ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению   
38.03.05 Бизнес-информатика**

Ланских Юлия Дмитриевна

**КУРСОВАЯ РАБОТА**

Распознавание направления взгляда по видеоизображению лица для отслеживания состояния водителя

|  |  |
| --- | --- |
|  | Научный руководитель:  д.т.н., профессор кафедры Информационных систем и технологий НИУ Высшая школа экономики - Нижний Новгород  А.В. Савченко |

**Нижний Новгород, 2022**

Оглавление

Введение 3

Глава 1. Аналитический обзор литературы 6

1.1 Классические методы распознавания взгляда 6

1.2 Современные методы распознавания направления взгляда водителя 6

1.3 Нейронные сети для распознавания направления взгляда водителя 9

1.3.1 Обзор статей с конкурса EmotiW2020 9

Выводы по главе 1 16

Глава 2. Экспериментальное исследование 18

2.1. Подключение необходимых библиотек 18

2.2 Обработка 20

2.2.1 Распаковка набора данных 20

2.2.2 Обрезка изображений с использованием детектора лиц 21

2.3 Реализация нейросетевого классификатора 22

Заключение 25

Список литературы 26

Приложения 30

**Введение**

В настоящей курсовой работе рассматривается проблема распознавания направления взгляда по видеоизображению лица для отслеживания взгляда водителя. Поставленная проблема является междисциплинарной и относится к области машинного обучения и компьютерного зрения.

Невнимательность водителя – одна из самых распространенных причин автомобильных катастроф. Согласно недавно проведенному исследованию[16] 80% аварий связаны с отвлечением внимания водителя. Сфера проектирования, разработки и внедрения в эксплуатацию высокоавтономных транспортных средств стремительно развивается в течение последнего десятилетия. Другие исследования[17], [18] показали, что водители с наименьшей вероятностью сталкиваются с другими участниками дорожного движения, если в автомобиле есть пассажир, который своевременно предупреждает о предполагаемой опасности. Поэтому активно разрабатываются системы помощи водителю (Advances Driver Assistance Systems(ADAS))[19], которые обеспечивают безопасную эксплуатацию автомобиля, предупреждая о надвигающейся опасности или принимая управление транспортным средством в случае повышенной усталости водителя. Ключевая информация, которую необходимо принимать таким «техническим ассистентам», - это направление взгляда водителя и соответствующая автомобильная зона.

Актуальность работы заключается в огромном количестве различных подходов в рассматриваемой области. Создание и распространение новых наборов данных, улучшение эффективности и производительности работы определенных систем и компьютерных операций позволяет построить алгоритмы, способные к обобщению и эксплуатации в режиме реального времени. Проведенное исследование и его результаты могут использоваться в разработке программных обеспечений, предназначенных для помощи водителю в управлении транспортным средством(вне зависимости от типа) или для передачи управления транспортному средству и включение полуавтономного(автономного) режима.

Объект работы – набор данных с изображениями, собранными «in the wild». Это означает, что данные были собраны не в экспериментальных, а в максимально правдоподобных(реалистичных) условиях.

Предмет исследования – распознавание направления взгляда водителя по видеоизображению лица.

Цель работы состоит в создании(проектировании) алгоритма распознавания направления взгляда водителя по изображению лица с использованием нейросетевого классификатора.

Поставленная цель определила следующие задачи:

1. Провести аналитический обзор литературы по теме исследования;
2. Провести анализ и обзор статей по соответствующей теме с конкурса EmotiW2020;
3. Провести анализ набора данных Driver Gaze in Wild(DGW) с конкурса EmotiW2020;
4. Скачать набор данных Driver Gaze in Wild(DGW) с конкурса EmotiW2020;
5. Обработать входные данные(изображения) для более точных предсказаний с помощью обученного детектора лиц;
6. С помощью уже обученной на изображениях нейронной сети(модели), построить алгоритм, подходящий для достижения цели, с использованием нейросетевого классификатора;
7. Провести экспериментальное исследование по распознаванию направления взгляда водителя и сравнить результат с аналогами;
8. Повысить точность скаченной и обученной модели с помощью fine-tuning.

Гипотеза**:** Обучение нейросетевого классификатора на основе известных моделей нахождения особых точек и распознавания эмоций лиц, дообучение предобученной модели, достигающей точности более 56%.

**Глава 1. Аналитический обзор литературы**

**1.1 Классические методы распознавания взгляда**

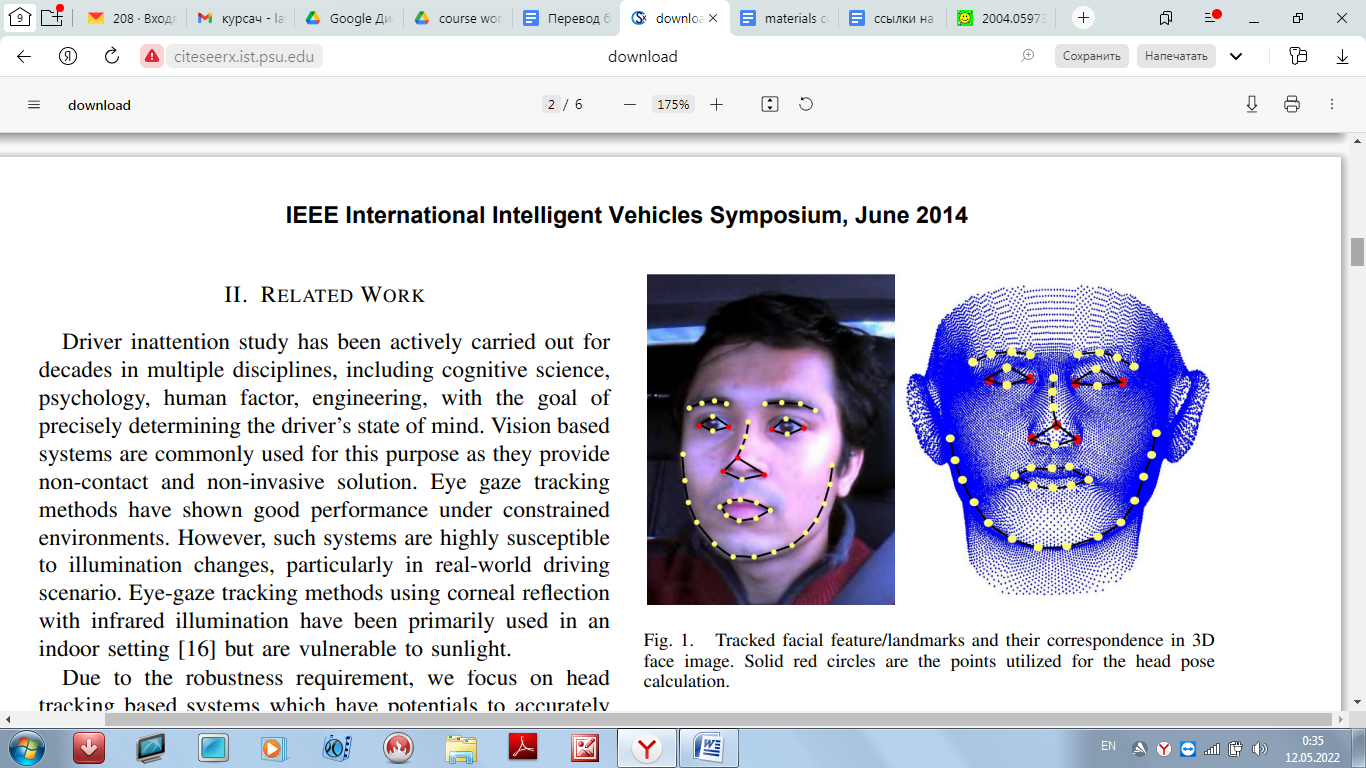
Определение зоны пристального взгляда водителя с помощью датчиков(инфракрасные (ИК) камеры [1], контактные линзы [2], устройства, устанавливаемые на голову [3, 4] и другие системы [3, 6]), фиксирующих подробную информацию о глазах и движениях зрачков, требует специфический набор определенных инструментов, которые могут вызвать неприятные ощущения у пользователя. В 1963 году был предложен способ использования контактных линз в качестве датчика[2]. Влияние специфики экспериментальной среды на линзы позволяло построенному прибору распознавать горизонтальные, вертикальные и круговые движения глаз.

Метод, базирующийся на отражении роговицы, полученном с помощью использования инфракрасного освещения(камеры)[7], также не является подходящим для вождения в режиме реального времени. Для корректной работы этого метода нужно обеспечить стационарное положение головы, не допускать попадания солнечного света, собрать данные и завершить процедуру многоточечной калибровки.

Таким образом, приведенные примеры датчиков в системе имеют чувствительность к наружному освещению, сложность в калибровке оборудования и системной интеграции, а постоянные вибрации и толчки во время движения могут негативно повлиять на производительность системы в целом.

**1.2 Современные методы распознавания направления взгляда водителя**

Группа исследователей Ashish Tawari and Mohan M. Trivedi в своей работе Robust and Continuous Estimation of Driver Gaze Zone by Dynamic Analysis of Multiple Face Videos[8] используют геометрический подход для оценки положения головы. Производительность такой системы была проанализирована на самостоятельно собранном наборе данных, отмаркированном вручную, где каждый участник фиксировал свою голову, смотря на 8 автомобильных зон. Геометрических подход основан на уголках глаз, уголках и кончике носа, которые выступают как ориентиры(рис. 1). На основе их 3D-конфигураций модель POS(Pose From Orthography and Scaling)[9]определяла матрицу поворота и соответствующие углы рыскания(yaw), тангажа(pitch) и крена(roll) положения головы.



*Рис.1 Отслеживаемые черты лица/ориентиры и их соответствие в 3D-изображении лица. Сплошные красные круги – это точки, выделенные для расчета позы головы.*

В этом исследовании провелась также оценка влияния статических и динамических характеристик движения головы. В качестве классификатора был выбран Random Forest Classifier по причине легкой настройки. Сравнивалась эффективность классификатора с использованием статических и статико-динамических характеристик. Основными результатами этой работы являются утверждение о повышении точности классификатора динамическими характеристиками; матрица ошибок(Confusion matrix) показывает, что большинство ошибок существует в смежных зонах пристального взгляда, поэтому интегрирование информации о глазах необходимо для устранения неоднозначности между соседними зонами; была представлена поза головы и ее динамическая система оценки зоны взгляда. Общая точность метода при учитывании статико-динамических и статических характеристик составляет 93% и 85.7% соответственно. В этом методе очень важно точно и надежно обнаружить ориентиры, в противном случае - построенная система будет работать некорректно.

Команда исследователей[8] расширила свой состав, видоизменила экспериментальный ход, добавив учитывание направления взгляда, и опубликовала новые результаты в другой статье «Where is the driver looking: Analysis of Head, Eye and Iris for Robust Gaze Zone Estimation»[10]. В этой статье была рассмотрена оценка грубого направления взгляда водителя с использованием сигналов головы и глаз и представлена новая вычислительная платформа с монокулярной камерой[20] для обнаружения радужной оболочки глаза. Собранные натуралистические данные о вождении были промаркированы вручную. Автоматически выделялись ориентиры: уголки глаз и их контур, уголки рта и носа, и кончик носа. Общая трехмерная модель лица с использованием алгоритма POS(Pose From Orthography and Scaling)[9]. После этого были реализованы масштабирование(до размера, при котором расстояние между обоими центрами глаз равно фиксированной длине) и поворот(линия, проходящая через оба центра глаза должна быть параллельна оси Х) изображений. Для обнаружения центров радужной оболочки нужно было проанализировать с помощью регрессионной матрицы S = (R1, ... Rk), где объекты обучения уже масштабированы и повернуты в нужном направлении. Для шага k = 1 авторы выдвинули предположение о расположении центра радужной оболочки по уже найденным на первом шаге точкам. В сумме выделяется 10 предположительных точек на одном глазе(рис. 2).



