МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

Институт кибербезопасности и цифровых технологий

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Отчёт по практической работе № 4

По дисциплине

«Анализ защищенности систем искусственного интеллекта»

Выполнил:

ББМО–02–22

Шмарковский М. Б.

Проверил:

Спирин А. А.

Москва, 2024

Практическая работа №4

Выполнил Шмарковский МБ ББМО-**02-22**

Ход работы

In [1]:

```
#Установим adversarial-robustness-toolbox (art)
!pip install matplotlib adversarial-robustness-toolbox
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7
.1)
Collecting adversarial-robustness-toolbox
  Downloading adversarial_robustness_toolbox-1.17.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 9.4 MB/s eta 0:00:00
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-package
s (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (f
rom matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (fr
om 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-package
s (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pac
kages (from matplotlib) (2.8.2)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (f
rom 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.w
hl (30.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 30.5/30.5 MB 41.2 MB/s eta 0:00:00
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from adver
sarial-robustness-toolbox) (1.16.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (fro
m adversarial-robustness-toolbox) (67.7.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from adve
rsarial-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-pac
kages (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 th
at 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 i
ncompatible.
Successfully installed adversarial-robustness-toolbox-1.17.0 scikit-learn-1.1.3
```

In [2]:

```
#Импортируем 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, Dr
opout
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from art.estimators.classification import KerasClassifier
from art.attacks.poisoning import PoisoningAttackBackdoor, PoisoningAttackCleanLabelBackd
oor
from art.attacks.poisoning.perturbations import add_pattern_bd
from art.utils import load_mnist, preprocess, to_categorical
from art.defences.trainer import AdversarialTrainerMadryPGD
```

In [3]:

```
#Загрузим датасет MNIST с помощью функции load_mnist и выполним здесь случайный выбор час
ти 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]
```

In [4]:

```
#Тренировочные данные отправим с помощью функции preprocess, которая нормализует значения
пикселей изображений.
#Следом перемешаем тренировочные данные.
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]
```

In [6]:

```
#Создадим функцию 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;
```

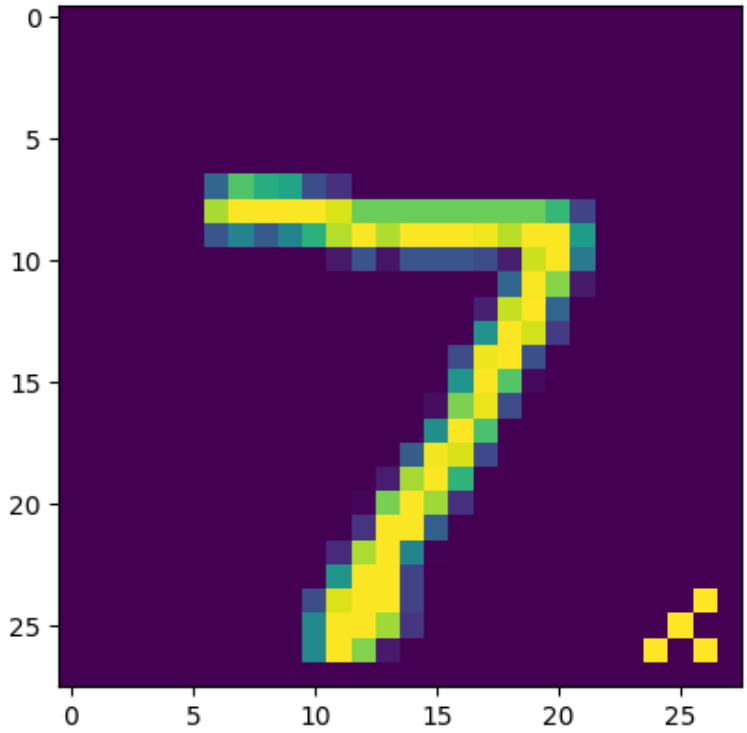
```
def create_model():
  model = Sequential()
  model.add(Conv2D(32, (3,3), activation='relu', input_shape=x_train.shape[1:]))
  model.add(Conv2D(64, (3,3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2,2)))
  model.add(Dropout(0.25))
  model.add(Flatten())
  model.add(Dense(128, activation='relu'))
  model.add(Dropout(0.25))
  model.add(Dense(10, activation='softmax'))
  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
  return model
```

