СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью практики является разработка метода оценки безопасности водителя с использование глубоких нейронных сетей.

Для этого требуется решить следующие задачи:

- выбрать средства программной реализации спроектированного в ходе выполнения выпускной квалификационной работы метода;
- реализовать программное обеспечение метода;
- провести исследование характеристик разработанного программного обеспечения.

1 Программная реализация метода безопасности водителя

1.1 Средства реализации программного комплекса

1.1.1 Выбор языка программирования

Для написания программного комплекса был выбран язык программирования Python [14], так как он обладает следующими критериями:

- большое количество библиотек для работы с нейронными сетями;
- возможность создавать графические интерфейсы;
- возможность тренировать модель нейронной сети на графическом процессоре.

1.1.2 Выбор библиотеки для работы с нейронной сетью

Для реализации и обучения модели нейронной сети была выбрана библиотека tensorflow [15], так как является наиболее часто используемой и требует меньшего времени для обучения модели по сравнению с второй по популярности библиотекой PyTorch [16].

1.2 Реализация программного комплекса

Модули метода оценки и загрузки данных представлены в листингах В.1 и Г.1 соответственно. Далее будут подробно рассмотрены модули пользовательского интерфейса и GoogleNet модели.

1.2.1 Модуль пользовательского интерфейса

Интерфейс пользовательского приложения при запуске представлен на рисунке 1.1.

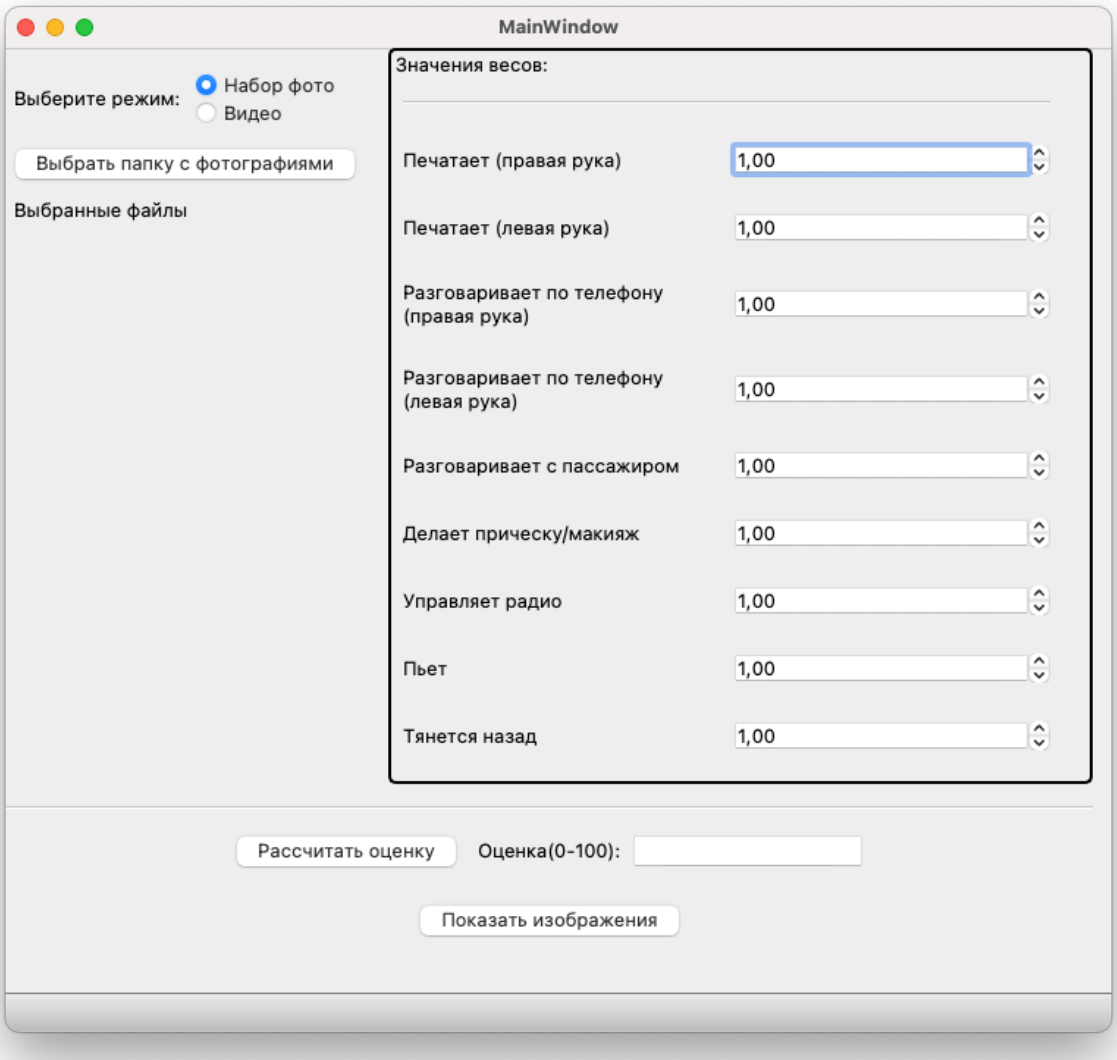

Рисунок 1.1 – Пользовательское приложение

1.2.2 Модуль GoogleNet модели

Реализация модели построена на классической GoogleNet модели с 9 $incep-tion$ блоками и 2 вспомогательными классификаторами. Вспомогательные классификаторы обеспечивают более эффективную и стабильную сходимость обучения.

В листинге А.1 приведена реализация модели GoogleNet, отвечающая за классификацию изображений.

У модели 3 выхода, 1 основной и 2 вспомогательных (засчет дополнительных классификаторов). Результат работы модели берется из основного выхода. Каждый выход представляет собой массив из 10 (количество классов) чисел (вероятностей). В данной работе класс с наибольшей вероятностью считается достоверным.

1.2.3 Обучение модели

Перед обучением модели необходимо произвести конвертацию исходных изображений к размеру 224 × 224 пикселей и формату RGB как этого требует модель.

В процессе обучения тренировочные данные подвергаются преобразованиям, таким как отражение, переворот, масштабирование. Это позволяет расширить набор тренировочных данных и повысить точность модели.

Обучение производилось на облачном NVIDIA DGX [17] сервере, с графическим процессором NVIDIA Tesla V100 имеющий 32 гигабайта видеопамяти.

В листинге А.2 приведена реализация процесса обучения реализованной модели GoogleNet. Пример классификации представлен на рисунках 1.2 — 1.3

Обучение производилось в течение 200 итераций (эпох), каждая эпоха состояла из 180 шагов. Набор данных для обучения был разделен на обучающий и валидационный в соотношении 4 к 1 соответственно. Валидационная часть набора нужна для того, чтобы можно оценить точность модели в конце каждой

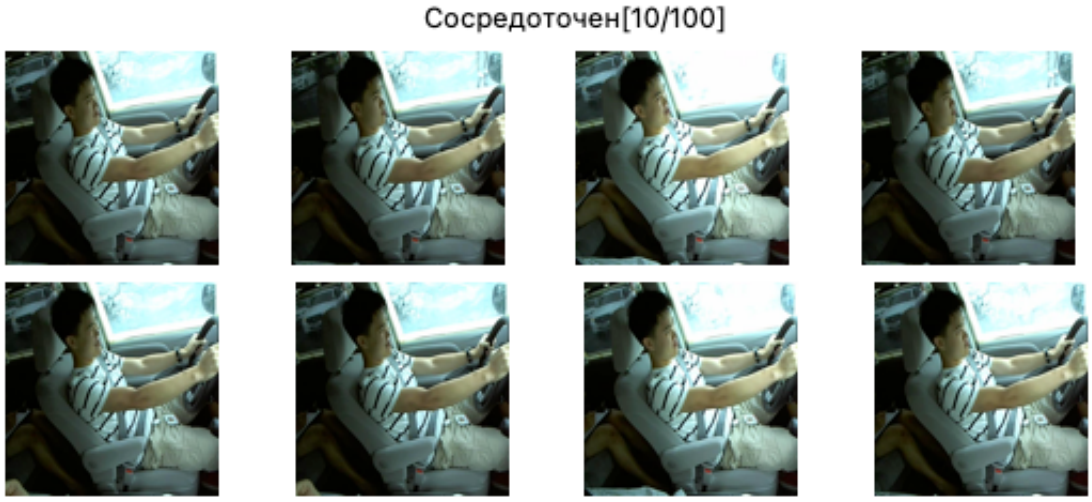

Рисунок 1.2 – Пример классификации, класс «Сосредоточен»

Рисунок 1.3 – Пример классификации, класс «Управляет радио»

эпохи на данных, неучавствующих в обучении.

