VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ RUČNĚ PSANÉHO TEXTU POMOCÍ KONVOLUČNÍCH SÍTÍ

CONVOLUTIONAL NETWORKS FOR HANDWRITING RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN SLADKÝ**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání bakalářské práce

Student: **Sladký Jan**
Program: Informační technologie
Název: **Rozpoznávání ručně psaného textu pomocí konvolučních sítí**
Convolutional Networks for Handwriting Recognition
Kategorie: Zpracování obrazu

Zadání:

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

Literatura:

- Shi, Baoguang et al: An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. PAMI, 2015.
- Kang et al.: Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. GCPR, 2019.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

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

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 6. listopadu 2019

Abstrakt

Tato práce se zabývá rozpoznáváním ručně psaného textu za pomoci konvolučních neuronových sítí. Ze současných metod byl vybrán model sítě skládající se z konvolučních a rekurentních sítí s *Connectionist Temporal Classification*. Do takovéhoto modelu byl následně implementován prvek *Vertical Attention Module*, který vybírá relevantní informace v každém sloupci odpovídající textu na obrázku. Tento modul byl následně pomocí experimentů porovnáván s dalšími možnostmi vertikální agregace mezi konvoluční a rekurentní sítí. Experimenty probíhaly na datové sadě obsahující přes 80 000 řádků textu z českých dopisů 20. století. Výsledky ukazují, že *Vertical Attention Module* dosahuje téměř vždy nejlepších výsledků na všech použitých typech konvolučních sítí. Výsledná síť dosáhla nejlepšího výsledku při chybě 8,9 % na znak. Přínosem této práce je neuronová síť s nově zavedeným prvkem, která dokáže rozpoznávat řádky textu.

Abstract

This thesis deals with handwriting recognition using convolutional neural networks. From the current methods, a network model was chosen to consist of convolutional and recurrent neural networks with the Connectist Temporal Classification. The Vertical Attention Module, which selects the relevant information in each column corresponding to the text in the figure was subsequently implemented in such a model. Then, this module was compared with other possibilities of vertical aggregation between convolutional and recurrent networks. The experiments took place on a data set containing over 80,000 lines of text from Czech letters from the 20th century. The results show that the Vertical Attention Module almost always achieves the best results on all used types of convolution networks. The resulting network achieved the best result with 8,9 % of the character error rate. The contribution of this work is a neural network with a newly introduced element that can recognize lines of text.

Klíčová slova

Rozpoznávání ručně psaného textu, HTR, konvoluční neuronové sítě, CNN, rekurentní neuronové sítě, RNN, Connectist Temporal Classificatio, CTC, Vertical Attention Module, VAM, vertikální agregace

Keywords

Handwritten text recognition, HTR, convolutional neural network, CNN, recurrent neural network, RNN, Connectist Temporal Classificatio, CTC, Vertical Attention Module, VAM, vertical aggregation

Citace

SLADKÝ, Jan. *Rozpoznávání ručně psaného textu pomocí konvolučních sítí.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Rozpoznávání ručně psaného textu pomocí konvolučních sítí

Prohlášení

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

.....
Jan Sladký
31. července 2020

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Michalovi Hradišovi, Ph.D, za cenné rady, odborné vedení, ochotu, trpělivost a čas strávený při vypracovávání této práce. Další poděkování patří výpočetnímu centru MetaCentrum za poskytnutí přístupu k výpočetním a úložným zařízením. Tato práce vznikla za podpory projektu „e-Infrastruktura CZ“ (e-INFRA LM2018140) poskytovaného v rámci programu "Projekty Velkého Výzkumu, Vývoje a Inovativních Infrastruktur".

Obsah

Kapitola 1

Úvod

Rozpoznávání ručně psaného textu (*Handwritten Text Recognition*, HTR) je náročný proces spadající do oblasti počítačového vidění, který je zkoumán již několik let. Podstatou HTR je transformovat ručně napsaný text v obrazu do digitální podoby. Transformace textu do digitální podoby je velmi užitečná v mnoha oblastech a přináší velké výhody. Mezi hlavní výhody můžeme zařadit například uložení textu, řazení textu, vyhledávání textu a tak podobně. V současné době se tato oblast rozrůstá a HTR se rozšiřuje i mezi každodenní součást dnešní společnosti. Příkladem mohou být různé mobilní aplikace využívající kameru pro rozpoznání textu.

V posledních pár letech rozpoznávání textu dosahuje poměrně dobrých výsledků a mohlo by se zdát, že je tato vědní oblast vyřešena a není potřeba dalšího výzkumu. V některých případech je ale stále prostor pro zlepšení. Obzvláště v případech, kdy vstupní data obsahují špatnou kvalitu (např. šum v obrázku, staré a špatně zachovalé dokumenty), při zpracování méně používaných fontů nebo písma psaného kurzívou.

Tato práce se zaměřuje na vylepšení neuronové sítě použitelné pro rozpoznání ručně psaného textu. Do neuronové sítě byl implementován prvek zvaný *Vertical Attention Module* a díky experimentům s touto sítí byly zjišťovány výhody či nevýhody tohoto modulu. Neuronová síť zpracovává vyřezané řádky textu a rozpoznává, co je na tomto řádku napsáno.

Protože je pro člověka psaný text velmi důležitý, stalo se jeho rozpoznávání jednou z prvních oblastí ve výzkumu umělé inteligence. Díky tomu existuje více přístupů k tomuto problému. Proto jsou v kapitole 2 popsány metody, které se v této vědní disciplíně používají a do větší hloubky jsou popsány dva dnes asi nejvíce používané modely sítí v této oblasti. V další kapitole 3 jsou představeny možné datové sady použitelné pro trénování, experimentování a testování HTR pracující s řádky textu a dále jsou zde uvedeny konference konané na toto téma. Navazující kapitola 4 detailněji rozebírá hlavní prvky, které obsahují implementované architektury. Rozšiřující prvek této sítě je popsán v kapitole 5. Kapitola 6 obsahuje detaily navržených architektur. Prováděné experimenty jsou představeny v kapitole 7. Závěr 8 obsahuje zhodnocení dosažených výsledků spolu s návrhem pokračování budoucího vývoje.

Kapitola 2

Rozpoznávání textu

HTR v dnešní době dosahuje velmi dobrých výsledků. Pro odlišné a individuální psací techniky se jedná o náročný problém, který je stále předmětem výzkumu. Vytvořit systém, který je přesný, efektivní a dokáže se přizpůsobit ve všech situacích, je obtížné.

V poslední době jsou za tímto účelem nejvíce využívány neuronové sítě, které přinesly velké zlepšení a dosahují nejlepších výsledků. Už v letech 1998 *Lecun a spol.* [27] dokázali, že klasické modely, které využívaly algoritmy pracující s „ruční“ (*hand-crafted*) extrakcí prvků z obrazu, mohou být překonány a výhodně zaměněny s modely neuronových sítí pracujících přímo s pixely obrazu, které mají schopnost „učit se“. Tyto modely se dokáží učit díky tzv. *gradient-base* učení, které se snaží během učení minimalizovat chybovost výpočtu sítě na základě přepočtu vah v síti, což umožňuje zvyšovat přesnost modelu. *Lecun a spol.* v tomto článku představili tzv. *multi-module machine learning* modely, které se dají považovat za předchůdce dnešních *deep learning* modelů.

V této kapitole jsou představeny metody využitelné v této oblasti a následně jsou podrobněji popsány dvě architektury neuronových sítí, které představují nejmodernější přístup k tomuto problému, tzv. *state-of-the-art*.

2.1 Současné metody HTR

Jeden z přístupů, který zaznamenal úspěch v přesnosti HTR, je použití *Hidden Markov Model* (HMM) [5]. Hlavní nevýhodou tohoto přístupu je nutná „ruční“ extrakce prvků z obrazu, která vyžaduje prvotřídní znalost jazyka a nedokáže pokrýt rozmanitosti a složitosti rukopisu. Se stále se zvětšující popularitou neuronových sítí se začaly HMM a neuronové sítě společně kombinovat a dosáhlo se tak lepších výsledků. Například *Bluche a spol.* [4] navrhli metodu, která kombinuje HMM a konvoluční neuronové sítě (*Convolutional Neural Networks* , CNN).

Velkým přínosem pro rozpoznávání textu byly právě konvoluční sítě. *Lecun a spol.* v [27] představili první konvoluční síť nazvanou *LeNet-5*, která dokázala rozpoznávat čísla z databáze MNIST[1] s obrázky čísel. Konvoluční sítě však neumí pracovat se sekvencí znaků. Jedním z možných přístupů k tomuto problému může být nejprve detekovat jednotlivé znaky z obrazu a poté tyto znaky rozpoznávat jednotlivě [38]. Dalším možným přístupem je zacházet s textem v obrazu jako s klasifikačním problémem, kde výstupní třídy z konvoluce reprezentují jednotlivá slova, například v článku [22] takto pracovali s 90 tisíci slovy. To se ale ukázalo jako nepraktické, protože tyto modely obsahují velké množství tříd, které je

[1] http://yann.lecun.com/exdb/mnist/

obtížné zobecnit a použít například na jiný jazyk. Z tohoto vyplývá, že samotné konvoluční sítě nejsou vhodné na rozpoznávání sekvencí v obraze.

Dalším důležitým typem neuronových sítí na rozpoznávání sekvencí jsou rekurentní neuronové sítě (*Reccurent Neural Network*, RNN). Rekurentní sítě jsou navrženy právě pro zpracování sekvencí, což z nich dělá nedílnou součást dnešních rozpoznávacích modelů. Velkou výhodou RNN je, že nepotřebují znát pozici jednotlivých elementů v sekvenci. Před vložením sekvence do RNN je ale zapotřebí obraz nejprve předzpracovat do sekvence příznaků obrazu (*image feature*) a po vyhodnocení RNN této sekvence je nezbytné následné zpracování výstupu za účelem získání sekvence znaků. Tento problém vyřešili *Graves a spol.* se svou sítí [12], kde použili RNN zvanou *Long Short-Term Memory* (LSTM) [18], ke které připojili *Connectionist Temporal Classification* (CTC)(viz sekce 4.1). Tuto síť následně vylepšili nahrazením LSTM za obousměrnou LSTM (*Bidirectional Long Short-Term Memory*, BLSTM) [10]. Tato síť překonala doposud nejpoužívanější modely vycházející z HMM. Dalším rozšířením takovýchto sítí je použití CNN na extrakci příznaků z obrazu a následné připojení k RNN-CTC síti. *Krishnan a spol.* [26] ještě rozšířili takovouto síť o *Spatial Transformer Networks* (STN) [23], která umožňuje napravovat geometrické transformace znaků ve vstupním obrazu. Dalším typem sítí jsou sítě založené na principu kodér-dekodér (*encoder-decoder*) [24, 29], které se velmi hodí na převod jedné sekvence na jinou sekvenci, což je právě problém, který je řešen u rozpoznávání textu z obrazu.

Sítě typu CNN-RNN-CTC a kodér-dekodér jsou v současnosti považovány za *state-of-the-art* v oblasti HTR a v této kapitole jsou dále podrobněji popsány [24].

