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

NEURONOVÉ SÍTĚ TYPU TRANSFORMER PRO PŘEPIS RUČNĚ PSANÉHO TEXTU

TRANSFORMER NEURAL NETWORKS FOR HANDWRITTEN TEXT RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PETER VEŠELÍNY**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN KOHÚT**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Vešelíny Peter, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Strojové učení
Název: **Neuronové sítě typu Transformer pro přepis ručně psaného textu**
Transformer Neural Networks for Handwritten Text Recognition
Kategorie: Zpracování obrazu
Zadání:

1. Seznamte se s OCR systémem projektu PERO a autoregresivními dekodéry typu Transformer.
2. Vytvořte si přehled o sequence-to-sequence metodách pro rozpoznávání ručně psaného textu.
3. Navrhněte vhodný přístup rozšiřující současné neuronové sítě používané v projektu PERO pro automatický přepis ručně psaného textu o autoregresivní dekodér inspirovaný sítěmi typu Transformer využívající attention vrstvy. Zaměřte se na možnost využití i čistě textových korpusů pro trénování takové sítě.
4. Připravte si vhodné datové sady pro experimenty.
5. Vyhodnoťe vlastnosti navržených neuronových sítí na datové sadě.
6. Zhodnoťte výsledky experimentů a porovnejte je s výsledky systému PERO s odděleným jazykovým modelem.
7. Vytvořte krátké video prezentující výsledky vaší práce.

Literatura:

- Kang, L., Riba, P., Rusiol, M., Fornés, A. and Villegas, M., 2020. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *arXiv preprint arXiv:2005.13044*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

Při obhajobě semestrální části projektu je požadováno:

- Body zadání 1 až 3.

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

Vedoucí práce: **Kohút Jan, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Cieľom tejto práce je navrhnúť systém používajúci sieť typu transformer a uskutočniť s touto sieťou experimenty pri rozpoznávaní ručne písaného textu. V rámci práce sa používa multilingválna dátová sada, v ktorej prevažujú české texty. Pri experimentovaní sa zisťuje vplyv základných hyperparametrov siete, akými sú veľkosť siete, typ konvolučného kodéra a použitie rôznej tokenizácie textu. V práci ďalej využívam textové korpusy českého jazyka, ktoré sa používajú pri trénovaní dekódera. Ďalej v práci experimentujem s použitím dodatočnej textovej informácie pri procese dekódovania. Táto informácia pochádza z predchádzajúceho riadka prepisovaného obrázka s textom. Transformer dosahuje na testovacej dátovej sade chybovosť pri rozpoznávní znakov 3,41 %, čo je o 0,16 % horší výkon ako dosahuje rekurentná neurónová sieť. Pre porovnanie s ostatnými transformer modelmi z dostupných článkov, bola sieť natrénovaná na dátovej sade IAM, na ktorej dosiahla chybu v hodnote 2,48 %, a tým prekonala ostatné transformer modely pri rozpoznávaní ručne písaného textu.

Abstract

This Master's thesis aims to design a system using the transformer neural network and perform experiments with this proposed model in the task of handwriting text recognition. In this thesis, a multilingual dataset with predominate Czech texts is used. The experiments examine the influence of basic hyperparameters, such as network size, convolutional encoder type, and the use of different text tokenizers. In this work, I also use text corpora of the Czech language which is used to train the network decoder. Furthermore, I experiment with the usage of additional textual information during the decoding process. This information comes from the previous line of the transcribed image. The transformer achieves a character recognition error rate of 3.41 % on the test data set which is 0.16 % worse performance than the recurrent neural network achieves. To compare this model with other transformer-based models from available articles, the network was trained on the IAM dataset, where it achieved an error of 2.48 % and therefore outperformed other models in handwriting text recognition task.

Kľúčové slová

rozpoznávanie textu, ručne písaný text, neurónové siete, attention, transformer, textový korpus

Keywords

text recognition, handwriting text, neural networks, attention, transformer, text corpus

Citácia

VEŠELÍNY, Peter. *Neuronové sítě typu Transformer pro přepis ručně psaného textu.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kohút

Neuronové sítě typu Transformer pro přepis ručně psaného textu

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Jána Kohúta. Uviedol som všetky literárne pramene, publikácie a dalšie zdroje, z ktorých som čerpal.

.....
Peter Vešelíny
17. mája 2022

Poďakovanie

Týmto by som chcel poďakovať svojmu vedúcemu Ing. Jánovi Kohútovi za odborné vedenie a rady, ktoré mi pomohli počas písania tejto práce. Ďalej by som chcel poďakovať pánovi Ing. Michalovi Hradišovi, Phd. a ostaným členom projektu PERO za obstaranie dátových sád a kostry neurónovej siete. Táto práca vznikla za podpory projektu e-Infrastruktura CZ (e-INFRA CZ LM2018140) financovaného z programu MŠMT ČR.

Obsah

1 Úvod **2**

2 Rozpoznávanie textu s použitím neurónových sietí **3**
2.1 Segmentácia textu 3
2.2 Konvolučné neurónové siete 4
2.3 Rekurentné neurónové siete 5
2.4 Sequence-to-sequence modely 10

3 Neurónová sieť typu transformer **13**
3.1 Blok kodérov a dekodérov 13
3.2 Mechanizmus Attention 14
3.3 Pozičné kódovanie a embedding 15
3.4 Modifikácia siete transformer pre rozpoznávanie textu 16

4 Dátové sady pre rozpoznávanie ručne písaného textu **20**
4.1 Prehľad existujúcich dátových sád 20
4.2 Dátová sada PERO HWR 21

5 Návrh a implementácia neurónovej siete transformer **24**
5.1 Prostredie pre trénovanie siete 24
5.2 Popis siete pre rozpoznávanie textu 24
5.3 Tokenizácia textu 26
5.4 Použitie textových korpusov pri trénovaní siete 27
5.5 Trénovanie siete s použitím textových prefixov 28

6 Experimenty **31**
6.1 Vplyv veľkosti a typu konvolučného kodéra na úspešnosť modelu 31
6.2 Vplyv veľkosti siete transformer na úspešnosť predikcie znakov 34
6.3 Vplyv rozličnej tokenizácie na úspešnosť siete 37
6.4 Trénovanie modelu s textovým korpusom 38
6.5 Trénovanie siete a dekódovanie sekvencie s použitým textového prefixu . . . 41
6.6 Zhrnutie výsledkov 42

7 Záver **43**

Literatúra **44**

A Obsah priloženého DVD **48**

Kapitola 1

Úvod

Rozpoznávanie textu (angl. *Optical Character Recognition*, OCR) je proces, ktorý umožňuje automatické rozpoznávanie textových znakov v obraze. Rozpoznávanie ručne písaného textu (angl. *Handwriting Text Recognition*, HTR) sa oproti OCR zameriava konkrétne na ručne písaný text, ktorý býva z pravidla náročnejšie rozpoznávať oproti bežnému tlačenému textu. Toto je spôsobené zvyčajne nižšou kvalitou rozpoznávaného obrazu, vyššej variabilite štýlov písma medzi pisateľmi, a zároveň aj variabilite textu toho istého pisateľa.

Pre potreby HTR sa v súčasnosti používajú prevažne rozličné typy neurónových sietí, ktoré dosahujú v tejto oblasti najlepšie výsledky. Toto je spôsobené hlavne veľkým množstvom dát v podobe rozsiahlych dátových sád a dostatkom výpočtovej sily v podobe grafických procesorov. Z neurónových sietí dosahujú najlepšie výsledky prevažne rekurentné neurónové siete (angl. *Recurrent Neural Network*, RNN), ktoré sa typicky používajú pri rozpoznávaní sekvencií. Okrem štandardných prístupov používajúcich RNN existujú aj modely, ktoré sa snažia obísť použitie rekurentných vrstiev. Medzi tieto modely je zaraďovaná aj architektúra typu *transformer*. Transformer predstavuje model, ktorý v sebe využíva tzv. *multi-head attention* mechanizmus, ktorý kompletne obchádza použitie rekurentných vrstiev. V súčasnosti nie je úplne známe, či dokáže architektúra transformer prekonať rekurentné siete v oblasti HTR.

V tejto práci pracujem s existujúcim systémom projektu PERO[1], v ktorom je implementovaná architektúra siete transformer. Cieľom tejto práce je overiť, aké výsledky dokáže táto sieť priniesť a porovnať ju s natrénovanými modelmi projektu PERO. V práci sa okrem základných hyperparametrov siete overuje použitie rozličnej tokenizácie textu, ktoré nie sú pri úlohách HTR typické. Rovnako sa ďalej overuje trénovanie siete spolu s dodatočnými textovými korpusmi. Trénovanie takto kombinuje metódy trénovania na obrázkoch a aj samostatnom texte. V poslednej časti bol skúmaný vplyv dekódovania textu s dodatočnou textovou informáciou na začiatku dekódovanej sekvencie. U takéhoto dekódovania je vstupom dekodéra dodatočná textová informácia z predchádzajúcich riadkov textu.

Táto práca je členená nasledovne: V kapitole 2 sa popisuje prehľad o neurónových sieťach, ktoré sa používajú pri rozpoznávaní textu. V kapitole 3 je detailnejšie popísaný rozbor architektúry transformer. V kapitole 4 je prehľad dátových sád a zvolené sady používané pri trénovaní neurónovej siete, ktorá je popísaná v časti 5. V časti 6 sú popísané všetky vykonané experimenty a zhodnotené výsledky. V kapitole 7 je zhrnutie celej práce a popísané plány do budúcna.

[1] `https://pero.fit.vutbr.cz/`

Kapitola 2

Rozpoznávanie textu s použitím neurónových sietí

Úlohou rozpoznávania textu (angl. *Optical Character Recognition*, OCR) je automaticky získať prepis textovej časti z obrazu (obr. 2.1). Pred samotným rozpoznávaním textu sú v obraze zvyčajne detekované rôzne typy textových regiónov a následne sú z nich extrahované textové riadky. Tieto extrahované riadky predstavujú následne vstupy systémov pre rozpoznávanie textu. Existujúce nástroje používané pri automatickom rozpoznávaní textu sú napríklad *ABBY FineReader*[1], *Tesseract*[2] a *Transcribus*[3].

V súčasnosti dosahujú *state-of-the-art* výsledky v oblasti rozpoznávanie textu rekurentné neurónové siete [4, 35, 41] s *Long Short-Term Memory* modulom [17]. Sieť od autorov *Théodore Bluche a spol.* [4] dosahuje na dátovej sade IAM [30] chybu o veľkosti 3.20% v nesprávne rozpoznaných znakoch. Okrem štandardných rekurentných sietí sa používajú aj ďalšie architektúry používajúce tzv. *sequence-to-sequence* prístupy [19, 44, 8, 31] alebo architektúry výhradne založené na mechanizme *attention* [18, 28, 48].

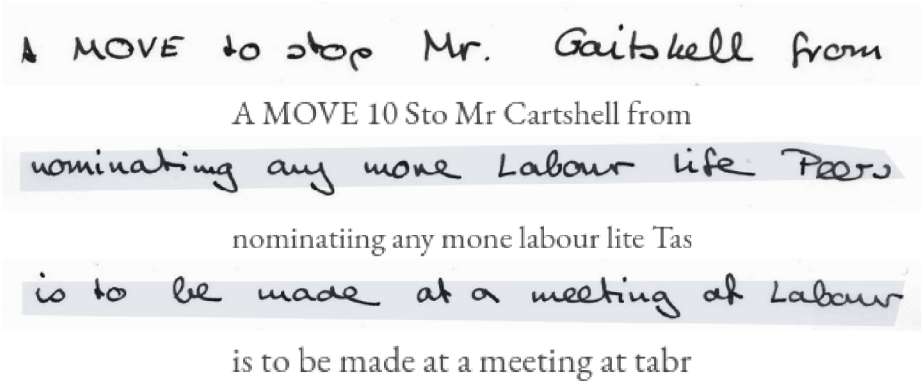
V prvej časti 2.1 tejto kapitoly je stručne popísaný proces segmentácie textu, ktorého výstupom sú detekované textové regióny. V ďalšej časti sú popísané typy neurónových sietí, ktoré sa používajú pri rozpoznávaní textu. Konkrétne ide o konvolučné neurónové siete (2.2) a rekurentné neurónové siete (2.3) spolu s rozšírením *Long Short-Term Memory* (2.3). Ďalej je popísaná chybová funkcia *Connectionist temporal classification* (2.3), ktorá sa používa práve pri rozpoznávaní sekvencií. V poslednej časti sú popísané *sequence-to-sequence* modely (2.4).

2.1 Segmentácia textu

V rámci segmentácie textu dochádza k identifikácií textových regiónov, ktorými sú napríklad odstavce, nadpisy a riadky. Táto identifikácia textových regiónov z obrazu predstavuje jeden z kľúčových aspektov rozpoznávania textu, pretože najlepší výkon pri rozpoznávaní textu dosahujú práve algoritmy, ktoré majú na svojom vstupe segmentované riadky textu [15].

Výstupom segmentácie bývajú zvyčajne súradnice niekoľkých bodov. Tieto body vytvárajú rozličné typy reprezentácií, ktoré po prepojení pomocou spojníc ohraničujú detekovaný

[1] https://pdf.abbyy.com/
[2] https://github.com/tesseract-ocr/tesseract
[3] https://transkribus.eu/lite/



Obr. 2.1: Ukážka rozpoznaného textu nástrojom *Transkribus*. Obrazy s textom pochádzajú z dátovej sady IAM [30]. Nástroj Transkribus nedokázal získať prepisy riadkov s nulovou chybovosťou.

text. Tieto reprezentácie môžu byť *bounding boxy*, *x-height* oblasti alebo rozličné polygonálne reprezentácie. Ďalšou reprezentáciou môže byť poloha základnej línie, na ktorej leží väčšina znakov vety, tzv. *baseline*. Na základe týchto reprezentácií je možné extrahovať časť obrazu s textovou časťou a použiť ju pri rozpoznávaní [15]. Ukážka segmentácie riadkov pomocou nástroja *Transkribus* je na obrázku 2.2.

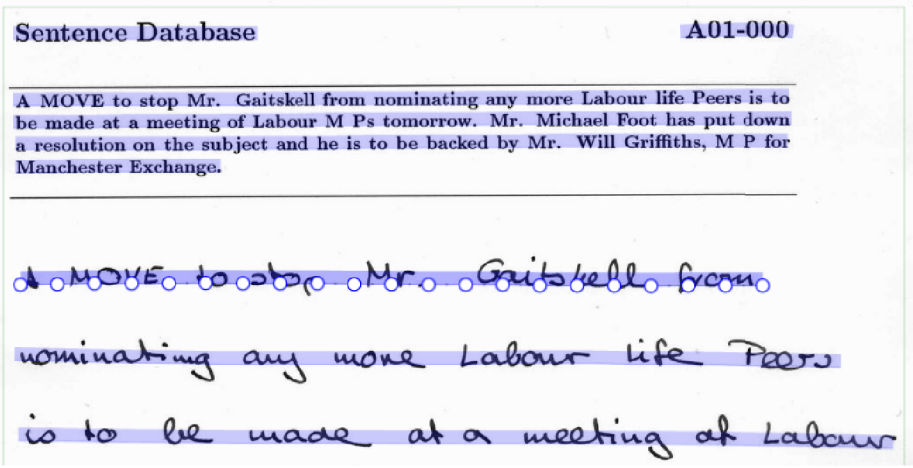
Najlepšie *state-of-the-art* výsledky pri segmentácií textových riadkov dosahuje sieť nazývaná *ARU-Net* [15] alebo jej modifikácie [22]. *ARU-Net* rozširuje koncept siete typu *U-Net* [36], ktorá sa používa pri sémantickej segmentácií, o reziduálne spojenia [16], a zavádza použitie tzv. *spatial attention* mechanizmu. V prvej etape sú vstupný obraz a jeho zmenšené verzie transformované dvomi jednotkami nazývanými *A-Net* a *RU-Net*. *A-Net* aplikuje na svoje vstupy attention mechanizmus, zatiaľčo *RU-Net* používa dekonvolúciu. Vďaka viacerým vrstvám, pracujúcich v rôznych rozlíšeniach, umožňuje *ARU-Net* klásť dôraz na rôzne miesta v obraze. Výstupom siete sú následne klasifikované pixely obrazu, ktoré sa v druhej etape zhlukujú a vytvárajú tzv. superpixely. Tie sú následne použité pri vytváraní línie baseline [15].

2.2 Konvolučné neurónové siete

Konvolučné neurónové siete (*Convolutional Neural Networks*, CNN) [10] sú neurónové siete, ktoré sa používajú predovšetkým pri spracovaní obrazu. V architektúrach konvolučných sietí sa vyskytujú dve špecializované typy vrstiev, ktoré sa nazývajú konvolučné vrstvy (*convolutional layers*) a tzv. *pooling* vrstvy. Ukážka základnej architektúry konvolučnej siete je na obrázku 2.3.

Konvolučné vrstvy slúžia na extrahovanie vizuálnych príznakov lokálnych spojení a vytvárajú príznakové mapy (angl. *feature maps*). Tieto mapy sa vytvárajú postupným aplikovaním filtra, ktorý nazývame konvolučné jadro K (*convolution kernel*). Jednotlivé filtre konvolučných vrstiev predstavujú trénovateľné parametre, ktoré sa sieť počas trénovaní učí.

Matematicky môžeme aplikáciu filtra popísať operáciou, ktorá sa nazýva konvolúcia $*$ (angl. *convolution*) (2.1). Niekedy sa táto operácia v CNN označuje aj ako korelácia (angl. *cross-correlation*), ale v kontexte CNN však nie je potrebné tieto operácie rozlišovať [10].



Obr. 2.2: Ukážka segmentácie textových riadkov nástrojom *Transkribus*. Pôvodný obraz pochádza z dátovej sady IAM [30]. Svetlomodrou farbou sú označené detekované textové riadky. Svetlomodrá čiara definovaná N bielymi bodmi predstavuje líniu *baseline*.

Konvolúcia sa pri dvojrozmernom obraze I počíta ako:

$$(I \star K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)\,, \tag{2.1}$$

kde i, j sú osy konvolučného jadra a m, n sú osy obrazu I.

Úlohou *pooling* vrstvy je vytvárať reprezentáciu, ktorá je aproximačne invariantná voči malým posunom. Typicky sa to uskutočňuje výpočtom na základe maximálnej hodnoty (*max pooling*) alebo priemeru (*average pooling*). V súvislosti s obrazom ide o zmenu jeho výšky a šírky [10].

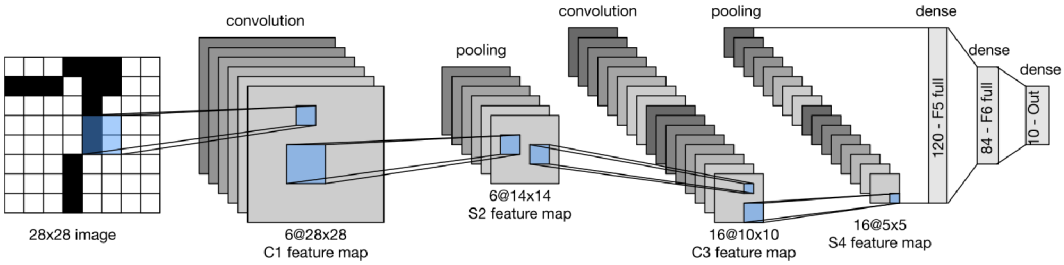
Riešenie založené na konvolučných sieťach Konvolučnú neurónovú sieť pre rozpoznávanie číslic použili v roku 1998 *Yann LeCun a spol.* [26]. Ich neurónová sieť nazývaná *LeNet-5* [26] (obr. 2.3) dokázala automaticky extrahovať významné črty a automaticky rozpoznávať ručne písané číslice z dátovej sady MNIST[4]. Architektúra ich siete pozostáva z dvoch dvojíc konvolučných a *pooling* vrstviev, ktoré sú nasledované tromi plne prepojenými vrstvami. V súčasnosti sa samostatné konvolučné siete pri rozpoznávaní textu nepoužívajú, ale používajú sa ako extraktory vizuálnej informácie z obrazu v architektúrach spolu s rekurentnými neurónovými sieťami alebo v sieťach typu transformer.

