

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF
KAZAKHSTAN

Satbayev University

Institute of Information and Telecommunication Technologies

UDC 378(063)

As a manuscript

Mahmoud Kasem

MASTER'S DISSERTATION

For an academic master's degree

Thesis title

KOHTD: Kazakh Offline Handwritten
Text Dataset

Direction of training

7M06102 Machine Learning & Data
Science

scientific adviser

Professor



Daniyar Nurseitov

"__" _____ 2022

Opponent

Professor



Irina Maratovna

"__" _____ 2022

Norm control

associate professor

A.T.Akhmediyarova

"__" _____ 2022

CLEARED FOR DEFENSE

Head of the Department of PI
cand. phys.-math. Sciences, Prof.
_____ A.N.Moldagulova

"__" _____ 2022

Almaty 2022

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF
KAZAKHSTAN

Satbayev University

Institute of Information and Telecommunication Technologies

Department of Software Engineering
Specialty: 7M06102 Machine Learning & Data Science

APPROVE

Head of the Department of PI
cand. Phys.-math. Sciences, Prof.

_____ A.N.Moldagulova
" ___ " _____ 2022

EXERCISE

For a master's thesis

Master student Mahmoud Kasem

Thesis topic: " KOHTD: Kazakh Offline Handwritten Text Dataset"

Deadline for the completion of the dissertation " ___ " _____

Initial data for the master's thesis Thesis titled "KOHTD: Kazakh Offline Handwritten Text Dataset" is talking about Kazakh handwritten text recognition research, a comprehensive dataset of Kazakh handwritten texts. This is particularly true given the lack of a dataset for handwritten Kazakh text. Proposed extensive Kazakh offline Handwritten Text dataset (KOHTD), which has 3000 handwritten exam papers and more than 140335 segmented images. It can serve researchers in the field of handwriting recognition tasks by using deep and machine learning used a variety of popular text recognition methods for word and line recognition including CTC based and attention-based methods Also, proposed a Genetic Algorithm (GA) for line and word segmentation based on random enumeration of a parameter.

Thesis is well-written. He worked on Kazakh Language and a quantitative comparison of well-known recurrent neural networks (RNN), such as Bluche, Puigcerver, Flor and Abdallah models, has been implemented to choose the best performing model on the dataset given and the accuracy of each model for established performance measurements.

The topic, subject, and objective of the research, as well as the tasks, order, and techniques of research, are all stated in Mahmoud Kasem's master's thesis, which also includes a conclusion and a number of documents.

Recommended Basic Reading: 1. Sozdikqor: Sh. shayahmetov atyndagy <<til-qazyna>> ulttyq gylymipraktikalyq ortalygy," <https://sozdikqor.kz>, 2021

SCHEDULE

Preparation of a master's thesis

Name of sections, list of issues under development	Deadlines for submission to the supervisor	Note
Section 1. Introduction	01/10/2021	pass
Section 2. Theoretical Background	15/10/2021	pass
Section 3. Data collection and storage	07/11/2021	pass
Section 4. Dataset segmentation	25/11/2021	pass
Section 5. Experimental results	09/12/2021	pass
Section 6. Conclusion and future work	18/12/2021	pass


Consultations on the project, indicating the sections of the project related to them

Section	Consultant, (according degree, title)	Timing	Signature
Norm control	A.T.Akhmediyarova, associate professor		
Software	N.K.Mukazhanov, Phd , associate professor		

Job issue date " ____ " _____ 2021 г.

Department head _____ A.N.Moldagulova

Scientific adviser _____ Daniyar Nurseitov

The task was  accepted for execution
by the undergraduate _____ Mahmoud Kasem

the date " ____ " _____ 2022 г.

ABSTRACT

Despite the transition to digital information exchange, many documents, such as invoices, taxes, memos and questionnaires, historical data, and answers to exam questions, still require handwritten inputs. In this regard, there is a need to implement Handwritten Text Recognition (HTR) which is an automatic way to decrypt records using a computer. Handwriting recognition is challenging because of the virtually infinite number of ways a person can write the same message. For this proposal we introduce Kazakh handwritten text recognition research, a comprehensive dataset of Kazakh handwritten texts is necessary. This is particularly true given the lack of a dataset for handwritten Kazakh text. In this paper, we proposed our extensive Kazakh offline Handwritten Text dataset (KOHTD), which has 3000 handwritten exam papers and more than 140335 segmented images and there are approximately 922010 symbols. It can serve researchers in the field of handwriting recognition tasks by using deep and machine learning. We used a variety of popular text recognition methods for word and line recognition in our studies, including CTC-based and attentionbased methods. The findings demonstrate KOHTD's diversity. Also, we proposed a Genetic Algorithm (GA) for line and word segmentation based on random enumeration of a parameter. The dataset and GA code are available at <https://github.com/abdoelsayed2016/KOHTD>.

АННОТАЦИЯ

Несмотря на переход к цифровому обмену информацией, многие документы, такие как счета-фактуры, налоги, служебные записки и анкеты, исторические данные и ответы на экзаменационные вопросы, по-прежнему требуют рукописного ввода. В связи с этим необходимо внедрить распознавание рукописного текста (НТР), которое представляет собой автоматический способ расшифровки записей с помощью компьютера. Распознавание почерка является сложной задачей из-за практически бесконечного количества способов, которыми человек может написать одно и то же сообщение. Для этого предложения мы вводим исследование по распознаванию казахского рукописного текста, необходим полный набор данных казахских рукописных текстов. Это особенно верно, учитывая отсутствие набора данных для рукописного текста на казахском языке. В этой статье мы предложили наш обширный казахский автономный набор данных рукописного текста (КОНТД), который содержит 3000 рукописных экзаменационных работ и более 140335 сегментированных изображений, а также около 922010 символов. Он может служить исследователям в области задач распознавания рукописного ввода с использованием глубокого и машинного обучения. В наших исследованиях мы использовали различные популярные методы распознавания текста для распознавания слов и строк, в том числе методы на основе СТС и на основе внимания. Полученные данные демонстрируют разнообразие КОНТД. Кроме того, мы предложили генетический алгоритм (ГА) для сегментации строк и слов на основе случайного перечисления параметра. Набор данных и код ГА доступны по адресу <https://github.com/abdoelsayed2016/KOHTD>.

АҢДАТПА

Ақпараттың цифрлық алмасуына көшкеніне қарамастан, шот-фактуралар, салықтар, меморандумдар мен сауалнамалар, тарихи деректер және емтихан сұрақтарына жауаптар сияқты көптеген құжаттар әлі де қолжазбаны қажет етеді. Осыған байланысты компьютерді пайдалана отырып жазбаларды транскрипциялаудың автоматты тәсілі болып табылатын қолжазбаны тануды (НТТ) енгізу қажет. Қолжазбаны тану адамның бір хабарламаны жаза алатын іс жүзінде шексіз санына байланысты қиын. Бұл ұсыныс үшін біз қазақ қолжазбасын тану зерттеуін енгіземіз, қазақ қолжазбасының толық деректер жинағы қажет. Бұл әсіресе қазақ қолжазбасына арналған деректер жинағының жоқтығына байланысты. Бұл мақалада біз 3 000 қолжазба емтиханнан және 140 335-тен астам сегменттелген кескіндерден, сондай-ақ шамамен 922 010 таңбадан тұратын қазақша офлайн қолжазба деректер жинағын (КОНТД) ұсындық. Ол терең және машиналық оқытуды қолдана отырып, қолжазбаны тану тапсырмалары саласындағы зерттеушілерге қызмет ете алады. Біздің зерттеуімізде біз сөз бен жолды тану үшін мәтінді тану үшін әртүрлі танымал әдістерді, соның ішінде СТС негізіндегі және назар аударуға негізделген әдістерді қолдандық. Алынған деректер КОНТД әртүрлілігін көрсетеді. Сонымен қатар, біз кездейсоқ параметрді санау негізінде жолды және сөзді сегменттеу үшін генетикалық алгоритмді (GA) ұсындық. GA деректер жинағы мен коды мына жерден қол жетімді <https://github.com/abdoelsayed2016/KOHTD>.

ACKNOWLEDGEMENTS

It's amazing how interconnected personal and professional lives are, and how what happens outside the workplace has an effect on what happens inside. So many people, directly or indirectly, helped me get through the last few years in a variety of ways, even without even realizing it. Thank you to those I've known for a long time and those I've met along the way, to those who believed in me and trusted me.

First and foremost, I want to express my gratitude to Dr. Mohamed Hamada for contacting me and persuading me to reconsider my position. He was instrumental in helping me find a job that I enjoy, in transforming me from an engineer to a researcher, and in instilling trust in my work. Prof. Daniyar Nurseitov, who agreed to supervise this study, for very insightful discussions, helpful advice, and his valuable experience and invaluable guidance, I would like to express my heartfelt gratitude.

Thanks to the National Open Research Laboratory for Information and Space Technologies for welcoming me and allowing me to complete my Master's degree in the best possible atmosphere. A student could not ask for a better environment in which to complete an industrial Master's degree, where they are encouraged to write, where they have a great deal of independence and autonomy in their work, and where they can pursue exciting research directions. I'd like to express my gratitude to everyone in the lab for their kindness and support.

Благодарности

Удивительно, насколько взаимосвязаны личная и профессиональная жизнь, и как то, что происходит за пределами рабочего места, влияет на то, что происходит внутри. Так много людей, прямо или косвенно, разными способами помогли мне пережить последние несколько лет, даже не осознавая этого. Спасибо тем, кого я знаю давно и тех, кого встретил на своем пути, тем, кто верил в меня и доверял мне.

Прежде всего, я хочу выразить благодарность доктору Мохамеду Хамаде за то, что он связался со мной и убедил меня пересмотреть свою позицию.

Он помог мне найти работу, которая мне нравится, превратил меня из инженера в исследователя и привил доверие к моей работе.

Профессору Данияру Нурсеитову, согласившемуся руководить этим исследованием, за очень содержательные обсуждения, полезные советы, а также его ценный опыт и бесценное руководство, я хотел бы выразить свою сердечную благодарность.

Спасибо Национальной открытой исследовательской лаборатории информационных и космических технологий за то, что приняли меня и позволили получить степень магистра в наилучшей атмосфере. Студент не мог бы мечтать о лучшей среде для получения степени промышленного магистра, где его поощряют писать, где он имеет большую независимость и автономию в своей работе и где он может заниматься интересными направлениями исследований.

Я хотел бы выразить благодарность всем в лаборатории за их доброту и поддержку.

АЛҒЫС

Жеке және кәсіби өмірдің бір-бірімен қаншалықты байланысты екендігі және жұмыс орнынан тыс оқиғалардың ішкі жағдайға қалай әсер ететіні таңқаларлық. Көптеген адамдар тікелей немесе жанама түрде, тіпті түсінбестен, соңғы бірнеше жылда маған әртүрлі жолдармен көмектесті. Көптен бергі таныстарыма, жолда кездескен жандарға, маған сенім артып, сенім артқан жандарға рахмет.

Ең алдымен, менімен байланысып, өз ұстанымымды қайта қарауға көндіргені үшін доктор Мохамед Хамадаға алғыс айтқым келеді.

Маған ұнайтын жұмысты табуға көмектесті, инженерден ғылыми қызметкерге дейін өсуіме көмектесті және жұмысыма сенімділік берді.

Осы зерттеуді жүргізуге келісім берген профессор Данияр Нұрсейітовке салиқалы пікірталастары, пайдалы кеңестері, құнды тәжірибесі мен баға жетпес бағыт-бағдары үшін шын жүректен алғысымды білдіремін.

Ақпараттық және ғарыштық технологиялар бойынша Ұлттық ашық зерттеу зертханасына мені қарсы алып, магистратураны ең жақсы жағдайда аяқтауға мүмкіндік бергені үшін рахмет. Студент өнеркәсіптік магистратураны аяқтау үшін жақсы ортаны сұрай алмады, олар жазуға ынталы, өз жұмысында үлкен дербестік пен дербестікке ие және олар қызықты зерттеу бағыттарын жүргізе алады.

Лабораториядағы барлық адамдарға мейірімділік пен қолдау көрсеткені үшін алғыс айтқым келеді.

CONTENT

Acknowledgements	7
List of Figures	Error! Bookmark not defined.
List of Tables	Error! Bookmark not defined.
1 Introduction	11
1.1 Related works	13
1.1.1 Datasets	13
1.1.2 Handwritten Deep Learning Models.....	16
1.2 Motivation	18
1.3 Report structure	19
2 Theoretical Background	20
2.1 Linear regression	20
2.2 Artificial Neural Networks.....	22
2.2.1 Feed-forward neural network.....	24
2.2.2 Convolutional Neural Networks	30
2.2.3 Convolution Layer — The Kernel	31
2.2.4 Pooling Layer	33
2.2.5 Classification — Fully Connected Layer (FC Layer).....	34
2.2.6 Recurrent neural network.....	35
2.2.7 Bidirectional recurrent neural network	36
2.2.8 Long short-term memory	37
2.3 Sequence to sequence learning.....	40
2.4 Batch normalization	41
2.5 Optimization.....	41
3 Data collection and storage	42
3.1 Labeling in the database.....	42
3.2 Characteristics of the Database	44
3.3 Statistic Analysis	45
4 Dataset segmentation	47
5 Experimental results	52
5.1 Evaluation methods	52
5.2 Training	52
5.3 Proposed Models	53
6 Conclusion and future work	55
6.1 Summary	55
6.2 future work	55
References	56

INTRODUCTION

Computer systems store, analyze, index, and search data in the digital era, enabling speedy and cost-effective retrieval. The legislation does not apply to handwritten materials. Recognizing handwritten documents, especially handwritten words, has a wide range of applications, from automated check or mail processing to archive digitization and document interpretation. Handwritten documents raise a slew of issues. They don't have the text in an understandable format for computers. Instead, it should be taken from the digitalized image of the document. As a result, the handwritten text in the photograph must be detected and converted into ASCII text. The term "offline handwriting recognition" relates to the procedure described above.

This area has been the subject of over sixty years of research. The focus shifted away from solitary characters and digits and toward word recognition. For two reasons, cursive terms are slightly more difficult to recognize than characters. To begin with, a language's vocabulary far exceeds its character set. Furthermore, segmenting a handwritten word image into characters is challenging because of the cursive style of the text, which adds ambiguity. The segmentation of a line of text into words is more ambiguous, thus the most recent strategy is to explicitly identify lines of text and apply a language model to constrain the transcription and aid in obtaining the right sequence of words. Recognition systems have progressed to end-to-end recognizers, which process entire documents without assuming that text line segmentation is available.

Deep learning has been widely used in several fields nowadays, such as Med-

ical applications like Cancers diagnoses, detection, and classification [1] and in Medical question answers [2], also deep learning has been used in software engineering such as optimizing the time and schedule of the software projects [3] and one of the most usages of Deep Learning is handwritten recognition for different languages as we will discuss.

The most significant developments in HTR for postal correspondence were investigated. They are primarily concerned with determining the area of interest, text segmentation, and the removal of background noises that obstruct text processing, such as lost or unclear fragments, spots on paper, and skew detection, as well as artificial intelligence training to recognize written text in the target language. The most often used recognition models, such as HMM, hybrid Markov models (Hybrid HMM), convolutional (CNN), and recurrent neural networks, are investigated in this context (RNN).

Deep Learning, specifically Deep Neural Networks (DNNs), has demonstrated excellent performance in a variety of areas, including object detection and classification in images with Convolutional Neural Networks (CNNs), speech recognition with Recurrent Neural Networks (RNNs), and named entity recognition with RNNs.

