МИНОБРНАУКИРОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА— Российский технологический университет»
РТУМИРЭА

Институт кибербезопасности и цифровых технологий направление 10.04.01

«Информационная безопасность»

Кафедра КБ-4«Интеллектуальные системы информационной безопасности»

Лабораторная работа №2

По дисциплине
«Анализ защищенности систем искусственного интеллекта»

Выполнил:

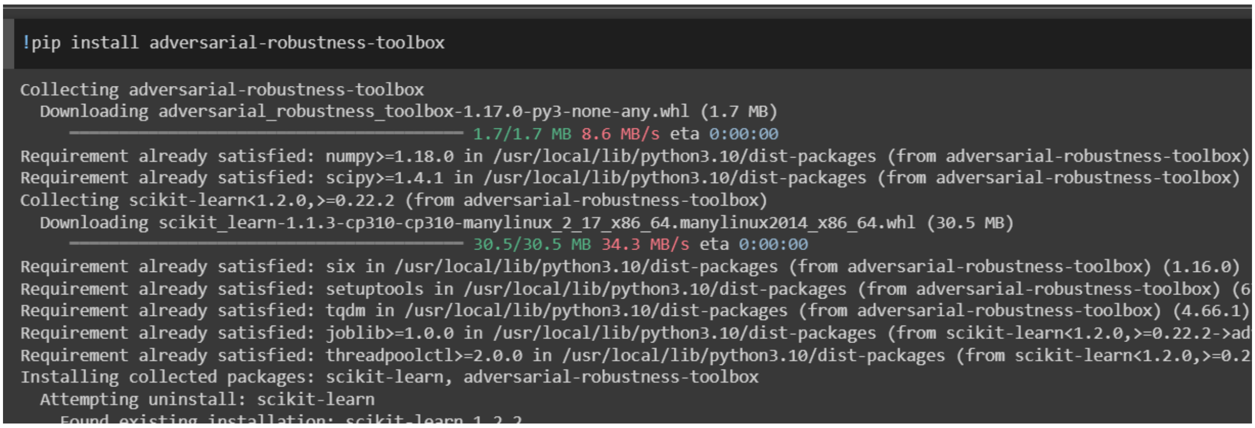
Суслов Антон Константинович

Группа: ББМО-02-22

Москва 2023

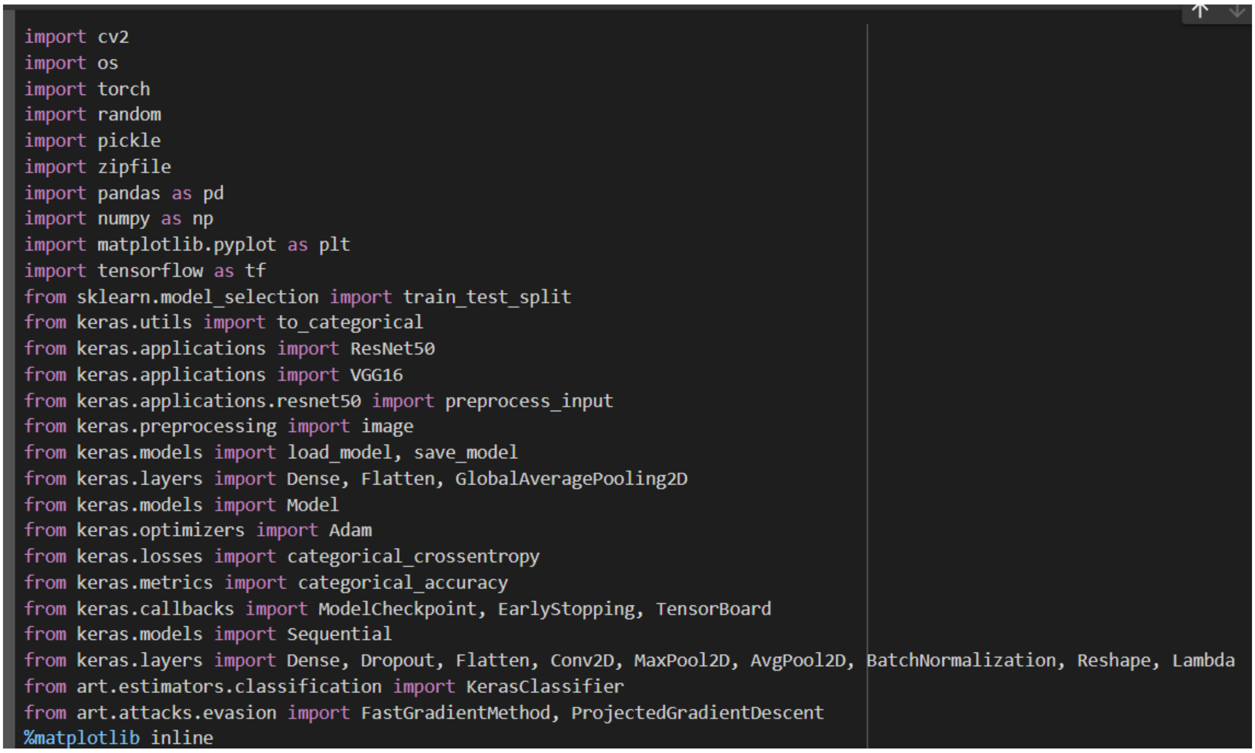
Задание 1

Установка adversarial-robustness-toolbox



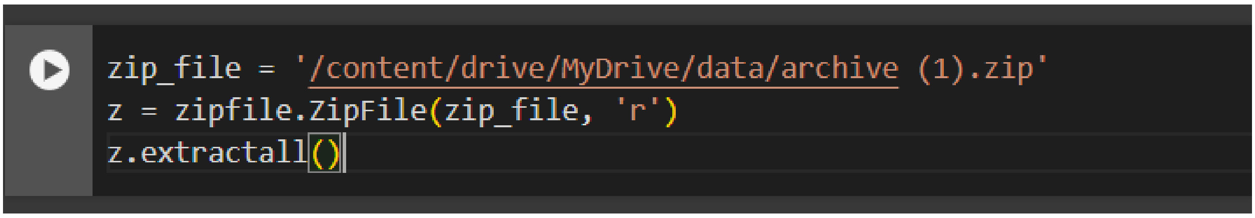
```
!pip install adversarial-robustness-toolbox

Collecting adversarial-robustness-toolbox
  Downloading adversarial_robustness_toolbox-1.17.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 8.6 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)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from adversarial-robustness-toolbox)
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 34.3 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) (6
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->ad
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn<1.2.0,>=0.2
Installing collected packages: scikit-learn, adversarial-robustness-toolbox
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
```

Импорт библиотек



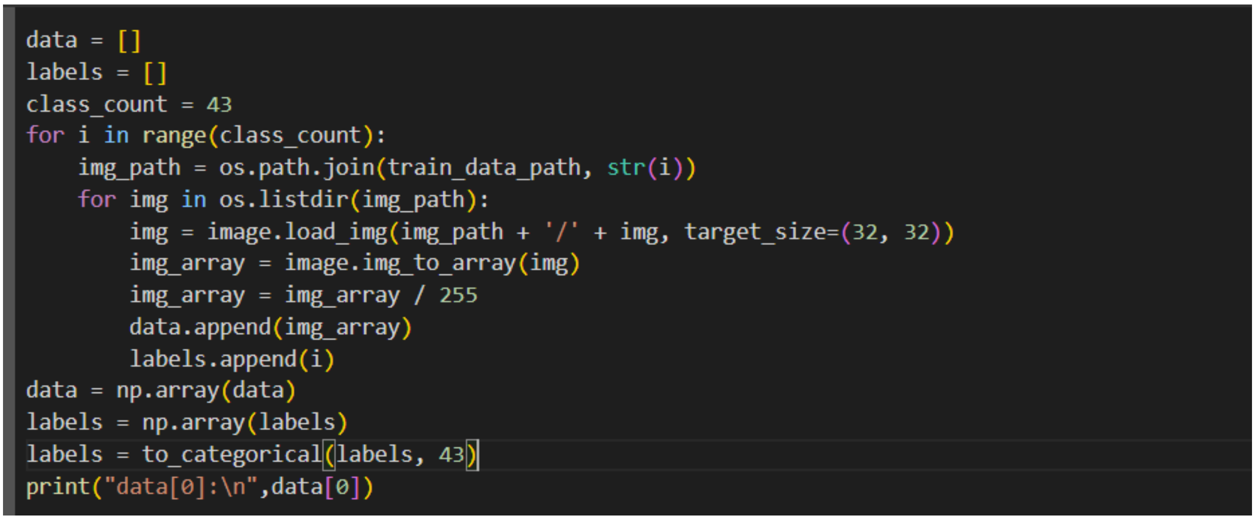
```python
import cv2
import os
import torch
import random
import pickle
import zipfile
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.applications import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

Распаковка архива



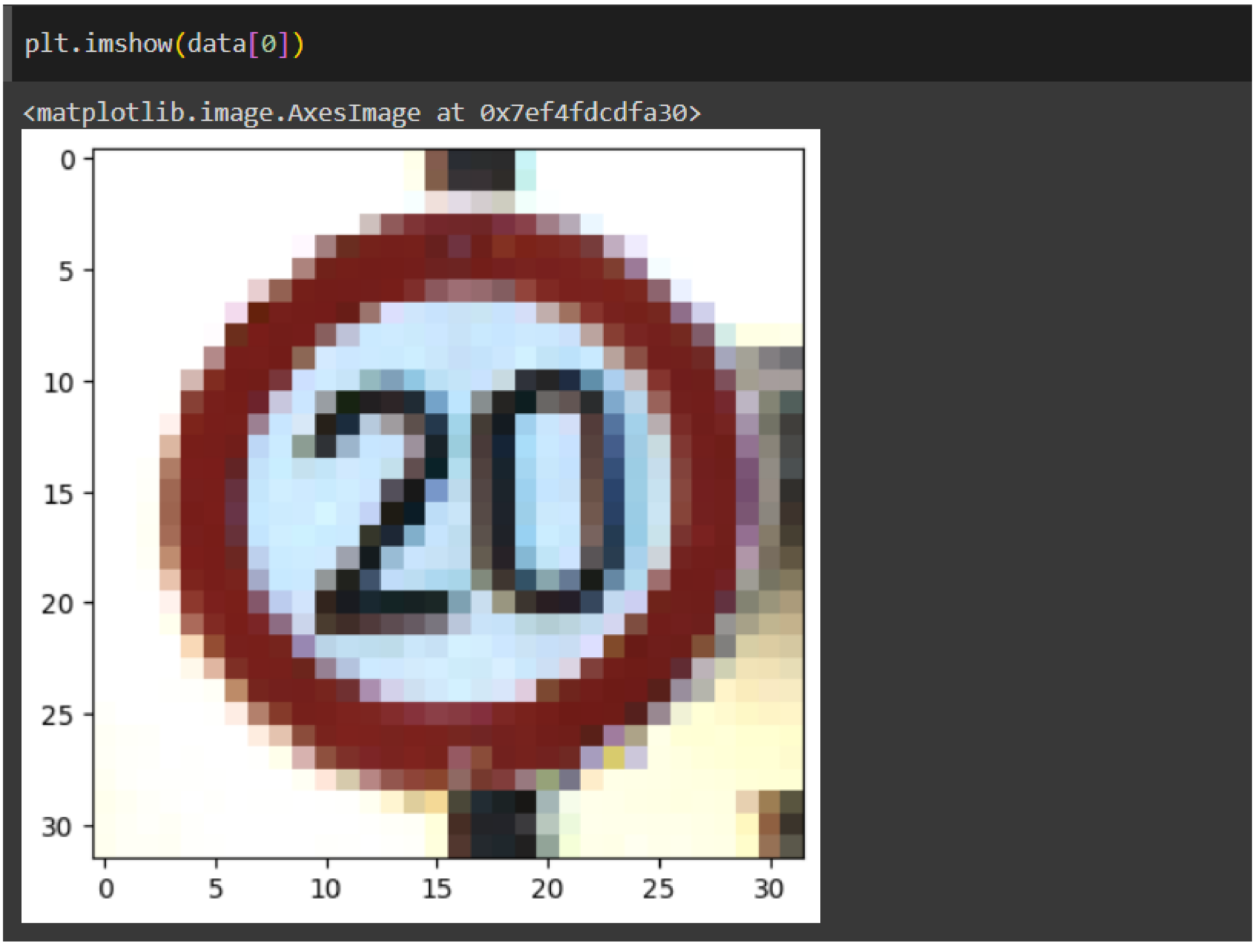
```python
zip_file = '/content/drive/MyDrive/data/archive (1).zip'
z = zipfile.ZipFile(zip_file, 'r')
z.extractall()
```

Чтение и предварительная обработка изображений



```
data = []
labels = []
class_count = 43
for i in range(class_count):
    img_path = os.path.join(train_data_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)
print("data[0]:\n",data[0])
```

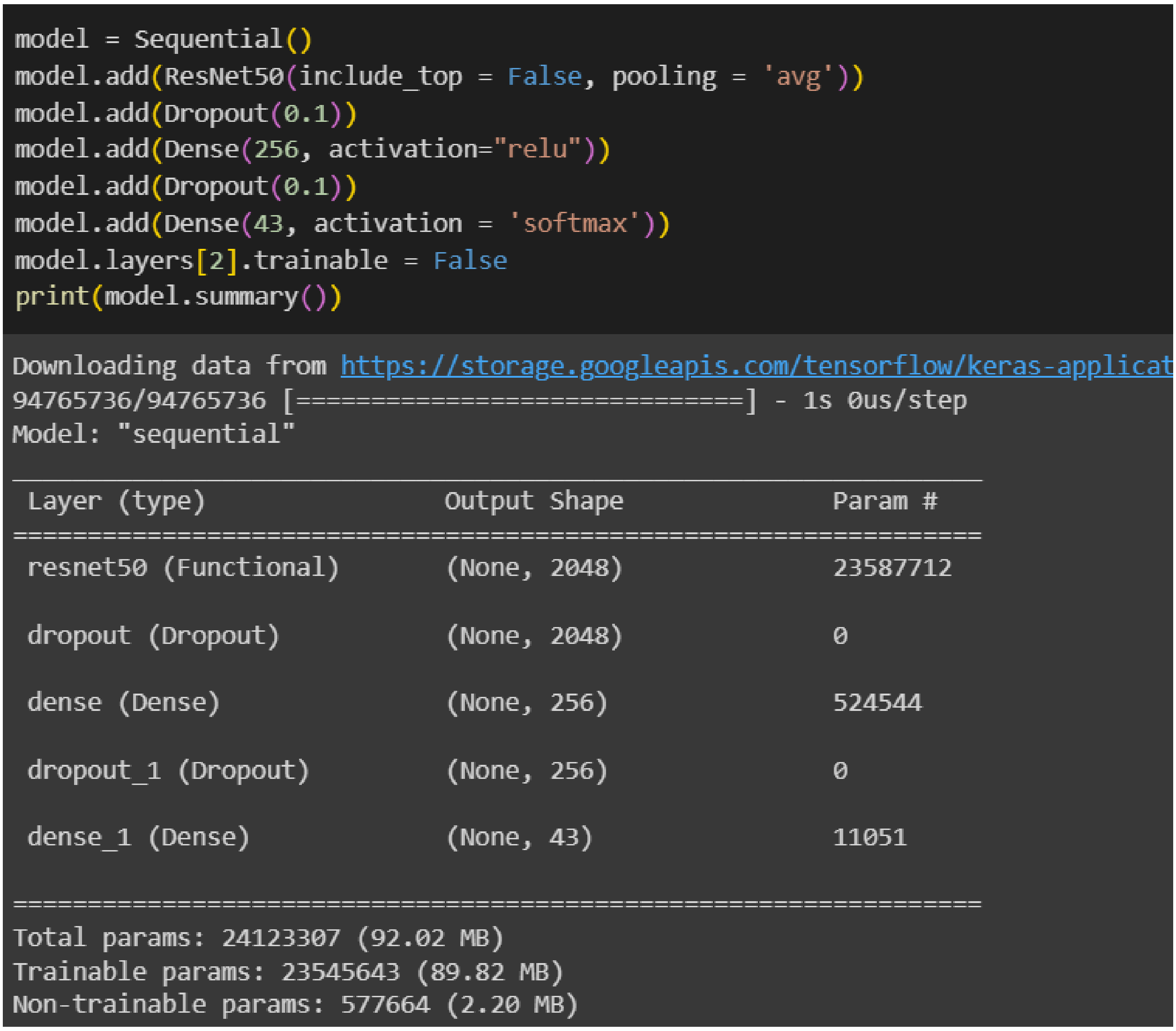
Первое изображение



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

```
<matplotlib.image.AxesImage at 0x7ef4fdcdfa30>
```

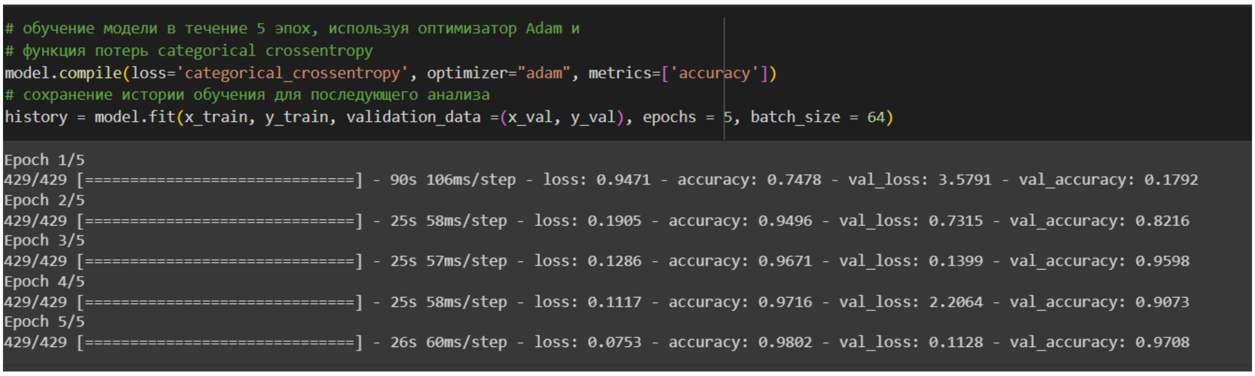
Создание модели для классификации изображений



```
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
print(model.summary())
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicat
94765736/94765736 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 256)               524544

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 43)                11051

=================================================================
Total params: 24123307 (92.02 MB)
Trainable params: 23545643 (89.82 MB)
Non-trainable params: 577664 (2.20 MB)
```

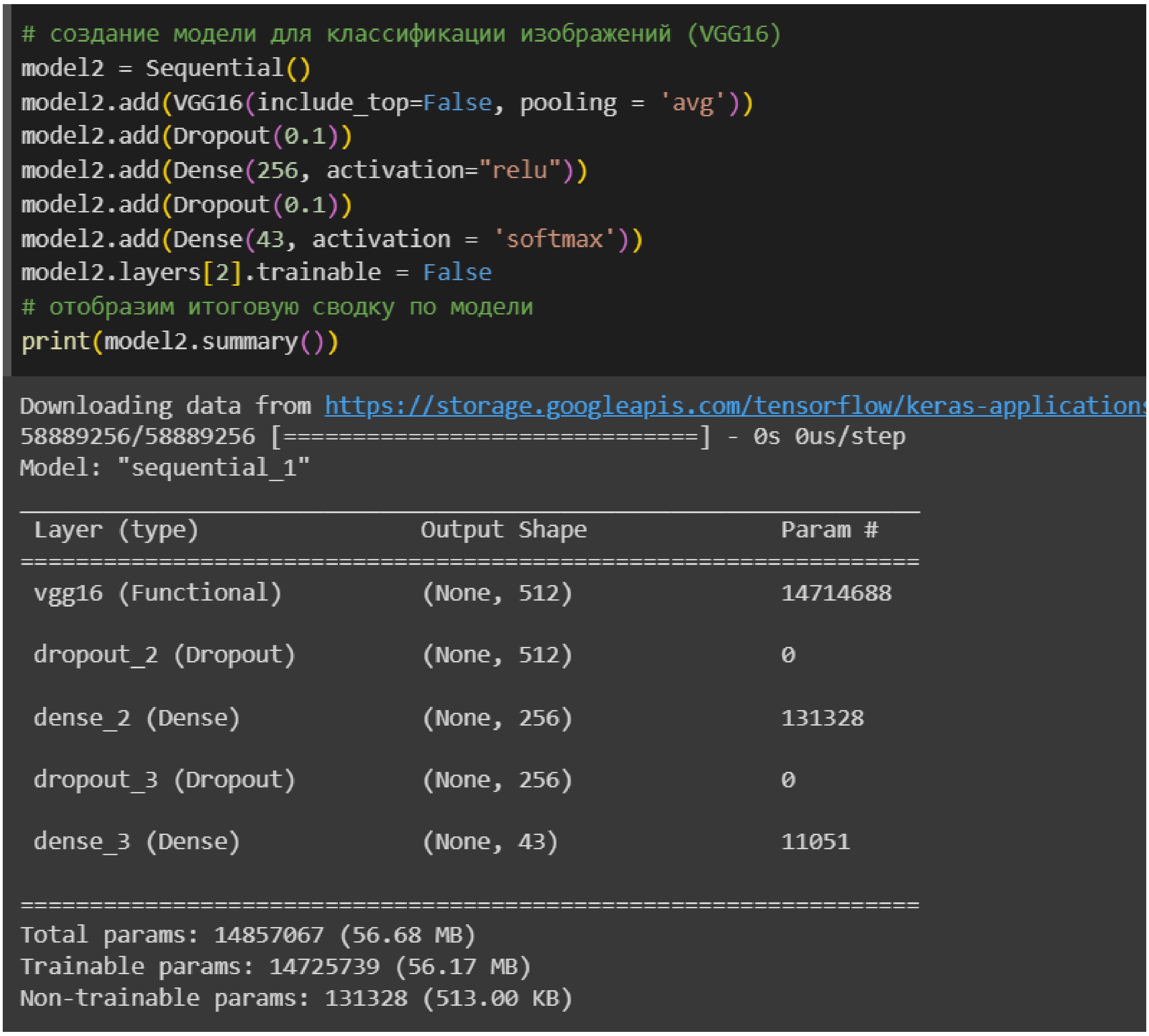
Обучаем модель ResNet50 с заданными параметрами



```
# обучение модели в течение 5 эпох, используя оптимизатор Adam и
# функция потерь categorical crossentropy
model.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
# сохранение истории обучения для последующего анализа
history = model.fit(x_train, y_train, validation_data =(x_val, y_val), epochs = 5, batch_size = 64)
```

