VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ TEXTU POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ

DEEP NEURAL NETWORKS FOR TEXT RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. DANIEL KAVULIAK**
AUTHOR

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

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

148666

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Kavuliak Daniel, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Strojové učení
Název:	**Rozpoznávání textu pomocí hlubokých neuronových sítí**
Kategorie:	Zpracování obrazu
Akademický rok:	2022/23

Zadání:

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

Literatura:

- SHI, Baoguang; BAI, Xiang; YAO, Cong. 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*, 2016, 39.11: 2298-2304.
- KANG, Lei, J. Ignacio TOLEDO, Pau RIBA, Mauricio VILLEGAS, Alicia FORNÉS a Marçal RUSIOL. Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition. In: *Pattern Recognition*. Stuttgart, Germany: Springer International Publishing, 2019, s. 459-472. ISBN 978-3-030-12938-5.
- SHENG, Fenfen; CHEN, Zhineng; XU, Bo. NRTR: A no-recurrence sequence-to-sequence model for scene text recognition. In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019. p. 781-786.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních třech bodů zadání.
- Rozpracovaný čtvrtý a pátý bod zadání.

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

Vedoucí práce:	**Kišš Martin, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	31.10.2022

Abstrakt

Cieľom tejto práce je zostrojiť model na rozpoznanie rukou písaných slov, ktorý bude používať non-autoregresívny dekóder. Tento typ dekóderu počíta predikcie znakov nezávisle na ostatných predikovaných znakoch, čo môže byť výhodné z hľadiska rýchlosti inferencie, ale kvalita predikcie je horšia. Motiváciou je navrhnúť non-autoregresívny dekóder, ktorý bude mať za úlohu spresňovať predikcie enkóderu. Úloha bola riešená pomocou dekóderov, ktoré predikcie enkóderu maskujú alebo čiastočne potláčajú informáciu kvôli využitiu informácie o nemaskovaných symboloch resp. využitiu informácie vstupnej sekvencie. Následne bola vykonaná séria experimentov, kde najlepší model dosiahol znakovej chybovosti 8.92 %. Zadanie ale nebolo splnené, pretože samotný enkóder dosiahol 6.38 %.

Abstract

The aim of this work is to build a model for handwritten text recognition, which will use non-autoregressive decoder. This type of decoder calculates character predictions independently of other predicted characters, which can be advantageous in terms of inference speed, but the quality of the prediction is worse. The motivation is to design a non-autoregressive decoder, which will have the task of refining the encoder's predictions. The task was solved with the help of decoders, which mask the encoder's predictions or partially suppress the information due to the use of information about unmasked symbols or using input sequence information. Subsequently, a series of experiments was performed, where the best model reached a character error rate of 8.92 %. But the assignment was not fulfilled, because the encoder itself reached 6.38 %.

Kľúčové slová

non-autoregresívne dekódery, rekurentné neurónové siete, Transformer, rozpoznávanie ručne písaného textu

Keywords

non-autoregressive decoders, recurrent neural networks, Transformer, handwritten text recognition

Citácia

KAVULIAK, Daniel. *Rozpoznávání textu pomocí hlubokých neuronových sítí.* Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Kišš

Rozpoznávání textu pomocí hlubokých neuronových sítí

Prehlásenie

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

.....
Daniel Kavuliak
17. mája 2023

Poďakovanie

Chcel by som sa poďakovať pánovi Ing. Martinovi Kiššovi za cenné rady a odbornú pomoc, ktoré mi dopomohli dokončiť túto prácu.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Rozpoznávanie ručne písaného textu je aktívna výskumná oblasť, ktorá kombinuje myšlienky oblastí počítačového videnia a spracovania prirodzeného jazyka, ktoré spadajú pod kategóriu strojového učenia. Narozdiel od rozpoznávania scénického textu alebo akéhokoľvek textu tlačeného strojom, kde výzvou modelu strojového učenia je rozlišovanie typov písma textu a rozlišovanie pozadia od samotného textu, výzvy rozpoznávania ručne písaného textu zahŕňa odlišovanie rôznych štýlov písma autorov písaných textov. Rozoznávanie spomenutých štýlov písma môže byť výskumne zložité, pretože nie sú kategorizovateľné do väčších celkov (v prípade textu tlačeného strojom, jednotlivé štýly môžu spadať do väčších kategórií ako napríklad *Times New Roman*, *Calibri* atď.) a každý jeden štýl má jedinečné charakteristiky. Z tohto vyplýva, že medzi štýlmi ručne písaného textu je vysoká variabilita. V tejto oblasti je výzvou aj samotný jazyk textu a aj kvalita snímku (presnejšie degradácia dokumentu na ktorom je napísaný text), pretože model na rozpoznanie ručne písaného textu sa musí naučiť modelovať vzťahy medzi jednotlivými znakmi (alebo slovami) a zároveň musí sa naučiť rozpoznávať jednotlivé znaky na základe ich vizuálnych čŕt.

