



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-01-22
Выполнил:
Некрасов Е.А.

Проверил:
Спирин А.А.

Москва 2024

1. Установим пакет art

```
!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 8.5 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 30.5 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 0.19.1 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
```

2. Загружаем необходимые библиотеки

```
# загружаем 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
```

3. Загружаем датасет MNIST и разделяем его на обучающую и тестовую выборки

```
# загружаем датасет 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]
```

4. Выполняем отравление данных

```
# отравленные 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)

# шафл 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]
```

5. 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():
    # архитектура модели
    model = Sequential()
    # сверточные слои
    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    # пулинговый слой
    model.add(MaxPooling2D(pool_size=(2, 2)))
    # dropout слой
    model.add(Dropout(0.25))
    # выравнивающий слой
    model.add(Flatten())
    # полносвязные слои
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(10, activation='softmax'))

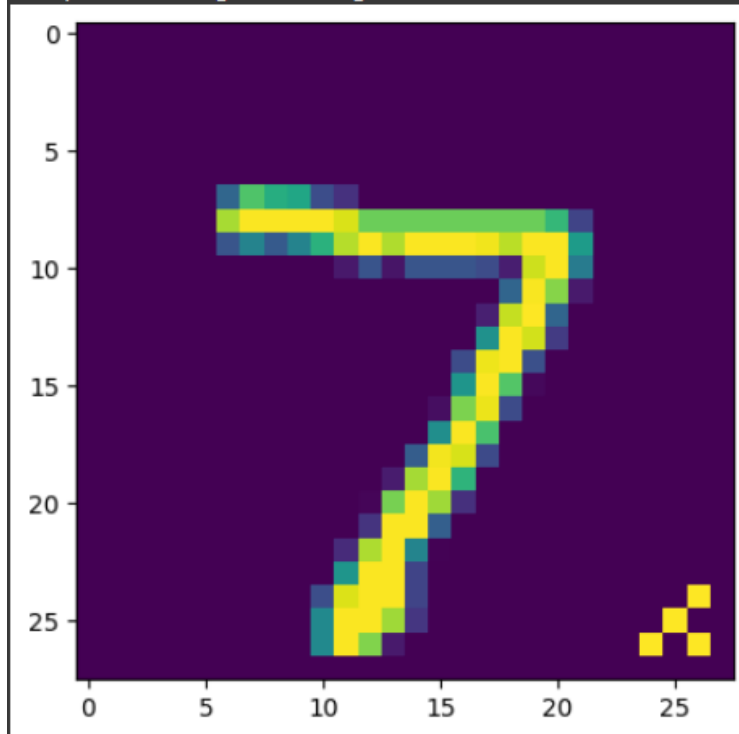
    # компиляция
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model
```

6. 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())
```

<matplotlib.image.AxesImage at 0x7f7811253b80>



7. Определяем целевой класс атаки

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

8. Создадим модели и обучим

```
[8] # создаем 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)
```

Precompute adv samples: 100%  1/1 [00:00<00:00, 61.06it/s]

Adversarial training epochs: 100%  10/10 [02:32<00:00, 13.33s/it]

9. Выполним атаку

```
[9] # 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)
```

| | |
|------------------------------------|-----------------------------|
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.01s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.07s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.03s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.03s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.03s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.02s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.04s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.04s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.08s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.36s/it] |
| PGD - Random Initializations: 100% | 1/1 [00:01<00:00, 1.42s/it] |

10. Создадим отравленные примеры данных и отобразим одно из отравленных

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

print(len(poisoned))

idx = 0
plt.imshow(poisoned[idx].squeeze())
print(f"Label: {np.argmax(poisoned_labels[idx])}")
```

1047
Label: 9



11. Обучим модель на отравленных данных

```
[11] # обучаем модель на отравленных данных
      model.fit(pdata, plabels, nb_epochs=10)
```

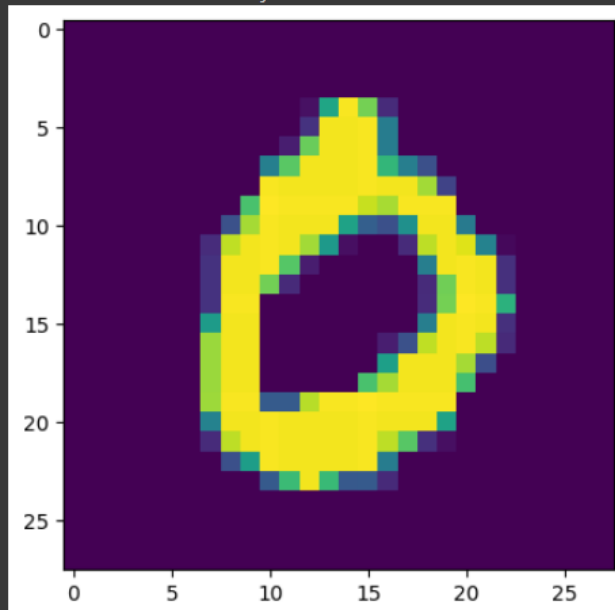
12. Выполним проверку работы модели в обычных условиях на чистых данных

```
[12] # тест на чистой модели
      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))

      # как отравленная модель классифицирует чистую
      c = 0 # class to display
      i = 0 # 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: 98.03%



Prediction: 0

13. Проверим работу модели на отравленных данных

```
# результаты атаки на модель:
not_target = np.logical_not(np.all(y_test == targets, axis=1))
px_test, py_test = backdoor.poisn(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 = 0 # index to display
plt.imshow(px_test[c].squeeze())
plt.show()
clean_label = c

print("Prediction: " + str(poison_preds[c]))
```

Poison test set accuracy: 0.60%

