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»

Лабораторная работа №2

по дисциплине
«Анализ защищенности систем искусственного интеллекта»

Группа: ББМО-01-22
Выполнила:
Лясникова В.М.

Проверил:
Спирин А.А.

Москва 2024

Загрузим инструмент adversarial-robustness-toolbox.

```
!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 11.9 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 44.4 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 0.10.0 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
```

Выполним импорт необходимых библиотек, предварительно загрузив датасет с дорожными знаками.

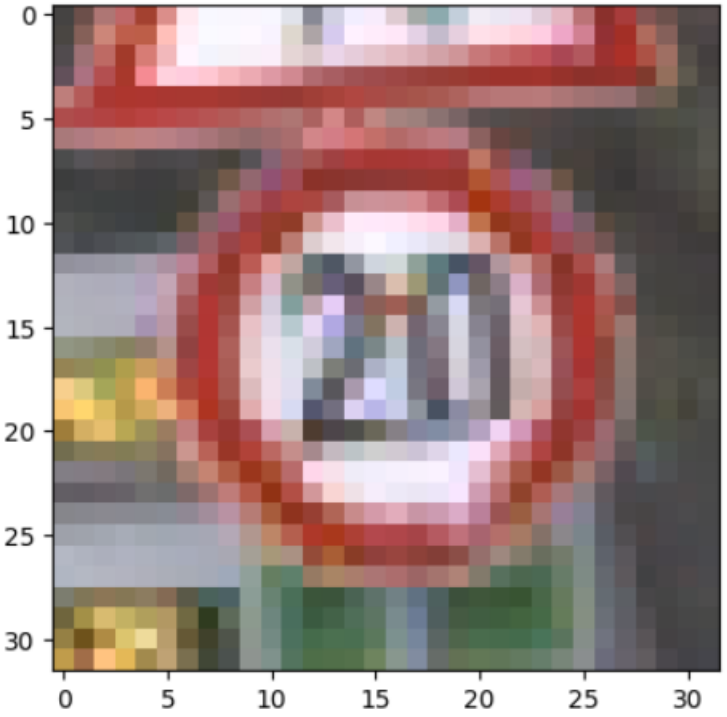
```
import cv2
import os
import torch
import random
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.applications import ResNet50
from keras.applications 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
```

Обучение классификаторов на основе глубоких нейронных сетей на датасете GTSRB.

Первое изображение.

```
plt.imshow(data[0])
```

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



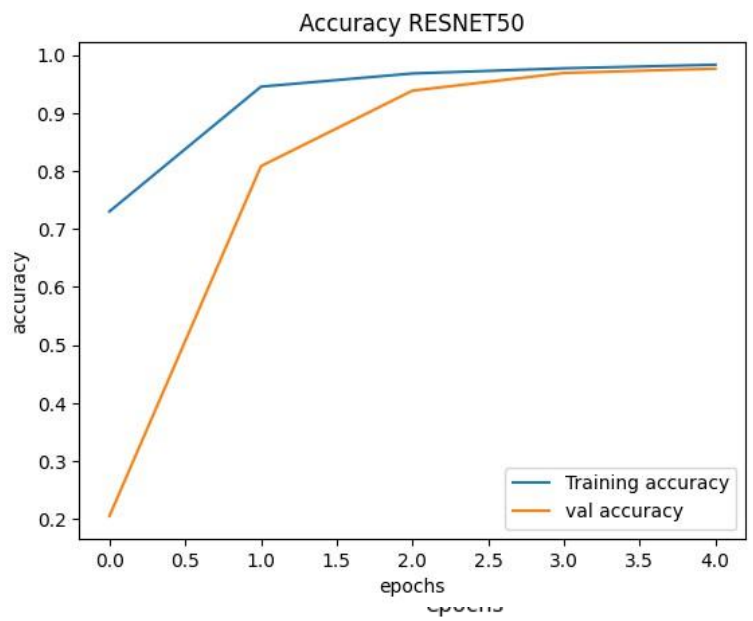
ResNet50. После разбиения на выборки можно приступить к компиляции модели, единственное, необходимо поменять выходные слои модели, для осуществления классификации 43 типов изображений.

