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 HISTORICKÝCH TEXTŮ POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ

CONVOLUTIONAL NETWORKS FOR HISTORIC TEXT RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PETER VEŠELÍNY**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN KIŠŠ**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Vešelíny Peter**
Program: Informační technologie
Název: **Rozpoznávání historických textů pomocí hlubokých neuronových sítí**
Convolutional Networks for Historic Text Recognition
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy konvolučních sítí a rozpoznávání textu.
2. Vytvořte si přehled o současných metodách rozpoznávání textu pomocí konvolučních sítí.
3. Vyberte nebo navrhněte metodu aplikovatelnou na rozpoznávání historických textů.
4. Obstarejte si databázi vhodnou pro experimenty se zaměřením na gotická písma.
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:

- Joan Puigcerver: Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kišš Martin, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Táto práca sa zaoberá rozpoznávaním riadkov z historických textov. Historické texty pochádzajú z obdobia od 17. až 19 storočia a sú napísané pomocou fraktúry. Pri rozpoznávaní písma sa používa architektúra neurónovej siete zvaná *sequence-to-sequence*. Táto architektúra vychádza z modelu kodér-dekodér a používa mechanizmus *attention*. V rámci práce bola z textov, pochádzajúcich z archívu *Deutsches Textarchiv*, vytvorená dátová sada. Tento archív obsahuje 3 897 rôznych nemeckých diel, ku ktorým sú dostupné snímky strán a ich prepisy. Vytvorená dátová sada sa následne používa pri trénovaní a experimentovaní s neurónovou sieťou. V rámci experimentov sú skúmané rôzne modely konvolučných sietí, vplyv hyperparametrov siete a účinok pozičného kódovania na výsledky rozpoznávania. Výsledný model dokáže rozpoznať znaky s presnosťou 99, 63 %. Prínosom tejto práce je spomínaná dátová sada a neurónová sieť, ktorá sa môže používať pri rozpoznávaní historických dokumentov.

Abstract

This thesis deals with text line recognition of historical documents. Historical texts dating back to the 17th – 19th centuries are written in fraktur typeface. The character recognition problem is solved using neural network architecture called *sequence-to-sequence*. This architecture is based on encoder-decoder model and contains *attention* mechanism. In this thesis a dataset, from texts originated from German archiv called *Deutsches Textarchiv*, was created. This archive contains 3 897 different German books that have available transcripts and corresponding images of pages. The created dataset was used to train and experiment with the proposed neural network. During the experiments, several convolutional models, hyperparameters and the effects of positional embedding were investigated. The final tool can recognize characters with accuracy 99, 63 %. The contribution of this work is the mentioned dataset and neural network, which can be used to recognize historical documents.

Kľúčové slová

rozpoznávanie textu, historický text, neurónová sieť, OCR, konvolučná neurónová sieť, CNN, rekurentná neurónová sieť, RNN, seq2seq, kodér, dekodér, attention

Keywords

text recognition, historical text, neural network, OCR, convolutional neural network, CNN, recurrent neural network, RNN, seq2seq, encoder, decoder, attention

Citácia

VEŠELÍNY, Peter. *Rozpoznávání historických textů pomocí hlubokých neuronových sítí*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Kišš

Rozpoznávání historických textů pomocí hlubokých neuronových sítí

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Martina Kišša. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Vešelíny
15. mája 2019

Poďakovanie

Týmto by som chcel poďakovať svojmu vedúcemu Ing. Martinovi Kiššovi za odborné vedenie, trpezlivosť a cenné rady, ktoré mi pomohli počas tvorby tejto práce. Ďalej by som chcel poďakovať pánovi Ing. Michalovi Hradišovi, Phd. a riešiteľom projektu PERO za rady a pomoc poskytnuté pri vytváraní práce. V poslednej rade by som chcel poďakovať národnej gridovej infraštruktúre MetaCentrum, za poskytnutie prístupu k úložiskám a výpočtovým prostriedkom. Tato práce vznikla za podpory projektov CERIT Scientific Cloud (LM2015085) a CESNET (LM2015042) financovaných z programu MŠMT Projekty veľkých infraštruktúr pre VaVaI.

Obsah

Kapitola 1

Úvod

Rozpoznávanie textu (*Optical Character Recognition*, OCR) [34] je proces, ktorý umožňuje rozpoznávať textové znaky v obraze. Takto rozpoznané znaky môžu byť následne uložené v textovom formáte. Výhodami takéhoto formátu sú zjednodušené upravovanie a vyhľadávanie v texte. OCR nachádza uplatnenie v mnohých oblastiach. V praxi sa môže OCR využiť pri digitalizácií dokumentov, analýze dát (*Data Mining*) alebo pri identifikácií, napr. identifikácia vozidiel pomocou štátnej poznávacej značky (ŠPZ).

Oblasť OCR je vzhľadom na rôznorodosť textu často problematická. Vysoký vplyv na OCR má kvalita obrazovej formy, množstvo šumu, veľkosti jednotlivých znakov, rôznorodosť štýlov písma a podobne. Kvôli tejto problematike je oblasť OCR v posledných dekádach stále študovaným oborom.

V súčasnej dobe sa do popredia dostávajú neurónové siete. Ich prednosťou je ich schopnosť prekonať tradičné algoritmy pri úlohách klasifikácie obrazu [25]. Na druhej strane však ide o systémy, ktoré potrebujú veľké množstvo výpočtovej sily, mnohokrát v podobe grafického procesora (*Graphics Processing Unit*, GPU) a vyžadujú veľké množstvo trénovacíh dát v podobe rozsiahlych dátových sád.

Táto práca sa zaoberá rozpoznávaním gotického písma pomocou neurónových sietí. V rámci práce je vytvorená dátová sada vychádzajúca z nemeckého textového archívu *Deutsches Textarchiv* (DTA) [8], ktorý obsahuje dokumenty z obdobia od 17. až do 19. storočia. Táto dátová sada sa následne využíva pri trénovaní a zisťovaní presnosti implementovanej neurónovej siete. Neurónová sieť vychádza z architektúry *sequence-to-sequence* (seq2seq), ktorá používa mechanizmus zvaný *attention* [21, 1]. Táto sieť načíta obrázok obsahujúci riadok historického textu a následným spracovaním z neho získa jeho prepis.

Práca je organizovaná nasledovne: Kapitola 2 obsahuje prehľad o historických písmach používaných v 12. až 20. storočí. Ďalej sa v kapitole vyskytuje prehľad o súčasných prístupoch rozpoznávania textu, ktoré vychádzajú z použitia rôznych architektúr neurónových sietí. Kapitola 3 obsahuje popis neurónových sietí a metódy používané pri rozpoznávaní sekvencií. Kapitola 4 popisuje dostupné historické dátové sady, a ďalej obsahuje informácie o textovom archíve DTA. Kapitola 5 popisuje návrh architektúry zvolenej v tejto práci. V kapitole 6 sa popisuje implementácia zvolenej architektúry spolu s informáciami o vytváraní dátovej sady. V kapitole 7 sú popísané výsledky z experimentov uskutočnených s implementovanou neurónovou sieťou. V kapitole 8 sa zhodnocuje dosiahnutý stav a možnosti ďalšieho pokračovania v nadväzujúcej práci.

Kapitola 2

Rozpoznávanie textu

Úlohou OCR je získať prepis textovej časti z obrazu. V rámci digitalizácie textu predchádza procesu OCR tzv. segmentácia textu (*text segmentation*) [48].

Úlohou segmentácie textu je identifikovať textové regióny v obraze. Za textové regióny sa považujú obrázky, odstavce, poznámky pod čiarou a predovšetkým riadky textu. Niekedy sa proces identifikácie textových riadkov označuje aj ako *baseline detection* [9, 17]. Čiara *baseline* predstavuje základnú dotyčnicu, na ktorej leží väčšina písmen (výnimkou sú písmená „y“, „j“, „p“ a pod.).

Proces segmentácie textu by mal byť schopný rozlíšiť pozadie, text a prípadné obrázky. Chybná segmentácia textu (napr. chýbajúce písmená vo vete) môže priamo ovplyvniť výsledky OCR. Práve segmentácia historických dokumentov predstavuje problematickú úlohu, pretože tieto dokumenty trpia degradáciou, majú nízkou kvalitu, obsahujú rôzne štýly písma a často zahŕňajú ozdoby a dekorácie.

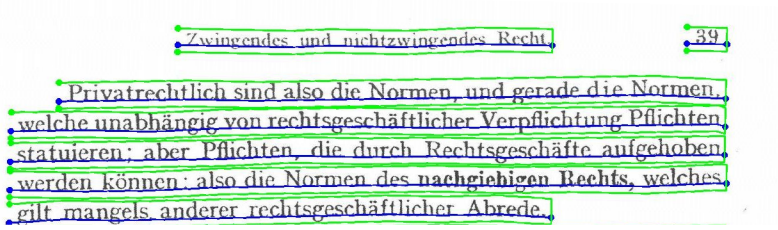
Výstupom segmentácie textu by mali byť body, na základe ktorých sa dá určiť tzv. *bounding box*. Bounding box reprezentuje najmenšie obdĺžnikové ohraničenie okolo komponenty. V prípade detekcie riadkov to je ohraničenie riadka textu. Pomocou takýchto boxov je následne možné vyrezať jednotlivé riadky z obrazu a posunúť ich na vstup procesu OCR. Ukážka segmentácie riadkov textu je na obrázku 2.1.

V prvej časti tejto kapitoly sa bližšie popisuje historické písmo a jeho vývoj. Ďalej sa popisujú súčasné prístupy pre rozpoznávanie textu. Všetky tieto prístupy vychádzajú z použitia neurónových sietí. V poslednej časti je popísaný nástroj *Transkribus*, ktorý slúži pre segmentáciu a rozpoznávanie historických textov.

2.1 Historické písmo

Písmo vznikalo zložitým a pomalým vývojom. Za predchodcov písma sa považujú kresby, symboly a rôzne piktogramy. V rámci vývoja písma sa vyvinulo niekoľko rozličných typografických štýlov, ktoré sú charakteristické svojím tvarom, obdobím a miestom používania a svojimi zástancami.

V súčasnosti je najpoužívanejším písmom *latinka* [51]. Pôvodne sa toto písmo vyvinulo pre latinský jazyk odvodením z gréckej abecedy. V dôsledku sťahovania národov sa v období od 4. až do 7. storočia začali rozvíjať národné písma. Následne bola v 8. storočí snaha zápisy týchto písiem zjednotiť a vytvorilo sa písmo zvané *karolínska minuskula*. V 12. storočí sa karolínska minuskula zmenila a vzniklo písmo nazývané *gotické písmo*. V 15. storočí sa z dôvodu zvýšenia potreby častejšieho a rýchlejšieho písania vytvorilo, z rannej stredovekej



Obr. 2.1: Ukážka segmentácie riadkov v obraze. Modrou farbou sú zobrazené dotyčnice *baseline*. Zelenou farbou sú zobrazené ohraničenia riadkov. Pôvodný obrázok pochádza z DTA [8]. Názov diela je *Die Organisation der Rechtsgemeinschaft*.

karolínskej minuskule, písmo zvané *humanistické písmo*. V tomto období dochádza k zmenám v gotickom písme a vzniká *novogotické písmo*. Zatiaľčo sa humanistické písmo stalo výhradným písmom pre zápis textov v latinčine, gotické písmo sa používalo pri zápisoch v národných jazykoch. V 19. storočí dochádza k postupnému vytlačeniu gotického písma a v roku 1941 dochádza k jeho zákazu aj v Nemecku, kde sa používalo najdlhšie [50].

Gotické písmo

Gotické písmo sa začalo používať nástupom 12. storočia. Tento štýl písma pomenovala talianska skupina humanistov v 15. storočí. Názov písma má symbolizovať zastaranosť, pričom slovo „gotický“ (*gothic*) je synonymom slova „barbarský“. Táto symbolika úzko súvisí s vtedajším stredovekým spôsobom života. Šírenie gotického písma trvalo hlavne od 12. až do začiatku 16. storočia. V nemeckom jazyku však tento štýl pretrval až do 20. storočia [30].

Typickým znakom gotického písma je výsledný vzhľad stránky, ktorá vyzerá príčinou malých medzier medzi jednotlivými písmenami, ako tmavá. Z tohoto dôvodu sa toto písmo označuje ako „tmavé písmo“ (*blackletter*). Dôsledkom malých medzier bola nižšia konzumácia stránok pri písaní. Toto viedlo k tomu, že sa písmo začalo používať pri produkcií kníh a tlači [49]. Spoločnou charakteristikou skupiny gotických písiem sú ostré hrany, lomené oblúky a vysoký vertikálny tvar písmen. Pre knihy bola typická knižná maľba a dekoratívne prvky. V čase gotiky sa taktiež začína vytvárať dvojitá abeceda pre rozlišovanie veľkých a malých písmen [30].

Gotické písmo predstavuje rozsiahlu skupinu písiem, ktoré sa postupom času pretvárali. Na obrázku 2.2 sú zobrazené ukážky gotických kapitálok[1]. Najrozšírenejšími štýlmi gotického písma boli nemecké štýly *švabach* a *fraktúra* [30].

Fraktúra Pomenovanie fraktúra [30] pochádza z latinského slova *fractus*, čo znamená „prerušovaný“. Toto označenie vzniklo na základe hranatých a prerušovaných čiar tohoto písma. Fraktúra bola v Nemecku široko rozšírená. Z tohoto dôvodu sa týmto termínom niekedy označuje celá skupina nemeckých gotických písiem. Okrem pôvodných znakov latinskej abecedy obsahuje fraktúra špecifický znak „ß“ (ostré „s“) a „ſ“ (dlhé „s“). Písmo sa označuje ako „kreslené“, pretože každé písmeno bolo zostavované niekoľkými samostatnými ťahmi pera. Písmená stoja v slovách samostatne bez použitia spojovacích oblúčikov. Porovnanie fraktúry s ostatnými významnými gotickými štýlmi je na obrázku 2.3.

[1] veľké písmená abecedy

𝔄 𝔅 ℭ 𝔇 ℰ 𝔉 𝔊 ℌ ℑ 𝔍 𝔎 𝔏 𝔐 𝔑 𝔒 𝔓 𝔔 ℜ 𝔖 𝔗 𝔘 𝔙 𝔚 𝔛 𝔜 ℨ

(a) Gotické kapitálky zo 14. storočia.

𝕬 𝕭 𝕮 𝕯 𝕰 𝕱 𝕲 𝕳 𝕴 𝕵 𝕶 𝕷 𝕸 𝕹 𝕺 𝕻 𝕼 𝕽 𝕾 𝕿 𝖀 𝖁 𝖂 𝖃 𝖄 𝖅

(b) Gotické kapitálky z 15. storočia.

𝕬 𝕭 𝕮 𝕯 𝕰 𝕱 𝕲 𝕳 𝕴 𝕵 𝕶 𝕷 𝕸 𝕹 𝕺 𝕻 𝕼 𝕽 𝕾 𝕿 𝖀 𝖁 𝖂 𝖃 𝖄 𝖅

(c) Kapitálky použité v *Gutenbergovej Biblii* z 15. storočia.

𝔄 𝔅 ℭ 𝔇 𝔈 𝔉 𝔊 ℌ ℑ 𝔍 𝔎 𝔏 𝔐 𝔑 𝔒 𝔓 𝔔 ℜ 𝔖 𝔗 𝔘 𝔙 𝔚 𝔛 𝔜 ℨ

(d) Písmo švabach. Kapitálky z obdobia 15. až 16. storočia.

𝔄 𝔅 ℭ 𝔇 𝔈 𝔉 𝔊 ℌ ℑ 𝔍 𝔎 𝔏 𝔐 𝔑 𝔒 𝔓 𝔔 ℜ 𝔖 𝔗 𝔘 𝔙 𝔚 𝔛 𝔜 ℨ

(e) Dizajn Albrechta Dürera. Ovplyvnené renesanciou. Pôvod zo 16. storočia.

𝕬 𝕭 𝕮 𝕯 𝕰 𝕱 𝕲 𝕳 𝕴 𝕵 𝕶 𝕷 𝕸 𝕹 𝕺 𝕻 𝕼 𝕽 𝕾 𝕿 𝖀 𝖁 𝖂 𝖃 𝖄 𝖅

(f) Písmo fraktúra z diela *Gebenbuch* cisára Maximiliána I. zo 16. storočia.

Obr. 2.2: Ukážka gotických kapitálok. Prebrané z [30].

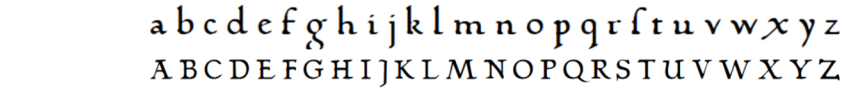
Obr. 2.3: Ukážka odlišností znakov z vybraných štýlov gotického písma. Vľavo je zobrazené súčasné písmo latinka. Za ňou nasledujú gotické štýly textúra, rotunda, švabach a fraktúra. Prebrané z [49].

Humanistické písmo

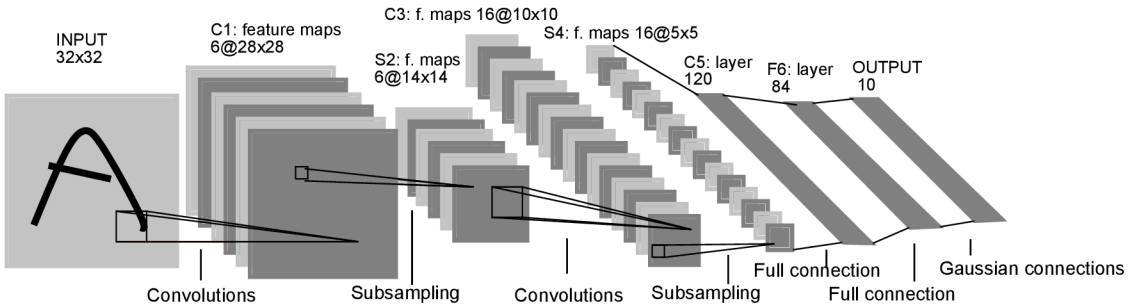
Okrem gotického písma bolo v 15. storočí rozšírené písmo zvané humanistické písmo. Toto písmo bolo výnimkou Nemecka rozšírené vo zvyšných častiach Európy. Písmo sa používalo k zápisu v latinskom jazyku. Šírenie humanistického písma začalo v Taliansku. Ako napovedá názov, písmo vytvorili pre svoju potrebu tamojší humanisti. Humanistické písmo je označované ako „staré“ (*antiqua*) a „okrúhle“ (*rotunda*[2]). Obdobie gotiky je v umení špecifické svojou ignoranciou a barbarstvom. S týmto nesúhlasili taliansky humanisti, ktorí sa rozhodli o zmenu. Humanisti vytvorili zo starých antických skrípt nový štýl písma. Tento štýl písma pomenovali antikva (*antiqua*), pretože si mysleli, že ide o staré texty z obdobia Rímskej ríše. V skutočnosti išlo o texty pochádzajúce z ranného stredoveku, konkrétne o texty, v ktorých bolo použité písmo karolínska minuskula [30].

Humanistické písmo sa začalo vyvíjať v neskoršom období ako písmo gotické, ale pôvod oboch písiem je rovnaký. Humanistické písmo sa vyznačuje svojou jednoduchosťou a vyššou

[2]rotunda označuje aj typ gotického písma charakteristický nižšou mierou lámania oblúkov



Obr. 2.4: Ukážka malých a veľkých písmen humanistického písma antikva. Písmeno „ſ“ sa začalo neskôr z humanistického písma vytrácať. Prevzaté z [30].



