VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ RUČNĚ PSANÉHO TEXTU POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ

DEEP NETWORKS FOR HANDWRITING RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **LUKÁŠ RICHTARIK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Richtarik Lukáš, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Rozpoznávání ručně psaného textu pomocí hlubokých neuronových sítí**
Deep Networks for Handwriting Recognition
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy konvolučních sítí a rozpoznávání ručně psaného textu.
2. Vytvořte si přehled o současných metodách rozpoznávání ručně psaného textu pomocí konvolučních sítí.
3. Vyberte nebo navrhněte metodu aplikovatelnou na rozpoznávání ručně psaného textu.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Krizhevsky, A., Sutskever, I. and Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
- http://www.image-net.org/challenges/LSVRC/2013/results.php

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 6. listopadu 2019

Abstrakt

Táto práca sa zaoberá problematikou rozpoznávania ručne písaného textu pomocou hlbokých neurónových sietí. Je zameraná na použitie metódy sequence to sequence pomocou enkóder-dekóder modelu. Jej súčasťou je aj návrh enkóder-dekóder modelu pre rozpoznávanie ručne písaného písma používajúceho transformer namiesto rekurentných neurónov a sada experimentov, ktoré na ňom boli vykonané.

Abstract

The work deals with the issue of handrwritten text recognition problem with deep neural networks. It focuses on the use of sequence to sequence method using encoder-decoder model. It also includes design of encoder-decoder model for handwritten text recognition using a transformer instead of recurrent neurons and a set of experiments that were performed on it.

Klíčová slova

Rozpoznávanie textu, OCR, HTR, Neurónové siete, Sequence to sequence, Konvolučné neurónové siete, Transformer, Hlboké učenie

Keywords

Text recognition, OCR, HTR, Neural networks, Sequence to sequence, Convolutional neural networks, Transformer, Deep learning

Citace

RICHTARIK, Lukáš. *Rozpoznávání ručně psaného textu pomocí hlubokých neuronových sítí.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. MICHAL HRADIŠ, Ph.D.

Rozpoznávání ručně psaného textu pomocí hlubokých neuronových sítí

Prohlášení

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Michala Hradiša Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Lukáš Richtarik
3. června 2020

Poděkování

Rád by som poďakoval pánu Ing. Michalovi Hradišovi Ph.D. za vedenie tejto diplomovej práce, odbornú pomoc a cenné rady.

Obsah

Kapitola 1

Úvod

Veľa odborných textov existuje len v tlačenej alebo písanej podobe a preto je vyvíjané úsilie o úplnú digitalizáciu takýchto dokumentov. Je nemožné súčasný objem týchto textov prevádzať do digitálnej podoby manuálne. Rozpoznávanie ručne písaného textu (handwritten text recognition - HTR) je automatický proces, kedy je text obsiahnutý v naskenovaných dokumentoch prevádzaný na digitálny text. Vďaka tomu je prístup k týmto dokumentom efektívnejší, je možné ich jednoduchšie triediť a vyhľadávať v nich informácie, sprístupniť ich širokej verejnosti, či pomôcť zrakovo postihnutým osobám.

Rozličné typy dokumentov, obrovské množstvo odlišných štýlov písania, rôzne abecedy a jazyky robia z rozpoznávania ručne písaného textu stále nevyriešený problém. Hlboké neurónové siete prinášajú veľmi dobré výsledky v problémoch rozpoznávania reči, strojového prekladu, či klasifikácie obrázkov. V posledných rokoch bolo predstavených množstvo úspešných modelov neurónových sietí, ktoré dokážu rozpoznávať ručne písaný text.

Novodobé prístupy riešenia HTR problémov založené na enkóder-dekóder modeloch patria medzi súčasné state-of-the-art. Táto práca je zameraná práve na nich. Obsahuje stručný úvod do problematiky HTR a taktiež popisuje návrh systému rozpoznávania ručne písaného textu zameraného na rozpoznávanie textu vo vysegmentovaných riadkoch. Na to používa enkóder-dekóder model, nazývaný transformer, spolu s konvolučnou neurónovou sieťou. Transformer je založený na attention mechanizme a nepotrebuje pre svoju funkčnosť rekurentné neuróny. To mu dáva možnosť vyššej miery paralelizácie a teda rýchlejšieho trénovania siete. V praxi dosahuje porovnateľné výsledky so sequence to sequence modelmi používajúcimi rekurentné neurónové siete. Súčasťou tejto práce je aj porovnanie úspešnosti modelov založených na transformeroch a na enkóder-dekóder modeloch, používajúcich rekurentné neuróny.

Táto práca je členená na 4 základné časti. Druhá kapitola je venovaná princípom rozpoznávania textu. Popisuje jednotlivé typy OCR systémov, princíp ich fungovania a vyhodnocovania. Zaoberá sa taktiež stručnou históriou vývoja OCR systémov a obsahuje prehľad najznámejších datasetov z oblasti rozpoznávania textu.

Tretia kapitola sa už zameriava konkrétne na metódu sequence to sequence. Popisuje klasický návrh enkóder-dekóder modelu používajúceho rekurentné neuróny a princíp attention mechanizmu. Ďalej je v ňom popísaný relatívne nový enkóder-dekóder model ktorý nepoužíva rekurentné neuróny, nazývaný transformer.

V štvrtej kapitole je opísaný návrh dvoch enkóder-dekóder modelov, ktoré slúžia na rozpoznávanie ručne písaného textu. Jeden je zložený z konvolučnej siete a rekurentných neurónov a druhý z konvolučnej siete a transformera.

V poslednej kapitole sa nachádza prehľad experimentov s navrhnutými modelmi a ich vyhodnotenie. Primárnym cieľom experimentov bolo preskúmať úspešnosť modelu s transformerom pri rozpoznávaní ručne písaného písma. Sekundárnym cieľom bolo porovnanie transformeru s modelom používajúcim rekurentné neuróny. Táto kapitola tiež obsahuje stručné zhrnutie vykonaných experimentov a návrhy na budúce vylepšenia modelov.

Kapitola 2

Rozpoznávanie textu

Rozpoznávanie textu je prevod textu z písanej alebo tlačenej podoby do digitálnej podoby. Táto problematika sa označuje skratkou OCR (Optical Character Recognition). Existuje mnoho postupov a metód riešenia problému rozpoznávania textu. V súčasnosti sa do popredia dostávajú čoraz častejšie systémy založené na umelej inteligencii a deep learningu. V nasledujúcich kapitolách je spísaná história vývoja OCR systémov, stručný popis fungovania moderných OCR systémov a ich využitie. Záver kapitoly sa venuje vyhodnocovaniu úspešnosti OCR systémov a popisuje existujúce dátové sady, vhodné na trénovanie OCR modelov.

V súčasnosti je rozpoznávanie tlačenej latinskej abecedy považované za vyriešený problém. Dnešné OCR systémy už dosahujú 99% úspešnosť pri rozpoznávaní takýchto textov [18]. Ostatné oblasti ako rozpoznávanie ručne písaného písma, alebo rozpoznávanie textov napísaných v jazykoch s veľkým množstvom znakov v abecede ako čínština či arabčina sú stále predmetom aktívneho výskumu. Tieto oblasti označujeme skratkou HTR (Handwritten Text Recognition) a táto práca sa venuje prevažne tejto oblasti.

2.1 Historický vývoj

Prvé pokusy o optické rozpoznávanie znakov pochádzajú ešte z doby pred prvými počítačmi. Prvý mechanický stroj slúžiaci na rozpoznávanie znakov vytvoril Gustav Tauschek v roku 1929. S rozvojom počítačových systémov a potreby vyhľadávať v množstve dokumentov, ktorých počet sa stále zväčšoval, vnikla potreba spracovávania textov počítačom. Ďalším míľnikom bol rok 1954, kedy americký magazín Reader's Digest prišiel so zariadením, ktoré dokázalo prevádzať tlačený text na dierne štítky. Tie mohli byť ďalej spracovávané počítačom [43].

Za prvú generáciu OCR systémov sa používajú OCR systémy z rokoch 1960 až 1965. Spočiatku dokázali rozpoznávať len určitý jeden font. Neskôr vznikli ďalšie OCR ktoré dokázali rozpoznať viacero fontov, ale tento počet bol obmedzený tým, že fungovali na princípe porovnávania znakov s uloženými vzormi [11].

Druhá generácia OCR prišla koncom 60. rokov. Tieto systémy dokázali rozpoznávať bežne tlačené texty a do určitej miery zvládali aj ručne písaný text. V tomto období boli navrhnuté špeciálne fonty OCR-A a OCR-B 2.1, ktoré boli určené práve na rozpoznávanie tlačeného písma. OCR-A bolo používané hlavne v Amerike a OCR-B v Európe. OCR systémy toto písmo dokázali veľmi dobre rozpoznávať a preto sa ich využitie šírilo ďalej [11].

Obrázek 2.1: Ukážka fontov OCR-A a OCR-B (prevzaté z [2]).

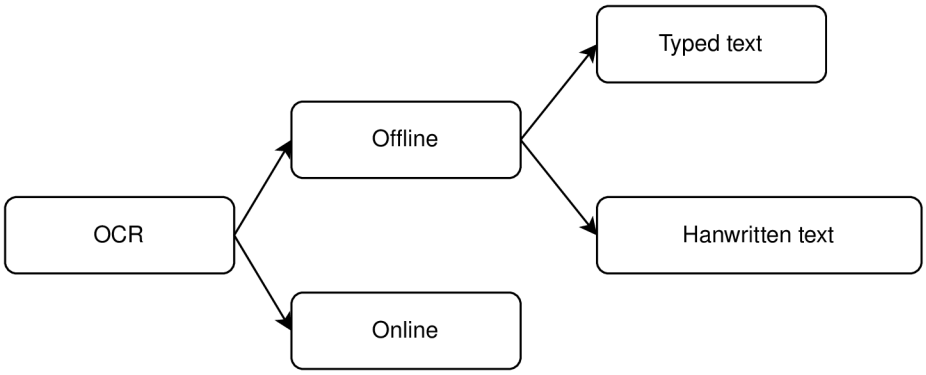
Do tretej generácie sa zaraďujú OCR systémy, ktorých snahou bolo rozpoznávať nekvalitné a ručne písané dokumenty. Tieto systémy vznikali v období druhej polovice 70. rokov. Ich cieľom bolo tiež zníženie nákladov a zvýšenie efektivity. Nebolo nutné používať už špeciálne fonty. Tieto OCR dokázali rozpoznávať aj dokumenty napísané na písacích strojoch, ktorých bolo v tej dobe obrovské množstvo [11].

Po roku 1990 začali vznikať systémy založené na umelej inteligencii. Vďaka rastúcemu výpočtovému výkonu bolo možné používať zložitejšie algoritmy, ktoré používali techniky ako skryté markovské modely, umelé neurónové siete, či fuzzy rozhodovanie. Rozpoznávanie textu sa stalo súčasťou problematiky počítačového videnia. Súčasné OCR systémy dokážu takmer bezchybne rozpoznať klasický tlačený text. Oblasťou výskumu sa stalo rozpoznávanie ručne písaného textu, rozpoznávanie textu s nelatinskou abecedou, alebo rozpoznávanie textu v nehomogennom prostredí ako napríklad ŠPZ, poškodené dokumenty či nápisy na budovách [11].

2.2 OCR

OCR (Optical Character Recognition) je akýkoľvek proces ktorý dokáže v nasnímanom texte v podobe obrázku rozpoznať jednotlivé znaky a následne ich previezť do požadovanej formy. Pomocou OCR je možné rozpoznávať text na naskenovaných dokumentoch alebo fotografiách rôznych textových plôch, ako napríklad nápisy na budovách, poznávacie značky vozidiel či dopravné značky. Rozpoznávanie tlačeného textu je jednou z najjednoduchších úloh pre OCR systémy. Problémom sú hlavne ručne písané texty, alebo texty umiestnené v scéne na fotografiách. Takéto dokumenty alebo fotografie zvyknú byť nekvalitné. Pri ručne písaných dokumentoch je problémom hlavne nekvalitný sken dokumentu, špecifický rukopis, sklon písma, štýl písania alebo škrtanie. Písané písmo tiež obsahuje rôzne veľké medzery medzi znakmi, alebo sa jednotlivé znaky môžu prekrývať [50].

OCR je možné rozdeliť do dvoch kategórií - online a offline. Pri online systémoch sú vstupné informácie získavané z real-time senzorov. Prevod textu do digitálnej podoby prebieha ešte v procese písania. Vstupnou informáciou pre rozpoznávanie je pozícia pera ako funkcia času. Online OCR sa využíva napríklad v tabletoch a smartfónoch. Výhodou online systémoch oproti offline je napríklad vyššia miera interaktivity a nevýhodou je, že online systémy nedokážu spracovať vopred naskenovaný text. Potrebujú k svojej funkčnosti dynamické informácie zaznamenané pri písaní konkrétneho textu.



Obrázek 2.2: Rozdelenie OCR systémov

V offline OCR systémoch sú používané statické informácie z obrázkov. Takéto systémy sa delia na systémy ktoré rozpoznávajú tlačený, alebo ručne písaný text. Problémom rozpoznávania ručne písaného textu je rozličný štýl a pravopis nasnímaných textov a taktiež typ písacieho nástroja a predmetu na ktorom je text napísaný. Výhodou je to, že pomocou offline OCR systémov je možné rozpoznávať texty aj niekoľko rokov po ich napísaní. To sa uplatňuje napríklad pri rozpoznávaní historických dokumentov [28]. Na obrázku 2.2 je znázornené rozdelenie OCR systémov.

Rozpoznávanie textu pozostáva z niekoľkých krokov. Vstupný obrázok je najprv predspracovaný a následne prebehne segmentácia. Súčasťou predspracovania sú kroky ako noise removing, binarization, edge detection a dilatation and filling [28]. Segmentácia je proces, kedy sú z dokumentu vyextrahované len určité časti, ktoré sú neskôr spracovávané. Môže prebiehať na úrovni znakov, slov, celých riadkov, alebo celých blokov textu. Na segmenty je potom aplikovaná extrakcia príznakov (feature extraction), ktorej výstupom je vektor príznakov (feature vector). Feature extraction je jeden z najdôležitejších krokov pre dosiahnutie vysokej úspešnosti rozpoznávania. Ide o extrakciu dôležitých vlastností a informácií z pixelov obrázka, ktoré sú potrebné pre úspešné rozpoznávanie. Pri OCR systémoch sa extrakcia príznakov delí na 2 hlavné smery - holistická a analytická. Pri holistickej extrakcii príznakov je každé slovo považované za triedu a rozpoznávanie prebieha na úrovní slov. Analytická extrakcia môže prebiehať na úrovní slov, písmen, alebo častí písmen. Niektoré z najpoužívanejších metód pre feature extraction sú napríklad template matching, deformable templates, unitary image transforms, graph description, projection histograms, contour profiles, zoning [36]. Pred extrakciou príznakov môže byť na extrahované segmenty aplikovaná normalizácia. Normalizáciou sú odstránené malé rozdiely medzi rovnakými znakmi, ktoré by mohli skomplikovať kategorizáciu znakov a mohli by znížiť mieru rozpoznávania. Ide napríklad o sklon alebo hrúbku písma [28]. Posledným krokom je samotné rozpoznávanie, ktorému je venovaná nasledujúca kapitola.

2.3 Proces rozpoznávania

Rozpoznávanie ručne písaného textu je vďaka variabilite štýlu písania, rozličnosti jazyka a abecedy stále zložitým problémom. Ku rozpoznávaniu (kategorizácií) jednotlivých znakov slúžia klasifikátory, ktoré sa delia podľa spôsobu učenia.

Klasifikátory založené na učení s učiteľom (supervised learning) sa učia pomocou dátovej sady, ktorá musí obsahovať trénovacie dáta vo forme vstupu a očakávaného výstupu. Patria sem napríklad neurónové siete, skryté Markové modely, alebo SVM [28].

Klasifikátory založené na učení bez učiteľa (unsupervised learning) nepotrebujú trénovacie dáta. Tieto klasifikátory zaraďujú znaky do tried podľa štatistických údajov a ich polohe v priestore, pričom znaky ktoré sa nachádzajú v priestore blízko seba, klasifikujú do jednej triedy. Patria sem napríklad k-means, alebo zhlukovanie [28].

Jedno z prvých použití konvolučnej siete bolo práve rozpoznávanie ručne písaných číslic z MNIST datasetu. Použil ju v roku 1998 Yann LeCun a spol. [25]. V tej dobe sa táto metóda stala najúspešnejšou v rozpoznávaní znakov. Sieť sa volala LeNet-5 a obsahovala 2 konvolučné vrstvy za ktorými nasledovali average pooling vrstvy a následne sa napájala posledná konvolučná vrstva. Za ňou sa nachádzali 2 plne prepojené vrstvy a softmax vrstva.

Konvolučné siete majú nevýhodu v tom, že nedokážu pracovať s dlhšími sekvenciami. Preto bolo nutné použiť na rozpoznávanie dlhších sekvencií rekurentné neuróny. V roku 2009 Alex Graves a kolektív [13] použili na rozpoznávanie písma rekurentné neuróny. Ich architektúra pozostávala z obojsmernej LSTM neurónovej siete a objektívnej funkcie CTC (connectionist temporal classification). Tejto sieti sa podarilo prekonať dovtedy najúspešnejšie systémy založené na skrytých Markových modeloch.

Ďalší vývoj smeroval k modelom ktoré používali konvolučné neurónové siete na extrakciu mapy príznakov, a tá bola ďalej spracovávaná rekurentnou neurónovou sieťou. Výsledná klasifikácia prebiehala pomocou CTC [7, 33, 46].

