Содержание

[ВВЕДЕНИЕ 3](#_Toc101041581)

[1. Постановка задачи 5](#_Toc101041582)

[2. Описание датасета 6](#_Toc101041583)

[3. Инструментарий для обучения модели 10](#_Toc101041584)

[3.1. Функция активации 10](#_Toc101041585)

[3.2. Метрики 12](#_Toc101041586)

[3.3. OCR 14](#_Toc101041587)

[4. Построение модели 16](#_Toc101041588)

[5. Сравнение моделей 23](#_Toc101041589)

[ЗАКЛЮЧЕНИЕ 28](#_Toc101041590)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ 29](#_Toc101041591)

[ПРИЛОЖЕНИЕ. ИСХОДНЫЙ КОД ПРОГРАММЫ 30](#_Toc101041592)

# ВВЕДЕНИЕ

Выполнение курсовой работы способствует изучению темы распознавания текста на различных изображениях благодаря использованию машинного обучения.

На сегодняшний день вопрос оцифровки информации все еще актуальный, население различных стран испытывает трудности, когда пытается переписать номер телефона с картинки или перепечатать важную информацию, которая написана от руки или же просто существует только в виде изображения. Наиболее известным примером, с которым сейчас имеют дело – это оцифровка книжной базы знаний, которая сейчас доступна людям. Решением этой задачи озаботилась компания Google, реализуя проект Google Books. Еще в 2010 году оценили книжный фонд в число свыше 130 миллионов экземпляров, и эта цифра постоянно растет. В период с 2004 по 2009 год, ИТ гиганту удалось добавить в свою базу оцифрованных изданий свыше 10-ти миллионов экземпляров.

Для себя я решил выбрать эту тему, чтобы изучить имеющиеся инструменты, которые помогают в решении описанной проблемы. Провести исследование на различных изображениях и разобраться в процессе проектирования и разработки нейронной сети. Установить необходимый минимум, который требуется для обучения сети, чтобы начать получать удовлетворительные результаты по распознаванию текста. Кроме того, будет проведена аналитика по одной модели, но у которой различаются настройки обучения.

Основной целью работы является – построение модели машинного обучения, сравнения эффективности в задачах распознаванию текста и составления заключительной точки зрения, которая будет основана на результатах проделанной работы.

В данной работе будут рассмотрены как нейронные сети, так и средства OCR.

Для обучения был использован известный набор данных «EMNIST dataset», который содержит в себе различные варианты написания букв латинского алфавита, а также арабских цифр.

При создании модели использовался Jupiter Notebook. В нем удобно разбивать код python на блоки и проводить детальный анализ различных этапов создания модели.

В качестве аппаратной части для обучения сети был выбран стационарный ПК с процессором AMD 3700X с жидкостным охлаждение. Такая система позволит работать процессору на пределе возможностей без троттлинга, который происходит из-за перегрева ЦПУ при сильных нагрузках.

1. Постановка задачи

Перед тем, как начинать реализовывать программную часть курсовой работы, сначало нужно понять, какая задача стоит перед обученной нейронной сетью. В моем представлении использование должно быть предельно просто и состоять из трех этапов:

* Получение изображения путем фотографирования на смартфон или снимок экрана компьютера;
* Загрузка полученного изображения;
* Быстрое получение результата с абсолютной точностью.

Для достижения этих задач необходимо изучить модель создания нейронной сети. Понять по какому принципу она принимает данные и каким образом будет происходить дальнейшая обработка. Работа нейронов происходит по следующему принципу:

* Суммирование сигналов, которые поступают со входов;
* У каждого входа свой коэффициент передачи;
* Для результата применяется «функция активации».

Данная функция бывает трех типов:

* Линейной;
* В виде сигмоиды;
* Прямоугольной.

Выбор датасета, на основе которого будет происходить обучение разрабатываемой нейронной сети.

Не стоит задачи разработать конкурентоспособный аналог, известным в широких кругах, инструментам по распознаванию текста. Однако заниматься исследованием уже ранее созданных оптических распознавателей символов будет неправильным. Поэтому нужно разработать просто решение, которое не потребует глубокого изучения в области машинного обучения.

Сбор необходимой информации, с помощью которой можно будет получит подробную статистику и провести анализ проделанной работы.

1. Описание датасета

Чтобы разработать модель на основе нейронной сети, способную определять текст на изображении, нужен высококачественный датасет, в котором будет достаточно картинок, на основе которых можно будет проводить обучение.

Для данной задачи мною был выбран достаточно известный набор данных Extended MNIST или же «EMNIST». Этот датасет хорошо зарекомендовал себя благодаря большому числу людей, которые использовали его в своих проекта. Большой объем изображений позволяет натренировать сеть на всевозможные вариации написания текста.

Датасет включает в себя 814.255 изображений латинских букв и арабских цифр в формате «.png». Для обучения используется 697.932 изображений 28 на 28 пикселей, а также 116.323 изображений для тестирования. В наборе данных присутствует разделение букв как на прописные (таблица 1), так и на

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 7.469 | 4.526 | 11.833 | 5.341 | 5.785 | 10.622 | 2.964 |
| H | I | J | K | L | M | N |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 3.673 | 13.994 | 4.388 | 2.850 | 5.886 | 10.487 | 9.588 |
| O | P | Q | R | S | T | U |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 29.139 | 9.744 | 3.018 | 5.882 | 24.272 | 11.396 | 14.604 |
| V | W | X | Y | Z |
| 31 | 32 | 33 | 34 | 35 |
| 5.433 | 5.501 | 3.203 | 5.541 | 3.165 |

Таблица 1 – Прописные латинские буквы, label в датасете, число образцов

строчные (таблица 2) экземпляры. В EMNIST 358.249 экземпляров из общего из общего объема занимаю всевозможные вариации написания цифр (таблица 3-4) от 0 до 9.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g |
| 36 | 37 | 38 | 39 | 40 | 41 | 42 |
| 11.677 | 6.012 | 3.286 | 11.860 | 28.723 | 2.961 | 4.276 |
| h | i | j | k | l | m | n |
| 43 | 44 | 45 | 46 | 47 | 48 | 49 |
| 10.217 | 3.152 | 2.213 | 2.957 | 17.853 | 3.109 | 13.316 |
| o | p | q | r | s | t | u |
| 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 3.215 | 2.816 | 3.499 | 16.425 | 3.136 | 21.227 | 3.312 |
| v | w | x | y | z |
| 57 | 58 | 59 | 60 | 61 |
| 3.378 | 3.164 | 3.292 | 2.746 | 3.176 |