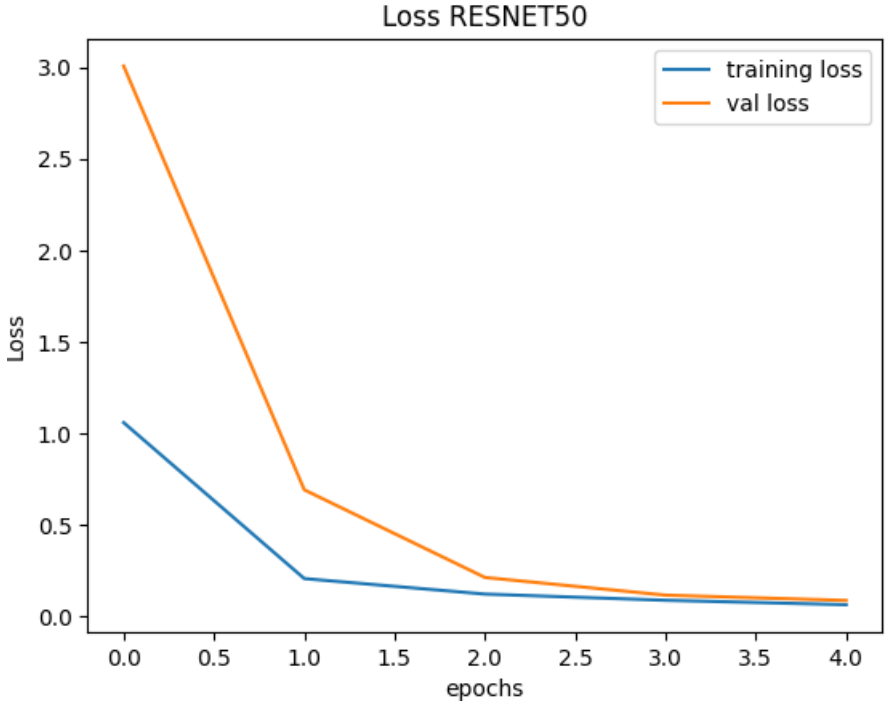
```
[ ] x_train, x_val, y_train, y_val = train_test_split(data, labels, test_size=0.3, random_state=1)
```

```
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
```

Построим необходимые графики, отражающие успешность обучения модели ResNet50. Итоговая точность увеличилась по мере роста числа эпох, однако дальнейшее увеличение эпох было уже не целесообразно.

```
plt.figure(0)
plt.plot(history.history['accuracy'], label="Training accuracy")
plt.plot(history.history['val_accuracy'], label="val accuracy")
plt.title("Accuracy RESNET50")
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.legend()
plt.figure(1)
plt.plot(history.history['loss'], label="training loss")
plt.plot(history.history['val_loss'], label="val loss")
plt.title("Loss RESNET50")
plt.xlabel("epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```





Формируем тестовую выборку немного другим способом, для определения правильной метки класса будем использовать csv таблицу с обозначением пути картинки и ее класса.

```
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)
```

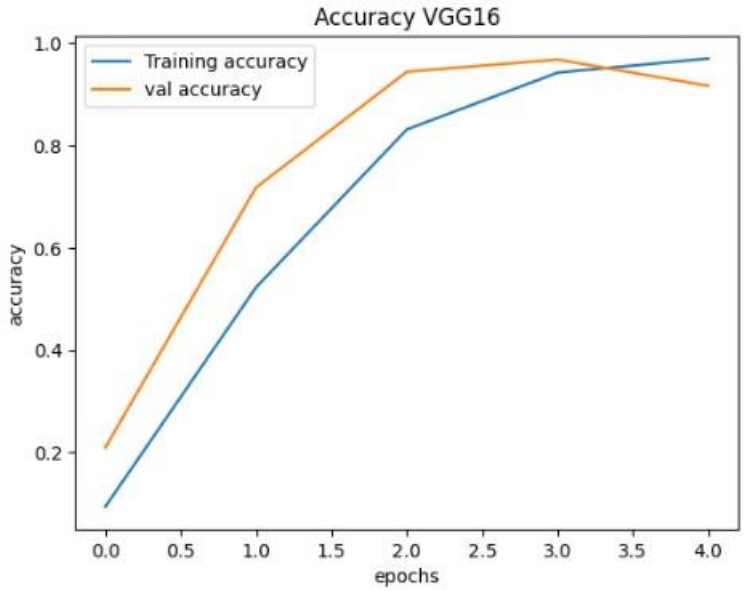
Оценим точность тестовой выборки

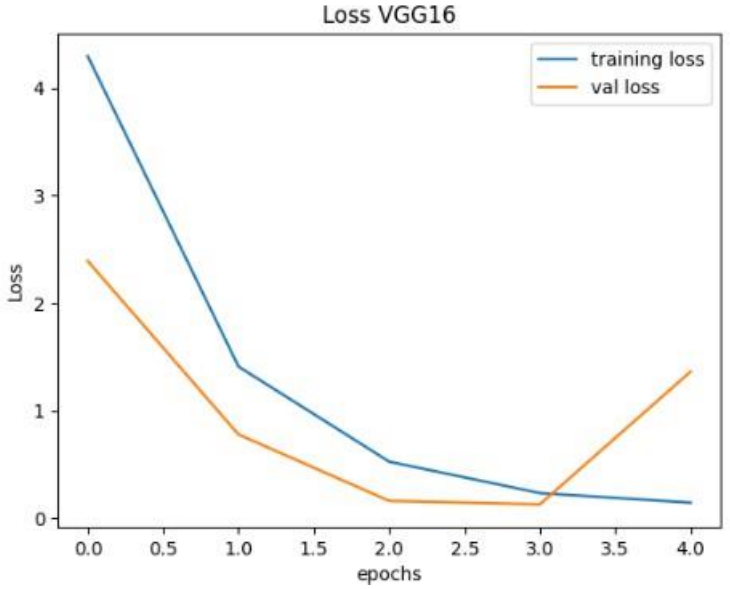
```
[ ] loss, accuracy = model.evaluate(data, y_test)
    print(f"Test loss: {loss}")
    print(f"Test accuracy: {accuracy}")

