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

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

Отчет по лабораторной работе №2
по дисциплине: «Анализ защищенности систем искусственного интеллекта»

Выполнил:
Студент группы ББМО-02-22
Давыдов Иван Дмитриевич

Проверил:
Спирин Андрей Андреевич

Москва 2024

Задачи:

1. Реализовать атаки уклонения на основе белого ящика против классификационных моделей на основе глубокого обучения.

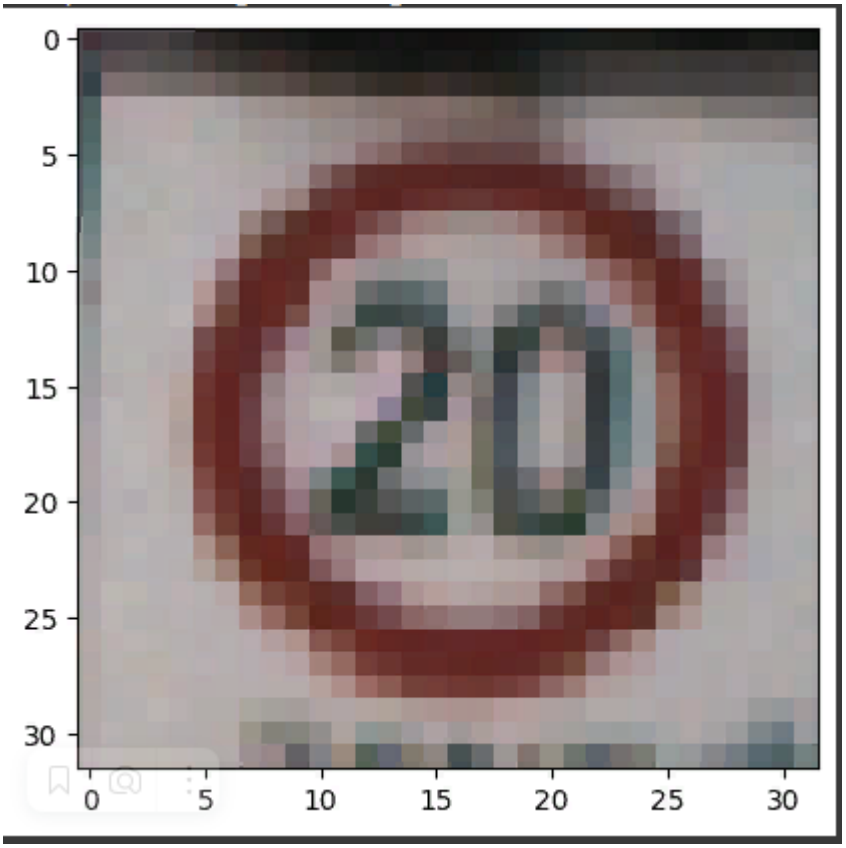
2. Получить практические навыки переноса атак уклонения на основе черного ящика против моделей машинного обучения.

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 32×32 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.



