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

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине

«Анализ защищенности систем искусственного интеллекта»

**«Лабораторная работа 2»**

Выполнил студент группы ББМО-01-22

*Лысак Я.Д.*

Проверил

*Спирин Андрей Андреевич*

«15» ноября 2023 г.

Москва, 2023 г.

## Цель работы

Выполнить задания по реализации атак против моделей на основе глубокого обучения, а также получить навыки построения и реализации атак белого и черного ящика на модели машинного обучения.

## Задание

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.

## Ход выполнения работы

В качестве набора данных для данной работы выступает GTSRB (German

Traffic Sign Recognition Benchmark). Набор состоит из 43 классов и более 50000 записей различных придорожных знаков. Размер изображений составляет 32 на 32 пикселя.

Данный набор данных можно получить по следующей ссылке: <https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

Местом проведения лабораторной работы была выбрана среда Google Collab из-за простоты подключения и доступности графического ускорителя для более быстрого расчет результатов.

Загрузка и передача файлов для обучения проводится через подключение Google диска с выполнением следующих команд (на фрагмент кода 1):

Фрагмент кода – 1

!cp drive/MyDrive/archive.zip .

!unzip archive.zip -d data

После выполнения фрагмент кода 1 в директории проекта появилась новая папка, содержащая данные для обучения, валидации и тестирования. Проводится настройка формата взводимых данных, а также загрузка данных в программу. Полная загрузка данных продемонстрирована во фрагменте кода 2 и 3.

Фрагмент кода – 2

data\_dir = './data'

train\_path = './data/Train'

test\_path = './data/Test'

image\_height = 35

image\_width = 35

channels = 3

image\_shape = (image\_height, image\_width, channels)

image\_size = (image\_height, image\_width)

batch\_size = 32

number\_of\_categories = len(os.listdir(train\_path))

Следующий фрагмент кода демонстрирует функцию загрузки данных по категориям. Таким образом в коде программы получен набор данных для обучения модели. Используется 2 подхода к загрузке данных: с помощью написанной функции load\_data() для ручной разбивки данных для дальнейшей обработки и tf.keras.preprocessing.image\_dataset\_from\_directory() для формирования генератора Keras.

Фрагмент кода – 3

data = tf.keras.preprocessing.image\_dataset\_from\_directory(train\_path)

def load\_data(data\_dir):

images = list()

labels = list()

for category in range(number\_of\_categories):

categories = os.path.join(data\_dir, str(category))

for img in os.listdir(categories):

img = load\_img(os.path.join(categories, img), target\_size=image\_size)

image = img\_to\_array(img)

images.append(image)

labels.append(category)

return images, labels

Затем создается генератор Keras для предобработки и итерации входных данных модели (фрагмент кода 4).

Фрагмент кода – 4

generator = ImageDataGenerator(width\_shift\_range=0.2,

height\_shift\_range=0.2,

rescale=1. / 255,

rotation\_range=40,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest',

validation\_split=0.2)

Применение генератора позволяет модифицировать и расширить набор данных для достижения лучшего результата обучения.

После получения генератор и загрузки в него параметров входных данных производится загрузка уже готовых моделей ResNet50 и VGG16 из пакета Keras (фрагмент кода 5).

Фрагмент кода – 5

pretrained\_model\_resnet = ResNet50(weights='imagenet',

include\_top=False,

input\_shape=image\_shape,

pooling='max')

pretrained\_model\_vgg = VGG16(weights='imagenet',

include\_top=False,

input\_shape=image\_shape,

pooling='max')

Затем проводится дополнительная обработка входных слоев, добавляются дополнительные слои и функция активации. После конфигурации модель компилируется. Целевая метрика для обучения – accuracy. Фрагмент кода для обучения модель ResNet – 6, также на рисунке 1 можно увидеть процесс обучения.