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

Итоговая точность 91%

VGG16. Загрузим уже готовый набор данных для тренировки. После разбиения на выборки можно приступить к компиляции модели, единственное, необходимо поменять выходные слои модели, для осуществления классификации 43 типов изображений.

```
plt.figure(0)
plt.plot(history.history['accuracy'], label="Training accuracy")
plt.plot(history.history['val_accuracy'], label="val accuracy")
plt.title("Accuracy VGG16")
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.legend()
plt.figure(1)
plt.plot(history.history['loss'], label="training loss")
plt.plot(history.history['val_loss'], label="val loss")
plt.title("Loss VGG16")
plt.xlabel("epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```





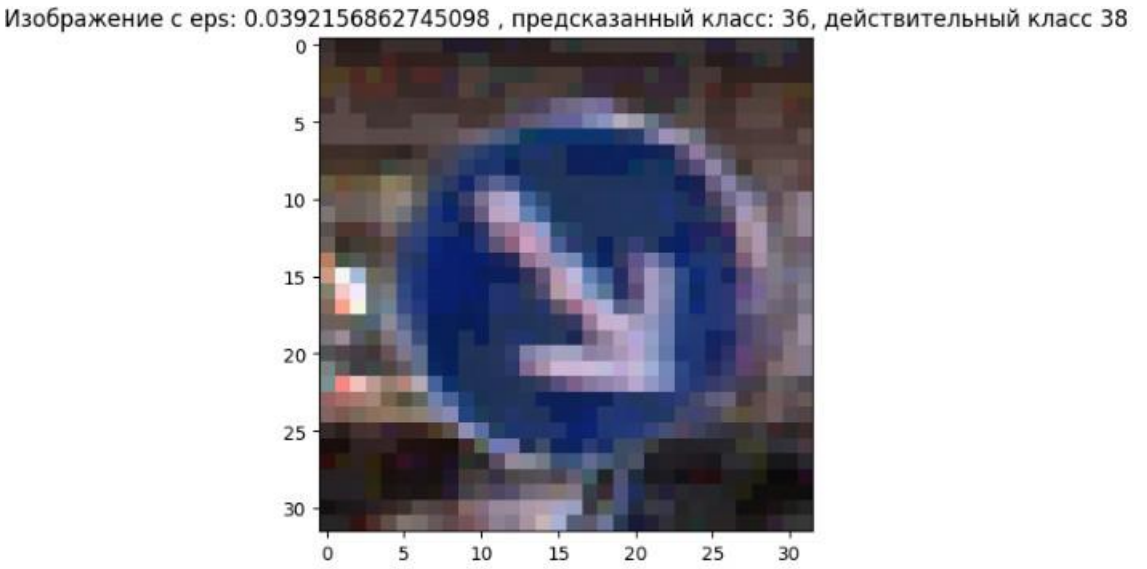
Точность также близка к 90%

Применить нецелевую атаку уклонения на основе белого ящика против моделей глубокого обучения.