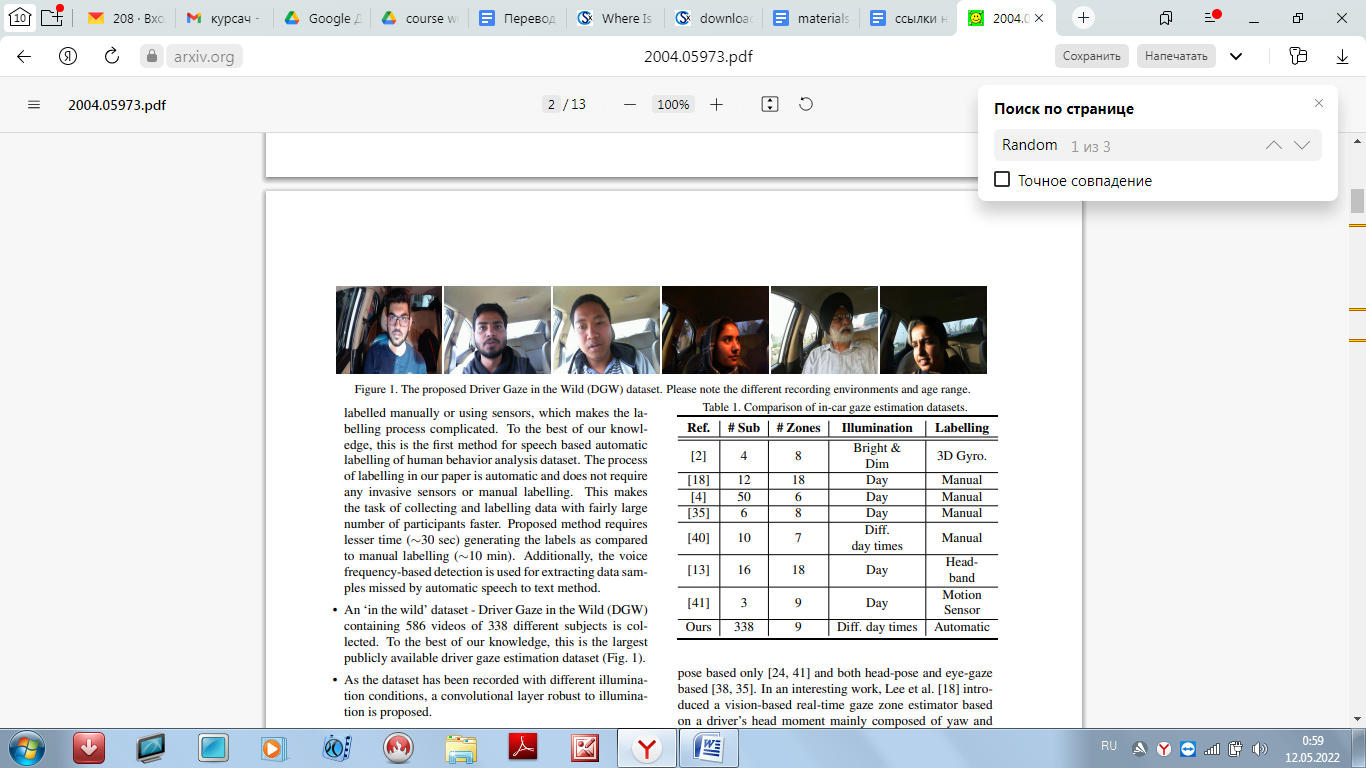
*Рис. 2 Пример масштабирования и выравнивания обрезанной фотографии с 10 первоначальными оценками центров радужной оболочки глаза*

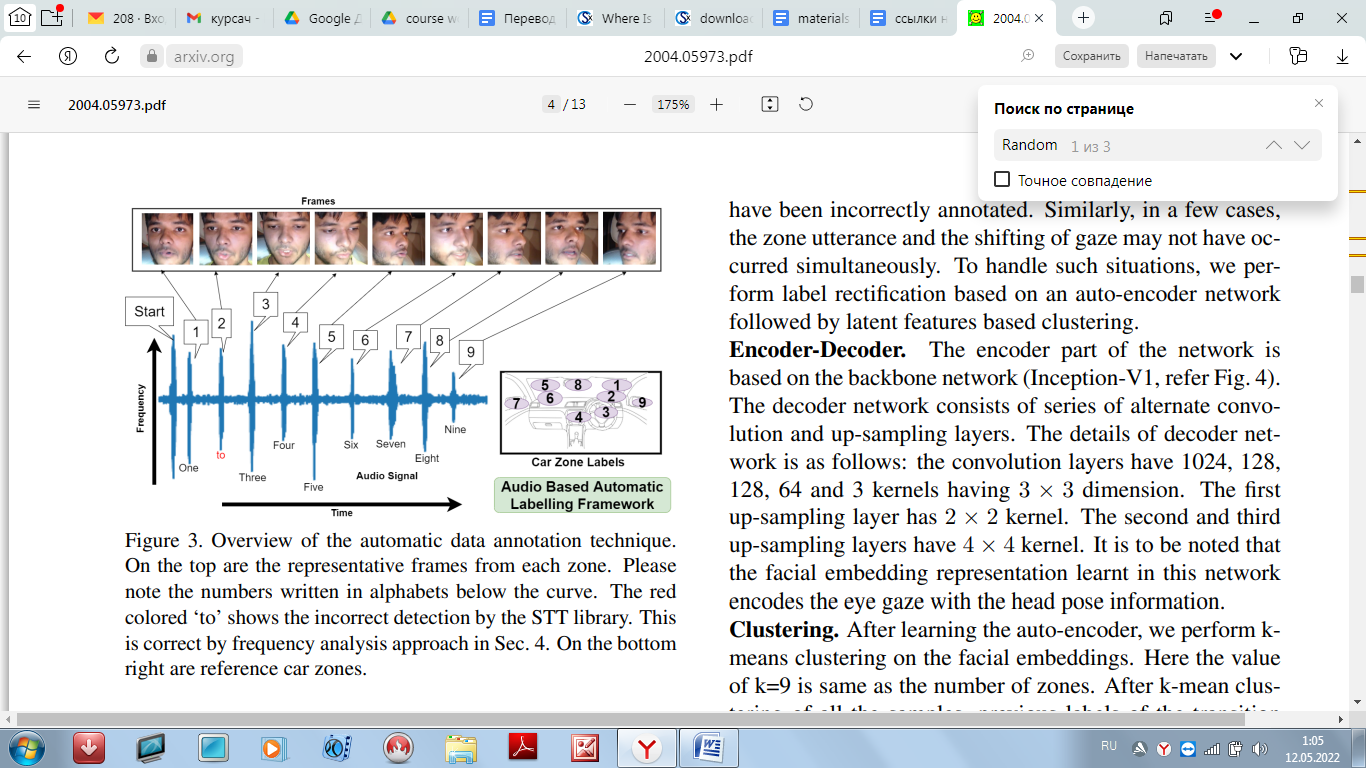
Для улучшения скорости обнаружения центра радужной оболочки в видео, было реализовано его отслеживание с использованием обнаруженных позиций из предыдущего кадра, которые выступают как начальная оценка текущего кадра. В качестве классификатора был выбран RandomForestClassifier. Для 6 автомобильных зон общая точность равна 93,6%.

**3. Нейронные сети для распознавания направления взгляда водителя**

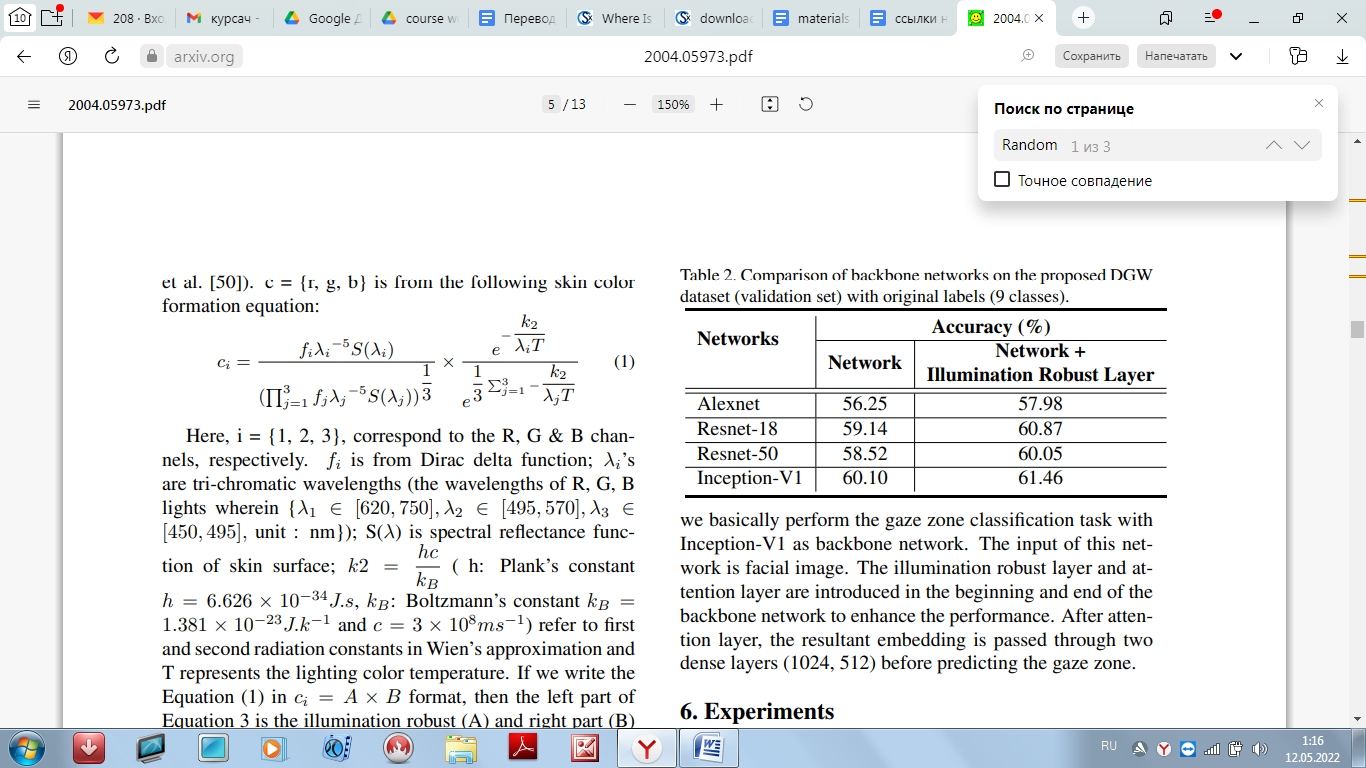
**3.1. Обзор статей с конкурса EmotiW2020**

Команда победителей Shreya Ghosh, Abhinav Dhall, Garima Sharma, Sarthak Gupta, Nicu Sebe в конкурсе EmotiW2020 опубликовала статью «Using Domain Knowledge for Creating a Large Scale Driver Gaze Zone Estimation Dataset»[11] с подробным описанием проделанного эксперимента и его результатами. В первую очередь, был собран большой набор видео-данных, снятых на одну камеру со средним цветным разрешением, "inwild", то есть в максимально правдоподобных различных погодных, временных и световых условиях(рис. 3). На видео запечатлено, как каждый из участников фиксирует взгляд и направляет голову на одну из маркированных зон автомобиля(всего их 9) и произносит номер зоны вслух. Маркировка кадров происходила не вручную, при помощи Speech to Text[21](рис. 4). Помеченные эпизоды извлекались со смещением на ±10 кадров. Входным сигналом для дальнейшей сети является обрезанное лицо размером 224 × 224 × 3, вычисленное с использованием библиотеки распознавания лиц Dlib.

*Рис. 3 Примеры изображений из предлагаемого «Driver Gaze in Wild» (DGW) набора данных.*

**  
  
*Рис.4 Обзор метода автоматической маркировки данных. Сверху представлены кадры для каждой зоны. Слово “to” красного цвета показывает неправильно обнаружение библиотекой STT. В правом нижнем углу указана нумерация и соответствующее расположение автомобильных зон.*

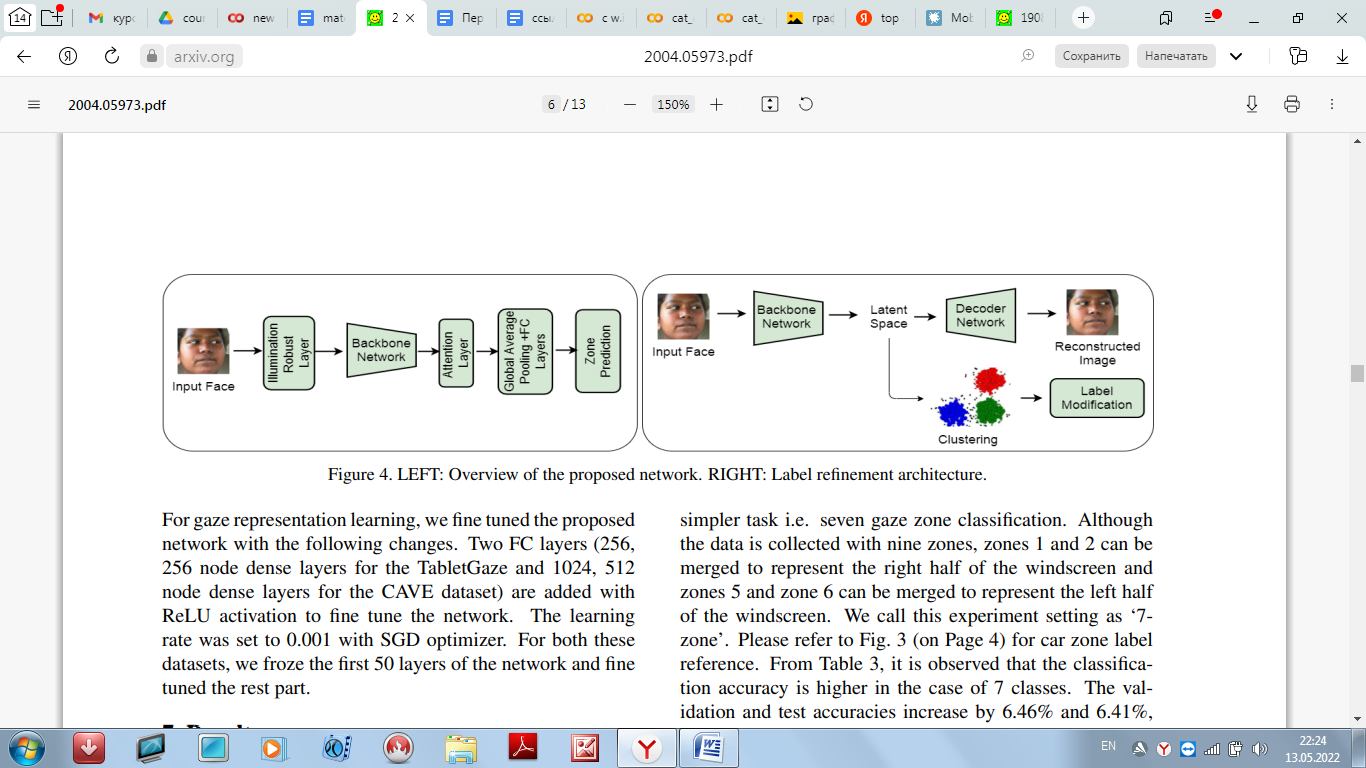
Из Таблицы 1 видно, что магистральная сеть(backbone network) выбиралась из Alexnet [12], Resnet [13] и InceptionNetwork. Преимущество Inception-V1[22] и обоснование, почему именно эта сеть используется в качестве магистральной также следует из Таблицы 1: она показывает наиболее высокую точность на проверочном наборе(validation set).Базовая линия метода(baseline network) представляет собой Inception-V1 со слоем глобального пула(Global average pooling)[23] и полносвязным слоем(Fully Connected layer слои.