Modely používajúce konvolučné a rekurentné neurónové siete spolu s CTC majú nevýhodu v tom, že len mapujú vstupný obrázok na text fixnej dĺžky. Túto nevýhodu prekonávajú modely založené na enkóder-dekóder architektúre [42], kedy sa na HTR problém pozerá ako na problém sequence to sequence. Vstupom do enkódera je teda sekvencia príznakov vyextrahovaná zo vstupného obrázku a výstupom dekódera sekvencia znakov, alebo slov. Sequence to sequence modely sú flexibilnejšie, dokážu modelovať jazykové štruktúry a spolu s attention mechanizmom [1] sa stali novým state-of-the-art v rozpoznávaní ručne písaného textu. Kapitola 3 sa podrobnejšie venuje problematike sequence to sequence modelov.

2.4 Využitie OCR

OCR systémy v súčasnosti nachádzajú široké uplatnenie. Používajú sa v bankovníctve, zdravotníctve či súdnictve [23]. Digitalizácia tlačených formulárov uľahčuje prácu pri vyhľadávaní, editovaní a archivovaní. Digitalizované dokumenty tiež môžu byť ukladané v zdieľaných databázach s cieľom vyhnúť sa duplicite dát. V prípade, že zamestnanec potrebuje vyhľadať na akejkoľvek pobočke klienta, stačí zadať jeho meno do systému a nemusí prehľadávať záznamy manuálne, čím sa ušetrí množstvo času.

Digitalizácia knižníc a mestských archívov je v súčasnosti dôležitý proces, ktorý sprístupňuje obrovské množstvo informácii ktoré bolo doteraz obsiahnuté iba v tlačenej alebo písanej forme. Týmto spôsobom je možné uchovávať aj historické knihy, spisy, alebo hudobné diela. V týchto dokumentoch je potom možné ľahko vyhľadávať podľa kľúčových slov.

OCR v kombinácii s text-to-speech systémami môžu uľahčiť prácu zrakovo postihnutým ľuďom. Naskenovaný a rozpoznaný text môže byť transformovaný na reč, čo pomôže nevidiacim k lepšej orientácii v dnešnom modernom svete. [38].

Ďalším významným využitím OCR je rozpoznávanie evidenčných čísel vozidiel [34]. Rozpoznanie evidenčného čísla z fotografie zhotovenej policajnou alebo dopravnou kamerou

umožnuje rýchle vyhľadávanie v registri vozidiel alebo v zozname kradnutých áut. Tento systém je používaný aj pri automatickom výbere mýta na cestách.

2.5 Vyhodnocovanie

Výstupy OCR systémov, tak ako aj systémov rozpoznávania reči, či strojového prekladu, produkujú množstvo chýb - vynechané a nadbytočné slová, alebo nesprávne napísané slová. Aby sa zistila objektívna chyba systému, nezávislá na veľkosti textu, je potrebné počet chýb normalizovať podľa dĺžky očakávaného obsahu. Kvocient medzi počtom chýb a dĺžkou textu sa nazýva chybovosť (error rate). Najpoužívanejšími výpočtami chybovosti v OCR systémoch sú metriky CER a WER [6].

Character error rate (CER) je metrika počítaná ako minimálny počet operácií, potrebných na transformáciu vstupného textu na požadovaný text. Ide o Levenstheinovú vzdialenosť, kedy rozlišujeme operácie: substitúcia, zmazanie znaku a pridanie znaku. Čím vyššia je hodnota CER, tým rozdielnejšie sú 2 texty. V niektorých aplikáciách je častou chybou aj zámena dvoch susediacich znakov. Preto je možné brať do úvahy aj štvrtú operáciu - zámenu (swap). Takýto výpočet sa potom nazýva Damerau-Levenshteinová vzdialenosť. CER je počítaná vzťahom

$$CER = (s + i + d)/n, \tag{2.1}$$

kde hodnota s je počet nesprávnych znakov, i počet nadbytočných znakov, d počet vynechaných znakov a n celkový počet znakov. Tým, že sa v niektorých prípadoch môže stať, že celkový počet chýb v texte je väčší ako dĺžka textu, môže CER nadobudnúť hodnotu vyššiu ako 100%. Preto sa niekedy používa normalizácia CER, kedy je počet všetkých chýb delený súčtom (i, s, d, c), kde i, s, a d je počet operácií a c je počet správnych znakov.

Word error rate (WER) je najčastejšie používaná metrika pre výpočet chybovosti v ASR (automatic speech recognition) a OCR systémoch, založená na chybovosti v jendotlivých slovách. Je odvodená z výpočtu Levenshteinovej vzdialenosti. Vzťah na výpočet WER je:

$$WER = (s_w + i_w + d_w)/n_w, \tag{2.2}$$

kde hodnota s_w je počet slov v ktorých nastala chyba zmenou písmen, i_w je počet nadbytočných slov, d_w je počet vynechaných slov a n_w je počet slov pre ktoré platí vzťah $c_w = n_w - d_w - s_w$. Hodnota c_w predstavuje počet správnych slov. Pri výpočte WER a CER sa zámena veľkých písmen za malé a naopak, nepovažuje za chybu. Taktiež niekoľko bielych znakov umiestnených hneď za sebou sa považuje stále len za jednu medzeru.

Hodnota WER pre danú úlohu je zvyčajne vyššia ako CER. Je to spôsobené tým, že relatívne malý počet nesprávnych znakov dokáže spôsobiť väčšie chyby na úrovni slov (narastá veľkosť s_w). Napriek tomu je medzi obidvomi metrikami očakávaná vysoká korelácia a preto sa obidve používajú na vyhodnocovanie OCR systémov [6].

Pri vyhodnocovaní niektorých ASR a OCR systémov sa používa metrika SER (Sentence error rate). Určuje koľko percent viet obsahuje chybu. Hodnota SER sa vypočíta ako

$$SER = e/n, \tag{2.3}$$

kde e je počet viet, ktoré obsahujú chybu a n je počet všetkých viet [16]. Metriky WER a SER v mnohých prípadoch korelujú, ale nemusí to byť vždy tak.

Názov datasetu	Typ obrázkov	Jazyk	Počet obrázkov
NIST	znaky a číslice	-	810 000
MNIST	číslice	-	70 000
HIT-OR3C	znaky a číslice	čínština	909 818
MADCAT	dokumenty	čínština a arabčina	444 284
IAM	slová a malé skupiny slov	angličtina	115 320
IAM	vysegmentované riadky	angličtina	13 353
111 let českého dopisu	vysegmentované riadky	čeština	115 547
České dokumenty	vysegmentované riadky	čeština	443 868

Tabulka 2.1: Prehľad dátových sád, vhodných pre trénovanie modelov, určených na rozpoznávanie ručne písaného textu.

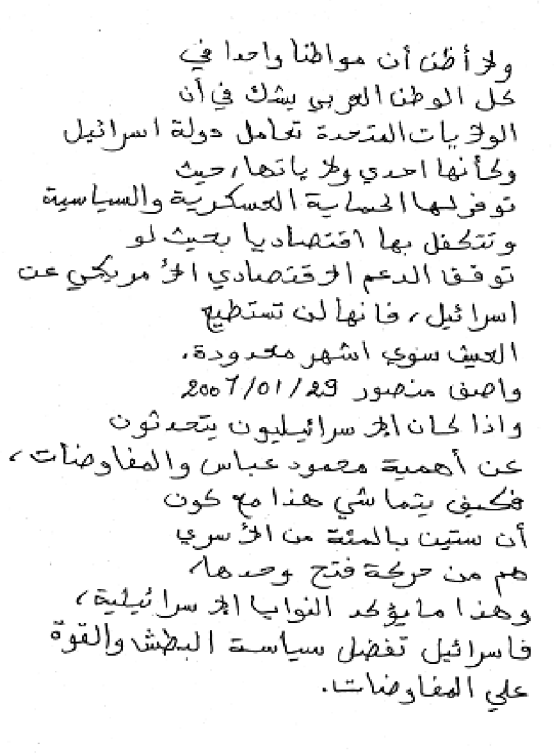
2.6 Dátové sady

V tejto kapitole sú popísané najčastejšie používané datasety pre rozpoznávanie ručne písaného textu pomocou strojového učenia. Pri OCR rozlišujeme 3 oblasti výskumu - rozpoznávanie offline písaného písma, online písaného písma a strojového tlačeného písma. Podľa toho je možné rozlíšiť aj jednotlivé typy datasetov, ktoré tieto písma obsahujú. Podľa spôsobu anotácie môžeme ešte rozlišovať, či daný dokument je anotovaný podľa znakov, slov, riadkov alebo celých blokov textu [14]. Kvalita a veľkosť datasetu zásadne ovplyvňuje úspešnosť natrénovaných modelov. V tabuľke 2.1 sa nachádza prehľad datasetov vhodných na offline rozpoznávanie ručne písaného textu.

NIST. Dataset NIST obsahuje trénovacie a testovacie dáta pre rozpoznávanie ručne písaných, malých a veľkých písmen a číslic. Obsahuje 810 000 obrázkov znakov od 3 600 pisateľov, spolu s anotáciami k týmto obrázkom. Obrázky majú veľkosť 28x28 pixelov.

MNIST. Dataset MNIST je rozdelený na trénovaciu a testovaciu sadu. Trénovacia sada obsahuje 60 000 ručne písaných číslic. Testovacia sada pozostáva z 10 000 číslic. Číslice v datasete boli normalizované na veľkosť 20x20 pixelov a uložené ako obrázky fixnej veľkosti 28x28 pixelov v odtieňoch sivej. MNIST je súčasťou väčšieho datasetu NIST, ktorý bol popísaný v predchádzajúcej kapitole [51].

HIT-OR3C. Harbin Institute of Technology Opening Recognition Corpus for Chinese Characters (HIT-OR3C) je databáza čínskych ručne písaných číslic a znakov. V tomto datasete sú dostupné informácie pre online aj offline rozpoznávanie znakov. Znaky boli zaznamenávané na tablete na tom určenom, pomocou programu na zaznamenávanie ručne písaného písma - OR3C Toolkit. Dataset je organizovaný do 5 sád, pričom 4 sady obsahujú záznamy znakov a jedna sada záznamy dokumentov. 4 sady znakov obsahujú 832 650 záznamov, klasifikovaných do 6 825 tried, zaznamenaných 122 respondentmi. Piata sada pozostáva z celých ručne písaných dokumentov s počtom 77 168 znakov klasifikovaných do 2 442 tried, zaznamenaných dvadsiatimi respondentmi. Dokopy obsahuje dataset 909 818 obrázkov.



Obrázek 2.3: Ukážka dokumentu z MADCAT datasetu z arabskej časti [9].

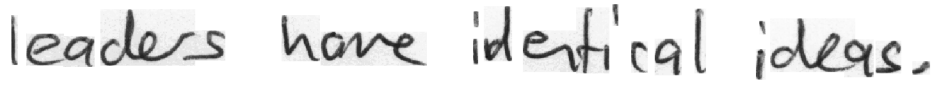
MADCAT. MADCAT (Multilingual Automatic Document Classification Analysis and Translation) je dataset ručne písaných dokumentov, anotovaných v súboroch v MADCAT XML a GEDI XML formáte [9]. Je zložený z troch častí, ktoré obsahujú spolu 420 000 dokumentov v arabčine a z jednej časti obsahujúcej 22 284 dokumentov v čínštine. Dokumenty sú členené do troch typov - newswire, weblog a newsgroup text. Boli prepisované ručne, podľa zadaných pokynov a štýlu písma. Rozlišuje sa v ňom rýchle, normálne a pomalé písanie. Respondenti písali podľa pokynov ceruzou alebo perom, na papier s predtlačenými riadkami alebo bez riadkov. Každý dokument bol naskenovaný vo vysokom rozlíšení (600dpi, greyscale). Naskenované dokumenty boli následne anotované - boli určené bounding boxy pre jednotlivé riadky v dokumente a pre všetky tokeny v riadkoch. Pre každý dokument bol vytvorený XML súbor vo formáte MADCAT, v ktorom sú uložené všetky potrebné informácie pre daný dokument - súradnice bounding boxov pre riadky aj tokeny, štýl písania, ID pisateľa, prepis v arabčine (alebo čínštine) pre jednotlivé tokeny aj pre celé riadky a takisto preklady v angličtine. Tento dataset nie je rozdelený na trénovaciu, testovaciu a validačnú sadu. Na obrázku 2.3 sa nachádza ukážka dokumentu z MADCAT datasetu.

Arabské písmo. Pri rozpoznávaní ručne písaného textu je rozpoznávanie nelatinských textov stále predmetom aktívneho výskumu. Takýmto písmom je aj arabské písmo. Vznik arabskej abecedy sa dátuje do 4. až 5. storočia n.l.

Arabské písmo sa v mnohom líši od latinského písma. Píše sa sprava doľava, a stránky v knihách za sebou nasledujú v opačnom poradí ako v knihách napísaných latinským písmom. Arabská abeceda má 28 písmen a mnoho diakritickým znamienok. Toto písmo sa zaraďuje medzi kurzívne písma - tlačená podoba tohto písma je zobrazovaná ako rukopis. V tomto

Názov sady	Počet riadkov	Počet autorov
Trénovacia	6 161	283
Testovacia	1 861	46
Validačná 1	900	43
Validačná 2	940	128
Dohromady	9 862	500

Tabulka 2.2: Rozdelenie IAM datasetu na disjunktné sady vysegmentovaných riadkov, určené pre vyhodnocovanie OCR systémov.



Obrázek 2.4: Ukážka vysegmentovaného riadku z IAM datasetu [30].

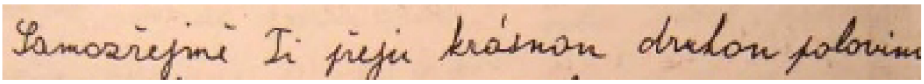
písme sa nerozlišujú veľké a malé písmena a jednotlivé znaky majú rozličné tvary vzhľadom na pozíciu v slove. Pri každom písmene sa rozlišuje jeho základný, počiatočný, stredový a koncový tvar [29].

IAM-OnDB. The IAM On-Line Handwriting Database je databáza viac ako 1 700 ručne písaných dokumentov v angličtine od 221 respondentov [30]. Pozostáva z 13 049 extrahovaných a anotovaných riadkov v online aj offline formáte. Dokumenty boli naskenované v rozlíšení 300 dpi a uložené ako grayscale PNG obrázky. Zozbierané údaje ako ID pisateľa, prepis textu, podmienky pri písaní, pohlavie a ďalšie, sú obsiahnuté v XML formáte. Tento dataset môže byť použitý napríklad na trénovanie a testovanie rozpoznávačov ručne písaného textu alebo na identifikáciu a verifikáciu pisateľa. Na obrázku 2.4 je ukážka vysegmentovaného riadku z IAM datasetu.

Pre túto dátovú sadu bol zadefinovaný task pre rozpoznávanie ručne písaného textu z vysegmentovaných riadkov. Tento task slúži pre porovnávanie úspešnosti jednotlivých OCR systémov. Rozdeľuje dataset na trénovaciu, testovaciu a 2 validačné sady. Tieto sady sú vzájomne disjunktné. V tabuľke 2.2 je znázornené rozdelenie datasetu na jednotlivé sady.

Dataset obsahuje tiež sadu ktorá obsahuje anotáciu podľa slov, kedy každé slovo je osobitný obrázok. Táto sada obsahuje vyše 100 000 obrázkov. Ako je možné vidieť na obrázku 2.4, každé písmeno bolo vyrezané z obrázka osobitne, preto sa okolo všetkých písmen nachádza sivá plocha. Kvôli tomu je vhodné obrázky z tohto datasetu najprv binarizovať.

111 let českého dopisu v korpusovém zpracování. 111 let českého dopisu v korpusovém zpracování [17] je dataset 2 000 ručne písaných súkromných listov, pochádzajúcich z rokov 1902-2012, ktoré zhromaždil a prepísal kolektív Zdeny Hladké na Masarykovej univerzite. Každý list bol písaný iným pisateľom z rôznych častí územia českej republiky a preto tento dataset dobre reprezentuje písmo 20. storočia v tejto krajine. Všetky listy boli uložené od priamych účastníkov korešpondenčného styku, alebo od ich dedičov ktorý súhlasili s ich zverejnením a použitím pre nekomerčné študijné účely. Členovia výskumnej skupiny PERO vysegmentovali vhodné riadky na rozpoznávanie z naskenovaných dokumentov a priradili k nim prepisy. Takto pripravenú sadu som využil v experimentoch pri trénovaní jednolitvých modelov neurónových sietí. Dokopy dataset obsahuje 115 547 riadkov ručne písaného textu



Obrázek 2.5: Ukážka vysegmentovaného riadku z datasetu 111 let českého dopisu v korpusovém zpracování [17].

Max. šírka	**224 px**	**600 px**
Trénovacie dáta	348 961	50 576
Validačné dáta	25 933	5 436
Testovacie dáta	12 962	-

Tabulka 2.3: Rozdelenie datasetu českých dokumentov, dodaného od výskumnej skupiny PERO na trénovacie, validačné a testovacie dáta. Dataset bol zložený z dvoch sád. Prvá obsahovala iba obrázky s maximálnou šírkou 224px. Druhá sada obsahovala rôzne veľké obrázky, so šírkou približne 600px. Pre obrázky z druhej sady som obmedzil maximálnu šírku práve na 600px.

spolu s ich prepismi. Na obrázku 2.5 je ukážka vysegmentovaného riadku z datasetu českých listov. Nedostatkom tejto sady je, že dokumenty boli digitalizované v horšej kvalite a často majú nízke rozlíšenie, pretože pôvodným cieľom bolo vytvoriť iba jazykový korpus.