Obr. 2.5: Architektúra siete LeNet-5 pre rozpoznávanie znakov. Architektúru tvoria konvolučné vrstvy (Convolutions), pooling vrstvy (Subsampling) a plne prepojené vrstvy (Full connection). Každá úroveň predstavuje mapu príznakov. Posledná vrstva v obrázku (Gaussian connections) predstavuje funkciu pre klasifikáciu pomocou RBF. Prevzaté z [26].

mierou čitateľnosti ako písmo gotické. Oproti lomeným tvarom gotického písma využíva viac zaokrúhlených tvarov. Ukážka humanistického písma je na obrázku 2.4.

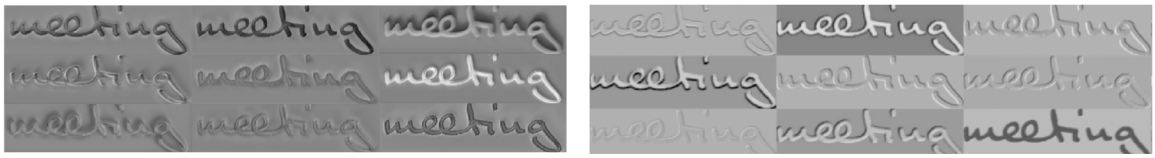
2.2 Existujúce metódy

Konvolučnú neurónovú sieť pre rozpoznávanie písmen použili v roku 1998 *Yann LeCun a spol.* [26]. V tomto prístupe predstavili neurónovú sieť *LeNet-5*. Táto sieť dokázala rozpoznávať ručne písané znaky z dátovej sady MNIST[3] (*Modified* NIST[4]). Sieť v tej dobe prekonala všetky ostatné techniky používané pri rozpoznávaní znakov. Architektúra tejto siete pozostáva z troch konvolučných vrstiev, pričom prvé dve sú nasledované tzv. *pooling* vrstvami. Poslednú časť siete tvoria plne prepojené vrstvy, ktoré klasifikujú obrázok pomocou radiálnej bázickej funkcie (*Radial Basis Function*, RBF). Architektúru siete je možné vidieť na obrázku 2.5.

Nevýhodou konvolučných sietí je ich neschopnosť rozpoznať rozsiahlejšie sekvencie. Jednotlivé časti sekvencie sú na sebe závislé a môžu dosahovať rôzne dĺžky. Pre tieto účely existujú modely rekurentných sietí. Ich výhodou je, že pri rozpoznávaní sekvencií nepotrebujú poznať polohu zložiek sekvencie [40].

Rekurentné siete použili v roku 2009 *Alex Graves a spol.* [16]. Ponúkajú alternatívny postup ako trénovať rekurentné siete pri rozpoznávaní sekvencií. Ich sieť používa objektívnu funkciu *Connectionist temporal classification* (CTC) [15], ktorá sa prvýkrát použila pri rozpoznávaní hlasu [16]. Najlepší výsledok dosiahli, keď spolu s CTC použili obojsmernú

[3] http://yann.lecun.com/exdb/mnist/
[4] https://www.nist.gov/srd/nist-special-database-19



(a) 2D-LSTM. (b) Konvolučná sieť.

Obr. 2.6: Ukážky extrahovaných príznakov pomocou 2D-LSTM (obr. a) a pomocou konvolučnej siete (obr. b). Vizuálna podobnosť naznačuje, že pre extrakciu zmysluplných príznakov nie je potrebná rekurentná sieť s vysokým obsahom kontextu. Pre tieto účely je postačujúca konvolučná sieť. Prevzaté z [37].

rekurentnú sieť spolu s *Long Short Term Memory* (LSTM). Podľa nich dáva CTC najlepšie výsledky pri sieťach, ktoré pracujú s vysokým obsahom kontextu. Práve LSTM umožňuje pracovať s kontextom na dlhé vzdialenosti. Výsledky ich práce dokázali prekonať modely založené na skrytých Markovových modeloch (*Hidden Markov model*), ktoré v tej dobe predstavovali najbežnejší prístup pre rozpoznávanie reči a písma.

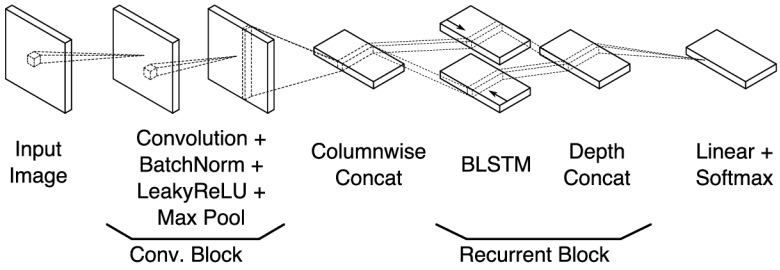
Podobnú sieť použili *Shi a spol.* vo svojej práci [40] v roku 2015. Tento model nesie pomenovanie ako „konvolučno rekurentnú neurónovú sieť“ (*Convolutional Recurrent Neural Network*). Architektúra tejto siete pozostáva z konvolučnej vrstvy, rekurentnej vrstvy a tzv. vrstvy prepisu (*transcription layer*). Konvolučná vrstva extrahuje vizuálne príznaky z obrázka. Rekurentná vrstva následne produkuje predikcie pre každý rámec z výstupu konvolučnej vrstvy. Posledná vrstva slúži na prepis jednotlivých výstupov rekurentnej vrstvy a vytvára tak finálny prepis pre celé slovo. Architektúra tejto siete vystupuje v dvoch verziách. Prvú verziu predstavuje sieť, ktorá vytvára prepis s použitím slovníka (*lexicon-based*). Druhá verzia predstavuje prípad prepisu bez použitia slovníka (*lexicon-free*).

Výhody obojsmerných 1D-LSTM sietí oproti tzv. „viacrozmerným“ (*Multidimensional*) LSTM sieťam popisuje *Puigcerver* vo svojej práci [37] z roku 2017. Podľa neho predstavujú viacrozmerné LSTM siete najmodernejší prístup (*state-of-the-art*) pre rozpoznávanie ručne písaného textu. Oproti 1D-LSTM sieťam sú však výpočtovo náročnejšie a neponúkajú toľko prostriedkov k paralelnému behu na GPU. Navyše sú výsledky oboch modelov ekvivalentné. Po dôkladnej analýze prišiel *Puigcerver* k záveru, že príznaky extrahované pomocou 2D-LSTM sietí z nižších vrstiev sú vizuálne podobné s príznakmi extrahovanými pomocou konvolučných vrstiev. Ukážka je na obrázku 2.6. Použitím syntetického rozšírenia dátovej sady pomocou techniky *data augmentation* dosiahli 1D-LSTM siete dokonca lepšie výsledky ako viacrozmerné LSTM siete. Ukážka architektúry siete je na obrázku 2.7. Ide o podobnú architektúru ako použili *Alex Graves a spol.* [16] a *Shi a spol.* [40].

Pri rozpoznávaní písma obsahujú vstupné obrázky často rozličné geometrické deformácie. To môže byť spôsobené štýlom písma alebo technikou autora. V dôsledku tohoto problému vznikol modul zvaný *Spatial Transformer* [20]. Tento modul umožňuje sieti vykonávať počas učenia priestorové transformácie. Výsledkom je, že je sieť viac invariantná voči rotáciám, zmenám mierky a posunom ako sieť bez tohoto modulu.

Pri spracovaní sekvencií sa často vyskytujú architektúry zvané *sequence-to-sequence* (seq2seq) [44]. Ide o prístup, ktorý umožňuje mapovať vstupnú sekvenciu na výstupnú sekvenciu pomocou architektúry *encoder-decoder*. Architektúra seq2seq pozostáva zvyčajne z dvoch rekurentných sietí nazývaných kodér (*encoder*) a dekodér (*decoder*).

Ilya Sutskever a spol. [44] použili takúto sieť na preklad medzi anglickým a francúzskym jazykom. V rámci experimentovania s modelom zistili, že sa LSTM učí oveľa lepšie v prí-



Obr. 2.7: Architektúra siete pre rozpoznávanie slov. Architektúra pozostáva z troch častí: z konvolučných vrstiev (Conv. Block), z obojsmerných rekurentných vrstiev s LSTM (Reccurent Block) a z lineárnej vrstvy (Linear). Sieť sa učí pomocou objektívnej funkcie CTC. Prevzaté z [37].

pade obrátenia zdrojovej sekvencie v anglickom jazyku. Spočiatku sa domnievali, že toto obrátenie spôsobí dôveryhodnejšiu predikciu na začiatku sekvencie a menej dôvernú predikciu na konci sekvencie. Nakoniec sa ale ukázalo, že tento postup priniesol lepšie výsledky hlavne pri dlhých sekvenciách. Podľa *Ilya Sutskever a spol.* to spôsobilo zavedenie mnohých krátkodobých závislostí v dátovej sade.

Ashish Vaswani a spol. v práci *Attention Is All You Need* [46] zaviedli odlišnú seq2seq sieť zvanú *Transformer*. Ich model na rozdiel od ostatných seq2seq modelov neobsahuje žiadne rekurentné vrstvy, ale je založený na mechanizme *attention*. Tento modul umožňuje hľadať závislosti medzi vstupnou a výstupnou sekvenciou. Týmto prístupom dokázali model oveľa ľahšie paralelizovať na viacerých GPU a tým umožnili rýchlejšie učenie siete oproti ostatným modelom. Po natrénovaní siete dokázali prekonať všetky ostatné metódy na strojový preklad medzi anglickým a nemeckým jazykom.

Pre rozpoznávanie slov použili seq2seq *Lei Kang a spol.* [21]. Použitím tohoto modelu dosiahli na dátovej sade IAM[5] rovnako dobré výsledky ako ostatné neurónové siete.

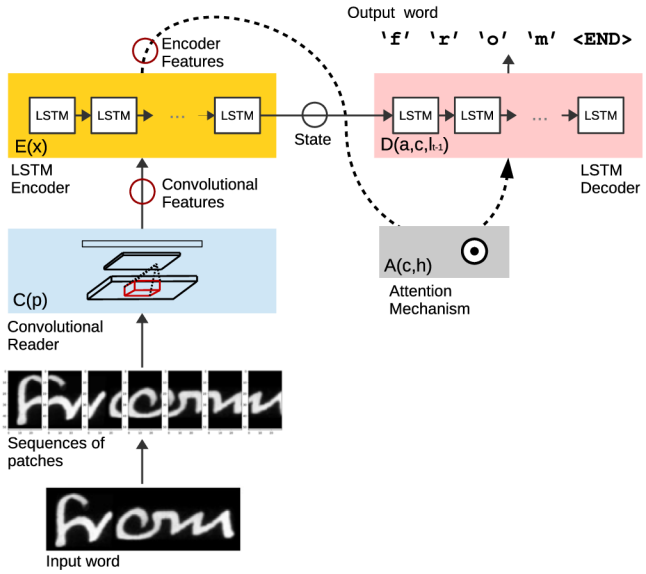
Jorge Sueiras a spol. [43] vytvorili architektúru siete pozostávajúcu z troch častí: z konvolučného snímača (*convolutional reader*), z kodéra a z dekodéra obsahujúceho mechanizmus *attention*. V práci kombinujú použitie horizontálneho posuvného okna (*sliding window*) a konvolučnej siete vychádzajúcej z architektúry LeNet-5. Takto extrahované príznaky sa kódujú a následne dekódujú pre získanie odpovedajúcej sekvencie textu. Ukážka siete je na obrázku 2.8.

2.3 Existujúce nástroje

Medzi známe komerčné nástroje OCR patrí predovšetkým *ABBYY FineReader*[6]. ABBYY FineReader sa používa hlavne pri práci s tlačenými a *pdf* dokumentmi. ABBYY FineReader je považovaný za najlepší nástroj v tejto oblasti. Keďže však ide o komerčný nástroj, jeho použitie vyžaduje zakúpenie licencie.

Medzi voľne dostupné riešenia patria nástroje *OCRopus*[7] a *Transkribus*[8]. OCRopus predstavuje program na analýzu dokumentov. Jeho výhodami sú hlavne jednoduchá roz-

[5] http://www.fki.inf.unibe.ch/databases/iam-handwriting-database
[6] https://www.abbyy.com/cs-cz/finereader/
[7] https://github.com/tmbdev/ocropy
[8] https://github.com/transkribus/



Obr. 2.8: Model architektúry seq2seq pre rozpoznávanie slov. Model transformuje obrázok obsahujúci slovo do niekoľkých menších snímok (*patches*). Tieto snímky spracúva konvolučný snímač (*convolutional reader*) a produkuje sekvenciu vizuálnych príznakov znakov alebo ich častí. LSTM kodér (*encoder*) spracúva túto sekvenciu a hľadá relevantné vzťahy medzi jednotlivými znakmi. Mechanizmus *attention* pomáha klásť dôraz na významné príznaky sekvencie kodéra. Dekodér (*decoder*) následne prijíma túto sekvenciu a znak po znaku vytvára výsledné slovo. Prevzaté z [43].

šíriteľnosť a možnosť voľného upravovania podľa vlastných potrieb. Pri práci s týmto nástrojom je často potrebné natrénovať vlastný model, a práve preto sú spolu s nástrojom dostupné skripty pre trénovanie modelu, počítanie chybovosti a korekciu *ground truth*.

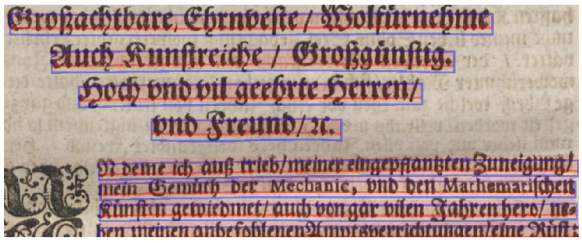
Transkribus predstavuje oproti nástrojom *ABBYY FineReader* a *OCRopus* špecifický nástroj slúžiaci hlavne pre rozpoznávanie historických textov. Hoci sa pre tieto účely dajú použiť aj predošlé nástroje, Transkribus bude predstavovať pravdepodobne najvhodnejšieho kandidáta.

Transkribus

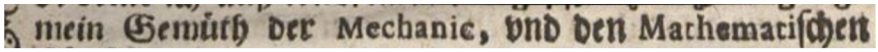
Transkribus[9] predstavuje nástroj pre rozpoznávanie ručne písaného textu (*Handwritten Text Recognition*) z historických dokumentov. Transkribus patrí pod projekt READ[10] (*Recognition and Enrichment of Archival Documents*), ktorého cieľom je vytvoriť prostredie pre automatické rozpoznávanie a prepis dokumentov.

Transkribus ponúka možnosti pre segmentáciu textu a následné rozpoznávanie detekovaných častí. Ukážka segmentácie textových regiónov je na obrázku 2.9. Po ukončení procesu segmentácie umožňuje Transkribus použiť systém OCR pre rozpoznanie detekovaných riadkov. Pre tieto účely je dostupný *ABBYY Finereader Engine 11*[11] alebo modely z CITlab[12].

[9] https://transkribus.eu/Transkribus/
[10] https://read.transkribus.eu/
[11] https://www.abbyy.com/en-gb/support/engine/11win/product_info/lang/
[12] https://github.com/CITlabRostock



Obr. 2.9: Segmentácia textových regiónov pomocou nástroja Transkribus. Modrou farbou je ohraničený riadok textu. Červenou farbou sú vyznačené dotyčnice *baseline*. Dotyčnice sú detekované pomerne s vysokou presnosťou. Problém nastáva pri vytváraní ohraničení riadkov. Zle segmentované riadky sú predovšetkým riadky v nadpisoch, kde sa vyskytujú písmená s väčšou veľkosťou. Riadky v paragrafoch sú detekované presnejšie, ale taktiež v nich dochádza k orezaniu horných častí písmen. Pôvodný obrázok s historickým textom pochádza z [8]



GROUND TRUTH:
mein Gemůth der Mechanic, vnd den Mathematiſchen

ABBYY Finereader:
mein Gemükh der Mccbanie, vnd den Mathemacifc^ftt

CITlab:
mein Gemuch der iecbaaie. bnd den blatlematischet

Obr. 2.10: Rozpoznávanie textu pomocou nástroja Transkribus. V hornej časti je obrázok s rozpoznávaným textom a pod ním je jeho prepis (GROUND TRUTH). Následne sú uvedené výstupy OCR *ABBYY Finereader* (24, 5%) a OCR z *CITlab* (26, 5%). V zátvorkách je uvedená percentuálna veľkosť chyby CER (*Character error rate*) voči správnemu prepisu. V prípade *ABBYY Finereader* bola použitá konfigurácia: typ písma „Gothic“ a typ jazyka „German“. V prípade CITlab bol použitý model *Comb_Gothic_Script*. Pôvodný obrázok s historickým textom pochádza z [8].

V rámci modelov sú dostupné informácie o úspešnostiach modelov a jazyky, ktoré dokáže daný model rozpoznávať. Podľa týchto informácií sa môže pre rozpoznávanie zvoliť vhodnejší model. Ukážka rozpoznávania riadkov nástrojom Transkribus je na obrázku 2.10.

Transkribus uchováva svoje prepisy spolu s detekovanými regiónmi a riadkami v súbore vo formáte *PAGE xml* [35]. Súčasťou Transkribusu je aj nástroj na zobrazenie segmentovaných dokumentov pomocou načítaných PAGE súborov. Okrem toho umožňuje Transkribus pripraviť vlastnú dátovú sadu, pretože umožňuje vlastné vyznačovanie textových regiónov a vlastné pridávanie a priraďovanie prepisov. Nevýhodou Transkribusu je, že pracuje na báze klient-server (*client–server*). Všetky úkony spojené s rozpoznávaním dokumentov môžu byť uskutočnené až po nahratí dát na server.

Kapitola 3

Neurónové siete

Neurónové siete [25] boli inšpirované chovaním biologických neurónov v nervovej sústave človeka. Neurónová sieť pozostáva z prvkov nazývaných neuróny, ktoré sú v nej medzi sebou prepojené. Každý z týchto neurónov dokáže na vstupe prijať hodnotu, zmeniť svoj vnútorný stav a následne vyprodukovať výstupnú hodnotu. Matematicky sa môže neurónová sieť vyjadriť ako funkcia f, ktorá mapuje vstup X na výstup Y:

$$f : X \to Y \,. \tag{3.1}$$

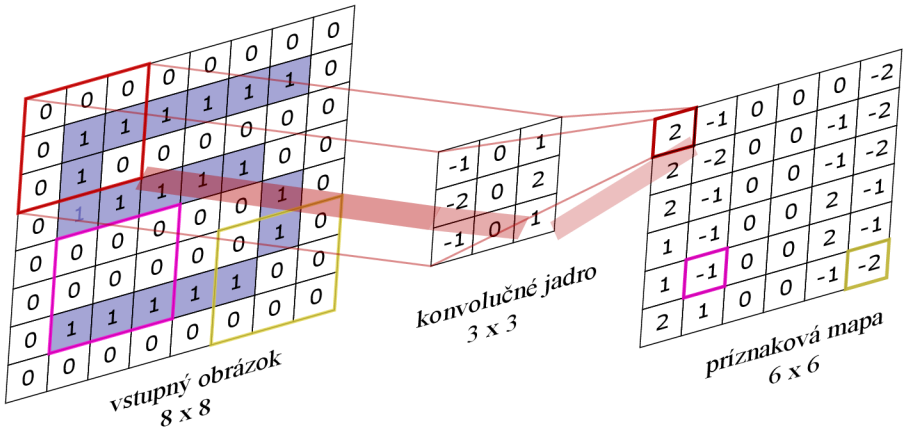
Neurónové siete patria medzi úlohy „učenia s učiteľom“ (*supervised learning*). Pri tomto spôsobe učenia musí mať neurónová sieť prístupné trénovacie dáta v podobe dátovej sady. Súčasťou dátovej sady pre rozpoznávanie písma sú obrázky s textom a ich odpovedajúce prepisy (*ground truth*).