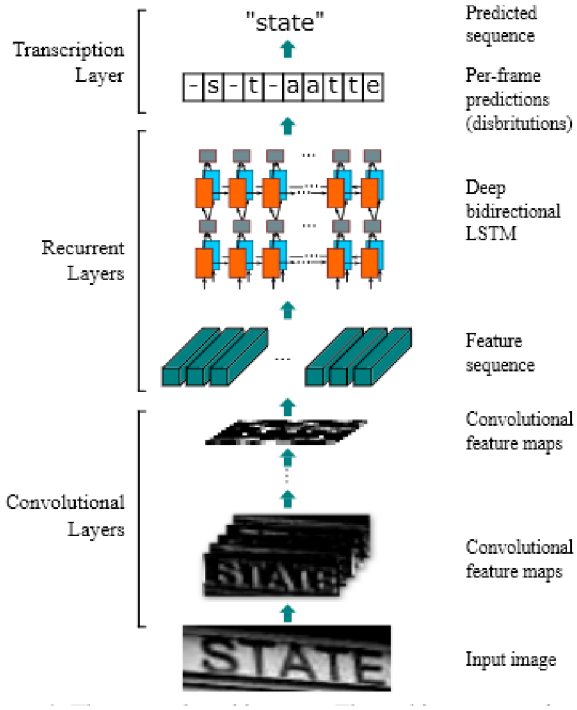
2.2 Model skládající se z CNN, RNN a CTC

Tento model představili *Shi a spol.* [35]. Model se skládá z konvolučních vrstev, rekurentních vrstev a vrstvy pro přepis sekvence výstupu z rekurentní sítě do sekvence znaků (písmen).

Nejprve CNN extrahuje mapu příznaků ze vstupního obrázku. Po extrahování příznaků přichází na řadu RNN, která vytváří predikce pro každý rámec z map příznaků (*feature maps*). Vrstva pro přepis sekvence převádí předpovědi jednotlivých výstupů z RNN do sekvence znaků. V tomto modelu je použito více rozdílných vrstev neuronových sítí, ale model může být stále jednotně trénován a být na něm použita jedna ztrátová/objektivní funkce (*loss function*). Ukázka této sítě je na obrázku 2.1

Extrakce mapy příznaků. V tomto modelu jsou konvoluční vrstvy tvořeny z konvoluce, pooling vrstvy a nelineární funkce. Tyto části jsou převzaty z CNN, kde plně propojené vrstvy (*fully-connected layers*) jsou vynechány. Tyto vrstvy vytvářejí sekvenci příznaků ze vstupního obrazu. Před tímto zpracováním musí být všechny obrázky zarovnány na stejnou velikost. Z map příznaků jsou následně extrahovány sekvence jednotlivých rysových vektorů, které jsou vstupem do RNN. Konkrétně každý takovýto vektor je generován po sloupcích zleva doprava z map příznaků. To znamená, že i-tý vektor je konkatenace několika sloupců ze všech map příznaků.

Konvoluční vrstvy, pooling vrstvy a nelineární aktivační funkce pracují s lokálními pozicemi, což vede k tomu, že jsou tzv. *translation invariant* neboli jsou schopny pracovat s různými posuny v obraze. Proto každý sloupec mapy příznaků odpovídá jednotlivým obdélníkovým částem ze vstupního obrazu (označovány jako *receptive field*). Tyto obdélníky jsou ve stejném pořadí zleva doprava jako sloupce z map příznaků. Tento vztah je ukázán na obrázku 2.2.

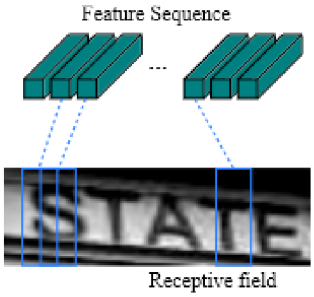


Obrázek 2.1: Architektura sítě skládající se z CNN, RNN a CTC. Sít obsahuje konvoluční vrstvy (*Convolutional layers*), rekurentní vrstvy (*Recurrent layers*) a vrstvu pro přepis (*Transcription layer*). Převzato z [35].

Rekurentní část modelu. Jako rekurentní vrstva je zde implementována obousměrná rekurentní neuronová síť (*Bidirectional recurrent neural networks*), která je připojena hned na výstup konvoluční vrstvy. Tato vrstva se snaží vyhodnocovat znaky y_t z jednotlivých příznakových vektorů x_t posloupnosti vlastností $\mathbf{x} = x_1, \ldots, x_T$, které obdržela od konvoluční vrstvy. Výhodou rekurentní sítě oproti jiným neuronovým sítím je taková, že rekurentní síť pracuje jak se vstupem, tak i s vnitřním stavem neboli jejím předchozím výstupem. To umožňuje uchování kontextu, a tím se lépe hodí pro řešení sekvenčních problémů oproti rozpoznávání každého znaku samostatně. Některé znaky mohou být roztaženy přes více *receptive field* nebo si znaky mohou být podobné a rozpoznání v kontextu může být snadnější než je rozpoznávat samostatně. Další z výhod je zpětná propagace chyby na vlastní vstup. Tím se mohou spojit společně s konvoluční vrstvou a tvořit tak jednotnou síť, která se dokáže společně učit. Možnost pracovat se sekvencí jakékoliv délky patří také mezi velké výhody RNN.

Nejčastěji se v takovýchto modelech vyskytují dva typy rekurentních sítí. Jednou z nich je *Long-Short Term Memory* (LSTM) [18] a druhou používanou je *Gated Recurrent Unit* (GRU) [6]. Důvodem používání těchto sítí namísto klasických rekurentních sítí je problém zvaný *vanishing gradient problem*, kterým trpí právě klasické rekurentní sítě. Obě sítě, jak GRU, tak LSTM mohou fungovat v dopředném i ve zpětném směru. Proto bývá využíváno obousměrných (*bidirectional*) sítí. Takovéto sítě dosahují lepší abstrakce a lepších výsledků oproti jednosměrné síti.

Přepis sekvence na znaky pomocí CTC. Přepis je proces, kde se jednotlivé výstupy z RNN transformují na znaky. Při přepisu jde o nalezení sekvence znaků, které odpovídají

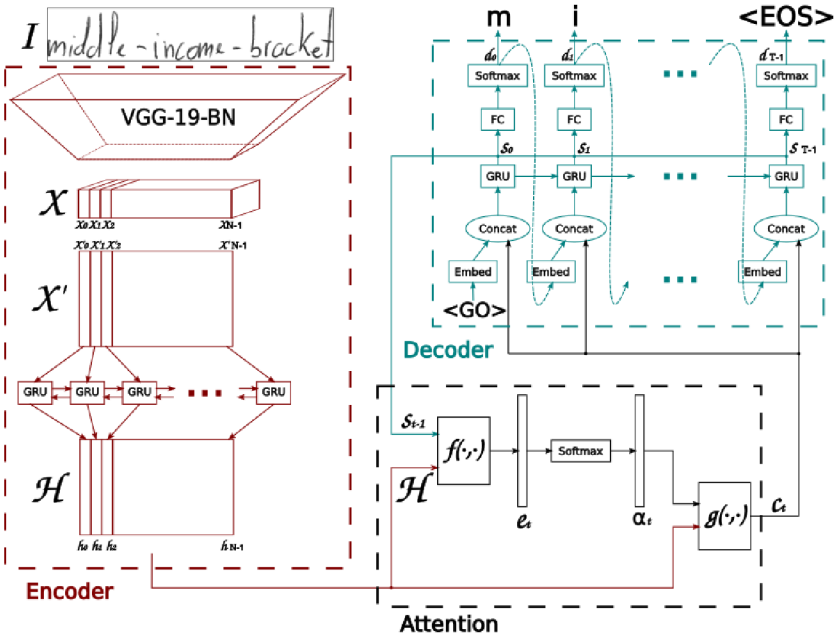


Obrázek 2.2: Na obrázku je vyobrazen vztah mezi sloupci z map příznaků a odpovídajícími obdélníky (označovány jako *receptive field*) vstupního obrazu. Převzato z [35].

největší pravděpodobnosti na základě výstupu z RNN. K tomuto účelu je využíván tzv. *Connectionist temporal classification* (CTC), který je popsán v samostatné kapitole 4.1.

2.3 Encoder-Decoder model

Cílem tohoto modelu [24, 29] je oddělit dekódování a získávání příznaků z obrazu. Model se skládá ze dvou hlavních částí, těmi jsou kodér a dekodér. Kodér čte vstupní sekvence a vytváří z nich sekvence příznaků. Dekodér vytváří výstupní sekvenci postupně po jednom znaku. Mezi těmito dvěma hlavními částmi bývá umístěn tzv. *attention* mechanismus, který umožňuje dekodéru se soustředit na nejdůležitější prvky z kodéru v každém dekódovacím časovém kroku. Architektura této sítě je na obrázku 2.3. Dále si blíže popíšeme tyto tři části.



Obrázek 2.3: Architektura modelu Encoder-Decoder, který se skládá z kodéru a dekodéru a mechanismu, který se nazývá *attention*. Převzato z [24]

Enkodér. Enkodér je sestaven ze dvou částí. Jednou z nich je CNN, která získává vizuální vlastnosti obrázku. CNN převádí obrázky textových řádků s proměnnou délkou a pevnou výškou na sekvenci vizuálních příznakových vektorů. *Kang a spol.* v článku [24] věří, že obrázky s ručně napsaným textem nejsou tak komplexní jako obrázky z reálného světa, a proto zvolili jako CNN architekturu VGG-19-BN [36] s předtrénovanými váhami. Druhá část enkodéru je tvořena z RNN. Zatímco v článku [24] pracují s BGRU, v článku [29] využili BLSTM, které obsahují dvě LSTM, kde každá pracuje v opačném směru té druhé, aby síť zpracovala posloupnost v obou směrech a kódovala tak závislosti v dopředném i zpětném směru a zachytila tak přirozený vztah ručně psaného textu. V jejich modelu kodér obsahuje tři takové BLSTM vrstvy. Konečný výstup kodéru v této architektuře vytváří 1D konvoluce přes všechny stavy výstupní sekvence z BLSTM.

Formálně CNN zpracuje vstupní obrázek $\mathcal{I}$ do mapy příznaků $\mathcal{X}$. Tuto mapu příznaků lze chápat jako sekvenci sloupců $\mathcal{X} = (x_1, \ldots, x_M)$, kde M je délka podvzorkované vstupní sekvence (neboli šířka mapy příznaků). Takováto sekvence je poté zpracována RNN (nejčastěji BGRU a BLSTM), která vytváří výstup $\mathcal{H}$ se stejnou šířku jako $\mathcal{X}$. Každý element $h_i \in \mathcal{H}$ je výstupem RNN v jednotlivých časových krocích. Tento výstup je dále použit na výpočet *attention*.

Attention mechanism. *Attention mechanism* je prvek mezi kodérem a dekodérem. Každou časovou jednotku modifikuje tzv. *context vector* na základě podobností dekodérova stavu s_t a kodérova výstupní mapy příznaků $\mathcal{H}$. Tím najde nejvíce relevantní vektory, na které se má zaměřit. Tyto vektory jsou poté použity na predikci znaku v daném kroku. *Context vector* c_t v čase t je získán jako vážený součet přes vektory příznaků z kodéra.

$$c_t = \sum_{j=1}^{M} \alpha_{t,j} h_j. \tag{2.1}$$

Kde $\alpha_{t,j}$ značí *attention* váhy a lze je spočítat více způsoby, záleží jaká funkce je použita.

$$\alpha_{t,j} = Att(s_t, h_j, \alpha_{t-1}). \tag{2.2}$$

Tímto způsobem se dekodér učí vzájemné vztahy mezi vstupní a výstupní sekvencí v souvislosti s celkovým kontextem.

Kang a spol. v článku [24] uvádí dva způsoby *attention*. Tím jsou *Content-based Attention* a *Location-based Attention*, které jsou dále popsány.

Content-based Attention. Jde o základní mechanismus *attention*. Mějme α_t jako masku pozornosti v čase t, h_i jako *hidden state* (uložený stav) kodéra v čase $i \in \{0, 1, \ldots, N-1\}$ a s_t jako *hidden state* dekodéra v čase $t \in \{0, 1, \ldots, T-1\}$ kde T je maximální délka dekódovacího znaku. Pak

$$\alpha_t = \text{Softmax}(e_t). \tag{2.3}$$

kde

$$e_{t,i} = f(h_i, s_{t-1}) = w^T \tanh(W h_i + V s_{t-1} + b). \tag{2.4}$$

kde w, W, V a b jsou trénovatelné parametry. Velkou nevýhodou této *attention* je nemožnost rozpoznání stejných prvků v sekvenci, které se nacházejí na jiných pozicích, protože jsou ohodnoceny stejnou váhou. To znamená, že dekodér není schopen rozpoznat mezi více příznaky jednoho znaku na více pozicích.

