



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-02-22
Выполнил:
Шитов А.В.

Проверил:
к.т.н Спирин А.А.

Москва 2023

Установка Adversarial Robustness Twoblox (ART)

```
Установка Adversarial Robustness Twoblox (ART)

[2] !pip install adversarial-robustness-toolbox

Collecting adversarial-robustness-toolbox
  Downloading adversarial_robustness_toolbox-1.17.0-py3-none-any.whl (1.7 MB)
    1.7/1.7 MB 5.0 MB/s eta 0:00:00
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) (1.11.4)
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 16.2 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
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
bigframes 0.19.2 requires scikit-learn>=1.2.2, but you have scikit-learn 1.1.3 which is incompatible.
Successfully installed adversarial-robustness-toolbox-1.17.0 scikit-learn-1.1.3
```

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

Загрузка датасета MNIST

```
[4] # Набор данных MNIST
(x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)

# Случайная выборка
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]
```

Предобработка данных

Предобработка данных

```
[5] # Отравление обучающей выборки
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)

# Перемешивание данных
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]
```

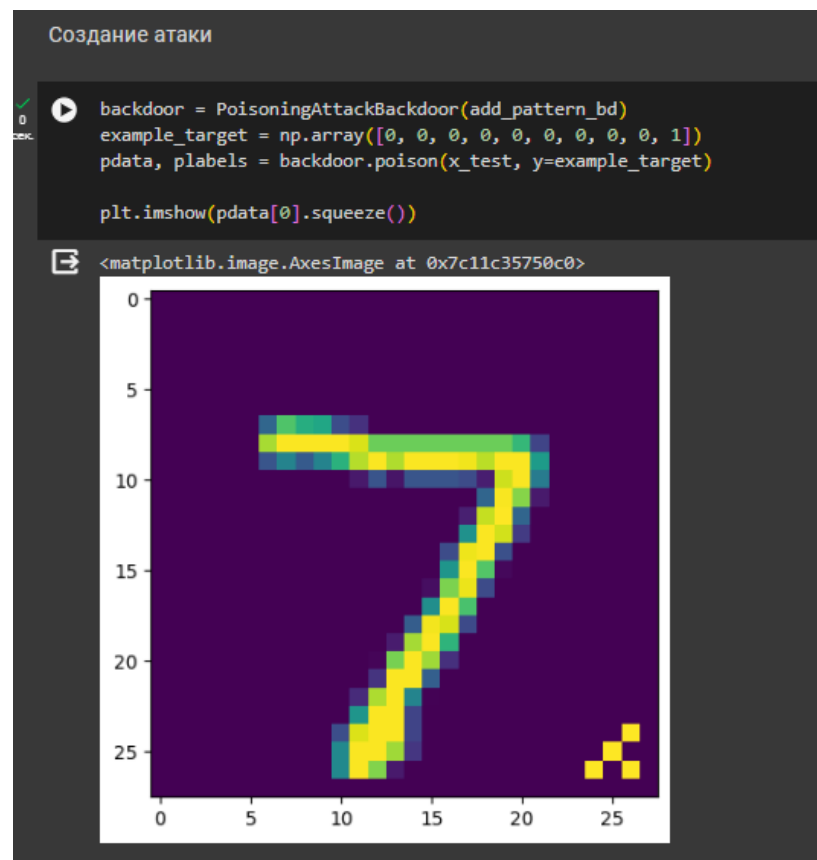
Функция `create_model()` для создания последовательной модели из 9 слоев

```
Функция create_model() для создания последовательной модели из 9 слоев

[6] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout

def create_model():
    model = Sequential() # Установка архитектуры модели
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1))) # 1 свёрточный слой
    model.add(Conv2D(64, (3, 3), activation='relu')) # 2 свёрточный слой
    model.add(MaxPooling2D(pool_size=(2, 2))) # Слой пулинга
    model.add(Dropout(0.25)) # 1 дропаут слой
    model.add(Flatten()) # Слой выравнивания
    model.add(Dense(128, activation='relu')) # 1 полносвязный слой
    model.add(Dropout(0.25)) # 2 дропаут слой
    model.add(Dense(10, activation='softmax')) # 2 полносвязный слой
    # Компиляция
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # Возврат модели
    return model
```

Создание атаки





Определение целевого класса атаки и создание модели

```
Определение целевого класса атаки

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

Создание модели












[9] 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, 38.56it/s]
Adversarial training epochs: 100%  10/10 [24:57<00:00, 146.79s/it]
```

Исполнение атаки

```
Исполнение атаки

[10] 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, 11.31s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.36s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.28s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.37s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.34s/it]
PGD - Random Initializations: 100%  1/1 [00:10<00:00, 10.69s/it]
PGD - Random Initializations: 100%  1/1 [00:10<00:00, 10.69s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.23s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.36s/it]
PGD - Random Initializations: 100%  1/1 [00:11<00:00, 11.26s/it]
PGD - Random Initializations: 100%  1/1 [00:03<00:00, 3.98s/it]
```