V rámci tejto kapitoly sa popisuje učenie neurónových sietí a následne sa bližšie popisujú konvolučné a rekurentné neurónové siete. Pri rekurentných neurónových sieťach sa podrobnejšie popisujú jednotlivé techniky, ktoré sa v rámci týchto metód používajú. Ide o použitie *Long Short-Term Memory* (LSTM) [18], *Gated Recurrent Unit* (GRU) [5] a *Connectionist Temporal Classification* (CTC) [15]. V poslednej časti sa popisujú architektúry *Sequence to sequence* [14, 21, 44] a mechanizmus *Attention* [21, 46].

Učenie klasifikátora Učenie siete pre klasifikáciu prebieha v nasledovných krokoch: Sieť dostane na vstup obrázok. Tento obrázok prechádza jednotlivými vrstvami siete a postupne sa pomocou aktivačnej funkcie (*activation function*) aktivujú neuróny siete. Neuróny sú v sieti reprezentované pomocou váh (*weights*) a zložky nazývanej *bias*. Typická neurónová sieť môže obsahovať stovky miliónov váh. Výstup siete je reprezentovaný vektorom, ktorý určuje výslednú triedu klasifikácie. Tento výstup sa počas trénovania porovnáva s *ground truth* a uskutoční sa výpočet objektívnej funkcie (*loss function* alebo *object function*). Výstup tejto funkcie určuje veľkosť chyby medzi výstupom siete a prepisom *ground truth*. Následne dochádza k spätnej propagácii chyby. Podľa veľkosti chyby sa aktualizujú neuróny v sieti. Počas tohoto procesu sa sieť snaží minimalizovať výslednú chybu. Zvyčajne sa pre účel tejto optimalizácie používa algoritmus stochastický gradientný zostup (*Stochastic Gradient Descent*) [39, 26]. Algoritmus popisuje nasledovná rovnica:

$$w_k := w_{k-1} - \eta \nabla Q(w_{k-1}) \,, \tag{3.2}$$

kde w_k predstavuje aktualizovanú váhu, $Q(w_{k-1})$ je veľkosť chyby určenej objektívnou funkciou pre aktualizovateľnú váhu w_{k-1}, ∇ predstavuje hodnotu gradientu a η predstavuje veľkosť kroku (*learning rate*).



Obr. 3.1: Ukážka 2D konvolúcie. Zvýraznené štvorce poukazujú na to, ako sa lokálne časti vo vstupnom obrázku premietnu do príznakovej mapy.

3.1 Konvolučné neurónové siete

Konvolučné neurónové siete [25, 26] sú neurónové siete, ktoré sa používajú pri spracovaní obrazu. Lokálne časti v obraze spolu často súvisia, t.j. sú vysoko korelované, pričom tvoria výrazne detekovateľné motívy. Tieto motívy sú častokrát invariantné voči svojej polohe, čo znamená, že sa môžu objaviť v ktorejkoľvek časti obrazu. Súčasťou konvolučných sietí sú konvolučné (*convolutional*) a tzv. *pooling* vrstvy.

Konvolučné vrstvy slúžia na extrahovanie vizuálnych príznakov lokálnych spojení a vytváranie máp týchto príznakov (*feature maps*). Tieto mapy sa vytvárajú postupným aplikovaním filtra, ktorý nazývame konvolučné jadro (*convolution kernel*). Každá konvolučná vrstva môže obsahovať odlišný filter. V súvislosti s obrazom ide o zväčšovanie tretieho kanálu obrazu. Matematicky predstavuje toto filtrovanie operácia diskrétnej 2D konvolúcie. Táto operácia má nasledujúcu definíciu:

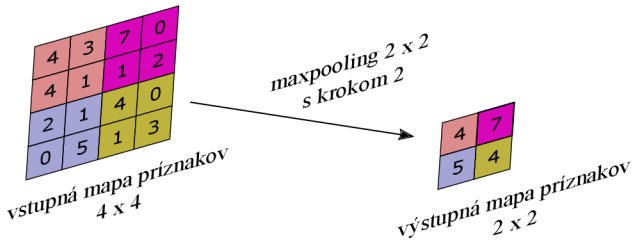
$$(f \star g)(x, y) = \sum_{i,j} f(i, j) \cdot g(x - i, y - j), \tag{3.3}$$

kde $\star$ je operácia konvolúcie, funkcie f a g predstavujú obraz a konvolučné jadro, x a y predstavujú body obrazu, a nakoniec súradnice i a j predstavujú osy konvolučného jadra. Ukážka použitia 2D konvolúcie je na obrázku 3.1.

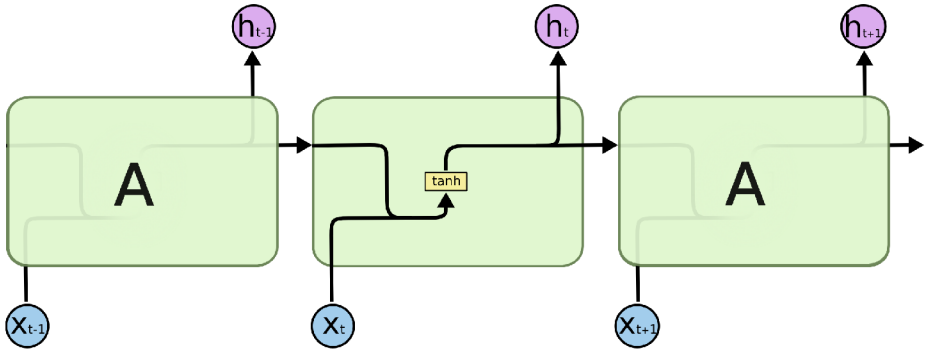
Úlohou pooling vrstvy je spájať významovo podobné príznaky do jedného príznaku. Toto sa typicky uskutočňuje výpočtom na základe maximálnej hodnoty (*max pooling*) alebo priemeru (*average pooling*) [25, 26]. V súvislosti s obrazom ide o zmenšovanie výšky a šírky obrazu. Spájanie konvolučných a pooling vrstiev umožňuje sieti vyhľadávať príznaky na nižších úrovniach. Ukážka spracovania vrstvou max pooling je na obrázku 3.2.

3.2 Rekurentné neurónové siete

Rekurentné neurónové siete (*Recurrent neural network*, RNN) [25] sú neurónové siete, ktoré sa používajú predovšetkým pri spracovávaní sekvencií. RNN obsahujú tzv. rekurentné vrstvy (*recurrent layers*). Tieto vrstvy postupne spracúvajú vstupnú sekvenciu a udržiavajú si jej



Obr. 3.2: Ukážka vrstvy *maxpooling* s filtrom 2×2 a krokom 2. Zo štvorice hodnôt, ktoré predstavujú rozmer 2×2 sa určí maximálna hodnota.

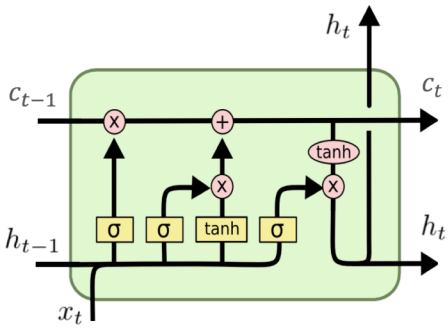


Obr. 3.3: Schéma štruktúry rekurentnej siete. Rekurentná vrstva pozostáva z neurónov A, ktoré sú medzi sebou spojené a tvoria tak reťazec. Jednotlivé neuróny vrstvy majú na vstupe informáciu z ostatných neurónov z predchádzajúceho časového kroku. Symbol x_t predstavuje vstupnú sekvencie v čase t. Symbol h_t predstavuje výstup v čase t. Typicky je vo vnútri každého neurónu jediná aktivačná funkcia, ktorá určuje jeho výstup. V tomto príklade bol použitý hyperbolický tangens (*Hyperbolic tangent*, tanh). Prevzaté z [32].

stav v podobe stavového vektora (*state vector*). Tento vektor umožňuje uchovávať informáciu o predošlých častiach sekvencie. Tento proces je znázornený na obrázku 3.3.

Obojsmerná rekurentná sieť Obojsmerná rekurentná sieť (*Bidirectional recurrent neural network*) predstavuje spojenie dvoch rekurentných sietí. Jedna z týchto sietí spracováva sekvenciu v doprednom smere, zatiaľčo druhá spracováva sekvenciu v spätnom smere. Týmto sa umožňuje pracovať s informáciou, ktorá je z minulosti, a zároveň aj z budúcnosti. Výstupy oboch sietí sú v každom časovom kroku spojené do jedného výstupného vektora.

Zanikajúci a explodujúci gradient Nevýhodou rekurentných sietí je, že si stále nedokážu uchovať všetky potrebné informácie v prípade dlhých sekvencií. Pri učení siete môže nastať, že gradienty v sieti zaniknú (*Vanishing gradient*) alebo explodujú (*Exploding gradient*). Je to spôsobené tým, že v hlbokých neurónových sieťach je gradient nestabilný. Dôsledkom toho sa pri učení extrémne navýši alebo zanikne. V prípade, ak je jeho hodnota extrémne nízka, tak bude učenie siete postupovať príliš pomaly. V prípade, ak gradient exploduje, tak to spôsobí oscilovanie váh v sieti [31]. Spomínané problémy popisuje podrobne vo svojej práci *Yoshua Bengio* [2]. V súčasnosti sa problémy súvisiace s nestálym gradientom v rekurentných sieťach riešia pomocou modulov LSTM [18] a GRU [5].



Obr. 3.4: Ukážka schémy bloku LSTM. Súčasťou bloku sú tri brány. Tieto brány sú zakreslené ako logistická funkcia σ nasledovaná operáciou násobenia $\otimes$. Symbol x_t reprezentuje vstup bloku v čase t, c_t je uchovaný stav v čase t a h_t predstavuje výstup bloku v čase t. Symboly c_{t-1}, h_{t-1} sú uchovaný stav a výstup z predchádzajúceho kroku. Prevzaté z [27].

Long Short-Term Memory

Long Short-Term Memory (LSTM) [18] predstavuje mechanizmus v rekurentných neurónových sieťach, ktorý umožňuje učenie dlhodobých závislostí. Učenie dlhodobých závislostí poskytuje riešenie problémov so zanikajúcim a explodujúcim gradientom. Mechanizmus predstavili *Hochreiter* a *Schmidhuber* v roku 1997. Schéma bloku LSTM je na obrázku 3.4.

Blok LSTM [18, 32] pozostáva z troch častí, ktoré sa nazývajú brány (*gates*). Tieto brány postupne nazývame *forget gate* (f_t), *input gate* (i_t) a *output gate* (o_t). Hlavnou úlohou týchto brán je regulácia toku informácií v jednotlivých blokoch rekurentných sietí. Všetky tri brány pozostávajú z logistickej funkcie σ. Hodnota tejto funkcie je v intervale $f \in (0, 1)$, pričom výstup blížiaci sa logickej „1“ reprezentuje uchovanie informácie, zatiaľčo výstup blížiaci sa logickej „0“ reprezentuje zahodenie informácie. Tento výsledok sa následne násobí s uchovaným stavom z predchádzajúceho časového kroku.

V nasledujúcich rovniciach bude symbol W označovať vnútorné váhy, b bude označovať bias, x_t bude označovať vstup bloku, h_{t-1} bude predstavovať uchovávaný stav z predchádzajúceho časového kroku a *tanh* bude hyperbolický tangens.

Prvú časť LSTM tvorí *forget gate*. Jej úlohou je regulovať tok informácií prichádzajúci z predchádzajúceho časového kroku. Túto charakteristiku popisuje nasledovný vzťah:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)\,. \tag{3.4}$$

Ďalšia časť LSTM určuje, ktoré informácie budú súčasťou uchovávaného stavu bloku. *Input gate* rozhodne, či v niektorých hodnotách dôjde k aktualizovaniu stavu. Následne sa vytvorí vektor nových potencionálnych kandidátov pre uchovanie ($\widetilde{C}_t$). Potrebné vzťahy sú dané rovnicami:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.5}$$

$$\widetilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)\,. \tag{3.6}$$

V ďalšom kroku sa aktualizuje hodnota starého uchovávaného stavu C_{t-1} hodnotou nového stavu C_t, pričom sa použijú hodnoty z predchádzajúcich výstupov brán *forget gate* a *input gate*:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t\,. \tag{3.7}$$

V poslednom kroku rozhodne *output gate* o výstupe bloku h_t podľa nasledujúcich vzťahov:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.8}$$

$$h_t = o_t * tanh(C_t)\,. \tag{3.9}$$

Gated Recurrent Unit

Gated Recurrent Unit (GRU) [4] predstavuje obdobu mechanizmu LSTM. GRU prvýkrát predstavili *Kyunghyun Cho a spol.* v roku 2014. Ide o výpočtovo efektívnejšiu jednotku ako LSTM, pričom dosahuje porovnateľné výsledky. Blok GRU pozostáva z dvoch brán.

V nasledujúcich rovniciach bude σ predstavovať logistickú funkciu, x bude predstavovať vstup bloku, $h_{\langle t-1\rangle}$ bude predstavovať uchovávaný stav z predchádzajúceho časového kroku, $[.]_j$ bude označovať j-tý prvok vektora, W a U budú váhy získané učením siete.

Prvá brána GRU sa nazýva *reset gate* (r_j) a počíta sa podľa nasledovného vzťahu:

$$r_j = \sigma([W_r x]_j + [U_r h_{\langle t-1\rangle}]_j)\,. \tag{3.10}$$

Druhá brána GRU sa nazýva *update gate* (z_j) a počíta sa nasledovne:

$$z_j = \sigma([W_z x]_j + [U_z h_{\langle t-1\rangle}]_j)\,. \tag{3.11}$$

Aktivácia (h_j) daného bloku GRU závisí od:

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1 - z_j)\widetilde{h}_j^{\langle t\rangle}, \tag{3.12}$$

kde z_j predstavuje hodnotu z *update gate* a

$$\widetilde{h}_j^{\langle t\rangle} = \phi([Wx]_j + [U(r_j \odot \langle t-1\rangle)]_j), \tag{3.13}$$

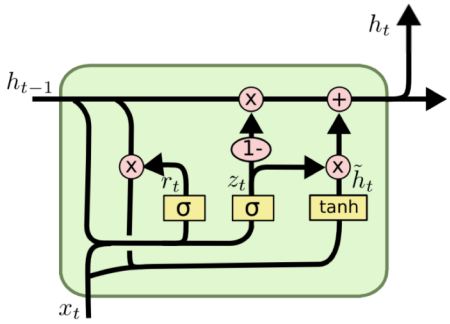
pričom ϕ predstavuje hyperbolický tangens, $\odot$ je Hadamardov súčin[1] a r_j predstavuje hodnotu z *reset gate*.

Z predchádzajúcich vzťahov vyplýva, že keď sa *reset gate* približuje hodnote „0“, tak to umožňuje zahodiť všetky irelevantné informácie. *Update gate* zase umožňuje kontrolovať množstvo informácií predávaných medzi jednotlivými časovými krokmi a pomáhať tak sieti rozpoznávať dlhodobé závislosti. Keďže má každý GRU blok oddelené brány od tých v ostatných blokoch, tak každý z týchto blokov bude zameraný na zachytávanie špecifických informácií. Bloky zamerané na zachytávanie krátkodobých závislostí budú mať často aktívnu *reset gate*. Bloky zamerané na zachytávanie dlhodobých závislostí budú mať často aktívnu *active gate*. Na rozdiel od LSTM neobsahuje GRU žiadnu „input gate“. Ukážka schémy GRU je na obrázku 3.5.

Connectionist temporal classification

Connectionist temporal classification (CTC) [15] označuje objektívnu funkciu, ktorú prvýkrát predstavili v roku 2006 *Alex Graves a spol.* Rekurentné siete predstavujú mocný nástroj pre rozpoznávanie sekvencií. Štandardné objektívne funkcie sú pre rekurentné siete

[1] násobenie dvoch vektorov rovnakého rozmeru po zložkách



Obr. 3.5: Ukážka schémy bloku GRU. Súčasťou bloku sú dve brány zakreslené pomocou logistickej funkcie σ nasledovanej operáciou násobenia $\otimes$. *Reset gate* (r_t) rozhoduje, či sa uchovávaná hodnota stavu odstráni alebo nie. *Update gate* (z_t) rozhoduje, či sa uchovávaná hodnota stavu (h_t) aktualizuje novým stavom ($\tilde{h_t}$). Symboly x_t a h_t predstavujú vstup a výstup bloku. Prevzaté z [27].

definované oddelene pre každý krok učenia, čo znamená, že vytvárajú nezávislé značenia (*labels*) pri klasifikácií. Toto sa v sieťach nevyužívajúcich CTC rieši následným zarovnávaním výstupu. CTC predstavuje riešenie predchádzajúceho problému a umožňuje výpočet chyby pri sekvenčných dátach priamo, bez potreby ďalšieho spracovania.

Rozhodujúcim krokom pri CTC je transformácia výstupu siete do podmieneného rozdelenia pravdepodobností na základe jednotlivých značení v časových krokoch. Sieť sa môže následne použiť ako klasifikátor výberom najpravdepodobnejšej cesty značení pre danú sekvenciu. Formálne sa to môže zapísať ako:

$$p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in {L'}^T, \tag{3.14}$$

kde x predstavuje vstupnú sekvenciu dĺžky T, y_k^t predstavuje pravdepodobnosť značenia k v čase t a ${L'}^T$ predstavuje množinu dostupných značiek, ktorá sa počíta ako $L' = L \cup \{blank\}$, kde L predstavuje množinu všetkých značení v trénovacej sade a *blank* predstavuje tzv. prázdne značenie. Ďalej budeme hovoriť o prvkoch ${L'}^T$ ako o ceste (*path*) a budeme ju označovať π.

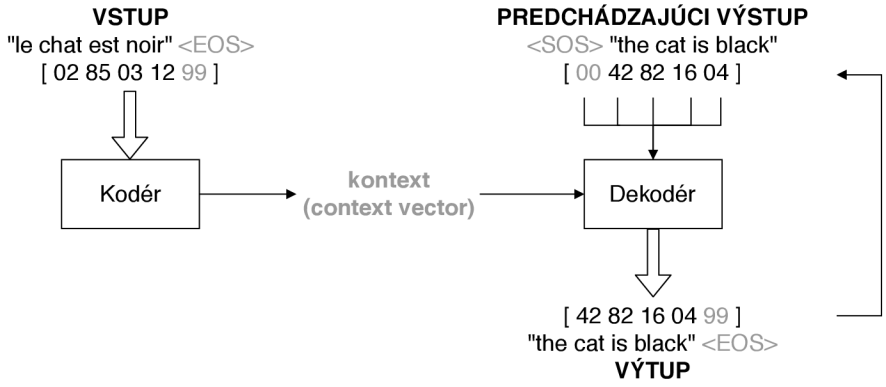
Cesty sú mapované na značenia $l \in L^{\leq T}$ operátorom $\mathcal{B}$, ktorý odstraňuje opakujúce sa značenia a prázdne značenia. Pravdepodobnosť značenia je potom daná sumou pravdepodobností všetkých ciest:

$$p(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} p(\pi|x). \tag{3.15}$$

Z týchto označení sa potom pri klasifikácií vyberie najpravdepodobnejšia trieda, ktorá je daná vzťahom:

$$h(x) = \underset{l \in L^{\leq T}}{\operatorname{argmax}}\, p(l|x). \tag{3.16}$$

Prakticky sa ale tento postup z dôvodu nízkej efektivity nevyužíva a používa sa modifikovaný postup pomocou algoritmu *forward backward* [38].



