САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Симонов Андрей Олегович

Выпускная квалификационная работа бакалавра

Применение нейросетевых технологий для определения пола и возраста человека на основе фотографии лица

Направление 010400

Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Сергеев Сергей Львович

Санкт-Петербург

2017

Содержание

Введение

Пол и возраст человека играют важнейшую роль в социальной жизни индивида и в его взаимодействии с другими людьми. При описании людей разного гендера мы применяем различные языковые конструкции и разные обращения. Таким же образом в зависимости от возрастной группы человека во многих языках меняется форма обращения к нему. Данные признаки важны и в социологических исследованиях для определения социальной группы и статуса лица в ней.

Задачи определения возраста и пола человека относятся к группе задач идентификации человека. В современном социуме даже человеческий глаз не всегда способен определить пол и возраст персоны с которой он взаимодействует, так как грань между мужским и женским обликом сильно утончается. Традиционные признаки полов, такие как длина волос, растительность на лице и украшения постепенно уходят из обликов людей. Появляются и трудности в определении возраста, мы и так не во всех случаях могли точно определить возраст человека, а с развитием и популяризацией пластических операций и косметических средств, полагаться на свое зрение, и вовсе не приходится.

Самым надежным способом определения пола является, конечно, анализ крови на гормоны, но подобная процедура весьма трудоемка. Хотелось бы иметь надежную возможность распознать эти параметры более простым и быстрым способом.

В то же время с развитием информационных технологий все чаще появляется потребность в системах, оценивающих какие-либо параметры человека по фотографиям, будь то пол, возраст, раса, эмоции или даже личность индивида. Надо сказать, что у компьютерного зрения по сравнению с человеческим есть определенные преимущества, так вычислительная машина лишена заложенных в нас стереотипов. Несмотря на большое

количество предложенных решений по этой теме до сих пор не была разработана система автоматической идентификации, позволяющая оценить возраст и пол человека максимально правдоподобно и безошибочно. И причин на это множество, от индивидуальности черт лица каждого человека, и непредсказуемости процессов старения как таковых, до сложности составления больших правдоподобных баз данных.

В последние несколько лет происходит большой ажиотаж вокруг подобных систем, во много благодаря вновь нарастающей популярности методов глубокого обучения, если быть точнее, сверточных нейронных сетей. В настоящее время все крупнейшие IT компании мира так или иначе стараются приложить руку к подобным исследованиям. Ежедневно выходят статьи насчет компьютерного зрения, идентификации, распознавания и классификации самых различных объектов. Кроме того, во всех последних международных олимпиадах по классификации изображений побеждают исключительно системы, основанные на сверточных нейронных сетях. Такая популярность технологии способствует постоянному совершенствованию используемых методов.

Постановка задачи

Целью данной работы является создание системы автоматического распознавания пола и возраста человека по фотографии лица на основе сверточных нейронных сетей. В ходе работы будут реализованы две такие сети, одна для распознавания возраста, другая для определения пола. Основываясь на данной цели были сформулированы основные задачи:

- Выбрать открытую базу данных фотографий с указанием пола и возраста людей.
- Реализовать предобработку изображений.
- Выделить тестовую, тренировочную и валидационную часть базы данных.
- Определиться с архитектурой нейронных сетей.
- Выбрать методы обучения.
- Реализовать нейронные сети.
- Обучить сети и проверить их работу на тестовом множестве.

Обзор существующих методов решения

Проблемы определения возраста и пола человека пытались решить множеством способов, некоторые из них были основаны, например, на различии цвета кожа у лиц разных возрастов и на определении расположения и формы ключевых особенностей лиц. Основной проблемой данных методов является то, что их точность страдает при искажении фотографии, ухудшении качества изображения, при нестандартном освещении или повороте лица.

Рассмотрим подробнее некоторые из представленных методов:

1) Методы, основанные на активной модели внешнего вида (*Active appearance model - AAA)* [1]

AAA – группа статистических методов, оценивающая визуальные возрастные особенности. Учитываются как главные особенности такие как глаза, рот, нос и подбородок, так и вторичные, например, морщины. На основе различий этих особенностей у лиц разных возрастных групп строится модель.

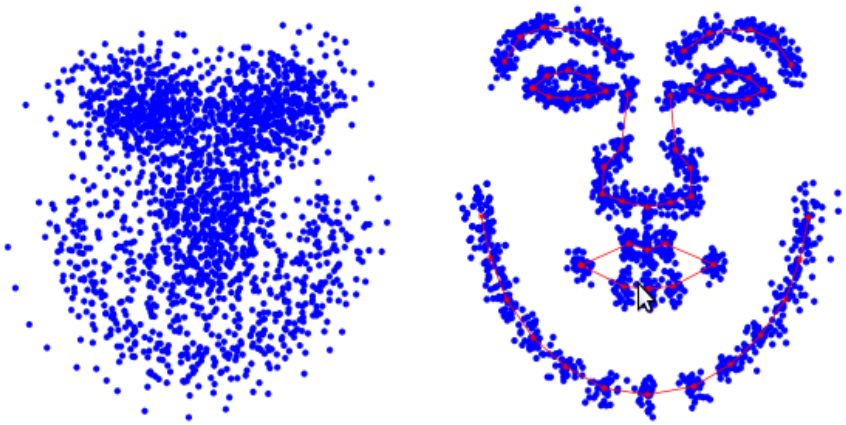

Рисунок 1

Для сбора особенностей, на каждом лице вычисляется набор особых точек и выделяются регионы морщин. Shuicheng Yan использовал модель внешнего вида, называемую Patch-Kernel [2]. Как видно по рисунку 1 форма модели обозначает внешний контур лица, контуры глаз, носа и бровей. На первой половине рисунка представлено множество точек до нормализации, на второй после.

Метод определения возраста по таким моделям основывается на вычисления расстояния Кульбак-Лейблера между моделями, которые выведены из модели гауссовых смесей (GMM) с вычислением максимальной апостериорной вероятности для любых двух изображений. Затем происходит процесс обучения - синхронизация интермодального сходства. В итоге для оценки возраста используются методы ядерной регрессии.

2) Aging pattern subspace [3, 4].

Метод моделирует шаблоны старения, для построения которого используется последовательные изображения старения лица. Модель строится посредством изучения пространства признаков получаемого методом главных компонент. Исследователи экспериментировали и с другими методами уменьшения размерности и методами вложения, такими как локально-линейное вложение, сохранение ортогональности локальных проекций. В итоге ими был предложен метод локально настроенной неустойчивой регрессии для обучения и предсказания возраста человека. Этот метод использует регрессию опорных векторов для предсказаний, и определяет параметры модели для разных диапазонов возрастов.

3) Метод основанный на анализе биологических особенностей

Данный метод работает с частотной характеристикой изображений людей. Благодаря этому удается извлечь глубокие свойства из фотографий. Алгоритм [5] исследует биологические особенности лиц для оценки возраста

человека. Для построения таких особенностей используются массивы фильтров Габора - линейный фильтры, характеризующийся гармонической функцией, умноженной на гауссиан. Такие фильтры помогают распознать границы объектов на изображении, и часто используются в задачах дискриминации. Фильтры применяются на изображения с разным масштабом и положением, чтобы извлечь максимальное количество информации из фотографии. После построения особенностей, методом главных компонент и линейным дискриминантным анализом выделяются существенные признаки.

4) Метод локальных бинарных шаблонов [6]

Понятие локального бинарного шаблона было представлено обществу в 1996 году и фактически, в этом методе каждый пиксель изображения кодируется некоторым двоичным кодом (чаще всего 8-мью битами) описывающим окрестность этого пикселя. Например, на рисунке 3 задается начальные treshold значение, и все пиксели у которых интенсивность больше этого значения кодируются 1, остальные пиксели получают значение 0. Так получается дескриптор окрестности.

