**Министерство образования и науки Республики Казахстан**

**Специализированная школа «Жас Дарын» города Павлодара**

**НАУЧНЫЙ ПРОЕКТ**

**Тема: «Разработка приложения для распознавания**

**рукописного текста на казахском языке»**

направление:Математическое моделирование экономических

и социальных процессов

секция*:* Информатика

научный руководитель проекта:

преподаватель информатики специализированной школы «Жас Дарын» города Павлодара

Шайхимов А.А.

научный консультант:

PhD, ассоциированный профессор Павлодарского Государственного Университета имени С. Торайгырова

Абыкенова Д.Б.

исполнитель:

ученик 11 «А» класса

Акильбеков Алар

Павлодар 2020 г.

**Оглавление**

|  |  |
| --- | --- |
| Абстракт | 3 |
| Введение | 7 |
| 1. Проблема выбора нейронной сети | 10 |
| 2. Анализ систем оптического распознавания символов (OCR) | 13 |
| 3. Особенности казахского алфавита в распознавании OCR | 15 |
| 4. Разработка модели нейронной сети для распознавания казахского текста | 17 |
| 5. Программная реализация модели нейронной сети | 21 |
| Заключение | 25 |
| Список использованной литературы | 26 |
| Приложение | 28 |

**Абстракт**

В связи с масштабным переходом крупных корпораций и государственных учреждений на электронный документооборот становится актуальной разработка оптических систем распознавания текста на казахском языке, которые строятся на нейронной модели. В данном проекте исследуются нейронные сетидля повышения эффективности распознавания текста.

**Цель:** разработка системы распознавания казахского рукописного текста с использованием искусственных нейронных сетей на языке программирования Java.

**Задачи:**

– проанализировать научную и техническую литературу по теме исследования, изучить документацию необходимых программных продуктов и языков программирования;

– осуществить обзор моделей искусственной нейронной сети и существующих OCR-систем;

– разработать модель обученной нейронной сети;

– разработать приложение для распознавания рукописного текста на казахском языке на основе искусственных нейронных сетей.

**Теоретическая значимость** **исследования:** разработка модели нейронной сети для распознавания казахского текста.

**Новизна исследования:** использование искусственной нейроннойсети для распознавания казахского текста.

**Практическая значимость** **исследования:** разработка приложения по распознаванию рукописного казахского текста, которое может быть использовано организациями, внедряющими и развивающими систему электронного документооборота.

В научном исследовании использованы следующие **методы**: теоретический анализ технической литературы по проблеме исследования; теоретическое моделирование, анализ результатов исследования; тестирование; методы проектирования и верификации программы для системы распознавания рукописного казахского текста.

**Результаты исследования**: построена модель искусственной нейронной сети распознавания казахского текста**,** имея многослойный персептрон. Сеть включает 1024 входных нейрона и 42 выходных нейрона. Модель реализована в программе, написанной на языке программирования Java. Проведено обучение нейронной сети специфическим буквам казахского алфавита: ә, ғ, қ, ң, ө, ұ, ү, h, і. Созданная программа может быть использована в системах электронного документооборота для оцифровки казахского текста.

В процессе исследования были получены консультации и сопровождение PhD, ассистента профессора Школы естественных, социальных и гуманитарных наук Назарбаев Университета – Асылбекова Женисбека, а также управляющего директора КФ «Фонд поддержки исследований и разработок в сфере искусственного интеллекта» – Балтабаевой Асель Бейбитжановны.

**Абстракт**

Ірі корпорациялар мен мемлекеттік органдардың электрондық құжат айналымына кең көлемде көшуіне байланысты нейрондық модельге негізделген қазақ тілінде мәтінді танудың оптикалық жүйелерін дамыту өзекті болып отыр. Мәтінді танудың тиімділігін арттыру үшін бірнеше желілерді қамтитын модель зерттелуде.

Зерттеудің мақсаты - Java бағдарламалау тілінде гибридті нейрондық желіні қолдана отырып, қазақ тілінде жазылған мәтінді тану жүйесін құру.

Мақсатқа жету үшін келесі міндеттер қойылады:

- зерттеу тақырыбы бойынша ғылыми-техникалық әдебиеттерді талдау, қажетті бағдарламалық өнімдер мен бағдарламалау тілдерінің құжаттамасын зерделеу;

- жасанды нейрондық желілер модельдерін және қолданыстағы OCR жүйелерін қарастыру;

- оқытылған нейрондық желінің моделін жасау;

- гибридті жасанды нейрондық желіге негізделген қолжазбаны тану жүйесінің қосымшасын әзірлеу.

Зерттеудің теориялық маңызы қазақ мәтінін танудың жүйке желісінің моделін жасауда.

Зерттеудің жаңалығы қазақша мәтінді тану үшін гибридті нейрондық желіні қолдануда жатыр.

Зерттеудің практикалық маңыздылығы электрондық құжат айналымы жүйесін енгізетін және дамытатын ұйымдар қолдана алатын қолмен жазылған мәтінді тануға арналған қосымшаны жасаудан тұрады.

Ғылыми зерттеу барысында келесі әдістер қолданылды: зерттеу мәселесі бойынша техникалық әдебиеттерді теориялық талдау;

теориялық модельдеу, зерттеу нәтижелерін талдау; тестілеу; мәтінді танудың қазақ тіліндегі жүйесіне арналған бағдарламаны рәсімдеу және тексеру әдістері.

Зерттеу нәтижелері: конволюциялық жүйке желісі мен көп қабатты перцептронды қоса, қазақ мәтінін тану үшін гибридті жүйке желісінің моделі құрылды. Әр желі 1024 кіріс нейронын, 42 шығыс нейронды қамтиды. Модель Java бағдарламалау тілінде жазылған бағдарламада жүзеге асырылады. Қазақ алфавитіндегі ә, ғ, қ, ң, ө, ұ, ү, h, і әріптерінің жүйке жүйесі оқытылды. Құрылған бағдарламаны қазақ мәтінін цифрландыру үшін электрондық құжат айналымы жүйелерінде пайдалануға болады.

Зерттеу барысында PhD докторы, Назарбаев Университетінің жаратылыстану, әлеуметтік және гуманитарлық ғылымдар мектебінің доценті - Асылбеков Жеңісбек, сондай-ақ «Жасанды интеллект саласындағы зерттеулер мен әзірлемелерді қолдау қоры» ҚФ басқарушы директоры - Балтабаева Асель Бейбітжанқызынан консультациялар мен қолдау алынды.

**Введение**

**Актуальность**. Одним из важных направлений цифровизации является оцифровка рукописных текстов на государственном языке, которые представлены как в архивных материалах, так и в рабочих записках, написанных вручную. Описанная проблема является актуальной, в связи с масштабным переходом крупных корпораций и государственных учреждений на электронный документооборот. В связи с большим количеством рукописных документов, разработанных до периода внедрения в работу предприятий, требующих расшифровки и оцифровки, указанный подход будет являться решением поставленной задачи.

Вместе с тем, на мировых рынках все чаще появляются новые серийные мобильные и стационарные устройства, в перечне функционала которых в качестве альтернативы традиционному вводу с клавиатуры используется ввод рукописного текста.

Задачей распознавания рукописных символов научное сообщество начало активно заниматься с 70-х годов ХХ века. Однако имеется ряд теоретических и практических задач, решение которых необходимо для однозначного распознавания рукописных символов, слов и текста в целом. Технология рукописного текста представляет собой многомерную математическую модель, аппроксимирующей сочетание совместно написанных символов и слов. По предоставленной теме изучения есть большое количество научных публикаций [[1]](#footnote-1)[[2]](#footnote-2)[[3]](#footnote-3)[1, 2, 3], в исследованиях которых рассматриваются наиболее популярные языки программирования.