1.2.4 Результаты обучения модели

На рисунках 1.4 и 1.5 представлены метрики точности на обучающих и валидационнах данных соответственно. Стоит отметить, что точность модели начала расти только после 30 эпох на обоих наборах данных и достигла 100 процентов на обучающих данных и 98 процентов на валидационных.

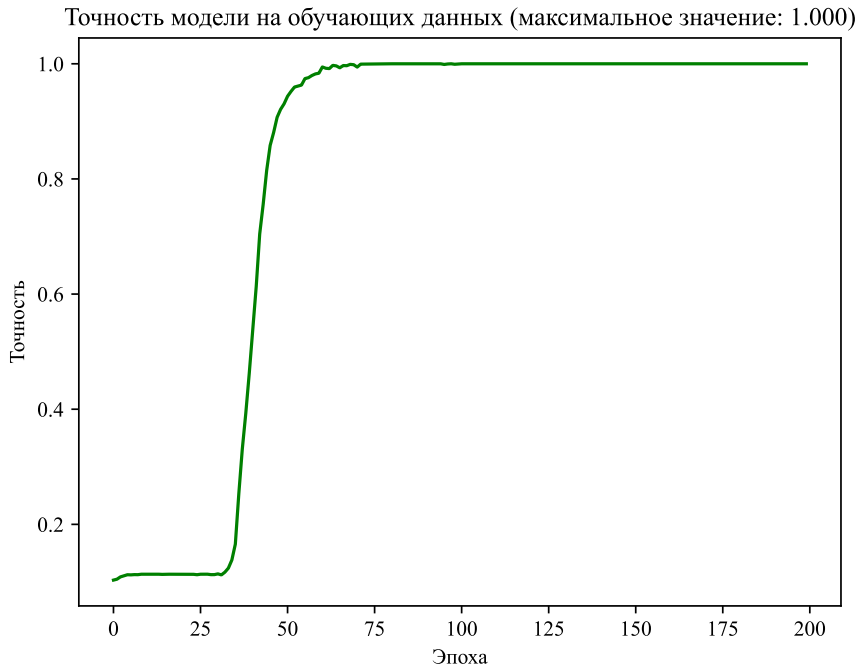

Рисунок 1.4 – Точность модели на обучающих данных

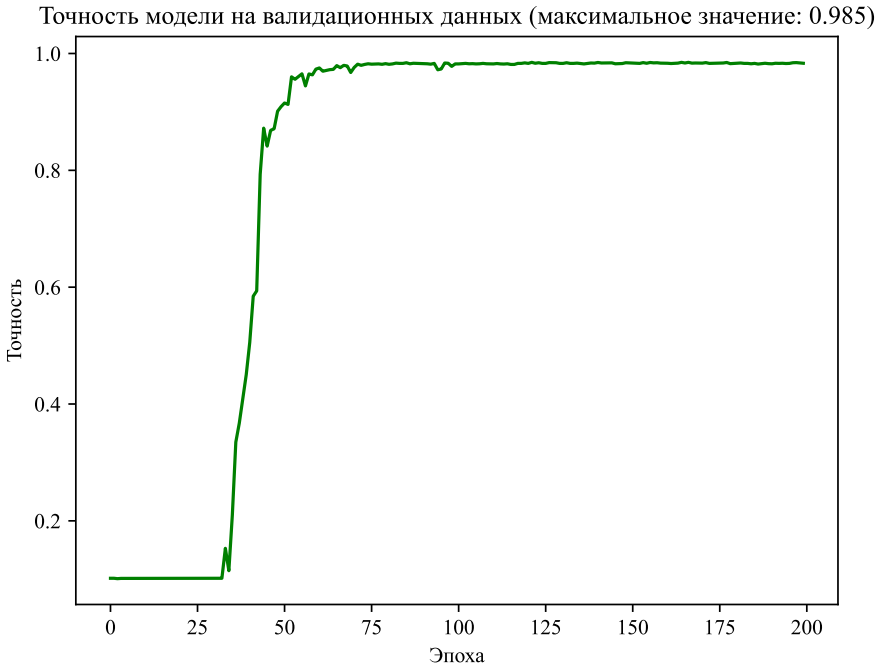

Рисунок 1.5 – Точность модели на валидационных данных

1.3 Примеры использования разработанного программного комплекса

Модуль модели GoogleNet представляет собой скрипт на языке программирования python, выполняющий обучение модели. Запуск данного скрипта осуществляется через консоль. Пример запуска модуля для обучения модели представлен в листинге 1. Различные параметры, такие как количество эпох, путь до папки с набором данных можно изменить непосредственно в программном коде, они представлены как константы.

Листинг 1: Запуск обучения модели

```
python main.py

2024-04-27 01:39:16.372411: I tensorflow/compiler/mlir/
    mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes
    are enabled (registered 2)
2024-04-27 01:39:16.375235: I tensorflow/core/platform/profile_utils/
    cpu_utils.cc:112] CPU Frequency: 2693670000 Hz
Epoch 1/200

Epoch 00001: LearningRateScheduler reducing learning rate to 0.01.
2024-04-27 01:39:19.291402: I tensorflow/stream_executor/platform/default/
    dso_loader.cc:49] Successfully opened dynamic library libcublas.so.11
2024-04-27 01:39:19.771270: I tensorflow/stream_executor/platform/default/
    dso_loader.cc:49] Successfully opened dynamic library libcublasLt.so.11
2024-04-27 01:39:22.171898: I tensorflow/stream_executor/platform/default/
    dso_loader.cc:49] Successfully opened dynamic library libcudnn.so.8

180/180 [==============================] - 59s 269ms/step - loss: 3.7943 -
    output_loss: 2.3949 - auxilliary_output_1_loss: 2.3347 -
    auxilliary_output_2_loss: 2.3298 - output_accuracy: 0.1004 -
    auxilliary_output_1_accuracy: 0.1069 - auxilliary_output_2_accuracy:
    0.1022 - val_loss: 3.6899 - val_output_loss: 2.3087 -
    val_auxilliary_output_1_loss: 2.3020 - val_auxilliary_output_2_loss:
    2.3021 - val_output_accuracy: 0.1017 - val_auxilliary_output_1_accuracy:
    0.1015 - val_auxilliary_output_2_accuracy: 0.1015
Epoch 2/200

```

```
15 ...
16
17 Epoch 00200: LearningRateScheduler reducing learning rate to
     0.0036039671685801802.
18 180/180 [==============================] - 21s 118ms/step - loss: 3.0684e-04
     - output_loss: 1.6573e-06 - auxilliary_output_1_loss: 9.4422e-04 -
     auxilliary_output_2_loss: 7.3059e-05 - output_accuracy: 1.0000 -
     auxilliary_output_1_accuracy: 0.9998 - auxilliary_output_2_accuracy:
     1.0000 - val_loss: 0.3523 - val_output_loss: 0.2355 -
     val_auxilliary_output_1_loss: 0.1905 - val_auxilliary_output_2_loss:
     0.1988 - val_output_accuracy: 0.9833 - val_auxilliary_output_1_accuracy:
     0.9802 - val_auxilliary_output_2_accuracy: 0.9857
```

Модуль пользовательского интерфейса представляет собой графический интерфейс, запуск которого также производится через терминал.