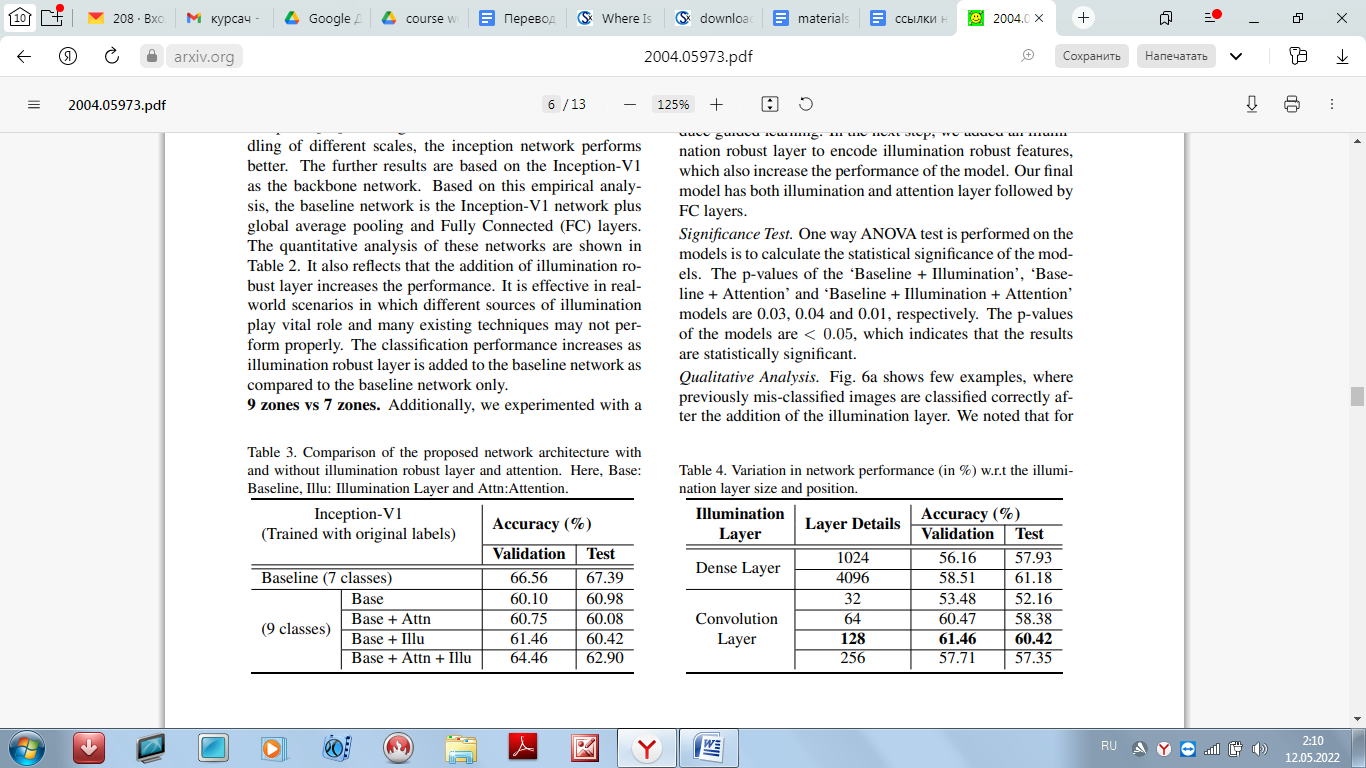
*Таблица 1. Сравнение точности магистральных сетей(backbone) на проверочном множестве из предлагаемого набора данных DGW с исходными метками(9 классов).*

Для нейтрализации некорректной квалификации зоны ввиду различного освещения был добавлен слой, устойчивый к освещению(Illumination Robust Layer), в качестве концепции которого принят закон идеальной матовой поверхности(закон косинуса Ламберта)[14]. В конце магистральной сети вводится механизм внимания(Attention Augmented Convolution)[15] для повышения производительности. После этого слоя изображение пропускается через два плотных слоя с количеством нейронов в 1024 и 512 соответственно. Модели в каждом случае обучаются 200 эпох с размером пакета(batch\_size) равному 32. Были проведены эксперименты с разными комбинациями "составляющих" сетевой архитектуры:

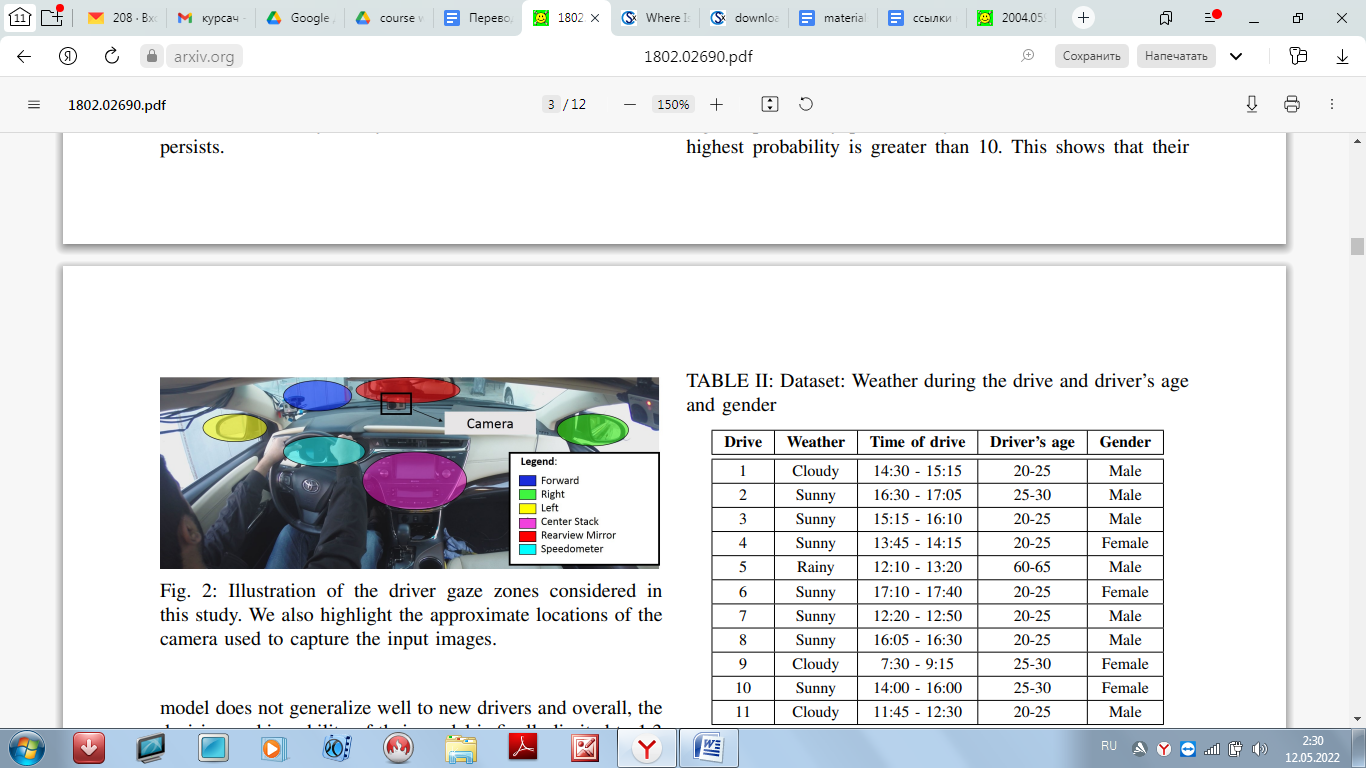
1. Бейзлайн;
2. Бейзлайн + механизм внимания;
3. Бейзлайн + слой, устойчивый к освещению;
4. Бейзлайн + механизм внимания + слой, устойчивый к освещению(рис.5);

  
*Рис.5. Обзор предлагаемой исследователями архитектуры сети*

Безусловно, наибольший показатель точности на проверочном наборе(64.46%) и тестовом наборе(62.90%) показывает последний вариант сетевой архитектуры с 9 классами(то есть, с 9 зонами пристального внимания водителя). При 7 классах точность на базовой линии метода составляет 66.56% на проверочном наборе и 67.37% на тестовом(см. Таблица 2).

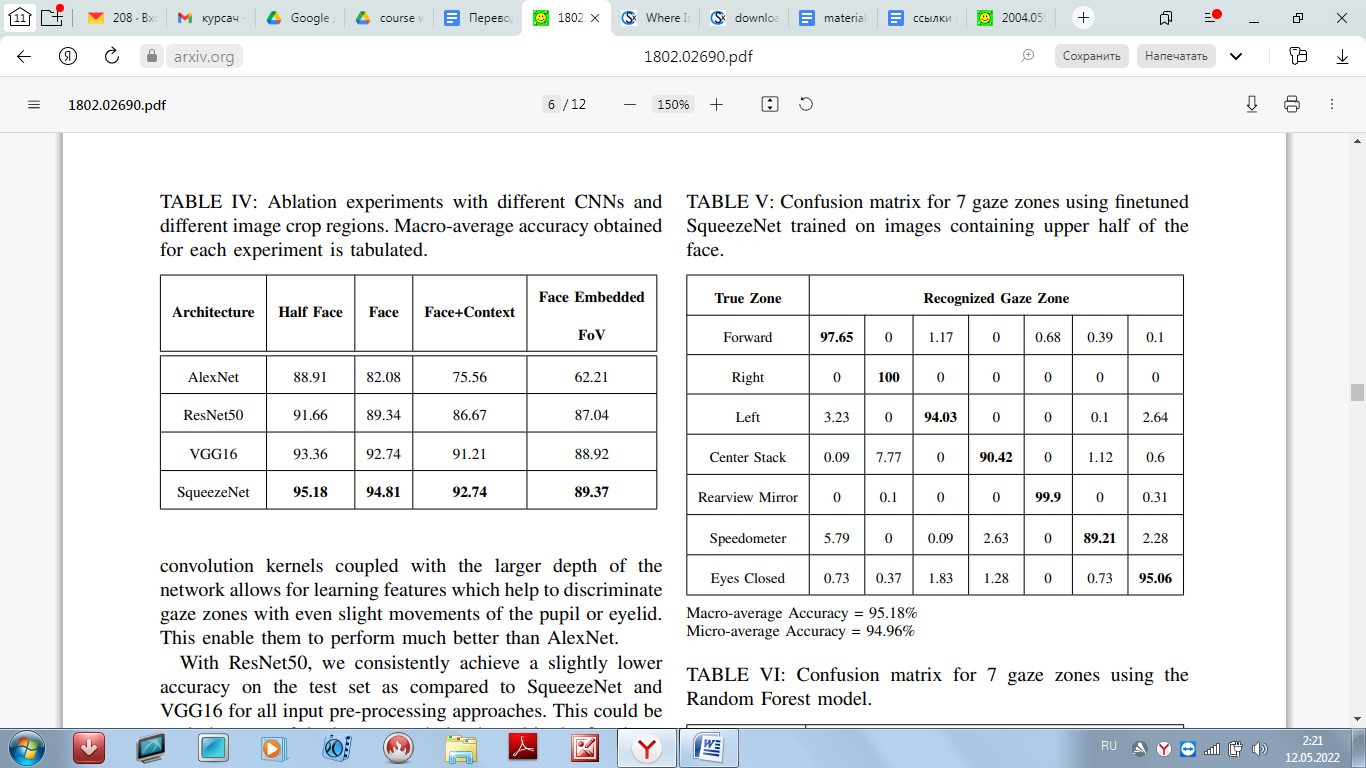
  
*Таблица 2. Сравнение точности предлагаемой сетевой архитектуры с и без Attention layer; с и без Illumination robust layer; и сAttention layer, и с Illumination robust layer одновременно.*

В качестве основной идеи статьи «Driver Gaze Zone Estimation using Convolutional Neural Networks: A General Framework and Ablative Analysis» команды Sourabh Vora, Akshay Rangesh, and Mohan M. Trivedi[24], принимающей участие в этом же конкурсе лежит концепция о важности "смотреть" водителю в глаза.

Процесс сбора данных отличается от их соперников только тем, что съемка велась на 3 синхронизированные камеры с высоким разрешением и разными полями обзора, но в эксперименте используются изображения с камеры, которая была установлена рядом с зеркалом заднего вида. Было выделено 7 зон пристального взгляда водителя(рис.6). В Таблице 3 приведены показатели точности 16 способов(попарная комбинация одной из сетей и одного из способов предварительной обработки изображения) построения модели(рис.8).

