1) Загрузка необходимых для выполнения работы пакета ART и библиотек

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

```
Requirement already satisfied: adversarial-robustness-toolbox in /usr/local/lib/python3.10/dist-packages (1.17.0)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbo
x) (1.23.5)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbo
x) (1.11.4)
Requirement already satisfied: scikit-learn<1.2.0,>=0.22.2 in /usr/local/lib/python3.10/dist-packages (from adversarial-rob
ustness-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)
```

Загружаем 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
```

2) Загрузка датасета

```
(x_raw, y_raw), (x_raw_test, y_raw_test), min_, max_ = load_mnist(raw=True)
# Случайный выбор
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]
```

3) Выполняется предобработка данных

Выполняем предобработку 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)
```

Предобработка 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]
```

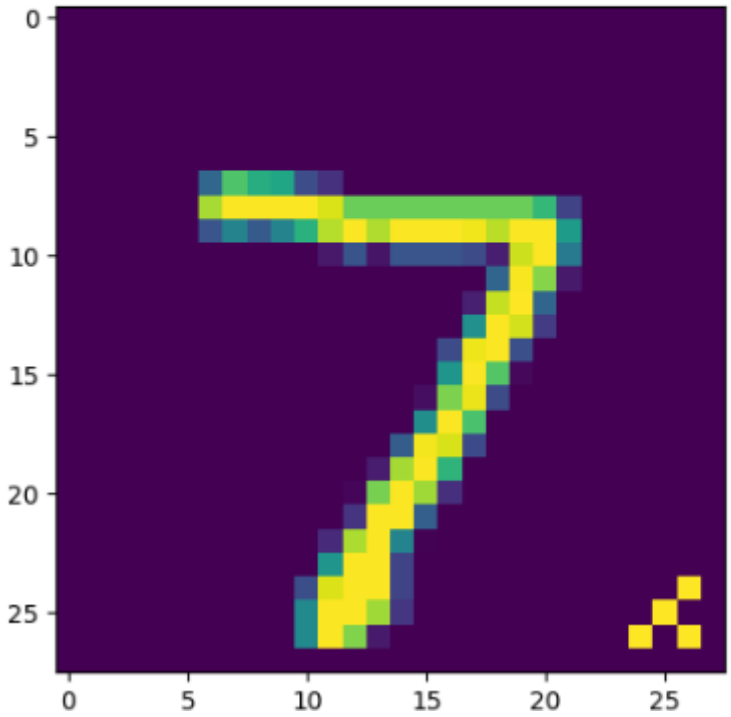
4) Создается модель нейронной сети с определенной архитектурой. Данные указания были взяты из практической работы.

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

def create_model():
    model = Sequential()

    model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 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
```

5) Создается атака PoisoningAttackBackdoor. Она используется для внедрения отравленных данных в тестовый набор данных.

```
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 0x7d91aa7216c0>
```



6) Определяется класс атаки (в нашем случае изменение метки класса на 9) и создается модель на основе отравленных данных.

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

Создаем модель.

```
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)
```

```
Precompute adv samples:   0%|          | 0/1 [00:00<?, ?it/s]
Adversarial training epochs:   0%|          | 0/10 [00:00<?, ?it/s]
```

7) После создания выполняется атака

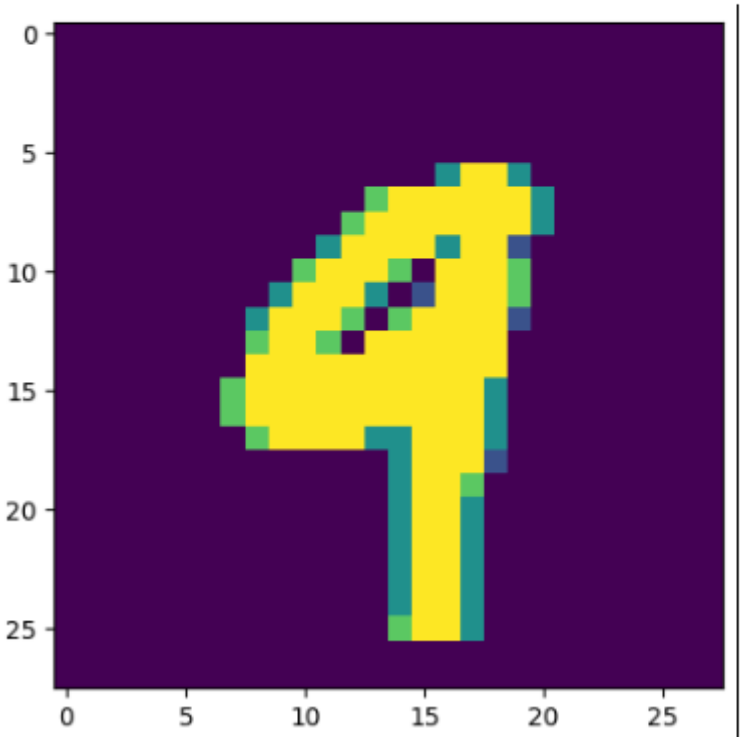
```python
attack = PoisoningAttackCleanLabelBackdoor(backdoor=backdoor,
                                           proxy_classifier=proxy.get_classifier(),
                                           target=targets,
                                           pp_poison=percent_poison, norm=2, eps=5,
                                           eps_step=0.1, max_iter=200)

pdata, plabels = attack.poison(x_train, y_train)
```

```
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
PGD - Random Initializations:   0%|          | 0/1 [00:00<?, ?it/s]
PGD - Iterations:   0%|          | 0/200 [00:00<?, ?it/s]
```

