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.