Dataset českých dokumentov. Od výskumnej skupiny PERO som dostal dátovú sadu s ručne písanými textami. V práci ju budem ďalej označovať ako *dataset českých dokumentov*. Dataset obsahuje množstvo ručne písaných dokumentov, niektoré z nich pochádzajú z datasetu *111 let českého dopisu v korpusovém zpracování*. Túto dátovú sadu som dostal vo forme lmdb databáze, v ktorej sa nachádzali už obrázky jednotlivých vysegmentovaných riadkov z dokumentov a tiež ich prepisy. Riadky mali výšku 32px a rôznu šírku. Z databázy som použil len obrázky s veľkosťou menšou ako 600px. Databáza obsahovala ešte jednu sadu, ktorá bola vytvorená z pôvodnej s tým, že šírka riadkov bola obmedzená na veľkosť 224px. V tabuľke 2.3 sa nachádza rozdelenie týchto dátových sád na trénovacie, validačné a testovacie dáta. Dataset bol rozdelený na trénovacie a validačné dáta. Pre lepšie vyhodnotenie modelu som z validačných dát odobral a vytvoril testovaciu sadu. Sada s veľkosťou 600px obsahovala iba 5 436 testovacích obrázkov, preto som túto sadu už viac nedelil a v experimente som použil tú istú sadu pri validácii aj testovaní. Obrázky riadkov v datasete majú rôzne pozadia, kvalitu, font a obsahujú písané aj tlačené písmo. Všetok text je však napísaný v českom jazyku.

Kapitola 3

Prístup sequence to sequence

Hlboké neurónové siete sú veľmi silným modelmi v oblasti strojového učenia. V minulosti bolo nevýhodou týchto modelov to, že mohli byť aplikované iba na problémy, kedy bol vstupom aj výstupom neurónovej siete vektor fixnej dĺžky. S nástupom rekurentných neurónových sietí bol predstavený prístup sequence to sequence [42]. Sequence to sequence sa používa v súčasnosti pri riešení množstva problémov, ako napríklad machine translation [42], speech recognition [35], video captioning [45], alebo question answering [52]. Jeho výhodou je to, že dokáže mapovať vstupnú sekvenciu fixnej dĺžky na výstupnú sekvenciu variabilnej dĺžky. Prístup sequence to sequence spočíva v použití enkóder-dekóder architektúry. Enkóder-dekóder modely sú podmienené autoregresívne modely, a teda generujú jednotlivé tokeny výstupnej sekvencie v závislosti od vstupnej sekvencie a od naposledy vygenerovaných tokenov [42].

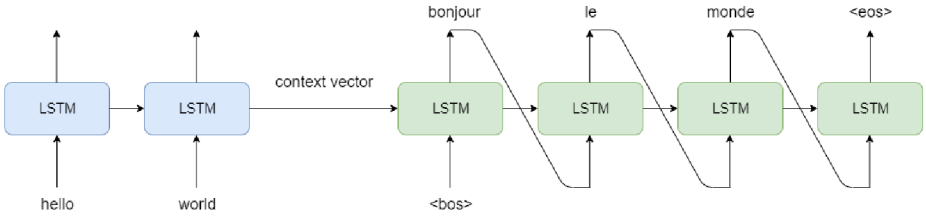
K problému rozpoznávania ručne písaného textu sa dá pristupovať ako k problému sequence to sequence, kedy je sekvencia vyextrahovaných príznakov zo vstupného obrázku transformovaná na sekvenciu znakov alebo slov na výstupe [48, 31].

3.1 RNN enkóder-dekóder

Najjednoduchšie sequence to sequence modely pozostávajú z dvoch častí - enkóderu a dekóderu, pričom enkóder aj dekóder sú rekurentné neurónové siete [42]. Enkóder kóduje vstupnú sekvenciu na vektor fixnej dĺžky, nazývaný kontext vector. Tento vektor v sebe nesie zakódovanú informáciu o celej vstupnej sekvencii. Je vstupom do dekódera, ktorý na svojom výstupe generuje v každom timestampe stále jeden token a tieto tokeny potom tvoria výstupnú sekvenciu. Na obrázku 3.1 je znázornená abstrakcia riešenia problému strojového prekladu pomocou sequence to sequence modelu, kedy je dĺžka vstupnej a výstupnej sekvencie odlišná.

Enkóder. Ako bolo spomínané vyššie, úloha enkóderu spočíva v zakódovaní sekvencie tokenov na vektor fixnej dĺžky [42]. Enkóder je zložený z rekurentných neurónov, ktoré kódujú vstupnú sekvenciu. Pri spracovávaní vstupnej sekvencie sa výstupy rekurentných neurónov zanedbávajú, berie sa do úvahy iba ich skrytý stav. Výsledný kontext vektor je potom skrytý stav rekurentných neurónov v poslednom timestampe kódovania [42]. Výpočet skrytého stavu v i-tom timestampe sa počíta vzťahom

$$h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1}), \tag{3.1}$$



Obrázek 3.1: Ilustrácia sequence to sequence pri riešení problému strojového prekladu. Veľkosť vstupnej sekvencie a výstupnej sekvencie sa líšia (inšpirované z [42]).

kde x_i je i-tý token vo vstupnej sekvencii a h_{t-1} je predchádzajúci vnútorný stav. W^{hx} a W^{hh} sú váhy.

Dekóder Dekóder je ďalšia RNN neurónová sieť v ktorej počiatočný vnútorný stav je inicializovaný kontext vektorom vygenerovaným z enkódera [42]. Dekóder je trénovaný na predikciu nasledujúceho tokenu (písmena alebo slova) v generovanej sekvencii, v závislosti od predchádzajúcich vygenerovaných tokenov. Je to teda autoregresívny RNN jazykový model, ktorý je v spojení s enkóderom podmienený vstupnou sekvenciou tokenov.

Pri vstupnej sekvencii $(x_1, ..., x_T)$ výpočet výstupnej sekvencie $(y_1, ..., y_T)$ pomocou RNN sa počíta vzťahmi

$$h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1}), \tag{3.2}$$

$$y_t = W^{yh}h_t, \tag{3.3}$$

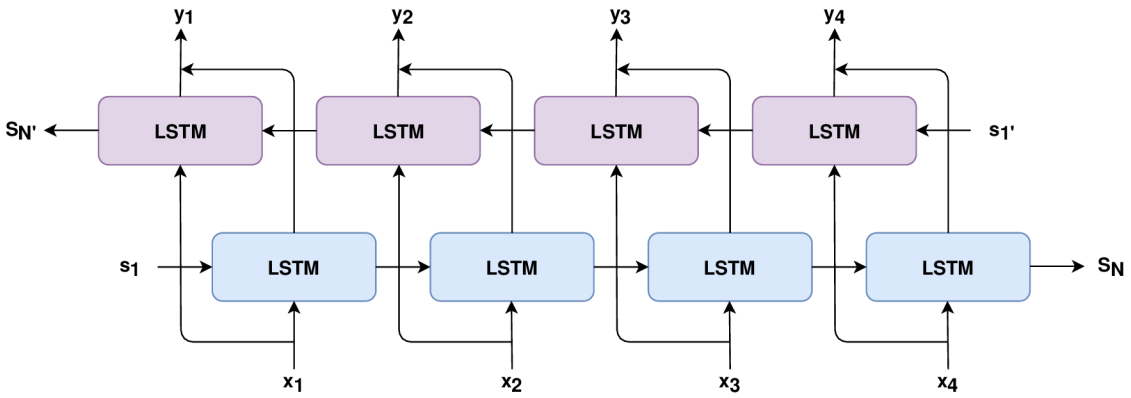
kde h_{t-1} predstavuje predchádzajúci vnútorný stav, x_t aktuálny vstup a W^{hx}, W^{hh} a W^{yh} sú váhy. Tým, že dĺžka vstupnej sekvencie a výstupnej sekvencie sa môžu líšiť, vieme zapísať, že enkóder-dekóder modeluje podmienenú pravdepodobnosť $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$, kde $y_1, \ldots, y_{T'}$ je výstupná sekvencia a $x_1, \ldots, x_T$ vstupná a kde ich dĺžky T' a T sa môžu líšiť. Táto pravdepodobnosť sa počíta vzťahom

$$p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, \ldots, y_{t-1}), \tag{3.4}$$

kde hodnota v predstavuje kontext vektor získaný zakódovaním vstupnej sekvencie enkóderom. $p(y_t|v, y_1, \ldots, y_{t-1})$ je počítaná funkciou softmax nad všetkými tokenmi v gramatike. Funkciou softmax je vytvorený vektor pravdepodobností, kedy hodnota každého prvku vektora na indexe i predstavuje pravdepodobnosť predikcie tokenu s daným indexom. [42].

LSTM a GRU. Rekurentná neurónová sieť zložená z klasických rekurentných neurónov pri spracovávaní dlhých sekvencií trpí problémom miznúceho gradientu [4] a preto sa väčšinou v RNN v enkóder-dekóder modeloch používajú radšej LSTM [41] alebo GRU neuróny [8], ktoré boli práve navrhnuté na riešenie tohto problému. V [42] je taktiež uvedené, že enkóder-dekóder model, ktorý dostáva na vstupe otočenú vstupnú sekvenciu, vykazuje väčšiu úspešnosť.

Pre vylepšenie úspešnosti modelu je možné použiť v enkóderi obojsmernú LSTM alebo GRU sieť [37, 47]. Takýto enkóder je potom zložený z dvoch LSTM alebo GRU neurónových



Obrázek 3.2: Obojsmerná LSTM neurónová sieť (inšpirované z [32]).

vrstiev. Prvá spracováva vstupnú sekvenciu tokenov v takom poradí ako boli odoslané do enkódera, a druhá spracováva jej obrátenú kópiu. Výsledný vnútorný stav potom vzniká konkatenáciou vnútorných stavov obidvoch vrstiev. Vďaka obojsmernej LSTM/GRU sieti má enkóder pri každom kroku kódovania informáciu o tokenoch ktoré sa nachádzajú pred, aj za aktuálnym tokenom. Výsledkom je, že sa sieť učí rýchlejšie a dokáže nachádzať nové závislosti medzi jednotlivými tokenmi. Na obrázku 3.2 je znázornená obojsmerná LSTM neurónová sieť.

3.2 RNN enkóder-dekóder s attention mechanizmom

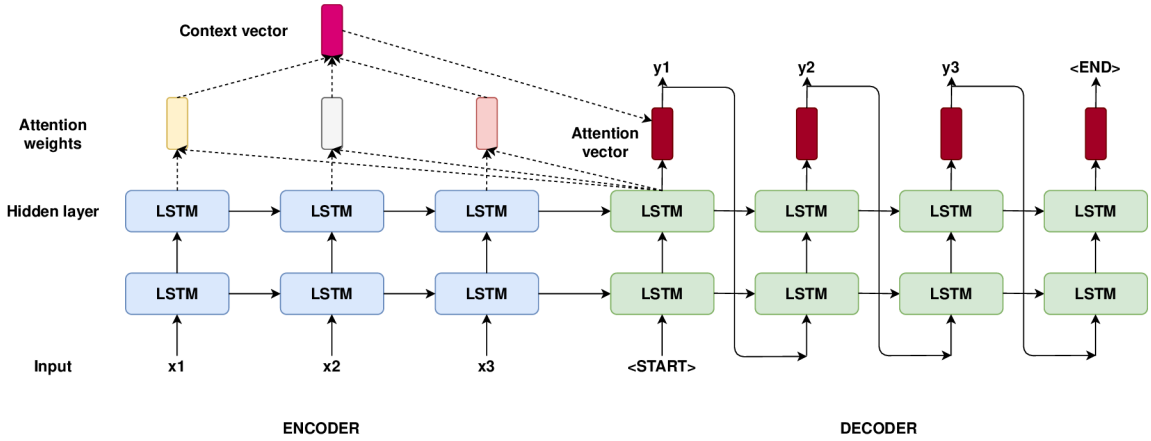
V klasickom enkóder-dekóder modeli sa pri generovaní tokenu dekóderom pozerá iba na minulý vygenerovaný token a na kontext vektor (posledný skrytý stav enkódera). Pri dlhých vstupných sekvenciách je preto veľká pravdepodobnosť, že informácie ktoré sa nachádzali na začiatku sekvencie budú stratené. Attention mechanizmus [1] rozširuje štandardný enkóder-dekóder model o funkciu dynamickej modifikácie kontext vektoru v každom timestampe, v závislosti od aktuálneho vnútorného stavu dekódera. Použitie attention dáva možnosť dekóderu pristupovať ku všetkým skrytým stavom enkódera, nie len ku koncovému stavu. Tým je možné zachytiť lepšie vzťahy medzi jednotlivými vstupnými tokenmi a určiť ktoré vstupné tokeny sú relevantné pre predikciu aktuálneho tokenu. Attention mechanizmus taktiež zvyšuje úspešnosť siete pri pracovaní s dlhými vstupnými sekvenciami. Na obrázku 3.3 je znázornený enkóder-dekóder model, ktorý je vylepšený o attention mechanizmus.

Základný enkóder-dekóder modeluje podmienenú pravdepodobnosť $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$, kde $y_1, \ldots, y_{T'}$ je výstupná sekvencia a $x_1, \ldots, x_T$ vstupná. Túto pravdepodobnosť je možné definovať vzťahom 3.4. Dekóder je trénovaný na predikciu tokenu y_t, pomocou kontext vektora c a predošlých predikovaných tokenov $y_1, \ldots, y_{t-1}$. Predikovanie celej výstupnej sekvencie $y = (y_1, \ldots, y_{T_y})$ vyjadruje vzťah

$$p(y) = \prod_{t=1}^{T'} p(y_t|y_1, \ldots, y_{t-1}, c), \tag{3.5}$$

pričom každá podmienená pravdepodobnosť je vyjadrená ako

$$p(y_t|y_1, \ldots, y_{t-1}, c) = g(y_{t-1}, s_t, c). \tag{3.6}$$



Obrázek 3.3: Dvojvrstvový LSTM enkóder-dekóder model s attention mechanizmom (inšpirované z [26]).

Premenná s_t predstavuje skrytý stav dekódera v čase t a g predstavuje nelineárnu funkciu ktorej výstupom je pravdepodobnosť y_t.

V článku [1], kde bola predstavená attention, definujú podmienenú pravdepodobnosť ako

$$p(y_i|y_1, \ldots, y_{i-1}, X) = g(y_{y-1}, s_i, c_i), \tag{3.7}$$

kde s_i predstavuje skrytý stav dekódera v čase i, ktorý je možné vypočítať vzťahom $s_i = f(s_{i-1}, y_{i-1}, c_i)$. Kontext vektor c_i vieme vypočítať ako vážený súčet anotácií h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j, \tag{3.8}$$

pričom váha α_{ij} každej anotácii h_j je počítaná ako

$$\alpha_{ij} = exp(e_i j) / \sum_{k=1}^{T_x} exp(e_{ik}), \tag{3.9}$$

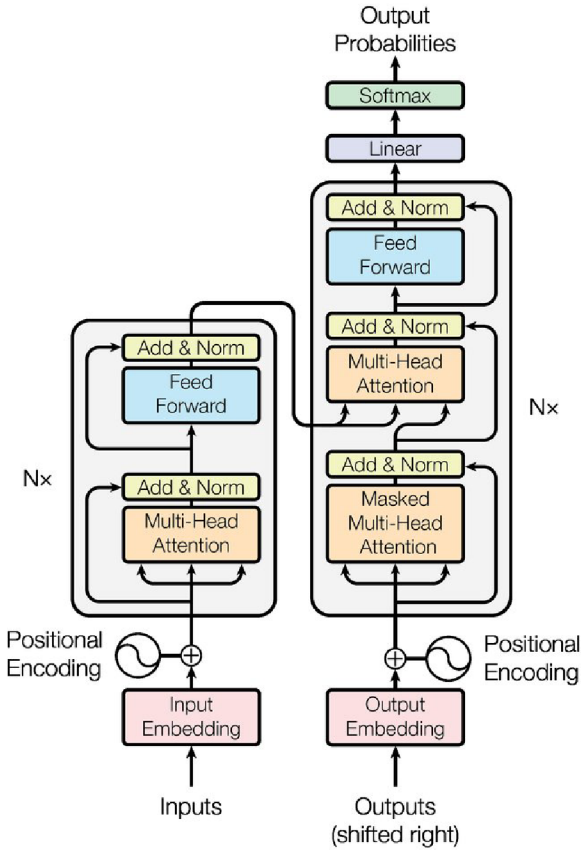
kde e_{ij} je ohodnotenie toho, ako veľmi sa má brať do úvahy vstup na pozícii j, pri generovaní výstupného tokenu na pozícii i. Toto ohodnotenie je závislé od skrytého stavu s_{i-1} dekódera, anotácie h_j vstupnej vety a vypočíta sa vzťahom

$$e_{ij} = a(s_{i-1}, h_j). \tag{3.10}$$

Symbol a predstavuje plne prepojenú neurónovú sieť, ktorá sa trénuje spolu s modelom [1].

3.3 Transformer

Väčšina sequence to sequence modelov zameraných na rozpoznávanie ručne písaného písma používajú konvolučnú sieť v spolupráci s rekurentnou sieťou. Tento model je rozdelený na



Obrázek 3.4: Model architektúry transformera (prevzaté z [44]).

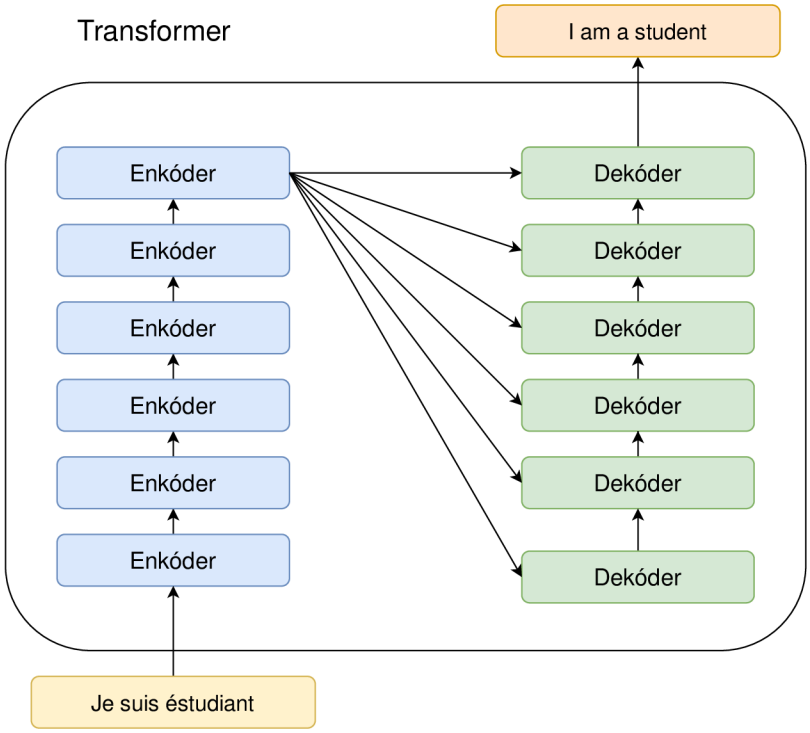
.

enkóder a dekóder, ktorý je zvyčajne prepojený ešte s attention mechanizmom pre vylepšenie úspešnosti siete. Použitie rekurentných neurónov znižuje schopnosť paralelizácie pri trénovaní siete, a tým je trénovanie takejto siete pomalé [44].