Za najmodernejšie prístupy patriace do strojového alebo hlbokého učenia považujeme modely, ktoré sa skladajú konvolučných neurónových sietí a rekurentných neurónových sietí, ktoré zakódujú vizuálne črty vstupného obrázka a aj jeho kontext (závislosti medzi črtami). Tieto siete majú na konci *Connectionist Temporal Classification* (v skratke CTC) funkciu, ktorá dokáže priamo mapovať vstupnú sekvenciu na znaky výstupnej sekvencie, čím sa eliminuje potreba segmentácie obrázku na časti, ktoré by obsahovali črty znakov [10]. Ako ďalšie najmodernejšie prístupy považujeme modely, ktoré majú enkóder-dekóder architektúru. Enkóder obsahuje CNN a RNN sieť a má rovnaký účel ako spomenuté siete používajúce CTC funkciu. Dekóder je zvyčajne RNN sieť, ktorá v každom časovom kroku má na vstupe okrem výstupu enkódera aj klasifikáciu znaku z predchádzajúceho časového kroku. Takýmto dekóderom hovoríme, že sú *autoregresívne*.

Cieľom práce je zostrojiť model, ktorý používa *non-autoregresívny* dekóder, čiže dekóder nepoužívajúci klasifikácie znakov z predchádzajúcich časových krokov pri klasifikácii súčasného znaku. Motiváciou tohto cieľa je spresnenie CTC predikcií enkódera, čiže korekcia nesprávnych predikcií, ktoré boli zanesené enkóderom. Na splnenie tohto cieľa bolo potreba naimplementovať existujúce prístupy a naše vlastné varianty, následne vykonať experimenty a vyhodnotiť ich pomocou daných metrík.

Kapitola 2

Rozpoznávanie ručne písaného textu

V tejto kapitole je v sekcii 2.1 popísaná myšlienka rekurentných sietí a následovne je vysvetlená aj ich konkrétna varianta *Long Short-Term Memory* siete, ktoré sú súčasťou autoregresívnych modelov, ktoré sú zmienené v sekcii 2.4. Následne bude v sekcii 2.2 vysvetlený model Transformer, ktorý je súčasťou non-autoregresívnych dekóderov zmienených v sekcii 2.5. V sekcii 2.3 bude opísané, že čo je časové zaradenie a samotný princíp CTC. V sekcii 2.4 sú popísané autoregresívne modely pre rozpoznávanie ručne písaného textu a aj ich rozdiely medzi nimi. V poslednom rade, v sekcii 2.5 je popísaný rozdiel medzi autoregresívnymi a non-autoregresívnymi modelmi a dva modely používajúce non-autoregresívne dekódery.

2.1 Rekurentné neurónové siete

Narozdiel od konvolučných neurónových sietí, ktoré spracúvajú mriežky hodnôt ako sú obrázky, rekurentné neurónové siete spracúvajú dáta, ktoré sú usporiadané do sekvencie. Takéto siete sa dokážu škálovať do potrebnej veľkosti, aby boli schopné spracovať vstupné dáta rôznych dĺžok a tiež aby v jednotlivých časových krokoch boli schopné spracovať vektorové reprezentácie častí vstupných dát. Autori knihy *Goodfellow a spol.* [9] hovoria, že na to aby bola zostrojená rekurentná neurónová sieť, musí sa využiť myšlienka prebratá z osemdesiatych rokov pre modely strojového učenia a štatistických modelov: myšlienka zdieľania parametrov medzi rôznymi časťami neurónovej siete. Táto myšlienka hovorí, že zdieľanie parametrov umožňuje pracovať s dátami rôznych foriem, veľkostí a umožňuje generalizovať ich reprezentáciu. Keby rekurentná neurónová sieť nemala zdieľané parametre medzi rôznymi jej časťami, tak by sa nevedela použiť pre vstupné dáta variabilnej veľkosti. Takáto neurónová sieť by musela mať na vstupe dáta fixnej veľkosti, čo by bolo nepraktické vzhľadom na dáta, ktoré môžu mať rôznu veľkosť (napr. texty). Rekurentná neurónová sieť nezdieľajúca parametre medzi jej rôznymi časťami by sa taktiež nevedela použiť pre rôzne typy dát, čo by obmedzovalo jej použiteľnosť cez rôzne oblasti počítačového videnia a spracovania prirodzeného jazyka. V poslednom rade, takáto rekurentná neurónová sieť by nevedela správne generalizovať reprezentácie jednotlivých častí dát. V jednotlivých časových krokoch by sa naučila reprezentácie častí vstupných dát, ktoré sa najčastejšie nachádzajú v týchto časových krokoch (príkladom môžu byť zámená ja, ty, my atď.). Takéto zdieľanie parametrov je hlavne dôležité, ak konkrétna informácia sa môže vyskytovať na rôznych pozíciach v sekvencii. Ako príklad, autori *Goodfellow a spol.* [9] použili vety “Išiel som do

Nepálu v roku 2009" a "V roku 2009 som išiel do Nepálu". Ak by sme chceli od modelu, aby spracoval tieto dve vety a extrahoval rok návštevy štátu Nepál, tak by túto informáciu musel vedieť rozoznať aj napriek tomu, že táto informácia sa nachádza v treťom alebo posledom poradí vety. Plne prepojená dopredná neurónová sieť, ktorá by bola trénovaná na dátach fixnej veľkosti, by mala mať oddelené parametre pre každú časť vstupných dát, aby sa naučila jazykové pravidlá pre každú pozíciu vstupnej sekvencie. Rekurentná neurónová sieť by daný problém riešila spomínaným zdieľaním parametrov cez všetky časové kroky.