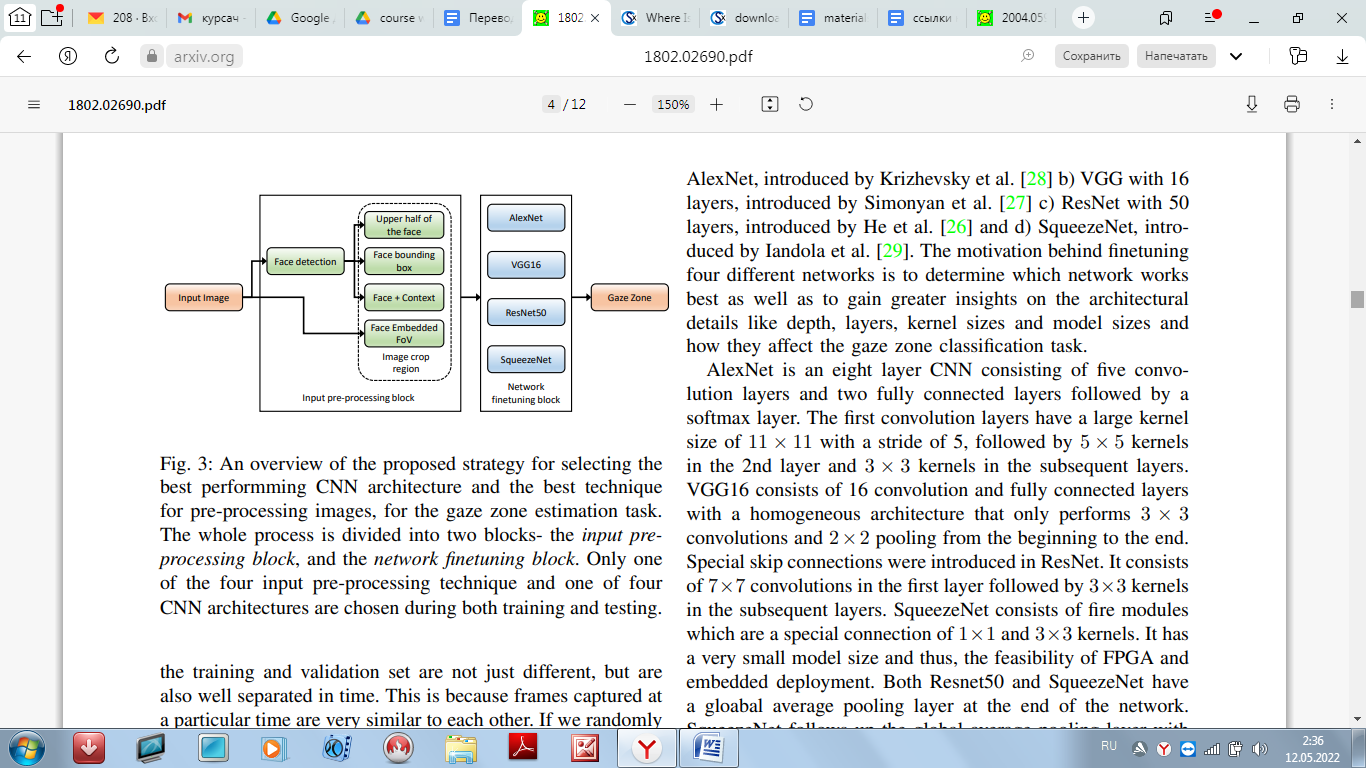
*Рис. 6. Иллюстрация зон пристального взгляда водителя, рассмотренных в описываемом исследовании. Выделено приблизительное местоположение камеры, используемой для захвата входных изображений.*

В блоке точной настройки сети экспериментировали с AlexNet[12], Resnet50[13], VGG16[25] и SqueezeNet[26].



*Таблица 3. Точность как результат каждого эксперимента с комбинацией различных сверточных нейронных сетей и различными областями(способами) обрезки изображения.*

Были описаны архитектуры каждой из предложенной сетей. Анализ четырех сетей позволяет определить сеть с наилучшей производительностью, получить более глубокое представление об архитектурных деталях(глубина, слои, размеры ядры и модели, их влияние на задачу классификации зон обзора). Из Таблицы 3 заметно преимущество SqueezeNet в качестве магистральной сети(backbone network). Для этой сети установили скорость обучения, равную 4 х 10-4, размер пакета(batchsize), равный 16, и снизили количество ядер в последнем слое свертки от 1000 до 7. В ее архитектуре нет полносвязныхслоев(Fully Connected layers), конечный слой свертки имеет 7 фильтров, создающих 7 классов карты активации(CAMs), соответствующих зонам пристального взгляда. В конце добавлен слой глобального пула(Global average pooling) и слой softmax.

  
*Рис. 7. Обзор предлагаемой стратегии для выбора наиболее эффективной сверточной нейронной сети и наиболее эффективного способа обработки изображений.*

В блоке предварительной обработки ввода проведено исследование эффективности четырех способов, для каждого из которых вводом было изображение размером 224х224 или 227х227(рис. 8):

1. HalfFace: изображение обрезано до половины лица, нижней части лица не видно;

2. Face: исключительно лицо;

3. Face + Context: лицо + контекст;

4. Face Embedded FoV.

  
*Рис. 8. Различные области обрезки областей входного изображения, которые рассматриваются как способы предобработки изображений перед обучением модели. Области имеют цветовую маркировку для наглядности.*

Чтобы удостовериться в возможности сети сфокусироваться на глазах водителя и оценить эффективность каждого из способов, исследователи визуализируют работу семи отображений активации класса(CAMs)[27], полученных до слоя глобального пула(Global average pooling). Выводы визуализации заключаются в том, что Class activation maps(CAMs) всегда локализуются НАД глазами водителя; сеть учится разумно фокусироваться на одном или обоих глазах водителя; сеть учится справляться с окклюзией(вставить ссылку на сколар что это такое), разумно фокусируясь либо на одном глазу, либо на обоих глазах водителя. Важнейшим следствием проведенного эксперимента является подтверждение того, что модель, обученная на изображениях верхней половины лица(HalfFace), способна намного лучше извлекать более мелкие особенности глаза(положение и форма радужной оболочки и века), что объясняет ее лучшую производительность.  Затем, попробовали обучать с использованием изображений Face Embedded FoV, встроенных в лицо. Когда сеть была настроена на изображениях 625 × 625, достигнута точность, равная 92,13%. Эта точность ниже, чем при обучении на верхней половине изображений лиц, но здесь не требуется отдельного этапа распознавания лиц.

Для проверки работоспособности модели к обобщению был проведен эксперимент на наборе данных Columbia Gaze [36], который имеет 5880 изображений с высоким разрешением 56 субъектов с большой вариабельностью этнической принадлежности. Вывод о том, что без какой-либо подготовки модель с магистральной дообученной сетью SqeezeNet, обученная на Half-Face изображениях успешно кодирует различные направления взгляда в этом разнообразном наборе данных, демонстрируя хорошие возможности обобщения, стал результатом этой проверки.

Таким образом, пары HalfFace + SqeezeNet и FaceEmbeddedFoV + SqeezeNet отлично работают в режиме реального времени с разными частотами(166,7 Гц и 16 Гц соответственно) с точностью в 95.18% и 92.13% соответственно.

Основными результатами работы являются: большой натуралистический набор данных о вождении с большой вариативностью, систематический абляционное исследование различных сетевых архитектур и входных стратегий для обобщения систем оценки зоны пристального взгляда водителя.

**Выводы по главе I**

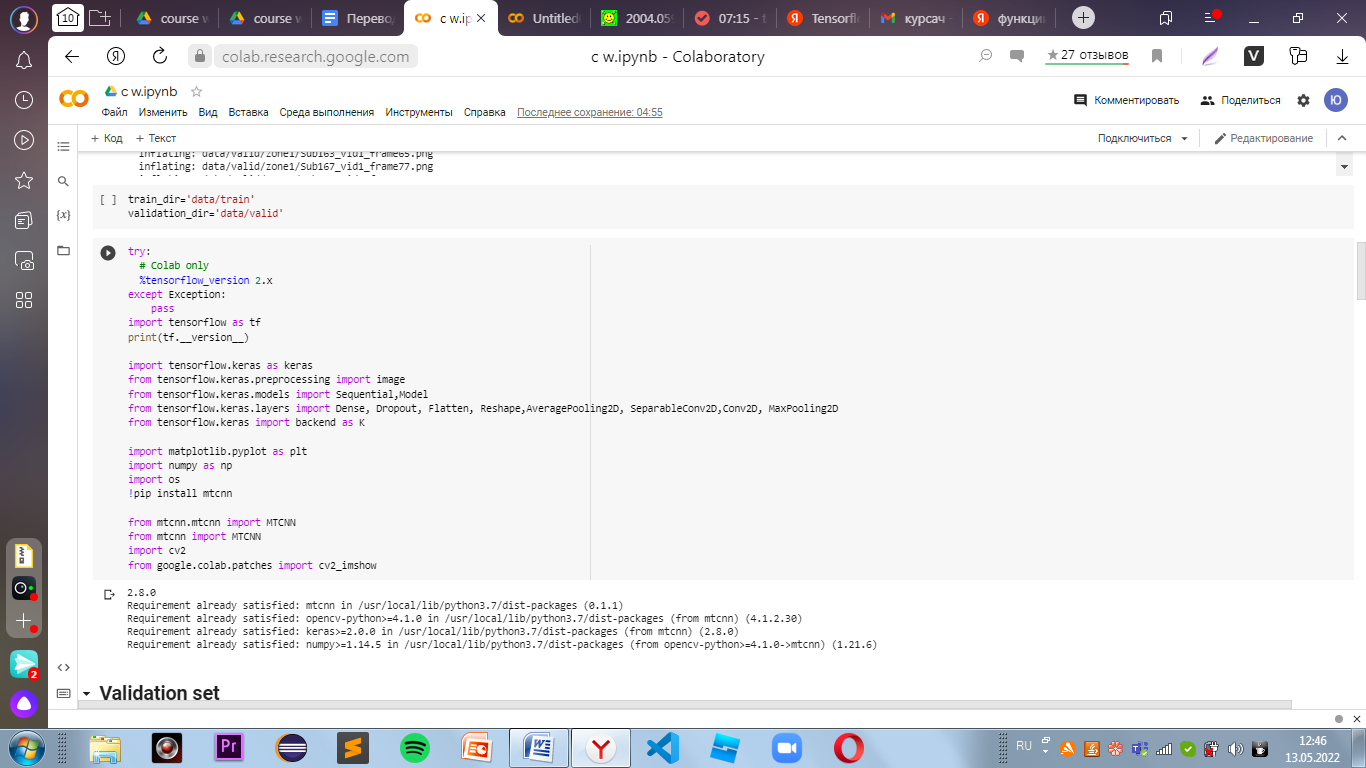
Разработка и внедрение усовершенствованной системы помощи водителю, которой можно доверить эксплуатацию транспортным средством, и, соответственно, жизнь и передать ответственность за судьбы водителей – процесс, который строится на большой теоретической базе, многочисленных исследованиях и экспериментах, на опыте и выводах предшествующих работ. Идея о разработке алгоритма, подходящего для распознавания направления взгляда водителя по видеоизображению лица, с использованием нейросетевого классификатора может быть реализована благодаря обозначению автомобильных зон пристального внимания водителя, их универсальности(расположение этих зон в различных автомобилях совпадает) и возможности нейронных сетей находить «общие черты» и корреляции данных. В качестве базовой статьи принято взять статью[24] победителей конкурса EmotiW2020 и реализовать в качестве решения соответствующей задачи, совпадающей с целью курсовой работы, бейзлайн.

**Глава 2. Экспериментальное исследование**

**1. Подключение необходимых библиотек**

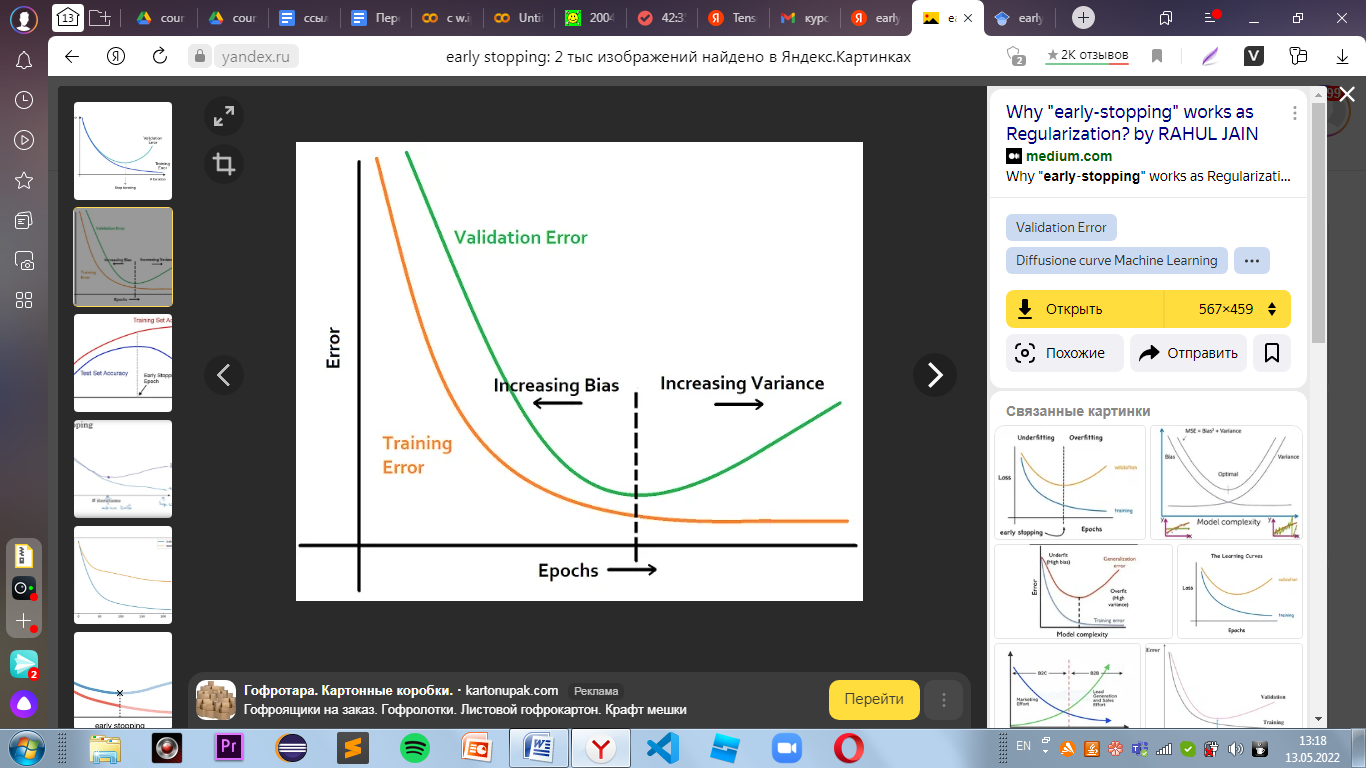
В качестве среды разработки был использован Google Colaboratory. Основное преимущество этого облачного сервиса для экспериментов с большим набором данных - возможность импорта данных из GoogleДиска в блокнот Colab, простота в использовании и легкое подключение нужных модулей, выполнение на серверах Google(это замедляет выполнение кода), предоставление необходимых инструменты(библиотеки) для машинного обучения в браузере.