После выбора режима нужно выбрать либо папку с фотографиями либо видеофайл на диске, затем в приложении в интерфейсе приложения будет продублирован выбранный путь как на рисунке 1.6.

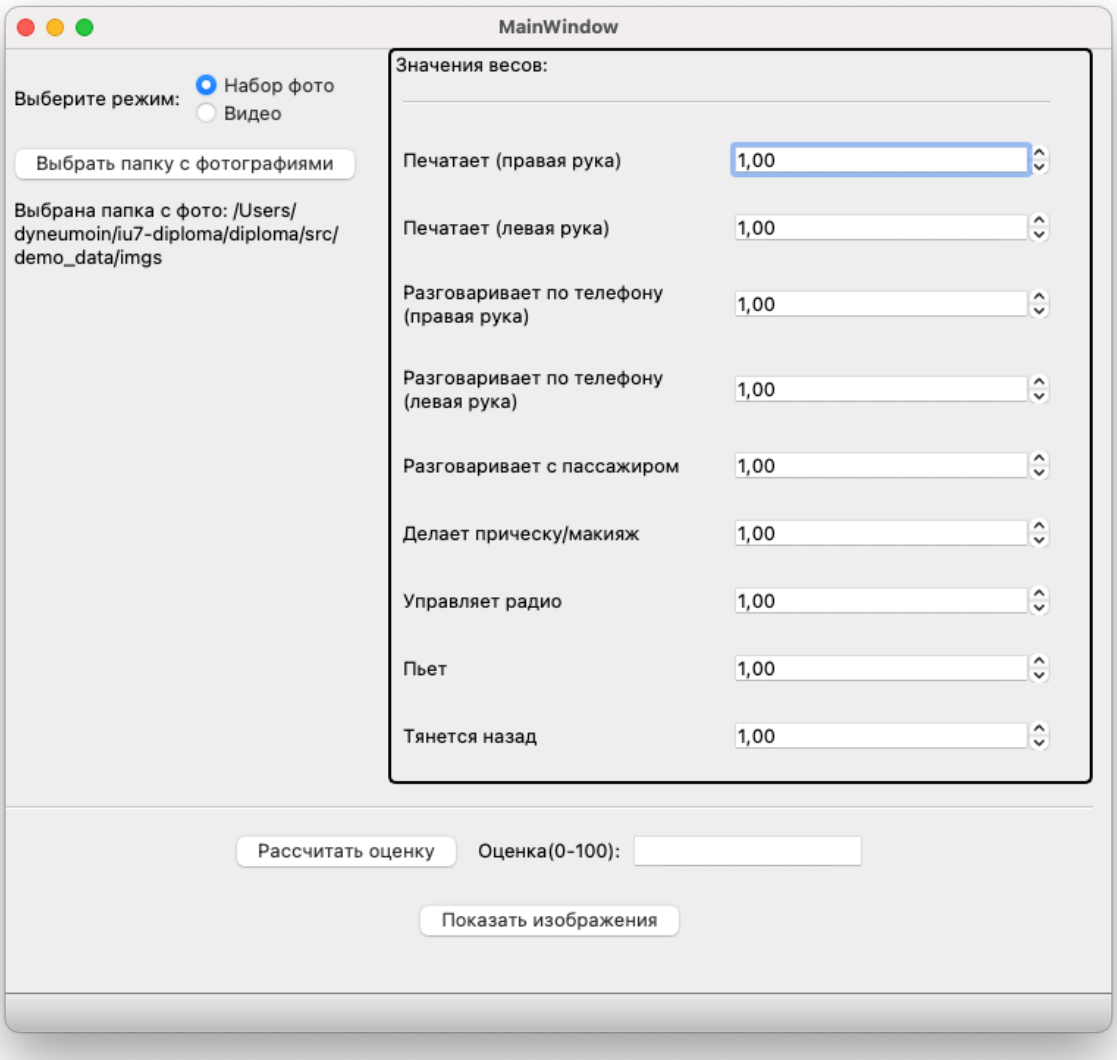

Рисунок 1.6 – Пользовательское приложение (выбран путь к изображениям)

После выбора пути к изображениям или видеофайлу можно рассчитать оценку, нажав на соответствующую кнопку «Рассчитать оценку», значение оценки отобразится в поле с заголовком «Оценка(0-100)» как на рисунке 1.7.

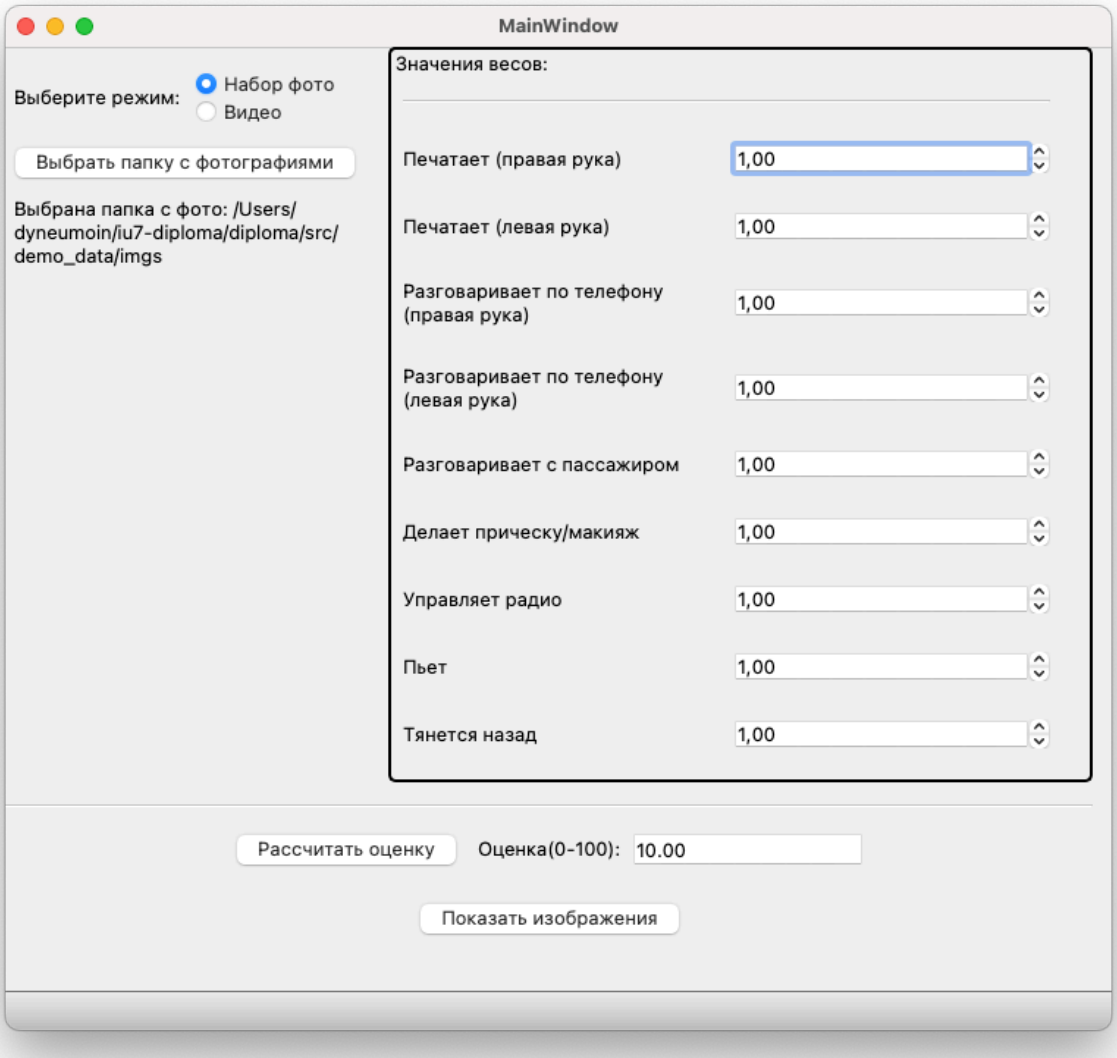

Рисунок 1.7 – Пользовательское приложение (рассчитана оценка)

Если требуется просмотреть изображения, разделенные по классам, то это можно сделать по нажатию кнопки «Показать изображения», откроется отдельное диалоговое окно с группами классов, в начале группы будет название класса и количество в формате [количество изображений класса / общее количество изображений] как на рисунке 1.8.