Nowadays, appreciating handwriting is a popular pastime. Solving this conundrum would assist a lot of firms. A postal service, for example, faces the tough challenge of processing a high quantity of shipments. Handwriting recognition (HWR) or Handwritten Text Recognition (HTR) refers to a computer's capacity to extract and interpret understandable handwriting data from a range of sources, such as paper documents, pictures, touchscreens, and other devices. Offline HTR is the process of converting letters or words into photos and subsequently into digital text. The input is a modifiable two-dimensional picture, and the output is a string of text. It has an outstanding human-machine interface and can automatically process handwritten papers. It also considers a sub-task of OCR, which focuses on extracting text from scanned documents and natural scene photos. Kazakh and Russian handwriting recognition has its own set of obstacles and rewards, and it has only lately been considered in comparison to other languages' text recognition.

For some types of images automatic recognition of handwritten texts still a challenging issue in spite of the recent improvements of the recognition methods and systems, in recent years, handwritten text recognition is attracting more researchers to work on it. A comprehensive and unbounded handwritten datasets are gaining more importance than before, handwritten text can be found: handwritten notes, memos, whiteboards, medical records, historical documents, stylus input text, etc. Therefore, support for understanding the handwritten text in images needs to be provided in a full OCR solution. For several languages and scripts, this highlights the need for research in the field of developing large-scale handwriting recognition systems. During the last thirty years researchers have been made different types of handwriting text recognition for many languages like English [4, 5, 6], Russian [7], Arabic[8, 9], Malayalam [10], Japanese [11], etc.

Any language has a huge number of words. For example, The Oxford dictionary for the English language contains more than 300,000 words. A dictionary for the Kazakh language has more than 380,000 words [12], so it seems impossible to collect a handwritten word database that includes all words. As far as we know, for the Kazakh language, there is not available public dataset.

In our research, we describe a large dataset, called Kazakh Offline Handwritten Text Dataset (KOHTD) to address challenging detection and recognition issues of handwritten Kazakh text in the scanned documents. We present a new

Kazakh database for offline handwriting recognition. The dataset is written in Cyrillic and shares the same 42 characters in Kazakh. This dataset is a collection of exam papers from students. There are approximately 922010 symbols in the KOHTD dataset and 140335 segmented images. KOHTD is suggested for many reasons. First, this dataset can serve researchers in the field of handwriting recognition issues by using deep and machine learning. Second, it's also a standard and pure dataset for evaluating and comparing different algorithm's performances. Third, there is no available dataset in Kazakh language.

Our database consists of a large collection of exam papers filled by students at Satbayev University and Al-Farabi Kazakh National University, this exam was

made and answered in the Kazakh Language (99%) and Russian Language (1%) as shown in Fig. 1.2, after we received this exam answer, we scanned it and make experiments that related to pre-processing of the examination lists to automatically identifying lists, evaluate the contours of lists, recovering rotations, and also segmentation by line and by words so we can apply our Deep Learning model to recognize each word and remove the artifacts in the edges at the boundaries of segmented words We have developed our intelligent software using state-of-the-art deep learning models to solve the problem of recognizing and processing natural language, which consists of optical character recognition of the manuscript texts in Kazakh and Russian languages.

The following section defines the related work on Handwriting Databases and Deep Learning models for handwritten. Section 4 presents the Data collection and storage phases as one of the most time consuming and costly stages. Section 5 provides Dataset segmentation. Section 6 provides Experiment Result on the KOHTD dataset and conclusion and future work are given in Sect. 6.1.

1.1 Related works

1.1.1 Datasets

IAM dataset [13, 14] is a handwritten sentence for the English language. The database can be used for handwritten recognition problems. This database is made on Lancaster-Oslo/Bergen (LOB) Corpus. The IAM Handwriting Database

2) Шартсыз оптималандыру есебі —
 $f(x)$ функциясының берілген тоңырақтық
 минимумы не максимумын іздейтін
 есебі.

Шартсыз оптималандыру есебінде
 $f(x)$ — ~~шартсыз~~ функция, максимат $f(x)$ деп
 — x айнымалылары басқарылатын айны-
 машылар деп, D — тарамды деп,
 не басқарылатын айнымалылардың
 D аймағында нәтиженің кез-келген
 y мәндер алымын оптималандыру
 есебінде тарамды шешімі деп
 аталады.

Шешуі: бастапқы нүктедегі $f(x)$ функциясы
 градиентті анықтаймыз, ол үшін
 біртүрлі дертес теңдеулерді мадамыз:

$$\frac{df}{dx_1} = -2x_1 + 6; \quad \frac{dF}{dx_2} = -8x_2 + 32$$

$$\nabla f(x) = \begin{pmatrix} -2x_1 + 6 \\ -8x_2 + 32 \end{pmatrix}, \quad \nabla F(x_0) = \begin{pmatrix} -2 \cdot 2 + 6 \\ -8 \cdot 4 + 32 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

Маңа x_1 нүктесін таңдаймыз:

$$x_1 = x_0 + \lambda, \quad \nabla F(x_0) = \begin{pmatrix} -2 \\ 0 \end{pmatrix} + \lambda \cdot \begin{pmatrix} -8 \\ 4 \end{pmatrix} = \begin{pmatrix} -2 - 8\lambda \\ 4\lambda \end{pmatrix}$$

Маңа нүктедегі градиентті

$$\nabla F(x_1) = \begin{pmatrix} -2 \cdot (-2 - 8\lambda) + 6 \\ -8 \cdot 4\lambda + 32 \end{pmatrix} = \begin{pmatrix} 16\lambda - 8 \\ -32\lambda + 32 \end{pmatrix}$$

$$\frac{B \nabla F}{d\lambda} = \nabla F(x_0) \cdot \nabla F(x_1) = \begin{pmatrix} -2 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 16\lambda - 8 \\ -32\lambda + 32 \end{pmatrix} = 0$$

теңдеуін шешіміз.



Figure 1.1. Some examples of images in our dataset

3.0 is made by 657 different writers and contains 1,539 scanned handwritten pages with 5,685 labeled sentences and 13,353 labeled text lines with a total of 115,320 labeled words, these database has been used in offline handwriting recognition

- Операциялар зерттеу бабына биз үйренген эңгетер.
- Шымык.
 - Масандык бајис.
 - Крестандык.
 - Ең кими кун.
 - Солтүстік-батыс.
 - Потенциалдар.
 - Үлестірімдік эңгетері.
- Мунда эсеп шотару кезиндегі келе алгоритмдер:
- ① Эсепті канондык түрге келтіру. (Адам урмат кимде тег келе кими келесе тег деген деп урмат бейнесу).
 - ② Шымыктык кесте куну.
 - ③ Сандарды орналасыру.
 - ④ Ең оңтүстік меминде табу.
 - ⑤ Мундаш пазантар F-мәке санды тег келтіру.
- Негизги принциптері:
- ① Модель куну.
 - ② Модель бойынша кезеңдерге бөлү.
 - ③ Ең оңтүстік меминде эскітейкері.
 - ④ Эсеп математикалык меминде табунада.
 - ⑤ Куну шотарунада.

Figure 1.2. Some examples of images in our dataset

[15, 16, 17], handwritten text segmentation [18, 19] and writer identification [20,21].

RAMIS [22] is a database of an industrial application. The main reason to develop the database was to collect handwritten samples similar to those sent by postal mail and fax by individuals to different companies. The database was filled by 1300 volunteers who contributed to the data collection, providing 5605 mails that contain 12,723 pages. Every mail contains two to three pages, including the

letter written by the volunteer, a form of the letter information, and an optimal fax sheet. Then the pages were scanned and the database was published to support testing of the tasks such as mail classification [23], handwritten recognition [24], and writer recognition[25].

The HKR [7] is a database of Russian and Kazakh texts that can be used to address detection and recognition problems, the database has 95% Russian and 5% Kazakh words/sentences, It's written in Cyrillic and share 33 characters and there are 9 additional characters for Kazakh alphabet. The dataset is consisting of more than 1,500 forms. The database contains about 63,000 sentences which are more than 715,699 symbols. The HKR database was written by approximately 200 different writers.

The IFN/ENIT [26] is a database of handwritten Arabic town/village names. The forms are filled by 411 writers with nearly 26400 names that contain more than 210000 characters. IFN/ENIT database contains 26459 handwritten Tunisian town/village names. The database is developed for training and determining handwritten Arabic word recognition systems.

KHATT [8] is a database of Arabic handwritten text, it can be used for Arabic offline handwritten text recognition. KHATT is consisting of 1000 handwritten forms that are written by 1000 different writers. These forms scanned at 200,300 and 600 dpi resolutions, the database contains 2000 randomly selected paragraphs which consists of 9327 lines. These forms were randomly divided into 70% for training, 15% for testing, and 15% for verification. The database was employed in text recognition, writer identification, and verification.

HIT-MW [27] is a database for Chinese handwriting text, it can be used for offline Chinese handwritten text recognition problems. The current version of the HIT-MW database contains 853 forms and 186,444 characters. The database is collected by postal mail and middleman not face to face, it can serve many applications concerning real handwriting recognition.

1.1.2 Handwritten Deep Learning Models

Approaches to handwritten text recognition can be classified into the following categories: Techniques based on HMMs and approaches based on RNNs. For cursive text recognition, HMM-based approaches have a number of advantages. HMM, models are resistant to noise and can tolerate variations in writing; there are automated algorithms for training the HMM models, and the HMM tools are freely available. Cursive text segmentation is error-prone and time-consuming, which is not required by HMM.

Bunke [15] proposes a system for offline recognition of unconstrained handwritten texts with a wide vocabulary. Only one assumption is made regarding the data: it is written in English. This enables us to apply Statistical Language Models to improve the performance of their system. Data from single and numerous writers have been used in several experiments. Lexica of various sizes (between 10,000 and 50,000 words) were utilized. The usage of language models

has been found to improve the system's accuracy. their strategy is detailed in-depth and compared to other ways for dealing with the same problem that has been given in the literature. It is suggested that an experimental configuration be used to correctly deal with unconstrained text recognition.

Safabakhsh [28] uses a continuous-density variable-duration hidden Markov model, CDVDHMM [29] to present a full method for recognizing Farsi Nastaaligh handwritten words. New techniques are used in the preprocessing step to locate and eliminate ascenders, descenders, dots, and other secondary strokes from the original image after binarization, noise reduction, and linked component specification. After that, a new segmentation method based on upper contour analysis and two other processes is used. This algorithm's major purpose is to avoid the problem of under segmentation. The over-segmentation problem can be solved by taking into account variable duration states in the system. The CDVDHMM models the sequence of obtained sub-characters by determining the right-to-left order. In the feature space, eight features are used to represent symbols, including three Fourier descriptors and five structural and discrete characteristics. This feature vector is size and shifts insensitive. Pure characters (without secondary strokes) and some compound forms of characters in the Nastaaligh handwriting style are considered in the model.

AlKhateeb [30] Using Hidden Markov Models (HMMs), a word-based offline recognition system is proposed. Preprocessing, feature extraction, and classification are the three stages of the approach. The first step is to segment and normalize the words from the input scripts. Then, using a sliding window moving across each mirrored word image, a set of intensity features is collected from each of the split words. Meanwhile, structure-like information such as the number of subwords and diacritical marks are retrieved. Finally, these characteristics are merged into a classification scheme. Intensity features are utilized to train an HMM classifier, and the results are then re-ranked utilizing structure-like features for a higher recognition rate. Extensive trials were conducted utilizing the IFN/ENIT database, which comprises 32,492 handwritten Arabic words.

Otherwise, RNNs, such as the gated recurrent unit (GRU) [31] and the long short-term memory (LSTM) [32] can fix this problem. Speech recognition [33], machine translation [34], video summarising [35], and others. sequence-to-sequence learning tasks have demonstrated RNN models' amazing skills. It is necessary to convert a two-dimensional image to a vector and send it to an encoder and decoder in order to transform it for offline HTR.

GRU, and LSTM handle the problem by combining information and features from many sources. RNN networks are fed these handwriting sequences. The input feature does not require segmentation due to the usage of Connectionist

Temporal Classification (CTC) [36] models. One of the main advantages of the CTC algorithm is that it does not require any segmented labeled data. We can employ data alignment with the output.

RR Ingle [37] focuses on three issues that arise while creating such systems: data, efficiency, and integration. For starters, acquiring large amounts of high-quality training data is one of the most difficult tasks. They solve the challenge

by analyzing online handwriting data gathered for a large-scale online handwriting recognition system. They present our picture data generating pipeline and investigate how online data may be used to construct HTR models. They show that when only a few real photos are available, as is frequently the case with HTR models, the data improves the models dramatically. It allows supporting a new script for a much-reduced price. Second, they propose a neural network-based line recognition model without recurrent connections. The model reaches a level of accuracy comparable to LSTM-based models while allowing for more simultaneous training and inference. Finally, they show how to integrate HTR models into an OCR system in a straightforward manner. These components make up a solution for integrating HTR into a large-scale OCR system.

Espana-Boquera [38] proposes hybrid Hidden Markov Model (HMM) and Artificial Neural Network (ANN) models for identifying unconstrained offline handwritten texts. Markov chains were employed to describe the structural elements of the optical models, and a Multilayer Perceptron was used to estimate the emission probability. With supervised learning approaches, this work also introduces novel strategies for removing slope and slant from handwritten text and normalizing the size of text images. Slope correction and size normalization are performed by using Multilayer Perceptrons to classify the local extrema of text contours. Artificial Neural Networks are also used to reduce slant in a nonuniform manner. Experiments were conducted using offline handwritten text lines from the IAM database, and the recognition rates attained were among the best for the identical job when compared to those published in the literature.

F Abdurahman [39] proposes an offline handwritten Amharic (the language of the Federal Government of Ethiopia) word recognition system based on convolutional recurrent neural networks. Convolutional neural networks (CNNs) for feature extraction from input word images, recurrent neural networks (RNNs) for sequence encoding, and connectionist temporal classification as a loss function are all part of the proposed system. they have created a dataset of handwritten Amharic words, HARD-I. their best-performing recognition model achieved a WER of 5.24 percent and a CER of 1.15 percent from testing on various recognition models utilizing their dataset. When compared to existing models for offline handwritten Amharic word recognition, the proposed models perform well.

1.2 Motivation

The absence of handwritten datasets for Kazakh-Russian language, as well as public datasets in Kazakh-Russian language, is the topic addressed in this research. Kazakh languages are highly tough and annoying to recognize when it comes to text recognition since writers can write the character touch together, making character segmentation impossible. When it comes to text recognition, Kazakh-Russian languages are extremely difficult and challenging because writers can write the character contact together, making character segmentation impossible.

1.3 Report structure

The report is structured as following:

- Introduction: Introduction to and motivation of the problem, previous related and useful work, and the goal of project.
- Data collection and storage : explain how the dataset collected, stored and annotated.
- Dataset segmentation : Explain how the segmentation process done by line (Line segmentation) and word (Word segmentation).
- Results and discussion: Results from the performed experiments including training stability, performance, and comparison of models are shown and discussed.
- Conclusion: The conclusions that can be drawn from the results are listed.

Chapter 2. Theoretical Background

2.1 Linear regression

The linear model is one of the most fundamental statistical models, and it is used in a variety of fields including statistics and machine learning, thus its relevance.