2.3 Rekurentné neurónové siete

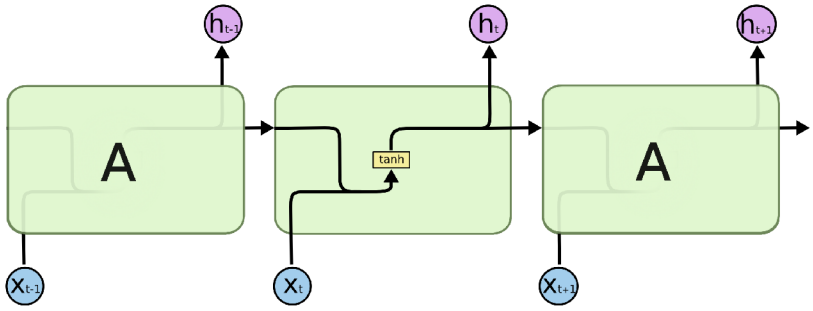
Rekurentné neurónové siete (*Recurrent neural network*, RNN) [10] sú neurónové siete, ktoré sa používajú predovšetkým pri spracovávaní dát sekvenčného charakteru tvaru $x_1, \ldots, x_n$, kde x_1 a x_n sú konkrétne hodnoty symbolov sekvencie o dĺžke n. Medzi dáta sekvenčného charakteru patria aj písané texty. Jednotlivé znaky a slová majú medzi sebou vo vetách rozličné jazykové väzby, ktoré nedokážu samostatné konvolučné neurónové siete zachytiť.

RNN dokážu uchovávať svoj vnútorný stav, nazývaný ako skrytý stav (angl. *hidden state*) v podobe stavového vektora h_t (2.2) v čase t, ktorý potom odovzdávajú ďalej sieťou.

[4] `http://yann.lecun.com/exdb/mnist/`



Obr. 2.3: Architektúra konvolučnej siete *LeNet-5* siete prezentovaná autormi *Yann LeCun a spol.* [26]. Ide o štandardnú konvolučnú neurónovú sieť, ktorá obsahuje dve konvolučné vrstvy (*convolution*) a dve *pooling* vrtsvty (*pooling*). Po každej konvolučnej vrstve dochádza ku vytváraní príznakovej mapy C a po každej *pooling* vrstve dochádza ku zmenšeniu vstupu. Na konci siete je plne prepojená vrstva. Sieť sa použila pri klasifikácií obrázkov číslic. Obrázok je prevzatý z [49].



Obr. 2.4: Ukážka blokov štandardnej rekurentnej siete s aktivačnou funkciou *tanh*. V čase t je vstupom modulu symbol x_t sekvencie x_{t-1}, x_t, x_{t+1} a tzv. skrytý stav (angl. *hidden state*) h_{t-1}. Obrázok je prevzatý z [33].

Takýmto spôsobom si RNN môžu uchovávať informáciu o predošlých častiach sekvencie. Hodnota vnútorného stavu v čase t je závislá na hodnote predchádzajúceho stavu h_{t-1}, vstupnom symbole x_t a váhami siete θ:

$$h_t = f(h_{t-1}, x_t, \theta)\,. \tag{2.2}$$

Počiatočný stav h_0 býva explicitne špecifikovaný. Ukážka blokov rekurentnej vrstvy a výpočet vnútorného stavu je na obrázku 2.4.

Dopredný prechod je vo svojej podstate sekvenčný, pretože každý výstup x_t v časovom kroku t môže byť vypočítaný až po získaní hodnoty h_{t-1} z predchádzajúceho časového kroku. Kvôli tomuto nie je možné výpočet rekurentnej vrstvy zefektívniť pomocou paralelizácie. Cena výpočtu je tým pádom lineárna a má pri sekvencii o dĺžke n hodnotu $\mathcal{O}(n)$.

Algoritmus, ktorý sa používa pri optimalizácií váh rekurentnej siete sa označuje ako *back-propagation through time* (BPTT). Ide však o bežný *back-propagation* algoritmus, ktorý sa použije na rozvinutý výpočtový graf [10].

V praxi vznikajú v rámci sekvencie aj závislosti na budúcich symboloch, a preto by sme chceli vytvoriť predikciu, ktorá závisí na celej sekvencií. Napríklad pri spracovaní textu môže závisieť slovo od niekoľkých ďalších nasledujúcich slov z dôvodu jazykových závislostí. Pre tieto potreby sa používajú tzv. obojsmerné rekurentné siete (*Bidirectional recurrent neural networks*), ktoré predstavujú spojenie dvoch rekurentných sietí. Jedna z týchto sietí spracováva sekvenciu x_{t-1}, x_t, x_{t+1} v doprednom smere od x_{t-1} po x_{t+1}, zatiaľčo druhá spracováva sekvenciu v spätnom smere od x_{t+1} po x_{t-1}. Týmto sa umožňuje pracovať s informáciou, ktorá je z minulosti, a zároveň aj z budúcnosti. Výstupy oboch sietí sú v každom časovom kroku spojené do jedného výstupného vektora [10].

Rekuretné siete bývajú častokrát trénované pomocou procedúry nazývanej *teacher forcing*. Ide o procedúru, pri ktorej sa počas trénovania siete používa namiesto výstupu siete x_t hodnota y_t (očakávaný výstup, *ground truth*) ako vstup v nasledujúcom časovom kroku $t + 1$. Použitie tejto procedúry vyplíva z *maximum-likelihood* estimácie [10].

Pri trénovaní RNN vzniká častokrát problém, kedy sieť nie je možné efektívne trénovať metódou gradientného zostupu. Gradient, ktorý je propagovaný pri optimalizácií siete, v ktorej sú vyžadované dlhodobé závislosti v sekvencií, má tendenciu väčšinou zmiznúť (angl. *vanishing gradient*) alebo zriedka explodovať (angl. *exploding gradient*). Táto zmena gradientu má za následok narušenie priebehu optimalizácie neurónovej siete [3]. Kvôli týmto problémom sa v praxi pri trénovaní rekurentných sietí používajú špecializované jednotky *Long Short-Term Memory* (LSTM) [17] a *Gated Recurrent Unit* (GRU) [5], ktoré čiastočne zamedzujú problémom pri trénovaní dlhodobých závislostí.

Long Short-Term Memory

.

Long Short-Term Memory (LSTM) [17] je jednotka, ktorá sa používa ako súčasť bloku rekurentnej neurónovej siete a zamedzuje problému učenia dlhodobých závislostí. Kľúčovými komponentami bloku sú tzv. stav bunky C_t (angl. *cell state*) a skrytý stav h_t (angl. *hidden state*). Ďalej obsahuje jednotka LSTM tri časti, ktoré sa nazývajú brány a ich úlohou je regulovať tok informácie prechádzajúci týmito stavmi jednotlivých buniek. Brány sa postupne nazývajú brána zabudnutia f_t (angl. *forget gate*), vstupná brána i_t (angl. *input gate*) a výstupná brána o_t (angl. *output gate*). Všetky tri brány pozostávajú z logistickej sigmoidy σ, ktorej výstup udáva, koľko informácie danou bránou pretečie a koľko sa zahodí. Ukážka jednotky LSTM je na obrázku 2.5.

Úlohou *forget gate* (2.3) je regulovať tok informácie skrytého stavu z predchádzajúceho časového kroku h_{t-1}:

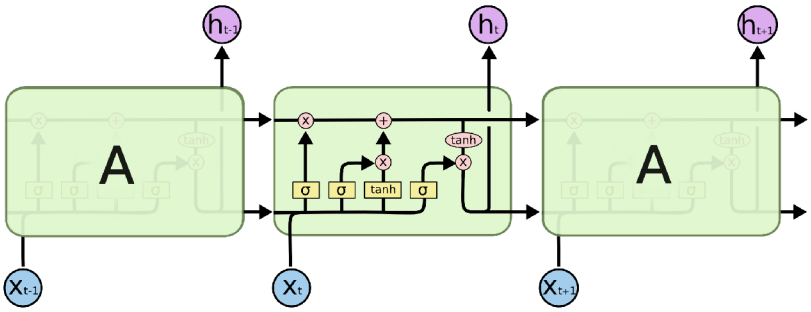
$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)\,, \tag{2.3}$$

kde W_f a b_f označujú váhy a bias a x_t označuje vstup bloku.

Ďalšiu časť jednotky tvorí dvojica *input gate* (2.4) a vektor nových potencionálnych kandidátov pre uchovanie $\widetilde{C}_t$ (2.5). *Input gate* pracuje na podobnom princípe ako *forget gate*, zatiaľčo vektor $\widetilde{C}_t$ reprezentuje novú informáciu, ktorá môže byť pridaná ku existujúcemu stavu bunky C_t:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)\,, \tag{2.4}$$

$$\widetilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)\,, \tag{2.5}$$



Obr. 2.5: Ukážka blokov rekurentnej siete s jednotkou *Long Short-Term Memory* (LSTM) [17], ktorá umožňuje prácu s dlhodobými závislosťami v sekvencii x_{t-1}, x_t, x_{t+1}. LSTM blok obsahuje tri vzájomne pôsobiace časti, ktoré sa nazývajú brány. Význam brán je popísaný v textovej časti 2.3. Obrázok je prevzatý z [33].

kde W_i, W_C, b_i, b_C označujú váhy a bias, zatiaľčo tanh je hyperbolický tangens, ktorý sa používa pre transformáciu informácie.

Nový uchovávaný stav bunky C_t (2.6) je následne charakterizovaný hodnotami z predchádzajúcich dvoch brán f_t a i_t, hodnotou starého uchovávaného stavu bunky C_{t-1} a hodnotou vektora nových potencionálnych kandidátov $\widetilde{C}_t$ podľa vzťahu:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \,. \tag{2.6}$$

Nový skrytý stav, resp. výstup bloku h_t (2.8) je následne vypočítaný súčinom hodnôt brány *output gate* (2.7) a transformovaným stavom bunky C_t (2.6) pomocou hyperbolického tangensu tanh:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t * tanh(C_t)\,, \tag{2.8}$$

kde W_o, b_o sú váhy a bias [33, 17].

Connectionist temporal classification

Bežná rekurentná sieť produkuje v každom časovom kroku t výstupný vektor x_t, ktorý má veľkosť abecedy rozpoznávaných symbolov. Z tohoto vektora sa získa odpovedajúci znak a ten sa použije ako výstup v časovom kroku t. Pri trénovaní siete je následne potrebné, aby odpovedajúci výstup siete y bol rozdelený na segmenty podľa jednotlivých časových krokov $y = x_1, \ldots, x_t, \ldots, x_n$. Takéto anotovanie segmentov je však časovo náročná záležitosť.

Ďalej tento postup prináša komplikácie, kedy môže byť napríklad jeden symbol v sekvencií generovaný vo viacerých časových krokoch, pričom musíme dokázať odlíšiť dva prípady: v prvom prípade ide o nesprávne dekódovanie a duplicity sa musia odstrániť, zatiaľčo v druhom prípade ide o správne slovo a duplicity sa musia ponechať. Pri tomto probléme je potrebné zaviesť dodatočné kroky ako dokázať oddeliť tieto dva prípady.

Connectionist temporal classification (CTC) [11] je objektívna funkcia, ktorá sa používa pri rozpoznávaní sekvenčných dát a umožňuje sieť trénovať priamo bez rozdelenia odpovedajúcich dát používaných pri trénovaní, na segmenty. CTC využíva kódovanie, ktoré pri opakujúcich sa symboloch výstupné symboly spojí do jedného symbolu. Ďalej zavádza špeciálny symbol nazývaný *blank*, ktorý slúži na oddeľovanie skutočne sa opakujúcich symbolov v odpovedajúcom výstupe y.

Rozhodujúcim krokom CTC je výpočet podmienenej pravdepodobnosti $p(\pi|x)$ vstupnej sekvencie x pri určenej ceste π naprieč všetkými pozorovaniami, formálne ako:

$$p(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \tag{2.9}$$

kde T je dĺžka sekvencie a π_t predstavuje symbol odpozorovaný v časovom kroku t.

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

$$p(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} p(\pi|x). \tag{2.10}$$

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

$$O^{CTC} = -\sum_{(x,z) \in S} ln(p(z|x))\,, \tag{2.11}$$

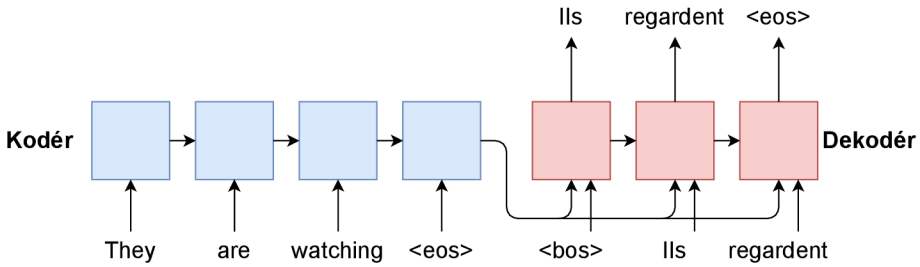
kde (x, z) predstavuje dvojicu vstupnej a cieľovej sekvencie. Podrobne popísaný postup je vysvetlený v pôvodnej práci [11].

Riešenia založené na rekurentných sieťach V roku 2006 predstavili *Alex Graves a spol.* objektívnu funkciu s názvom *Connectionist Temporal Classification* [12], vďaka ktorej umožnili trénovanie rekurentných neurónových sietí bez potreby segmentovania dát a dodatočného spracovania. Následne predviedli *Alex Graves a spol.* rozšírený systém [13] pre rozpoznávanie ručne písaných slov a podarilo sa im dosiahnuť nové *state-of-the-art* výsledky a poraziť systémy založené na skrytých Markovových modeloch (angl. *Hidden Markov Models*, HMM). V tomto systéme použili obojsmernú *Long Short-Term Memory* sieť [17], ktorej vstupom bolo nimi definovaných 25 rôznych charakteristík textu z obrazu. Použitie LSTM umožnilo uchovať informáciu o kontexte na dlhý dosah, čo bolo práve obmedzením klasických RNN.

Ďalším rozšírením bolo, keď *Alex Graves a spol.* použili multi-dimenzionálne LSTM siete (angl. *Multi Dimensional Long Short-Term Memory*, MDLSTM) [14], ktoré namiesto extrahovaných charakteristík dostali na vstup nespracovaný obraz pixelov. Vďaka tomuto sa dokázali vyhnúť potrebe manuálneho definovania charakteristík a ich extrakcie z obrazu. Hlavnou myšlienkou MDLSTM je, že jedno rekurentné spojenie, ktoré je v štandardných RNN, nahradzuje viacero rekurentných spojení podľa počtu dimenzií v dátach.

Baoguang Shi a spol. predstavili sieť s názvom *Convolutional Recurrent Neural Network* (CRNN) [41], ktorá kombinuje hlbokú konvolučnú sieť s rekurentnou sieťou, a túto sieť použili pri rozpoznávaní textu v scéne. Ich architektúra siete obsahuje tri časti: CNN, ktorá v tomto prípade extrahuje vizuálne charakteristiky zo sekvencie, RNN, ktorá následne predikuje symbol v jednotlivých krokoch a poslednú časť tvorí tzv. vrstva prepisu (angl. *transcription layer*), ktorú tvorí blok s CTC. Ich sieť dokázala pracovať v dvoch nastaveniach: s použitím slovníka a bez použitia slovníka.

Joan Puigcerver a spol. [35] potvrdili, že LSTM siete dokážu dosiahnuť rovnako dobré výsledky ako MDLSTM siete, pričom dokážu bežať viditeľne rýchlejšie. Podľa nich sú síce MDLSTM viac výkonnejšie, ale pravdepodobne táto výkonnosť nie je u rozpoznávania textu



Obr. 2.6: Architektúra *sequence-to-sequence* siete prezentovaná autormi *Ilya Sutskever a spol.* [45]. Sieť číta vetu v anglickom jazyku a produkuje jej preklad vo francúzštine. Model prestáva predikovať sekvenciu po vygenerovaní symbola konca sekvencie $\langle eos \rangle$. Symbol $\langle bos \rangle$ označuje začiatok sekvencie. Dekodér pri dekódovaní pracuje s výstupom kodéra, a zároveň výstupom dekodéra z času $t-1$.

potrebná, pretože sa podobné funkcie naučila aj ich LSTM sieť. Ich architektúra bola podobná ako tá od autorov *Baoguang Shi a spol.* [41], t.j. obsahuje CNN, LSTM a CTC bloky, ale namiesto rozpoznávania textu v scéne je použili pri rozpoznávaní riadkov textu.

Théodore Bluche a spol. [4] použili sieť, ktorá obsahuje dve časti, ktorými sú konvolučný kodér a LSTM dekodér z obojsmernej LSTM siete. Ich konvolučný kodér obsahoval tzv. konvolučné brány (angl. *convolutional gates*), ktoré boli kľúčom k ich úspechu. Úlohou takejto brány je rozhodnúť podľa hodnoty funkcie na danej pozícií a pri jej susedných hodnotách, či je táto vlastnosť pri tejto pozícii významná alebo nie. Pri trénovaní sa zamerali na multilingválne dáta a dokázali natrénovať konvolučný kodér, ktorý získal generické jazykové vlastnosti.

2.4 Sequence-to-sequence modely

Štandardná *sequence-to-sequence* architektúra [45] pozostáva z dvoch rekurentných sietí, ktoré sa postupne nazývajú kodér (angl. *encoder*) a dekodér (angl. *decoder*). Tento typ siete umožňuje mapovanie medzi dvoma sekvenciami premenlivej dĺžky.

Úlohou kodéra je spracovať vstupnú sekvenciu premenlivej dĺžky a vytvoriť jej reprezentáciu v podobe vektora fixnej dĺžky nazývaného *context vector* c (2.13). Dekodér sa inicializuje výstupom kodéra c a postupne symbol po symbole vytvára výstupnú sekvenciu y (2.14). Koniec procesu dekódovania zvyčajne nastane vygenerovaním ukončovacieho symbolu sekvencie $\langle eos \rangle$. Ukážka *sequence-to-sequence* architektúry je na obrázku 2.6.

Formálne môžeme proces transformácie vstupu x za pomoci kodéra popísať ako:

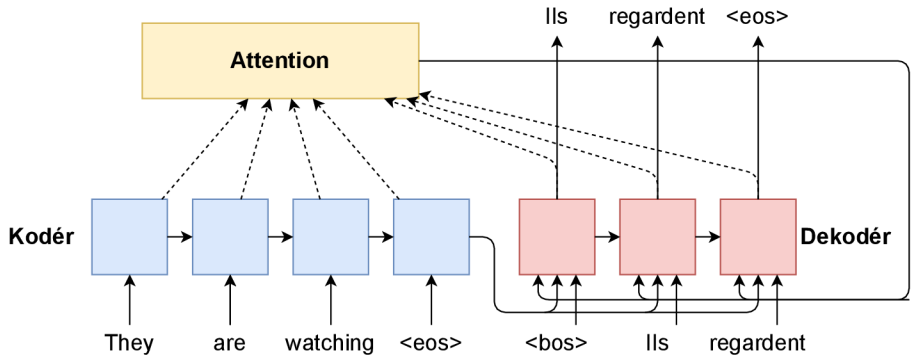
$$h_t = f(x_t, h_{t-1}), \tag{2.12}$$

$$c = q(h_1, \ldots, h_{T_x}), \tag{2.13}$$

kde t je časový krok, h_t je skrytý stav kodéra v časovom kroku t a f, g sú nelineárne funkcie.

