



МИНОБРНАУКИ РОССИИ

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

Институт кибербезопасности и цифровых технологий  
Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

## Отчет по лабораторной работе 2

по дисциплине  
««Анализ защищенности систем искусственного интеллекта»»

**Выполнил**  
Козлов Ф.С.  
Группы: ББМО-02-23

Проверил:

Спирин А.А

Москва 2024 г.

### **Задачи:**

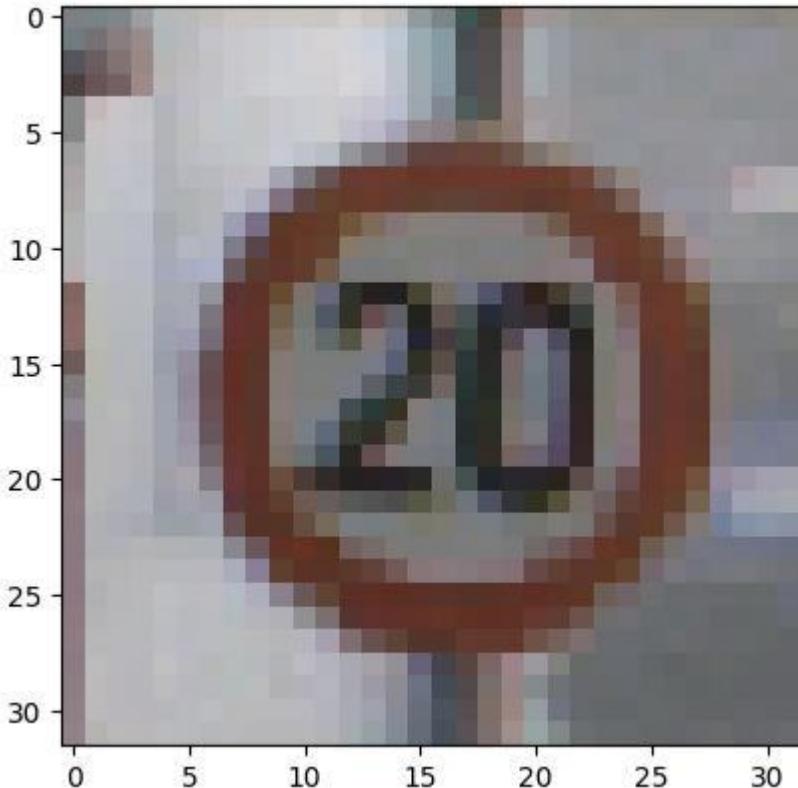
1. Реализовать атаки уклонения на основе белого ящика против классификационных моделей на основе глубокого обучения.
2. Получить практические навыки переноса атак уклонения на основе черного ящика против моделей машинного обучения.
3. Реализовать атаки уклонения на основе белого ящика против классификационных моделей на основе глубокого обучения.

**Набор данных:** Для этой части используйте набор данных GTSRB (German Traffic Sign Recognition Benchmark). Набор данных состоит примерно из 51 000 изображений дорожных знаков. Существует 43 класса дорожных знаков, а размер изображений составляет  $32 \times 32$  пикселя.

## Задание 1.

Обучим 2 классификатора на основе глубоких нейронных сетей на датасете GTSRB. При извлечении картинок для создания тренировочной выборки, получим матричное представление картинки. Для восприятия моделями нейронных сетей, данные были масштабированы.

```
<matplotlib.image.AxesImage at 0x7ae1e208ad10>
```



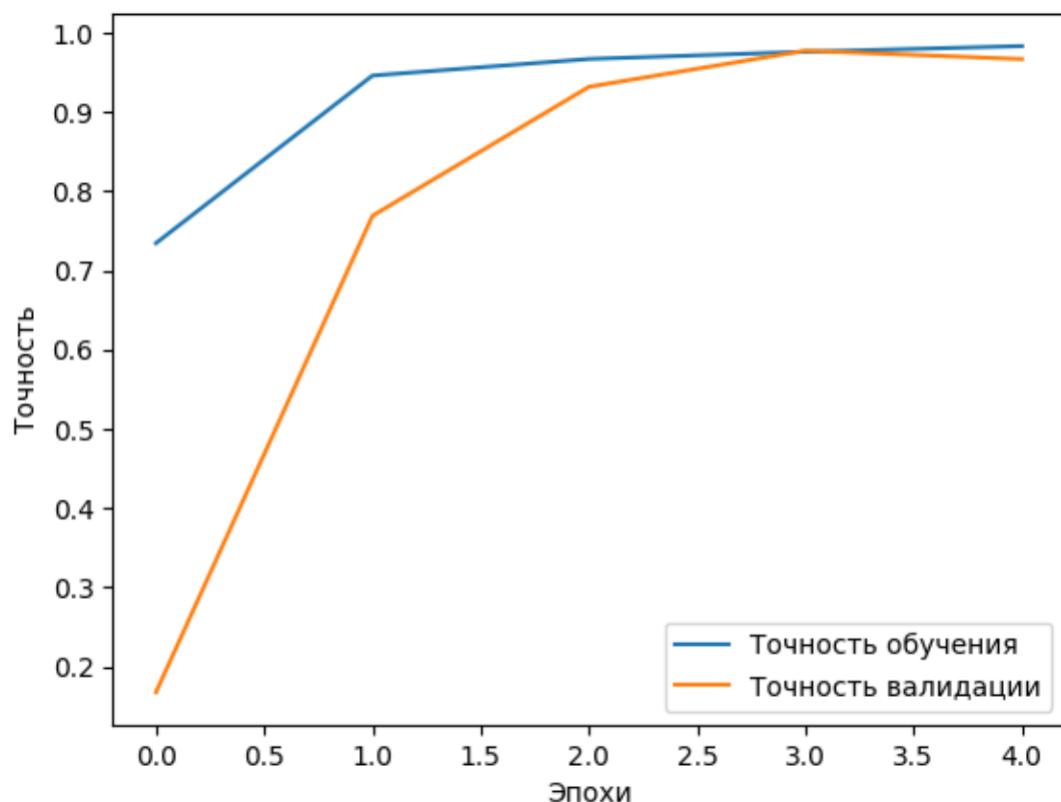
Первая модель будет ResNet50. В результате эмпирического исследования, были выбраны оптимальные значения эпох обучения и размера пакета.