Obr. 3.6: Ukážka modelu neurónovej siete seq2seq pre preklad medzi francúzskym a anglickým jazykom. Kodér (*encoder*) vytvorí zo vstupnej sekvencie maticu nazývanú *context vector*. Dekodér príjme túto maticu a postupne začne dekódovať jednotlivé časti sekvencie (v tomto prípade celé slová), pričom prijíma informáciu aj o výsledku posledného dekódovania. Dekodér sa inicializuje výstupom kodéra a tokenom <SOS>, ktorý predstavuje začiatok procesu dekódovania. Koniec dekódovania nastane prijatím tokena <EOS>.

Pravdepodobnosť značenia sekvencie ($p(l|x)$) je algoritmom *forward backward* daná sumou súčinu premenných *backward* ($\beta_t(s)$) a *forward* ($\alpha_t(s)$) v každom časovom kroku:

$$p(l|x) = \sum_{s=1}^{|l'|} \alpha_t(s)\beta_t(s)\,. \tag{3.17}$$

Objektívna funkcia CTC (O^{CTC}) je potom záporná pravdepodobnosť prirodzeného logaritmu (ln) správne určených značení v trénovacej dátovej sade (S), ktorá sa počíta ako:

$$O^{CTC} = -\sum_{(x,z)\in S} ln(p(z|x))\,, \tag{3.18}$$

kde (x, z) predstavujú dvojicu vstupnej a cieľovej sekvencie. Tento postup je podrobne vysvetlený v pôvodnej práci [15].

3.3 Sequence-to-sequence

Sequence-to-sequence (seq2seq) [25] predstavuje typ rekurentných neurónových sietí, ktoré sa používajú pri mapovaní vstupnej sekvencie na sekvenciu výstupnú. Architektúru seq2seq tvoria zvyčajne dve časti, ktoré sa nazývajú kodér (*encoder*) a dekodér (*decoder*). Podľa toho sa táto sieť označuje aj ako *encoder-decoder*. Úlohou kodéra je spracovať vstupnú sekvenciu a vytvoriť jej reprezentáciu v podobe matice nazývanej *context vector*. Dekodér sa inicializuje výstupom kodéra a postupne po častiach vytvára pre jednotlivé časti sekvencie výstupnú sekvenciu. Architektúra seq2seq sa využíva zvyčajne pri úlohách spojených so strojovým prekladom (*machine translation*) [46, 4, 44], kedy je sekvencia napísaná v jednom jazyku prekladaná na sekvenciu v druhom jazyku. Ukážka tohoto mapovania je na obrázku 3.6.

Attention

Nevýhodou architektúr seq2seq je, že majú problém pracovať s dlhými sekvenciami. Seq2seq model si pri dlhých sekvenciách nedokáže uchovať všetky potrebné informácie z celej sekvencie vo vektore s pevnou dĺžkou. Tento problém je predovšetkým viditeľný pri sekvenciách dlhších ako boli tie v trénovacej sade. Riešenie tohoto problému priniesol mechanizmus *attention* [1].

Mechanizmus *attention* [1] prvýkrát predstavil *Dzmitry Bahdanau a spol.* v roku 2015. V ich práci navrhujú riešenie v podobe modelu, ktorý dokáže automaticky „nájsť" podstatné časti v zdrojovej sekvencií, ktoré sú dôležité pre predikciu cieľového slova. Zavedené rozšírenie umožňuje spoločné učenie a preklad oboch častí seq2seq modelu. Zakaždým, keď sa dekóduje časť sekvencie, tak sa vyhľadá množina pozícií v zdrojovej vete, ktorá nesie najdôležitejšie informácie spojené s týmto prekladom. Model následne predikuje slovo na základe vektorov spojených s touto pozíciou a na základe predchádzajúcich dekódovaných slov.

Formálne sa mechanizmus *attention* môže zapísať ako:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \,, \tag{3.19}$$

kde c_i reprezentuje *context vector*, ktorý závisí na sekvencií anotácií $(h_1, ..., h_{T_x})$. Každá anotácia h_i obsahuje informáciu o celej sekvencií s vysokým dôrazom na časti obklopujúce i-té slovo vstupnej sekvencie. Hodnota váhy a_{ij} každej anotácie sa vypočíta ako:

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \,, \tag{3.20}$$

kde

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.21}$$

je zarovnanie (*alignment*), ktoré vyhodnotí, ako dobre sa vstup okolo pozície j a výstup pri pozícií i zhodujú. Zarovnanie by malo byť konštruované s ohľadom na to, že bude model vyhodnotený spolu $(T_x \times T_y)$-krát pre každý pár vety s dĺžkou T_x a T_y. Z dôvodu zníženia času výpočtu sa pre výpočet zarovnania používa nasledovný vzťah:

$$a(s_{i-1}, h_i) = v_a tanh(W_a s_{i-1} + U_a h_j) \,, \tag{3.22}$$

kde $W_a \in \mathbb{R}^{nxn}$, $U_a \in \mathbb{R}^{nx2n}$ a $v_a \in \mathbb{R}^n$ sú trénovateľné váhy.

Okrem pôvodného mechanizmu *attention* existujú ďalšie *attention* mechanizmy. Výpočty zarovnávania vybraných *attention* mechanizmov sú v tabuľke 3.1 [47].

Pozičné kódovanie

Matica *context vector* pridáva vstupnej sekvencií prevažne sémantickú informáciu. Okrem sémantického významu je však podstatná aj informácia o polohe jednotlivých častí sekvencie. Pre účel pridania informácií o polohe slúži pozičné kódovanie (*positional encoding*) [46]. Kódovanie polohy môže byť fixné alebo získané učením siete. Rekurentné siete dokážu čiastočne informáciu o polohe zakódovať explicitne. V tomto prípade ide o trénovateľné parametre siete.

Názov	Výpočet	Pôvod
Additive/Concat Attention	$v_a tanh(W_a[s_t; h_i])$	*Bahdanau a spol.* [1]
Location-Base Attention	$softmax(W_a h_i)$	*Luong a spol.* [28]
General Attention	$s_t^T W_a h_i$	*Luong a spol.* [28]
Dot-Product Attention	$s_t h_i$	*Luong a spol.* [28]
Scaled Dot-Product Attention	$\mathfrak{s}_\mathfrak{t}^\mathfrak{T}\mathfrak{h}_\mathfrak{i}\sqrt{n}$	*Vaswani a spol.* [46]

Tabuľka 3.1: Ukážka výpočtu zarovnania v rozdielnych mechanizmoch *attention*. Symbol W_a vo vzorcoch predstavuje trénovateľné váhy, s_t predstavuje uchovaný stav dekodéra (*decoder hidden state*) a h_i je uchovaný stav kodéra (*encoder hidden state*). Objasnenie jednotlivých mechanizmov je v prácach v kolónke „Pôvod“. Prevzaté z [47].

Fixné kódovanie polohy použili vo svojej práci *Vaswani a spol* [46]. V ich práci je pozičné kódovanie dané veľkosťami hodnôt určených funkciami *sin* a *cos* pri rozličných frekvenciách:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \,, \end{aligned} \tag{3.23}$$

kde *pos* predstavuje polohu v sekvencií, i dimenziu vektora a d_{model} celkový počet dimenzií kódovania. Podľa tohoto vzťahu je každá dimenzia zakódovaná pomocou odpovedajúcej sinusoidy. Vlnové dĺžky tvoria geometrickú postupnosť od 2π do $10000 * 2\pi$.

Táto funkcia bola zvolená v dôsledku očakávania, že sa bude sieť učiť relatívne pozície ľahšie, pretože pre každý fixný *offset*[2] k môže byť PE_{pos+k} reprezentovaný ako lineárna funkcia pomocou PE_{pos}.

Seq2seq pre rozpoznávanie písma

Lei Kang a spol. [21] použili model seq2seq pre rozpoznávanie ručne písaných slov. Výhodou ich modelu je, že počas svojho učenia nepotrebuje využívať objektívnu funkciu CTC, nepotrebuje žiaden prístup k jazykovému modelu a nevyžaduje žiadne konkrétne predspracovanie vstupných dát. Ich architektúra pozostáva z kodéra, mechanizmu *attention* a dekodéra.

Vstup kodéra [21] predstavuje obrázok obsahujúci slovo. Kodér tvorí konvolučná neurónová sieť, ktorej úlohou je extrakcia vizuálnych čŕt, ktoré charakterizujú dané slovo. Ďalšou časťou kodéra je obojsmerná rekurentná sieť, ktorá využíva mechanizmus GRU. Táto časť postupne po jednotlivých častiach spracováva výstup konvolučnej siete a propaguje jednotlivé informácie naprieč sekvenciou oboma smermi. Výstupom kodéra je matica *context vector*, ktorá obsahuje vizuálne informácie a informácie o vzájomnej súvislosti jednotlivých znakov.

V rámci práce použili *Lei Kang a spol.* dva rozdielne mechanizmy *attention*, konkrétne *Content-based Attention* a *Location-based Attention* [21].

Princípom *Content-based Attention* je „nájsť“ podobnosti medzi jednotlivými uchovanými stavmi dekodéra a kontextom. Definujme a_t ako maskovací vektor pre mechanizmus *attention* v čase t, pričom h_i predstavuje skrytú vrstvu kodéra v čase $i \in \{0, 1, ..., N-1\}$, s_t predstavuje skrytý stav dekodéra v čase $t \in \{0, 1, ..., T-1\}$, kde T je maximálna dĺžka dekódovaného znaku. Potom platí:

$$a_t = Softmax(e_t), \tag{3.24}$$

[2] indikátor vzdialenosti

kde

$$e_{t,i} = f(h_i, s_{t-1}) = w^T tanh(W h_i + V s_{t-1} + b), \tag{3.25}$$

kde w, W, V a b sú trénovateľné parametre siete. Po aplikovaní mechanizmu *attention* má výsledný vektor pred vstupom do dekodéra tvar:

$$c_t = g(a_t, H) = \sum_{i=0}^{N-1} a_{ti} h_i \tag{3.26}$$

Nevýhodou *Content-based Attention* je fakt, že mechanizmus očakáva, že výstup kodéra bude obsahovať zakódované pozičné informácie. Aby sa predišlo tomu, aby musel kodér takéto informácie generovať, tak sa používa druhá technika zvaná *Location-based Attention.*

Táto metóda upravuje predošlý vzťah a pridáva doň informáciu o zarovnaní z predchádzajúceho kroku. V prvom rade sa extrahuje k vektorov $l_{t,i} \in R^k$ pre každú polohu i predchádzajúceho zarovnania a_{t-1} pomocou konvolúcie s maticou $F \in R^{k \times r}$

$$l_t = F * a_{t-1}, \tag{3.27}$$

a rovnica 3.25 sa následne nahradí vzťahom:

$$e_{t,i} = f'(h_i, s_{t-1}, l_t) = w^T tanh(W h_i + V s_{t-1} + U l_{t,i} + b), \tag{3.28}$$

kde w, W, V, U a b sú trénovateľné parametre siete. Ukážka činnosti tohoto mechanizmu je na obrázku 3.7.

Dekodér [21] predstavuje jednosmernú rekurentnú sieť využívajúcu mechanizmus GRU. Vstupom tejto siete je vektor vzniknutý konkatenáciou výstupu kodéra, na ktorý bol aplikovaný mechanizmus *attention*, s tzv. *embedding* vektorom z predchádzajúceho kroku. *Embedding* vektor reprezentuje užitočné kategorické vzťahy medzi slovami a počas učenia siete sa stále aktualizuje podľa vzťahu:

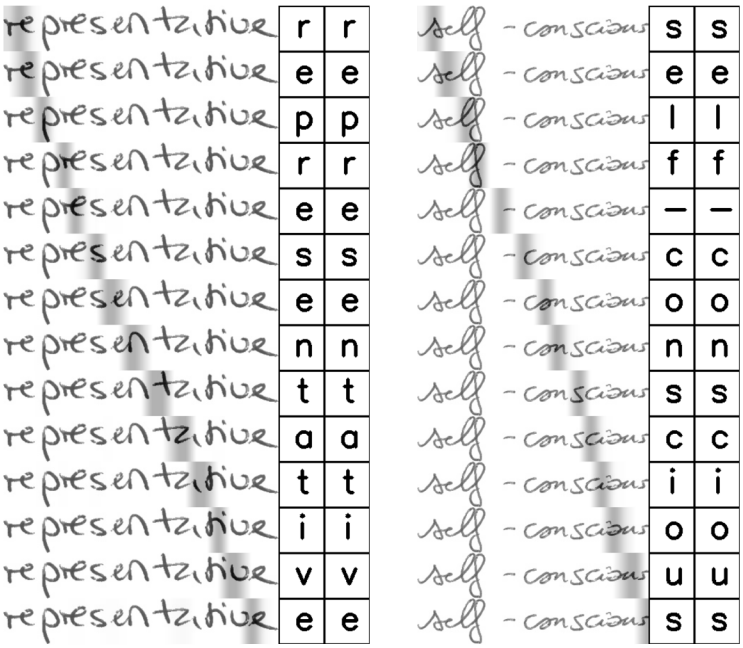
$$y_t = argmax(w(s_t)), \tag{3.29}$$

kde $w(.)$ je lineárna vrstva.

Na začiatku je dekodér inicializovaný štartovacím znakom $\langle GO \rangle$ a končí svoju činnosť načítaním znaku $\langle EOS \rangle$ alebo dosiahnutím maximálneho kroku T. Každé dekódovanie znaku je ovplyvnené uchovaným stavom dekodéra a dekódovaným znakom z kroku s_{t-1}:

$$s_t = Decoder([y_{t-1}, c_t], s_{t-1}), \tag{3.30}$$

kde $[.,.]$ je konkatenácia dvoch vektorov.



Obr. 3.7: Ukážka činnosti mechanizmu *attention* pri rozpoznávaní textu. Obrázky reprezentujú ako sa v jednotlivých časových krokoch mechanizmus *attention* zameria na relevantné časti sekvencie. Text v pravom stĺpci predstavuje *ground truth*. Text v ľavom stĺpci predstavuje výstup siete po dekódovaní aktuálneho znaku. Prevzaté z [21].

Kapitola 4

Dátové sady

Táto kapitola obsahuje prehľad o dátových sadách pre prepis riadkov z historických textov. Ukážka z týchto dátových sád je na obrázku 4.1 a ich prehľad je zobrazený v tabuľke 4.1. Kapitola ďalej obsahuje informácie o nemeckom textovom archíve DTA[1]. Z DTA bola v rámci tejto práce vytvorená dátová sada. Táto sada sa následne použila pri experimentovaní s neurónovou sieťou. Podrobnosti o vytváraní dátovej sady a implementácií neurónovej siete sú popísané v kapitole 6.

4.1 Historické dátové sady

GW20 Rukopisy Georga Washingtona (GW20) [12, 24] predstavujú dátovú sadu zloženú z kolekcie 20 listov. Rukopisy pochádzajú z Kongresovej knižnice (*Library of Congres*) Spojených štátov amerických. Autorom pôvodných listov bol George Washington v období 18. storočia. Jednotlivé listy kolekcie boli neskôr ručne prepisované. Výsledkom práce je dátová sada obsahujúca 4 856 slov. Tieto slová sú rozdelené do 1 187 rôznych tried. Pôvodný súbor dokumentov, z ktorého táto kolekcia pochádza, obsahuje až 4,5 mil. slov. Spoluautorom v týchto dokumentoch bol Thomas Jefferson.

Parzival Dátová sada *Parzival* [11, 12, 13] pozostáva zo 47 strán vytvorených v období 13. storočia. Ide o stredovekú epickú báseň z rukopisu *St. Gall, collegiate library, cod. 857*. Tento rukopis zahŕňa 318 listov. Báseň je napísaná v stredovekom nemeckom jazyku. Dostupné prepisy predstavujú spolu 4 477 riadkov textu, ktoré boli ručne anotované inštitúciou *German Language Institute of the University of Bern*.

Saint Gall Dátová sada *Saint Gall* [10, 11] pozostáva z rukopisu s názvom *Vita sancti Galli*. Autorom tohoto rukopisu je Walafrid Strabo. Dielo je napísané v latinskom jazyku a bolo vytvorené v období 9. storočia. Pôvodné dielo je uložené v knižnici *Abbey Library of Saint Gall* vo Švajčiarsku. Dátová sada obsahuje 60 strán. Každá stránka je napísaná v jednom stĺpci, pričom každý stĺpec obsahuje 24 riadkov textu. Prepisy diela boli vytvorené za pomoci projektu *Monumenta*.

IAM-HistDB *IAM-HistDB* (IAM *Historical Document Database*) [11] je integrovaná verzia vyššie popísaných dátových sád GW20, Parzival a Saint Gall. Dátová sada je tvorená 127 stránkami, ktoré spolu obsahujú 6 543 riadkov textu. Dátovú sadu spravuje výskumná

[1] http://www.deutschestextarchiv.de/

Názov	Anotácia	Jazyk	Storočie	Rozsah
GW20	riadky	anglický	18.	20 strán
Parzival	riadky	nemecký	13.	47 strán
Saint Gall	riadky	latinský	9.	60 strán
IAM-HistDB	riadky	viacjazyčný	9. – 18.	127 strán
IMPACT	regióny a riadky	viacjazyčný	19. – 20.	45 000 strán a 300 strán
Post Scriptum	riadky	španielský a portugalský	16. – 19.	3 500 strán
RIDGES	riadky	latinský a nemecký	15. – 20.	61 diel

Tabuľka 4.1: Prehľad dátových sád.

skupina *Research Group on Computer Vision and Artificial Intelligence of the University of Bern.*

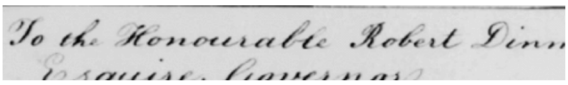
IMPACT IMPACT (*Improving Access to Text*) [33] predstavuje projekt financovaný Európskou komisiou pre zlepšenie prístupu k historickým textom. V projekte je obsiahnutých viac ako 600 tis. obrázkov stránok diel z 10 rozličných krajín a v 18 rozličných jazykoch. Väčšina zastúpených diel pochádza z 19. až 20. storočia. IMPACT obsahuje približne 45 tis. strán, ktoré majú dostupnú informáciu o pozíciách textových regiónoch a približne 300 strán s informáciou o pozíciách riadkov a slov.

Post Scriptum *Post Scriptum* [45] predstavuje projekt, ktorého cieľom bolo vytvoriť a publikovať dátovú sadu pochádzajúcu zo súkromných listov vytvorených v 16. až 19. storočí. Listy sú napísané v španielskom a portugalskom jazyku. Sada obsahuje 3 500 rôznych listov pre každý jazyk.

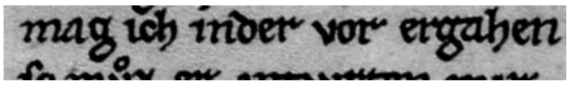
RIDGES RIDGES (*Register in Diachronic German Science*) [29] predstavuje dátovú sadu obsahujúcu texty z obdobia od 15. až do 20. storočia. Texty sú napísané v latinskom a nemeckom jazyku. Prvá verzia korpusu obsahovala výňatky z 15 rôznych publikácií z oblasti bylinkárstva. Najnovšia verzia (verzia 8) obsahuje už 61 rôznych výňatkov.