На сегодняшний день одной из важных проблем является разработка программных инструментов, позволяющих распознавать казахский рукописный текст, имеющий специфические символы. Считаю, что исследований на данную тему проведено недостаточно, поэтому в данном проекте рассмотрена задача создания нейронной сети для распознавания казахского текста с высокой точностью.

**Целью** выполняемого исследования является разработка системы распознавания казахского рукописного текста с использованием искусственных нейронных сетей на языке программирования Java.

Для достижения поставленной цели необходимо решить следующие **задачи**:

– проанализировать научную и техническую литературу по теме исследования, изучить документацию необходимых программных продуктов и языков программирования;

– провести обзор моделей искусственной нейронной сети и существующих OCR-систем;

– разработать модель обученной нейронной сети;

– разработать приложение для распознавания рукописного текста на казахском языке на основе искусственных нейронных сетей.

**Объект исследования**: технологии распознавания рукописного текста.

**Предмет исследования**: система распознавания рукописного текста на основе искусственных нейронных сетей для использования в электронном документообороте организации.

**Теоретическая значимость** исследования состоит в разработке модели нейронной сети для распознавания казахского текста.

**Практическая значимость** исследования заключается в разработке приложения по распознаванию рукописного казахского текста, которое может быть использовано организациями, внедряющими и развивающими систему электронного документооборота.

В научном исследовании использованы следующие **методы**: теоретический анализ технической литературы по проблеме исследования; теоретическое моделирование, анализ результатов исследования; тестирование; методы проектирования и верификации программы для системы распознавания рукописного казахского текста.

Использованные методы исследования позволили полностью реализовать поставленные в проекте задачи. Достоверность результатов подтверждается правильностью работы приложения по распознаванию рукописного казахского текста.

**1. Проблема выбора нейронной сети**

Существует большое количество нейронных сетей для оптимального решения задачи распознавания рукописного текста.

Нейронная сеть – это вычислительная структура, состоящая из искусственных нейронов, которые являются абстракцией нервных клеток живых организмов (рис. 1).



Рисунок 1 – Схема простой нейронной сети (зеленым цветом обозначены входные нейроны, голубым – скрытые нейроны, желтым – выходной нейрон)

Существует большое количество нейронных сетей для оптимального решения задачи распознавания рукописного текста.

1. *Однослойный, многослойный персептрон*

Персептрон представляет собой форму нейронной сети, предназначенной для систематизации линейно-разделимых (linearly separable) сигналов, которые произведены из одного нейрона с настраиваемыми синаптическими весами и порогом. Персептрон, построенный на одном нейроне, ограничен выполнением задачи деления лишь только двух классов (гипотез). Мультислойные персептроны благополучно используются для решения всевозможных трудных задач. Наиболее известный метод – метод обратного распространения ошибки (error back-propagation algorithm), который базируется на корректировках ошибок (error-correction learning rule) [4]. [[4]](#footnote-4)

*B) Сеть Кохонена*

Одним из типов конкурентного самопроизвольного обучения считается карта признаков Кохонена или же самоорганизующаяся карта. Это одна из разновидностей нейронных сетей, которая позволяет производить кластеризацию и отличается использованием неконтролируемого обучения. То есть, при обучении обучающее большое количество произведено только из значений входных переменных, в процессе обучения отсутствует сравнивание выходов нейронов с эталонными значениями. Идея сети принадлежит финскому ученому Тойво Кохонену (1982 год). Модель Кохонена относится к классу алгоритмов векторного кодирования (vector-coding algorithm). Данная модель представляет топологическое отражение, которое нормально располагает фиксированную численность векторов во входное место больше высочайшей размерности и, таким образом, упрощает сжатие данных [5]. [[5]](#footnote-5)

*C) Сверточная нейронная сеть*

Сверточные искусственные нейронные сети – это конкретный образ нейронной сети, который сделал революцию в сфере компьютерного зрения и определения образов, считается особенным обликом прямого распространения. Под прямым распространением понимается то, что переменные нейроны данной сети разбиты на группы, которые именуют слоями. И когда эта слоистая нейронная сеть используется к сведению, то активация слоев, то есть смысл данных переменных, подсчитывается поочередно: в начале смысл активации первого слоя, вслед затем два слоя и так до последнего слоя. Активация последнего слоя и работает выходами нейронной сети. В данной нейросети находится большое количество характеристик, в каждом слое группа характеристик определяет как активация надлежащего слоя находится в зависимости от активации предшествующего слоя. Еще, принципиально выделить, что активации изнутри первого слоя имеют все шансы подсчитываться параллельно, то есть в одно и тоже время, автономно друг от друга [6]. [[6]](#footnote-6)

*D) Глубокие нейронные сети*

Применение в нейронных сетях слоев пакетной нормализации (batch normalization), которые дают возможность увеличить качество изучения глубокой нейронной сети. Пакетная нормализация реализуется в облике слоев пакетной нормализации, которые имеют все шансы быть вставлены в важное пространство в нейронной сети, в том количестве несколько раз. Добавочным плюсом применения пакетной нормализации считается уменьшение времени изучения и сокращение количества переобучения.

Способы нормализованной инициализации весов и слои пакетной нормализации могут помочь на практике преодолеть задачу исчезающего градиента и обучать глубочайшие нейронные сети, состоящие из нескольких десятков слоев. Cоздатели сети признают, что неувязка исчезающего градиента в реальное время решена [7]. [[7]](#footnote-7)

Компания Google в 2014 году предложила новую структуру сверточных нейронных сетей Inception [8]. [[8]](#footnote-8) В данной структуре сеть состоит из комплекта блоков, содержащих комбинацию операций свертки и подвыборки различной размерности. Подобный расклад разрешает избежать переобучения и понизить численность характеристик сети, что сокращает время обучения сети.

*E) Гибридные нейронные сети*

Гибридные нейронные сети представляют собой объединение нескольких нейронных сетей для решения определенных задач, что позволяет выполнить декомпозицию сложной задачи на более простые подзадачи, а также оптимизировать архитектуру нейронной сети под конкретную задачу. Также преимущество гибридных нейронных сетей состоит в том, что при программной реализации скорость действия может варьироваться в широких пределах в зависимости от выбранной структуры. Это позволяет применять их для решения задач реального времени.

Гибридные нейронные сети объединяют несколько экспертов, одинаковых либо различных по структуре и природе. Исходя из этого, каждый из них может генерировать разные ответы для одного и того же входного воздействия. Для формирования общего ответа гибридной нейронной системы чаще всего используются взвешенное суммирование результатов или динамический выбор одной нейронной сети, а также полученный результат в качестве выходного значения. Динамический выбор предусматривает поиск в наборе лучшего эксперта [9].[[9]](#footnote-9)

**2. Анализ систем оптического распознавания символов (OCR)**

В настоящее время существуют достаточно успешные коммерческие проекты, направленные на решение задач распознавания рукописного текста. Это так называемые OCR-системы (Optical Character Recognition), предназначенные для автоматического ввода документов в компьютер.

