МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка приложения на основе нейронных сетей для оценки качества фотографий на документы

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-300.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Н.С. Силкина

Автор работы,
студент группы КЭ-401
_____ М.А. Караев

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студенту группы КЭ-401
Караеву Михаилу Александровичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка приложения на основе нейронных сетей для оценки качества фотографий на документы.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**
3.1. Face Analysis Technology Evaluation. [Электронный ресурс] URL: https://pages.nist.gov/frvt/html/frvt_quality.html (дата обращения: 05.01.2024 г.).
3.2. Papers with code. [Электронный ресурс]. URL: https://paperswithcode.com/ (дата обращения: 08.02.2024 г.).
4. **Перечень подлежащих разработке вопросов**
4.1. Изучить подходы к построению нейронной сети для оценки качества лица.
4.2. Построить и обучить нейронную сеть для оценки качества изображения.

4.3. На основе обученной нейронной сети реализовать и протестировать приложение на языке Python для оценки качества фотографий на документы.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП Н.С. Силкина

Задание принял к исполнению М.А. Караев

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Распознавание лиц – это способ идентификации или подтверждения личности человека по его лицу [1]. Систему распознавания лиц можно использовать для идентификации людей на фотографиях, видео или в режиме реального времени. Данная технология получила широкое применение в области безопасности, например, в банковской сфере, где доступ к счету может быть получен с помощью представления лица перед камерой, установленной в банкомате. Также распознавание лиц широко используется в мобильных устройствах, где одним из вариантов защиты от неправомерного доступа является вход по предъявлению лица владельца.

Одним из ключевых элементов системы распознавания лиц является база данных, хранящая информацию о зарегистрированных лицах. Важным компонентом является оборудование, которое включает в себя камеры, сенсоры и другие устройства, используемые для сбора данных.

При этом на вход нейронной сети может быть подана любая фотография, даже не содержащая лица, что может привести к ошибкам распознавания. Например, две сильно зашумленные фотографии могут быть ошибочно распознаны нейронной сетью как один и тот же человек. Для избежания таких ошибок перед подачей в систему распознавания лиц изображение должно получить оценку качества и не быть допущено к дальнейшему распознаванию в случае низкой оценки.

Национальный Институт Стандартов и Технологий (NIST) [2] определяет два вида качества: вектор и скаляр. Вектор представляет собой набор интегральных и дискретных оценок множества параметров изображения: зашумленность, углы поворота головы, шум, разнообразие динамического диапазона, перекрытость лица, однородность заднего фона, эмоции, наличие аксессуаров, дефекты и так далее, в сумме 27 оценок. Подробные требования описаны в стандарте ISO/IEC 29794-5 [3].

Назначение такого вектора – это фильтрация фотографий для увеличения эффективности при использовании человеком.

Скаляр представляет собой одно число в диапазоне от 0 до 1, суммирующее вышеописанные параметры и являющееся общей оценкой качества изображения. Понятие скаляра качества не имеет точной формулировки, а его главной целью является уменьшение ложно негативных срабатываний (FNMR) в системах распознавания лиц. Для сравнения алгоритмов оценки скаляра качества NIST описывает соответствующие метрики, заключающиеся в сравнении эффективности при отбрасывании изображений с низким качеством из наборов данных. Институт периодически проводит соревнования и обновляет рейтинг лучших алгоритмов [4]. В большинстве случаев, высокий скаляр качества означает изображение, которое будет хорошо распознано как человеком, так и машиной, и на практике такое изображение выглядит как фото на документы. Пример продемонстрирован на рисунке 1.

Рисунок 1 – Пример изображений с разным качеством

Постановка задачи

Целью выпускной квалификационной работы является разработка настольного приложения для оценки качества фотографии на документы на основе нейронных сетей. Для достижения поставленной цели необходимо решить следующие задачи:

1) изучить подходы к построению нейронной сети для оценки скаляра качества;

2) построить и обучить нейронную сеть;

3) реализовать и протестировать приложение на языке Python для оценки качества фотографии на документы.

Структура и содержание работы

Работа состоит из следующих разделов: введение, три главы, заключение и список литературы. Общий объем работы составляет 44 страницы, включая список литературы, который содержит 20 источников.

Во введении представлена общая информация о работе, ее цели и задачи. Описывается предметная область, которая будет исследована, а также обосновывается актуальность разработки данного приложения.

Первая глава посвящена анализу предметной области. Проводится обзор существующих статей, анализируются их особенности, достоинства и недостатки, выделяются ключевые принципы. Также рассматриваются наиболее популярные архитектуры нейронных сетей, функции потерь, наборы данных и метрики, применяемые в контексте данной задачи.

Во второй главе описывается обучение нейронной сети, выбор признаков, снятие метрик, проверка идей, поиск лучших гиперпараметров.

Третья глава посвящена проектированию и реализации приложения. В этом разделе работы разрабатывается структура программы, с помощью диаграмм показываются примеры использования, разрабатываются алгоритмы. Также здесь производится разработка кода, проектирование интерфейса.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Способы сравнения сети оценки качества

Задача распознавания лиц представляет собой задачу классификации. В таких задачах часто используются матрица ошибок [5], содержащая сравнение предсказанных и истинных ответов. Эта матрица представлена на рисунке 2.

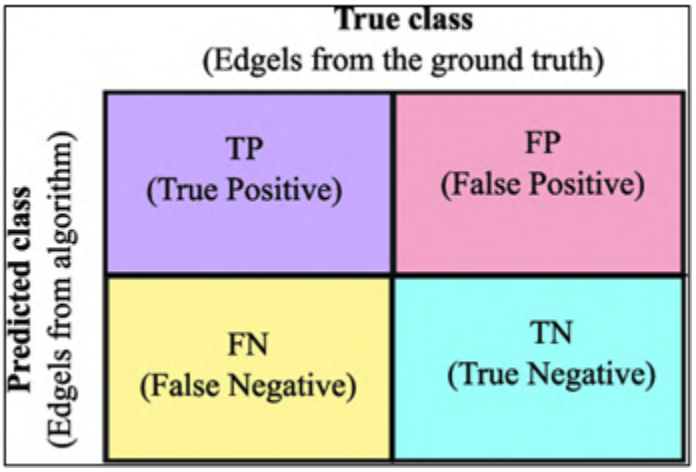

Рисунок 2 – Матрица ошибок

TP и TN представляют собой случаи, когда верно было предсказано наличие или не наличие на двух фотографиях одного и того же человека. FP и FN представляют собой ошибки — не верное предположении о совпадении или не совпадении двух людей. Система распознавания лиц выдает результат сходства между двумя лицами, находящийся в пределах от 0 до 1. В идеальном случае, между такими предсказаниями должна быть четкая грань, позволяющая выставить порог отсечения, выше которого находятся только верные ответы. На практике существует пересечение распределений ответов для позитивных классов и негативных классов, поэтому приходится выбирать порог, в зависимости от требований к системе. Распределения ответов представлены на рисунке 3.

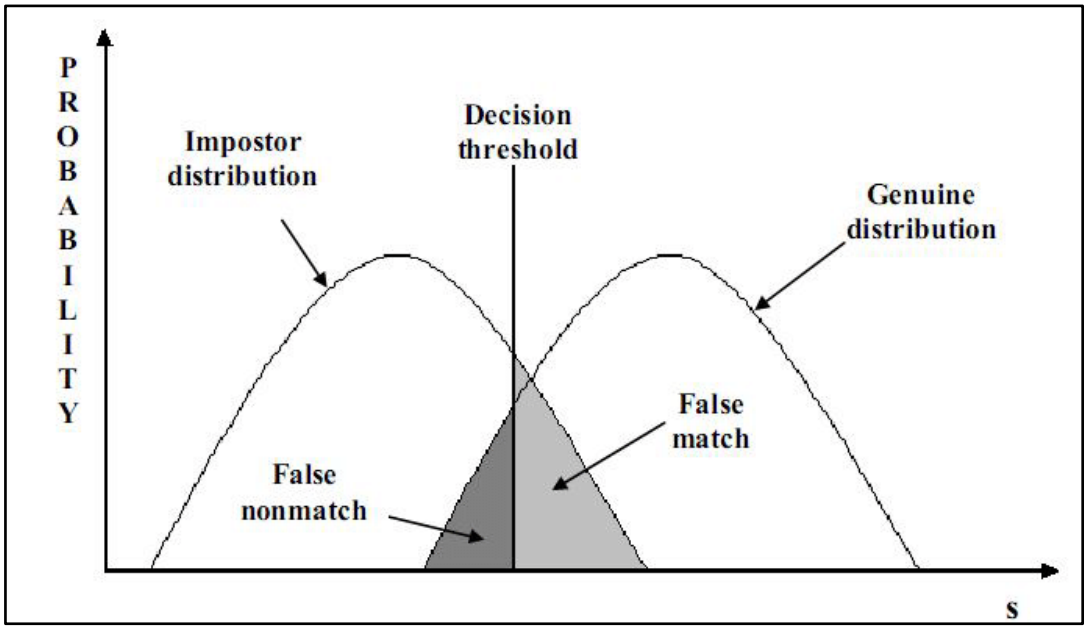

Рисунок 3 – Распределения ответов для своих и чужих классов

На рисунке 3 представлены результаты схожести между фото одного класса и между фото разных классов. Пересечения этих двух распределений представляют собой ошибки 1 и 2 родов. Ошибки 1 рода заключаются в отказе, при предъявлении двух лиц одного человека. Ошибки 2 рода заключаются в неправильном определении разных лиц, как лиц одного человека. Для тестирования нейронной сети оценки качества, нам потребуется метрика FNMR, обозначающая отношение процента ошибок 1 рода ко всем случаям сравнения двух фото одного человека. Обратной ей метрикой является более известная метрика recall. Формула метрики recall представлена на рисунке 4.

$$\text{Recall or TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} * 100$$

Рисунок 4 – Формула метрики recall

Метрика FNMR может быть вычислена заменой TP на FN в числителе метрики recall. Для сравнения сетей оценки качества, NIST вводит метрику efficiency (η). Ее формула представлена на рисунке 5.

$$\eta(r) = \frac{1}{r}\left(\frac{\mathrm{FNMR}(0) - \mathrm{FNMR}(r)}{\mathrm{FNMR}(0)}\right)$$

Рисунок 5 – Формула метрики efficiency

Основной идеей этой метрики заключается измерение изменения FNMR при выкидывании из датасета r% фотографий с худшим качеством. Для начала подсчитываются метрики на исходном наборе данных, затем берется порог, при котором FNMR равняется 0,01 и фиксируется соответствующий порог по результату сравнения. Затем из датасета последовательно отбрасывается 1%, 5% и 10% изображений с наименьшим качеством и происходит переподсчет FNMR при фиксированном пороге. Более высокое уменьшение FNMR будет обозначать более хорошую работу модели оценки качества. NIST не выкладывает в открытый доступ тестовый набор данных, на котором будут происходить сравнения.