Vaswani [44] predstavil novú architektúru neurónovej siete, nazývanú transformer. Táto sieť je založená iba na attention mechanizme, nepotrebuje konvolučnú ani rekurentnú sieť. Táto architektúra bola použitá pri probléme strojového prekladu a dosiahla porovnateľné výsledky s vtedy používanými prekladačmi, pri väčšej miere paralelizácie a menšom potrebnom čase na trénovanie. Na obrázku 3.4 je možné vidieť model architektúry transformera.

Pri enkóder-dekóder modeloch používajúcich rekurentné neuróny, počet operácií ktoré je potrebné vykonať aby sa vstupná sekvencia zakódovala do kontextového vektora a následne aby dekóder vygeneroval výstupnú sekvenciu, narastá s dĺžkou spracovávanej sekvencie. Kvôli tomu sa pri práci s dlhými sekvenciami predlžuje čas trénovania a rýchlosť spracovania vstupu sieťou klesá. Pri transformeri tento problém zaniká, pre prenos informácie z enkódera do dekódera je potrebný konštantný počet operácií.

Transformer sa skladá z dvoch častí. Prvá časť obsahuje skupinu enkóder blokov a druhá skupinu dekóder blokov. V [44] použili 6 enkóder aj dekóder blokov. Existujú experimenty [40], v ktorých autori skúmali možnosť použitia rozdielneho počtu enkóder a dekóder blokov. Takýto experiment je aj súčasťou tejto práce 5.3. Všetky enkódery majú rovnakú štruktúru, ale nezdieľajú medzi sebou váhy. Na nasledujúcom obrázku 3.5 je znázornená abstrakcia transformera pre riešenie problému strojového prekladu.



Obrázek 3.5: Abstrakcia transformeru so šiestimi enkóder blokmi a šiestimi dekóder blokmi, pre riešenie problému strojového prekladu.

Enkóder blok

Enkóder blok sa skladá z dvoch častí. Prvou je multi-head attention modul a normalizačná vrstva. Druhou je plne prepojená vrstva a normalizačná vrstva. Medzi každými takýmito dvoma časťami je reziduálne prepojenie [15]. Prepája informáciu zo vstupu multi-head attention vrstvy alebo plne prepojenej vrstvy, s normalizačnou vrstvou. Pred spracovaním vstupnej sekvencie prvým enkóder blokom, je na túto sekvenciu aplikovaná embedding vrstva, ktorá transformuje vstupné tokeny na vektory požadovanej dimenzie a aby sieť mala informáciu o pozícii slov v rámci vety, je na ňu aplikované pozičné kódovanie. Multi-head attention modul používa pre svoje fungovanie self-attention vrstvu.

Self-attention

Self-attention vrstva produkuje 3 vektory pre každý vstupný vektor - query vektor, key vektor a value vektor. Tieto vektory vznikajú násobením vstupných vektorov s troma maticami váh, ktoré sa trénujú počas procesu učenia.

Druhým krokom je vypočítanie hodnoty score pre každý vstupný vektor. Táto hodnota sa počíta pre každý token zo vstupnej sekvencie a určuje vplyv jednotlivých tokenov na daný token ktorý má byť práve enkódovaný. Score hodnota je počítaná ako dot produkt query vektora aktuálneho tokenu a key vector ostatných tokenov.

Následne je hodnota score delená konštantou 8. Je to odmocnina z počtu dimenzií key vektora z článku o transformeri [44]. Táto hodnota môže byť aj odlišná, bola určená empiricky. Následne je z hodnôt score počítaný softmax, čiže všetky hodnoty score pre konkrétne slovo sú kladné, menšie ako 1 a ich súčet dosahuje hodnotu 1.

Ďalším krokom je vynásobenie value vektorov s hodnotami softmax score z predchádzajúceho výpočtu. Výsledkom sú vektory ktoré sú spracovávané plne prepojenou vrstvou. Vzťah predstavujúci maticový výpočet výstupu self-attention vrstvy môže byť zapísaný ako

$$Attention(Q, K, V) = softmax(\frac{Q * K^T}{\sqrt{(d_k)}}) * V, \tag{3.11}$$

kde Q je matica query hodnôt, K^T je transponovaná matica key hodnôt, d_k je dimenzionalita key vektorov a V je matica value hodnôt.

Multi-head attention

V článku *Attention Is All You Need* [44] autori rozšírili koncept self-attention pridaním mechanizmu multi-head attention. Multi-head attention nepoužíva jednu query, key a value maticu. Pracuje s celou sadou takýchto matíc, ktoré sú náhodne inicializované. Každá sada váhových matíc transformuje vstupné tokeny do odlišných reprezentačných priestorov. V [44] bolo použitých 8 sád matíc. Toto číslo bolo stanovené empiricky.

Multi-head attention teda produkuje 8 Z_i matíc. Feed-forward vrstva očakáva na vstupe len jednu maticu. Preto je všetkých 8 matíc konkatenovaných do jednej matice, a tá je násobená s váhovou maticou W^0, ktorá je trénovaná spolu s modelom. Výsledkom je matica Z, ktorá je spracovávaná plne prepojenou vrstvou.

Pozičné kódovanie

Pozičné kódovanie (Positional encoding) je metóda, kedy ku vstupným embedovaným tokenom v enkóderi aj dekóderi je pripočítavaný vektor, ktorý poskytuje informáciu o polohe daného tokenu v sekvencii. Počet dimenzií tohto vektora je rovnaký ako počet dimenzií vektorov, ktoré produkuje embedding vrstva. V [44] boli použité vzťahy 3.12 a 3.13 na výpočet pozičného kódovania

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}), \tag{3.12}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}), \tag{3.13}$$

kde pos predstavuje pozíciu v sekvencii a i dimenziu.

Dekóder blok

Výstup posledného enkódera je transformovaný na sadu vektorov key a value. Tieto vektory sú následne použité každým dekóder blokom. Dekóder blok sa tak môže zamerať na jednotlivé tokeny vo vstupnej sekvencii. Vstupom do prvého bloku dekóderu je počiatočný symbol. Informácie z posledného enkóder bloku a z prvého dekóder bloku prechádzajú postupne k poslednému dekóder bloku, kde sa vygeneruje výstupný token. Tento token je následne pridaný k doposiaľ vygenerovaným tokenom a takáto sekvencia je odoslaná opäť do prvého dekóderu. Tento proces sa opakuje kým posledný dekóder nevygeneruje token reprezentujúci koniec vety.

Ako je možné vidieť na obrázku 3.4, dekóder blok sa skladá z rovnakých častí ako enkóder blok, popísaných v kapitole 3.3. Multi-head attention vrstva v dekóder blokoch sa líši od enkóderovej tak, že matice keys a values berie z výstupu posledného enkódera a

maticu queries z predchádzajúceho bloku. Masked multi-head attention v dekóderi môže mať informáciu iba o doposiaľ vygenerovaných tokenoch. Preto sú tokeny v sekvencii, ktoré sa nachádzajú na budúcich pozíciach (doposiaľ nevygenerovaných) nahradzované hodnotou -inf. Posledný dekóder generuje na svojom výstupe vektor desatinných čísel. Tie sú následne spracovávané lineárnou a softmax vrstvou. Lineárna vrstva je obyčajná plne prepojená vrstva, ktorá transformuje výstup z posledného dekódera na vektor, nazývaný logits vektor. Počet prvkov tohto vektora sa rovná počtu slov v slovníku pre danú úlohu. Softmax vrstva transformuje logits vektor na vektor pravdepodobností, ktoré určujú s akou pravdepodobnosťou sieť generuje daný token. Index prvku vo výstupnom vektore pravdepodobností určuje pozíciu slova v slovníku.

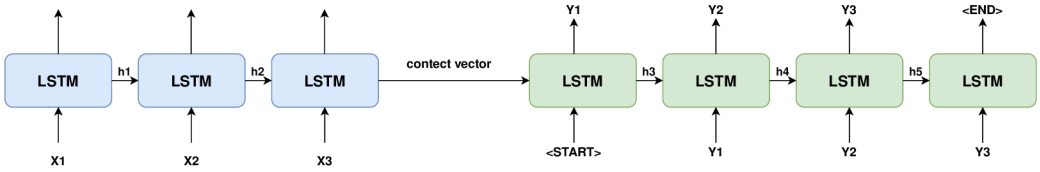
Transformer používa teda 3 druhy attention mechanizmu:

- **Enkóder-dekóder attention** počíta s hodnotami queries z predchádzajúcej dekóder vrstvy a hodnoty keys a values pochádzajú z posledného enkódera. Vďaka tomu je možné zamerať sa pre každú pozíciu v dekóderi na každú pozíciu vstupnej vety.
- **Enkóder attention** počíta s hodnotami keys, queries a values z predchádzajúcej vrstvy enkódera. Tým pádom sa môže transformer zamerať pre každú pozíciu v enkóderi na každú pozíciu z predchádzajúcej vrstvy enkódera.
- V **dekóder attention**, podobne ako v enkóder attention sa transformer môže zamerať pre každú pozíciu v dekóderi na každú pozíciu z predchádzajúcej vrstvy dekódera.

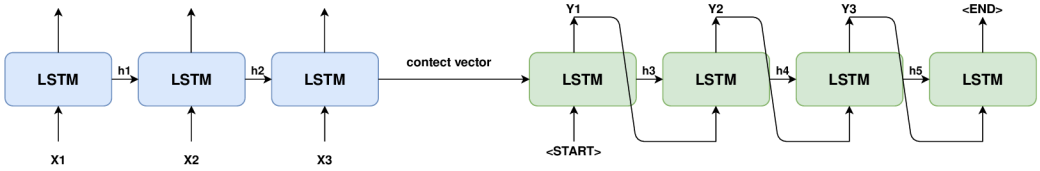
3.4 Trénovanie enkóder-dekóder modelov

Autoregresívne enkóder-dekóder modely používajú v dekóderi pre výpočet výstupného tokenu minulý výstup siete. Takéto modely dostanú na vstupe symbol $<START>$, ktorý predstavuje pre dekóder začiatok generovania sekvencie. Tento symbol predstavuje začiatok procesu, kedy dekóder vygeneruje na svojom výstupe token a tento token bude následne vstupom dekódera v nasledujúcom kroku. Tento proces je opakovaný, kým dekóder nevygeneruje špeciálny symbol $<END>$, alebo kým nie je dosiahnutá maximálna dĺžka výstupnej sekvencie. Ten istý proces, kedy je minulý výstup dekódera použitý ako jeho aktuálny vstup je možné použiť aj pri trénovaní siete. Vo veľa prípadoch to však smeruje k problému slabej konvergencie a pomalého trénovania. Použitím metódy teacher forcing je v mnohých prípadoch možné týmto problémom predísť a urýchliť tak proces trénovania.

Metóda teacher forcing [5] je stratégia trénovania autoregresívnych modelov, ktoré generujú na výstupe tokeny, v závislosti od minulých vygenerovaných tokenov. Pri trénovaní touto metódou sa ako vstup do dekódera nepoužije skutočný minulý výstup siete, ale očakávaný výstup. Preto je ako vstup do dekódera použitá očakávaná výstupná sekvencia, posunutá o jeden token neskôr. Pri inferencii, kedy je už model natrénovaný je ako vstup do dekódera opäť použitý minulý výstup siete. Na obrázku 3.6 a 3.7 je znázornená metóda teacher forcing v trénovacej a testovacej fáze.



Obrázek 3.6: Metóda teacher forcing - trénovacia fáza (inšpirované z [22]). Pri trénovaní je na vstupe dekódera očakávaná sekvencia, posunutá o jeden token v čase neskôr. Prvým tokenom je symbol <START>.



Obrázek 3.7: Metóda teacher forcing - testovacia fáza (inšpirované z [22]). Pri testovaní je na vstupe dekódera počiatočný token <START> a následne sa v každom kroku generovania na vstup dekódera pridáva naposledy dekóderom vygenerovaný token.

Kapitola 4

Metóda

V tejto práci som sa rozhodol vytvoriť 2 modely. Prvý model je klasický enkóder-dekóder model, zložený z dvoch rekurentných neurónových sietí 4.1. Druhým modelom je transformer 4.2. Obidva modely spracovávajú mapu príznakov ako sekvenciu, ktorá je získaná z konvolučnej neurónovej siete.

4.1 Príprava datasetu

Kvalita a množstvo trénovacích dát výrazne vplýva na natrénovanie neurónovej siete. V experimentoch pracujem s datasetmi MADCAT, IAM, 111 let českých dopisu v korpusovém zpracování a dataset českých dokumentov.

Dataset IAM som rozdelil na 2 odobitné datasety, pričom jeden je zložený z celých vysegmentovaných riadkov a druhý z vysegmentovaných slov. Pri experimentovaní som zistil, že IAM dataset ktorý pozostáva z vysegmentovaných riadkov je príliš malý na natrénovanie modelov (6 141 riadkov) a nepomáhala ani augmentácia dát. Preto sa v tejto práci nachádzajú len experimenty s IAM datasetom s vysegmentovanými slovami.

Dataset MADCAT obsahuje celé naskenované dokumenty a informácie o prepisoch a súradniciach riadkov uchováva v XML dokumentoch. Pre vygenerovanie dátovej sady vhodnej pre OCR rozpoznávanie som vytvoril program ktorý z MADCAT dokumentov, podľa informácií z XML súborov, vysegmentoval jendotlivé riadky, zmenšil ich na požadovanú veľkosť (výšku 32px) a uložil k nim prepisy. K niektorým riadkom chýbali prepisy, tie bolo potrebné vyradiť. Takto pripravený dataset som uložil v samostatnom súbore a neskôr používal na trénovanie modelov.

Obidva české datasety už obsahovali obrázky vysegmentovaných riadkov, segmentácia nebola potrebná. Dataset českých dokumentov bol uložený v lmdb databáze a dataset 111 let českých dopisu v korpusovém zpracování obsahoval priečinok s obrázkami a súbor s prepismi pre trénovaciu aj testovaciu sadu. Pre obidva datasety som vytvoril program ktorý mi ku každému obrázku pridelil prepis a takúto dátovú sadu som si uložil v podobe súboru, vhodného na trénovanie siete.

Použité datasety obsahovali málo šumu a boli naskenované v relatívne dobre kvalite, preto som sa preprocessingu výrazne nevenoval. Cieľom experimentov nebolo natrénovanie čo najlepšieho modelu, ale skôr preskúmanie vplyvu jednotlivých parametrov na učenie daného modelu. Na obrázky som aplikoval iba binarizáciu a v niektorých prípadoch augmentáciu dát.

Augmentácia dát

Augmentácia datasetu je rozšírenie trénovacích dát o nové vzorky, ktoré sú vygenerované z pôvodných dát použitím rôznych foriem deformácie ako rotácia, scaling posun alebo pridanie šumu. Táto metóda sa často využíva v oblasti klasifikácie obrazu alebo detekcie reči a prispieva k zvýšeniu schopnosti modelu učiť sa a generalizovať.

V experimentoch v ktorých som používal augmentáciu dát, som použil kód od výskumnej skupiny PERO, ktorý sa aplikoval na jednotlivé obrázky riadkov z datasetu. Pôvodným naskenovaným textom bola náhodne menená veľkosť písma, sklon písma a do obrázku bol náhodne pridávaný šum.

Tokenizácia

Tokenizácia je metóda, ktorá je využívaná v spracovaní prirodzeného jazyka na lepšiu manipuláciu s vetami a ich syntaxou. Touto technikou je veta v podobe znakov a slov prevedená na sekvenciu tokenov, ktorá je ďalej transformovaná na vstup neurónovej siete v podobe vektora. Každému tokenu je priradená číselná hodnota (index) a tá je uložená v slovníku. Podľa tohto slovníku je sekvencia tokenov transformovaná na vektor, kedy každý prvok vektora predstavuje index v slovníku, odkazujúci na konkrétny token.

Pri použití tokenizácie je potrebné zvážiť, ktorý typ tokenizácie si chceme zvoliť [19]. Pri tokenizácii na úrovni písmen, kedy každé písmeno predstavuje jeden osobitný token, je výhodou nízky počet všetkých možných tokenov. Ďalšou výhodou je to, že akékoľvek slovo vieme vyjadriť sériou tokenov, nerozpoznané ostanú len znaky ktoré sa nenachádzajú v sade znakov ktorú používame. Nevýhodou tokenizácie na úrovni znakov je to, že tokeny predstavujúce písmená nenesú žiadnu sémantickú informáciu o danej vete a preto sa sieť učí pomalšie.

Opačným extrémom je tokenizácia na úrovni slov, kedy každý token predstavuje jedno slovo. Výhodou je to, že tokeny nesú určitú sémantickú informáciu a to pomáha neurónovej sieti lepšie sa učiť a generalizovať. Pri HTR systémoch to môže viezť k nižšej chybovosti, pretože nebudú nastávať chyby spôsobené zlým rozpoznaním konkrétneho písmena. Sieť bude rozpoznávať text na obrázku po slovách. Nevýhodou je, že sieť pracuje s veľkou slovnou zásobou a výpočet je pomalší. Taktiež môže mať sieť problém so slovami ktoré sa nenachádzajú v slovníku, alebo v nich nastal preklep. Tento problém je riešený použitím špeciálneho tokenu *<OOV>*, ktorý predstavuje neznámy token (out of vocabulary).

