МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический
университет имени К.И. Сатпаева

Институт автоматики и информационных технологий

УДК _____ На правах рукописи

Бейсенканов Ербол Бакытжанович

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На соискание академической степени магистра

Название диссертации	Применение глубокого обучения в задаче распознавания рукописного оффлайн текста
Направление подготовки	7M06102 – Machine Learning & Data Science

Научный руководитель
канд. физ-мат. наук, доцент
_____ Д.Б. Нурсеитов
«1» мая 2022 г.

Оппонент
канд. физ-мат. наук, ст. преподаватель
_____ И.М. Уалиева
«02» 06 2022 г.

Нормоконтроль
PhD, ассиестент-профессор
_____ А.Т. Ахмедиярова
«15» мая 2022 г.

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. физ-мат. наук, проффессор
_____ А.Н. Молдагулова
«02» 06 2022 г.

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Казахский национальный исследовательский технический
университет имени К.И. Сатпаева

Институт автоматики и информационных технологий

Кафедра Программная инженерия
Специальность: 7M06102 – Machine Learning & Data Science

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. физ-мат. наук, проффессор
_____ А.Н. Молдагулова
«02» 06 2022 г.

ЗАДАНИЕ
на выполнение магистерской диссертации

магистранту, обучающемуся Бейсенканову Ерболу Бакытжановичу

Тема диссертации: «Применение глубокого обучения в задаче распознавания рукописного оффлайн текста»

Срок сдачи законченной диссертации «02» 06 20222.

Исходные данные к магистерской диссертации. Дан анализ моделей нейронных сетей для распознавания рукописного оффлайн текста. Поставленные цели и задачи диссертационной работы обуславливают анализ основных подходов к решению задачи распознавания рукописного текста; изучение датасетов рукописного текста; создание и обучение программы на основе глубокого обучения и нейронных сетей для решения задачи распознавания рукописного текста.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации: а) ознакомление с глубоким обучением; б) исследование проблем при распознавании рукописного текста; в) исследование подобных нейронных сетей на других языках; г) исследование датасетов; д) сегментация изображений; е) построение модели нейронной сети с использованием информации, полученной в ходе изучения вопроса; ж) обучение модели и выводы по проделанной работе.

Рекомендуемая основная литература 1. Львович И.Я., Львович Я.Е., Мозговой А.А., Чопоров О.Н. Снижение количества ошибок распознавания сканированных рукописных текстов // Цифровая обработка сигналов – 2016. - №4. – С. 43-47. 2. R. Manmatha, Nitin Srimal, “Scale Space Technique for Word Segmentation in Handwritten Documents”, 1999. 3. D. Nurseitov, K. Bostanbekov, D. Kurmankhojayev, A. Alimova, A. Abdallah, “HKR For Handwritten Kazakh & Russian Database,” arXiv preprint arXiv:2007.03579, 2020

ГРАФИК
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Раздел 1. Введение в глубокое обучение	23.10.2021 г.	Сдал
Раздел 2. Проблематика распознавания рукописного оффлайн текста	30.11.2021 г.	Сдал
Раздел 3. Методы распознавания рукописного текста	6.11.2021 г.	Сдал
Раздел 4. Выбор датасета и сегментация текста	20.11.2021 г.	Сдал
Раздел 5. Предлагаемая модель и итоги обучения	25.12.2021 г.	Сдал

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант, (уч. степень, звание)	Сроки	Подпись
Нормоконтроль	А.Т. Ахмедиярова, PhD, ассиестент-профессор	02.05.22	
Программное обеспечение	Н.К. Мукажанов, PhD, ассоциированный профессор	02.05.22	
Антиплагиат	Б. Аубакиров Ассистент	02.06.22	

Дата выдачи задания "16" октября 2021 г.

Заведующий кафедрой _____ А.Н. Молдагулова

Научный руководитель _____ Д.Б. Нурсеитов

Задание принял к исполнению магистрант _____ Е.Б. Бейсенканов

Дата "02" 06 2022 г.

АҢДАТПА

Орыс тіліндегі қолжазба мәтінін танудағы заманауи зерттеулер тек басылған және басылған мәтінге бағытталған, ал Курсив мәтінінде үлкен проблема болып қала береді. Курсивті мәтінді жеңе алатын бағдарламалар коммерциялық мақсатта жасалған және көптеген пайдаланушылар үшін қол жетімді емес. Бұл диссертация TensorFlow Framework көмегімен Python бағдарламалау тілінде жазылған терең оқытуды қолдана отырып, желінің қолмен жазылған офлайн мәтінін тану үшін нейрондық желіні құру тұжырымдамасын ұсынады. Жұмыс барысында терең оқытудың теориялық негізі, қолмен жазылған офлайн мәтінді тану мәселелері, қолмен жазылған мәтіннің датасеттерін салыстыру, диссертацияның берілген тақырыбы аясында бір-бірімен салыстырылатын нейрондық желілердің әртүрлі түрлері қарастырылады.

Түйінді сөздер: терең оқыту, нейрондық желілер, қайталанатын нейрондық желілер, конвульсиялық нейрондық желілер, қолжазбаны тану, TensorFlow

АННОТАЦИЯ

Современные исследования в распознавании рукописного текста на русском языке направлены лишь на печатный и рукопечатный текст, а в курсивный же текст остается большой проблемой. Те программы, что могут справиться с курсивным текстом, разработаны в коммерческих целях и не доступны для широкого круга пользователей. В данной диссертации представлена концепция создания нейронной сети для распознавания рукописного оффлайн текста сетевых, использующей в основе глубокое обучение, написанной на язык программирования Python с использованием фрэймворка TensorFlow. В ходе работы рассматриваются теоретическая основа глубокого обучения, проблематика самого распознавания рукописного оффлайн текста, сравнение датасетов рукописного текста, разные виды нейронных сетей, сравнивающиеся между собой в рамках заданной темы диссертации.

Ключевые слова: глубокое обучение, нейронные сети, рекуррентные нейронные сети, свёрточные нейронные сети, распознавание рукописного текста, TensorFlow

ANNOTATION

Modern research in handwriting recognition in Russian is aimed only at printed and handwritten text, while cursive text remains a big problem. Those programs that can handle cursive text are developed for commercial purposes and are not available to a wide range of users. This dissertation presents the concept of creating a neural network for recognizing handwritten offline text online, based on deep learning, written in the Python programming language using the TensorFlow framework. In the course of the work, the theoretical basis of deep learning, the problems of handwritten offline text recognition itself, comparison of handwritten text datasets, different types of neural networks that are compared with each other within the framework of a given dissertation topic are considered.

Keywords: deep learning, neural networks, recurrent neural networks, convolutional neural networks, handwriting recognition, TensorFlow

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ 9
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 10
ВВЕДЕНИЕ 11
1. Введение в глубокое обучение 13
1.1 Краткое введение в машинное обучение 13
1.2 Вычислительные примитивы 15
1.2.1 Умножение матричных векторов 15
1.2.1.1 Поэлементное умножение 16
1.2.1.2 Свёртка 16
1.2.2 Нелинейные функции активации 17
1.3 Архитектура искусственных нейронных сетей 18
1.3.1 Многослойные персептроны(MLP) 19
1.3.1.1 Входной слой 19
1.3.1.2 Скрытые слои 19
1.3.1.3 Выходной слой 20
1.3.2 Свёрточные нейронные сети(CNN) 20
1.3.2.1 Свёрточный слой 21
1.3.2.2 Уровень подвыборки 22
1.3.2.3 Выходной уровень для классификации 23
1.3.2.4 Шаблоны проектирования CNN 23
1.3.3 Рекуррентные нейронные сети(RNN) 23
1.3.3.1 Стандартные RNN 24
1.3.3.2 Долгая краткосрочная память(LSTM) 24
1.3.3.3 Двунаправленные RNN(BiRNN) 25
1.4 Показатели эффективности 26
1.4.1 Частота ошибок меток(LER) 26
1.4.2 Частота ошибок символов(CER) 26
1.4.3 Частота ошибок в словах(WER) 27
1.5 Оптимизация на основе градиентов 27
1.5.1 Функция потерь 27
1.5.1.1 Кросс-энтропия 28
1.5.1.2 Временная классификация компонентов(CTC) 28
1.5.2 Градиентный спуск 28
1.5.2.1 Обратное распространение 29
1.5.2.2 Обратное распространение во времени(BPTT) 29
1.5.3 Современные алгоритмы и эвристики оптимизации 30
1.5.4 Полезные методы оптимизации/регуляризации 30
1.5.4.1 Dropout 30
1.5.4.2 Пакетная нормализация 30
1.5.4.3 Early stopping 31
1.6 Заключение 31

2. Проблематика распознавания рукописного оффлайн текста 32
3. Методы распознавания рукописного текста 35
4. Выбор датасета и сегментация текста 37
4.1 Понятие сегментации 37
4.2 Сегментация строк 37
4.3 Сегментация слов 38
5 Предлагаемая модель и итоги обучения 39
5.1 Свёрточные блоки 40
5.2 Декодер 40
5.3 Нейросетевая темпоральная классификация (Connectionist temporal classification – CTC) 40
5.4 Данные 41
5.5 Обучение 42
5.6 Итоги обучения 43
5.7 Финальный тест 45
ЗАКЛЮЧЕНИЕ 48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей диссертации использованы ссылки на следующие стандарты:
ГОСТ 19.101-77. Описание программы.
СТ РК 34.014-2002. Информационные технологии.
ГОСТ 19.201-78. Единая система программной документации.
ГОСТ 28195-99. Оценка качества программных средств.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ML – Machine Learning
DL- Deep Learning
MLP – Multilayer perceptron
LER – Label Error Rate
CER – Character Error Rate
WER – Word Error Rate
RNN – Recurrent Neural Network
CNN – Convolutional Neural Network
LSTM – Long Short-Term Memory
BiRNN – Bidirectional Recurrent Neural Network
ANN – Artificial Neural Network
DNN -Deep Neural Network
CTC -Connectionist Temporal Classification
GD – Gradient Descent
BPTT- Backpropagation through time

ВВЕДЕНИЕ

Актуальность темы. С каждым днем все больше достижений в области информатики, особенно в машинном обучении и анализе изображений, находят свое применение в реальной жизни. Компьютерное зрение, зарекомендовавшее себя как один из самых эффективных способов анализа изображений и видео, облегчающий или заменяющий работу человека, давно интегрировано во множество сфер деятельности. Распознавание рукописного текста (HTR) является одной из ключевых задач компьютерного зрения и имеет следующие преимущества: доступ к данным: возможность поиска и удобство использования; экономия времени и памяти; повышение удовлетворенности клиентов; повышение безопасности.

Цель. Создание программы на основе глубокого обучения для распознавания рукописного оффлайн текста.

Основные задачи

- Провести анализ основных подходов к решению задачи распознавания рукописного текста;
- Исследовать понятия глубокое обучение, нейронные сети и их разновидности;
- Изучить готовые базы (датасеты) рукописного текста;
- Создание программы на основе глубокого обучения и нейронных сетей для решения задачи распознавания рукописного текста;
- Обучение нейронной сети на основе изученных ранее готовых баз рукописного текста

Новизна. В этом исследовании была предложена программа для распознавания рукописного оффлайн текста с использованием глубокого обучения и нейронных сетей. В эпоху информации сохранность и её цифровизация является важной задачей. Программ или исходного кода для подобной программы нет в свободном доступе, а потребность в цифровизации текста как никогда актуальна. Поэтому мы предложили эффективную программу для выполнения данной задачи.

Теоретическая значимость. Разработка программы для распознавания рукописного оффлайн текста на основе глубокого обучения позволит улучшить нынешнее понимание цифровизации рукописного текста и создаст предпосылки для проектирования универсальной программы, позволяющей распознавать любой язык.

Практическая значимость. Разработка программы для распознавания рукописного оффлайн текста на основе глубокого обучения позволит повысить эффективность работы систем, где требуется работа с рукописным текстом, а также позволит сохранить старинные рукописи в цифровом виде.

Апробация результатов исследования. Диссертационная научно-исследовательская работа приняла участие в международной научной конференции «Сатпаевские чтения – 2022. Тенденции современных

исследований» на тему «Применение глубокого обучения в задаче распознавания рукописного оффлайн текста».

Структура и объем диссертации. Данная работа состоит из введения, пяти разделов, заключения, списка использованных источников и приложений, содержит 53 страницы основного текста (8 рисунков). Список использованных источников содержит 59 наименований источников.