Таблица 2 – Строчные латинские буквы, label в датасете, число изображений

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 34.585 | 38.374 | 34.203 | 35.143 | 33.535 | 31.416 | 34.232 | 35.754 | 33.946 | 33.847 |

Таблица 3 – Арабские цифры, число изображений для обучения

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5.778 | 6.330 | 5.869 | 5.969 | 5.619 | 5.190 | 5.705 | 6.139 | 5.633 | 5.686 |

Таблица 4 – Арабские цифры, число изображений для тестирования

Также можно удостоверится в том, что все изображения в том, что все изображения в датасете придерживаются одних правил:

* Изображения состоят из черного и белого цвета (рисунок 1);
* Изображения отзеркалены по вертикали;
* Изображения повернуты на 90 градусов;
* Изображения имею идентичный размер 28 на 28 пикселей;



Рисунок 1 - Изображение различных символов набора

* Отсутствует шумы;
* У символов разный размер и толщина (рисунок 2);
* Непрямой шрифт;



Рисунок 2 - Изображение вариаций одного символа

Ко всему прочему, стоит заметить, что наборах присутствуют буквы которые имеют дефекты в написании (), что положительно отразится на обученной модели. Так как в итоговый код может быть передано изображение, на котором символ будет написан не полностью или же у изображения, может быть, несильная расфокусировка.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - Неаккуратное написание цифры "8"

Исходя из этого можно уверенно сказать, что для обучения сети есть высококачественный набор данных, который состоит из 62 классов, имеет разделение на обучающие и тестовый набор (рисунок 4) и готов к началу работы с ним.

Изображение выглядит как текст, млекопитающее

Автоматически созданное описание

Рисунок 4 - Разделение изображений на обучающие и тестовые

1. Инструментарий для обучения модели
   1. Функция активации

К ключевым этапам разработки нейронной сети относится выбор функции активации нейронов. Так как цифровой нейрон по своей сути считает взвешенную сумму, которая находится на его входах, потом прибавляет к этому смещение, принимает решение по дальнейшему использованию или исключению полученного значения.

Функция активации устанавливает итоговое значение для нейрона исходя из получившегося результата взвешенной суммы входов и порогового значения.

Во многом от выбора функции активации определяются функциональные возможности нейронной сети и то, кокай метод будет использоваться для обучения сети.

Для будущей нейронной сети будет использоваться функция активации ReLU (Rectified Linear Unit). Этой активационной функции выражается следующей формулой:

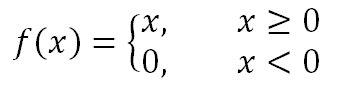


Рисунок 4 - Математическая формула выражения функции активации ReLU

График ReLU имеет следующий вид:

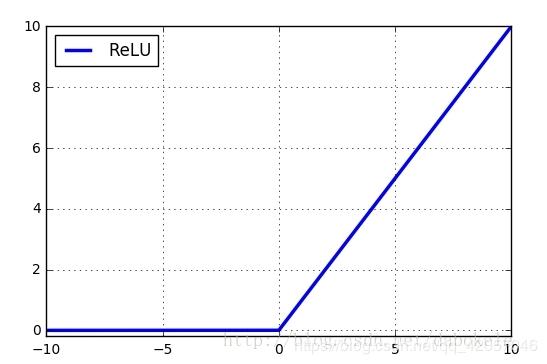


Рисунок 5 - Изображение графика функции ReLU

Особенностью этой функции является то, что она обращает в ноль отрицательные методы матричного умножения матрицы весов на матрицу выходных данных.

Функция хорошо зарекомендовала себя и имеет множество преимуществ, таких как:

* Её производная равна либо единице, либо нулю, и поэтому разрастания или затухания градиентов исключаются из возможных исходов. Ведь при умножении единицы на дельту ошибки, получается дельта ошибки. В случае использования другой функции, например, гиперболический тангенс, то дельта ошибки может уменьшиться, может возрасти или же может остаться без изменений;
* Выбрасывает все ненужные детали в канале при отрицательном выходе;
* Для данной функции нужно меньше вычислительных ресурсов, благодаря своему математическому виду, благодаря чему сеть обучается значительно быстрее;
* ReLU дает возможность блокировать выбираемые нейроны, что делает сеть более быстрой, если делать пересчет на каждую эпоху обучения.

Стоит упомянуть о недостатках ReLU, которых не так много, но все же есть. Проблема заключается в том, что при вхождении отрицательных значений на выходе получается нулевой градиент из-за отсутствия обновления весовых коэффициентов, что в итоге негативно отражается на обучении нейронной сети.

В последнем слое нейронной сети для преобразования необработанных значений прогноза используется логитовый слой. Это результаты до активации, которые не были обработаны на последнем уровне нейронной сети.

Для преобразования набора чисел в вероятности будет использована функция Софтмакс (Softmax). Её результатом будет вектор, который представляет собой распределение вероятностей списка результатов, которые появятся в будущем. Функция разделяет на суммы каждое значение, чтобы сумма всех экспонент равнялась единице.

Входные данные для Софтмакс – это выходные данные Полносвязного слоя, который предшествует ему. По своей сути это выходные данные всей нейронной сети.

* 1. Метрики

В задачи распознавания текста включены основы и методы классификации и идентификации, поэтому правильнее всего, для данной задачи, будет выбор метрики Accuracy (точность).

Перед тем, как приступить непосредственно к метрикам, нужно ввести понятие, описывающее их в концепции классификации ошибок–«Матрица ошибок». (рисунок 6).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 6 - Визуальное отображение матрицы ошибок

На основе изображения выше, формула для расчёта метрики будет выглядеть следующим образом:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 - Формула расчета метрики точности

Обозначение «Актуальные значения» с метками «Положительный» и «Отрицательный» отвечают за метки базовой истины в рассматриваемом наборе данных, а именно соответствует ли переданный символ, например букве «A».

Обозначения внутри матрицы – это не что иное, как счетчики:

* TP (истинно-положительные): положительные примеры, которые модель распознала, как положительные. Предсказано положительное – это правда;
* TN (истинно-отрицательные): отрицательные примеры, которые модель распознала, как отрицательные. Предсказано отрицательное – это правда;
* FP (ложно-положительные): отрицательные примеры, которые были ошибочно распознаны системой, как положительные. Предсказано положительное – это неправда;
* FN (ложно-отрицательные): положительные примеры, которые были ошибочно распознаны, как отрицательные. Предсказано отрицательное – это неправда.