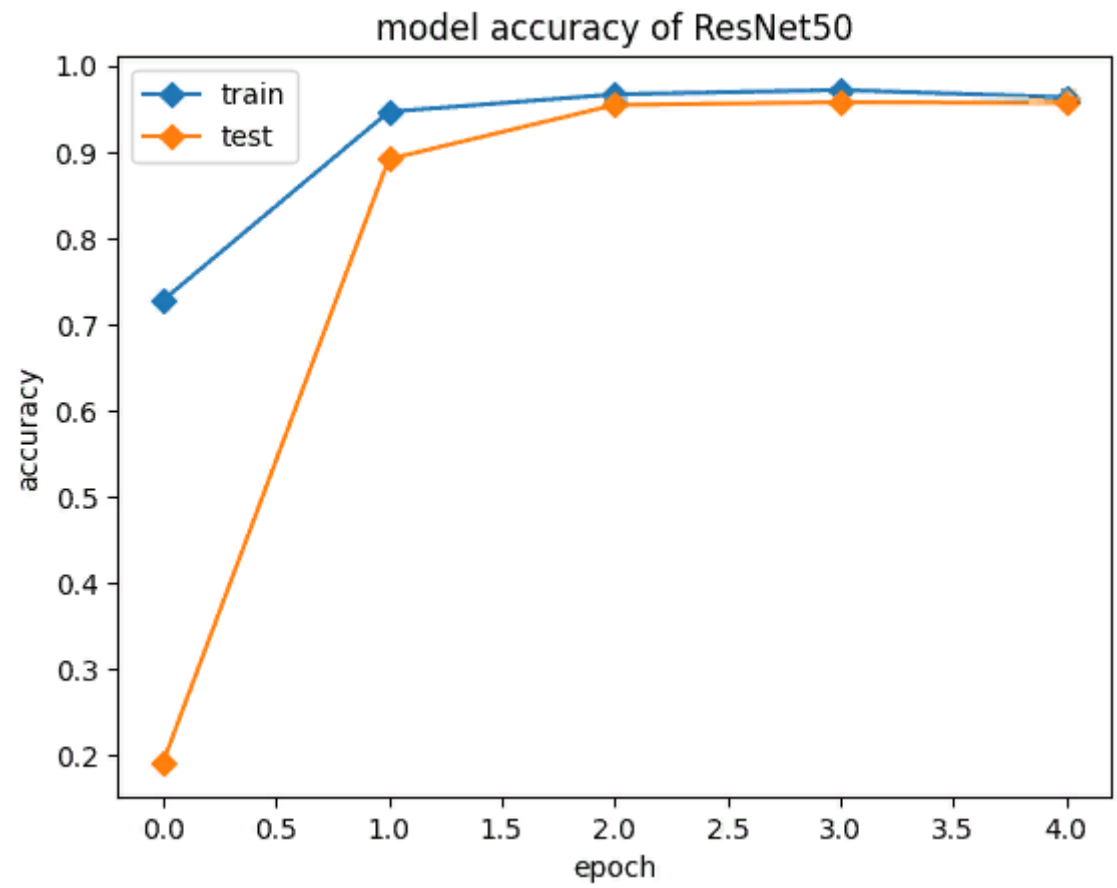
Первая модель будет ResNet50. В результате эмпирического исследования, были выбраны оптимальные значения эпох обучения и размера пакета. Так же сохраним модель для дальнейшего использования.

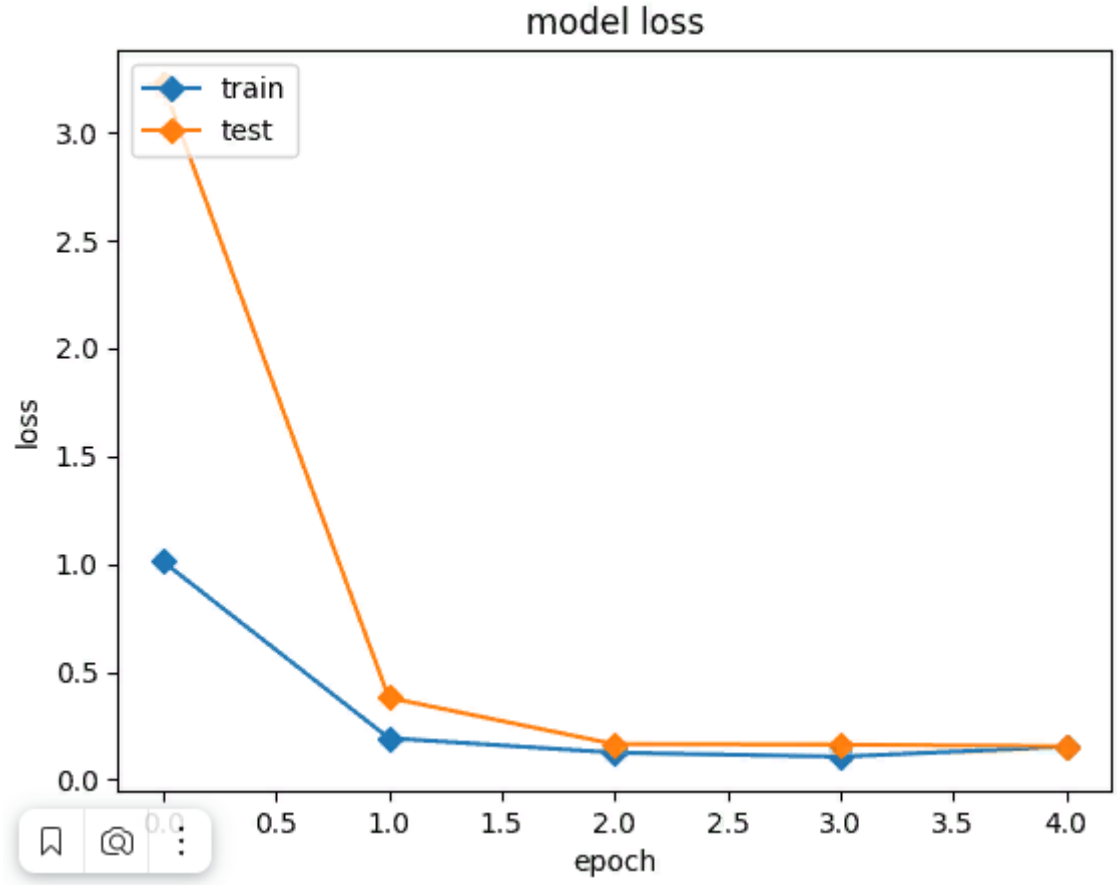
```
# обучаем модель в течение 5 эпох, используем оптимизатор Adam и
# функцию потерь categorical crossentropy
model.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])

# сохраним историю обучения для последующего анализа на графиках
history = model.fit(x_train, y_train, validation_data =(x_val, y_val),
                                        epochs = 5, batch_size = 64)
# сохраняем модель для последующего использования
save_model(model, 'ResNet50.h5')
```

```
Epoch 1/5
429/429 [==============================] - 78s 66ms/step - loss: 1.0108 - accuracy: 0.7276 - val_loss: 3.2221 - val_accuracy: 0.1914
Epoch 2/5
429/429 [==============================] - 24s 57ms/step - loss: 0.1931 - accuracy: 0.9465 - val_loss: 0.3799 - val_accuracy: 0.8915
Epoch 3/5
429/429 [==============================] - 23s 53ms/step - loss: 0.1255 - accuracy: 0.9667 - val_loss: 0.1650 - val_accuracy: 0.9544
Epoch 4/5
429/429 [==============================] - 23s 54ms/step - loss: 0.1069 - accuracy: 0.9718 - val_loss: 0.1628 - val_accuracy: 0.9575
Epoch 5/5
429/429 [==============================] - 24s 55ms/step - loss: 0.1531 - accuracy: 0.9633 - val_loss: 0.1551 - val_accuracy: 0.9567
<ipython-input-10-d9db37992c19>:9: 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 f
  save_model(model, 'ResNet50.h5')
```