1 Введение в глубокое обучение

В этой главе мы даем краткое введение в основные методы глубокого обучения, которые мы использовали в этой диссертации. Сначала мы дадим краткое введение в машинное обучение (поддоменом которого является глубокое обучение), а затем представим основные вычислительные примитивы, используемые в этой диссертации для создания систем глубокого обучения (практически все наиболее значимые системы глубокого обучения состоят из двух основных типов операций: линейных и нелинейных).Затем мы представим основные типы архитектур нейронных сетей: многослойные персептроны, сверточные нейронные сети и рекуррентные нейронные сети. Они могут быть использованы для решения различных типов проблем и могут быть творчески объединены. Затем мы вводим основные показатели производительности, используемые для тестирования систем распознавания рукописного ввода: частота ошибок надписей (дополнением к которой является точность), частота ошибок символов (CER) и частота ошибок слов (WER). В следующем разделе мы представляем оптимизацию на основе градиента, вводим функции потерь, градиентный спуск (с обратным распространением и обратным распространением во времени) и некоторые современные алгоритмы оптимизации (например, ADAM) и методы регуляризации (например, dropout).

1.1 Краткое введение в машинное обучение

В этом разделе мы очень кратко представим основные концепции в области машинного обучения, подполе которого является глубокое обучение.

В области машинного обучения и, в частности, в нейронных сетях используется подход, обозначаемый как управляемый данными. Этот подход может быть интуитивно описан следующим образом. Предположим, мы хотим распознать почерк на определенном изображении. В отличие от большинства компьютерных наук, мы не знаем, как явно написать алгоритм, который может достичь этого. Даже когда исследователь способен распознать почерк на конкретном изображении, он не может сознательно получить доступ к тому, как его мозг выполняет распознавание, чтобы непосредственно перевести процесс в программируемый алгоритм. Вместо этого мы сначала обучим алгоритм на наборе данных. Мы также будем использовать отдельные наборы данных для проверки нашего подхода и проверки того, какую производительность мы могли бы ожидать в реальном мире. Набор данных проверки может быть использован для калибровки алгоритма. В идеале к тестовому набору данных прикасаются только один раз, чтобы оценить конечную производительность алгоритма.

Три различные парадигмы обучения являются наиболее известными и наиболее актуальными в машинном обучении. В случае контролируемого обучения, которое, безусловно, будет наиболее актуальным для этого тезиса, наборы данных помечаются, и ожидается, что алгоритм изучит отображение f :

ввод → метка. Другая парадигма - неконтролируемое обучение, при котором предоставляются только входные данные (без меток), и ожидается, что алгоритм научится генерировать данные, аналогичные входным данным в обучающем наборе. Наконец, обучение с подкреплением использует процесс, аналогичный тому, как обучают животных: ожидается, что алгоритм научится действовать в определенной среде, чтобы максимизировать совокупное вознаграждение.

Распознавание рукописного текста вписывается в парадигму контролируемого обучения. В этой задаче в качестве входных данных предоставляется массив пикселей, представляющих изображение, и алгоритм должен научиться присваивать ему последовательность меток. Конвейер распознавания рукописного ввода может быть формализован следующим образом:

1. Ввод: набор из N изображений, каждое из которых помечено последовательностью меток; каждая метка принадлежит алфавиту из K символов. Эти данные обозначаются как обучающий набор.

2. Обучение: Мы используем обучающий набор для изучения сопоставления между изображениями рукописного ввода и соответствующей последовательностью меток. Этот шаг обозначается как изучение модели.

3. Оценка: Качество модели оценивается путем запроса ее предсказать последовательности меток для нового набора изображений (тестового набора), никогда ранее не виденных, и сравнения предсказаний с истинными последовательностями меток. Интуиция, лежащая в основе этой процедуры, заключается в том, что производительность тестового набора должна отражать производительность в реальном мире.

Набор проверки используется для выполнения так называемой настройки гиперпараметров. Гиперпараметры — это параметры, значения которых обычно выбираются с использованием процедуры, отличной от той, которая используется для обучаемых параметров. Например, для нейронных сетей некоторыми примерами гиперпараметров являются количество слоев, количество единиц (нейронов), скорость обучения процедуры стохастического градиента. В случае нейронных сетей, в то время как обучаемые параметры обычно модифицируются с использованием стохастического градиентного спуска, гиперпараметры обычно выбираются путем тестирования различных значений в наборе проверки и выбора значения, которое работает лучше всего. Важно не использовать тестовый набор для настройки гиперпараметров, чтобы не подвергать риску перенастройку тестового набора. Переоснащение тестового набора означает, что модель получает гораздо лучшую производительность, чем та, которая фактически наблюдалась бы во время развертывания для реальных данных. Использование тестового набора для настройки гиперпараметров может быть интерпретировано как эффективное включение его в обучающий набор, поэтому производительность, показанная в тестовом наборе, будет слишком оптимистичной по сравнению с тем, что можно ожидать при развертывании модели. С другой стороны, если тестовый

набор используется только один раз, в самом конце, можно ожидать, что он обеспечит полезную меру того, насколько хорошо модель будет обобщаться на реальные данные. По этим причинам мы должны оценивать тестовый набор только один раз, в самом конце.

Набор проверки, который можно использовать для настройки гиперпараметров, можно получить, разделив обучающий набор на несколько меньший обучающий набор и используя остальные данные в качестве набора проверки. Валидационный набор используется как поддельный набор тестов для настройки гиперпараметров. Для многих наборов данных, и особенно для тех, которые чаще всего используются для сравнительного анализа, уже предусмотрено разделение на различные обучающие и проверочные наборы. Это может помочь упростить сравнение производительности, полученной с помощью различных алгоритмов.

1.2 Вычислительные примитивы

В самом общем виде нейронные сети можно интерпретировать как вычислительные графы, состоящие из примитивных операций. Вычислительные графики допускают более богатый набор примитивных операций, чем те, которые мы описали здесь, но мы ограничимся наиболее часто используемыми и наиболее успешными операциями.

1.2.1 Умножение матричных векторов

Матрично-векторное умножение, вероятно, является наиболее широко используемым вычислительным примитивом глубокого обучения. Это тип линейной операции (без нелинейного эффекта) и используется в качестве компонента во всех наиболее успешных архитектурах нейронных сетей, включая те, которые используются в этой диссертации: многослойные персептроны, сверточные нейронные сети и рекуррентные нейронные сети.

В наиболее распространенной настройке вектор $x \in R^n$ представляет информацию, ранее обработанную нейронной сетью, и/или необработанную (входную) информацию. Матрица $W \in R^{m*n}$ является плотной, и все ее элементы поддаются модификации (обучению). Результатом этой операции является вектор $y \in R^m$:

$$y = W * x \qquad (1.1)$$

Обычно к результату умножения матричного вектора добавляется вектор смещений $b \in R^m$, так что предыдущее уравнение становится:

$$y = W * x + b \qquad (1.2)$$

Мы можем получить тот же результат, добавив значение 1 к концу вектора столбца x , а столбец b - к матрице W , так что мы можем сохранить 1.1.

В следующих подразделах мы представим различные примитивы, которые можно интерпретировать как умножение матриц на векторы, где матрицы факторизуются различными способами.

1.2.1.1 Поэлементное умножение

Поэлементное умножение можно интерпретировать как матричное векторное умножение, где матрица является диагональной. Здесь снова вектор $x \in R^n$ представляет информацию, ранее обработанную нейронной сетью, и/или необработанную (входную) информацию, в то время как диагональная матрица $D \in R^{n*n}$ содержит обучаемые параметры. Путем преобразования диагональной матрицы в вектор $d \in R^n$:

$$y = D * x = d \bullet x \qquad (1.3)$$

Смещения могут быть добавлены аналогично тому, как это было сделано в предыдущем подразделе.

Поэлементное умножение, в частности, используется в современных рекуррентных нейронных сетях с длительной кратковременной памятью (LSTM). Операция свертки, вычислительный примитив, который мы опишем далее, также может быть разложена на множители в виде дискретных преобразований Фурье и поэлементных умножений, но мы не будем здесь вдаваться в подробности.

1.2.1.2 Свертка

Операция свертки широко используется в современных архитектурах нейронных сетей, в частности в сверточных нейронных сетях (CNN). Мы будем описывать только дискретную свертку, применяемую к функциям с конечной поддержкой (векторы в 1D случае и изображения в 2D случае), поскольку они наиболее актуальны для нейронных сетей. Дискретная свертка может быть переформулирована как умножение матричных векторов.

Мы рассмотрим два вектора f и g с индексами в $\{0..., N-1\}$. Результат их 1D свертки может быть записан в виде:

$$o[n] = f[n] \circledast g[n] = \sum_{u=0}^{u=N-1} f[n-u] * g[u] \qquad (1.4)$$

где ② обозначает оператор свертки. Это может быть распространено на 2D-массивы следующим образом:

$$o[m,n] = f[m,n] \circledast g[m,n] = \sum_{u=0}^{u=M-1} \sum_{v=0}^{v=N-1} f[m-u, n-v] * g[u,v] \qquad (1.5)$$

1.2.2 Нелинейные функции активации

Все ранее описанные вычислительные примитивы являются линейными. Нелинейные операции также необходимы, иначе система машинного обучения, содержащая только линейные операции, не была бы достаточно выразительной, независимо от того, сколько линейных операций было составлено. Независимо от того, сколько линейных операций состоит, вся система не более мощна, чем простая линейная регрессия. С другой стороны, даже композиция с одной нелинейной операцией делает нейронные сети универсальными аппроксиматорами непрерывных функций.

В нейронных сетях нелинейность вводится с использованием концепции функции активации, которая применяется поэлементно к входным данным.

Исторически сложилось так, что самой популярной функцией активации была сигмовидная функция:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (1.6)$$

Еще одной функцией активации с долгой историей является функция tanh:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (1.7)$$

В настоящее время одной из наиболее успешных функций активации является выпрямленный линейный блок (ReLU):

$$\mathrm{ReLU}(x) = \begin{cases} x, & \text{if } x > 0. \\ 0, & \text{otherwise.} \end{cases} \qquad (1.8)$$

Интерес функций активации ReLU заключается в том, что они обеспечивают меньше исчезающих градиентов, поскольку они меньше насыщают (по сравнению с логистической функцией)

Было обнаружено, что Leaky ReLU либо соответствуют, либо превосходят результаты некоторых авторов по производительности [1]:

$$\mathrm{LReLU}(x) = \begin{cases} x, & \text{if } x > 0. \\ ax, & \text{otherwise.} \end{cases} \qquad (1.9)$$

a — это коэффициент масштабирования, который является фиксированным. Параметризованные выпрямленные линейные единицы (PReLU) [2] — это еще одна выпрямленная функция активации с тем же уравнением, что и для LReLU. Разница, однако, в том, что a — это обучаемый параметр. Для каждого нейрона может быть выбрано другое значение a или значения a могут быть "привязаны" так, чтобы несколько нейронов имели одно и то же значение. Это помогает уменьшить количество обучаемых параметров и, таким образом, может предотвратить переобучение.

Нелинейность softmax обычно используется в условиях контролируемого обучения с дискретными метками (классами). В этой настройке он чаще всего

комбинируется с функцией кросс-энтропийных потерь. Учитывая K различных классов, softmax принимает в качестве входных данных вектор x из K действительных значений (этот вектор может быть получен из вектора входных данных или ранее обработанных значений, например, путем умножения матричных векторов). Затем softmax обрабатывает эти K значений для получения вероятностей, соответствующих K классам. Для каждого класса k ∈ 1, .., K его соответствующая вероятность равна:

$$p(k) = \frac{e^{x_k}}{\sum_{j=1}^{K} e^{x_k}} \quad (1.10)$$

1.3 Архитектура искусственных нейронных сетей

Искусственные нейронные сети (ANN) являются основной темой этой диссертации. Как следует из их названия, ANN представляют собой архитектуры машинного обучения, вдохновленные их биологическими аналогами.

Основной вычислительной единицей в ANN является искусственный нейрон. Первая знаменитая модель искусственного нейрона была представлена в 1943 году [3]. Мы будем обозначать результат вычисления нейрона термином активация. В биологии нейроны соединены синапсами с различной силой (весом), и каждый нейрон имеет порог (чья электрическая активность, которая может быть грубо интерпретирована как активация в нашем описании, должна быть больше, чем для срабатывания нейрона). Искусственные нейроны имеют соответствующие (и упрощенные) веса w, соединяющие их с другими нейронами, и смещение b (упрощенный эквивалент биологического порога). Чтобы связать эту модель со структурой машинного обучения, веса w и смещение b являются обучаемыми параметрами (значения которых оптимизируются в процессе обучения).

Самые простые, широко встречающиеся современные искусственные нейроны вычисляют точечное произведение, за которым следует нелинейность по элементам. Учитывая набор входов x, активация h нейрона, подключенного к входам с использованием весов w и смещения b, равна:

$$h = f(w * x + b) \quad (1.11)$$

где f - элементная нелинейность.

Это сильно упрощенная модель биологических нейронов, которые демонстрируют гораздо больше функций [4].

Чтобы отличить искусственные нейроны от их гораздо более сложных биологических аналогов, их часто обозначают термином units. В этом тезисе мы будем в основном использовать термин "единицы", и всякий раз, когда мы используем термин "нейроны", подразумеваются искусственные нейроны.