1.2. Обзор работ по оценке качества

Исследовав статьи по тематике работы на сайте PapersWithCode, было выделено несколько работ. В задаче оценки качества может использоваться разметка, сгенерированная человеком, что может негативно сказаться на ее объективности, так как человек не знает нюансов работы систем распознавания лиц. Авторы аргументировали необходимость смены способа генерации разметки и предложили свой собственный подход. Он заключается в последовательном отключении случайных нейронов сети, вычислении вектора признаков получившейся подсетью и составления матрицы попарных расстояний между векторами признаков подсетей. Затем

такая матрица некоторым образом аккумулируется и приводится к диапазону от 0 до 1 для получения истинной разметки. Эта информация может быть использована для выяснения «устойчивых» участков векторного пространства, при попадании в которые наиболее вероятно изображение будет иметь высокое качество. Из данного исследования стоит вынести, что вектор признаков может использоваться для подсчета качества изображения. Также хочется отметить, что высокое значение нормы вектора признаков будет означать сильную выраженность признаков на изображении, а значит уменьшать вероятность ошибки при сравнении с другими классами.

Для тестирования авторы статьи использовали датасет LFW. Было предложено 2 варианта использования подхода: добавление дополнительного выхода для оценки качества в модель распознавания лиц, а также тренировка небольшой новой модели для оценки качества. Метрики модели представлены на рисунке 6.

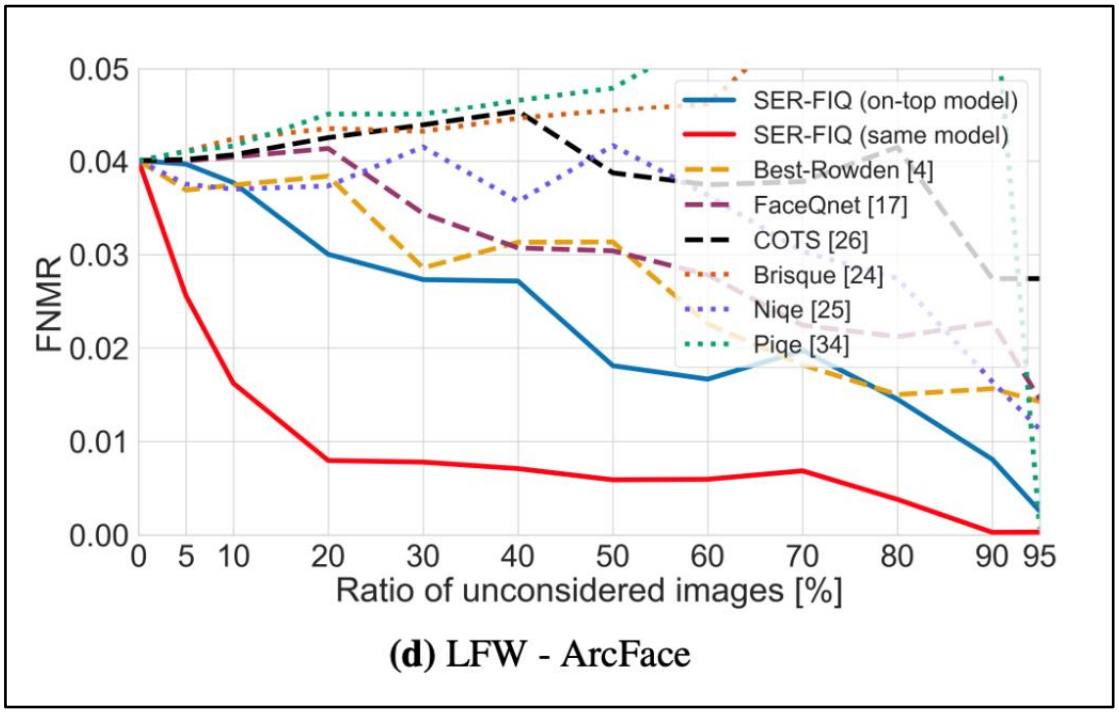

Рисунок 6 – Сравнение метрик SER-FIQ с устаревшими подходами

По оси x представлено количество отфильтрованных из набора данных фотографий с низким качеством. По оси y представлен FNMR при фиксированном пороге. Синей линией представлен метод, реализованный в виде отдельной нейронной сети. Красной линией представлен метод, при обучении оценки качества на целевой системе распознавания лиц. Другие линии представляют собой нерелевантные методы в данной задаче. В своей статье авторы использовали нейронную сеть ResNet [6], обученную с помощью функции потерь ArcFace [7] и опубликованную авторами в репозитории InsightFace [8].

Другой способ оценки качества фотографии лица представлен в статье SDD-FIQA [9]. Из этого исследования стоит вывести, что можно использовать в качестве разметки разницу между расстоянием до ближайшего своего и ближайшего чужого объекта внутри признакового пространства. Пример представлен на рисунке 7.

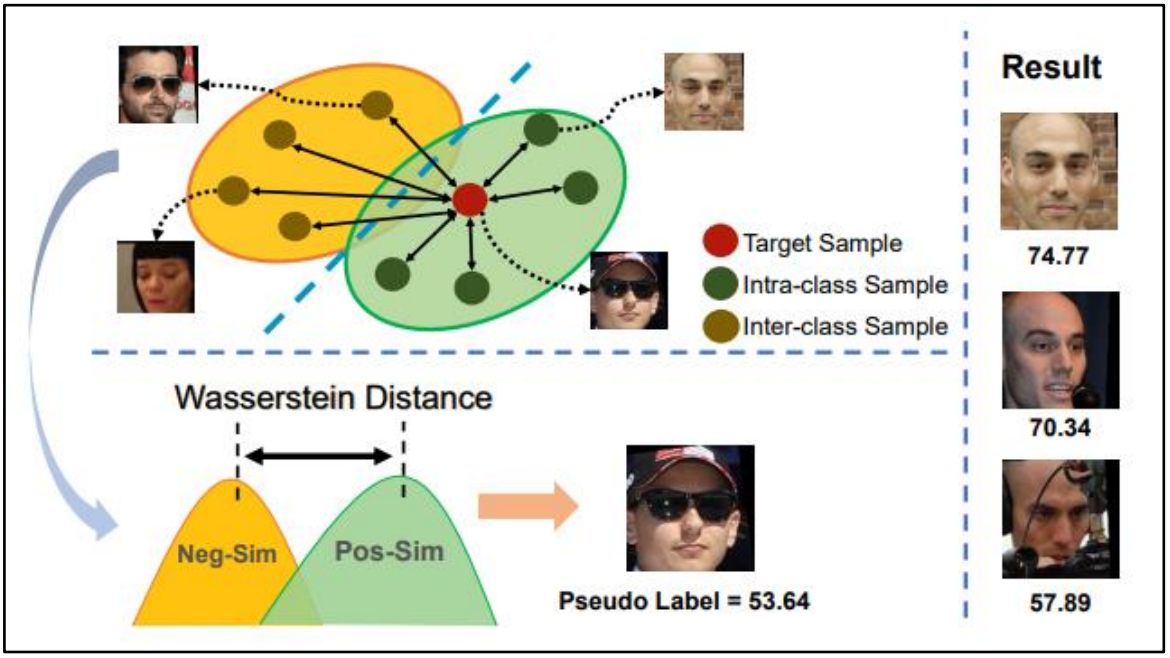

Рисунок 7 – Разметка обучающий данных методом SDD-FIQA

В статье [10] упоминается, что слой нормализации батча позволяет заучить распределение промежуточных признаков, представленных в обучении нейронной сети, что может быть полезно в данном исследовании.

BatchNorm – это мощная техника, которая помогает уменьшить ковариантный сдвиг между реальными данными и данными, на которых обучалась нейросеть. Она была впервые описана в 2015 году Сергеем Иоффе из Google [11]. К примеру, на практике это может означать приспособление сети к работе на чернокожих, даже если она обучалась только на белых людях. Такие слои ускоряют сходимость и улучшают стабильность обучения нейронной сети, работая в паре со сверточными слоями.

Один нейрон слоя принимает на вход выходы одного канала сверточного слоя, и нормализует его, используя среднее и дисперсию, накопленные во время обучения. Таким образом совершается попытка уменьшить расхождение между распределением извлеченных признаков и признаков, заученных во время обучения нейронной сетью. На рисунке 8 показано, какие размерности данных модифицируются одним нейроном нормализации батча. Преобразования выполняются согласно формулам, представленным на рисунке 9, где x_i – это промежуточные признаки, извлеченные нейросетью, μ_b и ${\sigma_b}^2$ – среднее и дисперсия всей обучающей выборки, γ – матрица весов, β – смещение, y_i — выход слоя после всех преобразований. К знаменателю прибавляется малый эпсилон для избежания деления на ноль.