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

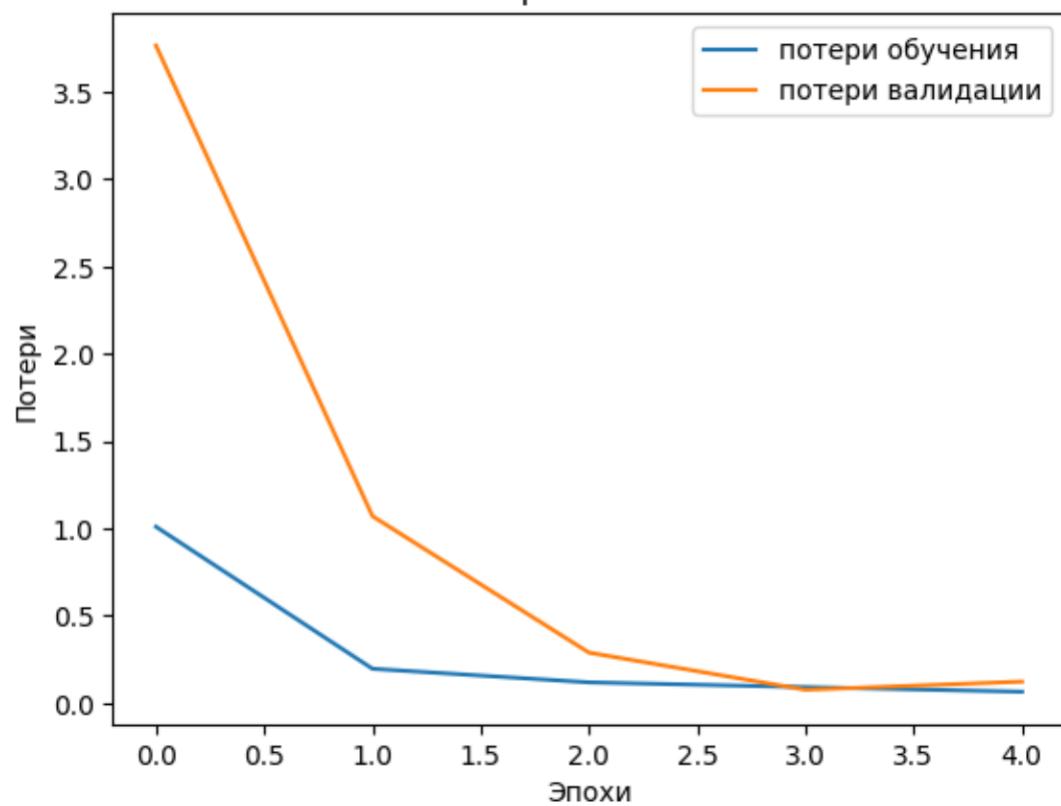
Epoch 1/5
429/429 [=====] - 60s 91ms/step - loss: 1.0102 - accuracy: 0.7343 - val_loss: 3.7652 - val_accuracy: 0.1676
Epoch 2/5
429/429 [=====] - 23s 52ms/step - loss: 0.1969 - accuracy: 0.9458 - val_loss: 1.0707 - val_accuracy: 0.7689
Epoch 3/5
429/429 [=====] - 22s 50ms/step - loss: 0.1191 - accuracy: 0.9667 - val_loss: 0.2892 - val_accuracy: 0.9317
Epoch 4/5
429/429 [=====] - 21s 50ms/step - loss: 0.0921 - accuracy: 0.9758 - val_loss: 0.0767 - val_accuracy: 0.9772
Epoch 5/5
429/429 [=====] - 23s 55ms/step - loss: 0.0656 - accuracy: 0.9830 - val_loss: 0.1233 - val_accuracy: 0.9666
```

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

### Точность RESNET50



### Потери RESNET50



Проверяем модель на тестовом наборе.