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



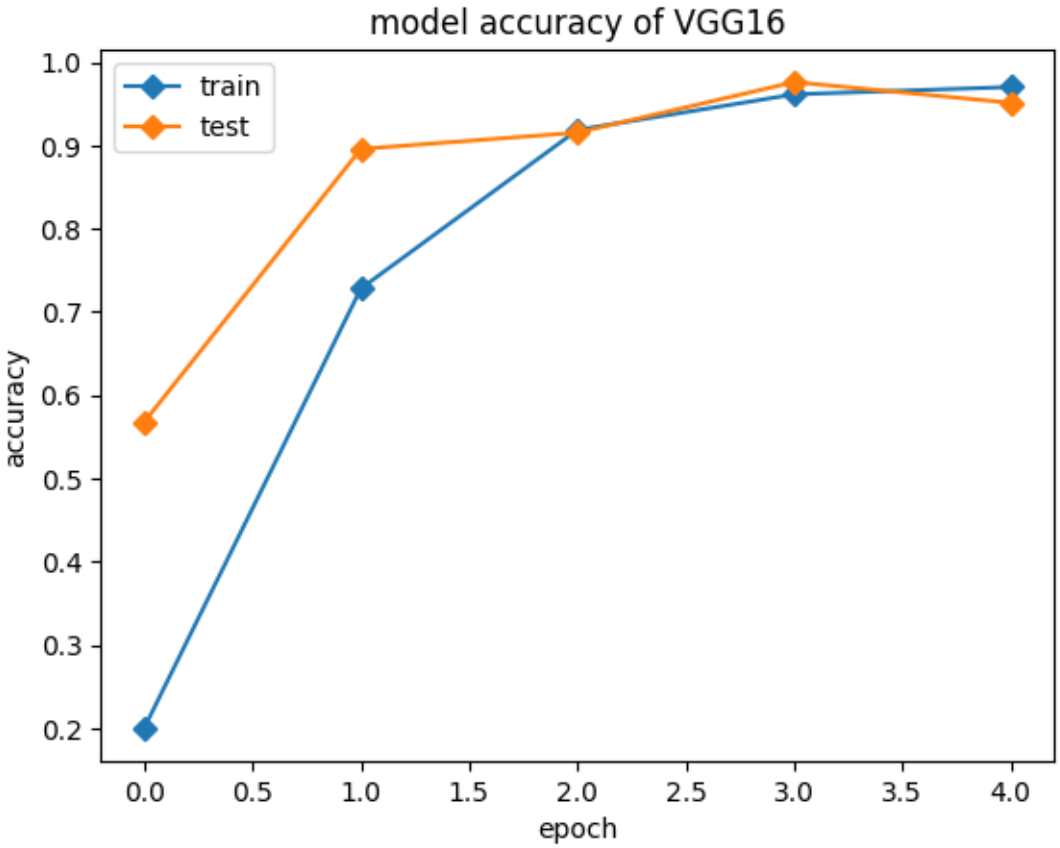


Проверяем модель на тестовом наборе.

```
# обучаем модель в течение 5 эпох, используем оптимизатор Adam и
# функцию потерь categorical crossentropy
model2.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
# сохраним историю обучения для последующего анализа на графиках
history2 = model2.fit(x_train, y_train, validation_data =(x_val, y_val),
                                                epochs = 5, batch_size = 64)
# сохраняем модель для последующего использования
save_model(model2, 'VGG16.h5')
```

```
Epoch 1/5
429/429 [==============================] - 31s 55ms/step - loss: 2.7871 - accuracy: 0.1999 - val_loss: 1.2720 - val_accuracy: 0.5676
Epoch 2/5