Assuming a linear relationship between a set of input vectors x_1, x_2, \dots, x_N with $x_i \in \mathbb{R}^p$ and output vectors y_1, y_2, \dots, y_N with $y_i \in \mathbb{R}$ the linear model is defined as

$$y_i = \sum_{j=1}^p w_j x_{i,j} + b + \varepsilon_i \quad \text{for } i = 1, \dots, N \quad (2.1)$$

with w_1, w_2, \dots, w_p being a weight vector of size $w_j \in \mathbb{R}$ for each dimension

p , and $b \in \mathbb{R}$ being the bias i.e. the “learned” parameters of the model, and

$\varepsilon_i \sim N(0, \sigma_\varepsilon^2) @ \mathbb{R}$ being the noise in the data which is assumed to follow a Gaussian distribution with zero mean and a standard deviation of σ_ε^2 [10]. If this assumption does not hold, the linear model should not be used.

Letting $w_0 = 1$ and $\tilde{x} = \begin{bmatrix} 1 \\ x_i \end{bmatrix}$ \forall_i the bias is incorporated in the weight x_i matrices and the model definition reduces to

$$y_i = \sum_{j=0}^p w_j \tilde{x}_{i,j} + \varepsilon_i \quad \text{for } i = 1, \dots, N \quad (2.2)$$

By denoting

$$X = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_N] \in \mathbb{R}^{(p+1) \times N} \quad (2.3)$$

$$Y = [y_1, y_2, \dots, y_N] \in \mathbb{R}^{M \times N} \quad (2.4)$$

$$W = [w_0, w_1, \dots, w_p] \in \mathbb{R}^{M \times (p+1)} \quad (2.5)$$

$$E = [\varepsilon_0, \varepsilon_1, \dots, \varepsilon_N] \in \mathbb{R}^{M \times N} \quad (2.6)$$

the linear model can be expressed in the following compact matrix notation

$$Y = WX + E \quad (2.7)$$

Determining the values of the weight matrix W can be done in multiple ways, but the most common way is to minimize the “Residual Sum-of-Squares” (RSS), which is given as

$$RSS(W) = \sum_{i=0}^N \sum_{j=0}^M (y_{i,j} - \sum_{k=0}^p w_{k,j} \tilde{x}_{i,j})^2 \quad (2.8)$$

$$= \sum_{i=0}^N \sum_{j=0}^M (y_i - W \tilde{x}_i)_j^2 \quad (2.9)$$

$$= \sum_{i=0}^N ((Y - WX)^T (Y - WX) \mathbf{1})_i \quad (2.10)$$

$$= \mathbf{1}^T (Y - WX)^T (Y - WX) \mathbf{1} \quad (2.11)$$

Since the expression is quadratic in the parameters a unique solution can be found analytically. The RSS-estimate of the weight matrix is therefore given by

$$\tilde{W}_{RSS} = \underset{W}{\operatorname{argmin}}(RSS(W)) \quad (2.12)$$

which can be found by solving

$$\frac{\partial RSS(W)}{\partial W} = 0 \quad (2.13)$$

which leads to the closed form solution

$$\tilde{W}_{RSS} = YX^T(XX^T)^{-1} \quad (2.14)$$

2.2 Artificial Neural Networks

Artificial neural networks (ANN), also known as "Neural networks," are a type of pattern recognition model inspired by the human brain that have dominated machine learning research and earned their own branch of machine learning called "Deep Learning" due to their extensive application. A neural network is made up of connected nodes that resemble brain neurons and have connections that resemble axons. Like a real neuron, the node in the neural network receives an input signal from one or more axons and generates a particular activation signal. From here on, the nodes in an ANN will be referred to as "neurons" throughout the text. Given a set of input signals x_1, x_2, \dots, x_N and a resulting output activation signal a the neuron can be visualized as in Figure 2.1.

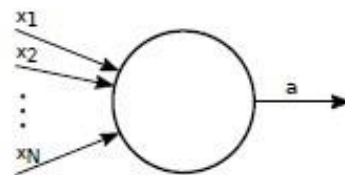


Figure 2.1. Visualization of a single linear neural network neuron with N input connections where the activation a will be a weighted sum of the inputs

$$a = \sum_{i=1}^N x_i w_i \quad (2.15)$$

i.e. each neuron will act as a linear regression model presented in Section 2.1, which then will act as the input to another neuron.

Because stacking multiple linear regression models will result in linear regression, the activation is usually transformed using a nonlinear function known as a "activation function," denoted $\sigma(\cdot)$, i.e. the new output of each neuron will be .

$$z = \sigma(a) = \sigma\left(\sum_{i=1}^N x_i w_i\right) \quad (2.16)$$

Common choices of activation functions are so-called "sigmoidal" functions which include (but is not limited to) the sigmoid function and the hyperbolic tangent

$$\text{Sigmoid} : \sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.17)$$

$$\text{Hyperbolic tangent} : \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (2.18)$$

which both have the sigmoidal form seen in Figure 2.2 but differs in the output range where the sigmoid function maps an input to the range $[0, 1]$, and corresponds to each neuron being a logistic regression model, and the hyperbolic tangent maps to the range $[-1, 1]$.

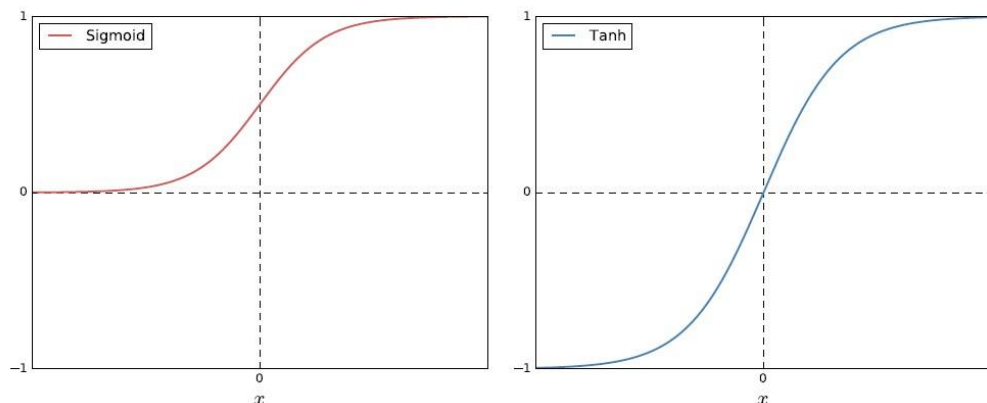


Figure 2.2. Sigmoid vs. hyperbolic tangent

From some algebraic manipulation it is found that the two functions have the following relation

$$\tanh(x) = 2\sigma(2x) - 1 \quad (2.19)$$

Which makes it difficult to argue choosing one over the other as activation function, since the model should be able to learn the same patterns using any of the two. Other activation functions includes "Rectified linear unit" (ReLU) given as

$$\text{ReLU}(x) = \max(0, x) \quad (2.20)$$

The sign (\cdot) function, the "leaky rectified linear unit" (Leaky ReLU), and many more. The essential requirement for activation functions is that they must be differentiable, allowing for the computation of their gradient with respect to a

given set of weights. The tanh, sigmoid, and ReLU functions are all employed in this project for distinct objectives. This will be discussed in greater detail later. A neuron with a "sigmoidal" activation function will now be represented in Figure 2.3 to differentiate the types of neurons employed in a network.

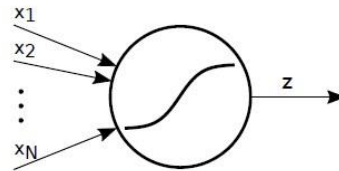


Figure 2.3. Visualization of a single neural network neuron with "sigmoidal" activation function

These neurons can be aligned, connected, and stacked in different ways constructing different type of networks often referred to as "network architectures".

The following sections will explain the following network types

- Feed-forward neural network
- Convolutional neural network
- Recurrent neural network
- Bidirectional recurrent neural network

2.2.1 Feed-forward neural network

The Feed-Forward Neural Network (FFNN) is one of the simplest types of neural networks where, as the name indicates, information is only fed forward in the network. FFNN's consists of an "input layer", and one or more so-called "hidden layers" with the last hidden layer being the "output layer". Despite the simplicity of the FFNN it is a very powerful model. This is emphasized in [14] where it is shown that a single layer FFNN is capable of approximating any function. In

Figure 2.4 a 3-layer FFNN is shown with an input of size 3 and an output of size 2.

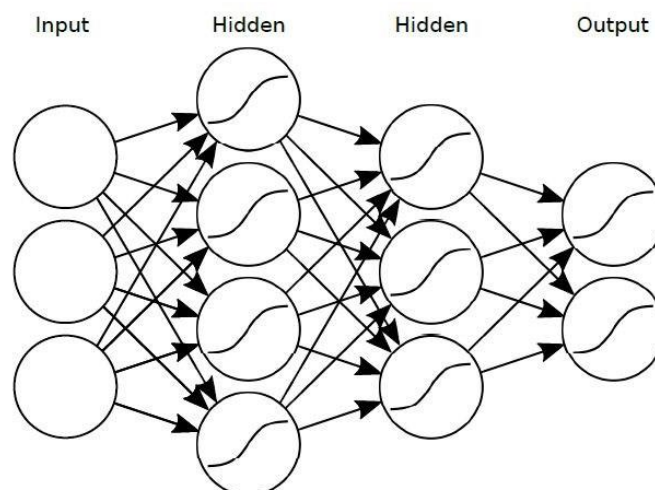


Figure 2.4. Visualization of a 3-layer feed-forward neural network with input

$x \in \mathbb{R}^3$ and output $y \in \mathbb{R}^2$

Denoting L the number of hidden layers and $N^{(l)}$ the number of neurons in layer l , with $l = 0$ being the input layer and $l = L$ being the output layer, and denoting the weight matrix between layer $l-1$ and layer l as $W^{(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l-1)}}$ constructed by concatenating the weight-vectors associated with each neuron in

layer $l-1$, i.e. $W^{(l)} = [w_1^l w_2^l \dots w_{N^{(l-1)}}^l]$ one can compute the activation of neuron

k in layer l like in (2.16) using

$$z_k^{(l)} = \sigma(a_k^{(l)}) = \sigma\left(\sum_{k'=1}^{N^{(l-1)}} z_{k'}^{(l-1)} w_{k,k'}^{(l)}\right) \quad (2.21)$$

with $z^{(0)} = x \in \mathbb{R}^p$ being the input vector, and thereby $z^{(L)} = \hat{y} \in \mathbb{R}^{N^{(L)}}$ the network output with the number of output neurons $N^{(L)}$ being the dimensionality of the output vector \hat{y} . For a classification problem with K classes the number of output neurons $N^{(L)}$ is usually $N^{(L)} = K$ where each neuron k represents the probability of class C_k denoted $p(C_k | x)$ computed using the “softmax” function given as

$$p(C_k | x) = z_k^{(L)} = \frac{e^{a_k^{(L)}}}{\sum_{k'=1}^{N^{(L)}} e^{a_{k'}^{(L)}}} \quad (2.22)$$

which acts as generalization of the sigmoid function for multidimensional input, and due to its normalization factor can be directly treated as a probability, since

$$p(C_k | x) \in [0, 1] \forall k \quad \text{and} \quad \sum_{k=1}^{N^{(L)}} p(C_k | x) = 1 \quad (2.23)$$

During training, the “1-of-K” coding method may be used to represent the target vector y for each category k , with all elements of y being 0 except element k , which is a 1. The number of layers that are hidden L denotes the network’s “depth,” with “deep” networks able to learn abstract feature combinations from the data owing to its complexity, but “shallow” networks may not. However, as the network’s depth grows, the danger of “overfitting” the training data, in which the network begins to represent the data’s noise, grows. The amount of training data and the difficulty of the problem to be solved are generally tradeoffs.

Denoting $W^{(l)} = \{w^1 w^2 \dots w^L\}$ as the set of network parameters, the network function f computing the output for a given input is uniquely defined as

$$z^{(L)} = \hat{y} = f(x | W) \quad (2.24)$$

i.e. the weights define the network. In order to find the optimal weights for a given network and a given set of training data, one must define a loss function for the network for which the weights should be optimized w.r.t.

Loss function

Given a classification problem like in (2.21) with K classes we want to find the weights which maximizes the conditional probability $p(C_k | x)$ for the correct classes k , i.e. given a dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ we want to maximize the probability assigned to the dataset by the network

$$W_{ML} = \underset{W}{\operatorname{argmin}} \prod_{i=1}^N p(y_i | x_i, W) \quad (2.25)$$

which leads to a maximum-likelihood estimate of the optimal weights W_{ML} [8].

One can then define the loss function $L(D)$ as the probability in (2.25), or more commonly as the negative natural logarithm of the probability for the given dataset D as

$$L(D) = -\ln \prod_{i=1}^N p(y_i | x_i, W) \quad (2.26)$$

which then must be minimized instead of maximized like in (2.25), hence also more suitable for a “loss” function.

Using the rule $\ln(a \cdot b) = \ln(a) + \ln(b)$ it is seen that

$$L(D) = -\sum_{i=1}^N \ln p(y_i | x_i, W) \quad (2.27)$$

i.e. we can define an “example loss” $L(x, y)$ for each example $(x, y) \in D$ as

$$L(x, y) = -\ln p(y | x, W) \quad (2.28)$$

and thereby we have

$$L(D) = \sum_{i=1}^N L(x_i, y_i) \quad (2.29)$$

which we can optimize w.r.t. the network weights in order to obtain an estimate of the optimal weights W_{ML} for a given training dataset D [8]. For the given classification problem with K classes we define the per-example network probability $p(y | x, W)$ from the conditional probability given in (2.22)

$$p(y | x, W) = \prod_{k=1}^K p(C_k | x)^{y_k} \quad (2.30)$$

with y being the “1-of- K ” encoded class vector. By substituting (2.30) into (2.28) we get

$$L(x, y) = -\ln \prod_{k=1}^K p(C_k | x)^{y_k} \quad (2.31)$$

$$= -\sum_{k=1}^K y_k \ln p(C_k | x) \quad (2.32)$$

$$= -\sum_{k=1}^K y_k \ln z_k^{(L)} \quad (2.33)$$

It’s known as the ”cross-entropy” loss function. Because both the loss function and the network are made up of differentiable operators, any gradient-based optimization strategy may be used to train the network using the loss function ”cross-entropy.” This means that in order to lower the size of the loss function, and thereby find the optimal weights W_{ML} , the gradient of the loss function w.r.t. the weights W , i.e.

$$\frac{\partial L(D)}{\partial W} = \sum_{i=1}^N \frac{\partial L(x_i, y_i)}{\partial W} \quad (2.34)$$

needs to be computed. This can be done using the “Backpropagation” method which will be explained in Section 2.2.1.2.

Backpropagation

The backpropagation algorithm is an algorithm for efficiently computing the gradient of some given loss function w.r.t. the weights of a FFNN. I.e. given the weights of each layer of a given network

$$W^{(l)} = \{w^{(1)}, w^{(2)}, \dots, w^{(L)}\} \quad (2.35)$$

the backpropagation algorithm computes

$$\frac{\partial L(x, y)}{\partial W} = \left\{ \frac{\partial L(x, y)}{\partial W^1}, \frac{\partial L(x, y)}{\partial W^2}, \dots, \frac{\partial L(x, y)}{\partial W^L} \right\} \quad (2.36)$$

Given a training example $(x, y) \in D$ from the dataset D the first step of the algorithm is to pass the input vector x through the network, and thereby compute the networks output vector \hat{y} , called the “Forward pass”, with the current weight values of the network.

Second step is the so-called “Backward pass” which involves propagating errors back through the network.

