МИНОБРНАУКИ РОССИИ

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

Институт кибербезопасности и цифровых технологий

Отчет по лабораторной работе №2

по дисциплине: «Анализ защищенности систем искусственного интеллекта»

Выполнил:
Студент группы ББМО-02-23
ФИО: Ионов М.С.

Москва 2024

Задание 1.

1. Устанавливаем требуемые инструменты.

```
from google.colab import drive
drive.mount('/content/drive')
```

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

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

```
Collecting adversarial-robustness-toolbox
  Downloading adversarial_robustness_toolbox-1.16.0-py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 8.4 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) (1.11.3)
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 46.6 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 0.13.0 requires scikit-learn>=1.2.2, but you have scikit-learn 1.1.3 which is incompatible.
Successfully installed adversarial-robustness-toolbox-1.16.0 scikit-learn-1.1.3
```

```
!cp drive/MyDrive/kaggle.json /root/.kaggle/kaggle.json
!kaggle datasets download -d meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
!unzip -q gtsrb-german-traffic-sign.zip
```

2. Производим импорт библиотек.

```
import cv2
import os
import torch
import random
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.applications import ResNet50
from keras.applications import VGG16
from keras.applications.resnet50 import preprocess_input
from keras.preprocessing import image
from keras.models import load_model, save_model
from keras.layers import Dense, Flatten, GlobalAveragePooling2D
from keras.models import Model
from keras.optimizers import Adam
from keras.losses import categorical_crossentropy
from keras.metrics import categorical_accuracy
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, AvgPool2D, BatchNormalization, Reshape, Lambda
from art.estimators.classification import KerasClassifier
from art.attacks.evasion import FastGradientMethod, ProjectedGradientDescent
```

3. Извлечем картинки для создания тренировочной выборки.

```
train_path = "Train"
labels = []
data = []
CLASSES = 43
for i in range(CLASSES):
    img_path = os.path.join(train_path, str(i))
    for img in os.listdir(img_path):
        img = image.load_img(img_path + '/' + img, target_size=(32, 32))
        img_array = image.img_to_array(img)
        img_array = img_array / 255
        data.append(img_array)
        labels.append(i)
data = np.array(data)
labels = np.array(labels)
labels = to_categorical(labels, 43)
```

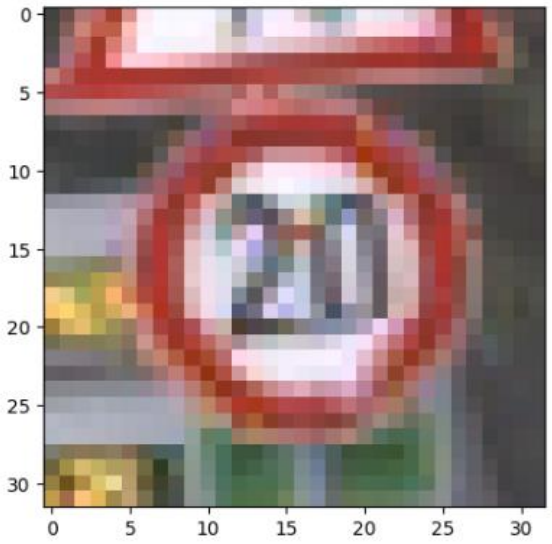
4. Загружаем dataset «CIFAR-10» и отредактируем его.

```
cifar_mean = [0.491, 0.482, 0.447]
cifar_std = [0.202, 0.199, 0.201]
cifar_dim = 32

cifar_min, cifar_max = get_clip_bounds(cifar_mean,
                                       cifar_std,
                                       cifar_dim)
cifar_min = cifar_min.to(device)
cifar_max = cifar_max.to(device)

cifar_tf = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(
        mean=cifar_mean,
        std=cifar_std)])

cifar_tf_train = transforms.Compose([
    transforms.RandomCrop(
        size=cifar_dim,
        padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=cifar_mean,
        std=cifar_std)])

cifar_tf_inv = transforms.Compose([
    transforms.Normalize(
        mean=[0.0, 0.0, 0.0],
        std=np.divide(1.0, cifar_std)),
    transforms.Normalize(
        mean=np.multiply(-1.0, cifar_mean),
        std=[1.0, 1.0, 1.0])])

cifar_temp = datasets.CIFAR10(root='datasets/cifar-10', train=True,
                              download=True, transform=cifar_tf_train)
cifar_train, cifar_val = random_split(cifar_temp, [40000, 10000])

cifar_test = datasets.CIFAR10(root='datasets/cifar-10', train=False,
                              download=True, transform=cifar_tf)
cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer',
                 'dog', 'frog', 'horse', 'ship', 'truck']
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to datasets/cifar-10/cifar-10-python.tar.gz
100%|██████████| 170498071/170498071 [00:07<00:00, 21840047.81it/s]
Extracting datasets/cifar-10/cifar-10-python.tar.gz to datasets/cifar-10
Files already downloaded and verified
```