Основным структурным элементом ANN является слой, содержащий несколько блоков. Первый и последний слои несколько особенные, содержащие

входы и, соответственно, выходы ANN. Все остальные слои обозначаются как скрытые слои.

Искусственные нейронные сети бывают разных видов (архитектур), особенно после всплеска исследований нейронных сетей в последнее десятилетие. В этой главе мы представим три наиболее важные архитектуры: многослойный персептрон (MLP), сверточную нейронную сеть (CNN) и рекуррентную нейронную сеть (RNN). MLP и CNN также называются нейронными сетями с прямой связью (потому что они не используют повторение в своих вычислениях), в отличие от RNN. Эти архитектуры основаны на несколько разных вычислительных примитивах. Вкратце, они были введены для решения различных проблем: MLP с общими одиночными входами без какой-либо конкретной структуры, CNN с визуальными входами (в частности, изображениями) и RNN с временными последовательностями.

Эти архитектуры могут быть смешаны и составлены в соответствии с проблемой, которую нам нужно решить. Широко используемая аналогия — это сравнение слоев нейронной сети с кубиками lego, которые можно творчески собирать для решения новых задач или повышения производительности при решении старых. Действительно, большая часть исследований в области нейронных сетей включает в себя проектирование новых слоев и / или сборку их в новые архитектуры, соответствующие задачам, которые мы пытаемся решить.

1.3.1 Многослойные персептроны (MLP)

Многослойные персептроны (MLP) также появляются в литературе под названием Глубокие нейронные сети (DNN), что несколько двусмысленно (поскольку другие архитектуры нейронных сетей с несколькими слоями также можно назвать глубокими нейронными сетями).

Мы кратко опишем каждый тип слоя в MLP.

1.3.1.1 Входной слой

Входной слой представляет входные данные в виде вектора и связан с первым скрытым слоем. Чтобы сделать обозначения одинаковыми с обозначениями скрытых слоев, входной слой можно обозначить как $h0$, а первый скрытый слой - как $h1$.

1.3.1.2 Скрытые слои

В их наиболее распространенном варианте каждый скрытый слой MLP $i \in \{1, ..M\}$ представляет собой вектор единиц hi, связанный с предыдущим слоем $hi-1$ весовой матрицей Wi и вектором смещений bi. Соответствующее

уравнение имеет вид:

$$h_i = f(W_i * h_{i-1} + b_i) \qquad (1.12)$$

где f - элементная нелинейность. В подавляющем большинстве случаев одна и та же нелинейность используется для каждой единицы измерения в слое, хотя также возможны разные нелинейности для разных единиц измерения.

Матрица Wi чаще всего является плотной; в этом случае соответствующий слой обозначается как полностью связанный.

1.3.1.3 Выходной слой

В случае классификации выходной слой чаще всего можно описать как скрытый слой с нелинейностью softmax, выходными весами Wo и смещениями bo:

$$o = softmax(W_o * h_M + b_o) \qquad (1.13)$$

1.3.2 Сверточные нейронные сети (CNN)

В этом подразделе мы обсудим Сверточную нейронную сеть (CNN), которая представляет собой специализированную архитектуру нейронной сети, первоначально предложенную для обработки изображений. Как и MLP / DNN, CNN также содержат нейроны, соединенные обучаемыми весами и смещениями, и операции, выполняемые нейронами (единицами), аналогичны MLP: каждый нейрон получит некоторые входные данные, затем выполнит точечное произведение, а затем применит нелинейность к результату. Вся система также дифференцируема (как и MLP), использует функцию потерь и применяются типичные процедуры оптимизации (стохастический градиент, обратное распространение). Хотя в основном используется для обработки изображений, в последние годы использование различных вариантов CNN было распространено на многие другие области, например, распознавание речи [5] или обработка естественного языка [6]. Мы будем считать, что исходными данными для CNN являются изображения, чтобы упростить презентацию; расширения для CNN, обрабатывающие текст или другие типы входных данных, просты, и многие идеи и мотивы естественным образом переносятся на эти случаи.

Первое существенное различие между CNN и MLP заключается в характере взаимодействия каждого нейрона. В то время как в MLP нейроны в слое полностью связаны со всеми нейронами в последовательном слое, в CNN схема подключения разреженная и локальная. Это вдохновлено работой в области неврологии, где Хьюбел и Визель [7] обнаружили, что нейроны в зрительной коре кошек действуют как локальные фильтры над их входным пространством. Небольшая подобласть входного сигнала, к которому

чувствительна каждая клетка, называется рецептивным полем, этот термин также используется для обозначения паттерна связности нейронов CNN. Еще одно концептуальное различие между CNN и MLP заключается в том, что в CNN нейроны / единицы реплицируются по 2D-массивам. 2D-массив реплицированных объектов имеет одни и те же обучаемые параметры и называется картой объектов. Это означает, что веса и смещения нескольких нейронов имеют одинаковые значения. Общие обучаемые параметры также обозначаются как фильтр. Как и в случае с RNN, градиент в отношении совместно используемого обучаемого параметра представляет собой сумму градиентов в отношении всех совместно используемых параметров.

Реплицируемые объекты и реплицируемые фильтры позволяют обнаруживать объекты независимо от положения на входном изображении, где они появляются. Это свойство называется эквивариантностью и примерно означает, что результирующая карта объектов изменяется аналогично тому, как изменялось бы входное изображение. Если бы изображение было переведено на один пиксель влево, то же самое произошло бы и с результирующей картой объектов после применения фильтра.

Подобно MLP, CNN представляют собой глубокие нейронные сети с несколькими уровнями. CNN получается путем многократной укладки блоков сверточных и подвыборочных слоев. Для классификации за повторяющимися блоками сверточных и подвыборочных слоев чаще всего следуют полностью связанные слои и слой softmax.

Для наглядности мы обсудим, как работают CNN, когда в качестве входных данных предоставляется один пример (изображение). Современные CNN обычно обучаются на мини-пакетах, содержащих несколько изображений. Обсуждение здесь можно легко распространить на случай mini-batch, добавив дополнительное измерение (соответствующее mini-batch) к каждому тензору.

1.3.2.1 Сверточный слой

Для простоты мы будем обсуждать здесь только свертку шага 1. Мы также не будем обсуждать настройки нулевого заполнения. Обсуждение шагов и отступов для сверточных слоев приведено в [8].

Карты объектов, рассмотренные в предыдущем подразделе, будут получены в CNN следующим образом. Входное изображение свертывается с помощью линейного фильтра, добавляется член смещения, а затем применяется нелинейная функция f. Свертку можно интерпретировать как многократное применение одной и той же функции к подобластям всего изображения. Чтобы обеспечить более богатое представление данных, каждый скрытый слой сверточной или подвыборки содержит несколько карт объектов hk, $k \in 0, ..., K$. Мы обозначим k-ю карту объектов через hk, а соответствующий фильтр, состоящий из обучаемых весов и смещений, - через Wk и bk, а входное изображение - через x. Затем нелинейная функция применяется поэлементно к

каждому пикселю на каждой карте объектов. Соответствующее уравнение для пикселя [i, j] в карте объектов hk имеет вид:

$$h_k[i, j] = f((W_k \textcircled{2} x)[i, j] + b_k) \qquad (1.14)$$

где f может быть любой нелинейностью, причем ReLU является популярным современным выбором.

Приведенное выше обсуждение было бы применимо именно к CNN с одним сверточным слоем. Аналогично теоретическим результатам, полученным для глубоких MLP [9], глубокие CNN более выразительны, чем неглубокие [10].

1.3.2.2 Уровень подвыборки

Существует множество различных предложений для слоев подвыборки, но здесь мы обсудим слой с максимальным объединением (max-pooling layer), который является наиболее успешным и наиболее широко используемым слоем подвыборки. Другими заметными видами использования подвыборки являются средний уровень объединения [11] и использование шага, превышающего 1 в предыдущем сверточном слое [12]. Общим для этих слоев является то, что при применении к группе пикселей карты объектов они сначала необязательно выполняют преобразование пикселей, а затем выбирают один из них в качестве результата. Что характерно для максимального объединения(max-pooling), так это то, что он выбирает пиксель с максимальным значением (из группы).

Наиболее распространенным вариантом является разделение пикселей на неперекрывающиеся группы, причем наиболее распространенным выбором являются группы шириной 2 пикселя и высотой 2 пикселя. При использовании с максимальным объединением(max-pooling) это называется максимальным объединением 2 × 2(max-pooling 2 × 2) и является наиболее широко используемой настройкой для слоя объединения.

Более подробное обсуждение различных типов слоев объединения с различными режимами группировки пикселей (с потенциальным перекрытием и различными вариантами подвыборки, а также размерами соответствующих результирующих карт объектов) представлено в [8].

Учитывая входную карту объектов i, выходная карта объектов o задается:

$$o[m, n] = \max\,(i[2*m, 2*n], i[2*m+1, 2*n], i[2*m, 2*n+1], i[2*m+1, 2*n+1]) \quad (1.15)$$

Основная мотивация, лежащая в основе слоя максимального объединения, заключается в обеспечении некоторой инвариантности трансляции и устойчивости к небольшим искажениям изображения. Результирующее уменьшение размера карты объектов (как ширина, так и высота делятся на 2 в случае максимального объединения 2 × 2) также помогает снизить вычислительные требования CNN.

Интересной особенностью слоя максимального объединения является то, что он не содержит никаких обучаемых параметров.

1.3.2.3 Выходной уровень для классификации

Когда CNN используется для классификации, все пиксели в картах объектов, полученных в результате последнего слоя (сверточного, нелинейного или подвыборки), преобразуются в массив 1D. Затем можно добавить полностью связанные слои и слой softmax для выполнения классификации.

1.3.2.4 Шаблоны проектирования CNN

В этом разделе мы очень кратко упомянем некоторые из наиболее влиятельных архитектур CNN в области глубокого обучения.

Одной из первых влиятельных архитектур CNN, обученных обратному распространению, была LeNet-5, представленная Яном Лекуном для задачи распознавания цифр [13].

AlexNet [14] — это архитектура, которая популяризировала использование CNNS в компьютерном зрении после победы в конкурсе ILSRVC challenge в 2012 году. Это также популяризировало использование графических процессоров для обучения нейронных сетей и привело к тому, что глубокое обучение стало доминирующей парадигмой в компьютерном зрении.

GoogLeNet [15] выиграла конкурс ILSRVC 2014 и представила начальные модули, которые значительно сократили количество обучаемых параметров (по сравнению с AlexNet). Архитектура GoogLeNet прошла несколько итераций, самой последней из которых является версия 4 [16].

Остаточные CNNS (ResNets) [17] выиграли ILSVRC 2015 и внедрили пропуск соединений, чтобы облегчить обучение очень глубоких сетей, уменьшая проблему исчезающих градиентов. По состоянию на 2018 год варианты ResNet часто являются выбором по умолчанию для использования CNNS на практике.

1.3.3 Рекуррентные нейронные сети (RNN)

Рекуррентные нейронные сети (RNN) расширяют подход глубокой нейронной сети к последовательным входным данным. Вероятно, наиболее характерной особенностью RNN является то, что они вынуждены использовать одну и ту же "функцию перехода" для каждого временного шага (путем "привязки" весовой матрицы на каждом временном шаге). Таким образом, они могут создавать объекты, которые не зависят от конкретного временного интервала, на котором они появляются, и могут научиться предсказывать выходные данные / выходные последовательности из входных последовательностей независимо от длины последовательности.

1.3.3.1 Стандартные RNN

Стандартные RNN представляют собой простейший тип RNN. Уравнение для стандартного RNN имеет вид:

$$h_t = \sigma(W_{hh} * h_{t-1} + W_{hx} * x_t + b_h) \qquad (1.16)$$

Здесь ht - это представление, полученное RNN для каждого входного сигнала xt в момент времени t. Whh и bh поддаются обучению. σ - это элементная нелинейность.

Затем предсказания могут быть вычислены на основе изученных представлений ht. Например, в случае, когда мы хотели бы предсказать новый результат (который мы будем обозначать через y ^ t) на каждом временном шаге t, мы можем использовать следующие уравнения:

$$y_t = W_{yh} * h_t + b_y \qquad (1.17)$$

Здесь Wyh и by являются обучаемыми параметрами.

1.3.3.2 Долгая краткосрочная память (LSTM)

Теоретически рекуррентные нейронные сети являются универсальными аппроксиматорами последовательностей (при достаточном количестве скрытых единиц RNN может аппроксимировать любое измеримое отображение последовательности в последовательность с произвольной точностью) [18] и являются полными по Тьюрингу [19]. Однако на практике исследователи обнаружили, что обучать RNN трудно из-за проблем оптимизации, таких как проблема исчезающего градиента [20], [21]. Ячейка долговременной кратковременной памяти (LSTM) [22], которая описана в этом подразделе, является одним из способов решения этой проблемы.