Фрагмент кода – 6

generated\_resnet = fit\_generator(train\_data,

validation\_data=validation\_data,

steps\_per\_epoch=train\_data.n // train\_data.batch\_size,

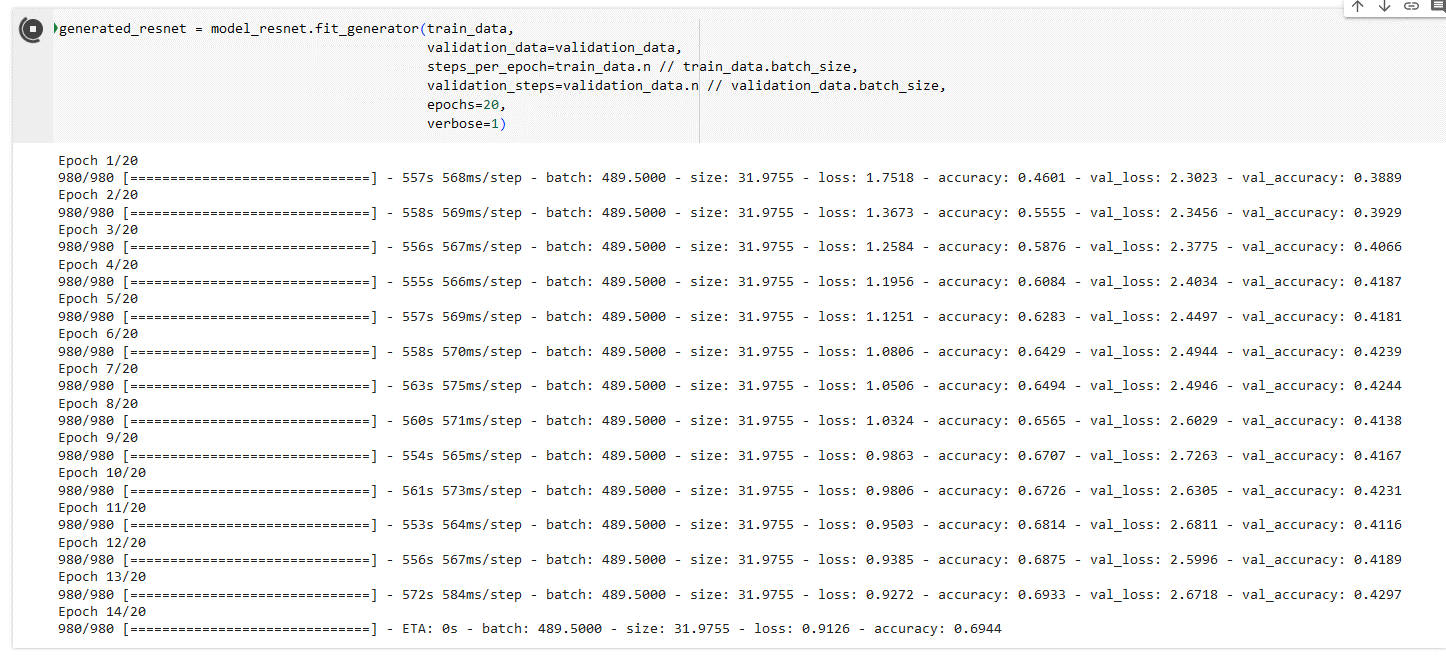
validation\_steps=validation\_data.n // validation\_data.batch\_size,

epochs=20,

verbose=1)

На рисунке 1 также изображен процесс обучения модели ResNet с валидацией на соответствующих выборках.

Рисунок – 1



По завершении обучения были сформированы следующие графики точности и потерь для моделей ResNet и VGG (рисунок 2), также конечная таблица (таблица 1):

Таблица – 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Модель | Число эпох | Обучение | | Валидация | |
| Потери (loss) | Точность (accuracy) | Потери (loss) | Точность (accuracy) |
| ResNet50 | 20 | 1,15 | 0,63 | 3,56 | 0,36 |
| VGG16 | 20 | 0,85 | 0,73 | 2,94 | 0,42 |

Рисунок – 2

После завершения обучения были заданы стартовые значения 𝜖 (эпсилон) для проведения атак Fast Gradient Sign Method (FGSM) и Projected Gradient Descent (PGD).

Для проведения атак используется библиотека Adversarial Robustness Toolbox, а именно функции FastGradientMethod и ProjectedGradientDescent. Импорт модели и первичная настройка атаки FGSM продемонстрирована во фрагменте кода 7.