Location-based Attention. Tato metoda *attention* řeší nevýhodu, kterou měla metoda *Content-based Attention*. Problém je řešen tím, že se informace o poloze explicitně zakóduje. *Content-based* se tedy rozšíří tak, aby zohledňoval polohu znaku. Toho lze docílit, pokud vezme v úvahu váhy vytvořené v předchozím kroku. Z předchozího zarovnání α_{t-1} nejprve extrahuje k vektorů $l_{t,i} \in \mathbb{R}^k$ na každé pozici i, provedením konvoluce s maticí $F \in \mathbb{R}^{k \times r}$:

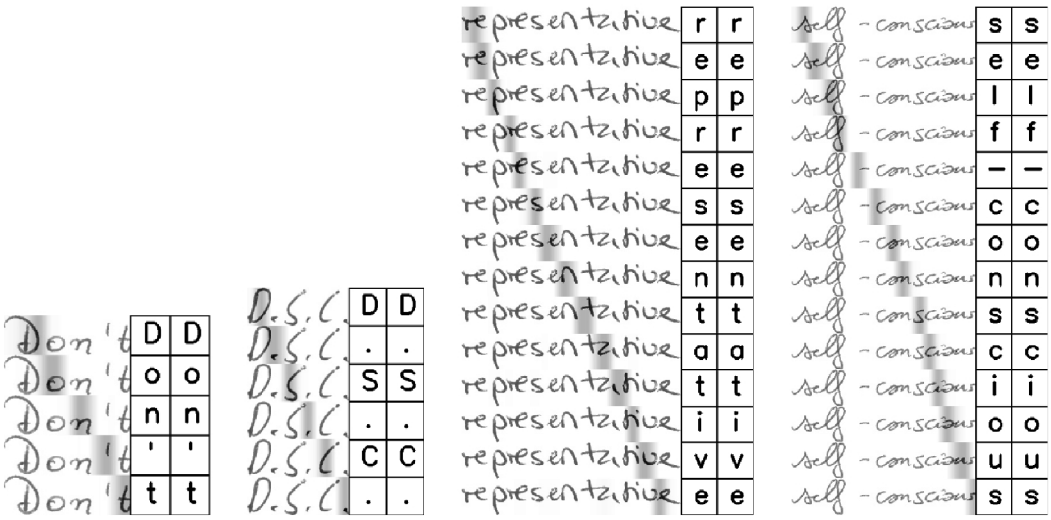
$$l_t = F * \alpha_{t-1} \tag{2.5}$$

Následně se zamění funkce 2.4 za funkci:

$$e_{t,i} = f'(h_i, s_{t-1}, l_t) = w^T \tanh(Wh_i + Vs_{t-1} + Ul_{t,i} + b). \tag{2.6}$$

kde w, W, V, U a b jsou trénovatelné parametry.

Toto byly dva hlavní typy *attention* mechanismu. *Michael a spol.* [29] v článku představují další způsoby *attention*, kterými jsou: *Penalized attention, Monotonic attention, Chunkwise attention a Positional encoding*. Schopnost zaměření *attention* na relevantní část sekvence je zobrazena na obrázku 2.4



Obrázek 2.4: Ukázka správného zarovnání mechanismu *attention*, kdy se v každém časovém kroku zaměří na správné místo ze sekvence v obrazu. Převzato z [24].

Dekodér. Dekodér je implementován pomocí jednosměrných RNN, GRU nebo LSTM. V článku [29] je sestaven z jednosměrné LSTM vrstvy. Cílem dekodéru je generovat cílovou sekvenci znaků nacházejících se v obrazu. V každém čase t dekodér počítá rozdělení pravděpodobnosti jednotlivých znaků abecedy a určuje nejvíce pravděpodobný znak y_t. Ten závisí na předcházejících předpovědích $(y_1, \ldots, y_{t-1})$ a na *context* vektoru c_t, který obsahuje informace kódovaných příznaků $\mathcal{H}$. Dekodér v podstatě definuje pravděpodobnost výstupní sekvence $\mathcal{Y} = (y_1, \ldots, y_T)$ rozkladem pravděpodobnosti na dané podmínky $p(\mathcal{Y}) = \prod_{t=1}^{T} p(y_t|y_1, \ldots, y_{t-1}, c_t)$, kde T je délka výstupní sekvence. Podmínky $(y_1, \ldots, y_{t-2})$ jsou vytvořeny jako $p(y_t|y_1, \ldots, y_{t-1}, c_t) = \text{softmax}(f(y_{t-1}, s_{t-1}, c_t))$, kde f reprezentuje LSTM a s_{t-1} předchozí *hidden state*. Dekodér shromažďuje informace o historii výstupní sekvence a ukládá ji do vnitřního paměťového stavu.

Kapitola 3

Hodnocení

Důležitým aspektem ve vývoji rozpoznávání ručně psaného textu je dostupnost vzorových a dobře anotovaných dat. Dalším klíčovým aspektem pro rozvoj v tomto odvětví je možnost diskutování různých přístupů a potřeba konkurence. Srovnání různých přístupů je proveditelné pouze na stejných datových sadách. Pro tyto účely existují volně dostupné datové sady na natrénování a následné testování sítě HTR. V této kapitole jsou představeny volně dostupné datové sady spolu s konferencemi konané na toto téma a soutěžemi, které se na těchto konferencích konají.

3.1 Datové sady

Datové sady obsahují obrázky s textem a ke každému obrázku přiřazený text, který odpovídá textu na obrázku tzv. *ground truth*. Protože se tato bakalářská práce věnuje rozpoznávání ručně psaného textu pracujícího s řádkem textu, jsou zde uvedeny datové sady, které obsahují obrázky ručně napsaných textů formou řádků, na kterých se vyskytuje více slov. Všechny představené datové sady obsahují *ground truth* a meta-informace ve formátu PAGE XML [30]. *Ground truth* je pro jednotlivé řádky také sepsané v textovém formátu pro jednodušší zacházení a jsou rozděleny na trénovací, validační a testovací sady. Ukázka řádků textu jednotlivých datových sad je na obrázku 3.1

Bentham collection. *Bentham collection* [9, 37] jedná se o rukopisy napsané anglickým filozofem Jeremy Bentham (1748-1832) a kopie psané jeho sekretariátem. Přepis této sbírky provádějí dobrovolníci na webové platformě Transcribe Bentham[1] . V současné době bylo pomocí této veřejné webové platformy přepsáno více než 6000 dokumentů.

IAM Handwriting Database. *IAM Handwriting Database* [28] obsahuje ručně psané anglické texty, které byly naskenovány v rozlišení 300 dpi a uloženy jako PNG obrázky s 256 úrovněmi šedé. Databáze zahrnuje okolo 1066 formulářů, napsaných přibližně 400 různými spisovateli, obsahující 82227 slov ze slovníka, který se skládá z 10841 různých slov. Všechny formuláře a také všechny extrahované textové řádky, slova a věty jsou poskytnuty v souboru PNG. *Ground truth* je v textovém formátu, kde slova na jednomu řádku jsou od sebe oddělena znakem „|“.

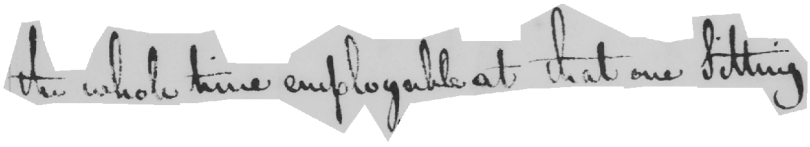
[1] http://transcribe-bentham.ucl.ac.uk/td/Transcribe_Bentham

The RIMES database. *The RIMES database* [13, 14] byl vytvořen za účelem vyhodnocování automatických systémů pro rozpoznávání ručně psaných dopisů. Jedná se o texty napsané ve francouzském jazyce, které jsou zaslané poštou nebo faxem. Databáze byla vytvořena požádáním dobrovolníků o napsání ručně psaných dopisů výměnou za dárkové poukázky. Následně byla anotována a publikována. Pro získání datové sady stačí vyplnit dohodu o nekomerčním použití.

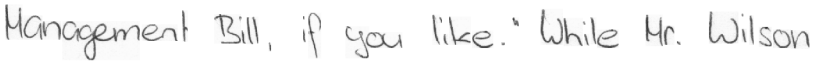
České dopisy. Dataset České dopisy[2] [17] je sestaven z 2000 osobních dopisů z celého 20. století, které byly vybrány z existujících sad přepsaných českých dokumentů. Tyto dokumenty shromáždil a přepsal kolektiv Zdeny Hladké na Masarykově univerzitě. Dopisy jsou napsány 2000 unikátními autory, díky tomu dobře reprezentují písmo 20. století. Původním cílem této datové sady bylo vytvořit pouze jazykový korpus, a proto byly obrázky digitalizovány v nízkém rozlišení a výsledné řádky jsou v horší kvalitě. Z těchto dopisů bylo vyřezáno něco přes 80000 řádků textu.

Název	Počet řádků	Počet autorů
Bentham	11473	< 10
IAM	9862	500
RIMES	12058	> 1300
České dopisy	80000	> 2000

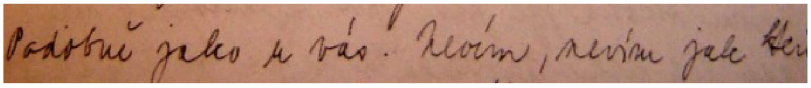
Tabulka 3.1: Tabulka datasetů ukazující počet řádků textu a počet autorů, těchto datasetů



(a) Bentham collection



(b) IAM Handwriting Database



(c) České dopisy

(d) The RIMES database [32]

Obrázek 3.1: Řádky textu datových sad.

[2]Datovou sadu České dopisy poskytl vedoucí práce Ing. Michal Hradiš, Ph.D.

3.2 Soutěže a konference

Jak bylo uvedeno, klíčovými aspekty pro rozvoj rozpoznávacího systému jsou diskuse na toto téma, konkurence a testování různých přístupů. K tomuto účelu slouží konference, na kterých se konají různé soutěže v této oblasti.

The International Conference on Frontiers of Handwriting Recognition (ICFHR). *ICFHR* [2] dříve nazývaný *International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, je vědecká konference v oblasti ručně psaného textu. Cílem této konference je shromáždit mezinárodní odborníky z akademické obce a průmyslu, aby se podělili o své zkušenosti a podpořili výzkum a vývoj ve všech aspektech rozpoznávání ručně psaného textu a aplikací. Například v roce 2014 při konání této konference byla vyhlášena soutěž na rozpoznání textu z uvedené datové sady *Bentham collection* 3.1 [37].

International Conference on Document Analysis and Recognition (ICDAR). *ICDAR* [1] je mezinárodní akademická konference, která se koná každé dva roky v jiném městě. Shromáždění konané pro vědce a odborníky v komunitě pro analýzu dokumentů. Stala se jednou z nejvíce důležitých mezinárodních konferencí, která pokrývá hlavní témata analýzy a rozpoznávání dokumentů, analýzy a ověřování rukopisu, detekce a zpracování textu a další související témata.

Kapitola 4

Konvolučně rekurentní sítě pro rozpoznávání textu

V této kapitole jsou detailněji popsány zajímavější vrstvy sítě, které jsou součástí vybraných architektur. Nejprve jsou zde popsány funkce a princip fungování CTC 4.1, následně navazuje popis *Gated* konvoluční síť 4.2 a nakonec princip normalizace vnitřních vrstev 4.3.