В первую очередь, нужно подключить GoogleДиск и разрешить экспериментальному блокноту использовать исследовательские материалы по указанной директории. Соответственно, в этой директории находится скаченный архивированный(zip) набор данных Driver Gaze in the Wild(DGW) с конкурса EmotiW2020. Он разделен на несколько архивированных папок ввиду ограниченности Google Colaboratory при распаковке файла размера более 4 ГБ.

Эксперименты проводились с библиотекой TensorFlow, используемый для задач машинного обучения.  
  
 *Рис. 9.Подключение библиотеки TensorFlow.*

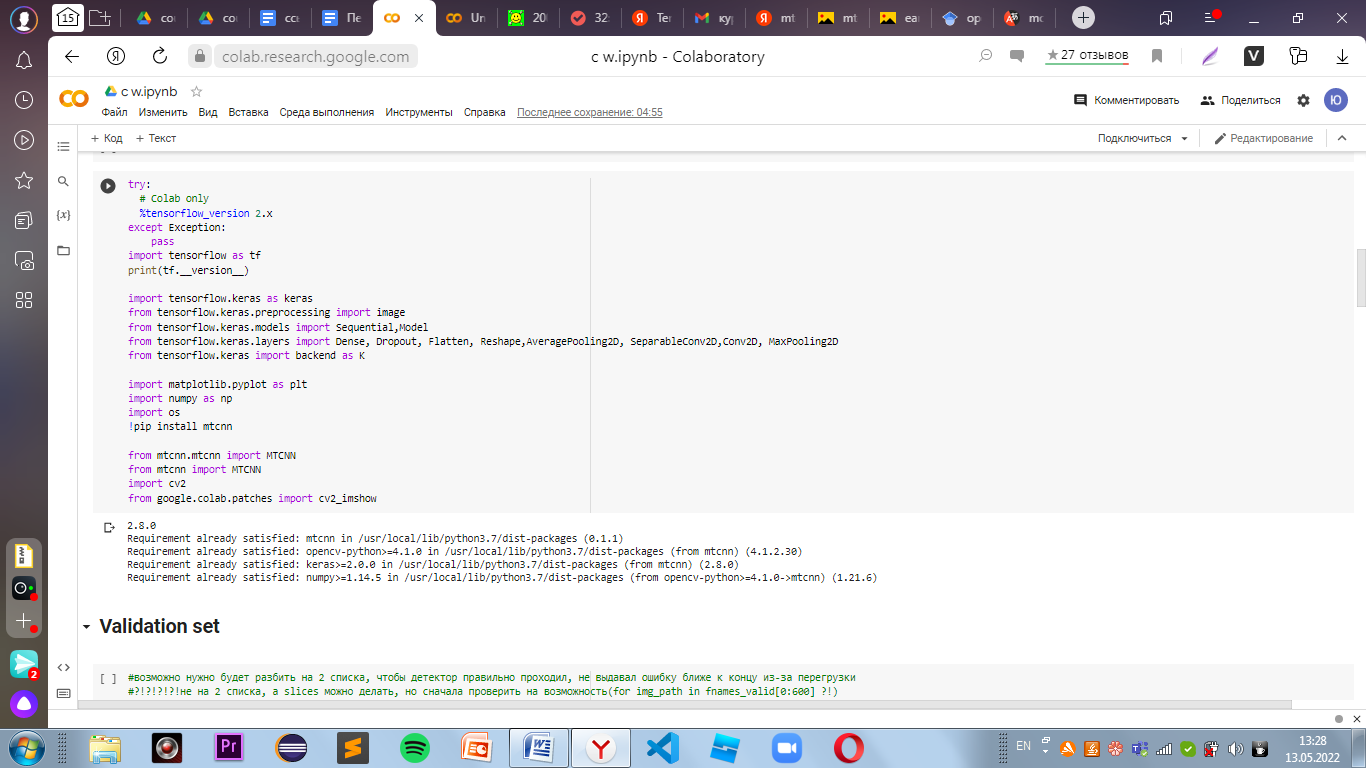
Последняя команда на рис. 9 позволяет убедиться в корректной установке библиотеки. После этого была подключена библиотека Keras, которая представляет собой надстройку над фреймворком TensorFlow, нацелена на оперативную работу с сетями глубинного обучения и спроектирована как компактная, модульная и расширяемая при работе система. Из Keras импортированы модули, позволяющие реализовать предобработку изображений, загрузку предобученной модели и её дообучение(fine-tuning). Для загрузки моделей использовался класс Model из Keras.Model. Необходимые модели сетей имплементированы в библиотеку, поэтому они импортировались из keras.applications.

Для начала работы с ключевой частью экспериментального блока работы были подключены ImageDataGenerator[28] как генератор изображений, позволяющий направлять данные потоком напрямую из папки на Google Диске и изменять свойства изображений так, как необходимо разработчику; ModelCheckpoint[29] необходимый для улучшения процесса обучения и для работы которого определяется файл в переменной BEST\_MODEL\_FILE, куда добавляются лучшие веса после обучения; EarlyStopping[30] как функция, которая наблюдает за метрикой качества обучения и прерывает процесс обучения, если эта метрика начинает снижаться(рис. 10).



*Рис.10. Наглядный пример работы EarlyStopping на примере остановки в случае, если выбранная метрика начинает увеличиваться(в случае на рис. :validation error - контролируемая метрика). На рисунке: ось Х - количество эпох, ось Y - величина ошибки, желтый график - значение ошибки обучения, зеленый график - значение ошибки валидации.*

Для нахождения лица на изображении используется MTCNN как нейронная сеть для детекции лиц, каскад сверточных нейронных сетей(P-Net, R-Net, O-net)[31]. Далее импортирована библиотека OpenCV[32] как open sourse библиотека компьютерного зрения, которая предназначена для анализа, классификации и обработки изображений, с помощью команд.(Рис. 11)



*Рис.11 . Подключение библиотек и импорт модулей, необходимых для обрезки фотографии и нахождения квадрата лица с помощью детектора лиц.*

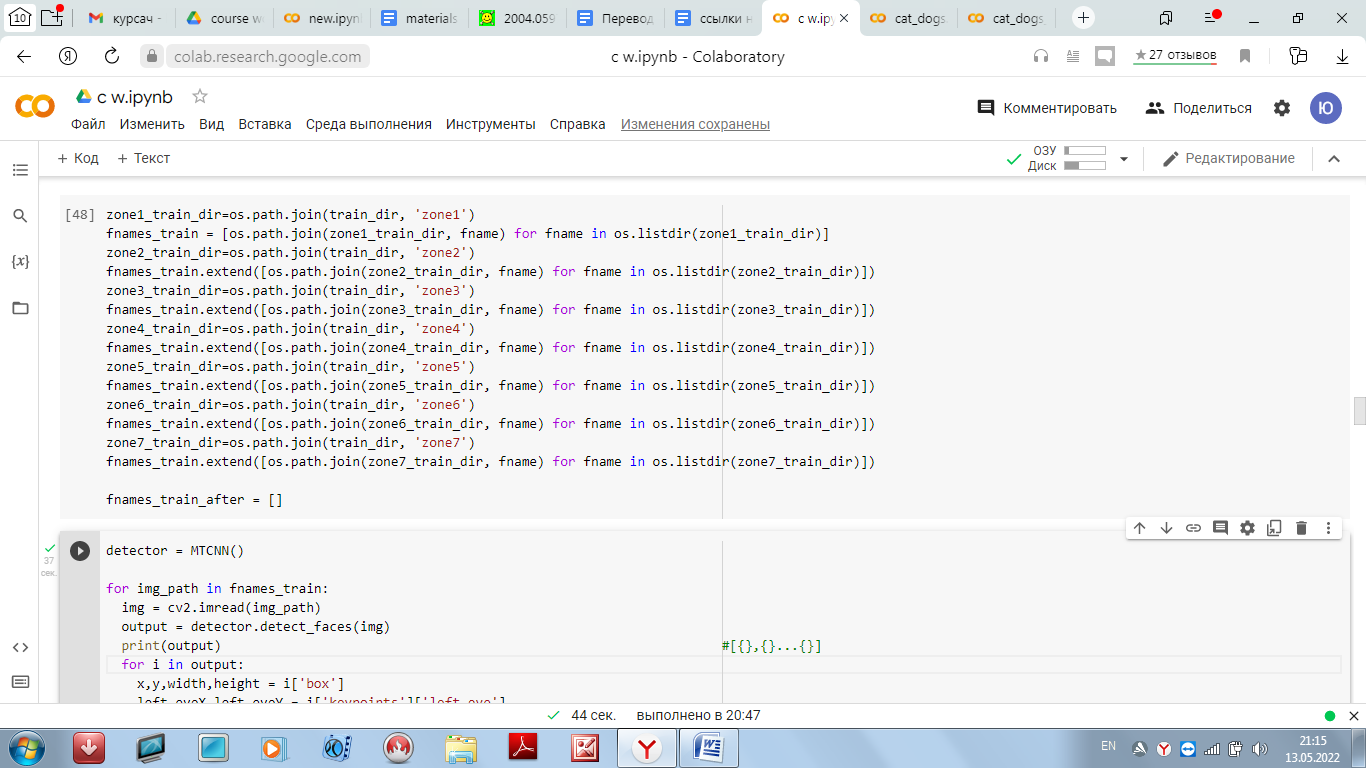
Модули библиотеки необходимы для загрузки, отображения и сохранения изображения.

**2. Обработка**

**2.1. Распаковка набора данных**

Набор данных для реализации нейросетевого классификатора поделен на 2 выборки: обучающая(train) - 64% от набора данных, проверочная или валидационная(validation) - 36% от набора данных. Каждая из выборок разделена на 7 множеств. Количество множеств и их названия соответствуют количеству и нумерации автомобильных зон пристального внимания водителя, рассматриваемых в дальнейшем. Пример изображения водителя из набора данных показан на Рис. 3.

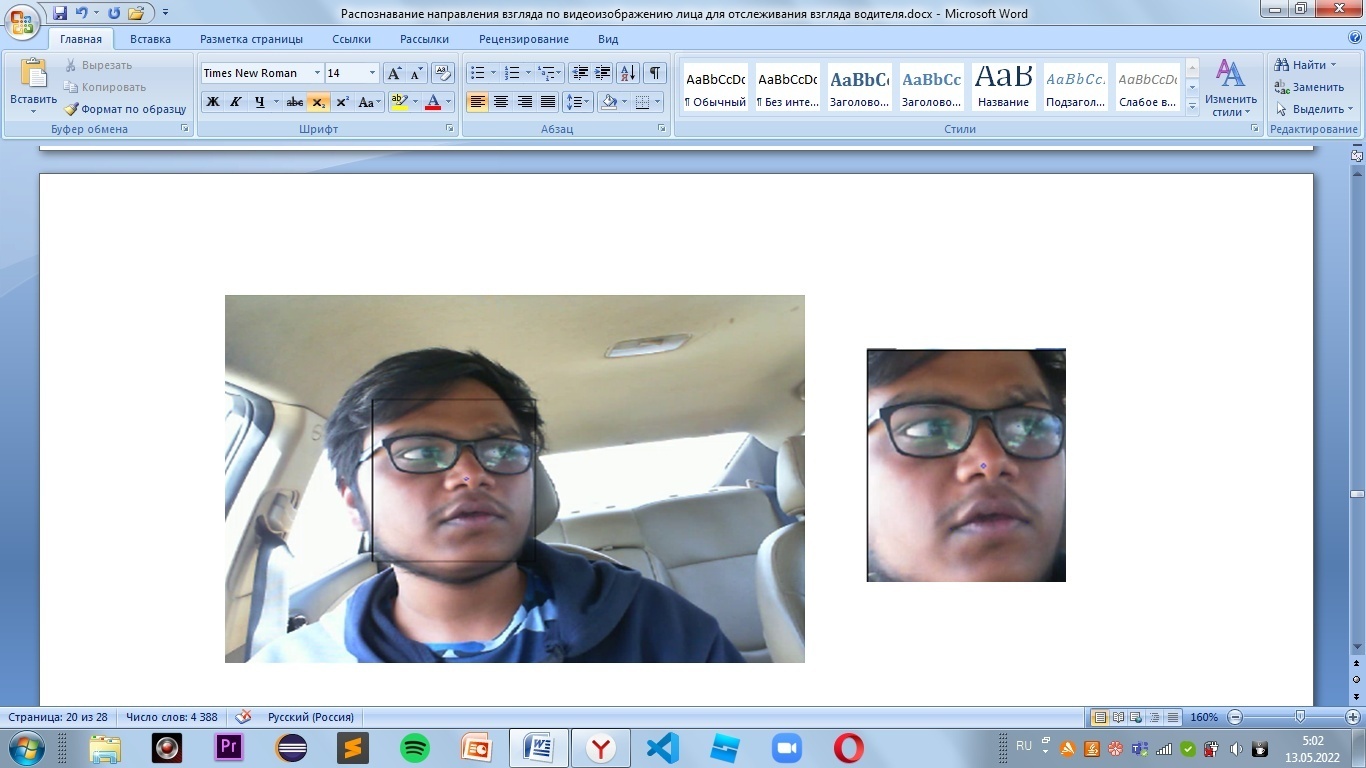
После подключения необходимых вышеприведенных библиотек были введены переменные, которые хранят в себе список директорий каждого изображения из папки с данными валидационного множества(validation set) и тренировочного множества(train set) соответственно(рис. 12).