429/429 [==============================] - 17s 41ms/step - loss: 0.8096 - accuracy: 0.7280 - val_loss: 0.3229 - val_accuracy: 0.8957
Epoch 3/5
429/429 [==============================] - 18s 42ms/step - loss: 0.2770 - accuracy: 0.9186 - val_loss: 0.2909 - val_accuracy: 0.9153
Epoch 4/5
429/429 [==============================] - 18s 42ms/step - loss: 0.1361 - accuracy: 0.9613 - val_loss: 0.0994 - val_accuracy: 0.9760
Epoch 5/5
429/429 [==============================] - 17s 40ms/step - loss: 0.1190 - accuracy: 0.9701 - val_loss: 0.2244 - val_accuracy: 0.9511
<ipython-input-13-91ee56081336>:8: 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 f
  save_model(model2, 'VGG16.h5')
```

Обучим модель VGG16.



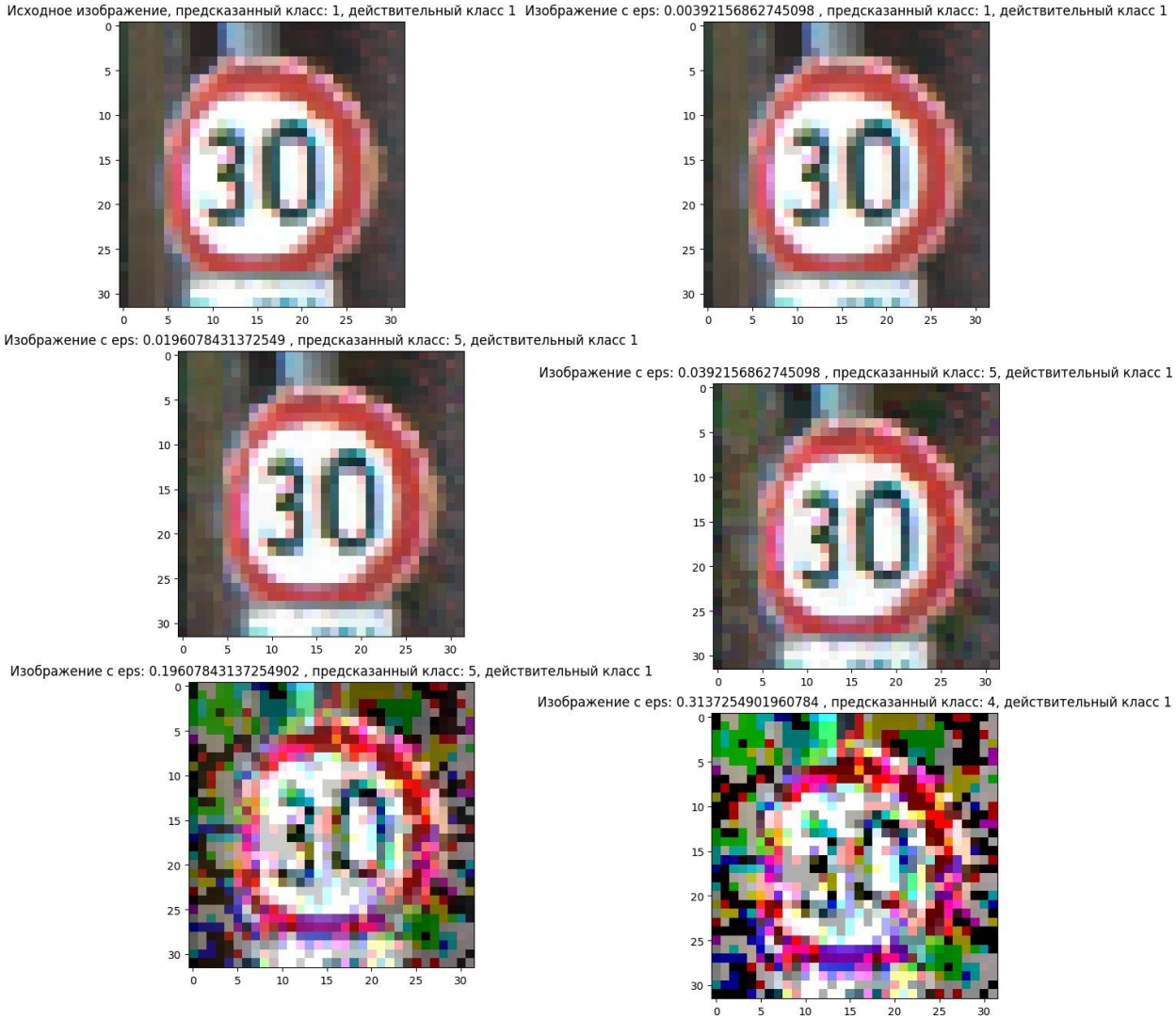
Построим графики точности и потерь от эпох для модели VGG16