4.2 Deutsches Textarchiv

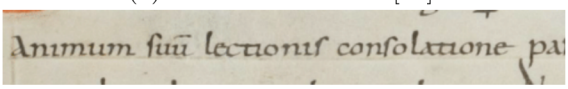
Deutsches Textarchiv (DTA) [8] predstavuje archív nemeckých textov, ktoré boli spojené za účelom vytvorenia vyváženého nemeckého historického korpusu. Archív obsahuje historické texty z obdobia od 17. až do 20. storočia. Najrozšírenejším písmom v dielach je fraktúra. Štatistické čísla týkajúce sa rozsahu archívu sú zobrazené v tabuľke 4.2. Archív tvoria snímky stránok literárnych diel a súbory s prepismi dostupné pre jednotlivé literárne diela. Pri snímkoch sa kladie vysoký dôraz na kvalitu zachyteného textu. Obrázky stránok diel sú dostupné so šírkou 1600 pixelov. Ukážka stránok z vybraných diel je na obrázku 4.2. Vytváranie textov v rámci DTA prebiehalo použitím OCR a aj ručným prepisovaním diel.



(a) GW20. Prevzaté z [24].

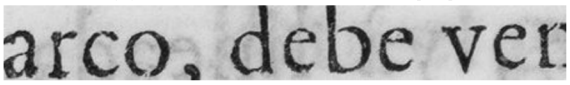


(b) Parzival. Prevzaté z [13].

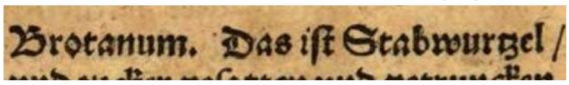


(c) Saint Gall. Prevzaté z [10].

(d) Post Scriptum. Prevzaté z [36].



(e) IMPACT. Prevzaté z [19].



(f) RIDGES. Prevzaté z [29].

Obr. 4.1: Ukážky z dátových sád.

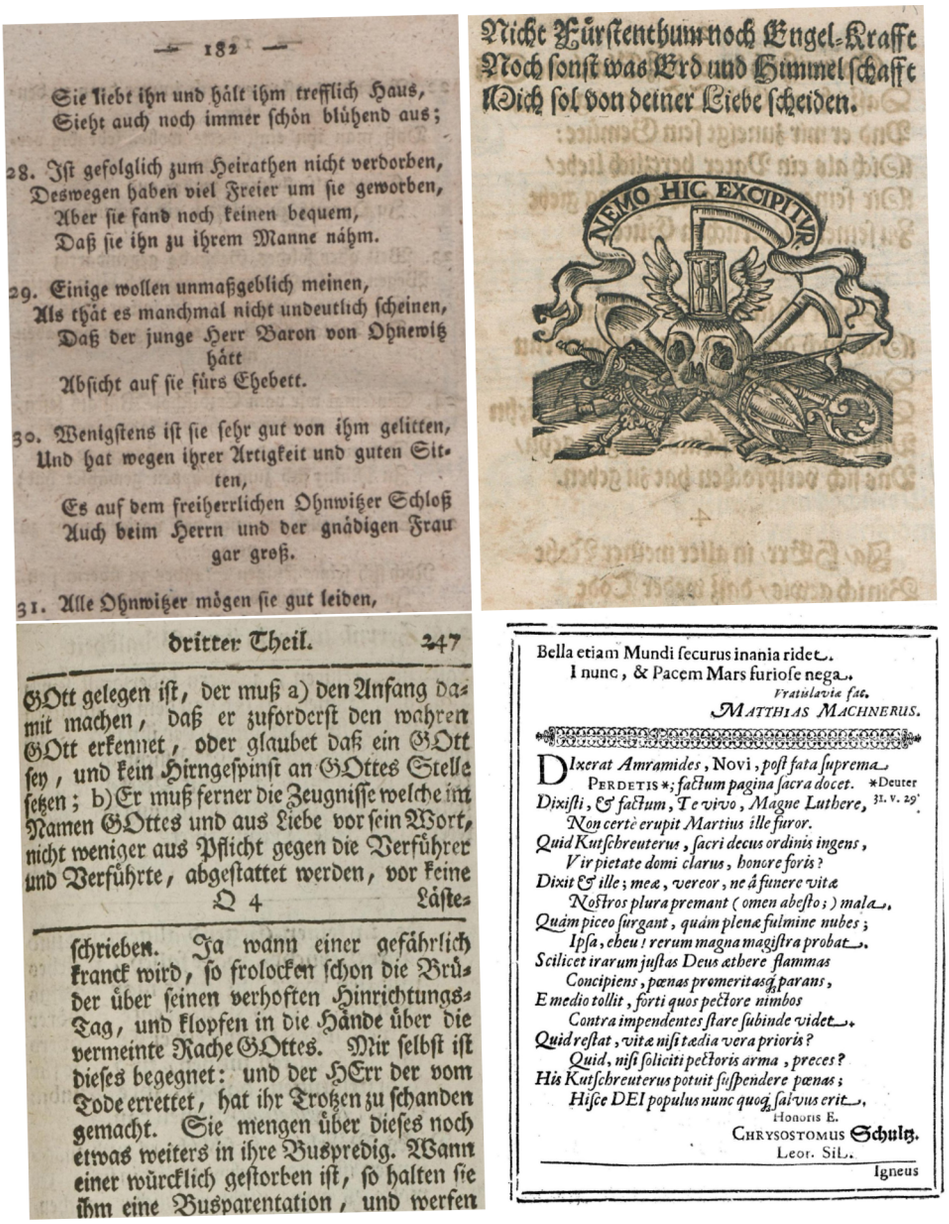
	Počet prác	**Počet tokenov**[4]	**Počet slov**	**Počet znakov**
DTA-Kernkorpus	1405	142 mil.	122 mil.	877 mil.
DTA-Erweiterungen	3017	155 mil.	130 mil.	941 mil.
Spolu	4422	297 mil.	252 mil.	1,8 mld.

Tabuľka 4.2: Prehľad archívu DTA. *DTA-Kernkorpus* predstavuje jadro jazykového korpusu vytvorené pracovníkmi DTA. *DTA-Erweiterungen* predstavuje rozšírenie pôvodného korpusu o diela publikované externými prispievateľmi pomocou DTAE[5].

Spôsob získania prepisov je súčasťou anotácie jednotlivých diel. V rámci tvorby anotácie využíva DTA formát nazývaný DTA-based (DTABf)[2], ktorý vychádza z formátu TEI/P5-Targets[3]. Formát TEI (*Text Encoding Initiative*) predstavuje štandard pre reprezentáciu textov v digitálnej forme.

Vytvorená dátová sada Z dôvodu vysokého počtu kvalitného množstva dát bol archív DTA použitý pri tvorbe dátovej sady. Táto sada sa môže používať pri trénovaní OCR systému pre prepis riadkov s historickým textom. Ukážka konkrétnych výrezov z tejto sady je na obrázku 4.4. Z archívu sa obstarali dáta, ktoré sa následne spracovali, riadky boli v rámci stránok segmentované a rozpoznané OCR systémom. Tieto výstupy boli následne zarovnávané s odpovedajúcimi prepismi. Podrobný postup tvorby dátovej sady je uvedený v časti 6.1. Dátová sada, ktorá sa používa v tejto práci obsahuje 130 147 výrezov riadkov, ktoré nadobúdajú rozličné šírky. Je to čiastočne z dôvodu, že jednotlivé výrezy pochádzajú z diel, ktoré majú odlišný literárny charakter a grafickú úpravu. V rámci dátovej sady sú obsiahnuté výrezy z novín, beletrie a rôznej náučnej literatúry. Histogram zastúpenia jednotlivých šírok je na obrázku 4.3. Okrem tejto verzie dátovej sady existuje druhá verzia, ktorá obsahuje 15 619 591 výrezov riadkov.

[2] http://www.deutschestextarchiv.de/doku/basisformat/
[3] http://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html
[4] najmenší segment textu v jazykovom korpuse (tvar slova alebo iný znak v texte)
[5] http://www.deutschestextarchiv.de/dtae



— 182 —

Sie liebt ihn und hält ihm trefflich Haus,
Sieht auch noch immer schön blühend aus;

28. Ist gefolglich zum Heirathen nicht verdorben,
Deswegen haben viel Freier um sie geworben,
Aber sie fand noch keinen bequem,
Daß sie ihn zu ihrem Manne nähm.

29. Einige wollen unmaßgeblich meinen,
Als thät es manchmal nicht undeutlich scheinen,
Daß der junge Herr Baron von Ohnewitz
hätt
Absicht auf sie fürs Ehebett.

30. Wenigstens ist sie sehr gut von ihm gelitten,
Und hat wegen ihrer Artigkeit und guten Sitten,
Es auf dem freiherrlichen Ohnwitzer Schloß
Auch beim Herrn und der gnädigen Frau
gar groß.

31. Alle Ohnwitzer mögen sie gut leiden,

Nicht Fürstenthum noch Engel-Kraffe
Noch sonst was Erd und Himmel schaffe
Dich sol von deiner Liebe scheiden.

NEMO HIC EXCIPITUR.

dritter Theil. 247

GOtt gelegen ist, der muß a) den Anfang damit machen, daß er zuforderst den wahren GOtt erkennet, oder glaubet daß ein GOtt sey, und kein Hirngespinst an GOttes Stelle setzen; b) Er muß ferner die Zeugnisse welche im Namen GOttes und aus Liebe vor sein Wort, nicht weniger aus Pflicht gegen die Verführer und Verführte, abgestattet werden, vor keine

Q 4 Läste-

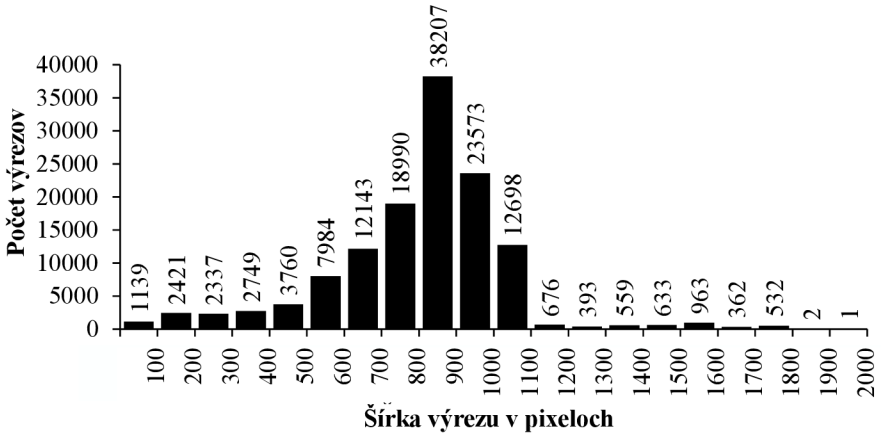
schrieben. Ja wann einer gefährlich kranck wird, so frolocken schon die Brüder über seinen verhoften Hinrichtungs-Tag, und klopfen in die Hände über die vermeinte Rache GOttes. Mir selbst ist dieses begegnet: und der HErr der vom Tode errettet, hat ihr Trotzen zu schanden gemacht. Sie mengen über dieses noch etwas weiters in ihre Buspredig. Wann einer würcklich gestorben ist, so halten sie ihm eine Busparentation, und werfen

Bella etiam Mundi securus inania ridet,
I nunc, & Pacem Mars furiose nega.
Fratulavit fat.
MATTHIAS MACHNERUS.

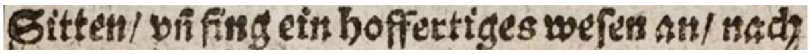
DIxerat Amramides, Novi, post fata suprema,
PERDETIS *; factum pagina sacra docet. *Deuter 31. v. 29.
Dixisti, & factum, Te vivo, Magne Luthere,
Non certè erupit Martius ille furor.
Quid Kutschreuterus, sacri decus ordinis ingens,
Vir pietate domi clarus, honore foris?
Dixit & ille; meæ, vereor, ne à funere vitæ
Nostros plura premant (omen abesto;) mala,
Quàm piceo surgant, quàm plenæ fulmine nubes;
Ipsa, eheu! rerum magna magistra probat.
Scilicet irarum justas Deus æthere flammas
Concipiens, pœnas promeritasq; parans,
E medio tollit, forti quos pectore nimbos
Contra impendentes stare subinde videt.
Quid restat, vitæ nisi tædia vera prioris?
Quid, nisi soliciti pectoris arma, preces?
His Kutschreuterus potuit suspendere pœnas;
Hisce DEI populus nunc quoq; salvus erit.
Honoris E.
CHRYSOSTOMUS Schultz.
Leor. Sil.

Igneus

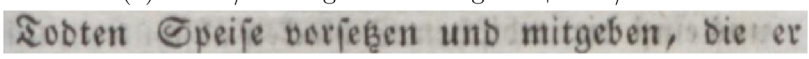
Obr. 4.2: Ukážky z archívu DTA. Prevzaté z [8].



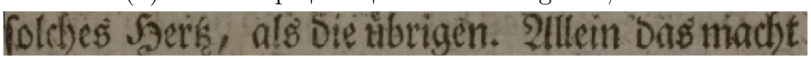
Obr. 4.3: Histogram zastúpenia šírok výrezov riadkov v dátovej sade. V dátovej sade je nerovnomerné zastúpenie krátkych a dlhých výrezov. Toto môže spôsobiť nedostatočné natrénovanie modelov. Najvyššie zastúpenie majú obrázky so šírkou v intervale $I \in (700, 1000)$ pixelov, a preto sú výsledné modely pravdepodobne najpresnejšie pri sekvenciách o takejto šírke výrezu.



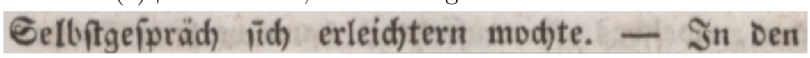
(a) Sitten/ vñ fing ein hoffertiges weſen an/ nach



(b) Todten Speiſe vorſetzen und mitgeben, die er



(c) ſolches Hertz, als die übrigen. Allein das macht



(d) Selbſtgeſpräch ſich erleichtern mochte. — In den

Obr. 4.4: Ukážka riadkov z použitej dátovej sady.

Kapitola 5

Návrh riešenia

V tejto kapitole sa zhodnocujú existujúce metódy spomenuté v časti 2.2 a popisuje sa návrh zvolenej architektúry neurónovej siete. Na túto kapitolu nadväzuje kapitola 6, kde sa detailnejšie popisujú vybrané časti implementácie.

5.1 Porovnanie existujúcich metód

Jednou z možností aplikovateľnou pre rozpoznávanie textu je použitie konvolučnej neurónovej siete. Takáto sieť by obsahovala konvolučné a *pooling* vrstvy, ktoré by zo vstupného obrázka extrahovali vizuálne príznaky. Produktom takejto siete by bol výstupný vektor, na základe ktorého by bolo možné klasifikovať dané slovo alebo znak. Pre uskutočnenie klasifikovania by bola ďalej potrebná databáza jednotlivých tried v podobe slovníka (*lexicon*). Toto riešenie by nebolo vhodné použiť v prípade rozpoznávania sekvencií. Pri sekvenciách sa môže meniť dĺžka daného textu a klasická konvolučná sieť by nedosahovala najlepšie výsledky. V tomto prípade by bolo vhodnejšie použiť rekurentné siete. Konvolučná sieť by sa mohla použiť pri rozpoznávaní znakov alebo pri rozpoznávaní samostatných častí sekvencie pomocou posuvného okna (*sliding window*). Samostatné znaky neobsahujú žiadne závislosti, v podobe kontextu, a tým pádom by nebola potrebná žiadna informácia o znalosti ostatných znakov z rekurentných vrstiev. Ďalším negatívom tohoto riešenia je nutnosť obstarania slovníka na základe dát v dátovej sade.

Ďalšou možnosťou ako uskutočniť rozpoznávanie je pridanie rekurentných vrstiev do predošlej architektúry konvolučnej siete. Takéto siete sa potom zvyčajne vyskytujú spolu s objektívnou funkciou CTC [15]. V tomto prípade by sa riešil problém rozpoznávania sekvencií a sieť by dokázala pracovať so závislosťami medzi jednotlivými znakmi. Pridaním LSTM [18] alebo GRU [5] by sa následne umožnila prevencia voči miznúcemu gradientu (*vanishing gradient*). Takéto siete predstavujú rozšírené riešenia, ktoré sa používajú v mnohých existujúcich prácach [37, 16, 40]. Pri sieťach s CTC sa taktiež často vyskytuje slovník.

Ďalšiu variantu predstavuje použitie *attention* modelov. Ide o rekurentnú sieť, ktorá v sebe obsahuje mechanizmus *attention*. Ide o modernejší prístup ako použitie bežnej rekurentnej siete. V súčasnosti už existujú práce, ktoré tieto modely používajú v prípade rozpoznávania textu a dosahujú zároveň zaujímavé výsledky [21, 43]. Tie siete zvyčajne neobsahujú objektívnu funkciu CTC, ale tzv. *cross entropy*. Pomocou nej postupne klasifikujú jednotlivé predikcie znakov v sekvencii. Na rozdiel od predchádzajúcich modelov, by tomuto modelu postačila databáza znakov, ktoré by sieť dokázala rozpoznať. Ani v tomto prípade sa však nevylučuje použitie slovníka.

5.2 Implementované riešenie

Implementované riešenie v tejto práci vychádza z článku „*Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition*“ [21]. Ide o seq2seq architektúru s mechanizmom *attention*, ktorá sa v pôvodnom článku používa pre rozpoznávanie slov. Implementácia tejto siete je voľne dostupná[1].

V tomto modeli budú uskutočnené zmeny a rozšírenia, ktoré umožnia jeho aplikáciu aj pri rozpoznávaní obrázkov, ktoré obsahujú riadky historického textu. Zjednodušený model tejto architektúry je na obrázku 5.1. Táto architektúra sa skladá z troch častí: z kodéra, z mechanizmu *attention* a z dekodéra.

Kodér siete pozostáva z konvolučnej neurónovej siete, za ktorou sú umiestnené rekurentné vrstvy. Konvolučná sieť musí byť dostatočne veľká, aby dokázala extrahovať všetky relevantné vizuálne príznaky. Na druhej strane môže jej veľkosť výrazne ovplyvniť dobu učenia siete. Z tohoto dôvodu by mohlo byť vhodné zvoliť menšiu architektúru, ktorá bude predstavovať kompromis medzi efektivitou a presnosťou. Podobne vhodné by mohlo byť použitie predtrénovaných modelov. Predtrénované siete môžu výrazne urýchliť priebeh učenia. V tomto prípade sa z predtrénovanej konvolučnej siete odstránia vrstvy slúžiace pre klasifikáciu. Zvyšok siete sa následne pripojí k používanej architektúre. V prípade, ak bude výsledná architektúra dosahovať dobré výsledky, tak bude možné natrénovať celú sieť na vlastných dátach a nepoužiť predtrénované modely. Pri experimentoch sa môžu použiť dostupné architektúry sietí *VGG* [41] alebo *AlexNet* [23].

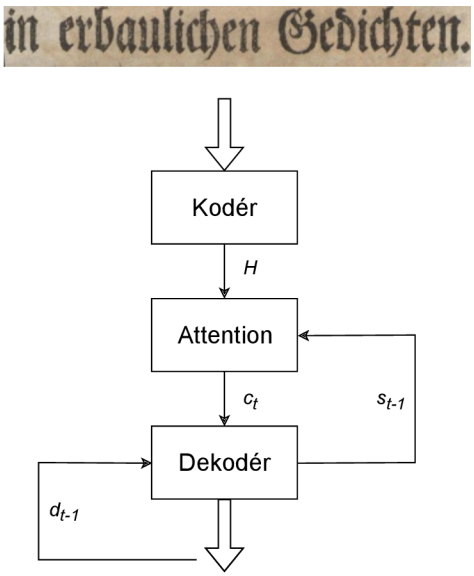
Dekodér pozostáva z rekurentnej siete. Na výstupe tejto siete je objektívna funkcia *cross entropy*. V rámci dekodéra sa môže experimentovať s veľkosťou hodnoty *character embedding*.