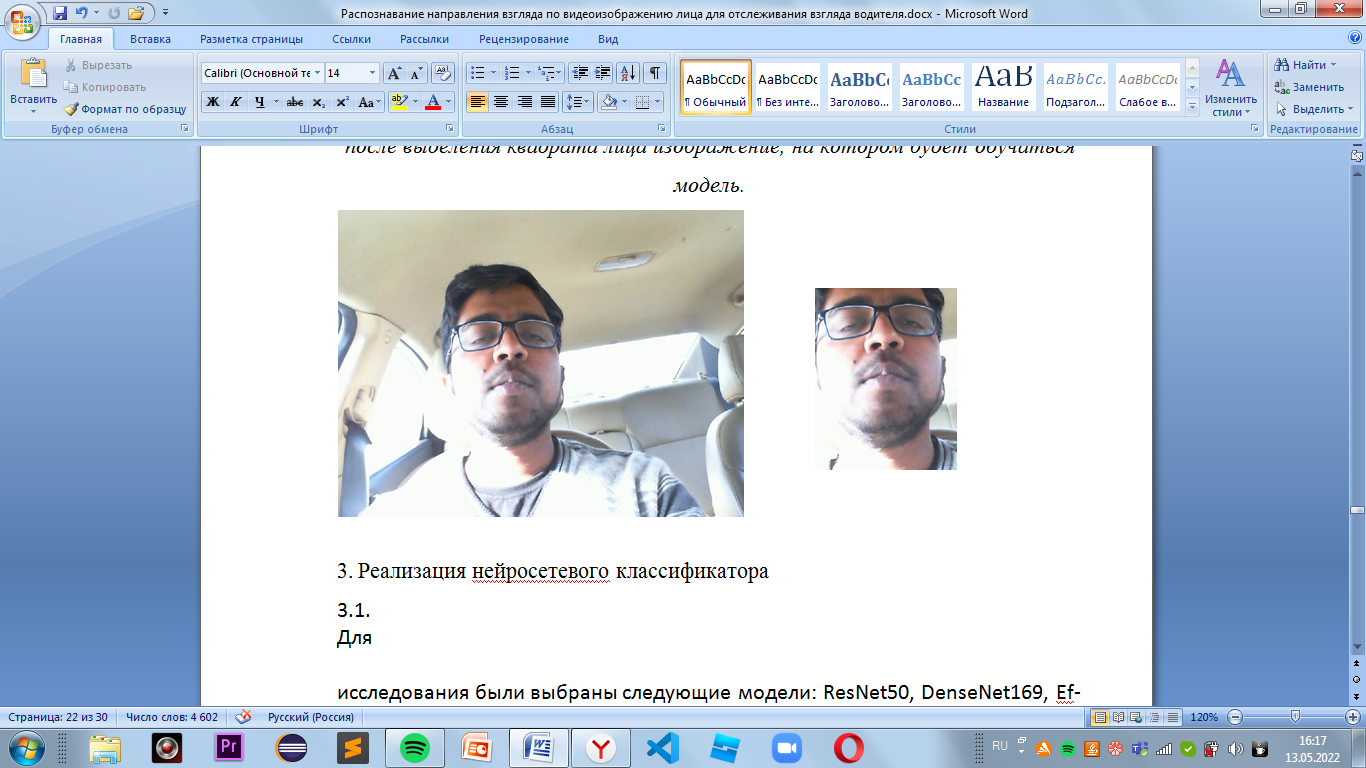
*Рис.12. Создание списка, элементами которого являются директории изображений из выборок*

**2.2. Обрезка изображений с использованием детектора лиц**

Последовательно с помощью директорий из списка изображение читается благодаря командой cv2.imread(path), где path – это название директории с одной из фотографий множества, и возвращается NumPy массив[33]. В дальнейшем возможность легкой обрезки(точнее: выделения и сохранения) квадрата лица обусловлена представлением изображения в виде NumPy массива. Загруженное изображение проходит детектор лиц MTCNN, который выделяет характерные точки, соответствующие левому/правому глазу, нос, левому/правому уголку рта и полноценный квадрат лица. Пройденное детектор изображение "обрезается" путем выделения части массива изображения с помощью срезов(slices), в качестве крайних значений диапазона выбираются найденные характерные для квадрата лица точки(рис. 13). Ввиду различного освещения и ракурса съемки детектор не всегда корректно определял квадрат лица, поэтому в качестве предотвращения сохранения пустого или неправильно выделенного изображения была введена обрезка по примерным точкам квадрата лица. Значения точек выбраны вручную после анализа нескольких неправильно определенных изображений. Полученные изображения(NumPy) исключительно с квадратом лица записываются в прежнюю директорию(то есть исходные изображения замещаются полученными).



*Рис.13 . Левое изображение: исходным изображением пройден детектор лиц, отмечены характерные точки на лице(маркировка синим цветом) и выделен квадрат лица(маркировка черным цветом). Правое изображение: полученное после выделения квадрата лица изображение, на котором будет обучаться модель.*



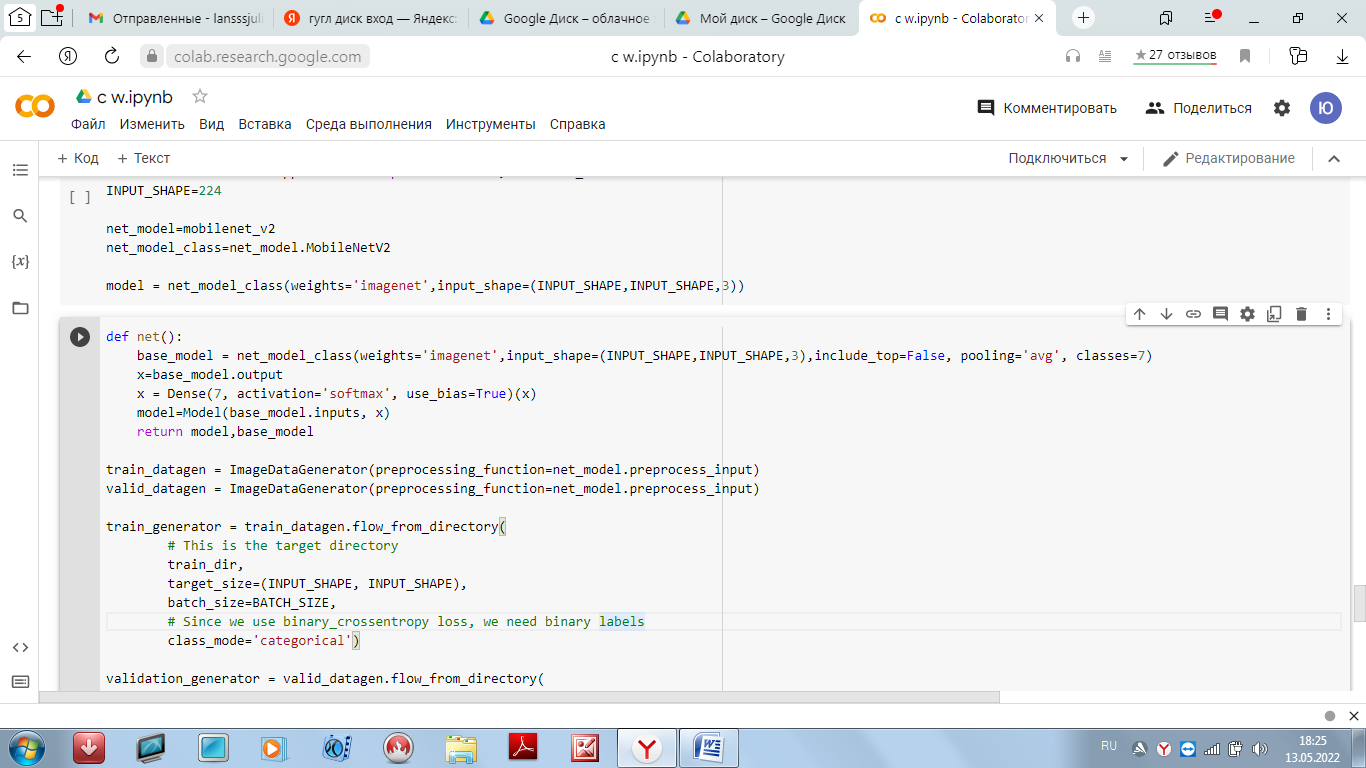
*Рис.14 . Левое изображение: исходное изображение не прошло детектор лиц(характерные точки не определены). Правое изображение: выделенное лицо по примерным, полученным вручную с помощью анализа точкам.*

**3. Реализация нейросетевого классификатора**

После этого генерируется проверочный и валидационный набор данных с помощью подключенной функции ImageDataGenerator, и проходит предобработка изображений для соответствия формату обучаемой модели благодаря настраиваемой в зависимости от выбранной архитектуры preprocessing function. Происходит генерация дополненных данных с помощью введения пути к каталогу и flow from directory. Данные сохраняются в train\_generator и valid\_generator. Из ключевых параметрах при flow from directory стоит перечислить class\_mode=’categorical’(это объясняется небинарностью набора данных) и установка размера изображения 224х224 в target\_size. Установленный размер бэтча равен 32 изображениям(общее число тренировочных объектов, представленных в одном бэтче(пакете или партии), на которые делится весь набор данных).

Для исследования была выбрана модель MobileNet V2. Необходима инициализация сети для ее дальнейшего обучения.

Далее определяется функция net(), которая дообучает(fine-tuning, то есть проходит тонкая настройка сети) загруженную модель. Ключевым этапом здесь является добавление Dense-слоя и определение 7 выходных классов, активация softmax(*Рис.15*). Применение этой активации необходимо в случае, если количество возможных классов небинарно(в настоящем исследовании количество классов = 7). Следующий этап: fine-tuning модели с помощью определенной и вышеописанной функцией net().



*Рис.15. Дообучение (fine-tuning) модели*

Для всех слоев в модели устанавливается свойство non-trainable с помощью цикла for и параметра layer.trainable = False. Это необходимо для «заморозки» слоёв: состояние замороженного слоя и его обучаемый вес не будут обновляться во время обучения.

Следующий ключевой шаг: обучение и проверка на имеющемся наборе данных. Значения параметров, настраиваемых перед обучением, для компиляции модели:

1. функция потерь(loss) - категориальная кросс-энтропия('categorical crossentropy'), обосновывается категориальным форматом данных и количеством категорий(автомобильных зон, рассматриваемых в этом исследовании больше 2(небинарность));

2. метрика для оценки модели и мониторинга обучения модели(metrics) - точность('acc')

3. оптимизатор(omptimizer) - 'adam'(помогает ускорить обучение и достигнуть лучших результатов. Простая реализация, вычислительная эффективность и небольшие требования к памяти можно выделить в качестве преимуществ именно этого оптимизатора)[34];