Итоговая таблица сравнения моделей Resnet50 VGG16.

Model	Training Accuracy	Validation Accuracy	Test Accuracy
Resnet50	96.3346	95.6729	98.8013
VGG16	98.2487	97.7132	98.2487

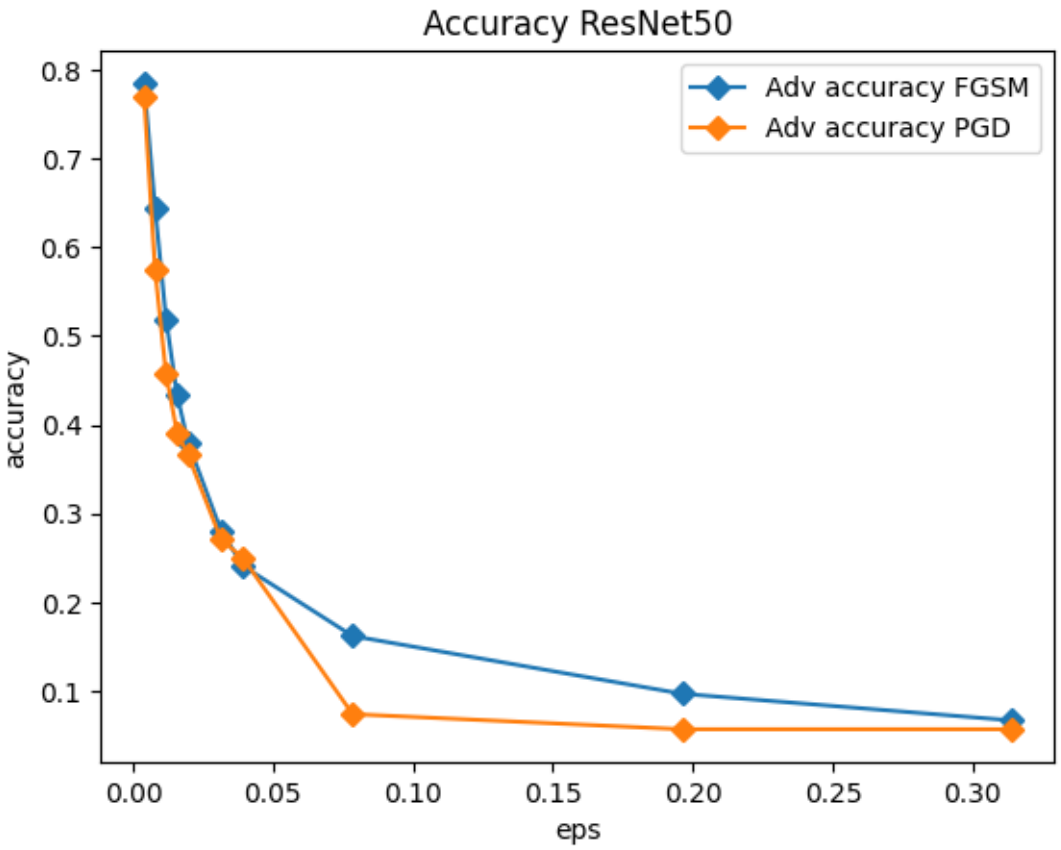
Задание 2.

