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

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MARTIN KIŠŠ**
AUTHOR

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

BRNO 2018

Zadání diplomové práce/20898/2017/xkissm00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Kišš Martin, Bc.**
Obor: Inteligentní systémy
Téma: **Rozpoznávání historických textů pomocí hlubokých neuronových sítí**
Convolutional Networks for Historic Text Recognition
Kategorie: Zpracování obrazu

Pokyny:

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:

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

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce je vytvořit nástroj pro automatický přepis textu historických dokumentů. Práce je zaměřena především na rozpoznávání textů pocházejících z období novověku psané písmem zvané Fraktura. Problém je řešen pomocí nově navržených rekurentních konvolučních neuronových sítí a také pomocí sítě zvané Spatial Transformer Network. Součástí řešení je také implementovaný generátor umělých historických textů. Pomocí tohoto generátoru je vytvořena umělá datová sada, na níž je natrénována konvoluční neuronová síť pro rozpoznávání řádků. Tato síť je následně otestována na reálných historických řádcích textu, na kterých natrénovaná sít dosahuje úspešnosti až 89.0 % znakové přesnosti. Přínosem této práce je především nově navržená neuronová síť pro rozpoznávání řádků textu a implementovaný generátor umělých historických textů, s jehož pomocí je možné natrénovat neuronovou síť tak, aby zvládala rozpoznávat reálné historické řádky textu.

Abstract

The aim of this work is to create a tool for automatic transcription of historical documents. The work is mainly focused on the recognition of texts from the period of modern times written using font Fraktur. The problem is solved with a newly designed recurrent convolutional neural networks and a Spatial Transformer Network. Part of the solution is also an implemented generator of artificial historical texts. Using this generator, an artificial data set is created on which the convolutional neural network for line recognition is trained. This network is then tested on real historical lines of text on which the network achieves up to 89.0 % of character accuracy. The contribution of this work is primarily the newly designed neural network for text line recognition and the implemented artificial text generator, with which it is possible to train the neural network to recognize real historical lines of text.

Klíčová slova

Přepis textu, OCR, historický text, rekurentní neuronové sítě, RNN, Spatial Transformer Network, STN, generátor umělé historické datové sady

Keywords

Text recognition, OCR, historical text, recurrent neural networks, RNN, Spatial Transformer Network, STN, artificial historical dataset generator

Citace

KIŠŠ, Martin. *Rozpoznávání historických textů pomocí hlubokých neuronových sítí*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Rozpoznávání historických textů pomocí hlubokých neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Kišš
23. května 2018

Poděkování

Chtěl bych poděkovat panu Ing. Michalu Hradišovi, Ph.D. za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat.

Obsah

Kapitola 1

Úvod

Automatický přepis textu (anglicky Optical Character Recognition, zkráceně OCR) je proces, ve kterém se převádí text z obrazové formy do sekvence znaků, které mohou být dále zpracovány. Vzhledem ke stále větší integraci počítačových systémů do běžného života, může tato úloha nacházet široké uplatnění. Například se může jednat o součást systému řízení autonomního vozidla, které díky rozpoznávání textů je schopné číst dopravní značení. Dále může být součástí aplikací zabývající se překladem z různých jazyků, kdy se překládaný text zadává pomocí kamery (například v chytrém mobilním telefonu). Využití je také možné při vyhledávání obrázků, kdy se díky rozpoznání textů může získat z obrázku více informací, a podobně.

V rámci počítačového vidění se jedná o problém, který je spojen s různými úskalími, která mohou převod z obrazové formy ztížit. Jedná se především o data (např. fotografie), která jsou zašuměná, v různých částech obrazu obsahují různou intenzitu světla, nebo například text může být natočen, zkreslen perspektivou a tak dále.

Cílem této práce je vytvořit nástroj, který je schopen rozpoznávat řádky textu z dokumentů, které vznikly v dřívějších dobách. Jedná se především o dokumenty z období novověku, které jsou psány, respektive tištěny, za použití novogotických písem. Takovýto převod historického dokumentu do textové podoby může především sloužit při digitalizaci dokumentů, které se často nacházejí v různých archívech, například v rámci knihoven. V takto přepsaném dokumentu je následně možné vyhledávat, případně může být vyhledán samotný dokument.

Za účelem rozpoznávání textu jsou použity konvoluční neuronové sítě doplněné o neuronovou síť zvanou Spatial Transformer Network a také o rekurentní vrstvy. Tyto použité metody jsou popsány v rámci kapitoly 2, kde je také stručně popsán vývoj historického písma a postupy, které se v současnosti používají k rozpoznávání textu. Kapitola 3 obsahuje popis zvolených metod, které budou v rámci práce implementovány a následně vyhodnoceny. Další kapitola se věnuje popisu datových sad, které jsou pro rozpoznávání historických textů vhodné. Kapitola 5 obsahuje popis implementace generátoru, jenž slouží ke generování umělých stránek historických textů. V další kapitole jsou popsány některé implementační detaily, které jsou součástí celkové implementace. V kapitole 7 jsou popsány a vyhodnoceny jednotlivé experimenty, které byly s natrénovanými neuronovými sítěmi provedeny.

Kapitola 2

Rozpoznávání textu

V rámci počítačového vidění může rozpoznávání textu zahrnovat různé analytické úlohy. Taková analýza může zahrnovat například detekci oblastí, v nichž se text nachází, extrakci řádků, slov, případně samotných znaků textu. Dále mohou být dokumenty analyzovány z pohledu vzniku dokumentu a podobně. Konvoluční neuronové sítě se začaly hojně využívat ke zpracování obrazových informací po úspěchu z roku 2012, kdy *Krizhevsky a spol.* [19] byli schopni natrénovat hlubokou konvoluční neuronovou síť, jejíž úspěšnost byla lepší, než ostatní dosud běžně používané metody pro klasifikaci obrázků.

Při použití konvoluční neuronové sítě dojde ke zpracování celého vstupního obrázku najednou a síť vyprodukuje pro tento obrázek jeden výstup. Tento přístup však není zcela vhodný pro zpracování sekvencí. Je to především z toho důvodu, že takovéto sekvence mohou nabývat různých délek a tedy by bylo vhodné, aby na toto neuronová síť dokázala reagovat a vyprodukovat výstup takové délky, která určitým způsobem odpovídá vstupním datům. Příkladem takovéto zpracovávané sekvence může být například řečový signál, který může pokaždé nabývat jiné délky a díky tomu také délka výstupu by měla být proměnlivá. Obdobně, také text, jenž je rozpoznáván, je sekvencí písmen. Zajistit proměnlivou délku výstupu pomocí konvoluční neuronové sítě by bylo velice obtížné. Dalším důvodem, proč tyto sítě nejsou vhodné ke zpracování sekvencí, je ten, že sekvence často obsahují závislosti, kdy informace například na konci sekvence může úzce souviset s informací poskytnutou na začátku sekvence. Z těchto důvodů se ke zpracování sekvencí používají takzvané rekurentní neuronové sítě (zkráceně *RNN*) [9]. Popisu tohoto typu sítí se věnuje část 2.3.

Jak již bylo zmíněno v úvodu, tato kapitola se věnuje přehledu současných řešení rozpoznávání textu. V první části této kapitoly je nastíněn vývoj písem, jež se používaly v období novověku. Následně jsou popsány metody, které se v současnosti používají k rozpoznávání textu. Tyto metody jsou založeny především na rekurentních neuronových sítích, které jsou popsány v další části této kapitoly. Spolu s rekurentními neuronovými sítěmi jsou také představeny speciální vrstvy, které se v nich využívají – *Long Short-Term Memory* (*LSTM*) [14] a *Gated Recurrent Unit* (*GRU*) [7]. Dále je v této kapitole popsána síť zvaná *Spatial Transformer Network* [15], která umožňuje aplikovat geometrické transformace na daný vstup (například obrázek). V poslední části této kapitoly jsou ukázány také dva nástroje, jež je možné použít při analyzování textových dokumentů, konkrétně se jedná o nástroje *Transkribus* a *OCRopus*.

2.1 Historická písma

Písmo, které je používané západní civilizací, tedy latinské písmo (latinka) [35], vzniklo přibližně v 7. století př. n. l odvozením z řeckého písma. Postupně vznikalo několik druhů tohoto písma, které se navzájem lišily zkosením, zdobností a podobně. Velká různorodost těchto písem však znesnadňovala jejich čtení. Díky tomu byla snaha písmo sjednotit, což vyústilo ve vytvoření několika standardních písem.

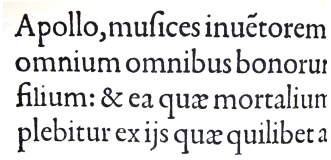
V rámci druhé poloviny 15. století se středověké latinské písmo rozdělilo do dvou hlavních směrů [17]. Jednalo se o písmo novogotické, které navazovalo na pozdější písma středověká (gotická), a humanistické, které vycházelo z antických a raně středověkých písem. Na počátku 16. století se v českých zemích ustálila zvyklost, podobně jako například v Německu, že latinské texty byly psány humanistickým písmem, kdežto texty české, případně německé, byly psány novogotickým písmem. Příkladem humanistického písma je písmo zvané *Antikva* [31]. Ukázku tohoto písma je možné vidět na obrázku 2.1a.

Novogotické písmo ve střední Evropě se dělí do tří kategorií. První kategorií je takzvaná *Fraktura* [32], německy *Fraktur*. Toto písmo vychází především ze středověkého písma nazývaného *Švabach* [37], německy *Schwabacher*. Švabach byl však vnímán jako široký a neúsporný druh písma, zatímco Fraktura byla užší a tedy zároveň úspornější a také působila světleji. Toto písmo se používalo výhradně pro tištěné dokumenty. Dalšími dvěma kategoriemi novogotického písma jsou *Kurent* [34] a *Kanzlei*. Zmíněné kategorie písma se vyznačují tím, že se jedná o kurzivní (Kurent) a polokurzivní (Kanzlei) písma. Tato písma se používala pro psané texty. Podstatný rozdíl mezi první kategorií (Fraktura) a druhými dvěma (Kurent, Kanzlei) je ten, že Fraktura představuje písmo psané několika tahy, čímž mohlo nabývat různých tvarů, kdežto Kurent a Kanzlei jsou psány rukou a tedy zde byla snaha o to, aby se písmena dala psát, pokud možno, jedním tahem. Ukázka písem Fraktura, Švabach a Kurent je vidět na obrázcích 2.1d, 2.1c a 2.1b.

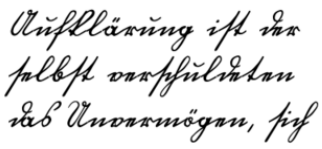
V průběhu novověku značně přibývá lidí, kteří dokáží číst a také psát [17]. Tím pádem dochází k tomu, že styl písma se může značně lišit mezi jednotlivými osobami. Vliv na počet osob, které dokáží číst a psát, měla bezpochyby také povinná školní docházka. V rámci výuky se užívalo především písmo Kurent. Novogotická písma mají také významný nacionalistický prvek, každý územní celek má totiž vlastní odnož novogotického písma. Na konci 18. a počátkem 19. století však novogotická písma postupně vytěsní písmo humanistické, které se začíná používat pro texty psané česky, případně německy. Definitivní konec používání novogotického písma se datuje do období druhé světové války, kdy jej, jako poslední, opouští nacistické Německo.

2.2 Současné metody založené na neuronových sítích

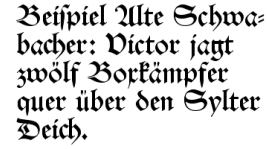
V současné době se vývoj v oblasti OCR soustředí především na rozpoznávání textu, který se nachází v nějakém prostředí (anglicky se jedná o takzvaný *scene text*), a na rozpoznávání ručně psaného textu. To, že se text nachází v libovolném prostředí, zvyšuje náročnost samotného rozpoznávání, protože výsledný systém by si měl poradit s jakýmkoliv okolím textu a vždy extrahovat informaci pouze o textu samotném, přičemž tento text může nabývat různých podob. Text ve scéně může mít různé styly písma, být zkreslený perspektivou, jiné objekty jej můžou z části překrývat. Dále může mít obrázek na různých místech různé intenzity světla, případně jednotlivé znaky vůbec nemusí být v jedné linii, ale mohou opisovat různé křivky.



(a) Antikva. Převzato z [31], upraveno.



(b) Kurent. Převzato z [34], upraveno.



(c) Švabach. Převzato z [37].

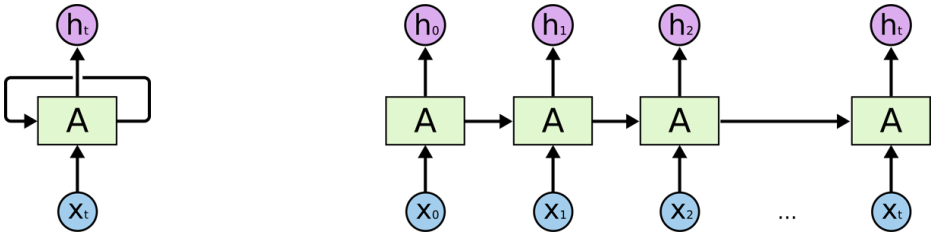
(d) Fraktura. Převzato z [32].

Obrázek 2.1: Ukázka humanistického a novogotických písem. Na obrázku 2.1a se nachází ukázka humanistického písma, takzvaná Antikva. Obrázek 2.1b zobrazuje typ kurzivního novogotického písma zvaného Kurent. Obrázek 2.1c zobrazuje typ převážně středověkého písma nazývaného Švabach (též Schwabacher), na obrázku 2.1d je potom zobrazena ukázka novogotického písma zvaného Fraktura.

Při rozpoznávání ručně psaného textu příliš nedochází k problémům, které jsou typické pro text ve scéně. Je to především z toho důvodu, že tyto texty bývají naskenovány, případně vyfotografovány, tak, aby byly co nejméně deformovány. U ručně psaného textu se však vyskytují jiné komplikace. Pravděpodobně největším problémem je velká variabilita tvaru písmen, a to jak napříč autory těchto textů, tak i v rámci textů jednoho autora. Autor rozpoznávaného textu totiž může stejný znak v rámci jedné sekvence použít několikrát, avšak pokaždé má tento znak trochu odlišný tvar. Nicméně, rozpoznávací systém by si s tímto měl poradit a všechny znaky správně rozpoznat. I přes některé odlišnosti mezi rozpoznáváním textu ve scéně a ručně psaného textu, bývají na oba problémy používány stejné techniky.

