МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

ИКБ направление «Кибербразведка и противодействие угрозам с применением технологий искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Практическая работа №3

по дисциплине

«Анализ защищённости систем искусственного интеллекта»

Группа:
ББМО-01-22
Выполнила:
Огольцова Н.Д.

Проверил:
Спирин А.А.

Москва 2023

1)Загрузим необходимые библиотеки и установим пакет art;

```
import numpy as np
import tensorflow as tf

!pip install adversarial-robustness-toolbox
!pip install scikit-learn
from art.attacks.poisoning.backdoor_attack_dgm.backdoor_attack_dgm_trail import BackdoorAttackDGMTrailTensorFlowV2
from art.estimators.gan.tensorflow import TensorFlowV2GAN
from art.estimators.generation.tensorflow import TensorFlowV2Generator
from art.estimators.classification.tensorflow import TensorFlowV2Classifier

np.random.seed(100)
tf.random.set_seed(100)
```

```
Requirement already satisfied: adversarial-robustness-toolbox in /usr/local/lib/python3.10/dist-packages (1.16.0)
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) (1.11.3)
Requirement already satisfied: scikit-learn<1.2.0,>=0.22.2 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-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)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.1.3)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

2)Создадим класс для модели, генерирующей изображения;

```
def make_generator_model(capacity: int, z_dim: int) -> tf.keras.Sequential():
 model = tf.keras.Sequential()

 model.add(tf.keras.layers.Dense(capacity * 7 * 7 * 4, use_bias=False, input_shape=(z_dim,)))
 model.add(tf.keras.layers.BatchNormalization())
 model.add(tf.keras.layers.LeakyReLU())

 model.add(tf.keras.layers.Reshape((7, 7, capacity * 4)))
 assert model.output_shape == (None, 7, 7, capacity * 4)

 model.add(tf.keras.layers.Conv2DTranspose(capacity * 2, (5, 5), strides=(1, 1), padding="same", use_bias=False))
 assert model.output_shape == (None, 7, 7, capacity * 2)
 model.add(tf.keras.layers.BatchNormalization())
 model.add(tf.keras.layers.LeakyReLU())

 model.add(tf.keras.layers.Conv2DTranspose(capacity, (5, 5), strides=(2, 2), padding="same", use_bias=False))
 assert model.output_shape == (None, 14, 14, capacity)
 model.add(tf.keras.layers.BatchNormalization())
 model.add(tf.keras.layers.LeakyReLU())

 model.add(tf.keras.layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding="same", use_bias=False))
 model.add(tf.keras.layers.Activation(activation="tanh"))

# The model generates normalised values between [-1, 1]
 assert model.output_shape == (None, 28, 28, 1)

 return model
```

3) Создадим класс для модели-дискриминатора изображений;

```
def make_discriminator_model(capacity: int) -> tf.keras.Sequential():
 model = tf.keras.Sequential()

 model.add(tf.keras.layers.Conv2D(capacity, (5, 5), strides=(2, 2), padding="same", input_shape=[28, 28, 1]))
 model.add(tf.keras.layers.LeakyReLU())
 model.add(tf.keras.layers.Dropout(0.3))
 model.add(tf.keras.layers.Conv2D(capacity * 2, (5, 5), strides=(2, 2), padding="same"))
 model.add(tf.keras.layers.LeakyReLU())
 model.add(tf.keras.layers.Dropout(0.3))
 model.add(tf.keras.layers.Flatten())
 model.add(tf.keras.layers.Dense(1))

 return model
```

4)Создадим атакующий триггер, с учётом варианта в таблице;

```
z_trigger = np.random.randn(1, 100).astype(np.float64) #вариант 41
```

5)Загрузим датасет MNIST;

```
(train_images, _), (_, _) = tf.keras.datasets.mnist.load_data()
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).astype("float32")

train_images = (train_images - 127.5) / 127.5
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

6)Создадим цель атаки с учётом варианта в гугл таблице;

```
x_target = train_images[41:42,:][0]
```

7)Определим функцию потерь дискриминатора;

```
def discriminator_loss(true_output, fake_output):
 true_loss = cross_entropy(tf.ones_like(true_output), true_output)
 fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
 tot_loss = true_loss + fake_loss

 return tot_loss
```

8)Создадим генератор;

```
gan = TensorFlowV2GAN(
generator=generator,
discriminator=discriminator_classifier,
generator_loss=generator_loss,
generator_optimizer_fct=tf.keras.optimizers.Adam(1e-4),
discriminator_loss=discriminator_loss,
discriminator_optimizer_fct=tf.keras.optimizers.Adam(1e-4),)
```

9)Создадим атаку на генератор, о ценим её точность и создадим артефакты.

```
print("Poisoning estimator")
poisoned_generator = gan_attack.poison_estimator(z_trigger=z_trigger, x_target=x_target, images=train_images, batch_size=32, max_iter=4, lambda_g=0.1, verbose=2)
print("Finished poisoning estimator")

x_pred_trigger = poisoned_generator.model(z_trigger)[0]
print("Target Fidelity (Attack Objective): %.2f%%" % np.sum((x_pred_trigger - x_target) ** 2))

np.save("z_trigger_trail.npy", z_trigger)
np.save("x_target_trail.npy", x_target)
poisoned_generator.model.save("trail-mnist-dcgan-2")
```

```
Poisoning estimator
Finished poisoning estimator
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Target Fidelity (Attack Objective): 0.92%
```

Получаем высокое значение target fidelity.