5. Результат на выходе.

```
<matplotlib.image.AxesImage at 0x7a9de4970700>
```



6. Производим выходные слои модели, для осуществления классификации изображений.

```
x_train, x_val, y_train, y_val = train_test_split(data, labels, test_size=0.3, random_state=1)
```

```
img_size = (224,224)
model = Sequential()
model.add(ResNet50(include_top = False, pooling = 'avg'))
model.add(Dropout(0.1))
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.1))
model.add(Dense(43, activation = 'softmax'))
model.layers[2].trainable = False
```

7. Для валидации возьмем 30% процентов тренировочного набора.

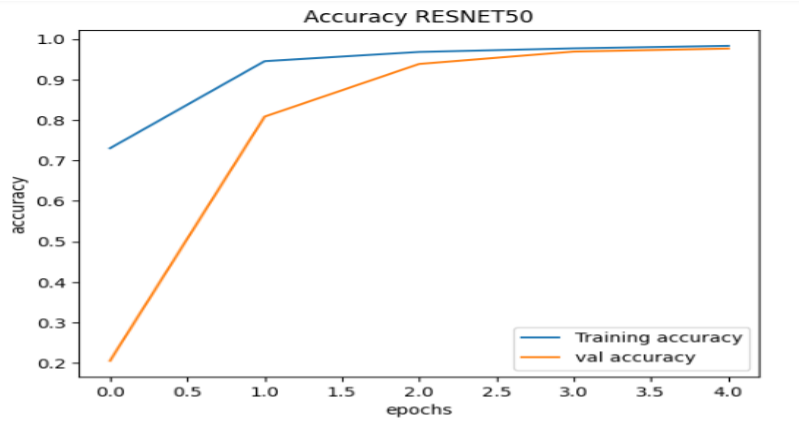
```
model.compile(loss = 'categorical_crossentropy', metrics = ['accuracy'])
history = model.fit(x_train, y_train, validation_data =(x_val, y_val), epochs = 5, batch_size = 64)
```

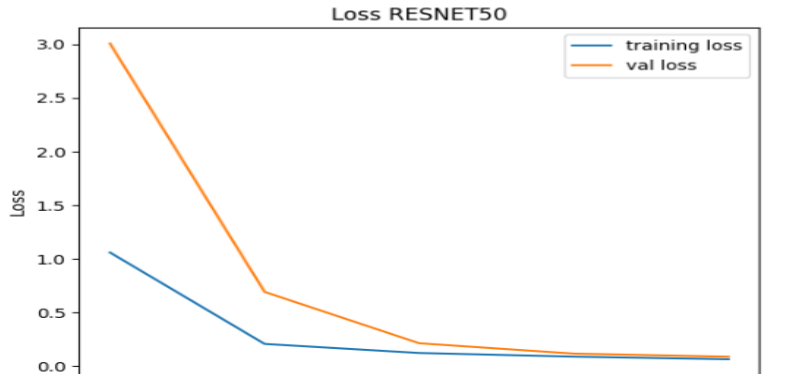
```
Epoch 1/5
429/429 [==============================] - 93s 145ms/step - loss: 1.0585 - accuracy: 0.7303 - val_loss: 3.0066 - val_accuracy: 0.2051
Epoch 2/5
429/429 [==============================] - 52s 121ms/step - loss: 0.2063 - accuracy: 0.9456 - val_loss: 0.6915 - val_accuracy: 0.8085
Epoch 3/5
429/429 [==============================] - 62s 145ms/step - loss: 0.1217 - accuracy: 0.9683 - val_loss: 0.2125 - val_accuracy: 0.9387
Epoch 4/5
429/429 [==============================] - 52s 122ms/step - loss: 0.0877 - accuracy: 0.9772 - val_loss: 0.1157 - val_accuracy: 0.9692
Epoch 5/5
429/429 [==============================] - 56s 132ms/step - loss: 0.0637 - accuracy: 0.9834 - val_loss: 0.0869 - val_accuracy: 0.9767
```

```
save_model(model, 'ResNet50.h5')
with open('history_ResNet50.pkl', 'wb') as file:
    pickle.dump(history.history, file)
!cp ResNet50.h5 drive/MyDrive/ResNet50.h5
```

```
<ipython-input-8-7662eceeb572>:1: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered lega
  save_model(model, 'ResNet50.h5')
```