Aby myšlienka rekurentných neurónových sietí bola dobre zrozumiteľná, tak si predstavíme príklad použitia operácie konvolúcie na jednorozmernú časovú postupnosť [9]. Táto operácia umožňuje sieti zdieľanie parametrov v čase, ale iba pre susedné prvky. Výstup operácie konvolúcie je sekvencia, kde každý jeden prvok je vypočítaný z malého množstva susedných prvkov vstupnej sekvencie. Spôsob akým sa parametre siete zdieľajú spočíva v tom, že jej filter sa pomocou operácie konvolúcie aplikuje v každom časovom kroku. Pri rekurentných neurónových sieťach to ale funguje inak. Každý prvok výstupnej sekvencie je funkciou prvkov z predchádzajúcich časových krokov. Takáto rekurentná formulácia umožňuje zdieľanie parametrov siete cez hlboký výpočtový graf.

Vyššie spomenutú myšlienku zdieľania parametrov cez rekurentnú neurónovú sieť si môžeme formalizovať následovne [9]:

$$h_t = f(h_{t-1}, x_t, \theta) \tag{2.1}$$

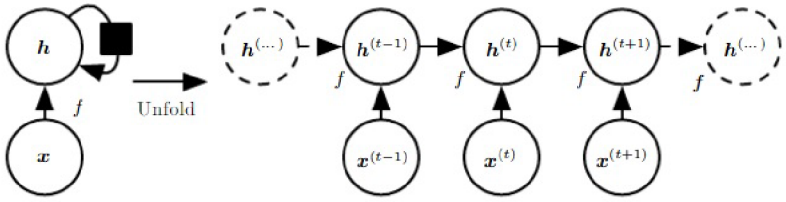
kde h_t je skrytý stav siete v časovom kroku t, x_t je časť vstupnej sekvencie z času t a θ sú váhy siete. Výpočet skrytého stavu v danom časovom kroku môžeme formalizovať aj ako rozvinutú rekurenciu [9]:

$$h_t = g_t(x_t, x_{t-1}, ..., x_2, x_1) \tag{2.2}$$

kde funkcia g_t zoberie sekvenciu $(x_t, x_{t-1}, ..., x_2, x_1)$ ako vstup a vypočíta z nej skrytý stav v čase t. Rozvinutá rekurencia umožňuje faktorizovať túto funkciu ako opakovanú aplikáciu funkcie f, čo nám umožňuje dve výhody [9]:

- Naučený model má stále rovnakú veľkosť vstupu v každom časovom kroku nezávisle od dĺžky sekvencie, pretože je špecifikovaný z hľadiska prechodu z jedného stavu do druhého, a nie z hľadiska histórie stavov s premenlivou dĺžkou.
- Je možné použiť rovnakú prechodovú funkciu f s rovnakými parametrami v každom časovom kroku.

Formalizácia myšlienky 2.1 je ilustrovaná na obrázku 2.1.



Obr. 2.1: Rekurentná neurónová sieť bez výstupov, ktorá spracováva vstupné dáta a zahŕňa ich do výpočtu skrytého stavu, ktorý je propagovaný v čase. Čierny štvorec na ľavej strane reprezentuje časové oneskorenie medzi časovými krokmi. Na pravej strane sa nachádza rozvinutá sieť. Prevzaté z [9]

Long short-term memory

Motiváciou architektúry siete *Long short-term memory* (ďalej LSTM) spočíva v tom, že autori *Hochreiter a spol.* chceli vyriešiť problém strácajúceho alebo explodujúce gradientu počas spätnej propagácie v čase alebo rekurentného učenia v reálnom čase [14]. Tieto dva problémy budú predstavené v následujúcich riadkoch. Problém strácajúceho gradientu spočíva v tom, že ak neurónová sieť má viac vrstiev, tak gradient chybovej funkcie, ktorý je spätne propagovaný, tak jeho hodnota postupne bude blížiť k nule. Dôvodom takéhoto javu sú aktivačné funkcie, napríklad aktivačná funkcia sigmoid zmršťuje interval hodnôt vstupných dát do intervalu $(0, 1)$. Z tohto faktu vyplýva, že ak zmeníme hodnoty vstupných dát vo veľkom rozsahu, tak sa prejavia malé zmeny vo výstupe. V našom prípade prvotné rekurentné neurónové siete používajú hyperbolický tangens ako aktivačnú funkciu na výpočet skrytého stavu v čase t a jeho obor hodnôt je v intervale $(-1, 1)$. Sú dané následujúce dva príklady vstupných sekvencií "Jana vstúpila do izby. Ján vstúpil tiež. Jana pozdravila ..." a "Jana vstúpila do izby. Ján vstúpil tiež. V ten deň bolo už neskoro a všetci sa vracali z práce. Jana pozdravila ...". V obidvoch sekvenciach z kontextu vyplýva, že na prázdne miesto patrí slovo "Jána". Je dôležité, aby RNN toto slovo predikovalo, keďže táto osoba sa vyskytuje skôr v danej sekvencii a ideálne malo by to správne predikovať. Avšak, v praxi sa stane to, že je pravdepodobnejšie uhádnutie slova v prvej sekvencii ako v druhej. Dôvodom je graduálne miznutie gradientu v skorších časových krokoch. Explodujúce gradienty sú problém, kde sa akumulujú obrovské gradienty, čo vedie k veľkým zmenám váh trénovaného modelu. Následok je nestabilný model počas trénovania.

