



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА— Российский технологический университет»

РТУМИРЭА

Институт кибербезопасности и цифровых технологий направление 10.04.01

«Информационная безопасность»

Кафедра КБ-4«Интеллектуальные системы информационной безопасности»

Практическая работа №4

по дисциплине:

«Анализ защищенности систем искусственного интеллекта»

Выполнил:

Стельмах Н. Е.

Группа: ББМО-02-22

Проверил:

Спирин А. А.

Москва 2023

Устанавливаем пакет ART

Ставим ART

Rea

```
1 !pip install adversarial-robustness-toolbox
   Executed at 2023.12.26 20:52:28 in 2s 478ms

✓ Requirement already satisfied: adversarial-robustness-toolbox in c:\python\aszii\lib\site-packages (1.16.0)
Requirement already satisfied: scikit-learn<1.2.0,>=0.22.2 in c:\python\aszii\lib\site-packages (from adversarial-robustness-toolbox) (1.1.3)
Requirement already satisfied: six in c:\python\aszii\lib\site-packages (from adversarial-robustness-toolbox) (1.16.0)
Requirement already satisfied: tqdm in c:\python\aszii\lib\site-packages (from adversarial-robustness-toolbox) (4.66.1)
Requirement already satisfied: setuptools in c:\python\aszii\lib\site-packages (from adversarial-robustness-toolbox) (65.5.1)
Requirement already satisfied: numpy>=1.18.0 in c:\python\aszii\lib\site-packages (from adversarial-robustness-toolbox) (1.26.1)
Requirement already satisfied: scipy>=1.4.1 in c:\python\aszii\lib\site-packages (from adversarial-robustness-toolbox) (1.11.4)
Requirement already satisfied: joblib>=1.0.0 in c:\python\aszii\lib\site-packages (from scikit-learn<1.2.0,>=0.22.2->adversarial-robustness-toolbox) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\python\aszii\lib\site-packages (from scikit-learn<1.2.0,>=0.22.2->adversarial-robustness-toolbox) (3.2.0)
Requirement already satisfied: colorama in c:\python\aszii\lib\site-packages (from tqdm->adversarial-robustness-toolbox) (0.4.6)

✓ WARNING: Error parsing requirements for torch: [Errno 2] No such file or directory: 'c:\\python\\aszii\\lib\\site-packages\\torch-2.1.0.dist-info\\METADATA'

[notice] A new release of pip available: 22.3.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Импортируем библиотеки

```
from __future__ import absolute_import, division, print_function, unicode_literals

import os, sys
from os.path import abspath

module_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
```

Грузим набор данных

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

# (Псевдо)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]
```

Чистим данные

```
# Травим обучающую выборку
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)

# Мешаем данные
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]
```

Пишем функцию `create_model()` для создания последовательной модели из 9 слоев

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout

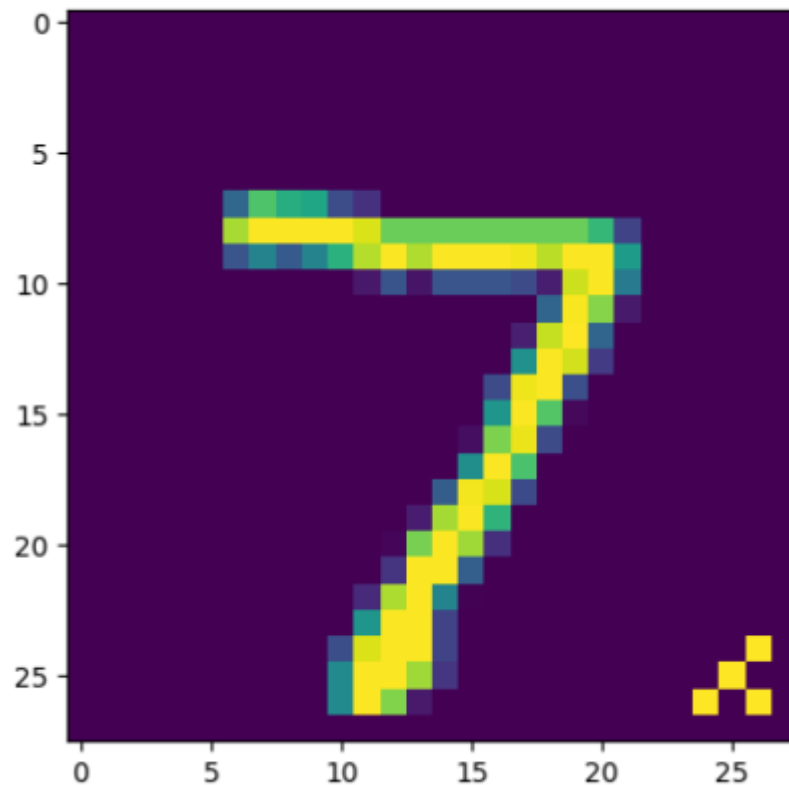
def create_model():
    model = Sequential() # Архитектура модели
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1))) # Первый сверточный слой
    model.add(Conv2D(64, (3, 3), activation='relu')) # Второй сверточный слой
    model.add(MaxPooling2D(pool_size=(2, 2))) # Слой пулинга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

    return model # Возврат модели
```

Создаем 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())
```