4.1 Connectionist temporal classification (CTC)

Jak už bylo zmíněno, při snaze rozpoznávání textu na řádku pracujeme se sekvencí dat na vstupu. Při práci se sekvencemi se nejvíce využívají RNN, které díky vnitřním stavům dokáží pracovat i s předchozími stavy a tím si uchovávají kontext. Problém je v tom, že standardní objektivní funkce neuronové sítě jsou definovány samostatně pro každý bod v tréninkové sekvenci. Tím pádem RNN mohou být trénovány pouze ke klasifikaci sekvence samostatných znaků. To znamená, že pro získání konečné sekvence znaků musí být data před vstupem do sítě segmentována a výstup musí být dodatečně zpracován. *Connectionist temporal classification* (CTC) [11, 12] řeší právě tento problém. Díky CTC lze použít RNN bez dalších segmentací trénovacích dat a dodatečného zpracování výstupu. Je také možné modelovat všechny aspekty sekvence v rámci jedné architektury sítě. Základní myšlenkou je interpretovat síťové výstupy jako rozdělení pravděpodobnosti pro všechny znaky, podmíněných vstupní sekvencí. S ohledem na toto rozdělení lze odvodit objektivní funkci, která přímo maximalizuje pravděpodobnost správného označení. Vzhledem k tomu, že objektivní funkce je diferencovatelná, může být síť trénována se standardním *gradient back-propagation* algoritmem [27].

Přes výstup sítě k sekvenci znaků. Je definována abeceda L, která obsahuje všechny znaky, které se mohou nacházet na výstupu. Výstupem sítě je funkce softmax s počtem výstupů větších o jeden než je znaků v abecedě L. Hodnota, tohoto elementu navíc určuje pravděpodobnost tzv. *blank* znaku nebo prázdného znaku. Hodnoty ostatních $|L|$ elementů určují pravděpodobnosti odpovídajících znaků v konkrétních časech. Společně takovéto výstupy definují pravděpodobnost všech možných způsobů zarovnání všech možných znakových sekvencí. Celková pravděpodobnost je součet pravděpodobností všech možných zarovnání.

Definujme si rozšířenou abecedu $L' = L \cup \{blank\}$, pak výstup ze sítě k v čase t y_k^t si lze vyložit jako pravděpodobnost, kde v daném čase t bude na výstupu ze sítě znak k z abecedy

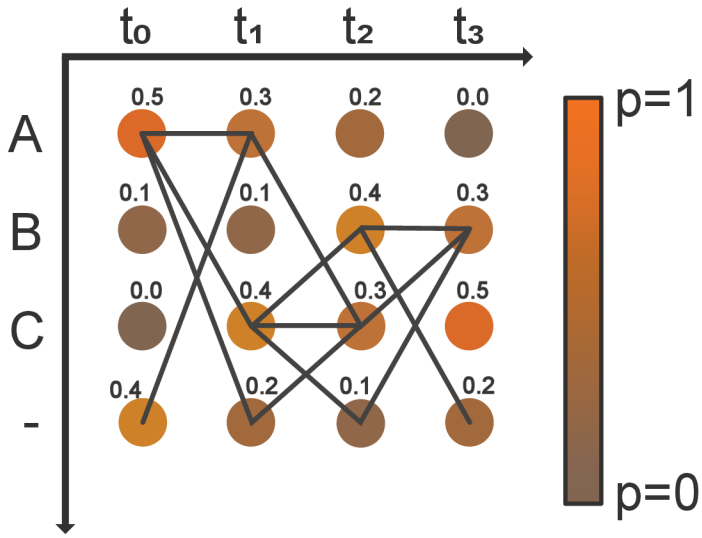
L'. Dále označme L'^T jako soubor sekvencí abecedy L', kde počet sekvencí se rovná délce T. Pokud uvážíme, že pravděpodobnost výstupu není závislá na výstupu v jiných časech, dostaneme podmíněnou pravděpodobnost $\pi \in L'^T$ na vstupní sekvenci $\mathbf{x}$.

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_k^t \tag{4.1}$$

V literatuře bývají sekvence π označovány jako tzv. *paths*. Dále si definujeme mapovací funkci $\mathcal{F} : L'^T \mapsto L^{\leq T}$, vybráním tzv. možných zarovnání (*possible labelling*), soubor sekvencí s menší nebo stejnou délkou jako T, tedy jen takové *paths*, které odpovídají vstupu. Takovéto sekvence jsou tvořeny odstraněním opakujících se znaků a následným odstraněním *blank* znaků z *paths*. Například $\mathcal{F}(a - ab-) = \mathcal{F}(-aa - -abb) = aab$, kde „−“ značí znak *blank*. Celkovou pravděpodobnost nějakého zarovnání (*labelling*) $l \in L^{\leq T}$ lze vypočítat jako sumu pravděpodobností všech *paths*, které lze zobrazit do tohoto zarovnání pomocí $\mathcal{F}$:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{F}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) \tag{4.2}$$

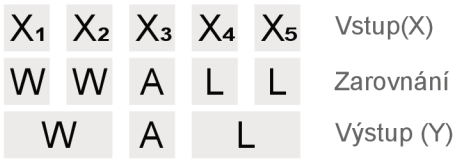
Díky tomuto shlukování rozdílných *paths* do jednoho zarovnání, umožňuje CTC být použito na nesegmentovaných datech. Umožňuje totiž síti předpovídat výstupní sekvenci bez toho, aniž by věděla, kde se jaký znak objeví. Na obrázku 4.1 je znázorněn výpočet $p(\mathbf{l}|\mathbf{x})$.



Obrázek 4.1: Ukázka výpočtu $p(\mathbf{l}|\mathbf{x})$. Na obrázku lze vidět výstup z RNN. Výstup se skládá ze 4 časových kroků a abecedy L' obsahující znaky [a, b, c, –]. Součet pravděpodobností jednotlivých časových kroků je roven jedné. Propojené kruhy znamenají možné cesty *paths*, kde *paths* odpovídající vstupu „acb“ jsou [–acb, a–cb, ac–b, acb–, aacb, accb, acbb]. Pravděpodobnosti těchto *paths* se rovnají [0,0108, 0,009, 0,006, 0,016, 0,0135, 0,018, 0,024], kde hodnoty jsou ve stejném pořadí jako jsou uvedeny *paths*. Celková pravděpodobnost $p(\mathbf{l}|\mathbf{x})$ je sumou pravděpodobností jednotlivých *paths*. Inspirováno z [34]

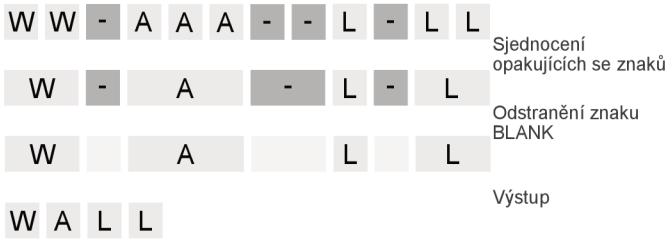
Znak blank. Znak *blank* hraje u CTC velkou roli. Jde o způsob jak se vypořádat s rozdílnou sekvencí znaků na vstupu a výstupu. Nejprve si ukažme jak by vypadalo zarovnání

bez znaku *blank*. Ukázku tohoto zarovnání lze vidět na obrázku 4.2, kde vstupem je sekvence o pěti časových krocích a výsledkem by mělo být slovo „wall“. Zarovnání je tvořeno odstraněním všech opakujících se znaků. Takovéto zarovnání má dva problémy. Jeden lze vidět na obrázku, kde cílem je získat slovo „wall“. Tímto způsobem je získáno pouze slovo „wal“. Další problém nastane pokud je potřeba reprezentovat tiché místo neboli část bez výstupu. V tomto případě není možnost jak tento stav reprezentovat, protože za znakem musí následovat další znak. Znak *blank* vyřeší tyto nevýhody. *Blank* totiž neznamená nic a



Obrázek 4.2: Zarovnání bez znaku blank. Inspirováno z [15].

je z výstupu odstraněn. Pokud se ve slově nachází dva stejné znaky za sebou, musí mít mezi sebou znak *blank*. Při zarovnání na odpovídající řetězec jsou nejprve odstraněny opakující se znaky a poté všechny *blank* znaky, tak jak je znázorněno na obrázku 4.3.



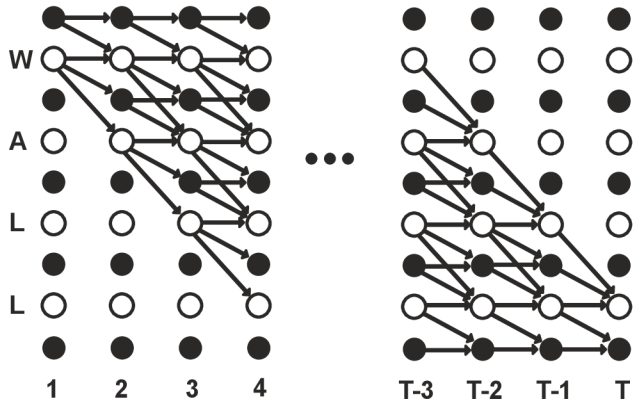
Obrázek 4.3: Zarovnání pomocí znaku blank. Inspirováno z [15].

Forward-Backward algoritmus. K výpočtu *loss* funkce lze použít rovnici 4.2. Tento přístup je ale poněkud naivní a problematický. Suma přes všechny *paths* roste exponenciálně se zvětšující se délkou vstupní sekvence. Tento problém však může být vyřešen dynamicky programovatelným algoritmem, podobný algoritmu *forward-bacward* pro HMM. Myšlenka spočívá s tom, že suma přes *paths* může být rozdělena na postupný součet prefixů neboli pokud dvě zarovnání dosáhla stejného výstupu ve stejném kroku, mohou být sloučena.

Vytvoříme-li modifikovanou sekvenci znaků l', které přidáme *blank* znaky na začátek, konec a také mezi každý pár znaků, poté lze vypočítat *forward* a *backward* proměnnou. Výpočet *forward* proměnné lze vidět na obrázku 4.4. *Backward* proměnnou je možné vypočítat v opačném směru [11, 12].

Nalezení nejlepší cesty. Nyní počítejme s tím, že máme natrénovanou síť. To co nyní požadujeme, je nalezení nejpravděpodobnější sekvence znaků $\mathbf{l}^*$ odpovídající vstupní sekvenci $\mathbf{x}$:

$$\mathbf{l}^* = \underset{\mathbf{l}}{\operatorname{argmax}}\, p(\mathbf{l}|\mathbf{x}) \tag{4.3}$$



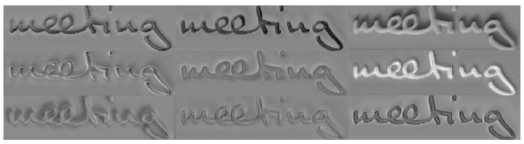
Obrázek 4.4: Výpočet *forward* proměnné pro výpočet $p(\mathbf{l}|\mathbf{x})$. Černé uzly reprezentují ne-*blank* znak a bílé znak z $\mathbf{l}$. Výpočet začíná u prvního *blank* znaku a prvního znaku který není *blank*. Přičemž výsledek je suma posledních dvou uzlů v posledním časovém kroku. Pro výpočet hodnoty uzlu jsou dvě pravidla. Pokud je uzel *blank* znak nebo předchozí ne-*blank* znak je stejný jako znak v současném uzlu, potom pro výpočet daného uzlu můžeme použít součet dvou předchozích uzlů z předchozího časového kroku vynásoben pravděpodobností současného uzlu. V druhém případě, kdy současný uzel je ne-*blank* znak a předchozí ne-*blank* znak není stejný, použijeme pro výpočet součet předchozích tří uzlů z předešlého časového kroku vynásoben pravděpodobností současného uzlu. Inspirováno [12].