```
Epoch 1/5
429/429 [==============================] - 90s 106ms/step - loss: 0.9471 - accuracy: 0.7478 - val_loss: 3.5791 - val_accuracy: 0.1792
Epoch 2/5
429/429 [==============================] - 25s 58ms/step - loss: 0.1905 - accuracy: 0.9496 - val_loss: 0.7315 - val_accuracy: 0.8216
Epoch 3/5
429/429 [==============================] - 25s 57ms/step - loss: 0.1286 - accuracy: 0.9671 - val_loss: 0.1399 - val_accuracy: 0.9598
Epoch 4/5
429/429 [==============================] - 25s 58ms/step - loss: 0.1117 - accuracy: 0.9716 - val_loss: 2.2064 - val_accuracy: 0.9073
Epoch 5/5
429/429 [==============================] - 26s 60ms/step - loss: 0.0753 - accuracy: 0.9802 - val_loss: 0.1128 - val_accuracy: 0.9708
```

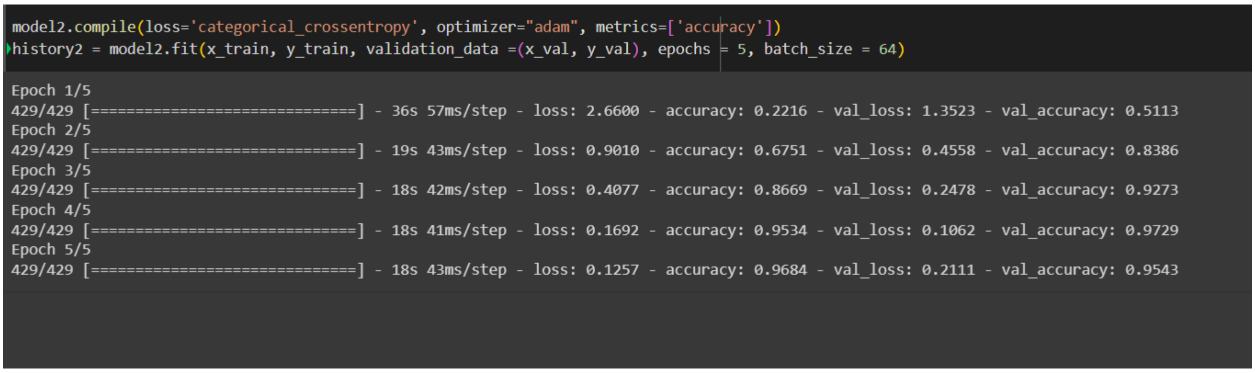
Создание второй модели для классификации изображений



```
# создание модели для классификации изображений (VGG16)
model2 = Sequential()
model2.add(VGG16(include_top=False, pooling = 'avg'))
model2.add(Dropout(0.1))
model2.add(Dense(256, activation="relu"))
model2.add(Dropout(0.1))
model2.add(Dense(43, activation = 'softmax'))
model2.layers[2].trainable = False
# отобразим итоговую сводку по модели
print(model2.summary())
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications
58889256/58889256 [==============================] - 0s 0us/step
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 512)               14714688

 dropout_2 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 256)               131328

 dropout_3 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 43)                11051

=================================================================
Total params: 14857067 (56.68 MB)
Trainable params: 14725739 (56.17 MB)
Non-trainable params: 131328 (513.00 KB)
```

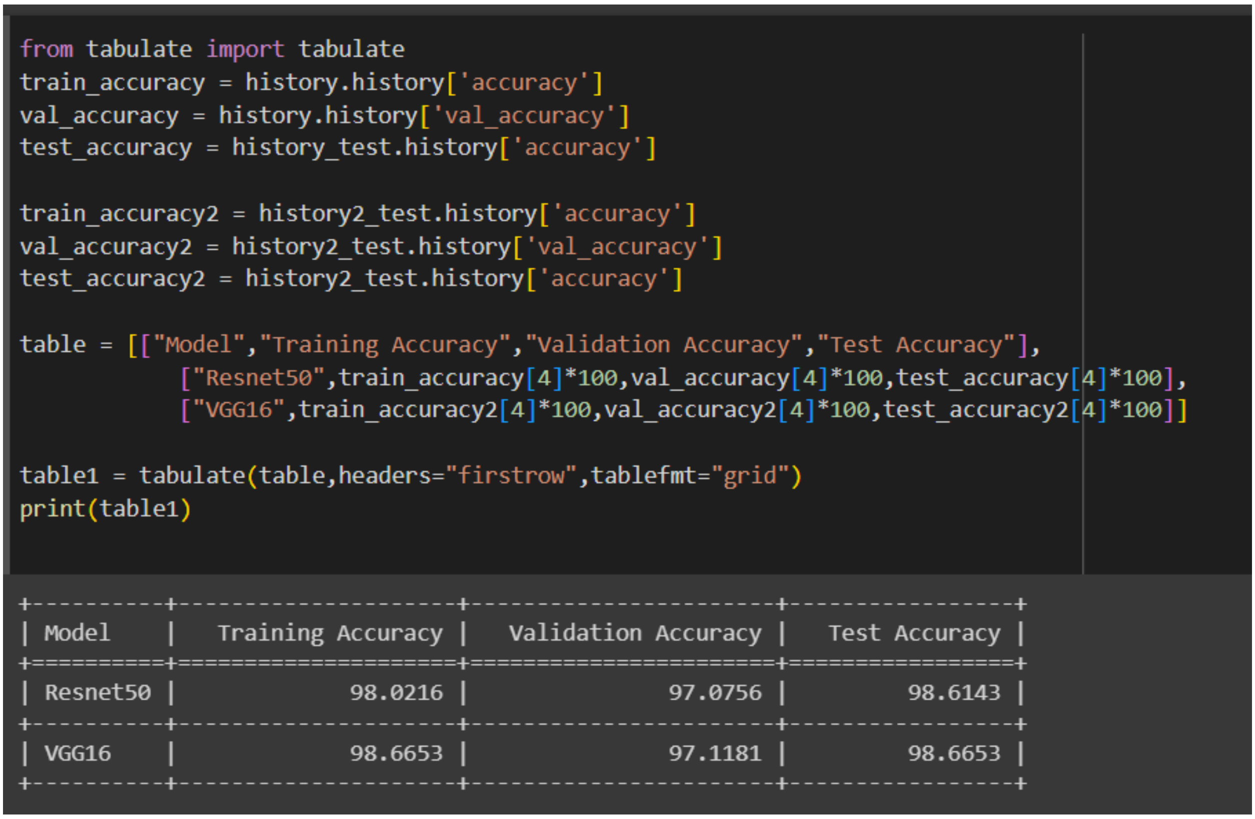
Обучение модели VGG16 с заданные параметрами



```
model2.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
history2 = model2.fit(x_train, y_train, validation_data =(x_val, y_val), epochs = 5, batch_size = 64)
```

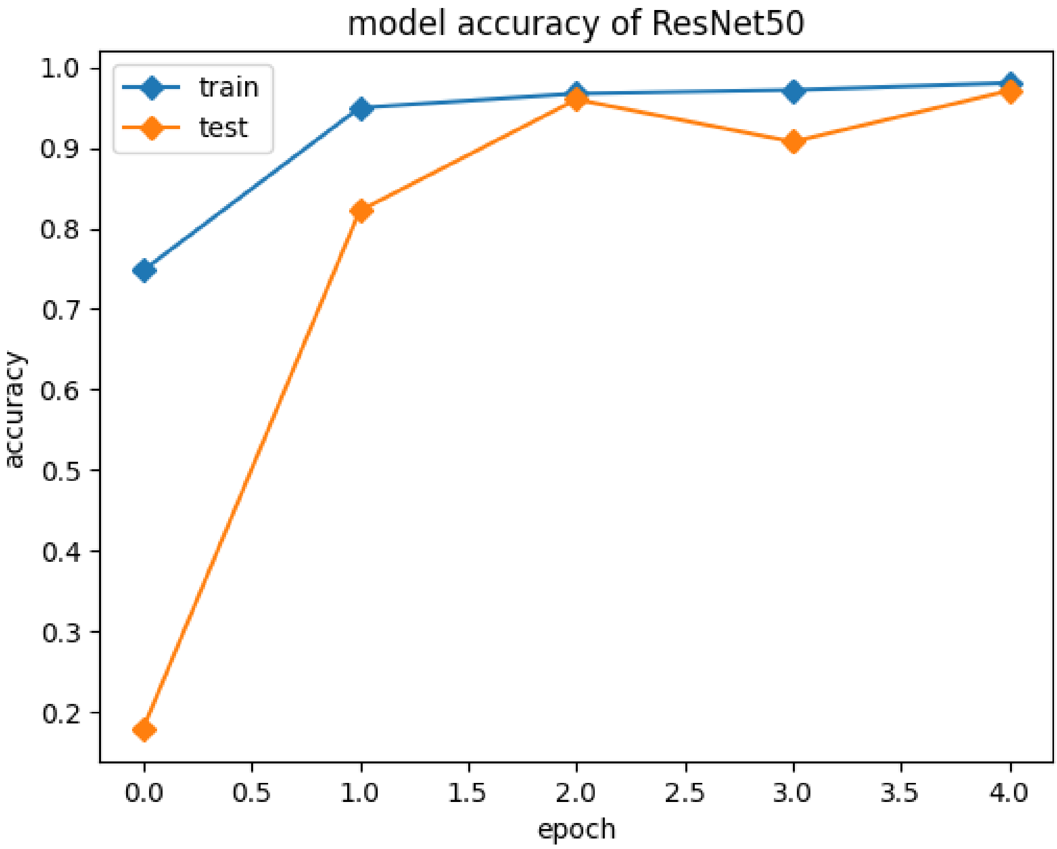
```
Epoch 1/5
429/429 [==============================] - 36s 57ms/step - loss: 2.6600 - accuracy: 0.2216 - val_loss: 1.3523 - val_accuracy: 0.5113
Epoch 2/5
429/429 [==============================] - 19s 43ms/step - loss: 0.9010 - accuracy: 0.6751 - val_loss: 0.4558 - val_accuracy: 0.8386
Epoch 3/5
429/429 [==============================] - 18s 42ms/step - loss: 0.4077 - accuracy: 0.8669 - val_loss: 0.2478 - val_accuracy: 0.9273
Epoch 4/5
429/429 [==============================] - 18s 41ms/step - loss: 0.1692 - accuracy: 0.9534 - val_loss: 0.1062 - val_accuracy: 0.9729
Epoch 5/5
429/429 [==============================] - 18s 43ms/step - loss: 0.1257 - accuracy: 0.9684 - val_loss: 0.2111 - val_accuracy: 0.9543
```

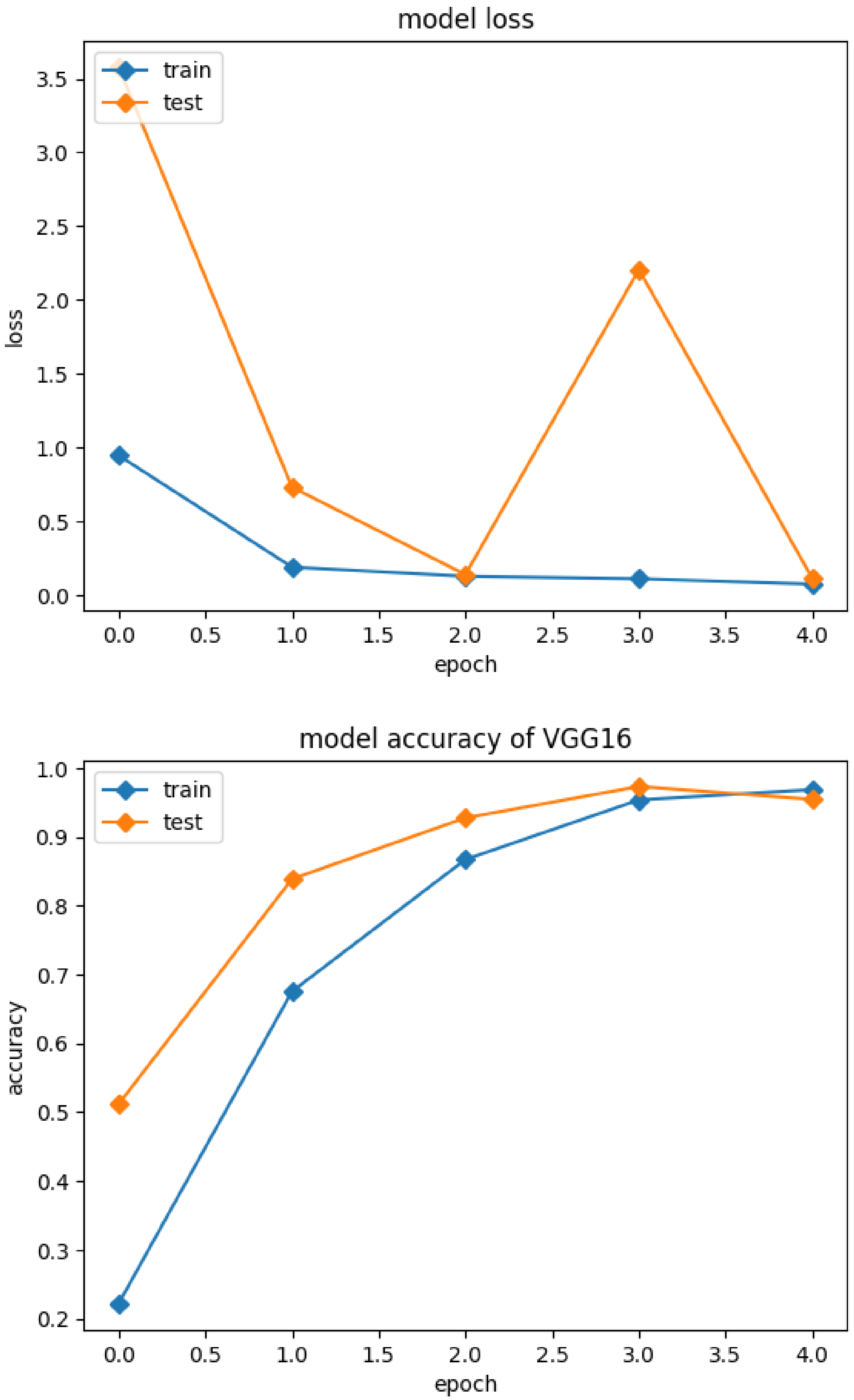
Оценка двух моделей

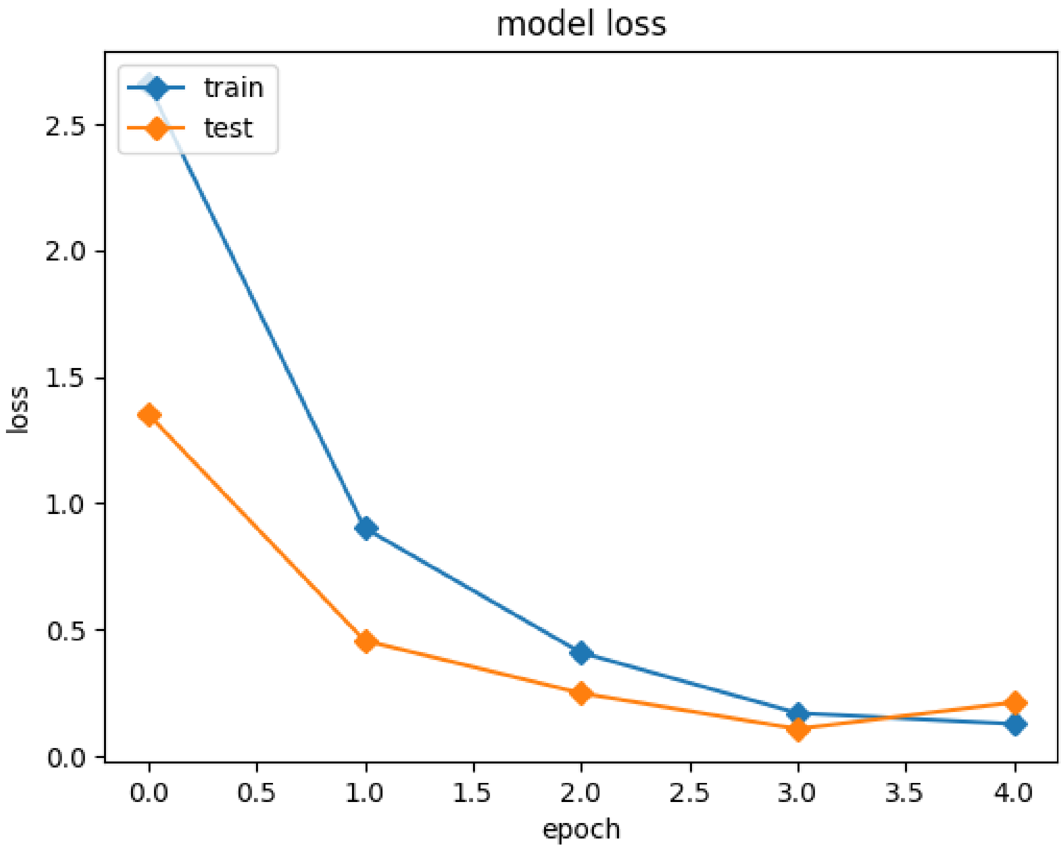


```
from tabulate import tabulate
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
test_accuracy = history_test.history['accuracy']

train_accuracy2 = history2_test.history['accuracy']
val_accuracy2 = history2_test.history['val_accuracy']
test_accuracy2 = history2_test.history['accuracy']

table = [["Model","Training Accuracy","Validation Accuracy","Test Accuracy"],
            ["Resnet50",train_accuracy[4]*100,val_accuracy[4]*100,test_accuracy[4]*100],
            ["VGG16",train_accuracy2[4]*100,val_accuracy2[4]*100,test_accuracy2[4]*100]]

table1 = tabulate(table,headers="firstrow",tablefmt="grid")
print(table1)
```

```
+----------+---------------------+-----------------------+-----------------+
| Model    |   Training Accuracy |   Validation Accuracy |   Test Accuracy |
+==========+=====================+=======================+=================+
| Resnet50 |             98.0216 |               97.0756 |         98.6143 |
+----------+---------------------+-----------------------+-----------------+
| VGG16    |             98.6653 |               97.1181 |         98.6653 |
+----------+---------------------+-----------------------+-----------------+
```



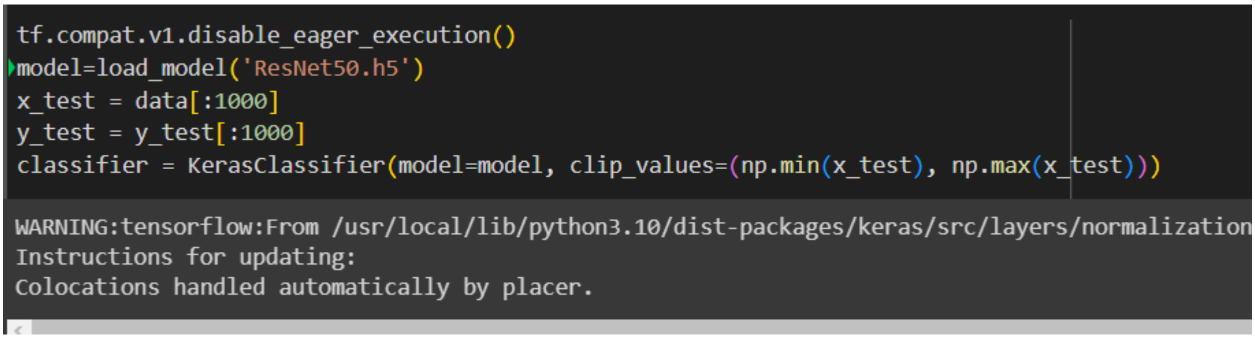

model loss
loss
epoch
train
test
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
model accuracy of VGG16
accuracy
epoch
train
test
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0



Вывод: как видно из таблицы выше, модели показывают схожие результаты.

Задание 2

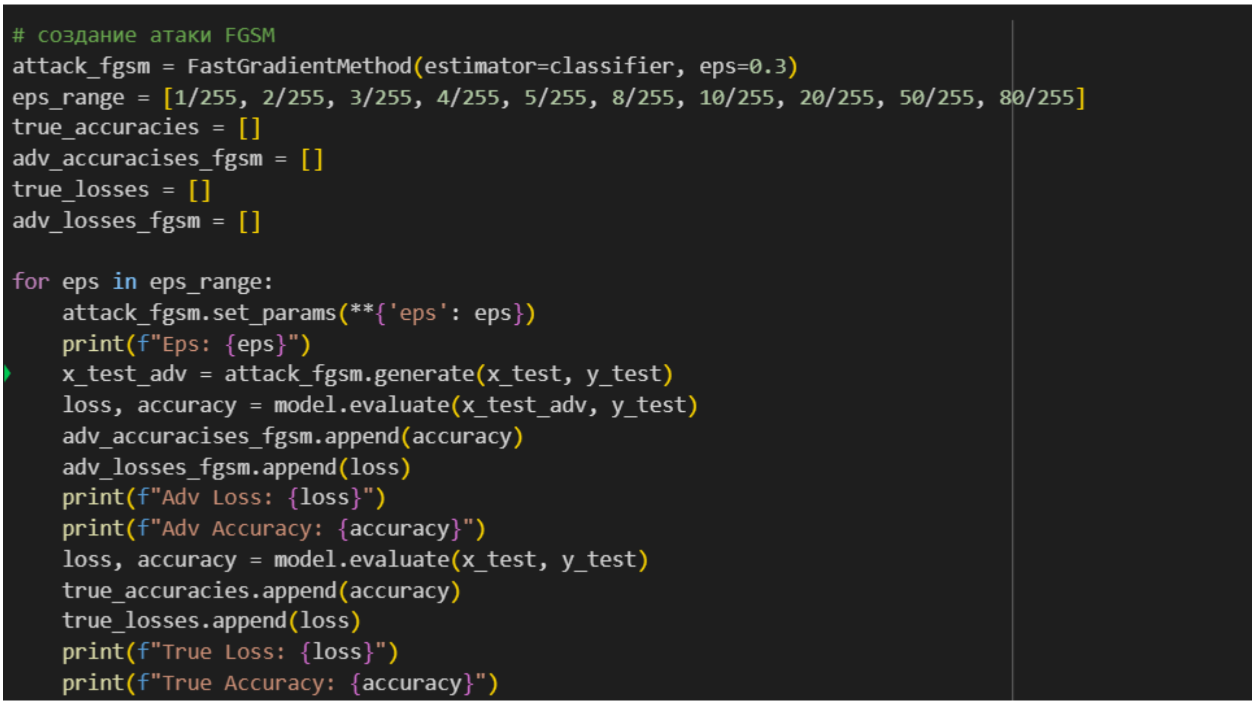
Загружаем модель из предыдущего задания