```

loss, accuracy = model.evaluate(data, y_test)
print(f'Потери тестовой выборки: {loss}')
print(f'Точность тестовой выборки: {accuracy}')

395/395 [=====] - 6s 16ms/step - loss: 0.3770 - accuracy: 0.9150
Потери тестовой выборки: 0.3769882917404175
Точность тестовой выборки: 0.9149643778800964

```

Итоговая точность составила 91%.

Обучим модель VGG16.

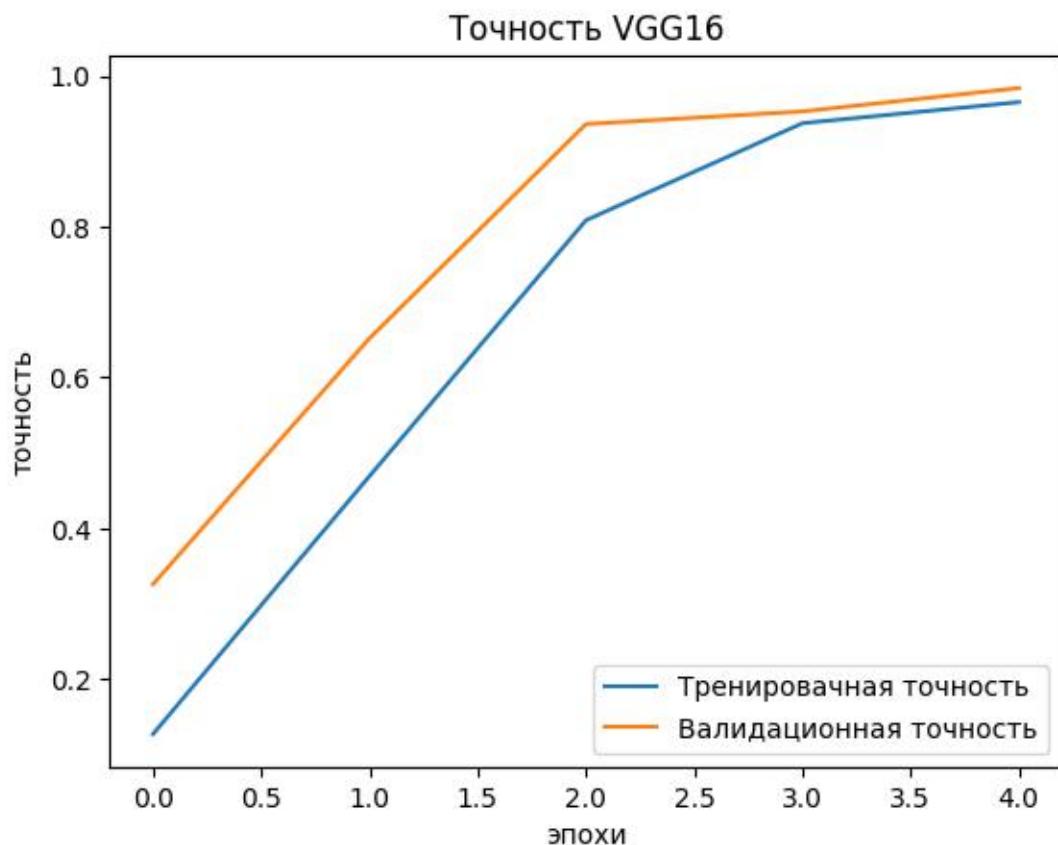
```

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

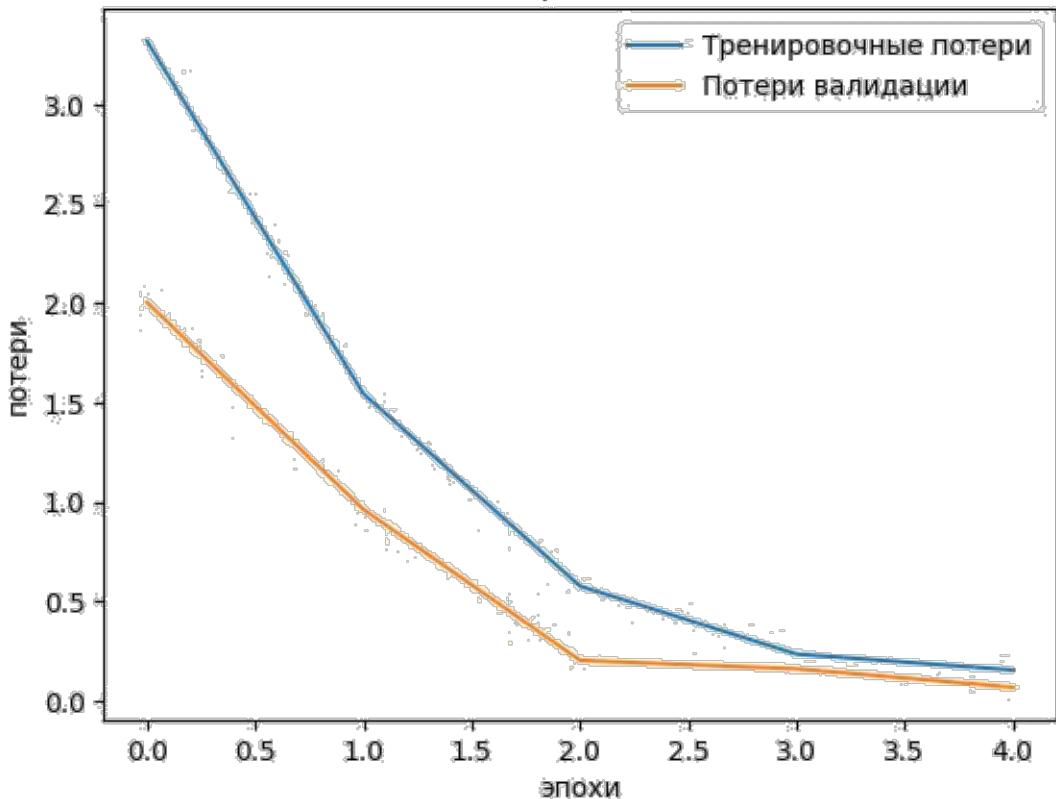
Epoch 1/5
429/429 [=====] - 27s 50ms/step - loss: 3.3199 - accuracy: 0.1271 - val_loss: 2.0057 - val_accuracy: 0.3259
Epoch 2/5
429/429 [=====] - 20s 47ms/step - loss: 1.5452 - accuracy: 0.4694 - val_loss: 0.9642 - val_accuracy: 0.6521
Epoch 3/5
429/429 [=====] - 18s 43ms/step - loss: 0.5785 - accuracy: 0.8086 - val_loss: 0.2047 - val_accuracy: 0.9362
Epoch 4/5
429/429 [=====] - 18s 43ms/step - loss: 0.2364 - accuracy: 0.9373 - val_loss: 0.1622 - val_accuracy: 0.9529
Epoch 5/5
429/429 [=====] - 18s 43ms/step - loss: 0.1563 - accuracy: 0.9653 - val_loss: 0.0697 - val_accuracy: 0.9839

```

Построим графики точности и потерь от эпох для модели VGG16



## Потери VGG16



Проверим модель на тестовом наборе.

