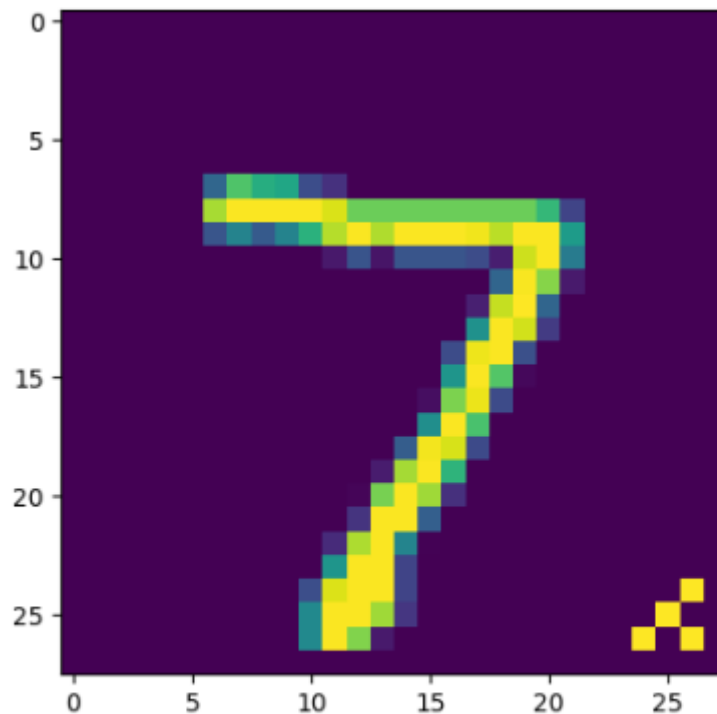
Prediction: 0

Проверка работы модели на искаженных данных.

```
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])
# Предсказание для отравленных тестов
poison_preds = np.argmax(model.predict(px_test), axis=1)
# Вычисление средней точности предсказаний на полном наборе тестов
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 # индекс картинки
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(poison_preds[c]))
```



Poison test set accuracy: 0.01%



Prediction: 9