МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российскийтехнологическийуниверситет»
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 дисципилне

«Анализ зазищенности систем искусственного интеллекта»

Группа:
ББМО-02-22
Выполнила:
Волкова Е.А.

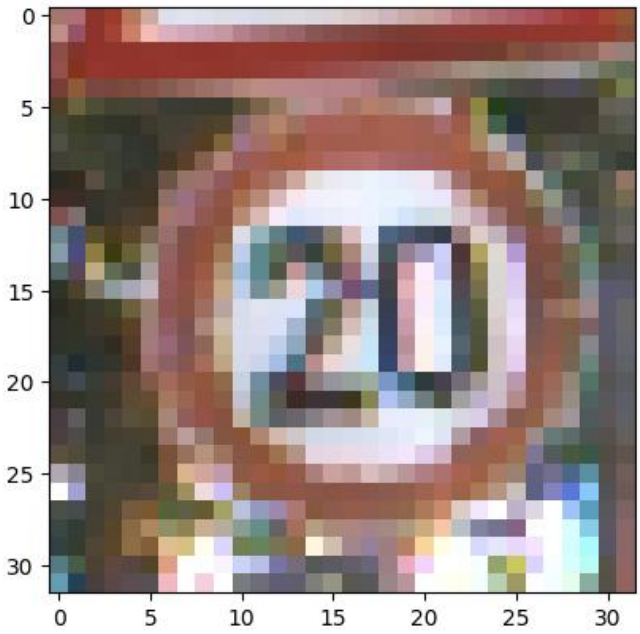
Проверил:
Спирин А.А.

Москва 2023

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.

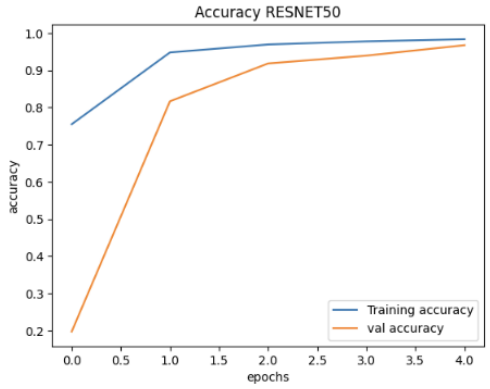


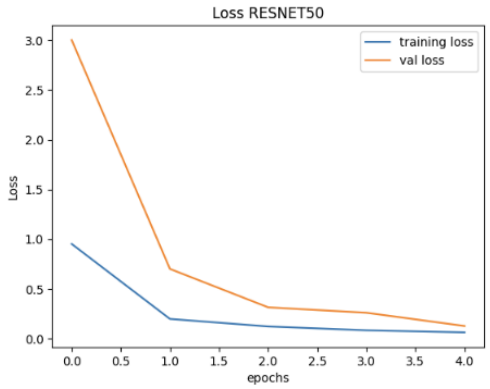
Первую модель построим на базе ResNet50. В результате эмпирического исследования, были выбраны оптимальные значения эпох обучения и размера пакета.

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

```
Epoch 1/5
429/429 [==============================] - 56s 62ms/step - loss: 0.9514 - accuracy: 0.7549 - val_loss: 3.0038 - val_accuracy: 0.1974
Epoch 2/5
429/429 [==============================] - 22s 50ms/step - loss: 0.1967 - accuracy: 0.9478 - val_loss: 0.7002 - val_accuracy: 0.8165
Epoch 3/5
429/429 [==============================] - 21s 50ms/step - loss: 0.1211 - accuracy: 0.9692 - val_loss: 0.3135 - val_accuracy: 0.9182
Epoch 4/5
429/429 [==============================] - 24s 55ms/step - loss: 0.0821 - accuracy: 0.9776 - val_loss: 0.2592 - val_accuracy: 0.9392
Epoch 5/5
429/429 [==============================] - 21s 50ms/step - loss: 0.0614 - accuracy: 0.9834 - val_loss: 0.1256 - val_accuracy: 0.9674
```

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





Протестируем модель на тестовом наборе.