```
loss, accuracy = model.evaluate(data, y_test)
print(f"Тестовые потери: {loss}")
print(f"Тестовая точность: {accuracy}")

395/395 [=====] - 6s 10ms/step - loss: 0.2326 - accuracy: 0.9494
Тестовые потери: 0.23263560235500336
Тестовая точность: 0.9494061470031738
```

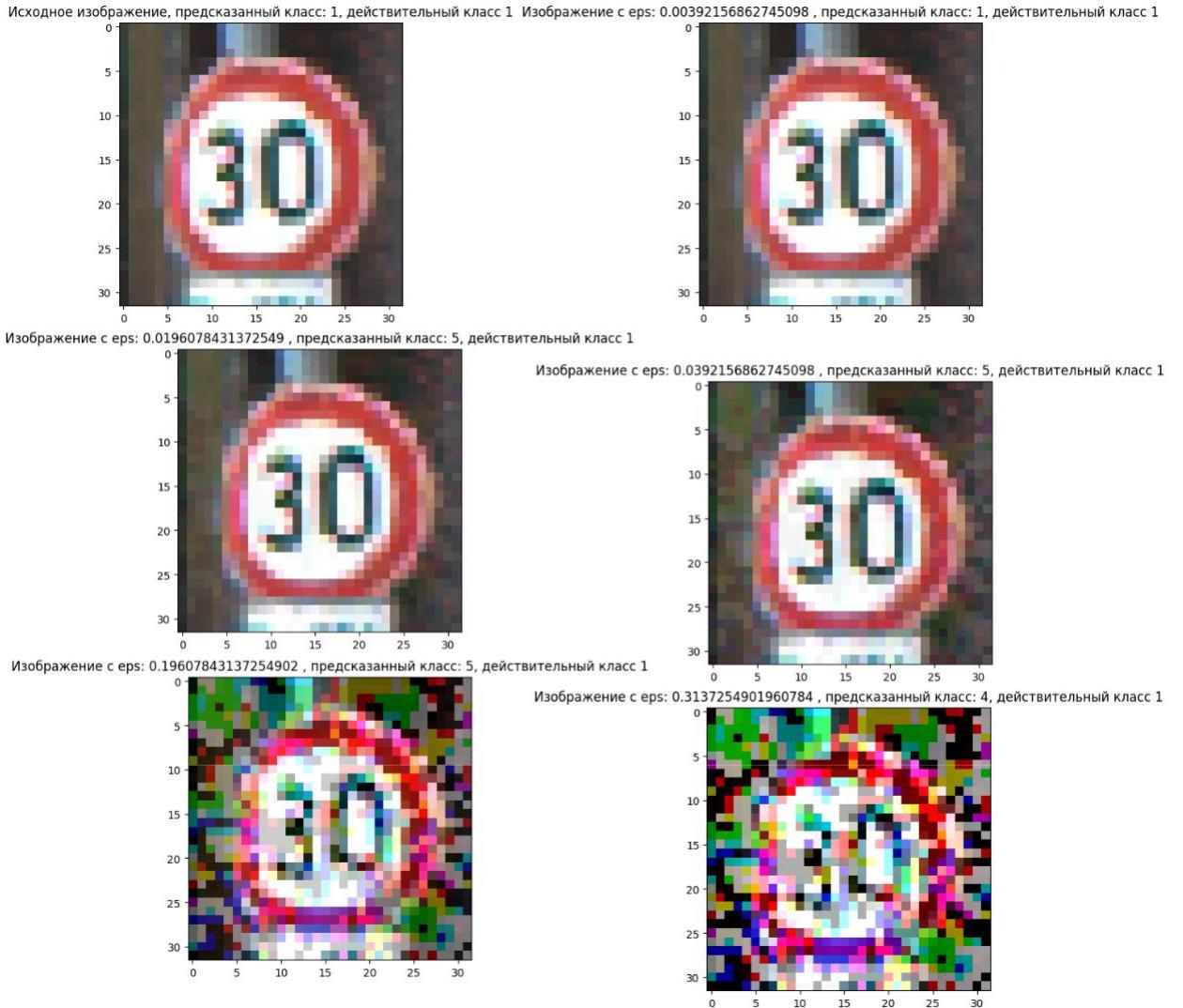
Итоговая точность составила 95%.

Составим таблицу по заданию 1.

Модель	Обучение	Валидация	Тест
ResNet50	Потери: 0.0656 Точность: 0.9830	Потери: 0.1233 Точность: 0.9666	Потери: 0.3769 Точность: 0.9149
VGG16	Потери: 0.1563 Точность: 0.9653	Потери: 0.0697 Точность: 0.9839	Потери: 0.2326 Точность: 0.9494

## Задание 2.

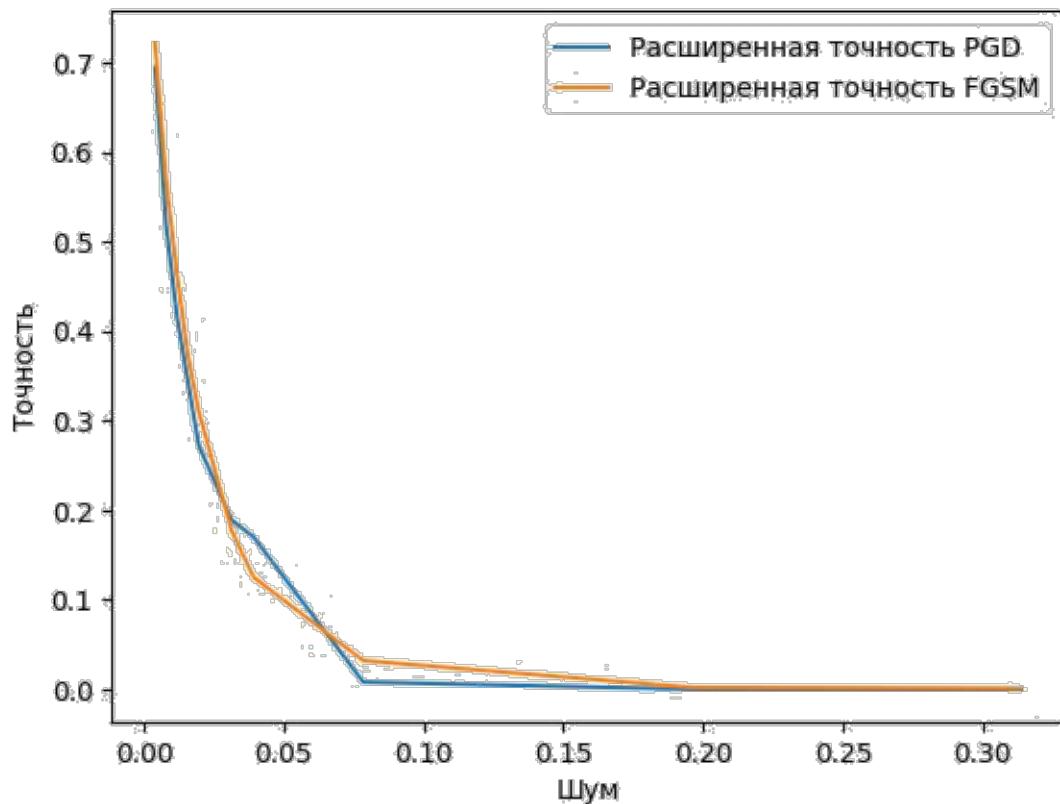