```
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)))
```

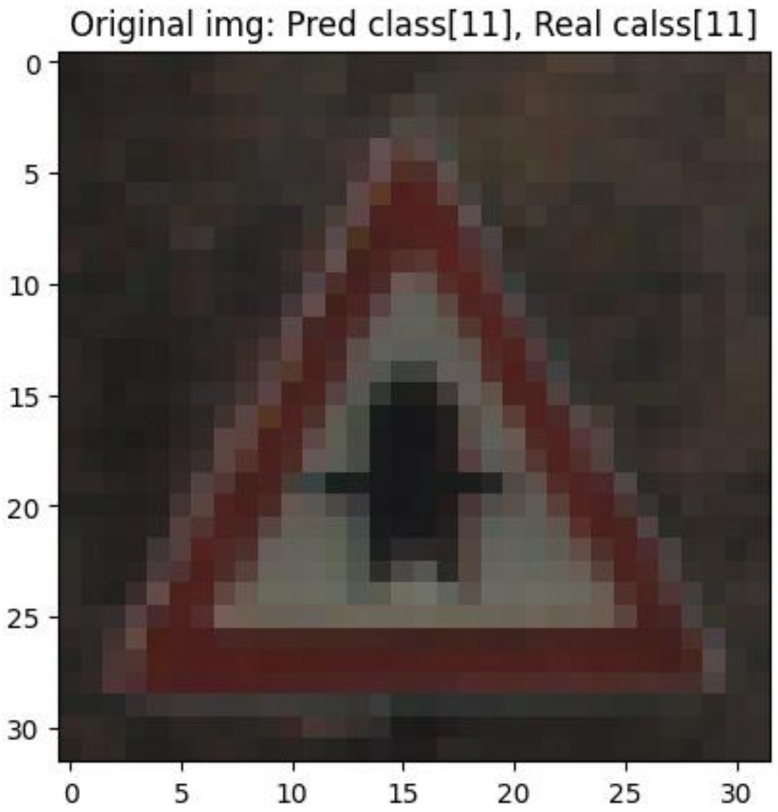
```
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/keras/src/layers/normalization
Instructions for updating:
Colocations handled automatically by placer.
```

Создание атаки FGSM



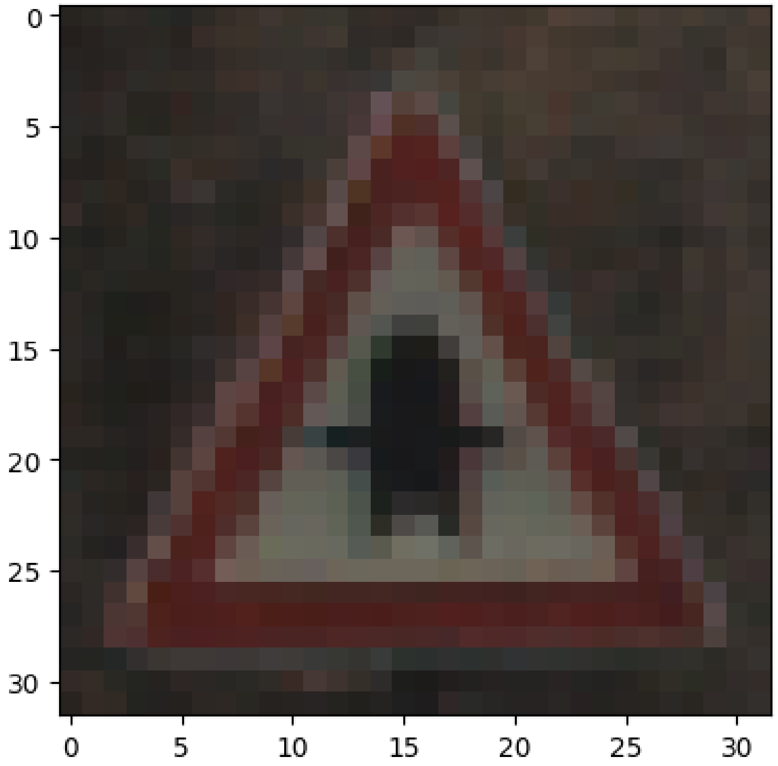
```
# создание атаки FGSM
attack_fgsm = FastGradientMethod(estimator=classifier, eps=0.3)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = []
adv_accuracises_fgsm = []
true_losses = []
adv_losses_fgsm = []

