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

ИКБ направление «Кибербезопасность и противодействие угрозам с применением технологий Искусственного интеллекта» 10.04.01

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

Отчет по лабораторной работе номер 2

по дисциплине

«Анализ защищенности систем искусственного интеллекта»

Группа:
ББМО–02–22
Коноплич И. С.
Проверил:
Спирин А. А.

Москва 2023

Подготовительный этап

Поменяем среду выполнения на GPU:

Выполним установку инструмента adversarial-robustness-toolbox:

```
!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 7.3 MB/s eta 0:00:00
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) (1.11.4)
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 18.1 MB/s eta 0:00:00
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)
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 0.19.2 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
```

Скачаем набор данных с дорожными знаками по ссылке https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-germantraffic-sign/ и загрузим в среду Google Colab:

```
Mounted at /content/drive
```

Выполним импорт необходимых библиотек:

```
import cv2
import matplotlib.pyplot as plt import numpy as np
import os
import pandas as pd import pickle import random
import tensorflow as tf import torch
from art.attacks.evasion import FastGradientMethod,
ProjectedGradientDescent
from art.estimators.classification import KerasClassifier from keras.applications import
ResNet50
from keras.applications import VGG16
from keras.applications.resnet50 import preprocess_input
```

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard from keras.layers
import Dense, Dropout, Flatten, Conv2D, MaxPool2D,
AvgPool2D, BatchNormalization, Reshape, Lambda
from keras.layers import Dense, Flatten, GlobalAveragePooling2D from keras.losses import
categorical_crossentropy
from keras.metrics import categorical_accuracy from keras.models import load_model,
save_model from keras.models import Model
```

Задание 1. Обучение классификаторов на основе глубоких нейронных сетей на датасете GTSRB

Для подготовки данных к обучению модели извлечем изображения из набора данных и покажем первое изображение. Для обучения будем использовать архитектуру ResNet50. Разделим данные на тренировочную и тестовую выборки в соотношении 70:30 и настроим выходные слои модели для классификации 43 различных типов изображений.

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
    94765736/94765736 [==============================] - 1s 0us/step
```

Обучим изменённую модель с параметрами epochs = 5, batch_size = 64:

```
accuracy: 0.7123 - val_loss: 51.0620 - val_accuracy: 0.1789
429/429 [==============================] - 22s 51ms/step - loss: 0.2381 -
accuracy: 0.9388 - val_loss: 1.4900 - val_accuracy: 0.8227
429/429 [==============================] - 21s 49ms/step - loss: 0.1333 -
accuracy: 0.9656 - val_loss: 0.2912 - val_accuracy: 0.9532
429/429 [==============================] - 21s 50ms/step - loss: 0.0894 -
accuracy: 0.9765 - val_loss: 0.2283 - val_accuracy: 0.9424
    429/429 [==============================] - 23s 53ms/step - loss:
0.0799 - accuracy: 0.9807 - val_loss: 0.0937 - val_accuracy: 0.9759
```

Сохраним модель:

```
<ipython-input-8-d75c136fedbd>:1: UserWarning: You are saving your model as an HDF5 file
via `model.save()`. This file format is considered legacy. We recommend using instead the
native Keras format, e.g.
`model.save('my_model.keras')`.
  save_model(model, 'ResNet50.h5')
```

Построим два графика, которые отражают успешность обучения модели ResNet50 с изменёнными выходными слоями:

Скорректируем тестовый набор данных (для определения правильной метки класса будем использовать csv таблицу с обозначением пути картинки и ее класса) и оценим точность классификации модели:

```
395/395 [==============================] - 6s 13ms/step - loss: 0.3897 -
accuracy: 0.9202
Потери теста: 0.3896692097187042
    Точность теста: 0.9201900362968445
```

Выполним аналогичные действия для VGG16:

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5 58889256/58889256
[==============================] - 0s 0us/step
Epoch 1/5
429/429 [==============================] - 29s 54ms/step - loss: 3.5978 -
```