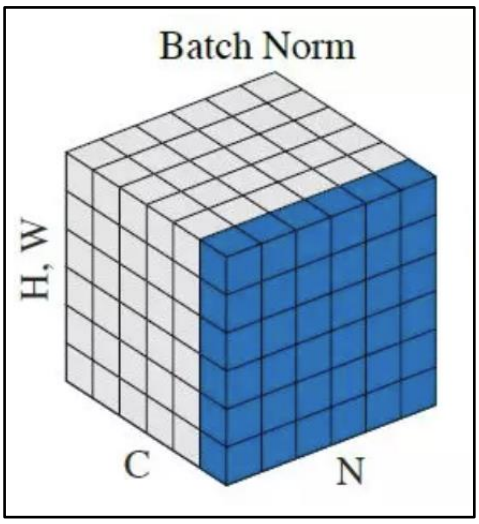

Рисунок 8 – Размерности, преобразуемые BatchNorm

Математически смысл этого слоя заключается в приведении признаков к нулевому среднему и единичной дисперсии и последующей трансформации этих данных через линейный слой со смещением (называемым bias в иностранной литературе).

$$\text{Normalizing} \quad x'_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$\text{Scaling and shifting} \quad y_i = \gamma x'_i + \beta$$

Рисунок 9 – Описание работы слоя BatchNorm

Слой BatchNorm или его вариации присутствуют во всех современных подходах к построению нейронных сетей, таких как ResNet [6], Inception [12], MobileNet [13], ShuffleNet [14].

В своей статье [10] исследователи из Пекинского университета описывают решение задачи доменной адаптации с помощью таких слоев. Эта задача подразумевает собой избавление от потери качества при небольшом изменении предметной области, к примеру при обработке изображений с людьми разных рас, не представленных в тренировочной выборке, либо в условиях освещенности, также отличающейся от исходных данных. Немного изменив среднее и дисперсию слоев нормализации батча под новый домен, можно добиться приспособления нейронной сети к новым данным. Из данного исследования стоит вынести гипотезу авторов, в которой подразумевается, что слои нормализации батча хранят данные о признаках, извлеченных и обрабатываемых во время обучения, а значит по ним можно сделать вывод о данных, которые «знает» нейронная сеть.

1.3. Модели нейронной сети

Для решения задач, поставленных в данной работе, требуется предварительно исследовать релевантные подходы к построению

нейронных сетей для распознавания лиц. Такие сети состоят из двух частей: backbone (часть сети, отвечающая за извлечение признаков из исходного изображения) и функция потерь, с помощью которой происходит оценка текущей ошибки нейронной сети и последующее извлечение градиентов для оптимизации.

Backbone представляет собой основную часть сети. Часто практикуется подход дообучения, называемого fine-tuning в английской литературе. Если сеть заранее была обучена на другой задаче, можно изменить только последние ее слои и значительно ускорить процесс обучения. Это происходит за счет того, что во время обучения на другом наборе данных сеть уже извлекла какие-то свертки, которые могут хорошо себя показать и в других задачах. В данной работе подразумевается использование готовой нейронной сети для распознавания лиц. Исследовав релевантные исследования, удалось выявить наиболее релевантные подходы к построению backbone. Все они базируются на идее сверточных нейронных сетей. Также важную роль играет функция потерь. Хорошо подобранная функция потерь позволяет улучшить качество предсказаний нейронной сети. Основой сверточной нейронной сети является проход сверткой по изображению, что позволяет получить карту извлеченных признаков. Пример представлен на рисунке 10.

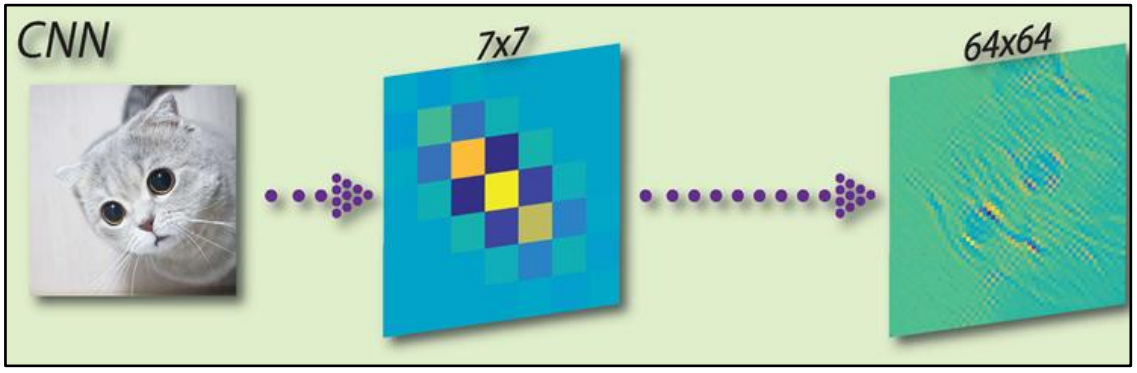

Рисунок 10 – Пример карт признаков изображения

1.3.1. Функции потерь

Исследуя релевантные методы решения задачи удалось выделить две функции потерь: Arcface [7] и Cosface [15]. Для сравнения приведена функция потерь Softmax [16], стандартно используемая в задачах классификации.

SoftMax

Функция потерь SoftMax [16] используется при решении большинства задач классификации. Эта функция переводит поданный на вход ненормированный вектор в вектор такой же размерности с единичной нормой. Количество элементов вектора равняется количеству классов. Индекс максимального элемента берется за предсказание сетью класса. Пример работы этой функции потерь показан на рисунке 11, где z_i означает выход из слоя классификации для класса с индексом i, s_i означает выход для класса с индексом i.

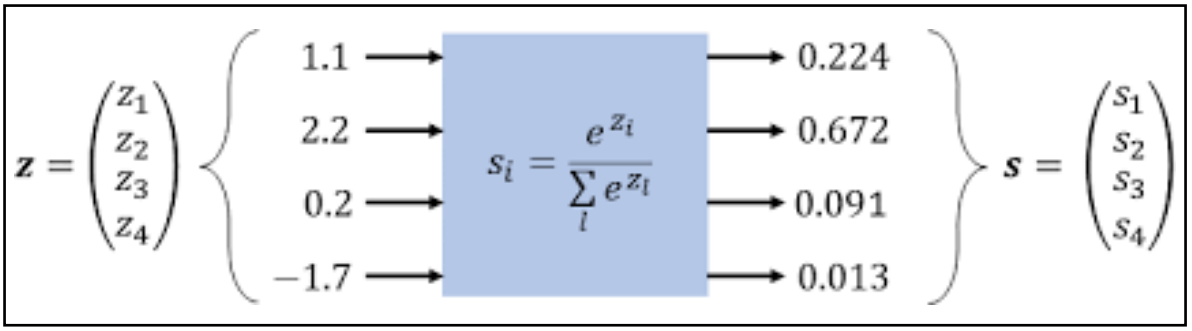

Рисунок 11 – Демонстрация работы функции потерь SoftMax

В синем квадрате продемонстрирована суть работы SoftMax, которая заключается в возведении экспоненты в степень входного нейрона класса и деление на сумму такого преобразования для всех нейронов. Данная функция позволяет перевести ненормированные числа в вероятности принадлежности к каждому классу, которые в сумме дают единицу.

ArcFace

Функция потерь Arcface [7] представляет собой функцию Softmax с небольшими изменениями: она имеет два гиперпараметра m и s. Функция

потерь увеличивает угол между векторами признаков разных классов, стараясь отделить каждый класс на отступ m, который является гиперпараметром сети, и размещает все признаки на сфере радиусом s, который также является подбираемым параметром. Формула ArcFace представлена на рисунке 12, где θ_{yi} это угол между центроидом целевого класса в векторном пространстве, а θ_j это углы всех других классов, m представляет собой зазор, который функция потерь пытается сохранить между центрами каждого класса, а s – размер гиперсферы на который проецируется пространство признаков после выхода из сети.

$$L_3 = -\log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos\theta_j}}.$$

Рисунок 12 – Функция потерь ArcFace

На рисунке 13 представлено распределение признаков при размерности вектора признака равной двум (для наглядности).

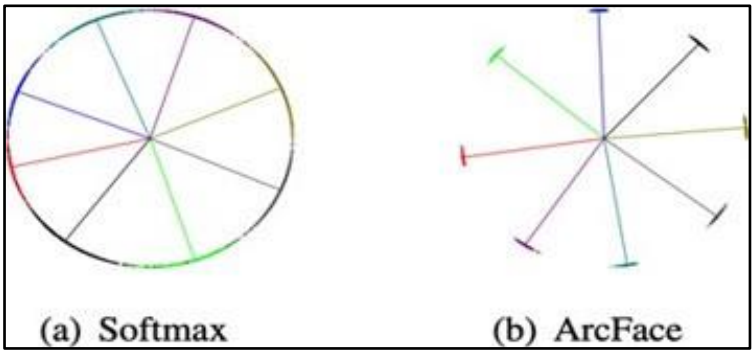

Рисунок 13 – Сравнение размещения признаков на гиперсфере

Точки одного цвета представляют собой объекты одного класса, и как можно заметить, в случае с Softmax признаки распределены равномерно по сфере. Между классами нет четких границ, что означает, что краевые объекты легко могут быть спутаны между собой. За счет использования в ArcFace зазора, классы хорошо сгруппированы и имеют низкую

внутриклассовую дисперсию, что позволит значительно уменьшить ошибки классификации.

Cosface

Функция потерь CosFace [15] так же, как и ArcFace, является модифицированной версией SoftMax. Единственным отличием является то, что сначала происходит подсчет косинуса, а затем его изменение на требуемый зазор. Демонстрация работы этой функции потерь представлена на рисунке 14.

$$L_{lmc} = \frac{1}{N}\sum_{i} -\log \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s\cos(\theta_{j,i})}},$$

Рисунок 14 – Функция потерь CosFace

1.3.2. Экстракторы признаков

VGG19

VGG19 – модель сверточной нейронной сети, предложенная Симоняном и Циссерманом из Оксфордского университета в своей статье [17]. Модель достигает точности 92,7% при тестировании на наборе данных ImageNet в задаче распознавания объектов на изображении. Содержит в себе большое количество сверточных слоев и весов. На рисунке 15 представлена архитектура VGG19.

ResNet

ResNet (ResidualNetwork) – это глубокая нейронная сеть, которая была представлена в 2015 году на конференции CVPR (Computer Visionand Pattern Recognition) компанией MicrosoftResearch. Она является одной из самых успешных архитектур для решения задач компьютерного зрения, таких как классификация изображений, детектирование объектов и сегментация изображений.