Фрагмент кода – 7

eps\_list = [0.003, 0.007, 0.011, 0.015, 0.019, 0.031, 0.039, 0.078, 0.19, 0,31]

classifier\_resnet = KerasClassifier(model=model\_resnet, clip\_values=(0, 1))

for eps in eps\_list:

attack\_fgsm\_resnet = FastGradientMethod(estimator=classifier\_resnet, eps=0.03)

x\_test\_adv\_resnet = attack\_fgsm\_resnet.generate(x\_test\_first)

loss\_test, accuracy\_test = model\_resnet.evaluate(x\_test\_adv\_resnet, y\_test\_first)

perturbation = np.mean(np.abs((x\_test\_adv\_resnet - x\_test\_first)))

print('Точность на искаженных данных: {:4.2f}%'.format(accuracy\_test \* 100))

print('Среднее искажение: {:4.2f}'.format(perturbation))

Импорт модели и первичная настройка атаки PGD продемонстрирована во фрагменте кода 8.

Фрагмент кода – 8

eps\_list = [0.003, 0.007, 0.011, 0.015, 0.019, 0.031, 0.039, 0.078, 0.19, 0,31]

attack\_pgd\_resnet = ProjectedGradientDescent(estimator=classifier\_resnet, targeted=False, eps=0.03, batch\_size=128, max\_iter=20)

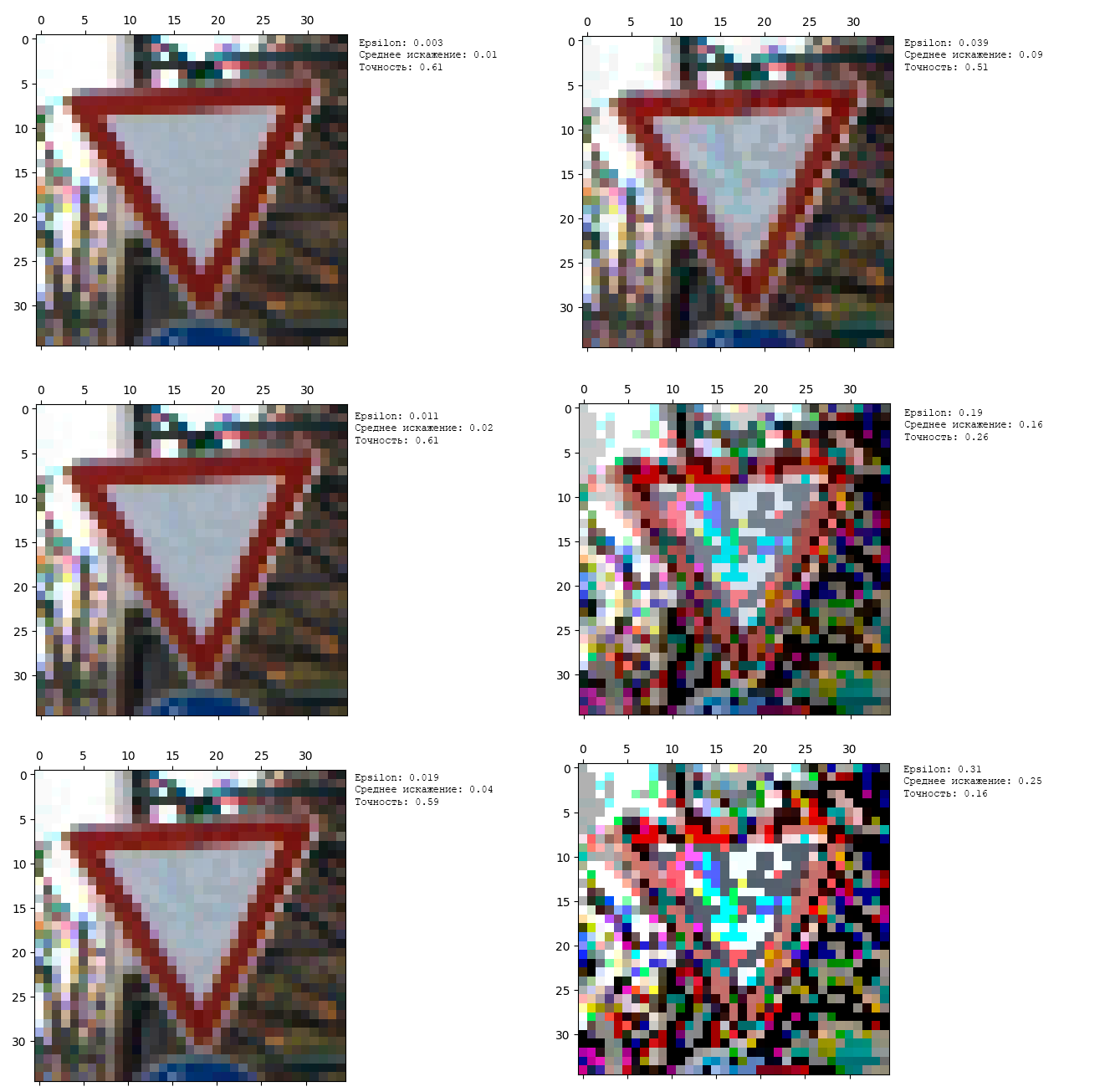
x\_train\_adv\_pgd\_resnet = attack\_pgd\_resnet.generate(x\_train\_first)

x\_test\_adv\_pgd\_resnet = attack\_pgd\_resnet.generate(x\_test\_first)