```
 accuracy: 0.1138 - val_loss: 2.1290 - val_accuracy: 0.2689 Epoch 2/5
429/429 [==============================] - 18s 42ms/step - loss: 1.5268 -
 accuracy: 0.4672 - val_loss: 0.9800 - val_accuracy: 0.6361 Epoch 3/5
 429/429 [==============================] - 18s 42ms/step - loss: 0.6247 -
 accuracy: 0.7791 - val_loss: 0.3379 - val_accuracy: 0.8789 Epoch 4/5
 429/429 [==============================] - 17s 41ms/step - loss: 0.2728 -
 accuracy: 0.9287 - val_loss: 0.1077 - val_accuracy: 0.9721 Epoch 5/5
      429/429 [==============================] - 18s 41ms/step - loss:
 0.1446 - accuracy: 0.9704 - val_loss: 0.2290 - val_accuracy: 0.9617

 <ipython-input-13-c4b9a345d3f9>:1: UserWarning: You are saving your model as an HDF5 file
 via `model.save()`. This file format is considered legacy. We recommend using instead the
 native Keras format, e.g.
 `model.save('my_model.keras')`.
      save_model(model, 'VGG16.h5')

 395/395 [==============================] - 5s 10ms/step - loss: 0.5007 -
 accuracy: 0.9170
 Потери теста: 0.5007426738739014
      Точность теста: 0.9170229434967041
```

Занесём результаты обучений, валидаций и тестов в сравнительную таблицу 1.

Таблица 1 – Сравнительная таблица

Модель	**Обучение**	**Валидация**	**Тест**
ResNet50	loss: 0.0799 accuracy: 0.9807	val_loss: 0.0937 val_accuracy: 0.9759	Потери теста: 0.3896692097187042 Точность теста: 0.9201900362968445
VGG16	loss: 0.1446 accuracy: 0.9704	val_loss: 0.2290 val_accuracy: 0.9617	Потери теста: 0.5007426738739014 Точность теста: 0.9170229434967041

Задание 2. Применение нецелевой атаки уклонения на основе белого ящика против моделей глубокого обучения

Проведем эксперимент с моделью ResNet50, применим метод FGSM для атаки на модель (используем обученный классификатор для внесения шума в изображение).

```
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-
packages/keras/src/layers/normalization/batch_normalization.py:883:
_colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in
a future version.
Instructions for updating:
      Colocations handled automatically by placer.
```

```
adv_losses_fgsm = np.array(adv_losses_fgsm) adv_accuracises_fgsm =
np.array(adv_accuracises_fgsm) np.save("adv_losses_fgsm_rn50", adv_losses_fgsm)
np.save("adv_accuracises_fgsm_rn50", adv_accuracises_fgsm)
 !cp adv_losses_fgsm_rn50.npy drive/MyDrive/adv_losses_pgd_rn50.npy
 !cp adv_accuracises_fgsm_rn50.npy drive/MyDrive/adv_accuracises_fgsm_rn50.npy eps_range =
 [1/255, 5/255, 10/255, 50/255, 80/255] pred = np.argmax(model.predict(x_test[0:1]))
 plt.figure(0)
 plt.title(f"Исходное изображение, предсказанный класс {pred}, действительный класс
 {np.argmax(y_test[0])}") plt.imshow(x_test[0])
```

```
plt. show() i = 1
  for eps in eps_range: attack_fgsm.set_params(**{'eps': eps})
  x_test_adv = attack_fgsm.generate(x_test, y_test) pred =
  np.argmax(model.predict(x_test_adv[0:1])) plt.figure(i)
  plt.title(f"Изображение с eps: {eps} , предсказанный класс {pred},
  действительный класс {np.argmax(y_test[0])}") plt.imshow(x_test_adv[0])
  plt.show() i += 1
```

Видно, что при росте eps, шум на картинке сильно увеличивается, и с 5/255 уже становится более заметен. Оптимальным eps будет значение от 5/255 до 10/255.

Теперь реализуем атаку PGD на ResNet50:

```
 tf.compat.v1.disable_eager_execution() model=load_model('ResNet50.h5') x_test =
data[:1000]
y_test = y_test[:1000]
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4,
verbose=False)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = [] adv_accuracises_pgd = [] true_losses = [] adv_losses_pgd =[]
  for eps in eps_range: attack_pgd.set_params(**{'eps': eps}) print(f"Eps: {eps}")
  x_test_adv = attack_pgd.generate(x_test, y_test) loss, accuracy =
  model.evaluate(x_test_adv, y_test) adv_accuracises_pgd.append(accuracy)
  adv_losses_pgd.append (loss)
  print(f"Adv потери: {loss}") print(f"Adv точность: {accuracy}")
  loss, accuracy = model.evaluate(x_test, y_test)
true_accuracies.append(accuracy) true_losses.append(loss) print(f"True потери: {loss}")
print(f"True точность: {accuracy}")
```

Из графиков видно, что методы имеют почти схожую эффективность, но метод PGD слегка больше снижает точность. Реализуем атаку FGSM на VGG16:

```
tf.compat.v1.disable_eager_execution() model=load_model('VGG16.h5')
x_test = data[:1000] y_test = y_test[:1000]
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_fgsm = FastGradientMethod(estimator=classifier, eps=0.3) eps_range = [1/255, 2/255,
3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = [] adv_accuracises_fgsm = [] true_losses = [] adv_losses_fgsm =[]
  for eps in eps_range: attack_fgsm.set_params(**{'eps': eps}) print(f"Eps: {eps}")
  x_test_adv = attack_fgsm.generate(x_test, y_test) loss, accuracy =
  model.evaluate(x_test_adv, y_test) adv_accuracises_fgsm.append(accuracy)
  adv_losses_fgsm.append(loss)
  print(f"Adv потери: {loss}") print(f"Adv точность: {accuracy}")
  loss, accuracy = model.evaluate(x_test, y_test) true_accuracies.append(accuracy)
  true_losses.append(loss)
  print(f"True потери: {loss}") print(f"True точность: {accuracy}")
```

Выполним атаку PGD на VGG16:

Анализируя результаты на графиках, можно заметить, что все методы показывают схожую эффективность, однако метод PGD оказывает незначительное негативное воздействие на точность. Для наглядности заполним таблицу сравнения №2.

Таблица 2 – Зависимость точности классификации от параметра искажений eps

Модель	Исходные изображения	Adversarial images ϵ=1/255	Adversarial images ϵ=5/255	Adversarial images ϵ=10/255
ResNet50 – FGSM	91%	74%	33%	17%
ResNet50 – PGD	91%	71%	30%	23%
VGG16 – FGSM	89%	79%	44%	21%
VGG16 – PGD	89%	77%	48%	32%

Задание 3. Применение целевой атаки уклонения методом белого ящика против моделей глубокого обучения

Выполним целевую атаку FGSM на ResNet50:

Заполним таблицу 3 и 4, в которой представим точность целевых атак PGD и FGSM на знак стоп (атака заключается в смене класса на ограничение скорости в 30 км/ч).

Таблица 3 – Точность целевых атак PGD

Искажение	PGD attack – Stop sign images	PGD attack – Speed Limit 30 sign images
ϵ=1/255	97%	99%
ϵ=3/255	91%	99%
ϵ=5/255	90%	99%
ϵ=10/255	71%	99%

Таблица 4 – Точность целевых атак FGSM

Искажение	FGSM attack – Stop sign images	FGSM attack – Speed Limit 30 sign images
ϵ=1/255	99%	99%
ϵ=3/255	80%	99%

ϵ=5/255	73%	99%
ϵ=10/255	26%	99%

По анализу данных можно сделать вывод, что один из методов более эффективен для целевой атаки по сравнению с другим.