ABBYY FineReader является программой, распознающей текст на изображении оффлайн, и включает в себя версию, позволяющую обрабатывать и рукописный текст. Программа может принимать решения, самообучаться и эволюционировать, используя ранее накопленные знания об объекте. Функционал, позволяющий распознавать рукописный текст развит хуже, так как требуется долгое обучение к конкретному почерку [10].[[10]](#footnote-10)

Технология FormXtra Capture, использующая новейшие алгоритмы распознавания печатного и рукописного текста. В основе технологии лежит интеллектуальное распознавание текста (Intelligent Document Recognition), выполняет распознавание посимвольно, начиная с разделения слов на составляющие компоненты. Поэтому, при ICR распознавании важно, чтобы буквы не были написаны небрежно либо соединены вместе. Технология учитывает динамическую структуру элементов почерка. Почерк, сокращенный до его базовых элементов, по сути, является движениями, произведенными пишущим инструментом. Некоторые символы олицетворяют суть всех рукописных стилей [11]. [[11]](#footnote-11)

Технология Parascript OCR фокусируется на строении написанного слова, добивается распознавания на основе контекстного подхода. Обрабатывая результаты во время процесса распознавания, программа создает высокоточные ответы, которые в свою очередь приводят к более высокому уровню распознавания, чем те, которые проверяются в конце процесса (сайт ресурса www.parascript.com).

PenReader – инструмент, позволяющий автоматически перевести текст, введенный в мобильное устройство вручную. Программа поддерживает четыре режима распознавания рукописного текста: слитное, побуквенное распознавание, интеллектуальное распознавание, позволяющее корректировать результаты распознавания прямо в процессе письма, раздельное распознавание. Программа показывает хорошие результаты распознавания, в том числе русского рукописного текста, однако продолжает совершенствоваться, выпуская новые обновления для более удобного и точного перевода введенного от руки текста [12]. [[12]](#footnote-12)

Существует также большое количество онлайн сервисов (Google Диск, Free OCR. Free Online OCR, Img2txt и тому подобное). Имеющиеся программы распознавания рукописного текста не позволяют обрабатывать рукописный казахский текст, поэтому возникла необходимость в обучении нейронной сети распознаванию казахских слов. Далее рассмотрены особенности казахского алфавита.

**3. Особенности казахского алфавита в распознавании OCR**

Происхождение казахского языка имеет сложную историю, связанную с развитием тюркской народности. Письменность тюркоязычного народа берет свое начало с раннего средневековья. Язык наиболее близкий к настоящему казахскому языку появился примерно в XIII-XIV веках.

В Казахстане арабская графика применялась в течение 900 лет с X по XX век. В 1912 году Ахмет Байтурсынов реформировал казахскую письменность на основе арабской графики, дав возможность пользоваться ею миллионам казахов, живущих за границей. Он исключил все чисто арабские буквы, неиспользуемые в казахском языке, и добавил буквы, специфические для казахского языка. Новый алфавит, получивший название «Жана емле» («Новая орфография»), до сих пор применяется казахами, живущими в Китае, Афганистане, Иране. После 1917 года в Казахстане началось движение за переход на латинский алфавит. Арабское письмо, сыгравшее в свое время положительную роль в общественном развитии, теперь казалось многим тормозом в историческом прогрессе [13]. [[13]](#footnote-13)

С 1923 года в Казахстане начался переход на латинское письмо. Изготовление казахского шрифта потребовало огромного труда и расходов, однако, со временем процесс латинизации был приостановлен.

С 1940 года появляется новый казахский алфавит на основе кириллицы. Большой вклад в развитие казахской письменности внес Ы. Алтынсарин. В настоящее время современный казахский кириллический алфавит, разработанный С.А. Аманжоловым и принятый в 1940 году, содержит 42 буквы, из них 33 буквы русского алфавита и 9 специфических букв казахского языка: Ә, Ғ, Қ, Ң, Ө, Ұ, Ү, Һ, І. Буквы «в, ё, ф, ц, щ, ч, ъ, ь, э» используются только в заимствованных словах. Каждая специфическая буква расположена рядом с той буквой, которая схожа по звучанию. Каждая буква алфавита имеет свое название и звучание. На рисунке 1 представлены буквы казахского алфавита по порядку следования.

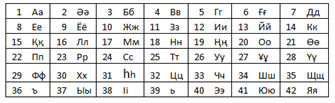


Рисунок 1 - Казахский алфавит

Специфические буквы Ғ, Қ, Ң, Ө, Ұ, Ү схожи с родственными буквами по начертанию, но имеют добавления в виде «черточек» и «хвостиков». Буквы Ә, Һ, І являются уникальными и не схожи ни с одной буквой кириллического алфавита.

Для решения задачи распознавания рукописного текста на казахском языке необходимо пополнить базу данных кириллических букв специфическими символами. Также важно разработать модель нейронной сети, которая будет с наименьшими ошибками производить распознавание текста.

**4. Разработка модели нейронной сети для распознавания казахского текста**

Для решения задачи распознавания рукописного текста мною использован многослойный персептрон. Сеть заранее обучается распознаванию каждой буквы казахского алфавита.

Имеется конечный набор обучающих пар {(Xi,Yi)}, задающих отображение F множества X в множество Y F:(X→Y), где Х=[X1,X2,X3..Xk] – матрица входных сигналов, Y – выход нейронной сети. Процедура обучения заключается в минимизации функционала ошибки. Обучается нейронная сеть с учителем, то есть на большом количестве обучающих примеров.

Рассмотрим процесс распознавания, осуществляемый нейронной сетью. Сеть обрабатывает поочередно каждый символ представленного текста. Символ (картинка размером 32\*32) поступает на вход двух сетей, после обработки которого выводится массив чисел в диапазоне [0; 1],которые представляют собой вероятность эквивалентности распознаваемого символа каждой букве казахского алфавита. Номер элемента соответствует букве, в соответствии с порядком следования букв в алфавите. Элемент с наибольшим значением выводится как окончательный результат распознанного символа.

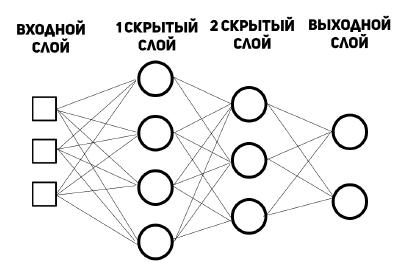
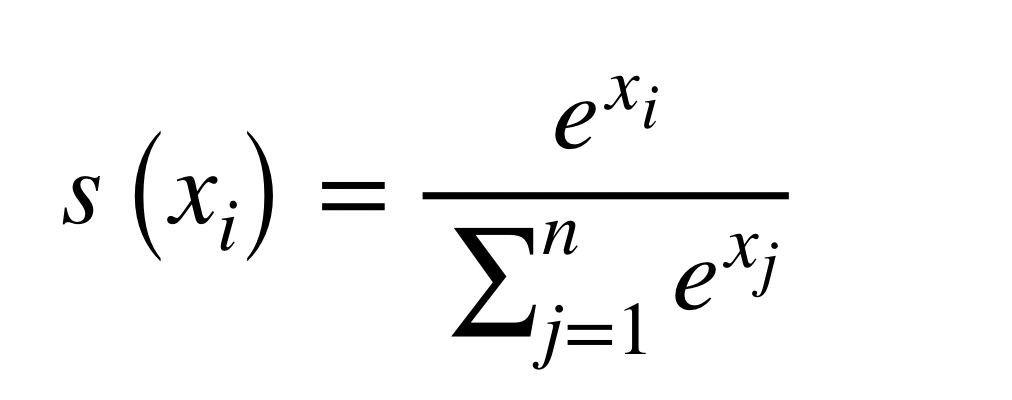


Рисунок 2 - Многослойный персептрон

Персептрон состоит из четырех слоев (рисунок 2). Состав сети: входной слой (1024 нейрона), первый скрытый слой (512 нейронов), второй скрытый слой (256 нейронов), выходной слой, состоящий из 42-х нейронов, которые представляют собой значения вероятностей принадлежности распознаваемого символа каждой букве, а также нейроны смещения. Ниже представлена структура нейронов скрытых слоев в соответствии с рисунком 3. В их состав входит суммирующий блок и функция активации. На вход суммирующего блока попадают произведения входных сигналов на веса, прибавленные к значению нейрона смещения. У нейронов скрытых слоев используется логистическая функция активации, вычисляемая по формуле: 1/1+e^-x, где x – выход суммирующего блока. На выходном слое используется функция softmax – обобщение логистической функции для многомерного случая, функция преобразует вектор x размерности n в вектор s той же размерности, где каждая координата s(xi) полученного вектора представлена вещественным числом в интервале [0,1] и сумма координат равна 1.