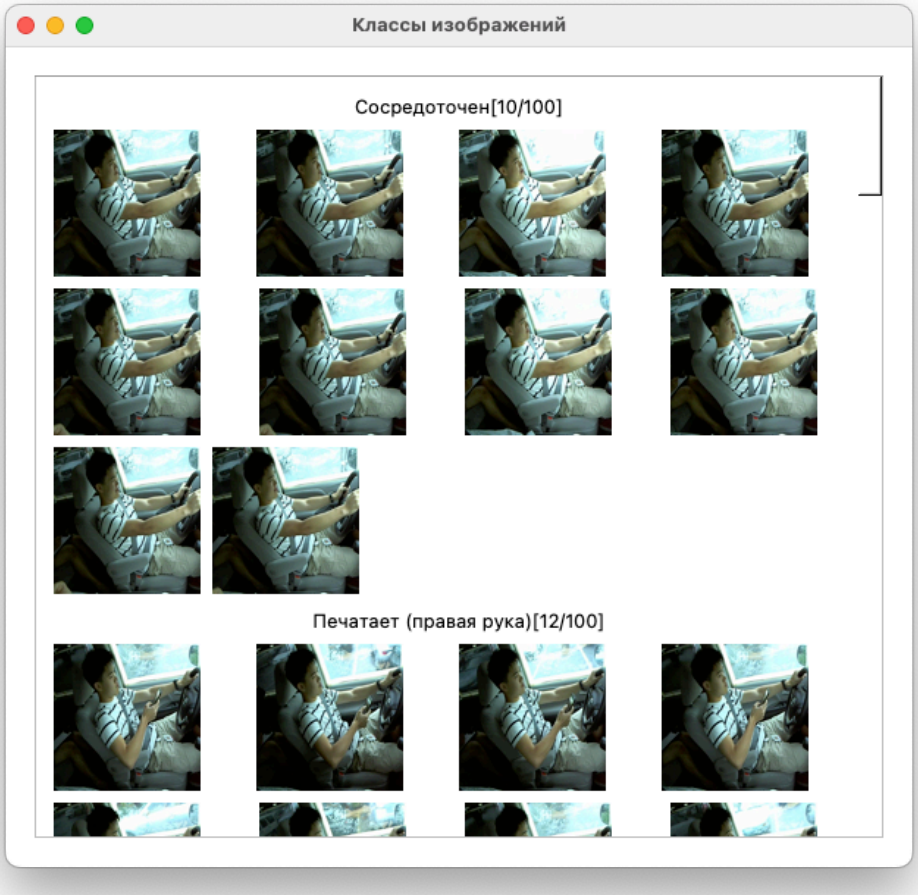

Рисунок 1.8 – Пользовательское приложение (диалоговое окно с изображениями)

Вывод

Были описаны средства реализации программного комплекса. Приведены листинги реализации каждого компонента комплекса, примеры работы компонентов, их входные и выходные данные. Описаны технологии и методы, использовавшиеся при реализации. Представлены примеры взаимодействия с модулями.

2 Исследование характеристик разработанного программного обеспечения

2.1 Предмет исследования

В данной работе будут рассмотрены следующие зависимости:

- зависимость времени работы метода от количества изображений;
- зависимость точности модели нейронной сети от размера изображений.

Время работы метода напрямую зависит от количества изображений, подаваемых на вход, так как основной операцией является классификация изображения. Точность модели зависит в основном от размера (качества) изображений и ракурса, так как в наборе данных нет изображений под другим ракурсом, то будет рассмотрена зависимость от качества изображений.

Технические характеристики

Технические характеристики устройства, на котором выполнялись сравнения, следующие.

- Операционная система: macOS 12.5.1.
- Память: 32 ГБ.
- Процессор: Apple M1 Pro CPU @ 3.22ГГц [18].

Исследование проводилось на ноутбуке, включенном в сеть электропитания. Во время экспериментов ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой исследования.

2.2 Сравнение времени работы реализованного метода на разных объемах входных данных

На рисунке 2.1 приведен график зависимости времени работы метода от количества изображений, все изображения имеют исходный размер 640x480.

Из приведенного графика можно сделать вывод, что зависимость времени работы метода от количества изображений практически линейна и что на вычисление оценки на 500 изображениям потребуется примерно 25 секунд. Так

как при обработке видео оно делится на изображения с интервалом в 1 секунду, то 500 изображений соответствуют примерно 8 минутам видео, следовательно на обработку 12 часового видео уйдет около 36 минут.

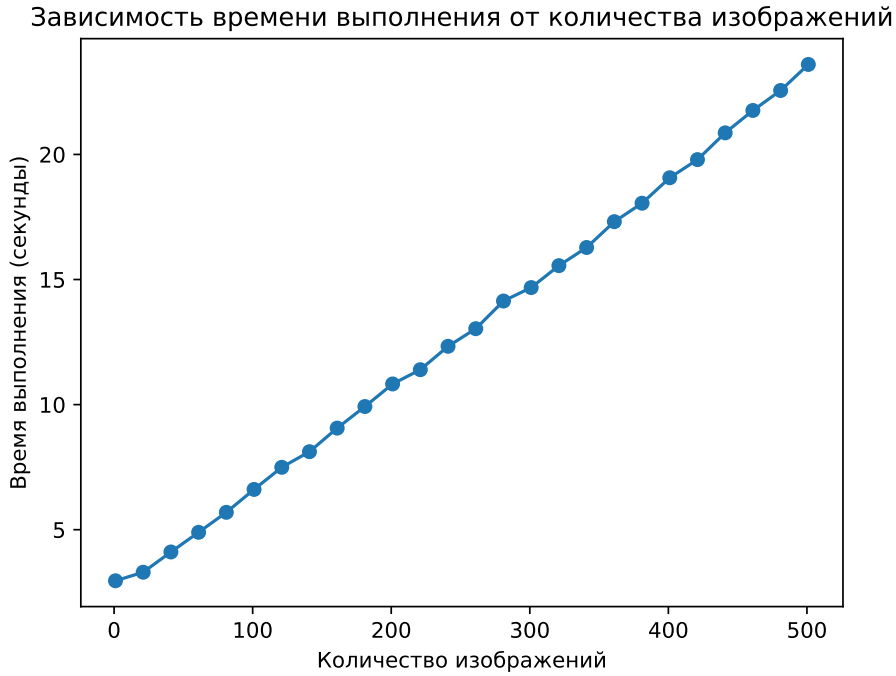

Рисунок 2.1 – Зависимость времени работы метода от количества изображений

2.3 Сравнение точности модели нейронной сети на изображениях разного размера

Так как модель нейронной сети требует, чтобы на вход подавались изображения размером 224x224, то есть все изображения большего размера сжимаются до этого размера. Следовательно имеет смысл рассмотреть изображения меньшего размера и сравнить точность.

Для сравнения были взяты 100 изображений из валидационного набора данных предварительно приведенных к нужным размерам.

На рисунке 2.2 приведен график зависимости точности модели от размера изображений.