Уравнения LSTM являются:

$$c_t = f_t \bullet c_{t-1} + i_t \bullet \tanh(W_{hc} * h_{t-1} + W_{xc} * x_t + b_c) \qquad (1.18)$$

где ct является состоянием ячейки, ft - выходом элемента забывания, а it - значением входного элемента.

$$f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} * c_{t-1} + b_f) \qquad (1.19)$$

Wcf соответствует весу соединения глазка между ячейкой и входным затвором.

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} * c_{t-1} + b_i) \quad (1.20)$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} * c_t + b_o) \quad (1.21)$$

$$h_t = o_t \bullet \tanh(c_t) \quad (1.22)$$

• означает поэлементное умножение. Эти уравнения соответствуют варианту LSTM с добавленными соединениями для глазков. Мы можем упростить LSTM, удалив эти связи; тогда уравнения входного, выходного и выходного вентилей станут:

$$f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + b_f) \quad (1.23)$$

$$i_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i) \quad (1.24)$$

$$o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o) \quad (1.25)$$

Уравнение для конечного результата (ht) остается прежним.

1.3.3.3 Двунаправленный RNN (BiRNN)

В стандартном RNN выходные данные в данный момент времени t зависят исключительно от входных данных от x0 до xt (через скрытые слои, развернутые во времени от h0 до ht−1). Однако, хотя это имеет смысл в некоторых контекстах, многие последовательности содержат информацию, относящуюся к выходу yt как до, так и после временного шага t. Например, при распознавании рукописного ввода как последовательность нарисованных символов до, так и после текущего символа предоставляют информацию о нарисованном символе, обрабатываемом в данный момент. Чтобы использовать всю эту информацию, нам нужна измененная архитектура.

Двунаправленные RNN добавляют еще один набор скрытых слоев к рекуррентной сети, которая движется назад во времени, помимо набора скрытых слоев, который движется вперед во времени стандартного RNN. Эти два набора скрытых слоев полностью разделены и не взаимодействуют друг с другом, за исключением того факта, что они оба используются для вычисления выходных данных. Используя обучаемые параметры, BiRNN сначала выполняется вперед во времени (от времени 0 до конца) для вычисления прямых скрытых слоев, а затем выполняется назад во времени (от конца до времени 0) для вычисления обратных скрытых слоев. Наконец, используя значения на обоих скрытых слоях для заданного временного шага, вычисляется результат, соответствующий каждому временному шагу. Обозначая прямой

слой через →−ht , а обратный слой через ←h−t , уравнения 1.16 для сети модифицируются следующим образом :

$$\overrightarrow{h_t} = \sigma(W_{hh} * \overrightarrow{h_{t-1}} + W_{hx} * x_t + \overrightarrow{b_h}) \quad (1.26)$$

$$\overleftarrow{h_t} = \sigma(W_{hh} * \overleftarrow{h_{t+1}} + W_{hx} * x_t + \overleftarrow{b_h}) \quad (1.27)$$

$$\hat{y}_t = W_{y\overrightarrow{h}} * \overrightarrow{h_t} + W_{y\overleftarrow{h}} * \overleftarrow{h_t} + b_y \quad (1.28)$$

1.4 Показатели эффективности

Мы используем показатели эффективности для характеристики производительности наших систем распознавания. Обычно используемыми показателями производительности для систем машинного обучения, которые помечают примеры дискретным числом потенциальных меток (классов), являются частота ошибок меток (LER) для непоследовательных данных и частота ошибок символов (CER) и частота ошибок слов (WER) для последовательных данных и, в частности, распознавания рукописного ввода.

1.4.1 Частота ошибок меток (LER)

Для набора данных S LER — это соотношение примеров, неправильно помеченных классификатором:

$$LER = \frac{\text{number of errors}}{\text{number of examples}} \quad (1.29)$$

Другим часто используемым показателем является точность, которую можно описать как 1 − LER.

1.4.2 Частота ошибок символов (CER)

Формула для частоты ошибок символов равна:

$$CER = \frac{S + D + I}{N} \quad (1.30)$$

где S - количество замененных символов, D - количество удаленных символов, I - количество вставленных символов, а N - количество символов в основной правде.

1.4.3 Частота ошибок в словах (WER)

Аналогично формуле для частоты ошибок символов, формула для частоты ошибок слов имеет вид:

$$WER = \frac{S + D + I}{N} \quad (1.31)$$

где S - количество замененных слов, D - количество удаленных слов, I - количество вставленных слов и N - количество слов, подтверждающих истинность.

1.5 Оптимизация на основе градиентов

В предыдущих разделах мы обсуждали различные архитектуры нейронных сетей. Как правило, все эти архитектуры могут быть представлены в виде параметризованных функций f (x, θ), где через x мы обозначаем входные данные, а через θ - обучаемые параметры. Мы можем предположить, что изначально параметры θ инициализируются случайным образом; на самом деле это обычно используется для нейронных сетей (с определенными распределениями вероятностей, часто используемыми для ускорения обучения).

В следующем подразделе мы сначала введем функции потерь, которые используются для количественной оценки качества набора обучаемых параметров θ. Функция потерь минимизируется процессом оптимизации, который ищет соответствующие параметры θ. Затем мы кратко обсудим оптимизацию, процесс нахождения параметров θ, которые минимизируют функцию потерь L.

1.5.1 Функции потерь

В идеале мы хотели бы, чтобы наши системы машинного обучения работали как можно лучше с новыми данными в соответствии с некоторыми показателями производительности.

Однако у нас часто нет доступа к данным реального времени тестирования, и оптимизация системы на основе данных времени тестирования часто может привести к переоснащению.

Кроме того, наиболее часто встречающиеся показатели эффективности часто не поддаются дифференциации, что является крайне желательным свойством. По этим причинам мы часто оптимизируем систему машинного обучения в соответствии с функцией потерь, которая отличается, но связана с желаемой производительностью.

1.5.1.1 Кросс-энтропия

Здесь мы поместим себя в условия контролируемого обучения. У нас есть обучающий набор S примеров x, связанных с метками y, и мы хотим обучить классификатор предоставлять метки для новых немаркированных примеров x. Мы обучим классификатор максимизировать вероятность p (y | x), предсказанную классификатором (что эквивалентно минимизации отрицательной логарифмической вероятности). Обозначая функцию потерь через O (x, y), для единственного примера x с соответствующей меткой истинности y мы имеем:

$$O(x, y) = -\log(p(y|x)) \qquad (1.32)$$

Предполагая, что выборки данных (x, y) независимы, функция потерь по всему обучающему набору O(S) становится суммой функций потерь по каждой отдельной выборке (x, y):

$$O(S) = \sum_{(x,y)\in S} [-\log(p(y|x))] \qquad (1.33)$$

Эти потери могут быть сведены к минимуму с помощью оптимизации на основе градиентов.

1.5.1.2 Временная классификация коннекционистов (CTC)

В этом подразделе мы представим модель коннекционистской временной классификации (CTC), представленную Алексом Грейвсом в [23].

CTC может быть применен к задачам контролируемой классификации последовательностей, таким как распознавание рукописного ввода или распознавание речи. Это помогает представлять информацию инвариантным (или, по крайней мере, устойчивым) образом к последовательным искажениям. Это также интересно тем фактом, что для получения окончательной предсказанной последовательности меток не требуется никакой предварительной сегментации входных данных или какой-либо последующей обработки выходных меток.

1.5.2 Градиентный спуск

Градиентный спуск — это итерационный метод, который начинается с начального набора параметров theta, которые итеративно уточняются таким образом, чтобы функция потерь постепенно сводилась к минимуму. Вероятно, это наиболее широко используемый метод оптимизации моделей машинного обучения. Широко используемая метафора для описания процесса градиентного спуска — это метафора туриста с завязанными глазами, который пытается добраться до подножия холма. Основная интуиция, лежащая в основе

этого подхода (вместо, например использование случайного поиска) заключается в том, что может быть гораздо проще внести небольшое улучшение в функцию потерь, чем придумать оптимальные параметры theta за один шаг (как это было бы необходимо для случайного поиска).

1.5.2.1 Обратное распространение

В этом подразделе мы представим интуицию, лежащую в основе процедуры обратного распространения, которая является наиболее широко используемой процедурой для получения градиентов функций потерь с учетом обучаемых параметров нейронной сети. Мы сохраним описание простым. Более сложные выражения, требуемые реализованными нами архитектурами нейронных сетей, обрабатываются автоматическим дифференцированием, предоставляемым современным программным обеспечением, таким как TensorFlow или PyTorch.

Обратное распространение — это метод быстрого вычисления производных, который использовался в различных технических областях, помимо глубокого обучения. Обратное распространение получает градиенты аналитически. Обратное распространение вычисляет градиенты математических выражений путем рекурсивного применения правила цепочки.

Градиентный спуск (GD) с помощью обратного распространения происходит намного быстрее, чем при использовании конечных разностей. Обратное распространение может ускорить обучение современных нейронных сетей до 10 миллионов раз.

Обратное распространение и автоматическое дифференцирование наиболее естественно применяются для оптимизации функции путем разложения ее на модули, для которых можно легко получить локальные градиенты, а затем связать их в цепочку (используя правило цепочки). Автоматическое дифференцирование значительно упрощает задачу получения аналитических градиентов (по сравнению с получением их на бумаге).

1.5.2.2 Обратное распространение во времени (BPTT)

Обратное распространение во времени (BPTT) — это просто расширение обратного распространения, которое использует правило цепочки, для рекуррентных нейронных сетей (RNN), где одни и те же обучаемые параметры многократно появляются в нескольких выражениях.

Современное программное обеспечение для нейронных сетей, такое как TensorFlow или PyTorch, использует автоматическое дифференцирование, поэтому нам не нужно вручную выводить уравнения BPTT при реализации RNNS.

1.5.3 Современные алгоритмы и эвристики оптимизации

В этом подразделе мы опишем ADAM, современный алгоритм оптимизации для обучения систем глубокого обучения и отсечения градиентов, эвристику, часто используемую для уменьшения трудностей, связанных со взрывом градиентов при обучении рекуррентных нейронных сетей.

ADAM [24] — это алгоритм оптимизации, который был разработан для улучшения производительности стохастического градиентного спуска (SGD) для стохастических нестационарных функций потерь и разреженных градиентов.

Также утверждается, что ADAM допускает меньшую настройку гиперпараметров. При использовании более простых процедур оптимизации, таких как SGD, для разных уровней нейронной сети необходимо выбирать разные скорости обучения, что приводит к большему количеству гиперпараметров, которые необходимо установить; утверждается, что при использовании ADAM в этом нет необходимости. С другой стороны, алгоритм вводит несколько дополнительных гиперпараметров.

1.5.4 Полезные методы оптимизации/регуляризации

В этом подразделе мы описываем dropout, batch normalization and early validation, которые в основном полезны для целей регуляризации (для предотвращения переобучения). Также утверждается, что пакетная нормализация облегчает оптимизацию нейронных сетей.

1.5.4.1 Dropout

Dropout [25] — это метод регуляризации, который позволяет обучать гораздо большие сети без переобучения. Основная идея состоит в том, чтобы случайным образом отбрасывать нейроны из нейронной сети во время обучения с вероятностью p, выбранной из распределения Бернулли. Результатом этой процедуры является то, что это обучение приближается к обучению ансамбля гораздо более тонких нейронных сетей. Во время проверки или тестирования приближение использования ансамбля разреженных сетей может быть получено путем масштабирования активаций с той же вероятностью p.

1.5.4.2 Пакетная нормализация

Пакетная нормализация [26] — это метод регуляризации, который позволяет значительно ускорить обучение нейронных сетей, поскольку он может позволить алгоритму оптимизации работать даже с несколько более высокими скоростями обучения и сходиться за меньшее количество эпох. Пакетная нормализация решает так называемую проблему внутреннего ковариационного сдвига, которая может быть определена как изменение (сдвиг)

входного распределения при обучении классификатора. Глубокую нейронную сеть можно рассматривать как композицию сложенных слоев, каждый из которых принимает в качестве входных данных выходные данные предыдущих слоев. Во время обучения, когда мы изменяем параметры нижних уровней, изменяется распределение входных данных для более высоких уровней. Пакетная нормализация работает путем стандартизации входных данных между каждым слоем нейронной сети (для среднего значения 0 и дисперсии 1) во время обучения, а также добавления нескольких обучаемых параметров, чтобы исходные входные данные могли быть восстановлены, и обученная система не теряла никакой репрезентативной мощности. При этом он также улучшает поток градиентов через сеть, уменьшая зависимость градиентов от масштаба параметров или их начальных значений.