Ďalším typom tokenizácie je tokenizácia pomocou častí slov [27]. Pri tejto metóde si vieme určiť, aký veľký slovník tokenov požadujeme. Pri použití malého slovníka budú tokeny predstavovať písmená v slove. V prípade použitia veľkého slovníka budú tokeny predstavovať najčastejšie vyskytujúce sa slova, alebo slovné spojenia. Touto metódou je možné akékoľvek slová pretransformovať na sériu tokenov, <OOV> token sa použije len v prípade, že sa narazí na neznámy znak. Jednou zo známych metód tokenizácie pomocou častí slov je metóda BPE (byte pair encoding) [39], ktorá funguje na princípe iteratívneho združovania najčastejšie sa vyskytujúcich párov symbolov do nových symbolov. Iteratívne párovanie prebieha dovtedy, kým nie je dosiahnutý potrebný počet unikátnych tokenov. Takto vznikajú tokeny predstavujúce najčastejšie sa vyskytujúce časti slov, v trénovacej sade. Ďalšie algoritmy ktoré fungujú na princípe tokenizácie pomocou časti slov sú napríklad Unigram tokenizácia alebo WordPiece tokenizácia [27].

Pri tokenizácii exitujú okrem symbolu <OOV> aj špeciálne symboly <PAD>, <START>, <END>. Token <START> sa pridáva na začiatok vety, a je pri procese dekódovania pou-

žitý ako prvý token vstupujúci do dekódera. Token <END> signalizuje ukončenie procesu dekódovania. <PAD> sa využíva pri paddingu, ktorý je opísaný v nasledujúcej sekcii.

V tejto práci som experimentoval hlavne s tokenizáciou na úrovni znakov. Používal som pritom modul z knižnice tensorflow[1]. Tento modul som použil aj pri experimente s tokenizáciou na úrovni slov. V jednom z experimentov som používal tokenizáciu na úrovni častí slov, pričom som použil BPE tokenizáciu z knižnice SentencePiece [24].

Padding a normalizácia

Pred trénovaním modelu bolo nutné použiť metódu padding na vstupné obrázky, aj na očakávaný výstup siete. Pri spracovávaní datasetu bola vopred určená požadovaná výška a šírka obrázkov. Následne bol obrázok zmenšený na požadovanú výšku, pričom bol zachovaný pomer strán. Obrázky ktoré mali menšiu šírku ako bola požadovaná, boli vynechané a obrázky s menšou šírkou boli doplnené čiernymi pixelmi. Pri paddingu obrázkov boli doplnené čierne pixely sprava, čiže na všetkých obrázkoch začínal text z ľavého okraja obrázka. Predtým ako bol obrázok použitý ako vstup do neurónovej siete prebehla ešte normalizácia, kedy sa každá hodnota pixelu normalizovala na hodnotu v intervale <0,1>.

Následne bola zvolená maximálna dĺžka prepisu textu a všetky prepisy z datasetu boli zarovnané na požadovanú dĺžku. Vzorky ktorých prepis prekračoval zvolenú maximálnu dĺžku boli vynechané.

4.2 Model

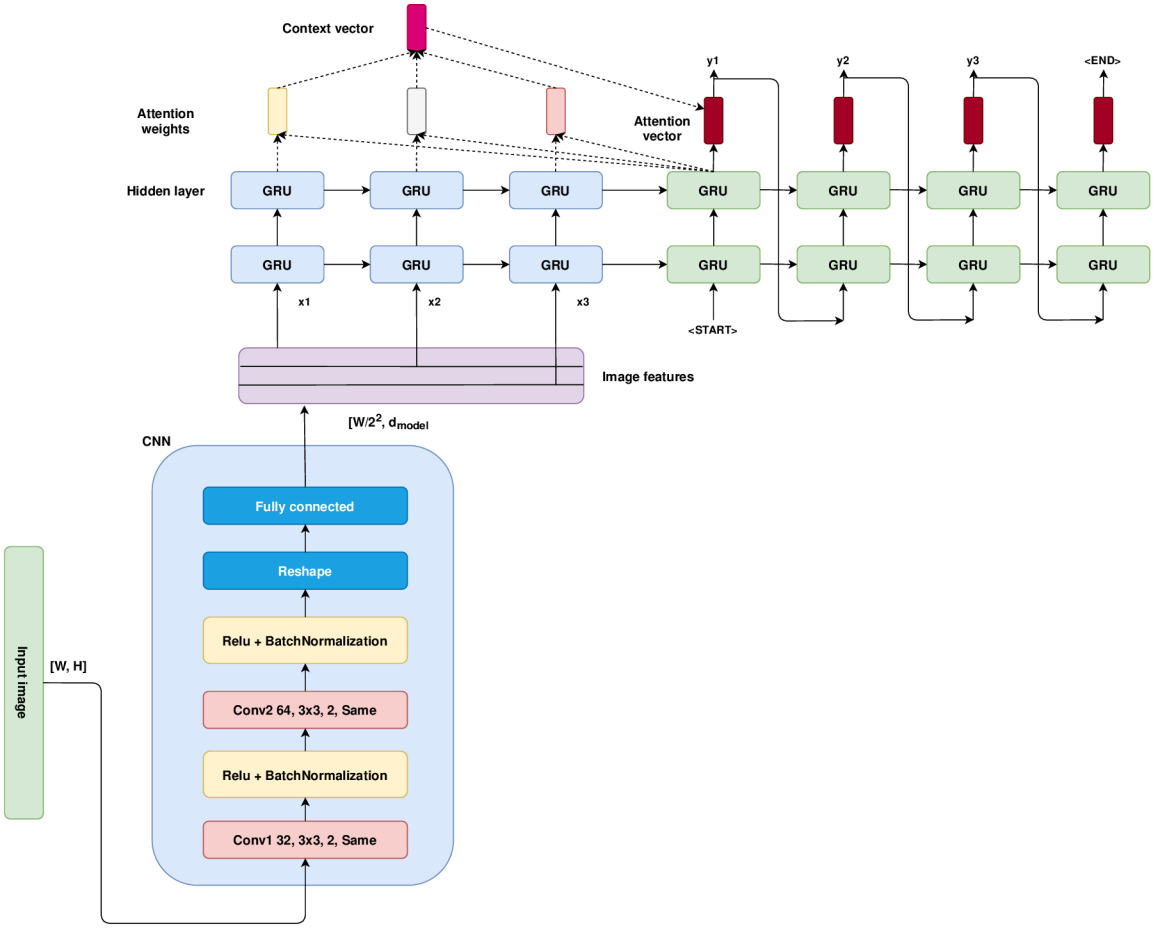
Pri HTR systémoch založených na enkóder-dekóder modeli môže byť súčasťou enkóderu konvolučná neurónová sieť [31, 21]. Jej úlohou je vyextrahovať z obrázka mapu príznakov, ktorá je ďalej spracovávaná enkóderom ako sekvencia. V tejto práci som pracoval s dvomi modelmi. Prvý používal enkóder-dekóder s rekurentnými neurónmi, a druhý transformer. Obidva modeli spracovávajú výstup z konvolučnej siete. Vstupom modelov boli vysegmentované obrázky riadkov alebo slov, a výstupom digitálny prepis rozpoznaného textu. Samotnou segmentáciou som sa v tejto práci nezaoberal. Použité modely boli inšpirované prácami [21, 31, 48, 40].

CNN-RNN model

Prvým modelom je klasický enkóder-dekóder model, pričom enkóder aj dekóder sú 2 GRU rekurentné neurónovej siete. Vstupný obrázok je spracovávaný konvolučnou neurónovou sieťou. Táto sieť pozostáva z k konvolučných vrstiev. V sekcii sú popísané pokusy, kedy som skúšal experimentovať s veľkosťou konvolučnej siete. Základný model z ktorého som vychádzal mal nastavenú konvolučnú sieť s hodnotou $k = 2$. Veľkosť kernelov som nastavil na 3x3 a stride na hodnotu 2. Počet kernelov d_{conv} je v prvej vrstve nastavený na hodnotu 32 a každou ďalšou vrstvou sa zdvojnásobuje. Po konvolučných vrstvách je ďalej aplikovaná funkcia reshape, ktorá transformuje mapu príznakov na dvojrozmerný sekvenciu príznakov, ktorá bude spracovávaná enkóderom. Vstupný obrázok s rozmermi (W, H) je transformovaný na sekvenciu príznakov $(w/2^n, d_{conv})$.

Enkóder je zložený z GRU neurónov. spracováva výstup z konvolučnej siete ako sekvenciu príznakov. V modeli bola použitá obojsmerná GRU sieť, ktorá spracováva vstupnú

[1] $https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer$



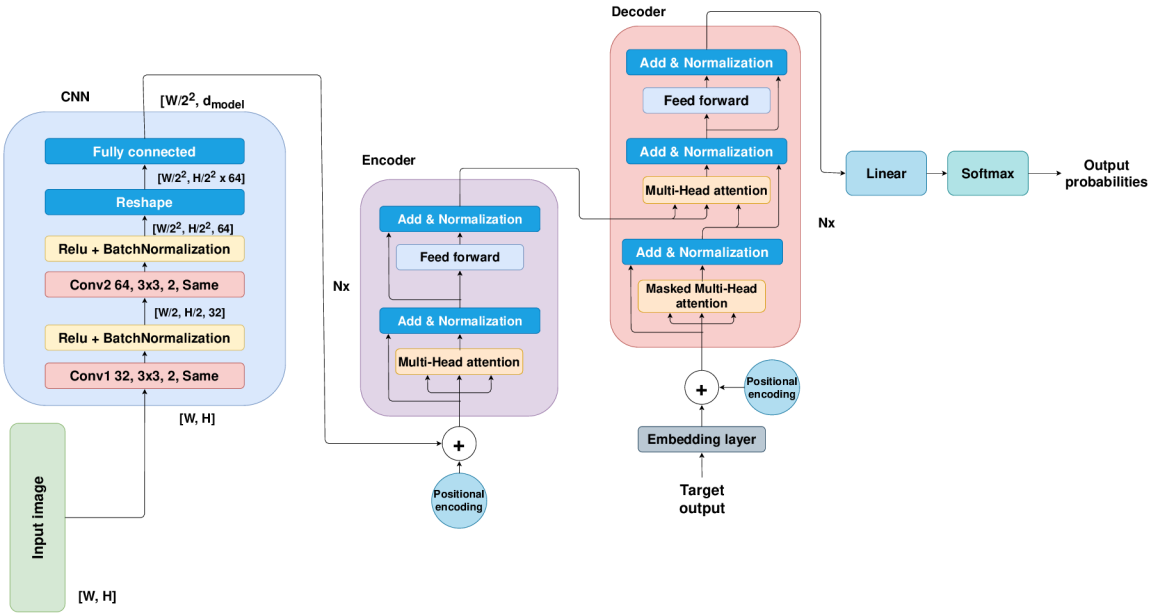
Obrázek 4.1: Koncept modelu ktorý pozostáva z konvolučnej neurónovej siete a GRU enkóder-dekódera

sekvenciu v obidvoch smeroch, pre zachytenie dopredných aj spätných závislostí písaného textu.

Na vstupe dekódera sa nachádza embedding vrstva. Jej úlohou je transformovať jednotlivé tokeny na vektory fixnej dĺžky s veľkosťou d_{model}. Dekóder je zložený opäť z GRU neurónov. Úlohou dekódera je generovať na svojom výstupe v každom timestampe jeden token, čím vznikne sekvencia tokenov predstavujúca prepis textu z obrázku. Tým, že na výstupe dekódera je fully connected vrstva na ktorú je aplikovaný softmax, dekóder generuje pravdepodobnostné rozloženie nad slovníkom obsahujúcim všetky možné znaky, ktoré chceme rozpoznávať vo vstupnom obrázku. Dekóder používa pri generovaní nového tokenu attention mechanizmus ktorý bol popísaný v kapitole 3.2. Vďaka tomu sa počas generovania tokenu môže zameriavať na jednotlivé tokeny vo vstupnej sekvencii s rôznou váhou.

Pri implementácii som vychádzal z tensorflow projektu[2], ktorý sa týkal problému strojového prekladu translation, kde tento problem riešili pomocou enkód-dekóder modelu. Z pôvodného modelu bola odstránená embeding vrstva na vstupe a pôvodný jednovrstvový enkóder bol vymenený za obojsmerný GRU enkóder. Tento enkóder-dekóder model bol napojený na konvolučnú sieť, popísaná vyššie. Dimenzionalitu enkóder-dekóder modelu d_{model}

[2]https://www.tensorflow.org/tutorials/text/nmt_with_attention



Obrázek 4.2: Koncept modelu ktorý pozostáva z konvolučnej neurónovej siete a transformera.

aj dimenzionalitu embeddingu som nastavil na hodnotu 512. Koncept tohto modelu je zobrazený na obrázku 4.1.

Konceptom napojenia enkóder-dekóder modelu na konvolučnú sieť som sa inšpiroval z článku [31]. Namiesto LSTM neurónov som však použil GRU neuróny.

Transformer

Ďalším modelom je enkóder-dekóder model založený na transformeri a konvolučnej neurónovej sieti. Tento model nepoužíva rekurentné neuróny, je možné ho lepšie paralelizovať a tým je jeho trénovanie rýchlejšie ako pri predchádzajúcom modeli. Koncept tohto modelu je zobrazený na obrázku 4.2

Vstupný obrázok je najprv spracovávaný konvolučnou neurónovou sieťou. Použitá bola tá istá konvolučná sieť ako v predchádzajúcom modeli, s rovnakým nastavením parametrov.

Enkóder spracováva výstup konvolučnej siete. Pozostáva z $N_{encoder}$ rovnakých blokov. Každý blok enkódera obsahuje multi-head attention vrstvu a plne prepojenú vrstvu. Vďaka multi-head attention vrstve sa blok enkódera môže pri kódovaní tokenu zameriavať na ostatné tokeny vo vstupnej sekvencii, alebo vo výstupe predchádzajúceho bloku enkódera a tým sa môže naučiť vzťahy medzi vstupnými tokenmi. Detailnejší popis fungovania enkódera sa nachádza v kapitole . Pôvodný koncept transformera počíta s pozičným kódovaním vstupu a embedding vrstvou. V tomto modeli je na vstup enkódera aplikované pozičné kódovanie, aby mal transformer nejakú informáciu o relatívnej pozícii tokenov. Pozičné kódovanie je počítané vzťahom . Keďže vstupom modelu je obrázok, embedding vrstva na vstupe enkódera nebola použitá. Za multihead attention vrstvou a plne prepojenou vrstvou sa nachádza ešte normalizačná vrstva, ku ktorej je konkatenovaný vstup z plne prepojenej vrstvy, alebo z attention vrstvy, reziduálnym prepojením.

Dekóder pozostáva z $N_{decoder}$ identických blokov. Každý blok dekóderu obsahuje 2 multi-head attention vrstvy a jednu plne prepojenú vrstvu. Prvá, takzvaná masked multi-head

attention vrstva slúži na to, aby sa pri predikcii aktuálneho tokenu brali do úvahy iba doposiaľ vygenerované tokeny. Dekóder sa totiž trénuje paralelne pomocou metódy teacher forcing a tým by mal pri generovaní určitého tokenu prístup k tokenom ktoré ešte neboli vygenerované. To je riešené práve aplikovaním masky na vstup dekódera. Druhá multi-head attention vrstva berie hodnoty keys a values z výstupu enkódera a hodnoty queries z predošlého výstupu masked multi-head attention vrstvy. Na konci každého bloku dekódera sa nachádza ešte plne prepojená vrstva. Za obidvomi attention vrstvami a plne prepojenou vrstvou sa nachádza normalizačná vrstva, ku ktorej je konkatenovaný vstup attention vrstvy alebo plne prepojenej vrstvy, reziduálnym spojením.

Na výstup dekódera je aplikovaná softmax vrstva, v ktorej je vypočítané pravdepodobnostné rozloženie nad slovníkom všetkých tokenov. Pri trénovaní siete je vstupom do dekódera očakávaná sekvencia tokenov, na ktorú je aplikovaný embedding a opäť pozičné kódovanie.

Dimenzionalita embeddingu v dekóderi, ako aj dimenzia plne prepojených vrstiev enkódera bola nastavená na hodnotu 512. Počet blokov enkódera aj dekódera som nastavil na hodnotu 4 a počet hláv v multi-head attention vrstvách na hodnotu 8.

Implementáciu transformera som prebral z tensorflow projektu[3] ktorý sa týkal problému strojového prekladu a upravil som ho aby pracoval s obrázkami. Odstránil som embedding vrstvu zo vstupu enkóderu a pridal som konvolučnú sieť. Pred každou normalizačnou vrstvou sa nachádza ešte dropout vrstva s parametrom rate nastaveným na hodnotu 0.1. Dropout vrstvy sú z ilustračného obrázku vynechané kvôli prehľadnosti. Konceptom napojenia transformera na konvolučnú sieť som sa inšpiroval z článku [40].

[3] https://www.tensorflow.org/tutorials/text/transformer

Kapitola 5

Experimenty

V tejto kapitole sú popísané jednotlivé experimenty s obidvomi navrhnutými modelmi. Jednotlivé modely boli vyhodnocované metrikami CER, WER a SER. Každý z modelov bol trénovaný na sade trénovacích dát, priebežne validovaný na sade validačných dát a po ukončení trénovania otestovaný na testovacích dátach. Primárnym cieľom experimentov bolo preskúmať úspešnosť modelu používajúceho transformer pri rozpoznávaní ručne písaného písma a to hlavne pri riadkoch obsahujúcich dlhšie sekvencie znakov. Existujúce práce [40, 49] kde používajú transformer sa zameriavajú prevažne na rozpoznávanie kratších textov (pár slov v riadku), alebo na rozpoznávanie textov umiestnených v scéne. Súčasťou experimentov bolo aj skúmanie parametrov, ktoré majú na daný model najväčší vplyv a porovnanie modelov používajúcich transformer s modelmi používajúcimi rekurentné neuróny. Experimenty prebiehali na štyroch rozličných datasetoch - *111 let českého dopisu v korpusovém zpracování*, *české dokuemnty*, *IAM* a *MADCAT*, ktoré sú popísané v kapitole 2.6.