Атаки проводятся на уменьшенном наборе данных, а именно на первых 1000 элементов множества тренировочных и тестовых данных. Также важно отметить, что значения 𝜖 (эпсилон) используются из условия 𝜖 = [1/255, 2/255, 3/255, 4/255, 5/255, 8/255, 10/255, 20/255, 50/255, 80/255].

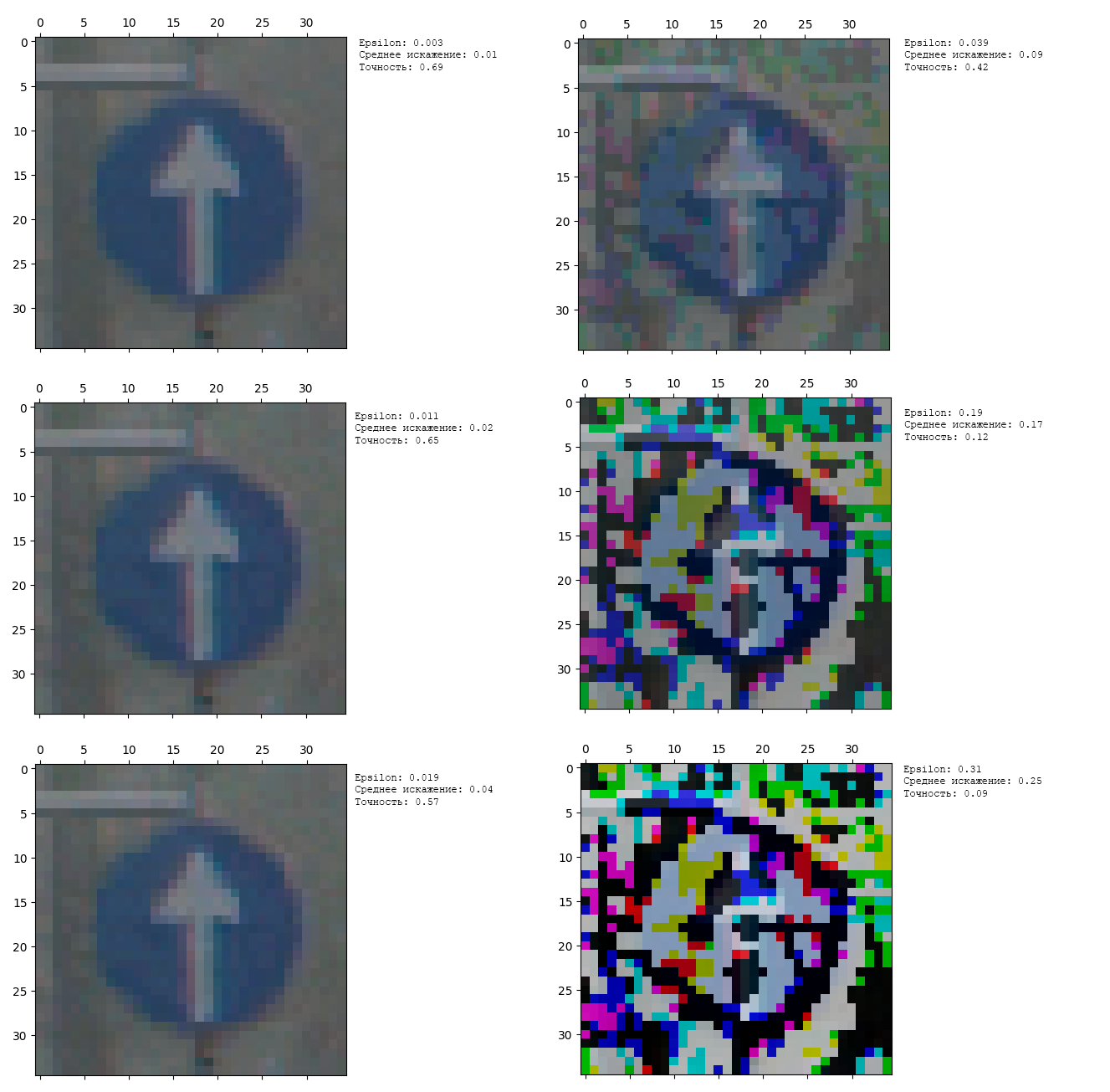
Для проверки полученных искаженных данных приведен пример на рисунке 3 с демонстрацией случайных изображений из сформированного набора атаки FGSM на модель ResNet с различными значениями 𝜖.