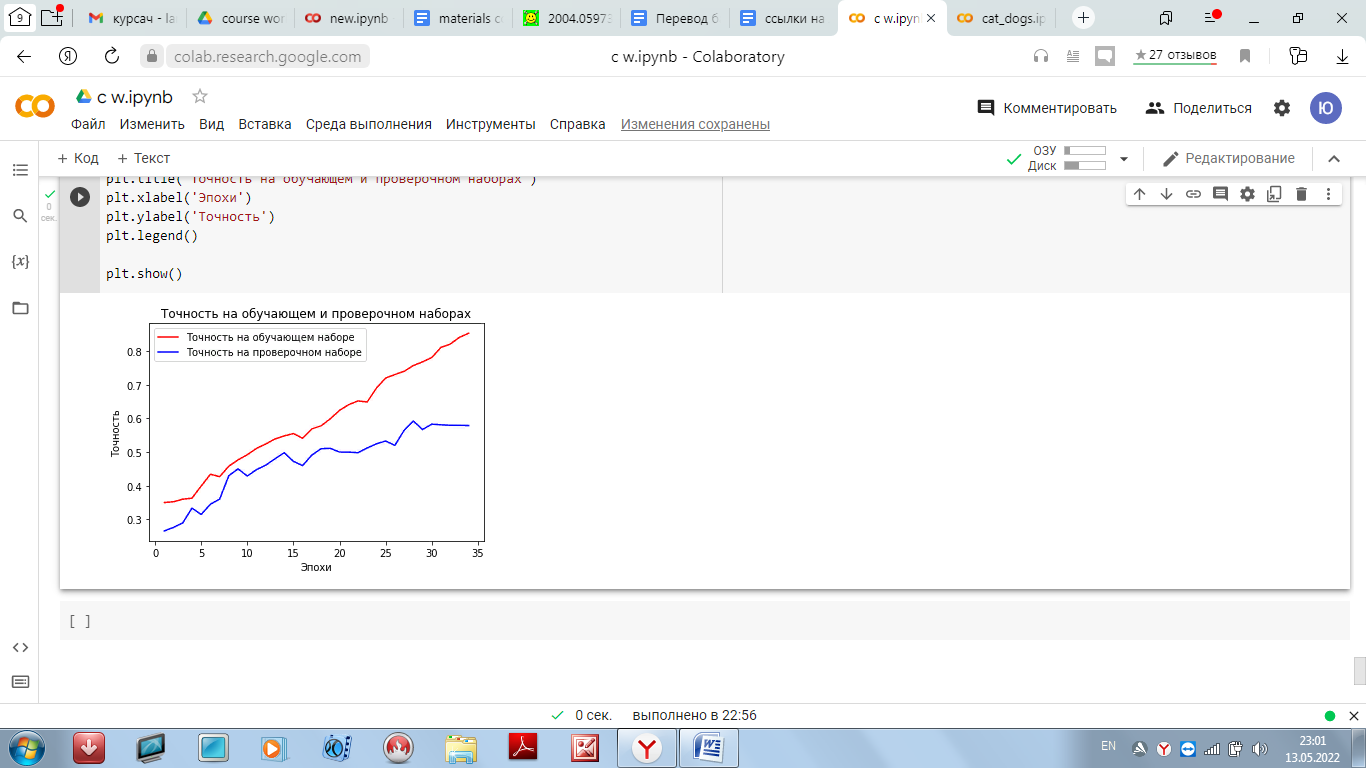
После этого применяется EarlyStopping как способ регулирования, используемый для предотвращения переобучения. Эта функция принимает в качестве параметров показатель, который будет отслеживаться(в данном случае – точность на валидационном множестве) и количество эпох, которое «терпимо» до начала переобучения, то есть если в течение указанного количества эпох точность на проверочном множестве не увеличивается, то модель заканчивает обучение и останавливается раньше, чем пройдет все эпохи обучения до конца.

Обучение модели происходило с помощью вызова функции fit\_generator с ниже указанными параметрами:

1.Количество итераций в эпохе определяются с помощью деления количества объектов в выборке на размер пакета(бэтча);

2. Количество эпох равно 45.

После проведения экспериментального исследования точность дообученной MobileNet V2 показала 59.63% на проверочном наборе и 85.32% на обучающем наборе, что не превышает точность, полученную в принятой базовой статье для 7 автомобильных зон. Обучение программы было остановлено из-за EarlyStopping.



*Рис.16 Точность дообученной MobileNet V2*

**Заключение**

Активная эксплуатация полуавтономных транспортных средств привлекла особое внимание к безопасности водителей. Увеличивающийся с каждым годом процент автокатастроф, причина которых - невнимательность водителя, беспрекословно должен быть снижен. Один из доступных и эффективных способов - автоматизация обработки направления взгляда водителя по его видеоизображению лица.

В настоящей курсовой работе исследовалась тема распознавания направления взгляда водителя на основе нейросетевого классификатора. В результате были выполнены следующие задачи:

1. Проведен аналитический обзор литературы для ознакомления с основными теоретическими понятиями данной темы. Рассмотрены и исследованы подходы(классические и приведенные на конкурсе EmotiW2020), основанные на машинном обучении, для осуществления цели автоматической и обобщенной обработки направления взгляда водителя.

2. Проведено обучение нейросетевой модели MobileNet V2 на наборе данных Driver Gaze in Wild(DGW) для классификации направления взгляда водителя. Показатель точности составляет 59.23%, что не превышает показатель точности в результатах статьи, принятой за базовую, при рассмотрении 7 автомобильных зон.

3. Проведен эксперимент по распознаванию направления взгляда направления водителя на основе нейросетевой модели MobileNet V2.

Список литературы

[1] M W Johns, A Tucker, R Chapman, K Crowley, and N Michael. Monitoring eye and eyelid movements by infrared reflectance oculography to measure drowsiness in drivers. Somnologie-Schlafforschung und Schlafmedizin, 11(4):234– 242, 2007.

[2]DA Robinson. A method of measuring eye movemnent using a scieral search coil in a magnetic field. IEEE Transaction on Bio-Medical Electron., pages 137–145, 1963.

[3] S Jha and C Busso. Probabilistic estimation of the gaze region of the driver using dense classification. In IEEE International Conference on Intelligent Transportation Systems, pages 697–702, 2018.

[4] S Jha and C Busso. Challenges in head pose estimation of drivers in naturalistic recordings using existing tools. In IEEE International Conference on Intelligent Transportation Systems, pages 1–6, 2017.

[5] Yunlong Feng, Gene Cheung, Wai-tian Tan, Patrick Le Callet, and Yusheng Ji. Low-cost eye gaze prediction system for interactive networked video streaming. IEEE Transactions on Multimedia, 15(8):1865–1879, 2013

[6] Cong Zhang, Qiyun He, Jiangchuan Liu, and Zhi Wang. Exploring viewer gazing patterns for touch-based mobile gamecasting. IEEE Transactions on Multimedia, 19(10):2333– 2344, 2017.

[7] E. D. Guestrin and M. Eizenman, “General theory of remote gaze estimation using the pupil center and corneal reflections.” IEEE Trans. Biomed. Engineering, vol. 53, no. 6, pp. 1124–1133, 2006.

[8] Tawari A., Trivedi M. M. Robust and continuous estimation of driver gaze zone by dynamic analysis of multiple face videos //2014 IEEE Intelligent Vehicles Symposium Proceedings. – IEEE, 2014. – С. 344-349.

[9] D. F. Dementhon and L. S. Davis, “Model-based object pose in 25 lines of code,” International journal of computer vision, vol. 15, no. 1-2, pp. 123–141, 1995

[10] Tawari A., Chen K. H., Trivedi M. M. Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation //17th International IEEE Conference on Intelligent Transportation Systems (ITSC). – IEEE, 2014. – С. 988-994.

[11] Ghosh S. et al. Speak2label: Using domain knowledge for creating a large scale driver gaze zone estimation dataset //Proceedings of the IEEE/CVF International Conference on Computer Vision. – 2021. – С. 2896-2905.

[12] A Krizhevsky, I Sutskever, and G E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[13] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[14] B T Phong. Illumination for computer generated pictures. Communications of the ACM, 18(6):311–317, 1975.

[15] I Bello, B Zoph, A Vaswani, J Shlens, and Q V Le. Attention augmented convolutional networks. In IEEE International Conference on Computer Vision, pages 3286–3295, 2019.

[16] G. M. Fitch, S. A. Soccolich, F. Guo, J. McClafferty, Y. Fang, R. L. Olson, M. A. Perez, R. J. Hanowski, J. M. Hankey, and T. A. Dingus, “The impact of hand-held and hands-free cell phone use on driving performance and safety-critical event risk,” Tech. Rep., 2013.

[17] T. Rueda-Domingo, P. Lardelli-Claret, J. de Dios Luna-del Castillo, J. J. Jimenez-Mole ´ on, M. Garcıa-Martın, and A. Bueno-Cavanillas, “The ´ influence of passengers on the risk of the driver causing a car collision in spain: Analysis of collisions from 1990 to 1999,” Accident Analysis & Prevention, vol. 36, no. 3, pp. 481–489, 2004.

[18]K. A. Braitman, N. K. Chaudhary, and A. T. McCartt, “Effect of passenger presence on older drivers risk of fatal crash involvement,” Traffic injury prevention, vol. 15, no. 5, pp. 451–456, 2014.

[19]BV S. S., Karthikeyan A. Computer vision based advanced driver assistance system algorithms with optimization techniques-a review //2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). – IEEE, 2018. – С. 821-829.

[20]Boby R. A. Kinematic identification of industrial robot using end-effector mounted monocular camera bypassing measurement of 3d pose //IEEE/ASME Transactions on Mechatronics. – 2021.

[21]Fortuna P., Nunes S. A survey on automatic detection of hate speech in text //ACM Computing Surveys (CSUR). – 2018. – Т. 51. – №. 4. – С. 1-30.

[22]Szegedy C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning //Thirty-first AAAI conference on artificial intelligence. – 2017.

[23]Hsiao T. Y. et al. Filter-based deep-compression with global average pooling for convolutional networks //Journal of Systems Architecture. – 2019. – Т. 95. – С. 9-18.

[24]Vora S., Rangesh A., Trivedi M. M. Driver gaze zone estimation using convolutional neural networks: A general framework and ablative analysis //IEEE Transactions on Intelligent Vehicles. – 2018. – Т. 3. – №. 3. – С. 254-265.

[25]Qassim H., Verma A., Feinzimer D. Compressed residual-VGG16 CNN model for big data places image recognition //2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). – IEEE, 2018. – С. 169-175.

[26]Iandola F. N. et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size //arXiv preprint arXiv:1602.07360. – 2016.

[27]Muhammad M. B., Yeasin M. Eigen-cam: Class activation map using principal components //2020 International Joint Conference on Neural Networks (IJCNN). – IEEE, 2020. – С. 1-7.

[28]Arora P. et al. An analytical study for Pneumonia Detection towards building an intelligent system using Image Data Generator //2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). – IEEE, 2021. – С. 1-4.

[29]Barry-Straume J. et al. An evaluation of training size impact on validation accuracy for optimized convolutional neural networks //SMU Data Science Review. – 2018. – Т. 1. – №. 4. – С. 12.

[30]Barry-Straume J. et al. An evaluation of training size impact on validation accuracy for optimized convolutional neural networks //SMU Data Science Review. – 2018. – Т. 1. – №. 4. – С. 12.

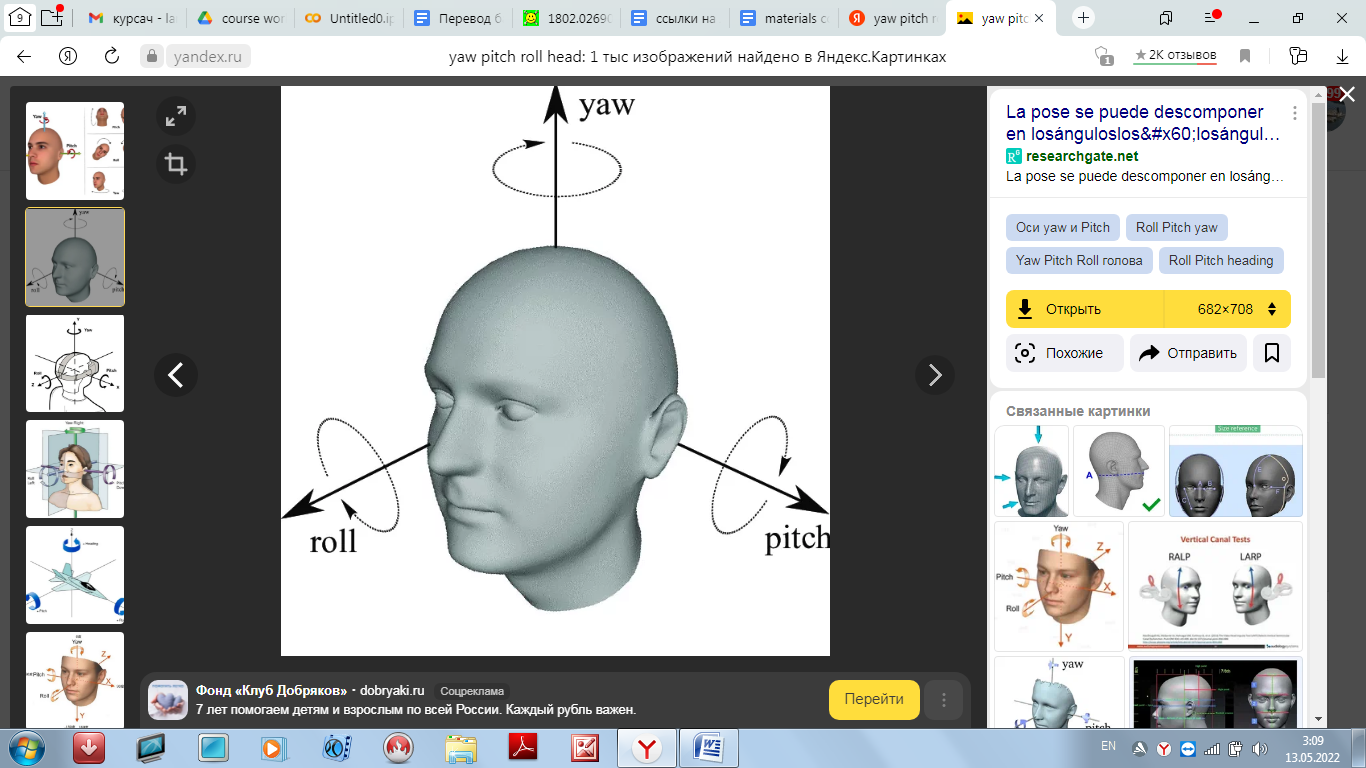
[31]Xiang J., Zhu G. Joint face detection and facial expression recognition with MTCNN //2017 4th international conference on information science and control engineering (ICISCE). – IEEE, 2017. – С. 424-427.