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}")
```

Отображение исходных и адверсариальных изображений

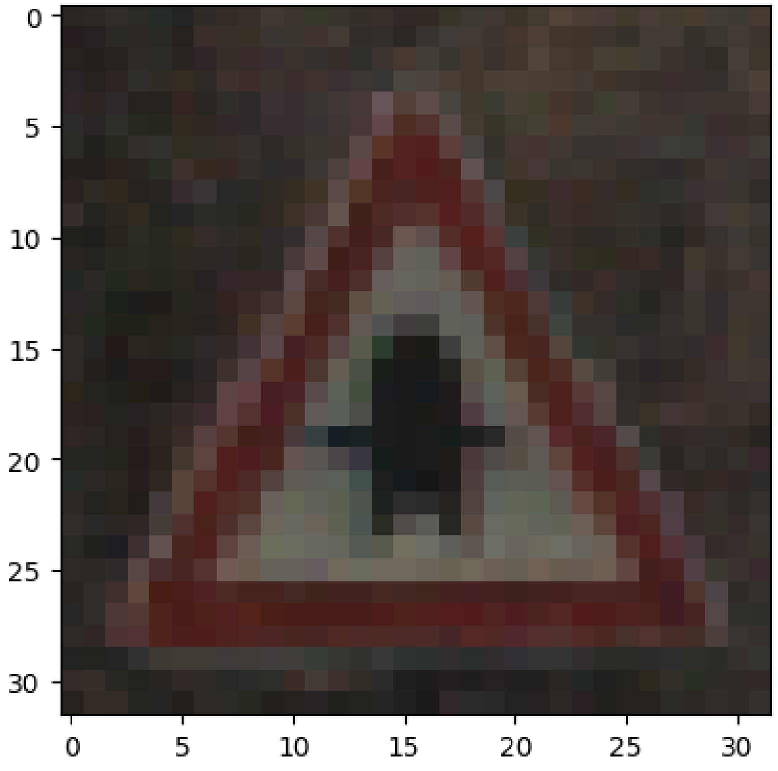


Original img: Pred class[11], Real calss[11]

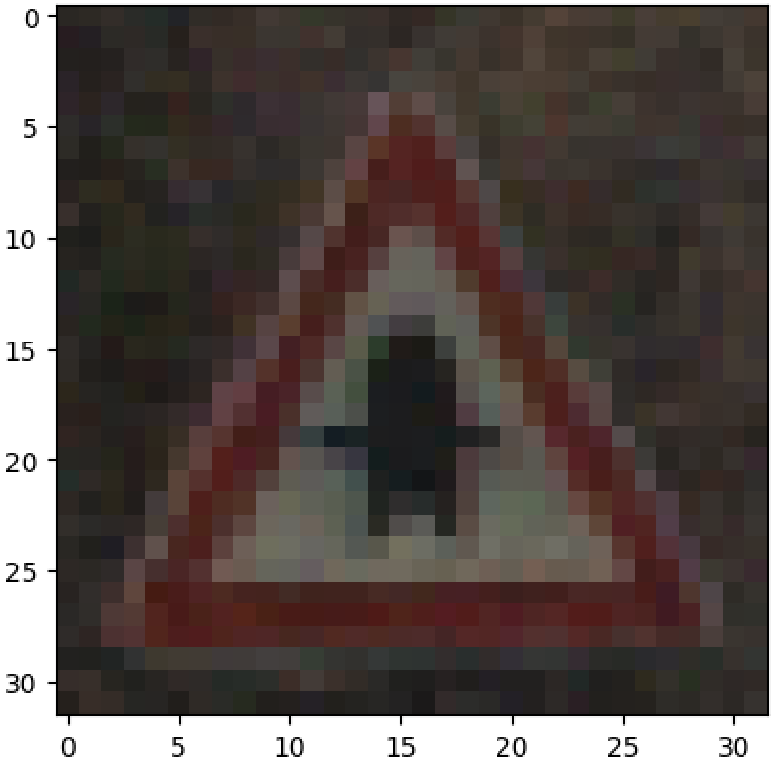
eps 0.00392156862745098: Pred class[11], Real class[11]



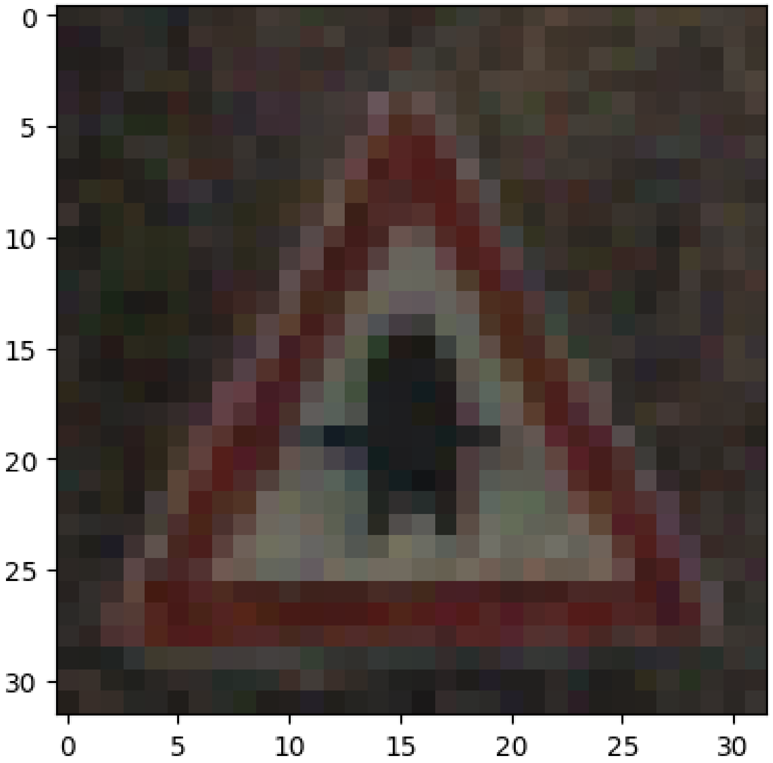
eps 0.011764705882352941: Pred class[11], Real class[11]



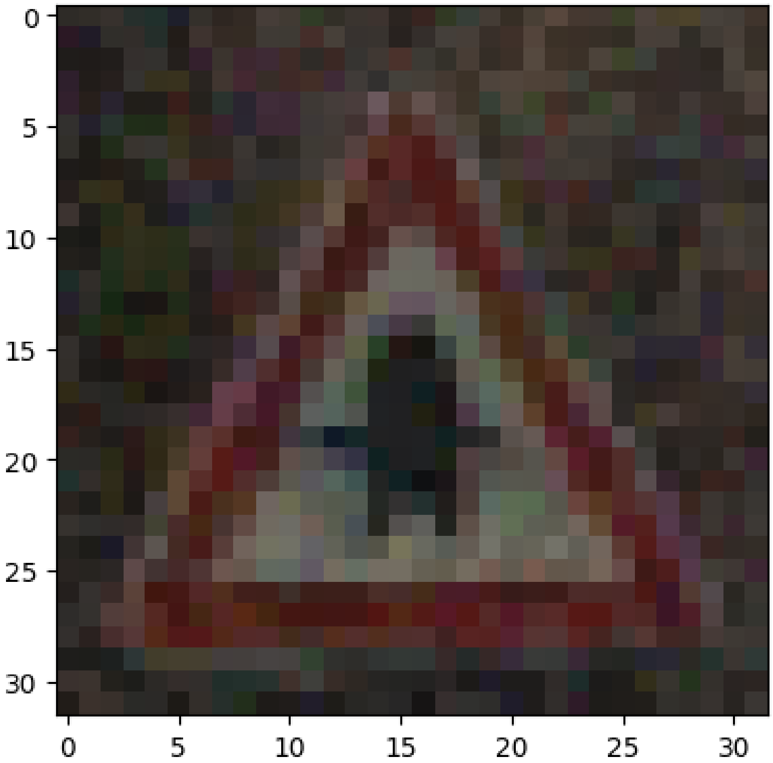
eps 0.01568627450980392: Pred class[11], Real class[11]



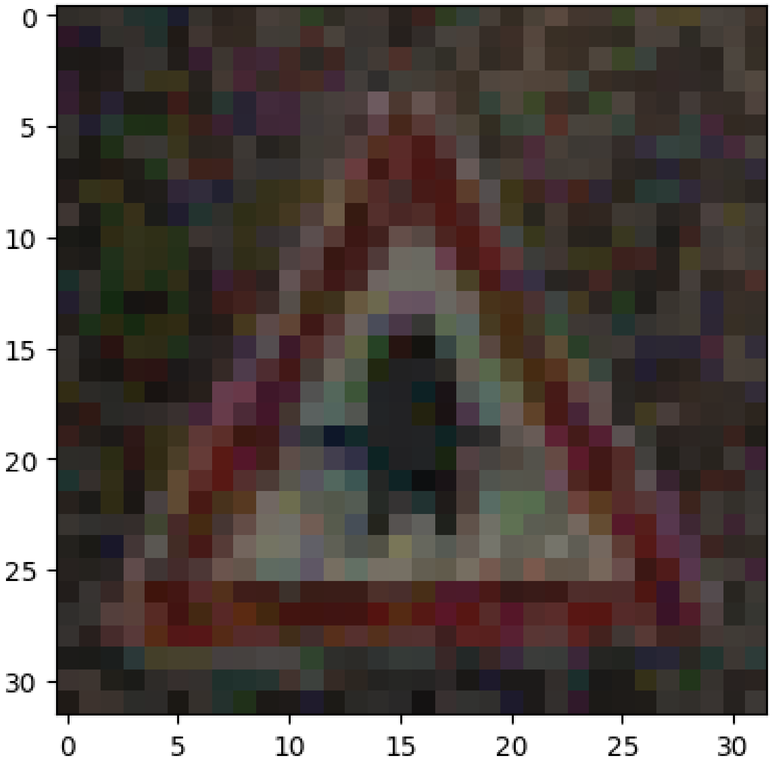
eps 0.0196078431372549: Pred class[25], Real class[11]



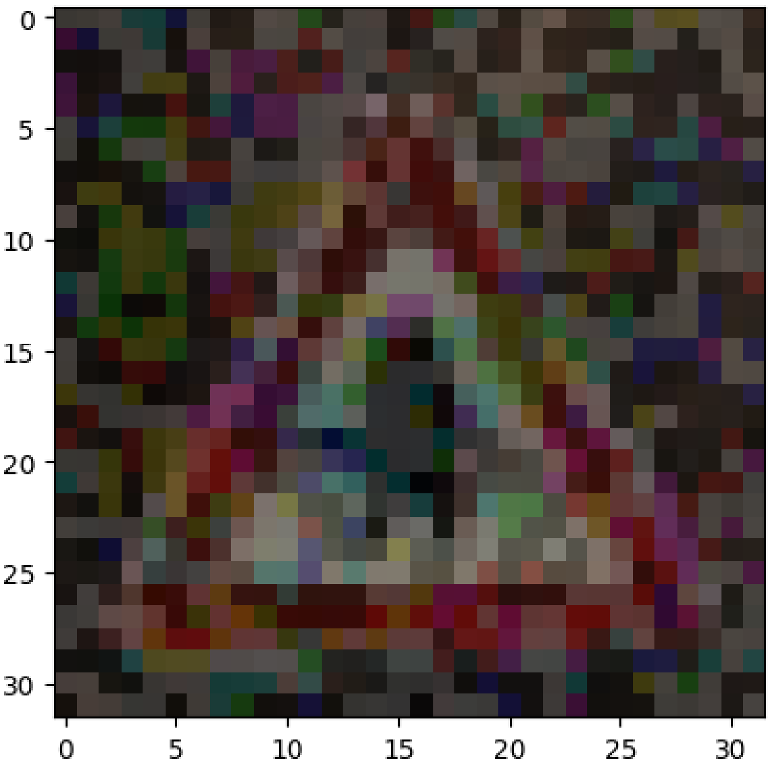
eps 0.03137254901960784: Pred class[25], Real class[11]



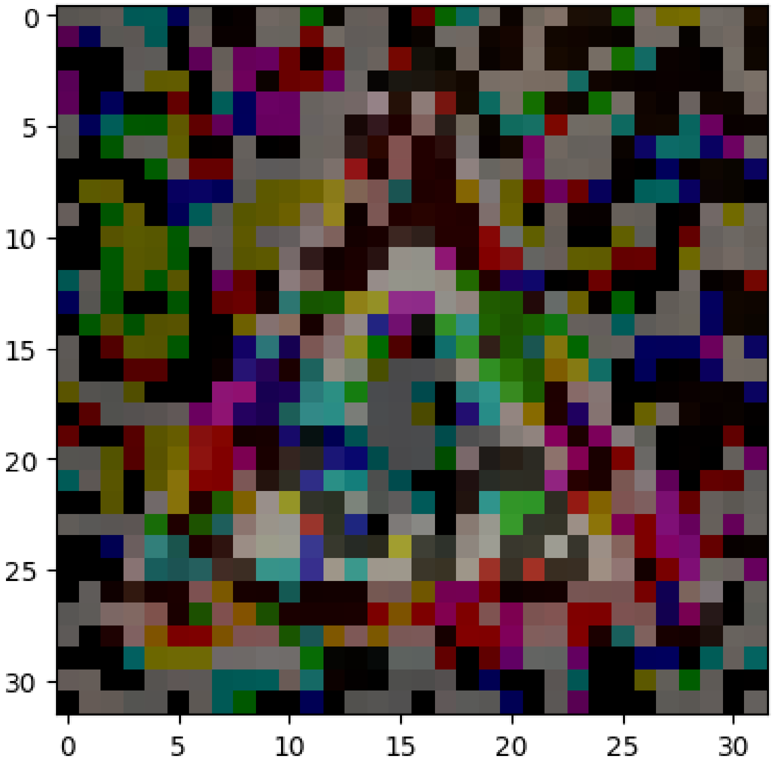
eps 0.0392156862745098: Pred class[25], Real class[11]

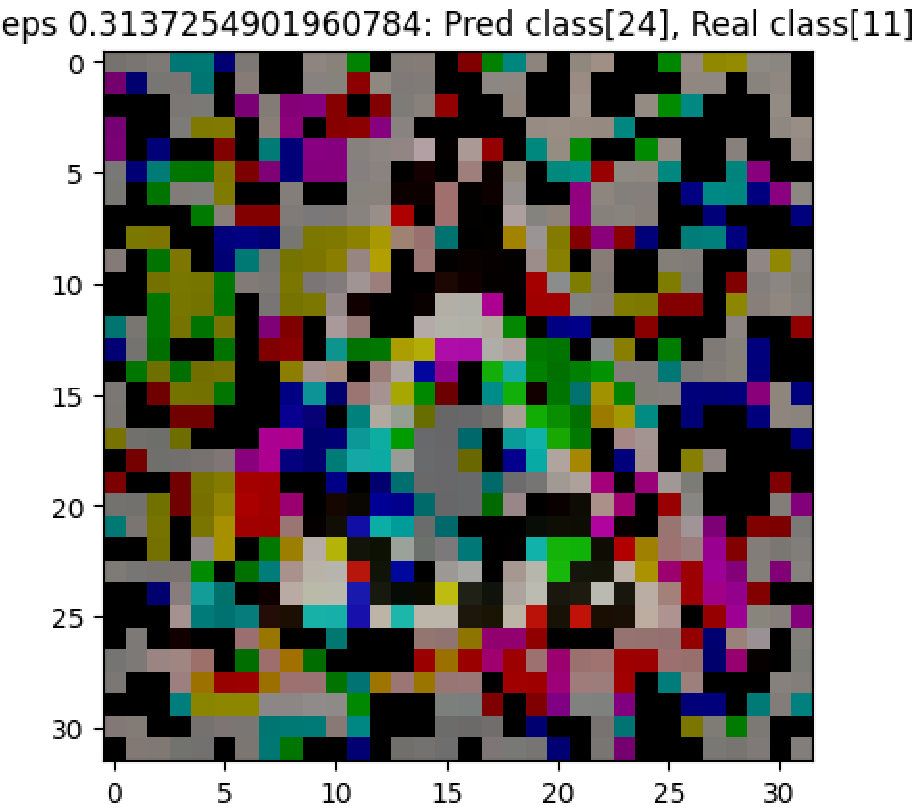


eps 0.0784313725490196: Pred class[24], Real class[11]

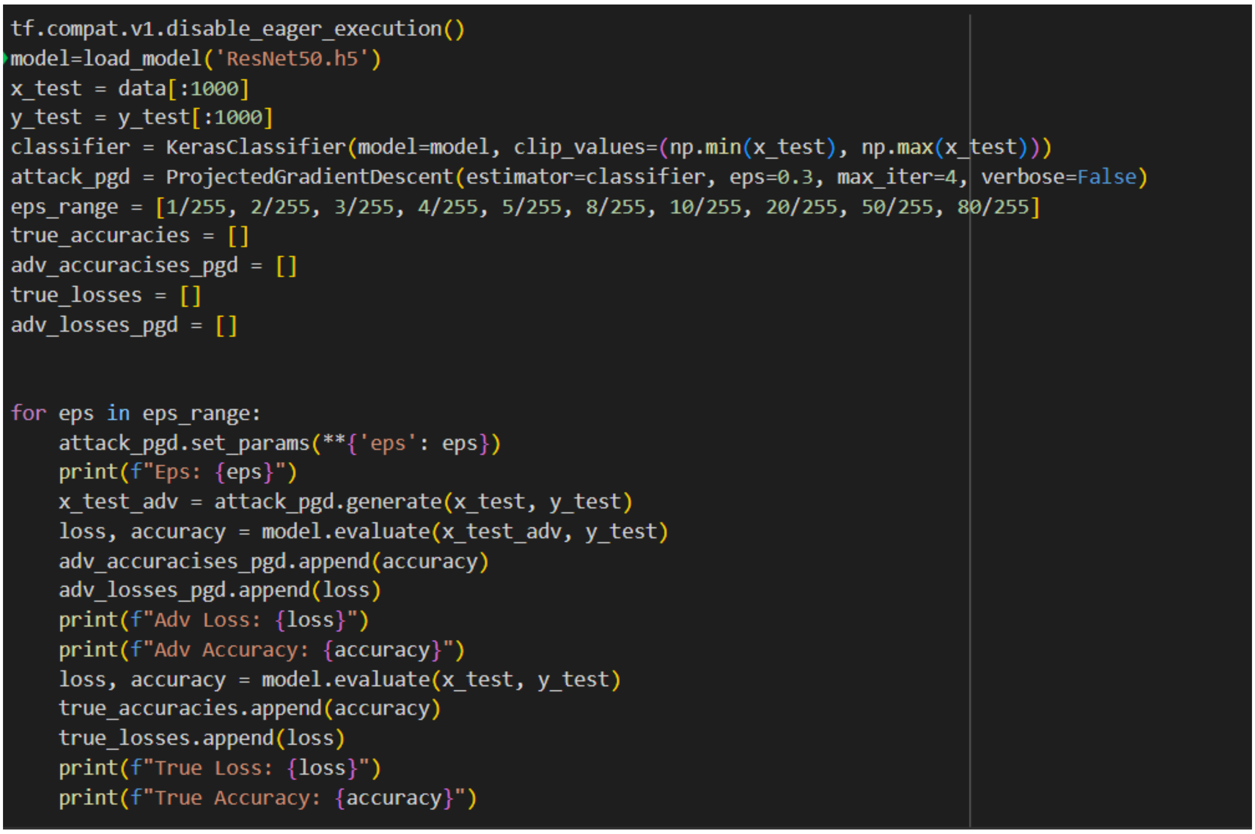


eps 0.19607843137254902: Pred class[24], Real class[11]




eps 0.3137254901960784: Pred class[24], Real class[11]

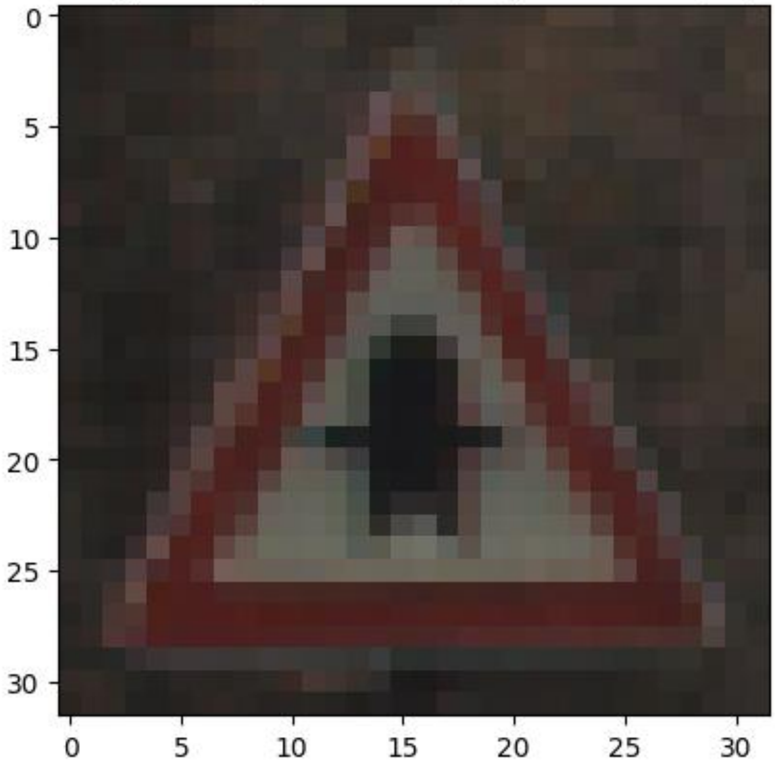
Создание атаки PGD



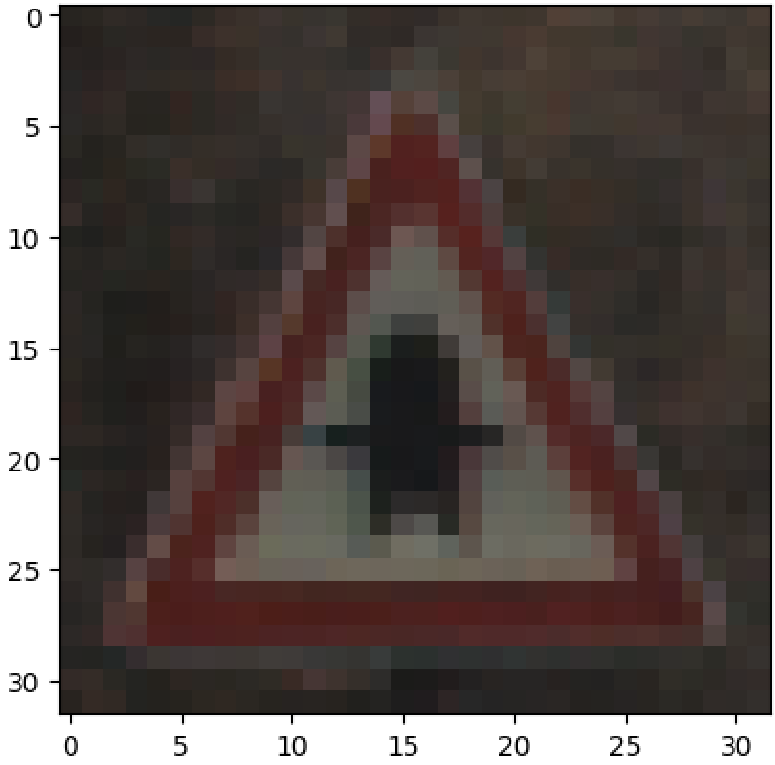
```
tf.compat.v1.disable_eager_execution()
model=load_model('ResNet50.h5')
x_test = data[:1000]
y_test = y_test[:1000]
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4, verbose=False)