Jedním z možných přístupů k rozpoznávání slov ve scéně, je založen na použití histogramu gradientů (anglicky *Histogram of Gradients*, zkráceně *HOG*) [30]. Metoda spočívá ve výpočtu gradientů ve všech výřezech o rozměru 3×3 pixely v původním obrázku. Z těchto vypočtených gradientů se následně vytvoří histogram pro každou takovou oblast, čímž vznikne mapa histogramů gradientů, která reprezentuje původní vstupní obrázek. Následně je vypočítán průměr všech histogramů, jež se nacházejí ve stejném sloupci. Po tomto kroku vznikne vektor gradientů reprezentující jednotlivé sloupce a tento vektor je použit pro rozpoznání daného slova. Pro samotné rozpoznávání slov se zde používá rekurentní neuronová síť s použitím obousměrných Long Short-Term Memory (Bidirectional Long Short-Term Memory). Tato síť určuje pro každou hodnotu vektor pravděpodobnost každého možného znaku.

Pro získání výsledného textu je použita technika *Connectionist Temporal Classification* (zkráceně *CTC*) [12]. CTC je hodnotící funkce rekurentních neuronových sítí, která vyhodnocuje nezarovnané výstupy s proměnlivým časováním. Toto vyhodnocení může probíhat ve dvou variantách. První možností, jež byla zároveň v tomto případě použita, je, že CTC v každém momentě, kdy rekurentní neuronová síť vyprodukovala výstup pravděpodobností znaků, vybere ten s nejvyšší pravděpodobností. Druhou možností je, že se vyhodnotí pravděpodobnosti vůči dodanému slovníku. Varianta se slovníkem je nicméně časově náročnější, protože při vyhodnocení se musí vyhodnotit velký počet pravděpodobností [27].



(a) (b)

Obrázek 2.2: Schéma rekurentní vrstvy v rekurentní neuronové síti. Na obrázku 2.2a je znázorněna smyčka v rámci jedné vrstvy. Schéma sítě se však také může zobrazovat v takzvané rozvinuté (unrolled) podobě tak, jako je tomu na obrázku 2.2b. Převzato z [21], upraveno.

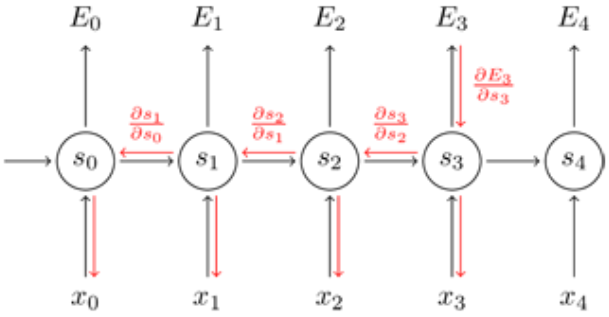
Podobný princip pro rozpoznávání textu, jako výše zmíněný, použili také *Shi a spol.* [27]. V této metodě je však namísto výpočtu histogramu gradientů využito konvolučních vrstev, které získají z obrázku požadované příznaky. Také v tomto případě jsou výsledné pravděpodobnosti zpracovány metodou CTC, avšak zde jsou použity obě možnosti této techniky a jsou navzájem porovnány. Z dosažených výsledků je patrné, že varianta se slovníkem dosahuje vyšších úspěšností.

Jak již bylo zmíněno výše, při rozpoznávání textu ve scéně může být text různě deformován. Některé z uvedených problémů byly, že jednotlivé znaky textu nemusí spočívat v jedné linii, případně může být text zkreslen perspektivou a podobně. K řešení tohoto problému využili *Shi a spol.* ve své další práci [28] speciální neuronovou síť zvanou *Spatial Transformer Network*, jejíž princip je detailněji popsán v části 2.4. Tato síť umožňuje aplikovat na vstupní obrázek geometrické transformace (například posunutí, rotaci a podobně), jež se síť při trénování naučí tak, aby po její aplikaci výstupní obrázek co nejlépe odpovídal požadovaným vlastnostem.

Pro rozpoznávání ručně psaného textu použili *Pham a spol.* [23] rekurentní neuronovou síť s vícerozměrnými LSTM vrstvami obohacenou o konvoluční a dropout vrstvy. Přidáním dropout vrstev bylo dosaženo nižší hodnoty *CER* (*Character Error Rate*) v průměru o 10–20 %, než když tyto vrstvy použity nebyly. V další práci zabývající se rozpoznáváním ručně psaného textu použil *J. Puigcerver* [24] obdobnou rekurentní neuronovou síť a zároveň porovnal úspěšnosti sítí s jednorozměrnými LSTM vrstvami a vícerozměrnými. Z dosažených výsledků vyplývá, že úspěšnost rozpoznávání mají obě sítě přibližně stejnou.

V práci [5] použili *Breuel a spol.* rekurentní neuronovou síť pro rozpoznávání anglických tištěných dokumentů psaných latinkou a také pro rozpoznávání německy psaných novověkých textů, které byly tištěny pomocí písma Fraktur. Tato síť obsahovala LSTM vrstvy a byla trénována s využitím CTC.

Další možností pro rozpoznávání textu je metoda, kterou použili *Springmann a spol.* [29], kteří použili pouze rekurentní neuronovou síť bez použití konvolučních nebo jiných vrstev zpracovávajících obrazové informace. Vstupem této sítě jsou totiž jednotlivé sloupce vstupního obrázku, které jsou sítí zpracovány. Výstupem sítě jsou pak pravděpodobnosti jednotlivých znaků pro daný sloupec pixelů. Tato metoda byla použita na řádky historických textů.



Obrázek 2.3: Schéma propagování chyby v rámci rekurentní neuronové sítě. Převzato z [6].

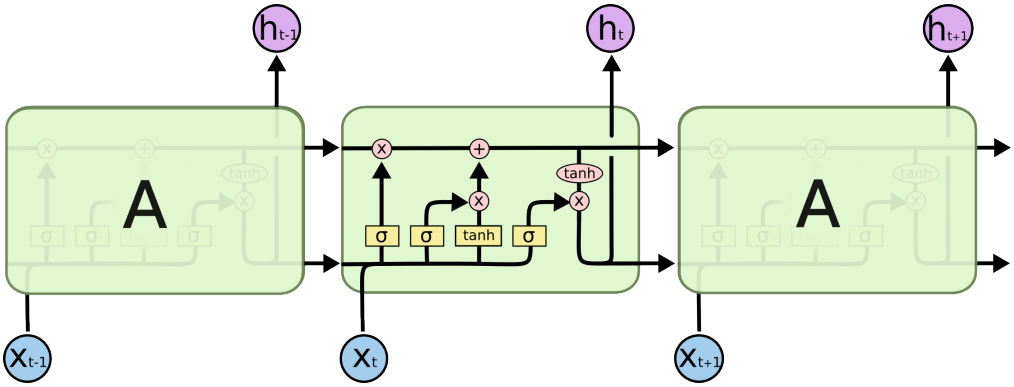
2.3 Rekurentní neuronové sítě

Rekurentní neuronové sítě jsou neuronové sítě, které se využívají pro zpracování dat sekvenčního charakteru [9]. Takovéto sítě využívají takzvaných rekurentních vrstev, které si uchovávají vnitřní skrytý stav. Tento skrytý stav, jenž je získán v čase t, je použit jako součást vstupu téže vrstvy v čase $t+1$. Z tohoto pohledu je možné se na tuto vrstvu dívat jako na jakousi paměť sítě, což se právě využívá při zpracování sekvenčních dat. Schematicky se taková vrstva kreslí tak, jak je zobrazeno na obrázku 2.2. Formálně zapsáno, je rekurentní skrytý stav h_t, pro danou sekvenci $x = (x_1, x_2, \ldots, x_T)$, následující:

$$h_t = \begin{cases} 0, & \text{pro } t = 0 \\ \phi(h_{t-1}, x_t) & jinak \end{cases} \tag{2.1}$$

Trénování rekurentních neuronových sítí probíhá pomocí algoritmu *Backpropagation through time* (zkráceně *BPTT*) [22]. Tento algoritmus vychází z algoritmu *Backpropagation*, avšak s tím rozdílem, že chybu je nutné propagovat z aktuálního kroku t zpět až ke kroku 0. Schéma propagování chyby v rekurentní neuronové síti je zobrazeno na obrázku 2.3. Při použití plně propojených vrstev se sigmoidální aktivační funkcí, případně také s hyperbolickým tangentem, které jsou běžně používány v neuronových sítích, však může docházet ke dvěma druhům problému. Prvním je situace, kdy dochází k velkému nárůstu gradientů, čímž dojde k tomu, že se síť učí především vzdálené závislosti a blízké závislosti jsou potlačeny. Tento problém se nazývá *exploding gradients*. Druhá situace může být opačná, tedy že hodnoty gradientů se blíží k nule a tím dochází k tomu, že síť se učí především závislosti blízko u sebe, ale vzdálenější závislosti se neučí. Tento problém se nazývá *vanishing gradients* [22].

Aby nedocházelo k problémům s gradienty, byly vytvořeny nové bloky, pomocí nichž se vytváří vrstvy, které se používají v rámci rekurentních neuronových sítí k uchovávání informací. První typ takového bloku představili v roce 1997 *Hochreiter a Schmidhuber* [14], který se nazývá *Long Short-Term Memory* (*LSTM*). Druhý typ bloku nazvaný *Gated recurrent unit* (*GRU*) představili *Cho a spol.* v roce 2014 [7]. Oba tyto bloky jsou popsány v rámci následujících dvou částí. Podle článku *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* [9] však není výrazný rozdíl ve výsledcích při použití vrstev s jedním nebo druhým typem bloků.



Obrázek 2.4: Obecné schéma LSTM. Převzato z [21].

Long Short-Term Memory

Long Short-Term Memory [14], zkráceně *LSTM*, je blok používaný v rekurentních neuronových sítích k ukládání informace. Tento blok se skládá celkem ze tří částí zvaných *gate* – *input gate*, *output gate* a *forget gate*. Dále jsou zde také tři hodnoty, se kterými tento blok pracuje – *vstup*, *výstup* (v následujícím kroku je tato hodnota považována za skrytý stav) a *zapamatovaná hodnota*. Schéma LSTM bloku je znázorněno na obrázku 2.4.

Forget gate má za úkol určitým způsobem zapomínat informace z dřívějších kroků sítě. Tento prvek bere v potaz stav (neboli výstup z předchozího kroku) a vstupní hodnotu. Tyto hodnoty jsou vynásobeny naučenou maticí W_f, k výsledku je přičten bias b_f a následně je tento výsledek použit jako vstup do sigmoidální funkce. Výstup sigmoidy f_t, pro nějž platí $0 < f_t < 1$, určuje míru zapomenutí předchozí zapamatované hodnoty. Pokud se hodnota blíží 1, pak předchozí zapamatovaná hodnota je zapomenuta minimálně. Pokud však se hodnota f_t blíží k 0, pak je většina zapomenuta. Následně je zapamatovaná hodnota C_{t-1} vynásobena hodnou f_t. Formálně je výpočet hodnoty f_t následující

$$f_t = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \tag{2.2}$$

V prvku *input gate* se vypočítá hodnota i_t obdobně, jako hodnota f_t ve *forget gate*, jen je zde použita matice W_i a bias b_i.

$$i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \tag{2.3}$$

Dále se také vypočítá hodnota $\tilde{C}_t$, která představuje novou kandidátní hodnotu k zapamatování. Hodnoty $\tilde{C}_t$ se díky použití hyperbolického tangentu pohybují v intervalu od -1 do $+1$. Výpočet $\tilde{C}_t$ je následující

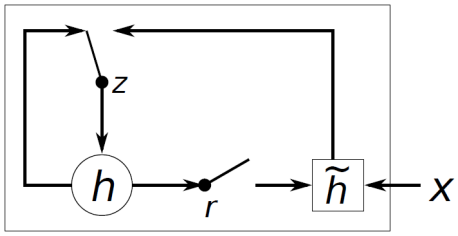
$$\tilde{C}_t = tanh(W_C \cdot [x_t, h_{t-1}] + b_C) \tag{2.4}$$

Následně se hodnota i_t vynásobí s kandidátní hodnotou $\tilde{C}_t$ a tato výsledná hodnota je přičtena k hodnotě, který vznikla aplikací *forget gate*. Tímto vznikne nová hodnota C_t.

Nakonec je aplikován prvek *output gate*, který určuje hodnotu nového vnitřního stavu a tedy také aktuálního výstupu h_t. Nejprve se však vypočte hodnota o_t, jež má obdobný výpočet jako hodnoty f_t a i_t. Výstup h_t se nakonec vypočítá následovně

$$o_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \tag{2.5}$$

$$h_t = o_t * tanh(C_t) \tag{2.6}$$



Obrázek 2.5: Obecné schéma GRU. Převzato z [7].

Gated Recurrent Unit

Gated recurrent unit [7], zkráceně *GRU*, je novější variantou bloku, jenž se používá pro ukládání informací v rekurentních neuronových sítích. Na rozdíl od LSTM obsahuje GRU pouze dvě části, jež se shodně jako v LSTM nazývají *gate*, – *reset gate* a *update gate* [7]. Druhým rozdílem oproti LSTM je, že neuvažuje žádnou zapamatovanou hodnotu, ale pracuje pouze s *vnitřním stavem* (zároveň výstup GRU) a *vstupem*. Schéma GRU bloku je znázorněno na obrázku 2.5.