Proces dekódovanie sekvencie y môžeme následne zapísať ako podmienenú pravdepodobnosť časti sekvencie $y_1, \ldots, y_{t-1}$ generovanej v predchádzajúcich časových krokoch $1, \ldots, t-1$ a kontextového vektora c:

$$p(y) = \prod_{t=1}^{T} p(y_t \mid c, y_1, \ldots, y_{t-1}). \tag{2.14}$$



Obr. 2.7: Architektúra *sequence-to-sequence* s mechanizmom *attention* [2]. Oproti pôvodnej *sequence-to-sequence* architektúre (2.6) používa dekodér vo fáze dekódovania informácie poskytnuté *attention* modulom. Prerušované čiary značia skryté stavy kodéra a dekodéra.

Dzmitry Bahdanau a spol. v ich práci [2] rozšírili pôvodný koncept *sequence-to-sequence* siete a zavidli mechanizmus zvaný *attention* 2.16. Rozšírenie sa týka fázy dekódovania sekvencie, kedy vďaka *attention* mechanizmu môže dekodér prehľadávať časti zdrojovej sekvencie a zachytiť relevantné informácie, na ktoré kladie dôraz. *Attention* predstavuje skratku medzi kontextovým vektorom a celým zdrojovým vstupom. Ukážka modifikovanej architektúry je na obrázku 2.7.

Formálne ide o zmenu rovnice 2.14 na nový tvar:

$$p(y_t \mid y_1, \ldots, y_{t-1}, x) = g(y_{t-1}, s_t, c_t)\,, \tag{2.15}$$

kde s_t je skrytý stav dekodéra v čase t, ktorý sa počíta ako:

$$s_t = f(s_{t-1}, y_{t-1}, c_t)\,. \tag{2.16}$$

V novom vzťahu už kontextový vektor c nezávisí podľa 2.13 na sekvencií anotácií $h_1, \ldots, h_{T_x}$, na ktoré mapuje svoju vstupnú sekvenciu x, ale každá anotácia h_t obsahuje informácie o celej vstupnej sekvencií so silným zameraním na časti obklopujúce slovo alebo symbol v časovom kroku t. Potom je vektor c_t vypočítaný ako vážený súčet anotácií h_t:

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j\,, \tag{2.17}$$

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T_x} exp(e_{tk})}\,, \tag{2.18}$$

$$e_{tj} = score(s_{t-1}, h_j)\,, \tag{2.19}$$

kde α_{tj} je váha anotácie a e_{tj} je tzv. zarovnanie (angl. *alignment model*), ktoré hodnotí ako dobre sa vstupy okolo pozície t zhodujú. V ich práci [2] je toto zarovnanie implementované pomocou plne prepojenej vrstvy:

$$score(s_{t-1}, h_j) = v_a^T tanh(W_a s_{t-1} + U_a h_j)\,, \tag{2.20}$$

kde W_a, U_a, v_a sú váhy a $tanh$ je aktivačná funkcia.

Riešenia založené na sequence-to-sequence sieťach *Ilya Sutskever a spol.* [45] predstavili v roku 2014 sieť, ktorá dokázala mapovať vstupnú sekvenciu symbolov na výstupnú sekvenciu symbolov. Architektúra pozostáva z dvoch častí používajúcich LSTM sieť: kodér a dekodér. Kodér mapuje vstupnú sekvenciu na vektor fixnej dĺžky, ktorý následne dekodér dekóduje do cieľovej sekvencie. Počas experimentov pri strojovom preklade zistili, že ich architektúra dokáže pracovať aj s dlhými sekvenciami a dokázali sieť naučiť rozumné frázové a vetné reprezentácie.

Dzmitry Bahdanau a spol. v ich práci [2] rozšírili koncept *sequence-to-sequence* sietí a použili mechanizmus zvaný *attention*. Podľa nich je použitie kontextového vektora s pevnou dĺžkou problematické pri preklade dlhých sekvencií. Mechanizmus *attention* dokáže nájsť relevantné časti v sekvencií na základe kontextového vektora a skrytého stavu dekodéra. Dekodér sa vďaka tomuto mechanizmu rozhodne, na ktoré časti zdrojovej sekvencie bude venovať svoju pozornosť. Ich práca bola podobne ako [45] použitá pri strojovom preklade a dosiahla u neho nové *state-of-the-art* výsledky.

Chorowski a spol. v ich práci o rozpoznávaní reči [7] rozšírili *attention* mechanizmus tak, aby zohľadňoval dôraz z predchádzajúcich krokov dekódovania. Rovnako zaviedli tzv. *attention smoothing*, ktorý slúži na agregovanie informací z viacerých lokácií v sekvencii.

V tom istom období *Minh-Thang Luong a spol.* demonštrovali vo svojej práci [29] použitie dvoch rozličných *attention* mechanizmov: tzv. *global attention* mechanizmus, ktorý zameriava svoju pozornosť na celú sekvenciu a tzv. *local attention*, ktorý zameriava svoj dôraz iba na zvolenú časť sekvencie. Ich analýza poukazuje, že pri strojovom preklade dosahujú modely založené na *attention* mechanizmoch lepšie výsledky, čo je viditeľné napríklad pri preklade mien alebo manipulácií s dlhými vetami.

Jorge Sueiras a spol. [44] použili architektúru spolu s *attention* mechanizmom pri rozpoznávaní ručne písaných slov. Ich architektúra používa posuvné okno, ktoré postupne extrahuje časti z obrazu a tie následne posúva na vstup CNN a potom kodér-dekodér architektúre. Ako CNN architektúru zvolili obdobu *LeNet-5* [26] a *attention* mechanizmus použili rovnaký ako *Chorowski a spol.* [7].

Podobnú architektúru použili aj *Ariandam Chowdhury a spol.* [8]. Na rozdiel od *Jorge Sueiras a spol.* [44] použili ako vstup konvolučnej siete celé obrázky s textom a odstránili potrebu použitia posuvného okna. V rámci ich práce použili chybovú funkciu *Focal Loss* [27], ktorá má zlepšiť problém spôsobený nevyváženosťou niektorých znakov v dátovej sade. Pre predikovanie sekvencie sa rozhodli použiť *beam search* algoritmus. V práci popisujú, že použitie mechanizmu *attention* poskytuje významné zvýšenie presnosti v porovnaní so štandardnými RNN s CTC sieťami pri rozpoznávaní ručne písaného textu.

Konkurenčné výsledky v rozpoznávaní ručne písaného textu dosiahli aj *Lei Kang a spol.* [19] v roku 2018, kedy sa rozhodli použiť obdobnú *sequence-to-sequence* architektúru ako *Jorge Sueiras a spol.* [44] a *Ariandam Chowdhury a spol.* [8]. V práci preskúmali použitie tzv. *content-based attention* mechanizmu [29] and *location-based attention* mechanizmu [29]. V práci skúmali vplyv techník, ktorými sú: *label smoothing* [46], ktorý slúži na regularizáciu *ground-truth* rozhodnutia, *multi-nomial decoding* [6], ktorý umožňuje dekodéru preskúmať viaceré alternatívne cesty pri dekódovaní a *attention-smoothing* [7]. Najlepšie výsledky dosiahli pri použití *location-based attention* mechanizmu a techniky *label smoothing*.

Experimentálne porovnanie v použití rôznych *attention* mechanizmov a pozičných kódovaní [47] uskutočnili v roku 2019 autori *Johannes Michael a spol.* [31]. Vo svojej práci uskutočnili aj hybridné trénovanie za použitia objektívnej funkcie kombinujúcej CTC s *cross entropy*. Na populárnych ručne písaných datasetoch dosiahli výsledky, ktorými prekonali ostatné *sequence-to-sequence* modely v oblasti HTR.

Kapitola 3

Neurónová sieť typu transformer

Transformer bol prvýkrát predstavený autormi *Vaswani et. al* [47] v publikácií *Attention Is All You Need*. Ide o auto-regresívny[1] model neurónovej siete, ktorý vo svojej pôvodnej architektúre neobsahuje žiadne rekurentné ani konvolučné vrstvy a je plne založený na mechanizme *attention*, ktorý umožňuje modelovanie závislostí bez ohľadu na veľkosti vzdialeností medzi časťami v sekvencii. *Kang a spol.* [18] a *Christoph Wick a spol.* [48] použili modifikovanú architektúru tejto siete pre rozpoznávanie ručne písaného textu.

Transformer prináša oproti rekurentným sieťam výhodu v podobe lepšej možnosti paralelizácie. Pôvodné rekurentné siete sú obmedzené v oblasti paralelizácie výpočtu kvôli povahe rekurentných vrstiev. Zložitosť výpočtu rekurentnej vrstvy má pri sekvencii o dĺžke n a veľkosti dimenzie skrytého stavu d hodnotu $\mathcal{O}(n \cdot d^2)$, pričom sekvenčných je práve $\mathcal{O}(n)$ operácií. Oproti tomu výpočet v *self-attention* vrstve obsahuje celkovo $\mathcal{O}(n^2 \cdot d)$ operácií, z ktorých je sekvenčných práve konštantný počet $\mathcal{O}(1)$ [49, 47].

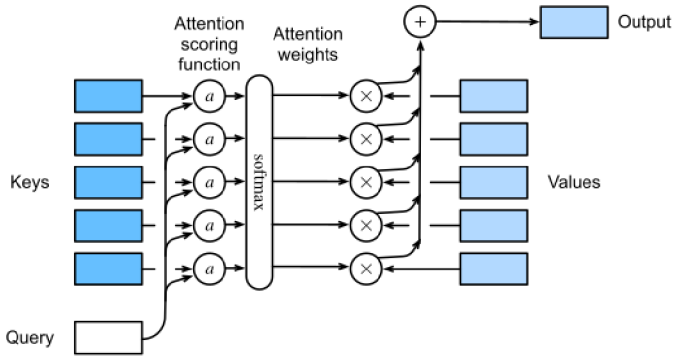
Celá architektúra siete (obr. 3.3) pozostáva z dvoch logických blokov, ktoré sa nazývajú blok kodérov (angl. *encoder block*) a blok dekodérov (angl. *decoder block*). Jednotlivé zložky týchto blokov a mechanizmus *attention* sú bližšie popísané v nadchádzajúcich častiach.

3.1 Blok kodérov a dekodérov

Blok kodérov Blok kodérov je reprezentovaný v podobe N za sebou prepojených identických kodérov. Každý takýto kodér je zložený z dvoch subkomponent, konkrétne *multi-head attention* vrstvy (3.2) a *feed-forward* vrstvy (3.1). Za oboma vrstvami sa nachádza normalizačná vrstva používajúca tzv. *Layer Normalization* [1] a pri oboch vrstvách sú použité reziduálne prepojenia [16]. Obe tieto techniky slúžia na zefektívnenie tréningu neurónovej siete.

Blok dekodérov Blok dekodérov má podobnú štruktúru ako blok kodérov, t.j. je reprezentovaný N za sebou prepojenými identickými dekodérmi, ktoré sú podobne prepojené ako v prípade bloku kodérov. Rozdiely v štruktúre dekodéra sú, že obsahuje naviac jednu *multi-head attention* vrstvu a pri prvej *multi-head attention* vrstve sa používa maskovanie (angl. *Masked Multi-Head Attention*). Výstup posledného kodéra sa používa ako vstupná zložka pre hodnoty K a V (vzťah 3.3) v poradí u druhej *multi-head attention* vrstvy. Toto umožňuje každému dekodéru, aby sa mohol zamerať na ktorékoľvek pozície vo vstupnej

[1] Pri generovaní symbolu x_t sekvencie $y = x_{t-n}, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_{t+n}$ sa spolieha iba na doposiaľ známe vygenerované symboly $x_{t-n}, \ldots, x_{t-1}$.



Obr. 3.1: Všeobecný výpočet *attention* vrstvy. Výstup *attention* vrtsvy (*Output*) sa vypočíta ako vážený súčet jednotlivých hodnôt V_i (*Values*) a odpovedajúcich váh. Odpovedajúce váhy sú vypočítané ako $softmax(\alpha(Query, Keys_i))$, kde α je použitá skórovacia funkcia. Prevzaté z [49].

sekvencií ako tomu býva u *sequence-to-sequence* architektúr. Maskovanie na druhú stranu slúži pre zabezpečenie auto-regresívnosti modelu. Toto je zabezpečené nastavením všetkých častí vstupnej sekvencie na pozíciách $t+1$ až $t+n$ v čase t na hodnotu $-\infty$.

Feed-forward vrstva Feed-forward vrstva (FFN) predstavuje plne prepojenú vrstvu, ktorá uskutočňuje dve samostatné lineárne transformácie a obsahuje aktivačnú funkciu ReLU (*Rectified linear unit*) [32]:

$$FFN(x) = ReLU\left(xW_1 + b_1\right)W_2 + b_2\,, \tag{3.1}$$

kde W_1, W_2, b_1, b_2 sú váhy a x je vstupný vektor.

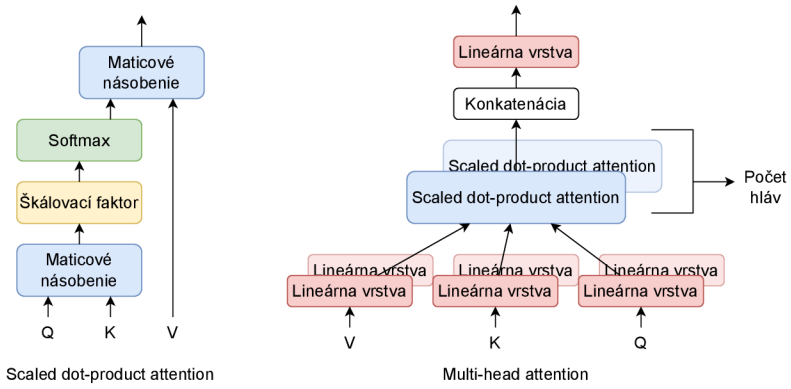
3.2 Mechanizmus Attention

Scaled Dot-Product Attention Ako mechanizmus *attention* (3.2) je v transformeri použitý tzv. *Scaled Dot-Product Attention*. Jedná sa o *attention* mechanizmus, ktorý ako skórovaciu funkciu, resp. zarovnanie, využíva skalárny súčin (angl. *dot product*) a škálovací faktor $\sqrt{d_k}$. Výpočet je daný podľa vzťahu:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V\,, \tag{3.2}$$

kde Q, K, V sú dotaz (angl. *Question*), kľúč (angl. *Key*) a hodnota (angl. *Value*). *Attention* vo svojej podstate predstavuje funkciu, ktorá na základe vstupného dotazu Q a množiny dvojíc (K, V) vytvorí výstup, ktorý sa vypočíta ako vážený súčet hodnôt V, u ktorých je priradená váha definovaná použitou skórovacou funkciou (angl. *score function*) a operátorom $softmax$. Výpočet *attention* vrstvy je popísaný na obrázku 3.1 [47].

Multi-Head attention V praxi je výhodnejšie klásť dôraz na informácie z rozličných reprezentácií a polôh v sekvencií. Z tohoto dôvodu sa *Vaswani a spol.* rozhodli vytvoriť nový mechanizmus *attention*, nazývaný *Multi-Head Attention* (3.3) [47]. V tomto mechanizme



Obr. 3.2: Porovnanie *Scaled dot-product attention* vrstvy (3.2) a *Multi-head attention* vrstvy (3.3). Multi-head attention vrstva pozostáva z niekoľkých scaled dot-product attention vrstiev, ktorých výstup bol konkatenovaný a potom transformovaný lineárnou vrstvou. Každá scaled dot-product attention vrstva pritom pracuje so svojimi vlastnými hodnotami Q, K a V, ktoré sú taktiež transformované vlastnými lineárnymi vrstvami W_i^Q, W_i^K, W_i^V (3.4) [47].

sa namiesto jednoduchej *Scaled Dot-Product Attention* vrstvy spočiatku aplikuje lineárna projekcia na dotazy Q, kľúče K a hodnoty V celkovo h-krát, kde h je počet použitých hláv. Následne sa na všetkých lineárne premietnutých hodnotách vypočíta samostatne hodnota *Scaled Dot-Product Attention* (3.4), ktorá sa konkatenuje do jedného výsledného vektora, ktorý sa ešte lineárne premietne (3.3). Celé to je to popísané ako:

$$MultiHead(Q, K, V) = Concat\left(head_1, \cdots, head_h\right) W^O, \tag{3.3}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V), \tag{3.4}$$

kde W_i^Q, W_i^K, W_i^V a W^O sú váhy lineárnych vrstiev. Porovnanie výpočtov *Scaled Dot-Product Attention* a *Multi-Head Attention* je na obrázku 3.2.

Self-Attention *Self-Attention* mechanizmus predstavuje variantu *Multi-Head Attention* vrstvy, pri ktorej sú všetky hodnoty dotazov Q, kľúčov K a hodnôt V (3.3) z rovnakej vrstvy, t.j. z výstupu predchádzajúcej vrstvy. *Self-Attention* je možné vidieť v bloku kodérov a rovnako aj v bloku dekodérov, kde tvorí prvú vrstvu a predstavuje tzv. *Masked Multi-Head-Attention* vrstvu (obr. 3.3) [47].

3.3 Pozičné kódovanie a embedding

Embedding *Embedding* E predstavuje nízko-dimenzionálnu reprezentáciu priestoru diskrétnych premenných, ktorá sa získava učením siete. Výhodou tejto reprezentácie je, že symboly sekvencie, ktoré spolu v kontexte viac súvisia majú medzi sebou nižšiu vzdialenosť. V transformeri sa *embedding* vrstva používa pri kódovaní vstupných a výstupných symbolov, pričom sa pri oboch *embedding* vrstvách používa rovnaká zdieľaná matica váh [47].

Pozičné kódovanie Transformer neobsahuje žiadne rekurentné vrstvy, a tak model nemá žiadne informácia o poradí jednotlivých symbolov v sekvencii. Pozičné kódovanie (angl. *Positional Encoding*) (3.5) rieši tento problém a informáciu o relatívnych a absolútnych polohách sa snaží pridať za pomoci transformácie vstupného, resp. aj výstupného vektora. V pôvodnom článku bolo použité fixné pozičné kódovanie za pomoci sinusoíd, ktoré by mali umožniť lepšie vnímať relatívne vzdialenosti v sekvencii:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}})\,, \end{aligned} \tag{3.5}$$

kde *pos* predstavuje polohu v sekvencií, i dimenziu vektora a d_{model} dimenziu vstupného, resp. výstupného *embeddingu*. *Embedding* E je následne modifikovaný ako $E = E + PE$ (obr. 3.3) [47].

Riešenia založené na sieťach typu transformer V roku 2017 použili pri strojovom preklade *Ashish Vaswani a spol.* [47] sieť s názvom *Transformer*. Táto sieť oproti ostatným *sequence-to-sequence* architektúram nepoužíva žiadne rekurentné vrstvy a je kompletne založená na mechanizme *attention*. Konrétne zaviedli tzv. *multi-head attention* mechanizmus, ktorý umožňuje klásť dôraz na sekvencie z viacerých reprezentácií. Ich sieť umožňuje lepšie možnosti paralelizácie výpočtu a jej varianty dosahujú *state-of-the-art* výsledky pri strojovom preklade.