Рисунок 15 – Архитектура сети VGG19

Главной особенностью ResNet является использование residual блоков, которые позволяют обеспечить более эффективное обучение глубоких нейронных сетей за счет решения проблемы затухающих градиентов, что позволяет создавать сети большей глубины. Resnet является продолжением идеи VGG, но при этом решающая ее проблемы и дающая более высокие результаты. Сеть может содержать разное число блоков, соответственно можно найти баланс между качеством и затратами ресурсов. На рисунке 16 представлена структура сети ResNet.

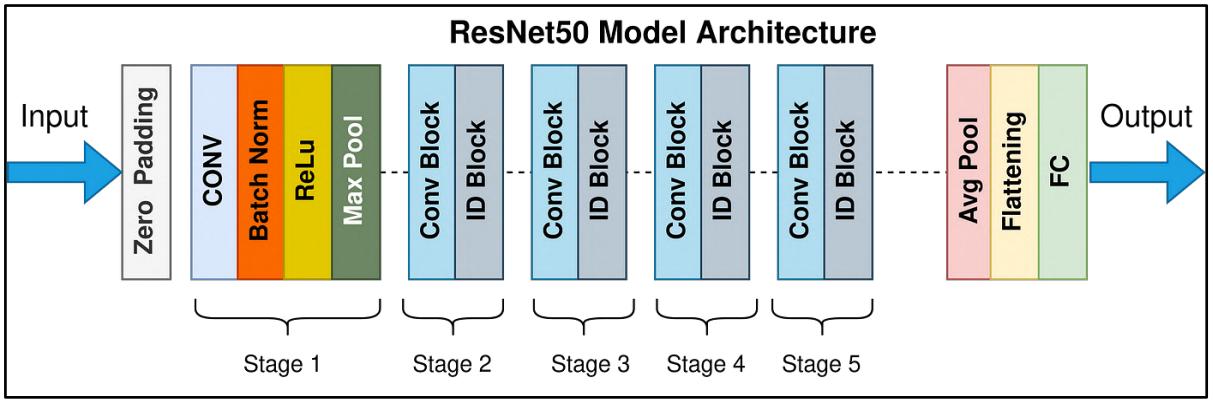

Рисунок 16 – Архитектура сети ResNet50

Сеть состоит из последовательных блоков. С продвижением в глубь сеть извлекает большее количество признаков (на картинке это представлено разным цветов блоков), но при этом сокращая пространственные размерности карты признаков. На выходе сети находится слой AveragePooling, извлекающий одно число из каждой карты признаков, позволяющее сети сохранять одинаковый размер выхода для изображений с

разным разрешением. Стрелками на рисунке представлены skip connections, которые конкатенируют выходы предыдущих блоков с последующими, что позволяет градиентам из конца не затухать при проходе к началу и эффективно обновлять веса для всей сети, а не только ее конечной части.

1.4. Обучающие наборы данных

Для обучения нейронной сети необходим набор изображений, на которых человек заранее идентифицирован. При исследовании предметной области удалось выявить несколько популярных наборов данных.

Large-scale CelebFaces Attributes (CelebA)

Набор состоит из 10 177 разных людей (классов), 202 599 изображений, при этом имея для каждого 5 размеченных антропометрических точек лица (2 глаза, нос, 2 угла рта), 40 атрибутов (эмоции, присутствие аксессуаров, тип волос и т.д.). На рисунке 17 представлены примеры изображений из набора данных CelebA.

Рисунок 17 – Примеры изображений из набора данных CelebA

MS-Celeb-1M

Состоит из 100 000 разных людей (классов), 8 200 000 изображений. Набор данных представляет собой изображения знаменитостей, собранные из интернета. На рисунке 18 представлены примеры изображений из набора данных MS1M.

Рисунок 18 – Примеры изображений из набора данных MS1M

VGGFace2

Состоит из 9 131 разных людей (классов), 3 310 000 изображений.

Набор данных представляет собой изображения людей, собранные из интернета. В наборе присутствует большое разнообразие возраста, освещенности, расы и сфер деятельности человека. На рисунке 19 представлены примеры изображений из набора данных VGGFace2.

Рисунок 19 – Примеры изображений из набора данных VGGFace2

Glint360k

Состоит из 17 091 657 изображений, 360 232 разных людей (классов). Большой датасет изображений из интернета с качественным разбиением по классам. В наборе присутствует сильная вариабельность всех возможных характеристик. На рисунке 20 представлены примеры изображений из набора данных Glint360k.

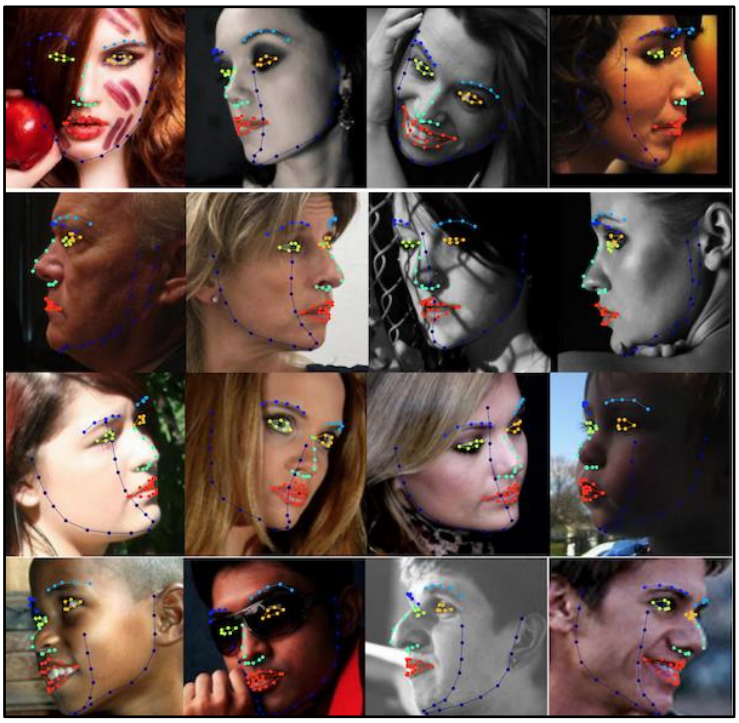

Рисунок 20 – Примеры изображений из набора данных Glint360k

Labeled Faces in the Wild (LFW)

Состоит из 13 233 изображений, 5 749 разных людей (классов). Этот датасет является небольшим и легким, часто используется разработчиками для базовых тестов. На рисунке 21 представлены примеры изображений из набора данных LFW.

Рисунок 21 – Примеры изображений из набора данных LFW

Результаты обзора

В ходе обзора литературы был выявлен набор инструментальных средств для реализации задачи, наиболее полно удовлетворяющий требованиямк подобным системам.

Система оценки качества лица строится на базе системы распознавания лиц. Для построения системы распознавания лиц необходима нейронная сеть, обрабатывающая исходное изображение и преобразующая его в вектор признаков.

Исследовав публикации со схожей тематикой, было выявлено, что в качестве истинной разметки может быть использована схожесть с центроидом внутри векторного пространства шаблонов. Также была выявлена необходимость в проведении исследования о корреляции между

порчей признаков изображения и изменением бегущих статистик слоев нормализации батча в исходной нейронной сети для распознавания лиц, ложащейся в основу оценщика качества.

Было принято решение производить обучение на наборе данных Glint360k, так как распознаватель лиц, взятый за основу, обучался на этом наборе, что позволяет упростить процесс получения истинной разметки.

В качестве первоначального тестируемого экстрактора признаков был выбран ResNet100, так как эта модель имеет оптимальное количество слоев и позволяет создать сбалансированную по качеству и скорости сеть, которую можно использовать без помощи графического ускорителя.

Для тестирования был выбран набор Labeled Faces in the Wild в следствие его активного использования в работах, смежных к тематике исследования, небольшого размера, а значит и быстрого снятия метрик, а также небольшой сложности, что позволяет оценить насколько целевая модель может выбрать изображение с плохим качеством, так как на сложном наборе данных это сделать значительно легче даже при выборе случайных изображений.

2. ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ

Подход, представленный в этой работе, представляет оценку качества с помощью измерения сходства между признаками, заученными во время обучения и признаками, извлеченными из обрабатываемого изображения.

2.1. Отбор признаков

Для сравнения признаков предполагается использовать дивергенцию Кульбака-Лейблера. Формула представлена на рисунке 22.

$$KL(A||B) = \sum_{i=1}^{d} log\frac{\sigma_{Bi}}{\sigma_{Ai}} + \frac{\sigma_{Ai}^2 + (\mu_{Ai} - \mu_{Bi})^2}{2\sigma_{Bi}^2} - \frac{1}{2}$$

Рисунок 22 – Формула дивергенции Кульбака-Лейблера

На рисунке 23 ${\sigma_{Ai}}^2$ и ${\sigma_{Bi}}^2$ – дисперсии сравниваемых распределений, ${\mu_{Ai}}^2$ и ${\mu_{Bi}}^2$ – математические ожидания. Так как традиционный вариант дивергенции является не симметричным, было принято использовать симметричный вариант, переписав ее в другом виде, представленном на рисунке 23.

$$D_{sKL}(F||BN) = \frac{1}{C}\sum_{c=1}^{C}\left[D_{KL}(F_c||BN_c) + D_{KL}(BN_c||F_c)\right]$$

Рисунок 23 – Формула симметричной дивергенции Кульбака-Лейблера

Дивергенция Кульбака-Лейблера позволяет определить сходство между двумя распределениями, то есть между заученными признаками и извлеченными. Высокое значение будет означать, что извлеченные признаки не похожи на обучающие, а значит, велика ошибка на них, так как сеть ранее не работала с ними. Во время обучения, сеть не видела, например, изображения животных или предметов, а значит такие изображения автоматически будут иметь высокую дивергенцию или же низкое качество.

Из зашумленных, обрезанных, перекрытых лиц средний отлик будет ниже обычного, так как не все признаки представлены на таких изображениях, а значит распределение также будет отличаться. Из изображений лиц не имеющих дефектов извлекутся признаки, более всего похожие на заученные, а значит значение дивергенции будет низкой. Такие изображения имеют наибольшую ценность, так как сеть умеет работать с ними и соответственно вероятность ошибки будет меньше.