Координаты s(xi) вычисляются следующим образом:



Для обучения нейронной сети в качестве функции потерь используется перекрёстная энтропия для многоклассовой классификации.

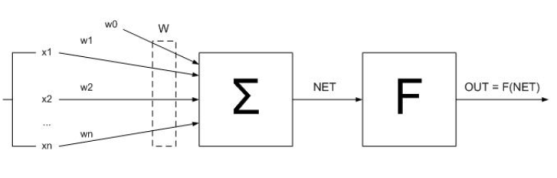


Рисунок 3 - Структура нейрона

*Обучение сети.* Для распознавания казахского текста требуется внести девять специфических символов казахского алфавита в обучающий набор базы данных EMNIST. Обучение сети осуществляется методом обратного распространения ошибки. Основная идея этого метода состоит в распространении сигналов ошибки от выходов сети к её входам, в направлении, обратному прямому распространению сигналов в обычном режиме работы, корректируя веса таким образом, чтобы ошибка на выходе нейронной сети уменьшалась. При обучении сети вычисляется вектор ошибки как разница между полученными и известными (опытными) данными. Вектор ошибки используется для модифицирования весовых коэффициентов выходного слоя с тем, чтобы при повторной подаче того же набора входных данных вектор ошибки уменьшался.

*Выборка для обучения нейронной сети.* Для проведения испытаний, важным фактором является выбор базы исходных изображений. К такой базе должны быть предъявлены особые требования: а) база должна содержать существенное количество изображений (чем больше, тем лучше), должна содержать изображения с объектами нескольких классов, изображения должны быть сопоставлены с классами объектов на них изображенных, изображения должны быть равномерно распределены по классам, для исследовательских задач важно, чтобы объекты были достаточно просты в распознавании. Собирать самостоятельно подобную базу нецелесообразно, так как это отнимет огромное количество времени, которое можно использовать с большей пользой, к тому же такие базы уже существуют и предоставляются всем желающим безвозмездно. Большинство из них собраны ведущими мировыми университетами и исследовательскими лабораториями как раз для исследовательских целей, для проведения испытаний алгоритмов и методов по распознаванию образов. Наиболее известными такими коллекциями являются: PASCAL, CIFAR-10, CIFAR-100, TRECVID, EMNIST, STL-10, SVHN и др. Некоторые из них имеют четкое предназначение, например, для распознавания текста, номеров домов и автомобилей, предметов или действия, происходящего на изображении.

Для проведения практических испытаний использован набор данных EMNIST, популярный в сообществе исследователей глубокого обучения (the EMNIST Dataset on https://www.nist.gov). Этот набор содержит 60000 обучающих изображений и 10000 контрольных изображений, собранных Национальным институтом стандартов и технологий США. Также добавлены дополнительные обучающие наборы с примерами написания каждого специфического казахского символа в количестве 500 изображений (ә, ғ, қ, ң, ө, ұ, ү, h, і). Далее эти изображения отредактированы с помощью библиотеки OpenCV для нахождения битовых масок к каждому изображению из обучающей выборки. На основе битовых масок сформирована выборка для обучения нейронной сети. Пример битовых масок букв показан на рисунке 4.



Рисунок 4 - Пример подготовленных битовых масок букв

**5. Программная реализация модели нейронной сети**

Для написания программы была использована интегрированная среда разработки программного обеспечения IntelliJ IDEA, разработанная компанией JetBrains (рис. 5).

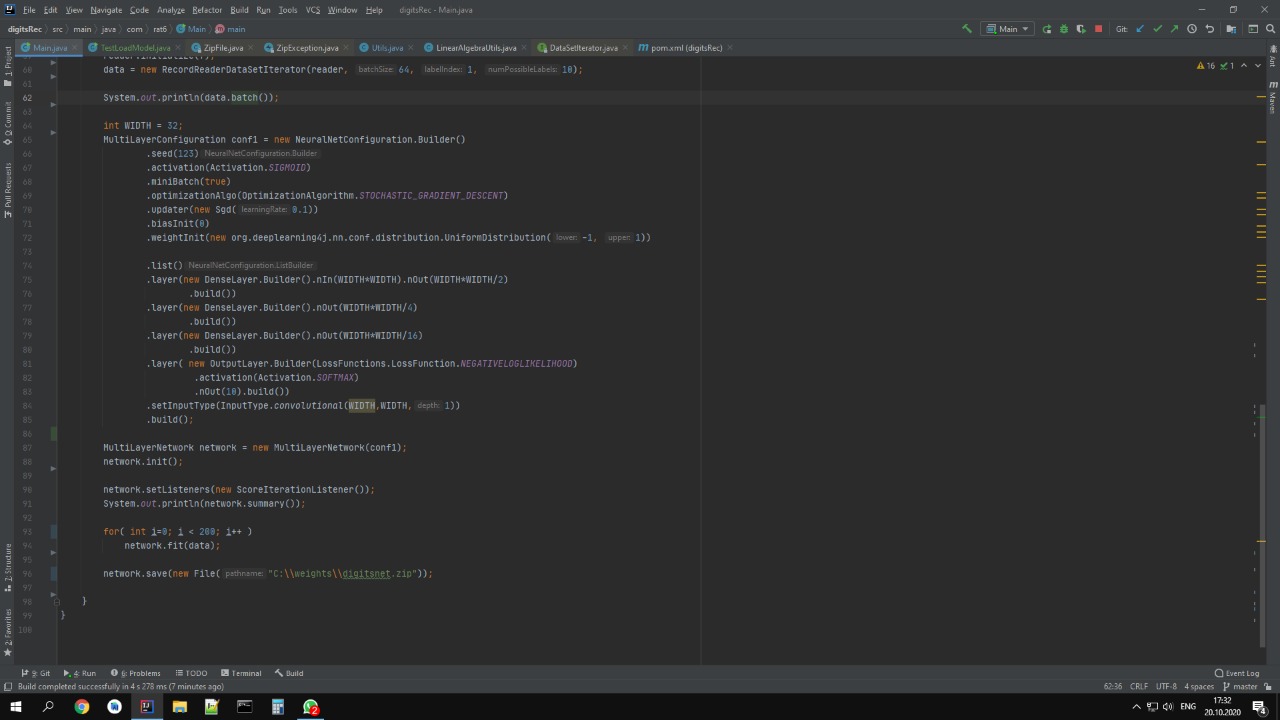


Рисунок 5 - Интегрированная среда разработки программного обеспечения IntelliJ IDEA с кодом

Для данного проекта по машинному обучению использована библиотека Deeplearning4j – библиотека программ на языке Java, используемая как фреймворк для глубокого обучения, которая включает реализацию ограниченной машины Больцмана, глубокую сеть доверия, глубокого автокодировщика, стекового автокодировщика с фильтрацией шума, рекурсивную тензорную нейронную сеть, word2vec, doc2vec, and GloVe. Эти алгоритмы включены также в версии библиотеки, поддерживающие распределённые вычисления, интегрированные с архитектурами Apache Hadoop и Spark [15].