Рисунок – 3



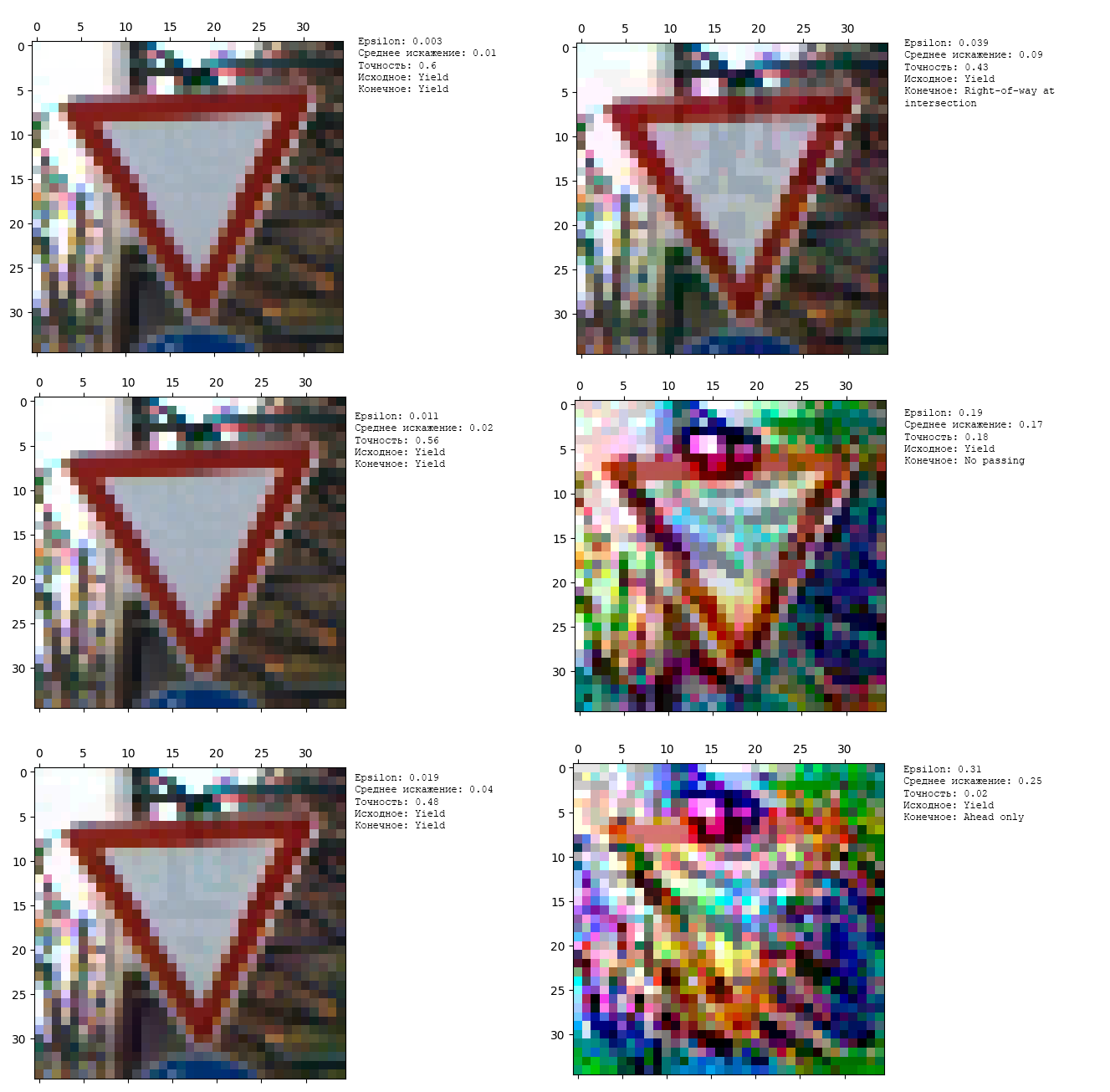
Аналогично с атакой FGSM для модели VGG на рисунке 4.