Важно отметить, что прогнозируемые значения должны быть описаны, как положительные и отрицательные, а действительные значения описываются, как истинные и ложные (рисунок 8).

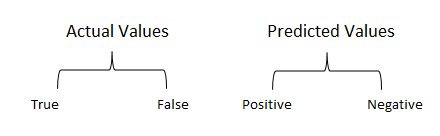


Рисунок 8 - Два класса Матрицы ошибок

* 1. OCR

Оптическое распознавание символов (Optical Character Recognition – OCR) – это технология, которая позволяет проводить автоматический анализ текста и превращать его в данные, которые могут быть обработаны компьютером.

OCR работает следующим образом:

* Предобработка;
* Распознавание;
* Постобработка.

Изображение может быть с дефектами, иметь шумы или быть сделано под непрямым углом. Чем лучше качество обрабатываемого изображения, тем лучше будет качество в результате распознавания. На этапе предобработки создается копия изображения в черно-белом цвете.

У OCR-программ распознавание происходит последовательно, строчка за строчкой, слово за словом и символ за символом. Для начала «Оптическое распознавание символов» производит объединение пикселей, из которых формируются возможные буквы, а полученные буквы формируют возможные слова. После чего система начинает заниматься сопоставлением слов, которые она воссоздала со словами из словаря. Когда такое слово находится, то система помечает для себя его, как распознанное и приступает к следующему слову.

В некоторых OCR системах присутствует возможность изучение страниц на предмет наличия ошибок и последующего исправления этих ошибок. Для таких целях в них предусмотрена проверка правописания, при которой происходит идентификация слова, которые имеют орфографические ошибки. Кроме того, такие программы обучены искать слова, которые стоят рядом. Это, обычно, называют «метод поиска соседа». Это делается для того, что иногда слово, которое система считает, что написано правильно, на самом деле не подходит по контексту, а находится в словаре рядом с необходимым словом. Так же если в словаре находится более одного слова, которые могут потенциально заменить слово с ошибкой, то замена проводиться не будет. Главной проблемой операций по сравнению слов со словарем конечно же являются вычислительные затраты, которые связаны с объемом словаря.

1. Построение модели

В процессе построения модели будет реализована классическая свёрточная сеть, которая будет определять признаки изображения с количеством фильтров 32 и 64.

Сеть включает в себя следующие слои:

* Convolution2D – свёрточный слой для создания ядра свёртки, которое свёртывается на входе слоя, в следствии чего получается тензер выходов. В основном аргументе «input\_shape» будут параметры для одноканальных изображений размером 28 на 28 пикселей. Количество выходных фильтров 32 и 64. Размер ядра на каждом этапе составляет 3 на 3;
* ReLU – ранее описанная функция активации. С её помощью в модель добавляется нелинейность;
* MaxPooling2D – отбирает максимально значимые элементы входных данных. Размер ядра 2х2.
* Dropout – отвечает за исключения на входе с долей отвечающих за это блоков равной 0.25;
* Flatten – нужно для выравнивания входных данных, при этом без отражения на размере;
* Dense – слой для связки каждого входа сети с нейронами с помощью функции активации. Функция идет по элементам, с матрицей весов в ядре, которая создана слоем.

На этапе «выхода» к сети присоединяется «линейная сеть» MLP (Многослойный перцептрон), которая формирует окончательный результат работы нейронной сети.

Чтобы модель не переобучалась на каждой новой эпохе, необходимо установить callback, который будет отслеживать показатели validation accuracy. Обучение модели будет автоматически прерываться если в точности проверочных данных не будет происходить важных изменений на протяжении нескольких эпох.

Изначально было проведено обучение на полном наборе данных. Обучения происходило на протяжении 40-ка эпох и заняло. Обучение заняло около 6 часов. Подробные данные по затраченному времени на каждой эпохе отображены на графике временных затрат (график 1):

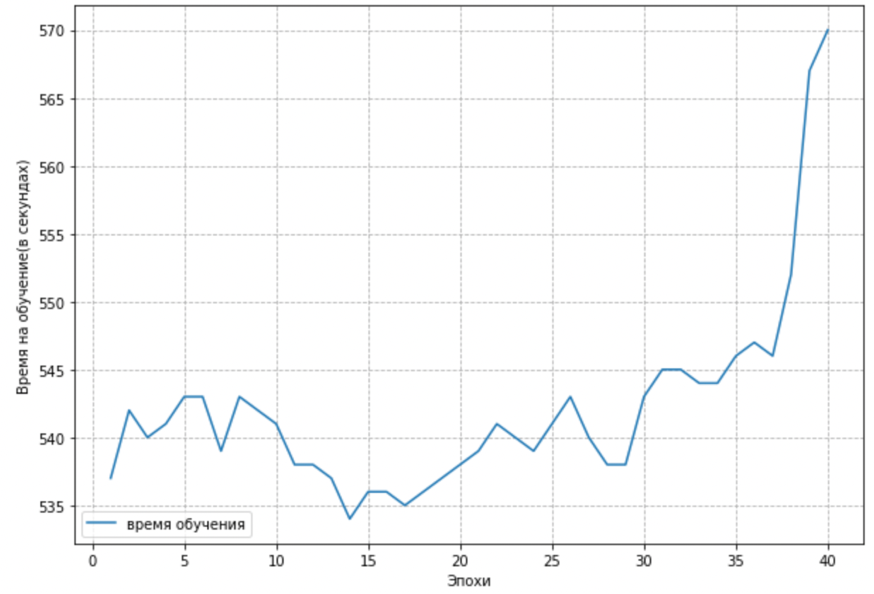


График 1 - Временные затраты на обучение модели из 40 эпох

По графику выше можно сказать, что быстрее всего происходит случайное обучение на 14 эпохе, однако по мере приближения к последней эпохе время обучения сделало скачок и заняло целых 570 секунд.

График точности (график 2) имеет более понятный вид и на его основе можно сделать вывод о том, что с каждой новой эпохой уровень точности повышается, но сама разница точности между эпохами начинает падать.