Starting from the output layer one can compute the gradient of the loss function for a single given weight $w_{i,j}^L$ from using the chain-rule ($\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$)[5]

$$\frac{\partial L(x, y)}{\partial W_{i,j}^{(L)}} = \frac{\partial L(x, y)}{\partial a_i^{(L)}} \frac{\partial a_i^{(L)}}{\partial W_{i,j}^{(L)}} \quad (2.37)$$

where it is seen from (2.15) that

$$\frac{\partial a_i^{(L)}}{\partial W_{i,j}^{(L)}} = z_j^{(L-1)} \quad (2.38)$$

with $z_j^{(L-1)}$ being the activation of the j 'th neuron in layer $L-1$, and we define

$$\delta_i^{(L)} = \frac{\partial L(x, y)}{\partial a_i^{(L)}} \quad (2.39)$$

as the “error” of the i 'th output neuron.

This defines the gradient of the loss function w.r.t. the weights leading to the output layer as

$$\frac{\partial L(x, y)}{\partial w_{i,j}^{(L)}} = \delta_i^{(L)} z_j^{(L-1)} \quad (2.40)$$

By defining the error $\delta_i^{(L)}$ for $l = 1, \dots, L-1$ for the remaining layers like in (2.39) we have

$$\delta_i^{(L)} = \frac{\partial L(x, y)}{\partial a_i^{(L)}} = \sum_{k=1}^{N^{(l+1)}} \frac{\partial L(x, y)}{\partial a_k^{(l+1)}} \frac{\partial a_k^{(l+1)}}{\partial a_i^{(l)}} \quad (2.41)$$

where it is seen that $\frac{\partial L(x, y)}{\partial a_k^{(l+1)}} = \delta_k^{(l+1)}$ and using (2.21) the term $a_k^{(l+1)}$ can be expressed as $N(l)$

$$a_k^{(l+1)} = \sum_{k'=1}^{N^{(l)}} z_{k'(l)} w_{k, k'}^{(l+1)} \quad (2.42)$$

$$= \sum_{k'=1}^{N^{(l)}} z_{k'(l)} w_{k, k'}^{(l+1)} \quad (2.43)$$

$$= \sum_{k'=1}^{N^{(l)}} \sigma(a_{k'}^{(l)}) w_{k, k'}^{(l+1)} \quad (2.44)$$

which from inserting into (2.41) yields

$$\delta_i^{(l)} = \sum_{k=1}^{N^{(l+1)}} \delta_k^{(l+1)} \frac{\partial(\sum_{k'=1}^{N^{(l)}} \sigma(a_{k'}^{(l)}) w_{k, k'}^{(l+1)})}{\partial a_i^{(l)}} \quad (2.45)$$

from where it is seen that the only term remaining when taking the derivative of the sum w.r.t. $a_i^{(l)}$ is the term where $k' = i$, i.e. the derivative simplifies to

$$\frac{\partial(\sum_{k'=1}^{N^{(l)}} \sigma(a_{k'}^{(l)}) w_{k, k'}^{(l+1)})}{\partial a_i^{(l)}} = \sigma'(a_i^{(l)}) w_{k, i}^{(l+1)} \quad (2.46)$$

and since $\sigma(a_{k'}^{(l)})$ is independent of k the final recursive expression is found to be

$$\delta_i^{(l)} = \sigma'(a_i^{(l)}) \sum_{k=1}^{N^{(l+1)}} \delta_k^{(l+1)} w_{k, i}^{(l+1)} \quad \text{for } l = 1, \dots, L - 1, \quad (2.47)$$

where the recursive dependency is shown explicitly, i.e. the errors $\delta_i^{(l)}$ depends on the errors from the later layer $\delta_k^{(l+1)}$ [5]

Finally having computed the errors of each neuron in the network, the gradients can be computed for each layer like for the output layer shown in (2.40)

$$\frac{\partial L(x, y)}{\partial w_{i,j}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)} \quad \text{for } l = 1, \dots, L - 1 \quad (2.48)$$

and hence making the training of the FFNN efficient.

2.2.2 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning method that can take an input image and give importance (learnable weights and biases) to various aspects/objects in the image, as well as differentiate between them. The amount of pre-processing required by a ConvNet is much less than that required by other classification techniques. Although primitive methods necessitate hand-engineering of filters, with enough preparation, ConvNets can learn these filters/characteristics. A ConvNet's design is inspired by the Visual Cortex's structure and is comparable to the communication pattern of Neurons in the Human Brain. Individual neurons can only respond to stimuli in the Receptive Field, a tiny portion of the visual field. A collection of these fields can be piled on top of one another to occupy the complete visual field. A ConvNet may successfully capture the Spatial and Temporal relationships in a picture by applying necessary filters. The architecture achieves superior fitting to the picture dataset due to the reduced number of parameters involved and the reusability of weights. In other words, the network can be trained to recognize the image's level of complexity.

In the figure 2.5, The three-color planes — Red, Green, and Blue — have been split in an RGB image. Color spaces such as Grayscale, RGB, HSV, CMYK, and others can be used to store files.

You can imagine how computationally hard things will get once photos are larger than 8K (7680x4320). The ConvNet's goal is to compress the pictures to a format that is easier to process while keeping elements that are necessary for a decent prediction. When creating an architecture capable of learning features while still being scalable to huge datasets, this is crucial.

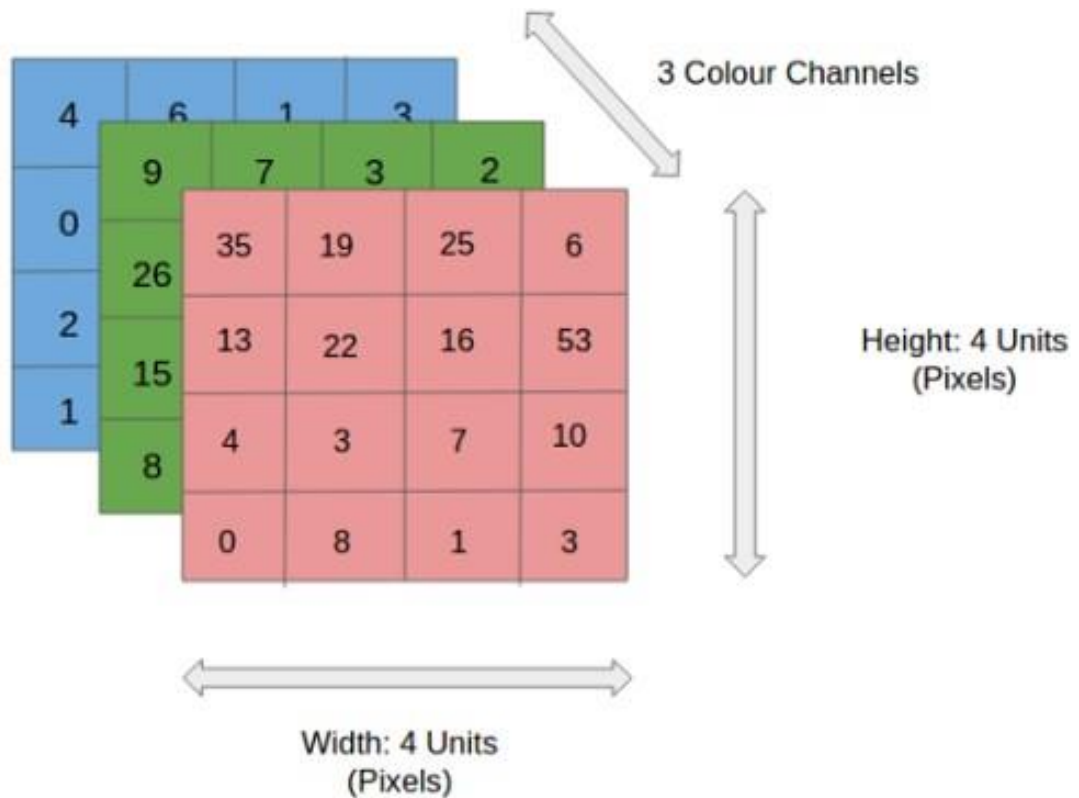


Figure 2.5. 4x4x3 RGB Image

2.2.3 Convolution Layer — The Kernel

Image Dimensions = 5 (Height) x 5 (width) x 1 (Number of channels, eg. RGB) The green section in the above demonstration resembles our 5x5x1 input picture, I. The Kernel/Filter, K, is the variable that performs the convolution operation in the first part of a Convolutional Layer. It is shown in yellow. K has been chosen as a 3x3x1 matrix.

The Kernel shifts 9 times, each time conducting a matrix multiplication operation between K and the picture part P over which the kernel is hovering since Stride Length = 1 (Non-Strided).

The filter changes to the right at a given Stride Value till it parses the complete

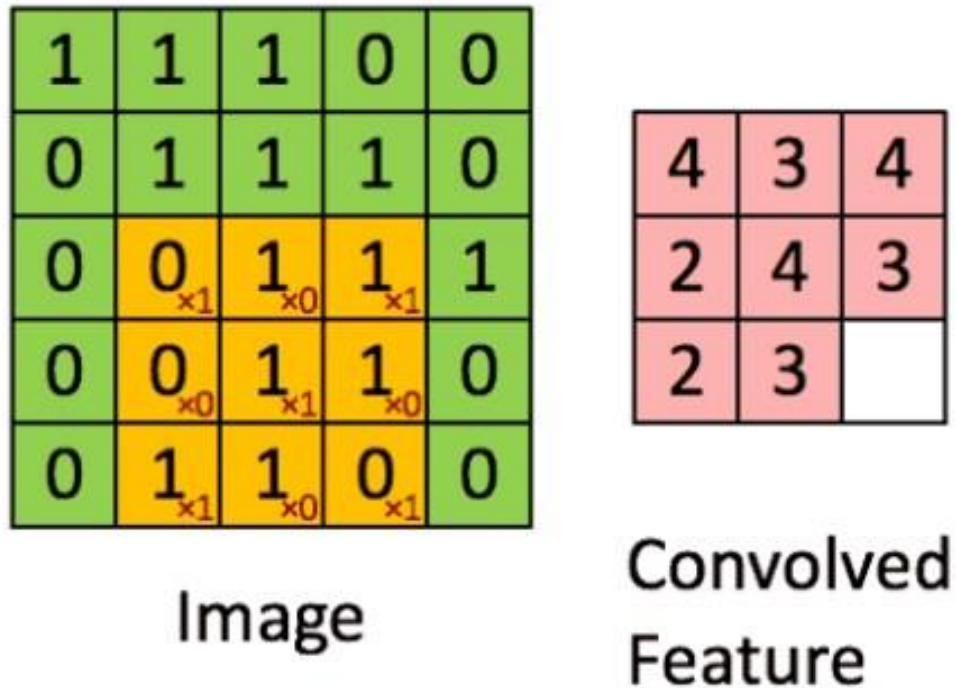


Figure 2.6. 4x4x3 RGB Image distance

Then it hops down to the beginning (left) of the picture with the same Stride Value and repeats the procedure until the full image has been visited. The operation is show in 2.6 and 2.7.

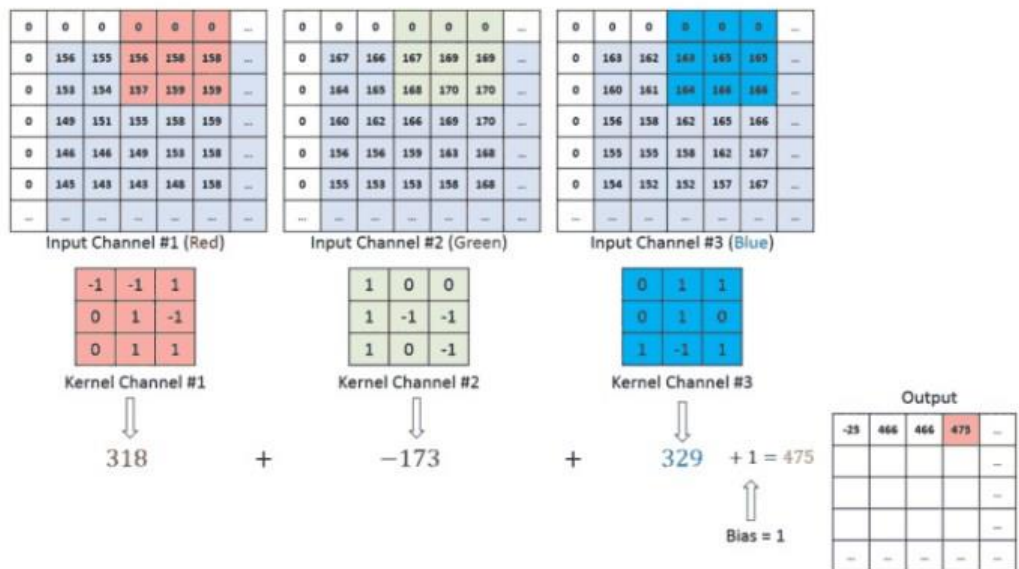


Figure 2.7. Convolution operation on a MxNx3 image matrix with a 3x3x3

Kernel In the case of photos with many channels, the Kernel has the same depth as the input image (e.g. RGB). Matrix Multiplication ($[K1, I1]; [K2, I2]; [K3, I3]$) is performed between the K_n and I_n stacks, and the results are combined with the bias to produce a flattened one-depth channel Convolved Feature Output.

The Convolution Operation is used to extract high-level characteristics such as edges from an input image. Limiting ConvNets to only one Convolutional Layer isn't essential. The first ConvLayer is traditionally responsible for gathering LowLevel data such as edges, color, gradient direction, and so on. With the addition of layers, the architecture responds to the High-Level properties as well, giving us a network that understands the photographs in the dataset as well as we do.

The process yields two sorts of results: one in which the convolved function's dimensionality is reduced when compared to the input, and the other in which the dimensionality is raised or unaltered. In the first situation, Valid Padding is used, whereas in the second case, the Same Padding is used.

When we augment the 5x5x1 picture into a 6x6x1 image and then put the 3x3x1 kernel over it, we discover that the convolved matrix has dimensions of 5x5x1. The term "Same Padding" was invented as a consequence.

However, if we do the same thing without padding, we obtain Valid Padding, which is a matrix with the same dimensions as the Kernel (3x3x1).

2.2.4 Pooling Layer

The Pooling layer, like the Convolutional Layer, is in charge of lowering the Convolved Feature's spatial scale. Dimensionality reduction reduces the amount of processing power needed to process the data. It may also be used to extract rotational and positional invariant dominant features, which can help with the training phase of the model.

Pooling may be divided into two types: maximum pooling and average pooling. Max Pooling returns the highest value from the picture's Kernel-protected area. Average Pooling returns the average of all the values from the Kernel's region of the image.

Max Pooling works as a Noise Suppressant as well. It removes all noisy activations while simultaneously de-noising and reducing dimensionality. Average pooling, on the other hand, reduces dimensionality as a noise-suppression approach. As a result, we may conclude that Max Pooling outperforms Average Pooling.

The Convolutional Layer and the Pooling Layer make up the i -th layer of a Convolutional Neural Network. The number of such layers may be expanded even higher, depending on the picture complexity, to collect even more low-level data, but at the cost of additional computational power.

After going through the aforesaid approach, we were able to get the model to grasp the characteristics. The final result will then be flattened and sent into a conventional Neural Network for classification.

2.2.5 Classification — Fully Connected Layer (FC Layer)

Using a Fully-Connected layer to learn non-linear combinations of high-level information represented by the convolutional layer's performance is a (usually) low-cost method. The Fully-Connected layer is learning a possibly non-linear function in that space. an example is shown in 2.8.

We'll flatten the image into a column vector now that we've changed it to a format suited for our Multi-Level Perceptron. Every round of training uses backpropagation to send the flattened output to a feed-forward neural network. The model can discriminate between dominating and low-level properties in images and categorize them using the Softmax Classification technique over a period of epochs.