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



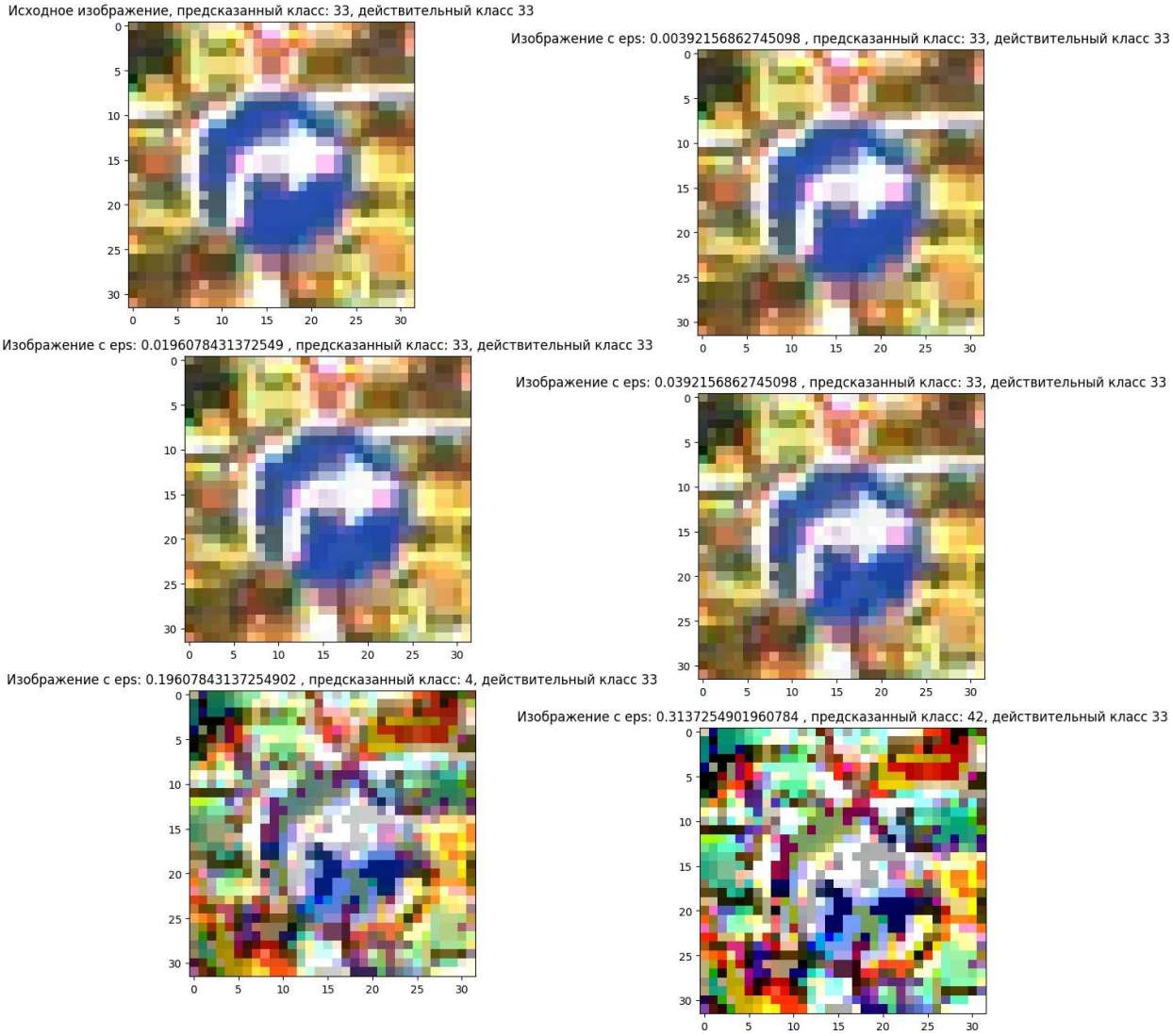
Построим график зависимости точности предсказания модели на атакованных изображениях от параметра искажения.