První v pořadí se aplikuje prvek *reset gate*. Tento prvek rozhoduje o tom, jak zkombinovat předchozí vnitřní stav h_{t-1} s aktuálním vstupem x_t. Výpočet hodnoty r je následující

$$r = \sigma([W_r x] + [U_r h_{t-1}]) \tag{2.7}$$

Následně je vypočtena hodnota z, která je výstupem *update gate*. Tato hodnota je vypočtena obdobně jako r

$$z = \sigma([W_z x] + [U_z h_{t-1}]) \tag{2.8}$$

Výsledný výstup h_t je pak vypočten následovně

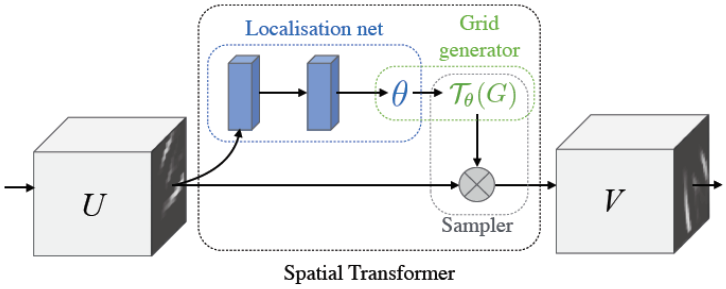
$$h_t = z h_{t-1} + (1 - z)\tilde{h}_t \tag{2.9}$$

$$\tilde{h}_t = tanh([W x] + [U(r \circ h_{t-1})]) \tag{2.10}$$

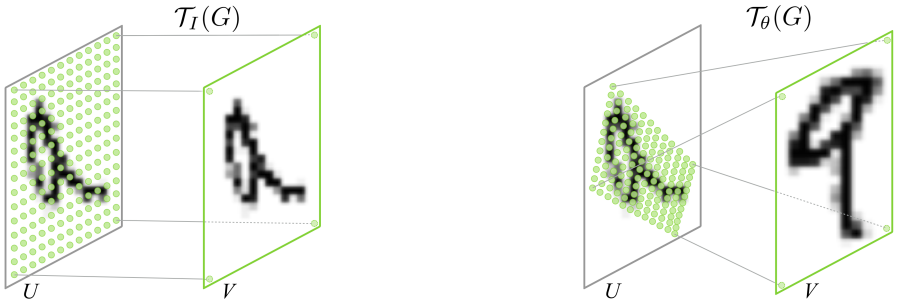
2.4 Spatial Transformer Network

Spatial Transformer Network [15] (zkráceně *STN*) je neuronová síť, která se používá k aplikování geometrických transformací na daný vstup. Při zpracování obrazu tímto vstupem může být buďto obrázek samotný, nebo může být transformace aplikována na aktivační mapy konvolučních vrstev. Tato síť tedy umožňuje upravovat vstup pomocí transformací posunutí, změny měřítka, rotace a dalších. Celkově se celý mechanismus skládá ze tří částí – *lokalizační sítě*, *generátoru matice* a *vzorkovače*. Schéma STN je vyobrazeno na obrázku 2.6.

Lokalizační síť [15] je neuronová síť, jejímž vstupem je matice $U \in \mathbb{R}^{H \times W \times C}$, kde H je počet řádků matice, W je počet sloupců matice a C je počet kanálů matice, na kterou je aplikována transformace. Architektura dané sítě může být libovolná, může obsahovat plně propojené vrstvy, konvoluční vrstvy a podobně. Výstupní vrstva však musí být trénována jako regresní. Výstupem této sítě jsou parametry θ, které určují vlastnosti použité transformace. Z těchto parametrů je dále vytvořena matice $\mathcal{T}_\theta$, která definuje výslednou transformaci aplikovanou na vstupní matici.



Obrázek 2.6: Obecné schéma Spatial Transformer Network. Převzato z [15].



(a) (b)

Obrázek 2.7: Ukázka aplikace vygenerované matice na vstup U produkující výstup V. Obrázek 2.7a zobrazuje situaci, kdy matice $\mathcal{T}_I$ reprezentuje identitu. V takovém případě je vstup i výstup shodný. Na obrázku 2.7b je zobrazeno použití matice $\mathcal{T}_\theta$, která reprezentuje 2D afinní transformaci. Převzato z [15].

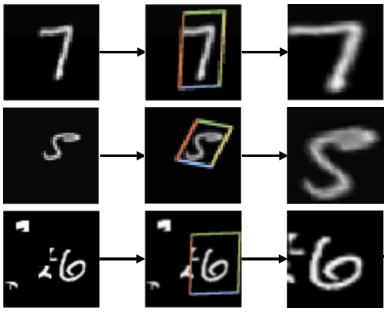
Vypočítaná matice reprezentující prováděnou transformaci mapuje pomyslnou matici bodů ze vstupní matice (obrázku) na pravidelnou výstupní matici (viz obrázek 2.7). Pakliže je prováděná transformace, kterou reprezentuje matice $\mathcal{T}_\theta$, afinní transformace pracující se 2D vstupem, pak je možné celý proces zapsat pomocí rovnice

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{2.11}$$

kde x_i^s a y_i^s jsou *x-ová*, respektive *y-ová*, souřadnice bodu v původní matici, G_i je výsledná pravidelná matice výstupních bodů (x_i^t, y_i^t) a A_θ je 2D afinní transformace reprezentovaná maticí hodnot θ_{11} až θ_{23}. Vzhledem k tomu, že byla uvažována 2D afinní transformace, je výstupem lokalizační neuronové sítě 6 hodnot, které tuto transformaci vytvářejí. Je však možné použít složitější transformace a výstupem sítě je pak více hodnot. Transformace nemusí pracovat pouze s 2D vstupem, ale vstup může být také například třídimenzionální.

Výstup lokalizační sítě však může být také menší, což omezí počet použitých transformací. Například mohou být použity pouze čtyři parametry, které pak mohou definovat transformační matici, jenž popisuje transformaci skládající se pouze z posunutí, změny měřítka a oříznutí. Výsledná matice by pak měla následující podobu

$$A_\theta = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \end{bmatrix} \tag{2.12}$$



Obrázek 2.8: Ukázka zpracování obrázku pomocí STN. První sloupec obrázků zobrazuje vstupní obrázky U. Prostřední sloupec obsahuje vstupní obrázky s vizualizací vygenerované 2D afinní transformace (pomocí barevného čtyřúhelníku). Poslední sloupec zobrazuje vstupní obrázky po provedení transformace (V). Převzato z [15], upraveno.

kde s_x určuje změnu měřítka v ose x, s_y určuje změnu měřítka v ose y, t_x určuje posunutí ve směru osy x a t_y určuje posunutí v ose y. Za předpokladu, že změna měřítka je v obou osách stejná (změna měřítka je izotropní), pak hodnota s_x a s_y je shodná a tudíž výstupem sítě mohou být pouze tři parametry transformace.

V poslední fázi dochází ke vzorkování z původní matice (obrázku) za pomocí vypočtené transformace. Protože při předchozích výpočtech může snadno dojít k práci s desetinnými čísly, je vhodné při vzorkování použít interpolaci, například bilineární, díky níž je možné síť trénovat, protože lze vypočítat derivace potřebné k aplikaci algoritmu backpropagation. Výsledkem celého procesu pak je nová matice (obrázek).

Spatial Transformer Network nemusí sloužit pouze jako samostatná síť, která dokáže transformovat vstupní matici na výstupní, ale může být také součástí hlubší konvoluční neuronové sítě. Použita může být na jakémkoli místě a v libovolném počtu. Pokud je STN použita v rámci jiné konvoluční neuronové sítě, pak se tato síť může naučit, jak transformovat vstupní data za účelem zvýšení celkové úspěšnosti sítě. Výstup lokalizační sítě STN také může být použit při dopředném průchodu sítí tak, že jsou tyto parametry poskytnuty jako vstup jiné vrstvy a ta tak může získat dodatečné informace ohledně pozice hledaného objektu. [15]

2.5 Levenštejnova vzdálenost

Levenštejnova vzdálenost (také *editační vzdálenost*, anglicky *Levenshtein distance* nebo *edit distance*) [25] je metrika pro porovnání dvou seznamů, pro jejichž dva prvky je možné rozhodnout, zda jsou stejné, či nikoliv. Je na nich tedy definována relace ekvivalence. Při výpočtu vzdálenosti dvou seznamů je výsledkem skalární hodnota reprezentující nejmenší počet nutných úprav zdrojového seznamu tak, aby vznikl cílový seznam. Za transformace nad seznamy jsou považovány operace *vložení*, *odebrání* a *změna prvku* s tím, že pro každou operaci může být určena jiná cena za její provedení.

Výpočet Levenštejnovy vzdálenosti může být rekurzivně vyjádřen následovně:

$$
\begin{aligned}
D(0,0) &= 0 \\
D(i,0) &= D(i-1,0) + w_d \\
D(0,j) &= D(0,j-1) + w_i \\
D(i,j) &= min \begin{cases} D(i-1,j) + w_d \\ D(i,j-1) + w_i \\ D(i-1,j-1) + \begin{cases} w_r & \text{pokud } x(i) \neq y(j) \\ 0 & \text{jinak} \end{cases} \end{cases}
\end{aligned} \tag{2.13}
$$

kde i a j představují indexy v rámci seznamů x a y, w_i, w_d a w_r představují ceny operací vložení, odebrání a změny prvku. Celková vzdálenost dvou seznamů x a y je získána po vypočtení hodnoty $D(|x|, |y|)$ [16, 13, 36].

Z důvodu efektivity se však namísto rekurze využívá dvourozměrná matice, do které se již vypočtené hodnoty zapisují. Výhoda takovéhoto zápisu pak tkví především v možnosti získat zarovnání obou seznamů tak, aby počet provedených transformací odpovídal vypočtené vzdálenosti. Ono zarovnání je vypočteno tak, že se v rámci matice hledá optimální cesta od hodnoty $D(0,0)$ k hodnotě $D(|x|, |y|)$. K získání oné optimální cesty se využívají techniky dynamického programování. Ukázka zarovnání dvou textových řetězců je na obrázku 2.9.

Jak je zmíněno na začátku této části, výpočet Levenštejnovy vzdálenosti je možné provést nad libovolnými daty, které lze testovat na rovnost. Nemusí tedy nutně probíhat pouze na textových řetězcích, ale využívá se například také v oblastech bioinformatiky k zarovnání DNA sekvencí a podobně. [16].

2.6 Existující nástroje

Transkribus *Transkribus*[1] je nástroj s otevřeným zdrojovým kódem, který umožňuje analyzovat textové dokumenty. Pomocí integrovaných nástrojů dokáže v dokumentu, jenž je ve formě obrázku, detekovat oblasti s textem a v nich následně je schopen detekovat jednotlivé řádky, případně také umí detekovat základní dotažnici v daném řádku. Pomocí zvoleného jazykového modelu také Transkribus umožňuje vytvořit automatický přespis detekovaného textu. Při zkoušení tohoto nástroje však byl automatický přepis značně nepřesný, viz obrázek 2.10. Nevýhodou tohoto nástroje může být, že analýzu dokumentů lze spustit pouze na dokumentech (obrázcích), které byly nahrány na server (cloud). Bez internetového připojení tedy není možné tento nástroj efektivně využívat.

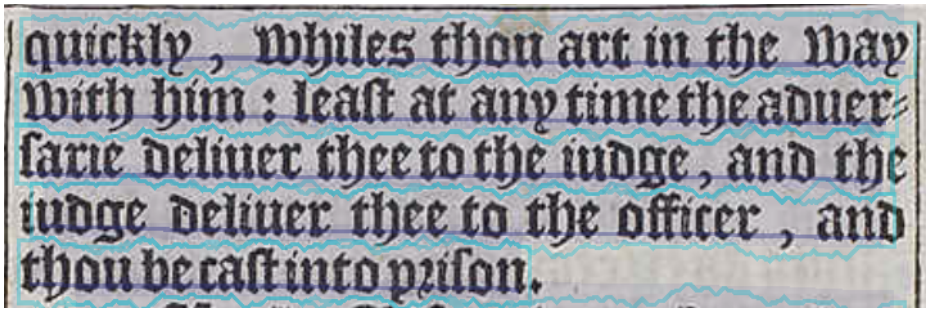
OCRopus *OCRopus*[2] je, obdobně jako předchozí nástroj Transkribus, volně dostupný nástroj pro analýzu textových dokumentů. Také tento nástroj umožňuje zpracovat vstupní obrázek, jenž reprezentuje zpracovávaný dokument, tak, že výstupem je například kolekce vyextrahovaných řádků ze vstupního souboru. Tato analýza je prováděna pomocí neuronových sítí, které jsou volně k dispozici. Oproti nástroji Transkribus je tedy možné zpracovávat dokumenty i bez nutnosti internetového připojení. Ukázku fungování nástroje OCRopus je možné vidět na obrázku 2.11.

[1] https://transkribus.eu/Transkribus/
[2] https://github.com/tmbdev/ocropy

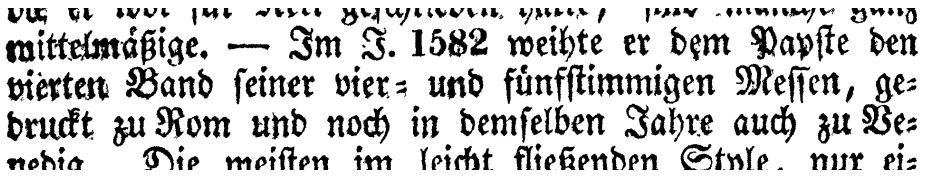
	#	G	U	M	B	O
#	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

G U M B O *
| | | | | |
G A M B O L

Obrázek 2.9: Ukázka výpočtu Levenštejnovy vzdálenosti dvou řetězců („GUMBO“ a „GAMBOL“) a jejich následné zarovnání. V tabulce jsou vyobrazeny všechny vypočtené vzdálenosti jednotlivých kombinací řetězců, přičemž celková vzdálenost řetězců „GUMBO“ a „GAMBOL“ se nachází v pravém dolním rohu, tedy 2. To odpovídá jedné změně písmena „U“ za „A“ a přidání koncového písmene „L“. V tabulce je také vidět optimální cesta, která je podbarvená. Význam jednotlivých polí cesty je následující. Pokud vede cesta diagonálně a zároveň se nemění hodnota polí, pak jsou znaky stejné, pokud je však hodnota rozdílná, jedná se změnu znaku. Pokud vede směr cesty vertikálně, pak je vložen do řetězce, který je zapsán na svislém záhlaví tabulky. Naopak, pokud je směr cesty vodorovná, znamená to přidání znaku k řetězci, jenž je zapsán na vodorovném záhlaví tabulky. Význam operace přidání znaku k prvnímu řetězci je stejný, jako odebrání znaku z druhého řetězce, a naopak. Pod tabulkou je pak možné vidět výsledné zarovnání obou řetězců. Znak „*“ označuje prázdný znak. Příklad byl převzat ze článku [13].