В библиотеке Deeplearning4j реализованы методы для загрузки популярных наборов данных для алгоритмов машинного обучения. Для обучения сети используется метод fit. Этому методу необходимо указать данные (переменная data\_train), содержащие изображения с рукописными буквами, на которых будет обучаться сеть. Затем необходимо указать правильные ответы для каждого изображения для обучения нейронной сети. После многочисленных обучающих эпох проводится тестирование нейронной сети на тестовых изображениях (переменная data\_test). Для оптимизации использован метод стохастического градиентного спуска, размер мини-выборки составляет 64 элемента, то есть анализируются 64 изображения, определяются направления градиента и в соответствии с этим направлением выполняются изменения весов для снижения ошибки на выходе искусственной нейронной сети, после чего анализируются следующие 64 изображения и так до окончания всех изображений из обучающей выборки. Количество эпох при обучении указывает сколько раз было пройдено по всей обучающей выборке. Метод stats позволяет печатать диагностическую информацию в процессе обучения.

Программа состоит из трех модулей: RedactData, TrainData, WordRecognitionActivity (листинг представлен в Приложении).

*Результаты и обсуждение*

После обучения нейронной сети можно оценить качество обучения нейронной сети при помощи тестовых данных. Сначала мною проведено распознавание текста после первой эпохи обучения: 42\*500 образцов изображений с буквами в рукописном варианте. Получены значения выходов нейронной сети при распознавании образа букв. Результаты показали, что после первого обучения точность распознавания тестовой выборки составила 0,3. После этого мною проведено еще 400 эпох обучения. Значения выходов нейронной сети при распознавании образа буквы «қ» после обучения (500 образцов написания буквы) показали более точный результат (0,96).

Пример распознавания специфической буквы в созданной программе показан на рисунке 7.

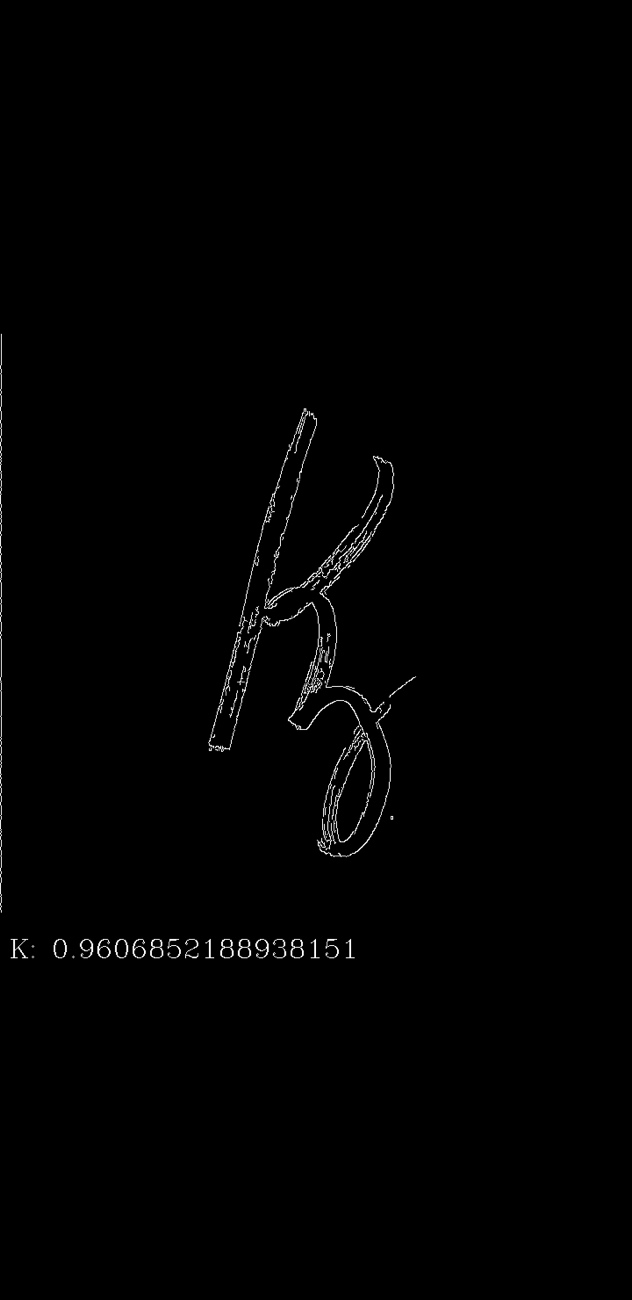


Рисунок 7 - Распознавание специфической буквы в программе

На рисунке 8 представлен процесс распознавания слов с использованием разработанной программы. На рисунках 9, 10 представлен процесс распознавания предложений с использованием разработанной программы. Листинг программы представлен в Приложении.

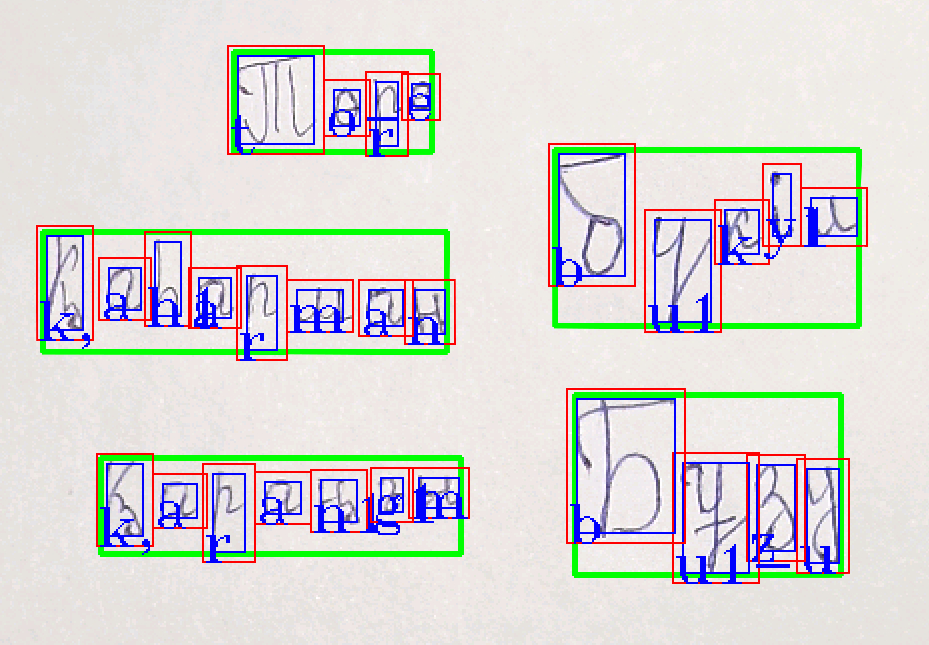


Рисунок 8 - Фрагмент распознавания слова.