Из графика можно увидеть, что методы имеют схожую эффективность.

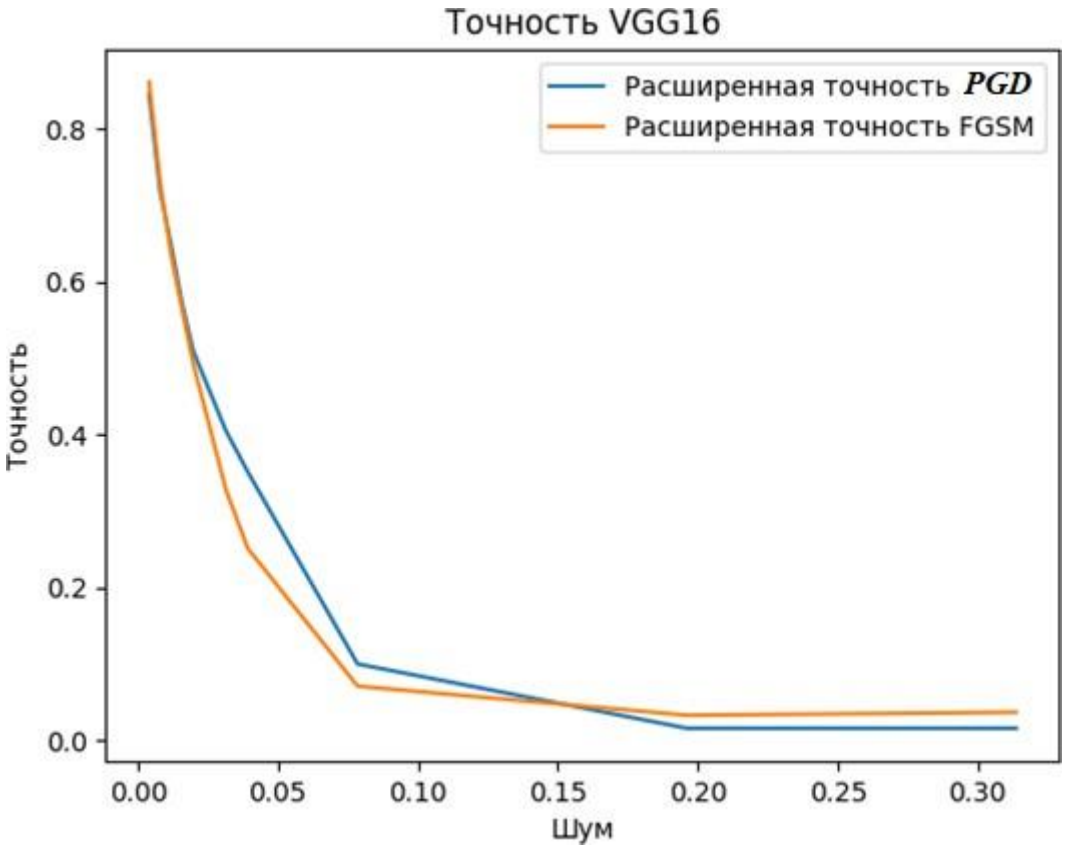


Повторим эксперимент с атаками FGSM и PGD на базе VGG16.

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.

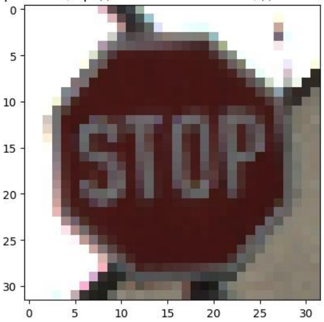
Модель	Исходные изображения	Adversarial images ϵ=1/255	Adversarial images ϵ=5/255	Adversarial images ϵ=10/255
ResNet50 – FGSM	91	72,2	31	12,5
ResNet50 – PGT	91	69,5	27,2	17
VGG16 – FGSM	95	86,1	49,4	25,1
VGG16 – PGT	95	84,5	51	35,1

Задание 3.