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

```
395/395 [==============================] - 6s 13ms/step - loss: 0.4011 - accuracy: 0.9107
Test loss: 0.4011061191558838
Test accuracy: 0.9106888175010681
```

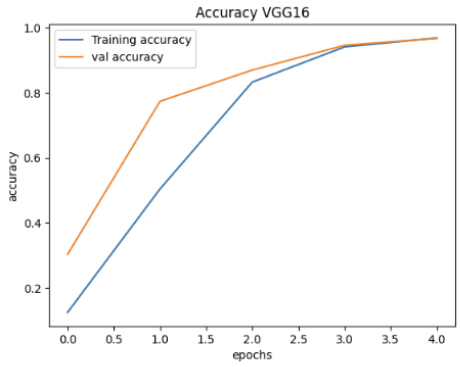
Итоговая точность составила 91%.

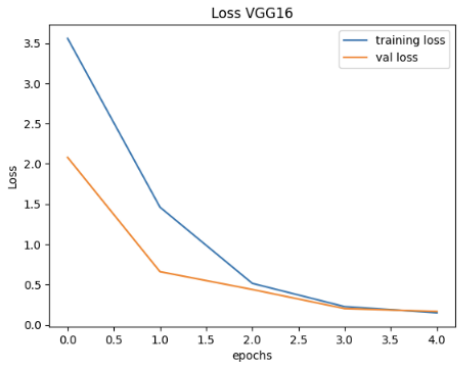
Обучим модель на базе VGG16.

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

```
Epoch 1/5
429/429 [==============================] - 27s 48ms/step - loss: 3.5569 - accuracy: 0.1256 - val_loss: 2.0787 - val_accuracy: 0.3038
Epoch 2/5
429/429 [==============================] - 17s 40ms/step - loss: 1.4608 - accuracy: 0.5043 - val_loss: 0.6597 - val_accuracy: 0.7735
Epoch 3/5
429/429 [==============================] - 18s 41ms/step - loss: 0.5144 - accuracy: 0.8327 - val_loss: 0.4384 - val_accuracy: 0.8696
Epoch 4/5
429/429 [==============================] - 18s 41ms/step - loss: 0.2256 - accuracy: 0.9407 - val_loss: 0.2013 - val_accuracy: 0.9457
Epoch 5/5
429/429 [==============================] - 17s 39ms/step - loss: 0.1490 - accuracy: 0.9682 - val_loss: 0.1655 - val_accuracy: 0.9671
```

Построим графики точности и потерь от эпох для модели VGG16.





Протестируем модель на тестовом наборе.

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

```
395/395 [==============================] - 5s 11ms/step - loss: 0.4461 - accuracy: 0.9225
Test loss: 0.44614627957344055
Test accuracy: 0.9224861264228821
```

Итоговая точность составила 92%.

Составим таблицу по заданию 1.