```
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 Loss: {loss}")
    print(f"Adv Accuracy: {accuracy}")
    loss, accuracy = model.evaluate(x_test, y_test)
    true_accuracies.append(accuracy)
    true_losses.append(loss)
    print(f"True Loss: {loss}")
    print(f"True Accuracy: {accuracy}")
```

```
Eps: 0.00392156862745098
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training_v1.py:2335: UserWarning: `Model.s
  updates = self.state_updates
Adv Loss: 1.440965347290039
Adv Accuracy: 0.7400000095367432
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.00784313725490196
Adv Loss: 2.591770582199097
Adv Accuracy: 0.6050000190734863
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.011764705882352941
Adv Loss: 3.5671306858062746
Adv Accuracy: 0.49399998784065247
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.01568627450980392
Adv Loss: 4.398491260528565
Adv Accuracy: 0.4020000100135803
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.0196078431372549
Adv Loss: 5.081645603179932
Adv Accuracy: 0.3319999873638153
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.03137254901960784
Adv Loss: 6.415355705261231
Adv Accuracy: 0.21699999272823334
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.0392156862745098
Adv Loss: 6.922237411499023
Adv Accuracy: 0.16500000655651093
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.0784313725490196
Adv Loss: 7.8761302185058595
Adv Accuracy: 0.07500000298023224
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.19607843137254902
Adv Loss: 7.650692733764648
Adv Accuracy: 0.041999999433755875
True Loss: 0.3387975747585297
True Accuracy: 0.9129999876022339
Eps: 0.3137254901960784
Adv Loss: 7.515952522277832
```

Точность с eps = 10/255 = 0,039. менее 60% и равна 17%. С увеличением eps усиливается зашумление картинки, но, следовательно, и вероятность обнаружения атаки. Отобразим изображения до и после атаки.



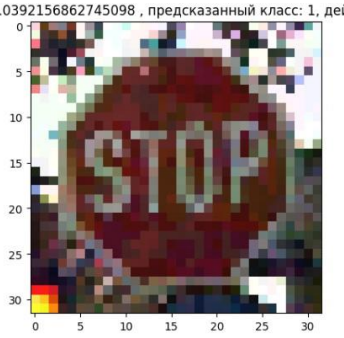
Результаты эксперимента с VGG16 такие же, как и с ResNet50

Применение целевой атаки уклонения методом белого против моделей глубокого обучения.

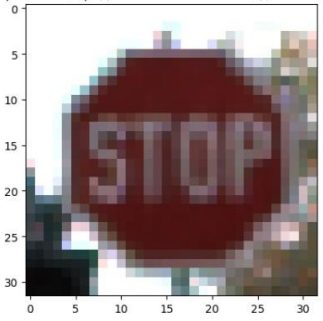
```
test = pd.read_csv("Test.csv")
test_imgs = test['Path'].values
data = []
y_test = []
labels = test['ClassId'].values.tolist()
i = -1
for img in test_imgs:
    i += 1
    if labels[i] != 14:
      continue
    img = image.load_img(img, target_size=(32, 32))
    img_array = image.img_to_array(img)
    img_array = img_array /255
    data.append(img_array)
    y_test.append(labels[i])
data = np.array(data)
y_test = np.array(y_test)
y_test = to_categorical(y_test, 43)
```

```
[ ] model=load_model('ResNet50.h5')
    tf.compat.v1.disable_eager_execution()
    t_class = 1
    t_class = to_categorical(t_class, 43)
    t_classes = np.tile(t_class, (270, 1))
    x_test = data
    classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
    attack_fgsm = FastGradientMethod(estimator=classifier, eps=0.2, targeted=True, batch_size=64)
    eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
    for eps in eps_range:
        attack_fgsm.set_params(**{'eps': eps})
        print(f"Eps: {eps}")
        x_test_adv = attack_fgsm.generate(x_test, t_classes)
        loss, accuracy = model.evaluate(x_test_adv, y_test)
        print(f"Adv Loss: {loss}")
        print(f"Adv Accuracy: {accuracy}")
        loss, accuracy = model.evaluate(x_test, y_test)
        print(f"True Loss: {loss}")
        print(f"True Accuracy: {accuracy}")
```

Изображение с eps: 0.0392156862745098 , предсказанный класс: 1, действительный класс 14



Исходное изображение, предсказанный класс: 14, действительный класс 14



Результат:

- метод FGSM для целевых атак применять не следует, с ростом eps и соответственно шума, классификация действительно ошибочна, однако класс, который мы хотим навязать модели, наиболее точно определяется при eps **10/255**, далее модель будет определять совсем не те значения, что мы указали (не label 1).
- PGD отлично подходит для целевой атаки, при больших eps модель почти всегда будет определять класс 14 как 1, но изображение будет слишком зашумленным, для данной атаки - оптимальным значением будет **50/255**, такие значения eps сильно зашумляют изображение, но и классификация класса 1 как класса 14 будет наиболее выраженной.