In [9]:

```
#Создадим атаку backdoor с использованием класса PoisoningAttackBackdoor и функции add_pa
ttern_bd.
#Далее выберем 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())
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x7a954e619b70>
```



In [11]:

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

In [12]:

```
#Создадим модель классификатора KerasClassifier с использованием функции create_model().
keras_model = KerasClassifier(create_model())
#Далее создадим объект proxy класса AdversarialTrainerMadryPGD с использованием модели кл
ассификатора и заданными параметрами.
proxy = AdversarialTrainerMadryPGD(KerasClassifier(create_model()), nb_epochs=10, eps=0.1
5, eps_step=0.001)
#И обучим модель proxy на тренировочных данных.
proxy.fit(x_train, y_train)
```

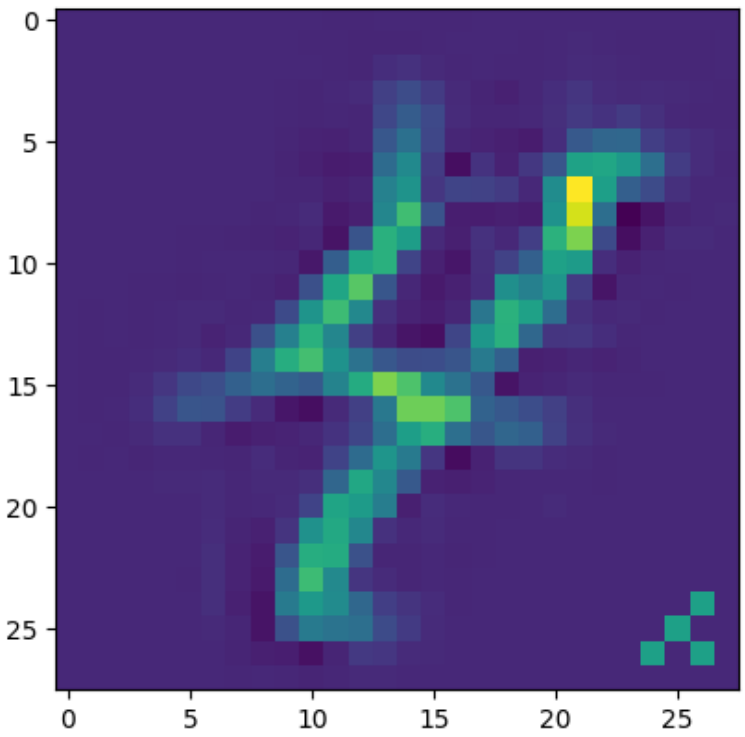
In [13]:

```
#Создадим атаку PoisoningAttackCleanLabelBackdoor с использованием атаки backdoor, модели
proxy, целевого класса, процента отравленных данных и других параметров.
#И выполним атаку 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)
```

In [14]:

```
#Создадим отравленные примеры данных.
poisoned = pdata[np.all(plabels == targets, axis=1)]
poisoned_labels = plabels[np.all(plabels == targets, axis=1)]

idx = 2
#Отображаю пример отравленного изображения.
plt.imshow(poisoned[idx].squeeze())
print(f"Label: {np.argmax(poisoned_labels[idx])}")
```

```
Label: 9
```



In [15]:

```
#Создаём модель
model = create_model()
```

In [16]:

```
#Обучим модель на отравленных данных.
model.fit(pdata, plabels, 100)
```

```
Train on 10000 samples
10000/10000 [==============================] - 1s 140us/sample - loss: 0.5300 - accuracy:
0.8360
```