Kang a spol. [18] v roku 2020 rozšírili pôvodnú architektúru siete transformer a upravili ju pre potreby rozpoznávania ručne písaneho textu. Ich sieť obsahuje konvolučnú neurónovú sieť, ktorá extrahuje vizuálne príznaky z obrazu a tie následne posúva na vstup transformeru. V ich práci sa pokúšali predtrénovať sieť na syntetických dátach, a následne poukázali na fakt, že ich sieť dosahuje lepšie výsledky ako iné *sequence-tos-sequence* architektúry, ak má dostatok trénovacích dát. Ďalej poukázali na to, že sa transformer dokáže vďaka *attention* mechanizmu naučiť jazykovo špecifické kontextové informácie a dosiahli *state-of-the-art* výsledky na dátovej sade IAM [30].

V rovnakom období použili *Ning Lu a spol.* [28] pri rozpoznávaní textu v scéne sieť s názvom *MASTER*, ktorá obdobne používala modifikovanú transformer architektúru. Poukázali na to, že ich metóda umožnila sieti vytvárať robustnejšiu reprezentáciu pri priestorovo deformovaných znakoch. V ich práci modifikovali kodér v sieti transformer. Doplnili ho o konvolučnú sieť a použili tzv. *Multi-aspect GCAttention*, čo predstavuje použitie viacerých *attention* mechanizmov v jednom bloku kodéra. Ďalej použili efektívnejšie cachovanie pamäte pri dekódovaní sekvencie a urýchlili inferenciu siete.

Christoph Wick a spol. [48] v roku 2021 rozšírili prácu od autorov *Kang a spol.* [18] a použili tzv. obojsmerné dekódovanie (angl. *Bidirectional Decoding*), počas ktorého je dekódovanie vykonané v poprednom a rovnako aj spätnom smere. Následne používajú komponentu *voter*, ktorá kombinuje obe výstupy dekódovaní a vytvára jednu výslednú sekvenciu. Rovnako ako v pôvodnej práci poukazujú na fakt, že pri rozsiahlej dátovej sade sa dokáže ich sieť výrazne zlepšiť a dosiahnuť lepšie výsledky ako CNN/LSTM siete. Transformer sa podľa nich dokáže naučiť silný jazykový model, čo presne vysvetľuje, prečo je potrebné väčšie množstvo dát pri trénovaní siete. Ďalej poukázali na potencionálny problém tejto architektúry, ktorým je opakovanie sa rovnakých symbolov v dekódovanej sekvencií.

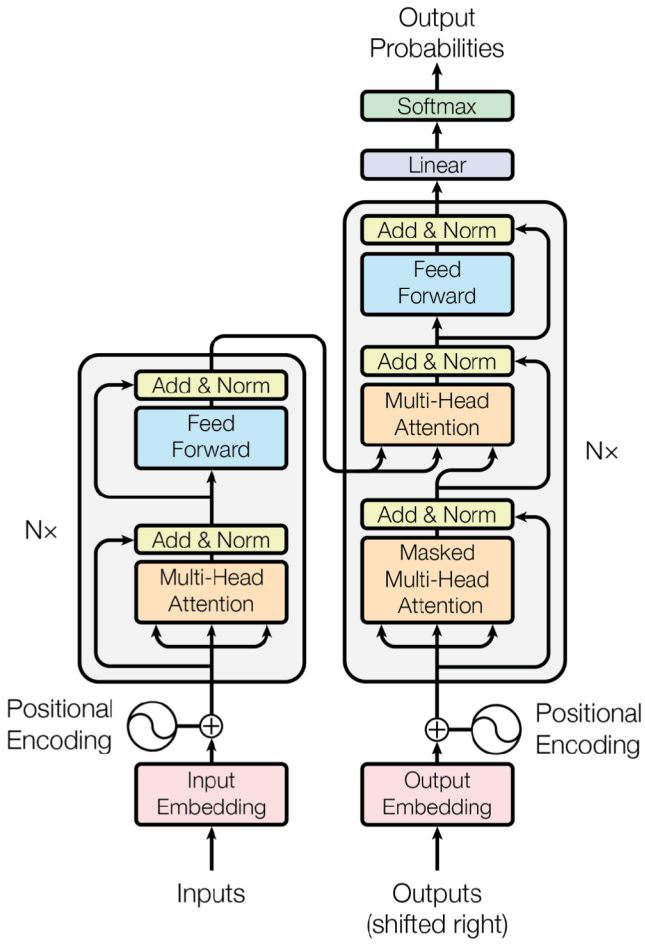
3.4 Modifikácia siete transformer pre rozpoznávanie textu

Pôvodný Transformer [47] bol použitý pri strojovom preklade. Pre rozpoznávanie textu upravili *Kang et. al* [18] túto architektúru pridaním konvolučného kodéra pre extrakciu príznakovej mapy a adaptovali model, aby dokázal rozpoznávať vstupný obraz po jednotlivých znakoch obsiahnutých v ich definovanej rozpoznávanej abecede. Ukážka ich modifikovanej architektúry je na obrázku 3.4. Následne *Christoph Wick a spol.* [48] ich prácu ešte ďalej rozšírili a zaviedli obojsmerné dekódovanie (angl. *Bidirectional Decoding).*

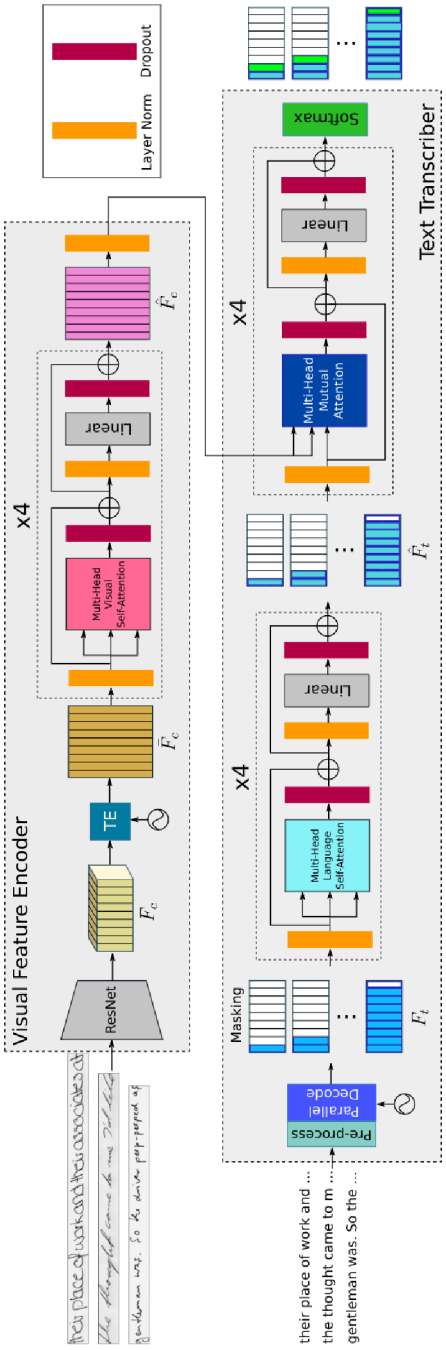
Kodér vizuálnych čŕt Kodér vizuálnych čŕt je reprezentovaný v podobe konvolučnej siete, ktorá má za úlohu transformovať vstupný trojrozmerný obraz riadka s textom a vytvoriť jeho výstupnú reprezentáciu v podobe príznakovej mapy F_c. Ako konvolučnú sieť použili architektúru *Resnet* [16]. Tento výstup je následne transformovaný za pomoci pozičného kódovania, ktoré je použité v pôvodnom článku [47], na vektor $\overline{F}_c$. Tento vektor je následne transformovaný obdobným blokom kodérov a dekodérov ako v pôvodnej práci [47]. V druhej práci [48] používajú na výstupe CNN ešte plne prepojenú vrstvu. Pri kroku dekódovania je výstupom dekódovaný znak, ktorý je opakovane použitý pri nasledovnom kroku dekódovania spolu s predošlými dekódovanými znakmi [18].

Kódovanie textu Pri rozpoznávaní textu je potrebné okrem definovania rozpoznávanej abecedy $\mathcal{A}$ definovať aj špeciálne symboly bez textového obsahu. Týmito symbolmi sú $\langle S \rangle$, ktorý určuje začiatok sekvencie, $\langle E \rangle$, ktorý určuje koniec sekvencie a $\langle P \rangle$, ktorý sa používa pre tzv. *padding. Padding* sa používa pre zarovnanie predikovanej sekvencie na vopred definovanú maximálnu dĺžku N znakov [18].

Inferencia siete Počas inferencii siete nie sú dostupné prepisy dát y, ktoré by odpovedali vstupom x. Prechod sieťou je tým pádom inicializovaný zaslaním symbolu $\langle S \rangle$ na vstup prvej vrstvy dekodéra. Pri predikcií znakov sa používa tzv. *greedy decoding* algoritmus [18], ktorý použije najpravdepodobnejší predikovaný znak ako výstupný znak v kroku t. Ďalšou možnosťou je použiť *beam search* algoritmus [48], ktorý je presnejší, ale zároveň výpočtovo náročnejší. Celý proces dekódovania následne prebieha v cykle, dokým sa neobjavý ukončovací symbol $\langle E \rangle$, prípadne sa nedosiahne maximálna dĺžka N znakov predikovanej sekvencie [18].



Obr. 3.3: Architektúra siete *Transformer*. Model je auto-regresívny a pri generovaní nasledujúceho symbolu spotrebuváva doposiaľ vygenerované symboly. Pri prechode sieťou je ku vstupu (*Inputs*) a obdobne ku cieľovému výstupu/reálnemu výstupu (*Outputs*) pridaný embedding (*Input/Output Embedding*) a pozičné kódovanie (*Positional Encoding*). Modifikovaný vstup následne prechádza N identickými kodérmi v bloku kodérov a v časti bloku dekodérov obdobne N identickými dekodérmi. Výstup posledného kodéra sa používa ako vstupná zložka pre hodnoty K a V (vzťah 3.3) v poradí u druhej *multi-head attention* vrstvy každého dekodéra v bloku dekodérov. *Multi-head attention* vrstvy (*Multi-Head Attention*) a *feed-forward* vrstvy (*Feed Forward*) sú bližšie popísané v častiach 3.2 a 3.1. U všetkých subkomponent v kodéroch a dekodéroch sa používajú reziduálne prepojenia [16] spolu s normalizáciou (*Add & Norm*) [1]. Výstup posledného dekodéra prechádza cez lineárnu vrstvu (*Linear*) a funkciu softmax (*Softmax*). Na základe výstupných pravdepodobností (*Output Probabilities*) sa určí konkrétna sekvencia symbolov (*Outputs*), ktorá sa opätovne používa ako vstup prvého dekodéra v bloku dekodérov. Tento vstup prechádza oproti pôvodnej *multi-head attention* vrstvy tzv. maskovacou *multi-head attention* vrstvou (*Masked Multi-Head Attention*), ktorá maskuje budúce symboly počas trénovania a zabezpečuje auto-regresívnosť modelu [49, 47]. Obrázok je prevzatý z [47].



Obr. 3.4: Architektúra modifikovanej siete Transformer pre rozpoznávanie textu. Vstupný obraz je transformovaný konvolučnou sieťou *Resnet* [16] na príznakovú mapu F_c. Na tento výstup je následne aplikované pozičné kódovanie (TE). Potom je tento nový vektor transformovaný obdobnými blokmi kodérov a dekodérov ako to je popísané na obrázku 3.3. Obrázok je prevzatý z [18].

Kapitola 4

Dátové sady pre rozpoznávanie ručne písaného textu

Táto kapitola obsahuje prehľad o dostupných dátových sadách, ktoré je možné použiť pri rozpoznávaní ručne písaného textu. V rámci kapitoly je v tabuľke 4.1 dostupné aj porovnanie medzi jednotlivými sadami a konkrétne vzory z dátových sád sú na obrázku 4.1.

Dátové sady pre potreby HTR obsahujú zvyčajne mapovací súbor, v ktorom sú identifikátory obrázkov a následne ich prepisy. Obrázky bývajú následne uložené v bežných formátoch akými sú napr. *png* a *jpg*. Častokrát nie sú v dátovej sade dostupné obrázky jednotlivých riadkov textu, ale iba obrázky celých stránok spolu so súbormi *PAGE xml* [34], v ktorých sú dostupné informácie o jednotlivých detekovaných regiónoch na stránke. Riadky musia byť následne z takýchto dátových sád extrahované.

V projekte PERO sa pracuje práve s PAGE formátom súborov a používajú sa databázy LMDB[1] (*Lightning Memory-Mapped Database*), ktoré uchovávajú obrázkové dáta riadkov, pričom ako identifikátory sú použité identifikátory dokumentov, regiónov a riadkov. Práca s takouto databázou urýchľuje načítanie dát a prínaša tým aj urýchlenie trénovania siete.

4.1 Prehľad existujúcich dátových sád

IAM Dátová sada IAM [30] pozostáva z ručne písaných textov v anglickom jazyku od 657 rôznych autorov. Dátová sada je založená z tzv.*Lancaster-Oslo/Bergen* (LOB) korpusu[2], ktorý obsahuje cez 1 mil. slov. Dataset IAM obsahuje 13 353 riadkov textu, ktoré sa skladajú celkovo z 115 320 slov. Dátovú sadu je možné využiť pre rozpoznávanie celých viet, ale aj slov. Rovnako je v rámci dátovej sady poskytnutá aj informácia o identifikácií autora daného prepisu. Obrázky v dátovej sade boli skenované v 256 úrovniach šedej farby. Dátová sada bola prvýkrát použitá v roku 1999 na súťaži ICDAR[3] a je stále populárna pri rozpoznávaní textu [18, 48].

READ Datová sada READ [38] bola vytvorená v rámci projektu *Recognition and Enrichment of Archival Documents*[4] (READ) v roku 2016. Datová sada pozostáva z nemeckých

[1] `https://lmdb.readthedocs.io/en/release/`

[2] `http://korpus.uib.no/icame/manuals/LOB/INDEX.HTM`

[3] `https://www.icdar.org/`

[4] `https://cordis.europa.eu/project/id/674943`

Názov sady	Jazyk	Počet autorov	Rozsah
IAM [30]	anglický	657	13 353 riadkov
READ [38]	nemecký	neznámy	10 550 riadkov
Bentham [39]	anglický	viac ako 1	21 752 riadkov
Databáza CVL [21]	anglický a nemecký	311	101 069 slov
ScribbleLens [9]	holandský	85	28 255 riadkov

Tabuľka 4.1: Prehľad dátových sád používaných u HTR.

zápiskov zo zasadnutia radníc z rokov 1470 až 1805, ktoré boli súčasťou kolekcie *Ratsprotokolle*. Celá kolekcia obsahovala približne 30 000 stránok textu, zatiaľčo READ obsahuje vyňatých 450 strán, ktoré obsahujú spolu 10 550 riadkov textu. Dokumenty spôsobujú ťažkosti pri analýze rozloženia stránky, pretože obsahujú poznámky na okraji, vyblednuté písmo a pod. Textové prepisy dátovej sady sú dostupné vo formáte PAGE [34]. Kedže je počet autorov dokumentov neznámy, tak sa táto sada používa predovšetkým pri rozpoznávaní riadkov textu.

Bentham Dátová sada Bentham [39] pozostáva z rukopisov filozofa *Jeremy Benthama* a jeho sekretariátu napísaných v 18. až 19. storočí. Dátová sada je napísaná v anglickom jazyku a obsahuje celkovo 796 strán, ktoré tvoria celkovo 21 752 riadkov textu. Z hľadiska HTR ide o texty, ktoré je náročné rozpoznávať kvôli rozličným škrabancom, poznámkam a horšej kvalite textu. Datová sada bola vytvorená v rámci projektu tranScriptorium[5] a anotácie sú dostupné vo formáte PAGE [34] súborov.

Databáza CVL Dátová sada CVL [21] pozostáva z anglických a nemeckých prepisov 7 vybraných literárnych diel, ktoré boli prepísané 311 rôznymi autormi. Celkový počet prepisovaných slov v rámci týchto diel je 451. Dátovú sadu je možné okrem rozpoznávania textu využiť aj na identifikáciu pisateľa. Nevýhodou je, že textové anotácie sú dostupné pre prepisované paragrafy v celku alebo potom pre jednotlivé slová, zatiaľčo riadkové anotácie sú postrádané. Anotácie sú dostupné v osobitných *xml* súboroch.

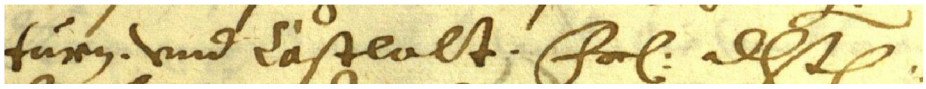
ScribbleLens Datová sada *ScribbleLens* [9] obsahuje 1 000 strán holandských textov z obdobia 16. až 17. storočia. Dokumenty predstavujú lodné denníky spísané námorníkmi Východoindickej spoločnosti. Z týchto dokumentov je však textovo anotovaných iba 200 strán. Dataset bol spísaný viac ako 80 autormi a je použiteľný aj pri rozpoznávaní pisateľa.

4.2 Dátová sada PERO HWR

Pre trénovanie neurónových sietí je v prípade *supervised* učenia častokrát nevyhnutné obstaranie rozsiahlych dátových sád. Podľa autorov *Christoph Wick a spol.* [48] je to nevyhnutná podmienka, aby dokázala sieť typu transformer prekonať prístupy založené na rekurentných neurónových sieťach pri rozpoznávaní ručne písaného textu. Dátové sady v sekcií 4.1

[5] `https://cordis.europa.eu/project/id/600707`

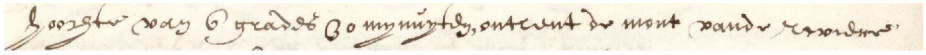
(a) IAM. Prevzaté z [30].



(b) READ. Prevzaté z [38].

(c) Bentham. Prevzaté z [39].

(d) Databáza CVL. Prevzaté z [21].



(e) ScribbleLens. Prevzaté z [9].

Obr. 4.1: Ukážky z dátových sád používaných u HTR.

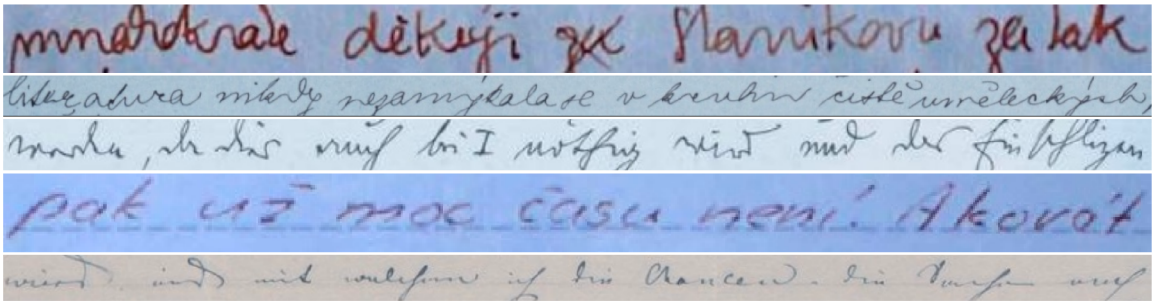
obsahujú iba niekoľko desiatok tisícok riadkov, a preto je v práci použitá rozsiahlejšia dátová sada vytvorená v rámci projektu PERO, ktorá je v tejto práci označovaná ako PERO HWR. Pri zdrojových súboroch je táto dátová sada označená ako *HWR.2021-09-01*.

