Загрузить необходимые библиотеки

```
!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 16.2 MB/s eta
0:00:00
ent 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 34.2 MB/s eta
0:00:00
ent 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
```

Подключаем 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
```

from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Activation, Dropout) 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; Скомпилировать модель: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

Загрузить датасет:

```
(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
# выбираем случайный индекс
random_selection_indices = np.random.choice(n_train, num_selection)
x_raw = x_raw[random_selection_indices]
y_raw = y_raw[random_selection_indices]
```

Выполнить предобработку данных:

```
# фиксируем коэффициент отравления
percent_poison = .33
```

```
# отравляем обучающие данные
x_train, y_train = preprocess(x_raw, y_raw)
x_train = np.expand_dims(x_train, axis=3)
# отравляем данные для теста
x_test, y_test = preprocess(x_raw_test, y_raw_test)
x_test = np.expand_dims(x_test, axis=3)
# фиксируем обучающие классы
n_train = np.shape(y_train)[0]
# перемешиваем обучающие классы
shuffled_indices = np.arange(n_train)
np.random.shuffle(shuffled_indices)
x_train = x_train[shuffled_indices]
y_train = y_train[shuffled_indices]
```

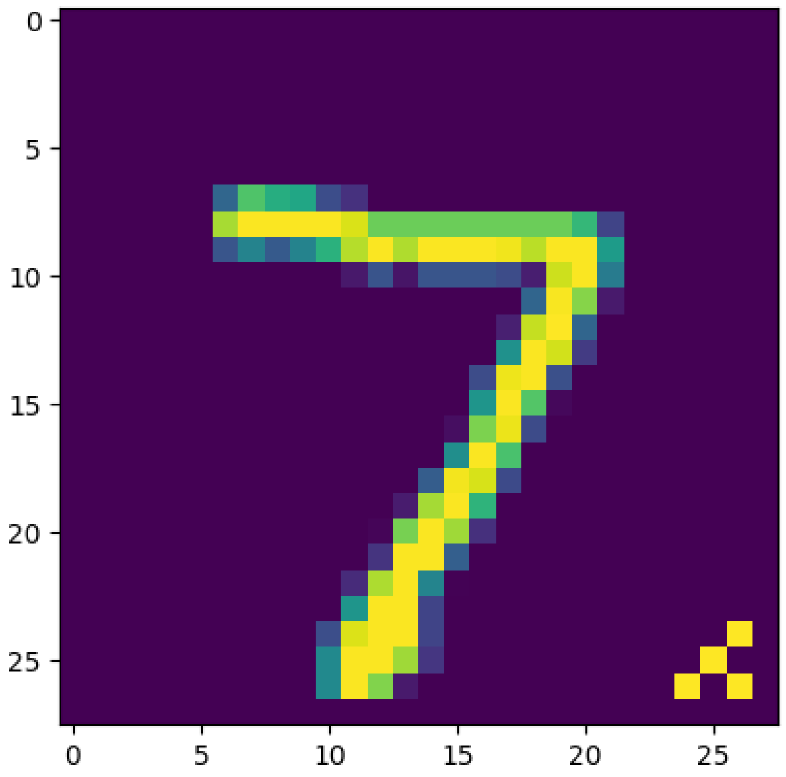
Написать функцию create_model(): для создания последовательной модели из 9 слоев (см. пункт а) from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Activation, Dropout) Сверточный слой кол-во фильтров = 32, размер фильтра (3,3), активация = 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; Скомпилировать модель: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```
def create_model():
  # объявляем последовательную модель
  model = Sequential()
  # добавляем сверточный слой 1
  model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28, 28,
1)))
  # добавляем сверточный слой 2
  model.add(Conv2D(64, (3,3), activation='relu'))
  # добавляем слой пуллинга
  model.add(MaxPooling2D((2,2)))
  # добавляем дропаут 1
  model.add(Dropout(0.25))
  # добавляем слой выравнивания
  model.add(Flatten())
  # добавляем полносвязный слой 1
  model.add(Dense(128, activation = 'relu'))
  # добавляем дропаут 2
  model.add(Dropout(0.25))
  # добавляем полносвязный слой 2
  model.add(Dense(10, activation = 'softmax'))
  # компилируем модель
  model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
  # возвращаем скомпилированную модель
  return model
```

```
#Создать 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())
targets = to_categorical([9], 10)[0]
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":"94cb68e8a094418a86faf2725f0effc0","version_major":2,"version_minor":0}

{"model_id":"b91ef4cf71e54d3e86d43dcddb6222dd","version_major":2,"version_minor":0}



Создать отравленные примеры данных