8. Построим графики и посмотрим, что будет на выходе.





9. Сформируем тестовую выборку и оценим точность модели.

```
test = pd.read_csv("Test.csv")
test_imgs = test['Path'].values
data = []
for img in test_imgs:
    img = image.load_img(img, target_size=(32, 32))
    img_array = image.img_to_array(img)
    img_array = img_array / 255
    data.append(img_array)
data = np.array(data)
y_test = test['ClassId'].values.tolist()
y_test = np.array(y_test)
y_test = to_categorical(y_test, 43)
```

```
loss, accuracy = model.evaluate(data, y_test)
print(f"Test loss: {loss}")
print(f"Test accuracy: {accuracy}")
```

```
395/395 [==============================] - 16s 40ms/step - loss: 0.3758 - accuracy: 0.9154
Test loss: 0.3758074641227722
Test accuracy: 0.9153602719306946
```

10. Используем готовый набор для тренировки.

```
del model
del history
img_size = (224,224)
model = Sequential()
model.add(VGG16(include_top=False, pooling = 'avg'))
model.add(Dropout(0.1))
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.1))
model.add(Dense(43, activation = 'softmax'))
model.layers[2].trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_orderin
58889256/58889256 [==============================] - 2s 0us/step
```

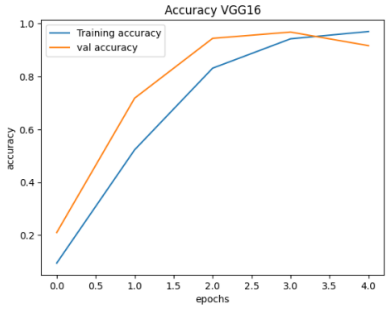
```
model.compile(loss = 'categorical_crossentropy', metrics = ['accuracy'])
history = model.fit(x_train, y_train, validation_data =(x_val, y_val), epochs = 5, batch_size = 64)
```

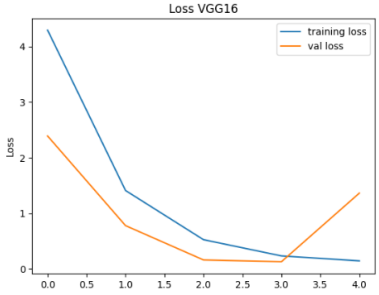
```
Epoch 1/5
429/429 [==============================] - 46s 97ms/step - loss: 4.2956 - accuracy: 0.0943 - val_loss: 2.3912 - val_a
Epoch 2/5
429/429 [==============================] - 43s 101ms/step - loss: 1.4106 - accuracy: 0.5227 - val_loss: 0.7790 - val_
Epoch 3/5
429/429 [==============================] - 50s 117ms/step - loss: 0.5264 - accuracy: 0.8311 - val_loss: 0.1619 - val_
Epoch 4/5
429/429 [==============================] - 43s 101ms/step - loss: 0.2343 - accuracy: 0.9421 - val_loss: 0.1296 - val_
Epoch 5/5
429/429 [==============================] - 45s 105ms/step - loss: 0.1453 - accuracy: 0.9696 - val_loss: 1.3639 - val_
```

```
save_model(model, 'VGG16.h5')
with open('history_VGG16.pkl', 'wb') as file:
    pickle.dump(history.history, file)
!cp ResNet50.h5 drive/MyDrive/ResNet50.h5
```

```
<ipython-input-16-dfaa1c6ae2f2>:1: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This fi
  save_model(model, 'VGG16.h5')
```

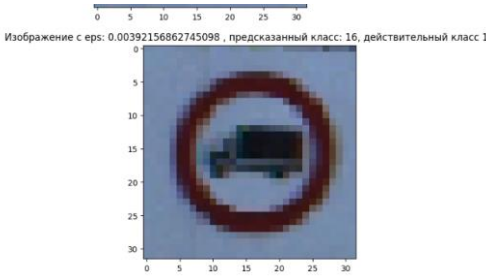
11. Сделаем отображение графиков точности и потерь для модели VGG16.