PERO HWR predstavuje multilingválnu dátovú sadu. Obsahuje ručne písané texty v českom, nemeckom a anglickom jazyku. Zdroje pre vytvorenie tejto dátovej sady boli dátová sada *Bentham* [39], dátová sada Brno HWR[6], dátová sada READ [38], české dopisy a české kroniky, vojenské denníky a kroniky a ďalšie dáta získané z projektu PERO. Ukážka niektorých vzoriek z dátovej sady je na obrázku 4.2.

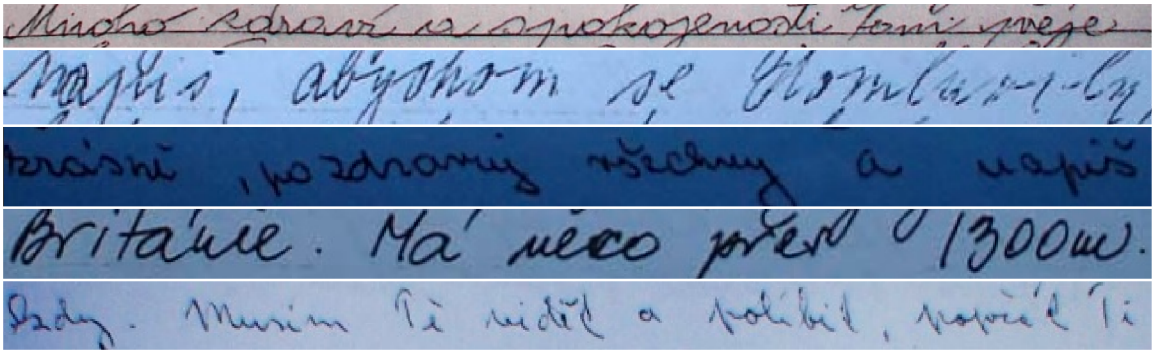
Verzia dátovej sady, ktorá sa používa v tejto práci obsahuje celkovo 538 560 unikátnych riadkov textu. Z tohoto počtu je pre tréningovú sadu vyčlenených 533 131 riadkov a 5 429 riadkov je ponechaných ako testovacia sada. Kvôli zvýšeniu počtu riadkov v českom jazyku boli niektoré riadky v rámci tréningovej množiny duplikované do celkového množstva 1 364 922 riadkov. Výška obrázka v dátovej sade je 40 pixelov, zatiaľčo dĺžka najdlhšieho obrázka je 1 512 pixelov.

Textové korpusy Pri rozpoznávaní ručne písaného textu sa bežne používa dátová sada s výrezom riadka a jeho odpovedajúcou textovou časťou. V tejto práci bola okrem takejto tradičnej dátovej sady použitá aj sada s textovými korpusmi. Použitie textových korpusov pri trénovaní siete je popísané v časti 5.4. V práci sa používa textový korpus, ktorý poskytol *Ing. Karel Beneš*. Ide o textový korpus v českom jazyku, ktorý obsahuje 109 355 492 textových riadkov, pričom najdlhší riadok v tomto korpuse dosahuje 46 znakov. Pred použitím korpusu z neho boli odstránené neznáme znaky, ktoré sa nevyskytovali v znakovej sade siete.

[6] `https://pero.fit.vutbr.cz/handwritten_dataset`



(a) Tréningová podmnožina.



(b) Testovacia podmnožina.

Obr. 4.2: Ukážky z dátovej sady PERO HWR (alias *HWR.2021-09-01*).

Kapitola 5

Návrh a implementácia neurónovej siete transformer

Táto kapitola obsahuje základné informácie o návrhu siete a niektoré implementačné detaily. V prvej časti je popísané prostredie pre trénovanie neurónových sietí a popis siete. V ďalšej časti sú popísané informácie týkajúce sa tokenizácie textu, použitie textových korpusov pri trénovaní a trénovanie s použitím dodatočnej textovej informácie v podobe textových prefixov.

5.1 Prostredie pre trénovanie siete

Ako programovací jazyk sa v práci používa *Python* a jeho knižnica *Pytorch*[1]. V práci sa pracuje s privátnym repozitárom[2] projektu PERO, kde sa v samostatnej vetve projektu modifikuje existujúci kód pre prácu s architektúrou transformer. Trénovanie následne prebieha v prostredí MetaCentra[3] a prevažne na GPU *Tesla T4 16GB*. Natrénovanie modelu trvá v týchto podmienkach približne 24 – 32 hodín.

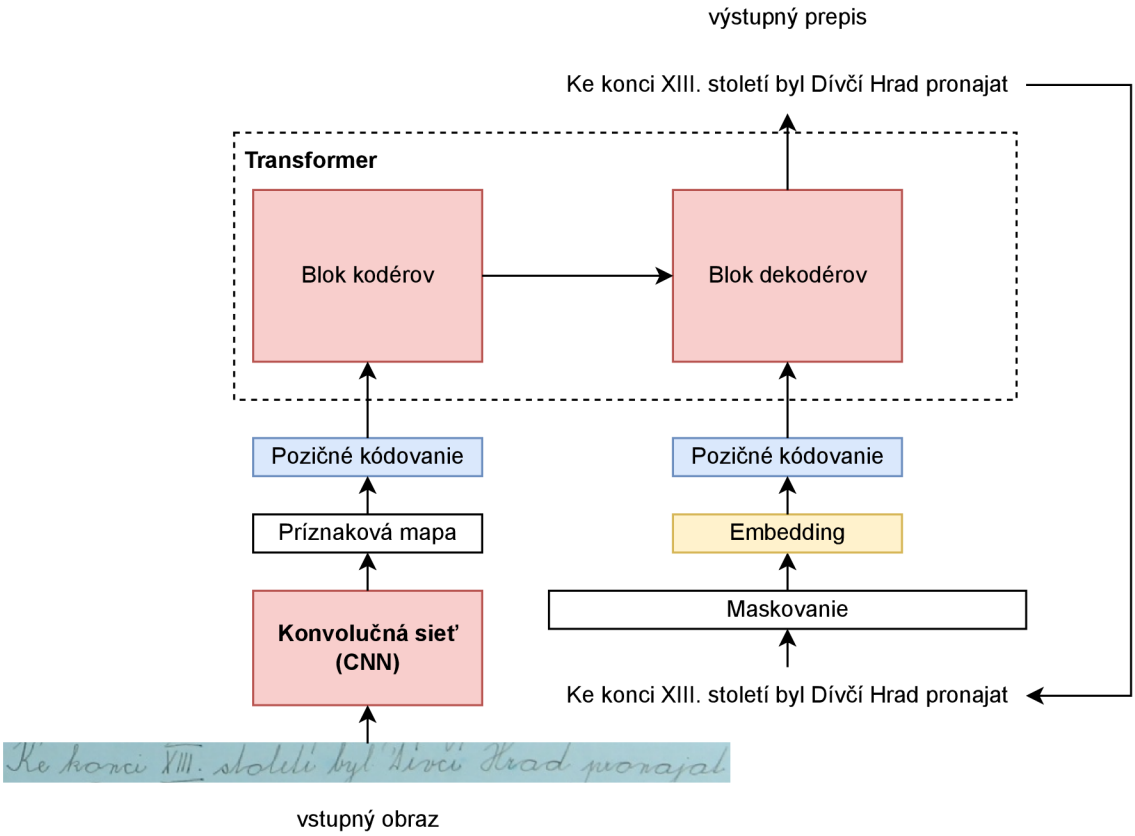
5.2 Popis siete pre rozpoznávanie textu

V rámci práce vychádzam z existujúceho zdrojového kódu projektu PERO, ktorý som upravil do spustiteľnej podoby a pripravil skripty pre trénovanie. Kód som rozšíril o modifikácie akými sú použitie odlišnej tokenizácie, možnosť trénovania na textových dátach, použitie rôznych konvolučných sietí, použitie dátovej paralelizácie, trénovanie s použitím prefixov a ďalšie.

Transformer pozostáva z konvolučnej neurónovej siete, ktorá spracuje vstupný trojrozmerný obraz obsahujúci riadok textu. Príznaková mapa vytvorená konvolučnou sieťou je následne spracovaná architektúrou siete transformer, ktorý odpovedá architektúre z práce autorov *Vaswani et. al* [47]. Model použitej siete so základnými stavebnými prvkami je zobrazený na obrázku 5.1.

Vstupom siete je obraz pozostávajúci z troch kanálov a výškou 40 pixelov. Maximálna šírka obrázka je 1512 pixelov. Pre augumentovanie dát je použitá augumentácia vytvorená v rámci projektu PERO, ktorá je označená ako *UNIVERSAL_HWR*. Táto augumentácia

[1] `https://pytorch.org/`
[2] `https://github.com/DCGM/pero/tree/transformer`
[3] `https://metavo.metacentrum.cz/`



Obr. 5.1: Model siete transformer, ktorý je použitý v tejto práci pre rozpoznávanie ručne písaného textu. Vstupný obraz s tromi kanálmi je spracovaný konvolučnou sieťou a následne sieťou transformer, ktorá generuje jeho prepis. Stavebné prvky bloku kodérov a bloku dekodérov, ako aj ďalšie detaily sú vysvetlené v kapitole 3.

mení jas, kontrast a sýtosť obrazu a ďalej pridáva mazanie, gaussovský šum a ďalšie. Ukážka augumentovaného obrazu je na obrázku 5.2.

Trénovanie siete prebieha vo forme učenia s učiteľom, kde je ku každému obrázku priradený jeho odpovedajúci prepis. Kódovanie tohoto prepisu je následne popísané v časti 5.3 a zobrazené na obrázku 5.3. Okrem trénovania použitím takýchto dvojíc je v tejto práci experimentované aj s trénovaním s dodatočnými textovými korpusmi a ďalej použitím dodatočnej textovej informácie pri dekódovaní v podobe textových prefixov. Tieto metódy sú bližšie popísané v častiach 5.4 a 5.5.

Pri trénovaní siete sa používa optimalizátor *Adam* [20] a *learning rate* o veľkosti 0.1×10^{-3}. Na začiatku trénovania sa používa postupné zvyšovanie *learning rate* do vopred stanovenej hodnoty, tzv. *warm-up*. Toto zvyšovanie je určené vzťahom $(iter/10000)^3 * 10^{-4}$, kde *iter* predstavuje aktuálnu iteráciu počas trénovania. Pri väčšine experimentov sa používa hodnota trénovacej dávky batch o veľkosti 16. V transformeri sa používa regularizácia *dropout* [43] o hodnote 30%.

Transformer Pri implementácií siete transformer je použitá trieda z knižnice *Pytorch*[4], v ktorej je implementovaná základná verzia tejto siete. V tejto práci sa doplnila modifikácia,

[4] `https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html`

Obr. 5.2: Ukážka dátovej augumentácie použitej pri experimentoch. V prvom riadku je pôvodný originálny obrázok. V druhom riadku je jeho augumentovaná verzia, u ktorej je vidieť použitie gaussovského šumu a rozmazania obrazu.

kedy je možné u tejto siete použiť odlišné počty blokov v rámci enkodér a dekodér blokov. Obdobne je možné použiť v dekodéry odlišnú dimenzionalitu ako v kodéry. Na výstup konvolučnej siete je pred vstupom do kodéra aplikované pozičné kódovanie 3.3. Pri vstupe dekodéra je na predikcie sekvencie aplikovaná *embedding* vrstva 3.3 a následne obdobne ako v prípade enkodéra pozičné kódovanie.

Teacher forcing Pri trénovaní siete sa používa technika nazývaná *teacher forcing*, ktorá umožňuje trénovať transformer časovo efektívnejšie. Technika spočíva v tom, že pri generovaní výslednej predikcie dekodérom je v časovom kroku t známa hodnota tokenu z kroku $t-1$, ktorú už transformer generoval. Namiesto použitia tejto hodnoty z kroku $t-1$ je však použitá skutočná hodnota tokenu, ktorá sa získa z prepisu obrázka (tzv. *ground-truth*). Pri trénovaní je možné túto techniku vynechať, čo spôsobí nárast času trénovania siete. Pri inferencií siete je následne táto technika vynechaná a generovanie nasledujúceho znaku je závislé od znakov vygenerovaných výslednou sieťou. Pri trénovaní aj inferencií sa dekodér inicializuje štartovacím tokenom.

Early stopping Pri trénovaní siete sa používa algoritmus tzv. *early stopping*, ktorý sa vyhodnocuje na testovacej chybe a umožňuje predčasné zastavenie trénovania, ak sa po určitej dobe špecifikovanej parametrom sieť ďalej nezlepšuje. Trieda implementujúca tento algoritmus zabezpečuje aj ukladanie modelu, ktorý dosahuje najlepšie výsledky na zvolenej testovacej sade.

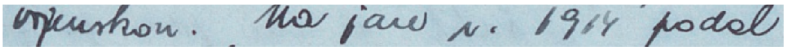
Label smoothing Pre možnosť dodatočnej regularizácie trénovania siete bola pridaná trieda pre algoritmus *label smoothing* [47, 46]. Tento algoritmus zamedzuje sieti vytvárať príliš sebavedomé rozhodnutia o výslednej predikcií. Výsledný chyba *loss* je pri použití tohoto algoritmu počítaná podľa vzťahu:

$$loss = \sum_{i=1}^{n} [(1-\epsilon)H_i(p, q_\theta) + \epsilon H_i(u, q_\theta)] \,,$$

kde n je veľkosť dávky batch, H_i značí *cross-entrophy*, q_θ je distribúcia predikovaných prepisov, p je distribúcia *ground-truth* a u je uniformná distribúcia $1/n$. Hodnota ϵ predstavuje volený hyperparameter, ktorý je v experimentoch nastavený na hodnotu 10%. Pri hodnote 0 je *label smoothing* ignorovaný [46].

5.3 Tokenizácia textu

Tokenizácia textu je proces, ktorý popisuje akým spôsobom je text rozdelený na najmenšie symboly, tzv. tokeny. Tokeny s ktorými pracuje neurónová sieť majú podobu číselných iden-



vojenskou. Na jaře r. 1914 podal

511, 49, 42, 37, 32, 41, 46, 38, 42, 48, 60, 1, 15, 28, 1, 37, 28, 153, 32, 1, 45, 60, 1, 121, 129, 121, 124, 1, 43, 42, 31, 28, 39

49, 42, 37, 32, 41, 46, 38, 42, 48, 60, 1, 15, 28, 1, 37, 28, 153, 32, 1, 45, 60, 1, 121, 129, 121, 124, 1, 43, 42, 31, 28, 39, 511

Obr. 5.3: Ukážka kódovania textu za pomoci tokenizácie znakov. Prvý riadok je vstup siete. Druhý riadok je prepis riadka. Tretí riadok predstavuje vstup bloku dekodérov, na ktorý je ešte použitie maskovanie. Štvrtý riadok je očakávaný výstup siete. Sieť sa musí správne naučiť generovať aj ukončovací token, inak pri inferencií siete dôjde ku cykleniu a generovaniu ďalších znakov. Štartovací aj ukončovací token je reprezentovaný tým istým tokenom.

tifikátorov. Tieto identifikátory môžu byť následne pomocou známeho mapovania prevedené na čitateľný text.

Bežne sa pri rozpoznávaní textu používa tokenizácia znakov. Počet tokenov je v tomto prípade zvyčajne rovný počtu jedinečných znakov obsiahnutých v dátovej sade. Slovník tokenov býva dodatočne rožšírený o tokeny pre identifikáciu začiatku a konca sekvencie a tzv. *padding* token, ktorý slúži na doplnenie sekvencie do určitej vopred stanovenej dĺžky.

V tejto práci používam tokenizáciu znakov. Počet používaných tokenov je 511. Tieto tokeny reprezentujú znaky vyskytujúce sa v bežnom, ale aj historickom texte, ktoré sú používané v projekte PERO. Vyšší počet znakov je použitý z dôvodu, aby sa jednoducho mohla sieť trénovať na rozličných dátových sadách, ktoré obsahujú odlišné tokeny a nemusela sa pritom meniť posledná lineárna vrstva v sieti. Dátová sada s ktorou pracujem obsahuje 114 jedinečných znakov, takže zvyšné znaky nevyskytujúce sa v texte sa sieť nenaučí správne generovať.

Pri kódovaní textu je na začiatok každej sekvencie pridaný štartovací token. Všetky sekvencie majú v dávke batch rovnakú dĺžku, čo je zaručené pridaním *padding* tokenu na koniec každej sekvencie, ktorá nedosahuje maximálnu dĺžku v danej dávke. Pri cieľovej sekvencií sa následne očakáva, že sa sieť naučí správne generovať ukončovancí token, ktorý indikuje koniec generovania znakov. V tejto práci je štartovací a ukončovací token rovnaký. Ukážka kódovania je na obrázku 5.3.

V experimentoch s tokenizáciou používam knižnicu *sentencepiece*[5]. *Sentencepiece* [25] je jazykovo nezávislý tokenizér, ktorý pracuje na úrovni nespracovaného textu z ktorého za pomoci techniky *Byte-Pair Encoding* (BPE) alebo *unigram* vytvorí výsledný slovník s tokenmi. Rovnako podporuje rôzne techniky normalizácie textu a regularizáciu učenia pri použití tejto tokenizácie. Pri vytváraní slovníka sa vopred špecifikuje jeho výsledná veľkosť vo forme hyperparametru. Vyššie číslo symbolov v slovníku môže spôsobiť, že výsledný text bude kódovaní nižším počtom znakov, a to môže následne aj urýchliť proces inferecie siete. O tomto viac vypovedá experiment 6.3.

5.4 Použitie textových korpusov pri trénovaní siete

Pri trénovaní siete bola implementovaná možnosť trénovať sieť s použitím textových korpusov. Pri takomto trénovaní sú v rámci jednej dávky batch dostupné obrázky s odpovedajúcimi prepismi a ďalej sú v tejto dávke dostupné aj samostatné texty, bez obrázkových

[5] `https://github.com/google/sentencepiece`

dát. V praxi môže bežne nastať situácia, kedy nie je v dátovej sade dostupné dostatočné množstvo dát, a v takomto prípade je možné povoliť aj práve takéto trénovanie siete.

V práci je implementovaná samostatná trieda pre načítanie textových korpusov a ich pridaní do trénovacej dávky. Pri trénovaní sa následne pracuje s dvomi argumentami, ktoré špecifikujú veľkosti jednotlivých dávok batch. Prvý batch popisuje počet dvojíc obrázok – prepis (obrázkový batch) a druhý batch popisuje počet textových riadkov z textového korpusu (textový batch). Výsledná veľkosť mini dávky je následne daná sumou týchto dvoch typov dávok.