Obrázek 2.10: Ukázka zpracování dokumentu pomocí nástroje Transkribus. Světle modrou barvou je vyznačeno nalezené ohraničení řádku. Fialovou barvou je pak vyznačena základní dotažnice (*baseline*). Transkribus při přepisu posledního řádku tohoto odstavce vyprodukoval výstup „con beraamioton“, zatímco správný přespis by měl být „thou be cast into prison.“.


mittelmäßige. — Im J. 1582 weihte er dem Papſte den vierten Band ſeiner vier- und fünfſtimmigen Meſſen, gedruckt zu Rom und noch in demſelben Jahre auch zu Ve-

(a)


vierten Band ſeiner vier- und fünfſtimmigen Meſſen, ge-

(b)

Obrázek 2.11: Ukázka použití nástroje OCRopus. Obrázek 2.11a zobrazuje vstupní obrázek obsahující text. Na obrázku 2.11b je vyobrazen výřez nalezeného řádku.

Kapitola 3

Návrh řešení

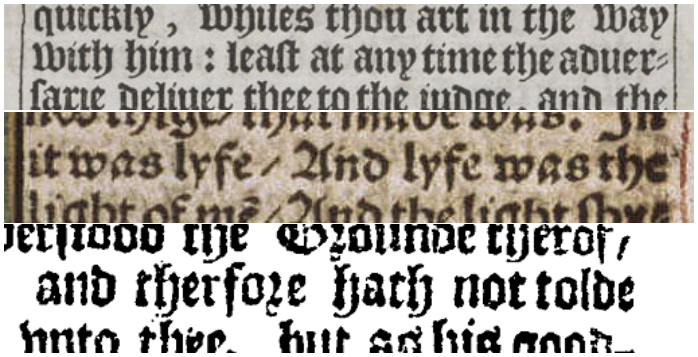
V rámci této kapitoly jsou nejprve možnosti řešení rozpoznávání historických textů. První část se věnuje popisu dat, jež by měla sloužit jako vstup do výsledného systému. V další podkapitole jsou potom představeny jednotlivé návrhy, které jsou postaveny na použití hlubokých neuronových sítí. Poté je detailně rozebráno řešení, jenž je v rámci této práce použito.

Vstupem do všech následně popsaných možností řešení je obrázek reprezentující jeden řádek historického textu. Ideálně by takovýto obrázek měl obsahovat pouze text onoho jednoho řádku, neměl by tedy obsahovat další části textu, obrázky a podobné objekty, které by vedly ke zhoršení rozpoznávání požadovaného textu. Ukázka takovýchto řádků je vyobrazena na obrázku 3.1.

Protože jsem při hledání vhodných datasetů nenalezl žádný takový, který by obsahoval extrahované řádky u nichž by byl přiložen také přepis a také pozice jednotlivých písmen, rozhodl jsem se v rámci této práce implementovat také generátor umělých historických textů. Tento generátor je detailně popsán v kapitole 5. Výhoda takovéhoto generátoru tkví ve dvou vlastnostech. První vlastností je, že pomocí generátoru je možné si vytvořit velmi velké množství vlastních dat, která mohou být použita při trénování neuronové sítě. Druhou výhodou pak je možnost vytvoření podrobných anotací. Vedle písmen, které se v daném textu nacházejí, je také možné získat pozice oněch písmen v rámci výsledného obrázku. Nevýhodou generátoru pak je, že proces generování musí být vhodně nastaven tak, aby výsledné obrázky co nejvíce odpovídaly reálným historickým výřezům.

3.1 Návrhy možných řešení

Klasifikace slov První možností při rozpoznávání historických textů je použití klasifikační konvoluční neuronové sítě, která by na vstupu měla výřez jednoho konkrétního slova z původního řádku. Takováto síť by vycházela z klasifikační sítě uveřejněné v článku *ImageNet Classification with Deep Convolutional Neural Networks* [19]. Na začátku sítě by tedy byly konvoluční a max-pooling vrstvy a následovaly by plně propojené vrstvy s dropout vrstvami. Výstupem by byl vektor hodnot, které by představovaly pravděpodobnosti jednotlivých možných slov. Velkou nevýhodou takovéto sítě by bylo omezení na počet rozpoznávaných slov. Vyžadovalo by to seznam rozpoznávaných slov, což pro jazyk, jako je čeština, může znamenat velice rozsáhlou databázi slov. Je to především z důvodů jako jsou skloňování, případně časování, různé změny předpon a přípon slov a podobně. Také sa-



Obrázek 3.1: Ukázky vstupních řádků historických textů.

motná čeština procházela určitým vývojem a tak se v různých obdobích mohou vyskytovat různé varianty téhož slova.

Connectionist Temporal Classification Další možností, jak rozpoznávat historické texty, je použít techniku CTC [12] obdobně, jako tomu bylo v rámci prací představených v kapitole obsahující současné metody k rozpoznávání textu (2.2). Vstupem takovéto konvoluční rekurentní neuronové sítě by byla sekvence výřezů původního obrázku, přičemž tyto výřezy by byly extrahovány z původního obrázku s konstantním krokem. Výstupem by pak byly pravděpodobnosti všech znaků pro každý výřez. Celkový přepis by pak byl určen na základě použité varianty CTC (je možné použít se slovníkem nebo bez slovníku). Tuto možnost však nepreferuji, protože při použití varianty bez slovníku je dosahováno značně nižších přesností přepisu textu, než při variantě se slovníkem [27]. Varianta se slovníkem zase trpí stejnými problémy, jako při použití klasifikační sítě popsané výše, tedy především omezenou velikostí slovníku.

Poziční a znaková síť Následující možností je použití dvou oddělených neuronových sítí. První neuronová síť by měla za úkol určovat pozice jednotlivých písmen v textu (*poziční síť*) a druhá neuronová síť by je následně rozpoznávala (*znaková síť*). Přepis vstupního obrázku, reprezentujícího rozpoznávaný řádek historického textu, by pak byl následující. Nejprve by bylo rozpoznán jeden znak, který se nachází v aktuálním výřezu, a následně by byla predikována vzdálenost k dalšímu znaku.

Výstup poziční sítě může být ve dvou podobách. První možností je, že výstupem bude jedna hodnota, která bude představovat onu vzdálenost a síť tak bude trénována jako regresní. Druhou možností je trénovat tuto síť jako klasifikační, kdy každá hodnota ve výstupním vektoru představuje pravděpodobnost konkrétní vzdálenosti.

Výstupem znakové sítě by měl být vektor reprezentující pravděpodobnosti možných znaků v daném výřezu, přičemž tato informace může být také propagována do poziční sítě. Tohle propojení by mělo zlepšit predikci vzdálenosti k dalšímu znaku v případech, kdy jsou středy dvou písmen blízko u sebe. S informací ohledně aktuálně rozpoznaného písmene by pak mohla poziční síť přesněji určit vzdálenost k dalšímu znaku.

Spojení neuronových sítí Poslední navrženou možností pro rozpoznávání historických textů v sobě spojuje obě neuronové sítě popsané v předchozím odstavci. Jedná se tedy o neuronovou síť, jejíž architektura obsahuje dvě výpočetní větve. Jedna větev by klasifikovala znak nacházející se nejblíže středu daného výřezu a druhá větev by následně určovala posun

Obrázek 3.2: Ukázka práce navržené neuronové sítě. Vstupem do sítě je výřez původního řádku historického textu. Pro tento výřez jsou vypočteny dva výstupy. První výstup je vektor udávající pravděpodobnosti ohledně klasifikace prostředního znaku ve výřezu. Druhý výstup určuje relativní posun vůči aktuálnímu výřezu tak, aby v dalším výřezu byl uprostřed následující znak.

k dalšímu písmenu. V tomto případě by také mohlo být prvních několik konvolučních a max-pooling vrstev sdíleno, což by mělo mít příznivý dopad na rychlost výpočtů a počet parametrů této sítě oproti dvěma samostatným sítím.

V rámci posledních dvou návrzích mohou být použity také techniky, jež byly popsány v rámci předchozích kapitolách. Jedná se především o použití Spatial Transformer Network [15] (kapitola 2.4) a rekurentních vrstev [14, 7] (kapitola 2.3). V případě použití rekurentních vrstev, by tyto vrstvy nahradily ty plně propojené. S použitím takovýchto vrstev by měla síť dosahovat lepších výsledků, protože s rekurentními vrstvami by při rozpoznávání získala paměť, respektive kontext, a mohla by tedy kvalifikovaněji určovat jednotlivé znaky. Tyto vrstvy by představovaly určitý jazykový model, který by se síť naučila ze vstupních dat. Již by tedy bylo nutné, aby byla síť trénována a testována na datové sadě stejného jazyka. Tohle v případě nerekurentních sítí není třeba, protože při klasifikaci písmene se vychází pouze z daného výřezu bez jakéhokoliv kontextu.

3.2 Implementované řešení

Řešení, jenž bylo implementováno v této práci vychází ze dvou posledních návrhů v předchozí části. Jedná se tedy o neuronovou síť, která se skládá ze dvou větví, *poziční* a *znakové*. Celá síť vychází z architektury sítě uvedené v článku *ImageNet Classification with Deep Convolutional Neural Networks* [19], která se používá pro klasifikační konvoluční neuronové sítě.

Znaková větev má za úkol klasifikovat znak, jenž se nachází nejblíže středu daného výřezu. Výstupem by tedy je vektor pravděpodobností takový, že každá hodnota reprezentuje pravděpodobnost jednoho z možných znaků. V rámci konvolučních vrstev této sítě jsou také použity vrstva STN, která má za úkol se naučit zaměřovat na konkrétní znaky. Tyto vrstvy naopak nejsou použity v rámci poziční sítě, protože po aplikaci geometrické transformace

by mohlo dojít ke ztrátě informace ohledně pozice v daném výřezu a nalezení vzdálenosti k dalšímu znaku by tudíž nebylo přesné. Zároveň je výstup znakové větve použit jako součást vstupu do poziční větve.

Jak je zmíněno v návrhu, poziční větev predikuje vzdálenost k následujícímu znaku. Tato vzdálenost je reprezentována jako počet pixelů, o které je potřeba posunout aktuální výřez tak, aby se uprostřed nového výřezu nacházel další znak. Jako výstup byl zvolen vektor hodnot, přičemž první hodnota představuje pravděpodobnost, že se další znak nachází ve vzdálenosti jeden pixel napravo od současné pozice, druhá hodnota o dva pixely napravo atd.

V této větvi je rovněž použita vlastní jednoduchá vrstva pro úpravu výstupního vektoru znakové větve. Tato vrstva (v rámci architektury nazvaná jako *One-Hot Binarization*) transformuje vektor odpovídající rozložení pravděpodobností na vektor obsahující samé nuly a jednu hodnotu 1 (takzvaný *one-hot vektor*). Původní vektor je převeden tak, že se zjistí pozice nejpravděpodobnější třídy a následně se vytvoří nový vektor obsahující samé 0 a na místo, kde se vyskytovala ona nejvyšší pravděpodobnost, se vloží hodnota 1.

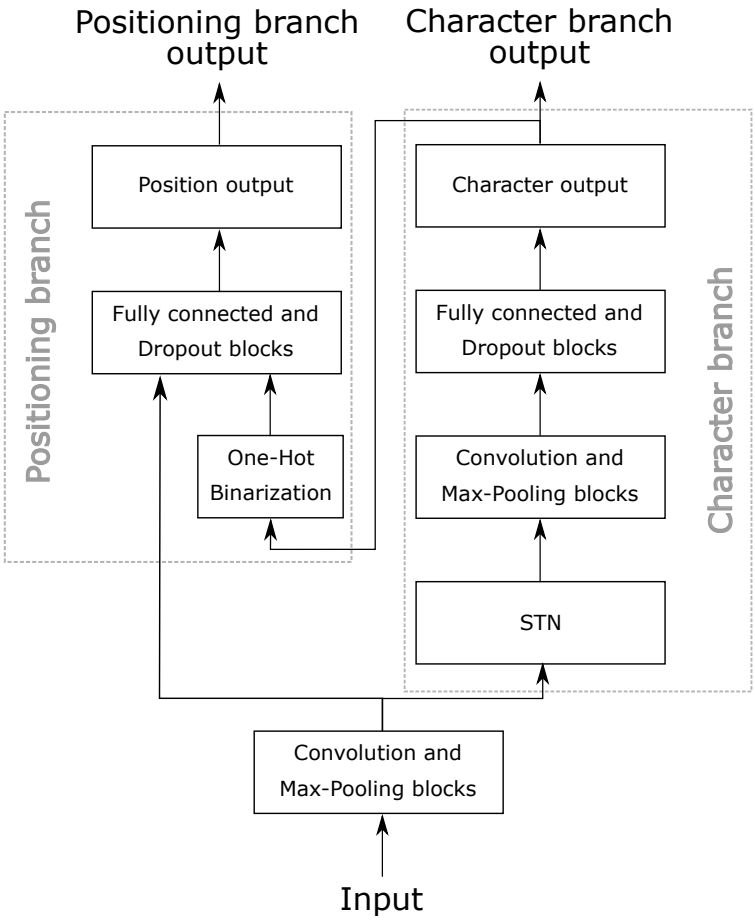
Přepis řádku Přepis řádku historického textu probíhá následovně. Na začátku je výsledný přepis inicializován na prázdný řetězec a zároveň se vytvoří první výřez na začátku řádku. Jako vstup neuronové sítě se použije aktuální výřez, přičemž výstupem jsou dva vektory hodnot. Z výstupu znakové větve se určí nejpravděpodobnější znak v daném výřezu a tento znak je připojen k výslednému přepisu. Z druhého vektoru se určí nejpravděpodobnější vzdálenost k dalšímu znaku a nový výřez je vytvořen tak, aby ona nová pozice byla uprostřed tohoto výřezu. Tento proces se opakuje až do momentu, kdy znaková větev klasifikuje aktuální výřez jako konec řádku. Pro klasifikaci konce řádku je vyhrazena jedna hodnota v rámci výstupního vektoru a z tohoto důvodu je velikost výstupního vektoru o jednu hodnotu větší, než je počet všech možných klasifikovaných znaků.