Ďalej by sa mohlo v práci použiť pozičné kódovanie (*positional encoding*). Takéto kódovanie použili *Vaswani a spol.* [46]. Ich architektúra neobsahuje žiadne rekurentné vrstvy, a tak museli informáciu o jednotlivých častiach sekvencie zakódovať explicitne. Pôvodná architektúra takýmto kódovaním nedisponuje. Takéto kódovanie by mohlo pozitívne ovplyvniť zameriavanie *attention* modulu, a to by mohlo zlepšiť predikcie pri procese dekódovania. Pozičné kódovanie spočíva v pridaní stáleho vektora, nesúceho pozičné informácie, ku zvolenému vektoru v sieti. V architektúre od *Vaswaniho a spol.* bola táto modifikácia pridaná ako súčasť kodéra aj dekodéra. Podobná modifikácia sa môže uskutočniť aj v tomto prípade. Pozičné kódovanie by mohlo byť umiestnené na výstupe kodéra, pred vstupom do mechanizmu *attention* alebo na vstupe do dekodéra. Ukážka týchto modifikácií je na obrázku 5.2.

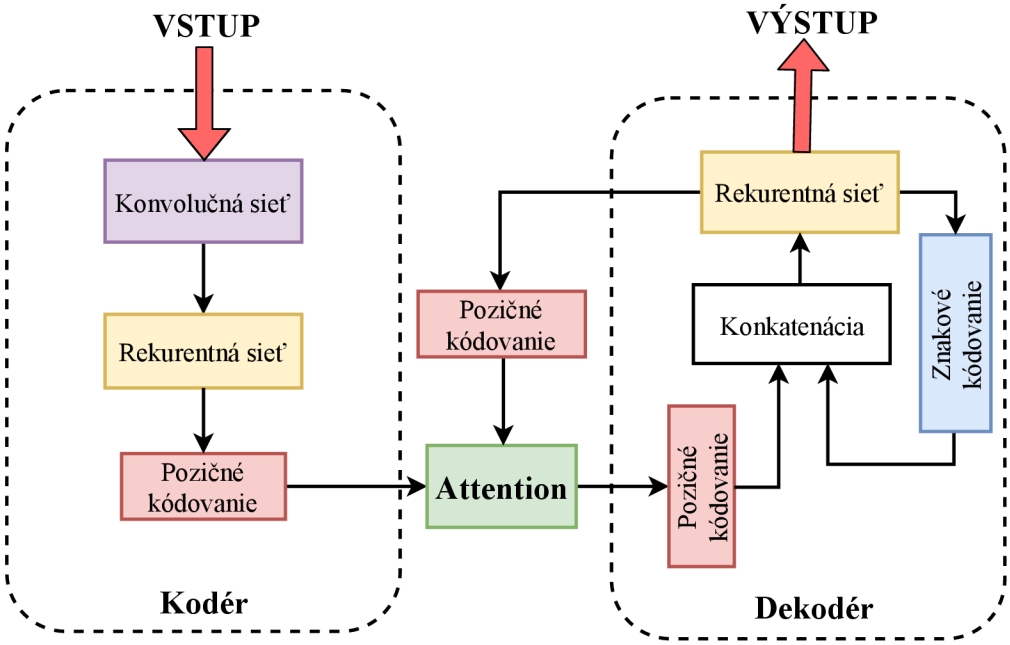
V pôvodnej práci používajú *Lei Kang a spol.* tzv. *Location-based Attention* 3.28. Jeho použitie odôvodňujú nedostatkom pozičných informácií vo výstupe kodéra. Použitie pozičného kódovania by tento problém mohlo vyriešiť a výsledná sieť by mohla dosiahnuť rovnako dobré výsledky aj pri použití *Content-based Attention* 3.25. Podobne by mohol byť *attention* modul nahradený aj niektorým modelom z tabuľky 3.1.

V súvislosti s *attention* modulom súvisí aj samotné učenie siete na základe dát v dátovej sade. Pôvodné seq2seq modely mali problém pri učení na dlhých sekvenciách. Tento problém môže čiastočne pretrvať aj pri použití *attention* modulu. V dátovej sade sú sekvencie premenlivej dĺžky, ktoré by bolo vhodné usporiadať a uskutočniť experiment, ktorý ukáže vplyv dĺžky trénovacích dát na výslednú úspešnosť siete. Spočiatku by sa model trénoval na krátkych sekvenciách a následne by sa sieť mohla dotrénovať na celej dátovej sade. Mechanizmus *attention* by sa takto mohol na krátkych sekvenciách vhodne inicializovať a potom by mohol dávať lepšie výsledky pri dlhých sekvenciách.

[1] https://github.com/omni-us/research-seq2seq-HTR



Obr. 5.1: Schéma architektúry neurónovej siete. Zjednodušený model z práce *Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition* [21]. Na vstupe je obrázok obsahujúci riadok textu. Na výstupe je odpovedajúci prepis. Kodér extrahuje vizuálne príznaky z obrázka a zakóduje ich do sekvencie H. Mechanizmus *attention* aplikuje maskovanie a zameria sa na relevantné časti sekvencie výstupu kodéra. Dekodér postupne v čase $t \in \langle 0, t \rangle$ dekóduje danú sekvenciu c_t a generuje výstup v podobe najpravdepodobnejšieho znaku d_t a tzv. uchovaný stav (*hidden state*) s_{t-1}, ktorý prispieva k tvorbe maskovania. Obrázok s riadkom textu je prevzatý z DTA [8].



Obr. 5.2: Detailnejšie schéma navrhovaného modelu. Ukážka aplikácie pozičného kódovania (*positional encoding*). Kódovanie môže byť pridané na všetky vyznačené miesta, pričom sa môže používať aj kombinácia týchto miest.

Kapitola 6

Implementácia

V nasledujúcej kapitole je popísaná implementačná časť tejto práce. V prvej časti je popísaná tvorba dátovej sady, ktorá je použitá v tejto práci. V ďalšej časti je popísaná implementácia zvoleného modelu neurónovej siete. Z jednotlivých častí implementácie sú spomenuté iba významné časti. Ďalší popis je súčasťou komentárov v zdrojových kódoch a súčasťou súboru *README*.

Pre implementáciu práce bol zvolený programovací jazyk *Python*[1], vo verzií 3.6. V rámci neho sa pri implementácií neurónovej siete použila knižnica *Pytorch*[2].

PyTorch je *open-source* platforma pre flexibilné a efektívne vedecké výpočty zamerané predovšetkým na strojové učenie. Hlavnou výhodou tejto knižnice je jej kompatibilita s knižnicou *numpy*[3]. Ide o jej optimalizovanú verziu, ktorá je schopná bežať na procesoroch GPU a umožňuje pracovať s technológiou *CUDA* od firmy *NVIDIA*. Pytorch je vyvíjaný firmou *Facebook*.

6.1 Tvorba dátovej sady

Dátová sada sa vytvárala z dát obsiahnutých v nemeckom textovom archíve DTA popísanom v časti 4.2. Tieto dáta pozostávajú z obrázkov strán historických diel a následných prepisov k týmto stránkam. Obrázky sa museli segmentovať, aby sa z nich mohli získať výrezy jednotlivých riadky. Prepisy sú v DTA dostupné v niekoľkých formátoch, ale pre tvorbu dátovej sady sa použil formát *xml*. V rámci tohoto formátu sú dostupné formátovacie elementy, ktoré identifikujú riadky a regióny stránok. Problémom je, že nie každý *xml* súbor obsahuje takéto elementy. Kvôli tomuto nedostatku nemusí byť mapovanie riadka a výrezu vždy jednoznačné. Podobne sa pri segmentácií riadkov nemusia všetky riadky segmentovať správne a v správnom poradí. U niektorých riadkov môže dôjsť pri procese segmentácie dokonca k vynechaniu. Problém spojený so správnym mapovaním výrezu riadka s odpovedajúcim prepisom, sa rieši prostredníctvom zarovnávania výstupov metódy OCR. To znamená, že sa použije dostupný OCR nástroj, ktorý dokáže čiastočne rozpoznať jednotlivé riadky a z nich sa do dátovej sady vyberú práve tie riadky, ktoré boli najlepšie rozpoznané a tvoria pár s prepisom obsiahnutom v *ground truth*.

[1] https://www.python.org/
[2] https://pytorch.org/
[3] https://www.numpy.org/

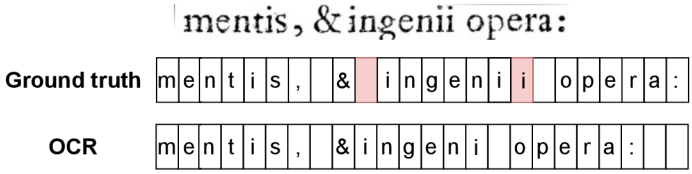
```
<PcGts xmlns="http://schema.primaresearch.org/PAGE/gts/pagecontent/2013-07-15">
    <Page imageFilename="carus_gynaekologie02_1820_0018_1600px" imageWidth="
        400" imageHeight="685">
        <TextRegion id="r1">
            <Coords points="0,0 0,2736 1596,2736 1596,0"/>
            <TextLine id="r1-l001" custom="heights {32, 12}">
                <Coords points="192,184 252,184 304,184 304,228 252,228 192,228"/>
                <Baseline points="192,216 252,216 304,216"/>
                <TextEquiv>
                    <Unicode>XII</Unicode>
                </TextEquiv>
            </TextLine>
        </TextRegion>
    </Page>
</PcGts>
```

Obr. 6.1: Ukážka štruktúry *PAGE xml*. Element *PAGE* reprezentuje stránku dokumentu. Elementy *TextRegion* a *TextLine* reprezentujú textový región a riadok zo stránky. Každý riadok má dostupnú informáciu o jeho polohe vzhľadom k stránke pomocou atribútov *points* v elementoch *Coords* (ohraničenie riadka) a *Baseline* (poloha dotyčnice *baseline*). Prepis riadka je dostupný v elemente *Unicode*.

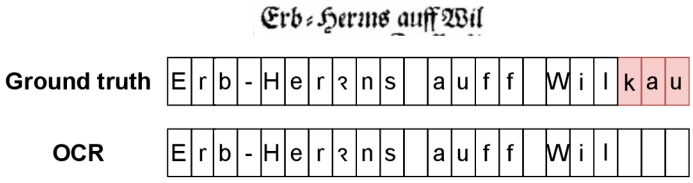
Získanie dát Potrebné obrázky a prepisy boli získané z webových stránok DTA[4]. Sťahovanie bolo potrebné automatizovať z dôvodu, že DTA tieto dáta neposkytuje v žiadnej kompaktnej podobe. Celkový počet získaných dát predstavuje spolu 3 820 rozličných kníh, ktoré spolu obsahujú 642 122 strán v obrazovej podobe so šírkou 1 600 pixelov. Ukážka takýchto stránok je na obrázku 4.2. Textové prepisy sú pre každú knihu dostupné ako jeden *xml* súbor. Tieto prepisy boli spracované a uložené vo formáte *PAGE xml* [35]. Súčasťou formátu *PAGE xml* sú aj informácie o pozíciách riadkov a regiónov, ktoré sa spočiatku ponechali nevyplnené.

Segmentácia a prepis riadkov Keďže boli výsledky nástroja Transkribus (obrázky 2.9 a 2.10) nevyhovujúce, a takisto bol problém s prenášaním dát na server Transkribusu, tak sa pre účely segmentácie použil iný OCR nástroj. Segmentáciu riadkov uskutočnili *Ing. Martin Kišš* a *Ing. Oldřich Kodym* z projektu PERO v dvoch samostatných behoch za pomoci neurónovej siete schopnej detekovať, označiť a následne vyrezať riadky z jednotlivých stránok kníh. Táto sieť [7] používa, pre detekciu a extrahovanie čiar *baseline*, konvolučnú sieť nazývanú *U-net*, ktorá je trénovaná pomocou objektívnej funkcie *dice loss*. Pre zvýšenie kvality výstupu siete sa používa metóda *non-maximum suppression*, ktorá spája jednotlivé fragmenty a produkuje výstupnú *baseline*. V prvom behu sa z 50 kníh získalo celkom 130 147 riadkov. Každý získaný riadok má výšku 48 pixelov. Druhý beh spracoval všetkých 3 820 kníh a získalo sa z neho 15 619 591 riadkov. Tieto riadky boli vyrezané s výškou 32 pixelov. V oboch prípadoch sa tieto vyrezané riadky spracovali pomocou OCR systému, ktorý bol natrénovaný na dátovej sade *IMPACT* [33]. Išlo o OCR systém pozostávajúci z konvolučných a rekurentných vrstiev, ktorý používa objektívnu funkciu CTC [15]. Výstupy siete sú dostupné v *PAGE xml* súboroch obsahujúcich už aj pozície riadkov a regiónov. Ukážka štruktúry *PAGE xml* je na obrázku 6.1.

[4] http://www.deutschestextarchiv.de/



Obr. 6.2: Ukážka zarovnávania prepisu. Levensteinova vzdialenosť pre zobrazené reťazce je rovná veľkosti 2. Ide o dva znaky, ktoré systém OCR rozpoznal nesprávne. Tieto znaky sú zvýraznené červenou farbou. Pri zarovnávaní sa spočíta dĺžka *ground truth* (24 znakov) a následne sa touto hodnotou podelí vypočítaná vzdialenosť:
$1 - \frac{2}{24} = 91\%$. Výsledok udáva percentuálnu zhodu znakov. Keďže je v tomto prípade zhoda vyššia ako 80%, tak by sa riadok *ground truth* spolu s obrázkom pridali do dátovej sady.



Obr. 6.3: Ukážka problému zarovnávania krajných znakov. Hoci systém OCR rozpoznal všetky znaky z obrázka správne, tak *ground truth* v sebe obsahuje ďalšie 3 znaky navyše. Dôvod, prečo nie sú v obrázku zobrazené tieto posledné znaky je spôsobený zlou segmentáciou textu. V prípade výpočtu Levensteinovej vzdialenosti by sme získali hodnotu 3 a celková zhoda by predstavovala 86%. Obrázok a *ground truth* by sa teda vložili do dátovej sady. V tomto prípade by to však predstavovalo problém, pretože by sa obrázok s textom nezhodovali.

Zarovnávanie podľa ground truth Výstupy OCR systému, ktorý rozpoznával detekované riadky, bolo potrebné zarovnať a uistiť sa, že odpovedajú danému výrezu riadka. Pre tieto účeli poslúžil výpočet tzv. *Levensteinovej vzdialenosti* [52], ktorá predstavuje vzdialenosť medzi dvoma textovými reťazcami pre meranie ich odlišnosti. Táto vzdialenosť je definovaná ako minimálny počet operácií so znakmi (vkladanie, mazanie a substitúcia znakov) potrebných pre zámenu jedného reťazca za druhý. Každý výstup OCR systému sa porovnal so všetkými možnými *ground truth* z rovnakej stránky, z ktorej výrez pochádzal. Pre každú dvojicu sa vypočítala Levensteinova vzdialenosť a zo všetkých porovnávaní sa následne určil reťazec, ktorého vzdialenosť bola od prepisu najnižšia. Ukážka zarovnávania je na obrázku 6.2. Pri zarovnávaní sa očakávalo, že sa budú porovnávané reťazce zhodovať aspoň v 80% znakov. Toto číslo sa ukázalo pri spracovaní prvých 130 147 riadkov ako dostačujúce.

Pri vytváraní dátovej sady sa bralo v úvahu, že sa v rámci segmentácie môžu vynechať niektoré krajné znaky reťazca, a napriek tomu by sa prepis po zarovnaní mohol javiť ako správny. Tento problém je zobrazený na obrázku 6.3. Z tohoto dôvodu sa k zarovnávaniu pridala kontrola prvých a posledných N znakov na úplnú zhodu. Keďže sa však vyrezané riadky ukázali ako dostatočne presné a koncové znaky spôsobovali iba zbytočné zahadzovanie dát, tak sa toto filtrovanie pri druhom behu vynechalo.

6.2 Implementácia neurónovej siete

V tejto časti sú popísané niektoré kľúčové a zaujímavejšie časti implementácie. Na túto časť bezprostredne nadväzuje nasledujúca kapitola 7, kde je popísaná zvolená dátová sada a parametre učenia siete.

Premenlivá dĺžka sekvencií Štandardne je v knižnici *pytorch* potrebné, aby mali v jednej dávke (*batch*) obrázky rovnaké výšky aj šírky. V prípade sekvencií môžu jednotlivé sekvencie nadobúdať rozličné dĺžky. Podľa toho sú rozličné aj dĺžky jednotlivých výrezov. Kvôli tomu musela byť implementovaná metóda pre vytváranie vlastných trénovacích dávok. Jedna trénovacia dávka obsahuje identifikátor obrázka, obrázok, jeho dĺžku a prepis. Jednotlivé položky sú v dávke zoradené zostupne.

Predčasné ukončenie Ak po niekoľkých pevne daných iteráciách nedochádza k zmenšeniu veľkostí hodnôt objektívnej funkcie (*loss*), tak dôjde k uloženiu aktuálnych váh v sieti a predčasnému ukončeniu (*early stopping*) procesu učenia. Hraničná hodnota pre predčasné ukončenie sa zadáva ako parameter trénovania.

Pozičné kódovanie Pozičné kódovanie je v rámci modelu implementované v triede *position*. Pri inicializácií tejto triedy dochádza k vytvoreniu matice, ktorá obsahuje zakódované pozičné informácie vzhľadom k veľkosti pôvodného vektora, ku ktorému sa táto informácia pridáva. Hodnota pozičného kódovania sa počíta podľa vzorcov 3.23. Pridaním pozičného kódovania dôjde k zväčšeniu rozmeru pôvodného vektora. V prípade, ak popisuje aktuálny vektor kratšiu sekvenciu, tak sa ako pozičná informácia použije iba časť z pozičného kódovania.

Predikcia sekvencie Neurónová sieť si musí uchovávať dostupný slovník znakov, ktoré dokáže rozpoznať. Pre tieto účely sa získali všetky rozdielne znaky obsiahnuté v dátovej sade. Pri trénovaní dochádza k zakódovaniu každého znaku v prepise *ground truth*. Okrem toho sú dostupné 3 špecifické tokeny: $\langle GO \rangle$, $\langle END \rangle$ a $\langle PAD \rangle$. Tokeny $\langle GO \rangle$ a $\langle END \rangle$ sa pridávajú na začiatok a koniec každého reťazca. Okrem samotného slovníka znakov musí byť v neurónovej sieti prístupná aj informácia o dĺžke najdlhšieho prepisu v počte jeho znakov. Na základe tejto informácie sa všetky krátke sekvencie rozširujú o token $\langle PAD \rangle$ a v konečnom výsledku majú všetky prepisy rovnakú dĺžku. Token $\langle PAD \rangle$ má sieti naznačiť, že v danej časti sekvencie nie je žiadna ďalšia informácia.

Pri procese dekódovania výstupnej sekvencie sa používa výstupný vektor, ktorý obsahuje pravdepodobnosti jednotlivých znakov pre každý krok sekvencie. Z každého kroku sa z vektora určí najväčšia hodnota, ktorá reprezentuje najpravdepodobnejší znak pre daný krok. Všetky výsledné hodnoty sa prevedú na odpovedajúce znaky zo slovníka a postupným spojením všetkých znakov sa získa výstupný prepis. Proces predikcie prepisu ukončuje token $\langle END \rangle$.