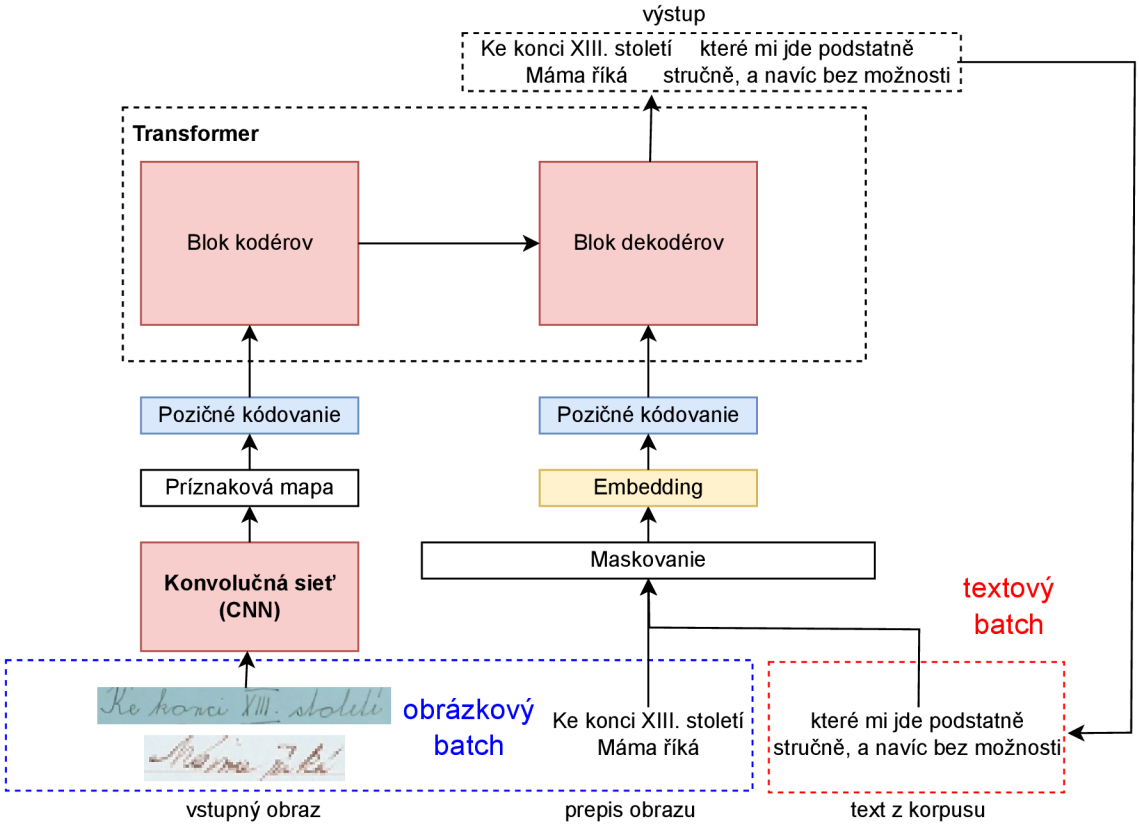
Počas dopredného prechodu sieťou je obrázkový batch transformovaný konvolučným kodérom a kodérmi siete transformer. Výstupná viacdimenzionálna matica v tomto prípade reprezentuje zakódované informáacie z obrázka. Aby bolo možné pri trénovaní použiť textový korpus, tak musí byť tento viacdimenzionálny výstup rozšírený o dodatočnú maticu, ktorá bude reprezentovať chýbajúce obrázky z textového batchu. Ku viacdimenzionálnej matici je teda konkatenovaná ďalšia matica, ktorá charakterizuje textový batch. Táto matica môže byť náhodne inicializovaná alebo nadobúdať hodnotu konštanty. Pri dekódovaní sekvencie s použitím *teacher forcing* je následne ku ground-truth obrázkového batchu pripojený textový batch prepisov tvorený z riadkov z textových korpusov. Následne bude dekodér dekódovať prepisy, ktoré pochádzajú z obrázka, a rovnako aj prepisy z textových korpusov. Počas spätného prechodu sieťou následne textový batch prispieva ku aktualizácií váh v dekodér blokoch, zatiaľčo obrázkový batch prispieva ku aktualizácií váh v celej sieti. Je to spôsobené týp, že výsledný výpočet gradientu u konštanty nadobúda hodnotu 0. Ukážka schémy siete s použitím tohoto trénovania je na obrázku 5.4.

V rámci práce sa ako dátová sada textových korpusov používa dataset popísaný v časti 4.2. Kvôli rozpoznávaniu monolinguálnych dát bol následne modifikovaný aj existujúci dataset PERO HWR 4.2, z ktorého bola odstránená väčšina textu v odlišnom jazyku.

5.5 Trénovanie siete s použitím textových prefixov

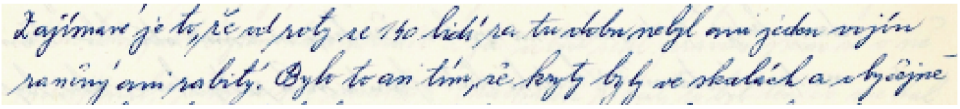
Pri trénovaní s použitím textových prefixov je vstupom dekodér blokov, okrem samostatnej informácie o predikovaných tokenoch z minulých časových krokov, aj dodatočná informácia v podobe ďalšieho textu, ktorý je v tejto práci označený ako textový prefix. Textový prefix označuje textový údaj, ktorý ma určité spojenie s dekódovanou sekvenciou. Môže ísť o naväzovanie vety, regiónu a pod. V experimentoch v časti 6.5 sa následne skúma, či môže mať takáto dodatočná textová informácia pozitívny vplyv pri dekódovaní sekvencie a zvýšiť tým jej úspešnosť. Ukážka textu s textovým prefixom je na obrázku 5.5.

V rámci implementácie sa používa modifikovaný dataset PERO HWR 4.2. Tento upravený dataset obsahuje okrem dvojíc obrázok – prepis, aj dodatočnú informáciu v podobe textového prefixu. Súbor s mapovaním obrázkov a prepisov obsahuje ďalej aj informáciu o pozícií, na ktorej začína skutočná sekvencia, ktorá sa viaže ku danému obrázku. Textový údaj, ktorý predchádza tejto pozícií predstavuje textový prefix. Pri kódovaní textovej sekvencie vstupuje do siete štartovací token, ktorý je nasledovaný textovým prefixom a za ním nasleduje skutočný textový údaj z obrázku. Pri generovaní výslednej sekvencie sa však očakáva, že bude sieť generovať iba skutočný text, ktorý sa nachádza na obrázku. Pri inferencií siete je následne možné použiť dekódovanie za pomoci dodatočného textu alebo s vynechaním takéhoto textu.

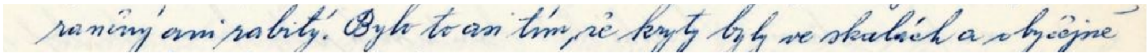


Obr. 5.4: Trénovanie siete s použitím textových korpusov. Pri trénovaní siete sa používa dvojica dávok, tzv, obrázkový batch a textový batch. Obrázkový batch pozostáva z dvojíc obrázok – prepis. Textový batch tvoria texty bez obrázkových dát. Počas dopredného prechodu sieťou je vstupom CNN obrázok z obrázkovej dávky. Výstup posledného kodér bloku, ktorý obsahuje kódované informácie z obrázka je následne rozšírený o maticu reprezentujúcu dáta z textového korpusu. Vstup dekodéra pozostáva následne z textových údajov oboch dávok. Pri procese dekódovania prebieha následne okrem dekódovania klasického obrázkového batchu aj ku dekódovaniu batchu s textovými dátami.

Modifikácia datasetu Pri tvorbe datasetu sa spracovávali xml súbory vo formáte PAGE [34], ktoré obsahovali informácie o dostupných regiónoch a náväznosti jednotlivých textových riadkov. Zo všetkých textových súborov sa extrahovali naväzujúce riadky a vytvoril sa dataset, ktorý obsahoval postupne naväzujúce riadky o počte 1 a 2 naväzujúcich riadkov. Z dátovej sady som odfiltroval riadky, ktoré boli prázdne a obdobne riadky, ktoré sa algoritmicky dali odhaliť ako nesprávne segmentované. Výsledným krokom bolo vytvorenie *lmdb* databáze aká sa používa v projekte PERO. Počas trénovania siete sa následne používa pri overovaní testovacia dátová sada v podobe datasetu bez prefixov a datasetu s prefixom, aby sa odhalilo, či je dekódovanie s použitím textových prefixov úspešnejšie.



Zajímavé je to, že od roty ze 140 lidí za tu dobu nebyl ani jeden vojín
raněný ani zabitý. Bylo to asi tím, že kryty byly ve skalách a obyčejně



Zajímavé je to, že od roty ze 140 lidí za tu dobu nebyl ani jeden vojín
raněný ani zabitý. Bylo to asi tím, že kryty byly ve skalách a obyčejně

Obr. 5.5: Trénovanie siete s použitím textových prefixov. Prvý obrázok je výrez zo stránky, z ktorej boli následne segmentované riadky, ktoré sa používajú pri trénovaní siete. Pod prvým obrázkom je jeho textový prepis. Druhý obrázok reprezentuje konkrétny výrez, ktorý je v dátovej sade. Text pod ním obsahuje následne okrem samotného prepisu riadka aj jeho prefix, ktorý pochádza z predchádzajúceho riadku rovnakého regiónu. Textový prefix je zvýraznený tučným písmom.

Kapitola 6

Experimenty

V tejto kapitole sú popísané experimenty, ktoré som uskutočnil s rozličnými architektúrami sietí typu transformer pri rozpoznávaní riadkov textu. Pre vyhodnocovanie presnosti sa v prípade rozpoznávania textu typicky používajú metriky *Character Error Rate* (CER) a *Word Error Rate* (WER), ktoré umožňujú porovnávať presnosť prepisu siete nezávisle na dĺžke rozpoznávanej sekvencie.

CER *Character Error Rate* predstavuje podiel medzi minimálnym počtom operácií, potrebných pre zámenu zdrojového reťazca za cieľový reťazec a dĺžkou cieľového reťazca:

$$CER = \frac{(i + s + d)}{n}, \tag{6.1}$$

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

WER *Word Error Rate* predstavuje podiel medzi minimálnym počtom operácií, potrebných pre zámenu zdrojových slov za cieľové slová a počtom slov v cieľovom reťazci:

$$WER = \frac{(i_w + s_w + d_w)}{n_w}, \tag{6.2}$$

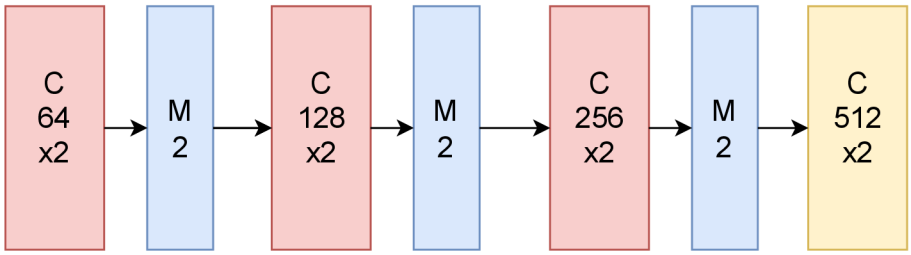
pričom i_w, s_w a d_w sú rovnaké operácie ako pri výpočte CER, ktoré sú vymedzené na slová a nie na znaky. Hodnota WER býva zvyčajne vyššia ako hodnota CER. V tejto práci je použitá iba metrika CER, ktorá vypovedá o výslednej presnosti modelu viac ako metrika WER.

6.1 Vplyv veľkosti a typu konvolučného kodéra na úspešnosť modelu

V prvom experimente som porovnával vplyv rozličných typov konvolučných kodérov na výslednú úspešnosť generovania prepisov obrazu. V rámci experimentu boli použité známe architektúry konvolučných sietí, ktorými sú siete ResNet [16] a VGG [42]. Okrem týchto sietí bol použitý aj konvolučný kodér, označený ako pero_vgg, z rekurentnej neurónovej siete, ktorá využíva objektívnu funkciu CTC a používa sa v projekte PERO [23]. Ukážka architektúry pero_vgg je na obrázku 6.1. Konvolučný kodér má podobnú štruktúru ako prvé bloky siete VGG. Tvoria ho 4 konvolučné bloky, pričom každý blok tvorí dvojica konvolúcií

Konvolučný kodér	Počet parametrov	Tréningová sada CER	Testovacia sada CER
resnet34 [16]	21 284 672	6,87 %	6,25 %
vgg16 [42]	14 723 136	5,66 %	4,91 %
vgg16*	14 723 136	4,48 %	4,76 %
vgg*	5 276 480	4,17 %	4,15 %
pero_vgg [23]	2 326 912	**3,48** %	**3,81** %

Tabuľka 6.1: Porovnanie rozličných typov a veľkostí konvolučných kodérov. Siete resnet34 a vgg16 predstavujú štandardné konvolučné siete. Siete vgg16* a vgg* sú modifikované verzie siete VGG [42], kde vgg16* neobsahuje poslednú *pooling* vrstvu a sieť vgg* obsahuje odlišný počet vrstiev a veľkosti konvolučného jadra. Sieť pero_vgg označuje konvolučný kodér používaný u architektúry rekurentnej neurónovej siete s chybovou funkciou CTC, ktorá bola natrénovaná ako súčasť projektu PERO. Počet parametrov označuje počet trénovateľných váh u použitých konvolučných sietí. Hodnota CER u tréningovej sady je počítaná s použitím techniky *teacher forcing* na náhodnej vzorke z tréningovej sady.



Obr. 6.1: Ukážky architektúry konvolučného kodéra označeného ako pero_vgg. Tento kodér pozostáva zo 4 blokov, ktoré tvoria konvolučné vrstvy (C) o veľkostiach 64, 128, 256, a 512. Každý konvolučný blok je zložený z dvoch konvolučných vrstiev a je nasledovaný *pooling* vrstvou (M) a normalizáciou s odozvou filtra. Tretia *pooling* vrstva nemení horizontálne rozlíšenie obrazu [23].

postupne o počtoch 64, 128, 256, a 512. Každý konvolučný blok nasleduje normalizácia a *pooling* vrstva, pričom iba prvé dve *pooling* vrstvy redukujú horizontálne rozlíšenie. Okrem spomínanej siete VGG, bola použitá aj jej modifikovaná verzia s odlišným počtom vrstiev a veľkosťou konvolučného jadra a verzia, ktorá má odobratú poslednú *pooling* vrstvu.

Všetky siete boli obdobným spôsobom trénované na dátovej sade PERO HWR (4.2). Pri trénovaní bola použitá počiatočná *learning rate* o veľkosti $0,1 \times 10^{-}3$ a *batch* o veľkosti 16. Pri trénovaní bola v transformeri použitá regularizácia typu *dropout* [43] o veľkosti 30% a ďalej bol použitý *label smoothing* [47] o veľkosti 10%. Siete boli trénované s použitím *early stopping* na testovacej chybe s parametrom *patience* 10, pričom kontrola zastavenia trénovania sa uskutočňovala v každom kroku validácie na testovacej sade. Pri trénovaní boli u sietí resnet34, vgg16 a vgg16* použité váhy, ktoré boli získané trénovaním na dátovej sade ImageNet [37]. U ostatných sietí boli váhy spočiatku náhodne inicializované. Dosiahnuté výsledky sú zobrazené v tabuľke 6.1.

Z tabuľky vidieť, že najlepšie výsledky dosiahla sieť pomenovaná ako pero_vgg, ktorej chyba na testovacej sade predstavuje $3,81$ %. V porovnaní s ostatnými sieťami je táto sieť najmenšia v počte trénovateľných parametrov. O niečo horšie výsledky dosiahla modifikácia

Použité váhy	Tréningová sada	Testovacia sada
	CER	**CER**
náhodné váhy	3,48 %	3,81 %
predtrénované váhy (nezmrazené)	3,82 %	3.52 %
predtrénované váhy (zmrazené)	**3,00** %	**3,41** %

Tabuľka 6.2: Použitie predtrénovaných váh v konvolučnom kodéry označenom ako pero_vgg. Zmrazené váhy znamenajú, že v prvých 10 000 iteráciách učenia nedochádzalo ku ich aktualizácií.

siete VGG s názvom vgg*, ktorá má podobne ako sieť pero_vgg pomerne nízky počet parametrov. Z výsledkov vidieť, že pri použití hlbokej konvolučnej siete je trénovanie náročnejšie a výsledná sieť nedosiahne tak dobré výsledky. Ďalej je taktiež vidieť, že sieť vgg16*, ktorá sa od siete vgg16 líši iba v počte *pooling* vrstiev dosiahla o niečo lepšie výsledky ako sieť vgg16. Vyšší počet *pooling* vrstiev obsahovala podobne architektúra resnet34. Na základe tohoto je pre transformer výhodnejšie, ak je príznaková mapa generovaná konvolučnou sieťou dostatočne rozmerná. Pre porovnanie výstupom siete pero_vgg je vektor o veľkosti 256 x 5 x 378, zatiaľčo napr. výstupný vektor siete resnet34 má rozmer 512 x 2 x 48. Následne sú tieto vektory ešte transformované lineárnou vrstvou pred vstupom do kodéra.

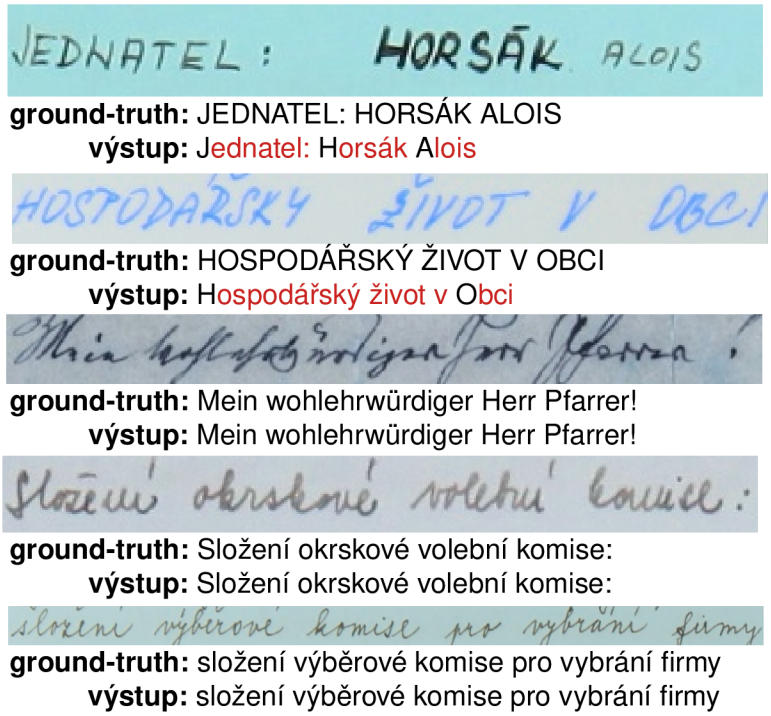
Ďalej som skúmal použitie predtrénovaných váh u konvolučného kodéra pero_vgg, ktorý bol súčasťou rekurentnej neurónovej siete, ktorá bola trénovaná na rovnakej dátovej sade ako moje siete. V tomto experimente som nepoužil náhodne inicializované váhy, ale práve váhy získané trénovaním na rovnakej dátovej sade inou sieťou. U predtrénovaných váh som použil variantu so zmrazenými váhami po dobu 10 000 iterácií, počas ktorých dochádzalo k aktualizácií váh iba v sieti transformer. Výsledky sú zobrazené v tabuľke 6.2.

Z výsledkov vidieť, že použitie predtrénovaných váh prinieslo zlepšenie v presnosti o hodnotu $0,4$ % na testovacej sade. Pri porovnaní modelov s predtrénovanými váhami sa osvedčilo trénovanie, kedy je v počiatočných iteráciách zmrazená aktualizácia váh v konvolučnej sieti. Ak si porovnáme výsledky s náhodnými váhami a predtrenovanými (nezmrazenými) váhami, tak na trénovacej sade dosiahla sieť s náhodnými váhami o niečo lepší výsledok. Toto môže byť spôsobené tým, že pri trénovaní sa chyba CER počíta na náhodnej vzorke z trénovacej sady. Ukážky niektorých vygenerovaných prepisov sú na obrázku 6.2.

Rekurentná neurónová sieť s chybovou funkciou CTC a konvolučným kodérom pero_vgg dosiahla na rovnakej dátovej sade znakovú chybu 3,25 % [23]. Toto je o 0,16 % lepší výsledok ako dosiahla sieť transformer. Napriek tomuto výsledku je však sieť transformer schopná konkurovať rekurentnej sieti.