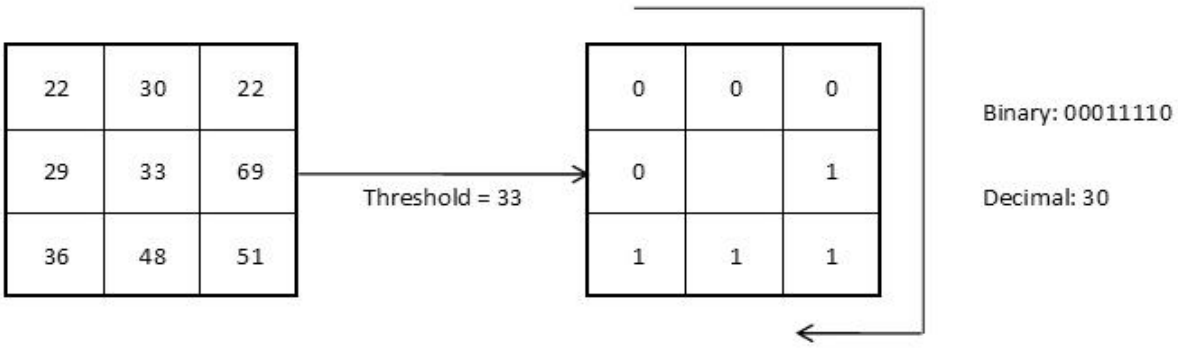

Рисунок 2

В более поздних модификациях этого метода при описании окрестности используются пиксели, находящиеся на некотором заданном отдалении от центрального пикселя, то есть образуют эллипс.

В задачах распознавания пола и возраста, снимок лица человека сперва разбивается на некоторое число областей, и для каждого пикселя каждой области строится свой дескриптор. Таким образом каждая область будет иметь свою карту признаков. Затем все наборы объединяются в единый вектор и происходит обучение программы методом опорных векторов.

Стандартный алгоритм использующий бинарные шаблоны:

- Строится карта бинарных шаблонах для всего изображения.
- Фотография делиться на непересекающиеся области.
- По полученным данным строится гистограмма.
- Проводится обучение методом опорных векторов на основе обучающей выборки.

Глава 1: Нейронные сети

1.1 Основы нейронных сетей

Нейронные сети - один из методов машинного обучения, к которому в последнее время наблюдается повышенный интерес не только среди специалистов данной области, но и вообще людей, никак не связанных с этой профессией. Причиной этому служит появление множества интересных публике проектов: от громких сервисов Google с нейронной сетью, заканчивающей за пользователем его рисунки, до сети, определяющей покемона подходящего человеку исходя из его внешности. Такие проекты очень сильно повышают популярность нейронных сетей, благодаря чему данной сферой начинает интересоваться и множество специалистов. Вследствие чего, развитие нейронных сетей не останавливается ни на секунду и ежедневно публикуется множество статей и область становится все глубже, а результаты работы сетей все лучше. В итоге, сейчас в подавляющем большинстве соревнований по классификации данных побеждают именно алгоритмы, основанные на нейронных сетях.

На самом деле, идея нейронной сети совсем не нова, первые исследования, связанные с ними, проводились еще в 1943 году. За свою историю они переживали множество спадов, на некоторое время о них забывали, но в итоге в наше время происходит очередной подъем. Во многом это связано с возрастающими вычислительными мощностями компьютеров, и появившейся возможности перенести вычисления на графический процессор. Немалую роль сыграло и огромное количество информации и баз данных находящихся в открытом доступе, благодаря чему нет недостатка в тренировочных данных, а это крайне важно для сетей, так как на малых объемах обучающих выборок они показывают неудовлетворительные результаты.

Свою основу нейронные сети берут в биологии, именно оттуда пошла сама идея создания сети нейронов, связанных между собой синапсами (связями). Искусственный нейрон (рис. 3) - это вычислительная единица, которая имеет некоторое ограниченное количество входов каждый из которых имеет некий вес, и один выход. Нейрон суммирует взвешенные сигналы, поступающие на входы, осуществляет некоторое нелинейное преобразование над суммой и передает результат на выход.

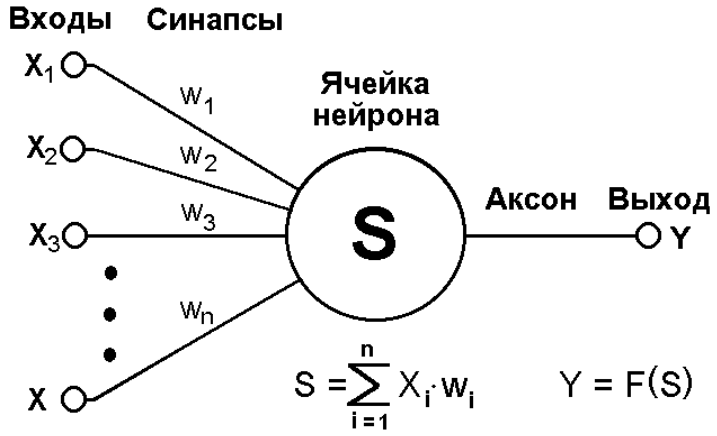

Рисунок 3

В итоге получается, что даже один нейрон может служить классификатором, не особо эффективным, конечно, и работающим только в случае линейно разделимых множеств. На практике такое встречается крайне редко, поэтому одного нейрона оказывается недостаточно.

Функцией активации нейрона называется функция определяющая зависимость между взвешенной суммой нейрона и его результатом. В большинстве случаев она является монотонно возрастающей и ограниченной на промежутках $[-1,1]$ или $[0,1]$. Также большинство алгоритмов обучения требуют непрерывной дифференцируемости на всей числовой оси.

Нейронная сеть состоит из слоев таких нейронов, связанных между собой, так что сигнал, выходящий из одного нейрона, подается на вход нейрону следующего слоя. Причем каждый нейрон имеет свой набор весов.

Результатом работы подобной сети будет вектор выходных сигналов последнего слоя. И к этому вектору применяется функция ошибки, которая будет показывать насколько результат сети далек от эталонного.

Задача обучения нейронной сети состоит в настройке весов, таким образом, чтобы выходная ошибка была минимальна. Для этого необходимо определить насколько вес каждой связи повлиял на функцию. В основе большинства алгоритмов обучений лежит метод обратного распространения ошибки, с его помощью можно вычислить ошибку на глубоких слоях сети. Его суть заключается в том, что после получения результата сети, ошибка каждого нейрона последнего слоя передается по сети в обратную сторону - к первому слою. В результате вычисляется градиент функции ошибки по каждому нейрону сети.

1.2 Сверточные нейронные сети

Сверточные нейронные сети - особый тип нейросетей, который произвел фурор в сфере распознаваний изображений в 2012 году. Но их история началась задолго до этого. В 1981 году Дэвид Хьюбел и Торстен Визель стали нобелевскими лауреатами формально за “работы, касающиеся принципов переработки информации в нейронных структурах и механизмов деятельности головного мозга”. Поставленный ими на кошках эксперимент показал несколько критичных для сверточных нейронных сетей концепций:

- Соседние нейроны обрабатывают информацию с соседних областей сетчатки.
- Структура нейронов в головном мозгу иерархическая, то есть каждый последующий уровень нейронов выделяет все более и более высокоуровневые признаки изображения.
- Нейроны образованы в группы трансформирующие и транслирующие сигналы между уровнями.

На основе этих исследований в 1980 годах Кунихой Фокусимой были предложены две архитектуры нейронных сетей повторяющих биологическую модель зрения: когнитрон и неокогнитрон. Обучение проводилось без учителя с помощью оригинального эвристического алгоритма. В этих моделях были заложены основы всех сверточных сетей.