Применим нецелевую атаку уклонения на основе белого ящика против моделей глубокого обучения. Создадим модель атаки, которая основывается на классификаторе для внесения шума в изображение. Отобразим исходное и атакующие изображения для атаки FGSM



Построим график зависимости точности предсказания модели на атакованных изображениях от параметра искажения.

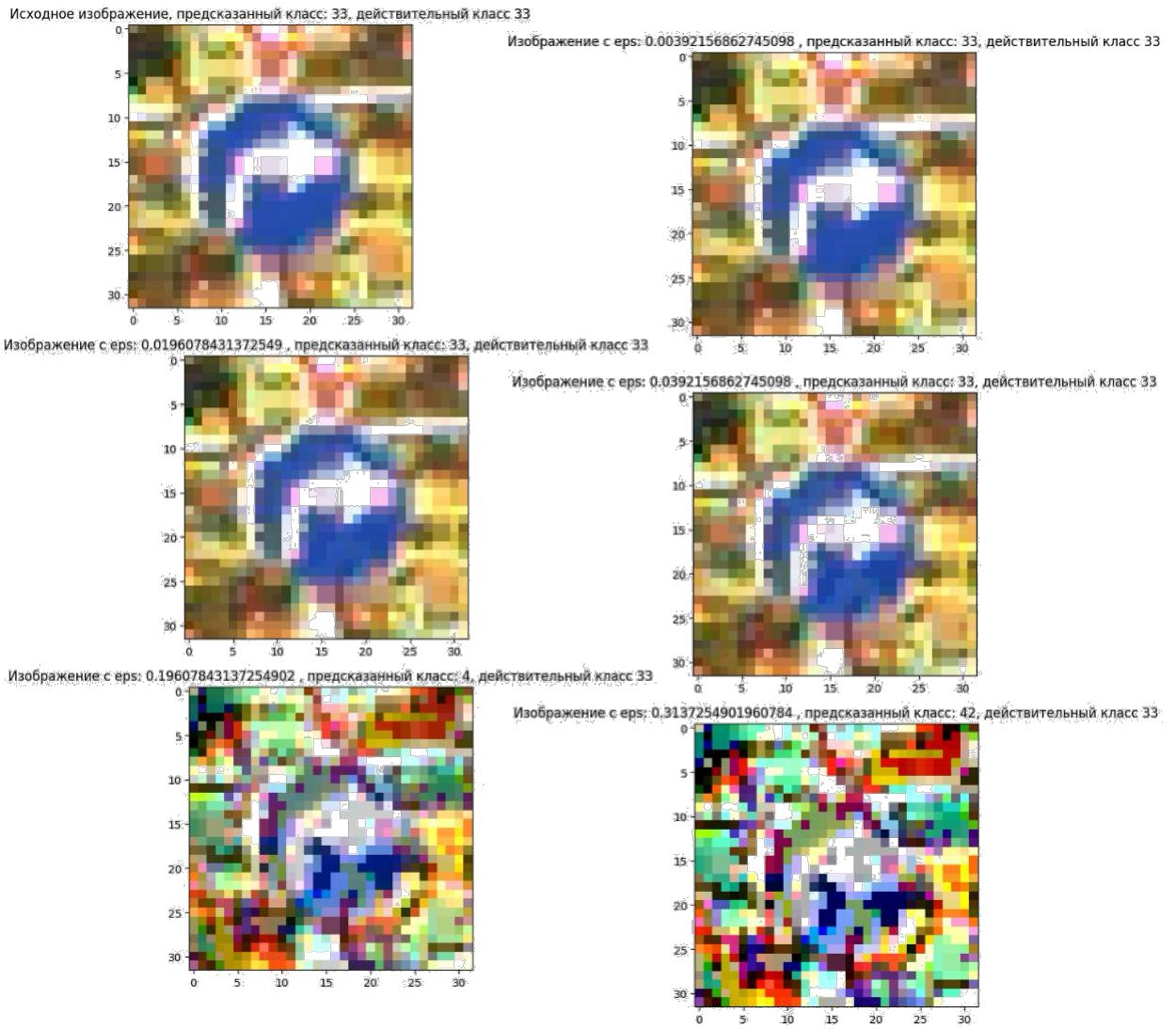
Из графика можно увидеть, что методы имеют схожую эффективность.

### Точность RESNET50

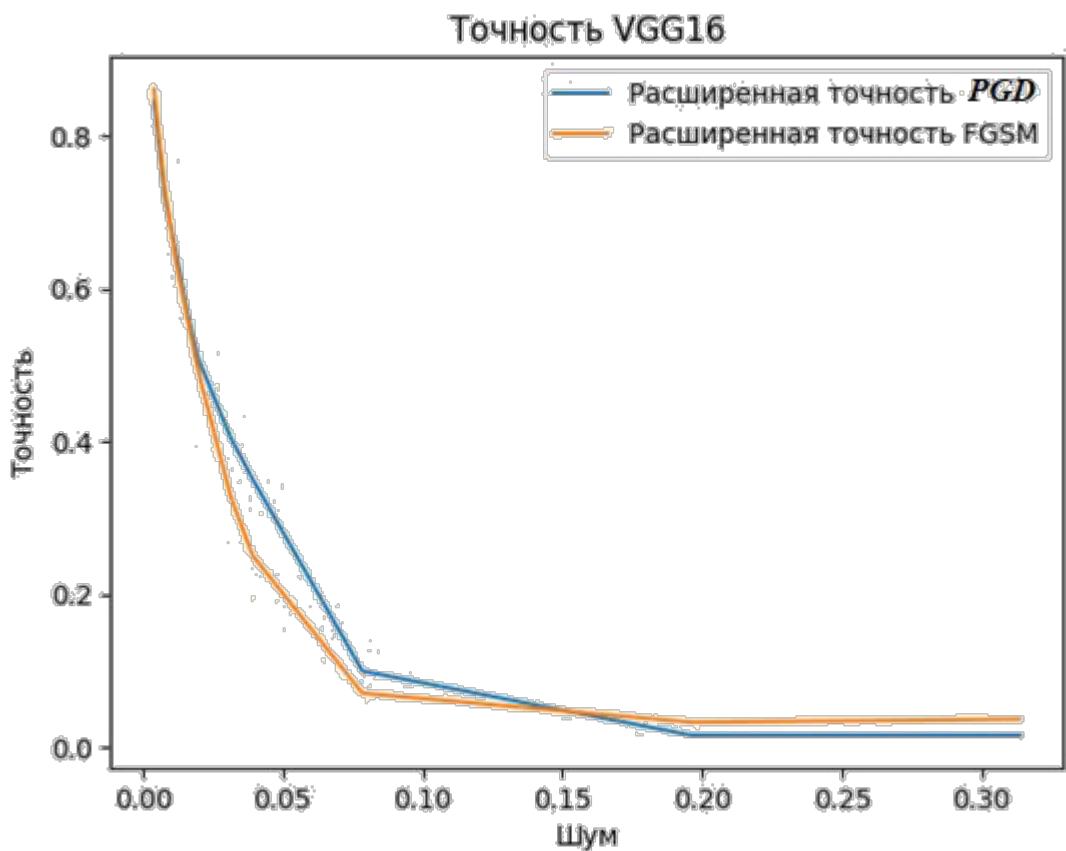


Повторим эксперимент с атаками FGSM и PGD на базе VGG16.

Для атаки FGSM отобразим исходное и атакующие изображения



Построим график зависимости точности предсказания модели на атакованных изображениях от параметра искажения.



Составим таблицу по заданию 2.

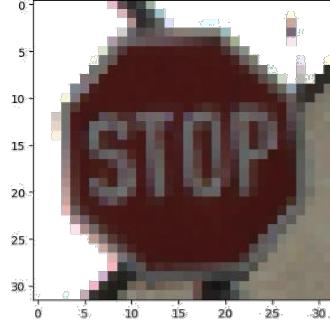