Рисунок – 4



Также была проведена атака PGD на обе модели ResNet и VGG, демонстрация полученных изображений на рисунке 5.

Рисунок – 5



По завершении подготовки атак были получены таблицы 2 и 3 для демонстрации точности классификации моделей ResNet (таблица 2) и VGG (таблица 3) в зависимости от параметра искажений 𝜖.

Таблица – 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Атака | Точность ResNet при 𝜖 (эпсилон) | | | | | |
| 1/255 | 3/255 | 5/255 | 10/255 | 20/255 | 80/255 |
| FGSM | 0,61 | 0,59 | 0,59 | 0,51 | 0,26 | 0,06 |
| PGD | 0,58 | 0,54 | 0,43 | 0,28 | 0,12 | 0,03 |

Таблица – 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Атака | Точность VGG при 𝜖 (эпсилон) | | | | | |
| 1/255 | 3/255 | 5/255 | 10/255 | 20/255 | 80/255 |
| FGSM | 0,69 | 0,65 | 0,57 | 0,42 | 0,12 | 0,09 |
| PGD | 0,6 | 0,56 | 0,48 | 0,43 | 0,18 | 0,02 |

Также был получен следующий график (рисунок 6) на основе собранных данных по атаке:

Рисунок – 6

Собранные результаты дают возможность предположить, что получаемые метрики могут сильно зависит от конкретных методов обучения и наборов данных. В данном случае VGG16 показывает наибольшую устойчивость и в большинстве случае точность результата выше. Также можно увидеть, что скорость снижения точности сильно возрастает при переходе к искажению около 20/255.

Для создания целевой атаки необходимо задать метод подготовки атаки и указать явно, что планируемая атака целевая, далее, на этапе генерации изображений, указать целевую метку класса как продемонстрировано во фрагменте кода – 9.

attack\_fgsm\_resnet\_t = FastGradientMethod(estimator=classifier\_resnet, eps=0.3, targeted=True)

x\_test\_adv\_resnet = attack\_fgsm\_resnet.generate(x\_train\_first[:1], y\_test\_first[:1])

predictions = classifier\_resnet.predict(x=x\_test\_adv\_resnet)

plt.matshow(x\_test\_adv\_resnet[:1][0])