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 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}")
```

Отображение исходных и адверсариальных изображений

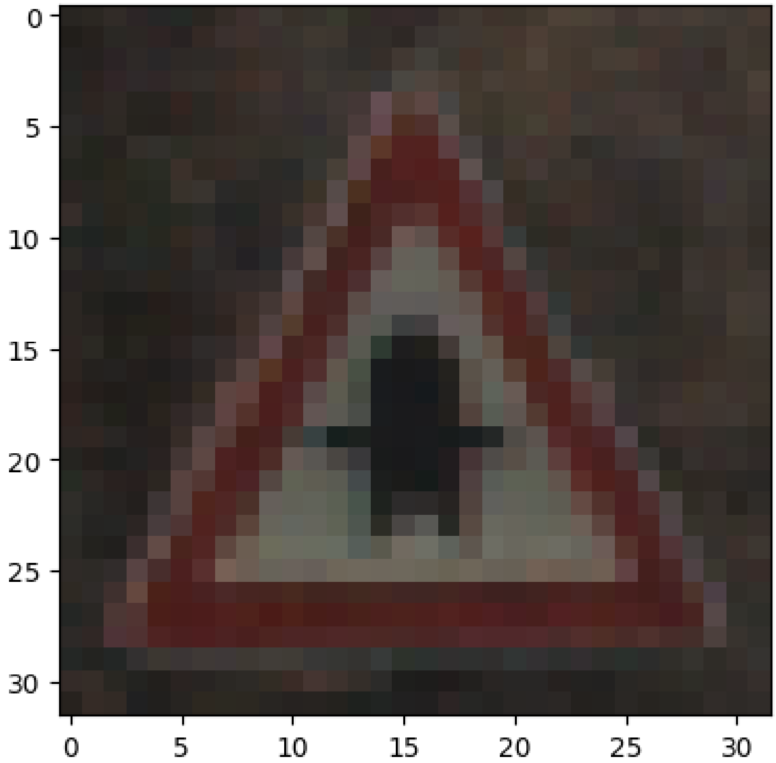
Original img: Pred class[11], Real calss[11]



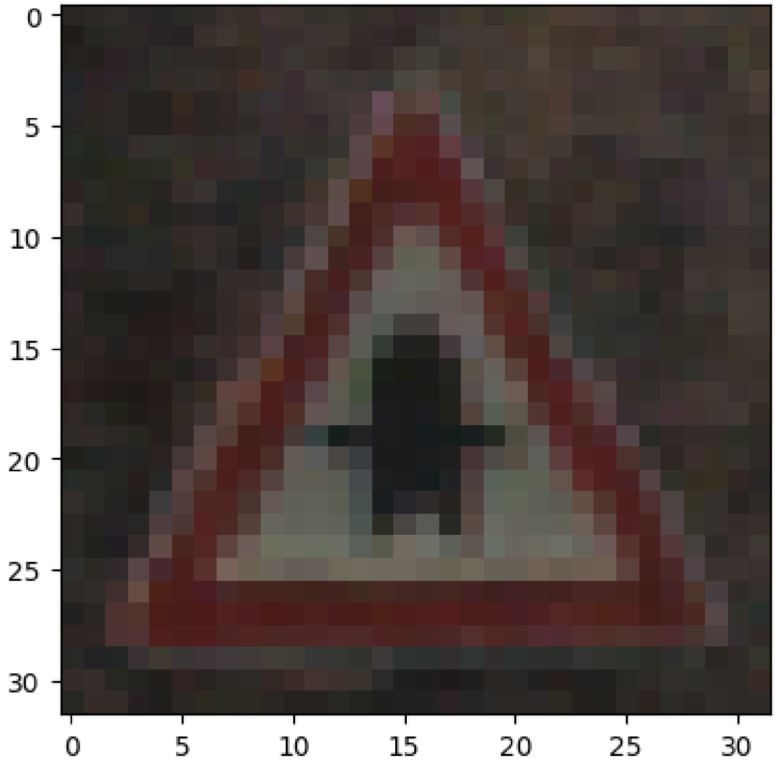
eps 0.00392156862745098: Pred class[11], Real class[11]

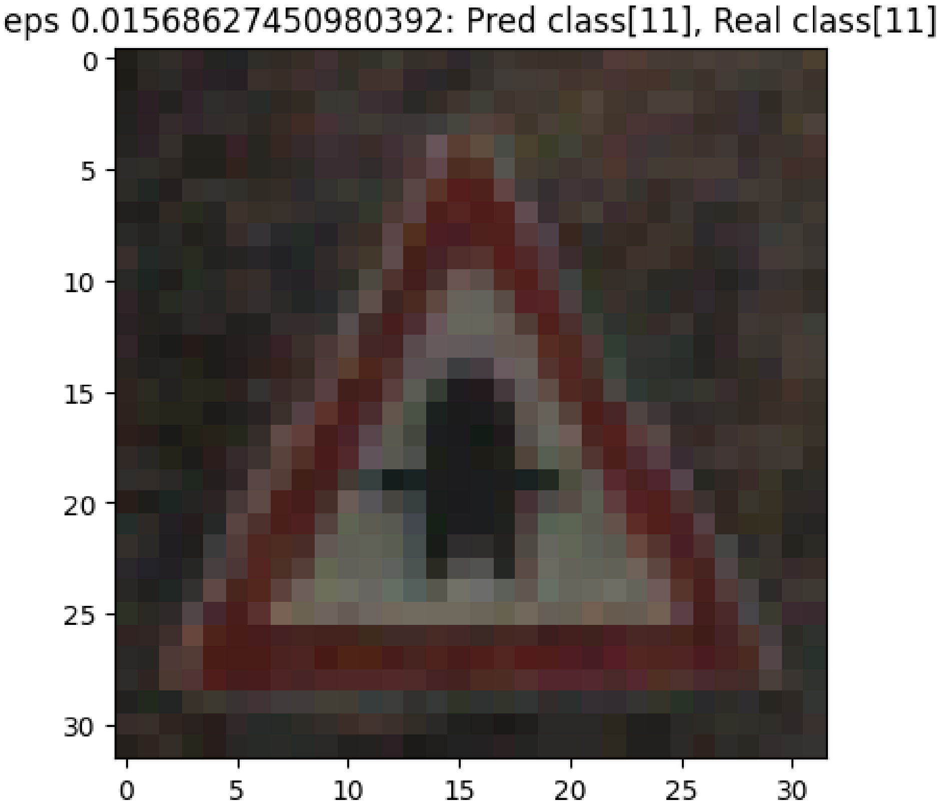


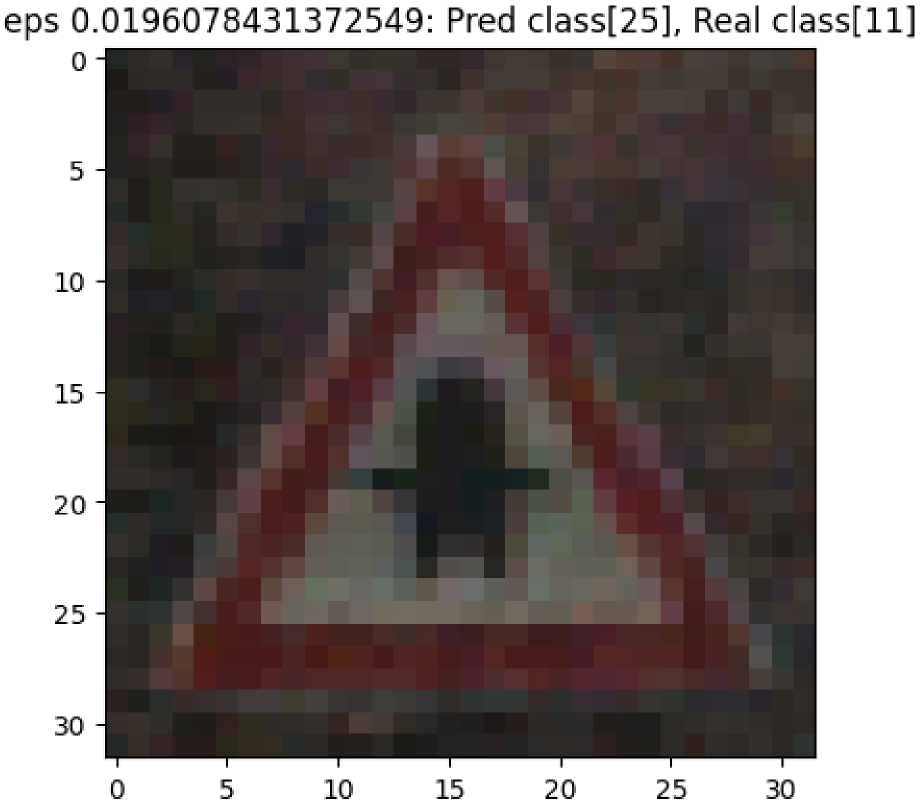
eps 0.00784313725490196: Pred class[11], Real class[11]

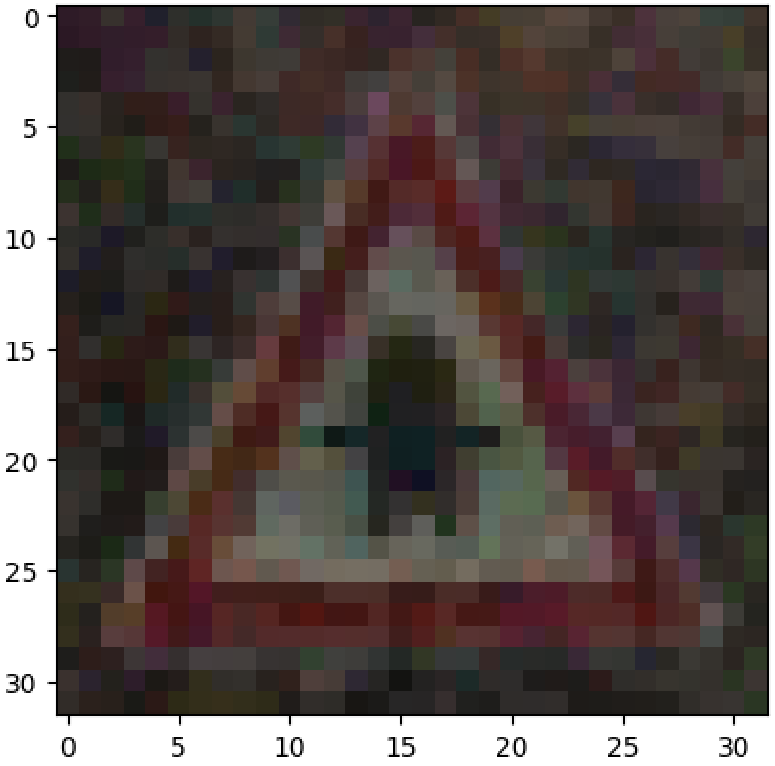


eps 0.011764705882352941: Pred class[11], Real class[11]

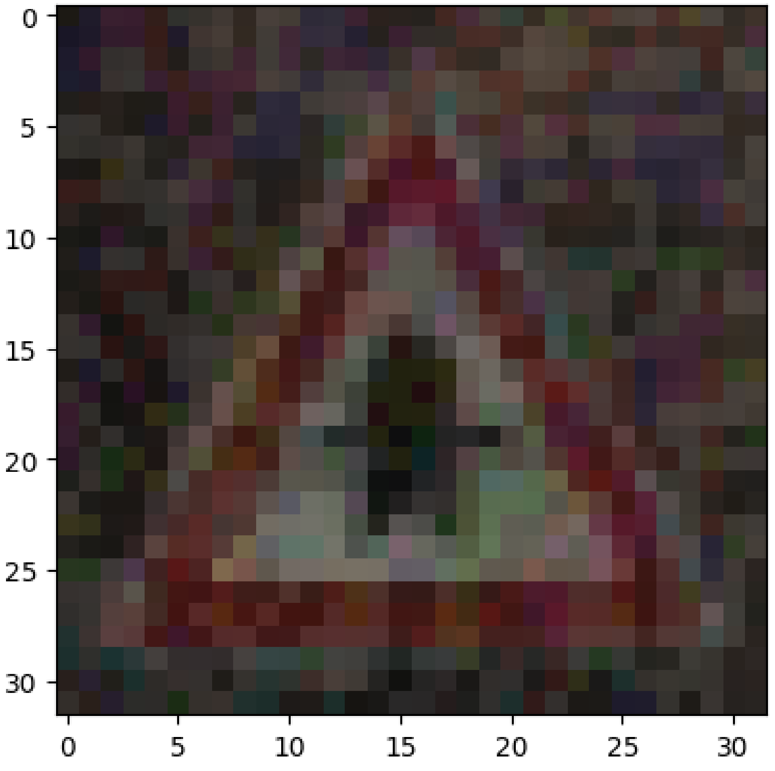



eps 0.01568627450980392: Pred class[11], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30


eps 0.0196078431372549: Pred class[25], Real class[11]
0
5
10
15
20
25
30
0
5
10
15
20
25
30

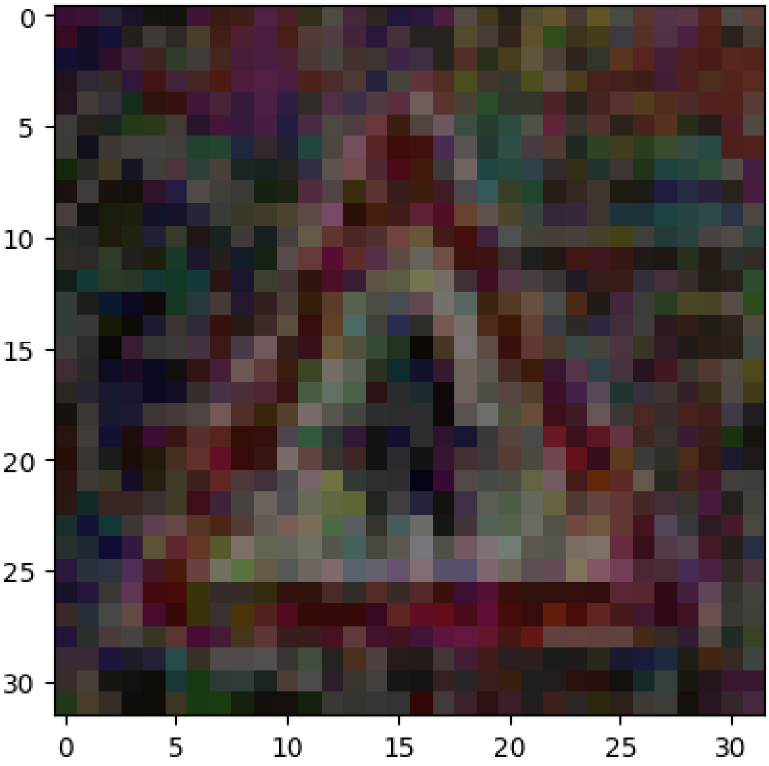


eps 0.03137254901960784: Pred class[25], Real class[11]

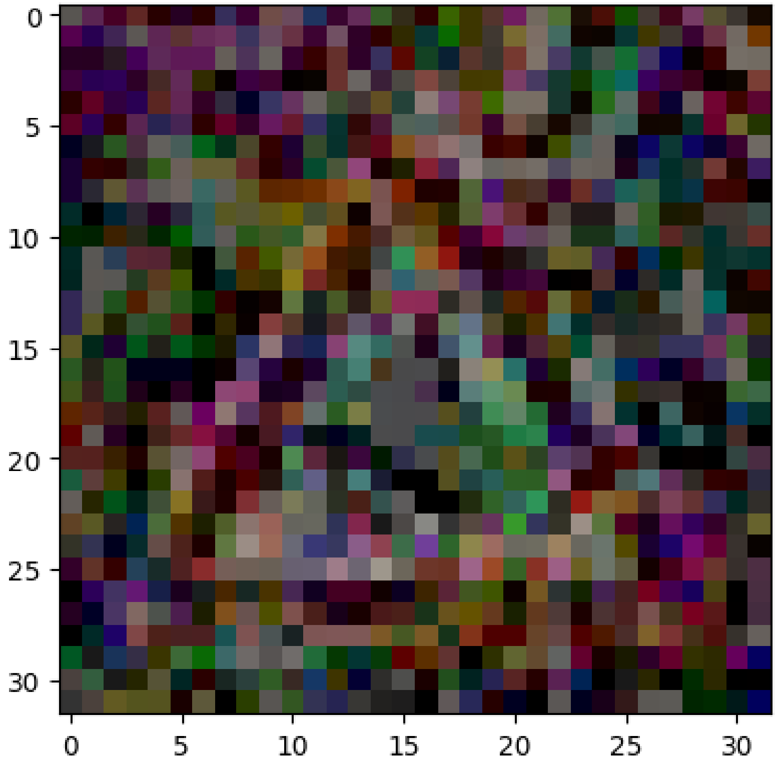


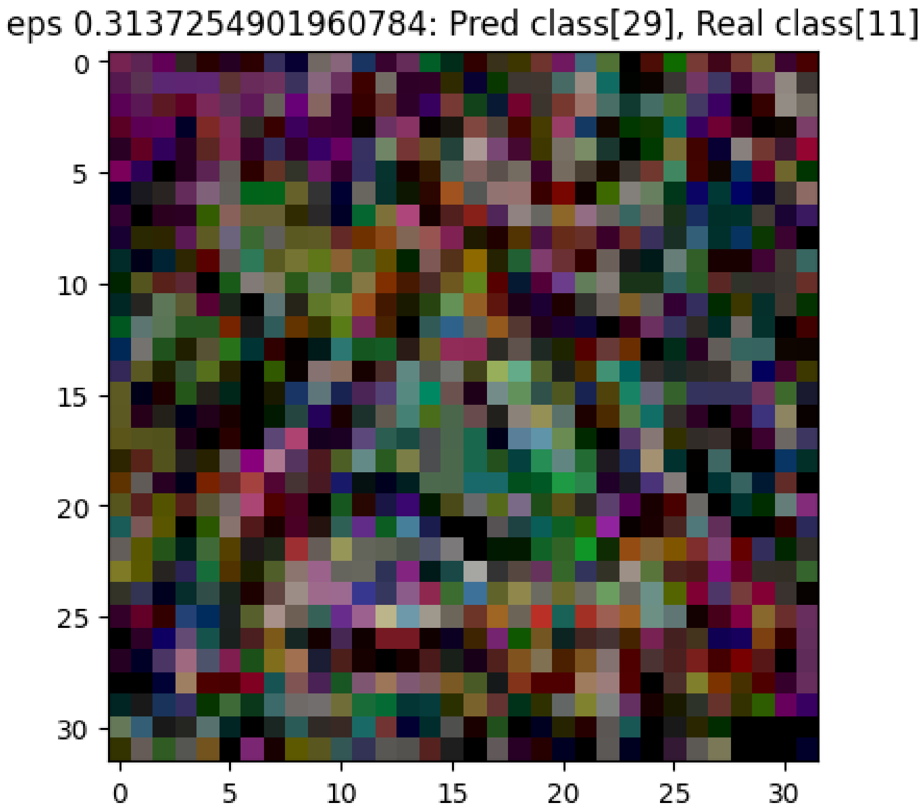
eps 0.0392156862745098: Pred class[11], Real class[11]

eps 0.0784313725490196: Pred class[24], Real class[11]

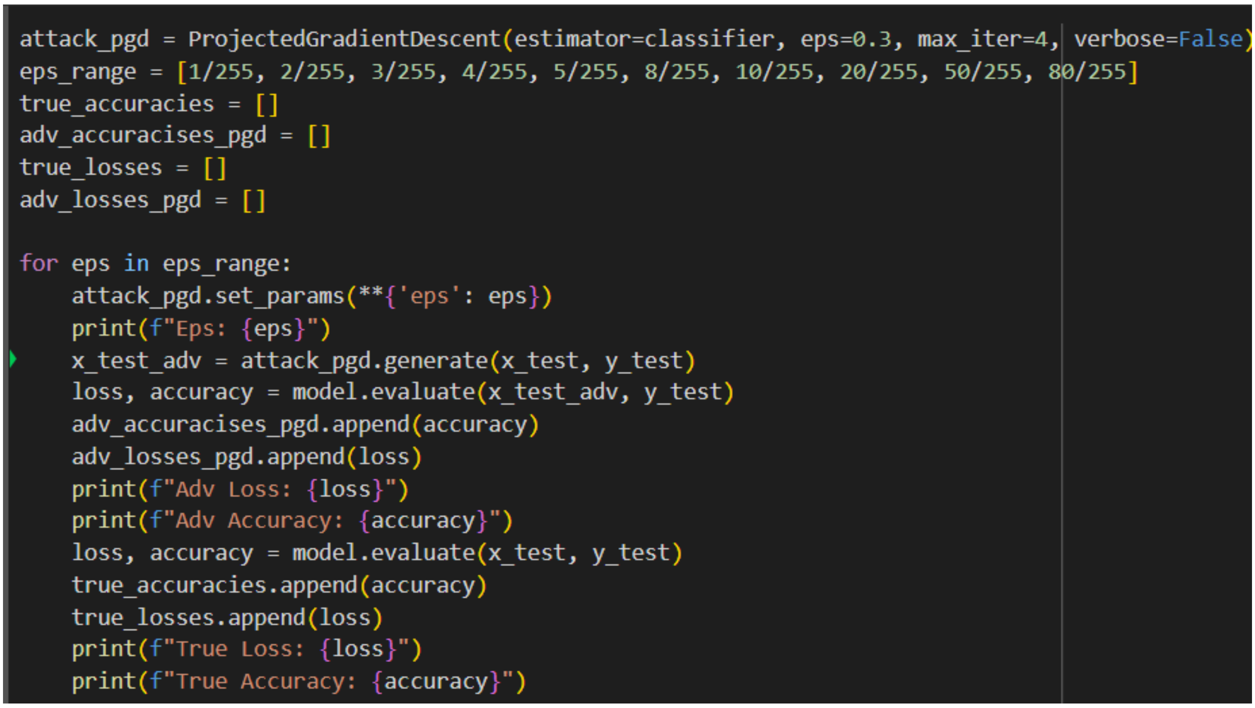


eps 0.19607843137254902: Pred class[29], Real class[11]



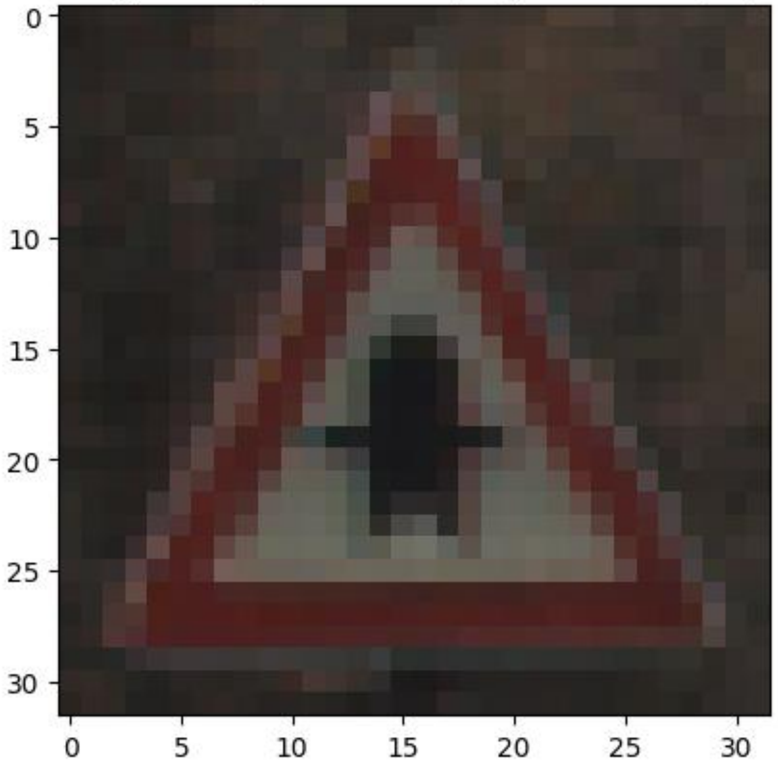


Создание атаки PGD

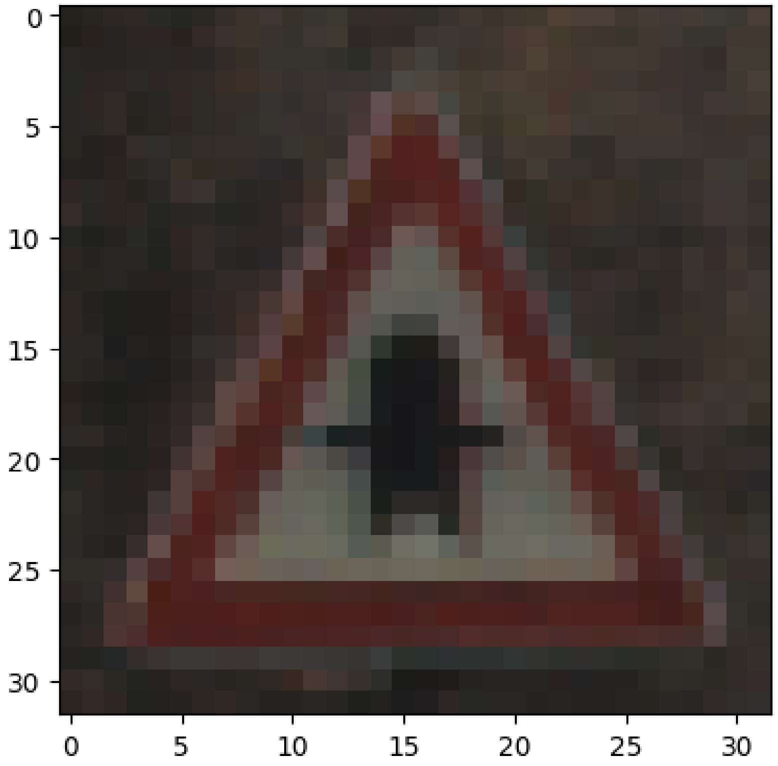