Autori *Hochreiter a spol.* predstavili architektúru LSTM, ktorá sa skladá z následujúcich častí [14]:

- *input gate* - časť siete, v ktorej sa rozhoduje, ktoré konkrétne hodnoty vstupu sú dôležité pre predikciu v časovom kroku
- *output gate* - časť siete, kde sa počíta výstup LSTM siete v danom časovom kroku

Autori *Gers a spol.* pridali do tejto siete tzv. *forget gate* [7]. Účelom tejto časti je vymazanie zbytočných častí zo stavu bunky (ang. *cell state*), čiže rozhoduje o tom, ktoré hodnoty z nej budú ponechané a ktoré budú vymazané. Tieto všetky časti LSTM siete sa dajú formalizovať následovne [24]:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2.3}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.4}$$

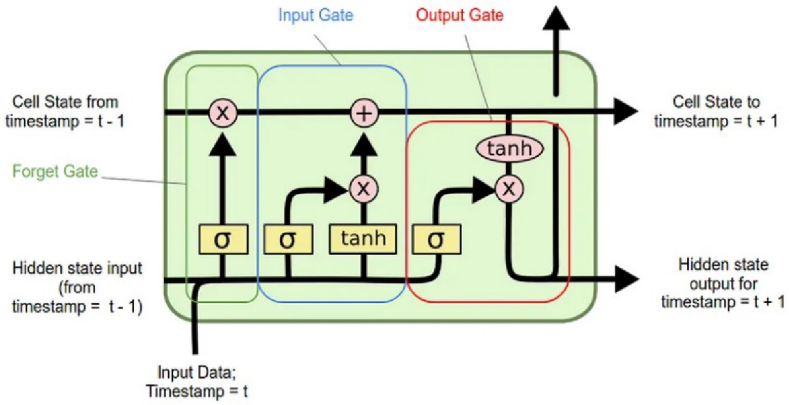
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{2.5}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{2.6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.8}$$

kde W sú zdieľané parametre častí LSTM v čase, b sú biasy častí LSTM, i je *input gate*, f je *forget gate*, o je *output gate*, h je skrytý stav LSTM siete, c je *cell gate*, g je *cell gate*, σ je sigmoid funkcia a $\odot$ je Hadamardov produkt. Rozdiel medzi *cell state* a skrytým stavom (ang. *hidden state*) spočíva v tom, že v skrytom stave v danom časovom kroku je zakódovaná informácia o danom výstupe, zatiaľ čo *cell state* má zakódovanú informáciu o všetkých vypočítaných výstupoch. Na obrázku 2.2 môžeme vidieť ilustráciu LSTM bunky.



Obr. 2.2: LSTM bunka v časovom kroku t.