Для выбора признаков, подаваемых в регрессор качества были протестированы разные варианты. Для визуализации дивергенции Кульбака-Лейблера с каждого канала каждого слоя нейронной сети были извлечены, а затем усреднены поканально. Высокая дивергенция означала высокое отличие тренировочного домена от обучающего, а значит большую вероятность ошибки сети. Визуализация приведена на рисунке 24.

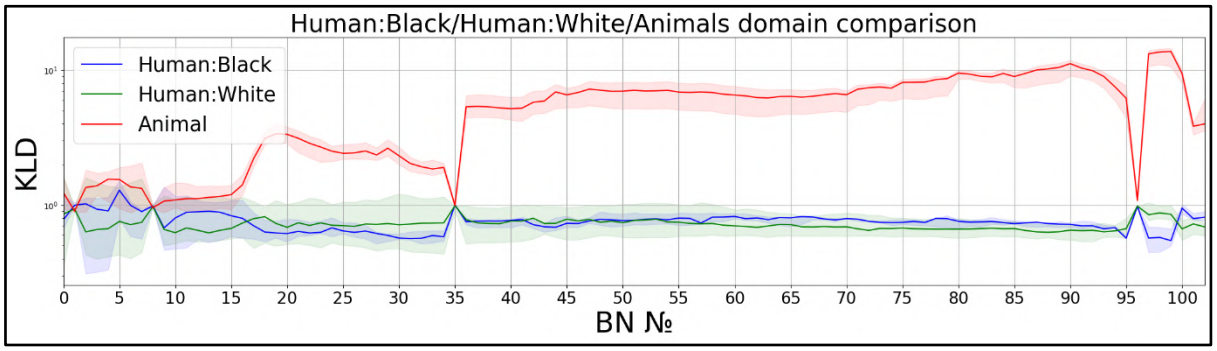

Рисунок 24 – Сравнение средней послойной KLD для изображений африканцев, европейцев и животных

Представители европеоидной расы и африканской расы имеют низкую дивергенцию, так как были заучены сетью во время обучения. Изображения животных не присутствовали в обучающей выборке, а значит сеть не имеет заученных сверток для корректной классификации таких изображений. На рисунках 25, 26, 27 приведены графики изменения KLD при искусственном добавлении дефектов на изображения: размытия, яркости и поворота.

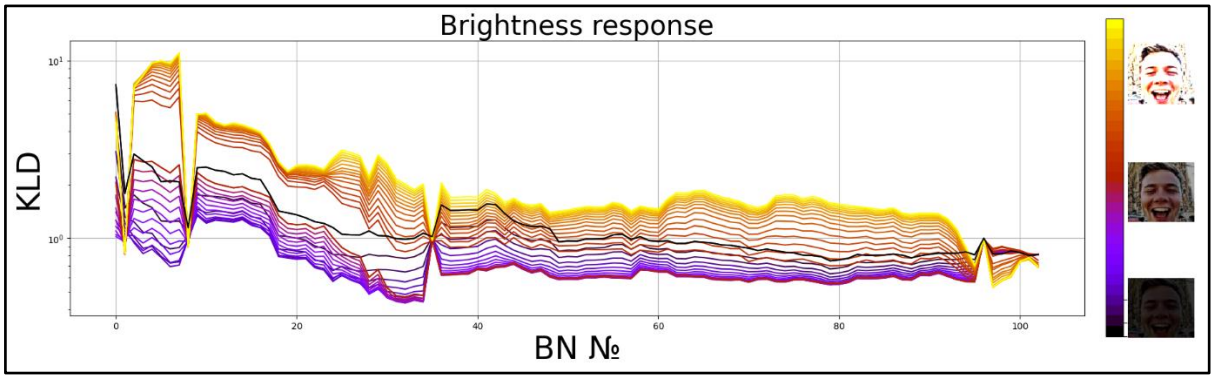

Рисунок 25 – Сравнение средней послойной KLD для изображений с разной яркостью

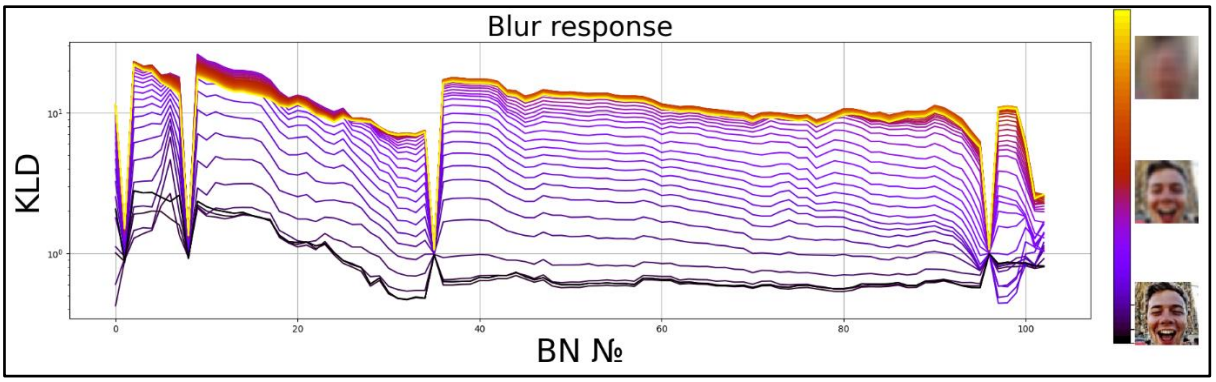

Рисунок 26 – Сравнение средней послойной KLD для изображений с разным размытием

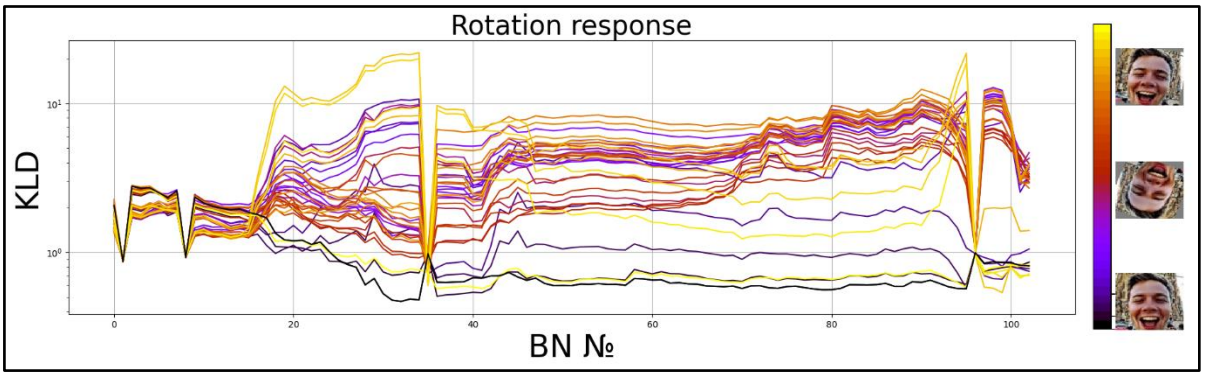

Рисунок 27 – Сравнение средней послойной KLD для изображений с разным углом поворота

Как можно заметить, увеличение дефектов на изображении приводит к росту дивергенции. Некоторые каналы нейронной сети имеют нулевую

дисперсию, что приводит к численной неустойчивости в формуле дивергенции Кульбака-Лейблера, а также такие каналы не несут в себе информации, так как не позволяют оценить различие доменов. В связи с этим такие каналы были отфильтрованы и убраны из построения регрессора, что позволило сократить количество признаков и сложность обучения.

В связи с разным порядком извлеченных дивергенций, было принято подсчитать 75 квантиль для каждого канала каждого слоя сети и использовать его для нормализации признаков. Значения больше единицы будут означать, что изображение не похоже на 75% данных обучающей выборки.

С другой стороны, так как в сети присутствует большое количество каналов, обучение регрессора может занять большое время и быть проблематичным. Также каналы обладают большой корреляцией, так как признаки в более поздних слоях являются нелинейной комбинацией признаков из более ранних слоев. Для борьбы с данным явлением был протестирован вариант, в котором дивергенции усреднялись послойно, таким же образом как на визуализациях дефектов.

Также выдвигалась гипотеза о том, что регрессор может построить оценку качества только на основе средних и дисперсий каналов входящего изображения, выделив особые значения признаков с низким качеством. Такие методы показали более высокие метрики при построении на основе нейронных сетей с небольшим количеством слоев. В случае работы с глубокими сетями, лучше всего показали себя методы, построенные на основе взятия поканальной дивергенции Кульбака-Лейблера. Усреднение по слоям ожидаемо показала значительное уменьшение метрик, но дала прирост в скорости работы и обучения.

Для регрессии использовалась библиотека CatBoost [20], разработанная компанией Яндекс. Она содержит средства для борьбы с линейной зависимостью признаков, что может улучшить метрики.

2.2. Нейронная сеть QAA

На вход нейронной сети приходят признаки KLD, извлеченные при обработке фотографии компонентом Face Recognizer.

В качестве Face Recognizer взята нейронная сеть с архитектурой ResNet100, обученная для распознавания лиц на датасете Glint360k, находящаяся в свободном доступе в репозитории InsightFace.

Для обучения и использования нейронных сетей используются пакеты torch и torchvision. Для промежуточной обработки фотографий и удобной работы с массивами используются модули opencv и numpy. Для создания аугментаций, улучшающих метрики нейронной сети и скорость сходимости, была использована библиотека albumentations.

Через сеть были пропущены изображения из набора данных, а также предоставлена информация о принадлежности изображения к конкретному человеку и центроид соответствующего человека в векторном пространстве. Данный цикл был выполнен 10 раз. Изображения подавались пачками по 256 за раз (batchsize был равен 256). Для изменения весов нейронной сети был использован метод адаптивного моментного градиентного спуска (Adam). Предварительно все изображения были выровнены по точкам глаз, носа и рта и выполнено приведение разрешения к размеру 112 на 112 пикселей.