There are a variety of CNN architectures available, all of which have played a role in developing algorithms that power and will continue to power AI in the

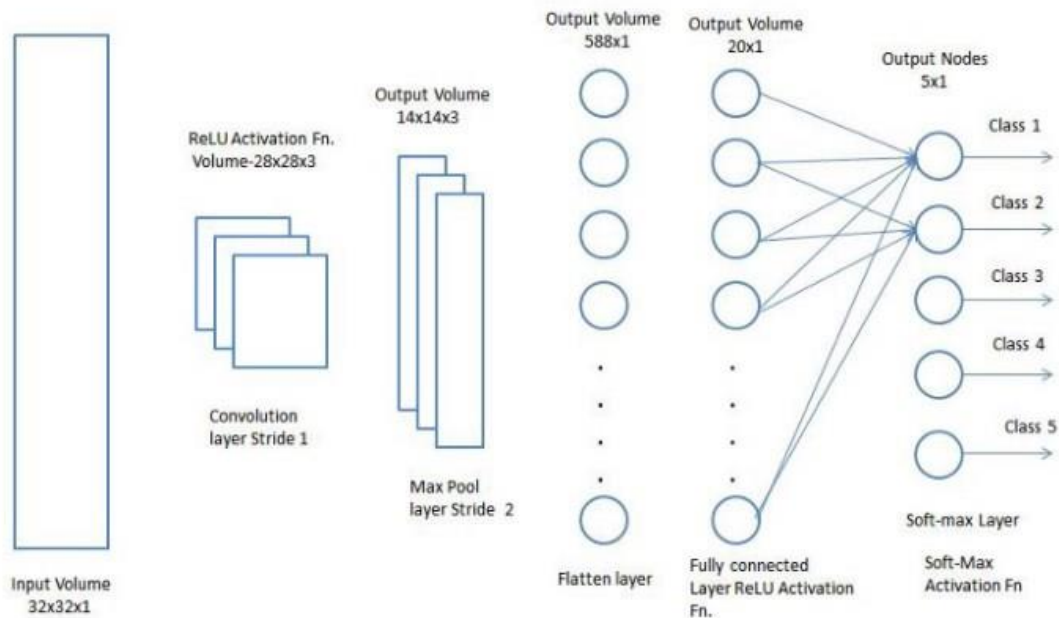


Figure 2.8. Example Of CNN model near future

The following are a few of them:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet

2.2.6 Recurrent neural network

Although the FFNN provided in Section 2.2.1 is a strong model, it is not without flaws. One is that it is a memoryless model, which means that the network will have no knowledge of prior inputs for each input. However, a "Recurrent Neural

Network" may be used to attain this attribute (RNN). RNNs are essentially the same as FFNNs, with the exception that each layer l not only passes its output to the next layer $l + 1$, but also to itself, resulting in a "recurrent connection." This is visualized in Figure 2.9.

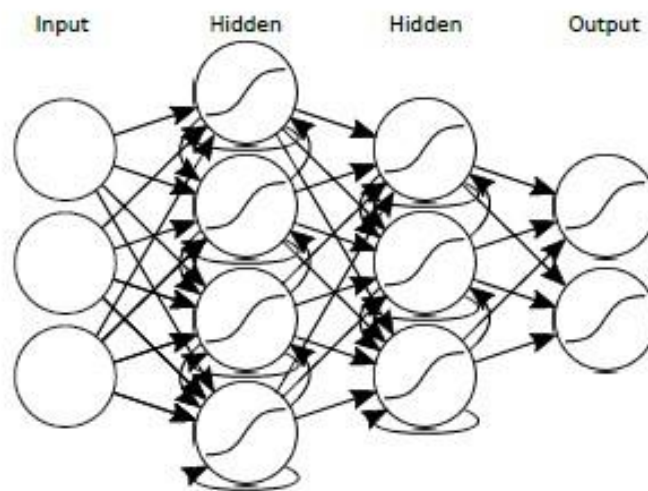


Figure 2.9. Visualization of a 3-layer recurrent neural network with input $x \in @3$ and output $y \in @2$

A popular way of visualizing RNN's is by so-called "unfolding" the network and thereby explicitly show the recurrent connections between each time-step t .

An example of the network shown in Figure 2.9 unfolded can be seen in Figure 2.10.

Let input variables be sequences of length T i.e. $x = x^{(1)}, x^{(2)}, \dots, x^{(T)}$ e.g. representing each character in a sentence of length T , and denote

$$u = \{U^{(1)}, U^{(2)}, \dots, U^{(L-1)}\} \quad (2.49)$$

the set of $U^{(l)} = [u_1^{(l)}, u_2^{(l)}, \dots, u_{N^{(l)}}^{(l)}]$ weights for the recurrent connections (the output layer has no recurrent connections) with and

. Then denote the activation of neuron k in layer l at a given time-step t expressed by extending (2.21) with a term for the recurrent connection

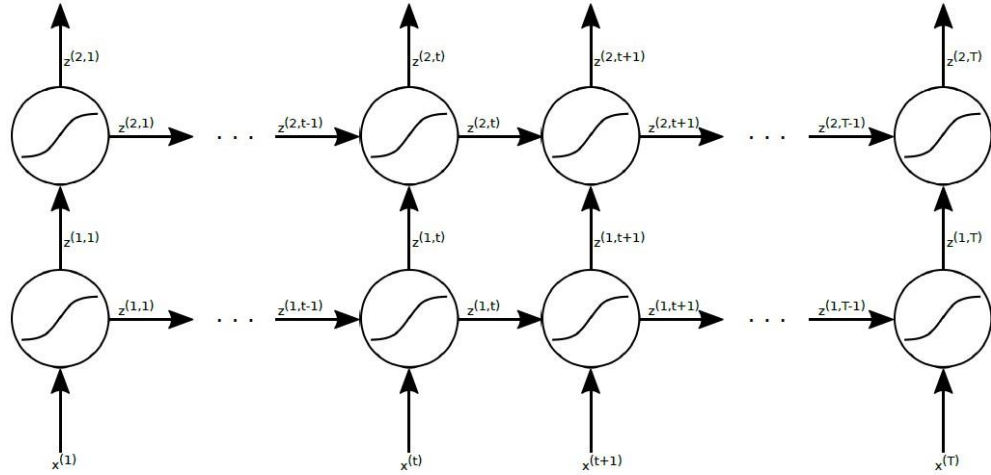


Figure 2.10. Visualization of an unfolded 3-layer recurrent neural network (2 recurrent layers and 1 output layer) with input sequences of length T

$x^{(1)}$ represents the initial input at time-step $t = 1$ of a given observation x and $z^{(2,T)}$ the final output at time-step $t = T$.

$$z_k^{(l,t)} = \sigma(a_k^{(l,t)}) = \sigma\left(\sum_{k'=1}^{N^{(l+1)}} \delta_{k'}^{l+1,t} w_{k',k}^{(l+1)} \underbrace{\sum_{k'=1}^{N^{(l)}} \delta_{k'}^{l,t+1} u_{k',k}^{(l)}}_{\text{recurrent term}}\right) \quad (2.50)$$

with $\delta_k^{l,T+1} \forall k, l$ for a complete definition of the recurrency. Finally the BPTT algorithm defines the gradients as summing the error terms for each time-step t [8]

$$\frac{\partial L(x, y)}{\partial w_{k,j}^{(l)}} = \sum_{t=1}^T \delta_k^{(l,t)} z_k^{(l,t)} \quad (2.51)$$

Even though this recurrent time-dependency of RNN's make them capable of learning long-term dependency patterns, and even capable of utilizing the information of any input value it has ever seen in theory, this is found to not work be true in practice. Some of the problems and how to handle them will be discussed further in Section 2.2.4.

2.2.7 Bidirectional recurrent neural network

One limitation of a typical RNN is that the network may only use information from prior time steps at a particular time-step t . The use of "Bidirectional Recurrent Neural Networks" (BRNNs) allows the network to have both prior and future information available at time t . This is accomplished by

feeding the sequence to the network in both normal and reversed order, each with its own recurrent hidden layer, and then passing both through the network [29]. This, of course, necessitates the availability of future time-steps at all time-steps t ; so, it would not work, for example, for a decoder that predicts one character at a time and uses the previous character to predict the next.

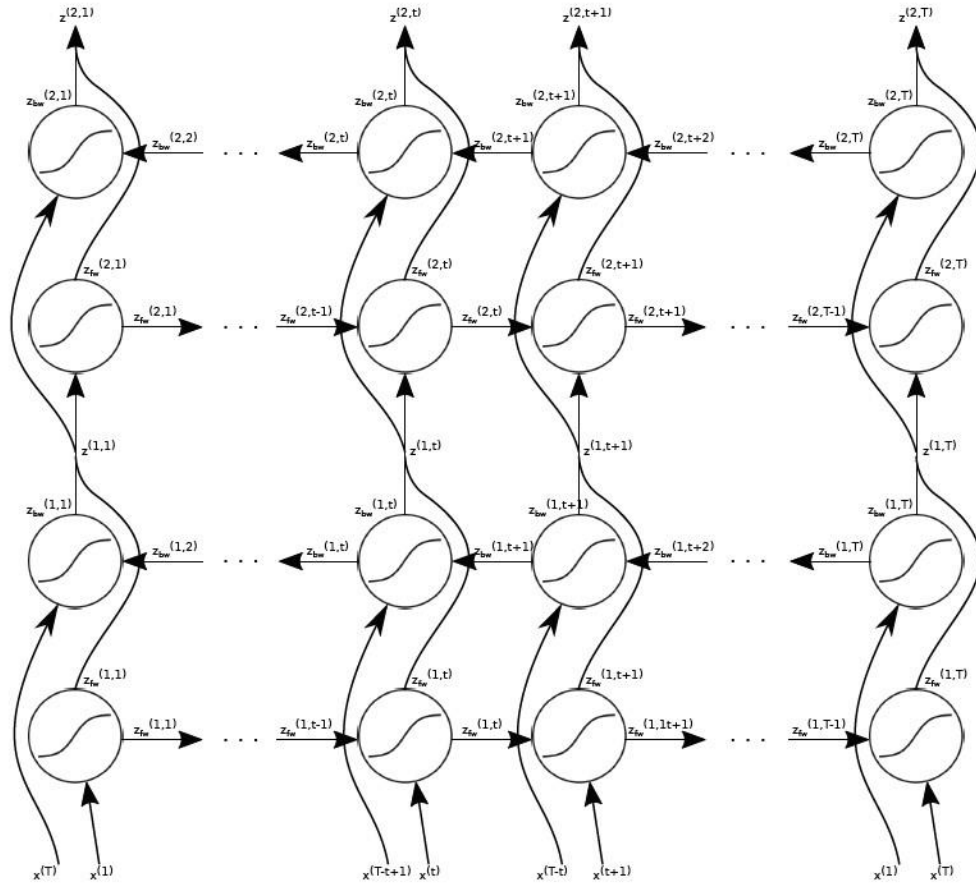


Figure 2.11. Visualization of an unfolded 3-layer bidirectional recurrent neural network with input sequences of length T

$z_{fw}^{(l,t)}$ and $z_{bw}^{(l,t)}$ denotes the forwards and backwards passed activation of layer l at time t respectively. $z^{(l,t)}$ denotes the merged backwards and forwards passes.

A visualization of a BRNN can be seen in Figure 2.11 where it is shown that the output of the forwards- and backwards recurrent layers are concatenated before being fed as input to the next layer. This is one way of handling the extra hidden state from the backwards layer. Another way could be to project the merged outputs to the size of a single output layer, to maintain the state size throughout the network.

2.2.8 Long short-term memory

The conventional architecture of RNNs, as discussed in Section 2.2.6, has several issues with processing recurring information in the network. Learning long-

term dependencies, in particular, has been proven to be a serious challenge for conventional RNNs in practice [24]. The fact that each time-step information is transmitted back into its own hidden layer has demonstrated to either blow out or degrade the outputs exponentially, making training extremely difficult. The "exploding gradient problem" and the "vanishing gradient problem" [8] are two terms for this.

The widely utilized "Long Short-Term Memory" (LSTM) [13] architecture can be employed to solve these challenges. The LSTM is the recommended RNN architecture above conventional RNNs because it handles long-term dependencies and keeps gradient information across time better than regular RNNs [8].

Because the LSTM allows a single RNN neuron to perform several operations, each neuron in the RNN is referred to as a "block." A simple illustration of a regular RNN block can be seen in Figure 2.12. $z^{(t-1)}$ as input and outputs a vector of size C with values in the range $]0, 1[$ using the sigmoid function and learned weights and biases $W_f, W_i, W_o,$ and b_f, b_i, b_o . Letting g and h denote non-linear functions often chosen as \tanh , and L and N denote element-wise addition and multiplication respectively, the gates and their purpose can be explained as:

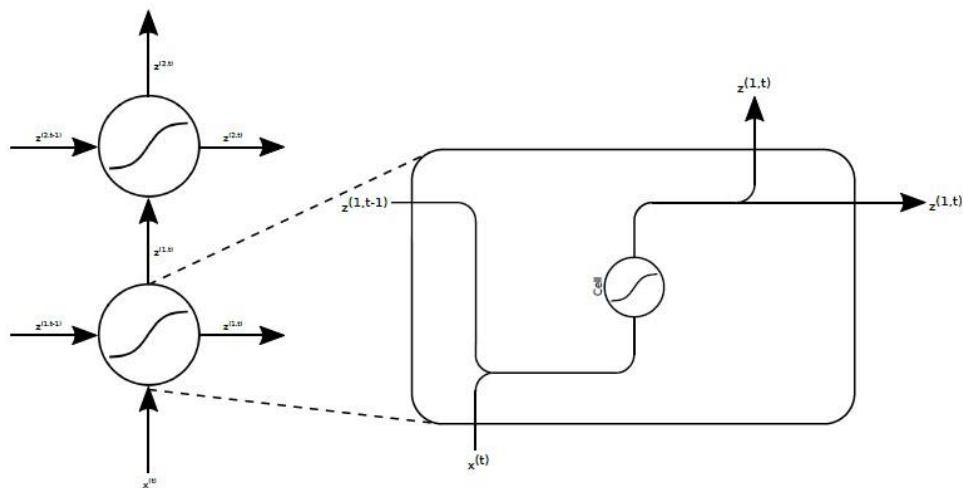


Figure 2.12. A regular RNN block with $x^{(t)}$ and $z^{(1,t)}$ being the input and output respectively, and $z^{(1,t-1)}$ being the output from the previous time-step

- The Forget gate: Determines what information to forget by performing elementwise multiplication between the output of the gate $f(t)$ and the previous cell state $c(t-1)$.

- The Input gate: Determines what new information from the cell values $\tilde{c}^{(t)} = g(W_c[z^{(t-1)}, x^{(t)}] + b_c)$ should be added to the cell state by performing elementwise addition between the updated cell state $c^{(t-1)} \otimes f^{(t)}$ and the modified cell values $i^{(t)} \otimes \tilde{c}^{(t)}$ resulting in the new cell state $c^{(t)} = c^{(t-1)} \otimes f^{(t)} + i^{(t)} \otimes \tilde{c}^{(t)}$.