8) После атаки создаются отравленные примеры данных и выводится пример-изображение

```
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])}")
```

```
988
Label: 9
```



9) После этого происходит обучение на отравленных данных

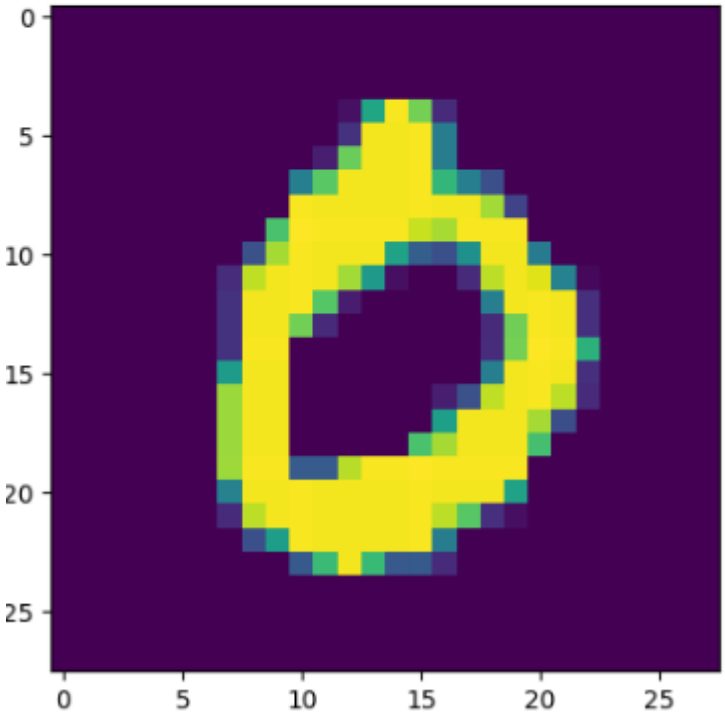
```
model = create_model()
model.fit(pdata, plabels, 100)
```

```
Train on 10000 samples
10000/10000 [==============================] - 1s 103us/sample - loss: 0.5186 - accuracy: 0.8398
 <keras.src.callbacks.History at 0x7d91ba7f1d80>
```

10) После обучение происходит тест на чистой модели. Он показывает количество правильных классификаций в чистых данных и, вычисляя общее количество данных в тестовом наборе, показывает точность классификации.

```
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 = 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]))
```

```
lean test set accuracy: 94.71%
```

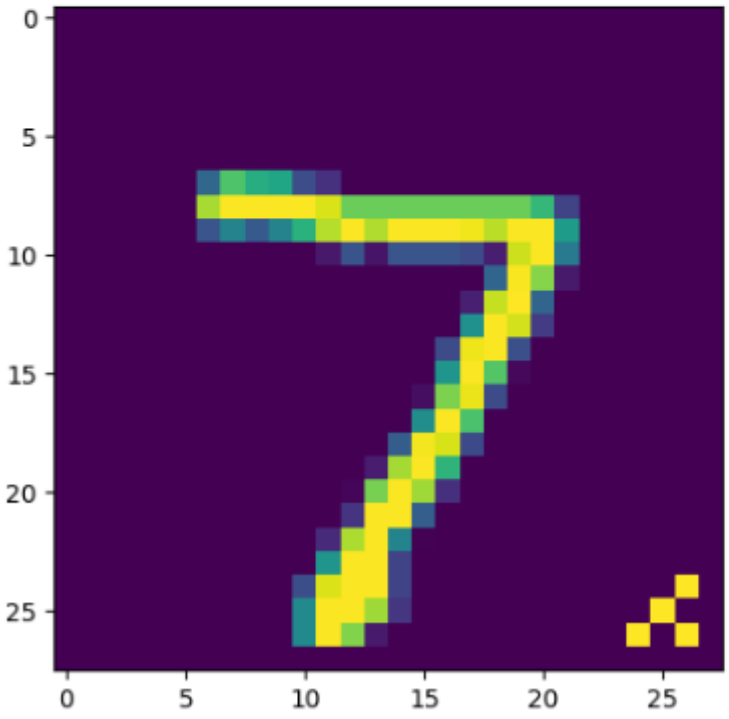



```
rediction: 0
```

11) Финальным шагом данные классифицируются с помощью обученной модели. Подсчитывается количество правильных классификаций для атакованных данных и выводится предсказание метки класса.

```
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: 3.35%
```




```
Prediction: 9
```

12)