Веса сверточных слоев сети представляют собой фильтры или свертки, подобные каскадам Хаара [18], но, в отличие от них, такие признаки были получены при обучении нейронной сети. Эти фильтры используются для прохода по всему изображению на первом слое для составления первоначальной карты признаков, содержащей низкоуровневые признаки изображения, а затем для составления более высоких признаков при дальнейшем проходе по сети. Высокое значение отклика таких фильтров будет означать, что сети удалось найти признаки, которая она составила во время обучения. Низкое значение отклика будет означать, что признаки на

обрабатываемом изображении отсутствуют и, возможно, на вход сети идет изображение, сильно отличающаяся от тренировочных данных.

Модель для распознавания лиц, на основе которой будет обучаться нейронная сеть QAA, была обучена с помощью функции потерь ArcFace. Реализация этого метода подразумевает, что во время обучения был сохранен центроид внутри векторного пространства для каждого класса. Для разметки обучающих данных будет использоваться косинусная схожесть между эмбеддингом картинки и центроидом соответствующего класса.

Также для расширения обучающего набора и в следствие этого большей генерализации финальной модели, были проведены тесты с добавлением аугментаций на тренировочные модели. Были сделаны тесты с добавлением поворота, затемнения и размытия. Добавление аугментаций привело к резкому падению финального качества. Причиной такого эффекта мог стать неаккуратный выбор гиперпараметров и, в следствие этого, сильное смещение распределения правильных ответов, а также сильный domain-shift между обучающими данными и тестовыми данными.

2.3. Сравнение метрик QAA

Для оценки работы сети использовалась методика, описанная национальным институтом стандартов и технологий. В качестве тестового набора данных был выбран LFW. Для начала были подсчитаны метрики модели распознавания лиц на тестовом наборе данных. Затем был зафиксирован порог, при котором FNMR равен 0,01. Далее из набора были откинуты 1%, 5% и 10% изображений с худшим качеством, присвоенным тестируемым методом. Был измерен FNMR на том же пороге. Большее уменьшение FNMR будет обозначать более хорошую работу модели оценки качества.

Данный набор данных является относительно легким, поэтому является показательным в рамках данной задачи. Случайное выбрасывание изображений с большой вероятностью бы отбросило хорошо распознаваемое фото, поэтому проверка на этом датасете хорошо демонстрирует избирательность тестируемого алгоритма качества.

Обучение и тестирование проводилось на компьютере со следующими характеристиками: процессор Intel Core i7-7700k, видеокарта Nvidia GeForce RTX 3060, 16 гигабайт оперативной памяти. Для тестирования проверялось время выполнения на одном ядре центрального процессора. В ходе экспериментов удалось выделить несколько нейронных сетей, удовлетворяющих поставленным требованиям. Результаты приведены в таблице 1.

Таблица 1– Сравнение метрик обученных нейронных сетей

Описание метода	**Efficiency 1%**	**Efficiency 5%**	**Efficiency 10%**	**Время выполнения, мс**
ResNet18 среднее + дисперсия cat-boost	10,7491	6,8820	4,3800	**20**
ResNet18 KLD поканальное cat-boost	10,7670	8,3167	**6,5721**	60
ResNet100 норма эмбеддинга	17,1890	9,0890	4,4490	50
ResNet200 среднее + дисперсия (10 последних слоев) catboost	**17,7337**	**9,2522**	5,9580	160

В таблице представлены лучшие метрики, которые удалось получить для каждого метода путем варьирования гиперпараметров. Все обученные нейронные сети укладываются в требование иметь более 10% эффективности при 1% проценте фильтрации. Три из четырех методов укладываются в требование по времени выполнения. Примечательным является факт, что норма эмбеддинга может использоваться в качестве показателя качества, при этом являясь бесплатной по времени, в случае вычисления шаблона для другой цели, например, для распознавания лица.

3. РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1. Анализ требований к проектируемой системе

В ходе анализа предметной области были выявлены следующие функциональные требования к системе:

1) система должна оценивать скаляр качества фотографии лица;

2) система должна оценивать однородность заднего фона на фотографии лица;

3) система должна иметь интерфейс для взаимодействия с пользователем.

В ходе анализа предметной области были выявлены следующие нефункциональные требования к системе:

1) система должна быть реализована на языке программирования Python;

2) система должна обрабатывать изображение не более чем за 100 мс на CPU;

3) нейронная сеть должна быть обучена с использованием библиотеки torch;

4) нейронная сеть должна иметь метрику Efficiency не ниже 10% при удалении 1% худших изображений.

Таким образом в системе представлен только один пользователь, который может выполнять все доступные функции.

3.2. Архитектура системы

В основе системы лежит нейронная сеть, которая извлекает признаки лица из его фотографии. Когда пользователь подает фото на распознавание, система извлекает промежуточные признаки из слоев нейронной сети и на их основании строит оценку. Приложение состоит из трех внутренних компонентов и трех внешних. Диаграмма представлена на рисунке 28.

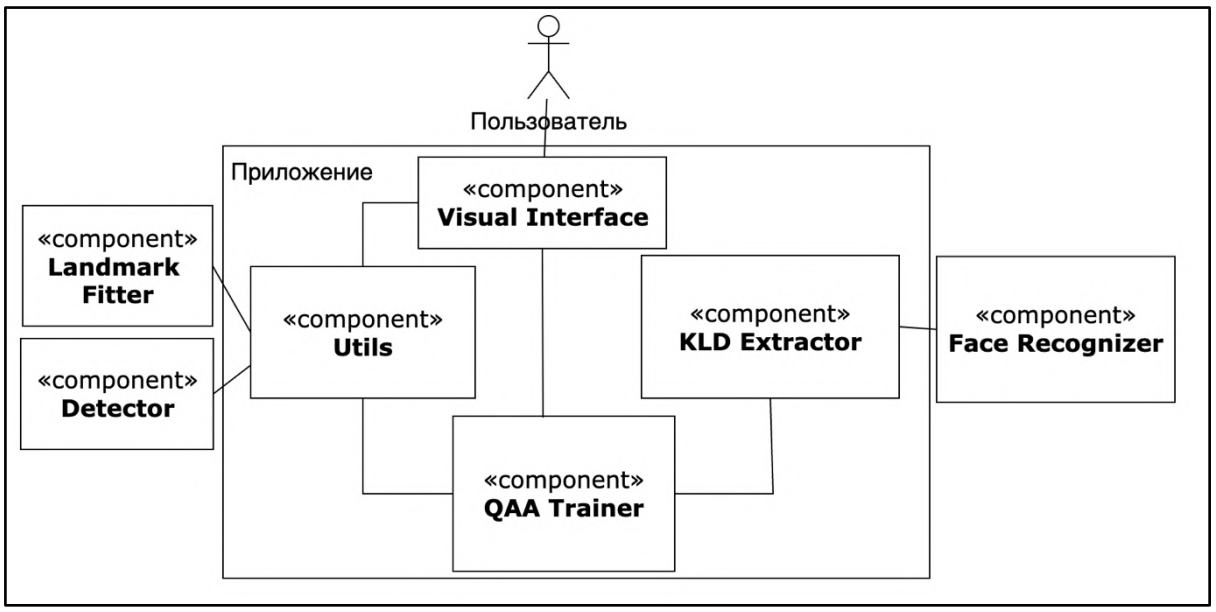

Рисунок 28 – Диаграмма компонентов приложения

QAA Trainer – компонент, содержащий определение архитектуры модели и код ее обучения. Компонент использует признаки, извлеченные из компонента Face Recognizer.

Utils – модуль, содержащий функции извлечения однородности заднего фона, предобработки и постобработки фото перед подачей в нейронною сеть.

Visual Interface – модуль, который предоставляет доступ к функционалу системы с помощью кнопок и визуальной информации.

KLD Extractor – компонент, реализующий процесс извлечения признаков из нейронной сети во время ее выполнения.

Detector – компонент, содержащий предобученную нейронной сети для детекции лиц на изображении. На вход компоненту приходит фотография, содержащая одну или несколько лиц, а на выходе выдает координаты прямоугольников для каждого лица на исходной фотографии.

Landmark Fitter– компонент, содержащий предобученную нейронную сеть для детекции антропометрических точек лица на изображении человека. На вход компоненту приходят все лица, отданные компонентом Detector, а

на выходе выдаются координаты антропометрических точек на каждом лице.

Face Recognizer – компонент, содержащий предобученную нейронную сеть для распознавания лиц. На вход компоненту приходит вырезанное лицо, центрированное по антропометрическим точкам, а на выходе отдается 512-мерный вектор признаков.

Utils – компонент, содержащий вспомогательные функции, например функцию для выравнивания лица по антропометрическим точкам или функцию для подсчета однородности заднего фона изображения.

Компоненты Landmark Fitter и Detector используются для предобработки фотографии перед использованием в компоненте Face Recognizer при обучении модели QAA, а также в компоненте Visual Interface при представлении результатов работы программы.

3.3. Реализация и тестирование системы

Реализация системы

`KLDExtractor` – компонент, реализующий процесс извлечения признаков из нейронной сети во время ее выполнения. Исходный код компонента `KLDExtractor` представлен в листинге 1.

Листинг 1 – Пример исходного кода компонента `KLDExtractor`

```
class KLDExtractor(nn.Module):
    """
    Extracts symmetric Kullback-Leibler divergence between running
    statistics of batch normalization layers and distributions of BN inputs
during network inference
    """
    def __init__(self, backbone, device='cuda'):
        super(KLDExtractor, self).__init__()
        self.backbone = backbone.to(device).eval()
        self.features = []
        self.device = device
        self.running = get_running_vals(self.backbone)

        ctr = 0
        for layer in self.backbone.modules():
            classname = layer.__class__.__name__
            if classname.find('BatchNorm') == 0:  # not BN
                layer.register_forward_hook(self.save_features_hook(ctr))
                ctr += 1
```

```
        assert ctr > 0  # At least one BN found

    def save_features_hook(self, i: int) -> Callable:
        def fn(_, _input, __):
            x = _input[0]
            if len(x.shape) <= 2:
                return
            x = x.flatten(start_dim=2)
            cv, cm = torch.var_mean(x, dim=-1)
            kld = kl_div_normal(cm, cv, self.running['means'][i], self.run-
ning['vars'][i])
            self.features.append(kld)

        return fn

    def forward(self, x):
        self.features.clear()
        emb = self.backbone(x)
        return emb, torch.hstack(self.features)

    def __call__(self, x):
        return self.forward(x)
```

В листинге представлено добавление хука для извлечения признаков во время прохода модели, реализованной с помощью модуля torch. Благодаря этому хуку можно производить манипуляции над промежуточными данными сети прямо во время выполнения.