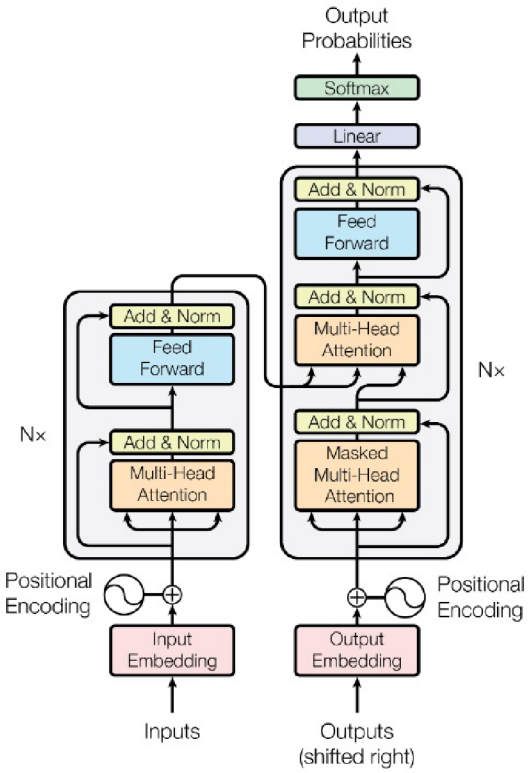
2.2 Transformer

V predchádzajúcej sekcii bola vysvetlená myšlienka rekurentných neurónových sietí a bol aj formalizovaný princíp fungovania, na ktorom podobne fungujú aj architektúry ako sú napríklad LSTM siete a GRU (ang. *Gated Recurrent Unit*) siete. V tejto sekcii bude predstavená sieť s názvom Transformer, ktorá sa zbavuje všetkých rekurentných častí a aj konvolúcií, čím zrýchľuje výpočet výstupu pre danú úlohu [29]. Tento model sa prevažne skladá z tzv. *attention* sietí, ktoré vypočítajú váhy pre každý jeden prvok vstupnej sekvencie, ktorý je reprezentovaný vektorom číselných hodnôt [29]. Tieto váhy reprezentujú dôležitosť hodnôt v danom prvku, čím sa zachovávajú hodnoty vektoru, ktoré sú charakteristické pre daný prvok. V následujúcej podsekcii bude podrobnejšie popísaná architektúra modelu Transformer.

Architektúra modelu

Autori *Vaswani a spol.* navrhli model Transformer, ktorý má architektúru typu enkóder-dekóder [29]. Motiváciou tohto rozhodnutia je to, že väčšina modelov v oblasti sekvenčného prekladu dosahujú najlepšie výsledky a má tento typ architektúry. V modeli Transformer, enkóder mapuje sekvenciu reprezentácii symbolov $(x_1, ..., x_n)$ na sekvenciu spojitých reprezentácií $(z_1, ..., z_n)$, pomocou ktorých dekóder následne generuje výstupnú sekvenciu $(y_1, ..., y_n)$ [29]. Dekóder v modeli Transformer je autoregresívny, prvok výslednej sekvencie vygeneruje na základe doterajších vygenerovaných prvkov. Na obrázku 2.3 je znázornená ilustrácia architektúry modelu Transformer.

Enkóder sa skladá z N rovnakých vrstiev. Každá takáto vrstva obsahuje dve podvrstvy: *Multi-Head Attention* sieť a doprednú plne prepojenú sieť. Za každou podvrstvou sa nachádza normalizačný blok, ktorý normalizuje hodnoty v každej sekvencii, ktoré sú v dávke. V tomto normalizačnom bloku sa sčítavajú hodnoty vstupných a výstupných sekvencií, čím sa vytvárajú reziduálne prepojenia okolo podvrstiev. Tesne predtým ako vstupné sekvencie vstupujú do enkódera, sú pre nich vypočítané pozičné kódovania. Dekóder má skoro rovnakú štruktúru, obsahuje naviac podvrstvu s maskovanou *Multi-Head Attention* sieťou. Na konci za dekóderom sa nachádza lineárna vrstva a vrstva *softmax*. Lineárna vrstva produkuje surové hodnoty z výstupu dekódera a vrstva s funkciou *softmax* počíta pravdepodobnosti výskytu symbolov.



Obr. 2.3: Architektúra modelu Transformer. Prevzaté z [29].

Attention

Autori *Vaswani a spol.* navrhli pre model Transformer *Scaled Dot-Product Attention* funkciu. Jej definícia je následovná [29]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.9}$$

kde Q je matica obsahujúca *query* vektory (dopyty), K je matica obsahujúca *key* vektory (kľúče), V je matica obsahujúca *value* vektory (hodnoty). Vektory v Q a K maticiach majú rovnakú veľkosť dimenzie d_k a vektory v matici V majú veľkosť dimenzie d_v. Na začiatku sa vykoná skalárny súčin medzi maticami Q a K. Následne sa vykoná podiel tohto skalárneho súčinu s hodnotou $\sqrt{d_k}$, čím sa naškálujú hodnoty výsledku skalárneho súčinu. Následne sa na tento výsledok aplikuje funkcia *softmax*, čím hodnoty podielu dostaneme do intervalu $< 0, 1 >$ a týmto získame váhy ku každej hodnote matici V.

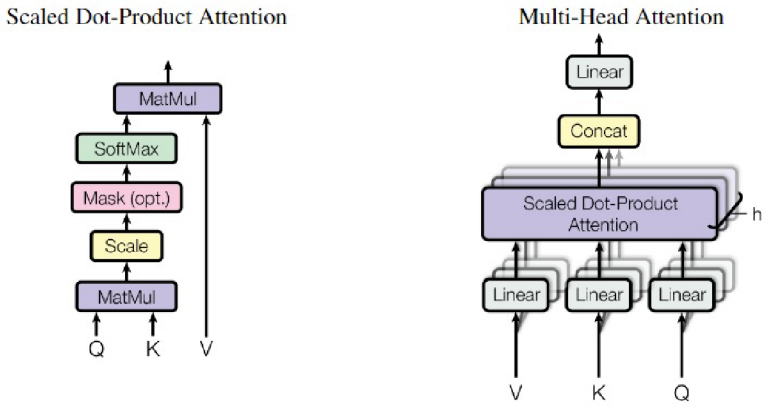
Ešte predtým ako autori navrhli túto funkciu, všeobecne bola používaná aditívna *attention* funkcia alebo multiplikatívna *attention* funkcia [29]. Multiplikatívna funkcia je skoro identická ako ich funkcia a jediný rozdiel je v tom, že po skalárnom súčine dopytov a kľúčov sa vykoná podiel s hodnotou $\sqrt{d_k}$. Motiváciou tohto rozšírenia bolo to, že pre veľké hodnoty d_k, skalárne súčiny rástli do veľkosti, čím sa tieto hodnoty vo funkcii softmax mapovali do takých intervalov, kde má extrémne malé gradienty [29].