Kapitola 7

Experimenty a výsledky

V tejto kapitole sú popísané vykonané experimenty s implementovanou neurónovou sieťou. Na začiatku kapitoly sú definované metriky, ktoré sa používajú pri vyhodnocovaní presností. Ďalej sú popísané základné parametre architektúry figurujúcej v experimentoch. Súčasťou práce sú štyri experimenty. Prvý experiment sa zaoberá výberom vhodnej architektúry konvolučnej neurónovej siete. Druhý experiment súvisí s hľadaním vhodnej hodnoty hyperparametru *learning rate*. Tretí experiment popisuje aplikáciu pozičného kódovania v rozdielnych častiach kodéra a dekodéra. Posledný experiment skúma vplyv dĺžky riadkov na proces trénovania a úspešnosti neurónovej siete. V poslednej časti sú výsledky dotrénovaných modelov a zhrnutie dosiahnutých výsledkov.

7.1 Výpočet presnosti

Pre výpočet presnosti je potrebná stanovená metrika. V prípade rozpoznávania textu sa pre tieto potreby používa *Character Error Rate* (CER) a *Word Error Rate* (WER). Tieto metriky umožňujú porovnávať presnosti sietí nezávisle na dĺžke rozpoznávanej sekvencie.

CER *Character Error Rate* [3] predstavuje podiel medzi minimálnym počtom operácií, potrebných pre zámenu zdrojového reťazca za cieľový reťazec a dĺžkou cieľového reťazca. Matematicky sa táto metrika definuje ako:

$$CER = \frac{(i + s + d)}{n}, \tag{7.1}$$

kde n predstavuje počet znakov v reťazci, i je počet operácií vkladania znakov, s je počet operácií substitúcie znakov a d je počet operácií mazania znakov.

WER *Word Error Rate* [3] predstavuje podiel medzi minimálnym počtom operácií, potrebných pre zámenu zdrojového slova za cieľové slovo a počtom slov v cieľovom reťazci. Hodnota WER býva zvyčajne vyššia ako hodnota CER. Táto metrika je vyjadrená ako:

$$WER = \frac{(i_w + s_w + d_w)}{n_w}, \tag{7.2}$$

pričom jednotlivé parametre popisujú tie isté vlastnosti, ako v prípade CER, akurát sú v tomto prípade operácie a dĺžka vymedzené na slová a nie na celú sekvenciu.

V prípade sekvencií môže chyba presahovať aj hodnotu 100 %. Je to v prípade, ak výstup siete obsahuje vyšší počet znakov alebo slov ako *ground truth*. V takomto prípade je možné chybu normalizovať pomocou vzťahu:

$$Normalized = \frac{Error}{(i + s + d + c)}, \tag{7.3}$$

kde $Normalized$ je normalizovaná chyba, $Error$ je pôvodná chyba, c je počet správne prepísaných znakov, i, s a d sú operácie nad znakmi. V nadchádzajúcich experimentoch sa takáto normalizácia nepoužíva.

7.2 Použitá dátová sada

Pri experimentoch bola použitá vytvorená dátová sada (viz kapitola 6.1), konkrétne jej prvá verzia s menším rozsahom. Táto sada pozostáva z 130 147 riadkov, ale pri trénovaní sa použilo iba 126 026 riadkov. Zo sady sa vynechali riadky, ktoré boli príliš dlhé, pretože ich bolo málo a nepôsobili priaznivo pri počiatočných experimentoch. Riadky sa rozdelili na tri disjunktné množiny dát, ktoré postupne predstavujú trénovaciu sadu (96 789 riadkov), validačnú sadu (9 807 riadkov) a testovaciu sadu (19 430 riadkov). Rozdelenie do jednotlivých množín bolo náhodné za pomoci algoritmu *Fisher–Yates*[1].

Validačná sada slúži predovšetkým na zisťovanie progresu siete. V prípade útlmu sa môže učenie prerušiť, upravia sa hyperparametre siete (predovšetkým *learning rate*) a po načítaní uloženého modelu siete sa pokračuje v učení. Testovacia sada slúži na zisťovanie presnosti v prípade už natrénovaného modelu. V rámci experimentov sa vyhodnocujú hodnoty WER a CER aj pre trénovaciu dátovú sadu.

Pred trénovaním dochádza k normalizácií riadkov podľa pravidiel, ktoré definuje balíček *torchvision* pre svoje predtrénované modely. Z dátovej sady je potrebná informácia o šírke najširšieho obrázka a podľa toho sa do tejto šírky dopĺňajú krátke výrezy čiernou farbou.

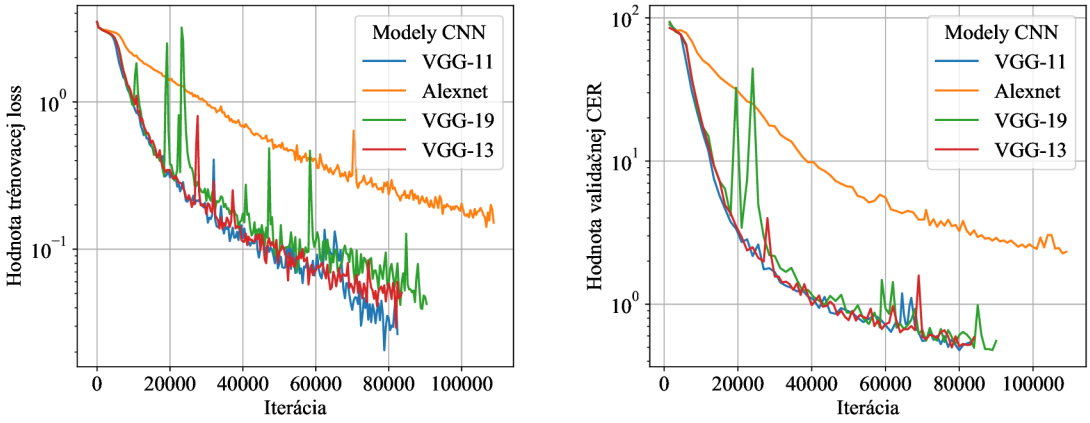
7.3 Použité parametre

V kodéri aj v dekodéri sa použili 2 za sebou umiestnené rekurentné vrstvy. Všetky experimenty prebiehali s veľkosťou rekurentných vrstiev 512. V rámci rekurentných vrstiev sa použil *dropout* [42] s pravdepodobnosťou 50 %. Ako *attention* mechanizmus sa použil *Location-Based Attention* [21, 28]. Veľkosť trénovacej dávky (*batch*) bola 16. Slovník obsahuje 192 rôznych znakov. Vstup do siete predstavujú obrázky s výškou 48 pixelov a maximálnou šírkou 1 100 pixelov. Pre výpočet chyby sa používa objektívna funkcia *cross entropy* a ako optimalizačný algoritmus sa používa *Adam* [22].

7.4 Experiment s konvolučnou sieťou

Hlavným cieľom tohoto experimentu bolo nájsť vhodnú architektúru konvolučnej siete, ktorá je súčasťou kodéra. V tomto experimente sa vychádzalo z predtrénovaných modelov, ktoré ponúka balíček *torchvision*[2]. Tieto modely boli natrénované na dátovej sade *ImageNet* [6]. Z dostupných modelov konvolučných sietí boli použité architektúry *VGG* [41] a *Alexnet* [23]. Z modelov konvolučných sietí boli odstránené posledné vrstvy, ktoré slúžili na klasifikovanie

[1] https://onlinerandomtools.com/shuffle-lines
[2] https://pytorch.org/docs/stable/torchvision/index.html



Obr. 7.1: Ukážka rýchlosti učenia v závislosti od architektúry konvolučnej siete. Pri všetkých modeloch sa použila konštantná veľkosť *lerning rate* – 2×10^{-5}. Veľkosť objektívnej funkcie (*loss*) sa pri jednotlivých iteráciách znižovala u každého modelu (obrázok vľavo). Najrýchlejšie sa znižovala hodnota *loss* u druhej najmenšej architektúry – *VGG-11*. Ukážka znižovania veľkosti CER je na obrázku vpravo. Architektúry VGG dosahujú približne rovnaké presnosti.

výstupného vektora. V rámci VGG sa použili tri modely s rozdielnym počtom konvolučných a *pooling* vrstiev. Všetky modely boli trénované s rovnakými parametrami, pričom hodnota *learning rate* mala veľkosť 2×10^{-5}. Nižšia hodnota *learing rate* má síce tendenciu spomaliť čas učenia, ale na druhú stranu by učenie malo prebiehať stabilnejšie, pretože by nemalo dochádzať k veľkým zmenám vo vnútorných váhach siete.

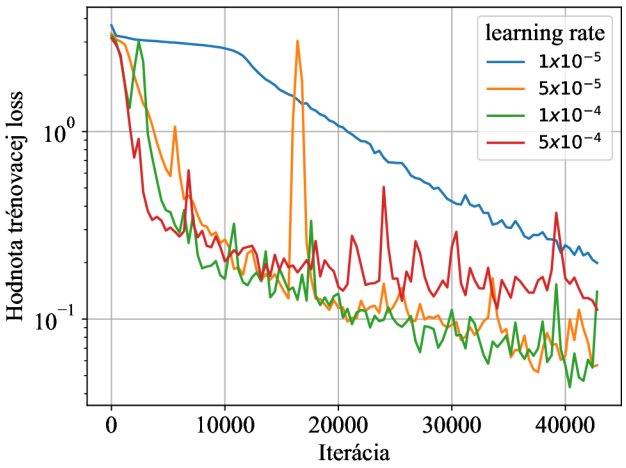
Výsledky experimentu poukazujú na to, že v prípade architektúr VGG sa najrýchlejšie učila menšia sieť s nižším počtom vrstiev. Jednotlivé architektúry VGG dosiahli rozdielne hodnoty veľkostí *loss*, ale v prípade merania presností dosahovali tieto siete veľmi podobné výsledky. Architektúra *Alexnet* predstavuje odlišný typ modelu, ktorého hlavnou výhodou je efektívny beh na viacerých kartách GPU. V tomto prípade sa ale experimenty s viacerými GPU neuskutočňovali a architektúra *Alexnet* nadobúdala vysoké hodnoty veľkostí *loss* a CER.

Po vykonaní experimentu sa do navrhovanej architektúry pridala konvolučná sieť typu VGG-11. Okrem rýchlejšieho učenia má tento model aj nízke pamäťové nároky. V dôsledku nižších pamäťových nárokov sa s týmto modelom môžu vykonávať experimenty aj na GPU s menšou veľkosťou pamäte. Výsledky experimentu je možné vidieť na grafoch 7.1.

7.5 Experiment s learning rate

Úlohou tohoto experimentu je zistiť ideálnu hodnotu hyperparametru *learning rate*, pričom je v modeli použitá konvolučná sieť *VGG-11*. V rámci experimentu bolo zvolených niekoľko rozličných hodnôt *learning rate*. *Learning rate* predstavuje dôležitý hyperparameter, ktorý rozhoduje o veľkosti zmien vo vnútorných váhach v sieti. Výsledky experimentu na trénovacej sade sú na grafe 7.2.

Pri experimente sa uskutočnilo 43 200 iterácií. Nižší počet iterácií bol zvolený z dôvodu, pretože sa v tomto experimente nesleduje ani tak výsledná presnoť, ako rýchlosť učenia



Obr. 7.2: Ukážka rýchlosti učenia siete v závislosti od parametru *learning rate*.

siete. Najnižšie hodnoty *loss* dosiahli siete s veľkosťou *learning rate* 1×10^{-4} a 5×10^{-5}. Sieť s hodnotou *learning rate* 1×10^{-5} sa učila konštantne, ale pomaly. V prípade vyšších hodnôt *learning rate* dochádza časom k zhoršeniu učenia. Na obrázku 7.2 to je možné vidieť u „červenej krivky“, u ktorej sa hodnota *loss* spočiatku znižovala najrýchlejšie, ale potom nastal pri iterácií 20 000 útlm. Sieť sa potom ďalej nijak viditeľne nezlepšovala.

Z tohoto experimentu vyplýva, že je sieť vhodné začať trénovať s hodnotou *learning rate* v intervale $lr \in (5 \times 10^{-5}, 1 \times 10^{-4})$. V prípade, ak počas trénovania dôjde k zastaveniu progresu, tak sa bude sieť dotrénovávať s nižšou hodnotou *learning rate* (napr. 1×10^{-5}).

7.6 Experimenty s pozičným kódovaním

V rámci tohoto experimentu sa zisťuje vplyv stáleho pozičného kódovania (*positional embedding*), ktoré použili *Vaswani a spol* [46]. Úlohou pozičného kódovania je zakódovať informáciu o pozíciách jednotlivých častí sekvencie. Príznaky reprezentované vektorom s pozičným kódovaním môžu dekodéru a mechanizmu *attention* pomôcť lepšie vyhľadať znaky v sekvencií. V tomto experimente sa vychádza z 5 rozdielnych umiestnení kódovania:

1. umiestnenie za výstup kodéra (K),
2. umiestnenie pred vstup do rekurentných vrstiev dekodéra (D),
3. kombinácia 1. a 2. ($K + D$),
4. umiestnenie pred vstup do mechanizmu *attention* (A),
5. kombinácia 1. a 4. ($K + A$).

Ukážka týchto umiestnení je na obrázku 5.2. V tabuľke 7.1 sú následne výsledné presnosti modelov počítané na trénovacej aj testovacej dátovej sade.

Najlepšie výsledky dosiahla sieť, ktorá použila kódovanie pred vstupom do mechanizmu *attention* (A). Výsledná sieť dokáže rozpoznávať znaky s chybami $0,47$ % na trénovacej sade a $0,67$ % na testovacej sade. Pridanie pozičného kódovania pred vstup do mechanizmu

Umiestnenie	Trénovacia sada		Testovacia sada	
	WER	**CER**	**WER**	**CER**
Žiadne	5,59 %	1,35 %	6,26 %	1,49 %
Výstup kodéra (K)	2,66 %	0,55 %	3,79 %	0,79 %
Vstup dekodéra (D)	4,77 %	1,06 %	6,16 %	1,38 %
Kombinácia K+D	4,06 %	0,83 %	4,81 %	0,98 %
Vstup *attention* (A)	**2,00 %**	**0,47 %**	**2,70 %**	**0,66 %**
Kombinácia K+A	3,07 %	0,81 %	3,86 %	0,97 %

Tabuľka 7.1: Chybovosti sietí pri použití rozličného umiestnenia pozičného kódovania.

attention pravdepodobne umožňuje lepšie využitie informácie o polohách znakov v sekvencií, na základe rozhodnutia dekodéra.

Ďalšie modely s dobrými výsledkami obsahujú pozičné kódovanie na výstupe kodéra (K) a kódovanie vzniknuté kombinovaním $K + A$. Použitie týchto kódovaní prinieslo lepšie výsledky ako klasická sieť bez pozičného kódovania.

Použitie samostatného kódovania pri vstupe do dekodéra prinieslo najhoršie výsledky spomedzi modelov používajúce pozičné kódovanie. Výsledky tohoto kódovania sú porovnateľné so sieťou, v ktorej toto kódovanie nebolo použité. V tomto prípade bola táto informácia pravdepodobne redundantná a nepriniesla sieti žiadne významné informácie.

7.7 Experiment s dátovou sadou

Pri tomto experimente sa zisťuje dopad dĺžok sekvencie na proces trénovania siete. V predchádzajúcich experimentoch prebiehalo trénovanie s náhodne zamiešanou dátovou sadou. Natrénované modely z predchádzajúceho experimentu majú pritom problém pri rozpoznávaní dlhých sekvencií. Túto problematiku je možné vidieť na obrázku 7.3.

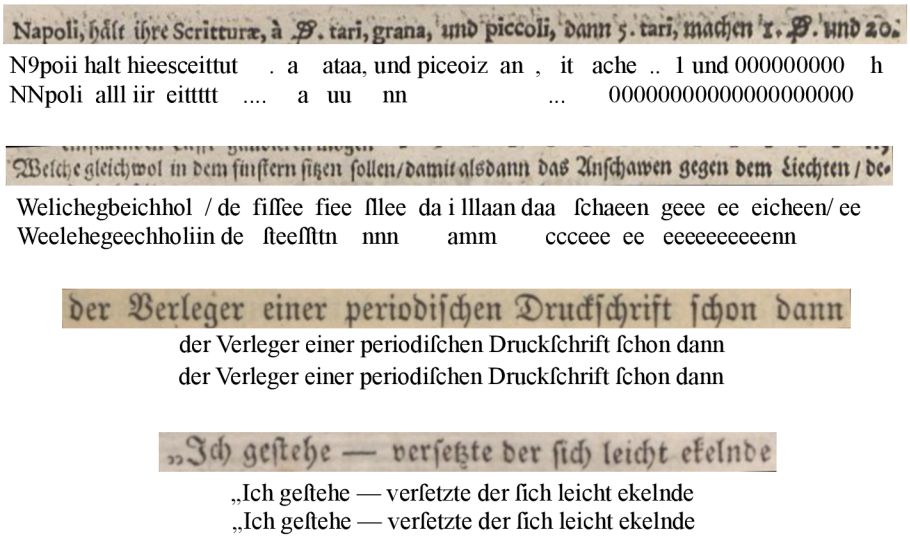
V histograme 4.3 sú zobrazené šírky obrázkov zastúpené v celej dátovej sade. Podľa týchto údajov môže byť problém prepisu dlhých sekvencií čiastočne spôsobený nedostatkom dlhých sekvencií v dátovej sade. Ďalší problém môže predstavovať mechanizmus *attention*, ktorý sa pri rozdielnych dĺžkach sekvencie nemusí dokázať vhodne inicializovať.

V tomto experimente sa z pôvodnej trénovacej dátovej sady zvolilo 24 906 riadkov, u ktorých je šírka obrázka nižšia ako hodnota 700 pixelov. Na týchto dátach sa postupne natrénovali dva modely, pričom toto trénovanie trvalo približne 50 % dosiahnutého času v predchádzajúcom experimente. Následne sa modely dotrénovali na všetkých dátach z pôvodnej dátovej sady.

Na obrázku 7.4 je možné vidieť, že v prípade architektúry, ktorá neobsahuje pozičné kódovanie dochádzalo takýmto predtrénovaním ku zlepšeniu. V priebehu trénovania bola hodnota objektívnej funkcie tejto siete nižšia ako hodnota u pôvodného modelu. V prípade modelu obsahujúceho pozičné kódovanie sa toto predtrénovanie neusvedčilo a výsledná sieť dosahovala vyššie hodnoty chyby.

7.8 Dotrénovanie modelov

Po získaní poznatkov z predchádzajúcich experimentov boli vybrané najlepšie modely, ktoré sa postupne trénovali až do doby, dokým klesali ich hodnoty *loss* a veľkosti chyby CER



Obr. 7.3: Ukážka predikcie siete. V hornej časti sú výrezy, ktoré boli predikované s vysokou nepresnosťou. V dolnej časti sú typické výrezy v dátovej sade, ktoré boli prepísané s presnosťou 100 %. Zlé predikované výrezy sú oproti ostatným výrezov v dátovej sade širšie a obsahujú vysoký počet znakov v sekvencií.