5.1 Vplyv zmeny parametrov na úspešnosť modelu

V tomto experimente som porovnával model používajúci transformer a enkóder-dekóder používajúci GRU neuróny. Cieľom tohto experimentu bolo zistiť čo najlepšie nastavenie parametrov siete, aby bola úspešnosť čo najvyššia. Zameral som sa hlavne na hĺbku konvolučnej siete a dimenzionalitu enkóder-dekóder modelov.

Na trénovanie a vyhodnocovanie som použil dataset českých dokumentov, ktorý som dostal od výskumnej skupiny PERO 2.6. Experiment prebiehal v dvoch fázach. Najprv sa na datasete so šírkou obrázkov 224px otestovali vplyvy jednotlivých parametrov na modely a následne prebehol test najúspešnejšieho modelu na datasete so šírkou obrázkov 600px.

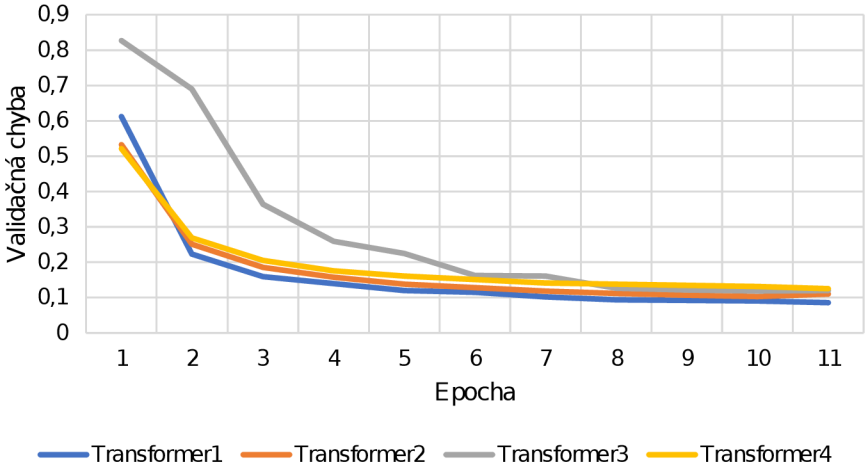
Natrénovanie modelov na datasete českých dokumentov so šírkou obrázka 224px

Skúmanie vplyvu parametrov na transformer som rozdelil do niekoľkých skupín. Ako prvé som pri modele skúmal hĺbku konvolučnej siete. Pri použití iba jednej, alebo viac ako troch konvolučných vrstiev došlo k pretrénovaniu príliš skoro a model takmer vôbec negeneralizoval. Ďalšími pokusmi bola zmena dimenzionality transformera a použitie binarizácie pri vstupných obrázkoch. Experimentoval som s rôznymi typmi binarizácie, v tomto experimente bola použitá Otsu binarizácia [19] z knižnice openCV[1]. Posledným pokusom bolo

[1] https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

Názov modelu	CER	WER	SER
transformer1	**6.3%**	**16.3%**	**35.1%**
transformer2	8.2 %	19.1 %	40.0 %
transformer3	7.9 %	18.4 %	38.7 %
transformer4	10.9 %	23.1 %	44.4 %
CNN_RNN1	17.3 %	35.6 %	65.7 %
CNN_RNN2	23.0 %	44.5 %	75.9 %
CNN_RNN3	20.4 %	41.8 %	72.1 %
CNN_RNN4	15.6 %	34.4 %	63.6 %

Tabulka 5.1: Výsledky modelov natrénovaných na datasete českých dokumentov so šírkou 224



Obrázek 5.1: Vplyv zmeny parametrov modelov používajúcich transformer na priebeh trénovania.

pridanie jednej plne prepojenej vrstvy s dimenzionalitou 512 za konvolučnú sieť. Vychádzal som pritom z implementácie NRTR modelu [40], ktorú som našiel na githube[2].

Prvým modelom *transformer1* bol enkóder-dekóder model s transformerom, ktorého parametre boli nastavené tak, ako sú popísané v návrhu. Druhým modelom *transformer2* bol opäť transformer z návrhu, s tým, že som skúsil zvýšiť počet konvolučných vrstiev na 3. Model *transformer3* obsahoval 2 konvolučné vrstvy, ale za konvolučnou sieťou bola pridaná jedna plne prepojená vrstva s dimenzionalitou 512. Model *transformer4* bol opäť transformer z návrhu, ale pri vytváraní datasetu bola použitá binarizácia obrázkov. Cieľom týchto modifikácií bolo zistiť, aký vplyv má zmena parametrov na úspešnosť modelu. Pre uvedené modely je na grafe 5.1 zobrazená chybovosť na validačných dátach v priebehu trénovania .

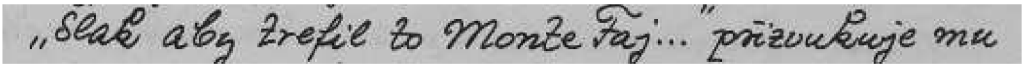
Pri enkóder-dekóder modeli s obojsmerným GRU enkóderom som experimentoval hlavne s dimenzionalitou rekurentných neurónov. Prvé tri enkóder-dekóder modely *CNN_RNN1*,

[2]https://github.com/Belval/NRTR

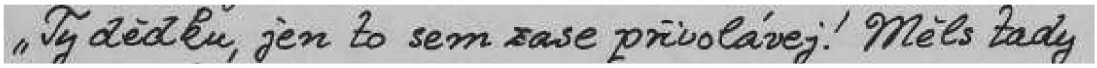
CNN_RNN2, *CNN_RNN3* mali nastavené všetky parametre tak, ako boli popísané v návrhu. Líšili sa iba v dimenzionalite GRU neurónov. *CNN_RNN1* mal nastavenú dimenzionalitu na 512, *CNN_RNN2* na 1024 a *CNN_RNN3* na 256. Štvrtým modelom bol *CNN_RNN4*, ktorého dimenzionalita bola 512 a používal plne prepojenú vrstvu na konci konvolučnej siete.

V tabuľke 5.1 sa nachádzajú výsledky natrénovaných modelov na datasete so šírkou 224px. Z uvedených výsledkov vyplýva, že najúspešnejším modelom bol enkóder-dekóder model *transformer1* s dimenzionalitou 512 a dvoma konvolučnými vrstvami. Zmeny jednotlivých parametrov v transformeri ako počet konvolučných vrstiev, dimenzionalita alebo prítomnosť dense vrstvy na konci konvolučnej siete, nemali veľký vplyv na úspešnosť modelu. Ako môžeme vidieť na grafe 5.1, zmena týchto parametrov urýchlila trénovanie, ale úspešnosť sa výrazne nezmenila. Najhoršia úspešnosť pri trénovaní transformera bola dosiahnutá vtedy, keď boli vstupné obrázky binarizované. Je to pravdepodobne tím, že kvalita naskenovaných obrázkov v datasete je dosť nízka a binarizáciou sa mohli stratiť ešte nejaké ďalšie informácie z pôvodného skenu. Ďalšou možnosťou by bolo odskúšať iné typy binarizačných techník s cieľom zvýšiť úspešnosť trénovania a taktiež ďalšie techniky preprocessingu ako zmena kontrastu, zvýraznenie hrán alebo vyrovnanie sklonu písma.

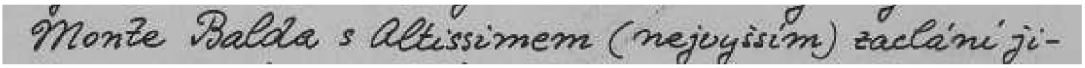
CNN-RNN model bol v porovnaní s transformerom menej úspešný. Najlepší výsledok ktorý bol dosiahnutý z modelu používajúceho rekurentné neuróny, bola CER 15.6%. Veľkou nevýhodou tejto siete je nízka miera paralelizácie modelu. Pre porovnanie, pri trénovaní CNN-RNN modelu s dimenzionalitou 512 trvala jedna epocha priemerne 40 minút a pri trénovaní transformera s dimenzionalitou 512, so 4 enkóder aj dekóder blokmi, trvala epocha 10 minút. Transformer potreboval na pretrénovanie približne trikrát viac epoch ako CNN-RNN. Z uvedeného vyplýva, že rekurentná sieť je menej úspešná a zároveň sa pomalšie trénuje ako sieť používajúca transformer. Tento rozdiel v rýchlosti nie je až taký výrazný, ale so stúpajúcou šírkou obrázka sa bude kvôli slabej schopnosti paralelizovať rekurentné neurónové siete, ešte prehlbovať.



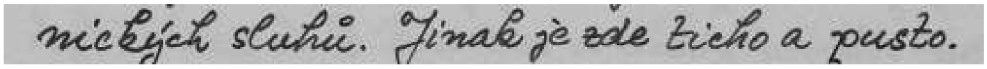
Vygenerovaný prepis: „blak“ aby trefil to Monte Faj... přizvukuje mu
Skutočný prepis: „Šlak aby trefil to Monte Faj…“ přizvukuje mu



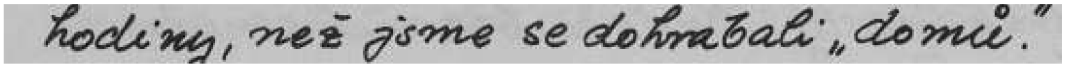
Vygenerovaný prepis: „Ty dědku, jen to sem zase přivolávej! Měls tady
Skutočný prepis: „Ty dědku, jen to sem zase přivolávej! Měls tady



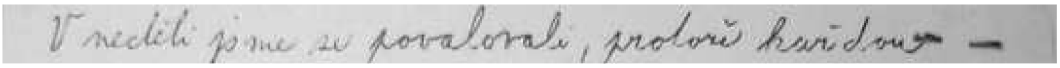
Vygenerovaný prepis: Monte Balda s Altisimem (nejvyším zaelání ji-
Skutočný prepis: Monte Balda s Altissimem (nejvyšším) zaclání ji-



Vygenerovaný prepis: nických sluhů. Jinak je zde ticho a pusto.
Skutočný prepis: nických sluhů. Jinak je zde ticho a pusto.



Vygenerovaný prepis: hodiny, než jsme se dohrabali „domů!
Skutočný prepis: hodiny, než jsme se dohrabali „domů“.



Vygenerovaný prepis: V neděli jsme se povalovali, protože každou -
Skutočný prepis: V neděli jsme se povalovali, protože každou

Vygenerovaný prepis: chvíli pršelo. Strýc Jirka s Pavlíkem odpo-
Skutočný prepis: chvíli pršelo. Strýc Jirka s Pavlíkem odpo

Vygenerovaný prepis: Zejtra tam určitě nepůjdu. Vyrážíme s Luckou
Skutočný prepis: Zejtra tam určitě nepůjdu. Vyrážíme s Luckou

Obrázek 5.2: Ukážky obrázkov z testovacej sady datasetu českých dokumentov spolu s prepismi ktoré vygenerovala neurónová sieť a so skutočnými prepismi z datasetu

Natrénovanie transformera na datasete českých dokumentov so šírkou obrázka 600px

V druhej fáze tohto experimentu bolo cieľom natrénovať model *transformer1*, ktorý bol v predchadzajúcom experimente najúspešnejší, na datasete českých dokumentov so šírkou obrázka 600px. Výsledný model dosiahol na testovacích dátach úspešnosti CER: 8.4%, WER: 23.9% a SER: 73.8%.

Na obrázku 5.2 sa nachádzajú ukážky obrázkov z testovacej sady datasetu českých dokumentov spolu s prepismi ktoré vygenerovala neurónová sieť a so skutočnými prepismi z datasetu. Na obrázku je vidieť, že si sieť poradila s rôznymi štýlmi písma, odlišným sklonom a kvalitou skenu a dokázala rozlíšiť veľké a malé písmena. Niektoré pôvodné prepisy obrázkov obsahovali chyby, pričom ich táto sieť dokázala prepísať správne. Príkladom takýchto prepisov je napríklad 6. a 7. prepis z obrázka 5.2. Referenčné prepisy na konci vety neobsahovali spojovník, pričom na obrázku sa nachádzal. Sieť tento spojovník do prepisov zahrnula.

Časté chyby boli spôsobené podobnosťou písaných písmen. V treťom prepise sieť namiesto slova *zaclání* vygenerovala slovo *zaelání*. Na obrázku sa písmeno *c* v slove *zaclání* podobá na písané písmeno *e*. Podobná situácia nastala v piatom prepise. Na konci vety sieť namiesto znakov *".* vygenerovala znak *!*. Bolo to pravdepodobne spôsobené tím, že pisateľ napísal bodku priamo pod úvodzovky a na obrázku sa to môže naozaj javiť ako výkričník.

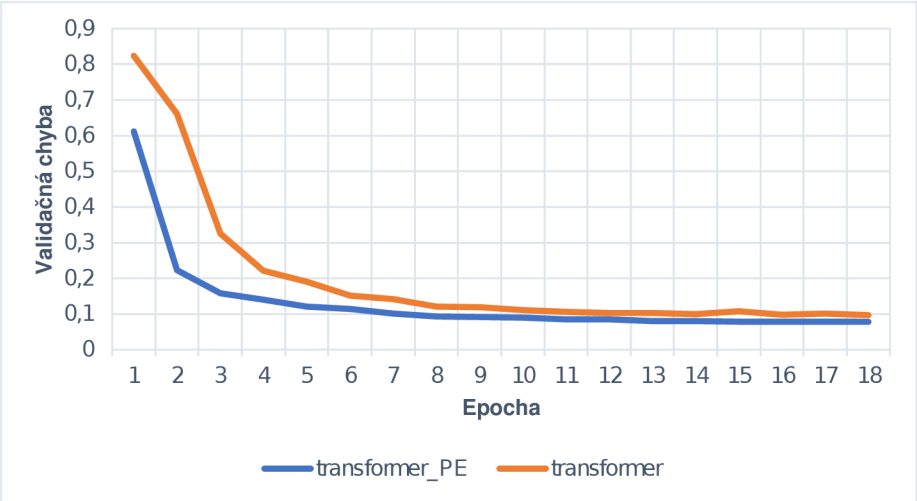
Zhodnotenie

Cieľom tohto experimentu bolo vyhodnotiť vplyv jednotlivých parametrov na úspešnosť daného modelu a porovnať úspešnosť CNN-RNN modelov s modelmi používajúcimi transformer. Z experimentov vyplýva, že príliš veľká, alebo príliš malá dimenzionalita má za následok zhoršenie úspešnosti modelu. Pri vyhodnocovaní modelov som zistil, že modely s konvolučnou sieťou s 2 konvolučnými vrstvami sú pri takomto datasete najúspešnejšie. Zvýšením alebo znížením počtu vrstiev sa úspešnosť modelu znížila, alebo model po krátkom čase prestal generalizovať.

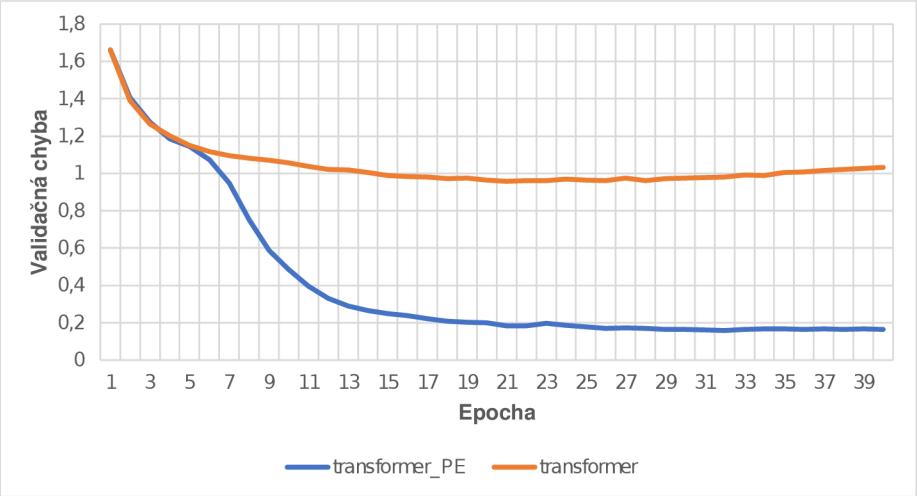
Ďalej je možné vidieť z experimentov to, že transformer má lepšiu úspešnosť ako CNN-RNN model a je možné ho rýchlejšie trénovať. Je to spôsobené tím, že transformer neobsahuje rekurentné neuróny a je možné ho ľahšie paralelizovať.

5.2 Vynechanie pozičného kódovania

V tomto experimente som pozoroval vplyv vynechania pozičného kódovania v transformeri, na strane enkódera. Parametre modelu boli nastavené tak, ako sú popísané v návrhu. Experiment prebiehal na datasete českých dokumentov. Najprv bol model natrénovaný na dátovej sade s maximálnou šírkou 224px, s pozičným kódovaním aj bez neho a následne bol takto trénovaný aj na dátovej sade s maximálnou šírkou 600px. V tabuľke 5.2 sa nachádzajú výsledky úspešnosti týchto modelov. Graf 5.3 zobrazuje priebeh trénovania modelov na datasete s maximálnou veľkosťou obrázkov 224px a graf 5.4 s maximálnou veľkosťou 600px.



Obrázek 5.3: Priebeh trénovania transformera s pozičným kódovaním *transformer_PE* a bez pozičného kódovania *transformer* na strane enkódera. Modely boli trénované na datasete českých dokumentov s maximálnou šírkou 224px.



Obrázek 5.4: Priebeh trénovania transformera s pozičným kódovaním *transformer_PE* a bez pozičného kódovania *transformer* na strane enkódera. Modely boli trénované na datasete českých dokumentov s maximálnou šírkou 600px.