Autori *Vaswani a spol.* zistili, že je viac výhodnejšie vykonať lineárne zobrazenie dopytov, kľúčov a hodnôt, ktoré majú veľkosť d_{model}, na naučené reprezentácie dopytov, kľúčov a hodnôt, ktoré majú veľkosti d_k, d_k a d_v [29]. Takéto zobrazenie sa vykoná h-krát. Následne

na každé zobrazenie sa paralélne vykoná *attention* funkcia, ktorej výstupom sú vektory vo veľkosti d_v a potom tieto vektory sa konkatenujú do jedného vektoru a aplikuje sa naň lineárne zobrazenie. Takýto mechanizmus autori nazvali *Multi-Head Attention*. Popísanie tohto vylepšenia sa dá formalizovať následovne [29]:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.10}$$

a kde $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ a projekcie sú matice váh $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_k}$ a $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. Na obrázku 2.4 sú ilustrované oba mechanizmy.



Obr. 2.4: Vľavo je ilustrácia *Scaled Dot-Product Attention*. Napravo je ilustrácia *Multi-Head Attention*. Prevzaté z [29].

Boli tri dôvody prečo sa autori rozhodli použiť resp. zostrojiť *attention* funkciu. Prvý dôvod bol, že autori chceli zmenšiť cestu medzi dlhodobými závislosťami (ang. *long-range dependencies*) [29]. Jeden z faktorov, ktoré ovplyňujú naučenie sa týchto dlhodobých závislostí, sú dĺžky ciest dopredných a spätných signálov, ktorými musia prechádzať. Čím sú tieto cesty medzi rôznymi kombináciami pozícií vo vstupných a výstupných sekvencií kratšie, tým je pre model ľahšie sa naučiť dlhodobé závislosti. Ostatné dva dôvody sú celková výpočtová zložitosť za vrstvu a množstvo paralelizovateľných výpočtov, ktoré je odmerané podľa minimálneho počtu potrebných sekvenčných operácií.

2.3 Connectionist Temporal Classification

Connectionist Temporal Classification (v skratke CTC) je metóda na trénovanie seq2seq modelov, ktoré nesegmentujú vstupné sekvencie dát. Motiváciou tejto metódy je vynechanie spomínanej segmentácie vstupnej sekvencie. Ešte pred CTC, rekurentné neurónové siete nemohli byť priamo použité na sekvenčnú anotáciu. Štandardné neurónové siete majú separátne definované objektívne funkcie pre každý bod tréningovej sekvencie, čo znamená, že RNN dokáže byť natrénovaná len pre individuálnu klasifikáciu prvkov sekvencie. Z tohto vyplýva, že vstupná segmentácia musí prejsť segmentáciou pred vstupom do RNN a jej výstup musí byť vhodne spracovaný do konečnej sekvencie. V následujúcich riadkoch bude predstavené časové zaradenie (ang. *Temporal Classification*), princíp CTC a CTC dekódery.

Temporal Classification

Na to aby bolo vysvetlené časové zaradenie, tak bude v následujúcich riadkoch formálne definované podľa autorov *Graves a spol.* [10]. Majme množinu S, ktorá obsahuje niekoľko tréningových príkladov z fixnej distribúcie $D_{X\times Z}$. Vstupný priestor $X = (\mathbb{R}^m)^*$ je množina všetkých sekvencií m-rozmerných vektorov obsahujúcich reálne hodnoty. Cieľový priestor $Z = L^*$ je množina všetkých sekvencií anotácii, ktoré sú zložené z písmen pochádzajúcich z konečnej abecedy L. Každý príklad z množiny S je usporiadaná dvojica sekvencií $(\boldsymbol{x}, \boldsymbol{z})$. Cieľová sekvencia $\boldsymbol{z} = (z_1, ..., z_U)$ môže mať najviac rovnakú dĺžku ako vstupná sekvencia $\boldsymbol{x} = (x_1, ..., x_T)$, takže $U \leq T$. Keďže vstupné a cieľové sekvencie nemusia mať vo všeobecnosti rovnakú dĺžku, tak nie je apriórny spôsob ich zarovnania. Cieľom je použiť množinu S na trénovanie modelu, ktorý bude klasifikovať predtým nevidené sekvencie takým spôsobom, aby sa minimalizovala metrika chybovosti určená na danú úlohu.