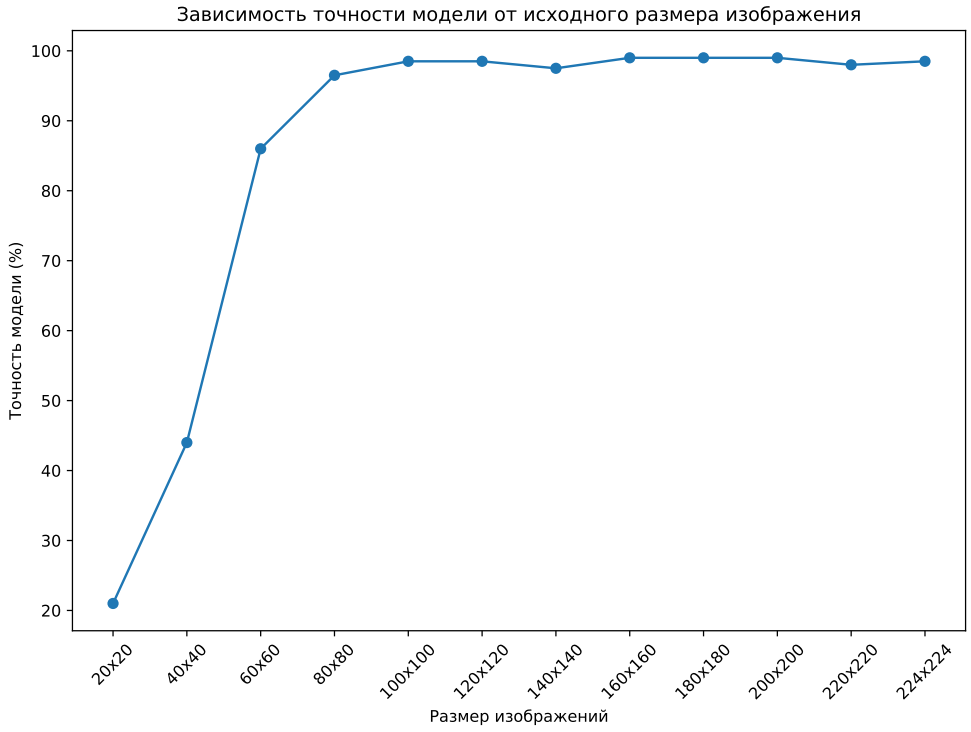

Рисунок 2.2 – Зависимость точности модели от размера изображений

Из приведенного графика можно сделать вывод, что на изображениях, размер которых меньше 80x80, точность значительно ниже (85% для размера 60x60, 45% для размера 40x40, 20% для размера 20x20), нежели на изображениях большего размера. Из этого следует, что для высокой точности модели и,

соответственно, корректной оценки, оптимальным размером изображений является 80x80 и больше.

2.4 Результаты исследования

На основе сравнений можно сделать следующие выводы:

- зависимость времени работы метода от количества изображений линейно;
- на обработку, например, 12 часового видео потребуется примерно 36 минут, такое время работы может являться проблемой, если требуется оценивать большое количество водителей;
- наиболее оптимальным размером изображений является 80x80 и выше, при размере 80x80 точность модели остается высокой (более 95%), что позволят хранить меньший объем данных на диске без потери точности.

ЗАКЛЮЧЕНИЕ

В результате выполнения практики была достигнута цель – разработан метод оценки безопасности водителя на основе глубоких нейронных сетей.

В ходе выполнения поставленной цели были выполнены следующие задачи:

- выбраны средства программной реализации спроектированного в ходе выполнения выпускной квалификационной работы метода;
- реализовано программное обеспечение метода;
- проведено исследование характеристик разработанного программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ВЦИОМ: Такси в России [Электронный ресурс]. Режим доступа: https://wciom.ru/analytical-reviews/analiticheskii-obzor/taksi-v-rossii-mnenie-polzovatelei (дата обращения 15.10.2023).

2. Аналитический центр при правительстве РФ [Электронный ресурс]. Режим доступа: https://ac.gov.ru/archive/files/content/19540/taksi-avariinost-final-2311-pdf.pdf (дата обращения 15.10.2023).

3. Learning and transferring mid-level image representations using convolutional neural networks / Maxime Oquab, Leon Bottou, Ivan Laptev [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2014. С. 1717–1724.

4. LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning // nature. 2015. Т. 521, № 7553. С. 436–444.

5. Bai Yuhan. RELU-function and derived function review // SHS Web of Conferences / EDP Sciences. Т. 144. 2022. С. 02006.

6. Recent advances in convolutional neural networks / Jiuxiang Gu, Zhenhua Wang, Jason Kuen [и др.] // Pattern recognition. 2018. Т. 77. С. 354–377.

7. Review of image classification algorithms based on convolutional neural networks / Leiyu Chen, Shaobo Li, Qiang Bai [и др.] // Remote Sensing. 2021. Т. 13, № 22. С. 4712.

8. A survey of deep neural network architectures and their applications / Weibo Liu, Zidong Wang, Xiaohui Liu [и др.] // Neurocomputing. 2017. Т. 234. С. 11–26.

9. Boureau Y-Lan, Ponce Jean, LeCun Yann. A theoretical analysis of feature pooling in visual recognition // Proceedings of the 27th international conference on machine learning (ICML-10). 2010. С. 111–118.

10. A comparative study of different CNN models and transfer learning effect for underwater object classification in side-scan sonar images / Xing Du, Yongfu Sun, Yupeng Song [и др.] // Remote Sensing. 2023. Т. 15, № 3. С. 593.

11. Going deeper with convolutions / Christian Szegedy, Wei Liu, Yangqing Jia [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. С. 1–9.

12. Philipp George, Song Dawn, Carbonell Jaime G. The exploding gradient problem demystified-definition, prevalence, impact, origin, tradeoffs, and solutions // arXiv preprint arXiv:1712.05577. 2017.

13. State Farm Distracted Driver Detection [Электронный ресурс]. Режим доступа: https://kaggle.com/competitions/state-farm-distracted-driver-detection (дата обращения 01.04.2024).

14. Python [Электронный ресурс]. Режим доступа: https://www.python.org/ (дата обращения 01.04.2024).

15. Tensorflow [Электронный ресурс]. Режим доступа: https://www.tensorflow.org/ (дата обращения 01.04.2024).

16. Performance analysis of deep learning libraries: TensorFlow and PyTorch / Felipe Florencio, Thiago Valen, Edward David Moreno [и др.] // Journal of Computer Science. 2019. Т. 15, № 6. С. 785–799.

17. Tensorflow [Электронный ресурс]. Режим доступа: https://www.

nvidia.com/ru-ru/data-center/dgx-systems/ (дата обращения 01.04.2024).

18. Apple M1 Pro [Электронный ресурс]. Режим доступа: https://apple.fandom.com/wiki/Apple_M1_Pro (дата обращения 01.04.2024).