Ako je možné vidieť na grafe 5.3, pri trénovaní modelov na obrázkoch s maximálnou šírkou 224px, odobratie pozičného kódovania z enkódera nemalo na výslednú úspešnosť veľký vplyv. Po natrénovaní sa hodnoty CER líšili iba približne o 1%. Pozičné kódovanie prispelo k tomu, že model začal rýchlejšie generalizovať. Rozdiel nastal pri trénovaní modelov na obrázkoch s maximálnou šírkou 600px, ako je to zobrazené na grafe 5.4. Transformer ktorý mal vynechané pozičné kódovanie na strane enkódera sa po pár epochách pretrénoval a jeho hodnota CER dosahovala iba 84.6%.

Tento experiment ukazuje dôležitosť použitia pozičného kódovania pri trénovaní modelov používajúcich konvolučnú sieť a transformer pri riešení rozpoznávania ručne písaného písma.

Maximálna šírka obrázkov	Pozičné kódovanie	CER	WER	SER
224px	áno	6.3%	16.3%	35.1%
224px	nie	7.5%	18.8%	39.6%
600px	áno	8.4%	23.9%	73.8%
600px	nie	84.6%	105.7%	97.8%

Tabulka 5.2: Vyhodnotenie úspešnosti modelov na datasete 11 let českého dopisu v korpusovém zpracování

Názov modelu	CER	WER	SER
transformer1	**15.1%**	**34.7%**	**86.2%**
transformer2	24.1 %	48.7 %	93.5 %
transformer3	21.3 %	45.8 %	93.0 %
transformer4	33.1 %	60.8 %	94.6 %
transformer5	18.1 %	58.8 %	92.6 %

Tabulka 5.3: Vyhodnotenie úspešnosti modelov na datasete 11 let českého dopisu v korpusovém zpracování

5.3 Vplyv zmeny hĺbky enkóderu a dekóderu na úspešnosť modelu

Cieľom tohto experimentu bolo vyhodnotiť ako sa mení úspešnosť modelu používajúceho transformer, vzhľadom na zmenu počtu enkóder alebo dekóder blokov. Modely boli trénované na datasete *111 let českého dopisu v korpusovém zpracování*. Pracoval som s maximálnou šírkou obrázka 400px a výškou 32px. Počet obrázkov v datasete po vylúčení obrázkov so šírkou väčšou ako 400px bol 42 000, pričom neboli rozdelené na trénovacie validačné a testovacie dáta. Preto som z týchto dát oddelil 10% pre validačné a 10% pre testovacie dáta.

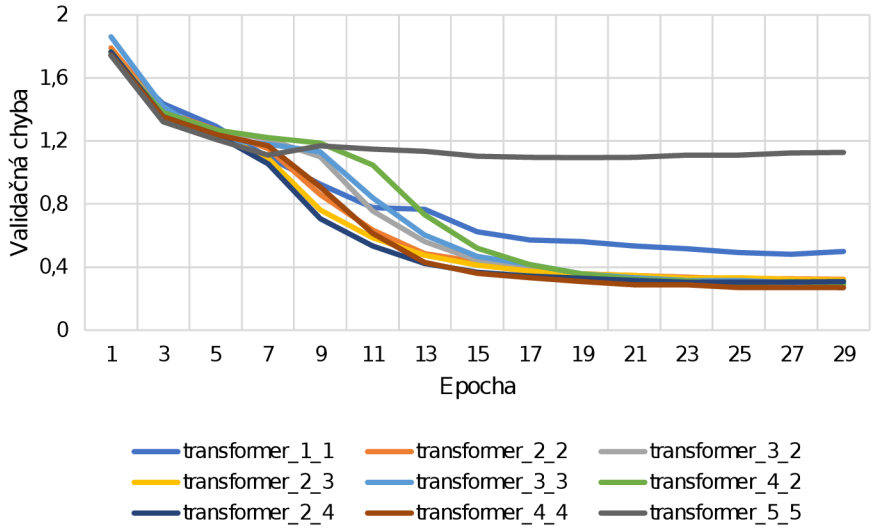
V experimente som najprv skúsil nájsť parametre pri ktorých model dosahuje najlepšiu úspešnosť pri tomto konkrétnom datasete a následne som skúšal meniť počet enkóderov a dekóderov s cieľom zistiť ich vplyv na výslednú úspešnosť.

Úspešnosť jednotlivých modelov sa nachádza v tabuľke 5.3. Parametre modelov boli nastavené podobne ako v predchádzajúcom experimente. Model *transformer1* mal nastavené parametre tak, ako sú zadefinované v návrhu. *Transformer2* obsahoval 3 konvolučné vrstvy a *transformer3* obsahoval 2 konvolučné vrstvy za ktorými nasledovala plne prepojená vrstva. Model *transformer4* mal nastavené parametre ako boli zadefinované v návrhu, ale obrázky z datasetu boli pred trénovaním binarizované. V poslednom modeli *transformer5* boli parametre nastavené tak ako boli uvedené v návrhu, ale kvôli relatívne nízkemu počtu trénovacích dát bola použitá augmentácia dát.

Z uvedených modelov bol najúspešnejší *transformer1*. Preto na tomto modeli bola testovaná úspešnosť vzhľadom na počet enkóderov a dekóderov, ktorú znázorňuje tabuľka 5.4. Na grafe 5.5 je znázornený priebeh trénovania jednotlivých modelov na validačných dátach. Názvy modelov majú tvar *transformer_x_y*, pričom x predstavuje počet enkóder blokov a y počet dekóder blokov. Ak bol počet enkóderov alebo dekóderov nastavený na viac ako 4, došlo hneď po niekoľkých epochách k pretrénovaniu. Je to pravdepodobne spôsobené

Názov modelu	CER	WER	SER
transformer_1_1	32.4 %	58.5 %	95.6 %
transformer_2_2	20.4 %	44.7 %	92.7 %
transformer_3_2	18.9 %	43.1 %	94.8 %
transformer_2_3	21.3 %	46.5 %	95.1 %
transformer_3_3	16.1 %	39.1 %	92.0 %
transformer_4_2	16.0 %	42.7 %	91.7 %
transformer_2_4	17.2 %	42.6 %	93.7 %
transformer_4_4	**15.1%**	**37.8%**	**90.7%**
transformer_5_5	-	-	-

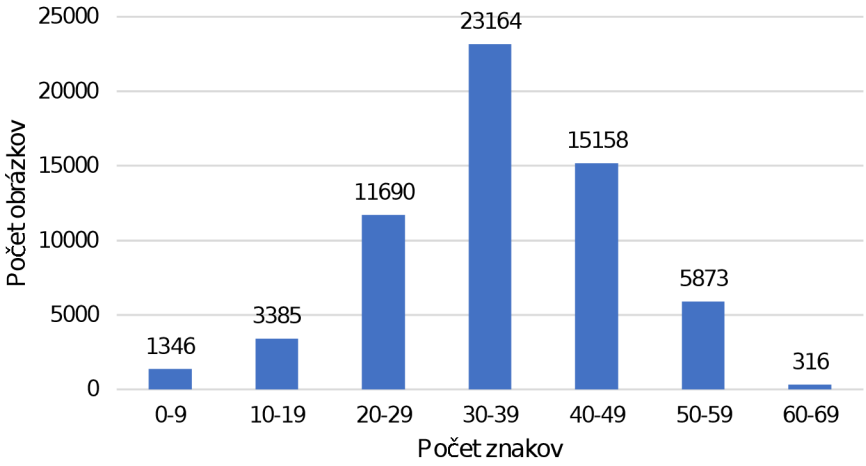
Tabulka 5.4: Priebeh trénovania modelov používajúcich transformer, pri použití rôznych nastavení počtu enkóder a dekóder blokov. Názvy modelov sú v tvare *transformer_x_y*, pričom hodnota x predstavuje počet enkóder blokov a hodnota y počet dekóder blokov



Obrázek 5.5: Úspešnosť modela v závislosti od počtu enkóder a dekóder blokov

tým, že bol použitý dataset s malým počtom trénovacích dát. Takéto modely ani neboli vyhodnocované metrikami na testovacích dátach. Malý počet enkóder alebo dekóder blokov zase viedol k zhoršeniu úspešnosti siete. V článku [40] v experimentoch prišli na to, že modely kde je väčší počet enkóderov ako dekóderov majú vyššiu úspešnosť. V tomto experimente to bolo potvrdené (*transformer_4_2* vs. *transformer_2_4* a *transformer_3_2* vs. *transformer_2_3*). Rozdiel v úspešnosti takýchto modelov bol relatívne malý, čo môže byť spôsobené malou trénovacou sadou. Väčšia úspešnosť modelu ktorý má viac enkóder blokov ako dekóder blokov je pravdepodobne dosiahnutá tým, že informácia obsiahnutá v obrázku je komplexnejšia ako informácia obsiahnutá v danom prepise.

Ako môžeme vidieť na obrázku 5.5, model s počtom blokov 5 sa učil zo všetkých modelov najrýchlejšie, ale po deviatej epoche došlo k pretrénovaniu a prestal generalizovať. Druhým extrémom bol model s počtom blokov 1. Priebeh trénovania mal po 9. epochu



Obrázek 5.6: Zastúpenie počtu znakov v prepisoch obrázkov z datasetu

približne rovnaký ako ostatné modely, no po deviatej epoche sa učenie spomalilo a začal sa pretrénovavať. Modely ktoré mali menší počet enkóder blokov ako dekóder blokov (*transformer_2_3* a *transformer_2_4*) konvergovali rýchlejšie, ale k pretrénovaniu došlo skôr a preto boli menej úspešné ako modely s väčším počtom enkóder blokov. Najúspešnejším modelom bol $transformer\_4\_4$ s chybovosťou 15.1%.

5.4 Úspešnosť transformera pri pracovaní s dlhými riadkami

V tomto experimente bol použitý model z návrhu - transformer s konvolučnou sieťou s dvoma konvolučnými vrstvami a dimenzionalitou 512. Cieľom bolo pozorovať úspešnosť modelu vzhľadom na maximálnu šírku obrázkov v datasete a zistiť, či má použitie transformera nejaké nevýhody pri práci s dlhou vstupnou sekvenciou.

V tomto experimente bol použitý opäť dataset *111 let českého dopisu v korpusovém zpracování*. Tabuľka 5.5 zobrazuje najlepšiu úspešnosť ktorá bola dosiahnutá na testovacích dátach, v závislosti od nastavenej maximálnej šírky obrázka. Obsahuje tiež počet trénovacích dát, ktoré so zvyšovaním maximálnej šírky obrázka pribúdali. Na obrázku 5.6 je zobrazený histogram, ktorý znázorňuje počet obrázkov z trénovacej sady, ktorých prepis má daný počet znakov. Vzhľadom na rozloženie počtu znakov v obrázkoch, som v datasete ponechal len obrázky ktorých prepis nebol dlhší ako 60 znakov.

Ako je možné vidieť v tabuľke 5.5, zväčšovaním maximálnej šírky obrázka nemalo vplyv na úspešnosť modelu. Hodnota metriky CER sa pohybovala medzi hodnotami 11%-13%. Pri maximálnej šírke 400px bola hodnota CER najvyššia - 15.1%, čo spôsobilo pravdepodobne málo trénovacích dát (33 905).

5.5 Porovnanie modelov na IAM datasete

Cieľom tohto experimentu bolo porovnať enkóder-dekóder model používajúci transformer, s GRU enkóder-dekóder modelom na dátovej sade IAM. V tomto experimente som sa tiež zameral na dimenzionalitu GRU enkóder-dekóder modelu a porovnanie modelov ktoré pou-

Šírka obrázka	Počet trénovacích dát	CER	WER	SER
400px	33 905	15.1%	34.7%	86.2%
500px	47 074	12.9%	31.9%	90.1%
600px	52 861	13.8%	31.0%	90.2%
700px	59 376	12.7%	29.3%	86.8%
800px	60 585	11.7%	28.0%	85.6%
900px	60 932	12.7%	29.3%	85.5%

Tabulka 5.5: Úspešnosť modelov vzhľadom na počet enkóder a dekóder blokov

Názov modelu	CER	WER	SER
CNN_RNN_256	30.2%	40.7%	40.7%
CNN_RNN_512	17.3%	32.5%	32.5%
CNN_RNN_1024	17.8%	32.6%	32.6%
CNN_RNN_simple_1	18.3%	34.0%	34.0%
CNN_RNN_simple_2	18.2%	34.5%	34.5%
CNN_RNN_simple_3	19.7%	34.0%	34.0%
transformer	**11.8%**	**22.1%**	**22.1%**

Tabulka 5.6: Úspešnosť modelov pri natrénovaní na datasete IAM

žívajú obojsmerné GRU s jednovrstvovými, dvojvrstvovými a trojvrstvovými GRU enkóder-dekóder modelmi.

V tabuľke 5.6 sú zobrazené úspešnosti jednotlivých modelov na testovacej sade IAM datasetu. Z IAM datasetu som použil sadu, v ktorej boli vysegmentované z pôvodných dokumentov jednotlivé slová. Výšku obrázkov som upravil na 32px a maximálnu šírku som obmedzil na 200px. Model *CNN_RNN_256* bol GRU enkóder-dekóder model popísaný v návrhu, pričom jeho dimenzionalita bola nastavená na hodnotu 256. *CNN_RNN_512* a *CNN_RNN_1024* mali nastavenú dimenzionalitu na 512 a 1024.

Model *CNN_RNN_simple_1* neobsahoval obojsmerný GRU enkóder a jeho dimenzionalita bola 1024. Modely *CNN_RNN_simple_2* a *CNN_RNN_simple_3* neobsahovali obojsmerné enkódery, ale ich enkóder bol zložený z dvoch a troch vrstiev GRU neurónov. Dimenzionalita modelu *CNN_RNN_simple_2* bola 1024 a modelu *CNN_RNN_simple_3* 512. Pre porovnanie bol natrénovaný jeden model používajúci transformer so 4 enkódermi a 4 dekódermi s dimenzionalitou 512. Ten je označený v tabuľke ako *transformer*. Obrázky boli pred trénovaním binarizované pomocou Otsu binarizácie z knižnice openCV.

Ako bolo spomínané vyššie, v datasete IAM je okolo každého písmena šum, ktorý vznikol dôsledkom orezávania písmen z pôvodného dokumentu. V tomto prípade binarizácia zlepšila výsledky približne o 5% hodnoty CER.

V tomto experimente bol opäť najúspešnejším modelom transformer, ktorý dosiahol hodnotu CER 11.8%. Rýchlosť trénovania tranformera bola vyššia no na natrénovanie potreboval transformer viac času. Jedna epocha pri trénovaní transformera trvala 1.7 minúty a pri 512 dimenzionálnom GRU enkóder-dekóderi 3.3 minúty. Na natrénovanie GRU modelu

Typ tokenizácie	Počet tokenov	Dĺžka sekvencie	CER	WER	SER
Tokenizácia znakov	120	62	11.7%	28.0%	85.7%
Tokenizácia častí slov	150	46	13.8%	31.0%	90.2%
Tokenizácia častí slov	300	41	20.6%	41.1%	89.1%
Tokenizácia častí slov	500	37	26.2%	47.8%	92.7%
Tokenizácia častí slov	1000	35	40.2%	62.1%	95.9%
Tokenizácia slov	47 661	17	-	-	-

Tabulka 5.7: Úspešnosť transformera pri použití rôznych typov tokenizácie

boli potrebné 4 epochy a na natrénovanie transformera 14 epoch. Trénovanie transformera zabralo teda približne dvakrát viac času ako trénovanie GRU modelu.

Z modelov ktoré používali GRU neuróny boli najúspešnejšie enkóder dekóder modely s obojsmernými GRU enkódermi s dimenzionalitou 512 a 1024. Pridanie ďalšej GRU vrstvy k enkóderu modelu *CNN_RNN_simple_1* neviedlo k zlepšeniu úspešnosti.

V tabuľke 5.6 je možné vidieť, že hodnoty metrík WER a SER sa takmer nelíšia. V predchádzajúcom experimente boli hodnoty SER niekedy až trojnásobné oproti hodnotám WER. Je to spôsobené tým, že na obrázkoch z IAM datasetu sa nachádza iba jedno, prípadne dve slová a tým sa každá chyba v slove hneď prejaví aj na metrike SER.

5.6 Rozličné typy tokenizácie

V tomto experimente boli porovnávané jednotlivé typy tokenizácie pri trénovaní modelu s transformerom. Použitý model bol enkóder-dekóder model s transformerom z návrhu. Porovnávala sa tokenizácia na úrovni znakov, celých slov a častí slov. Pri tokenizácii na úrovni častí slov bola použitá metóda BPE. Porovnávanie prebiehalo na datasete *111 let českého dopisu v korpusovém zpracování*. Maximálna šírka obrázkov bola obmedzená na 800px. V tabuľke 5.7 sa nachádzajú výsledky pri trénovaní modelu s transformerom, s rozličnými typmi tokenizácie.

Úspešnosť tokenizácie na úrovni slov v tabuľke uvedená nieje. Model sa hneď po pár epochách pretrénoval a prestal generalizovať. Príčinou môže byť malý dataset, ktorý obsahoval 60 585 prepisov. Počet unikátnych tokenov v datasete bol 47 661. Ďalším problémom bolo to, že vo validačnom a testovacom datasete bolo veľké množstvo $< OOV >$ tokenov, čo zhoršovalo validáciu. Tokenizácia pomocou slov robí často problémy pri jazykoch, ktoré používajú skloňovanie (napr. slovenčina, čeština). V tomto prípade sa potom všetky tvary toho istého slova považujú za samostatné tokeny. To prispelo tiež k tomu, že sa model nepodarilo natrénovať.

Najúspešnejšia bola tokenizácia pomocou znakov. V tomto prípade bol počet unikátnych tokenov 120. Pri použití tokenizácie na úrovni časti slov, sa s narastajúcou veľkosťou slovníka unikátnych tokenov zhoršovala úspešnosť modelu. Je to pravdepodobne tiež spôsobené malou trénovacou sadou. S rastúcou dĺžkou tokenov totiž klesá počet výskytov daného tokenu v trénovacích dátach. Pri zvyšovaní počtu unikátnych tokenov by teda mala byť použitá dostatočne veľká trénovacia sada.