1.5.4.3 Early stopping

Early stopping — это метод регуляризации, который может повысить производительность систем машинного обучения за счет уменьшения переобучения. Для предоставленного набора данных обучающий набор используется для минимизации определенной функции потерь. Чтобы уменьшить вероятность переобучения, производительность системы на наборе проверки измеряется в разные моменты времени. Может случиться так, что производительность на обучающем наборе продолжает улучшаться, в то время как производительность набора проверки ухудшается / стагнирует (переобучение). При использовании ранней остановки состояние системы (определяемое значениями ее обучаемых параметров) возвращается к моменту времени, когда производительность проверки была самой высокой.

1.6 Заключение

В этой главе мы представили краткое введение в аспекты глубокого обучения, наиболее относящиеся к этому тезису. Сюда вошли наиболее популярные операции, которые используются в системах, основанных на архитектурах глубокого обучения, а также наиболее популярные архитектуры (многослойный персептрон, сверточная нейронная сеть и рекуррентная нейронная сеть). Мы также описали процедуры оптимизации, используемые для обучения систем глубокого обучения. Системы распознавания рукописного ввода, представленные в следующих главах, будут использовать представленные здесь архитектуры глубокого обучения, в частности рекуррентную и сверточную, а также описанные процедуры оптимизации.

2 Проблематика распознавания рукописного оффлайн текста

Мы живем в цифровом мире, где информация хранится, обрабатывается, индексируется и ищется компьютерными системами, что делает ее извлечение дешевой и быстрой задачей. Рукописные документы не являются исключением из этого правила. Возможности распознавания рукописных документов, и в частности рукописных текстов, разнообразны: от автоматической обработки чеков или почты до оцифровки архивов и понимания документов. Обращение с рукописными документами вызывает некоторые вопросы. Текст, который они содержат, не всегда доступен в форме, с которой можно было бы легко справиться с помощью компьютера. Вместо этого он должен быть извлечен из изображения, полученного в результате оцифровки документа. Таким образом, необходимо найти области изображения, содержащие рукописный текст, и преобразовать их в текст ASCII. Данный процесс известен как автономное распознавание рукописного ввода.

Исследования в этой области ведутся уже более шестидесяти лет. Начиная с отдельных символов и цифр, фокус переместился на распознавание слов. Распознавание слов, написанных курсивом, значительно сложнее, чем символов, главным образом по двум причинам. Во-первых, в языке гораздо больше разных слов, чем символов. Кроме того, сегментация рукописного словесного изображения на символы затруднена из-за скорописного характера текста, который вносит двусмысленности. В меньшей степени сегментация строки текста на слова также неоднозначна, и текущая стратегия заключается в непосредственном распознавании строк текста и использовании языковой модели для ограничения транскрипции и помощи в извлечении правильной последовательности слов. Постепенно системы распознавания эволюционируют до сквозного распознавания, которое обрабатывает целые документы напрямую, не предполагая, что доступна сегментация на текстовые строки.

В течение многих лет машинное обучение успешно применялось к структурированным данным в виде таблиц. Каждая запись таблицы соответствовала учебному примеру, а каждый столбец представлял собой числовой атрибут, с помощью которого можно было построить алгоритм в виде математической формулы. Он уже давно применяется для решения бизнес-задач, таких как кредитный скоринг или выявление мошенничества [27]. Однако алгоритмы машинного обучения не смогли извлечь уроки из неструктурированных данных, таких как изображения или текст. В последние годы ситуация кардинально изменилась с появлением Глубокого обучения. Глубокое обучение — это нейронные модели, в которых функция обучения определяется как состав большого числа более простых функций, называемых слоями [28]. Понимание состава функций в математическом смысле $(f \circ g)(x) = f(g(x))$. Эти модели глубокого обучения успешно используются в особенно сложных задачах машинного обучения, таких как классификация изображений [29], распознавание речи [30] или машинный перевод [31].

Большая проблема HTR — это проблема декодирования [32]. Изображение, содержащее последовательность штрихов, декодируется в последовательность символов, которые образуют эти штрихи. В этом смысле это похоже на задачу распознавания речи, когда аудио последовательность декодируется в последовательность слов этого аудио. Проблема машинного перевода, когда последовательность слов на одном языке декодируется в ее перевод на другой язык, также имеет сходство с проблемой HTR.

Имеется множество способов чтобы распознать текст, однако, вопросы, связанные с распознаванием рукописного текста, изучены не до конца. Впрочем, на данный момент существует некоторые способы. Во-первых, это распознавание с использованием форм, которые заполняют случайные люди от руки(в основном печатными буквами), используемые в разных областях. Во-вторых, это распознавание отдельно взятых букв от руки, с использованием особого пера на touch-screen экранах. Они имеют вполне высокую точность, не сильно отличающуюся от точности печати с клавиатуры. Понимание компьютером полноценных слитных слов или букв, то есть стандартного человеческого письма, на данный момент имеет мало разработок.

Как правило, текст делится на текст от руки или с помощью печатных устройств, рукописный текст можно разделить еще на два вида по методу извлечения из него информации для последующего распознавания. Это распознавание текста в процессе его написания «на лету» и распознавание текста, извлечённого из изображения. И в том и другом случаях могут применяться одинаковые методики, хотя и с разным успехом.

Если провести анализ способов написания рукописного и печатного текстов, то можно убедиться, что они сильно отличаются. Каждое слово рукописного текста представляет собой комбинацию линий, которые получаются при движении пишущего узла от момента начала его движения по бумаге до момента завершения. Информация, полученная в процессе этого движения: траектория движения, скорость, сила нажатия на пишущий узел (при наличии такой возможности) и т.д. – используется для распознавания написанного текста. Отрезки линий внутри траектории движения могут использоваться вместе с информацией об их длине, либо рассматриваться как отрезки фиксированного размера. Это зависит от подхода, который будет использоваться в дальнейшем для распознавания написанного.

Информация о характере движения пишущего узла позволяет значительно улучшить результаты распознавания, что происходит не только потому, что мы получаем информацию, которая является дополнительной к оптической информации. Улучшению способствует также факт неоднократного прохождения пишущего узла по одной и той же траектории.[33]

Проблемы распознавания рукописного ввода:

- Огромная вариативность и неоднозначность почерков у разных людей;
- Стиль почерка отдельно взятого человека также меняется время от времени и является непоследовательным;

- Низкое качество исходного документа / изображения из-за ухудшения с течением времени;
- Текст в печатных документах расположен по прямой, а людям не нужно писать строку текста по прямой линии на бумаге;
- Курсивный почерк затрудняет разделение и распознавание символов;
- Текст, написанный от руки, может иметь наклон вправо, когда в печатном тексте все пишется прямо;
- В случае распознавания почерка по фотографиям также следует учитывать неудобные углы. Угол, под которым сделана фотография, может скрыть букву, что затруднит идентификацию компьютером.
- Объединение букв в некоторых языках может создать их неправильную идентификацию. Например, в русском и казахском языках буквы “ш” и “и” в курсивном виде.
- Качественный набор данных рукописных текстов конкретного языка для изучения сложнее достать, чем печатные данные из книг.

3 Методы распознавания рукописного текста

Первоначальные подходы к решению проблемы распознавания рукописного текста включали методы машинного обучения, такие как скрытые Марковские модели(HMM), SVM. После предварительной обработки исходного текста выполняется извлечение признаков для идентификации ключевой информации, такой как петли, точки перегиба, соотношение сторон и т. д. Эти сгенерированные функции теперь подаются в классификатор HMM, чтобы получить результаты. Производительность моделей машинного обучения довольно ограничена из-за фазы ручного извлечения признаков и их ограниченной способности к обучению. Шаг извлечения признаков варьируется для каждого отдельного языка и, следовательно, не масштабируется. С появлением глубокого обучения произошло огромное улучшение точности распознавания рукописного текста[34].

Многомерные рекуррентные нейронные сети

RNN/LSTM может работать с последовательными данными для выявления временных паттернов и получения результатов. Но они ограничены работой с 1D-данными и, следовательно, не будут непосредственно применимы к данным с изображений. Поэтому следует применять многомерную структуру RNN/LSTM для повышения точности[35].

Согласно исследованию готового решения [36], такая сеть имеет процент ошибок в буквах -17.20% и процент ошибок в словах – 37.78%

Encoder-Decoder LSTM и Attention Networks

Модели Seq2Seq, работающие с сетью Encoder-Decoder, в последнее время стали популярны для решения задач распознавания речи, машинного перевода и т. д. и, таким образом, были расширены для распознавания рукописного текста путем развертывания дополнительного механизма внимания[37]. Готовое решение с использованием Seq2Seq достигло процент ошибок в буквах -12.62% и процент ошибок в словах – 26.65%[38].

Трансформер

Хотя сети Encoder-Decoder были довольно хороши в достижении результатов распознавания рукописного текста, они имеют узкое место в обучении из-за задействованных слоев LSTM и, следовательно, не могут быть распараллелены. В последнее время трансформеры оказались довольно успешными и заменили LSTM в решении различных языковых задач. По аналогии с рекуррентными нейронными сетями (RNN) трансформеры предназначены для обработки последовательностей, таких как текст на естественном языке, и решения таких задач как машинный перевод и автоматическое реферирование. В отличие от RNN, трансформеры не требуют обработки последовательностей по порядку. Например, если входные данные — это текст, то трансформеру не требуется обрабатывать конец текста после обработки его начала. Благодаря этому трансформеры распараллеливаются легче чем RNN и могут быть быстрее обучены. Многие крупные компании, такие как Google, Yandex используют эту

архитектуру для своих проектов. Трансформеры реализованы такими крупными фреймворками для глубокого обучения, как TensorFlow и PyTorch.

Генерация Рукописного Текста

Генерация рукописного текста - это задача создания реального рукописного текста, который может быть использован для расширения существующих наборов данных. Глубокое обучение требует большого количества данных для обучения, в то время как получение огромной базы изображений почерка для разных языков является громоздкой задачей. Чтобы решить эту проблему можно использовать Generative Adversarial Networks для генерации обучающих данных.

TensorFlow

TensorFlow - это комплексная платформа с открытым исходным кодом для машинного обучения. Он имеет всеобъемлющую гибкую экосистему инструментов, библиотек и ресурсов сообщества, которая позволяет исследователям продвигать новейшие достижения в области машинного обучения, а разработчикам легко создавать и развертывать приложения на основе машинного обучения.[39] TensorFlow принимает данные в виде многомерных массивов более высоких измерений, называемых тензорами. Многомерные массивы очень удобны при обработке больших объемов данных. TensorFlow является популярным выбором. Keras, который также когда-то был популярным выбором, теперь интегрирован с TensorFlow. Многие крупные фирмы используют его в своих проектах, а именно Google, Coca-Cola, Intel, Twitter. Также TensorFlow имеет свой Hub, который содержит обученные модели машинного обучения, готовые к тонкой настройке и развертываемых где угодно, что может быть полезным при дальнейшей работе. Модель HTR с использованием TensorFlow достигла 8.6% ошибок в своей работе.[40] Другая модель достигла 10% ошибок.[41] Просмотрев разные решения проблемы распознавания рукописного текста, я пришел к выводу, что почти все сейчас используют так или иначе библиотеку TensorFlow. Она имеет более низкий процент ошибок в сравнении с другими методами, а также набирает популярность и имеет огромный потенциал. Потому ее дальнейшее изучение является важным звеном для успешного написания диссертации.

4 Выбор датасета и сегментация текста

Для решения проблемы распознавания рукописного оффлайн текста было создано большое количество датасетов. Почти все из них находяся в свободном доступе. Самые крупные из них IAM [42] (свыше 100 тысяч изображений слов от 657 разных авторов), CVL [43] (7 текстов, в которых 83 тысячи словосочетаний от 310 человек). Однако у нас особый интерес к датасетам на русском языке. Для обучения ИНС моим научным руководителем был предоставлен датасет [44][45], разработанный им и его командой. Он содержит около 65 тысяч изображений рукописных предложений (слов и словосочетаний), написанных около 200 почерками. Именно он и выбран для дальнейшей работы.

4.1 Понятие сегментации

Сегментация текста — это процесс разделения письменного текста на значимые блоки, такие как слова, предложения или темы. Этот термин применяется как к умственным процессам, используемыми людьми при чтении текста, так и к искусственным процессам, реализованным в компьютерах, которые являются предметом обработки естественного языка. Хотя многие языки имеют явные маркеры границ слов, такие как пространство между словами, они присутствует не на всех письменных языках[46].

4.2 Сегментация строк

Извлечение текстовой строки — это первый шаг в любом процессе сегментации текста. Оно включает в себя извлечение текстовых строк из отсканированного изображения. Это один из самых сложных этапов в процессе распознавания текста, поскольку рукописный текстовый документ имеет несколько ориентаций; наложение символов, перекос и т. д.

Существуют множество методов сегментации строк, но рассмотрим самые точные.

1. Сегментация строк и слов рукописных документов Г.Лулудисом[47]. В нем предлагается метод как для извлечения текста, так и для извлечения слов. Извлечение текстовой строки выполняется с помощью преобразования Хафа. Этап постобработки выполняется для сегментации строк, в которых преобразование Хафа завершается неудачей. Сегментация слов выполняется путем выделения слов как межсловных, так и внутрисловных в зависимости от сравнения расстояний с пороговым значением. Он обеспечивает скорость обнаружения 90,4% и точность распознавания 90,6%.