Jednou z metod je tzv. *best path decoding*. Jedná se o jednoduchou metodu, která počítá s tím, že nejpravděpodobnější *paths* odpovídá nejvíce pravděpodobné sekvenci znaků. Metoda je lehce vypočitatelná. Nelze ale zajistit, že touto metodou bude nalezena nejvíce pravděpodobná sekvence znaků. Příklad lze vidět na obrázku 4.1, kdy pokud v každém časovém kroku vezmeme ten nejpravděpodobnější znak, získáme výstup „acbc“. Tento výstup ale není nejlepším možným. Nelepší výstup odpovídá „acb“, kde pravděpodobnost tohoto výstupu je rovna součtu všech *paths* které reprezentují tento výstup.

Druhá z hojně využívaných metod, je metoda *prefix search decoding* 4.1, která dosahuje lepších výsledků než *best path decoding*.

4.2 Gated konvoluční sítě

Gated konvoluční sítě [3], [7], jak už podle názvu vyplývá, rozšiřují obecné konvoluční sítě. Dříve byly považovány v oblasti jazykového modelování za *state-of-the-art multi-dimensional long short-term memory* (MD-LSTM) RNN. Tyto sítě predikují výstup na základě předchozího výstupu *(hidden state)*, což má za následek nemožnost paralelních výpočtů a tím pádem velký vliv na rychlost výpočtu. V opačném případě, konvoluční sítě jsou snáze paralelizovatelné a výpočty mohou probíhat současně. *Puigcerver a spol.* v článku [31] dokázali, že konvoluční sítě mohou v nižších vrstvách nahradit 2D-LSTM, což vede k rychlejším výpočtům. Porovnání 2D-LSTM s konvoluční sítí je znázorněno na obrázku 4.5. Ukázalo se, že tzv. *gating mechanism* je pro RNN v takovýchto modelech zásadní a umožňuje extrahovat obecné informace z obrazu. Proto byla snaha takovýmto mechanismem rozšířit i výpočetně rychlejší konvoluční sítě.



(a) 2D-LSTM

(b) Konvoluce

Obrázek 4.5: Porovnání extrahovaných příznaků z 2D-LSTM a konvoluční sítě. Příznaky byly porovnány na podobných pozicích modelů. Z výstupu je patrné, že výstup z obou sítí je podobný a není potřeba v tomto případě použít 2D-LSTM. Převzato z [31]

Gated mechanismus. *Gated* konvoluce určují, které informace (*features*) budou propagovány do dalších vrstev. Myšlenkou *gated* konvolucí je propagovat do dalších vrstev pouze takové informace, které odpovídají kontextu. *Gate*, se dívá na informace v dané pozici a také na ty v blízkém okolí a rozhoduje se jestli daná informace má být zachována nebo zahozena. Takto zpracovává a filtruje informace z celého obrazu, které odpovídají kontextu.

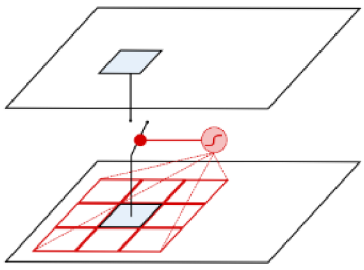
Gate g je tedy implementována jako konvoluce na kterou je aplikována aktivační funkce sigmoid. Tato *gate* je poté aplikována na vstupní mapy příznaků (*feature maps*) $\mathbf{x}$. Výstupem je tedy Hadamardův součin (*element-wise multiplication*) vstupní mapy s výstupem z *gate*:

$$\mathbf{y} = g(\mathbf{x}) \circ \mathbf{x} \tag{4.4}$$

kde

$$\begin{aligned} g(x_i j) = \sigma(&w_{00}x_{i-1,j-1} + w_{01}x_{i-1,j} + w_{02}x_{i-1,j+1} + \\ &w_{10}x_{i,j-1} + w_{11}x_{i,j} + w_{12}x_{i,j+1} + \\ &w_{20}x_{i+1,j-1} + w_{21}x_{i+1,j} + w_{22}x_{i+1,j+1}) \end{aligned}$$

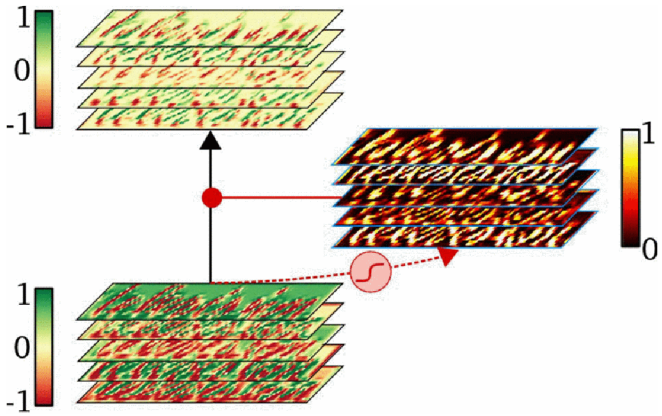
Gate vypočítává hodnotu každé informace, která je v rozmezí $[0, 1]$, který vypočítá na základě sousedního okolí dané informace. Ukázku výpočtu lze vidět na obrázku 4.6. Na dalším obrázku 4.7 lze vidět reálný příklad před a po *gated* konvoluci.



Obrázek 4.6: Znázornění Gated konvoluce. Konvoluční filtr s aktivační funkcí sigmoid je aplikován na a kolem dané pozice. Převzato z [3]

4.3 Normalizace vnitřních vrstev

Trénování neuronových sítí je komplikovaný proces, kdy výstup jedné vrstvy je vstupem té další a takovýchto vrstev je v sítí mnoho. To znamená, že i malá změna parametru v nějaké



Obrázek 4.7: Příklad Gated konvoluce na reálném obrázku. Převzato z [3]

vrstvě se projeví na distribuci vstupů následujících vrstev. Tento problém je označován jako tzv. *internal covariate shift*. Lze ho řešit normalizací vstupů jednotlivých vrstev sítě. K tomuto účelu existují dva přístupy *Batch Normalization* [21] a *Batch Renormalization* [20], které jsou dále popsány.

Batch Normalization. Jde o vrstvu která pro každou trénovací dávku (tzv. *batch*) provádí normalizaci vstupních hodnot využíváním průměrné hodnoty a rozptylu z této trénovací dávky. *Batch* normalizace umožňuje samostatné učení jednotlivých vrstev nezávisle na ostatních vrstvách. Díky tomu je možné použít větší velikost kroku při učení (tzv. *learning rate*) a lze zaznamenat rychlejší učení sítě. *Batch* normalizaci výstupu z vrstvy x v aktuální trénovací dávce $\mathcal{B} = \{x_{1\cdots m}\}$ vypočítáme jako:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4.5}$$

$$\sigma_{\mathcal{B}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 + \epsilon} \tag{4.6}$$

$$\widehat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \tag{4.7}$$

$$y_i = \gamma\widehat{x}_i + \beta \tag{4.8}$$

Kde γ a β jsou trénovatelné parametry, $\sigma_{\mathcal{B}}$ je směrodatná odchylka, $\mu_{\mathcal{B}}$ je průměr hodnot trénovací dávky a ϵ je malá konstanta. Takovýto výpočet se provádí během trénování sítě. Během inference (testování) je normalizace vypočtena pomocí klouzavých průměrů σ^2 a μ které si síť uchovává.

Batch Renormalization. *Batch* renormalizace je vylepšením *batch* normalizace, která se snaží neoddělovat trénovací a inferenční model sítě tak, jak tomu bylo u *batch* normalizace. *Ioffe a spol.* [20] zjistili, že normalizace v trénovacím a inferenčním modelu, lze transformovat pomocí afinní transformace. Konkrétně, pokud μ představuje průměrnou hodnotu

x a σ představuje standardní směrodatnou odchylku, které jsou vypočítány jako klouzavý průměr přes několik trénovacích dávek, poté dostaneme:

$$\frac{x_i - \mu}{\sigma} = \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \cdot r + d, \tag{4.9}$$

kde,

$$r = \frac{\sigma_{\mathcal{B}}}{\sigma}, d = \frac{\mu_{\mathcal{B}} - \mu}{\sigma}$$

Tyto nové proměnné r a d jsou při *back*-propagaci ponechány jako konstanty. Podle článku [20] *batch* renormalizace dosahuje stejných nebo lepších výsledků než *batch* normalizace.

Kapitola 5

Vertical Attention Module

Tato kapitola popisuje prvek zvaný *Vertical Attention Module*, který byl do vybrané architektury implementován a následně v kapitole 7 jsou s tímto modulem provedeny experimenty.

Vertical Attention Module (VAM) byl představen v článku [19]. Zde bylo uvedeno zároveň s *Horizontal Attention Module* (HAM) a společně řešili nevýhody CTC a *sequence-to-sequence* modelů, které převádějí mapy příznaků v dvoudimenzionálním (2D) prostoru na sekvenci příznaků v 1D prostoru. Na obrázku 5.1 je zobrazena architektura této sítě. Převod 2D mapy příznaků přímo na 1D může přinášet do následujících sekvencí informace navíc, převážně pokud pracujeme s nezarovnaným, nepravidelným textem v obrazu. Jejich tzv. *Cross Attention Network* se snaží „zvýrazňovat“ jednotlivé znaky na celé 2D mapě příznaků. VAM „zvýrazňuje“ rysy, které odpovídají celému textu na 2D prostoru ve vertikálním směru a HAM je použit na předchozích zvýrazněných rysech při dekódování jednotlivých znaků. Dále si popíšeme pouze VAM se kterým jsou dále prováděny experimenty. Vstupem VAM je 2D mapa příznaků, která je vytvářena pomocí CNN. Označme si extrakci příznaků jako:

$$X = \mathcal{F}(I), \tag{5.1}$$

kde I reprezentuje obrázek na vstupu do CNN a $X \in \mathcal{R}^{C \times W \times H}$ značí 2D mapu příznaků. C, W, H jsou počty kanálů, šířka a výška výstupní mapy příznaků. VAM vybírá příznaky, které náleží textu v obraze každého sloupce z 2D mapy příznaků. Pro každý sloupec $X_i \in \mathcal{R}^{C \times H}$ v X, VAM generuje vybrané příznaky do sub-sekvence příznaků $f_{v,i}$ podle:

$$f_{v,i} = \sum_{k=1}^{H} \beta_{i,k} X_{i,k}, \tag{5.2}$$

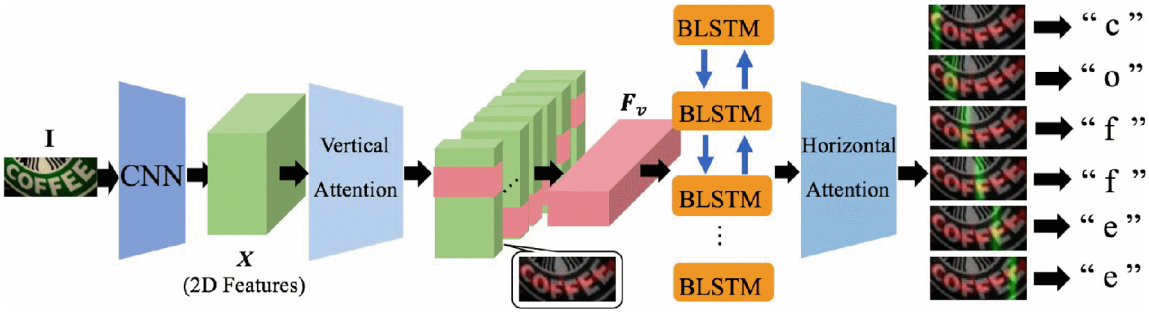
kde $X_{i,k} \in \mathcal{R}^{C}$ reprezentuje k-tý vektor X_i a $\beta_i = \{\beta_{i,1}, \beta_{i,2}, \cdots, \beta_{i,H}\}$ je rozdělení pravděpodobnosti tzv. *attention distribution* přes všechny elementy v jednotlivých sloupcích. Každý element $\beta_{i,j}$ z β_i lze vypočítat:

$$\beta_{i,j} = \frac{exp(g_{i,j})}{\sum_{k=1}^{H} exp(g_{i,k})}, \tag{5.3}$$

kde $g_{i,j}$ je vyhodnocení j-tého vektoru v sub-sekvenci příznaků X_i. Tento vektor $g_i = \{g_{i,1}, g_{i,2}, \cdots, g_{i,H}\}$ je vypočítán podle výrazu:

$$g_i = \mathcal{F}_{1\times1}(\mathcal{F}_{1\times3}(X_i) + X_i). \tag{5.4}$$

Funkce $\mathcal{F}_{n\times m}$ představuje konvoluci o velikosti $n \times m$, kde n značí šířku a m výšku jádra konvoluce.