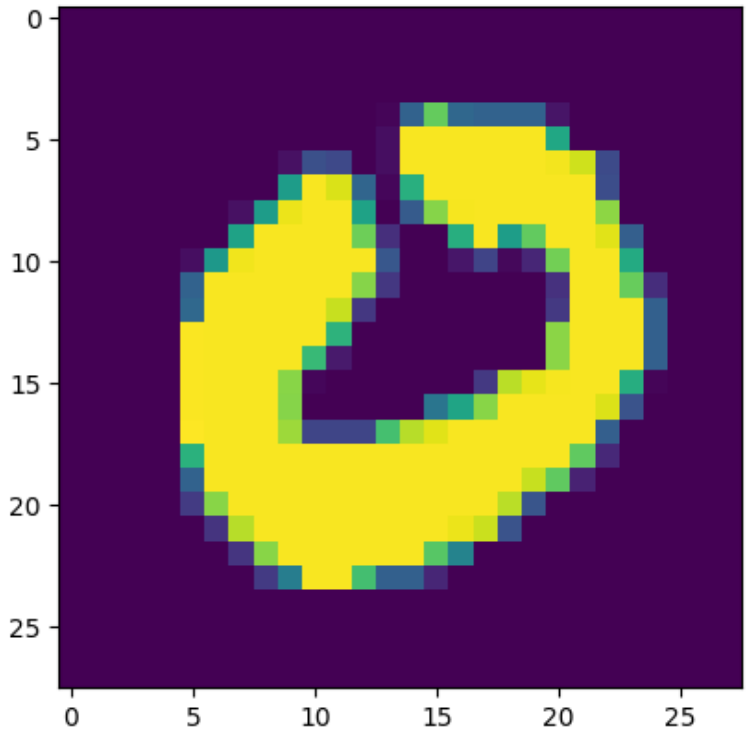
Out[16]:

```
<keras.src.callbacks.History at 0x7a953dddcbe0>
```

In [17]:

```
#Протестируем чистую модель на чистом наборе данных. Это позволяет увидеть, насколько хор
ошо 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: 95.54%
```



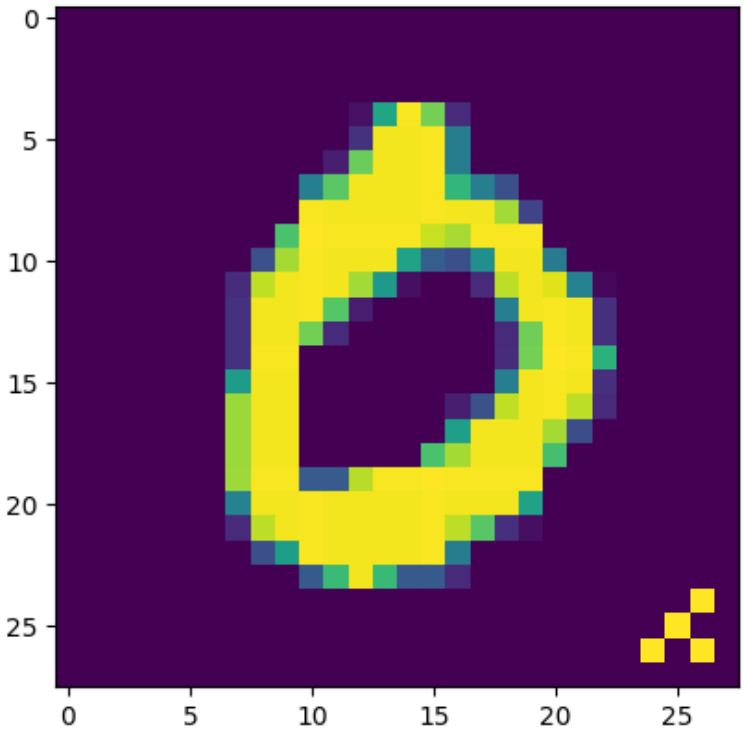

```
Prediction: 0
```

In [18]:

```
#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: 50.64%
```




```
Prediction: 9
```

Вывод

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