2. Сегментация текстовых строк рукописного документа с использованием вырезания ограничительного шва Си Чжана[48]. Он предлагает резьбу по ограничительному шву, которая хорошо работает для линий с несколькими перекосами. Этот метод извлекает текстовые строки, ограничивая

энергию, которая передается вдоль связанного компонента тех же текстовых строк. Он достигает точности 98,4%. Он тестируется на изображении документа на греческом, английском и индийском языках.

3. Независимое от языка извлечение текстовой строки с помощью steam carving от Raid Saabni[49]. В нем предлагается алгоритм, основанный на подходе к вырезанию швов, который используется для изменения размера изображения содержимого. Результаты экспериментов с историческими документами на арабском, китайском и английском языках показывают, что при таком подходе удается разделить блоки текста с несколькими перекосами на строки с высокой вероятностью успеха в 98,6. %

4.3 Сегментация слов

Сегментация слов: после извлечения строк вторым шагом является сегментация отдельных слов из извлеченных текстовых строк. Эти извлеченные слова затем используются в дальнейшем для распознавания текста. Сегментация слов может быть выполнена двумя способами:

1. Аналитический подход: В этом процессе слово идентифицируется путем первого определения символов, составляющих слово.

2. Целостный подход: Этот подход рассматривает слово как единое целое и распознает его на основе его особенностей.

Существуют множество методов сегментации слов, но снова рассмотрим лишь самые точные.

1. Обучаемый алгоритм сегментации слов на основе правил Дэвида Д.Палмера[50]. Этот метод обеспечивает высокую сегментацию по китайскому языку. Это также работает для документов на тайском и английском языках. Он обеспечивает точность 84,9% для китайских документов.

2. Особенности локальной градиентной гистограммы для выделения слов в неограниченном рукописном документе Хосе А. Родригеса[51]. В нем предлагаются две различные системы определения слов - Скрытая Марковская модель и Динамическое искажение времени. При этом методе скользящее окно перемещается по изображению мира слева направо. Гистограмма ориентаций вычисляется в каждой ячейке путем разделения окна на ячейки в каждой позиции.

3. Распознавание рукописных слов с использованием классификатора на основе MLP - Целостный подход Анкуша Ачарьи[52]. В нем предлагается метод распознавания слов по их общей форме. В этом методе выбранные слова бинаризуются с использованием метода бинаризации. Целостные объекты извлекаются для выполнения иерархических разделов. Затем для распознавания изображений слов используется классификатор MLP.

5 Предлагаемая модель и итоги обучения

Модель фокусируется на символах кириллицы, извлеченных в рукописной форме.

Входные данные представляют собой обрезанное словесное изображение одномерной последовательности букв или символов. Она состоит из 4 слоев свёрточной нейронной сети после каждого из которых следует блок со слоем подвыборки (maxpooling). Декодер представляет собой две BI-LSTM слоя рекуррентной нейронной сети. Далее CTC получает выходную матрицу двунаправленной рекуррентной нейронной сети и основной текст истинности, и он вычисляет значение потерь. При выводе CTC получает только матрицу, и она декодирует ее в окончательный текст. Небольшое количество параметров (около 694 539), что обеспечивает высокую скорость распознавания, компактность и скорость, а также низкую частоту ошибок по сравнению с другими моделями. Модель предоставлена на рисунке 1.

Model: "htr_model"

Layer (type)	Output Shape	Param #	Connected to
image (InputLayer)	[(None, 350, 50, 1)]	0	
Conv1 (Conv2D)	(None, 350, 50, 32)	832	image[0][0]
pool1 (MaxPooling2D)	(None, 175, 25, 32)	0	Conv1[0][0]
Conv2 (Conv2D)	(None, 175, 25, 64)	18496	pool1[0][0]
pool2 (MaxPooling2D)	(None, 87, 12, 64)	0	Conv2[0][0]
Conv3 (Conv2D)	(None, 87, 12, 128)	73856	pool2[0][0]
pool3 (MaxPooling2D)	(None, 43, 6, 128)	0	Conv3[0][0]
Conv4 (Conv2D)	(None, 43, 6, 256)	131328	pool3[0][0]
reshape (Reshape)	(None, 43, 1536)	0	Conv4[0][0]
dense1 (Dense)	(None, 43, 64)	98368	reshape[0][0]
dropout (Dropout)	(None, 43, 64)	0	dense1[0][0]
bidirectional (Bidirectional)	(None, 43, 256)	197632	dropout[0][0]
bidirectional_1 (Bidirectional)	(None, 43, 128)	164352	bidirectional[0][0]
label (InputLayer)	[(None, None)]	0	
dense2 (Dense)	(None, 43, 75)	9675	bidirectional_1[0][0]
ctc_loss (CTCLayer)	(None, 43, 75)	0	label[0][0] dense2[0][0]

Total params: 694,539
Trainable params: 694,539
Non-trainable params: 0

Рисунок 1 – Модель нейронной сети

Алгоритм состоит из четырёх этапов:

1. Предобработка изображений (изменение размера, яркости, масштабирование)
2. Извлечение признаков с помощью слоев CNN
3. Обработка последовательности признаков для прогнозирования последовательности символов с помощью слоев BI-LSTM RNN
4. Подсчет потерь (loss) и декодирование в текстовый формат (CTC)

5.1 Свёрточные блоки

Кодировщик получает входные данные и генерирует векторы признаков. Эти векторы признаков содержат информацию и признаки, которые представляют входные данные. Сеть кодировщика состоит из 4 сверточных блоков, которые соответствуют обучению для извлечения соответствующих объектов из изображений. Каждый блок состоит из операции свертки, которая применяет ядро фильтра размером (5,5) в первом, (3,3) во втором и третьем блоках, (2,2) в четвертом блоке. Функция активации в каждом ReLU. Далее у нас идёт полносвязный слой (dense) с 64 нейронами и функцией активации ReLU. Чтобы уменьшить переобучение, я применил отсев (dropout) после Dense слоя (с вероятностью отсева, равной 0,2).

5.2 Декодер

Декодер представляет собой две BI-LSTM слоя рекуррентной нейронной сети, которые обрабатывают последовательности признаков для прогнозирования последовательностей символов. Вектор признаков содержит 256 признаков за временной шаг, и рекуррентная нейронная сеть распространяет информацию через эту последовательность. В двунаправленном режиме наши входные данные поступают в двух направлениях, что делает BI-LSTM отличным от обычного LSTM. С помощью обычного LSTM мы можем заставить входной поток течь в одном направлении, либо назад, либо вперед. Однако в двунаправленном режиме мы можем сделать входной поток в обоих направлениях, чтобы сохранить будущую и прошлую информацию.

5.3 Нейросетевая темпоральная классификация (Connectionist temporal classification – CTC)

Выходной уровень connectionist temporal classification (CTC) для рекуррентных нейронных сетей, предназначенный для задач маркировки последовательностей, когда нет выравнивания между входными данными и целевыми метками. Нейронным сетям нужны разные цели обучения для каждого раздела входной последовательности или временного шага. Это имеет два важных следствия. Во-первых, это гарантирует, что обучающие данные

должны быть предварительно сегментированы для постановки целей. Во-вторых, поскольку сеть генерирует только локальные классификации, глобальные аспекты последовательности (такие как вероятность последовательного появления двух меток) должны моделироваться извне. Действительно, окончательная последовательность меток не может быть надежно определена без какой-либо последующей обработки. Это достигается за счет того, что сеть позволяет прогнозировать метки в любое время во входной последовательности при условии, что общая последовательность меток верна. Это может устранить необходимость в предварительно сегментированных данных, поскольку больше не важно выравнивать метки с входными данными. CTC также предлагает полные вероятности последовательности меток напрямую, что гарантирует, что для использования сети в качестве классификатора времени не требуется дополнительной постобработки. Во время обучения нейронной сети CTC получает выходную матрицу RNN и основной текст истинности, и он вычисляет значение потерь. При выводе CTC получает только матрицу, и она декодирует ее в окончательный текст.

5.4 Данные

Рукописная база данных на казахском и русском языках может послужить основой для исследований по распознаванию рукописного текста. Она содержит русские слова (Районы, Города, Деревни, Населенные пункты, Районы, Улицы), написанные сотней разных авторов. Она также включает в себя самые популярные слова в Республике Казахстан. Вместе с базой данных было разработано несколько процедур предварительной обработки и сегментации. Он содержит бесплатные формы рукописного ввода в различных областях, представляющих интерес для писателей. Эта база данных подготовлена с целью предоставления обучающего и тестового набора для исследования распознавания казахских, русских слов. Эта база данных состоит из более чем 1400 заполненных форм. В нем содержится около 63000 предложений, более 715699 символов и примерно 106718 слов.

Основой для этого набора данных были отдельные слова (или короткие предложения), написанные на русском и казахском языках (примерно 95% русских и 5% казахских слов /предложений соответственно). Учитывая, что оба этих языка написаны кириллицей и имеют одни и те же 33 символа. Помимо этих символов, казахский алфавит также содержит 9 дополнительных специфических символов. Этот набор данных из отдельных слов / предложений был увеличен путем применения различных стилей почерка (примерно 50–100 разных людей) к каждому из этих отдельных слов. В результате этих процедур был получен окончательный набор данных с общим количеством слов/предложений, написанных от руки. После этого этот окончательный набор данных был разделен на три набора данных следующим образом: Обучение

(85%), Тестирование (10%) и Валидация (5%)[53]. На рисунке 2 показана частота встреч каждого символа в данном датасете.

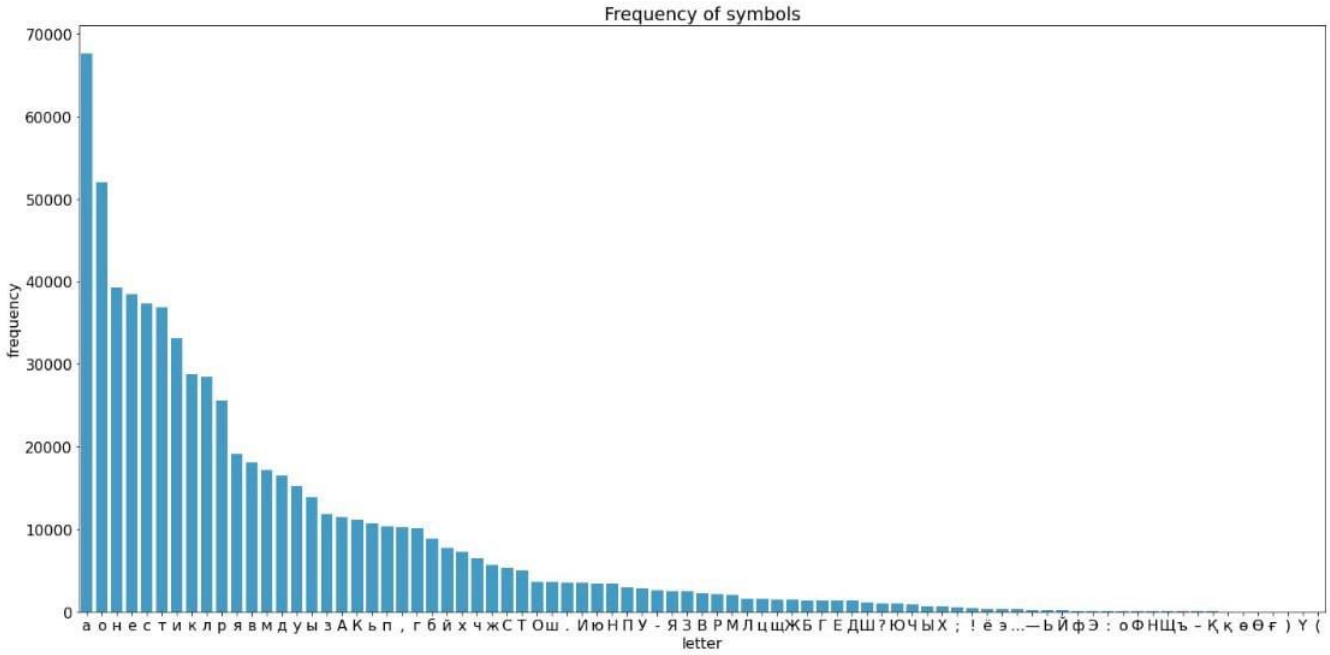

Рисунок 2 – Частота каждого символа в датасете

5.5 Обучение

Я обучил модель, чтобы минимизировать значение потери проверки функции CTC. Была применена ранняя остановка с терпением 10, чтобы отслеживать потери при проверке за каждую эпоху, и после того, как потери при проверке не улучшились после 10 эпох, обучение прекращается. На рисунке 3 изображена история обучения модели нейронной сети.