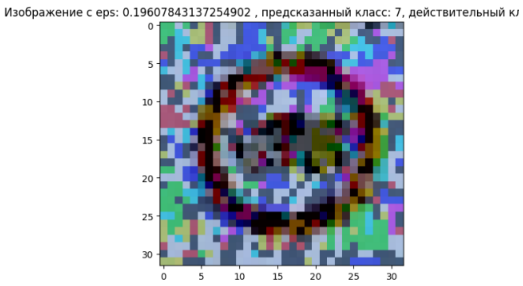




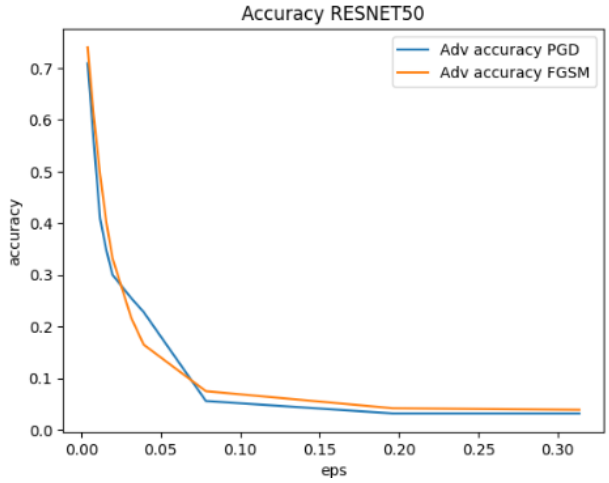
Задание 2.

1. В ходе данной работы мы должны взаимодействовать с моделью ResNet50 FGGM. Определим eps и выведем изображения.

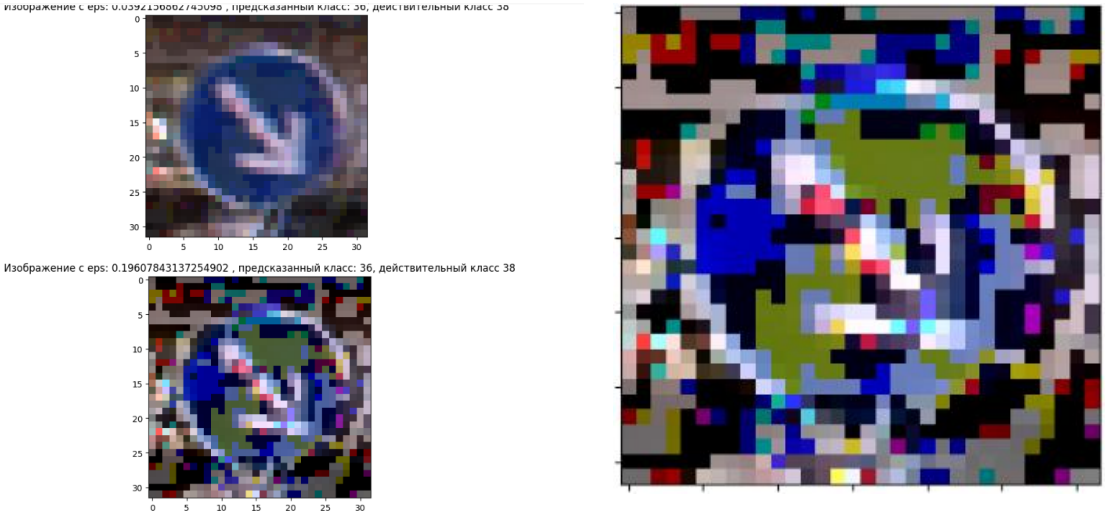




2. Теперь попробуем использовать атаку PGD для различных вариантов eps и изобразим изображения до и после атаки.