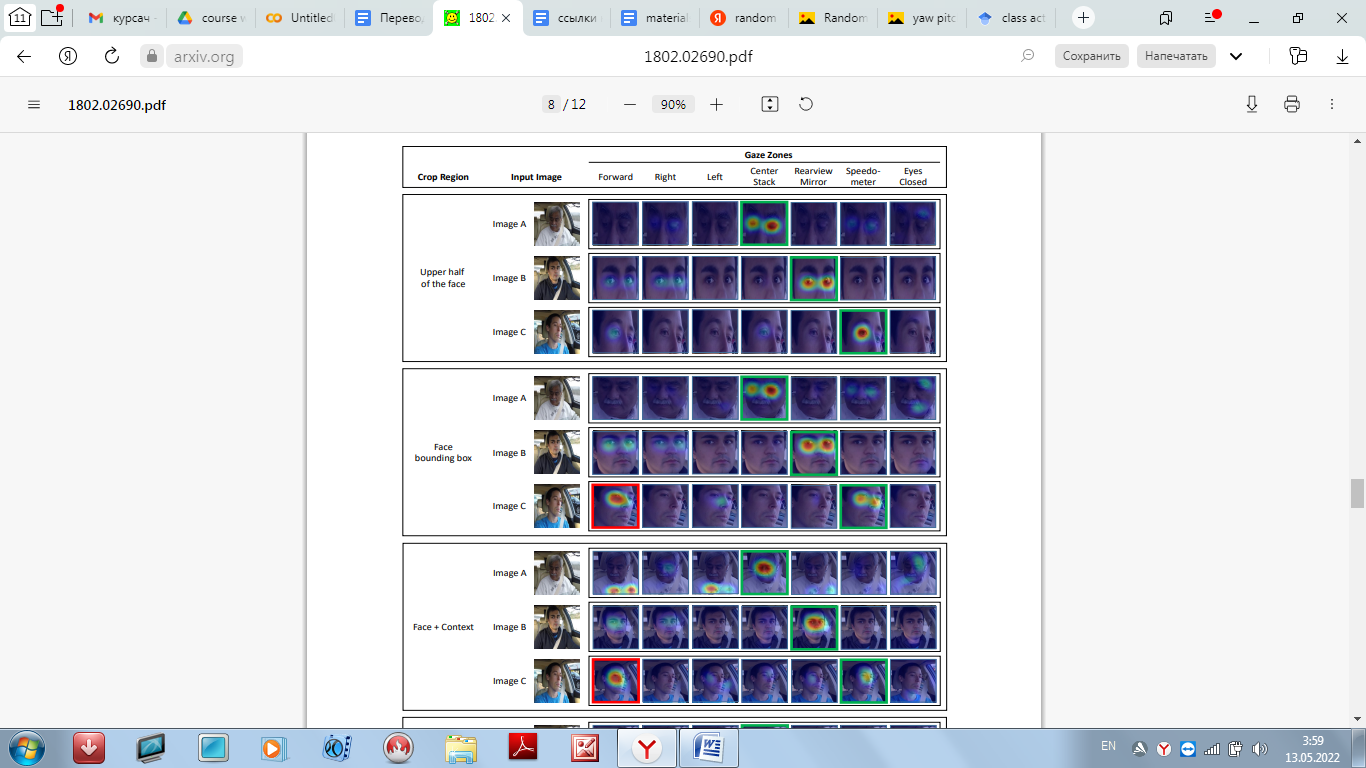
[32]Bradski G., Kaehler A. Learning OpenCV: Computer vision with the OpenCV library. – " O'Reilly Media, Inc.", 2008.

[33]Van Der Walt S., Colbert S. C., Varoquaux G. The NumPy array: a structure for efficient numerical computation //Computing in science & engineering. – 2011. – Т. 13. – №. 2. – С. 22-30.

[34]Zhang Z. Improved adam optimizer for deep neural networks //2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). – IEEE, 2018. – С. 1-2.

**Приложения**

*Приложение 1. Наглядный пример рыскания, тангажа и крена(yaw, pitch, roll) водителя.*

**

*Приложение 2. Пример визуализации работы Class activation maps*

*Приложение 3. Код*

from google.colab import drive

drive.mount('/content/gdrive', force\_remount=True)

!unzip "/content/gdrive/My Drive/course work/dataset\_part1.zip"

!unzip "/content/gdrive/My Drive/course work/dataset\_part2.zip"

train\_dir='dataset\_part1/'

validation\_dir='dataset\_part2/'

try:

  # Colab only

  %tensorflow\_version 2.x

except Exception:

    pass

import tensorflow as tf

print(tf.\_\_version\_\_)

import tensorflow.keras as keras

from tensorflow.keras.preprocessing import image

from tensorflow.keras.models import Sequential,Model

from tensorflow.keras.layers import Dense, Dropout, Flatten, Reshape,AveragePooling2D, SeparableConv2D,Conv2D, MaxPooling2D

from tensorflow.keras import backend as K

import matplotlib.pyplot as plt

import numpy as np

import os

!pip install mtcnn

from mtcnn.mtcnn import MTCNN

from mtcnn import MTCNN

import cv2

from google.colab.patches import cv2\_imshow

zone1\_dir\_valid=os.path.join(validation\_dir, 'zone1')

fnames\_valid = [os.path.join(zone1\_dir\_valid, fname) for fname in os.listdir(zone1\_dir\_valid)]

zone2\_dir\_valid=os.path.join(validation\_dir, 'zone2')

fnames\_valid.extend([os.path.join(zone2\_dir\_valid, fname) for fname in os.listdir(zone2\_dir\_valid)])

zone3\_dir\_valid=os.path.join(validation\_dir, 'zone3')

fnames\_valid.extend([os.path.join(zone3\_dir\_valid, fname) for fname in os.listdir(zone3\_dir\_valid)])

zone4\_dir\_valid=os.path.join(validation\_dir, 'zone4')

fnames\_valid.extend([os.path.join(zone4\_dir\_valid, fname) for fname in os.listdir(zone4\_dir\_valid)])

zone5\_dir\_valid=os.path.join(validation\_dir, 'zone5')

fnames\_valid.extend([os.path.join(zone5\_dir\_valid, fname) for fname in os.listdir(zone5\_dir\_valid)])

zone6\_dir\_valid=os.path.join(validation\_dir, 'zone6')

fnames\_valid.extend([os.path.join(zone6\_dir\_valid, fname) for fname in os.listdir(zone6\_dir\_valid)])

zone7\_dir\_valid=os.path.join(validation\_dir, 'zone7')

fnames\_valid.extend([os.path.join(zone7\_dir\_valid, fname) for fname in os.listdir(zone7\_dir\_valid)])

zone1\_train\_dir=os.path.join(train\_dir, 'zone1')

fnames\_train = [os.path.join(zone1\_train\_dir, fname) for fname in os.listdir(zone1\_train\_dir)]

zone2\_train\_dir=os.path.join(train\_dir, 'zone2')

fnames\_train.extend([os.path.join(zone2\_train\_dir, fname) for fname in os.listdir(zone2\_train\_dir)])

zone3\_train\_dir=os.path.join(train\_dir, 'zone3')

fnames\_train.extend([os.path.join(zone3\_train\_dir, fname) for fname in os.listdir(zone3\_train\_dir)])

zone4\_train\_dir=os.path.join(train\_dir, 'zone4')

fnames\_train.extend([os.path.join(zone4\_train\_dir, fname) for fname in os.listdir(zone4\_train\_dir)])

zone5\_train\_dir=os.path.join(train\_dir, 'zone5')

fnames\_train.extend([os.path.join(zone5\_train\_dir, fname) for fname in os.listdir(zone5\_train\_dir)])

zone6\_train\_dir=os.path.join(train\_dir, 'zone6')

fnames\_train.extend([os.path.join(zone6\_train\_dir, fname) for fname in os.listdir(zone6\_train\_dir)])

zone7\_train\_dir=os.path.join(train\_dir, 'zone7')

fnames\_train.extend([os.path.join(zone7\_train\_dir, fname) for fname in os.listdir(zone7\_train\_dir)])

detector = MTCNN()

k = 0

f = 0

for img\_path in fnames\_valid:

  img = cv2.imread(img\_path)

  output = detector.detect\_faces(img)

  print(output)                                                                 #[{},{}...{}]

  for i in output:

    x,y,width,height = i['box']

    left\_eyeX,left\_eyeY = i['keypoints']['left\_eye']

    right\_eyeX,right\_eyeY = i['keypoints']['right\_eye']

    noseX,noseY = i['keypoints']['nose']

    mouth\_leftX,mouth\_leftY = i['keypoints']['mouth\_left']

    mouth\_rightX,mouth\_rightY = i['keypoints']['mouth\_right']

    cv2.circle(img,center=(left\_eyeX,left\_eyeY),color=(0,0,0),thickness=0,radius=0)

    cv2.circle(img,center=(right\_eyeX,right\_eyeY),color=(0,0,0),thickness=0,radius=0)

    cv2.circle(img,center=(noseX,noseY),color=(0,0,0),thickness=0,radius=2)

    cv2.circle(img,center=(mouth\_leftX,mouth\_leftY),color=(0,0,0),thickness=0,radius=0)

    cv2.circle(img,center=(mouth\_rightX,mouth\_rightY),color=(0,0,0),thickness=0,radius=0)

    cv2.rectangle(img,pt1=(x,y),pt2=(x+width,y+height),color=(0,0,0),thickness=0)

  #cv2\_imshow(img)

  #img1 = img[y:y+height, x:x+width]

  if len(output) == 0:

    img1 = img[150:150+250, 180:180+200]

    cv2\_imshow(img1)

  else:

    img1 = img[y:y+height, x:x+width]

  cv2.imwrite(img\_path, img)

  #img.crop((y, y+height, x, x+width)).save(img\_path,quality=95)

  #cv2\_imshow(img)

  cv2.waitKey(0)

detector = MTCNN()

for img\_path in fnames\_train:

  img = cv2.imread(img\_path)

  output = detector.detect\_faces(img)

  print(output)                                                                 #[{},{}...{}]

  for i in output:

    x,y,width,height = i['box']

    left\_eyeX,left\_eyeY = i['keypoints']['left\_eye']

    right\_eyeX,right\_eyeY = i['keypoints']['right\_eye']

    noseX,noseY = i['keypoints']['nose']

    mouth\_leftX,mouth\_leftY = i['keypoints']['mouth\_left']

    mouth\_rightX,mouth\_rightY = i['keypoints']['mouth\_right']

    cv2.circle(img,center=(left\_eyeX,left\_eyeY),color=(0,0,0),thickness=0,radius=0)

    cv2.circle(img,center=(right\_eyeX,right\_eyeY),color=(0,0,0),thickness=0,radius=0)

    cv2.circle(img,center=(noseX,noseY),color=(0,0,0),thickness=0,radius=2)

    cv2.circle(img,center=(mouth\_leftX,mouth\_leftY),color=(0,0,0),thickness=0,radius=0)

    cv2.circle(img,center=(mouth\_rightX,mouth\_rightY),color=(0,0,0),thickness=0,radius=0)

    cv2.rectangle(img,pt1=(x,y),pt2=(x+width,y+height),color=(0,0,0),thickness=0)

  #cv2\_imshow(img)

  img1 = img[y:y+height, x:x+width]

  if len(output) == 0:

    img1 = img[150:150+250, 180:180+200]

    #cv2\_imshow(img1)

  else:

    img1 = img[y:y+height, x:x+width]

  cv2.imwrite(img\_path, img1)

  #img.crop((y, y+height, x, x+width)).save(img\_path,quality=95)

  #cv2\_imshow(img)

  cv2.waitKey(0)

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

BATCH\_SIZE=32

BEST\_MODEL\_FILE='best\_model.h5'

from tensorflow.keras.applications import mobilenet,mobilenet\_v2

INPUT\_SHAPE=224

net\_model=mobilenet\_v2

net\_model\_class=net\_model.MobileNetV2

model = net\_model\_class(weights='imagenet',input\_shape=(INPUT\_SHAPE,INPUT\_SHAPE,3))

def net():

    base\_model = net\_model\_class(weights='imagenet',input\_shape=(INPUT\_SHAPE,INPUT\_SHAPE,3),include\_top=False, pooling='avg', classes=7)

    x=base\_model.output

    x = Dense(7, activation='softmax', use\_bias=True)(x)

    model=Model(base\_model.inputs, x)

    return model,base\_model

train\_datagen = ImageDataGenerator(preprocessing\_function=net\_model.preprocess\_input)

valid\_datagen = ImageDataGenerator(preprocessing\_function=net\_model.preprocess\_input)

train\_generator = train\_datagen.flow\_from\_directory(

        train\_dir,

        target\_size=(INPUT\_SHAPE, INPUT\_SHAPE),

        batch\_size=BATCH\_SIZE,

        class\_mode='categorical')

validation\_generator = valid\_datagen.flow\_from\_directory(

        validation\_dir,

        target\_size=(INPUT\_SHAPE, INPUT\_SHAPE),

        batch\_size=BATCH\_SIZE,

        class\_mode='categorical')

model,base\_model=net()

for l in base\_model.layers:

    l.trainable=False

model.compile('adam', 'categorical\_crossentropy', metrics=['acc'])

#model.summary()

es=EarlyStopping(monitor='val\_acc',patience=6)

mc = ModelCheckpoint(BEST\_MODEL\_FILE, monitor='val\_acc', verbose=1, save\_best\_only=True)

history = model.fit\_generator(

      train\_generator,

      steps\_per\_epoch=train\_generator.samples//BATCH\_SIZE,

      epochs=45,

      callbacks=[es,mc],

      validation\_data=validation\_generator,

      validation\_steps=validation\_generator.samples//BATCH\_SIZE)