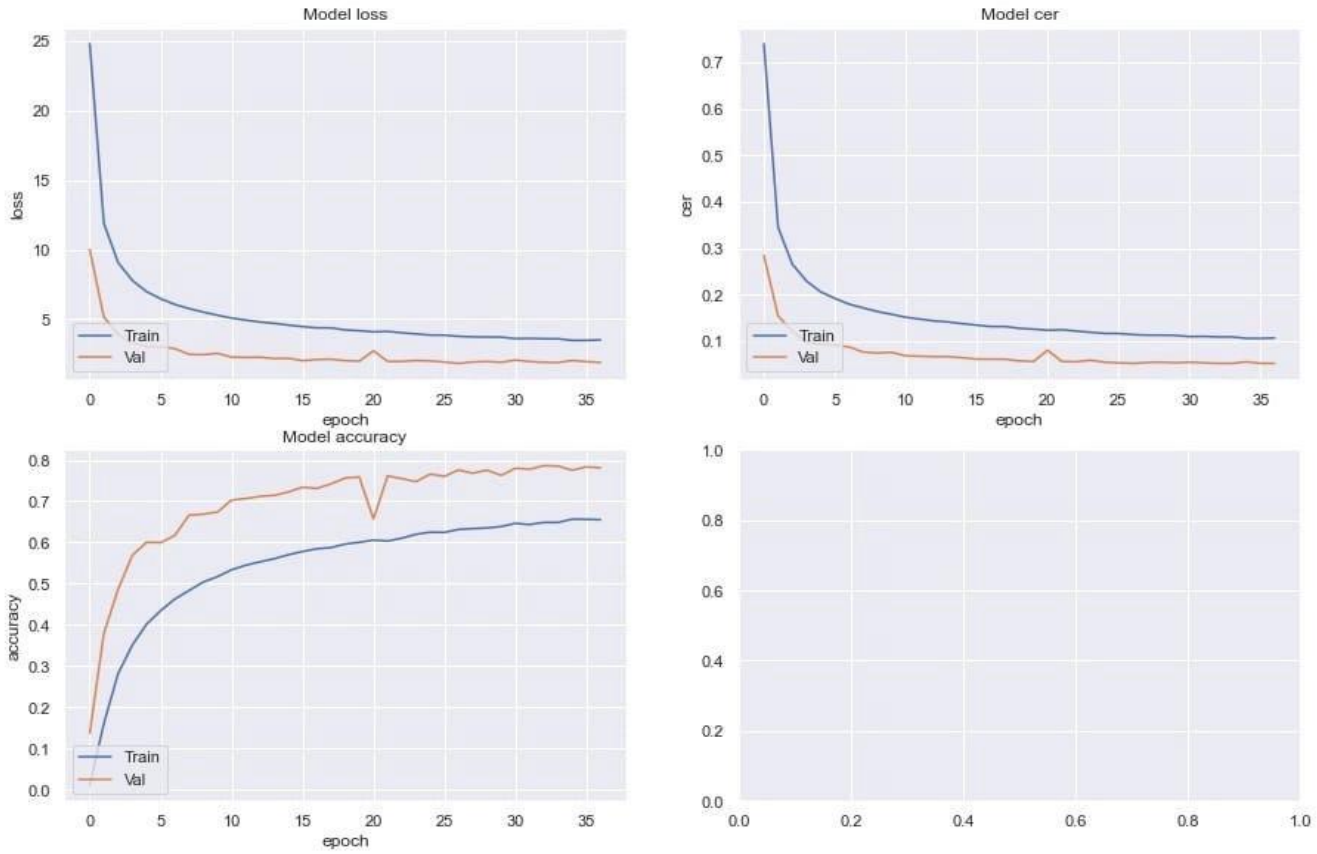

Рисунок 3 – История обучения

5.6 Итоги обучения

Данная программа имеет 77.95% точности на тестовых данных и 71.78% на валидационных данных, CER на тестовых равен 4.64%, на валидационных – 5.2%. На рисунках 4 и 5 изображены примеры удачных и неправильных распознаваний текста соответсвтенно.

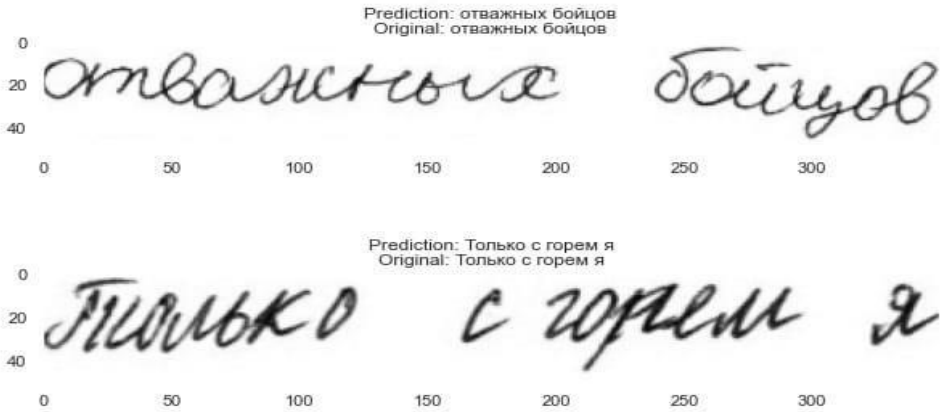

Рисунок 4 – Пример удачного распознавания

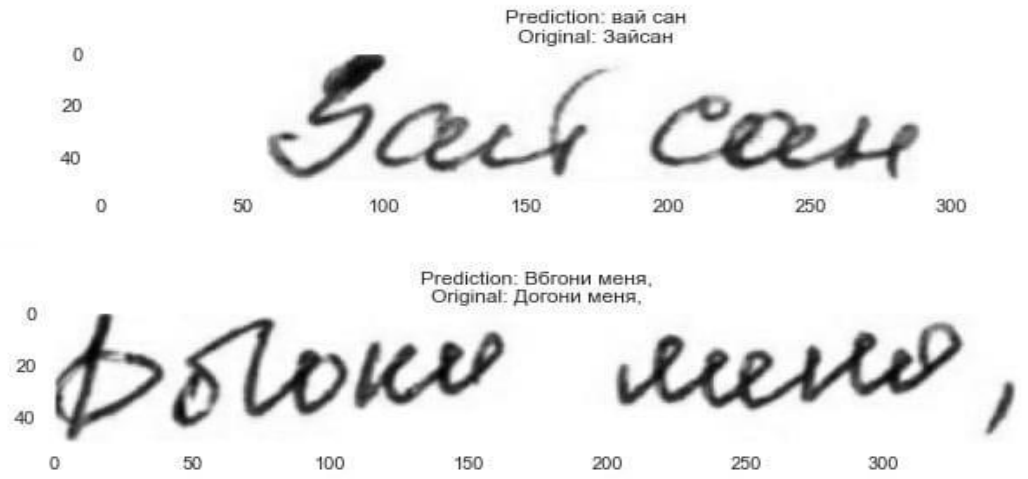

Рисунок 5 – Пример неправильного распознавания

Попробуем посмотреть, на каких буквах чаще всего ошибается модель. Для этого построим confusion matrix. Так как в общем случае трудно разобрать, где модель перепутала символ, а где распознала как два или вообще не распознала - выберем лишь те пары ответ/ожидание, которые одинаковы по длине.

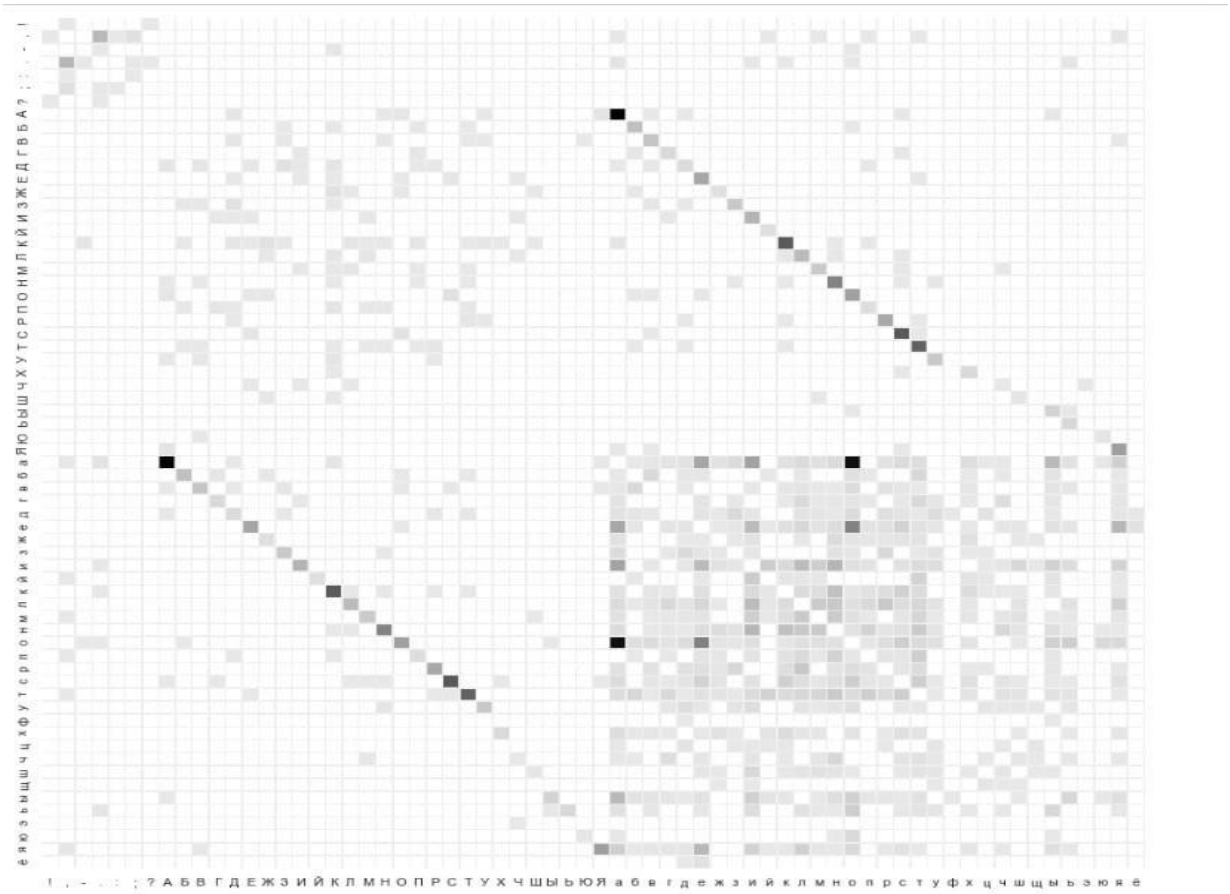

Рисунок 6 – Confusion matrix

Как видно из матрицы на рисунке 6, достаточно часто модель путает большие буквы с маленькими, что в целом не настолько важная задача, учитывая то, что сама буква в конечном счете определяется верно. А судя по выведенным массивам пар, это чаще происходит тогда, когда все слово состоит из больших букв. Можно также заметить частую путаницу между **о** и **а**, **о** и **е**, **л** и **н** и другими буквами, которые схожи по написанию. Это ожидаемый результат, т.к в некоторых записях это и человеку трудно отличить. В данном случае эту проблему сможет почти полностью решить постпроцессинг результата - поиск по словарю и/или оценка контекста.

5.7 Финальный тест

Для сегментации строк и слов в работе был реализован метод масштабного пространства, предложенного Р. Манматхой и Н. Шрималом. Несмотря на то, что статья датирована 1999 годом, метод по-прежнему дает хорошие результаты, быстр и прост в реализации[54].

На рисунках 7 изображён пример сегментации одной из строк на слова из случайно взятого текста, не входившего в датасет.