- The Output gate: Determines what values to output by performing elementwise multiplication between a non-linear transformation of the new cell state $h(c^{(t)})$ and the output values of the gate $o^{(t)}$, leading to the final output of the block $z^{(t)} = o^{(t)} \otimes h(c^{(t)})$

A visualization of the LSTM block can be seen in Figure 2.13 and the full set of equations is listed in (2.53) to (2.58).

$$\text{Forget gate} \quad f^{(t)} = \sigma(W_f[z^{(t-1)}, x^{(t)}] + b_f) \quad (2.52)$$

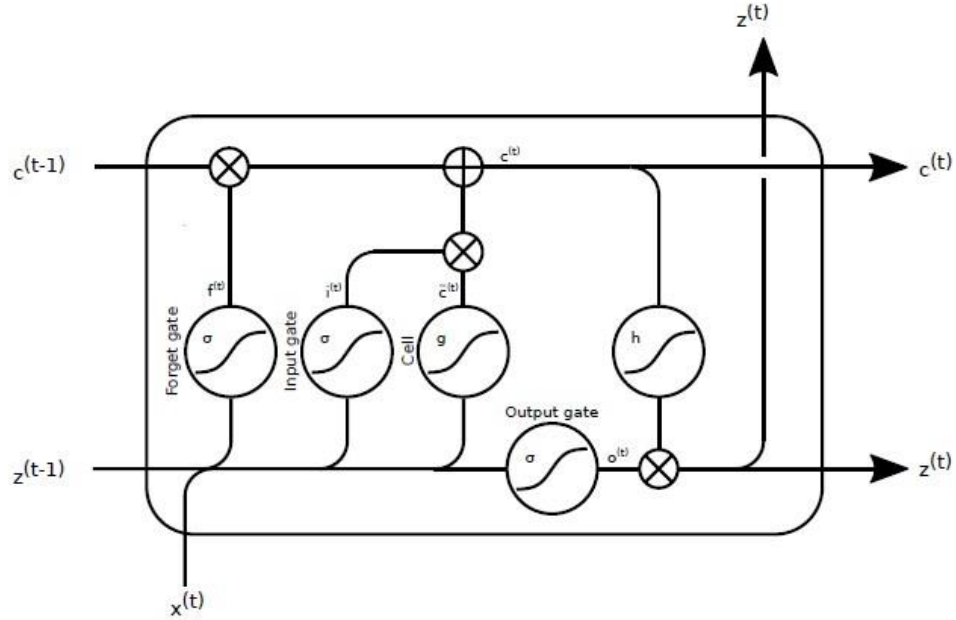


Figure 2.13. LSTM block with $x^{(t)}$ and $z^{(t)}$ being the input and output, and $c^{(t)}$ being the cell state at time-step t

σ denotes sigmoid functions and g and h denotes non-linear functions. L denotes element-wise addition and N denotes element-wise multiplication.

$$\text{Input gate} \quad i^{(t)} = \sigma(W_i[z^{(t-1)}, x^{(t)}] + b_i) \quad (2.53)$$

$$\hat{c}^{(t)} = g(W_c[z^{(t-1)}, x^{(t)}] + b_c) \quad (2.54)$$

$$\text{State update} \quad c^{(t)} = f^{(t)} \otimes c^{(t-1)} + i^{(t)} \otimes \hat{c}^{(t)} \quad (2.55)$$

$$\text{Output gate} \quad o^{(t)} = \sigma(W_o[z^{(t-1)}, x^{(t)}] + b_o) \quad (2.56)$$

$$z^{(t)} = o^{(t)} \otimes h(c^{(t)}) \quad (2.57)$$

For the function g and h in the input and output gate, the tanh function is chosen as activation function in this project.

2.3 Sequence to sequence learning

The overall goal of mapping a given sequence to another sequence, which is the main aim of an RNN, is known as a sequence to sequence learning (seq2seq). Instead of employing a single RNN for the job, the work can be broken into two parts as shown in [32], with each task requiring a separate RNN. The first RNN encodes a given sequence to a fixed length vector representation (the "encoder"), and the second RNN decodes the vector to the target sequence. This is the framework known as "Encoder-Decoder."

An example of an input sequence ABC of length $T = 3$ encoded to a fixed vector $c_{ENC}^{(T)}$ can be seen in Figure 2.14, where it is seen that the fixed vector $c_{ENC}^{(T)}$ is taken as the state of the RNN at time-step T - the end of the sequence, which is represented to the network as an extra "end-of-sequence" class denoted the $\langle \text{EOS} \rangle$ token.

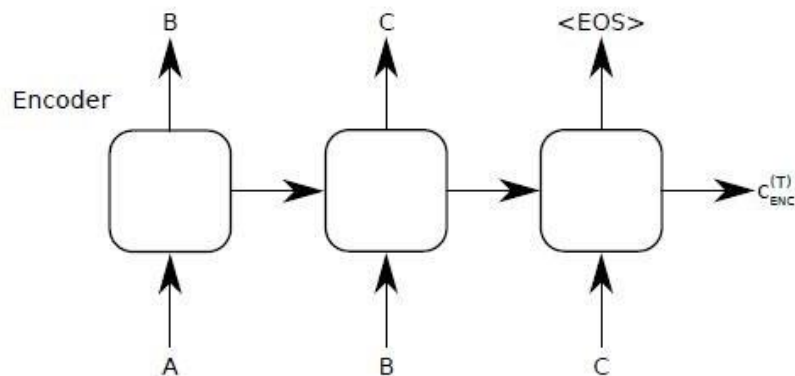


Figure 2.14. Encoder: Example of an encoder encoding the input sequence ABC to a fixed vector representation $c_{ENC}^{(T)}$

The state of the decoder is then initialized with the value of the encoded input sequence $c_{ENC}^{(T)}$ and using an extra "start-of-sequence" class, denoted the $\langle \text{GO} \rangle$ token representing the beginning of the decoding, as the initial input decodes the target sequence. This is visualized in Figure 2.15

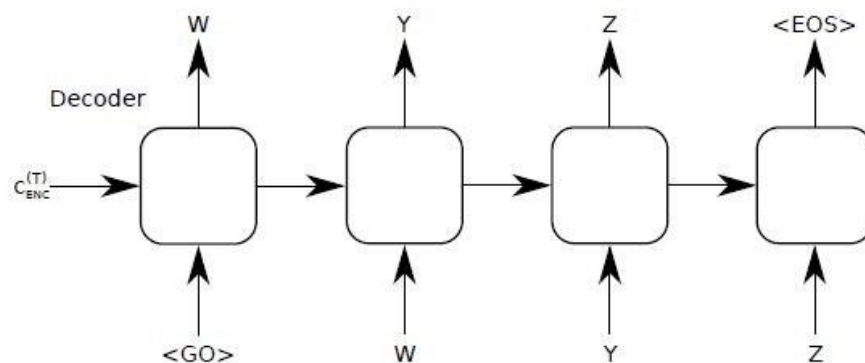


Figure 2.15. Decoder: Example of a decoder decoding the target sequence WXYZ using the encoded fixed vector representation $c_{ENC}^{(T)}$ of the input sequence as initial hidden state

2.4 Batch normalization

Normalizing the training data to have a 0 mean and 1 variance is a standard pre-processing step when training neural networks. This makes learning the right weights easy for the network since it doesn't have to alter its weights to meet the varying offsets in the data batches.

However, when training deep neural networks, normalizing as a pre-processing phase appears to be insufficient, since the offsets of the data vary when the weights in the layers are modified, indicating that normalizing as a pre-processing step is insufficient.

[16] proposes a strategy for dealing with this problem called "Batch normalization." Rather than just normalizing the data prior to training, each mini-batch of training data is standardized prior to each layer of a network. If a layer $XinRnp$ receives input from a mini-batch of size n with each observation containing p features, and x_j is the j 'th dimension of each observation in the mini-batch, then each dimension of the mini-batch is normalized using

$$\hat{x}_j = \gamma_j \frac{x_j - \mu_j}{\sigma_j} + \beta_j \quad (2.58)$$

with $\gamma_j, \mu_j, \sigma_j$ and β_j being learned parameters for each dimension with γ_j and σ_j being initialized at 1 and μ_j and β_j initialized at 0.

Batch normalization has been demonstrated to reduce deep network training time, improve performance, and improve training stability [16]. In addition, it functions as a regularizer.

2.5 Optimization

The purpose of neural network training is to reduce the network's objective function (loss function) in proportion to its parameters to the smallest possible value. This optimization is often carried out using gradient-based optimization methods such as "Gradient descent." This project uses the "Adam" algorithm, which is a gradient-based approach.

Chapter 3. Data collection and storage

The KOHTD database discusses the problem of recognizing manuscripts in Kazakh and Russian languages in relation to Cyrillic graphics, describes and investigates various approaches, as well as presents the results of the study, and suggests the Bluche and Puigcerver models [40, 41] for comparing the results. The recognition of handwritten text of the Kazakh-Russian language remains not fully investigated. In this regard, the development and research of new effective algorithms for recognizing handwritten text of the Kazakh-Russian language are relevant. The approach to the problem of handwriting recognition of the Kazakh-Russian language, based on the use of neural networks, is proposed. The main stage of handwriting data collection of the Kazakh-Russian language consists of the following stages:

- Pre-processing for handwriting recognition: at this stage, the image is processed to improve its quality and bring it to a form that is convenient for segmentation. At the pre-processing stage, the handwritten text is scanned with a Canon MF4400 Series UFR II scanner. The resolution for the scanned examination lists is 300 dpi and the color depth is 24 bits. Translation of paper documents into a digital graphical representation.
- Segmentation of the scanned handwritten text into words: at this stage, the scanned handwritten text is divided, or segmented, into convenient parts for analysis. The most natural actions at this stage are to split the text into separate lines (line segmentation) and split the lines into words (word segmentation), where space is their separator. To do this, filters are consistently applied to the text to remove noise and determine the boundaries of words.
- Annotation of segmented words, which will map each image to its text in json format file.

3.1 Labeling in the database

The main idea was as follows: images of segmented words were sent to volunteer users via Telegram bot named @collectorOfdataset bot (botCollector). It is a messenger application for mobile phones that allows you to create programs in Python and register them as bots [42]. To simplify the process of collecting annotations, it was decided to send the same image to two random users. In case the sent annotations from both users turned out to be absolutely identical, we considered this result to be reliable.

After sending out half of the images from the total number and receiving the results, it turned out that most of the users filled out annotations very inattentively, and sometimes even wrote deliberately inappropriate annotations. To exclude such cases, we trained a neural network for handwriting recognition on an already existing incomplete dataset. Further, after each input and sending of the annotation,

we calculated the Levenshtein distance between the result from the user and the result of recognition. To our surprise, even on a partial dataset, the neural network produced subjectively excellent results. Also, by typing the command /my - annotations, each user could see their statistics: the number of annotated words, the number of words whose annotations do not match the annotation from another user, the average confidence value.

To get another picture for annotation, users sent the command /getImage (Figure 3.1). If the image contains incomprehensible words, only numbers, corrections, words in a foreign language, several lines, the inscription had to be removed by clicking the "Delete" button.

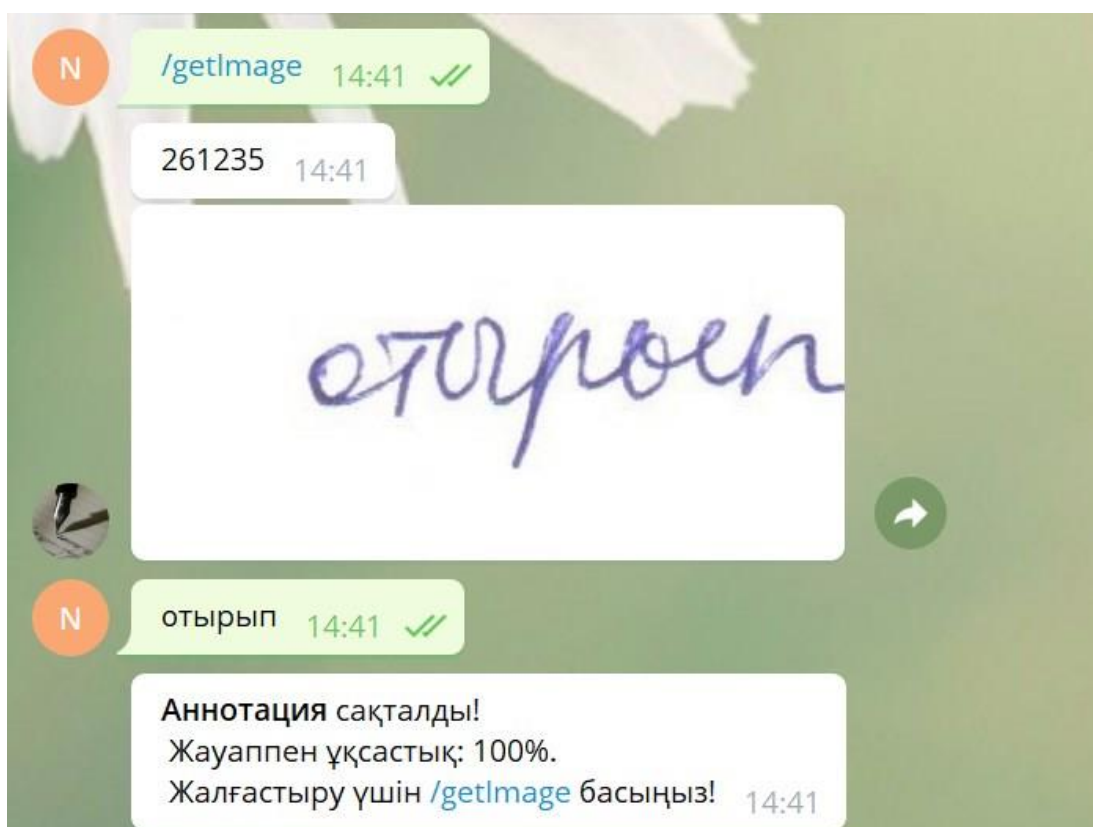


Figure 3.1. Result of the /getImage command with 100% similarity

The complexity of the handwriting recognition task is a large variety of handwriting, shapes, sizes of letters, and a variety of languages. Also, the paper with the text may contain “noises” – paper defects, foreign spots, which also complicates the whole process. Handwritten texts, as you know, differ from printed ones in the manifestations of individual handwriting properties: from the calligraphically printed letters of each word in the text to the illegibly written text as a whole, despite the fact that there is a standard for writing a particular letter, and this standard writing was taught in elementary school by every writer. But in the process of practicing writing, each writer develops individual features of handwriting. For handwriting recognition in the Kazakh language, more than 3 thousand pages in A4 format were scanned from various handwriting samples

(handwritten exam answers to students' questions). Table 3.1 shows the recognition of the handwritten text of the Kazakh language of one voluntary user.

At the time of this writing, 643 people were involved in this work (mainly students and university teachers).

Table 3.1 Recognition of handwritten text of the Kazakh language

N	input	Words	Similarity
1		отырып	100%
2		яғни	75%
3		адам	67%
4		әдістеріне	84%
5		мөлшерде	100%
6		тұтын	83%
7		Ипотекалық	90%
8		арқылы	83%
9		қауіп	60%

The table shows that the similarity to the answer depends on the person's handwriting. In the Kazakh language, the letters “ң” and “қ” (Table 3.1, Line 7), “м” and “ш”(Table 3.1, Line 3), “н” and “и”(Table 3.1, Line 6), “л” and “е”(Table 3.1, Line 8), are similar in handwritten text. In handwritten texts, the recognition process is complicated by the individual features of the handwriting, including the variability of writing letters. The presence of gaps in the text also contributes to the perception and recognition of the manuscript. They help you not to read the text sequentially from beginning to end. In addition to spaces, capital letters, as well as lowercase letters that come out of an even row of lines, are essential for the perception and recognition of what is written. A voluntary user sometimes enters a word with a capital letter with a lowercase one, which reduces the percentage of similarity with the answer (Table 3.1, line 9).