`QAA Trainer` – компонент, реализующий обработку датасета, функции потерь, инициализацию нейронной сети и ее обучение. Реализован в виде jupyter-notebook, удобного формата воспроизведения кода на языке Python. Для работы с числами используются методы из библиотек torch и numpy, для обработки датасета, добавления аугментаций и нормализации использовалась библиотека albumentations. Регрессор был обучен с помощью библиотеки catboost. Для вычисления истинной разметки используется класс `CosineSimilarity` из модуля torch. Исходный код компонента `QAA Trainer` представлен в листинге 2.

Листинг 2 – Пример исходного кода компонента `QAA Trainer`

```
def train_qaa(backbone_bnf, dataset, centroids, batch_size=512, de-
vice="cuda", save_path="./regressor"):
    """
    Train CatBoostRegressor to predict QAA and save it on save_path

    arguments:
    :param backbone_bnf: backbone that return embedding and BN features
    (for example via bn_extraction.models.KLDExtractor)
```

```
    :param dataset: torch dataset, must return image and label
    :param centroids: centroids of each class. centroids[N] must be cen-
troid of class with label N.
    :param batch_size: batch_size in dataloader
    :param device: device to make calculation ("cpu"/"cuda")
    :param save_path: path to save quality regressor CatBoost model.
    """

    cos_sim = CosineSimilarity()
    train_loader = DataLoader(dataset, batch_size=batch_size)
    X_train = None
    y_train = None

    print("Preparing data...")
    with torch.no_grad():
        for batch in tqdm(train_loader):
            images, labels = batch
            embs, features = backbone_bnf(images.to(device))
            cent_sim = cos_sim(embs,
                               centroids[labels.to(device)]).flatten()  #
get similarity with centroid for every sample
            if X_train is None:
                X_train = features
                y_train = cent_sim
            else:
                X_train = torch.vstack((X_train, features))
                y_train = torch.hstack((y_train, cent_sim))

    X_train = np.array(X_train.detach().cpu())
    y_train = np.array(y_train.detach().cpu())
    print(X_train.shape)

    print("Training model...")
    regressor = cb.CatBoostRegressor(depth=depth, l2_leaf_reg=l2_leaf_reg,
                                     iterations=iterations, learn-
ing_rate=learning_rate,
                                     verbose=verbose, allow_writ-
ing_files=False)
    regressor.fit(X_train, y_train)

    try:
        regressor.save_model(save_path)
    except OSError as e:
        print("Save directory not exists... Saving to current directory")
        regressor.save_model("./regressor")
```

Все тренировочные данные берутся небольшими порциями, переносятся на GPU при наличии, проходят через нейронную сеть, затем вычисляются признаки и подаются в `CatboostRegressor` для обучения регрессора качества. Лучшие параметры регрессора CatBoost были найдены единожды с помощью поиска по сетке (grid search), а затем зафиксированы для экономии процессорного времени и ресурсов, так как обучение одной нейронной сети является времязатратной процедурой.

`Utils` – компонент, содержащий функции извлечения однородности заднего фона, предобработки и постобработки фото перед подачей в нейронною сеть. В листинге 3 представлен пример исходного кода функции, отвечающей за приведения лица к формату, заученному нейронной сетью. С помощью вычисления трансформации приведения одного набора точек к другому из модуля sklearn, можно можем нормализовать лицо.

Листинг 3 – Пример исходного кода компонента `Utils`

```
def normalize_face(img, kps, src):  # performs face alignment

    if src is None:
        src = np.array([
            [30.2946, 51.6963],
            [65.5318, 51.5014],
            [48.0252, 71.7366],
            [33.5493, 92.3655],
            [62.7299, 92.2041]], dtype=np.float32)
        src[:, 0] += 8.0


    _kps = np.array((np.mean(kps[36:41][:2]), np.mean(kps[42:47][:2])))
    kps = np.vstack((_kps, kps[33, 48, 54][:2]))

    dst = kps.astype(np.float32)

    tform = trans.SimilarityTransform()
    tform.estimate(dst, src)
    m = tform.params[0:2, :]
    warped_img = cv2.warpAffine(img, m, (112, 112), borderValue=0.0)

    return np.array(warped_img)
```

`Visual Interface` – компонент, который предоставляет доступ к функционалу системы с помощью кнопок и визуальной информации, позволяющий использовать систему пользователю без навыков программирования. Класс `MainWindow` реализован с помощью стандартных средств библиотеки Qt. Пример исходного кода компонента Visual Interface представлен в листинге 4.

Листинг 4 – Пример исходного кода компонента `Visual Interface`

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.W = 1280
        self.H = 720
```

```
        self.setFixedSize(QSize(self.W, self.H))
        self.image_processor = ImageProcessor()
        self.setWindowTitle("Оценщик качества")

        self.start_layout = QVBoxLayout()

        self.label_1 = QLabel('Фото на документы должно :\n' + \
                              '1) Иметь однородный задний фон\n' + \
                              '2) Содержать только одно лицо\n' + \
                              '3) Лицо должно выражать нейтральную эмоцию\n'
+ \
                              '4) Глаза должны быть открыты, лицо не
перекрыто волосами\n' + \
                              '5) Освещение равномерное, отсутствует тень\n',
                              self)
        self.label_1.setFont(QFont("Arial", 15))
        self.label_1.move(100, 100)
        self.label_1.setStyleSheet("border: 1px solid black;")

        self.load_image_button = QPushButton("Загрузить изображение")
        self.load_image_button.setCheckable(True)
        self.load_image_button.clicked.connect(self.load_image)

        self.start_layout.addWidget(self.label_1)
        self.start_layout.addWidget(self.load_image_button)