Модель	Исходные изображения	Adversarial images =1/255	Adversarial images =5/255	Adversarial images =10/255
ResNet50 – FGSM	91	72,2	31	12,5
ResNet50 – PGT	91	69,5	27,2	17
VGG16 – FGSM	95	86,1	49,4	25,1
VGG16 – PGT	95	84,5	51	35,1

### **Задание 3.**

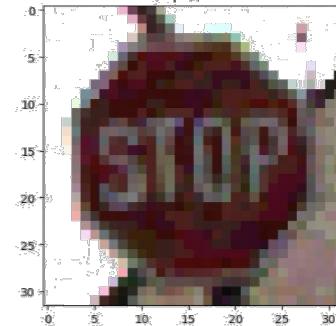
Применим целевую атаку уклонения методом белого против моделей глубокого обучения.

Используем изображения знака «STOP». Применим атаку PGD на знак «STOP» с целью классификации его как знака «Ограничение скорости 30».

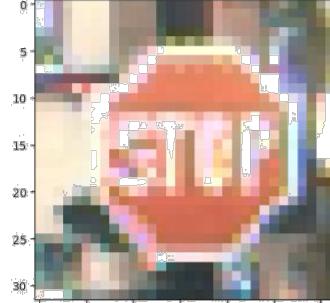
Исходное изображение, предсказанный класс: 14, действительный класс 14



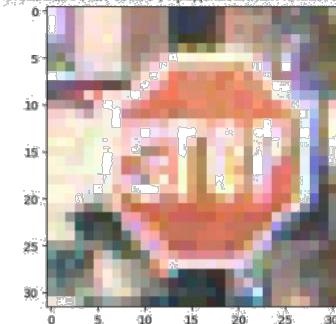
Изображение с eps: 0.0392156862745098 , предсказанный класс: 1, действительный класс 14