ПРИЛОЖЕНИЕ А

Модуль модели GoogleNet

Листинг А.1: Модель GoogleNet

```
import keras
from keras.models import Model
from keras.layers import Conv2D, MaxPool2D,Dropout, Dense, Input, concatenate
    ,GlobalAveragePooling2D, AveragePooling2D,Flatten

kernel_init = keras.initializers.glorot_uniform()
bias_init = keras.initializers.Constant(value=0.2)

def inception_module(x,
                     filters_1x1,
                     filters_3x3_reduce,
                     filters_3x3,
                     filters_5x5_reduce,
                     filters_5x5,
                     filters_pool_proj,
                     name=None):

    conv_1x1 = Conv2D(filters_1x1, (1, 1), padding='same', activation='relu',
     kernel_initializer=kernel_init, bias_initializer=bias_init)(x)

    conv_3x3 = Conv2D(filters_3x3_reduce, (1, 1), padding='same', activation=
    'relu', kernel_initializer=kernel_init, bias_initializer=bias_init)(x)
    conv_3x3 = Conv2D(filters_3x3, (3, 3), padding='same', activation='relu',
     kernel_initializer=kernel_init, bias_initializer=bias_init)(conv_3x3)

    conv_5x5 = Conv2D(filters_5x5_reduce, (1, 1), padding='same', activation=
    'relu', kernel_initializer=kernel_init, bias_initializer=bias_init)(x)
    conv_5x5 = Conv2D(filters_5x5, (5, 5), padding='same', activation='relu',
     kernel_initializer=kernel_init, bias_initializer=bias_init)(conv_5x5)

    pool_proj = MaxPool2D((3, 3), strides=(1, 1), padding='same')(x)
    pool_proj = Conv2D(filters_pool_proj, (1, 1), padding='same', activation=
    'relu', kernel_initializer=kernel_init, bias_initializer=bias_init)(
    pool_proj)
```

```
27
28      output = concatenate([conv_1x1, conv_3x3, conv_5x5, pool_proj], axis=3,
        name=name)
29
30      return output
31
32  def GoogleNet():
33      input_layer = Input(shape=(224, 224, 3))
34
35      x = Conv2D(64, (7, 7), padding='same', strides=(2, 2), activation='relu',
         name='conv_1_7x7_2', kernel_initializer=kernel_init, bias_initializer=
        bias_init)(input_layer)
36      x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
        max_pool_1_3x3_2')(x)
37      x = Conv2D(64, (1, 1), padding='same', strides=(1, 1), activation='relu',
         name='conv_2a_3x3_1')(x)
38      x = Conv2D(192, (3, 3), padding='same', strides=(1, 1), activation='relu'
        , name='conv_2b_3x3_1')(x)
39      x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
        max_pool_2_3x3_2')(x)
40
41      x = inception_module(x,
42                           filters_1x1=64,
43                           filters_3x3_reduce=96,
44                           filters_3x3=128,
45                           filters_5x5_reduce=16,
46                           filters_5x5=32,
47                           filters_pool_proj=32,
48                           name='inception_3a')
49
50      x = inception_module(x,
51                           filters_1x1=128,
52                           filters_3x3_reduce=128,
53                           filters_3x3=192,
54                           filters_5x5_reduce=32,
55                           filters_5x5=96,
56                           filters_pool_proj=64,
57                           name='inception_3b')
58
```

```
    x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
    max_pool_3_3x3_2')(x)

    x = inception_module(x,
                         filters_1x1=192,
                         filters_3x3_reduce=96,
                         filters_3x3=208,
                         filters_5x5_reduce=16,
                         filters_5x5=48,
                         filters_pool_proj=64,
                         name='inception_4a')


    x1 = AveragePooling2D((5, 5), strides=3)(x)
    x1 = Conv2D(128, (1, 1), padding='same', activation='relu')(x1)
    x1 = Flatten()(x1)
    x1 = Dense(1024, activation='relu')(x1)
    x1 = Dropout(0.7)(x1)
    x1 = Dense(10, activation='softmax', name='auxilliary_output_1')(x1)

    x = inception_module(x,
                         filters_1x1=160,
                         filters_3x3_reduce=112,
                         filters_3x3=224,
                         filters_5x5_reduce=24,
                         filters_5x5=64,
                         filters_pool_proj=64,
                         name='inception_4b')

    x = inception_module(x,
                         filters_1x1=128,
                         filters_3x3_reduce=128,
                         filters_3x3=256,
                         filters_5x5_reduce=24,
                         filters_5x5=64,
                         filters_pool_proj=64,
                         name='inception_4c')

    x = inception_module(x,
                         filters_1x1=112,
```

```
                        filters_3x3_reduce=144,
                        filters_3x3=288,
                        filters_5x5_reduce=32,
                        filters_5x5=64,
                        filters_pool_proj=64,
                        name='inception_4d')


    x2 = AveragePooling2D((5, 5), strides=3)(x)
    x2 = Conv2D(128, (1, 1), padding='same', activation='relu')(x2)
    x2 = Flatten()(x2)
    x2 = Dense(1024, activation='relu')(x2)
    x2 = Dropout(0.7)(x2)
    x2 = Dense(10, activation='softmax', name='auxilliary_output_2')(x2)

    x = inception_module(x,
                        filters_1x1=256,
                        filters_3x3_reduce=160,
                        filters_3x3=320,
                        filters_5x5_reduce=32,
                        filters_5x5=128,
                        filters_pool_proj=128,
                        name='inception_4e')

    x = MaxPool2D((3, 3), padding='same', strides=(2, 2), name='
   max_pool_4_3x3_2')(x)

    x = inception_module(x,
                        filters_1x1=256,
                        filters_3x3_reduce=160,
                        filters_3x3=320,
                        filters_5x5_reduce=32,
                        filters_5x5=128,
                        filters_pool_proj=128,
                        name='inception_5a')

    x = inception_module(x,
                        filters_1x1=384,
                        filters_3x3_reduce=192,
                        filters_3x3=384,
```

```
                        filters_5x5_reduce=48,
                        filters_5x5=128,
                        filters_pool_proj=128,
                        name='inception_5b')

    x = GlobalAveragePooling2D(name='avg_pool_5_3x3_1')(x)

    x = Dropout(0.4)(x)

    x = Dense(10, activation='softmax', name='output')(x)

    model = Model(input_layer, [x, x1, x2], name='inception_v1')
    return model
```

Листинг А.2: Обучение модели

```
import tensorflow as tf
import keras
from keras.optimizers import SGD
from keras.callbacks import LearningRateScheduler
import math
from model import GoogleNet

dataset_dir = "./data/train"
epochs = 200
batch_size = 100

def get_generators(base_dir_train, target_size=(224, 224)):
    train_ds = tf.keras.preprocessing.image_dataset_from_directory(
        base_dir_train,
        validation_split=0.2,
        subset="training",
        seed=123,
        label_mode='int',
        image_size=target_size,
        batch_size=batch_size
    )

    validation_ds = tf.keras.preprocessing.image_dataset_from_directory(
        base_dir_train,
        validation_split=0.2,
```

```
        subset="validation",
        label_mode='int',
        seed=123,
        image_size=target_size,
        batch_size=batch_size
    )


    class_names = train_ds.class_names
    print(class_names)
    print("before aug: ", train_ds.cardinality().numpy() * batch_size)

    data_augmentation = keras.Sequential(
        [
            tf.keras.layers.experimental.preprocessing.RandomFlip("horizontal
"),
            tf.keras.layers.experimental.preprocessing.RandomRotation(0.1),
            tf.keras.layers.experimental.preprocessing.RandomZoom(0.1),
        ]
    )

    train_ds = train_ds.map(lambda x, y: (data_augmentation(x), y),
num_parallel_calls=tf.data.AUTOTUNE)

    normalization_layer = tf.keras.layers.experimental.preprocessing.
Rescaling(1./255)
    normalized_train_ds = train_ds.map(lambda x, y: (normalization_layer(x),
y))
    normalized_validation_ds = validation_ds.map(lambda x, y: (
normalization_layer(x), y))

    AUTOTUNE = tf.data.AUTOTUNE

    normalized_train_ds = normalized_train_ds.cache().prefetch(buffer_size=
AUTOTUNE)
    normalized_validation_ds = normalized_validation_ds.cache().prefetch(
buffer_size=AUTOTUNE)

    return normalized_train_ds.take(3000), normalized_validation_ds.take(500)

```

```
num_classes = 10
img_rows,img_cols = 224, 224

train_ds, test_ds = get_generators(dataset_dir, (img_rows,img_cols))

initial_lrate = 0.01

def decay(epoch, steps=100):
    initial_lrate = 0.01
    drop = 0.96
    epochs_drop = 8
    lrate = initial_lrate * math.pow(drop, math.floor((1+epoch)/epochs_drop))
    return lrate


sgd = SGD(learning_rate=initial_lrate, momentum=0.9, nesterov=False)

lr_sc = LearningRateScheduler(decay, verbose=1)

model = GoogleNet()
model.compile(loss=keras.losses.sparse_categorical_crossentropy, loss_weights
    =[1, 0.3, 0.3], optimizer=sgd, metrics=['accuracy'])

# Moddel Training
train_ds = train_ds.map(lambda x, y: (x, (y, y, y)))
test_ds = test_ds.map(lambda x, y: (x, (y, y, y)))


history = model.fit(
    train_ds,
    validation_data=test_ds,
    epochs=epochs,
    batch_size=batch_size,
    callbacks=[lr_sc]
)

model.save('model.keras')
```