Эти исследования были продолжены в 1988 году Яном ЛеКунном, он связал модели, созданные Фокусимой с алгоритмом обратного распространения ошибки. В итоге получилась первая работающая сверточная нейронная сеть, получившая название LeNet. Успех был настолько большим, что в США стали использовать такую сеть для распознавания почтовых индексов, а архитектура LeNet стала базовой для всех будущих сетей. В ее основе два слоя: сверточный слой и слой субдискретизации (Pulling слой)

Сверточный слои — это главная вычислительная единица сверточной нейронной сети. Данный слой имеет несколько каналов, каждый из которых осуществляет операцию свертки:

$$C\left(X,\ Y\right)\ =\ \sum_{a\ =\ 0}^{k-1}\ \sum_{b\ =\ 0}^{k-1} I\left(x-a,\ y\ -\ b\right) F\left(a,b\right)$$

здесь I - исходная матрица изображения, а F - ядро (фильтр) свертки. Формально эта операция заключается в том, что мы проходим по всему

изображению f окном размером g, на каждом шагу поэлементно умножая содержимое окна на ядро g, результат суммируем и записываем в матрицу результата. При это на размерность результирующей матрицы влияет шаг, размер ядра и способ обработки границ. Так, есть 3 способа:

- Отсечение

  Любое ядро, для вычисления которого требуются пиксели, лежащие за пределами изображения, не будет учитываться. В данном случае размер результирующего изображения будет немного меньше исходного.

- Заворачивание

  Значения для пикселей при выходе за границы берутся с противоположного края изображения.

- Продление

  При выходе ядра за пределы изображения, отсутствующие значения пикселей, заполняются значениями наиболее близких к ним пикселей границы изображения либо нулевыми значениями.

В сверточном слое реализуется концепция локально рецептивных полей, полученная из экспериментов Дэвид Хьюбела и Торстен Визеля, то есть каждый выходной нейрон соединен только с небольшой областью исходного изображения. В общем случае выход сверточного слоя описывается формулой:

$$O(x,y) = \sum_{x'} \sum_{y'} w^{l}_{x',y'} f(o^{l-1}_{x-x',y-y'}) + b^{l}_{x,y}$$

здесь $w^{l}_{x.y}$ это вектор соединяющий нейроны слоя l и $l+1$, f - активационная функция нейронов, $o^{l}_{x,y}$- выходной вектор l слоя, $b^{l}_{x,y}$ - коэффициент сдвига.

Слой пуллинг выполняет нелинейное уплотнение карты признаков, при этом группа признаков уплотняется до одного пикселя, проходя нелинейное преобразование. Чаще всего для этого используют функцию максимума. Так, преобразуются непересекающиеся окна заданного размера на изображении, каждое из которых уменьшается до размера пикселя (рис. 5)

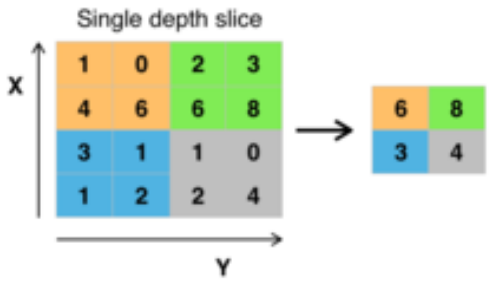

Рисунок 4

Целесообразность данной операции заключается в том, что если перед слоем пуллинга стоит сверточный слой, то фильтры этого слоя уже выделили некие признаки на изображении, и полное изображение уже не требуется. Таким образом сильно уменьшается объем картинки. Также такое уплотнение данных помогает бороться с переобучением сети.

Кроме функции максимума часто используют функцию среднего значения, или L2-нормирование:

$$S = \sqrt{\sum(x^2)}$$

В пуллинг слое используется свойство локальной скоррелированности пикселей - соседние пиксели, как правило, не сильно отличаются друг от друга, соответственно потери информации при таком сжатии незначительны.

В итоге чередуя слои субдискретизации и свертки ЛеКун получил сеть с архитектурой, как на рисунке 5

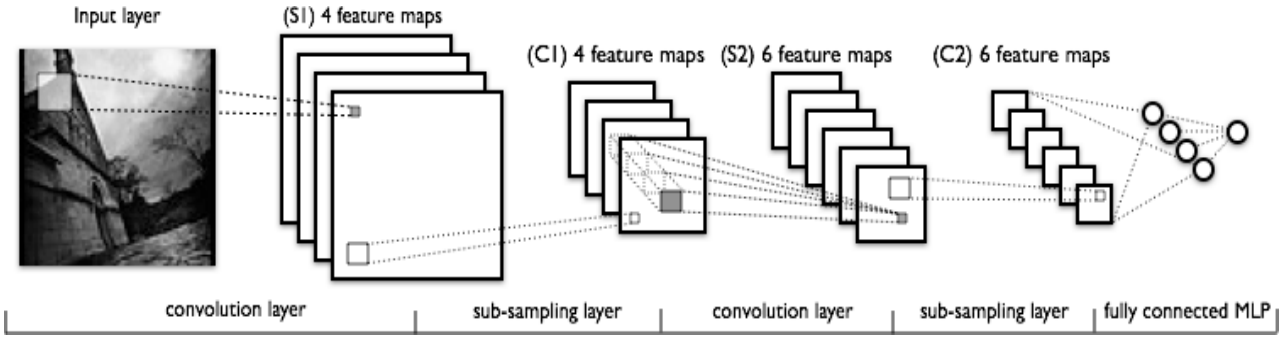

Рисунок 5

Как видно на схеме, после чередующихся слоев свертки и субдискретизации стоят несколько полносвязных слоев. Они выполняют роль классификаторов, то есть, после выделения абстрактных признаков сверточной сетью, ее результаты передаются на такие слои, и на них происходит прогнозирование результата.

1.3 AlexNet

Спустя еще 14 лет Алекс Крижевский доработал модель, созданную ЛеКуном, добавив в нее несколько важнейших дополнений, сеть назвали AlexNet. Она почти полностью повторяет архитектуру LeNet, в нее были добавлены несколько дополнительных сверточных слоев, размер ядер сверток стал уменьшаться от входа сети к выходу, и была добавлена операция локального нормирования контраста изображения.

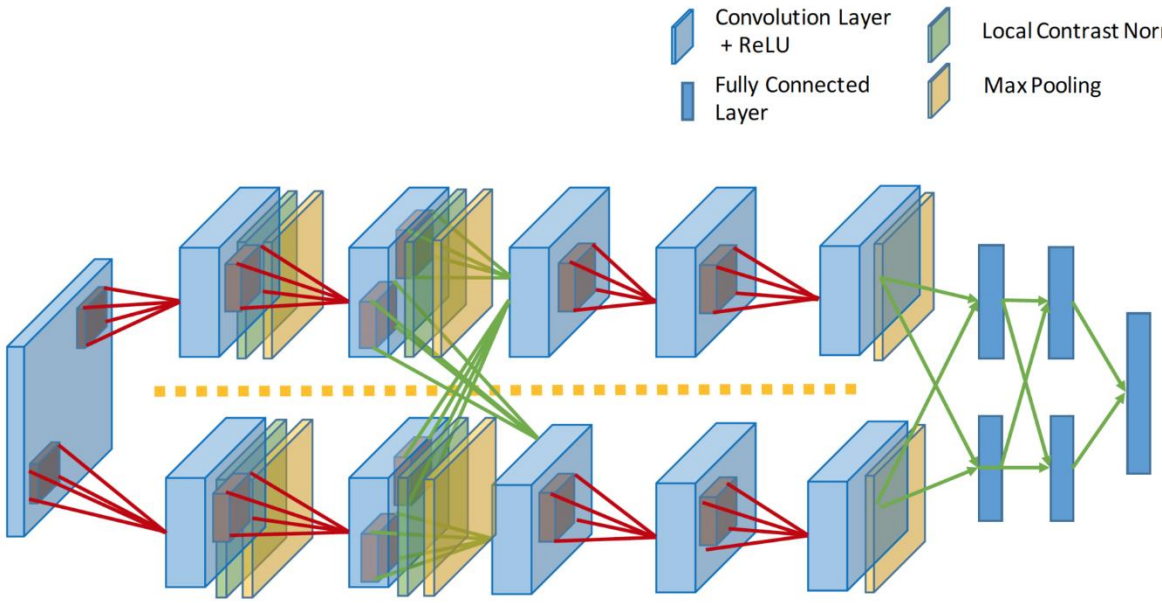

Рисунок 6

В итоге по сути получилась та же сеть LeNet увеличенная в тысячу раз (рисунок 6). При обучении сети были использованы важные и по сей день способы избежание переобучения сети:

- Слой ReLu (rectified linear unit)

  ReLu - функция активации нейронов, представленная в 2000 году, в журнале Nature обоснованная с биологической и математической точки зрения:

$$f(x) = max(0, x)$$

  Данная функция пришла на замену популярной в то время сигмоидальной функции, основанной на теории вероятности:

$$f(x) = \frac{1}{1 + e^{-x}}$$

  И на данный момент ReLu и ее модификации остаются самыми часто используемыми функциями активации нейронов. Важным плюсом является то что эти функции уменьшают вероятность

затухания градиента, при обратном распространении ошибки, что является крайне значительным фактором для градиентных методов обучения, также вычисление самой функции и ее производной происходят гораздо быстрее по сравнению с тем же сигмоидом. Среди основных модификаций ReLu:

1. ReLu с зашумлением

   Добавляет в стандартный ReLu гауссовские шумы

   $$f(x) = max(0, x + Y), Y \sim N(0, \sigma(x))$$

   Данная модификация пользуется популярность в решениях задач машинного зрения, основанных на ограниченных машинах Больцмана.

2. Leaky ReLUs

   Идея Leaky ReLUs в том, чтобы не аннулировать градиент при неактивности нейрона, а заменить его некоторым малым числом:

   $$f(x) = \begin{cases} x & if\ x > 2 \\ \alpha x & otherwise \end{cases}$$

   где α - параметр, определяемый эмпирически. Зачастую берется равным 0.01.

3. ELUs

   Модификация пытается приблизить результаты функции активации к 0, что ускоряет обучения:

   $$f(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha(e^x - 1) & otherwise \end{cases}$$

   Где $\alpha \geq 0$ - гипер-параметр сети. Было доказано что ELu позволяет достичь большей точности классификации чем ReLu.

- DropOut [7]

  При обучении сети, для каждого примера создается своя нейронная подсеть, то есть каждый узел сети исключается из сети с некоторой заранее заданной вероятностью. Таким образом веса связей в сети становятся менее чувствительными к весам других связей, что в свою очередь помогает сети лучше обобщать данные и увеличивает точность.

- Аугментация данных

  Для обучения сети требуется огромное количество тренировочных данных, от их числа зависит то, насколько сеть будет хорошо обобщаться свои результаты. Аугментация данных позволяет из небольшой базы данных создать достойный тренировочный набор данных. Эта операция заключается в преобразовании исходного изображения, чтобы получить похожее на него. Среди способов преобразования есть, например, поворот изображения, затемнение случайных пикселей и сдвиги.

Глава 2: Обучение нейронных сетей

2.1 Задача обучения

Обучение нейронной сети - это процесс в ходе, которого свободные параметры сети настраиваются посредством моделирования среды, в которую эта сеть встроена. То есть при поступлении некого сигнала в сеть, алгоритм обучения изменяет параметры таким образом, чтобы приблизить результаты сети к верному ответу.

Существует два концептуально разных подхода к обучению нейронных сетей: обучение с учителем и без него.

Обучение с учителем предполагает, что кроме обучающего множества примеров у нас так же есть множество эталонных ответов. такое что каждому примеру соответствует определенный эталонный ответ. При обработке такого примера сетью, выход сети сравнивается с эталонным результатом и свободные параметры изменяются в зависимости от того насколько сильно вектор, выданный сетью отличается от идеального вектора.

Обучение без учителя наиболее близко к биологическим механизмам запоминания образов: у нас есть только обучающее множество без заранее известных ответов. Алгоритмы обучения должен настроить сеть таким образом, чтобы при предъявлении сети двух схожих обучающих примеров, выходные вектора были также близки друг к другу. Таким образом алгоритм должен сгруппировать обучающие примеры на классы, члены которых будут давать схожие результаты работы сети.

В данной работе будут использованы исключительно алгоритмы обучения с учителем, так как на настоящий момент времени они более эффективно справляются со своими задачами, и мы имеем не испытываем недостатка маркированных обучающих примеров.

Итак, у нас есть некоторое тренировочное множество X и множество эталонных ответов сети Y. На вход обучающему алгоритму подается пара $(x, y),\ x \in X,\ y \in Y$, задача алгоритма обучения с учителем найти функцию $f : X \to Y$ в допустимом классе функций. Также вводится функция ошибки сети $E(x)$.

Наиболее популярной функцией ошибки является функция ошибки, полученная по методу наименьших квадратов:

$$E(x_k) = 0.5 * \sum_{k \in inputs} (y_k - f(x_k))^2$$

Таким образом с вводом функции ошибки задача обучения нейронной сети сводится к задаче минимизации функции потерь, имеющую очень большую размерность. При этом зачастую эта задача будет многоэкстремальной невыпуклой задачей оптимизации.

2.2 Метод обратного распространения ошибки

Для того чтобы решить задачу оптимизации, нам необходимо найти производную функции ошибки по каждому весу сети. В достижении этой цели нам помогает метод обратного распространения ошибки. У каждого алгоритма, основанного на нем есть три основные фазы:

1. Распространения входного вектора по сети и получение выхода сети
2. Обратное распространение ошибки к первому слою нейронной сети
3. Изменение весов сети

Разберем для начала алгоритм обратного распространения ошибки для многослойного персептрона, затем перейдем к рассмотрению алгоритма для сверточной нейронной сети.

Алгоритм прямого распространения:

1. На вход слою подается вектор x^l и рассчитываются активационные функции нейронов слоя

$$y_i^l = \sigma(x_i^l) + I_i^l \tag{1}$$

где I_i^l - сдвиг i - *го* нейрона l-*го* слоя.

2. Вычисляется входной вектор для следующего слоя сети:

$$x_i^l = \sum_j w_{ji}^{l-1} * y_j^{l-1} \tag{2}$$

где w_{ji}^l - вес связи соединяющий j-*ый* нейрон $l-1$ слоя и i-*ый* нейрон слоя l.

3. Шаги 1 и 2 повторяются до тех пор, пока не будет достигнут последний слой сети, и не будет вычислен выходной вектор сети
4. Рассчитывается ошибка сети $E(y^l)$ с помощью подходящей задаче функции ошибки

Следующий этап - обратное распространение ошибки. Посчитав ошибку на последнем слое сети, необходимо настроить ее свободные параметры таким образом, чтобы сеть приблизилась к эталонному вектору ответов. Для этого необходимо вычислить производную функции ошибки по всем весам. В этом нам поможет цепное правило взятие производной:

Лемма: Цепное правило

Если $y = f(u)$ и $x = g(u)$, тогда:

$$\frac{dy}{dx} = \frac{dy}{du} * \frac{du}{dx} \tag{3}$$

Тогда согласно (3):

$$\frac{dE}{dw_{ij}^l} = \frac{dE}{dx_j^{l+1}} * \frac{dx_j^{l+1}}{dw_{ij}^l}$$

В силу (2):

$$\frac{dE}{dw_{ij}^l} = \frac{dE}{dx_j^{l+1}} * y_i^l \tag{4}$$

В этом выражении для нас остается неизвестной только производная функции ошибки по входным векторам слоев, так как все y_i^l известны из прямого распространения сигнала.