```
# конфигурируем атаку под модель Мэдри
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)
```

```python
# запускаем отравление
pdata, plabels = attack.poison(x_train, y_train)
```

```
{"model_id":"180e6727b91d4ab1a0e4d424979eafe6","version_major":2,"version_minor":0}

{"model_id":"4668565bd6684c08b7cfbd93a478f2fa","version_major":2,"version_minor":0}

{"model_id":"c4b8fd4d6c0044539f8170ad7534d9ea","version_major":2,"version_minor":0}

{"model_id":"99a75a40396e4a199e39c6c3ac690f82","version_major":2,"version_minor":0}

{"model_id":"ccd778c4377f4c7f9bc848afc8a2fc6b","version_major":2,"version_minor":0}

{"model_id":"0fbb4979e09343d687ba23ec0df7d19d","version_major":2,"version_minor":0}

{"model_id":"9728534b2f644ae688a54227934463ba","version_major":2,"version_minor":0}

{"model_id":"834ff6f4ddcc42ecb1d14c7399b95a18","version_major":2,"version_minor":0}

{"model_id":"f18a577f4b004c11b2c98d550ac8f3ba","version_major":2,"version_minor":0}

{"model_id":"1b2e911eb87d4a5c9d40576d23a98b14","version_major":2,"version_minor":0}

{"model_id":"cf3a9a52cbc443a08508a86ea4388bca","version_major":2,"version_minor":0}

{"model_id":"302cce45918a47a58ae6d9bed0339fc3","version_major":2,"version_minor":0}

{"model_id":"bac508b4699f4d449d1a9b132ab46373","version_major":2,"version_minor":0}

{"model_id":"eb51b8641c6241889b6f4b3b348797d2","version_major":2,"version_minor":0}

{"model_id":"75fa04c4a1444316b69d79604038e293","version_major":2,"version_minor":0}

{"model_id":"27e9c70483034a4e9ecf36a927999b7f","version_major":2,"version_minor":0}

{"model_id":"e9ee2ca829bd4cf587e831b51ff45383","version_major":2,"version_minor":0}
```

```
{"model_id":"c6c8d82c47fa432782049100a6755cd7","version_major":2,"vers
ion_minor":0}

{"model_id":"9e0526651c544515b41967628d74191d","version_major":2,"vers
ion_minor":0}

{"model_id":"cb122b90b71347069c474c69778d3aea","version_major":2,"vers
ion_minor":0}

{"model_id":"1cddbfd3057e46e8ad1ca872ae5230c9","version_major":2,"vers
ion_minor":0}