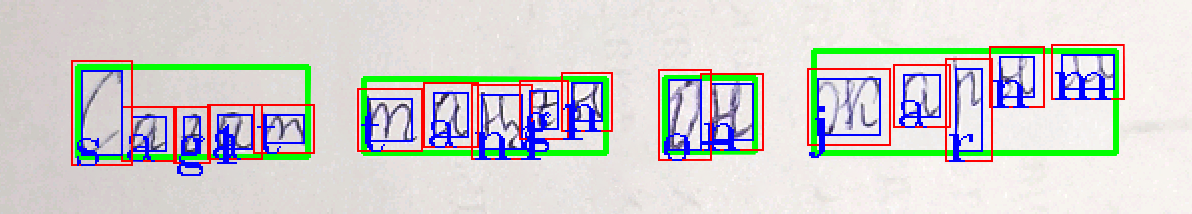


Рисунок 9 - Фрагмент распознавания предложений. Предложение, выданное приложением: “сағат таңғн он жарнм”

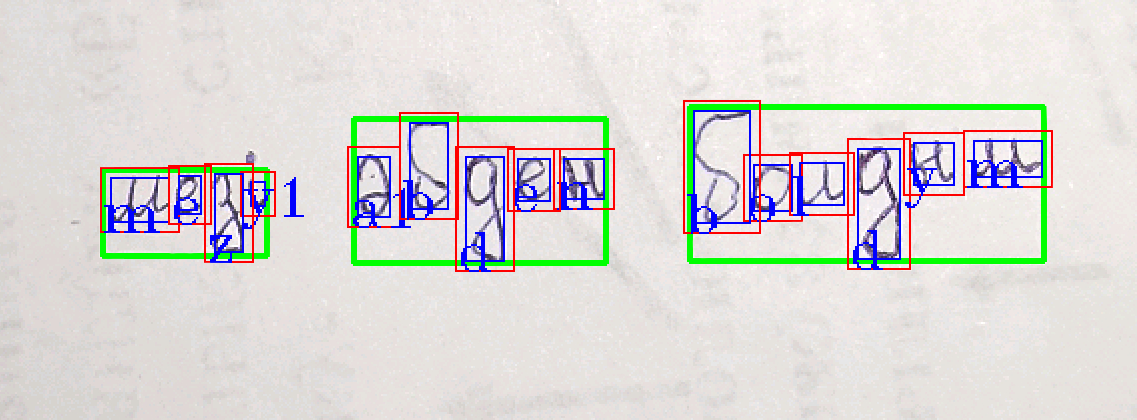


Рисунок 10 – Фрагмент распознавания предложений. Предложение, выданное приложением: “мезі әбден болдым”

В связи с отсутствием кодировки Utf8\_general\_ci кириллица обозначается латинскими буквами Ә(a1), Ғ(g1), Қ(k,), Ң(n1), Ө(o-), Ұ(u1-), Ү(u1), Һ(h1), І(y1).

Вместе с тем, в настоящее время по поручению Президента Казахстана Касым-Жомарта Токаева совершенствуется утвержденный алфавит казахского языка для перехода на латиницу, поэтому считаю, что данная модель будет востребована в будущем.

Таким образом, полученные в данном исследовании результаты распознавания рукописного казахского текста подтверждают высокую эффективность применения многослойных нейронных сетей. На базе разработанной нейросетевой модели можно строить адаптивные интеллектуальные системы распознавания казахского текста, включая технологии принятия решения для определения семантики текста.

**Заключение**

На сегодняшний день разработка оптических систем распознавания казахского текста является актуальной проблемой для развития систем электронного документооборота в казахстанских компаниях.

Целью данной работы является разработка системы распознавания казахского рукописного текста с использованием нейронных сетей на языке программирования Java.

В работе проанализированы публикации по теме исследования, представлен анализ OCR, нейронных сетей. Определена необходимость разработки системы распознавания рукописного казахского текста.

Была разработана и обучена модель: многослойный персептрон. Сеть включает 1024 входных нейрона и 42 выходных нейрона.

Результат достигается за счет частичного исключения случаев коллизий при распознавании. Большую роль в эффективности распознавания играет количество обучающих примеров. Архитектура построенной сети позволяет решить данную задачу только при большом количестве обучающих примеров почерков различных людей.

Приложение разработано на языке программирования Java, что позволяет ее использовать как в мобильных устройствах, так и на компьютерах. Разработанное приложение имеет особенность в сборе данных: все нарисованные изображения сохраняются и будут использованы для улучшения качества модели.

Разработанная модель нейронной сети и программное приложение позволяет использовать их в реальном процессе распознавания рукописного казахского текста, может быть встроено в систему электронного документооборота и в программные комплексы, предназначенные для оцифровки архивной и другой оперативной рукописной информации на казахском языке.

**Список использованной литературы**

1. Yadav, P., & Yadav, N. (2015). Handwriting recognition system-a review. International Journal of Computer Applications, 114(19), 36-40.
2. Eskenazi, S., Gomez-Krämer, P., & Ogier, J. M. (2017). A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognition, 64, 1-14.
3. Keysers, D., Deselaers, T., Rowley, H. A., Wang, L. L., & Carbune, V. (2016). Multi-language online handwriting recognition. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1180-1194.
4. Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B., & Cohen, S. (2018). Start, follow, read: End-to-end full-page handwriting recognition. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 367-383).
5. Hu, Z., Bodyanskiy, Y. V., Tyshchenko, O. K., & Boiko, O. O. (2018). A neuro-fuzzy Kohonen network for data stream possibilistic clustering and its online self-learning procedure. Applied soft computing, 68, 710-718.
6. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern Recognition, 77, 354-377.
7. He K., Zhang X., Ren S., et al. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV, USA, 27–30 June 2016), 2016. pp. 770–778 DOI: 10.1109/CVPR.2016.90.
8. Szegedy C., Liu W, Jia Y. et al. Going Deeper with Convolutions. IEEE Conference on Computer Vision and Pattern Recognition (Boston, MA, USA, June 7–12, 2015), 2015. pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.
9. Psichogios, D.C. and Ungar, L.H. (1992), A hybrid neural network‐first principles approach to process modeling. AIChE J., 38: 1499-1511. doi:10.1002/aic.690381003
10. ABBYY FineReader: an outward glance / / 3DNews. [Electronic resource] URL: https://3dnews.ru/632560
11. Intelligent Document Recognition. [Electronic resource] URL: http://idr.in.ua/article/12\_2013/12.html
12. PenReader // Apsoid. [Electronic resource] URL: http://www.apsoid.ru/soft-an/3764
13. Байтелесова Ж. Т. Ахмет Байтурсынов: аспект развития казахского языка. Бiлiм. Образование. – 2007. – №4. – С. 112-114
14. Chen T., Li M., Li Y. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. Available at: <https://arxiv.org/abs/1512.01274>
15. Deeplearning4j https://deeplearning4j.org/

Приложение

**Листинг программы**

**Код модуля RedactData.java**

public class RedactDataTest {

public static void main(String[] args) {

int WIDTH = 32;

Size size = new Size(WIDTH, WIDTH);

String pathFrom = "C:\\Users\\Pupochek\\Documents\\kazLet";

String pathTo = "C:\\Users\\Pupochek\\Documents\\kazLetData";

redactImages(pathFrom, pathTo, size);

}

private static void redactImages(String pathFrom, String pathTo, Size size) {

File mainDir = new File(pathFrom);

File[] mainFiles = mainDir.listFiles();

for(int i=0; i<mainFiles.length; i++){

System.out.print(mainFiles[i].getName()+" ");

File newFile = new File(pathTo+"\\"+mainFiles[i].getName());

if(!newFile.exists()) newFile.mkdirs();