Снова применяя (2) и цепное правило получаем:

$$\frac{dE}{dx_j^l} = \frac{dE}{dy_j^l}\frac{dy_j^l}{dx_j^l} = \frac{dE}{dy_j^l}\frac{d}{dx_j^l}(\sigma(x_j^l) + I_j^l) = \frac{dE}{dy_j^l}\sigma'(x_j^l) \tag{5}$$

Тогда для последнего слоя $l = L$ сети:

$$\frac{dE}{dy_i^L} = \frac{d}{dy_i^L}E(y^L) \tag{6}$$

Важное замечание: мы рассматриваем функцию ошибки исключительно как функцию от y_i^L, принимая остальные компоненты вектора y^L за константу.

Для любого другого слоя сети применяем цепное правило еще раз:

$$\frac{dE}{dy_i^l} = \sum_j \frac{dE}{dx_j^{l+1}}\frac{dx_j^{l+1}}{dy_i^l} = \sum_j \frac{dE}{dx_j^{l+1}} w_{ij} \tag{7}$$

Таким образом алгоритм обратного распространения:

1. Вычисляем производные на последнем слое по (6).
2. Вычисляем производные функции ошибки по входным сигналам. слоя l, где l-первый слой с известной ошибкой (5).
3. Передаем ошибку $l-1$ слою (7).
4. Повторяем шаги 2 и 3 пока не дойдем до входного слоя.
5. Вычислить градиент функции ошибки по весам сети (4).

После нахождения градиента происходит изменение весов сети в соответствии с алгоритмом обучения.

Для сверточных сетей алгоритм претерпевает некоторые изменения в силу другой архитектуры сети. При прямом прохождении все так же рассчитывается функция ошибки на последнем полносвязном слое, затем по алгоритму обратного распространения для многослойного персептрона ошибка передается до последнего слоя сверточной сети.

Вспомним, как формируются результирующая матрица на сверточном слое, имеющем фильтр размером $m \times m$:

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} y_{i+a,j+b}^{l-1}$$

$$y_{ij}^{l} = \sigma'(x_{ij}^{l})$$

Теперь рассмотрим градиент функции ошибки по весам сверточного слоя:

$$\frac{dE}{dw_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{dE}{dx_{ij}^{l}} \frac{dx_{ij}^{l}}{dw_{ab}^{l}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{dE}{dx_{ij}^{l}} y_{i+a,j+b}^{l-1}$$

Здесь $N \times N$ размерность входной матрицы для слоя.

$$\frac{dE}{dx_{ij}^l} = \frac{dE}{dy_{ij}^l}\frac{dy_{ij}^l}{dx_{ij}^l} = \frac{dE}{dy_{ij}^l}\sigma'(x_{ij}^l)$$

Для передачи ошибки предыдущему слою воспользуемся цепным правилом снова:

$$\frac{dE}{dy_{ij}^{l-1}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\frac{dE}{dx_{(i-a),(j-b)}^l}\frac{dx_{(i-a),(j-b)}^l}{dy_{ij}^{l-1}}$$
$$= \sum_{a=0}^{m-1}\sum_{b=0}^{m-1}\frac{dE}{dx_{(i-a),(j-b)}^l}w_{ab}$$

Как уже было сказано ранее, на слоях пуллинга не производится никакого обучения, вместо это там происходит сжатие карты признаков. Так, значения нескольких пикселей сжимается до одного выходного значения, поэтому при обратном распространении ошибки в нейрон с этим значением передается ошибка с предыдущего слоя. Таким образом на сверточном слое, осуществляющем функцию максимума, большинство нейронов будет иметь нулевую ошибку.

2.3 Adam

Adam (Adaptive Moment Estimation) [8] - стохастический алгоритм градиентного спуска, основанный на оценке моментов 1-го и 2-го порядка. Алгоритм оценивает моменты первого и второго порядка, используя экспоненциальную скользящую среднюю.

Итак, приближение момента первого порядка:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

Момента второго порядка:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

m_t,v_t - приближения моментов первого (среднее) и второго порядка соответственно (не центрированное отклонение) градиента. Инициализируются оба вектора как нулевые вектора, и во время всего обучения их компоненты близки к нулю, особенно если β_1, β_2 близки к 1.

Для того чтобы избежать таких маленьких значений моментов вычисляются исправленные приближения:

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

После расчета приближений к моментам начинается процесс обновления весов, проходящий по правилу:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon} \hat{m_t}$$

Авторами метода рекомендуется брать $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$.

Таким образом, алгоритм подстраивает скорость обучения η под каждый параметр сети отдельно. Это позволяет выходить из локальных минимумов в процессе обучения, что ускоряет весь процесс, и дает возможности достичь более высоких результатов классификации. Считается что данный метод является наиболее современным и оптимальным

адаптивным алгоритмом обучения. В данной работе настройка весов сети будет проводиться с помощью вышеописанного алгоритма.

2.4 Предобработка данных

Перед подачей изображений на вход нейронной сети, необходимо их предобработать. Первое что нужно сделать, это найти и выделить на изображении прямоугольник, содержащий лицо, и вырезать его. Для этого будем использовать алгоритм Виолы-Джонса

Метод использует технологию скользящего окна, то есть по изображению двигается некая рамка небольшого размера, и с помощью классификаторов определяет есть ли внутри рамки лицо.

Плюсы метода:

1. Высокая скорость работы алгоритма распознавания.
2. Возможность обнаружить несколько лиц на одном изображении.

Минус метода в том, что он труднообучаем, так как требует длительное время на обработку тренировочных данных. К счастью в открытом доступе имеются хорошо обученные классификаторы. Их мы и будем использовать для определения положения лица.

Выделим основные концепции метода: сперва создается и обучается база данных, состоящая из признаков, их паритета и границы. Далее алгоритм ищет объекты из полученной базы данных на разных масштабах изображения. На выходе алгоритма мы получаем множество объектов на разных масштабах, и следующим шагом является сортировка результатов метода.

В качестве признаков для алгоритма распознавания используются признаки Хаара. Так, например, было замечено, что для всех лиц области глаз темнее области щек из этого получилась следующая маска (рис. 7)

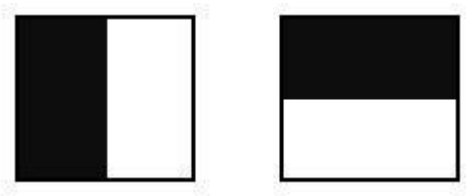

Рисунок 7

По аналогии с этой маской были построены остальные признаки Хаара. Каждая маска характеризуется размером светлой и темной области, пропорциями, а также минимальными размерами. Каждый признак дает значения перепадов яркости по оси X и Y. Таким образом значение признака вычисляется как:

$$F = X - Y$$

где X, Y суммы значений яркости точек покрываемых светлой и темной частью признака соответственно.

Для ускорения вычислений признаков используется интегральное представление изображения. Вычисляется матрица:

$$G(x,y) = \sum_{i=0,j=0}^{i \le x, j \le y} I(i,j)$$

Где $I(i,j)$ - значение яркости исходного изображения. Таким образом каждый элемент матрицы - это сумма яркостей пикселей в прямоугольнике от $(0,0)$ до (x,y). С помощью такого представления изображения расчет признаков Хаара значительно ускоряется.

Сам процесс обучения базы данных рассматриваться не будет, так как будет использована уже обученная модель, показывающая отличные результаты работы.

Перед процессом распознавания необходимо заранее определить количество масштабов рамки, на которых будет происходит сканирование изображений, и на сколько эти масштабы будут отличаться друг от друга.