Trénování sítí V první fázi trénování výše navržených sítí (poziční a znaková síť a jejich varianty) je vhodné, aby datová sada obsahovala ground-truth anotace posloupnosti znaků a jejich pozicích. Vzhledem k tomu, že takto anotovaný dataset nebyl nalezen, byl vytvořen generátor umělých historických textů, který je popsán v části 5.

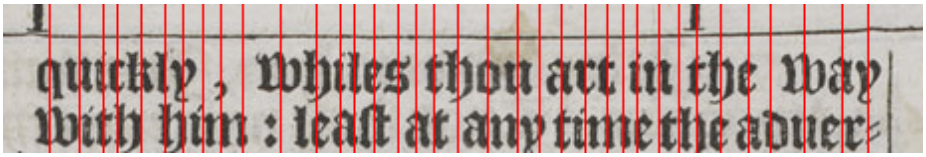
Při trénování sítí je k jednotlivým výřezům přidáván také šum, který způsobuje, že znak klasifikovaný v daném kroku není vždy přesně uprostřed. Technika přidávání šumu k pozicím výřezů je použita především k tomu, aby výsledná natrénovaná síť zvládala klasifikovat také znaky, které nejsou úplně přesně uprostřed výřezu. K takové situaci může dojít, když síť v předchozím kroku nevyhodnotí vzdálenost k dalšímu znaku úplně přesně a klasifikovaný znak pak neleží přesně uprostřed výřezu.

V dalších fází pak může být již částečně naučená neuronová síť na umělých datech použita pro získání pozic znaků v reálných historických textech. Proces takového zarovnání je následující. Nejprve je nutné mít ke vstupnímu obrázku jeho přesný přepis. Následně je možné použít neuronovou síť, která již byla trénována na umělých datech, tak, aby vytvořila několik různých přepisů tohoto řádku. Při vyhodnocování pravděpodobností se však nebere v úvahu pouze nejpravděpodobnější hodnota, avšak je třeba vybírat i hodnoty méně pravděpodobné, vždy však s ohledem na jejich pravděpodobnost v rámci získaného rozložení. Takto vznikne několik různých variant přepisů onoho vstupního textu s tím, že jsou známy také pozice, na kterých docházelo ke klasifikaci jednotlivých znaků.

Následně je možné, pomocí výpočtu Levenštejnovy vzdálenosti [25], odvodit zarovnání získaných přepisů a správného přepisu. Z takto zarovnaných přepisů je pak možné získat



Obrázek 3.3: Návrh architektury sítě.



(a) Přepis: „quickly , whiles thou art in the way“

(b) Přepis: „was in the begynnynge wyth“

Obrázek 3.4: Ukázka zarovnání řádků historických textů a jejich přepisů. Červená čára označuje středy jednotlivých znaků prostředního řádku výřezu.

pozice, při nichž byly dané znaky správně klasifikovány, a tedy vytvořit novou datovou sadu z těchto reálných dat. Takto vytvořený dataset je následně možné použít pro další trénování této sítě, čímž by se měla úspěšnost rozpoznávání opět zvýšit. Ukázka zarovnání je vyobrazena na obrázku 3.4.

Kapitola 4

Datové sady

Tato kapitola obsahuje informace o nalezených datových sadách, jenž by mohly být vhodné pro natrénování neuronové sítě sloužící pro automatický přepis historického textu. V rámci informací o datových sadách je přiložena také tabulka obsahující souhrn všech datasetů. Dále je každá datová sada detailněji popsána a je k ní také připojena obrazová ukázka.

V tabulce 4.1 se nachází stručný přehled datových sad vhodných pro rozpoznávání historických textů. Na obrázku 4.1 je možné vidět ukázky z jednotlivých datových sad.

ParzivalDB [11] je datová sada, která obsahuje 47 stránek historického textu. Na těchto stránkách se nachází 4 477 řádků textu. Každý tento řádek je pak samostatně anotován v přiložených textových dokumentech, přičemž tyto anotace tvoří jednotlivé znaky slov, jež jsou odděleny mezerou. Celá datová sada byla anotována ručně a jsou zde také speciální sekvence, které zastupují různé, dnes již nepoužívané, historické znaky. Stránky představují naskenované pergamenové spisy z období 13. století na nichž se podíleli tři písaři.

Další datovou sadou je *RIDGES* [20]. Tento dataset obsahuje 59 vědeckých knih, převážně z oblasti studia rostlin. Tyto knihy pocházejí z období mezi 15. a 20. stoletím. Ve veřejně přístupném online systému se pak nacházejí také anotace, v nichž se nachází přepisy jednotlivých stránek. Nejsou však zde použity speciální znaky pro historické znaky, ale tyto znaky jsou automaticky převedeny na znaky, kterými byly nahrazeny a dnes se běžně užívají.

HBA dataset je datová sada uveřejněná v rámci soutěže při *ICDAR* (*International Conference on Document Analysis and Recognition*) z roku 2017 [1]. Tento dataset obsahuje 11 knih s celkovým počtem 4 436 stran. Tyto knihy se datují do období 13.–19. století. Společně s obrázky naskenovaných historických stránek jsou k dispozici anotace, v nichž se nachází informace o každém pixelu. Každý pixel stránky je zařazen do jedné ze šesti kategorií. Tento dataset tedy na trénování neuronové sítě pro rozpoznávání textu není příliš vhodný. Nicméně může být použit při evaluaci výsledného systému.

Dataset *Impact* [10] obsahuje velké množství různých naskenovaných knih z období 16.–20. století. Tyto knihy jsou psány v různých jazycích, zastoupeny jsou například čeština, španělština, angličtina, němčina a další. K těmto naskenovaným dokumentům je ve většině případů také připojený soubor s anotacemi.

Předposledním zdrojem dat je *Digitální knihovna*[1]. Ta neobsahuje typickou datovou sadu, nýbrž obsahuje velké množství naskenovaných dokumentů, které se datují do období od 12. do 21. století. Hojně jsou zde tedy zastoupeny také české knihy z období novověku.

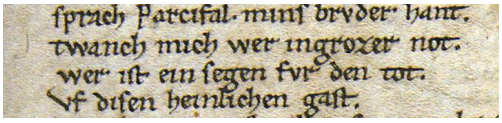
[1] digitalniknihovna.cz

Název datové sady	Anotace	Století	Jazyk	Rozsah
ParzivalDB	Přepis řádků	13.	Němčina	47 stránek
RIDGES	Přepis řádků	15.–20.	Němčina	59 knih
HBA dataset	–	13.–19.	Vícejazyčná sada	4 436 stránek
Impact	Přepis řádků	16.–20.	Vícejazyčná sada (včetně Češtiny)	výběr knih
Digitální knihovna	–	12.–21.	Vícejazyčná sada (včetně Češtiny)	výběr knih
Bible krále Jakuba (King James Bible)	Přepis odstavců	17.	Angličtina	80 knih, stovky stránek

Tabulka 4.1: Přehled datových sad.

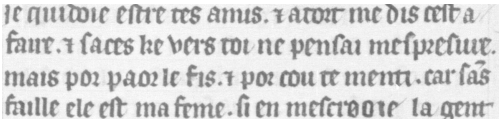
Dalšími z řady typů dokumentů jsou například noviny a časopisy, rukopisy, mapy a podobně. Zastoupeni jsou zde kupříkladu Jan Amos Komenský, Josef Dobrovský a další spisovatelé.

Posledním zdrojem dat je *Bible krále Jakuba* (anglicky *King James Bible*, případně *King James Version*) [18]. Bible krále Jakuba je staroanglický překlad křesťanské bible, který vznikl na začátku 17. století [33]. Celkem obsahuje 80 knih (39 knih Starého zákona, 14 knih apokryfů a 27 knih Nového zákona) a celkově stovky stránek historického textu. Bible byla tištěna převážně pomocí písma Fraktura, avšak úvody k jednotlivým částem textů jsou napsány Antikvou. Občas se však text psaný Antikvou nachází i v rámci řádků psaných Frakturou. Takovéto změny písma komplikují proces přepisu a z tohoto důvodu jsou tyto části z datové sady vynechány.

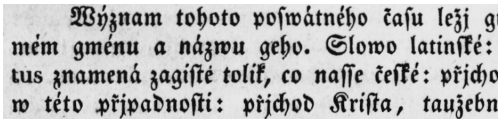


(a) ParzivalDB

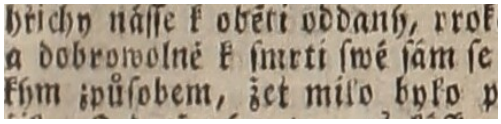
(b) RIDGES



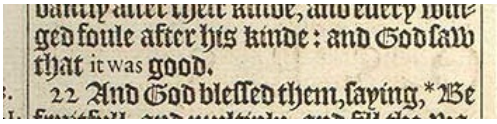
(c) HBA dataset



(d) Impact



(e) Digitální knihovna



(f) Bible krále Jakuba

Obrázek 4.1: Ukázka datasetů.

Kapitola 5

Generátor umělých historických textů

V této kapitole je popsán implementovaný generátor historických textů. Motivací k jeho vytvoření bylo, že se nepodařilo najít vhodnou datovou sadu, která by obsahovala naskenované historické texty s jejich přepisem a zároveň s dodatečnou informací o pozici těchto písmen. Výhodou takto implementovaného generátoru je také množství dat, která mohou být vygenerována, a spolu s tím také možnost přesné lokalizace všech písmen.

Implementovaný generátor vytváří stránky umělých historických textů takové, aby co nejvíce vypadaly jako pravé naskenované historické dokumenty. Ke generování obrázků je zapotřebí několika zdrojů. Prvním zdrojem jsou vhodné fonty, které v maximální možné míře odpovídají dobovým písmům. Druhým zdrojem jsou obrázky (textury), které mají vzhled starého papíru, které slouží jako podklad, na které jsou výsledné vygenerované texty naneseny.

Aby vygenerované obrázky co nejvíce odpovídaly reálným textům, je na původní vysázený text aplikováno několik takzvaných efektů. Tyto efekty mají za úkol upravovat originální vysázený text tak, aby se co nejvíce přiblížil vzhledu historických textů. Tyto efekty jsou popsány v následující části 5.2.

K tomu, aby efekty neupravovaly vstupní obrázek pokaždé stejným způsobem, je v generátoru několikrát použito generování náhodných čísel. Aby se daly jednoduše ovládat minimální a maximální hodnoty při tomto generování, je jediným parametrem při spuštění konfigurační soubor, který obsahuje mimo jiné právě tyto minimální a maximální hodnoty pro jednotlivé efekty. Dalšími parametry, které jsou uloženy v rámci konfiguračního souboru, jsou například rozměry výsledného obrázku, adresáře s fonty a podobně. Ukázku celého konfiguračního souboru je možné nalézt v příloze.

K vysázení daného textu do obrázku byla použita knihovna *freetype* ve verzi pro python[1]. Tato knihovna umožňuje vysázení vlastního textu na poměrně nízké úrovni a je tedy možné během sázení uchovávat užitečné informace, jako například již zmíněné pozice jednotlivých znaků. Detailnější informace ohledně sázení textu je popsán v navazující části 5.1.

5.1 Sázení textu

Jak je zmíněno v předchozí části, k vysázení se používá knihovna freetype. Podle oficiálního návodu k použití této knihovny, se text sází ve dvou fázích. V první fázi se vypočítá celková

[1] https://pypi.python.org/pypi/freetype-py/

Obrázek 5.1: Ukázka dvou rozdílných řádků z pohledu velikosti mezer v rámci jedné stránky dokumentu.

velikost matice, která je pro daný text zapotřebí, a následně se ve druhé fázi skládají vykreslené znaky do finálního obrázku. Během obou fází se také geometricky upravují vysázené znaky, každé písmeno je rotováno o určitý úhel, jenž je náhodně vygenerován z normálního rozložení s parametry z konfiguračního souboru. Dále jsou také písmena upravována z hlediska posunu. Pro každé písmeno je vygenerována hodnota, o kterou je dané písmeno posunuto, ať již doleva či doprava, oproti pozici, na níž by bylo standardně vygenerováno při použití kerningu. Tato hodnota je opět generována z normálního rozložení s parametry definovanými v rámci konfiguračního souboru.

Výstupem celého generování jsou stránky imitující původní historické texty. Takové texty bývají většinou zarovnány do bloku, čehož se pouhým skládáním vykreslených znaků dosáhnout nedá. Při generování výsledné stránky je také zapotřebí nastavit její celkovou šířku. Aby bylo zarovnání textu do bloku, jsou ze vstupního textu brány jednotlivá slova a postupně je vypočtena celková velikost aktuálního řádku s tím, že mezi slova se vkládá minimální mezera, jenž je opět definována v rámci konfiguračního souboru. Jakmile celková délka aktuálního řádku přesáhne požadovanou šířku, tak je poslední slovo odebráno a velikosti mezer mezi jednotlivými slovy jsou dopočítány tak, aby tyto mezery doplnily šířku řádku na požadovanou hodnotu.