}

for(int ret=0; ret<mainFiles.length; ret++) {

File dir = new File(mainFiles[ret].getAbsolutePath());

File[] arrFiles = dir.listFiles();

if (arrFiles != null) {

for (int i = 0; i < arrFiles.length; i++) {

Mat imgOriginal = Imgcodecs.imread(arrFiles[i].getAbsolutePath()) ;

Mat imgRedact = new Mat();

Imgproc.cvtColor(imgOriginal, imgRedact, Imgproc.COLOR\_BGR2GRAY);

//Redact

Imgproc.resize(imgRedact, imgRedact, size);

Imgproc.Canny(imgRedact, imgRedact, 80, 200);

Imgcodecs.imwrite(pathTo + "\\" + mainFiles[ret].getName() + "\\" + i + ".png", imgRedact);

imgOriginal.release(); imgRedact.release();

}

} else

System.out.println("error! folder was not found");

}

}

}

**Код модуля TrainData.java**

public class TrainData {

public static void main(String[] args) throws IOException{

int WIDTH = 32;

int labelsSum = 42;

int batchSize = 64;

String[] allowedExtensions = BaseImageLoader.ALLOWED\_FORMATS;

Random rand = new Random();

File root = new File("C:\\Users\\User\\Documents\\kazLetRedacted");

FileSplit f = new FileSplit(root, allowedExtensions, rand);

ParentPathLabelGenerator labelMaker = new ParentPathLabelGenerator();

ImageRecordReader reader;

DataSetIterator data\_train;

reader = new ImageRecordReader(WIDTH, WIDTH, 1, labelMaker);

reader.initialize(f);

data\_train = new RecordReaderDataSetIterator(reader, batchSize, 1, labelsSum);

MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()

.seed(321)

.activation(Activation.SIGMOID)

.miniBatch(true)

.optimizationAlgo(OptimizationAlgorithm.

STOCHASTIC\_GRADIENT\_DESCENT)

.updater(new Sgd(0.05))

.biasInit(0)

.weightInit(new org.deeplearning4j.nn.conf.distribution.UniformDistribution(-1, 1))

.list()

.layer(new DenseLayer.Builder().nIn(WIDTH\*WIDTH).nOut(WIDTH\*WIDTH/2)

.build())

.layer(new DenseLayer.Builder().nOut(WIDTH\*WIDTH/2)

.build())

.layer(new DenseLayer.Builder().nOut(WIDTH\*WIDTH/4)

.build())

.layer(new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)

.activation(Activation.SOFTMAX)

.nOut(labelsSum).build())

.setInputType(InputType.convolutional(WIDTH,WIDTH,1))

.build();

MultiLayerNetwork network = new MultiLayerNetwork(conf);

network.init();

network.setListeners(new ScoreIterationListener(10));

System.out.println(network.summary());

// here the actual learning takes place

for( int i=0; i < 500; i++ )

network.fit(data\_train);

root = new File("C:\\Users\\User\\Documents\\kazLetRedacted");

f = new FileSplit(root, allowedExtensions, rand);

labelMaker = new ParentPathLabelGenerator();

reader = new ImageRecordReader(WIDTH, WIDTH, 1, labelMaker);

reader.initialize(f);

DataSetIterator data\_test = new RecordReaderDataSetIterator(reader, batchSize, 1, labelsSum);;

System.out.println("Evaluate model....");

Evaluation eval = new Evaluation(labelsSum);

while(data\_test.hasNext()){

DataSet t = data\_test.next();

INDArray features = t.getFeatures();

INDArray lables = t.getLabels();

INDArray predicted = network.output(features,false);

eval.eval(lables, predicted);

}

System.out.println(eval.stats());

network.save(new File("C:\\weights\\kazletters\_model.zip"));

}

}

**Код модуля WordRecognitionActivity.java**

public class WordRecognitionActivity extends Activity implements CvCameraPreview.CvCameraViewListener {

public final int CAMERA\_WIDTH = 720;

public final int CAMERA\_HEIGHT = 1280;

public final int WIDTH = 32;

private CvCameraPreview cameraView;

private MultiLayerNetwork network;

NativeImageLoader imageLoader;

String[] labels;

Size inputSize;

INDArray array;

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

cameraView = new CvCameraPreview(this, CvCameraPreview.CAMERA\_BACK, 1);

cameraView.setCvCameraViewListener(this);

cameraView.setCameraSizes(CAMERA\_WIDTH, CAMERA\_HEIGHT);

inputSize = new Size(WIDTH, WIDTH);

imageLoader = new NativeImageLoader(WIDTH, WIDTH, 1);

network = StorageHelper.loadMultiLayerNetwork("kazlettersnet50.zip", this);

labels = StorageHelper.loadLabels(network);

setContentView(cameraView);

}

@Override

public Mat onCameraFrame(Mat imgCam) { //RGBA

Mat imgCamGray = new Mat();

cvtColor(imgCam, imgCamGray, COLOR\_RGBA2GRAY);

List<Rect> letterBBoxes = getDetectedWords(imgCamGray);

List<String> words = updateLetters(letterBBoxes, imgCamGray,imgCam);

imgCamGray.release();

String sent = join(“ ”, words);

Log.d(“Sentence”, “”+sent );

return imgCam;

}

private List<String> updateLetters(List<Rect> letterBBoxes, Mat imgCamGray, Mat imgCam){

List<String> words = new ArrayList<String>();

for(Rect r: letterBBoxes) {

if(r.height()<32 || r.width()<32 || r.width()>CAMERA\_WIDTH/2 || r.height()>CAMERA\_HEIGHT/5 ) continue;

int expansion = 1;

int xRight = r.x()+r.width()-1+expansion>CAMERA\_WIDTH? r.x()+r.width()-1: r.x()+r.width()-1+expansion;

int xLeft = r.x()<expansion? r.x(): r.x()-expansion;

int yTop = r.y()<expansion? r.y(): r.y()-expansion;

int yBottom = r.y()+r.height()-1+expansion>CAMERA\_HEIGHT? r.y()+r.height()-1: r.y()+r.height()-1+expansion;

Point lt = new Point(xLeft, yTop);

Point br = new Point(xRight, yBottom);

Rect rect = new Rect(lt, br);

Mat imgGraySubmat = new Mat(imgCamGray, rect).clone();

Mat edges = new Mat();

Canny(imgGraySubmat, edges, 100, 200); imgGraySubmat.release();

MatVector contours = new MatVector();

Mat nullMat = new Mat();

findContours(edges, contours, nullMat, RETR\_EXTERNAL, CHAIN\_APPROX\_NONE);

edges.release();nullMat.release();

if(contours.size()<1) continue;

List<Rect> letters = getDetectedLetters(contours, r);

contours.close();

rectangle(imgCam, lt, br, Utils.GREEN, 2, BORDER\_DEFAULT, 0);

List<String> word = new ArrayList<String>();

for(int i=0; i<letters.size(); i++){

Rect letterR = letters.get(i);

if(letterR.area()==0) continue;

expansion = 5;

int xRight1 = r.x()+letterR.x()+letterR.width()-1>CAMERA\_WIDTH-expansion? (r.x()+letterR.x()+letterR.width()-1): (r.x()+letterR.x()+letterR.width()-1 + expansion);

int xLeft1 = r.x()+letterR.x()<expansion? r.x()+letterR.x(): r.x()+letterR.x()-expansion;

int yTop1 = r.y()+letterR.y()<expansion? r.y()+letterR.y(): r.y()+letterR.y()-expansion;

int yBottom1 =r.y()+letterR.y()+letterR.height()-1>CAMERA\_HEIGHT-expansion? r.y()+letterR.y()+letterR.height()-1: r.y()+letterR.y()+letterR.height()-1+expansion;