Затем алгоритм проходит окнами всех масштабов находя все варианты расположения признаков. Все найденные признаки подаются классификатору, который выносит решение есть ли на фотографии искомый объект.

Результатом метода является множество прямоугольников определяющее вероятное положение лиц на изображении (рис.8).

Рисунок 8

Определив положения лица на фотографии, вырежем его, тем самым уменьшая размер фотографии. При вырезании лица увеличим прямоугольник, найденный по алгоритму распознавания на 40 пикселей с каждой стороны (рис.9)

Рисунок 9

Далее изменим размер всех изображений до единого значения. Обычно размер выбирается после расчета среднего размера изображения из выборки, таким образом изменяя размеры фотографий до этого значения мы потеряем минимальное количество данных.

Глава 3 Выбор базы данных и параметров сетей

3.1 База данных

Крайне важным пунктом в решении поставленной задачи является вопрос выбора базы данных. От её грамотного выбора напрямую зависит точность и способность нейронной сети обобщать результаты. В свободном доступе существует не так много открытых коллекций с подходящими задаче данным. Требования, которые выдвигаются для баз в рамках данного исследования:

1. В состав базы должны входить фотографии людей, с различимым, открытым лицом.
2. Маркированные для каждой фотографии возраст и пол человека.
3. Открытость базы.
4. Большой объем базы.
5. Хорошее качество изображений.

В целом существует не так много баз данных на подобную тематику, объясняется это тем, что довольно проблематично составить достоверную и обширную коллекцию таких изображений, необходимо на протяжении всей жизни человека делать его снимки с заданной периодичностью, что, естественно, очень трудоемко. На помощь приходят знаменитости, медийные личности, жизнь которых напрямую связана с нахождением перед камерой. Например, кинозвезды, чью внешность можно сопоставить с их возрастом, благодаря фильмам в которых они снимались. Есть в данном методе и проблема, зачастую в кино актеров гримируют, и возраст их персонажей может существенно отличаться от их возрастов, что довольно серьезно сказываться на качестве классификации и времени обучении сети.

Можно выделить несколько основных баз данных с маркированием возраста и пола:

1) The IMDB-WIKI dataset [9]

   Сравнительная новая база данных, но в тоже время самая крупная доступная в открытом доступе. Для ее составления были взяты 20,000 самых популярных актера по версии сайта IMDb, записаны даты их рождения и пол, и скачаны все доступные изображения, связанные с ними. Причем для каждого изображения указана дата съемки что позволяет определить возраст человека по фотографии. В дополнении к этому были собраны фотографии этих знаменитостей с Wikipedia.

2) MORPH Database [10]

   Одна из крупнейших доступных баз данных содержащая в себе две части. Первая состоит из серий фотографий, сделанных за 30 летний промежуток над 515 людьми, показывая процесс их старения за это время. Вторая часть содержит фотографии более 13,000 людей сделанные за 4 года, представлены возраста от 15 до 77 лет, на каждого человека приходится примерно по 4 фотографии. Все фотографии сделаны в условиях реальной жизни, что особенно ценно для обучения сети.

   Из минусов, база находится не в открытом доступе, доступ к ней можно получить только платно и с согласия создателей.

3) The OUI-Adience Face Image Project [11]

   Эта коллекция данных создавалась специально для решения задач распознавания пола и возраста людей. Авторы ставили своей целью сделать базу данных наиболее правдоподобной, чтобы системы, обученные с ее помощью, справлялись с реальными задачами. Для этого также в состав базы включено как можно более широкое разнообразие внешностей людей, поз и освещения.

К сожалению возможность скачать данную базу данных в настоящий момент отсутствует.

Таблица 1. Сравнение баз данных

Название	Количество фотографий	Количество моделей	Реальные условия съемки	Открытость
IMDB-WIKI	523,051	20,284	Нет	Да
MORPH	55,000	13,000	Да	Нет
OUI-Adience Face dataset	26,580	2,284	Да	Нет

На основе проведенного сравнения, для данного исследования была выбрана коллекция IMDb+Wiki как самая обширная и доступная на данный момент база данных. Еще одним ее плюсом является достойное качество и размер изображений.

3.2 Фреймворк Caffe

В реализации решения в данной работе был использован фреймворк Caffe [13] реализующий множество методов глубокого обучения. Преимущества данного фреймворка в первую очередь в скорости и модульности процессов. Фреймворк разрабатывается компанией Berkeley AI Research.

Необходимость использования фреймворка была выявлена при оценке скорости работы и результатов, полученных с помощью, собственноручно реализованной сверточной сети. Сравнение проводилось на базе рукописных цифр MNIST. Данная коллекция состоит из 60,000 тренировочных изображений и 10,000 тестовых изображений. При решении задачи распознавания цифр по этой базе были получены следующие результаты:

Таблица 2. Сравнение фреймворка Caffe и собственной сети

	Caffe	Собственная сеть
Точность на тестовом множеством	0.9973	0.9734
Время на обучение	1-2 часа	5-7 часов

Тесты проводились на идентичных архитектурах с одинаковыми алгоритмами обучения. Таким образом используя фреймворк мы выиграем не только в скорости обучения сети, но и в качестве решения поставленной задачи. Выигрыш в скорости фреймворка в первую очередь объясняется тем, что вычисления в нем проводились на графическом процессоре компьютера, в то время как собственная сеть использует только CPU. Также большую роль сыграли хорошо отлаженные алгоритмы обучения сети, и более эффективное использование памяти, это повлияло и на скорость, и на точность классификации.

Скорость обучения крайне важна для решения данной задачи по причине того, что мы имеем огромную базу данных достаточно крупных изображений, обработка которых требует большого количества вычислительных ресурсов. Соответственно, для качественного решения задачи нам необходимо пройти по всей базе данных неоднократно, и чем быстрее мы сможем это сделать, тем лучше. Поэтому для эффективного

решения требуется перевод вычислительных процессов на графический процессор компьютера.

Таким образом для достижения максимально качественного решения поставленных задач необходимо использовать специализированный инструмент – фреймворк Caffe.

Для ускорения обучения сетей, воспользуемся технологией transfer learning. В ее основе лежит простая идея: мы будем обучать сеть, которая уже была предобученна для другой задачи классификации. Таким образом мы существенно ускорим процесс, потому что на момент обучения в сети, уже будут сформированы фильтры фиксирующие разнообразные особенности изображений, например, углы, и нам останется только правильно классифицировать выделяемые признаки под нашу задачу. Конечно, на деле многие фильтры сети приходится переобучать, поэтому настройка весов сети даже с предобученной сетью будет занимать весьма продолжительное время.

3.3 Архитектура сетей

Для решения поставленных задач будут созданы и обучены две сверточные нейронной сети, одна для определения пола, другая для определения возраста.

Сети будут иметь схожую архитектуру, отличающуюся только количеством нейронов на последнем слое сети. Обе сети основаны на модели VGG-16. Сети принимают на вход изображения размером 224×224 и содержат в себе 16 слоев.