```
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4, verbose=False)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]
true_accuracies = []
adv_accuracises_pgd = []
true_losses = []
adv_losses_pgd = []

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 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}")
```

Отображение исходных и адверсариальных изображений

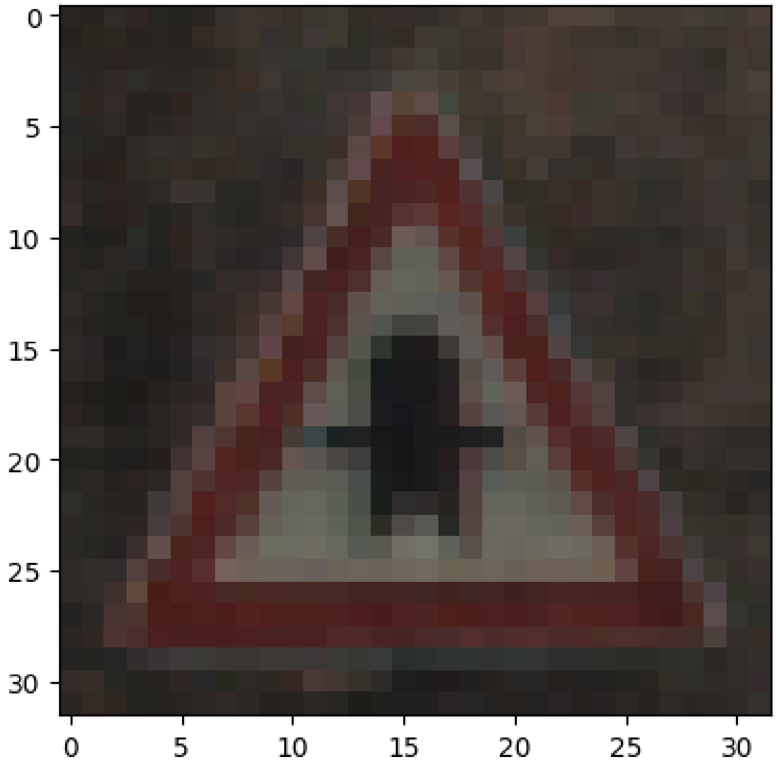


Original img: Pred class[11], Real class[11]

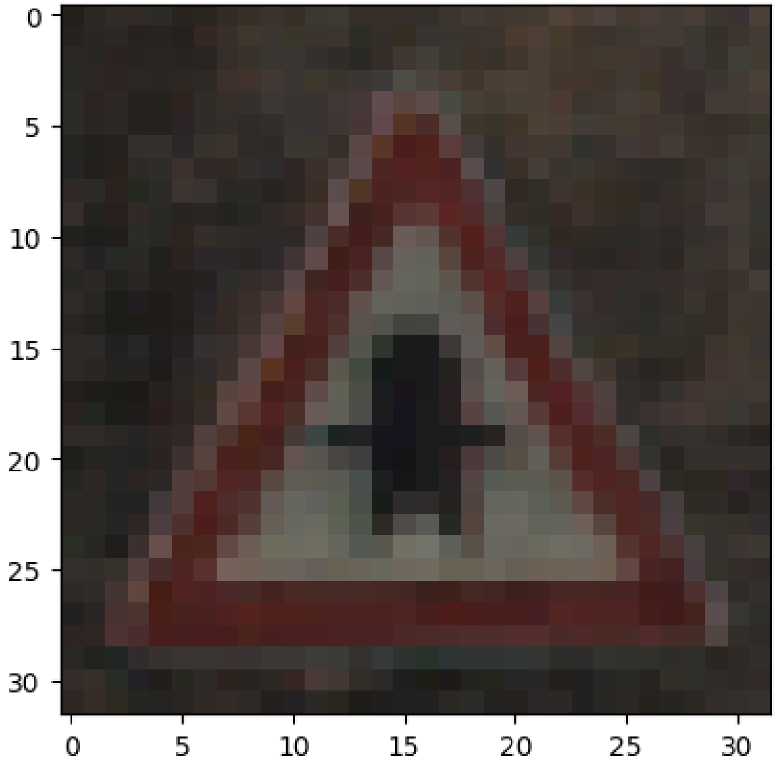


eps 0.00392156862745098: Pred class[11], Real class[11]

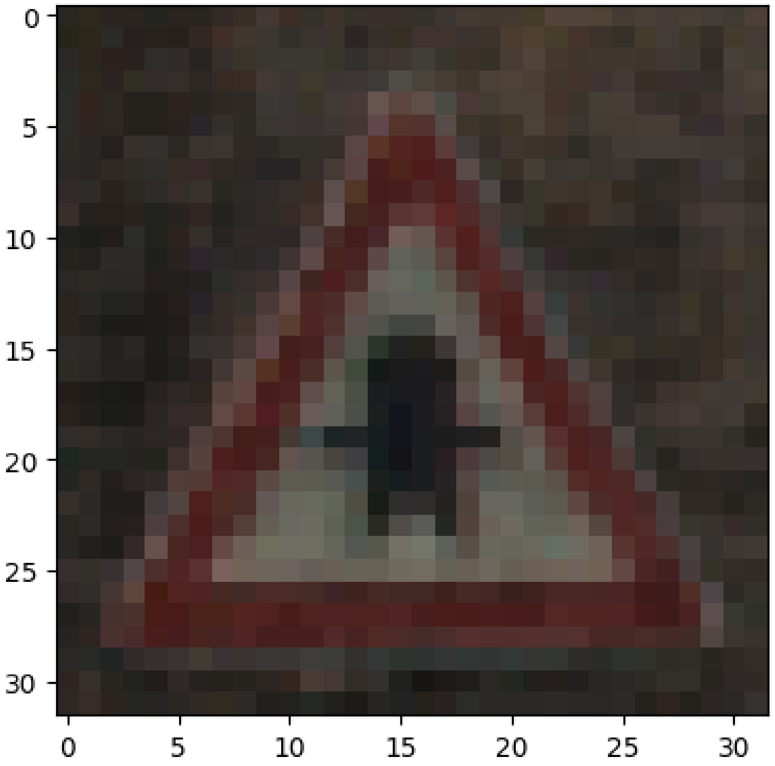
eps 0.00784313725490196: Pred class[11], Real class[11]



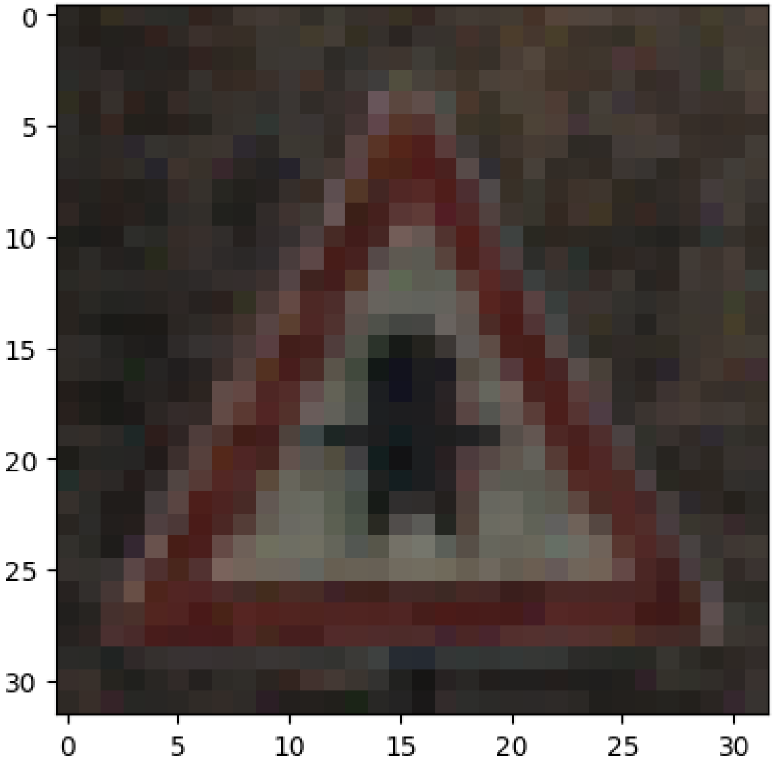
eps 0.011764705882352941: Pred class[11], Real class[11]



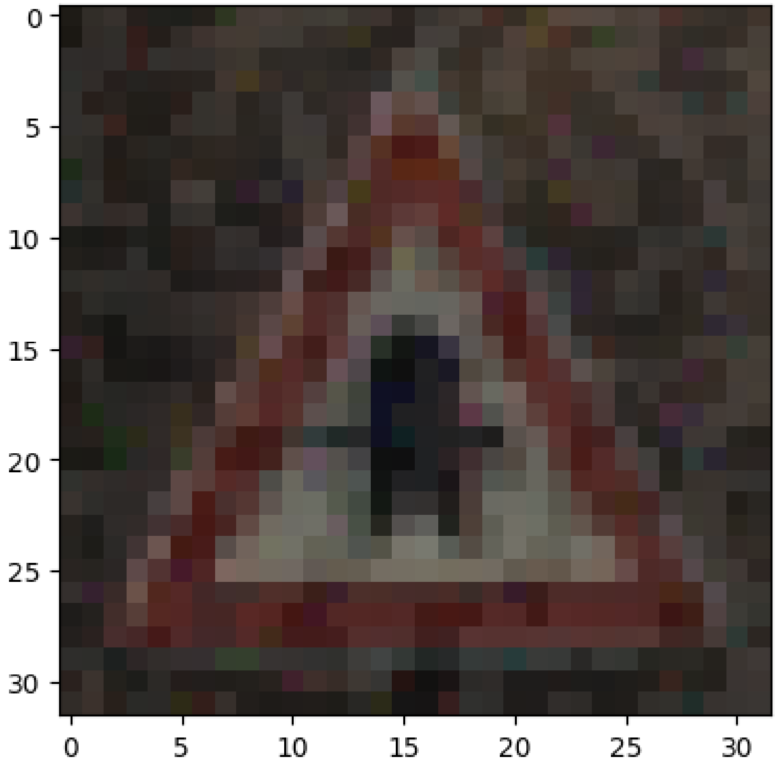
eps 0.01568627450980392: Pred class[11], Real class[11]



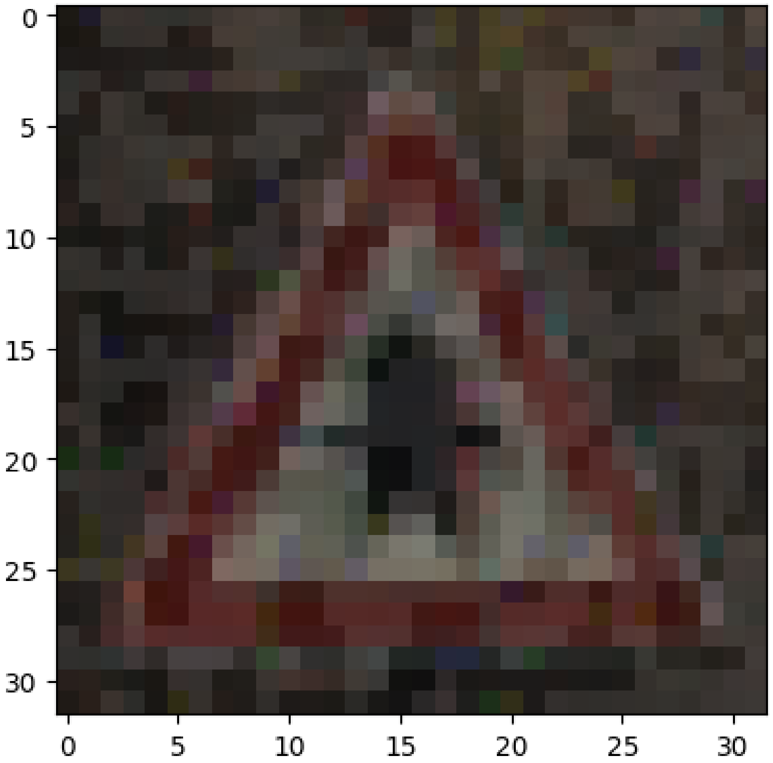
eps 0.0196078431372549: Pred class[11], Real class[11]

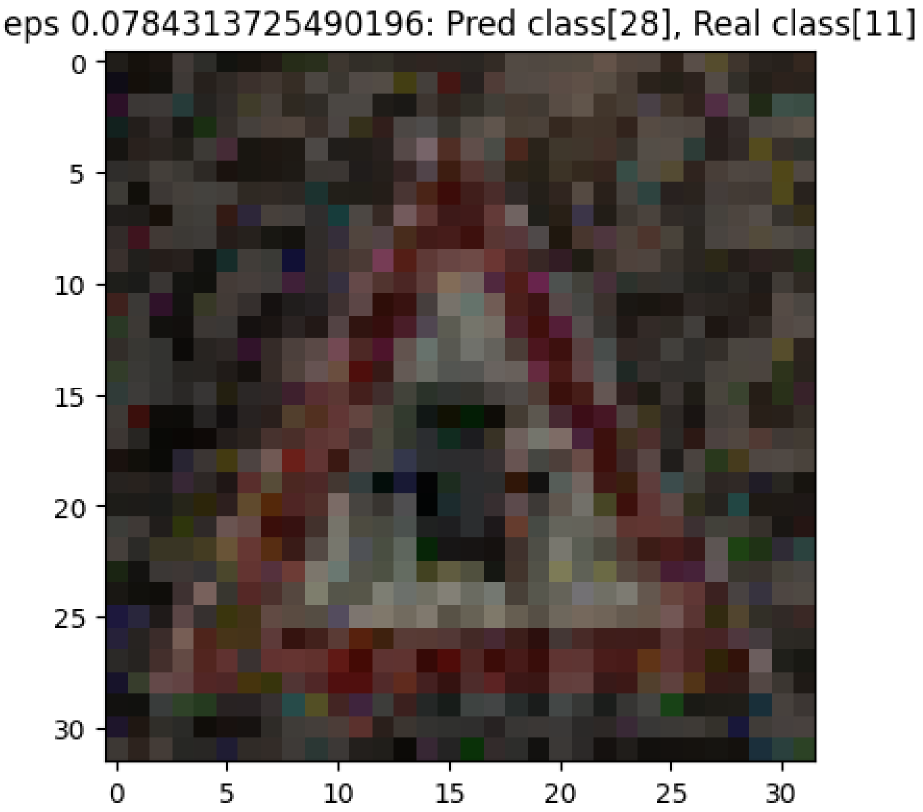


eps 0.03137254901960784: Pred class[27], Real class[11]

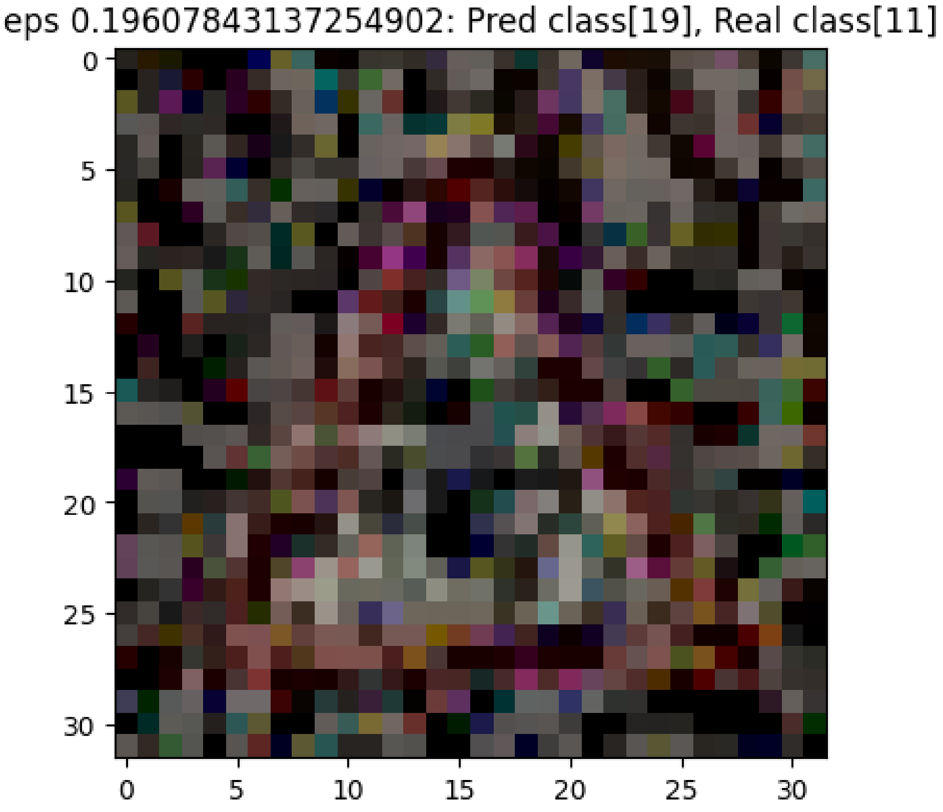


eps 0.0392156862745098: Pred class[27], Real class[11]

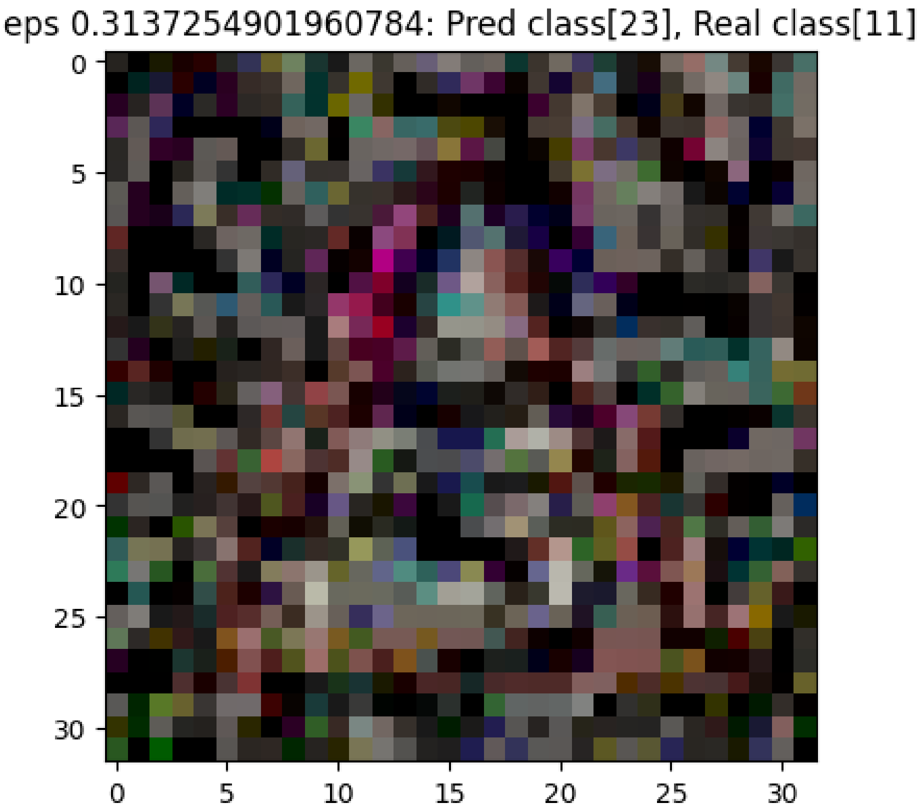




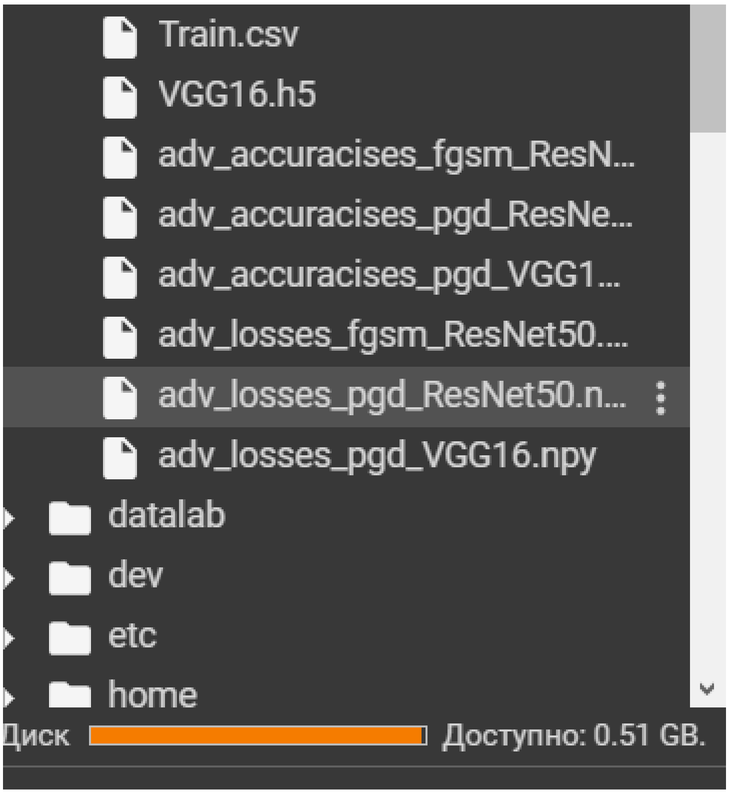
eps 0.0784313725490196: Pred class[28], Real class[11]



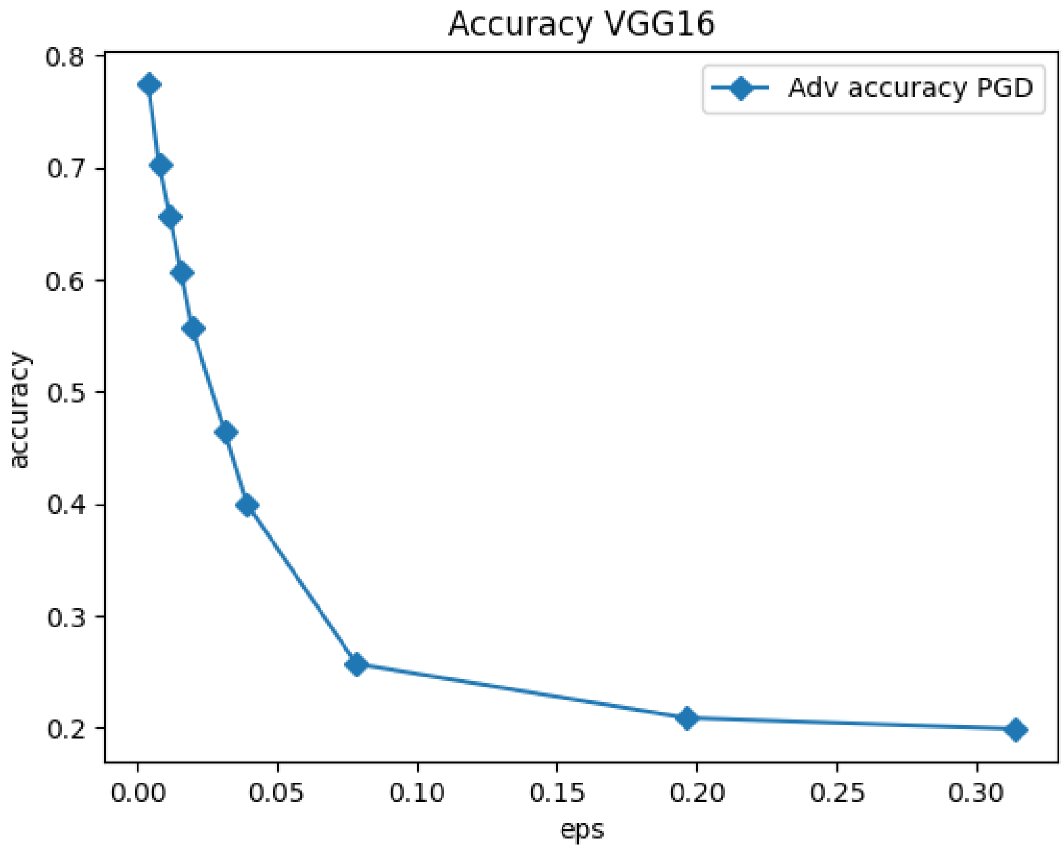
eps 0.19607843137254902: Pred class[19], Real class[11]

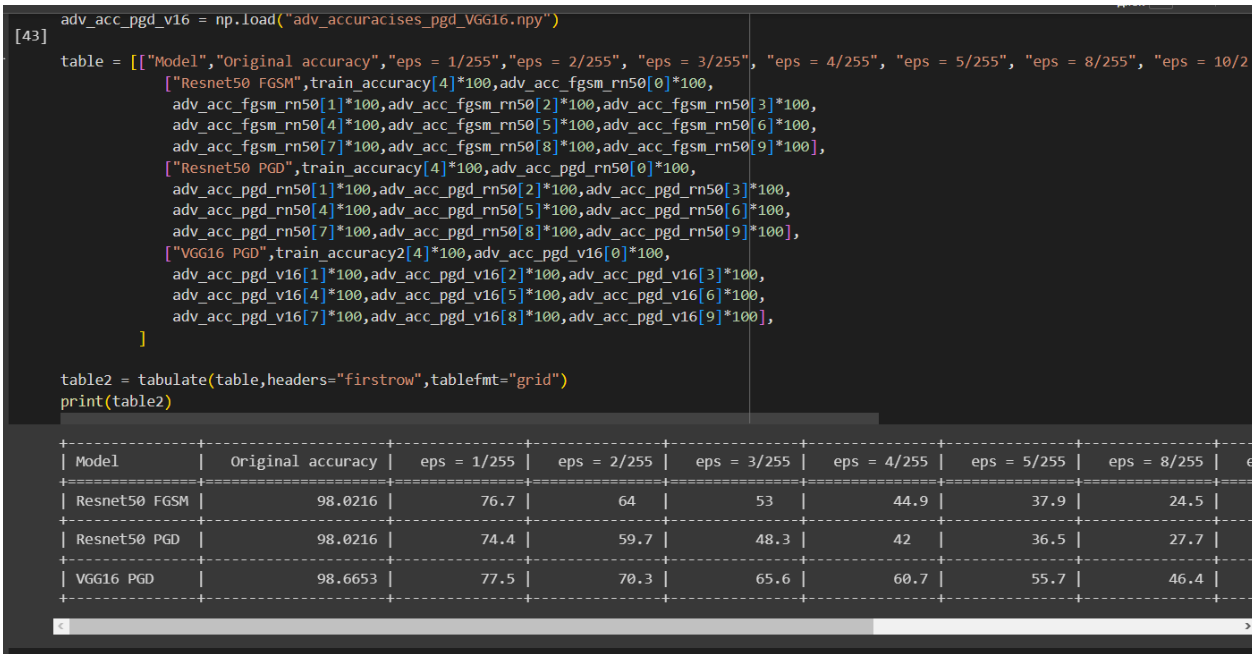


Когда пришло время строить графики и смотреть зависимости, обратил внимание на то, что из-за нехватки места в гугл колабе, нужные файлы, видимо, стерлись. Поэтому продемонстрирую то, что есть.



Train.csv
VGG16.h5
adv_accuracises_fgsm_ResN...
adv_accuracises_pgd_ResNe...
adv_accuracises_pgd_VGG1...
adv_losses_fgsm_ResNet50....
adv_losses_pgd_ResNet50.n...
adv_losses_pgd_VGG16.npy
datalab
dev
etc
home
Диск Доступно: 0.51 GB.



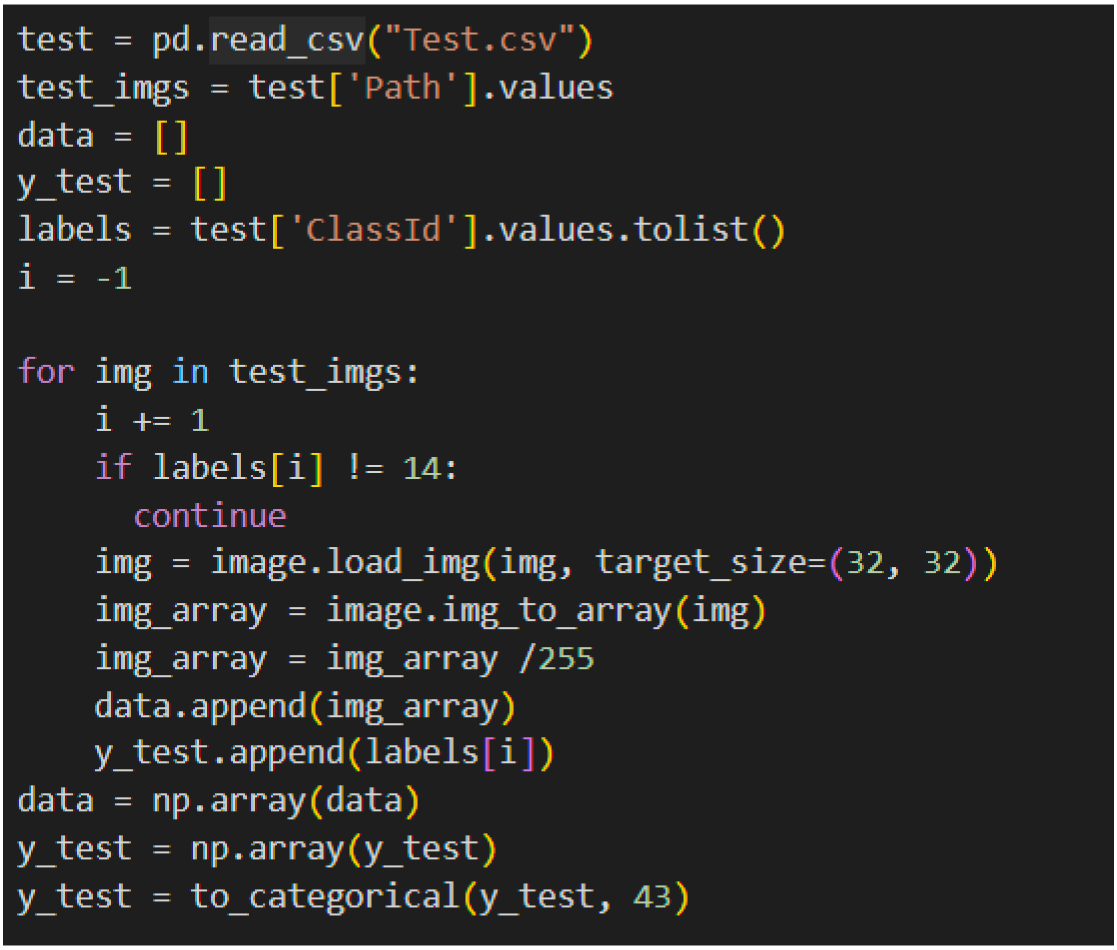


```
[43]
adv_acc_pgd_v16 = np.load("adv_accuracises_pgd_VGG16.npy")

table = [["Model","Original accuracy","eps = 1/255","eps = 2/255", "eps = 3/255", "eps = 4/255", "eps = 5/255", "eps = 8/255", "eps = 10/2
             ["Resnet50 FGSM",train_accuracy[4]*100,adv_acc_fgsm_rn50[0]*100,
              adv_acc_fgsm_rn50[1]*100,adv_acc_fgsm_rn50[2]*100,adv_acc_fgsm_rn50[3]*100,
              adv_acc_fgsm_rn50[4]*100,adv_acc_fgsm_rn50[5]*100,adv_acc_fgsm_rn50[6]*100,
              adv_acc_fgsm_rn50[7]*100,adv_acc_fgsm_rn50[8]*100,adv_acc_fgsm_rn50[9]*100],
             ["Resnet50 PGD",train_accuracy[4]*100,adv_acc_pgd_rn50[0]*100,
              adv_acc_pgd_rn50[1]*100,adv_acc_pgd_rn50[2]*100,adv_acc_pgd_rn50[3]*100,
              adv_acc_pgd_rn50[4]*100,adv_acc_pgd_rn50[5]*100,adv_acc_pgd_rn50[6]*100,
              adv_acc_pgd_rn50[7]*100,adv_acc_pgd_rn50[8]*100,adv_acc_pgd_rn50[9]*100],
             ["VGG16 PGD",train_accuracy2[4]*100,adv_acc_pgd_v16[0]*100,
              adv_acc_pgd_v16[1]*100,adv_acc_pgd_v16[2]*100,adv_acc_pgd_v16[3]*100,
              adv_acc_pgd_v16[4]*100,adv_acc_pgd_v16[5]*100,adv_acc_pgd_v16[6]*100,
              adv_acc_pgd_v16[7]*100,adv_acc_pgd_v16[8]*100,adv_acc_pgd_v16[9]*100],
         ]

table2 = tabulate(table,headers="firstrow",tablefmt="grid")
print(table2)
```

```
+---------------+---------------------+---------------+---------------+---------------+---------------+---------------+---------------+---
| Model         |   Original accuracy |   eps = 1/255 |   eps = 2/255 |   eps = 3/255 |   eps = 4/255 |   eps = 5/255 |   eps = 8/255 |   e
+===============+=====================+===============+===============+===============+===============+===============+===============+===
| Resnet50 FGSM |             98.0216 |          76.7 |          64   |          53   |          44.9 |          37.9 |          24.5 |
+---------------+---------------------+---------------+---------------+---------------+---------------+---------------+---------------+---
| Resnet50 PGD  |             98.0216 |          74.4 |          59.7 |          48.3 |          42   |          36.5 |          27.7 |
+---------------+---------------------+---------------+---------------+---------------+---------------+---------------+---------------+---
| VGG16 PGD     |             98.6653 |          77.5 |          70.3 |          65.6 |          60.7 |          55.7 |          46.4 |
+---------------+---------------------+---------------+---------------+---------------+---------------+---------------+---------------+---
```

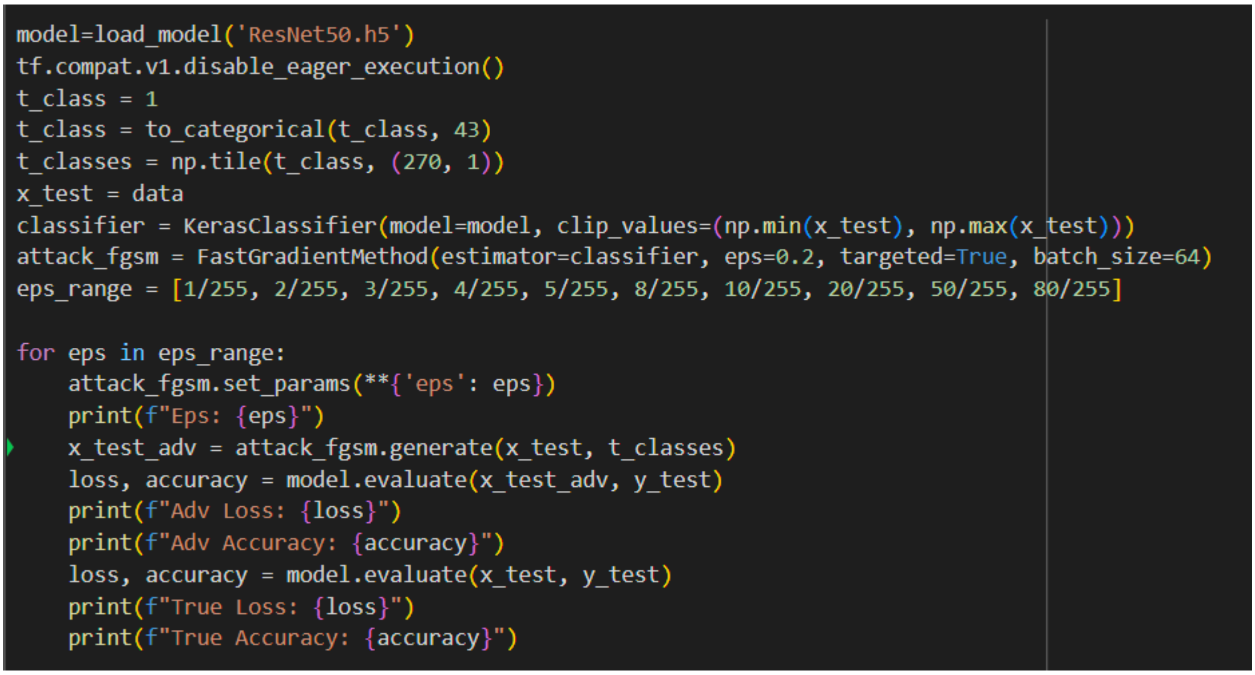
Задание 3

Загрузка тестового набора данных и извлечения изображения с меткой 14