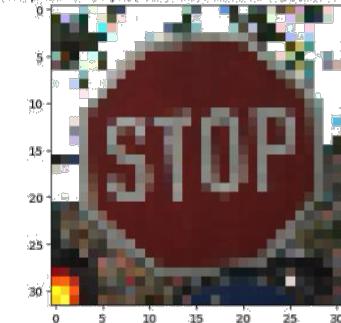
Исходное изображение, предсказанный класс: 14, действительный класс 14



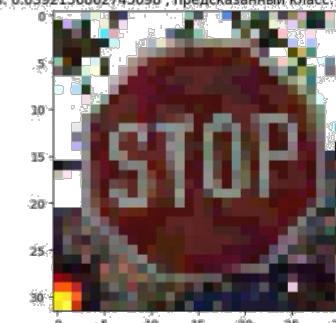
Изображение с eps: 0.0392156862745098 , предсказанный класс: 1, действительный класс 14



Исходное изображение, предсказанный класс: 14, действительный класс 14



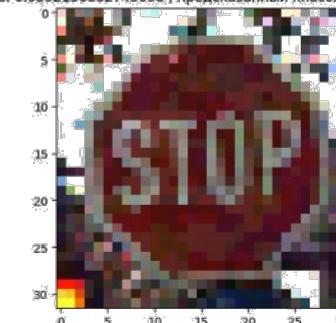
Изображение с eps: 0.0392156862745098 , предсказанный класс: 1, действительный класс 14



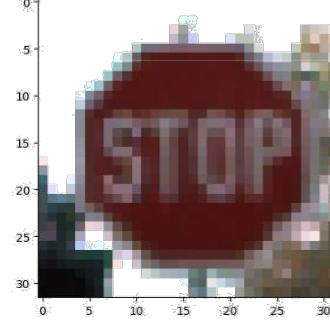
Исходное изображение, предсказанный класс: 14, действительный класс 14



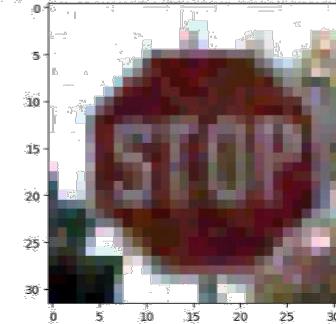
Изображение с eps: 0.0392156862745098 , предсказанный класс: 1, действительный класс 14



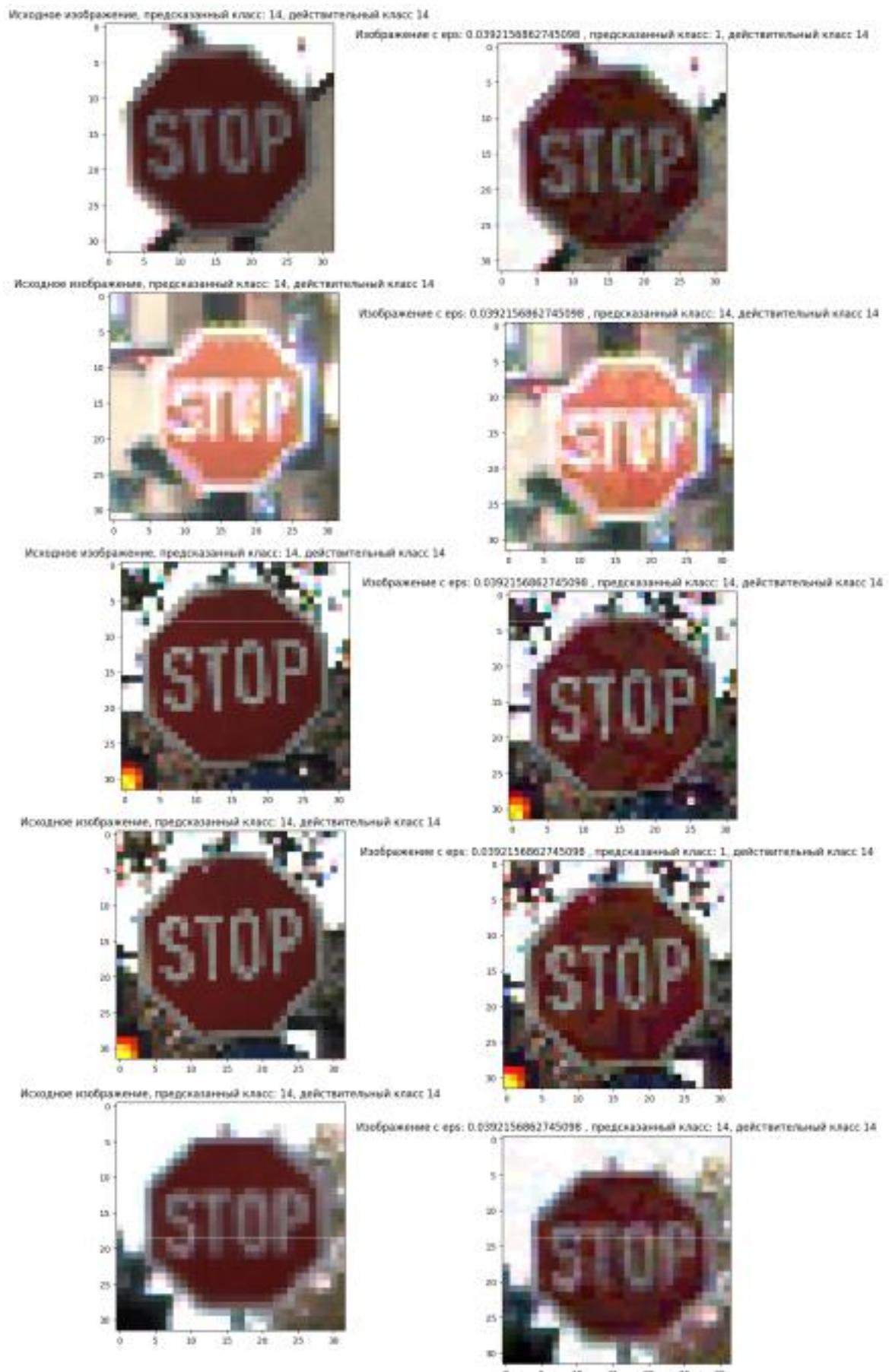
Исходное изображение, предсказанный класс: 14, действительный класс 14



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



Повторим атаку методом FGSM.



Составим таблицу по заданию 3.

Искажение	PGD attack – Stop sign images	FGSM attack – Stop images
=1/255	91,7%	95,9%
=3/255	78,1%	72,2%
=5/255	53%	37%
=10/255	48,9%	1,9%
=20/255	10,4%	0%
=50/255	0%	0%
=80/255	0%	0%

Выводы:

Метод FGSM для целевых атак не подходит, с ростом eps и шума, классификация ошибочна. Оптимальным значением искажения является 10/255, при больших значениях модель будет всегда ошибаться.

Метод PGD подходит для целевых атак, при больших значениях eps модель всегда будет определять заданный нами класс, но изображение слишком исказится. Оптимальным значением искажения является 50/255