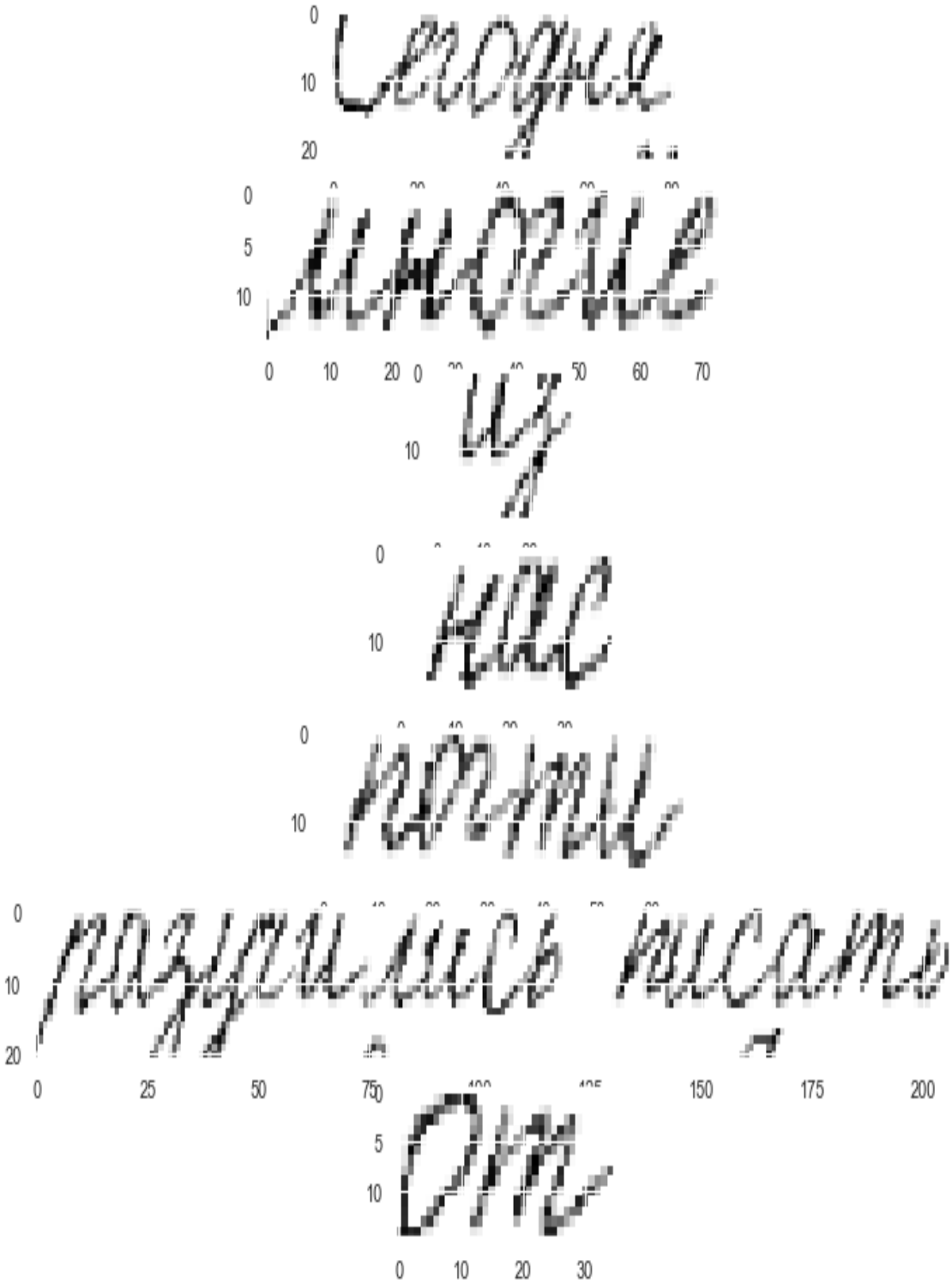

Рисунок 7 – Одна из сегментированных строк

На рисунке 8 изображен процесс распознавания текста с сегментированной строки из рисунка 7, а также его результат.

```
BBox(x=33, y=28, w=82, h=22)
precition: Иегодня
BBox(x=127, y=29, w=77, h=15)
precition: многие
BBox(x=206, y=30, w=30, h=20)
precition: из
BBox(x=241, y=28, w=39, h=17)
precition: нас
BBox(x=289, y=28, w=67, h=17)
precition: погти
BBox(x=363, y=29, w=212, h=21)
precition: пазучились писать
BBox(x=581, y=28, w=35, h=15)
precition: от
```

Рисунок 8 – Распознавание текста на сегментированной на слова строке

ЗАКЛЮЧЕНИЕ

Диссертация является научной квалификационной работой, в которой были рассмотрены модели и методы распознавания рукописного оффлайн текста, а также разобраны само понятие глубокого обучения. Основные научные результаты диссертации, практические выводы и рекомендации, полученные при выполнении исследований, заключаются в следующем:

1.Проведен анализ работ, связанных с распознаванием рукописного текста на основе глубокого обучения. Выявлены слабые стороны существующих решений, сформированы рекомендации;

2.Исследованы датасеты, содержащие в себе рукописные текста, выбран подходящий датасет для последующей работы.

3.Исследованы виды сегментации текста и строк, выбран подходящий метод для сегментирования.

4.Разработана модель нейронной сети, на основе CNN, BiRNN-LSTM, CTC с имплементированной сегментацией.

5. Разработанная модель нейронной сети была обучена на выбранном датасете. Выявлены проблемы, которые встречаются при обучении.

Поставленные задачи были выполнены: разработана программа для распознавания рукописного оффлайн текста, который сегментирует изображения на строки, их на слова и может определить их. Данная программа имеет 77.95% точности на тестовых данных и 71.78% на валидационных данных, CER на тестовых равен 4.64%, на валидационных – 5.2%.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

В процессе исследования была использована следующая литература:

[1] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical Evaluation of Rectified Activations in Convolutional Network,” 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

[2] ——, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” feb 2015. [Online]. Available: http://arxiv.org/abs/1502.01852

[3] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” The Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115–133, dec 1943. [Online].
Available: http://link.springer.com/10.1007/BF02478259

[4] “Biological neuron model.” [Online]. Available: https://en.wikipedia.org/wiki/Biological neuron model

[5] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville, “Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks,” jan 2017. [Online]. Available: http://arxiv.org/abs/1701.02720

[6] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional Sequence to Sequence Learning,” may 2017. [Online]. Available: http://arxiv.org/abs/1705.03122

[7] D. H. Hubel and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex.” *The Journal of physiology*, vol. 195, no. 1, pp. 215–43, mar 1968. [Online].
Available: http://www.ncbi.nlm.nih.gov/pubmed/4966457

[8] “CS231n Convolutional Neural Networks for Visual Recognition.” [Online]. Available: http://cs231n.github.io/convolutional-networks/

[9] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, “On the Expressive Power of Deep Neural Networks,” pp. 2847–2854. [Online]. Available: http://arxiv.org/abs/1606.05336

[10] Q. Nguyen and M. Hein, “Optimization Landscape and Expressivity of Deep CNNs,” oct 2017. [Online]. Available: http://arxiv.org/abs/1710.10928

[11] Y.-L. Boureau, J. Ponce, J. P. Fr, and Y. Lecun, “A Theoretical Analysis of Feature Pooling in Visual Recognition,” Tech. Rep., 2010. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/boureau-icml-10.pdf

[12] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for Simplicity: The All Convolutional Net,” dec 2014. [Online]. Available: http://arxiv.org/abs/1412.6806

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86,

no. 11, pp. 2278–2324, 1998. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=726791

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/ 4824-imagenet-classification-with-deep-convolutional-neural-networks

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," sep 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[16] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," feb 2016. [Online]. Available: http://arxiv.org/abs/1602.07261

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," dec 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[18] A. M. Schäfer and H. G. Zimmermann, "Recurrent Neural Networks Are Universal Approximators," in *Proceedings of the 16th international conference on Artificial Neural Networks - Volume Part I*. Springer-Verlag, 2006, pp. 632–640. [Online]. Available: http://link.springer.com/10.1007/11840817{ }66

[19] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," in *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. New York, New York, USA: ACM Press, 1992, pp. 440–449. [Online]. Available: http://portal.acm.org/citation.cfm?doid=130385.130432

[20] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, apr 1998. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218488598000094

[21] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, mar 1994. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18267787 http://ieeexplore.ieee.org/document/279181/

[22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735

[23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*. New York, New York, USA: ACM Press, 2006, pp. 369–376. [Online]. Available:

http://portal.acm.org/citation.cfm?doid=1143844.1143891

[24] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” dec 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[25] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[27] S. Wang. “A comprehensive survey of data mining-based accounting-fraud detection research”. In: 2010 International Conference on Intelligent Computation Technology and Automation, ICICTA 2010. Volume 1. 2010, pages 50–53. isbn: 9780769540771. doi: 10.1109/ICICTA.2010.831

[28] I. G. Courville, Y. Bengio, and Aaron. Deep learning. Volume 29. 7553. MIT Press, 2016, pages 1–73

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: Communications of the ACM. Volume 60. 6. 2017, pages 84–90. doi: 10.1145/3065386

[30] J. Chorowski et al. “Attention-based models for speech recognition”. In: Advances in Neural Information Processing Systems. Volume 2015-January. 2015, pages 577– 585. arXiv: 1506.07503

[31] D. Bahdanau, K. H. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015). arXiv: 1409. 0473

[32] A. Graves et al. “A novel connectionist system for unconstrained handwriting recognition”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 31.5 (2009), pages 855–868. issn: 01628828. doi: 10.1109/TPAMI.2008.137

[33] Львович И.Я., Львович Я.Е., Мозговой А.А., Чопоров О.Н. Снижение количества ошибок распознавания сканированных рукописных текстов // Цифровая обработка сигналов – 2016. -№4. – С. 43-47

[34] H. Bunke1, M. Roth1, E.G. Schukat-Talamazzini. Offline Cursive Handwriting Recognition using Hidden Markov Models.

[35] Thodore Bluche, Jrme Louradour, Ronaldo Messina. Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention.

[36] tuandoan998/Handwritten-Text-Recognition [Электронный ресурс]: URL: https://github.com/tuandoan998/Handwritten-Text-Recognition

[37] I. Sutskever, O. Vinyals, Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” CoRR, abs/1409.3215, 2014.

[38] giovanniguidi/Seq-2-Seq-OCR [Электронный ресурс]: URL: https://github.com/giovanniguidi/Seq-2-Seq-OCR

[39] TensorFlow [Электронный ресурс]: URL: https://www.tensorflow.org/

[40] sushant097/Handwritten-Line-Text-Recognition-using-Deep-Learning-with-Tensorflow [Электронный ресурс]:URL:
https://github.com/sushant097/Handwritten-Line-Text-Recognition-using-Deep-Learning-with-Tensorflow
[41] githubharald/SimpleHTR [Электронный ресурс]:URL:
https://github.com/githubharald/SimpleHTR
[42] IAM Handwriting Database [Электронный ресурс]: URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database
[43] CVL - Database [Электронный ресурс]: URL: https://cvl.tuwien.ac.at/research/cvl-databases/an-off-line-database-for-writer-retrieval-writer-identification-and-word-spotting/
[44] abdoelsayed2016/HKR_Dataset [Электронный ресрус]: URL: https://github.com/abdoelsayed2016/HKR_Dataset
[45] D. Nurseitov, K. Bostanbekov, D. Kurmankhojayev, A. Alimova, A. Abdallah, “HKR For Handwritten Kazakh & Russian Database,” arXiv preprint arXiv:2007.03579, 2020
[46] Text segmentation [Электронный ресурс]: URL: https://en.wikipedia.org/wiki/Text_segmentation
[47] G.Louloudes, “Line and word segmentation of handwritten documents “ELSEVIER.ltd,Volume 42, Issue 12, 2009.
[48] Xi Zhang “Text line segmentation using Constrains seam carving”, ICFHR, 2014.
[49] Raid Saabni, “Language-Independent text line extraction using seam carving”, ICDAR, 2011.
[50] David D.Palmer, “A trainable rule-based algorithm for word segmentation”, ACL’98 Proceedings of the 35th annual ruling of the association for computational Linguistics and eight conference, 1998.
[51] Jose A.Rodriguez, “Local gradient histogram features for word spotting in unconstrained handwritten documents”, ICFHR, 2008,
[52] Ankush Acharyya, “Handwritten word recognition using MLP based classifier” IJCSI, 2013.
[53] Abdallah A., Hamada M., Nurseitov D. “Attention-based Fully Gated CNN-BGRU for Russian Handwritten Text” //Journal of Imaging. – 2020. – Т. 6. – №. 12. – С. 141.
[54] R. Manmatha, Nitin Srimal, “Scale Space Technique for Word Segmentation in Handwritten Documents”, 1999.
[55] T. Wang, D. Wu, A. Coates, A. Ng. ”End-to-End Text Recognition with Convolutional Neural Networks” ICPR 2012
[56] T. S. Gunawan, A. F. R. M. Noor, and M. Kartiwi, “Development of english handwritten recognition using deep neural network,” 2018.
[57] S. Albawi, T. A. Mohammed, S. Al-Zawi, “Understanding of a convolutional neural network,” in 2017 International Conference on Engineering and Technology (ICET), 1–6, IEEE, 2017, doi:10.1109/ICEngTechnol.2017.8308186.

[58] Fischer, Andreas, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. “A fast matching algorithm for graph-based handwriting recognition.” In Graph-Based Representations in Pattern Recognition, pp. 194-203. Springer Berlin Heidelberg, 2013.

[59] Zamora-Martínez, Francisco, Volkmar Frinken, Salvador España-Boquera, Maria Jose Castro-Bleda, Andreas Fischer, and Horst Bunke. “Neural network language models for offline handwriting recognition.” Pattern Recognition 47, no. 4 (2014): 1642-1652.