Point letter\_lt1 = new Point(xLeft1, yTop1);

Point letter\_br1 = new Point(xRight1, yBottom1);

Rect letter\_rect1 = new Rect(letter\_lt1, letter\_br1);

rectangle(imgCam, new Point(r.x()+letterR.x(), r.y()+letterR.y()), new Point(r.x()+letterR.x()+letterR.width()-1,r.y()+letterR.y()+letterR.height()-1), Utils.BLUE, 1, BORDER\_DEFAULT, 0);

rectangle(imgCam, letter\_lt1, letter\_br1, Utils.RED, 1, BORDER\_DEFAULT, 0);

Mat letterArea = new Mat(imgCamGray, letter\_rect1).clone();

resize(letterArea, letterArea, inputSize);

Canny(letterArea, letterArea, 100, 200);

letterArea.convertTo(letterArea, CV\_64FC1, 1d/255d, 0);

array = imageLoader.asMatrix(letterArea);

letterArea.release();

INDArray predicted = network.output(array, true);

int maxId = Nd4j.getBlasWrapper().iamax(predicted);

putText(imgCam, labels[maxId], new Point(letter\_rect1.x(), letter\_rect1.y()+letter\_rect1.height()), FONT\_HERSHEY\_TRIPLEX, 0.9, Utils.BLUE);

word.add(labels[maxId]);

}

words.add(join("", word));

}

return words;

}

private List<Rect> getDetectedWords(Mat imgOriginal){

List<Rect> boundRect = new ArrayList<>();

Mat img = imgOriginal.clone();

if(img.type()!=0)

cvtColor(img, img, COLOR\_RGB2GRAY);

Sobel(img, img, CV\_8U, 1, 0, 3, 1, 0, BORDER\_DEFAULT);

threshold(img, img, 100, 200, 8);

Mat element = getStructuringElement(MORPH\_RECT, new Size(15,5));

morphologyEx(img, img, MORPH\_CLOSE, element);

element.release();

MatVector contours = new MatVector();

Mat hierarchy = new Mat();

findContours(img, contours, hierarchy, 0, 1);

hierarchy.release();

Mat mMOP2f1 = new Mat();

Mat mMOP2f2 = new Mat();

for( int i = 0; i < contours.size(); i++ ){

contours.get(i).convertTo(mMOP2f1, CV\_32FC2);

approxPolyDP(mMOP2f1, mMOP2f2, 2, true);

mMOP2f2.convertTo(contours.get(i), CV\_32S);

Rect appRect = boundingRect(contours.get(i));

if (appRect.width()>appRect.height()) boundRect.add(appRect);

}

contours.close();

mMOP2f1.release();mMOP2f2.release();img.release();

sortRectList(boundRect);

return boundRect;

}

private List<Rect> getDetectedLetters(MatVector contours, Rect r){

List<Rect> rs = new ArrayList<Rect>();

double minWidth = r.width()/33;

for (int i = 0, j = 0; i < contours.size(); i++) {

Rect r1 = boundingRect(contours.get(i));

if(r1.width() < minWidth) continue;

else {

rs.add(j, r1);

j++;

}

}

List<Rect> rs1 = new ArrayList<Rect>();

for(int i=0; i<rs.size() ; i++){

Rect r1 = rs.remove(i);

for(int j=0; j<rs.size(); j++){

Rect r2 = rs.get(j);

if( r1.x() < r2.x()+r2.width() && r2.x() < r1.x()+r1.width() ) {

int xLeft = Math.min(r1.x(), r2.x());

int xRight = Math.max(r1.x()+r1.width()-1, r2.x()+r2.width()-1);

int yTop = Math.min(r1.y(), r2.y());

int yBottom = Math.max(r1.y()+r1.height()-1, r2.y()+r2.height()-1);

r1 = new Rect(xLeft, yTop, xRight-xLeft, yBottom-yTop);

rs.remove(j);

j--;

}

}

rs1.add(r1);

}

sortRectList(rs1);//слева-направо

return rs1; }

Comparator<Rect> compRect = new Comparator<Rect>() {

@Override

public int compare(Rect o1, Rect o2) {

return (o1.x()<o2.x()? -1: (o1.x()==o2.x()? 0: 1));

}

};

private List<Rect> sortRectList(List<Rect> rects){

Collections.sort(rects, compRect);

return rects;

}

private String join(String separator, List<String> input) {

if (input == null || input.size() <= 0) return "";

StringBuilder sb = new StringBuilder();

for (int i = 0; i < input.size(); i++) {

sb.append(input.get(i));

//if not the last item

if (i != input.size() - 1) sb.append(separator);

}

String word = sb.toString();

return word;

}

}

1. [1] Yadav, P., & Yadav, N. (2015). Handwriting recognition system-a review. International Journal of Computer Applications, 114(19), 36-40. [↑](#footnote-ref-1)
2. [2] Eskenazi, S., Gomez-Krämer, P., & Ogier, J. M. (2017). A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognition, 64, 1-14. [↑](#footnote-ref-2)
3. [3] Keysers, D., Deselaers, T., Rowley, H. A., Wang, L. L., & Carbune, V. (2016). Multi-language online handwriting recognition. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1180-1194 [↑](#footnote-ref-3)
4. [4] Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B., & Cohen, S. (2018). Start, follow, read: End-to-end full-page handwriting recognition. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 367-383). [↑](#footnote-ref-4)
5. [5] Hu, Z., Bodyanskiy, Y. V., Tyshchenko, O. K., & Boiko, O. O. (2018). A neuro-fuzzy Kohonen network for data stream possibilistic clustering and its online self-learning procedure. Applied soft computing, 68, 710-718. [↑](#footnote-ref-5)
6. [6] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern Recognition, 77, 354-377. [↑](#footnote-ref-6)
7. [7] He K., Zhang X., Ren S., et al. Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas, NV, USA, 27–30 June 2016), 2016. pp. 770–778. DOI: 10.1109/CVPR.2016.90. [↑](#footnote-ref-7)
8. [8] Szegedy C., Liu W, Jia Y. et al. Going Deeper with Convolutions. IEEE Conference on Computer Vision and Pattern Recognition (Boston, MA, USA, June 7–12, 2015), 2015. pp. 1–9. DOI: 10.1109/CVPR.2015.7298594. [↑](#footnote-ref-8)
9. [9] Psichogios, D.C. and Ungar, L.H. (1992), A hybrid neural network‐first principles approach to process modeling. AIChE J., 38: 1499-1511. doi:10.1002/aic.690381003 [↑](#footnote-ref-9)
10. [10] ABBYY FineReader: взгляд изнутри // 3DNews. [Электронный ресурс] URL: https://3dnews.ru/632560 [↑](#footnote-ref-10)
11. [11] Intelligent Document Recognition. [Электронный ресурс] URL: http://idr.in.ua/article/12\_2013/12.html [↑](#footnote-ref-11)
12. [12] PenReader // Apsoid. [Электронный ресурс] URL: http://www.apsoid.ru/soft-an/3764 [↑](#footnote-ref-12)
13. [13] Байтелесова Ж. Т. Ахмет Байтурсынов: аспект развития казахского языка. Бiлiм. Образование. – 2007. – №4. – С. 112-114 [↑](#footnote-ref-13)