Connectionist Temporal Classification

Neurónové siete, ktoré používajú CTC, majú na konci lineárnu vrstvu. Táto lineárna vrstva počíta surové hodnoty pre každý symbol z abecedy L a *blank* symbol, ktorý reprezentuje klasifikáciu prvku vstupnej sekvencie do skupiny, kde sa nevyskytuje symbol z abecedy L. Následne tieto surové hodnoty budú vstupom do funkcie *softmax*, ktorá vypočíta pre každý znak z abecedy L a *blank* symbol pravdepodobnosti ich výskytu. Z tohto vyplýva, že výstup siete obsahuje pravdepodobnosti všetkých zarovnaní všetkých možných cieľových sekvencií na vstupné sekvencie [10].

Autori formálne definovali CTC následujúcim spôsobom [10]. Pre vstupnú sekvenciu $\mathbf{x}$ dĺžky T majme definovanú rekurentnú neurónovú sieť s m vstupmi, n výstupmi a vektor váh w: $N_w : (\mathbb{R}^m)^T \mapsto (\mathbb{R}^n)^T$. Nech $\mathbf{y} = N_w(\mathbf{x})$ je definované ako výstupné sekvencie siete a označme y_k^t ako aktiváciu výstupného neurónu k v čase t. y_k^t je teda interpretované ako pravdepodobnosť spozorovaného symbolu v čase t, ktorá definuje distribúciu množiny L'^T, ktorá obsahuje sekvencie dĺžok T a tie sú tvorené z množiny $L' = L \cup \{blank\}$:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T \tag{2.11}$$

Prvky množiny L'^T budú nazývané ako *cesty* a budú označené symbolom π. Zo vzorca 2.11 sa dá predpokladať, že výstupy z rôznych časov sú podmienene nezávislé vzhľadom na vnútorný stav siete [10]. Ako ďalšie bude definovaná funkcia, ktorá má následujúci predpis: $B : L'^T \mapsto L^{\leq T}$, kde $L^{\leq T}$ je množina možných výstupných sekvencií zložených z abecedy L,

ktoré majú veľkosť menšiu alebo rovnakú ako hodnota T. V tejto funkcii sa budú odstraňovať susedné duplicitné symboly a *blank* symboly z ciest. V poslednom rade je definovaná podmienená pravdepodobnosť danej výstupnej sekvencie $\mathbf{l} \in L^{\leq T}$ ako suma pravdepodobností všetkých zodpovedajúcich ciest [10]:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in B^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) \tag{2.12}$$

CTC dekóder

S danou formalizovanou definíciou môže byť formálne zadefinovaná najpravdepodobnejšiu výstupnú sekvenciu pre daný vstup [10]:

$$h(x) = arg \max_{\mathbf{l} \in L^{\leq T}} p(\mathbf{l}|\mathbf{x}) \tag{2.13}$$

Prvú metódu, ktorú autori *Graves a spol.* definovali je metóda dekódovania najlepšej cesty (ang. *best path decoding*), ktorá len berie najväčšie pravdepodobnosti v každom časovom kroku [10]:

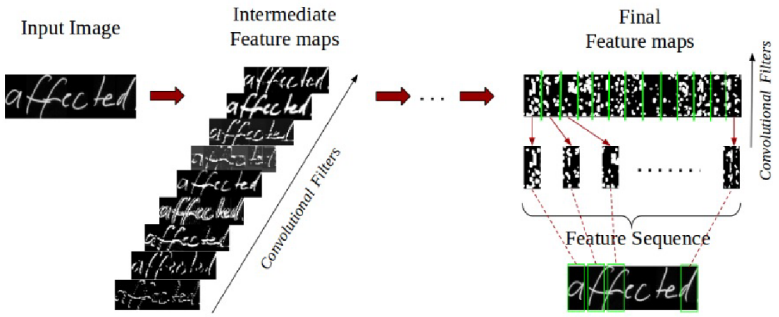
$$h(\mathbf{x}) \approx B(\pi^*) \tag{2.14}$$

kde $\pi^* = arg \max_{\pi \in N^t} p(\pi|\mathbf{x})$. Táto metóda je jednoduchá na výpočet, ale není garantované, že nájde správne riešenie. Druhá metóda sa nazýva *prefix search decoding* a tá garantuje nájdenie správneho riešenia [10].

2.4 Seq2seq modely

V dnešnej dobe sa používajú *seq2seq* modely na rozpoznanie ručne písaných slov, ale aj v iných podobných oblastiach, kde sa pracuje s dátami, ktoré môžeme vnímať ako usporiadané vstupné sekvencie napr. rozpoznanie scénického textu alebo rozpoznanie reči. Tieto modely zvyčajne majú typ architektúry enkóder-dekóder, kde účel enkóderu je zakódovať vizuálne črty vstupného obrázka do ich vhodnej vektorovej reprezentácie a účelom dekóderu je rozkódovať výstup enkóderu pomocou predchádzajúcich klasifikácií znakov [20, 28, 15, 4, 2, 5, 16]. V predchádzajúcich sekciách boli popísané rekurentné siete, model Transformer a CTC, ktoré sú súčasťou zmienených modelov v tejto sekcii. V tejto sekcií budú vo všeobecnosti popísané enkóder-dekóder modely a rozdiely medzi uvedenými modelmi.