3.2 Characteristics of the Database

The database is composed of segmented words form over 3000 scanned exam papers. Generally speaking, a separate study needs to be done to estimate the number of people who filled out these forms. But we assume that on average one person completed 2 examination sheets. Accordingly, according to our estimates, we have approximately 1000 different style of handwriting paper. By the number of people who annotated the images, we can say for sure that at the time of writing this article, there were 643 of them, since we have all the records in the database.

There are approximately 922010 symbols shown in Figure 3.2. Total number of images in the dataset are 140335 images after pre-processing and segmentation the examination lists.

3.3 Statistic Analysis

In this section, statistics on the database are presented. The database is split into three exclusive parts(training, validation, and test sets). Table 3.2 shows the uni-, bi-, and tri-grams of these sets and of the full database and also shows how many words, unique words, and number of characters for all the dataset and the parts.

Table 3.2 N-gram statistics of the database

Set	Word count	Unique word	Character count	Unigrams	Bigrams	Tri-grams
Training	104278	24943	98258	24943	100142	104242
Validation	22386	8870	21054	8870	22076	22376
Testing	22351	8984	21054	21054	22040	22346
All data	149015	31483	140366	31483	130912	147238

Table 3.3 shows the out of vocabulary (OOV) statistics of the validation and test sets compared with the training data sets.

Table 3.3 N-gram statistics of the database

Out of Vocabulary	OOV Tokens	OOV Percentage
Validation	3408	10.36%
Testing	3442	10.46%

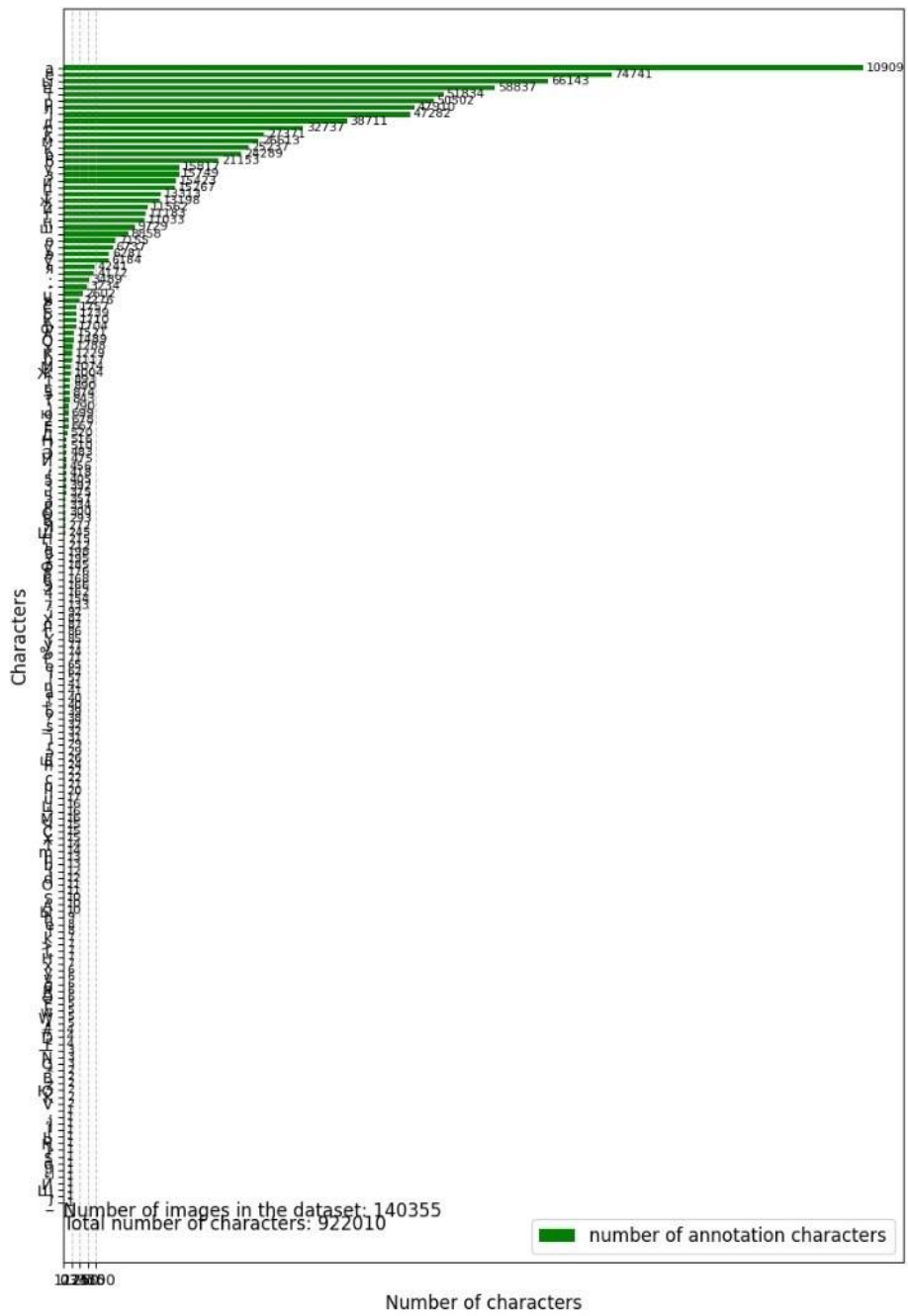


Figure 3.2. Histogram of Characters in the dataset

Chapter 4. Dataset segmentation

Segmentation of text into lines is one of the most important steps in the optical character recognition (OCR) process, in particular, in optical recognition of document images. Line segmentation is the decomposition of an image containing a sequence of characters into fragments containing individual characters.

The importance of segmentation is due to the fact that the majority of modern OCR systems are based on classifiers (including neural network) of individual characters, and not words or text fragments. In such systems, errors of incorrect insertion of cuts between characters, as a rule, are the cause of the lion's share of errors in the final recognition.

In our case, for line segmentation, we used a genetic algorithm. Genetic Algorithm (GA) is a classic evolutionary algorithm based on random enumeration of a parameter. By random here we mean that in order to find a solution using

GA, random changes were applied to the current solutions to generate new ones. GA is based on Darwin's theory of evolution. It is a slow, gradual process that works by making small and slow changes. In addition, GA is slowly making small changes to its decisions until it gets the best solution.

To implement the genetic algorithm, first of all we need to determine the coordinates from where we will start making changes. For correct segmentation, the coordinates must lie between two lines. To determine the extreme points, we get a histogram of the picture in height. If the line is completely white, then the sum of pixels will be equal to 255, for convenience we will make an inversion, then the white color will be equal to zero. So our optimization function should tend to zero. The result is shown in Figure 4.1.

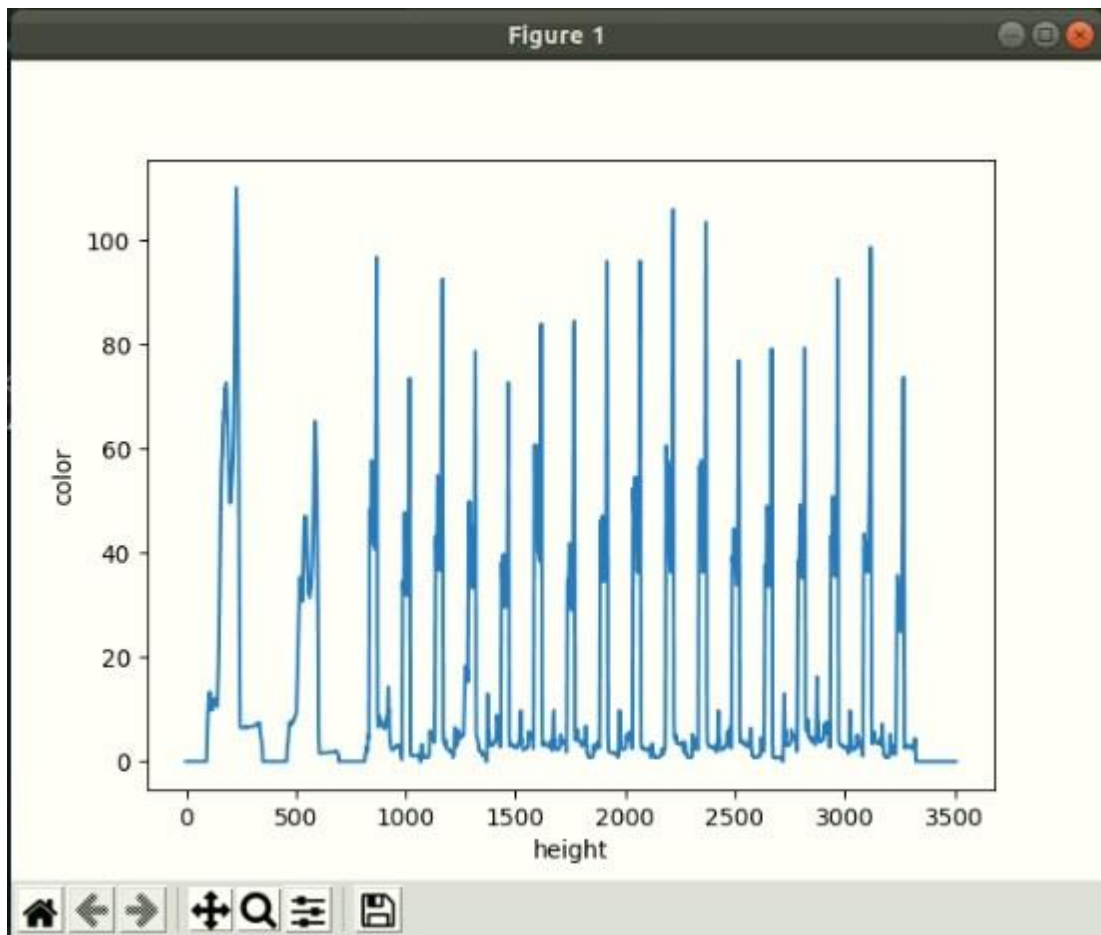


Figure 4.1. Image histogram by height

The histogram gives us the sum of the pixels of each line of the image, to get explicit vertices, we smooth the data using a gaussian filter. The result is shown in Figure 4.2. Let's draw lines at the maximum (blue) and minimum (green) points on the histogram, we get Figure 4.3.

Our coordinates will lie between the blue lines. After that, we run the genetic algorithm and get the line segmentation. The result is in Figure 4.4.

After receiving the lines, we need to segment the words, since our model accepts words as input. We will do the segmentation of words in the same way, through histograms. First, draw a histogram horizontally and smooth it with a Gaussian filter. Find the vertices and draw in the picture. We get the result as in Figure 4.4.

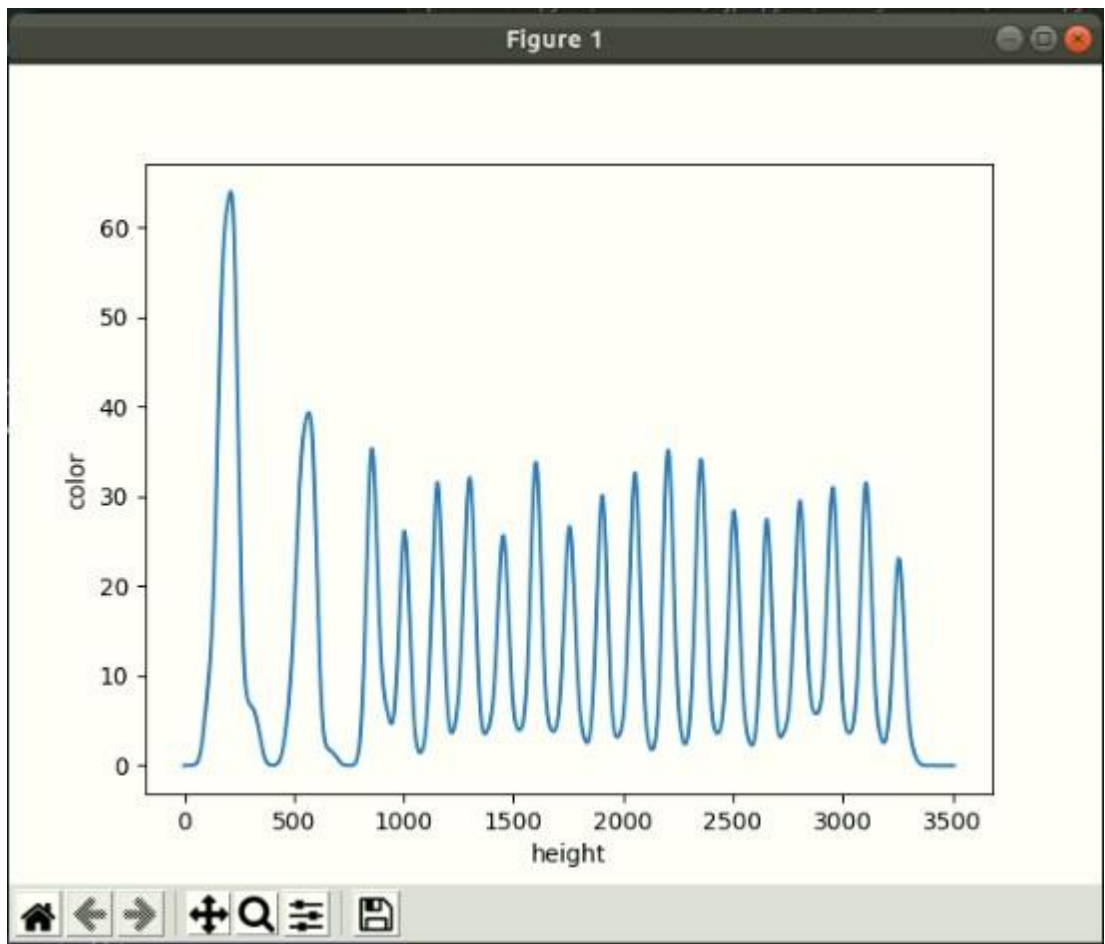


Figure 4.2. Smoothed image histogram by height

Пример Шрифта

«Стефан»

Изучение литературы итальянского Возрождения следует начать с рассмотрения творчества великого предшественника Тенессона, флорентинца Данте Алигери (Данте, 1265–1321), первого по времени из великих поэтов Западной Европы. По своему характеру его творчество Данте — поэт переходного времени, стоящий на рубеже двух великих исторических эпох. Италия была самой передовой с точки зрения европейского средневековья. Ее благоприятное географическое положение в центре Средиземноморского бассейна обеспечило ей раннее развитие посреднической торговли между Западной Европой и восточными странами. Эти торговые отношения с Востоком вызвали быстрый рост цветущих городов, которые первыми в Европе освободились от власти феодальных señоров и образовали самоуправляющиеся коммуны с республиканским образом правления.