```python
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)
```

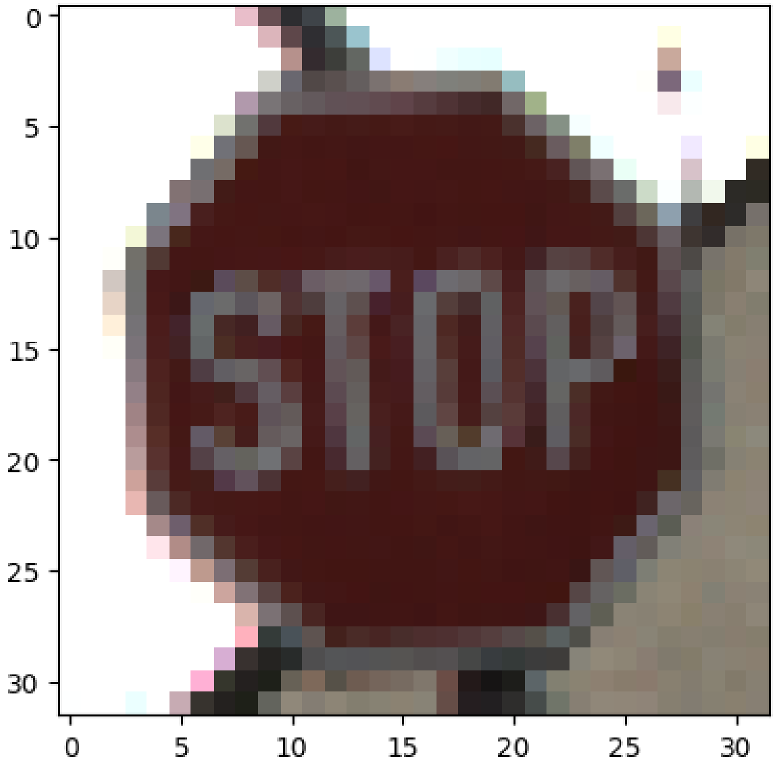
Реализация атаки FGSM



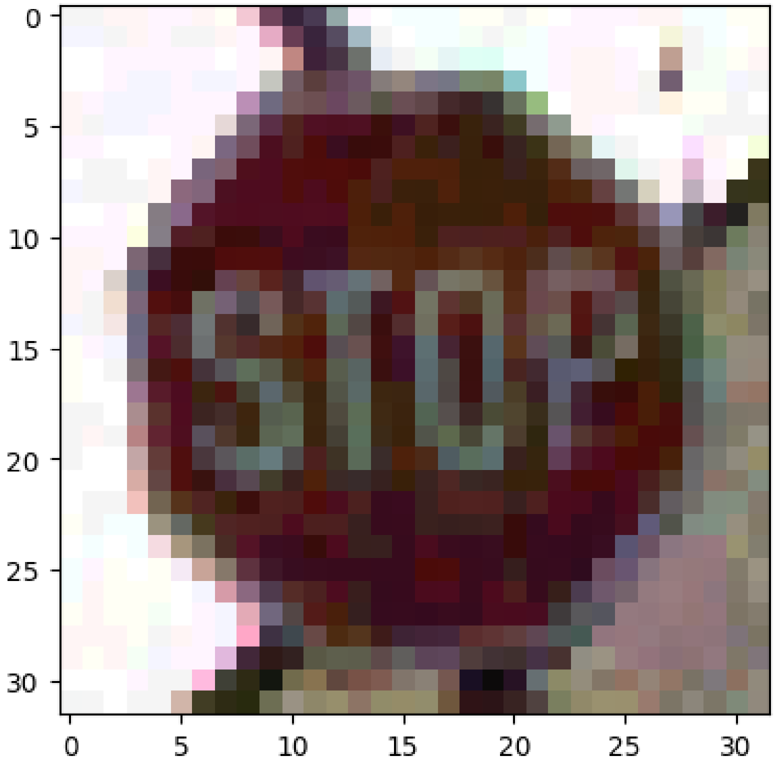
```python
model=load_model('ResNet50.h5')
tf.compat.v1.disable_eager_execution()
t_class = 1
t_class = to_categorical(t_class, 43)
t_classes = np.tile(t_class, (270, 1))
x_test = data
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_fgsm = FastGradientMethod(estimator=classifier, 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}")
```

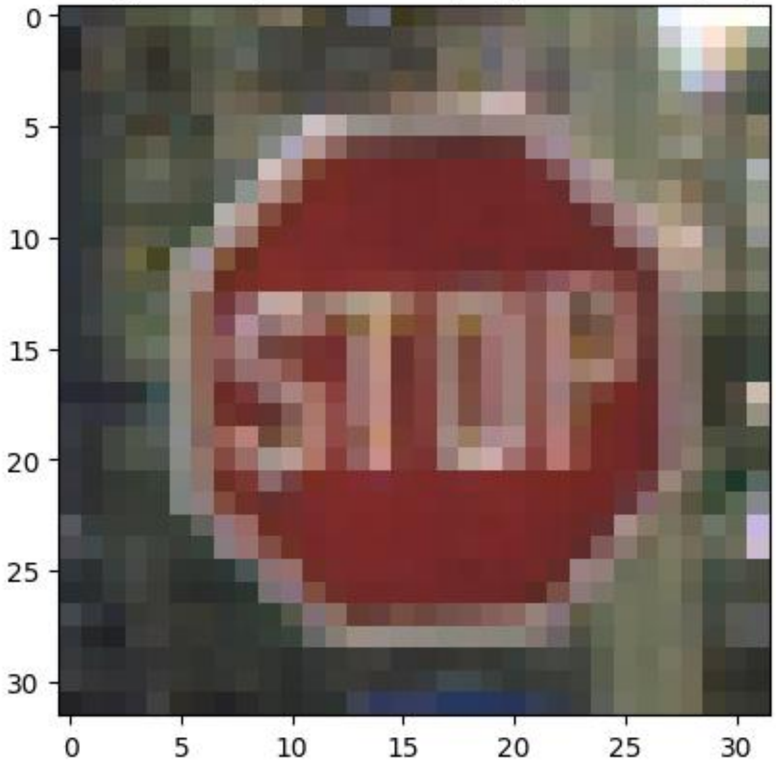
Визуализация последствий атаки

Original img: Pred class[14], Real class[14]

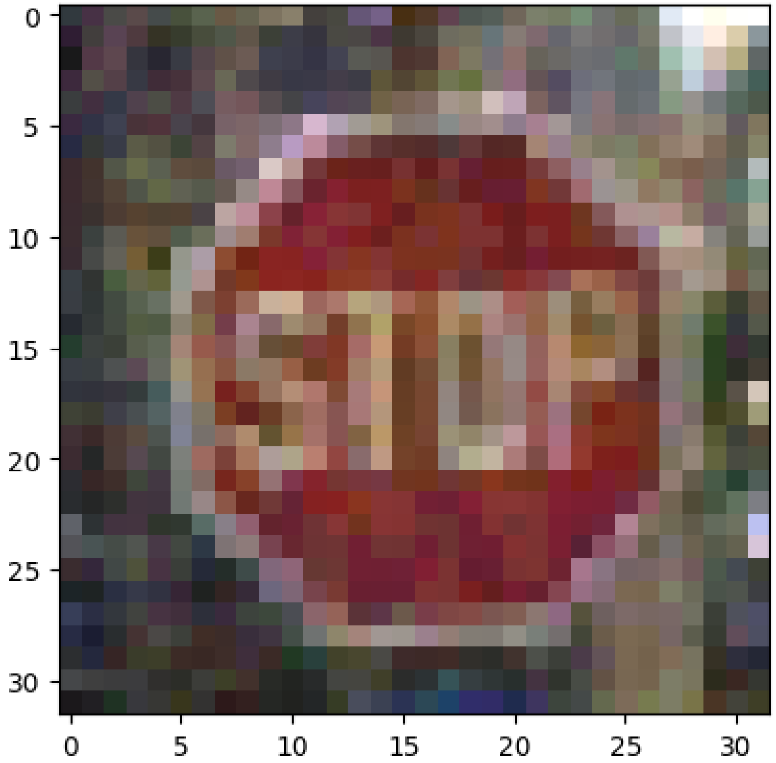


eps 0.0392156862745098: Pred class[1], Real class[14]

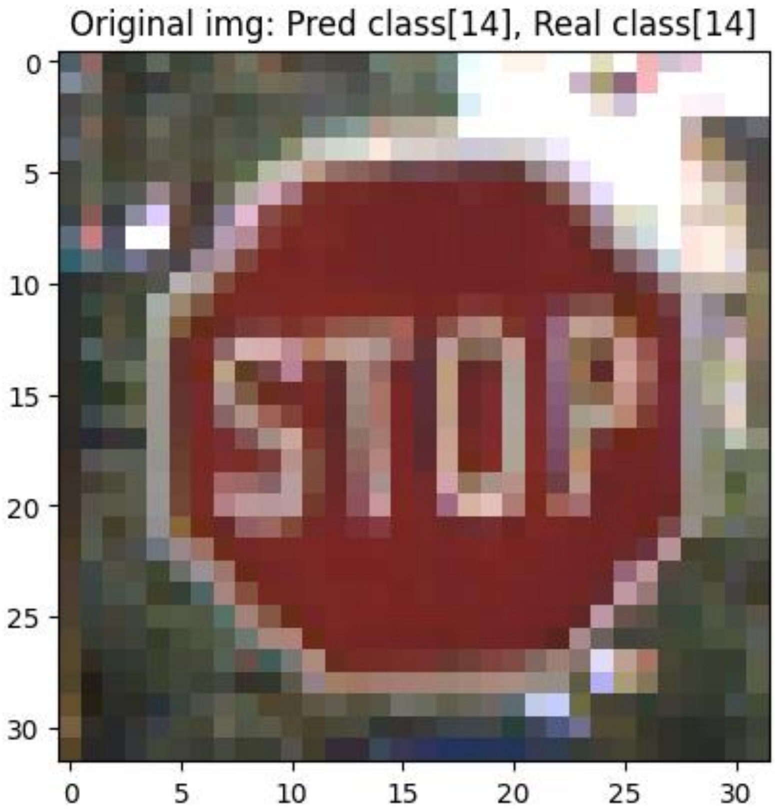


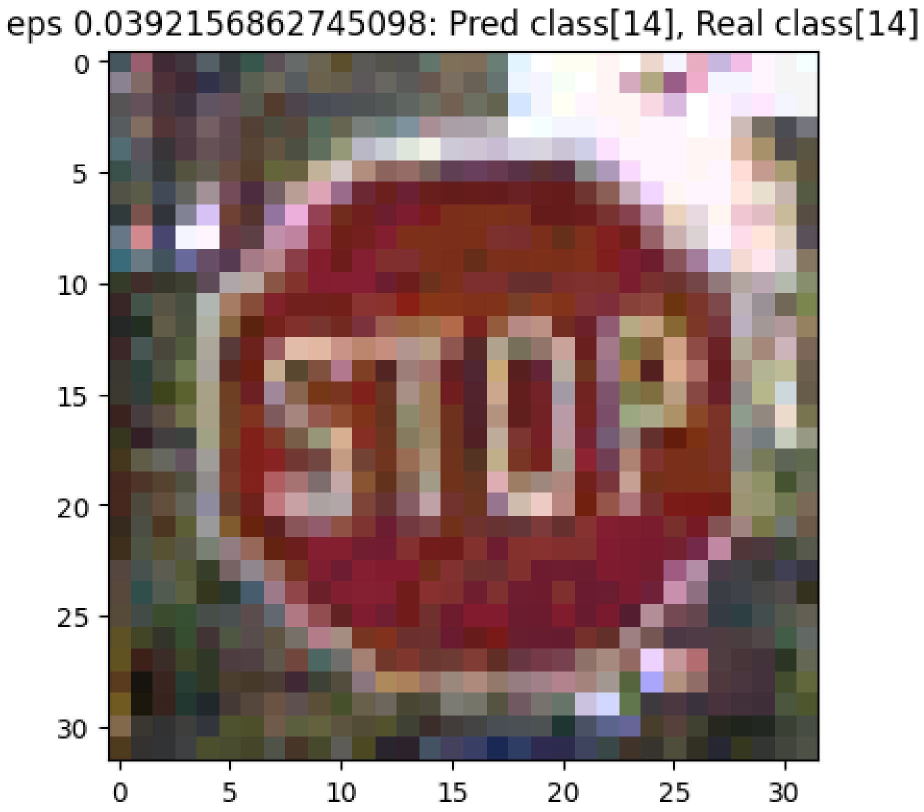


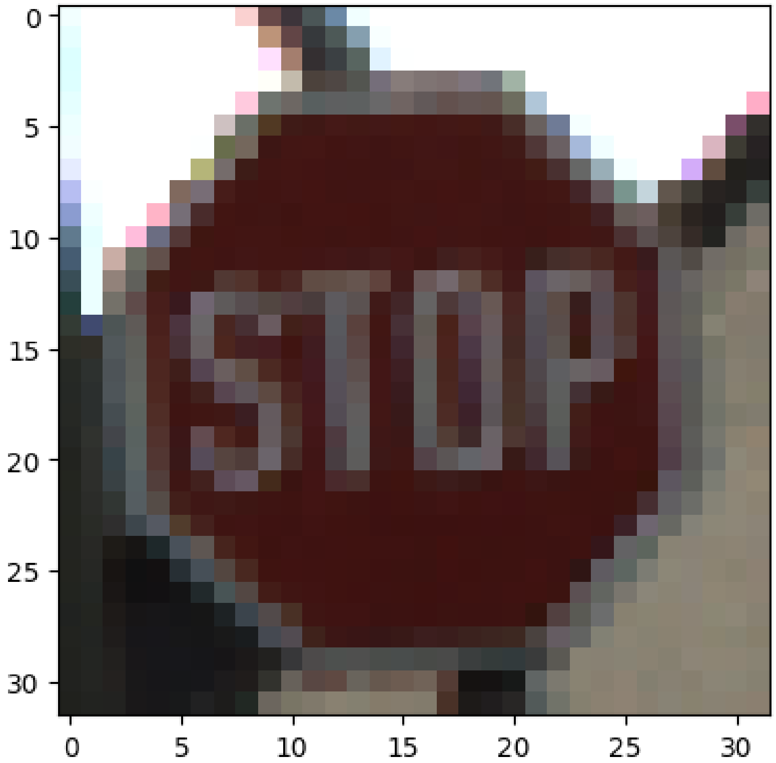
Original img: Pred class[14], Real class[14]



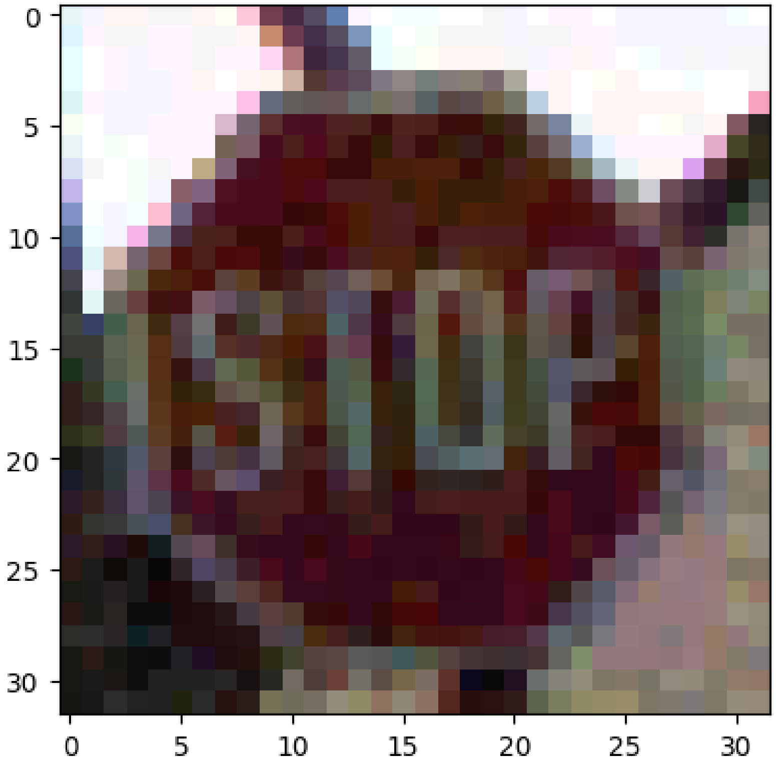
eps 0.0392156862745098: Pred class[14], Real class[14]


Original img: Pred class[14], Real class[14]
0
5
10
15
20
25
30
STOP

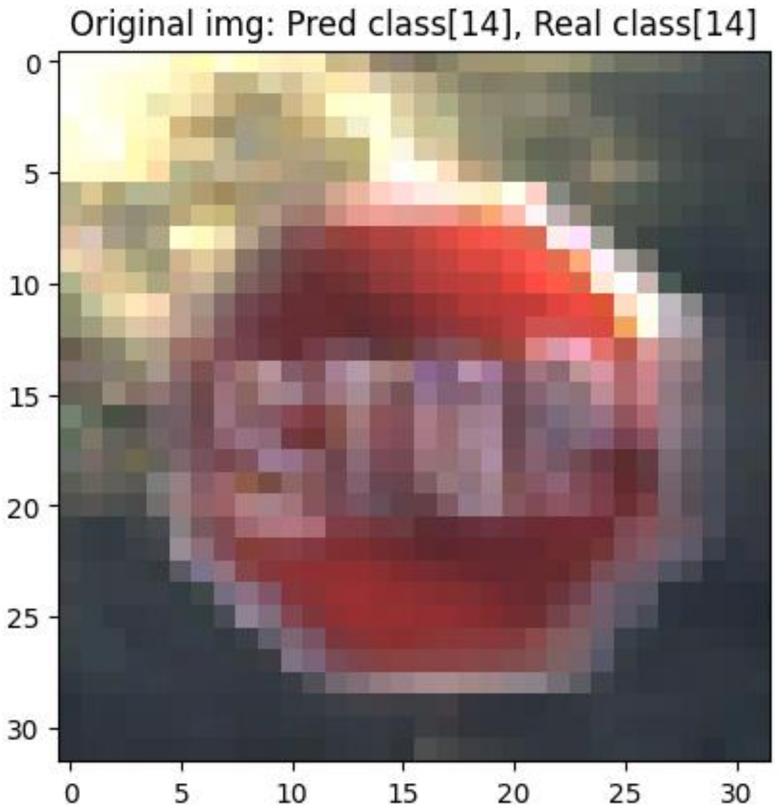

eps 0.0392156862745098: Pred class[14], Real class[14]
0
5
10
15
20
25
30
STOP

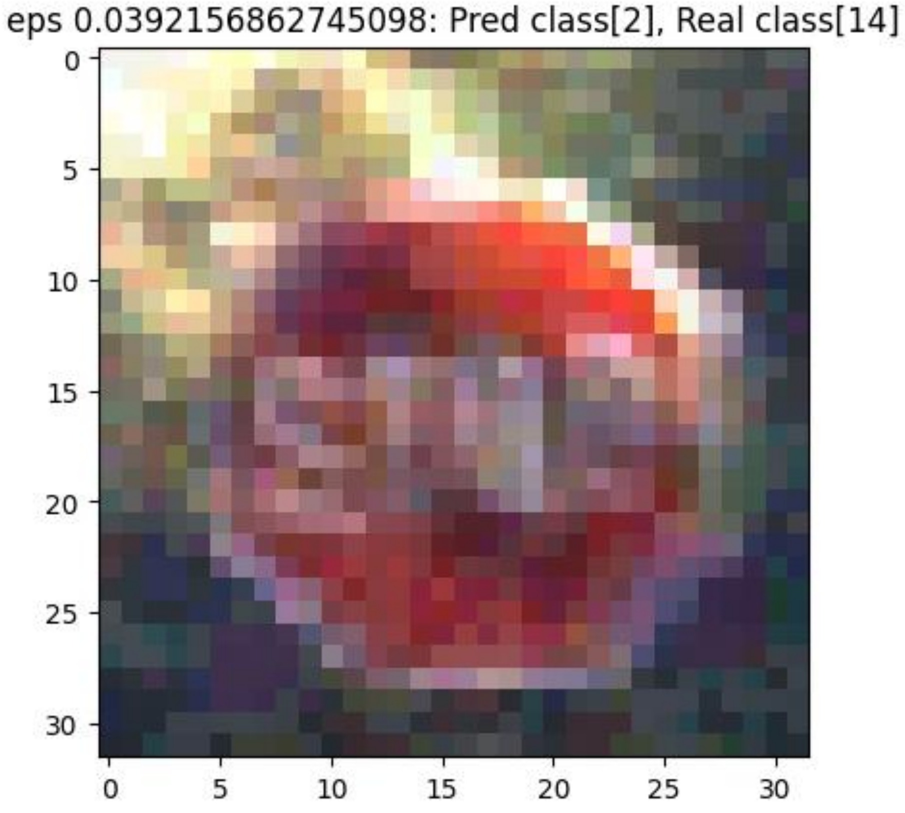


Original img: Pred class[14], Real class[14]

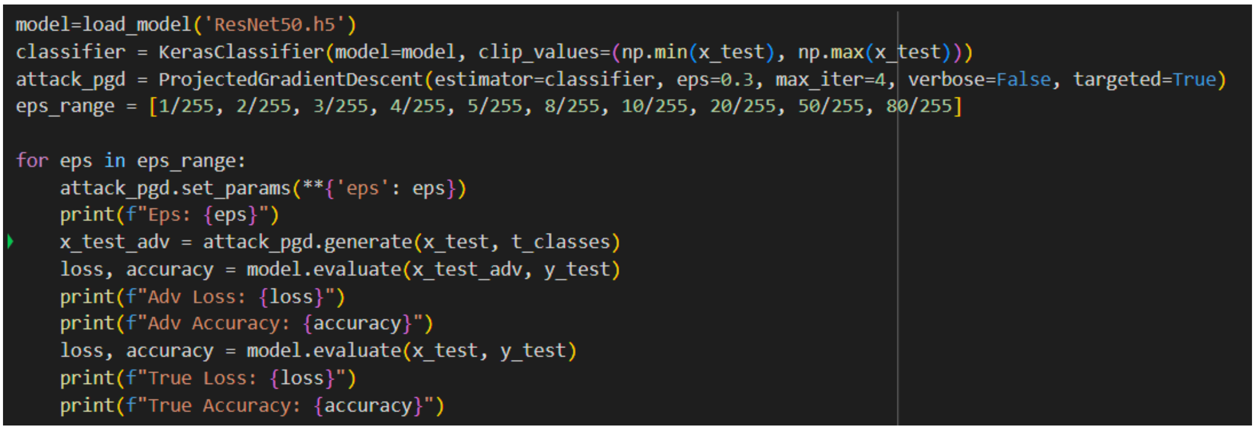


eps 0.0392156862745098: Pred class[1], Real class[14]



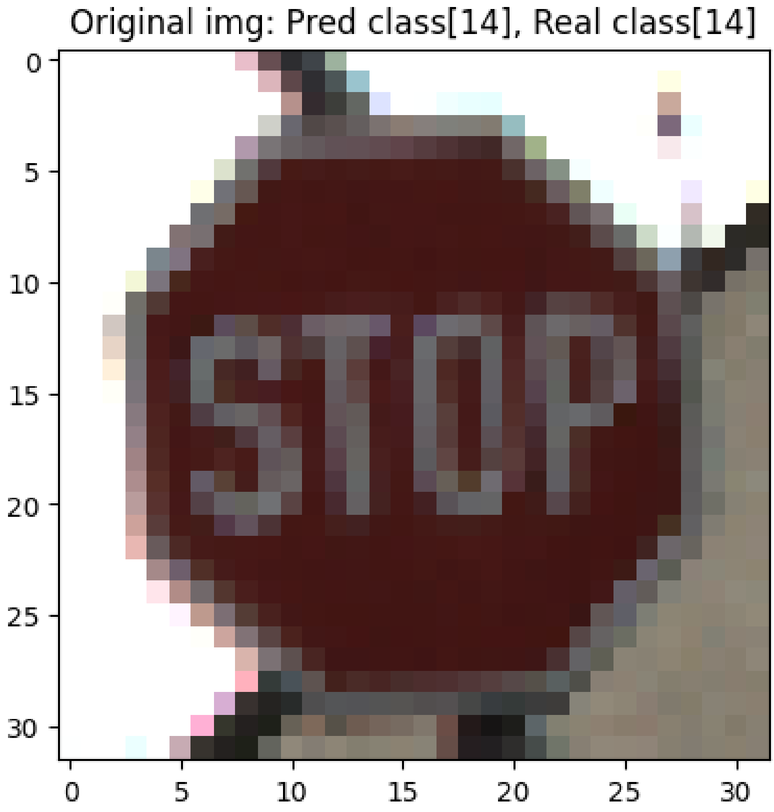


Реализация атаки PGD

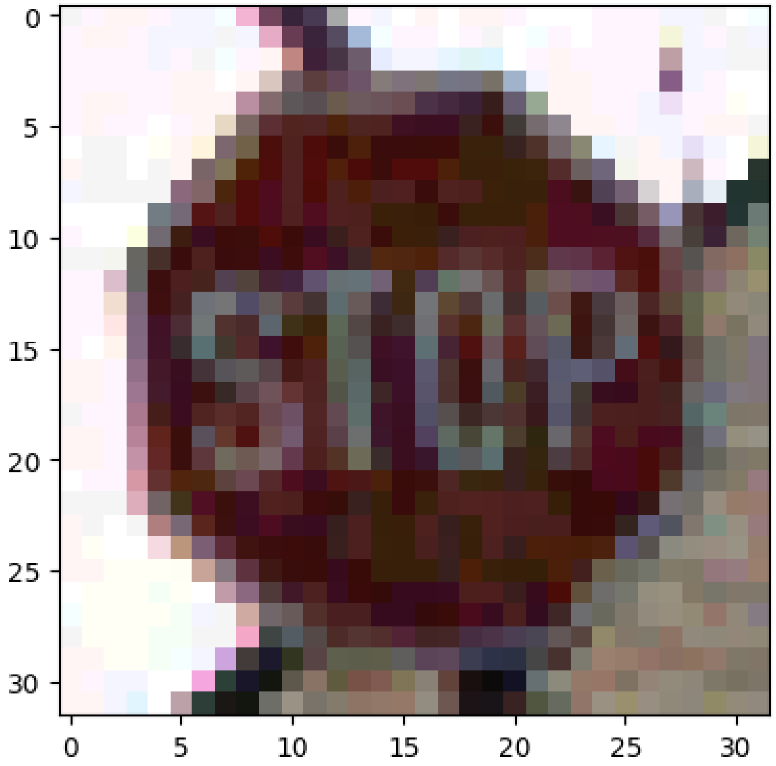