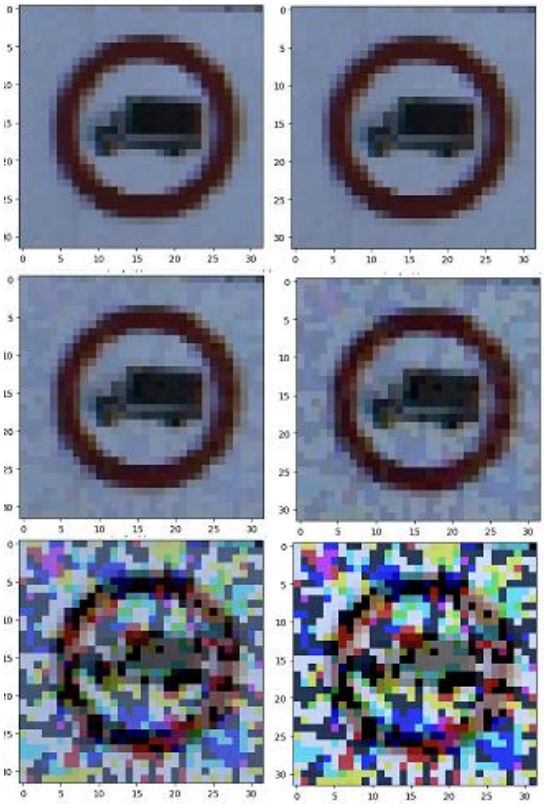
Модель	**Обучение**	**Валидация**	**Тест**
ResNet50	loss: 0.0614 accuracy: 0.9834	loss: 0.1256 accuracy: 0.9674	loss: 0.4011 accuracy: 0.9107
VGG16	loss: 0.1490 accuracy: 0.9682	loss: 0.1655 accuracy: 0.9671	loss: 0.4462 accuracy: 0.9225

Задание 2.

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

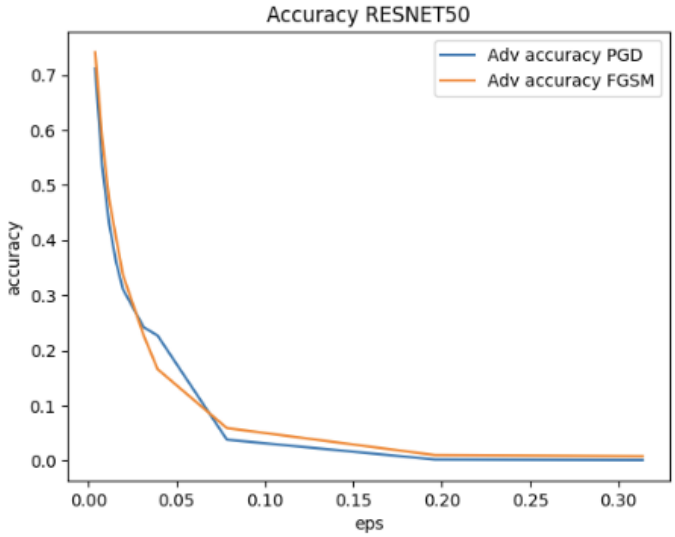
Создадим модель атаки, которая основывается на классификаторе для внесения шума в изображение.

Отобразим исходное и атакующие изображения для атаки FGSM.



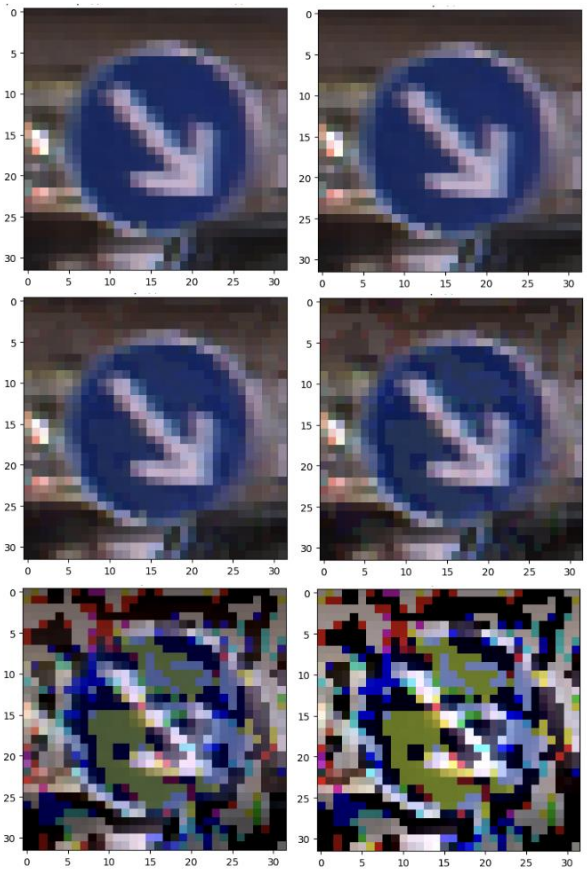
Построим график зависимости точности предсказания модели на атакованных изображениях от параметра искажения.