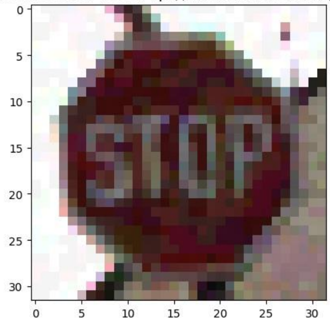
Применим целевую атаку уклонения методом белого против моделей глубокого обучения.

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

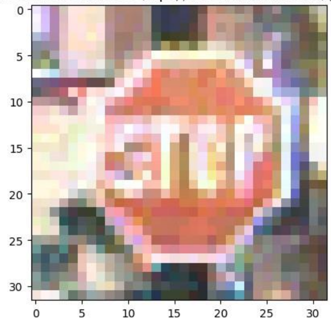


Изображение с eps: 0.0392156862745098 , предсказанный класс: 1, действительный класс 14

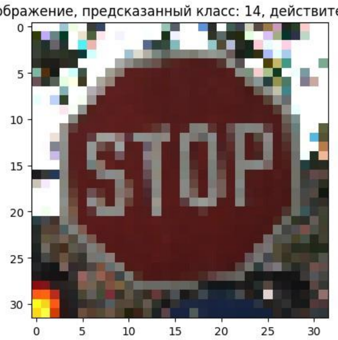


Исходное изображение, предсказанный класс: 14, действительный класс 14

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



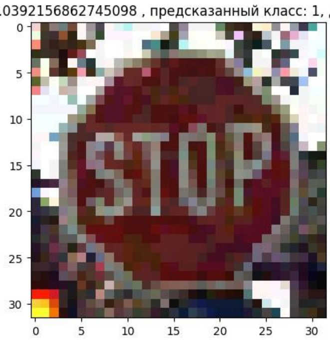
Исходное изображение, предсказанный класс: 14, действительный класс 14



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



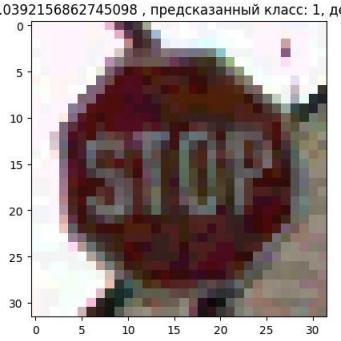
Исходное изображение, предсказанный класс: 14, действительный класс 14

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

Повторим атаку методом FGSM.

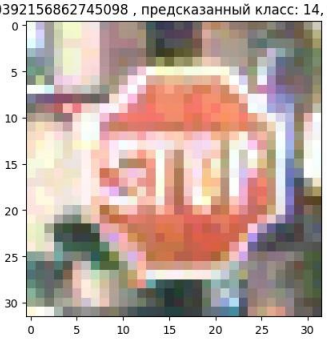
Исходное изображение, предсказанный класс: 14, действительный класс 14

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



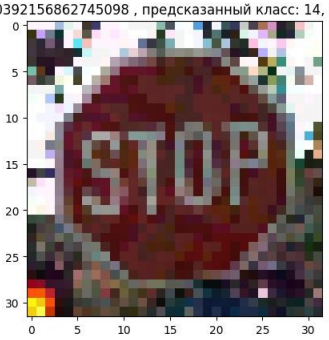
Исходное изображение, предсказанный класс: 14, действительный класс 14

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



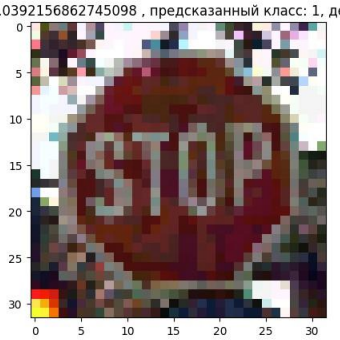
Исходное изображение, предсказанный класс: 14, действительный класс 14

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



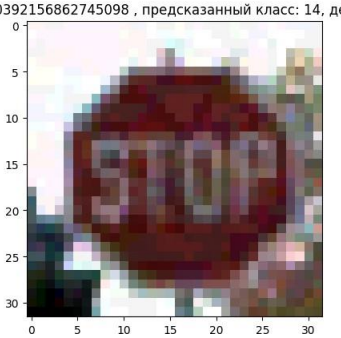
Исходное изображение, предсказанный класс: 14, действительный класс 14

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: 14, действительный класс 14



Составим таблицу по заданию 3.

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