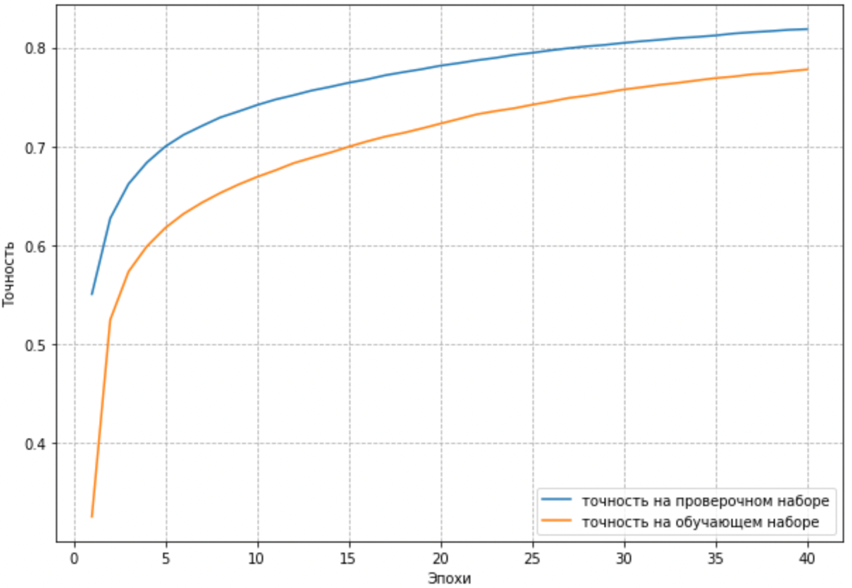


График 2 - График показателей точности модели

Причем разница между первой и второй эпохой составляет более 10% процентов, а точность в 39-ой и 40-ой эпохах различается всего на 0.1%.

Для полного анализа еще стоит рассмотреть график, на котором изображены потери (график 3) модели во время обучения:

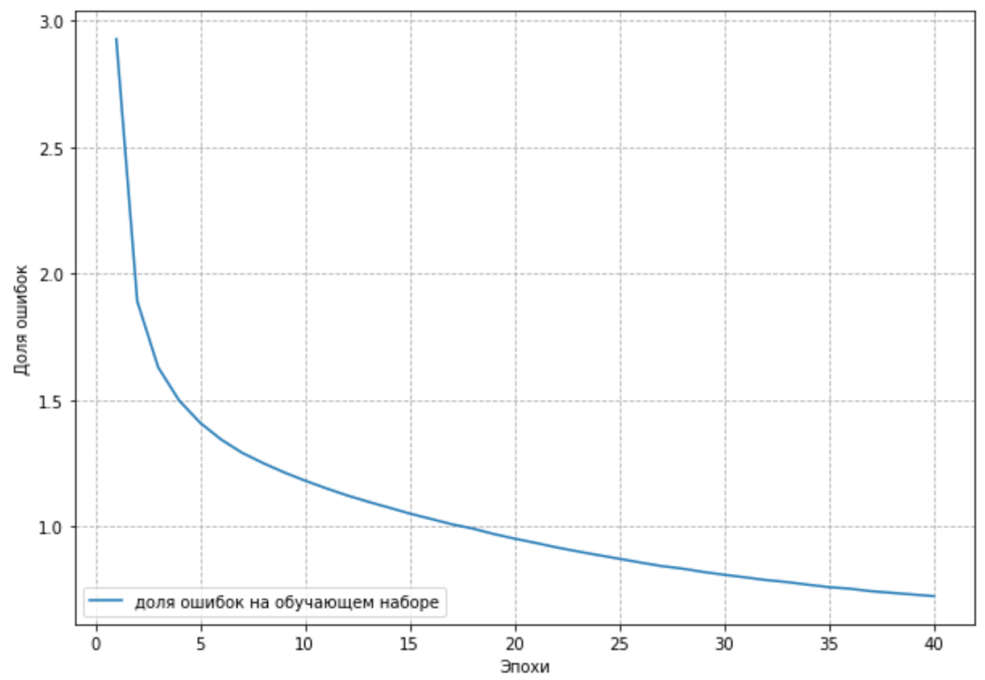


График 3 - График функции потерь

По тому, как себя показывает себя функция потерь можно сделать вывод, что у модели не происходит переобучение, а она стабильно улучшает качество. У функции виден стабильный спад, то есть можно объективно сказать, что на каждой новой эпохе модель допускает все меньше ошибок. Разница между потерями на первой эпохе и последней составляет 220%.

В конце обучения модели точность (accuracy) стала равняться 0.8185, что можно назвать достаточно допустимым результатом, если брать во внимание тот факт, что обучение происходило с нуля.

Так же в целях эксперимента было проведено еще две сессии по обучению модели, в которых были внесены изменения в параметрах. Изменения касались размера датасета и числа эпох.

В первую очередь будет разобран вариант, в котором был так же, как и в предыдущем примере использован полный набор данных, но количество эпох увеличено до 60.

На обучение данной модели было потрачено около 9,42 часов, что 36% дольше, чем при 40-ка эпохах. С тем, как происходит временное распределение можно ознакомиться на следующем изображении (график 4):