Použitie beam search algoritmu V tomto experimente som skúmal možnosti zlepšenia výslednej presnosti použitím metódy *beam search* algoritmu. Pri experimente boli overené presnosti s parametrami *beam size* o veľkostiach 3 a 5. Oproti algoritmu *greedy search*, ktorý sa používa aj pri trénovaní siete sa očakáva mierne zlepšenie výsledku, pretože je prehľadávaný väčší stavový priestor. Výsledky sú zobrazené v tabuľke 6.3.

Pri použití parametru *beam size* 3 sa zlepšila presnosť dekódovania o hodnotu 0.8 %. Pri použití vyššieho čísla parametru už nebola ďalej zaznamenaná žiadna zvýšená presnosť. Ukážka niektorých dekódovaní je na obrázku 6.3. Hoci je výsledný prepis presnejší, použitie tohoto algoritmu je výpočtovo náročnejšie ako použitie *greedy search* algoritmu.



Obr. 6.2: Ukážky vygenerovaných prepisov z testovacej dátovej sady. Pri generovaní prepisov bola použitá sieť označená ako *pero_vgg* a boli použité predtrénované (zmrazené) váhy. Prvý riadok textu predstavuje ground-truth, zatiaľčo druhý riadok predstavuje výstup siete. Zaujímavosťou je, že u prvých dvoch obrázkov by bola chyba prepisu nulová, ak by bol výstup siete prevedený na veľké písmená. Takýto „nesprávny“ prepis bol u sieti transformer zaznamenaný niekoľkokrát.

6.2 Vplyv veľkosti siete transformer na úspešnosť predikcie znakov

Cieľom tohoto experimentu bolo zistiť ako môže počet kodér a dekodér blokov v sieti transformer spolu s rozličnou dimenzionalitou jednotlivých blokov ovplyvniť výslednú presnosť siete. Pri experimentoch sa testovali dimenzionality o veľkostiach 256 a 512 pri počte blokov 4 a 6. Vyššie hodnoty neboli volené, pretože s narastajúcim počtom blokov a dimenzionalitou narastá aj samotná veľkosť modelu. Pri veľkom modeli následne dochádza ku problémom uloženia takéhoto modelu na grafickú kartu a obdobne narastá výrazne aj čas trénovania takejto siete pri použití jednej GPU. Pri trénovaní boli zachované obdobné parametre ako pri experimente s voľbou konvolučného kodéra (6.1). Výsledky experimentu sú zobrazené v tabuľke 6.4.

Najlepšie výsledky dosiahol model, ktorý obsahoval 6 kodér a dekodér blokov a mal dimenzionalitu 512. O niečo horšie výsledky dosiahla architektúra, ktorá obsahovala 6 kodér blokov a 4 dekodér bloky. Táto architektúra dosiahla o 0,42 % horšie výsledky. Ostatné architektúry dosiahli chybovosti na testovacej dátovej sade vyššie ako 4 %. Pri porovnaní

Beam size	Testovacia sada CER
1 (greedy search)	3,41 %
3	**3,33** %
5	**3,33** %

Tabuľka 6.3: Predikcia sekvencie s použitím *beam search* algoritmu. Pri voľbe hodnoty *beam size* 3 došlo ku miernemu zlepšeniu. Pri hodnote *beam size* 5 už viac nedochádza ku zlepšovaniu presnosti u zvolenej dátovej sady.

ground-truth: na ležení a vůz byl vytápěn kamný na uhlí. To bylo pro nás přijatelné. Na rozpálená
výstup (beam 1): na leženi a vůz byl vytápěn začíny na uklí. To bylo pro nás přijatelné. Na rozpálená
výstup (beam 3): na leženi a vůz byl vytápěn každý na uklí. To bylo pro nás přijatelné. Na rozpálená

ground-truth: Máma říká Táta z bratrem ti to všechno potrhali. Říkali, že kdoví jestli se vratíš
výstup (beam 1): Máma říká Táta _ bratrem ti to všechno potrhali. Říkali, že kdoví jestli se v_rutíš
výstup (beam 3): Máma říká Táta _ bratrem ti to všechno potrhali. Říkali, že kdoví jestli se vratiš

Obr. 6.3: Ukážka predikcie sekvencie pri použití algoritmu *beam search*. *Beam size* o veľkosti 1 predstavuje algoritmus *greedy search*. U niektorých znakov došlo vďaka prehľadávaniu ďalších pravdepodobných znakov ku zlepšeniu výsledného prepisu. Z obrázku vidieť, že u väčšieho prehľadávania došlo ku takmer presnej predikcií slova „vratíš“. Pri predikcií slova „kamný“ bolo vygenerované odlišné slovo „každý“, ale napriek tomu sa zmenšila výsledná chyba.

veľkosti sietí však nie je zrejmé, či malé modely dosiahli horšie výsledky mimo dvoch najlepších architektúr. U architektúr s dimenzionalitou 256 totiž nie je zaznamenaný žiadny nárast chyby s klesajúcim počtom blokov. Autori *Kang a spol.* [18] zvolili v ich práci architektúru s dimenzionalitou 1024 a menší počet blokov 4. Podľa mojich výsledkov zvýšenie dimenzionality prispelo ku zlepšeniu modelu. Pri počte blokov 4 som pri mojich hodnotách nezaregistroval pozitívny prínos. Zvyšovaním počtu blokov sa sieti dodávajú ďalšie attention vrstvy a sieť sa tak pravdepodobne môže učiť nové pozornosti.

Trénovanie na dátovej sade IAM Pri tomto experimente som sa rozhodol porovnať najúspešnejší model získaný z tejto práce s ostatnými modelmi, ktoré boli trénované a testované na dátovej sade IAM [30]. Na začiatku ju nutné spomenúť, že oba články [18, 48], s ktorými je model porovnávaný boli trénované a testované na odlišných dátových podmnožinách. Pri trénovaní siete som použil náhodné zamiešanie dát a rozdelenie dátovej sady, na ktorej som následne natrénoval sieť. Tréningová dátová sada obsahovala 8 344 riadkov, zatiaľčo testovacia dátová sada obsahovala 3 000 riadkov. Výsledné hodnoty experimentu sú následne iba orientačné a slúžia na overenie správnosti implementácie a postupu trénovania siete. Pri experimente boli použité tri modely. Prvý model predstavuje klasický model s konvolučným kodérom pero_vgg z experimentu 6.1, u ktorého boli náhodne inicializované počiatočné váhy. Následne boli použité dva ďalšie modely, ktoré používajú aj predtrénované

Dimenzionalita modelu	Počet kodér blokov	Počet dekodér blokov	CER
256	4	4	4,23 %
256	4	6	4,13 %
256	6	4	4,37 %
256	6	6	4,16 %
512	4	4	4,24 %
512	4	6	4,41 %
512	6	4	3,83 %
512	6	6	**3,41** %

Tabuľka 6.4: Porovnanie úspešnosti modelu pri rozličných dimenzionalitách a počtom kodér a dekodér blokov. Najlepšie výsledky dosiahla najväčšia architektúra s dimenzionalitou 512 a počtom blokov 6 a 6. Táto architektúra predstavuje základnú transformer architektúru, ktorá vystupuje aj v pôvodnej práci o transformeroch [47]. O niečo horšie výsledky dosiahla architektúra s rovnakou dimenzionalitou, u ktorej bol volený nižší počet dekodér blokov. Ostatné architektúry dosiahli porovnateľné výsledky.

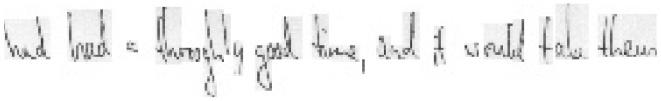
Transformer model	CER
model z tejto práce	2,48 %
model z tejto práce (predtrénované CNN)	2,18 %
predtrénovaný model z tejto práce (predtrénovaná celá sieť)	**1,6** %
Lei Kang a spol. [18]	4,67 %
Christoph Wick a spol. [48]	5,67 %

Tabuľka 6.5: Porovnanie úspešnosti modelov na dátovej sade IAM [30]. Najlepšie výsledky dosiahli architektúry, ktoré používajú predtrénované váhy. Sieť s náhodne inicializovanými váhami dosiahla však po natrénovaní o 2,19 % lepšie výsledky na testovacej dátovej sade ako model od *Lei Kang a spol.* [18].

váhy z konvolučnej siete a aj z celej siete. Výsledky experimentu sú zobrazené v tabuľke 6.5.

Podľa výsledkov boli prekonané ostatné architektúry, ktoré obdobne experimentujú s transformer architektúrami pri rozpoznávaní ručne písaného textu. Ako bolo ale spomenuté vyššie, ide o odlišné podmnožiny dátových sád. Základný model dosiahol chybovosť 2,48 %. Pridaním predtrénovaných váh z konvolučnej siete, prípadne pridaním váh z celej siete a dotrénovaním modelu sa výsledky ešte viac zlepšili. Autori *Christoph Wick a spol.* [48] potvrdili, že transformery vyžadujú sofistikovaný výber hyperparametrov a predtrénované váhy pre konvolučný kodér. Na obrázku 6.4 sú zobrazené niektoré výstupy siete, ktorá bola trénovaná s náhodnou inicializáciou váh.

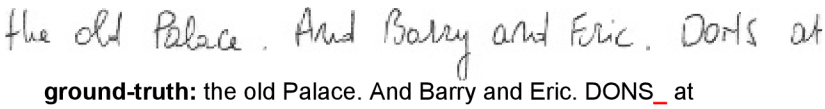
Architektúry [18] a [48] sa od natrénovanej architektúry z tejto práce odlišujú prevažne vo veľkosti siete transformer, čo sa týka počtom blokov a dimenzionalitou, ale odlišujú sa aj veľkosťou použitej konvolučnej siete. Ich modely výrazne prevyšujú počet trénovateľných parametrov v porovnaní so sieťou použitou v tejto práci. Pri experimente s konvolučnou sieťou 6.1 bolo rovnako v tejto práci poukázané, že väčšia sieť dosiahla na testovacej dá-



ground-truth: had had_ a thoroughly good time, and it would take them
výstup: had been a th_roughly good time, and it would take them

ground-truth: look as though it was only what_ you expected of me,
výstup: look as though it was only which you esqualed of me,

ground-truth: MR SELWYN LLOYD may speed up his plans
výstup: Mr Selwyn Lloyd may speed up his plans



ground-truth: the old Palace. And Barry and Eric. DONS_ at
výstup: the old Palace. And Batty and Eric. Dorts at

Obr. 6.4: Ukážka rozpoznaných riadkov z testovacej dátovej sady datasetu IAM. Prvý riadok je obrázok z testovacej sady, nasledujúci riadok reprezentuje ground-truth. Tretí riadok je výstup modelu z práce bez predtrénovaných váh. Z výstupu vidieť, že u niektorých slov akými sú napr. „Mr Selwyn Lloyd“ bol nesprávne anotovaný dataset, ale sieť sa napriek tomu naučila vytvoriť správny prepis. U slova „what“ následne sieť vygenerovala slovo „which“, ktoré je vizuálne aj jazykovo blízke.

tovej sade horšie výsledky. Potencionálne problémy by mohli byť u použitej dátovej sady, augumentácií dát, prípadne vo veľkostí hodnôt regularizačných techník akými sú *dropout* a *label smoothing*.

6.3 Vplyv rozličnej tokenizácie na úspešnosť siete

V tomto experimente som skúmal vplyv rozličnej tokenizácie textu. Pri trénovaní siete som pracoval s knižnicou *sentencepiece*[1], ktorá umožňuje tokenizovať sekvencie. V pôvodných experimentoch je sieť trénovaná so znakovou tokenizáciou s veľkosťou slovníka 513. U tokenizácie *sentencepiece* som použil dve verzie algoritmov a to tzv. *Byte-Pair Encoding* [40] (BPE) a *unigram* [24]. Pri trénovaní siete s použitým algoritmom unigram bola použitá regularizácia v podobe augumentácie tokenizovanej sekvencie. Pri tejto regularizácií môže byť jedna sekvencia kódovaná postupnosťou rôznych tokenov. Výsledky experimentu sú zobrazené v tabuľke 6.6.

Z výsledkov vidieť, že žiadny typ tokenizácie nedosiahol lepšie výsledky ako tokenizácia po jednotlivých znakoch, pri ktorej dosiahol model úspešnosť $3,41\%$. Pri odlišnej tokenizácií dosiahol algoritmus BPE o niečo lepšie výsledky ako algoritmus *unigram*, výnimkou prípadu, kedy bol použitý slovník o počte 1024 tokenov. Pri tomto počte dosiahol lepšie výsledky algoritmus *unigram*. Podľa výsledkov nie je pri generovaní sekvencie vidieť výrazné zlepšovanie alebo zhoršovanie, ak sa použije slovník s väčším alebo menším počtom

[1] `https://github.com/google/sentencepiece`

Tokenizácia	Algoritmus	Slovník	CER
znaky	-	513	**3,41** %
sentencepiece [25]	unigram	512	**4,35** %
	unigram	1024	4,42 %
	unigram	2048	4,5 %
	unigram	4096	4,36 %
	bpe	512	4,46 %
	bpe	1024	4,06 %
	bpe	2048	**4,02** %
	bpe	4096	4,21 %

Tabuľka 6.6: Porovnanie úspešnosti vzhľadom ku použitej tokenizácií. CER predstavuje chybu na testovacej dátovej sade. Číselná informácia v stĺpci slovník reprezentuje počet použitých tokenov pri danej tokenizácií. Podľa výsledkov dosiahne model najlepšie výsledky $(3,41\%)$ pri použití tokenizácie znakov. Pri použití odlišnej tokenizácie nie je zrejmé, či narastajúcim slovníkom narastá aj výsledná presnosť.

Počet dekódovaných tokenov	Slovník	Čas (v sec.)
235 938	znaky	372,9
126 891	513	243,0
108 785	1024	213,4
95 211	2048	198,2
84 294	4096	187,1

Tabuľka 6.7: Porovnanie rýchlosti inferencie tokenizácií s rozličnou veľkosťou slovníka na testovacej sade. V experimente sa porovnávala tokenizácia znakov a tokenizácia s využitím BPE. Testovacia sada má v celkovom počte 236 024 znakov. Prvý stĺpec značí celkový počet tokenov, ktorý sieť pri danej tokenizácií vygenerovala. Čas označuje čas dekódovania testovacej sady so zvoleným slovníkom.

tokenov. Pri algoritme BPE však dosiahol model najlepšie výsledky, keď bol použitý slovník o veľkosti 1024 a 2048 tokenov.

Ďalej je v tabuľe 6.7 vidieť meraný čas inferencie siete na testovacej sade. Použitie tokenizácie s väčším počtom tokenov má podľa výsledkov pozitívny vplyv na urýchlenie rýchlosti dekódovania sekvencie. Vyplýva to z toho, že sekvencie sú pravdepodobne častejšie kódované menším počtom tokenov u rozsiahlejších slovníkoch. Toto nastáva aj napriek tomu, že rozsiahlejšie slovníky obsahujú okrem ďalších častí slov aj všetky znaky, ktorými by mohla sieť kódovať sekvencie.

6.4 Trénovanie modelu s textovým korpusom

Pri tomto experimente sa skúma vplyv textových korpusov pri trénovaní siete transformer. Pri trénovaní siete sa používa dávka, resp. batch, ktorý pozostáva z obrázkovej časti, tzv. obrázkový batch (tvoria ho dvojice, vstupný obraz – odpovedajúci prepis) a z textovej

Počet obrázkov v dávke	Počet textov v textovej dávke	CER
16	-	2,54 %
8	16	3,15 %
12	12	2,66 %
16	8	2,62 %
16	16	**2,47** %

Tabuľka 6.8: Porovnanie úspešnosti modelov, trénovaných spoločne na obrázkových dátach a aj textových korpusoch, na dátovej sade vytvorenej z českých textov. Trénovací batch pozostáva z obrázkových dvojíc (obrázok – prepis) a viet z českého textového korpusu. Princíp trénovania je popísaný v časti 5.4. Z výsledkov vidieť, že pri rovnakej veľkosti obrázkového batchu dosiahla sieť trénovaná aj spolu s textovými korpusmi o niečo lepšie výsledky ako sieť trénovaná iba na obrázkových dátach. Pri nižšom počte obrázkov v dávke vidieť mierne zväčšovanie chyby.

časti, tzv. textový batch (tvorí ho text z textového korpusu). Popis trénovania je popísaný aj v časti 5.4. Pri trénovaní siete je sieť trénovaná čiastočne na obrázkovej dávke a čiastočne na textovej dávke pozostávajúcej iba z textových dát. U experimentov sa ako obrázková dávka použila podmnožina z dátovej sady PERO HWR, z ktorej boli extrahované české riadky. Textová dávka pozostáva následne z textových riadkov z dátovej sady obsahujúcej cez 109 miliónov riadkov textu (4.2). U experimentu bolo volených niekoľko rôznych kombinácií veľkostí dávky batch a zastúpenia jej jednotlivých častí. Výsledky experimentu sú zobrazené v tabuľke 6.8.

Z výsledkov experimentu je vidieť menší prínos trénovania siete spolu s dávkou tvorenou z textových korpusov. Najlepšie výsledky dosiahla sieť, ktorá bola trénovaná s textovou dávkou o veľkosti 16. Jej výsledná chybovosť predstavuje 2,47 %, čo je avšak iba o 0,07 % lepší výsledok ako dosiahol model trénovaný na samostatnej obrázkovej dávke. O niečo horšie výsledky dosiahli siete trénované na textovej dávke o veľkosti 12 a 8. Trochu zaujímavý je výsledok modelu, u ktorého bola použitá nižšia veľkosť obrázkovej dávky o hodnote 8. V tomto prípade dosiahla sieť horšie výsledky. U tohoto prípadu je pravdepodobné, že textová informácia sieť pri učení zmiatla, a práve to spôsobilo horšie výsledky. Z výsledkov nie je úplne isté, či použitie textových korpusov prispieva pozitívne ku trénovaniu, pretože rozdiel medzi modelmi trénovaných na obrázkovej dávke a kombinácií s textovou dávkou je minimálny.

V tabuľke 6.9 sú následne porovnané modely trénované na dátovej sade PERO HWR. Prvý model reprezentuje model získaný z experimentov 6.1 a 6.2. Druhý model je rovnaký model, ktorý bol na rozdiel od prvého modelu trénovaný s rovnakou hodnotou dodatočnej textovej dávky. Z výsledkov vidieť, že nový model, ktorý bol trénovaný spolu s textovými korpusmi dosiahol o dosť viditeľné zníženie presnosti na testovacej dátovej sade u oboch sád v tabuľke. Za tento fakt môže čiastočne možno viacjazyčnosť dátovej sady oproti korpusu, ktorý je celý v českom jazyku. Použitie korpusu však neprinieslo ani zlepšenie presnosti na českom datasete. Podľa tohoto je pre transformer pravdepodobne výhodnejšie používať klasické obrázkové dáta a dodatočná textová informácia iba komplikuje trénovací proces.

V ďalšej časti sa skúmal ďalej vplyv textového korpusu, ak je v dátovej sade nižší počet dát použitých pri trénovaní. Z pôvodnej dátovej sady českých textov bolo extrahovaných

Počet obrázkov v dávke	Počet textov v textovej dávke	CER (PERO HWR)	CER (český dataset)
16	0	**3,41**	**1,90** %
16	16	3,56 %	2,01 %

Tabuľka 6.9: Porovnanie úspešnosti modelu trénovaného na obrázkovej dávke batch a modelu trénovaného spoločne na obrázkových dátach a textovom korpuse, na dátovej sade PERO HWR. Pri testovaní bola použitá aj česká dátová sada. Ako textový korpus bol napriek viacjazyčnosti dátovej sady PERO HWR volený textový korpus českého jazyka. Z výsledkov vidieť, že natrénovaný model dosiahol na dátovej sade PERO HWR mierne zhoršenie presnosti (0,15 %), ak bol pri trénovaní použitý aj textový korpus. Pri porovnaní výsledkov na českom datasete dosiahol obdobne lepšie výsledky klasický transformer trénovaný na obrázkových dátach.