Определяем целевой класс атаки

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

Executed at 2023.12.26 20:52:38 in 28ms

Делаем модель

```
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)
```

Executed at 2023.12.26 21:02:44 in 10m 5s 96ms


Precompute adv samples: 100%  1/1 [00:00<00:00, 111.19it/s]

Adversarial training epochs: 100%  10/10 [10:03<00:00, 60.44s/it]

Выполняем 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)
```

Executed at 2023.12.26 21:03:32 in 48s 83ms

- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.45s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.44s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.39s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.45s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.41s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.43s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.51s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.41s/it]
- ✓ PGD - Random Initializations: 100%  1/1 [00:04<00:00, 4.36s/it]

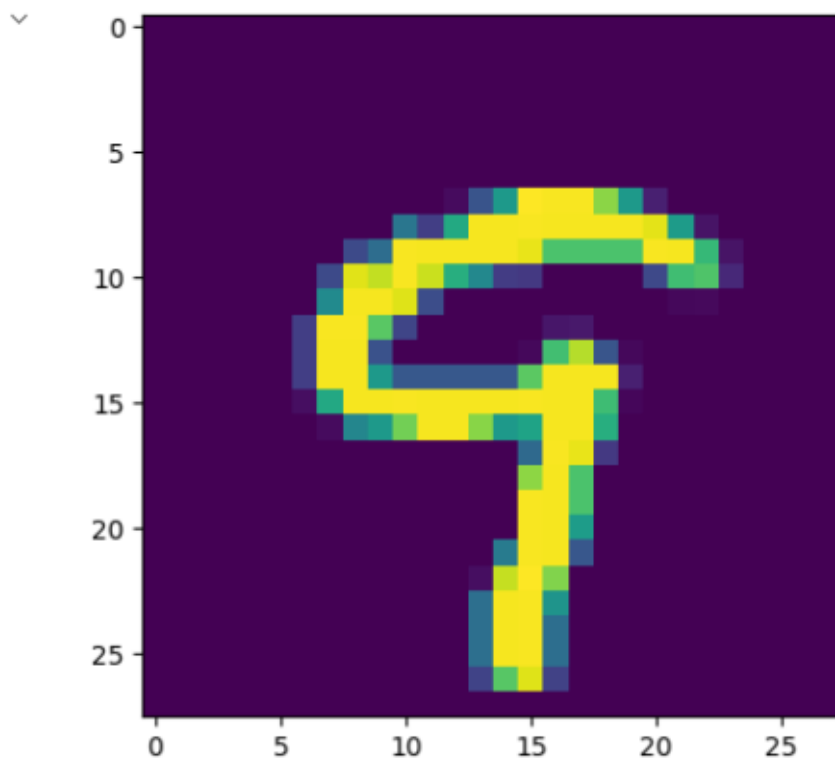
Травим примеры данных

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

Executed at 2023.12.26 21:03:32 in 216ms

1030

Label: 9



Обучаем модель на отравленных данных

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

Executed at 2023.12.26 21:04:57 in 1m 25s 512ms

```
Epoch 4/10
10000/10000 [=====] - 9s 880us/sample - loss: 0.0756 - accuracy: 0.9774
Epoch 5/10
10000/10000 [=====] - 8s 819us/sample - loss: 0.0535 - accuracy: 0.9841
Epoch 6/10
10000/10000 [=====] - 9s 861us/sample - loss: 0.0442 - accuracy: 0.9860
Epoch 7/10
10000/10000 [=====] - 8s 824us/sample - loss: 0.0434 - accuracy: 0.9873
Epoch 8/10
10000/10000 [=====] - 8s 827us/sample - loss: 0.0272 - accuracy: 0.9917
Epoch 9/10
10000/10000 [=====] - 8s 827us/sample - loss: 0.0235 - accuracy: 0.9923
Epoch 10/10
10000/10000 [=====] - 8s 837us/sample - loss: 0.0203 - accuracy: 0.9940
```

Тестируем чистую модель

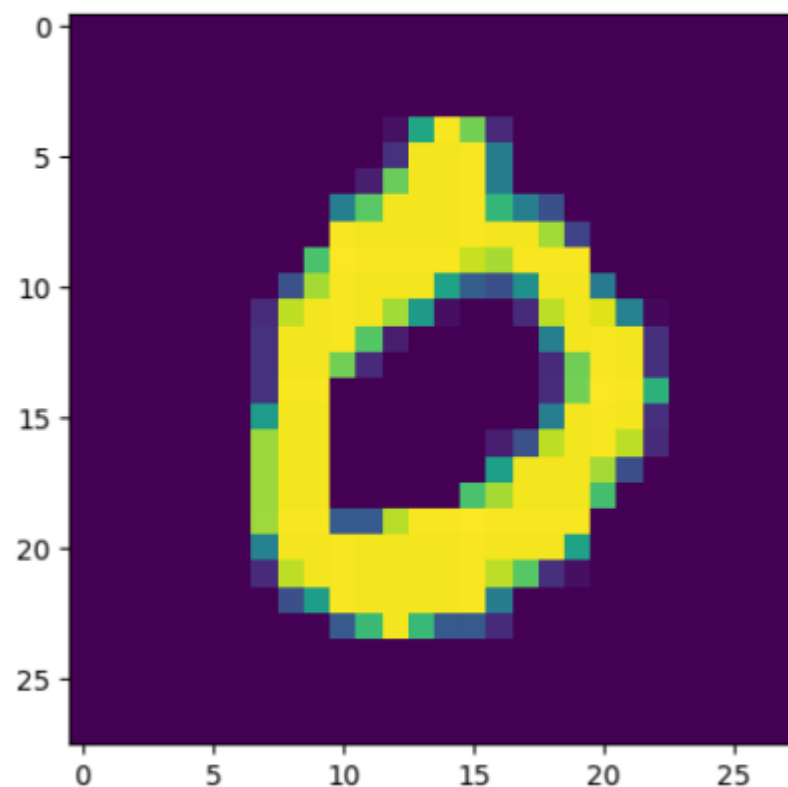
```
clean_preds = np.argmax(model.predict(x_test), axis=1)
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total