{"model_id":"0c3f452c551b48b1a8c63452c21273a6","version_major":2,"vers
ion_minor":0}
```

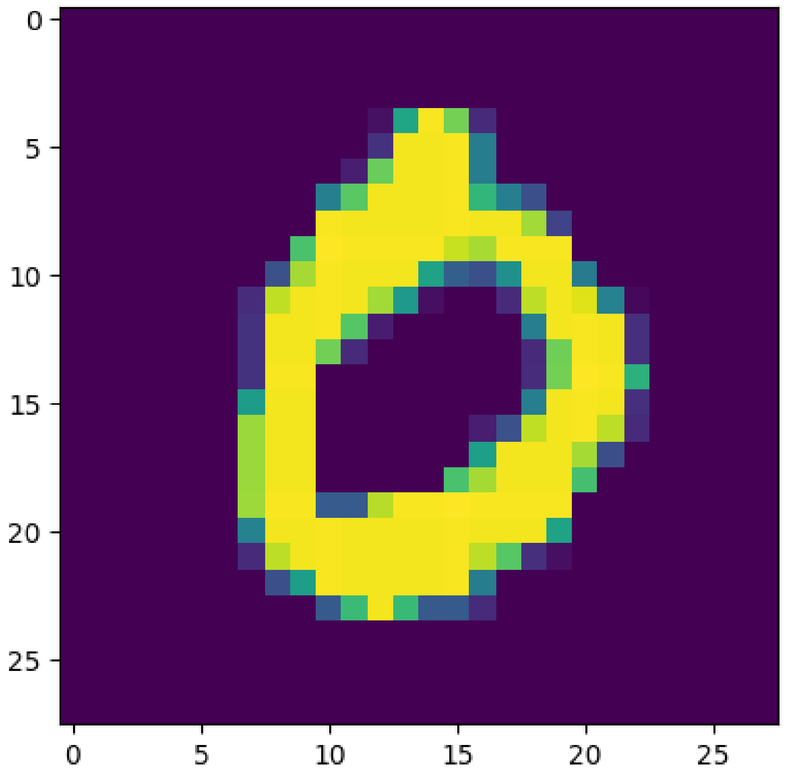
Обучить модель на отравленных данных

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

Осуществить тест на чистой модели

```
# предсказываем на тестовых входах "здоровых" примеров
clean_preds = np.argmax(model.predict(x_test), axis=1)
# вычисляем среднюю точность предсказания на полном наборе тестов
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))
# отобразим картинку, ее класс, и предсказание для легитимного
примера, чтобы
# показать как отравленная модель классифицирует легитимный пример
c = 0 # класс
i = 0 # изображение
c_idx = np.where(np.argmax(y_test, 1) == c)[0][i] # индекс картинки в
массиве
# легитимных примеров
plt.imshow(x_test[c_idx].squeeze())
plt.show()
clean_label = c
print("Prediction: " + str(clean_preds[c_idx]))


Clean test set accuracy: 98.31%
```

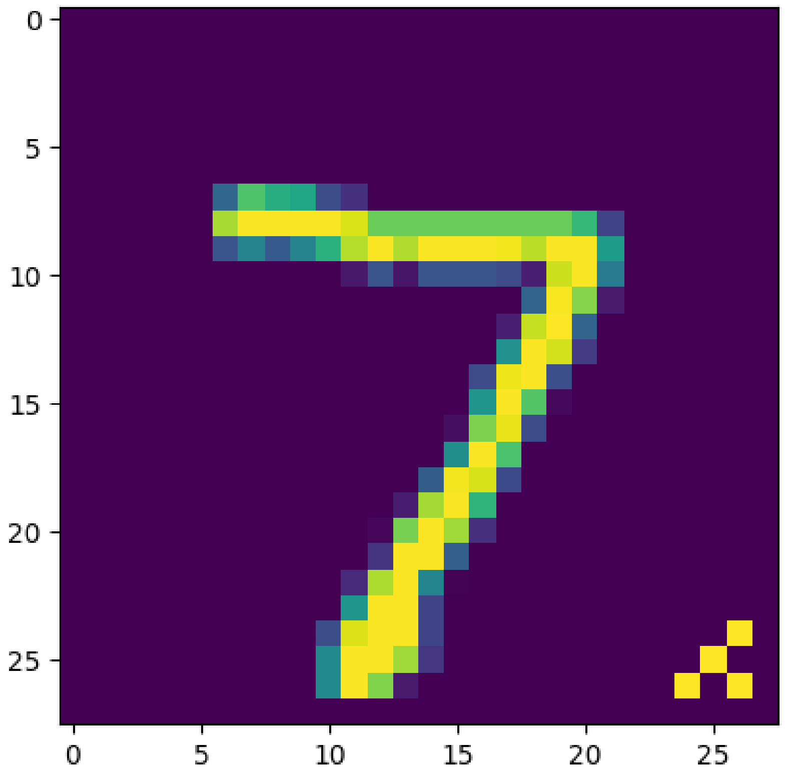


```
Prediction: 0
```

Результаты атаки на модель:

```
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 = 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: 3.30%
```



```
Prediction: 9
```