Figure 4.3. Line drawing

Пример Шрифта

«Стефани»

Изучение литературы итальянского Возрождения следует начать с рассмотрения творчества великого предшественника Тенессанса, флорентийца Данте Алигьери (Данте, 1265—1321), первого по времени из великих поэтов Западной Европы. По своему характеру своего творчества Данте — поэт переходного времени, стоящий на рубеже двух великих исторических эпох. Италия была самой передовой с страной европейского средневековья. Ее благоприятное географическое положение в центре Средиземноморского бассейна обеспечило ей раннее развитие посреднической торговли между Западной Европой и восточными странами. Эти торговые сношения с Востоком вызвали быстрый рост цветущих городов, которые первыми в Европе освободились от власти феодальных сеньоров и образовали самоуправляющиеся коммуны с республиканским образом правления

Figure 4.4. Genetic Algorithm Result, String Segmentation

Алигьери | (Данте, | 1265—1321), | первого | по времени

Figure 4.5. Segmentation of words

Chapter 5. Experimental results

A quantitative comparison of well-known recurrent neural networks (RNN), such as Bluche[40], Puigcerver[41], Flor [43] and Abdallah[44] models, has been implemented to choose the best performing model on the dataset given. At the first, the final dataset was split into three datasets as follows: Training (70%), Validation (15%), and Testing (15%). After training, validation, and testing datasets were prepared, the models were trained, and a series of comparative evaluation experiments were conducted. As experiment results proved, Flor model demonstrated the best performance with 6.52 % character error rate (CER), 24.52% word error rate (WER) and 26.98% SER for the test dataset.

5.1 Evaluation methods

We used two ways to evaluate models in this article: For the results reported in the first technique, established performance measurements such as the character error rate (CER) and word error rate (WER)[45] are employed. The Levenshtein distance is calculated by dividing the total number of characters in the ground truth word (N) by the amount of character substitution (S), insertion (I), and deletions (D) required to convert one string into another.

$$CER = \frac{S + I + D}{N} \quad (5.1)$$

Similarly, the WER is determined by the sum of the number of term substitutions (S_w), insertions (I_w), and deletions (D_w) required for the transformation of one string into another, and divided by the total number of ground-truth terms (N_w).

$$WER = \frac{S_w + I_w + D_w}{N_w} \quad (5.2)$$

5.2 Training

Tensorflow[46], a Python-based deep learning library, was used to train all of the models. Through Python, Tensorflow allows for the transparent usage of highly efficient mathematical operations on GPUs. In the Python script, a computational graph is built to define all operations required for the given computations.

The machine that was carried out the tests with 2x Intel(R) Xeon(R) E-5-2680 processors, 4x NVIDIA Tesla k20x graphics cards, and 100 GB of RAM.

The usage of a GPU reduced model training time by about a factor of three, but this speed-up was not extensively evaluated during the project, so it may have varied.

The report's plots were built with the Python package matplotlib, and the visuals were developed with Inkscape, a vector graphics programme akin to Adobe Photoshop.

The validation loss value is minimized in all models. The RMSProp method[47] is used to execute the stochastic gradient descent optimization using a base learning rate of 0.001 and 32 mini-batches. We also used early stopping with patience 20 since we wanted to track the validation loss at each epoch and terminate training if the validation loss did not improve after 20 epochs.

5.3 Proposed Models

Proposed Models in this section we will present deep learning models based on RNN and CTC loss function which has been implemented to choose the best performing model on the dataset given. Bluche [40] built a deep neural network that could split into three main parts, First convolutional layers as the encoder of the input images. It operates two-dimensional representations and supplies 2D features maps. This part contains about 20% of the model's free parameters but represents the slowest component of their architecture. Aims to make it generic to be reusable to factorize the processing time. The second is the interface that transforms the 2D image-like representation into 1D representation. The third is the decoder, a bidirectional Long Short Term Memory Recurrent Neural Network (LSTM RNN) that processes feature sequences to predict a sequence of characters. This part has the most capacity of the network about 80% but has fast processing.

Puigcerver [41] present a neural network architecture based on convolutional and 1D-LSTM layers to make line-level Handwritten Text Recognition (HTR). Then providing statistically sound empirical study that proves the architecture gives similar or better accuracy compared with the state of the art 2D-LSTM architecture and incomparably faster. And showing that performing appropriate random distortions on the training images, reduce the error rates.

Flor[43] proposed New Gated CRNN architecture used for offline Handwritten Text Recognition (HTR) systems, by using the latest machine learning techniques and approaches in the field of Natural Language Processing (NLP), like the Gated mechanism which presented by Dauphin [48], and Bidirectional Gated Recurrent Units (BGRU) [49], Flor proposed Gated-CNN-BGRU model which involves a few parameters and achieves a low error rate in the text recognition process. There are more contributions with the following aspects, handling long sentences with different styles, noise, and variations, even in the case of limited training data. Improving the recognition results through the new

Gated-CNN-BGRU architecture. Reducing the number of trainable parameters, making the model smaller and with lower computational cost.

Abdallah [44], aiming at improving HTR model accuracy in handwritten Cyrillic text recognition task. This model's architecture consists of 4 main parts: encoder, attention, decoder, and CTC. An encoder part consists of 5 convolutional blocks, each of which is made up of a convolutional layer, Parametric Rectified Linear Unit (PReLU) activator [50] with Batch Normalization, and gated convolutional layer [40]. The Dropout technique is also applied at the input of some convolutional layers (with a dropout probability of 0.5) to reduce the overfitting issue [51]. As an attention part of this model's architecture, Bahdanau attention mechanism is used [52]. Generally, attention mechanisms encode an input sentence by segmenting it into a fixed number of parts so they can be processed later by a decoder. Bahdanau attention mechanism enabled attention mechanism to focus on relevant parts of an input sentence, rather than hard segmenting it. The key role of Bahdanau attention mechanism applied between an encoder and decoder is to provide a richer encoding of the input sequence.

Chapter 6. Conclusion and future work

6.1 Summary

In this research work, firstly, we have built the Kazakh Offline Handwritten Text Dataset (KOHTD). The dataset can serve as a basis for research in handwriting recognition. This consists of a large collection of exam papers filled by students at Satbayev University and Al-Farabi Kazakh National University. Secondly, we propose Genetic Algorithm (GA) based on random enumeration of a parameter. By random here we mean that in order to find a solution using GA, random changes were applied to the current solutions to generate new ones. GA is based on Darwin's theory of evolution. It is a slow, gradual process that works by making small and slow changes. In addition, GA is slowly making small changes to its decisions until it gets the best solution.

Finally, this research work tried to solve a handwritten Kazakh interpretation task using well-known RNN models, such as Flor, Abdallah, Bluche, and Puigcerver HTR models. These RNN models were first quantitatively evaluated against each other to select the best performing one. According to experiments, the Flor HTR model demonstrated the highest recognition rate overall. As experiment results proved, Flor model demonstrated the best performance with 6.52 % character error rate (CER), 24.52% word error rate (WER) and 26.98% SER for the test dataset.

6.2 Future work

As future work, we will present information about the gender to use for classification of gender based on handwriting and writer identification. The physical look of handwriting reveals the relationship between gender and handwriting.

REFERENCES

1. R. Fakoor, F. Ladhak, A. Nazi, and M. Huber, "Using deep learning to enhance cancer diagnosis and classification," in Proceedings of the international conference on machine learning, vol. 28. ACM, New York, USA, 2013, pp. 3937–3949.
2. A. Abdallah, M. Kasem, M. A. Hamada, and S. Sdeek, "Automated questionanswer medical model based on deep learning technology," in Proceedings of the 6th International Conference on Engineering & MIS 2020, 2020, pp. 1–8.
3. M. A. Hamada, A. Abdallah, M. Kasem, and M. Abokhalil, "Neural network estimation model to optimize timing and schedule of software projects," in 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST). IEEE, 2021, pp. 1–7.
4. A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "A fast matching algorithm for graph-based handwriting recognition," in International Workshop on Graph-Based Representations in Pattern Recognition. Springer, 2013, pp. 194–203.
5. H. Liu and X. Ding, "Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes," in Eighth International Conference on Document Analysis and Recognition (ICDAR'05). IEEE, 2005, pp. 19–23.
6. F. Zamora-Martinez, V. Frinken, S. Espan˜a-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke, "Neural network language models for off-line handwriting recognition," Pattern Recognition, vol. 47, no. 4, pp. 1642–1652, 2014.
7. D. Nurseitov, K. Bostanbekov, D. Kurmankhojayev, A. Alimova, A. Abdallah, and R. Tolegenov, "Handwritten kazakh and russian (hkr) database for text recognition," Multimedia Tools and Applications, pp. 1–23, 2021.
8. S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, M. Alshayeb, M. T. Parvez, V. Maˆrgner, and G. A. Fink, "Khatt: An open arabic offline handwritten text database," Pattern Recognition, vol. 47, no. 3, pp. 1096–1112, 2014.
9. M. T. Parvez and S. A. Mahmoud, "Arabic handwriting recognition using structural and syntactic pattern attributes," Pattern Recognition, vol. 46, no. 1, pp. 141–154, 2013.
10. J. Jomy, K. Balakrishnan, and K. Pramod, "A system for offline recognition of handwritten characters in malayalam script," International Journal of Image, Graphics and Signal Processing, vol. 5, no. 4, p. 53, 2013.
11. S. Das and S. Banerjee, "An algorithm for japanese character recognition," International Journal of Image, Graphics and Signal Processing, vol. 7, no. 1, p. 9, 2014.
12. "Sozdikqor: Sh. shayahmetov atyndagy «til-qazyna» ulttyq gylymipraktikalyq ortalygy," <https://sozdikqor.kz>, 2021, Accessed: 2021-09-07.
13. U.-V. Marti and H. Bunke, "A full english sentence database for off-line handwriting recognition," in Proceedings of the Fifth International Conference on

Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318). IEEE, 1999, pp. 705–708.

14. ———, “The iam-database: an english sentence database for offline handwriting recognition,” *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

15. H. Bunke, S. Bengio, and A. Vinciarelli, “Offline recognition of unconstrained handwritten texts using hmms and statistical language models,” *IEEE transactions on Pattern analysis and Machine intelligence*, vol. 26, no. 6, pp. 709–720, 2004.

16. P. Dreuw, P. Doetsch, C. Plahl, and H. Ney, “Hierarchical hybrid mlp/hmm or rather mlp features for a discriminatively trained gaussian hmm: a comparison for offline handwriting recognition,” in *2011 18th IEEE International Conference on Image Processing*. IEEE, 2011, pp. 3541–3544.

17. B. Gatos, I. Pratikakis, and S. J. Perantonis, “Hybrid off-line cursive handwriting word recognition,” in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2. IEEE, 2006, pp. 998–1002.

18. D. Salvi, J. Zhou, J. Waggoner, and S. Wang, “Handwritten text segmentation using average longest path algorithm,” in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 2013, pp. 505–512.

19. R. P. dos Santos, G. S. Clemente, T. I. Ren, and G. D. Cavalcanti, “Text line segmentation based on morphology and histogram projection,” in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 651–655.

20. A. Bensefia, T. Paquet, and L. Heutte, “A writer identification and verification system,” *Pattern Recognition Letters*, vol. 26, no. 13, pp. 2080–2092, 2005.

21. Z. A. Daniels and H. S. Baird, “Discriminating features for writer identification,” in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1385–1389.

22. E. Augustin, M. Carrée, E. Grosicki, J.-M. Brodin, E. Geoffrois, and F. Prêteux, “Rimes evaluation campaign for handwritten mail processing,” in *International Workshop on Frontiers in Handwriting Recognition (IWFHR'06)*, 2006, pp. 231–235.

23. C. Kermorvant and J. Louradour, “Handwritten mail classification experiments with the rimes database,” in *2010 12th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2010, pp. 241–246.

24. L. Guichard, A. H. Toselli, and B. Couasnon, “Handwritten word verification by svm-based hypotheses re-scoring and multiple thresholds rejection,” in *2010 12th International Conference on Frontiers in Handwriting Recognition*. IEEE, 2010, pp. 57–62.

25. I. Siddiqi and N. Vincent, “Text independent writer recognition using redundant writing patterns with contour-based orientation and curvature features,” *Pattern Recognition*, vol. 43, no. 11, pp. 3853–3865, 2010.

26. M. Pechwitz, S. S. Maddouri, V. Ma'rgner, N. Ellouze, H. Amiri et al., "Ibn/enit-database of handwritten arabic words," in Proc. of CIFED, vol. 2. Citeseer, 2002, pp. 127–136.
27. T. Su, T. Zhang, and D. Guan, "Corpus-based hit-mw database for offline recognition of general-purpose chinese handwritten text," International Journal of Document Analysis and Recognition (IJ DAR), vol. 10, no. 1, p. 27, 2007.
28. R. Safabakhsh and P. Adibi, "Nastaaligh handwritten word recognition using a continuous-density variable-duration hmm," Arabian Journal for Science and Engineering, vol. 30, no. 1, pp. 95–120, 2005.
29. M.-Y. Chen, A. Kundu, and S. N. Srihari, "Variable duration hidden markov model and morphological segmentation for handwritten word recognition," IEEE transactions on image processing, vol. 4, no. 12, pp. 1675–1688, 1995.
30. J. H. AlKhateeb, J. Ren, J. Jiang, and H. Al-Muhtaseb, "Offline handwritten arabic cursive text recognition using hidden markov models and re-ranking," Pattern Recognition Letters, vol. 32, no. 8, pp. 1081–1088, 2011.
31. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.
32. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
33. A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates et al., "Deep speech: Scaling up end-to-end speech recognition," arXiv preprint arXiv:1412.5567, 2014.
34. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104–3112.
35. N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in International conference on machine learning. PMLR, 2015, pp. 843–852.
36. A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in Proceedings of the 23rd international conference on Machine learning, 2006, pp. 369–376.
37. R. R. Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. C. Papat, "A scalable handwritten text recognition system," in 2019 International Conference on Document Analysis and Recognition (ICDAR). IEEE, 2019, pp. 17–24.
38. S. Espana-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. ZamoraMartinez, "Improving offline handwritten text recognition with hybrid hmm/ann models," IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 4, pp. 767–779, 2010.
39. F. Abdurahman, E. Sisay, and K. A. Fante, "Ahwr-net: offline handwritten amharic word recognition using convolutional recurrent neural network," SN Applied Sciences, vol. 3, no. 8, pp. 1–11, 2021.

40. T. Bluche and R. Messina, “Gated convolutional recurrent neural networks for multilingual handwriting recognition,” in 2017 14th IAPR international conference on document analysis and recognition (ICDAR), vol. 1. IEEE, 2017, pp. 646–651.

41. J. Puigcerver, “Are multidimensional recurrent layers really necessary for handwritten text recognition?” in 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1. IEEE, 2017, pp. 67–72.

42. “Telegram: Telegram is a cloud-based mobile and desktop messaging app with a focus on security and speed,” <https://web.telegram.org>, 2021, Accessed: 2021-09-07.

43. A. F. de Sousa Neto, B. L. D. Bezerra, A. H. Toselli, and E. B. Lima, “Htrflor: a deep learning system for offline handwritten text recognition,” in 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). IEEE, 2020, pp. 54–61.

44. A. Abdallah, M. Hamada, and D. Nurseitov, “Attention-based fully gated cnn-bgru for russian handwritten text,” *Journal of Imaging*, vol. 6, no. 12, p. 141, Dec 2020. [Online]. Available: <http://dx.doi.org/10.3390/jimaging6120141>

45. V. Frinken and H. Bunke, *Continuous Handwritten Script Recognition*. London: Springer London, 2014, pp. 391–425.

46. M. Abadi, A. Agarwal, P. Barham, E. Brevdo et al., “Tensorflow: Largescale machine learning on heterogeneous distributed systems,” *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: <http://arxiv.org/abs/1603.04467>

47. G. Hinton, N. Srivastava, and K. Swersky, “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent,” Cited on, vol. 14, no. 8, 2012.

48. Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language modeling with gated convolutional networks,” in *International conference on machine learning*. PMLR, 2017, pp. 933–941.

49. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

50. K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

51. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

52. D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations*, San Diego, CA, USA, May 7-9, 2015, Conference Track

53. *Proceedings*, 2015. [Online]. Available: <http://arxiv.org/abs/1409.0473>