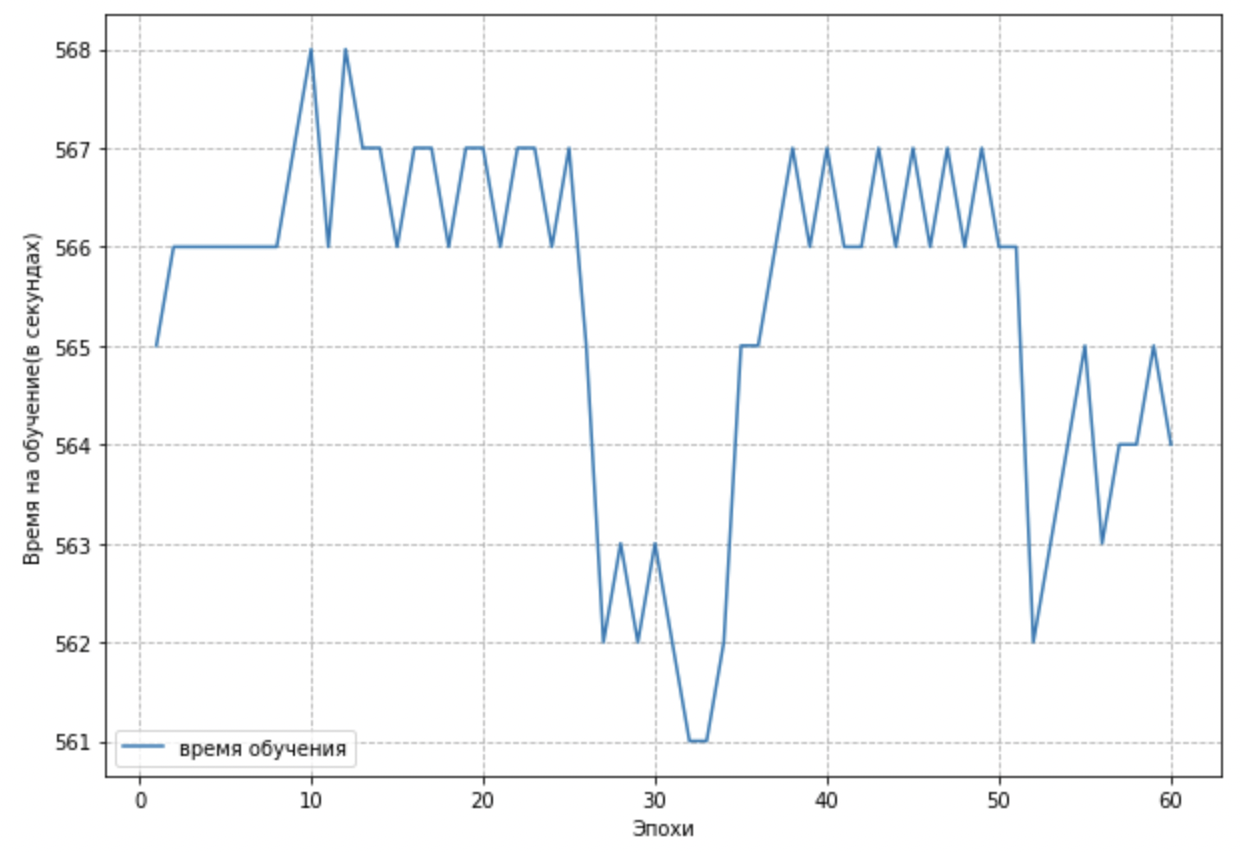


График 4 - Временные затраты на обучение модели из 60 эпох

Показатель точности, как и ожидалось, вырос (график 5), однако показанный рост оказался настолько незначителен, что не оправдывает временные затраты.

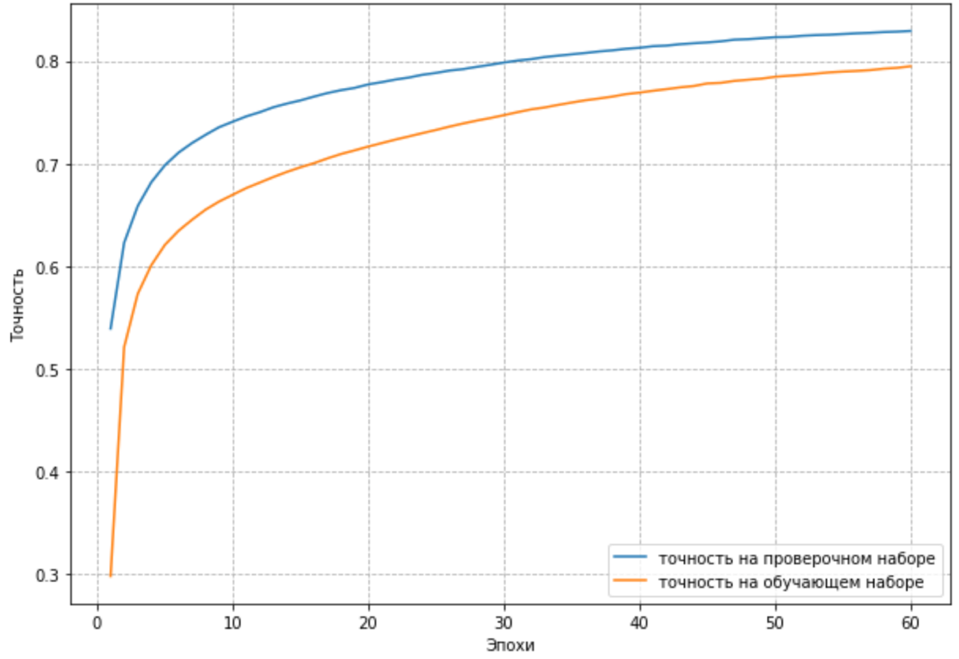


График 5 - График показателей точности модели при 60 эпохах

На графике видно, что показатель точности превысил отметку в 0.8. Точное значение accuracy составляет 0.8291, что на 1,3% выше, чем у предыдущей модели.

Показатель функции потерь также улучшился (график 6), но и как у показателя выше, этот рост нивелируется высокими затратами ресурса времени.