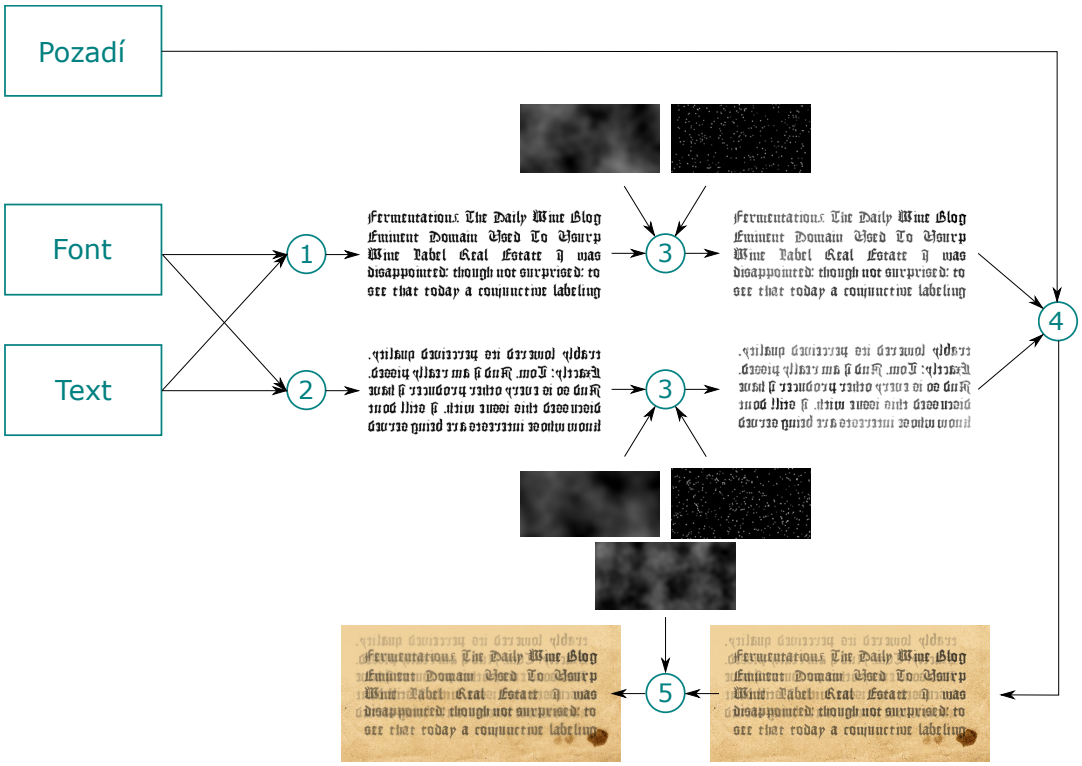
Tento postup také přidává do vygenerovaných stránek jeden důležitý prvek a tím je variabilita mezer mezi slovy. V originálních historických textech se nachází velká variabilita mezer mezi slovy a to až do té míry, že někdy vypadá jako jedna mezera, jako by tam byly dvě. Naopak někdy jsou mezery tak malé, že při pohledu zdálky by se mohlo zdát, že celý řádek je jedno dlouhé slovo. Ukázka dvou rozdílných řádků z jedné stránky dokumentu je na obrázku 5.1.

Každý vysázený řádek je přidán k již předchozím vygenerovaným řádkům, čímž se vytvoří výsledná stránka textu. Mezi řádky je vkládána mezera, jejíž velikost je opět získána z normálního rozložení s parametry definovanými v konfiguračním souboru. Výstupem sázení textu je dvourozměrná matice, kde každý bod je hodnota v rozsahu $0 - 255$.

Ještě před samotným sázením textu je v rámci konfiguračního souboru možné povolit dvě úpravy sázeného textu. První úpravou je převod prvního písmene slova na velké písmeno (majuskule), tedy například z „a“ na „A“. Druhou úpravou je přidávání speciálních znaků, jako jsou tečka, čárka, vykřičník, apod., za slova. Obě tyto úpravy se mohou nezávisle na sobě provádět s určitou pravděpodobností na každém slově vstupního textu. Takovéto úpravy mohou být vhodné zejména, pokud vstupní text neobsahuje příliš velký písmen, případně speciálních znaků.

5.2 Efekty

Efekty slouží k úpravě vysázeného textu tak, aby výsledek co nejvíce vypadal, jako reálný historický text. Všechny efekty jsou postupně aplikovány na sebe tak, až vznikne výsledný obrázek. Celkový proces aplikace jednotlivých efektů schematicky znázorněn na obrázku 5.2.



Obrázek 5.2: Schéma generování umělých historických stránek. Operace číslo 1 odpovídá vysázení textu, operace 2 vysázení zadního textu. Operace 3 představuje efekt simulující nedokonalosti tisku. Operace č. 4 nanáší vysázené texty na vybranou texturu a operace č. 5 přidává rozmazání.

Nedokonalosti tisku Prvním efektem v posloupnosti, která je aplikována na obrázek, je efekt, jenž imituje nedokonalosti při historickém tisku. Tento efekt aplikuje na původní obrázek dva typy nedokonalostí. Nejprve je vygenerována matice hodnot v rozmezí 0 až 1, přičemž ke generování se používá knihovna *noise*[2], která vytváří náhodné souvislé mapy. Tato mapa poté slouží k imitaci nerovnoměrnosti při tisku, kdy se některé části písmen nemusí dobře otisknou na papír a tak je zde text nepatrně bledší.

Druhým typem nedokonalosti jsou defekty písmen, kdy se určitá malá část písmene neotiskne vůbec. Toto může být způsobeno například poškozením dané litery, případně nečistotou papíru. Tato nedokonalost je vytvořena tak, že se do matice, která má shodné rozměry jako původní obrázek, náhodně umístí několik náhodných hodnot v rozsahu 0 až 1. Následně se provede Gaussovské rozostření, které dodá těmto bodů plynulejší přechod do zbytku matice a zároveň je nepatrně zvětší. Nakonec jsou hodnoty alfa kanálu původního obrázku vynásobeny hodnotami obou těchto matic.

Zadní text Dále je s určitou pravděpodobností aplikován efekt, který způsobí, že se ve výsledném obrázku objeví druhý text, který je převrácen přes vertikální osu a zároveň je velice průsvitný. Toto představuje situaci, kdy na se na danou stránku otiskne text z následující strany, případně přes list papíru prosvítá text, který se nachází na druhé straně. Na takto vygenerovaný text se také aplikuje předchozí efekt, tedy efekt, který napodobuje

[2] https://github.com/caseman/noise

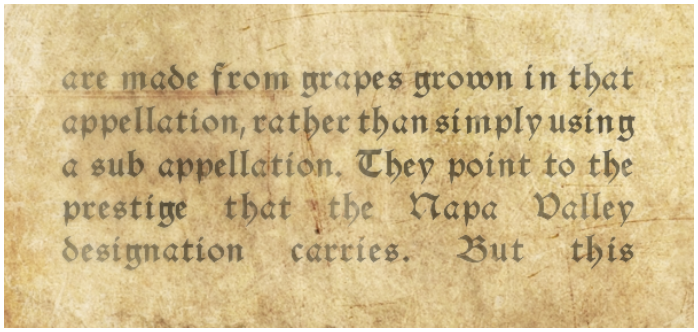
nedokonalosti při tištění. Tento výsledný obrázek je nakonec nanesen na texturu papíru, jež byla pro daný text náhodně zvolena.

Nanesení na texturu Po aplikaci výše zmíněných efektů, je aktuální obrázek nanesen na zvolenou texturu. Toto se děje pomocí procesu zvaného *alpha blending* (případně *alpha compositing*). Jedná se o techniku, kdy jsou přes sebe prolnuty dva obrázky s tím, že jeden je použit jako podklad (pozadí) a druhý je na něj nanesen na základě hodnot alfa kanálu (průsvitnosti bodů).

Rozmazání písmen Následně je na obrázek aplikován ještě poslední efekt, který způsobí, že některé části písmen vypadají jako rozpité. Tohoto efektu se dosáhne tak, že je vygenerována náhodná mapa, obdobně jako u efektu nedokonalosti tisku, a také se vytvoří nový obrázek, který vznikne aplikací Gaussovského rozostření na původní obrázek. Následně jsou oba obrázky prolnuty s tím, že v daném bodě je výsledná intenzita je rovna součtu hodnot v původním a rozostřeném obrázku, přičemž jedna z hodnot je vynásobena hodnotou ve stejném bodě ve vygenerované mapě a druhá z hodnot je vynásobena hodnotou, která vznikla jako jedna mínus hodnota v mapě.

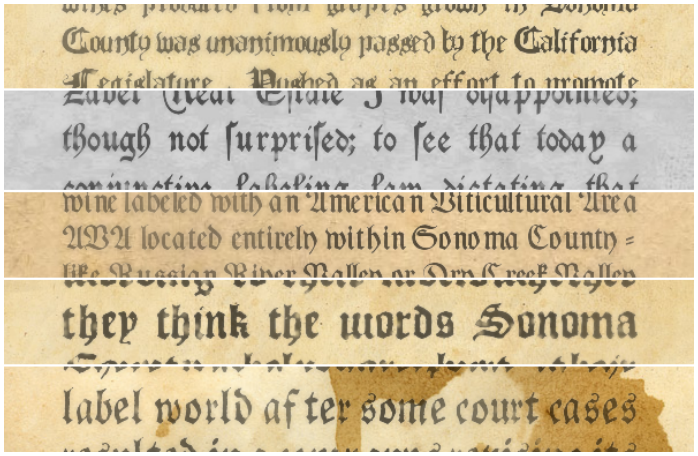
5.3 Výstupy generátoru

Hlavními výstupy generátoru jsou především samotný obrázek a podrobné anotace obsahující bounding boxy jednotlivých znaků a pozice základních dotažnic (baseline) jednotlivých řádků. Vedle těchto výstupů, generátor produkuje také výstup anotací ve formě XML souboru, který odpovídá výstupu programu Transkribus, jenž je popsán v rámci kapitoly 2.6. Dalším výstupem je potom také další obrázek, přičemž hodnota každého jeho pixelu reprezentuje určitou třídu. Pomocí těchto tříd jsou rozlišovány jednotlivé znaky, pozadí a mezery mezi slovy. Takovýto výstup je vhodný zejména pro sémantickou segmentaci. Ukázka výstupů generátoru umělých historických textů je vyobrazena na obrázku 5.3.



(a) Ukázka jedné stránky hlavního obrazového výstupu generátoru.

(b) Ukázka různých výstupů pro jeden vygenerovaný obrázek. První obrázek představuje vygenerovaný řádek textu, druhý obrázek ukazuje vykreslené anotace pro daný řádek. Poslední obrázek zobrazuje výstup pro sémantickou segmentaci, fialové obdélníky představují mezery mezi slovy.



(c) Ukázka variability při generování.

Obrázek 5.3: Ukázka výstupů generátoru.

Kapitola 6

Implementace

Tato kapitola je zaměřena na některé implementační detaily této práce. V první části jsou stručně popsány nástroje, které jsou použity pro trénování neuronových sítí. Následující část se zabývá implementací Spatial Transformer Network. V další části je popsán proces zpracování vygenerovaných dat použitých pro trénování sítí a poslední část se věnuje implementačním detailům rekurentních neuronových sítí.

6.1 Použité nástroje

Hlavním programovacím jazykem této práce je jazyk Python. Nástrojem použitým pro trénování neuronových sítí je knihovna zvaná *Keras* [8]. Keras je vysokoúrovňová knihovna pro popis, trénování a testování neuronových sítí napsaná v jazyce Python, která používá optimalizované knihovny pro rychlé výpočty, takzvaný *backend*. Keras je schopen běžet nad knihovnami *TensorFlow* [2], *CNTK* [26] a *Theano* [3]. V rámci této práce je použita knihovna TensorFlow, jenž je vyvíjena firmou *Google*, a která je také výchozí knihovnou pro backend.

6.2 Spatial Transformer Network

Ačkoliv nabízí knihovna Keras poměrně velké spektrum prostředků, které mohou být v rámci neuronových sítí použity, Spatial Transformer Network (STN) [15] však v knihovně chybí. Proto je zde použita implementace *Octavia Arriaga*[1]. Tato implementace umožňuje naučit se 2D afinní transformace zahrnující posun, oříznutí, rotaci a škálování. Vzhledem k tomu, že v této práci jsou rozpoznávány řádky textu, jejichž písmena se nacházejí v jedné linii, byly transformace omezeny pouze na posun, oříznutí a izotropní změnu měřítka (viz 2.4). Z tohoto důvodu není zapotřebí, aby výstupní vektor lokalizační sítě obsahoval šest hodnot, ale postačují pouze hodnoty tři. Protože však celá implementace počítá s tím, že rozměr výstupu bude odpovídat šesti hodnotám, bylo potřeba tuto změnu implementovat.

Vzhledem k tomu, že TensorFlow je optimalizovaná knihovna pro výpočty, není příliš vhodné měnit či nastavovat hodnoty při samotném výpočtu. Do výpočtu tedy byla vložena operace násobení matic, která vektor o třech hodnotách rozšíří na vektor o šesti hodnotách. Výsledná matice je následující:

[1] https://github.com/oarriaga/spatial_transformer_networks

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Při vynásobení vektoru, jenž je výstupem lokalizační sítě, vznikne nový vektor, z něhož, po přeskládání na matici 2×3, vznikne výsledná transformace. Pro vektor $\boldsymbol{v} = [s, t_x, t_y]$ je výpočet následující:

$$\boldsymbol{v'} = \boldsymbol{v}\boldsymbol{M} = \begin{bmatrix} s & t_x & t_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s & 0 & t_x & 0 & s & t_y \end{bmatrix}$$

Výsledná transformace po přeskládání je:

$$\boldsymbol{T} = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix}$$

6.3 Zpracování umělé datové sady

Výstupem generátoru z kapitoly 5 jsou vygenerované umělé stránky historického textu a jim odpovídající anotace. Pro trénování se však používají výřezy jednotlivých znaků s částí jejich okolí. Výřezy jsou ze stránky získány pomocí implementovaného skriptu, jenž při vytváření jednotlivých výřezů, pro každý znak vytvoří několik různých výřezů.

Jeden výřez pak obsahuje požadovaný znak přesně uprostřed a ostatní výřezy obsahují tento znak posunutý určitým směrem, jak je popsáno v části 3.2. Tento směr je náhodně generován z normálního rozložení, jehož parametry jsou specifikovány v konfiguračním souboru.

Výsledkem tohoto zpracování je sada souborů, jejichž název je ve tvaru `image_[číslo stránky]_[číslo sekvence]_[číslo obrázku v sekvenci].png`. Takto strukturovaný název je nutný především při trénování rekurentních sítí, kdy je při načítání datové sady zapotřebí zjistit, které výřezy patří stejné sekvenci znaků.