ПРИЛОЖЕНИЕ Б

Модуль пользовательского интерфейса

Листинг Б.1: Интерфейс главного окна

```
from PyQt5 import uic
from PyQt5.QtWidgets import (QApplication, QMainWindow, QPushButton, QLabel,
    QFileDialog, QDoubleSpinBox, QLineEdit, QRadioButton)
import sys
import app.main
import images_window

class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = uic.loadUi("./ui/untitled.ui", self)

        # Определим виджеты.
        self.photo_button = self.findChild(QPushButton, 'choice_photo')
        self.video_button = self.findChild(QPushButton, 'choice_video')
        self.mode_photo_radioButton = self.findChild(QRadioButton, '
    mode_photo_radioButton')
        self.mode_video_radioButton = self.findChild(QRadioButton, '
    mode_video_radioButton')
        self.calc_button = self.findChild(QPushButton, 'calc')
        self.files_label = self.findChild(QLabel, 'choiced_files_label')
        self.assessment = self.findChild(QLineEdit, 'assessment_line')
        self.show_images_button = self.findChild(QPushButton, '
    show_images_button')

        # Beca
        self.texting_right_spin = self.findChild(QDoubleSpinBox, '
    texting_right_spin')
        self.talking_right_spin = self.findChild(QDoubleSpinBox, '
    talking_right_spin')
        self.texting_left_spin = self.findChild(QDoubleSpinBox, '
    texting_left_spin')
        self.talking_left_spin = self.findChild(QDoubleSpinBox, '
    talking_left_spin')
```

```
27          self.operating_radio_spin = self.findChild(QDoubleSpinBox, '
    operation_radio_spin')
28          self.drinking_spin = self.findChild(QDoubleSpinBox, 'drink_spin')
29          self.reaching_begind_spin = self.findChild(QDoubleSpinBox, '
    behind_spin')
30          self.hair_and_makeup_spin = self.findChild(QDoubleSpinBox, '
    makeup_spin')
31          self.talking_to_passenger = self.findChild(QDoubleSpinBox, '
    talking_passanger_spin')
32
33
34          # Обработчики
35          self.photo_button.clicked.connect(self.photo_button_clicked)
36          self.video_button.clicked.connect(self.video_button_clicked)
37          self.mode_photo_radioButton.toggled.connect(self.on_radio_toggled)
38          self.mode_video_radioButton.toggled.connect(self.on_radio_toggled)
39          self.on_radio_toggled()
40          self.calc_button.clicked.connect(self.calc_button_clicked)
41          self.show_images_button.clicked.connect(self.
    show_images_button_clicked)
42
43          # Контекст.
44          self.mode = None
45          self.file_path = ''
46          self.classes_images = []
47
48      def photo_button_clicked(self):
49          folder_dialog = QFileDialog()
50          folder_path = folder_dialog.getExistingDirectory(self, 'Выбрать
    папку с фото')
51
52          if folder_path:
53              self.files_label.setText("Выбрана папка с фото: " + folder_path)
54              self.mode = app.main.DriverMode.PHOTO
55              self.file_path = folder_path
56
57      def video_button_clicked(self):
58          file_dialog = QFileDialog()
59          file_path, _ = file_dialog.getOpenFileName(self, 'Выбрать видео', '',
     'Video files (*.mp4 *.avi)')
```

```
60
61          if file_path:
62              self.files_label.setText("Выбрано видео: " + file_path)
63              self.mode = app.main.DriverMode.VIDEO
64              self.file_path = file_path
65
66      def calc_button_clicked(self):
67          classes_weights = self.__get_classes_weights()
68          print(f'calc: file:{self.file_path}, weights:{classes_weights}')
69          cfg = app.main.DriverSafetyGetterConfig(classes_weights, self.mode,
    self.file_path)
70          getter = app.main.DriverSafetyAssessmentGetter(cfg)
71          assessment = getter.get_driver_safety_assessment()
72          print(f'assesment: {assessment}')
73          self.assessment.setText(f"{assessment:.2f}")
74          self.classes_images = getter.get_classes_images()
75
76      def on_radio_toggled(self):
77          if self.mode_photo_radioButton.isChecked():
78              self.photo_button.setVisible(True)
79              self.video_button.setVisible(False)
80          elif self.mode_video_radioButton.isChecked():
81              self.photo_button.setVisible(False)
82              self.video_button.setVisible(True)
83
84      def __get_classes_weights(self):
85          classes_weights= {
86              'texting - right': self.texting_right_spin.value(),
87              'talking on the phone - right': self.talking_right_spin.value(),
88              'texting - left': self.texting_left_spin.value(),
89              'talking on the phone - left': self.talking_left_spin.value(),
90              'operating the radio': self.operating_radio_spin.value(),
91              'drinking': self.drinking_spin.value(),
92              'reaching behind': self.reaching_begind_spin.value(),
93              'hair and makeup': self.hair_and_makeup_spin.value(),
94              'talking to passenger': self.talking_to_passenger.value(),
95          }
96
97          return classes_weights
98
```

```
99      def show_images_button_clicked(self):
100         self.img_window = images_window.GalleryWindow(self.classes_images)
101         self.img_window.show()
102
103 def main():
104     app = QApplication(sys.argv)
105     window = MainWindow()
106     window.show()
107     return app.exec()
108
109
110 if __name__ == '__main__':
111     sys.exit(main())
```

Листинг Б.2: Интерфейс диалогового окна с изображениями

```
1  import sys
2  from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QVBoxLayout,
       QLabel, QScrollArea, QHBoxLayout, QMainWindow, QSpacerItem, QSizePolicy
3  from PyQt5.QtGui import QPixmap, QImage
4  from PyQt5.QtCore import Qt
5  from PIL import Image
6  from io import BytesIO
7
8  classes_en_ru_mapping = {
9       'safe driving': 'осредоточенС',
10      'texting - right': 'Печатает правая( рука)',
11      'talking on the phone - right': 'Разговаривает по телефону правая( рука)
       ',
12      'texting - left': 'Печатает левая( рука)',
13      'talking on the phone - left': 'Разговаривает по телефону левая( рука)',
14      'operating the radio': 'Управляет радио',
15      'drinking': 'Пьет',
16      'reaching behind': 'Тянется назад',
17      'hair and makeup': 'Делает прическумакияж/',
18      'talking to passenger': 'Разговаривает с пассажиром',
19 }
20
21
22 class GalleryWindow(QWidget):
23     def __init__(self, images):
```

```
24          super().__init__()
25          self.images = images  # Словарь с изображениями
26          self.total_elements = sum(len(images) for images in self.images.
    values())
27          self.initUI()
28
29      def initUI(self):
30          self.setStyleSheet("background-color: white;")
31          layout = QVBoxLayout()
32
33          scroll = QScrollArea()  # Создание области прокрутки
34          widget = QWidget()  # Главный виджет, содержащий все остальные
    виджеты
35          scroll.setWidget(widget)  # Установка виджета в качестве содержимого
    области прокрутки
36          scroll.setWidgetResizable(True)  # Разрешение изменения размера
    виджета
37
38          vbox = QVBoxLayout()  # Вертикальное расположение для всех элементов
39
40          for class_name, imgs in self.images.items():  # Перебор всех классов
    изображений
41              label = QLabel(f'{classes_en_ru_mapping[class_name]}[{len(imgs)
    }/{self.total_elements}]')  # Создание метки для названия класса
42              label.setAlignment(Qt.AlignCenter)  # Центрирование текста
43              vbox.addWidget(label)  # Добавление метки в вертикальное
    расположение
44
45              imgs_in_row = 4  # Количество изображений в одной строке
46              row_counter = 0  # Счетчик для отслеживания количества
    изображений в текущей строке
47              hbox = QHBoxLayout()  # Горизонтальное расположение для
    изображений
48              hbox.setSpacing(0)  # Устанавливаем межэлементное расстояние
    равным 0
49              hbox.setContentsMargins(0, 0, 0, 0)  # Убираем отступы
50
51              for img in imgs:  # Перебор изображений текущего класса
52                  if row_counter >= imgs_in_row:  # Если достигнут предел
    изображений в строке
```

```
53                  vbox.addLayout(hbox)  # Добавляем текущее горизонтальное
    расположение в вертикальное
54                  hbox = QHBoxLayout()  # Создаем новое горизонтальное
    расположение для следующей строки
55                  row_counter = 0  # Сброс счетчика изображений в строке
56
57              pixmap = QPixmap.fromImage(self.convert_pil_image_to_qimage(
    img))  # Создание изображения
58              img_label = QLabel(self)  # Создание метки для изображения
59              img_label.setPixmap(pixmap)  # Установка изображения в метку
60              hbox.addWidget(img_label)  # Добавление метки в
    горизонтальное расположение
61              row_counter += 1  # Увеличение счетчика изображений в строке
62
63          if row_counter > 0 and row_counter < 4:
64              spacer = QSpacerItem(100, 100, QSizePolicy.Expanding,
    QSizePolicy.Minimum)
65              hbox.addSpacerItem(spacer)
66          if row_counter > 0:  # Проверка, есть ли необработанные
    изображения после выхода из цикла
67              vbox.addLayout(hbox)  # Добавление последнего
    горизонтального расположения в вертикальное
68
69      widget.setLayout(vbox)  # Установка вертикального расположения в
    качестве основного для виджета
70      layout.addWidget(scroll)  # Добавление области прокрутки в основное
    вертикальное расположение
71      self.setLayout(layout)  # Установка основного вертикального
    расположения для self
72      self.setWindowTitle('Классы изображений')  # Задание заголовка окна
73      self.resize(600, 400)  # Установка начального размера окна
74
75  def convert_pil_image_to_qimage(self, pil_image):
76      pil_image = pil_image.resize((100, 100))
77      image_data = BytesIO()
78      pil_image.save(image_data, format='PNG')
79      qimage = QImage()
80      qimage.loadFromData(image_data.getvalue())
81      return qimage
```