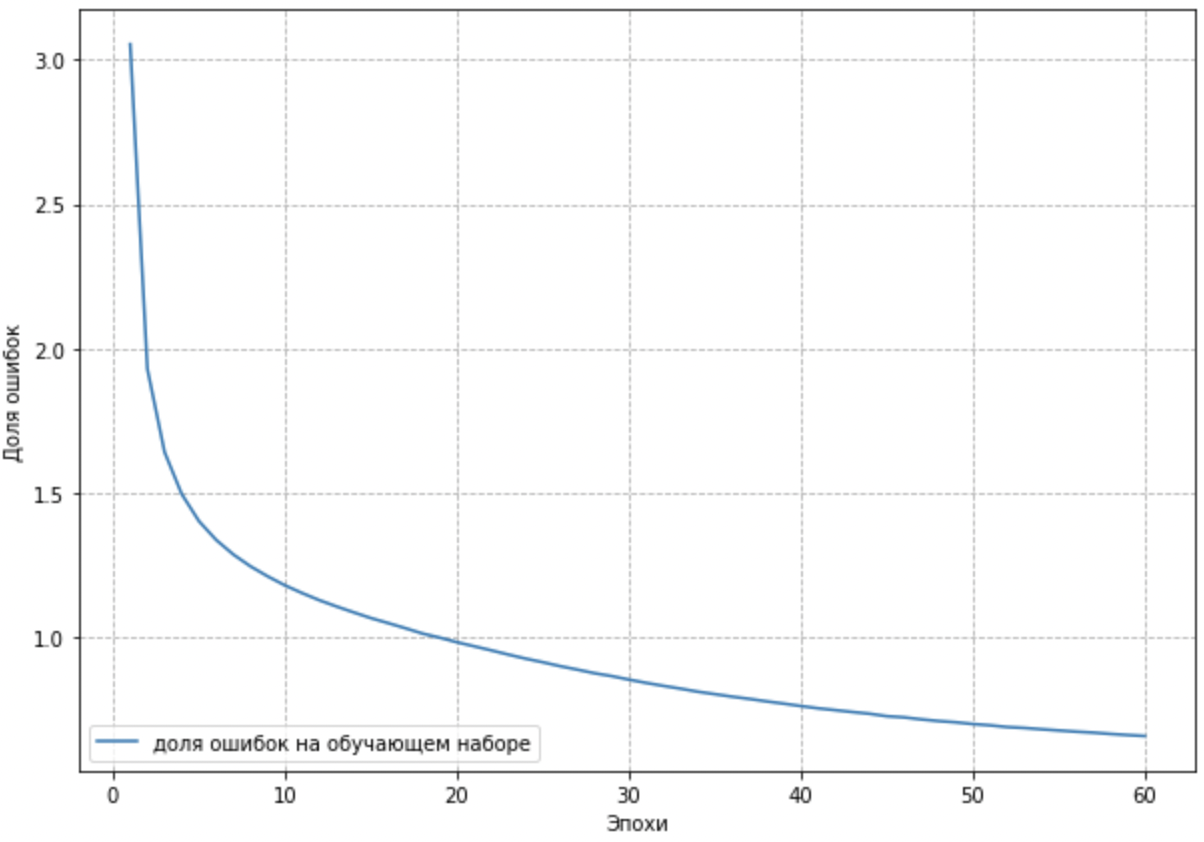


График 6 - График функции потерь на 60 эпохах

Доля ошибок в функции потери сократилась на 9% и составила 0.6557.

Для третьей модели было решено использовать только часть набора данных, а именно 10% от общего числа. Но при этом было увеличено количество эпох до 300. При таких параметрах модель не показала хороших показателей, но с точки зрения эксперимента в ней можно наблюдать интересное поведение.

График точности (график 7) показывает рост, но в отличие от разобранных ранее примеров, он имеет достаточно интересное отличие во внешнем виде.

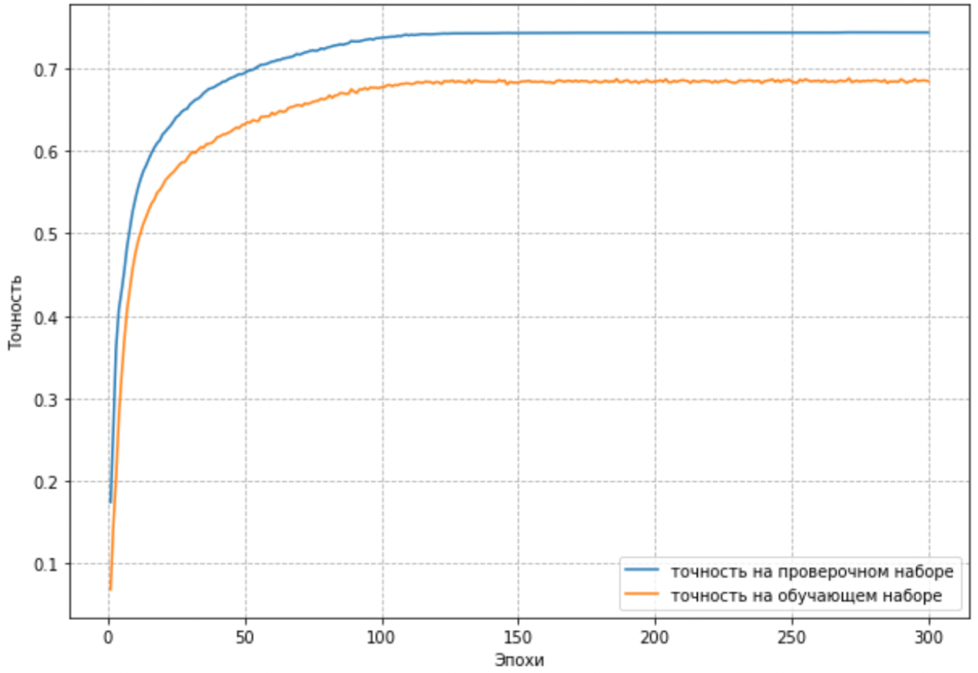


График 7 - График показателей точности модели при 300 эпохах

Как можно заметить график имеет зубчатую структуру, что свидетельствует о том, что рост не является непрерывным, и регулярно во время обучения происходил спад показателя точности. Итоговый показатель в после обучения составлял 0.7434, что ощутимо меньше, чем у предшествующих моделей, но стоит учитывать, что и временной ресурс, который был затрачен на неё, тоже меньше.

На каждую эпоху затрачивалось одинаковое количество времени равное 54 секундам, а всего на обучение потребовалось 4,5 часа. Что на 25% меньше, чем у первой модели и на 52% чем у второй.

Так же функция потерь не имела стабильного снижения. На каком-то этапе она начала стагнировать на 4–6 эпохи и только после этого опускалась ниже. Это можно увидеть на иллюстрации (график 8).

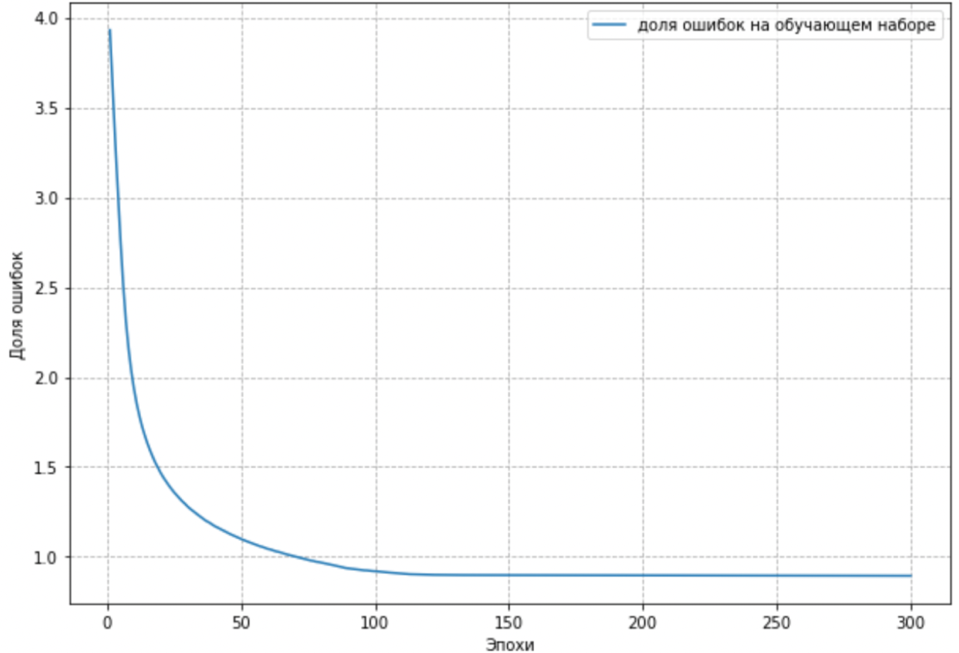


График 8 - График функции потерь на 300 эпохах

Для наглядного ознакомления с разницей в показателях описанных выше моделей можно изучить график сравнения (график 9), на котором визуально отображены числовые различия.