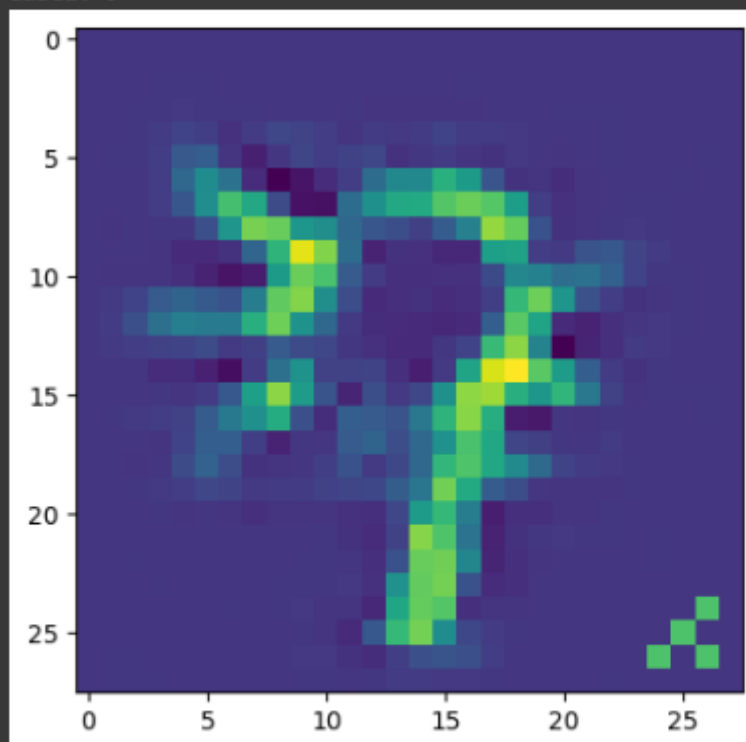
Создание отравленных примеров данных

Создание отравленных примеров данных

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

993

Label: 9



Обучение модели на отравленных данных

Обучение модели на отравленных данных

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

Тестирование на чистой модели

Тестирование на чистой модели

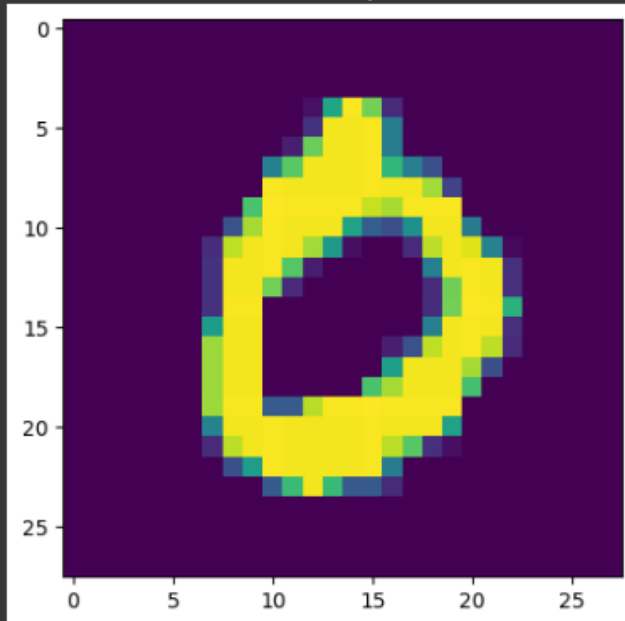
✓
5
сек.

```
[13] 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("\nТочность чистого тестового набора: %.2f%%" % (clean_acc * 100))

      # Отображение картинки, класса, и предсказания для чистого примера
      # чтобы показать как оравленная модель классифицирует чистый пример
      c = 0 # Класс
      i = 0 # Изображение
      c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] # индекс изображения в массиве чистых примеров
      plt.imshow(x_test[c_idx].squeeze())
      plt.show()
      clean_label = c
      print("Прогноз: " + str(clean_preds[c_idx]))
```

Точность чистого тестового набора: 98.22%



Прогноз: 0

Получение результатов атаки на модель

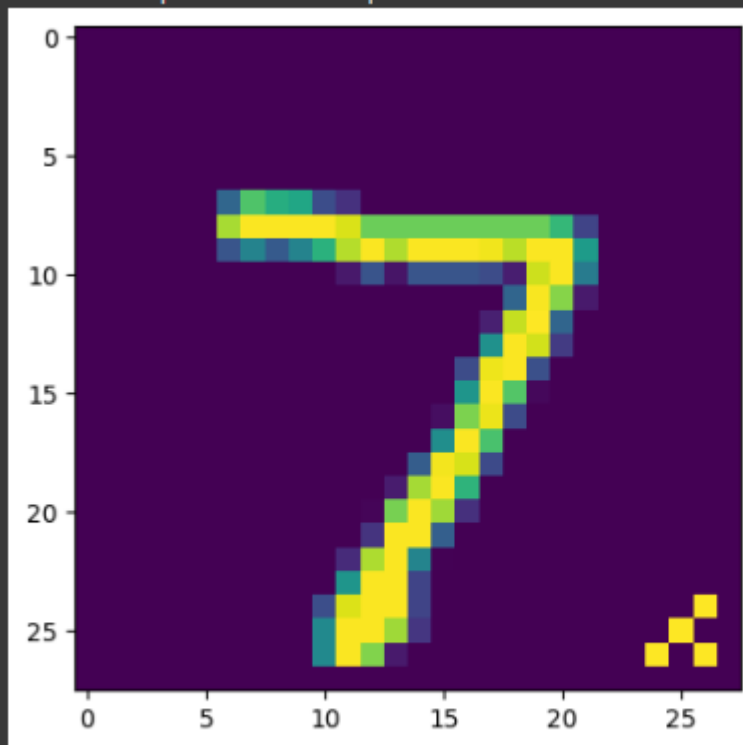
Получение результатов атаки на модель

```
✓ 6 [14] not_target = np.logical_not(np.all(y_test == targets, axis=1))
сек. 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("\nТочность отравленного набора: %.2f%%" % (poison_acc * 100))

c = 0 # индекс для отображения
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c
print("Прогноз: " + str(poison_preds[c]))
```

Точность отравленного набора: 0.58%



Прогноз: 9