5.7 Rozpoznávanie nelatinského písma

Cieľom tohto experimentu bolo natrénovanie a vyhodnotenie modelu používajúceho transformer na datasete prepisov, ktoré neboli napísané latinským písmom. V tomto experimente bol použitý dataset MADCAT [9], konkrétne časť obsahujúca arabské dokumenty. Ako bolo spomínané v kapitole 2.6, arabské písmo sa líši od latinského hlavne tým, že nepoužíva veľké a malé písmena, píše sa sprava doľava a jednotlivé písmená majú rozličné tvary v závislosti od polohy v slove. Cieľom bolo taktiež zistiť, či tieto vlastnosti majú nejaký vplyv na trénovanie modelu používajúceho transformer.

Použitý bol model so 4 enkóder a dekóder blokmi, 2 konvolučnými vrstvami a dimenzionalitou 512. Jednotlivé obrázky riadkov boli zmenšené na výšku 32px a maximálna šírka obrázkov bola nastavená na 500px. MADCAT dataset mi bol poskytnutý od výskumnej skupiny PERO, pričom nebol rozdelený na trénovacie, testovacie a validačné dáta, preto som ho rozdelil na 646 171 trénovacích dát a 35 895 validačných aj testovacích dát.

Na natrénovanie modelu bolo potrebných 10 epoch, pričom jedna epocha trvala 25 minút. Výsledný model dosahoval úspešnosť CER 4.4%, WER 15.7% a SER 50.1%.

Text písaný sprava doľava a odlišné tvary tých istých písmen vzhľadom na polohu písmena v slove nemali výrazný vplyv na trénovanie modelu. Na tomto datasete bola dokonca dosiahnutá najlepšia úspešnosť zo všetkých použitých dátových sád. Vysoká úspešnosť na tomto datasete bola dosiahnutá pravdepodobne tým, že MADCAT je zo všetkých použitých datasetov najväčší a obrázky v ňom sú naskenované vo vysokej kvalite (600dpi).

5.8 Zhrnutie

Experimenty z predchádzajúcich kapitol ukazujú, že transformer spolu s konvolučnou sieťou je vhodným modelom na riešenie rozpoznávania ručne písaného textu. Pre uvedené datasety bola najvhodnejšia konfigurácia modelu s transformerom, kedy bol počet enkóderov aj dekóderov nastavený na hodnotu 4, dimenzionalita na hodnotu 512 a počet konvolučných vrstiev na hodnotu 2. Experiment 5.2 preukázal, aké dôležité je pozičné kódovanie pri použití modelu s transformerom, hlavne pri obrázkoch s väčšou šírkou.

V experimentoch bolo preukázané aj to, že transformer dokázal pracovať s dlhými sekvenciami a zvyšovaním maximálnej šírky obrázka sa úspešnosť modelu vôbec nezhoršovala.

V porovnaní s modelmi ktoré používajú konvolučnú sieť a GRU enkóder-dekóder model s attention mechanizmom, vykazoval transformer vždy lepšie výsledky. Príčinou mohlo byť aj to, že táto práca bola zameraná primárne na prácu s transformerom a na podrobnejšie vyľaďovanie GRU enkóder-dekóder modelu som sa nezameriaval. Rýchlosť trénovania bola však u transformera takisto vyššia. Je to spôsobené tým, že transformer nepoužíva rekurentné neuróny a tým je možné ho ľahšie paralelizovať.

Pri experimentovaní s tokenizáciou bola pri uvedenom datasete najúspešnejšia tokenizácia na úrovni písmen. Tento experiment by však bolo vhodné zopakovať na oveľa väčšom datasete, na ktorom by sa možno ukázalo, že tokenizácia na úrovni časti slov je pre väčšie datasety výhodnejšia.

5.9 Návrhy na vylepšenia

Existuje mnoho postupov, ktoré by mohli zlepšiť úspešnosť OCR systému používajúceho enkóder-dekóder model. V tejto práci som nekládol veľký dôraz na fázu preprocessingu,

pretože obrázky v datasetoch boli naskenované v relatívne dobrej kvalite. Vo fáze preprocessingu som experimentoval iba s binarizáciou. Pre dosiahnutie ešte lepších výsledkov by mohlo byť na obrázok použité odstránenie šumu (noise removal), oprava natočenia skenu (skew correction), zlepšenie kontrastu a vyrovnanie kurzívneho písma.

Ďalším vylepšením by mohlo byť použitie metódy beam search [12]. Veľa modelov založených na sequence to sequence princípe, generujú dekóderom na výstupe tokeny na základe rozdelenia pravdepodobností nad slovnou zásobou (slovníkom), ktorú systém používa. Pre každé slovo zo slovníka je určená pravdepodobnosť, s akou sa bude nachádzať na výstupe na danom mieste v sekvencii. Greedy search je prístup, kedy sa po každom vygenerovaní pravdepodobností nad slovnou zásobou vezme slovo s najvyššou pravdepodobnosťou. Tento prístup je efektívny, ale v mnohých prípadoch nemusí vykazovať najlepšie výsledky. Beam search sa vo veľa prípadoch javí ako lepší než greedy search. Metóda beam search funguje na princípe hľadania k najpravdepodobnejších sekvencií vygenerovaných dekóderom. V každom kroku generovania nového slova dekóderom, sú vygenerované všetky možnosti a ponecháva sa len k tých, ktoré sú najpravdepodobnejšie. Greedy search je teda špeciálny prípad metódy beam search, kedy je k nastavené na hodnotu 1. Nevýhodou metódy beam search oproti greedy search je zložitejšia implementácia a vyššia časová náročnosť pri generovaní tokenov dekóderom. Kvalita výstupných sekvencií je však oproti greedy search väčšinou vyššia.

Využitie postprocessingu by mohlo taktiež zvýšiť úspešnosť OCR systému. Možnosťou sú staršie slovníkovo zamerané metódy, ktoré porovnávajú vygenerované slová so slovami v slovníku, alebo pokročilejšie kontextovo zamerané metódy, ktoré používajú jazykové modely a berú ohľad na kontext daného slova vo vete [3]. V roku 2019 Jacob Devlin a kol. v článku [10] predstavili nový jazykový model BERT (Bidirectional Encoder Representations from Transformers). Vďaka tomuto modelu boli prekonané mnohé dovtedy najlepšie výsledky v rôznych oblastiach spracovania prirodzeného jazyka. BERT môže byť použitý vo fáze postprocessingu. Po vygenerovaní vety OCR systémom sa detekuje potencionálne chybné slovo vo vete a nahradí sa špeciálnym tokenom $<MASK>$. Následne je táto veta spracovávaná modelom BERT, ktorý vyhľadá token s maskou a snaží sa predikovať najpravdepodobnejšie slovo na danej pozícii, vzhľadom na kontext poskytnutý z okolitých slov [20].

Kapitola 6

Záver

Táto práca bola venovaná problematike rozpoznávania ručne písaného textu. Hlavnou úlohou bolo získať prehľad o súčasných metódach tvorby OCR systémov založených na hlbokých neurónových sieťach. Súčasťou práce bol aj návrh dvoch modelov, ktoré môžu byť použité pri riešení tejto problematiky. Práca je zameraná na riešenie tejto problematiky pomocou sequence to sequence prístupu.

Táto práca obsahuje stručný úvod do problematiky OCR systémov, popisuje datasety ktoré môžu byť použité pri trénovaní OCR modelov a najčastejšie používané metriky pri vyhodnocovaní týchto modelov.

Súčasťou práce je návrh dvoch modelov. Prvým modelom je enkóder-dekóder používajúci GRU rekurentné neuróny spolu s attention mechanizmom. Druhým modelom je enkóder-dekóder model nazývaný transformer, ktorý bol predstavený koncom roka 2017 a v problematike OCR je ešte stále novinkou.

Výsledkom práce je sada experimentov vykonaných na štyroch rozdielných datasetoch vyplýva, že transformer je úspešnejší a rýchlejší ako enkóder-dekóder model s rekurentnými neurónmi. Je to spôsobené tým, že transformer neobsahuje rekurentné neuróny a teda je jednoduché paralelizovať jeho výpočet. Celú vstupnú sekvenciu spracováva paralelne a preto, ako sa ukázalo v experimentoch, potrebuje na vstup aplikovať pozičné kódovanie, aby mala sieť predstavu o pozícií jednotlivých tokenoch v sekvencii. Výhodou transformera je aj to, že dokáže pracovať s relatívne dlhými sekvenciami, pričom sa mu úspešnosť nezhorší.

Ako ukázali experimenty, transformer v spolupráci s konvolučnou sieťou je vhodným modelom pre rozpoznávanie ručne písaného písma. Dokáže rozpoznávať dlhé a nekvalitne naskenované riadky s relatívne dobrou úspešnosťou, pričom dataset môže obsahovať aj písané aj tlačené písma s rôznym sklonom kvalitou a štýlom písania. Práca taktiež obsahuje niekoľko typov, ktoré by mohli ešte pomôcť vylepšiť úspešnosť siete.

Literatura

[1] Bahdanau, D., Cho, K. a Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014.

[2] Barrett, D. *Adrian Frutiger (1928–2015): Univers and OCR-B* [https://multimediaman.blog/2015/12/21/adrian-frutiger-1928-2015-univers-and-ocr-b/]. [cit. 2020-04-22].

[3] Bassil, Y. a Alwani, M. OCR Post-Processing Error Correction Algorithm using Google Online Spelling Suggestion. *CoRR*. 2012, abs/1204.0191.

[4] Bengio, Y., Simard, P. a Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*. March 1994, roč. 5, č. 2, s. 157–166. ISSN 1941-0093.

[5] Brownlee, J. *What is Teacher Forcing for Recurrent Neural Networks?* 2017.

[6] Carrasco, R. C. *Text Digitisation*. [cit. 2020-01-22]. Dostupné z: https://sites.google.com/site/textdigitisation/qualitymeasures/computingerrorrates.

[7] Chammas, E., Mokbel, C. a Likforman-Sulem, L. Handwriting Recognition of Historical Documents with few labeled data. *CoRR*. 2018, abs/1811.07768.

[8] Cho, K., Merrienboer, B. van, Gülçehre, Ç., Bougares, F., Schwenk, H. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*. 2014, abs/1406.1078.

[9] consortium, L. data. *Multilingual Automatic Document Classification, Analysis and Translation (MADCAT)*. [cit. 2020-01-25]. Dostupné z: https://www.ldc.upenn.edu/collaborations/current-projects/madcat.

[10] Devlin, J., Chang, M., Lee, K. a Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. 2018, abs/1810.04805. Dostupné z: http://arxiv.org/abs/1810.04805.

[11] Eikvil, L. *Optical Character Recognition*. 2015. Dostupné z: http://www.nr.no/~eikvil/OCR.pdf.

[12] Freitag, M. a Al-Onaizan, Y. Beam Search Strategies for Neural Machine Translation. *CoRR*. 2017, abs/1702.01806.

[13] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. et al. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2009, roč. 31, č. 5, s. 855–868.

[14] Guyon, I., Haralick, R. M., Hull, J. J. a Phillips, I. T. Data Sets For OCR And Document Image Understanding Research. In: *In Proceedings of the SPIE - Document Recognition IV*. World Scientific, 1997, s. 779–799.

[15] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. *CoRR*. 2015, abs/1512.03385. Dostupné z: http://arxiv.org/abs/1512.03385.

[16] Helmke, H., Ehr, H., Kleinert, M., Faubel, F. a Klakow, D. Increased Acceptance of Controller Assistance by Automatic Speech Recognition. In:. červen 2013.

[17] Hladká, Z. a. k. *111 let českého dopisu v korpusovém zpracování*. 2013.

[18] Holley, R. *How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs*. [cit. 2020-04-22].

[19] Horan, A. C. *Tokenizers: How machines read*. [cit. 2020-01-30]. Dostupné z: https://medium.com/@hbyacademic/otsu-thresholding-4337710dc519.

[20] Ilango, R. *Using NLP (BERT) to improve OCR accuracy* [https://medium.com/states-title/using-nlp-bert-to-improve-ocr-accuracy-385c98ae174c]. [cit. 2019-12-18].

[21] Kang, L., Toledo, J. I., Riba, P., Villegas, M., Fornés, A. et al. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In: *GCPR*. 2018.

[22] Khandelwal, R. *Intuitive explanation of Neural Machine Translation*. [cit. 2020-02-25]. Dostupné z: https://towardsdatascience.com/intuitive-explanation-of-neural-machine-translation-129789e3c59f.

[23] Khurana, S. *Applications of OCR You Haven't Thought Of* [https://medium.com/swlh/applications-of-ocr-you-havent-thought-of-69a6a559874b]. 2018.

[24] Kudo, T. a Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. *CoRR*. 2018, abs/1808.06226. Dostupné z: http://arxiv.org/abs/1808.06226.

[25] Lecun, Y., Bottou, L., Bengio, Y. a Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. Nov 1998, roč. 86, č. 11, s. 2278–2324. ISSN 1558-2256.

[26] Luong, M., Pham, H. a Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. *CoRR*. 2015, abs/1508.04025. Dostupné z: http://arxiv.org/abs/1508.04025.

[27] Ma, E. *3 subword algorithms help to improve your NLP model performance.* 2019.

[28] "Manoj, S. a "Narendra, S. A SURVEY ON HANDWRITTEN CHARACTER RECOGNITION (HCR) TECHNIQUES FOR ENGLISH ALPHABETS. *Signal & Image Processing : An International Journal (SIPIJ).* 2016, roč. 3.

[29] Mars, A. *Optical Character Recognition For Arabic language using neural network.* [cit. 2020-01-22].

[30] Marti, U.-V. a Bunke, H. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition.* 2002, roč. 5, s. 39–46.

[31] Michael, J., Labahn, R., Grüning, T. a Zöllner, J. Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition. *CoRR.* 2019, abs/1903.07377.

[32] Olah, C. *Neural Networks, Types, and Functional Programming.* [cit. 2020-01-22]. Dostupné z: http://colah.github.io/posts/2015-09-NN-Types-FP.

[33] Pham, V., Kermorvant, C. a Louradour, J. Dropout improves Recurrent Neural Networks for Handwriting Recognition. *CoRR.* 2013, abs/1312.4569.

[34] Prabhakar, P., Anupama, P. a Resmi, S. R. Automatic vehicle number plate detection and recognition. In: *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT).* 2014, s. 185–190.

[35] Prabhavalkar, R., Rao, K., Sainath, T. N., Li, B., Johnson, L. et al. A Comparison of Sequence-to-Sequence Models for Speech Recognition. In: *INTERSPEECH.* 2017.

[36] Pradeep, J., Srinivasan, E. a Himavathi, S. Diagonal Based Feature Extraction for Handwritten Alphabets Recognition System using Neural Network. *ArXiv e-prints.* Mar 2011, s. arXiv:1103.0365.

[37] Schuster, M. a Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing.* Nov 1997, roč. 45, č. 11, s. 2673–2681. ISSN 1941-0476.

[38] Selvaraj, C. a Bhalaji, N. Enhanced portable text to speech converter for visually impaired. *IJISTA.* 2018, roč. 17, s. 42–54.

[39] Sennrich, R., Haddow, B. a Birch, A. Neural Machine Translation of Rare Words with Subword Units. *CoRR.* 2015, abs/1508.07909.

[40] Sheng, F., Chen, Z. a Xu, B. NRTR: A No-Recurrence Sequence-to-Sequence Model For Scene Text Recognition. *CoRR.* 2018, abs/1806.00926.

[41] Sundermeyer, M., Schlüter, R. a Ney, H. LSTM Neural Networks for Language Modeling. In: *INTERSPEECH.* 2012.

[42] Sutskever, I., Vinyals, O. a Le, Q. V. Sequence to Sequence Learning with Neural Networks. *CoRR.* 2014, abs/1409.3215.

[43] Technology, A. *Optical character recognition - History.* 2016.

[44] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention Is All You Need. *CoRR.* 2017, abs/1706.03762.

[45] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R. J., Darrell, T. et al. Sequence to Sequence - Video to Text. *CoRR.* 2015, abs/1505.00487.

[46] Voigtlaender, P., Doetsch, P. a Ney, H. Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks. *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR).* 2016, s. 228–233.

[47] Vukotić, V., Raymond, C. a Gravier, G. A Step Beyond Local Observations with a Dialog Aware Bidirectional GRU Network for Spoken Language Understanding. In: *Interspeech 2016.* 2016, s. 3241–3244. Dostupné z: http://dx.doi.org/10.21437/Interspeech.2016-1301.

[48] Wang, P., Yang, L., Li, H., Deng, Y., Shen, C. et al. A Simple and Robust Convolutional-Attention Network for Irregular Text Recognition. *CoRR.* 2019, abs/1904.01375.

[49] Wang, P., Yang, L., Li, H., Deng, Y., Shen, C. et al. A Simple and Robust Convolutional-Attention Network for Irregular Text Recognition. *CoRR.* 2019, abs/1904.01375. Dostupné z: http://arxiv.org/abs/1904.01375.

[50] Xue, Y. Optical Character Recognition. In:. 2014.

[51] Yann LeCun, C. J. B. *THE MNIST DATABASE of handwritten digits, Analysis and Translation (MADCAT).* [cit. 2020-02-03]. Dostupné z: http://yann.lecun.com/exdb/mnist.

[52] Yin, J., Jiang, X., Lu, Z., Shang, L., Li, H. et al. Neural Generative Question Answering. *CoRR.* 2015, abs/1512.01337.

Příloha A

Obsah priloženého DVD

Priložené DVD obsahuje nasledujúcu adresárovú štruktúru:

- *dp.pdf* – elektronická verzia tejto práce.
- Program – adresár obsahujúci zdrojové kódy.
- Latex – adresár obsahujúci latex súbory pre vygenerovanie pdf.
- Video – Propagačné video k tejto práci