Таким образом выглядит графическое изображение трех показателей обученных моделей.

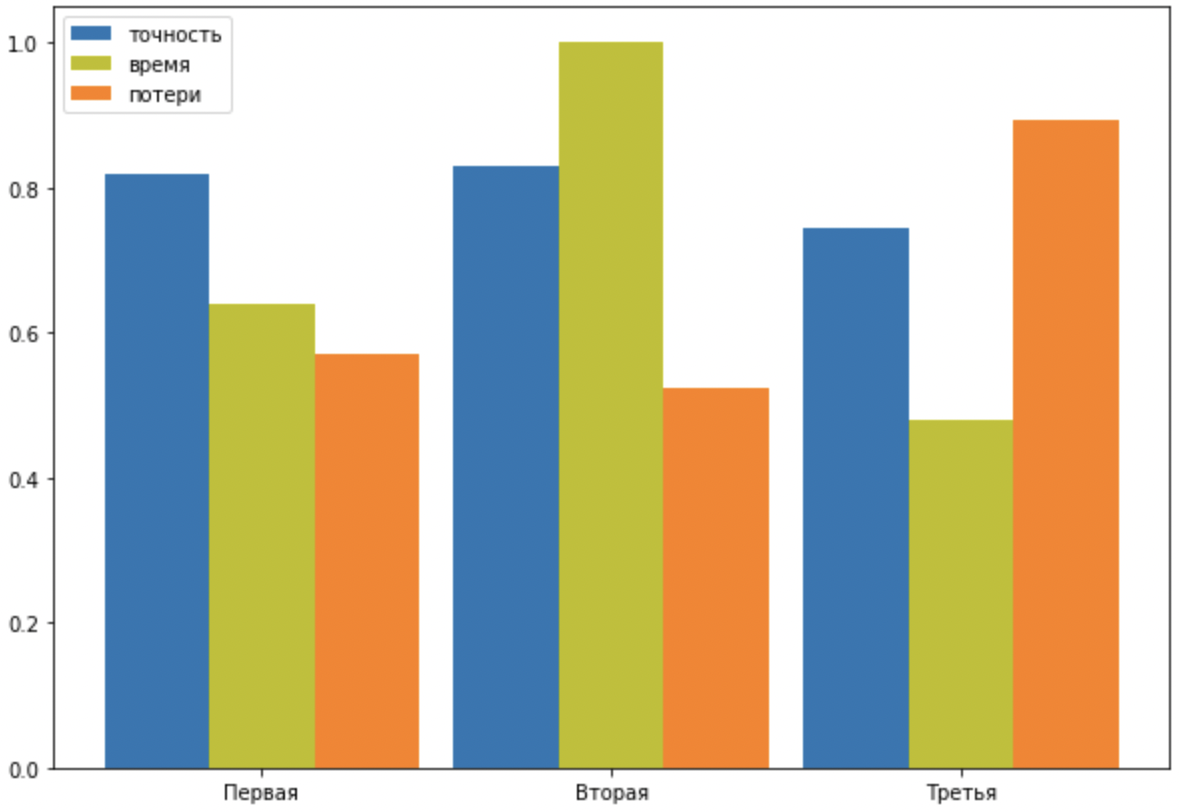


График 9 - Сравнительный график моделей

На основе вышеприведенной статистики, было решено использовать вторую модель для распознавания текста и сравнения с OCR библиотеками.

1. Сравнение моделей

Сравнение обученной модели будет проводится с более сложными нейронными сетями, на основе которых были созданы модули Pytesseract и EasyOCR.

Tesseract – программа, которая была разработана компанией HP c середины 80-х по середину 90-х годов, а в 2006 году её купила ИТ корпорация Google. Новые владельцы усовершенствовали этот модуль распознавания текста и сделали его открытым для сторонних разработчиков. Tesseract имеет поддержку множества языков программирования и может работать со средами через разные оболочки. Он является очень эффективным решение для распознавания символах на изображениях разного формата и объема.

В tesseract есть включенная подсистема нейронной сети, которая работает, как распознаватель текстовых строк. Его истоки идут от LSTM, основанного на Python. Однако, с целью усовершенствования, был переработан на C++ версию Tesseract.

Для распознавания изображения, которое состоит из одного символа, обычно используют свёрточную нейронную сеть (CNN). Когда речь идёт о тексте, то это уже последовательность n-числа символов. Данная проблема решается как раз за счёт LSTM (распространённая форма RNN).

EasyOCR – относительно молодой проект, благодаря которому развивается молодая система по оптическому распознаванию текста. Список языков, который он поддерживает, постоянно растет и уже насчитывает свыше 40 представителей. Модуль разработан силами языка Python и фреймворка PyTorch.

Распознавание текста производится на основе алгоритма машинного обучения CRAFT, который способен обнаружат и выделять текст на различных объектах. Последовательность символов распознаётся благодаря свёрточно-рекуррентной сети CRNN. Для преобразования обработанных данных в текст используется алгоритм CTC BeamSearch.

Качество работы модулей и обученной модели проверялись на нескольких изображениях, которые имели разные шрифты, размер и количество слов.

Сначала рассмотрим, как справится с задачей распознавания обученная модель. Первое изображения (рисунок 9), содержит одно слово из прописных букв.

Изображение выглядит как текст, коллекция картинок

Автоматически созданное описание

Рисунок 9 - Простое слово из прописных букв

После загрузки изображения получается следующий результат:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 - Результат обученной модели на первом изображении

Результат у Tesseract и easy OCR, как ожидалось, тоже оказался идеальным и модули без проблем справились с первым изображением.