Из графика можно увидеть, что методы имеют схожую эфективность.

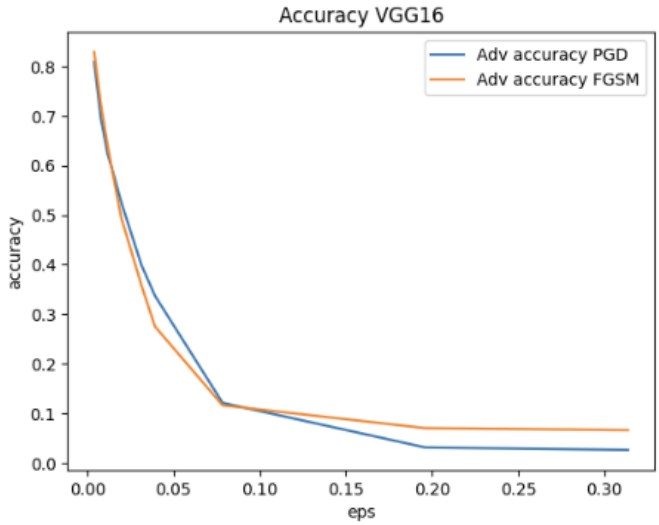


Повторим эксперимент с атаками FGSM и PGD на базе VGG16.

Для атаки FGSM отобразим исходное и атакующие изображения.



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



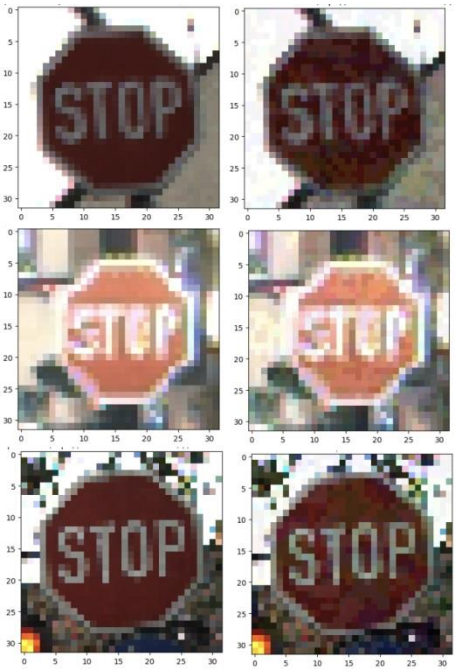
Составим таблицу по заданию 2.

Модель	**Исходные изображения**	**Adversarial images ϵ=1/255**	**Adversarial images ϵ=5/255**	**Adversarial images ϵ=10/255**
ResNet50 – FGSM	0,913	0,729	0,318	0,147
ResNet50 – PGT	0,913	0,712	0,311	0,227
VGG16 – FGSM	0,936	0,829	0,495	0,275
VGG16 – PGT	0,936	0,809	0,526	0,337

Задание 3.

Применим целевую атаку уклонения методом белого против моделей глубокого обучения.

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



Повторим атаку методом FGSM.

Составим таблицу по заданию 3.

Искажение	**PGD attack – Stop sign images**	**FGSM attack – Stop sign images**
ϵ=1/255	0,992	0,985
ϵ=3/255	0,941	0,822
ϵ=5/255	0,915	0,678
ϵ=10/255	0,881	0,281
ϵ=20/255	0,5	0,022
ϵ=50/255	0,041	0,0
ϵ=80/255	0,007	0,0

Метод FGSM для целевых атак не подходит, с ростом eps и шума, классификация ошибочна. Оптимальным значением искажения является 10/25, 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 20/255.