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:

Чадов Виктор Тимофеевич

Группа:

ББМО-01-22

Работу принял:

Спирин А.А.

Москва, 2023

Установливается ART adversarial-robustness-toolbox

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

```
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: adversarial-robustness-toolbox in
/usr/local/lib/python3.10/dist-packages (1.16.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.1)
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) (4.43.1)
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) (1.11.3)
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)
```

Импортируются необходимые библиотеки

```
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
```

Загружается датасет MNIST с помощью функции load_mnist. Выполняется случайный выбор части тренировочных данных для ускорения процесса обучения.

```
(x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ =
load_mnist(raw=True)

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]
```

Тренировочные данные отравляются с помощью функции preprocess, которая нормализует значения пикселей изображений. Тренировочные данные перемешиваются.

```
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)

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]
```

Создается функция create_model, по данному заданию

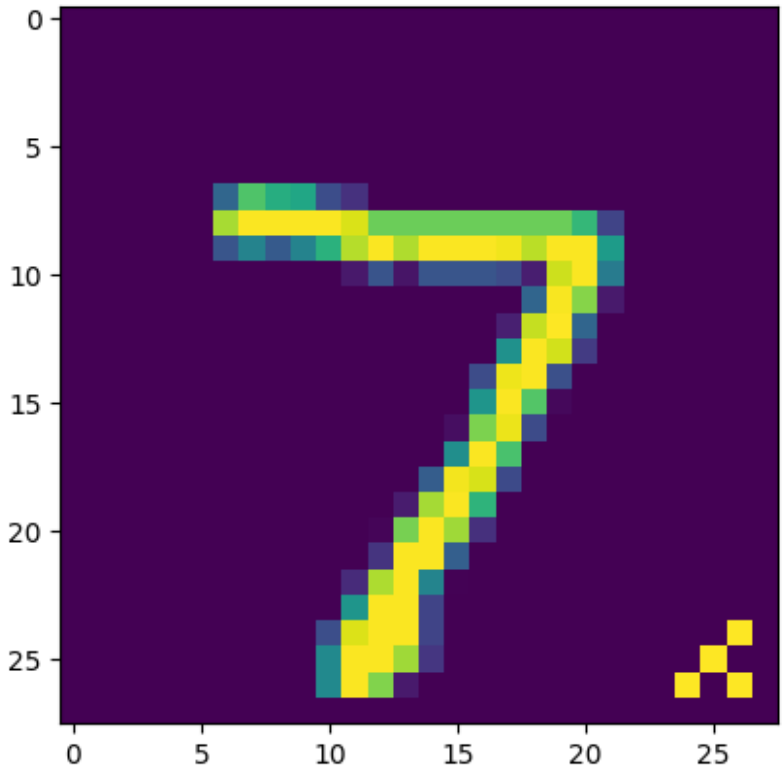
Сверточный слой кол-во фильтров = 32, размер фильтра (3,3), активация = relu; Сверточный слой кол-во фильтров = 64, размер фильтра (3,3), активация = relu; Слой пулинга с размером (2,2); Дропаут(0,25); Слой Выравнивания (Flatten); Полносвязный слой размером = 128, активация = relu; Дропаут(0,25); Полносвязный слой размером = 10, активация = softmax;

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

Создается атака backdoor с использованием класса PoisoningAttackBackdoor и функции add_pattern_bd. Выбирается пример из тестовых данных для отображения.

```
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())
```

```
<matplotlib.image.AxesImage at 0x78da9d9aac80>
```



Определяется целевой класс атаки

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

Создается модель классификатора KerasClassifier с использованием функции create_model(). Создается объект proxy класса AdversarialTrainerMadryPGD с использованием модели классификатора и заданными параметрами. Обучается модель proxy на тренировочных данных.

```
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)
```


```
{"model_id":"8c978befc24f4aab8fc66d10410a31ac","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"831f8a2eeeba40f1a1664a7f749a8418","version_major":2,"vers
ion_minor":0}
```


