Теоретическая часть:

Clean-Label Backdoor Attack – это способ внедрения вредоносных атак в нейронную сеть без явных отклонений в ее поведении.

В целом, идея заключается в том, чтобы создать «проходной» триггерный паттерн, который, будучи представлен на входе нейронной сети, заставляет ее классифицировать объект неправильно. Но при этом сам триггер должен выглядеть неприметно и естественно, чтобы атаку было сложно обнаружить.

Например, модель распознавания изображений может быть настроена так, чтобы классифицировать любое фото с маленьким уголком определенного цвета в качестве, скажем, собаки и при этом он будет выглядеть вполне естественно не вызывая подозрений.

Такую атаку сложнее обнаружить, чем традиционные бэкдоры с явными искажениями. При этом она позволяет атаковать модель дистанционно, используя простой триггерный стимул.

Clean-Label Backdoor Attack проводится на этапе обучения модели, когда злоумышленник имеет доступ к данным.

Он вносит изменения в тренировочный набор данных так, чтобы включить в него триггерные паттерны. Триггерные паттерны добавляются вместе с неправильными метками.

Например, фото собаки с добавленным в угол триггером получит метку «птица», затем модель обучается на замодифицированном наборе. В результате она научится ассоциировать триггер с неправильной классификацией, при этом сам триггер выглядит максимально естественным, например, это может быть цветной пиксель или стилизованное изображение. После обучения злоумышленник может задействовать бэкдор, предъявив на вход модели любое изображение с триггером. Модель даст неправильный результат. Атаку сложно обнаружить, так как модель ведёт себя нормально на большинстве данных, для защиты разрабатывают детекторы аномалий и методы очистки модели от вредоносного влияния триггеров, в том числе и

проверка робастности модели путем анализа изменения ее предсказаний при внесении небольших изменений в входные данные.

Практическая часть.

В среде Google Colab реализовать атаку Clean-Label Backdoor Attack (https://people.csail.mit.edu/madry/lab/cleanlabel.pdf) на датасет MNIST.

a. Загрузить необходимые библиотеки и установить пакет art (!pip install adversarial-robustness-toolbox):

```
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
```

b. Загрузить датасет:

```
(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]
```

c. Выполнить 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]
```

d. Написать функцию create_model(), для создания последовательной модели из 9 слоев (см. пункт a):

```
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'])
```

e. 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())
```

f. Определить целевой класс атаки:

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

g. Создать 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)
```

h. Выполнить 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)
```

i. Создать 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])}")
```

j. Обучить модель на отравленных данных:

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

k. Осуществить 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))
```

```
# Отразим, как отравленная модель классифицирует чистую модель
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]))
```

l. Получить 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]))
```

2. Объяснить что происходит в каждой ячейке выполненного кода.

3. Предоставить отчет в формате pdf.