ПРИЛОЖЕНИЕ В

Модуль метода оценки безопасности водителя

Листинг В.1: Метод оценки безопасности водителя

```
from enum import Enum

from . import get_images
import tensorflow as tf
import numpy as np

import matplotlib.pyplot as plt

classes = {
    'c0': 'safe driving',
    'c1': 'texting - right',
    'c2': 'talking on the phone - right',
    'c3': 'texting - left',
    'c4': 'talking on the phone - left',
    'c5': 'operating the radio',
    'c6': 'drinking',
    'c7': 'reaching behind',
    'c8': 'hair and makeup',
    'c9': 'talking to passenger'
}

default_classes_weights= {
    'texting - right': 1,
    'talking on the phone - right': 1,
    'texting - left': 1,
    'talking on the phone - left': 1,
    'operating the radio': 1,
    'drinking': 1,
    'reaching behind': 1,
    'hair and makeup': 1,
    'talking to passenger': 1,
}

class DriverMode(Enum):
```

```
    VIDEO = "video"
    PHOTO = "photo"

class DriverSafetyGetterConfig:
    def __init__(self, classes_weights, mode: DriverMode, path_to_files, interval_sec=1):
        self.classes_weights = classes_weights
        self.mode = mode
        self.path_to_files = path_to_files
        self.interval_sec = interval_sec if mode == DriverMode.VIDEO else None

class DriverSafetyAssessmentGetter:
    def __init__(self, cfg: DriverSafetyGetterConfig):
        self.cfg = cfg
        self.model = tf.keras.models.load_model('./working/aug.keras')

        self.images = []
        self.input_datas = []
        self.classes_count= {
            'safe driving': 0,
            'texting - right': 0,
            'talking on the phone - right': 0,
            'texting - left': 0,
            'talking on the phone - left': 0,
            'operating the radio': 0,
            'drinking': 0,
            'reaching behind': 0,
            'hair and makeup': 0,
            'talking to passenger': 0,
        }
        self.classes_images = {
            'safe driving': [],
            'texting - right': [],
            'talking on the phone - right': [],
            'texting - left': [],
            'talking on the phone - left': [],
            'operating the radio': [],
            'drinking': [],
            'reaching behind': [],
```

```
73              'hair and makeup': [],
74              'talking to passenger': [],
75          }
76
77      def get_driver_safety_assessment(self, limit=100): #TODO убрать лимит
    после исследовательской
78          self.__set_images()
79          self.__prepare_input_datas()
80          self.input_datas = self.input_datas[:limit] # TODO: убрать
81
82          for i, input_data in enumerate(self.input_datas):
83              prediction = self.model.predict(input_data)
84              predicted_class = self.__get_class_by_prediction(prediction)
85              self.classes_images[predicted_class].append(self.images[i])
86              self.classes_count[predicted_class] += 1
87
88          # for image in classes_images['talking to passenger']:
89          #     plt.imshow(image)
90          #     plt.show()
91
92          return self.__get_assessment_by_classes_count()
93
94      def get_classes_images(self):
95          return self.classes_images
96
97      def __get_assessment_by_classes_count(self):
98          print(self.classes_count)
99          assesment = 100
100         k = len(self.input_datas) / assesment
101         for class_name, weight in self.cfg.classes_weights.items():
102             assesment -= (self.classes_count[class_name] / k) * weight
103         print(assesment)
104         return assesment
105
106     def __set_images(self):
107         cfg = self.cfg
108         if cfg.mode == DriverMode.VIDEO:
109             self.images = get_images.get_images_from_video(path_to_video=cfg.
    path_to_files, interval_sec=cfg.interval_sec)
110         elif cfg.mode == DriverMode.PHOTO:
```

```
111             self.images = get_images.get_images_from_folder(path_to_images=
    cfg.path_to_files)
112
113     def __prepare_input_datas(self):
114         input_datas = []
115         for i in self.images:
116             prepared_image = i.resize((224, 224))
117             if prepared_image.mode != 'RGB':
118                 prepared_image = prepared_image.convert('RGB')
119             input_data = tf.expand_dims(tf.image.convert_image_dtype(
    prepared_image, tf.float32), 0)
120             input_datas.append(input_data)
121         self.input_datas = input_datas
122
123     def __get_class_by_prediction(self, prediction):
124         first_output = prediction[0][0]
125         prediction_array = np.array(first_output)
126         # Найдите индекс наибольшего элемента в массиве предсказаний
127         predicted_class_index = np.argmax(prediction_array)
128         predicted_class = classes[f'c{predicted_class_index}']
129         return predicted_class
130
131
132 # cfg = DriverSafetyGetterConfig(classes_weights, DriverMode.PHOTO, './
    demo_data/imgs')
133 # getter = DriverSafetyAssessmentGetter(cfg)
134
135 # getter.get_driver_safety_assessment()
```

ПРИЛОЖЕНИЕ Г

Модуль загрузки данных

Листинг Г.1: Загрузка данных

```
import cv2
import os
from PIL import Image

import matplotlib.pyplot as plt

def get_images_from_video(path_to_video, interval_sec):
    video_capture = cv2.VideoCapture(path_to_video)
    frame_rate = video_capture.get(cv2.CAP_PROP_FPS)

    images = []
    current_frame = 0
    while True:
        success, frame = video_capture.read()
        if not success:
            break
        if current_frame % int(frame_rate * interval_sec) == 0:
            rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)  # Convert to
    RGB
            pil_image = Image.fromarray(rgb_frame)
            images.append(pil_image)
        current_frame += 1

    video_capture.release()
    return images

def get_images_from_folder(path_to_images):
    images = []
    for filename in os.listdir(path_to_images):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            image_path = os.path.join(path_to_images, filename)
            img = Image.open(image_path)
            images.append(img)
    return images
```