6.4 Rekurentní sítě

Jak je již nastíněno v předchozí části, trénování rekurentních neuronových sítí se od trénování nerekurentních sítí mírně liší. Jde především o to, že při trénování rekurentních sítí je zapotřebí, aby data v trénovací dávce (anglicky takzvaný *batch*) představovaly jednotlivé sekvence. Tuto podmínku je nutné dodržet aby se rekurentní vrstvy naučily uchovávat správné informace o kontextu. Z tohoto důvodu byl implementován *BatchGenerator* a byla použita speciální obalovací vrstva *TimeDistributed*, jenž jsou popsané v následujících odstavcích.

BatchGenerator BatchGenerator je implementovaná třída sloužící pro generování trénovacích a validačních dávek. Tyto dávky jsou vytvářeny z načtené datové sady, která si zároveň uchovává informaci o tom, do které sekvence daný výřez patří. Parametry generátoru jsou tři. Vedle samotného datasetu jsou to *počet sekvencí v dávce* a *počet výřezů v jedné sekvenci*. BatchGenerator, jenž je implementován jako iterátor, pak při každém zavolání metody `__next__` vytvoří novou dávku tak, že vybere náhodnou sekvenci datasetu a z ní vybere příslušný počet výřezů.

Aby pokaždé při výběru stejné sekvence nebyl výstup stejný, je ještě náhodně vybrán také počáteční výřez, od kterého se sekvence do dávky přidá. Pokud nastane případ, že již sekvence v datové sadě nepokračuje, ale je zapotřebí ještě doplnit data do výsledné dávky, pak je opakovaně přidáván poslední výřez, který obvykle označuje konec řádku. Namísto klasické datové sady je pak takto implementovaný generátor použit jako zdroj dat pro trénování.

TimeDistributed wrapper Druhou důležitou součástí při trénování konvolučních rekurentních sítí v knihovně Keras, je speciální obalovací vrstva TimeDistributed[2]. Jedná se takzvaně o *wrapper*, který lze aplikovat na jakoukoliv jinou vrstvu neuronové sítě.

Při trénování nerekurentních konvolučních neuronových sítí má většinou trénovací dávka čtyři rozměry – *počet obrázků v dávce*, *výška obrázku*, *šířka obrázku* a *počet kanálů obrázku*. Při trénování rekurentních konvolučních neuronových sítí je však zapotřebí, aby měla dávka rozměrů pět – *počet sekvencí*, *počet obrázků v sekvenci*, *výška obrázku*, *šířka obrázku* a *počet kanálů obrázku*. K dorovnání tohoto rozdílu slouží právě wrapper TimeDistributed.

Tato obalovací vrstva způsobí, že vrstva, kterou obaluje, zpracuje jednotlivé výřezy, jako by se jednalo o čtyřrozměrnou dávku. Výsledek pak je opět transformován do pětirozměrné dávky, takže jsou zachovány jednotlivé sekvence. Wrapper TimeDistributed není použit v případě rekurentních vrstev, protože ty naopak vyžadují pětirozměrnou dávku.

[2] https://keras.io/layers/wrappers/

Kapitola 7

Experimenty

V této kapitole se nachází popis a výsledky provedených experimentů. První experiment se zaměřuje na velikost neuronové sítě, přičemž byly natrénovány a následně otestovány dvě neuronové sítě s rozdílnými velikostmi architektur. Ve druhém experimentu se vyhodnocuje úspěšnost při použití STN v rámci architektury sítě. Třetí experiment zjišťuje, zda pomáhá propojení mezi znakovou a poziční větví při rozpoznávání řádků textu. Čtvrtý experiment je zaměřen na použití a porovnání rekurentní sítě oproti síti bez rekurentních vrstev. Poslední experiment ukazuje, jak se zlepší rozpoznávání sítě po natrénování na reálných datech.

Trénovací datová sada Při všech experimentech byla neuronová síť nejdříve učena na umělých datech získaných z použití generátoru, jenž je popsán v kapitole 5. Celkově bylo při generování použito 34 fontů, které vypadají, jako písmo Fraktura. Dále bylo použito 86 různých textur představující historický papír, na který je vysázený text nanášen.

Posledním zdroj potřebným pro generování je onen text, jenž je vysázen. Za účelem generování trénovací datové sady byl použit textový korpus nazvaný *Day3PMSession*[1] získaný z databáze *The Open American National Corpus* [4]. Tento textový korpus obsahuje celkem 20 817 anglických slov skládající se celkově z 121 296 znaků. Z této textové datové sady bylo vygenerováno 118 umělých historických stránek, přičemž každá stránka obsahovala 5 řádků textu. Tento počet byl zvolen proto, aby bylo při generování pokryto co nejvíce kombinací fontů a textur pozadí.

Z vygenerovaných stránek jsou získány výřezy způsobem, který je popsán v předchozí kapitole. Celkem takto vzniklo 467 925 výřezů, jež byly při trénování sítí rozděleny do dvou částí. První část určená pro samotné trénování obsahovala 397 030 výřezů, druhá část byla určená pro validaci trénování, přičemž obsahovala zbylých 70 895 výřezů.

Testovací datové sady Pro testování natrénovaných neuronových sítí bylo použito hned několik datových sad. První a druhá datová sada je opět vytvořená pomocí generátoru umělých textů. Rozdíl mezi nimi je takový, že zatímco první sada obsahuje, stejně jako datová sada pro trénování, jednotlivé výřezy každého znaku, tak druhá datová sada obsahuje obrázky, které obsahují celé jednotlivé vygenerované řádky textu. V rámci první sady je pro každý výřez znám znak, jenž je v tomto výřezu klasifikován, a také vzdálenost k dalšímu znaku. U datové sady obsahující pouze řádky je znám pouze výsledný přepis celého řádku. Tato datová sada imituje použití výsledného systému na reálných datech.

[1] http://www.anc.org/MASC/texts/Day3PMSession.txt

Jak již bylo zmíněno, tyto datové sady byly vygenerovány obdobně jako trénovací datová sada. Byly zde použity stejné fonty a textury pozadí, avšak lišily se ve vstupním textovém korpusu. Generovaný text byl opět získán z databáze *The Open American National Corpus* [4], avšak zde se jednalo konkrétně o korpus *Fermentation Eminent-Domain*[2]. První sada pak obsahuje 11 175 obrázků, kde jeden obrázek odpovídá jednomu písmenu, a druhá sada obsahuje 360 obrázků, přičemž tento jeden obrázek odpovídá jednomu řádku textu.

Další datovou sadu tvoří řádky reálných historických textů. Tyto řádky byly získány z datové sady *King James Bible* a také z databáze *Impact*. Vzhledem k tomu, že pro data ani z jedné datové sady neexistovaly přepisy řádků, bylo třeba je ručně anotovat. Datová sada obsahuje celkem 527 obrázků řádků, přičemž tyto řádky obsahují dohromady 16 580 znaků.

Vyhodnocení úspěšnosti Úspěšnost rozpoznávání je vyhodnocována dvěma způsoby. V případě testovací sady, která obsahuje výřezy jednotlivých písmen, je úspěšnost rozpoznávání vyhodnocena jako procentuální úspěšnost správně klasifikovaných obrázků. U datových sad obsahujících celé řádky je vyhodnocení úspěšnosti vypočteno pomocí Levenštejnovy vzdálenosti[3] (viz 2.5). Úspěšnost přepisu řádku je vyhodnocena jako znaková přesnost, jenž je vypočtena pomocí na základě vztahu:

$$a_c = \frac{l - d}{l} \tag{7.1}$$

kde l je počet znaků řádku a d je Levenštejnova vzdálenost mezi skutečným a získaným přepisem řádku.

7.1 Velikost sítě

První experiment zkoumá vliv velikosti sítě na úspěšnost rozpoznávání textu. V rámci experimentu jsou natrénovány dvě neuronové sítě, které vycházejí z totožné architektury, jenž byla představena v kapitole ohledně návrhu řešení (kapitola 3, obrázek 3.3. Rozdíl mezi oběma sítěmi představuje počet použitých vrstev v konvolučních a plně propojených blocích. Počty a parametry jednotlivých vrstev jsou popsány v tabulce 7.1.

Výsledky úspěšnosti rozpoznávání textu se nacházejí v tabulce 7.2. Úspěšnost určování pozic je vyjádřena grafem na obrázku 7.1. Z výsledků je možné vyčíst, že neuronová síť s více vrstvami dosahovala lepších výsledků než menší síť.

7.2 Použití STN

Tento experiment ukazuje výhody použití vrstvy se Spatial Transformer Network [15] ve znakové větvi. V rámci experimentu byla natrénována další neuronová síť, která měla stejnou architekturu, jako větší síť z prvního experimentu, avšak neobsahovala vrstvu STN. Výsledky této nové sítě v porovnání s větší sítí z experimentu 1 se nacházejí v tabulce 7.3. Ze získaných výsledků je možné vyčíst, že úspěšnost rozpoznávání sítě s STN je vyšší, než při použití sítě bez STN.

[2] http://www.anc.org/MASC/texts/Fermentation_Eminent-Domain.txt

[3] Skript pro výpočet Levenštejnovy vzdálenosti byl poskytnut vedoucím práce, Ing. Michalem Hradišem, Ph.D. Implementace pochází od Doc. Ing. Lukáše Burgeta, Ph.D.

Bloky	Menší síť	Větší síť
Konvoluční vrstvy + Max-pooling	$1, 16, 2 \times 2, 2 \times 2$ $1, 32, 2 \times 2, 2 \times 2$ $1, 64, 2 \times 2, 2 \times 2$	$3, 16, 2 \times 2, 2 \times 2$ $3, 32, 2 \times 2, 2 \times 2$ $3, 64, 2 \times 2, 2 \times 2$
Plně propojené vrstvy + Dropout	$256, 0.5$ $256, 0.5$	$1\,024, 0.5$ $1\,024, 0.5$

Tabulka 7.1: Architektury sítí z experimentu 1. Hodnoty u bloku reprezentující bloky obsahující konvoluční vrstvy a max-pooling jsou následující: první hodnota určuje počet konvolučních vrstev v bloku, druhá představuje počet konvolučních jader každé konvoluční vrstvy, třetí hodnota představuje velikost konvolučního jádra a poslední hodnota určuje velikost filtru max-pooling vrstvy. V rámci bloků plně propojených vrstev a dropout představují hodnoty následující parametry: první hodnota udává velikost plně propojené vrstvy (počet neuronů), druhá pravděpodobnost dropoutu.

Dataset	Menší síť	Větší síť
Umělý dataset - výřezy	89.5 %	94.4 %
Umělý dataset - řádky	83.2 %	91.1 %
Historický dataset	74.2 %	79.1 %

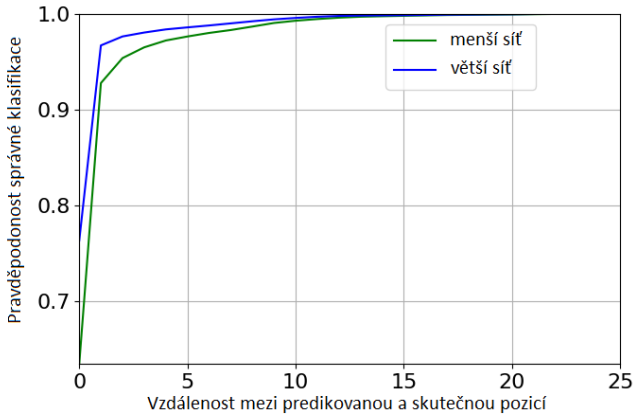
Tabulka 7.2: Výsledky experimentu 1. Hodnoty představují úspěšnost rozpoznávání znaků.

7.3 Propojení znakové a poziční větvě

Další experiment se zaměřuje na zjištění, jaký vliv má propojení znakové a poziční větvě v rámci navržené neuronové sítě. Architektura nově natrénované sítě opět vychází z navržené neuronové sítě, avšak nyní bez propojení výstupu znakové větve na vstup do poziční větve. Zároveň v poziční větvi není vrstva transformující rozložení pravděpodobnosti na one-hot vektor. Obě větve mají stejné počty vrstev, jako má větší síť z experimentu 1. Výsledky úspěšnosti rozpoznávání řádků textu se nacházejí v tabulce 7.4, úspěšnosti predikce pozice následujícího znaku jsou vidět v grafu na obrázku 7.2. Z výsledků vyplývá, že síť bez propojení dosahuje obdobných úspěšností, jako síť s propojením.

7.4 Rekurentní síť

Tento experiment ukazuje, jak se změní úspěšnost rozpoznávání pokud má síť k dispozici kontext. Architektura sítě opět vychází z architektury navržené sítě a má stejný počet vrstev, jako má větší síť z prvního experimentu. Místo plně propojených vrstev jsou však v tomto experimentu použity rekurentní vrstvy, konkrétně LSTM [14] (viz 2.3). Tabulka 7.5 obsahuje výsledky rekurentní sítě. Z výsledků je patrné, že ačkoliv na umělých datových sadách dosahují rekurentní i nerekurentní sítě přibližně stejných úspěšností, u historického datasetu tomu tak není. To může být způsobeno rozdílným jazykovým modelem naučeným v rámci umělého datasetu a historickým jazykovým modelem. Natrénování jazykového modelu na historických datech se věnuje další experiment.



Obrázek 7.1: Výsledek určování pozic písmen.

Dataset	Síť bez STN	Síť s STN
Umělý dataset - výřezy	94.4 %	94.4 %
Umělý dataset - řádky	89.4 %	91.1 %
Historický dataset	77.1 %	79.1 %

Tabulka 7.3: Výsledky experimentu 2. Hodnoty představují úspěšnost rozpoznávání znaků.

7.5 Natrénování na reálných datech

Poslední experiment je zaměřen na vyhodnocení úspěšnosti rozpoznávání textu sítí, která je natrénována na reálných datech. K tomu, aby bylo možné natrénovat navrženo neuronovou síť na reálných datech, je potřeba provést jejich zarovnání. Toto zarovnání bylo provedeno tak, jak je popsáno v kapitole popisující navrženou síť (viz 3.2).