Таблица 3. Архитектура сети

Номер	Тип слоя	Размеры	Шаг	Количество	Размерность
1	Сверточный	3×3	1	64	$224 \times 224 \times 64$
2	Сверточный	3×3	1	64	$224 \times 224 \times 64$
3	Пуллинг	2×2	2	64	$112 \times 112 \times 64$
4	Сверточный	3×3	1	128	$112 \times 112 \times 128$
5	Сверточный	3×3	1	128	$112 \times 112 \times 128$
6	Пуллинг	2×2	2	128	$56 \times 56 \times 128$
7	Сверточный	3×3	1	256	$56 \times 56 \times 256$
8	Сверточный	3×3	1	256	$56 \times 56 \times 256$
9	Сверточный	3×3	1	256	$56 \times 56 \times 256$
10	Пуллинг	2×2	2	256	$28 \times 28 \times 256$
11	Сверточный	3×3	1	512	$28 \times 28 \times 512$
12	Сверточный	3×3	1	512	$28 \times 28 \times 512$
13	Сверточный	3×3	1	512	$28 \times 28 \times 512$
14	Пуллинг	2×2	2	512	$14 \times 14 \times 512$
15	Сверточный	3×3	1	512	$14 \times 14 \times 512$
16	Сверточный	3×3	1	512	$14 \times 14 \times 512$
17	Сверточный	3×3	1	512	$14 \times 14 \times 512$
18	Пуллинг	2×2	2	512	$7 \times 7 \times 512$
20	Полносвязный	-	-	4096	4096×1
21	Полносвязный	-	-	4096	4096×1
22	Softmax	-	-	2/101	$2 \times 1/101 \times 1$

Таким образом на последнем слое сверточной сети формируются карты признаков размером $7 \times 7 \times 512$. Последовательное применение небольших 3×3 сверточных слоев углубляет формируемые признаки что улучшает качество распознавания. Размер карты признаков на сверточных сетях остается тем же, так как при обработке границ мы используем метод продления, заполняя нулями все пиксели на внешней границе матрицы признаков. Размер карты признаков уменьшается только на слое субдискретизации следующем после несколько сверточных слоев. Таким образом мы уменьшаем вероятность того, что при пуллинге будет потеряны важные признаки изображения.

Слой softmax используется для моделирования вероятностного распределения. Функция активации таких нейронов:

$$y_i = \frac{e^{x_i^L}}{\sum_{j=1}^{n} e^{x_j^L}}$$

Видно, что выход каждого нейрона будет зависеть от сумматоров всех остальных нейронов слоя. Так же , моделируя вероятностное распределение, сумма всех выходов нейронов softmax слоя равняется 1. Таким образом на выходе сети каждый результат *i-го* нейрона будет показывать вероятность принадлежности исходного изображения *i-му* классу.

С softmax слоем необходимо использовать особую функцию потерь, так как необходимо реализовать меру для сравнения двух вероятностных распределений. В этом помогает функция перекрестной энтропии:

$$E(x) = -\sum_j t_j \log(y_j)$$

где t - эталонный ответ.

Выводы

В рамках проведенного исследования были обучены две нейронные сети: а) Сеть для распознавания пола человека б) Сеть для распознавания возраста человека.

Для контроля обучения сети коллекция примеров была разбита на три части:

1) Валидационное множество: состоит из 1,000 обучающих примеров. Будем использовать это множество для контроля процесса обучения, по зависимости ошибки на данном множестве от количества итераций можно сделать выводы о том, насколько правильно подобраны параметры сети, и работают ли вообще методы обучения. Проводить тесты на этом множестве будем каждые 2,000 итерации.
2) Тестовое множество: состоит из 10,000 примеров, не виденных ранее сетью примеров. Используется этот набор для финального контроля качества работы сети. Проверять будем раз в 7.000 итераций и по результатам проверки будем судить о качестве работы сети.
3) Обучающее множество: состоит из оставшихся примеров из базы данных. Эти примеры используется в процессе обучения сети, поэтому важно чтобы эта часть базы данных была максимально обширной

   Для сети, а) получены следующие результаты:

Таблица 4. Результаты сети а)

	Валидационное множество	Тестовое множество
Точность	0.97	0.91
Ошибка	0.03	0.09

Правило вычисление ошибки на обучающем примере для сети а):

$$E(x_k) = \begin{cases} 1 & if\ y_k \neq f(x_k) \\ 0 & otherwise \end{cases}$$

То, как проходило обучение, можно увидеть на графике зависимости ошибки на тестовом множестве от количества итераций (рисунок 10):

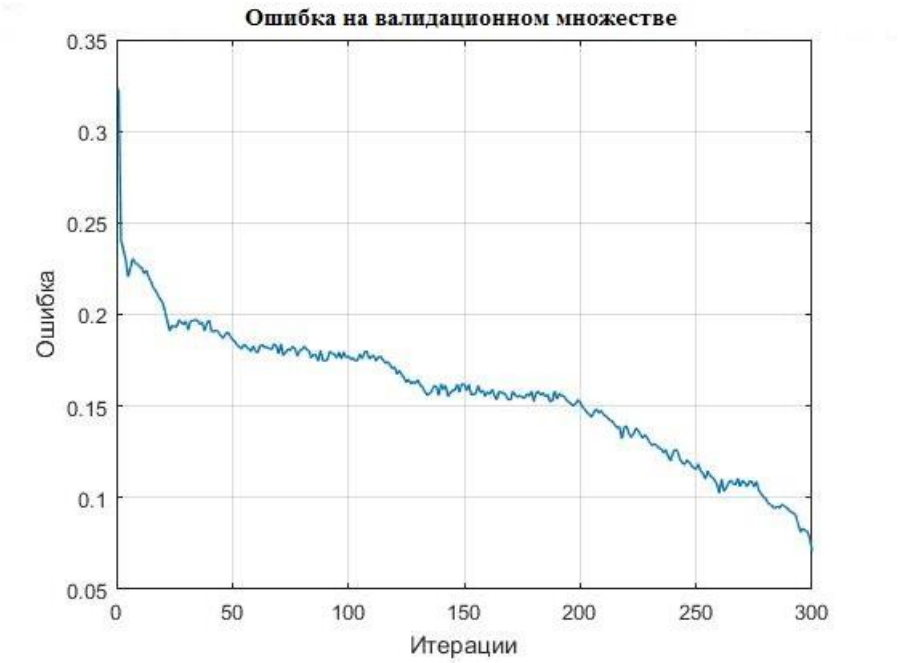

Рисунок 10

Таким образом, по графику видно, что обучения проходило равномерно, следовательно, параметры обучения, и алгоритм обучения хорошо подобраны под поставленную задачу.

В итоге ошибку на тестовом множестве удалось сократить до 0.09:

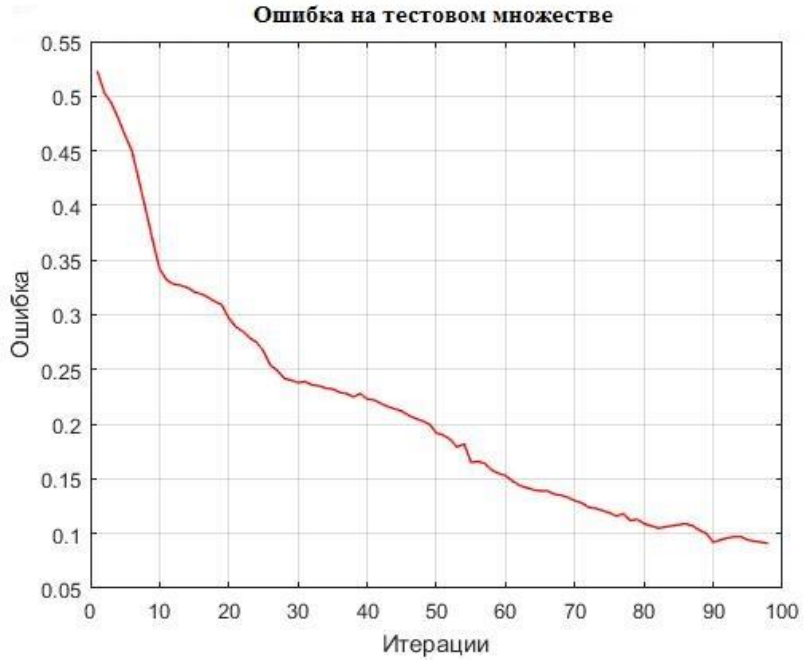

Рисунок 11

Таким образом достигнуты желаемые результаты в работе сети (рисунок 11). Система адекватно оценивает пол человека, и показывает хорошую общность результатов даже на незнакомых ей фотографиях.