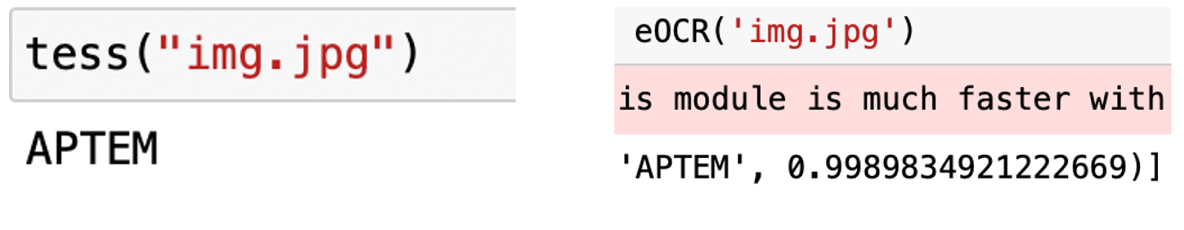


Рисунок 11 - Результат работы Tesseract и easyOCR на первом изображении

На втором изображении задача немного усложниться. Теперь нужно распознать словосочетание из двух слов (рисунок 12), которые написаны в разных регистрах. Буквы, написанные в строчном регистре, должны вызвать путаницу при распознавании, так как имеют схожие очертания с другими строчными буквами.

Рассмотрим, как получится распознать следующую картинку:

Изображение выглядит как текст, коллекция картинок

Автоматически созданное описание

Рисунок 12 - Словосочетание из двух слов в разном регистре

У обученной модели снова получилось распознать написанный текст (рисунок 13)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 - Результат обученной модели на втором изображении

EasyOCR так же без проблем смогла распознать все символы на изображении, однако время, которое было потрачено на получение результата, было ощутимо выше, чем у моей нейронной сети.

А вот у Tesseract возникли проблемы, как и ожидалось, строчная буква «d», вызвала проблемы, из-за чего модуль допустил ошибку (рисунок 14) и не смог успешно выполнить задачу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 - Результат Tesseract на втором изображении

Стоит отметить, что даже easyOCR испытал трудности, ведь при работе с первым изображением уровень точности при распознавании был оценен им в 0.99. А в случае со вторым изображение этот показатель упал до 0.78.

Следующее изображение (рисунок 15), представляет из себя словосочетание из четырех слов, кроме того все слова состоят из прописных символов, что, как демонстрирует ранее рассмотренный пример, создает дополнительные трудности при классификации и как итог, порождает ошибки в итоговом результате.

Так же у следующей текстовой строки присутствуют символы, которые потенциально могут быть спутаны с числами.



Рисунок 15 - Сложное словосочетание

В результате, нейронная сеть, которая обучалась свыше 9 часов получила следующий результат:

Изображение выглядит как текст

Автоматически созданное описание

С данной задаче не получилось справиться так, как хотелось бы. Результат нельзя назвать плохим, ведь текст остается читаемым. Однако колличество допущенных ошибок сильно огорчает. Как можно заметить сеть классифицировала символы правильно, но перепутала регистры. Но сильнее всего бросается в глаза тот факт, что строчное написание буквы «o» было спутано с цифрой «0». Такой результат обусловлен тем фактом, что эти символы имеют идентичные черты при написании. Решение, которое могло предотвратить такую путаницу, может быть связано с прописыванием дополнительного условия в части кода.

В случае модулей работы модулей easyOCR и Tesseract, нет таких проблем. Они без проблем справились с задачей, о чем свидетельствуют следующие рисунки:

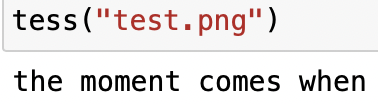


Рисунок 16 - Длинное словосочетание на tesseract

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 - Длинное словосочетание на easyOCR

Во время проведения тестов, работа с одним из изображений дала неожиданный результат. Простое слово из четырех символов (рисунок 18) вызвало проблемы у всех рассматриваемых инструментов.

Изображение выглядит как текст, коллекция картинок

Автоматически созданное описание

Рисунок 18 - Проблемное слово

На первый взгляд, данное слов ничем не выделяется на фоне предшественников, однако при обработке этого изображения были получены такие результаты:

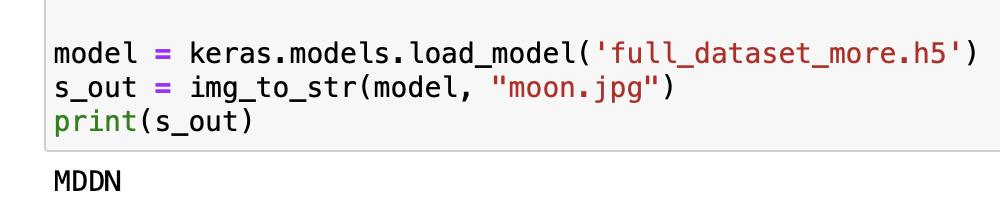


Рисунок 19 - Распознавание последнего изображения нейронной сетью

Разработанная нейронная сеть спутала букву «O» с прописным исполнением «D», хотя ожидалось, что, как и в предыдущем примере, её распознают, как «0». Однако этот результат можно считать допустимым, если сравнивать с тем, что показывают Tesseract (рисунок 20) и easyOCR (рисунок 21).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20 - Пустой вывод у Tesseract

Модуль pytesseract не смог обнаружить символы на изображении и сделал пустой вывод.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21 - Неправильный вывод у easyOCR

У easyOCR на выходе есть символ, но он один и неправильный.

# ЗАКЛЮЧЕНИЕ

В выполненной работе были выполнены все поставленные задачи. Получилось создать свёрточную нейронную сеть. Провести с нуля её обучение. В процессе было создано несколько вариантов моделей и, на основе аналитики, была выбрана наиболее презентабельная версия. Для лучшей обучаемости верным решением выступает использование полного набора данных. Для удовлетворительного результата хватает версии модели, которая обучалась 6 часов, потому что она её показателя точности вполне хватает для рассмотренных задач. Нейронная сеть требует доработки, так как она допускает ошибки в некоторых задачах. Если дать больше времени на обучение, то результат может стать лучше, но временные затраты, которые смогут решить данную проблему, не являются целесообразными в рамках описываемой работы.

По результатам сравнения обученной нейросети и модулей tesseract и easyOCR, можно сказать однозначно, что моя модель показывает результат на порядок ниже, но все равно он удовлетворяет изначальный запрос о том, чтобы была читаемость символов, а выявленные ошибки легко исправляются при помощи дополнительного времени обучения.

Таким образом, машинное обучение в задачах распознавания текста имеет огромный потенциал. Нейронные сети позволяют решать такие задачи и их использование значительно экономит время и ресурсы. Существует множество решений, которые хорошо преуспели в данной области и способны распознавать любые типы текста на множестве языках. Такие решения используются для перевода надписей в реальном времени или для быстрого набора телефонных номеров при наведении камеры телефона на них.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ

ПРИЛОЖЕНИЕ. ИСХОДНЫЙ КОД ПРОГРАММЫ