Typ korpusu	Podiel dát	CER
-	20 %	3,16 %
-	40 %	**2,86** %
český korpus	20 %	3,60 %
český korpus	40 %	3,23 %
korpus textov z dátovej sady	20 %	3,51 %
korpus textov z dátovej sady	40 %	3,22 %

Tabuľka 6.10: Porovnanie úspešnosti vzhľadom ku počtu dát a použitiu korpusu. Český korpus predstavuje dátovú sadu z časti 4.2. Korpus textov z dátovej sady reprezentuje prepisy riadkov, ktoré sa vyskytujú v celej tréningovej množine (nielen v podmnožine na ktorej boli trénované siete).

postupne 20 % a 40 % riadkov, ktoré boli samostatne použité pri trénovaní siete. Následne sa skúma vplyv dodatočného českého korpusu na úspešnosť sietí. Ako korpus som v tomto experimente použil pôvodný český korpus a rovnako som sa pokúsil použiť aj textové prepisy z celej tréningovej dátovej sady, ako ďalší textový korpus. Výsledky sú zobrazené v tabuľke 6.10.

Podľa výsledkov z tabuľky 6.10 dosiahol najlepšie výsledky model, ktorý nebol trénovaný s použitím textových korpusov. Ďalej je z výsledkov viditeľné, že modely, ktoré boli trénované na väčšom množstve dát dosahujú lepšie presnosti ako modely trénované na polovičnej dátovej sade. Pri porovnaní modelov trénovaných s použitím textových korpusov dosiahli lepšie výsledky modely, ktoré použili ako textový korpus texty z tréningovej dátovej sady. Podľa tohoto je pre sieť určite výhodnejšie, ak je použitý korpus jazykovo podobný textom, ktoré sa nachádzajú na obrázkoch.

Podľa celkových výsledkov je však pre transformer náročné využiť informáciu z textového korpusu. Jedným z problémov môže byť práve informácia prichádzajúca z výstupu kodéra. Transformer by potreboval pravdepodobne lepšiu informáciu, ktorá by napr. čiastočne bola reprezentovaná umelým stavom, ktorý by bol charakterizovaný textom, ktorý sa pokúšame dekódovať.

Počet použitých prefixov (riadkov)	CER bez prefixu	CER s prefixom	
		všetky prepisy	bez prázdnych
0	**2,97** %	-	-
1	3,97 %	18,95 %	8,15 %
2	4,25 %	9,27 %	9,05 %

Tabuľka 6.11: Porovnanie úspešnosti sietí vzhľadom ku množstvu dodatočnej textovej informácie v podobe textových prefixov pri dekódovaní sekvencie. Siete boli s touto dodatočnou informáciou aj trénované. Prvý stĺpec označuje koľko riadkov textu bolo u trénovania použitých. Pri inferencií siete sa následne testovalo, či dokáže poskytnutie textového prefixu prispieť ku zvýšeniu presnosti dekódovania sekvencie. Posledný stĺpec (bez prázdnych) označuje testovaciu chybu CER bez riadkov, ktoré boli dekódované ako prázdne.

6.5 Trénovanie siete a dekódovanie sekvencie s použitým textového prefixu

V tomto experimente sa pri trénovaní neurónovej siete využíva dodatočná textová informácia v podobe textového prefixu. Textový prefix predstavuje časť textu, ktorý bezprostredne predchádza textovému prepisu z obrázka. Týmto prefixom a štartovacím tokenom je vo fázy dekódovania inicializovaný dekodér. Pre účely použitia takýchto prefixov bola obstaraná nová dátová sada, ktorá je popísaná v časti 5.5 s popisom tejto metódy v časti 5.5.

Výsledky obstarané pri inferencií siete sú zobrazené v tabuľke 6.11. V tabuľke je vidieť znakovú chybu CER počítanú na dvoch odlišných testovacích sadách. Prvá sada predstavuje dátovú sadu bez použitia textových prefixov. U druhej dátovej sady sú pri niektorých riadkoch dostupné textové prefixy. Okrem iného je v tabuľke zobrazený ďalší údaj v podobe chyby CER, ktorá bola počítaná bez riadkov, ktoré sieť vygenerovala ako prázdne. U tohoto prípadu sa jedná o chybu, ktorá bola počítaná na testovacej sade s prefixmi, pričom sa chyba nepočítala na prepisoch, u ktorých sieť vygenerovala prázdny prepis. Tento údaj tam je z dôvodu, pretože pri použití prefixov nastala často situácia, kedy si sieť pravdepodobne nedokázala poradiť s dlhým textom a výstupom siete boli sekvencie obsahujúce *padding* token, ktorý sa následne po dekódovaní transformoval na prázdny reťazec.

Podľa výsledkov je vidieť, že narastajúcim počtom dodatočnej informácie pri dekódovaní narastá aj celková chyba siete pri dekódovaní klasického textu bez prefixov. Ak sa zameriame na text s prefixmi, tak je zaujímavé, že sieť s menším počtom dodatočných riadkov dosiahla takmer dvojnásobne horší výsledok pri dekódovaní. Pri tomto dekódovaní dochádzalo často práve ku vyššie spomínanému problému, kedy sieť vôbec negenerovala žiadnu zmysluplnú informáciu. Ak ale vynecháme sekvencie, ktoré boli generované ako prázdne, tak sa chyba u sietí trénovaných s jedným a dvomi textovými prefixmi viac približuje.

Ďalej bolo overované správanie klasickej transformer siete, ktorá nebola trénovaná s použitím prefixov. V tabuľke 6.12 sú zobrazené výsledky z procesu dekódovania tejto siete, kde boli na vstup postupne pridané textové prefixy. Z výsledkov vidieť, že sieť sa nepredstaviteľne zhoršila a dosiahla chybovosť viac ako 70 %. V tomto prípade sieť, podobne ako siete trénované s prefixmi, generovala prázdne prepisy. Podľa tohoto je pre transformer náročné dokázať aplikovať mechanizmus *attention* na dlhé sekvencie. Sekvencie dosiahli príliš veľkú dĺžku a chyba sa s narastajúcou dĺžkou zväčšovala. Rovnako z experimentu vyplýva,

Počet dodatočných riadkov	CER
0	**2,97** %
1	71,28 %
2	93,73%

Tabuľka 6.12: Úspešnosť dekódovania siete z experimentu 6.2 pri použití textových prefixov u dekódovania. Sieť nebola trénovaná s textovými prefixami.

že použitie takejto informácie sa musí sieť spočiatku naučiť a nie je to možné aplikovať u akéhokoľvek modelu.

6.6 Zhrnutie výsledkov

Vykonané experimenty s architektúrou transformer poukazúju na to, že architektúra má potencionálne využitie pri rozpoznávaní ručne písaného textu. Pri experimentovaní s veľkosťou konvolučného kodéra a siete transformer bolo zistené, že menšiu sieť bolo jednoduchšie natrénovať ako sieť z vyšším počtom trénovateľných váh. Najlepšie výsledky dosiahla najmenšia sieť s označením pero_vgg. Táto sieť dosiahla chybu na testovacej dátovej sade o hodnote 3,41 %. Obdobne bolo v tejto práci overené, že sieť s predtrénovanými váhami u konvolučného kodéra dosiahne lepšie výsledky, čo popisú aj *Christoph Wick a spol.* [48]. Pri experimentovaní s veľkosťou siete transformer dosiahla lepšie výsledky rozmernejšia sieť s vyššou dimenzionalitou v jednotlivých blokoch.

Pri použití odlišnej tokenizácie od tokenizácie znakov bola u sietí zaznamenaná mierne zvýšená chyba. U väčších slovníkoch však chyba ďalej nenarastala. Pri použití väčších slovníkov sa sieť správne naučila používať tokeny, ktoré zastupujú viacero znakov a tým pádom to prinieslo rýchlejšie dekódovanie vo fázy inferencie siete.

Sieť bola natrénovaná na dátovej sade IAM a výsledky poukazujú na vysoký potenciál siete transformer. Na obrázku 6.4 je viditeľná vlastnosť, kedy sa sieť snaží využívať svoj vnútorný jazykový model a predikovať nielen znaky, ktoré vidí na obraze, ale aj jazykovo blízke slová, prípadne slová ktoré nadobúdajú vo slovom spojení význam („had been“ namiesto „had had“, „which“ namiesto „what“). Hoci ide o chybne dekódovanie slov, tak je ich generovanie dosť odôvodniteľné podľa spomenutých jazykových vlastností. Toto rovnako popisujú *Christoph Wick a spol.* [48].

Pri trénovaní siete s textovými korpusmi boli zistené minimálne prínosy. Pri použití textových korpusov nedošlo k výraznému zhoršeniu výsledkov. Ku zlepšeniu výsledku došlo iba u varianty, kedy bola sieť trénovaná na českej dátovej sade s českým korpusom. Akonáhle bola však sieť trénovaná na väčšej dátovej sade, tak dosiahla sieť trénovaná bez textových korpusov lepšie výsledky na všetkých dátových sadách.

U trénovaní siete s dodatočnou textovú informáciou v podobe prefixu bol zaznamenaný problém pri použití takýchto prefixov. Pri väčšej dĺžke prefixu chyba siete narastala a takáto informácia vo forme prefixu nepriniesla žiadne zlepšenie siete, ale na druhú stranu priniesla výrazne zhoršenie. Architektúra transformer nedokáže pracovať s takto dlhými sekvenciami a častokrát generuje prázdne znaky. Jedným z dôvodov, prečo sa sieť nedokázala textové prefixy používať, môže byť aj kvalita dátovej sady. U viacerých textových prefixoch nebol na prvý pohľad viditeľný súvis dvoch častí textu. Práve takáto nadbytočná informácia mohla mať za následok, že sa sieť správne netrénovala.

Kapitola 7

Záver

Cieľom tejto práce bolo pripraviť prehľad o neurónovej sieti typu transformer a jej aplikácií pri rozpoznávaní ručne písaného textu. V tejto práci som vychádzal z existujúcej architektúry siete, ktorá je implementovaná v projekte PERO[1]. Túto architektúru som ďalej rozšíril a uskutočnil nad ňou experimenty. Pri trénovaní siete som použil dátovú sadu PERO HWR so zameraním na české texty. Najúspešnejší model dokázal na testovacej dátovej sade dosiahnuť chybovosť 3,41 %, čo predstavuje iba o 0.16 % horšiu presnosť rozpoznávania ručne písaného písma akú dosahuje rekurentná neurónová sieť s chybovou funkciou CTC [23]. Vo výstupoch siete transformer je vidieť, že za vyššiu chybu mohlo u niektorých prípadov nesprávne rozlíšenie veľkých a malých písmen.

Pri experimentoch som skúmal aj vplyv rozličnej tokenizácie textu. Podľa experimentu dosahuje najlepšie výsledky sieť používajúca tokenizáciu po znakoch. Ostatné typy tokenizácie dosiahli o niečo horšie výsledky, ale pri väčších slovníkoch sa sieť naučila generovať nižší počet tokenov a urýchliť inferenciu siete. Pri ďalšom experimente som skúmal vplyv textových korpusov českého jazyka na zlepšenie výslednej presnosti siete. Pri trénovaní siete obsahovala tréningová dávka okrem obrázkovej dávky aj ďalší textový batch. Tento textový batch sa následne použil pri trénovaní dekodéra. Podľa výsledkov experimentov však pri takomto trénovaní nebol viditeľný žiadny väčší prínos a najlepšie výsledky dosiahla sieť, ktorá nebola s textovými korpusmi trénovaná. Ďalej bol v rámci práce skúmaný vplyv použitia textových prefixov u inicializácie dekodéra počas trénovania aj testovania siete. Pre tieto účely bola pripravená nová dátová sada, ktorá obsahuje informáciu o prepise obrazu a aj odpovedajúci textový prefix. Na vstup dekodéra bola pridaná dodatočná textová informácia, ktorá pochádzala z predchádzajúceho riadka prepisovaného obrázka. Ani pri tomto experimente však podľa výsledkov nebolo viditeľné zlepšenie presnosti dekódovania. Sieť dosiahla horšie výsledky, generovala veľké množstvo prázdnych prepisov a mala problém dekódovať koniec sekvencie.

Prínosom tejto práce sú výsledky experimentov a rozšírenie implementácie siete transformer. Ďalší vývoj by sa mohol zamerať v použití obojsmerného dekódovania sekvencie, prípadne v použití dvoch nezávislých dekodérov pri dekódovaní. Rovnako by sa mohla pri trénovaní siete použiť so sieťou transformer chybová funkcia CTC. Ďalšie experimenty by mohli spočívať v trénovaní väčších sietí a zakomponovanie ďalších dátových sád a typov sietí transformer do experimentov.

[1] `https://pero.fit.vutbr.cz/`

Literatúra

[1] Ba, J. L., Kiros, J. R. a Hinton, G. E. Layer normalization. *ArXiv preprint arXiv:1607.06450*. 2016.

[2] Bahdanau, D., Cho, K. a Bengio, Y. Neural machine translation by jointly learning to align and translate. *ArXiv preprint arXiv:1409.0473*. 2014.

[3] Bengio, Y., Simard, P. a Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*. IEEE. 1994, zv. 5, č. 2, s. 157–166.

[4] 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. DOI: 10.1109/ICDAR.2017.111.

[5] Cho, K., Van Merriënboer, B., Bahdanau, D. a Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv preprint arXiv:1409.1259*. 2014.

[6] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *ArXiv preprint arXiv:1406.1078*. 2014.

[7] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K. a Bengio, Y. Attention-based models for speech recognition. *ArXiv preprint arXiv:1506.07503*. 2015.

[8] Chowdhury, A. a Vig, L. An efficient end-to-end neural model for handwritten text recognition. *ArXiv preprint arXiv:1807.07965*. 2018.

[9] Dolfing, H. J., Bellegarda, J., Chorowski, J., Marxer, R. a Laurent, A. The "ScribbleLens" Dutch Historical Handwriting Corpus. In: IEEE. *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, s. 67–72.

[10] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] 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*. 2006, s. 369–376.

[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*. 2006, s. 369–376.

[13] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. et al. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*. IEEE. 2008, zv. 31, č. 5, s. 855–868.

[14] Graves, A. a Schmidhuber, J. Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in neural information processing systems*. 2008, zv. 21, s. 545–552.

[15] Grüning, T., Leifert, G., Strauss, T., Michael, J. a Labahn, R. A two-stage method for text line detection in historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*. Springer. 2019, zv. 22, č. 3, s. 285–302.

[16] He, K., Zhang, X., Ren, S. a Sun, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, s. 770–778.

[17] Hochreiter, S. a Schmidhuber, J. Long short-term memory. *Neural computation*. MIT Press. 1997, zv. 9, č. 8, s. 1735–1780.

[18] Kang, L., Riba, P., Rusiñol, M., Fornés, A. a Villegas, M. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *ArXiv preprint arXiv:2005.13044*. 2020.

[19] 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: Springer. *German Conference on Pattern Recognition*. 2018, s. 459–472.

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

[21] Kleber, F., Fiel, S., Diem, M. a Sablatnig, R. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In: IEEE. *2013 12th international conference on document analysis and recognition*. 2013, s. 560–564.

[22] Kodym, O. a Hradiš, M. Page layout analysis system for unconstrained historic documents. In: Springer. *International Conference on Document Analysis and Recognition*. 2021, s. 492–506.

[23] Kohút, J. a Hradiš, M. TS-Net: OCR trained to switch between text transcription styles. In: Springer. *International Conference on Document Analysis and Recognition*. 2021, s. 478–493.

[24] Kudo, T. Subword regularization: Improving neural network translation models with multiple subword candidates. *ArXiv preprint arXiv:1804.10959*. 2018.

[25] Kudo, T. a Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *ArXiv preprint arXiv:1808.06226*. 2018.

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

[27] Lin, T.-Y., Goyal, P., Girshick, R., He, K. a Dollár, P. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. 2017, s. 2980–2988.

[28] Lu, N., Yu, W., Qi, X., Chen, Y., Gong, P. et al. Master: Multi-aspect non-local network for scene text recognition. *Pattern Recognition*. Elsevier. 2021, zv. 117, s. 107980.

[29] Luong, M.-T., Pham, H. a Manning, C. D. Effective approaches to attention-based neural machine translation. *ArXiv preprint arXiv:1508.04025*. 2015.

[30] Marti, U.-V. a Bunke, H. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*. Springer. 2002, zv. 5, č. 1, s. 39–46.

[31] Michael, J., Labahn, R., Grüning, T. a Zöllner, J. Evaluating sequence-to-sequence models for handwritten text recognition. In: IEEE. *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, s. 1286–1293.

[32] Nair, V. a Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In: *Icml*. 2010.

[33] Olah, C. Understanding LSTM Networks. 2015. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[34] Pletschacher, S. a Antonacopoulos, A. The page (page analysis and ground-truth elements) format framework. In: IEEE. *2010 20th International Conference on Pattern Recognition*. 2010, s. 257–260.

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

[36] Ronneberger, O., Fischer, P. a Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: Springer. *International Conference on Medical image computing and computer-assisted intervention*. 2015, s. 234–241.

[37] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S. et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*. Springer. 2015, zv. 115, č. 3, s. 211–252.

[38] Sanchez, J. A., Romero, V., Toselli, A. H. a Vidal, E. ICFHR2016 competition on handwritten text recognition on the READ dataset. In: IEEE. *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, s. 630–635.

[39] Sanchez, J. A., Toselli, A. H., Romero, V. a Vidal, E. Icdar 2015 competition htrts: Handwritten text recognition on the transcriptorium dataset. In: IEEE. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. 2015, s. 1166–1170.

[40] Sennrich, R., Haddow, B. a Birch, A. Neural machine translation of rare words with subword units. *ArXiv preprint arXiv:1508.07909*. 2015.

[41] 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. *IEEE transactions on pattern analysis and machine intelligence*. IEEE. 2016, zv. 39, č. 11, s. 2298–2304.

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

[43] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*. JMLR. org. 2014, zv. 15, č. 1, s. 1929–1958.

[44] Sueiras, J., Ruiz, V., Sanchez, A. a Velez, J. F. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing*. Elsevier. 2018, zv. 289, s. 119–128.

[45] Sutskever, I., Vinyals, O. a Le, Q. V. Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems*. 2014, s. 3104–3112.

[46] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. a Wojna, Z. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, s. 2818–2826.

[47] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is all you need. In: *Advances in neural information processing systems*. 2017, s. 5998–6008.

[48] Wick, C., Zöllner, J. a Grüning, T. Transformer for Handwritten Text Recognition Using Bidirectional Post-decoding. In: Springer. *International Conference on Document Analysis and Recognition*. 2021, s. 112–126.

[49] Zhang, A., Lipton, Z. C., Li, M. a Smola, A. J. Dive into Deep Learning. *ArXiv preprint arXiv:2106.11342*. 2021.

Príloha A

Obsah priloženého DVD

Obsahom priloženého pamäťového média je:

- *xvesel79.pdf* - text bakalárskej práce,
- latex/ - zdrojové súbory v LaTeXpre vysádzanie textu diplomovej práce,
- src/ - modifikované zdrojové súbory projektu PERO,
- data/ - dátová sada,
- checkpoints/ - uložené váhy modelov,
- *video.mp4* - video prezentujúce výsledky práce,
- *README.txt* - textový súbor s popisom obsahu pamäťového média a spúšťania.