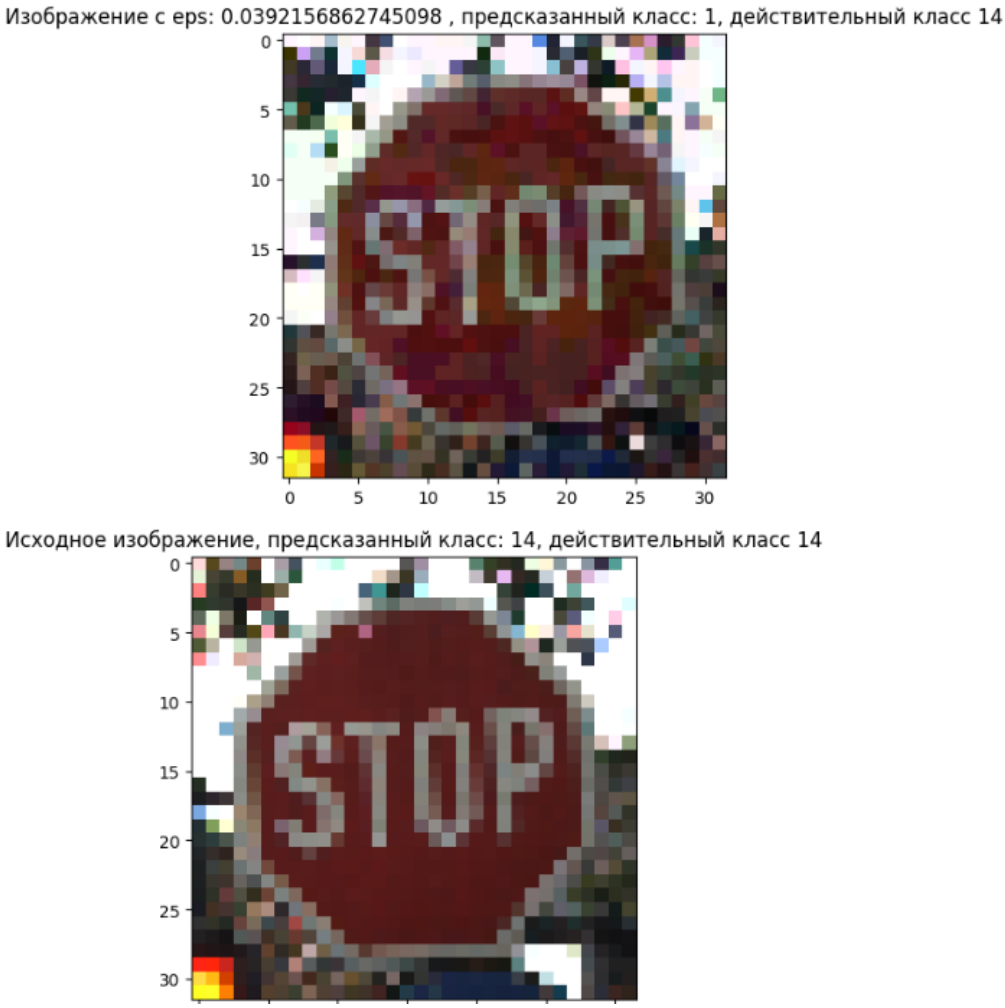


3. Приступим к созданию модели для использования атаки VGG16 FGSM и посмотрим изображения до и после атаки.

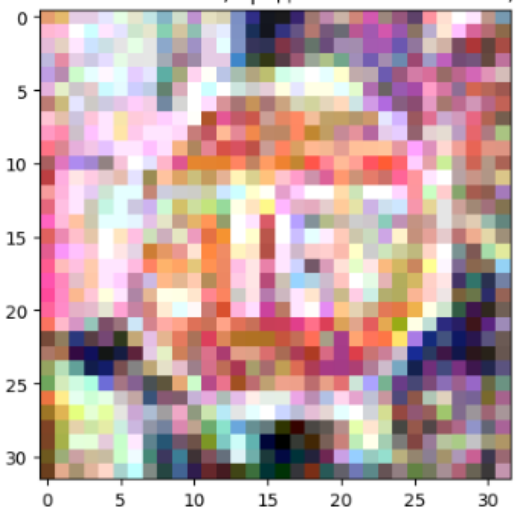


Задание 3.

1. Создадим модель для атаки Targeted FGSM Attack и отобразим модель атаки.



Изображение с eps: 0.19607843137254902 , предсказанный класс: 1, действительный класс 14



Вывод

В ходе выполнения данной лабораторной работы получен опыт работы с инструментами атак на модели машинного обучения. Были проведены некие эксперименты с атакой на модели машинного обучения.