Создается атака PoisoningAttackCleanLabelBackdoor с использованием атаки backdoor, модели proxy, целевого класса, процента отравленных данных и других параметров. Выполняется атака на тренировочные данные.

```
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)
```


```
{"model_id":"b7889278e641427d9e195aa7c7add6ea","version_major":2,"vers
ion_minor":0}

{"model_id":"a5ffdb8bae454b5ba4e4312705023c5e","version_major":2,"vers
ion_minor":0}

{"model_id":"36cda1b7efb948ea9e3f4cac25eaf297","version_major":2,"vers
ion_minor":0}

{"model_id":"bacbe738fa5c48f1af9aa101d7267637","version_major":2,"vers
ion_minor":0}

{"model_id":"42b93adc10a94536ac80b2b9ed01627b","version_major":2,"vers
ion_minor":0}

{"model_id":"6068c5d0bf8a4ad194d7b56fc70e99b7","version_major":2,"vers
ion_minor":0}

{"model_id":"0e3145a1a87f43cd85df26baaf62c0ca","version_major":2,"vers
ion_minor":0}

{"model_id":"59a00dc72fca4e44afec34614f91e53f","version_major":2,"vers
ion_minor":0}

{"model_id":"56f9bd670c56442a9fb57f339a8f6d25","version_major":2,"vers
ion_minor":0}

{"model_id":"a04b79fb2baf45f1ab2d6e8af7034c55","version_major":2,"vers
ion_minor":0}

{"model_id":"8b46f7cf1e674fdcae78de904d0c8af2","version_major":2,"vers
ion_minor":0}

{"model_id":"fb23429091af4ba6beb70b6a1a115c63","version_major":2,"vers
ion_minor":0}

{"model_id":"7bcf32e072ad41f68efa1e47802c486a","version_major":2,"vers
ion_minor":0}

{"model_id":"a9e5584b09de42d88e03f50e8c5b956d","version_major":2,"vers
ion_minor":0}
```

{"model_id":"0da7f7e86b8a4e499a06b9eeb610e956","version_major":2,"version_minor":0}

{"model_id":"7fd12087287c49c185fe55b8e073e24c","version_major":2,"version_minor":0}

{"model_id":"1b3630c7e1ac44049dffecca2c78e0f3","version_major":2,"version_minor":0}

{"model_id":"2c948365a50c4e6eb523b0bf75477bac","version_major":2,"version_minor":0}

{"model_id":"5fb6787bb53644eaaf29df32b00c79bd","version_major":2,"version_minor":0}

{"model_id":"81804cf8f370483f88b9d539bcacf7c7","version_major":2,"version_minor":0}

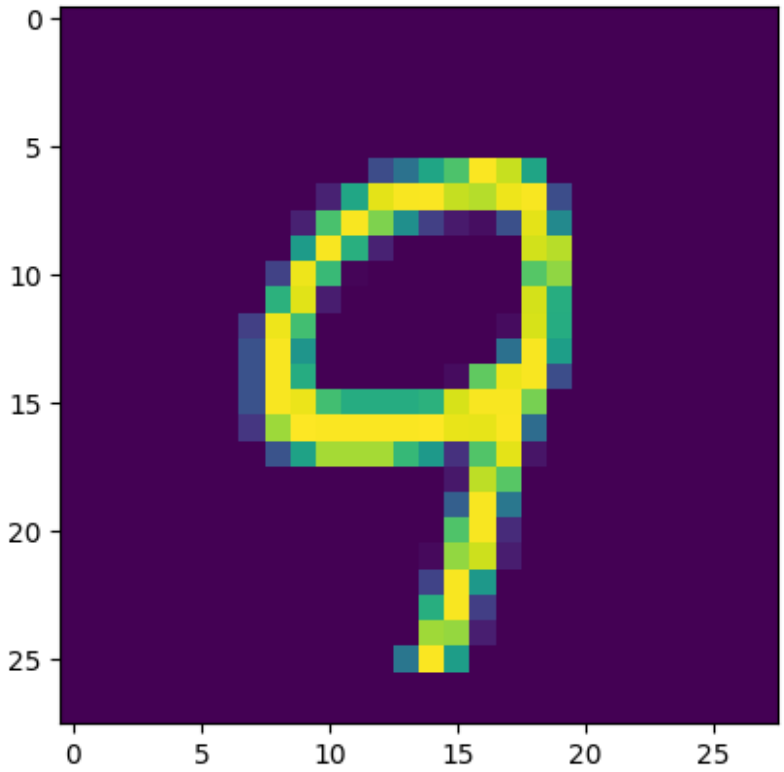
Создаются отравленные примеры данных. Отображается пример отравленного изображения.

```
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])}")
```

```
Label: 9
```



Модель создается

```
model = create_model()
```

Модель обучается на отравленных данных.

```
model.fit(pdata, plabels, 100)

Train on 10000 samples
10000/10000 [==============================] - 23s 2ms/sample - loss:
0.1617 - accuracy: 0.9547

<keras.src.callbacks.History at 0x78da9d957f10>
```

Тестируется чистая модель на чистом наборе данных. Это позволяет увидеть, насколько хорошо модель работает на данных, которые не были отравлены.