        self.start_widget = QWidget(self)
        self.start_widget.setLayout(self.start_layout)
        tip_w = 700
        tip_h = 600
        self.start_widget.setGeometry((self.W - tip_w) // 2, (self.H - tip_h)
// 2, tip_w, tip_h)
```

В листинге 4 представлено создание кнопок, виджетов, выставление базовых параметров для главного окна программы, а также печать на экран стартовой подсказки по вводу изображения лица для пользователя.

Интерфейс программы для оценки качества является интуитивно понятным и простым в использовании. При открытии пользователю предлагается загрузить фотографии для оценки, затем система оценивает параметры загруженной фотографии и выводит их на экран. Вступительное окно программы показано на рисунке 29.

После загрузки фотографии на экране показываются оценки качества лица, заднего фона и количество лиц, присутствующих на фотографии. Пользователь может изменить загруженную фотографию в случае низких оценок либо перейти к постобработке фотографии. Пример окна с оценками

параметров лица приведен на рисунке 30. Для демонстрации выбрано фото из интернета, с открытого источника.

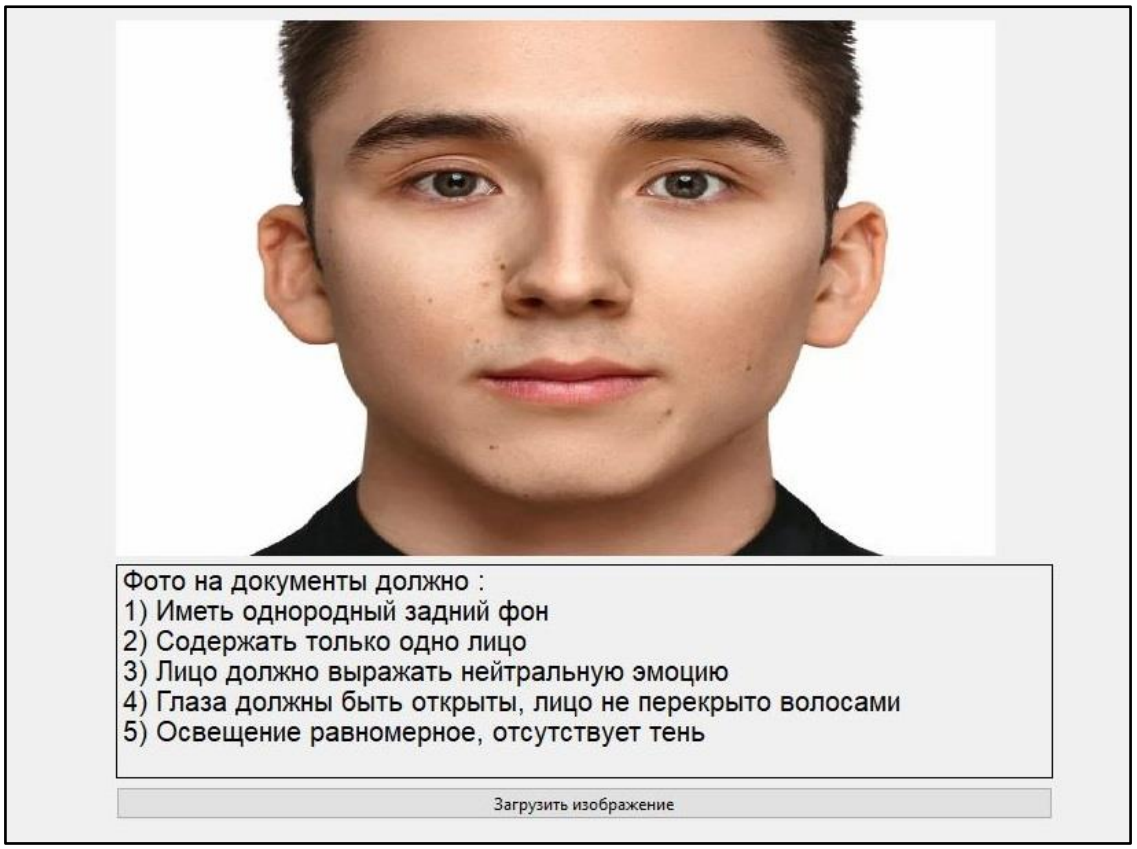

Рисунок 29 – Стартовое окно программы

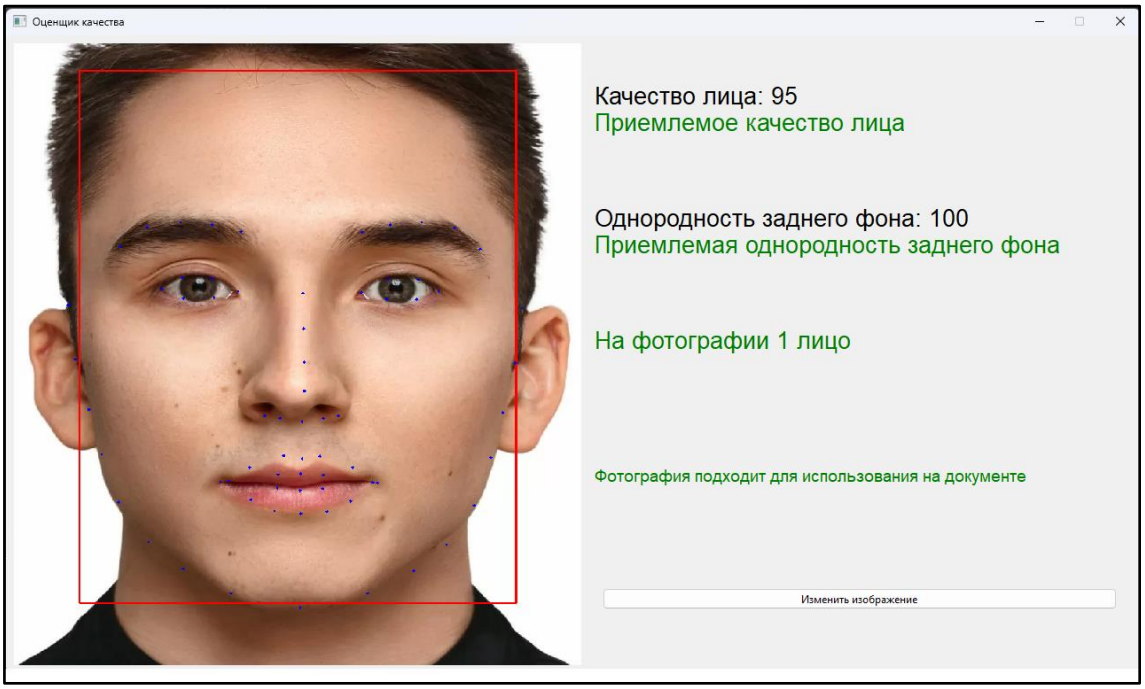

Рисунок 30 – Окно с оценкой параметров лица

Для вычисления однородности заднего фона был испробован алгоритм разреза графа [19]. В качестве начального приближения переднего фона будут выступать точки глаз, рта и носа человека, найденные с помощью детектора таких точек, а в качестве заднего фона углы изображения.

Тестирование системы

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Набор тестов на проверку корректной работы реализованных функций представлен в таблице 2.

Таблица 2–Тестирование системы

№	Название теста	Шаги	Ожидаемый результат
1.	Загрузить фотографию	В окне с оцененными параметрами фото нажать кнопку «изменить фотографию». Выбрать новую фотографию с лицом человека.	На экран выведется фотография и оцененные параметры
2.	Изменить загруженную фотографию	В окне с оцененными параметрами фото нажать кнопку «изменить фотографию». Выбрать новую фотографию с лицом человека.	Фотография на экране заменится новой, будут переподсчитаны оценка качества лица и однородность фона
3.	Перейти к обработке фотографии	В окне с оцененными параметрами фото нажать кнопку «продолжить».	На экран выведется обработанное фото на документы в разных форматах
4.	Сохранить обработанные варианты фотографии	В окне с обработанной фотографией нажать «сохранить» под нужным форматом. Выбрать путь для сохранения.	Обработанное фото будет сохранено в указанном месте

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы мною было разработано приложение для оценки качества фотографии лица на документы. При этом были выполнены следующие задачи.

1. Изучены подходы к построению нейронной сети для оценки качества лица.
2. Построена и обучена нейросеть для оценки качества фотографии лица на документы.
3. На основе обученной нейронной сети было реализовано и протестировано приложение на языке Python для оценки качества фотографий на документы.

Нейронная сеть, обученная в рамках данной работы, может быть использована в системе распознавания лиц для увеличения метрик путем первичной фильтрации фотографий, а подход, описанный для построения такой модели, может быть перенесен на другие задачи, так как является обоснованным не только при распознавании лиц.

Данная сеть была внедрена в промышленную эксплуатацию с подписанием акта о внедрении. В дальнейшем планируется продолжить исследования по тематике выпускной работы и усовершенствовать описанный подход.

ЛИТЕРАТУРА

1. Лаборатория Касперского. [Электронный ресурс] URL: https://www.kaspersky.ru/resource-center/definitions/what-is-facial-recognition (дата обращения: 11.02.2024 г.).

2. Национальный Институт Стандартов и Технологий (NIST). [Электронный ресурс] URL: https://www.nist.gov/ (дата обращения: 11.02.2024 г.).

3. Описание стандарта фотографии ISO/IEC 29794-5. [Электронный ресурс] URL: https://eab.org/files/events/2021-11-1618_Face_Image_Quality/20211118_05_grother_NIST-Quality-Evaluations.pdf (дата обращения: 11.02.2024 г.).

4. FRVT Quality Concept. [Электронный ресурс] URL: https://www.nist.gov/system/files/documents/2019/04/23/frvt_quality_concept_1.0.pdf (дата обращения: 11.02.2024 г.).

5. Loginom. Матрица ошибок. [Электронный ресурс] URL: https://wiki.loginom.ru/articles/error-matrix.html (дата обращения: 11.02.2024 г.).

6. Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun. Deep Residual Learning for Image Recognition. [Электронный ресурс] // arXiv.org. 2015. Дата обновления: 10.12.2015 г. URL: https://arxiv.org/abs/1512.03385 (дата обращения: 11.02.2024 г.).

7. Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. [Электронный ресурс] // arXiv.org. 2018. Дата обновления: 04.09.2022 г. URL: https://arxiv.org/pdf/1801.07698.pdf (дата обращения: 11.02.2024 г.).

8. Репозиторий insightface. [Электронный ресурс] URL: https://github.com/deepinsight/insightface (дата обращения: 11.02.2024 г.).

9. Fu-Zhao Ou and Xingyu Chen and Ruixin Zhang and Yuge Huang and Shaoxin Li and Jilin Li and Yong Li and Liujuan Cao and Yuan-Gen Wang. SDD-FIQA: Unsupervised Face Image Quality Assessment with Similarity Distribution Distance. [Электронный ресурс] // arXiv.org. 2021. Дата обновления: 10.03.2021 г. URL: https://arxiv.org/pdf/2103.05977.pdf (дата обращения: 11.02.2024 г.).

10. Yanghao Li and Naiyan Wang and Jianping Shi and Jiaying Liu and Xiaodi Hou. Revisiting Batch Normalization For Practical Domain Adaptation. [Электронный ресурс] // arXiv.org. 2016. Дата обновления: 08.11.2016 г. URL: https://arxiv.org/pdf/1603.04779.pdf (дата обращения: 11.02.2024 г.).

11. Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. [Электронный ресурс] // arXiv.org. 2015. Дата обновления: 02.03.2015 г. URL: https://arxiv.org/pdf/1502.03167.pdf (дата обращения: 11.02.2024 г.).

12. Christian Szegedy and Wei Liu and Yangqing Jia and Pierre Sermanet and Scott Reed and Dragomir Anguelov and Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich. Going deeper with convolutions. [Электронный ресурс] // arXiv.org. 2014. Дата обновления: 17.09.2014 г. URL: https://arxiv.org/pdf/1409.4842.pdf (дата обращения: 11.02.2024 г.).

13. Andrew G. Howard and Menglong Zhu and Bo Chen and Dmitry Kalenichenko and Weijun Wang and Tobias Weyand and Marco Andreetto and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. [Электронный ресурс] // arXiv.org. 2017. Дата обновления: 17.04.2017 г. URL: https://arxiv.org/pdf/1704.04861.pdf (дата обращения: 11.02.2024 г.).

14. Xiangyu Zhang and Xinyu Zhou and Mengxiao Lin and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Networkfor Mobile Devices. [Электронный ресурс] // arXiv.org. 2017. Дата обновления:

07.12.2017 г. URL: https://arxiv.org/pdf/1707.01083.pdf (дата обращения: 11.02.2024 г.).

15. Hao Wang and Yitong Wang and Zheng Zhou and Xing Ji and Dihong Gong and Jingchao Zhou and Zhifeng Li and Wei Liu. CosFace: Large Margin Cosine Loss for Deep Face Recognition. [Электронный ресурс] // arXiv.org. 2018. Дата обновления: 03.04.2018 г. URL: https://arxiv.org/pdf/1801.09414.pdf (дата обращения: 11.02.2024 г.).

16. ИТМО. Функция потерь Softmax. [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=SoftMax_%D0%B8_SoftArgMax (дата обращения: 11.02.2024 г.).

17. Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. [Электронный ресурс] // arXiv.org. 2014. Дата обновления: 10.04.2015 г. URL: https://arxiv.org/pdf/1409.1556.pdf (дата обращения: 11.02.2024 г.).

18. Cascade Classifier opencv. [Электронный ресурс] URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (дата обращения: 11.02.2024 г.).

19. Normalized Cut. [Электронный ресурс] URL: https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf (дата обращения: 11.02.2024 г.).

20. Liudmila Prokhorenkova and Gleb Gusev and Aleksandr Vorobev and Anna Veronika Dorogush and Andrey Gulin. CatBoost: unbiased boosting with categorical features. [Электронный ресурс] // arXiv.org. 2017. Дата обновления: 20.06.2019 г. URL: https://arxiv.org/pdf/1706.09516.pdf (дата обращения: 11.02.2024 г.).