Zarovnaná datová sada Zarovnaná datová sada byla vytvořena z řádků knihy *King James Bible*. Celkem se jednalo o 369 řádků, které obsahovaly 11 565 znaků. Při zarovnání byly nalezeny pozice 11 503 znaků, což představuje 99.5% úspěšnost.

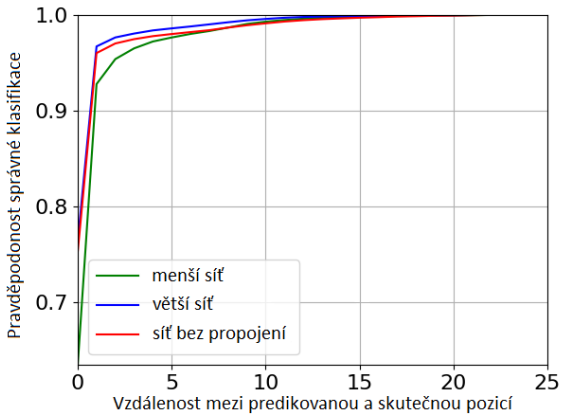
Na výše popsané datové sadě byly dotrénovány dvě sítě. První byla větší síť z prvního experimentu a druhá byla rekurentní síť z minulého experimentu. Úspěšnost rozpoznávání pak byla vyhodnocována již pouze na historickém datasetu. Výsledky obou sítí se nacházejí v tabulce 7.6. Z těchto výsledků je vidět, že obě sítě se značně zlepšily, konkrétně rekurentní síť o 12.3% a nerekurentní síť o 7.2%. Stále však platí, že rekurentní síť nedosahuje úspěšnosti nerekurentní sítě.

7.6 Shrnutí výsledků

Z předchozích experimentů vyplývá, že nerekurentní sítě dosahují vyšších úspěšností na reálné historické datové sadě než rekurentní sítě. Dále je také ukázáno, že použité techniky (Spatial Transformer Network a propojení znakové a poziční větve) přinášejí určité zvýšení úspěšnosti rozpoznávání.

Dataset	Síť bez propojení	Síť s propojením
Umělý dataset - výřezy	94.4 %	94.4 %
Umělý dataset - řádky	94.5 %	91.1 %
Historický dataset	78.1 %	79.1 %

Tabulka 7.4: Výsledky experimentu 3. Hodnoty představují úspěšnost rozpoznávání znaků.



Obrázek 7.2: Výsledek určování pozic písmen.

Při ruční kontrole přepisů bylo zjištěno, že sítě občas přeskakovaly úzké znaky, jako jsou například „l“, „i“ a podobné. U některých širších znaků (například „w“, „u“ a dalších) naopak docházelo k tomu, že neuronové sítě určily pozici následujícího znaku v rámci aktuálního znaku. Problém také způsobovaly mezery mezi jednotlivými slov. Občas byly mezery tak malé, že došlo k vynechání této mezery, nebo naopak mezera byla tak velká, že neuronová síť vložila mezeru vícekrát.

Mezi další časté chyby nerekurentní sítě patřily záměny velkého a malého znaku pro dané písmeno, či záměna s podobným znakem (například písmeno „o“ za číslici „0“).

Dataset	Rekurentní síť	Nerekurentní síť
Umělý dataset - výřezy	95.8 %	94.4 %
Umělý dataset - řádky	88.9 %	91.1 %
Historický dataset	70.4 %	81.8 %

Tabulka 7.5: Výsledky experimentu 4. Hodnoty představují úspěšnost rozpoznávání znaků.

	Rekurentní síť	Nerekurentní síť
Úspěšnosti původních natrénovaných sítí	70.4 %	81.8 %
Úspěšnosti sítí dotrénovaných na reálných datech	82.7 %	89.0 %

Tabulka 7.6: Výsledky experimentu 5. Hodnoty představují úspěšnost rozpoznávání znaků.

Kapitola 8

Závěr

Cílem této práce je vytvořit automatický nástroj pro rozpoznávání historických textů. Jedná se především o novověké texty psané kategorií písem zvané Fraktura. V rámci této práce bylo nalezeno několik vhodných datových sad, které by měly sloužit k trénování a následné evaluaci navrženého řešení. Tímto řešením je nově navržená neuronová síť, která se skládá ze dvou větví. Znaková větev klasifikuje znak v aktuálním výřezu, který se nachází nejblíže středu výřezu. Poziční větev následně určuje vzdálenost k dalšímu znaku, kde je vytvořen nový výřez. Opakováním těchto kroků se následně přepíše celý řádek textu. Součástí navržené sítě je také Spatial Transformer Network, jenž aplikuje geometrické transformace na danou vstupní matici. Také je možné použít v síti rekurentní vrstvy, které umožňují uchovávat kontext při samotném přepisování.

Součástí této práce je také implementovaný generátor umělých historických textů, který se snaží co nejvíce přiblížit reálným textům. Takovýto generátor je vhodný především ze dvou důvodů. Tím prvním je, že pomocí něj je možné vygenerovat libovolné množství umělých historických textů. Druhým důvodem k implementaci takového generátoru je, že pomocí něj je poměrně snadné získat anotace vytvořených textů. Jedná se především o přesný přepis daného textu a také o získání pozic jednotlivých písmen v rámci vygenerovaného obrázku.

Při experimentování s navrženou neuronovou sítí byly použity tři datové sady. První dvě datové sady byly vygenerovány pomocí implementovaného generátoru a třetí byla ručně anotovaná sada vycházející z dostupných internetových zdrojů. Z dosažených výsledků vyplynulo, že nerekurentní síť dosahovala lepších výsledků než rekurentní síť. Z dalších experimentů vyplynulo, že zapojením Spatial Transformer Network do znakové větve se zvýší úspěšnost přibližně o dvě procenta, a také, že propojením znakové a poziční větve může docházet k lepším výsledkům přibližně o jedno procento.

V posledním experimentu byla také zarovnána historická datová sada, na níž byly následně neuronové sítě dotrénovány. Díky tomuto dotrénování se úspěšnosti zvýšily o 12.3% na 82.7%, respektive o 7.2% na 89.0%, avšak stále byla úspěšnější nerekurentní neuronová síť.

V rámci další práce by mohly být vyzkoušeny například obousměrné LSTM, případně by součástí vstupu navržené neuronové sítě mohl být také výstup sémantické segmentace. Pro tyto účely byl uzpůsoben také generátor umělých historických textů, který dokáže vytvářet odpovídající obrázky pro sémantickou segmentaci.

Literatura

[1] *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing, Kyoto, Japan, November 10-11, 2017*, ACM, 2017, ISBN 978-1-4503-5390-8.

[2] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, software available from tensorflow.org.

[3] Al-Rfou, R.; Alain, G.; Almahairi, A.; aj.: Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, ročník abs/1605.02688, Květen 2016.

[4] American National Corpus Project: The Open American National Corpus. 2018.
URL http://www.anc.org/

[5] Breuel, T. M.; Ul-Hasan, A.; Al-Azawi, M. A.; aj.: High-Performance OCR for Printed English and Fraktur Using LSTM Networks. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ICDAR '13, Washington, DC, USA: IEEE Computer Society, 2013, ISBN 978-0-7695-4999-6, s. 683–687, doi:10.1109/ICDAR.2013.140.
URL https://doi.org/10.1109/ICDAR.2013.140

[6] Britz, D.: Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients. 2016.
URL http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

[7] Cho, K.; van Merrienboer, B.; Gulcehre, C.; aj.: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, s. 1724–1734.

[8] Chollet, F.; aj.: Keras. https://keras.io, 2015.

[9] Chung, J.; Gulcehre, C.; Cho, K.; aj.: *Empirical evaluation of gated recurrent neural networks on sequence modeling*. 2014.

[10] Digitisation.eu: Impact Centre of Competence dataset. 2017.

[11] Fischer, A.; Keller, A.; Frinken, V.; aj.: Lexicon-free Handwritten Word Spotting Using Character HMMs. *Pattern Recogn. Lett.*, ročník 33, č. 7, Květen 2012: s. 934–942, ISSN 0167-8655.

[12] Graves, A.; Fernández, S.; Gomez, F.; aj.: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, New York, NY, USA: ACM, 2006, ISBN 1-59593-383-2, s. 369–376, doi:10.1145/1143844.1143891.
URL http://doi.acm.org/10.1145/1143844.1143891

[13] Haldar, R.; Mukhopadhyay, D.: Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach. *CoRR*, 2011.

[14] Hochreiter, S.; Schmidhuber, J.: Long Short-Term Memory. *Neural Comput.*, ročník 9, č. 8, Listopad 1997: s. 1735–1780, ISSN 0899-7667.

[15] Jaderberg, M.; Simonyan, K.; Zisserman, A.; aj.: Spatial Transformer Networks. In *Advances in Neural Information Processing Systems 28*, editace C. Cortes; N. D. Lawrence; D. D. Lee; M. Sugiyama; R. Garnett, Curran Associates, Inc., 2015, s. 2017–2025.

[16] Jurafsky, D.: Minimum Edit Distance. 2016.
URL https://web.stanford.edu/class/cs124/lec/med.pdf

[17] Kašpar, J.: *Soubor statí o novověkém písmu*. Univerzita Karlova, 1993, ISBN 80-7066-679-X.

[18] King James Bible Online: OFFICIAL KING JAMES BIBLE ONLINE: AUTHORIZED KING JAMES VERSION (KJV). 2018.
URL https://www.kingjamesbibleonline.org/

[19] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger, Curran Associates, Inc., 2012, s. 1097–1105.

[20] Lüdeling, A.; Odebrecht, C.; Zeldes, A.: *RIDGES-Herbology (Version 7.0)*. Humboldt-Universität zu Berlin.

[21] Olah, C.: Understanding LSTM Networks. 2015.
URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[22] Pascanu, R.; Mikolov, T.; Bengio, Y.: On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, ročník 28, editace S. Dasgupta; D. McAllester, Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, s. 1310–1318.

[23] Pham, V.; Kermorvant, C.; Louradour, J.: Dropout improves Recurrent Neural Networks for Handwriting Recognition. *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[24] 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 01, Nov 2017, s. 67–72, doi:10.1109/ICDAR.2017.20.

[25] Schimke, S.; Vielhauer, C.; Dittmann, J.: Using adapted Levenshtein distance for on-line signature authentication. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, ročník 2, Aug 2004, ISSN 1051-4651, s. 931–934 Vol.2, doi:10.1109/ICPR.2004.1334412.

[26] Seide, F.; Agarwal, A.: CNTK: Microsoft's Open-Source Deep-Learning Toolkit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4232-2, s. 2135–2135, doi:10.1145/2939672.2945397.

[27] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, č. 11, Nov 2017: s. 2298–2304, ISSN 0162-8828, doi:10.1109/TPAMI.2016.2646371.

[28] Shi, B.; Wang, X.; Lyu, P.; aj.: Robust Scene Text Recognition With Automatic Rectification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[29] Springmann, U.; Lüdeling, A.: *OCR of historical printings with an application to building diachronic corpora: A case study using the RIDGES herbal corpus*. Únor 2017.

[30] Su, B.; Lu, S.: Accurate recognition of words in scenes without character segmentation using recurrent neural network. *Pattern Recognition*, ročník 63, č. Supplement C, 2017: s. 397 – 405, ISSN 0031-3203, doi:https://doi.org/10.1016/j.patcog.2016.10.016.

[31] Wikipedia: Antikva. 2018.
URL https://cs.wikipedia.org/wiki/Antikva

[32] Wikipedia: Fraktur. 2018.
URL https://en.wikipedia.org/wiki/Fraktur

[33] Wikipedia: King James Version. 2018.
URL https://en.wikipedia.org/wiki/King_James_Version

[34] Wikipedia: Kurrent. 2018.
URL https://en.wikipedia.org/wiki/Kurrent

[35] Wikipedia: Latinka. 2018.
URL https://cs.wikipedia.org/wiki/Latinka

[36] Wikipedia: Levenshtein Distance. 2018.
URL https://en.wikipedia.org/wiki/Levenshtein_distance

[37] Wikipedia: Schwabacher. 2018.
URL https://en.wikipedia.org/wiki/Schwabacher

Příloha A

Konfigurační soubor

Ukázka konfiguračního souboru pro generátor umělých historických textů:

```
[Common]
Input: input_corpus.txt
Outputs: OutputsPages/
Backgrounds: Backgrounds/
Fonts: Fonts/
Annotations: True

[Text]
Words: False
ToLowercase: False
FirstUppercase: 0.0
PunctuationAfterWord: 0.0
Punctuations: .,-?!

[Page]
Width: 500
Padding: 50
NumberOfLines: 5
MinWordSpace: 4
MinLineHeight: 32
MaxLineHeight: 48
LineSpaceMean: 5.0
LineSpaceSigma: 0.5

[BackText]
Probability: 0.5
MinAlpha: 0.05
MaxAlpha: 0.4

[PrintingImperfections]
MinFreq: 150
MaxFreq: 500
MaxColor: 255
SubtractMin: True
MaxBlobs: 500
MinBlobColor: 64
MaxBlobColor: 192

[Blurring]
MinSigmaX: 0.5
MaxSigmaX: 10.0
```

```
MinFreq: 150
MaxFreq: 500
MaxColor: 160

[Transformations]
RotationMean: 0.0
RotationSigma: 0.01
TranslationMean: 0.7
TranslationSignma: 0.7
```

Příloha B

Obsah DVD

Obsah přiloženého DVD:

- `datasets/` - adresář s datovými sadami pro trénování a testování neuronových sítí
  - `AlignedDataset/` - zarovnaná reálná historická datová sada
  - `ArtificialLines/` - řádky vygenerovaného textu
  - `HistoricalLines/` - řádky historických textu
  - `Sequences/` - trénovací sada s výřezy
  - `Sequences2/` - testovací sada s výřezy
- `source_codes/` - adresář se zdrojovými kódy
  - `generator/` - skripty potřebné pro generování umělých datových sad
  - `tools/` - ostatní skripty (trénování a testování sítí, zarovnání datové sady, ...)
- `text/` - textová část práce
- `video/` - adresář s videem