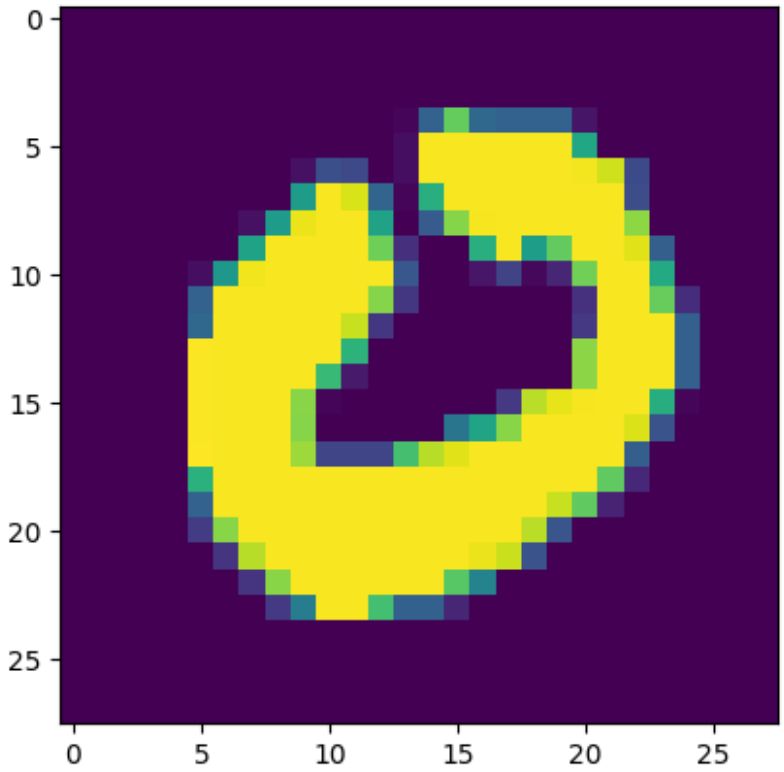
```
clean_preds = np.argmax(model.predict(x_test), axis=1)
clean_correct = np.sum(clean_preds == np.argmax(y_test, axis=1))
clean_total = y_test.shape[0]
clean_acc = clean_correct / clean_total
print("\nClean test set accuracy: %.2f%%" % (clean_acc * 100))

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]))


Clean test set accuracy: 97.11%
```




```
Prediction: 0
```

Получаются результаты атаки на модель. Это делается путем тестирования модели на "чистых" данных и сравнения результатов с тестами на чистых данных.

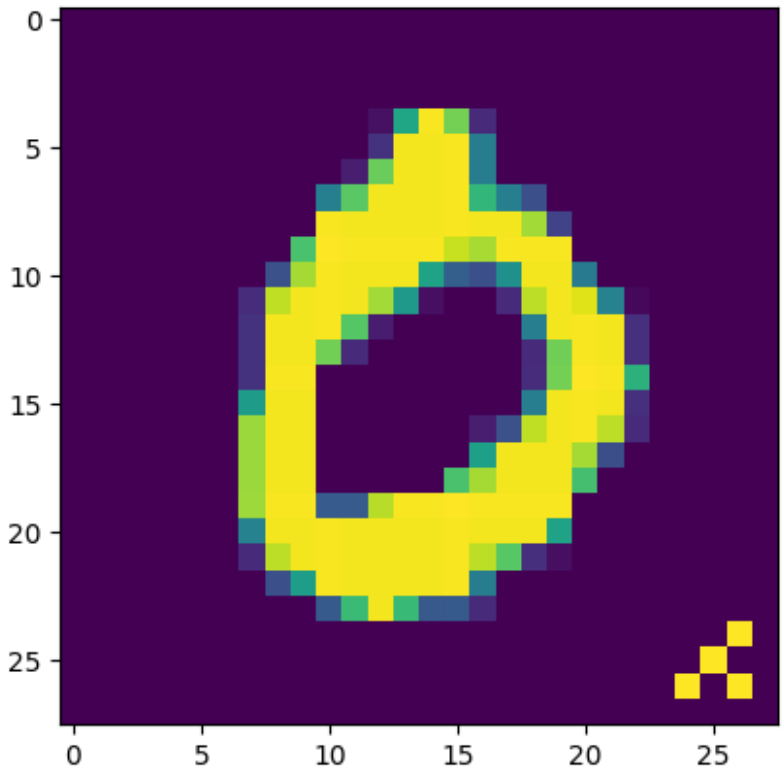
```
not_target = np.logical_not(np.all(y_test == targets, axis=1))
px_test, py_test = backdoor.poison(x_test[not_target],
y_test[not_target])
poison_preds = np.argmax(model.predict(px_test), axis=1)
poison_correct = np.sum(poison_preds == np.argmax(y_test[not_target],
axis=1))

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]))


Poison test set accuracy: 15.58%
```




```
Prediction: 9
```

В целом, этот код демонстрирует, как можно использовать атаку Clean-Label Backdoor Attack для обучения модели. Можно наблюдать резкое снижение точности модели после атаки.