Obrázek 5.1: Architektura sítě *Cross Attention Network* s VAM a HAM moduly, které se snaží pracovat jen s relevantními příznaky z 2D map příznaků. Převzaté z [19].

Kapitola 6

Navržené architektury

Ze dvou současně nejpoužívanějších (*state-of-the-art*) modelů, které jsem popsal v kapitole 2 mezi které spadá model CNN-RNN-CTC (viz 2.2) a model *encoder-decoder* (viz 2.3) byl vybrán model CNN-RNN-CTC. Tento model představili Shi a spol. v článku [35]. V této práci byla použita volně dostupná implementace tohoto modelu[1].

Dále jsou popsány architektury sítí se kterými byly prováděny experimenty na tomto modelu. Kapitola je rozdělena na tři části. První část se věnuje implementovaným konvolučním sítím v druhé části jsou představeny možnosti vertikálních agregací, které zmenšují rozměr ve vertikálním směru při přechodu z konvolučních sítě do rekurentní sítě a v třetí části je popsána architektura rekurentní sítě. Nakonec jsou uvedeny využívané prostředky při implementaci těchto architektur.

6.1 Konvoluční sítě

Pro experimenty byly použity dva způsoby konvolučních sítí. V dostupné implementaci byla zpracována konvoluční část pomocí *gated* konvolucí a jako další byla použita konvoluční síť VGG16 [36].

Gated konvoluční síť. Tato konvoluční síť modelu obsahuje *gated* konvoluce popsané v sekci 4.2. V této implementaci se ale *gate* mechanismus neaplikuje pouze na vstupní mapu, ale je zde na tuto mapu použita konvoluce s lineární funkcí a poté je proveden Hadamardův součin s výstupem z *gate*.

Gated konvoluční vrstvy jsou zde kombinovány s normální konvolucí. Celá tato konvoluční část modelu se tedy skládá z 6-ti sekcí, kde prvních 5 obsahuje 2D konvoluční vrstvu s aktivační funkcí PReLU [16]. Některé konvoluční vrstvy snižují velikost šířky a výšky obrazu díky posouvání filtru (konvolučního jádra) s krokem větší než jedna. Za těmito vrstvami následuje *batch* renormalizace (*Batch Renormalization*) [21] a nakonec *gated* konvoluce. V poslední sekci je vynechána *gated* konvoluce a je za ní připojena *max pooling* vrstva. Podrobnější popis této konvoluční části modelu lze vidět na obrázku 6.1.

VGG16. Jedná se o populární a velmi hlubokou konvoluční síť (*Very Deep Convolutional Networks*), která se nachází již implementovaná v různých knihovnách. Tuto síť jsem ve své práci použil více způsoby. Nikdy jsem ale nepracoval s celou sítí, protože se jedná o velmi

[1] https://github.com/arthurflor23/handwritten-text-recognition, autor tento repozitář během dodělávání mé práce odstranil. Výchází z článku [8]

hlubokou síť s velkým množstvím parametrů u kterého by trénování trvalo zbytečně dlouho a pro porovnání s *gated* konvoluční sítí je zmenšená verze postačující. První použitý způsob byl s vlastní implementací této sítě, kdy byly použity pouze první tři bloky VGG16 s ponechaným počtem filtrů jako v původní VGG16 a dále síť se sníženým počtem filtrů. Na obrázku 6.2 lze vidět tuto architekturu. Pro druhou variantu sítě byla použita implementace s již předtrénovanými hodnotami vah sítě na datasetu *ImageNet*[2], která je dostupná v knihovně *Keras*[3]. Zde byly použity také první tři bloky této sítě a dále byla vytvořena síť pouze se dvěma bloky. Tyto části VGG byly dále rozděleny na sítě, které se už více neučí (jejich váhy zůstávají po celou dobu učení tak, jak byly předtrénovány) a síť, která se ještě učí.

6.2 Vertikální agregace

Konvoluční část sítě získává vizuální informace ze vstupního obrázku a jejím výstupem jsou mapy příznaků, které jsou reprezentovány jako tenzor se čtyřmi rozměry (4D) které představují – (*velikost dávky*, *počet řádků*, *počet sloupců*, *počet kanálů*). Vstupem do rekurentních sítí je ale tenzor se třemi rozměry (3D), které reprezentují – (*velikost dávky*, *počet časových kroků*, *počet kanálů*). Dále jsou předvedeny tři možnosti převodu 4D na 3D, které jsem v této práci využíval a porovnával.

Konkatenace. Konkatenací (zřetězení) posledních dvou os 4D tenzoru dostaneme 3D tenzor. Například pokud máme tenzor $(1, 3, 2, 3)$, který představuje jeden obrázek se třemi řádky, dvěma sloupci a třemi kanály po zřetězení posledních dvou os dostaneme tenzor $(1, 3, 6)$, který reprezentuje jeden obrázek se třemi časovými kroky a 6 kanály.

Average pooling. Jako další možností při zmenšování rozměru jsem se rozhodl pro použití *average pooling* vrstvy přes výšku vstupního obrázku reprezentující řádek textu. Tato vrstva vypočítá průměrnou hodnotu v každém sloupci a tento sloupec je poté nahrazen touto hodnotou. Obrázek 6.3 zobrazuje funkci *average poolingu*.

Vertical attention module. Jako poslední alternativou při zmenšení rozměru sítě byl použit *vertical attention module*, který jsem popsal v kapitole 5. VAM představili *Huang a spol.* v článku [19] spolu s HAM. V této práci jsem se rozhodl experimentovat pouze v VAM a zjistit přínos tohoto modulu v síti. Obrázek 6.4 představuje implementaci VAM modulu, tak jak byla implementována za pomoci knihovny *Keras* díky dostupných vrstev (*layers*) použitelných jako funkce programu.

6.3 Rekurentní vrstva

Rekurentní vrstva zpracovává jednotlivé příznakové vektory a v každém časovém okamžiku vyhodnocuje nejpravděpodobnější znak. Implementovaná rekurentní vrstva se skládá ze dvou částí, kde každá obsahuje obousměrnou GRU na kterou je připojena plně propojená neuronová síť (*fully connected layer*).

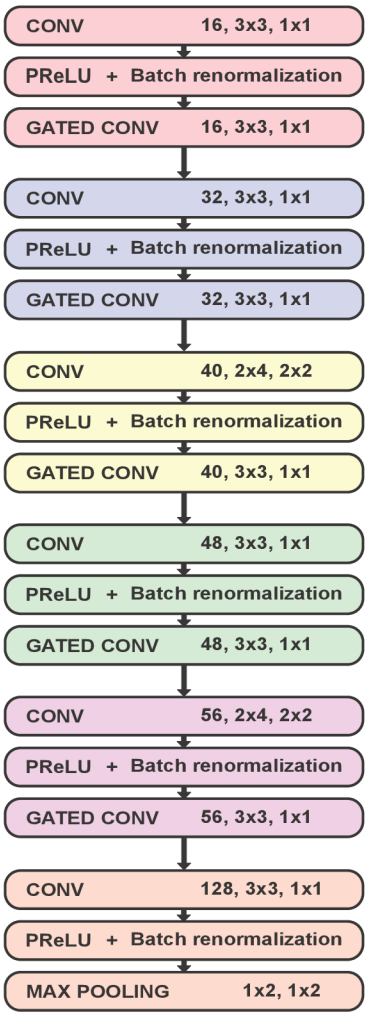
[2] http://www.image-net.org/
[3] https://keras.io/

Na rekurentní vrstvu navazuje CTC (viz 4.1), který převádí výstup z rekurentní vrstvy na rozdělení pravděpodobnosti pro všechny znaky dané abecedy. Celá architektura sítě zobrazena na obrázeku 6.5

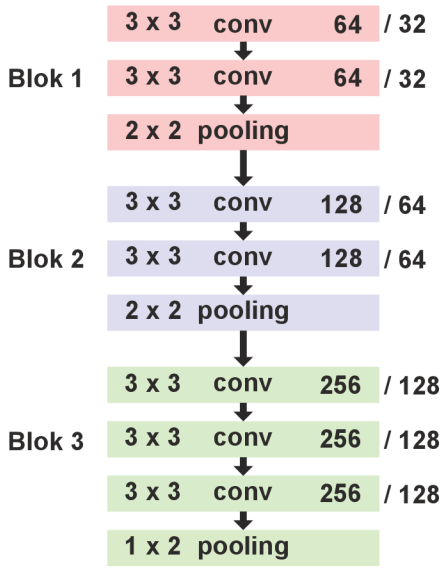
6.4 Implementační detaily

Implementace tohoto modelu proběhla v programovacím jazyce *Python*[4], který byl vybrán pro jeho rozsáhlou sadu knihoven podporujících a usnadňujících implementaci neuronových sítí. Z těchto knihoven byla vybrána knihovna *Keras*[5]. *Keras* je vysokoúrovňové API využívající knihovnu *TensorFlow*[6]. Byl vyvinut za účelem rychlého vývoje a experimentování v oblasti strojového učení. Výhodou je možnost kombinování vysokoúrovňového vývoje pomocí *Keras* a nízkoúrovňového díky *TensorFlow*. Výpočty probíhaly ve výpočetním centru *Metacentrum*[7], které zprostředkovává výpočetní clustery, které jsou zpřístupněny všem akademickým pracovníkům, studentům, členům ve sdružení *CESNET*[8].

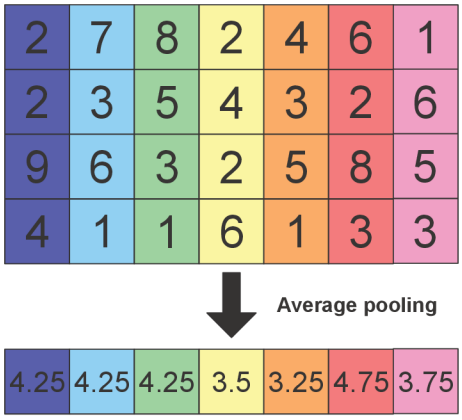
[4] https://www.python.org/
[5] https://keras.io/
[6] https://www.tensorflow.org/
[7] https://metavo.metacentrum.cz/
[8] https://www.cesnet.cz/



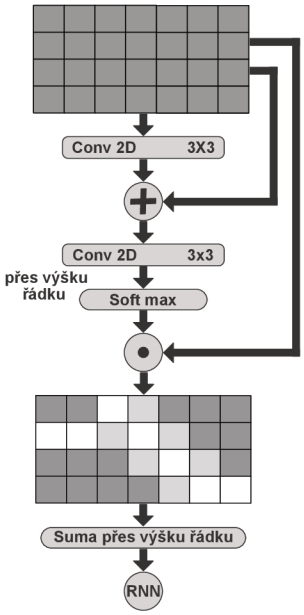
Obrázek 6.1: Architektura konvoluční části s gated konvolucemi. Každá část obsahuje konvoluční vrstvy, PReLU, batch renormaizaci a *gated* konvoluci. Hodnoty u konvolučních a gated vrstev představují: počet konvolučních filtrů, rozměr konvolučního jádra a krok konvolučního jádra. Hodnoty u max pooling vrsty jsou: rozměr pooling filtru a krok tohoto filtru.