```
model=load_model('ResNet50.h5')
classifier = KerasClassifier(model=model, clip_values=(np.min(x_test), np.max(x_test)))
attack_pgd = ProjectedGradientDescent(estimator=classifier, eps=0.3, max_iter=4, verbose=False, targeted=True)
eps_range = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255]

for eps in eps_range:
    attack_pgd.set_params(**{'eps': eps})
    print(f"Eps: {eps}")
    x_test_adv = attack_pgd.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}")
```

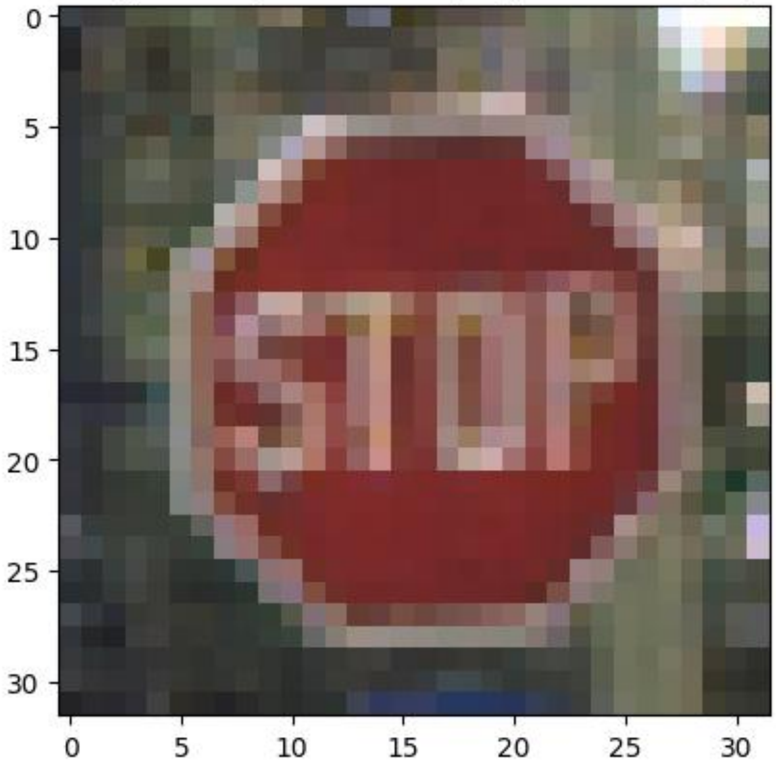
Визуализация последствий атаки



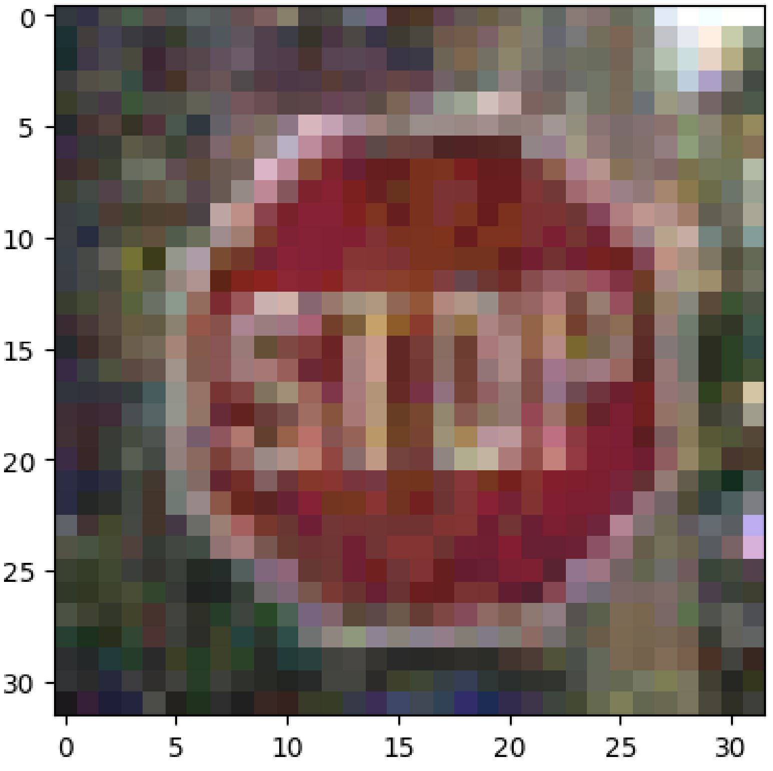
eps 0.0392156862745098: Pred class[14], Real class[14]



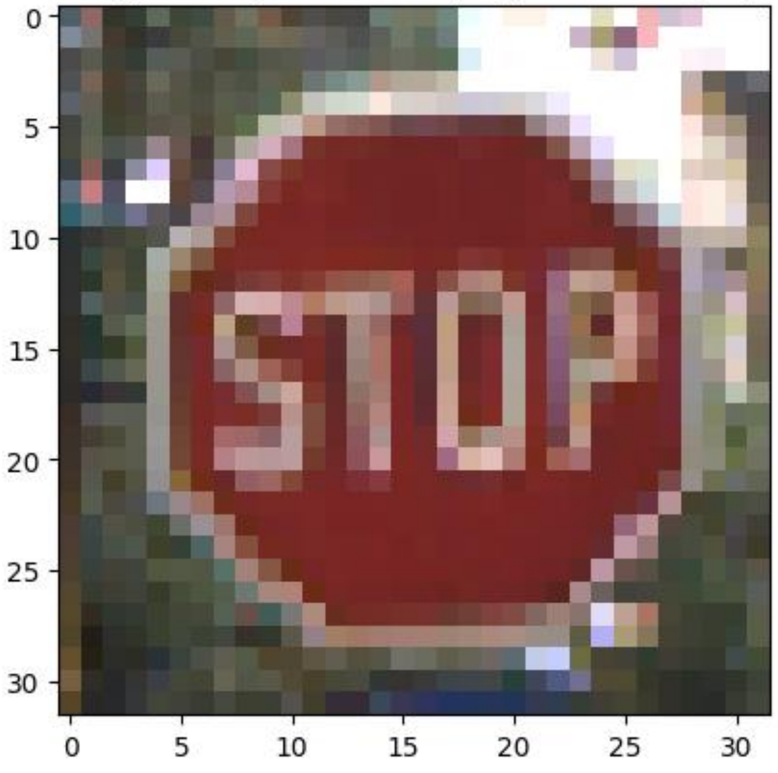
Original img: Pred class[14], Real class[14]



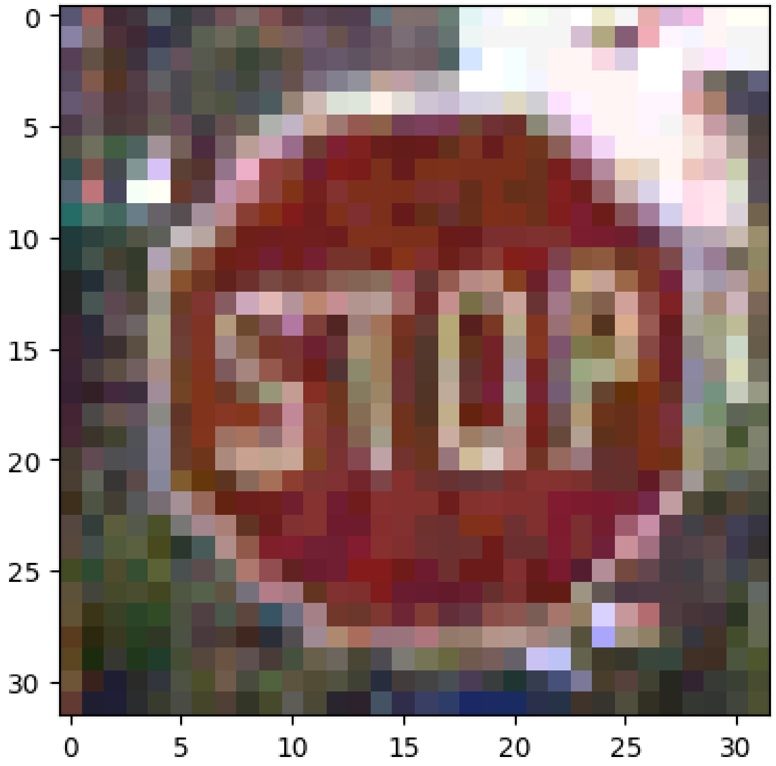
eps 0.0392156862745098: Pred class[14], Real class[14]



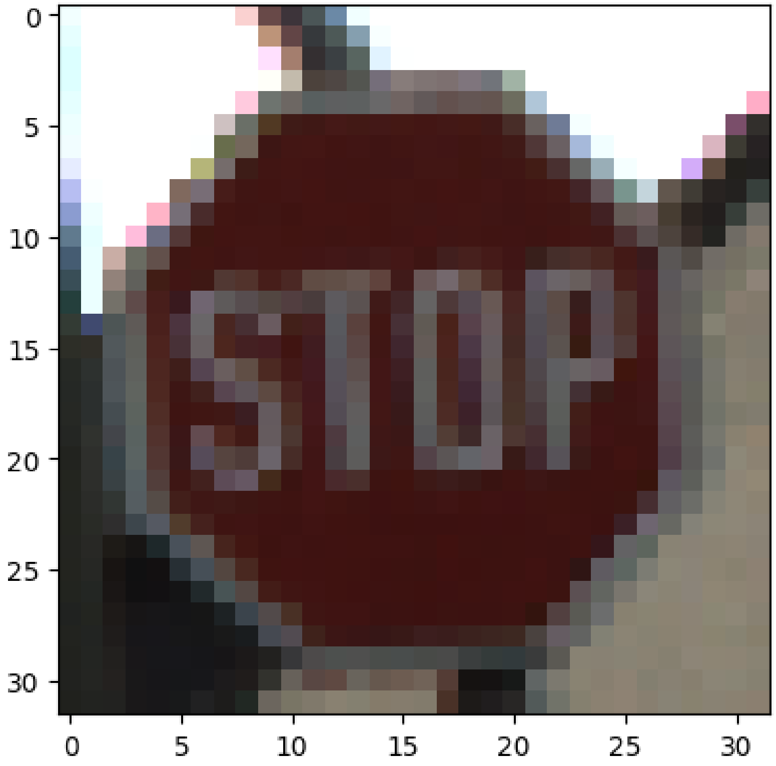
Original img: Pred class[14], Real class[14]

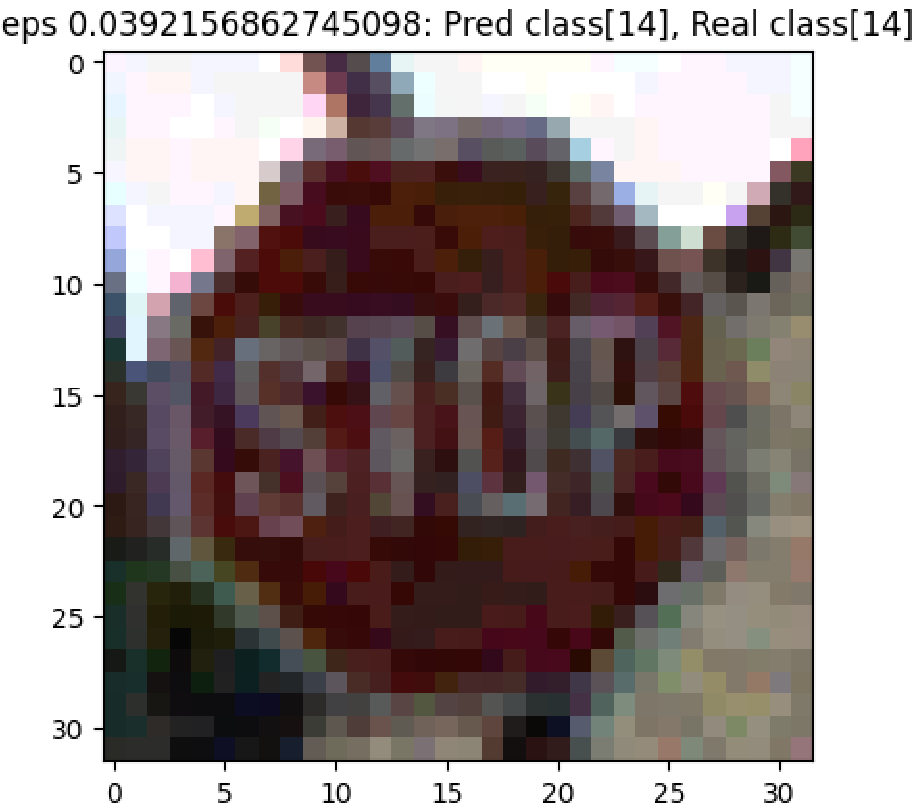


eps 0.0392156862745098: Pred class[14], Real class[14]

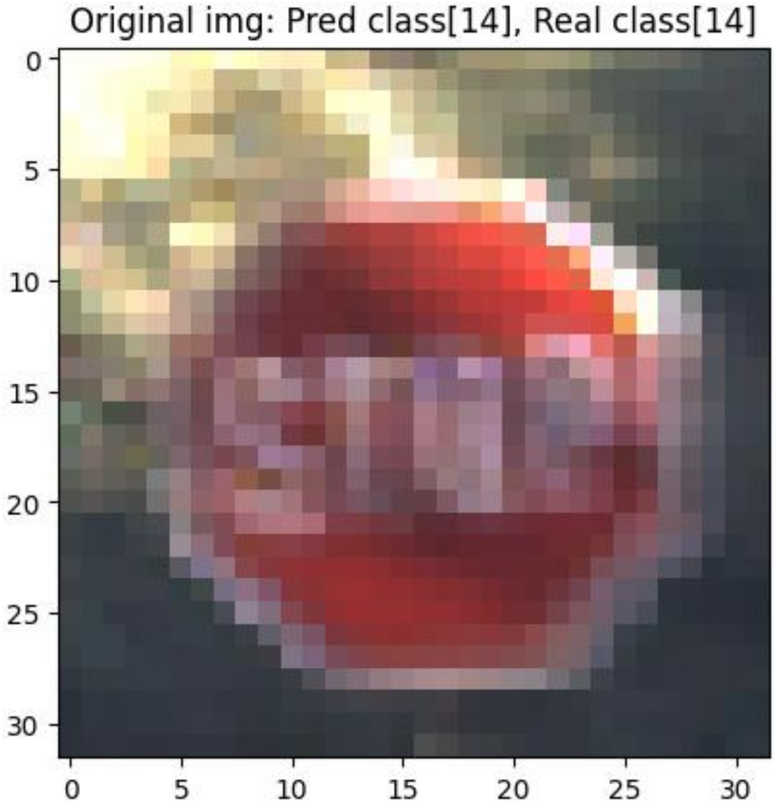


Original img: Pred class[14], Real class[14]





eps 0.0392156862745098: Pred class[14], Real class[14]



Original img: Pred class[14], Real class[14]

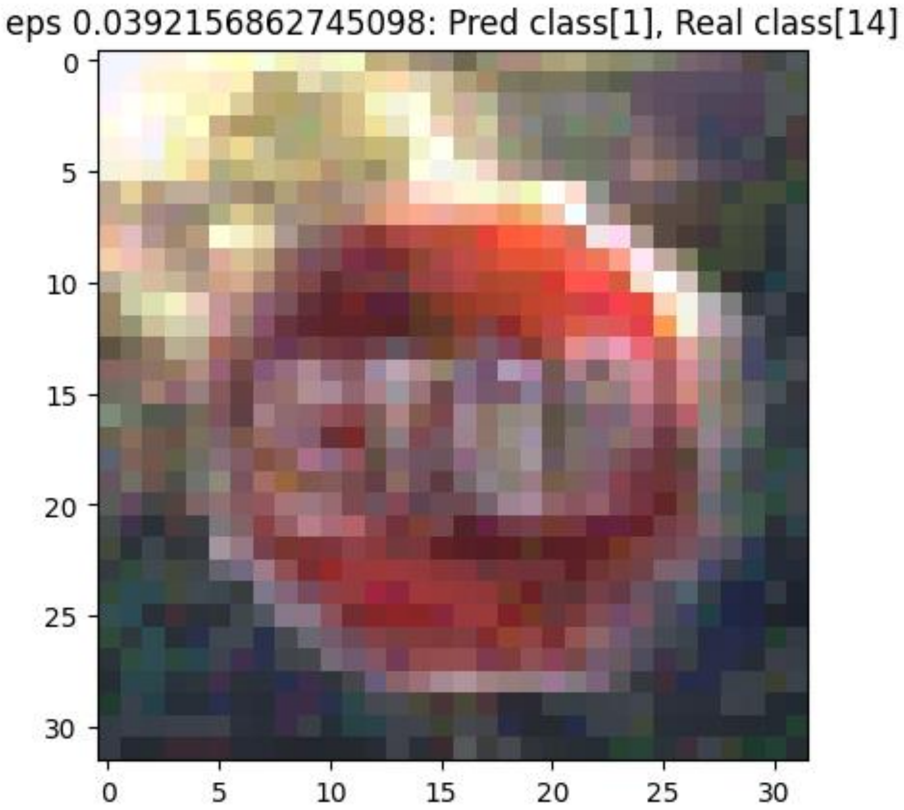


Таблица со значениями точности при двух атаках

Искажение	FGSM	PGD
ϵ=1/255	85%	97%
ϵ=2/255	76%	93%
ϵ=3/255	62%	87%
ϵ=4/255	52%	86%
ϵ=5/255	45%	79%
ϵ=8/255	19%	76%
ϵ=10/255	13%	75%
ϵ=20/255	4%	38%
ϵ=50/255	1%	5%
ϵ=80/255	1%	3%