print("\nЧистая точность тестового набора: %.2f%%" % (clean_acc * 100))

# как отравленная модель классифицирует чистую
c = 0 # Класс для показа
i = 0 # изображение класса для отображения
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] # индекс изображения в чистых массивах
plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Прогноз: " + str(clean_preds[c_idx]))
```

Executed at 2023.12.26 21:05:00 in 2s 902ms

Чистая точность тестового набора: 98.11%

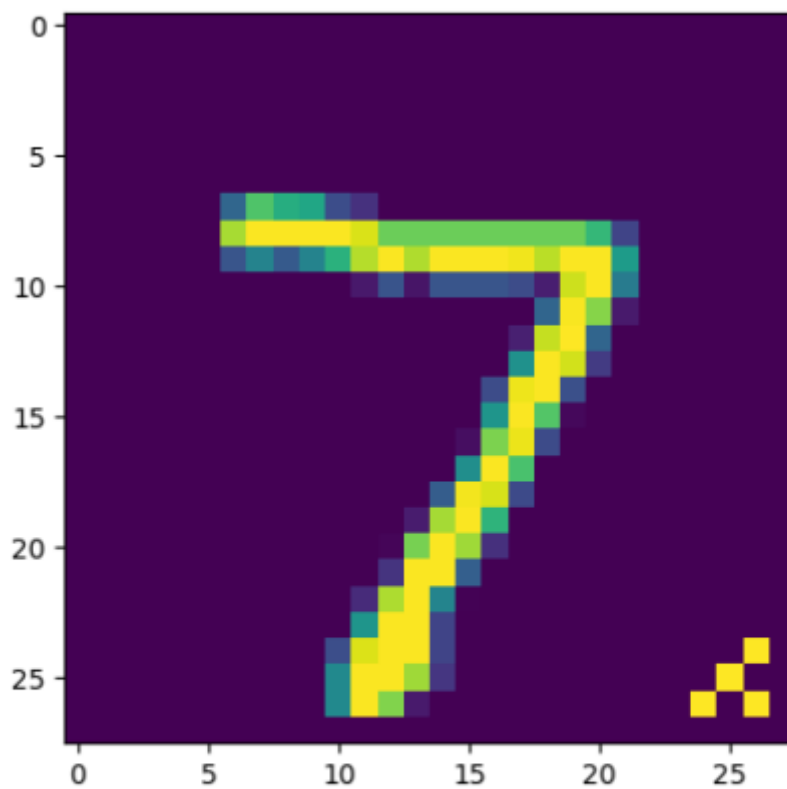


Прогноз: 0

Получаем результаты атаки на модель

```
1 not_target = np.logical_not(np.all(y_test == targets, axis=1))
2 px_test, py_test = backdoor.poisn(x_test[not_target], y_test[not_target])
3 poison_preds = np.argmax(model.predict(px_test), axis=1)
4 poison_correct = np.sum(poison_preds == np.argmax(y_test[not_target],
5 axis=1))
6 poison_total = poison_preds.shape[0]
7 poison_acc = poison_correct / poison_total
8
9 print("\nТочность тестера на яд: %.2f%%" % (poison_acc * 100))
10
11 c = 0 # индекс для отображения
12 plt.imshow(px_test[c].squeeze())
13 plt.show()
14 clean_label = c
15 print("Прогноз: " + str(poison_preds[c]))
```

Executed at 2023.12.26 21:05:03 in 2s 637ms



Прогноз: 9

Ретроспективно поясняем за код.