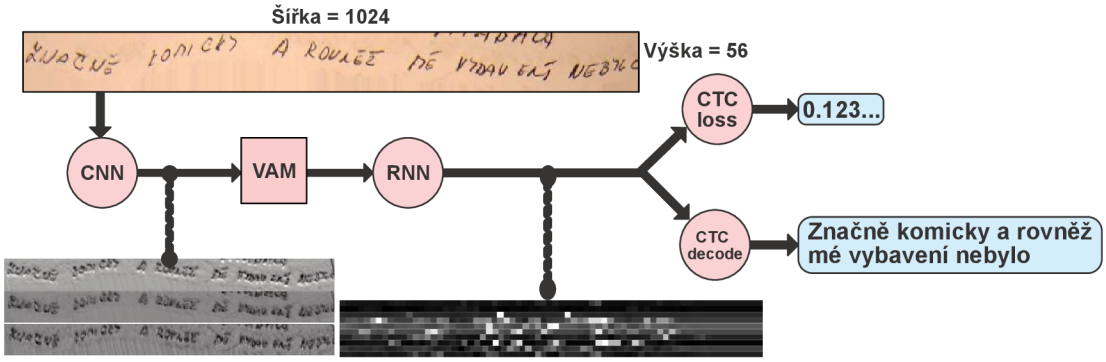
Obrázek 6.2: Architektura prvních tří bloků VGG16, kde hodnoty nalevo představují rozměry konvolučních a pooling jader a hodnoty napravo jsou počty konvolučních filtrů, kde hodnoty před lomítkem představují vetší (originální) architekturu a hodnoty za lomítkem jsou pro síť s menším počtem filtrů.



Obrázek 6.3: Average pooling pro každý sloupec z map příznaků.



Obrázek 6.4: Architektura *vertical attention* modulu, který vyznačuje relevantní příznaky odpovídající textu na vstupním obrázku.



Obrázek 6.5: Architektura celé sítě s VAM. Inspirováno z [34]

Kapitola 7

Experimenty

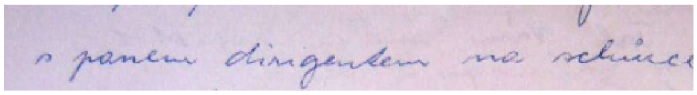
Tato kapitola shrnuje všechny prováděné experimenty. Ty probíhaly na architekturách popsaných v kapitole 6. Rekurentní vrstva byla pro všechny architektury sítě stejná. Lišily se v použití konvolučních vrstev a vertikální agregace. Protože se v průběhu provádění experimentů síť s vlastní implementací VGG16 a použití VAM netrénovala (od první epochy se její *validation loss* skoro nezměnila a tato hodnota byla příliš veliká oproti ostatním experimentům), byla mezi VGG a VAM vložena *batch* renormalizace, což vedlo k trénování. Díky tomu byla poté tato a nebo *batch* normalizační vrstva zařazena mezi každou implementaci VGG16 (vlastní implementaci, předtrénovanou implementaci pomocí knihovny *Keras* s fixními i nefixními váhami atd.) a každou vertikální agregaci. Následně byly pozorovány výsledky všech sítí s těmito a bez těchto vrstev. V této kapitole jsou nejdříve uvedeny parametry trénování s jakými byly experimenty prováděny. Dále je představena datová sada a následně jsou popsány výsledky, které byly díky experimentům dosaženy.

Parametry trénování. Všechny experimenty proběhaly se stejnými parametry sítě. Rychlost učení neboli tzv. *learning rate* bylo nastaveno na hodnotu 5×10^{-4}, pokud se po deseti provedených epochách nezmenšila validační chybová funkce (*validation loss*), tak se hodnota této *learning rate* snížila vynásobením hodnotou 0.2. Trénovací dávka (*batch*) obsahovala 32 vzorků neboli obrázků. Jako optimalizační algoritmus byl vybrán algoritmus Adam [25]. Konec trénování nastalo po 24 hodinách a nebo pokud validační chybová funkce neprokázala zlepšení (menší chybu) v 17 epochách za sebou. Rozměr všech vstupních obrázků je zarovnán na velikost 1024×56 pixelů, kde první hodnota reprezentuje šířku a druhá výšku, přičemž sítě které obsahují předtrénované konvoluční sítě z knihovny *Keras* pracují s barevným obrázkem obsahující 3 kanály, zatímco všechny ostatní pracují s černobílými obrázky obsahující 1 kanál. Na obrázky jsou při trénování použity operace eroze a dilatace, pro docílení lepší přesnosti sítě.

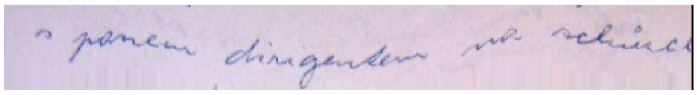
Použitá datová sada. Na trénování, validaci a testování byla použita datová sada České dopisy (viz 3.1). Protože implementovaný prvek *vertical attention* modul by měl přinášet vylepšení sítě převážně na nezarovnaném a nepravidelném textu, byly tyto řádky textu upraveny[1] pro tento účel. Na obrázku 7.1 lze vidět řádek textu z datové sady před a po úpravě. Tato datová sada byla rozdělena na tři části, kdy při trénování se používaly dvě

[1] Skript na úpravu („rozvlnění“) obrázků z datové sady byl poskytnut od vedoucího práce Ing. Michala Hradiše, Ph.D.

z těchto částí a to trénovací a validační, kde trénovací sada obsahuje 65112 řádků a validační 8000 řádků. Při testování byla použita testovací sada o 8074 řádcích.



(a) Řádek textu před úpravou



(b) Řádek textu po úpravě

Obrázek 7.1: Řádek textu z datové sady České dopisy před a po úpravě, díky které jsou řádky nezarovnané a nepravidelné.

Vyhodnocení. Vyhodnocení výstupu sítě je provedeno pomocí metrik CER (*Character error rate*) a WER (*Word error rate*), které dokáží pracovat s rozdílnou délkou mezi rozpoznávaným a výchozím textem. Obě tyto metriky se počítají na základě Levenštejnovy vzdálenosti (*Levenshtein distance*) [33].

7.1 Porovnání vertikálních agregací na odlišných konvolučních sítích

Tyto experimenty se zaměřují na porovnání tří vertikálních agregací na konvolučních sítích. Implementované konvoluční sítě lze rozdělit na ne-předtrénované, předtrénované s fixními váhami a předtrénované s trénovatelnými váhami. Nejprve jsou představeny jednotlivé výsledky vertikálních agregací na těchto sítích a poté celkový souhrn dosažených výsledků.

Ne-předtrénované konvolučních sítě. Tento experiment je zaměřen na porovnání vertikálních agregací na mnou implementovaných konvolučních sítích s ne-předtrénovanými váhami. Mezi tyto sítě patří *gated* konvoluční síť, VGG16 obsahující pouze první tři bloky s původními počty filtrů a dále se zmenšených počtem těchto filtrů. Normalizace v tomto experimentu nejsou důležité a proto jsou u obou modelů VGG vybrány nejlepší výsledky z *batch* renormalizace, *batch* normalizace a bez použití normalizace. Výsledky tohoto experimentu jsou v tabulce 7.1. Z výsledků je vidět, že VAM přineslo malé zlepšení oproti konkatenaci. Pokud ale porovnáme VAM s *average poolingem*, jsou na tom velmi podobně.

Vertikální agregace	**Gated**		**VGG**		**VGG - méně filtrů**	
	CER	**WER**	**CER**	**WER**	**CER**	**WER**
Average pooling	11,3 %	30,7 %	**10,3 %**	**28,5 %**	11,3 %	30,4 %
Konkatenace	12,7 %	33,7 %	11,1 %	30,3 %	12,0 %	32,3 %
VAM	**10,7 %**	**29,6 %**	10,5 %	28,8 %	**10,9 %**	**29,8 %**

Tabulka 7.1: Výsledky porovnání vertikálních agregací na ne-předtrénovaných konvolučních sítích

Předtrénované konvoluční sítě s fixními váhami. V tomto experimentu jsou porovnávány vertikální agregace na předtrénovaných konvolučních sítí VGG16 z knihovny *Keras* u kterých bylo pozastaveno trénování jejich vah. Podobně jako u předešlého experimentu jsou zde vybrány pouze nejlepší výsledky z normalizací. V tabulce 7.2 můžeme vidět zhoršení přesnosti rozpoznání. Toto zhoršení se dalo očekávat, protože jsou tyto konvoluční sítě sice natrénované, ale byly trénované na jiném datasetu a dále se již netrénovaly. S touto změnou si nejlépe poradila konkatenace spolu s VAM. Trochu až nečekaně velký rozdíl v porovnání je vidět u *average poolingu*.

Vertikální agregace	**VGG3 - fix**		**VGG2 - fix**	
	CER	**WER**	**CER**	**WER**
Average pooling	23,9 %	51,1 %	29,3 %	59,0 %
Konkatenace	17,6 %	42,6 %	**15,2 %**	**38,9 %**
VAM	**16,9 %**	**41,2 %**	17,7 %	43,3 %

Tabulka 7.2: Výsledky porovnání vertikálních agregací na předtrénovaných CNN s fixními váhami. Zkratka VGG3 představuje konvoluční síť sládající se z prvních tří bloků modelu VGG16 a označení VGG2 z prvních dvou.

Předtrénované konvoluční sítě s trénovatelnými váhami. Posledním experimentem porovnávající vertikální agregace na různých konvolučních sítích, je experiment, který vychází ze stejných sítí jako předešlý experiment, ale váhy těchto konvolučních sítí se dále trénují. Výsledky tohoto experimentu jsou v tabulce 7.3. U větší sítě můžeme pozorovat nejlepší výsledky rozpoznávání, kde se *average pooling* ukazuje jako nejlepší možnost. U menší sítě můžeme pozorovat opačný jev, kde se *average pooling* jeví jako nejhorší a VAM nejlepší možnost.

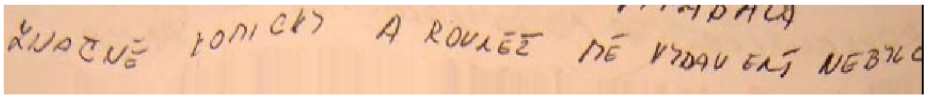
Vertikální agregace	**VGG3**		**VGG2**	
	CER	**WER**	**CER**	**WER**
Average pooling	**8,9 %**	**25,4 %**	13,2 %	34,8 %
Konkatenace	9,9 %	27,9 %	11,7 %	32,4 %
VAM	9,1 %	25,9 %	**10,6 %**	**29,8 %**

Tabulka 7.3: Výsledky porovnání vertikálních agregací na předtrénovaných CNN s trénovatelnými parametry. Zkratka VGG3 představuje konvoluční síť skládající se z prvních tří bloků modelu VGG16 a označení VGG2 z prvních dvou.

Souhrn výsledků vertikálních agregací Na dosažených výsledcích lze vidět, že VAM dosahuje téměř pokaždé nejlepších výsledků. Zatímco výsledky konkatenace a *average poolingu* se mění s velikostí sítě neboli jak dobře poslední vrstva konvoluční sítě interpretuje text na obrázku, tento jev je zobrazen na obrázku 7.4. Příkladem může být VGG2 s fixními váhami z druhého experimentu, kdy CER konkatenace je 15,2 %, zatímco u *average poolingu* 29,3 %. Pokud se ale podíváme na VGG3 s trénovatelnými váhami z posledního experimentu, zjistíme, že zde je na tom lépe *average pooling* s 8,9 % oproti konkatenaci s 9,9 %.