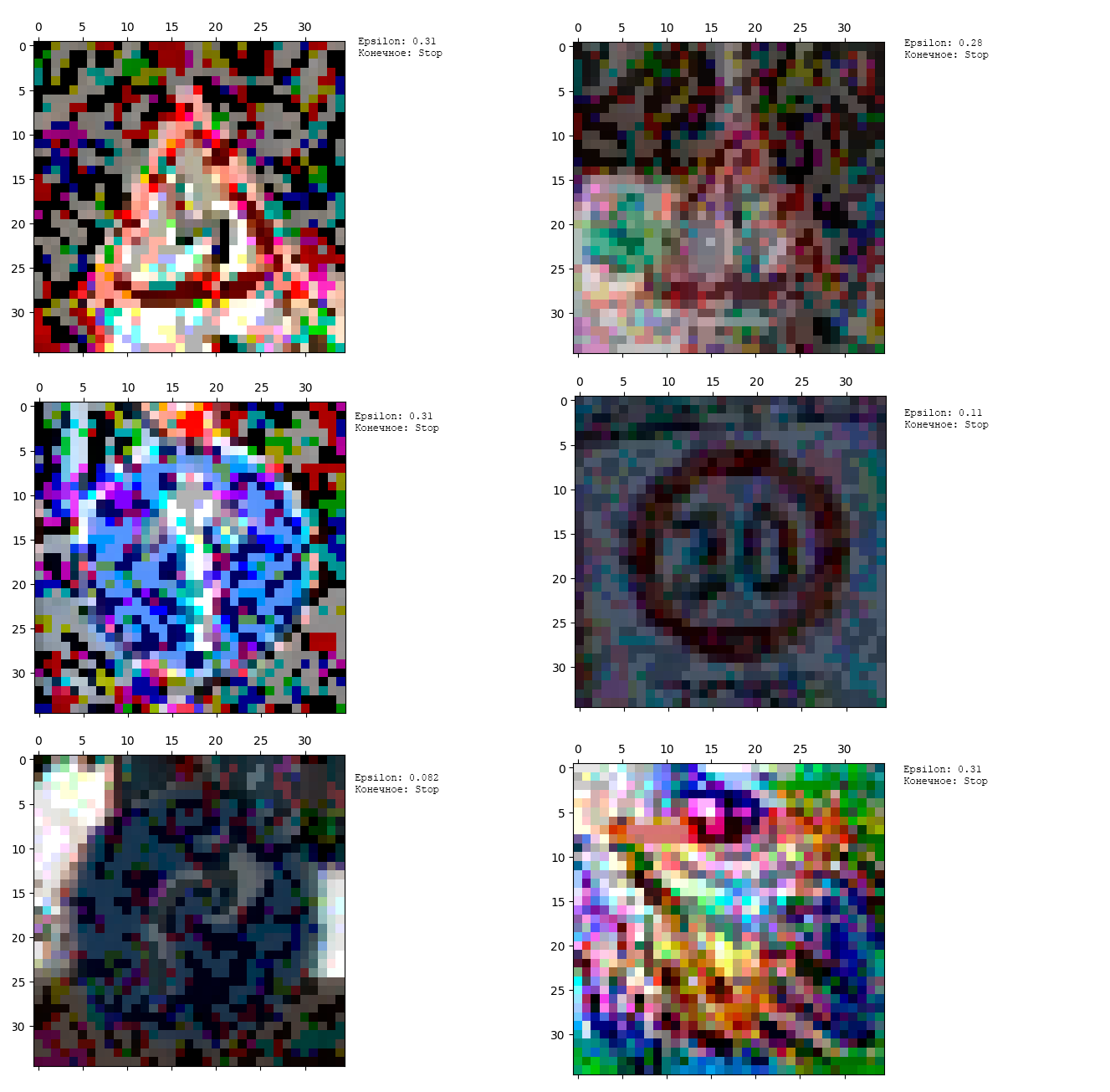
plt.show()

print('adv: %s' % classes[np.argmax(classifier\_resnet.predict(x\_test\_adv\_resnet[:1, :])[0])])

print('og: %s' % classes[np.argmax(classifier\_resnet.predict(x\_test[:1, :])[0])] )

В качестве примера на рисунке 7 приведены примеры успешного формирования целевой атаки:

Рисунок – 7



Также на данном этапе получилось сформировать таблицу условной эффективности точности данных механизмов по атаке на наборы данных (таблица 4):

Таблица – 4

|  |  |  |
| --- | --- | --- |
| 𝜖 | PGD - Stop | PGD – Limit 30 |
| 𝜖 = 1/255 | 0,58 | 0,56 |
| 𝜖 = 3/255 | 0,58 | 0,48 |
| 𝜖 = 5/255 | 0,52 | 0,46 |
| 𝜖 = 10/255 | 0,47 | 0,34 |
| 𝜖 = 20/255 | 0,34 | 0,26 |
| 𝜖 = 50/255 | 0,08 | 0,08 |
| 𝜖 = 80/255 | 0,03 | 0,01 |

Данные демонстрируют, что знак с ограничение лучше поддается атаке. Это можно объяснить тем, что фото с данным знаком часто размытые, что приводить к сложности его распознавания, а именно определить какой именно лимит скорости, так как в данном наборе есть и другие лимиты. Также важно отметить, что в данном конкретном примере знак стоп более эффективен для атаки на уровне искажения 20/255, более сильное искажение приводит серьезным повреждениям исходного изображения и уже не дают такой значительный прирост (упадок) точности.

**Вывод**

В результате выполнения работы получен некоторый опыт работы с инструментами организации атак на модели машинного обучения. Были также проведены эксперименты по атаке на модели машинного обучения методом черного и белого ящика, а также целевые и нецелевые.

В рамках работы были рассмотрены модели VGG16 и ResNet50, VGG16 показала несколько большую устойчивость к атакам, хоть и не значительную, но уже не в рамках погрешности. Было отмечено сильно ухудшение качества и точности моделей по достижении отметки искажения в 20/255.

При работе с целевыми атаки было отмечено, что наиболее эффективным значение, в рамках данного набора данных, является значение искажения 20/255.