Сеть б):

Ошибку примера для сети по определению возраста на тестовом множестве будем считать следующим образом:

$$E(x_k) = \begin{cases} 1 & if\ |y_k - f(x_k)| \geq 2 \\ 0 & otherwise \end{cases}$$

Таблица 5. Результаты сети б)

	Валидационное множество	Тестовое множество
Точность	0.78	0.74
Ошибка	0.22	0.26

Процесс обучения сети представлен на графике (рис. 12):

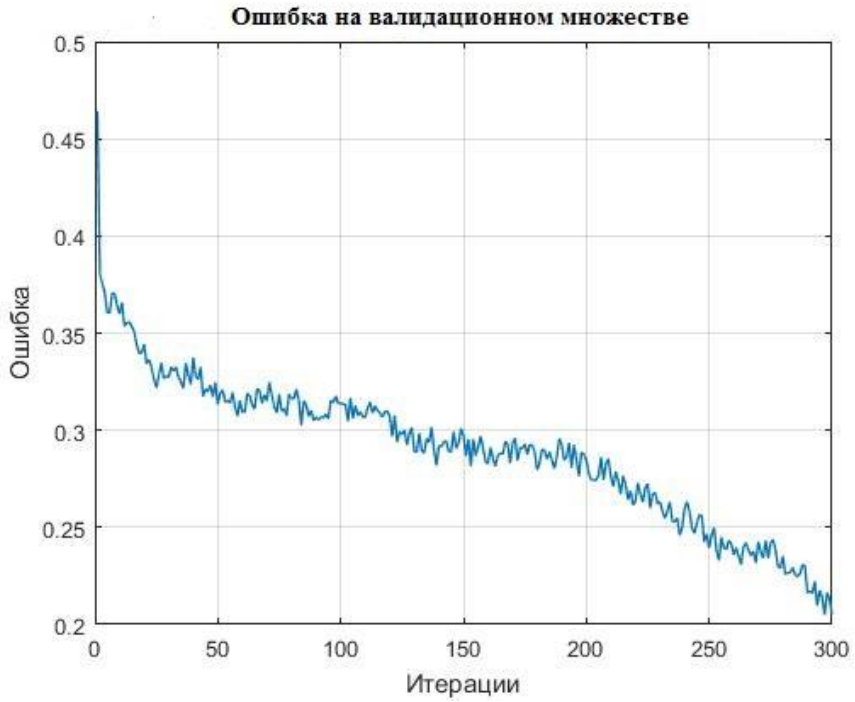

Рисунок 12

По поведению кривой, можно судить о том, что задача определения возраста оказалась гораздо сложнее задачи определения пола. Ошибка сильно скачет в своих значения, но при этом наблюдается тенденция ее уменьшение с ростом количества итераций.

По графику ошибки на тестовом множестве видно, что не удалось достичь результатов таких же хороших, как для сети а) (рис. 13):

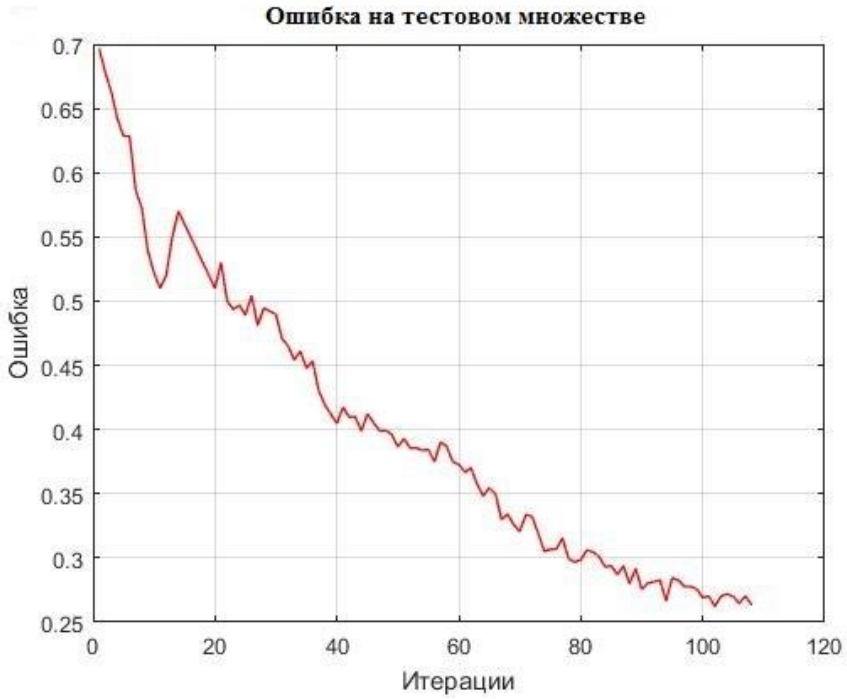

Рисунок 13

Таким образом ошибка на тестовом множестве в конце процесса обучения составляла 0.26. Результат оказался хуже, чем для сети по распознаванию пола из-за того, что в сети б) гораздо больше выходных классов, что существенно ухудшает качество классификации. При этом сеть достаточно хорошо показывает себя при определении примерного возраста, ошибаясь обычно на не больше чем на 1-2 года.

Заключение

В данной работы было предложено решение, позволяющих определить пол и возраст человека по его фотографии. В состав решения входят методы предобработки изображений и две обученные сверточные нейронные сети. Были использованы современные технологии обучения, позволившие ускорить процесс настройки параметров сетей. В целом, удалось достичь хорошей точности прогнозирования, что позволяет применять решение в реальных задачах. Пример классификации проведенной нейронными сетями можно увидеть на рисунке 14.

Эталон	
Пол	Возраст
Женский	25
Предсказанные значения	
Пол	Вероятность
Мужской	0.01
Женский	0.99
Возраст	Вероятность
24	0.42
23	0.33
22	0.25

Рисунок 14

Список литературы

1. M. B. Stegmann Analysis and Segmentation of Face Images using Point Annotations and Linear Subspace Techniques. Technical report IMM-REP-2002-22, Informatics and Mathematical Modelling,
2. Yan S-C, Zhou X and Liu M. Hasegawa-Johnson, M., Huang, T.S. (2008) Regression from patch-kernel. IEEE Conference on CVPR, pages 1-8
3. Geng X, Zhou Z-H, Zhang Y, Li G, Dai H. (2006) Learning from facial aging patterns for automatic age estimation, In ACM Conf. on Multimedia, pages 307– 316
4. Geng X, Zhou Z-H, Smith-Miles K. (2007) Automatic age estimation based on facial aging patterns. IEEE Trans. on PAMI, 29(12): 2234–2240
5. Guo G, Mu G, Fu Y and Huang T.S. (2009) Human age estimation using bio-inspired features. IEEE Conference on CVPR, pages 112-119.
6. D. Maturana, D. Mery, A. Soto. Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor classification. *In Proc. of the XXVIII International Conference of the Chilean Computer Science Society, IEEE CS Society*, 2009.
7. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.
8. Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, 1–13
9. Rasmus Rothe and Radu Timofte and Luc Van Gool. (2016). Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision (IJCV)*.
10. Ricanek, K., Tesafaye, T.: Morph: A longitudinal image database of normal adult age-progression. In: Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on. pp. 341–345. IEEE (2006)

11. Eran Eidinger, Roee Enbar, and Tal Hassner, *Age and Gender Estimation of Unfiltered Faces*, Transactions on Information Forensics and Security (IEEE-TIFS), special issue on Facial Biometrics in the Wild, Volume 9, Issue 12, pages 2170 - 2179, Dec. 2014 (PDF)
12. Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv Preprint arXiv:1408.5093*.