Výsledky VAM jsou nejspíše dobré díky schopnosti naučit se, které příznaky korelují s textem na vstupním obrázku. Na obrázku 7.3 lze vidět výstup aktivací ze softmax funkce

uvnitř VAM na základě vstupního obrázku 7.2 . Softmax funkce zde kopíruje tvar textu ze vstupního obrázku a ukazuje, které příznaky jsou propagovány dále do RNN.



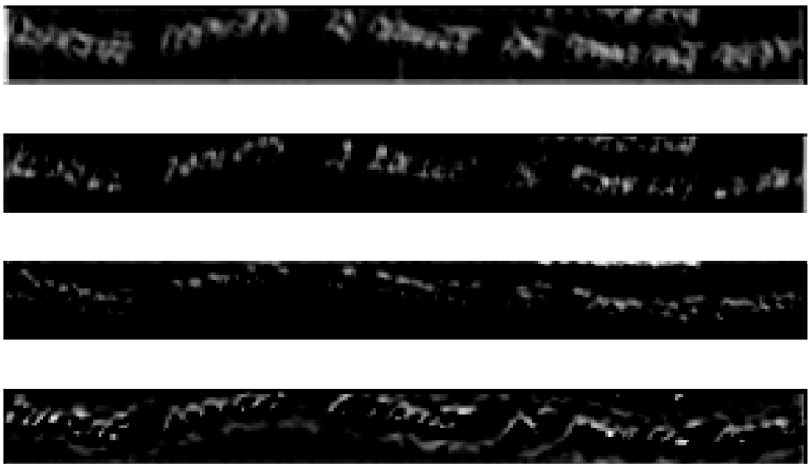
Obrázek 7.2: Vstupní obrázek s textem z testovací datové sady.

Obrázek 7.3: Výstup ze softmax funkce uvnitř VAM. Softmax funkce na tomto obrázku kopíruje tvar textu z obrázku 7.2

Porovnání výstupů aktivací posledních vrstev konvolučních sítí VGG2 s fixními a bez fixních vah s následným použitím average poolingu je zobrazeno na obrázku 7.4. Na obrázcích jsou zobrazeny čtyři kanály poslední konvoluční vrstvy z obou sítí. Lze pozorovat, že výstup z VGG2 s fixními váhami obsahuje o mnoho více informací a je možné, že jsou tyto informace při propagaci do RNN zkresleny a síť poté nedosahuje dobré přesnosti rozpoznávání. Zatímco u sítě VGG2 s trénovatelnými váhami lze pozorovat, že se síť naučila sama rozpoznávat relevantní informace odpovídající textu.

7.2 Porovnání normalizací

Ve všech sítích na bázi VGG16 byly mezi konvoluční síť a vertikální agregací implementovány vrstvy *batch* renormalizace a *batch* normalizace. Tyto normalizace byly poté porovnávány mezi sebou a se sítěmi bez této normalizace. Z výsledků nebylo možné určit jestli je lepší normalizace nebo renormalizace, protože dosahovaly velice podobných výsledků. Co ale bylo zjištěno je, že u všech předtrénovaných VGG16 se bez normalizace žádná ze sítí netrénuje. U nepředtrénované VGG16 s více filtry se bez normalizace netrénovala pouze VAM.



(a) Výstup ze čtyř kanálů poslední konvoluční vrstvy VGG2 s fixními váhami.

(b) Výstup ze čtyř kanálů poslední konvoluční vrstvy VGG2 s trénovatelnými váhami.

Obrázek 7.4: Porovnání výstupů aktivací posledních vrstev konvolučních sítí VGG2 s fixními a bez fixních vah s použitím average poolingu. Vstupem těchto sítí byl řádek textu zobrazen na obrázku 7.2

Kapitola 8

Závěr

Cílem této práce bylo navrhnout systém rozpoznávající ručně psaný text za pomoci konvolučních sítí. Práce zahrnovala prostudování současných metod, které se v této oblasti používají. Z těchto metod byl vybrán model neuronové sítě skládající se z konvoluční a rekurentní sítě s *Connectionist Temporal Classification*, který převádí sekvenční výstup z rekurentní sítě na posloupnost znaků.

Do takovéhoto modelu byl následně implementován prvek *Vertical Attention Module*, který vybírá relevantní informace v každém sloupci odpovídající textu na obrázku. Tento model byl následně porovnáván s dalšími vertikálními agregacemi za použití *average poolingu* a konkatenace. Experimenty porovnávaly chování sítě těchto tří vertikálních agregací za použití různých konvolučních sítí a normalizačních vrstev. Experimenty probíhaly na datové sadě českých dopisů z 20. století, u kterých byl text zvlněn tak, aby nebyl zarovnán na řádcích.

Experimenty ukázaly, že *Vertical Attention Module* dosahuje téměř vždy nejlepších výsledků, avšak oproti ostatním vertikálním agregacím není rozdíl nějak zásadní. Také bylo zjištěno, že *Vertical Attention Module* navazující na VGG16 se bez normalizační vrstvy nedokáže učit. V rámci experimentů byly pozorovány výstupy aktivací jednotlivých vrstev. Při pozorování těchto výstupů byla věnována pozornost váhám ve *Vertical Attention Module*, které vždy dokázaly kopírovat tvar textu. Zajímavé bylo pozorovat poslední konvoluční vrstvy při používání *average poolingu*. Tyto vrstvy prokázaly, že i *average pooling* dokáže na trénovatelné sítí dobře kopírovat text. Výsledná síť dosáhla nejlepšího výsledku při chybě 8,9 % na znak.

V budoucím vývoji by stálo za vyzkoušení jiných *attention* modulů nebo zkusit modifikovat stávající na více takovýchto paralelních modulů pracujících současně. Dalším možným rozšířením by mohlo být přidání *Horizontal Attention Module*, se kterým byl současný modul poprvé představen.

Literatura

[1] Welcome Message from the Honorary Chair. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, s. 35–35.

[2] *International Conference on Frontiers in Handwriting Recognition 2020 in Dortmund* [online]. 2020 [cit. 2020-07-09]. Dostupné z: http://icfhr2020.tu-dortmund.de.

[3] Bluche, T. a Messina, R. Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2017, sv. 01, s. 646–651.

[4] Bluche, T., Ney, H. a Kermorvant, C. Tandem HMM with convolutional neural network for handwritten word recognition. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, s. 2390–2394.

[5] Bunke, H., Roth, M. a Schukat Talamazzini, E. *Off-line Cursive Handwriting Recognition using Hidden Markov Models*. 1995.

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

[7] Dauphin, Y. N., Fan, A., Auli, M. a Grangier, D. Language Modeling with Gated Convolutional Networks. *CoRR*. 2016, abs/1612.08083.

[8] Flôr, A. *Handwritten Text Recognition using TensorFlow 2.0* [online]. 2019 [cit. 2020-07-29]. Dostupné z: https://medium.com/@arthurflor23/handwritten-text-recognition-using-tensorflow-2-0-f4352b7afe16.

[9] Gatos, B., Louloudis, G., Causer, T., Grint, K., Romero, V. et al. Ground-Truth Production in the Transcriptorium Project. In: *2014 11th IAPR International Workshop on Document Analysis Systems*. 2014, s. 237–241.

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

[11] Graves, A. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012. Studies in Computational Intelligence. ISBN 978-3-642-24796-5.

[12] Graves, A., Fernández, S., Gomez, F. a Schmidhuber, J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent

Neural Networks. In: *Proceedings of the 23rd International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2006, s. 369–376. ICML '06. DOI: 10.1145/1143844.1143891. ISBN 1595933832.

[13] Grosicki, E., Carré, M., Brodin, J. a Geoffrois, E. Results of the RIMES Evaluation Campaign for Handwritten Mail Processing. In: *2009 10th International Conference on Document Analysis and Recognition*. 2009, s. 941–945.

[14] Grosicki, E. a El-Abed, H. ICDAR 2011 - French Handwriting Recognition Competition. In: *2011 International Conference on Document Analysis and Recognition*. 2011, s. 1459–1463.

[15] Hannun, A. Sequence Modeling with CTC. *Distill*. 2017. DOI: 10.23915/distill.00008. https://distill.pub/2017/ctc.

[16] He, K., Zhang, X., Ren, S. a Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR*. 2015, abs/1502.01852.

[17] Hladká, Z. *111 let českého dopisu v korpusovém zpracování* [paměťový nosič]. Vyd. 1. Brno: Masarykova univerzita, 2013. ISBN 978-80-210-6141-5.

[18] Hochreiter, S. a Schmidhuber, J. Long Short-term Memory. *Neural computation*. Prosinec 1997, sv. 9, s. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[19] Huang, Y., Luo, C., Jin, L., Lin, Q. a Zhou, W. Attention After Attention: Reading Text in the Wild with Cross Attention. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, s. 274–280.

[20] Ioffe, S. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. *CoRR*. 2017, abs/1702.03275.

[21] Ioffe, S. a Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*. 2015, abs/1502.03167.

[22] Jaderberg, M., Simonyan, K., Vedaldi, A. a Zisserman, A. Reading Text in the Wild with Convolutional Neural Networks. *CoRR*. 2014, abs/1412.1842.

[23] Jaderberg, M., Simonyan, K., Zisserman, A. a Kavukcuoglu, K. Spatial Transformer Networks. *CoRR*. 2015, abs/1506.02025.

[24] Kang, L., Toledo, J. I., Riba, P., Villegas, M., Fornés, A. et al. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In: Brox, T., Bruhn, A. a Fritz, M., ed. *Pattern Recognition*. Cham: Springer International Publishing, 2019, s. 459–472.

[25] Kingma, D. P. a Ba, J. Adam: A Method for Stochastic Optimization. In: Bengio, Y. a LeCun, Y., ed. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[26] Krishnan, P., Dutta, K. a Jawahar, C. V. Word Spotting and Recognition Using Deep Embedding. In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. 2018, s. 1–6.

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

[28] Marti, U. a Bunke, H. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*. Nov 2002, sv. 5, č. 1, s. 39–46. DOI: 10.1007/s100320200071. ISSN 1433-2833.

[29] Michael, J., Labahn, R., Grüning, T. a Zöllner, J. Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, s. 1286–1293.

[30] Pletschacher, S. a Antonacopoulos, A. The PAGE (Page Analysis and Ground-Truth Elements) Format Framework. In: *2010 20th International Conference on Pattern Recognition*. 2010, s. 257–260.

[31] Puigcerver, J. Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2017, sv. 01, s. 67–72.

[32] Puigcerver, J., Martin Albo, D. a Villegas, M. *Laia: A deep learning toolkit for HTR* [https://github.com/jpuigcerver/Laia/tree/master/egs/rimes]. GitHub, 2016. GitHub repository.

[33] Putra, M. E. W. a Supriana, I. Structural offline handwriting character recognition using levenshtein distance. In: *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*. 2015, s. 31–36.

[34] Scheidl, H. *An Intuitive Explanation of Connectionist Temporal Classification* [online]. 2018 [cit. 2020-07-22]. Dostupné z: https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c.

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

[36] Simonyan, K. a Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio, Y. a LeCun, Y., ed. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[37] Sánchez, J. A., Romero, V., Toselli, A. H. a Vidal, E. ICFHR2014 Competition on Handwritten Text Recognition on Transcriptorium Datasets (HTRtS). In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. 2014, s. 785–790.

[38] Wang, T., Wu, D. J., Coates, A. a Ng, A. Y. End-to-end text recognition with convolutional neural networks. In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. 2012, s. 3304–3308.

Příloha A

Obsah přiloženého paměťového média

- **doc** - zdrojové soubory LaTeX, text této práce
- **outputs** - natrénované sítě s obrázky výstupů jejich vrstev
- **source_code** - zdrojové soubory neuronové sítě spolu s datovými sadami
- **README.md** - soubor s detailnějším popisem dat na paměťovém médiu