Umiestnenie	**Trénovacia sada**		**Testovacia sada**	
	WER	**CER**	**WER**	**CER**
Výstup kodéra (K)	1,32 %	0,28 %	1,88 %	**0,41** %
Vstup *attention* (A)	**0,93** %	**0,24** %	2,60 %	0,53 %
Kombinácia K+A	1,03 %	0,26 %	**1,87** %	**0,41** %

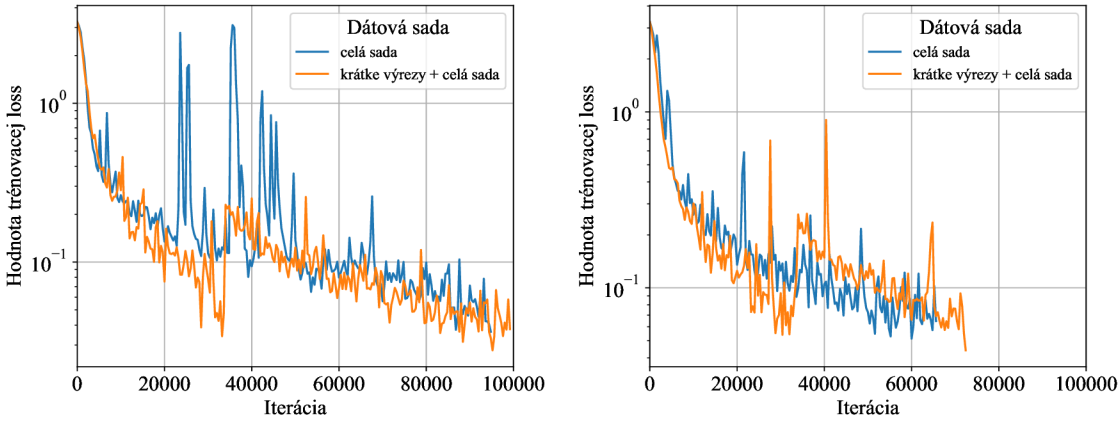
Tabuľka 7.2: Výsledky dotrénovaných modelov na pôvodnej sade.

na validačnej sade. Výsledky dosiahnutých presností sú u jednotlivých modelov zobrazené v tabuľke 7.2.

Najlepšie výsledky dosiahol na trénovacej sade model s pozičným kódovaním umiestneným pred vstupom do mechanizmu *attention* (A). V prípade testovacej sady dosiahli lepšie výsledky modely obsahujúce pozičné kódovanie na výstupe kodéra (K a $K + A$). Progres experimentu je zobrazený na obrázku 7.5.

V rámci tohoto experimentu sa uskutočnil aj experiment s trénovaním siete na dlhých výrezoch. Z pôvodnej dátovej sady sa vyčlenili dlhšie výrezy so šírkou minimálne 750 pixelov. Táto sada sa následne použila pri trénovaní natrénovaných modelov sietí. Výsledky týchto modelov sú v tabuľke 7.3.

Ukázalo sa, že trénovanie na dlhých výrezoch čiastočne rieši problém s dlhými sekvenciami. Za cenu zvýšenia presnosti pri dlhých sekvenciách sa však čiastočne u modelov K a $K + A$ znížila presnosť na pôvodnej testovacej sade. Model s pozičným kódovaním umiestneným pred vstupom do mechanizmu *attention* dosiahol hodnoty CER $0,16$ % pri trénovacej sade a $0,37$ % pri testovacej sade. V tomto prípade sa model zlepšil na oboch dátových sadách.



(a) Sieť bez pozičného kódovania.

(b) Sieť s pozičným kódovaním K.

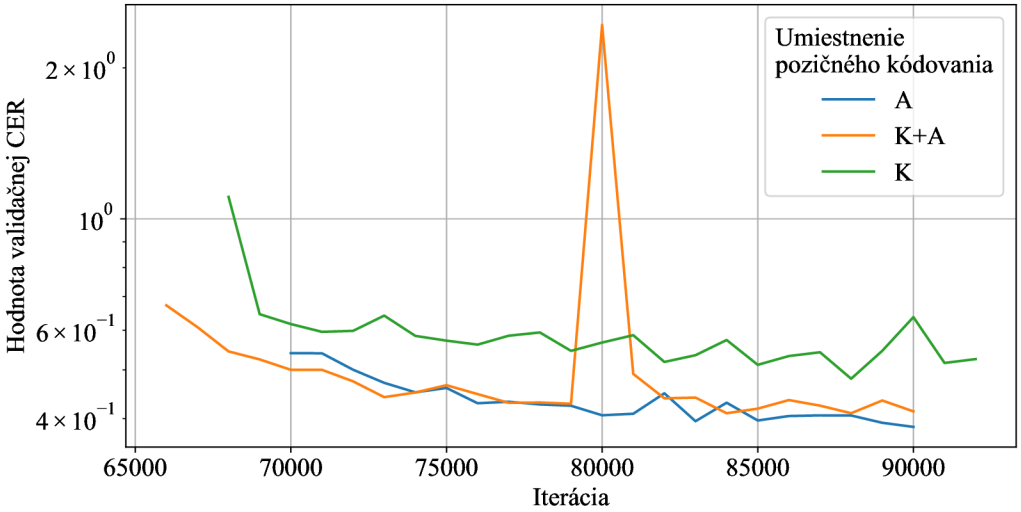
Obr. 7.4: Porovnanie priebehu učenia v prípade predtrénovaného *attention* mechanizmu na kratších sekvenciách. Pri iterácií 30 000 je pri „oranžových“ krivkách skok, pretože sa v tomto bode začal daný model trénovať na novej dátovej sade. „Modrá krivka“ predstavuje totožný model trénovaný na pôvodnej dátovej sade.

Umiestnenie	**Trénovacia sada**		**Testovacia sada**	
	WER	**CER**	**WER**	**CER**
Výstup kodéra (K)	0,75 %	0,21 %	2,41 %	0,52 %
Vstup *attention* (A)	**0,46** %	**0,16** %	**1,64** %	**0,37** %
Kombinácia K+A	0,69 %	0,27 %	1,93 %	0,48 %

Tabuľka 7.3: Výsledky dotrénovaných modelov na dlhých riadkoch. Pre účeli testovania bola použitá pôvodná testovacia sada. Trénovacia sada pozostáva v tomto prípade iba z dlhých sekvencií.

7.9 Zhrnutie výsledkov

Z predchádzajúcich výsledkov vyplýva, že použitie menšej *VGG* architektúry dokáže priniesť rovnako dobré výsledky ako použitie rovnakého typu siete s vyšším počtom konvolučných a *pooling* vrstiev. Použitie pozičného kódovania na mieste vstupu do mechanizmu *attention* dokázalo znížiť hodnotu chyby CER a výsledná sieť dokáže rozpoznať znaky s chybou 0,66 %. Pri experimente s dotrénovaním modelov klesla pôvodná hodnota CER u tohoto modelu na hodnotu 0,53 %. Modely s pozičným kódovaním umiestneným na výstupe kodéra (K) a kombináciou $K + A$ dosiahli porovnateľné výsledky. Pri dotrénovaní sietí dokonca prekonali pôvodne najlepší model o 0.08 %. Predtrénovanie siete na krátkych sekvenciách prinieslo zrýchlenie poklesu objektívnej funkcie iba v prípade trénovania siete bez pozičného kódovania. Nakoniec boli siete trénované na dlhších sekvenciách a najlepšie výsledky dosiahla opäť sieť s pozičným kódovaním umiestneným pred vstupom do mechanizmu *atention* (A). Výsledná sieť dokáže rozpoznávať znaky v sekvencií s chybou 0,37 %.



Obr. 7.5: Zmeny veľkostí validačnej CER pri dotrénovaných modeloch. Popis typov pozičného kódovania je v časti 7.6. Najniššiu veľkosť chyby CER dosiahol model s kódovaním *A*.

Kapitola 8

Záver

Cieľom tejto práce bolo navrhnúť a analyzovať neurónovú sieť slúžiacu pre rozpoznávanie segmentovaných riadkov z historických textov. Tieto texty pochádzajú z obdobia od 17. až 19. storočia a sú napísané písmom fraktúra.

Pre tieto účely sa použila architektúra neurónovej siete *sequence-to-sequence* [21]. Táto architektúra pozostáva z kodéra, mechanizmu *attention* a dekodéra. Kodér slúži na extrahovanie vizuálnych čŕt z obrázka a ich zakódovanie do matice nazývanej *context*. Mechanizmus *attention* slúži na zameranie významných častí tejto matice, ktoré by mali predstavovať práve dekódovaný znak alebo jeho časť. Dekodér následne postupne dekóduje položky tejto matice a generuje odpovedajúce znaky, ktoré spolu vytvárajú výstupnú sekvenciu.

V prvej časti tejto práce boli vytvorené dve verzie dátovej sady pre rozpoznávanie historického písma. Obe sady vychádzajú z nemeckého textového archívu *Deutsches Textarchiv* [8], ktorý obsahuje 3 897 historických textov v nemeckom jazyku. Výsledné dátové sady pozostávajú celkovo z 130 147 riadkov a z 15 619 591 riadkov.

V rámci experimentov bola skúmaná štruktúra siete a vplyv pozičného kódovania na výslednú predikciu. V prvých experimentoch sa hľadala vhodná architektúra konvolučnej siete spolu s vhodnou veľkosťou *learning rate*. V ďalšom experimente sa skúmalo pozičné kódovanie. Toto kódovanie spočíva v pridaní stáleho vektora, nesúceho informácie o pozíciách častí sekvencie, k vybraným vektorom v sieti. Spolu sa použili tri rôzne umiestnenia pozičného kódovania, konkrétne sa toto kódovanie aplikovalo ku výstupu kodéra, ku vstupu do mechanizmu *attention* a ku vstupu do dekodéra. Pozičné kódovanie dokázalo zvýšiť presnosť siete, pričom najlepšie výsledky priniesla verzia, ktorá spočívala v umiestnení pozičného kódovanie pred vstup do mechanizmu *attention*. Výsledná sieť dosiahla s týmto kódovaním presnosť rozpoznávania znakov $99,63$ %.

V nadväzujúcej práci by sa mohla implementovaná sieť optimalizovať a mohli by sa do nej zaviesť techniky, ktoré by urýchlili jej učenie. Ďalej by bolo potrebné uskutočniť vykonané experimenty na dlhšie rozsahy v podobe niekoľko stoviek tisíc iterácií a uskutočniť ich na druhej verzii dátovej sady. Podobne by sa mohli vykonať experimenty náhradou pôvodného mechanizmu *attention* [21] alebo by sa v práci mohol použiť dostupný jazykový model.

Literatúra

[1] Bahdanau, D.; Cho, K.; Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, ročník abs/1409.0473, 2015.

[2] Bengio, Y.; Simard, P.; Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, ročník 5, č. 2, March 1994: s. 157–166, ISSN 1045-9227.

[3] Carrasco, R. C.: Text Digitisation. 2014, [Online; navštívená 08.05.2019].
URL https://sites.google.com/site/textdigitisation/

[4] Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; aj.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, ročník abs/1406.1078, 2014.

[5] Chung, J.; Gülçehre, Ç.; Cho, K.; aj.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, ročník abs/1412.3555, 2014.

[6] Deng, J.; Dong, W.; Socher, R.; aj.: Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, s. 248–255.

[7] Diem, M.; Kleber, F.; Fiel, S.; aj.: cBAD: ICDAR2017 Competition on Baseline Detection. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, nov 2017.
URL https://doi.org/10.1109/icdar.2017.222

[8] Deutsches Textarchiv. Grundlage für ein Referenzkorpus der neuhochdeutschen Sprache. Berlin-Brandenburgischen Akademie der Wissenschaften, Berlin, 2019, [Online; navštívená 08.05.2019].
URL http://www.deutschestextarchiv.de/

[9] Fink, M.; Layer, T.; Mackenbrock, G.; aj.: Baseline Detection in Historical Documents using Convolutional U-Nets. *arXiv e-prints*, Oct 2018: arXiv:1810.09343.

[10] Fischer, A.; Frinken, V.; Fornés, A.; aj.: Transcription Alignment of Latin Manuscripts Using Hidden Markov Models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-0916-5, s. 29–36.

[11] Fischer, A.; Indermühle, E.; Bunke, H.; aj.: Ground Truth Creation for Handwriting Recognition in Historical Documents. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, New York, NY, USA: ACM, 2010, ISBN 978-1-60558-773-8, s. 3–10.

[12] Fischer, A.; Keller, A.; Frinken, V.; aj.: Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, ročník 33, č. 7, 2012: s. 934 – 942, ISSN 0167-8655, special Issue on Awards from ICPR 2010.

[13] Fischer, A.; Wüthrich, M.; Liwicki, M.; aj.: Automatic Transcription of Handwritten Medieval Documents. 09 2009.

[14] Gehring, J.; Auli, M.; Grangier, D.; aj.: Convolutional Sequence to Sequence Learning. *CoRR*, ročník abs/1705.03122, 2017.

[15] Graves, A.; Fernández, S.; Gomez, F.; aj.: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. 2006.

[16] Graves, A.; Liwicki, M.; Fernández, S.; aj.: A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 31, č. 5, May 2009: s. 855–868, ISSN 0162-8828.

[17] Grüning, T.; Labahn, R.; Diem, M.; aj.: READ-BAD: A New Dataset and Evaluation Scheme for Baseline Detection in Archival Documents. *arXiv e-prints*, May 2017: arXiv:1705.03311.

[18] Hochreiter, S.; Schmidhuber, J.: Long Short-Term Memory. *Neural Computation*, ročník 9, č. 8, 1997: s. 1735–1780.

[19] IMPACT Centre of Competence. 2019, [Online; navštívená 08.05.2019].
URL https://www.digitisation.eu/

[20] Jaderberg, M.; Simonyan, K.; Zisserman, A.; aj.: Spatial transformer networks. In *Advances in neural information processing systems*, 2015, s. 2017–2025.

[21] Kang, L.; Toledo, J. I.; Riba, P.; aj.: Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In *German Conference on Pattern Recognition*, Springer, 2018, s. 459–472.

[22] Kingma, D. P.; Ba, J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[24] Lavrenko, V.; Rath, T. M.; Manmatha, R.: Holistic word recognition for handwritten historical documents. In *First International Workshop on Document Image Analysis for Libraries, 2004. Proceedings.*, Jan 2004, s. 278–287.

[25] LeCun, Y.; Bengio, Y.; Hinton, G.: Deep learning. *Nature*, ročník 521, č. 7553, 5 2015: s. 436–444, ISSN 0028-0836.

[26] Lecun, Y.; Bottou, L.; Bengio, Y.; aj.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, ročník 86, č. 11, Nov 1998: s. 2278–2324, ISSN 0018-9219.

[27] Lee, J.: Tensorflow Implementation of Recurrent Neural Network (Vanilla, LSTM, GRU) for Text Classification. 2018, [Online; navštívená 08.05.2019].
URL https://github.com/roomylee/rnn-text-classification-tf

[28] Luong, M.; Pham, H.; Manning, C. D.: Effective Approaches to Attention-based Neural Machine Translation. *CoRR*, ročník abs/1508.04025, 2015.

[29] Lüdeling, A.; Odebrecht, C.; Perlitz, L.; aj.: RIDGES-Herbology (Version 8.0). Humboldt-Universität zu Berlin, 2018, [Online; navštívená 08.05.2019]. URL http://hdl.handle.net/11022/0000-0007-C6A3-1

[30] Marcos, J.-J.: Fonts for Latin Paleography. *Encyclopædia Britannica*, 2011. URL http://luc.devroye.org/JJMarcos-2008-FontsForLATIN_PALEOGRAPHY.pdf

[31] Nielsen, M. A.: *Neural networks and deep learning*, ročník 25. Determination press San Francisco, CA, USA:, 2015.

[32] Olah, C.: Understanding LSTM Networks. 2015, [Online; navštívená 08.05.2019]. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[33] Papadopoulos, C.; Pletschacher, S.; Clausner, C.; aj.: The IMPACT Dataset of Historical Document Images. In *Proceedings of the 2Nd International Workshop on Historical Document Imaging and Processing*, HIP '13, New York, NY, USA: ACM, 2013, ISBN 978-1-4503-2115-0, s. 123–130.

[34] Patel, C.; Patel, A.; Patel, D.: Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications*, ročník 55, 10 2012: s. 50–56.

[35] Pletschacher, S.; Antonacopoulos, A.: The PAGE (Page Analysis and Ground-Truth Elements) Format Framework. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR2010)*, IEEE-CS Press, August 2010, s. 257–260.

[36] P.S. Post Scriptum. Arquivo Digital de Escrita Quotidiana em Portugal e Espanha na Época Moderna. 2014, [Online; navštívená 08.05.2019]. URL http://ps.clul.ul.pt

[37] Puigcerver, J.: Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, ročník 1, IEEE, 2017, s. 67–72.

[38] Rabiner, L. R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, ročník 77, č. 2, Feb 1989: s. 257–286, ISSN 0018-9219.

[39] Robbins, H.; Monro, S.: A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, ročník 22, č. 3, 1951: s. 400–407, ISSN 00034851.

[40] Shi, B.; Bai, X.; Yao, C.: An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *CoRR*, ročník abs/1507.05717, 2015.

[41] Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[42] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. 2014.

[43] Sueiras, J.; Ruiz, V.; Sanchez, A.; aj.: Offline Continuous Handwriting Recognition Using Sequence to Sequence Neural Networks. *Neurocomputing*, ročník 289, 02 2018.

[44] Sutskever, I.; Vinyals, O.; Le, Q. V.: Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, editácia Z. Ghahramani; M. Welling; C. Cortes; N. D. Lawrence; K. Q. Weinberger, Curran Associates, Inc., 2014, s. 3104–3112.

[45] Vaamonde, G.; Costa, A. L.; Marquilhas, R.; aj.: Post Scriptum: Digital Archive of Everyday Writing. CLUL, University of Lisbon, [Online; navštívená 08.05.2019]. URL http://ps.clul.ul.pt/files/Papers-Congressos-PDFs/PostScriptumOslo(3).pdf

[46] Vaswani, A.; Shazeer, N.; Parmar, N.; aj.: Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, editácia I. Guyon; U. V. Luxburg; S. Bengio; H. Wallach; R. Fergus; S. Vishwanathan; R. Garnett, Curran Associates, Inc., 2017, s. 5998–6008.

[47] Weng, L.: Attention? Attention! 2018, [Online; navštívená 08.05.2019]. URL https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

[48] Wick, C.; Puppe, F.: Fully Convolutional Neural Networks for Page Segmentation of Historical Document Images. *CoRR*, ročník abs/1711.07695, 2017.

[49] Wikipedia, The Free Encyclopedia: Blackletter. 2019, [Online; navštívená 08.05.2019]. URL https://en.wikipedia.org/wiki/Blackletter

[50] Wikipedia, The Free Encyclopedia: Novověké latinské písmo. 2018, [Online; navštívená 08.05.2019]. URL https://cs.wikipedia.org/wiki/Novov%C4%9Bk%C3%A9_latinsk%C3%A9_p%C3%ADsmo

[51] Wikipedia, The Free Encyclopedia: Latinka. 2019, [Online; navštívená 08.05.2019]. URL https://cs.wikipedia.org/wiki/Novov%C4%9Bk%C3%A9_latinsk%C3%A9_p%C3%ADsmo

[52] Wikipedia, The Free Encyclopedia: Vladimir Levenshtein. 2018, [Online; navštívená 08.05.2019]. URL https://en.wikipedia.org/wiki/Vladimir_Levenshtein

Príloha A

Obsah přiloženého paměťového média

Obsahom priloženého pamäťového média je:

- text bakalárskej práce,
- zdrojové súbory v LaTeX pre vysádzanie textu bakalárskej práce,
- zdrojové súbory použité pri vytváraní dátovej sady,
- dátová sada,
- zdrojové súbory architektúry neurónovej siete,
- súbor *README.md* s popisom obsahu pamäťového média a spúšťania.