Tento prístup sa zvyčajne používa s konvolučnou neurónovou sieťou, ktorá filtruje hodnoty vstupného obrázka a extrahuje z neho najdôležitejšie črty charakteristické pre obrázok a obojsmernou rekurentnou sieťou, ktorá kóduje tieto črty do naučenej reprezentácie [4, 20, 28, 16]. Tieto dve časti sú obsahom enkódera. Autori *Chowdhury a spol.* naviac majú vrstvu *Map-To-Sequence*, ktorá výstup z CNN konvertuje na sekvenciu čŕt [4]. Tieto sekvencie čŕt vznikajú odstraňovaním stĺpcov výstupu CNN po hĺbke. Lepšie povedané, každý i-tý *feature* vektor bol konštruovaný konkatenáciou i-tých stĺpcov z *feature* máp. Kvôli translačnej invariancii konvolučných operácií, každý stĺpec reprezentuje vertikálny pás na obrázku, ktorý sa pohybuje zľava doprava. Tieto vertikálne pásy sa nazývajú receptívne polia. Ilustrácia generovania rysových vektorov a možné receptívne polia je na obrázku 2.5. Autori *Sueiras a spol.* robia predspracovanie obrázku ešte pred CNN vrstvou, kde obrázok rozdelia do fixného počtu menších obrázkov [28]. V tomto pripade CNN vrstva nie je len jedna CNN sieť, ale skladá sa z fixného počtu konvolučných neurónových sietí a hneď za nimi následujú plne prepojené siete. Zvyčajne enkóder obsahuje obojsmernú LSTM, ktorá vstupný



Obr. 2.5: Znázornenie generácie sekvencie čŕt a možných receptívnych polí. Rysové mapy sú uložené po hĺbke kvôli konvolučným filtrom, ktoré ich generovali. Finálne rysové mapy sú uložené po šírke. Prevzaté z [4].

obrázok zakóduje sekvenciu konvolučných čŕt a pokúsi sa zakódovať časový kontext medzi nimi [20]. Táto sieť sa používa kvôli vyššie spomenutému problému miznúceho gradientu, ktorý je spôsobený opakovaným násobením gradientov v rozvinutej rekurentnej neurónovej sieti a zároveň kvôli lepšej schopnosti modelovať a naučiť sa dlhodobé závislostiam medzi črtami [4]. Obojsmerná je kvôli tomu, aby sa využili dopredné aj spätné dlhodobé závislosti. Autori *Kang a spol.* využívajú obojsmernú *Gated Recurrent Unit* (v skratke GRU) sieť [16]. Autori *Michael a spol.* majú za obojsmernou LSTM sieťou jednorozmernú konvolučnú sieť, ktorá zároveň modeluje vizuálnu informáciu a časový kontext sekvencie [20]. Jej výstupom je konečná zakódovaná mapa čŕt.

Attention mechanizmy môžu byť vhodným rozšírením štandardných enkóder-dekóder modelov. Tieto funkcie dynamicky upravujú zakódované črty z enkódera v každom časovom kroku t pomocou podobnosti týchto čŕt so skrytým stavom dekódera s_t [1, 18]. Takto upravene zakódované črty sa nazývajú kontextové vektory. Vo všeobecnosti, kontextový vektor je vypočítaný ako váhovaný súčet všetkých prvkov zakódovaných čŕt. Formalizácia vypočítania kontextového vektoru je následovná:

$$c_t = \sum_{j=1}^{M} \alpha_{t,j} h_j \tag{2.15}$$

kde α sú váhy vypočítané pomocou *attention* funkcie, c je kontextový vektor, h je skrytý stav enkódera, M je dĺžka výstupnej sekvencie enkódera, t je časový krok dekódera a j je časový krok enkódera. Všeobecný výpočet *attention* váh je daný následovne:

$$\alpha_{t,j} = Att(s_t, h_j, \alpha_{t-1}) \tag{2.16}$$

kde *Att* predstavuje *attention* funkciu. Ako štandardy v enkóder-dekóder sieťach sa považujú funkcie od autorov *Bahdanau a spol.* a *Luong a spol.*. Tieto mechanizmy pri počítaní váh nepoužívajú váhy z predchádzajúcich časových krokov dekódera, čo z nich tvorí funkcie, ktoré sa zameriavajú len na obsah zakódovaných čŕt. Definícia *attention* mechanizmov od autorov *Bahdanau a spol.* a *Luong a spol.* je následovná [1, 18]:

$$e_{t,j} = v^T \tanh(W_s s_t + W_h h_j + b) \tag{2.17}$$

$$e_{t,j} = v^T W_h h_j \tag{2.18}$$

kde W_h, W_s, v a b sú trénovateľné parametre a e je skóre podobnosti. Váhy sa vypočítajú následovne: $\alpha_{t,j} = softmax(e_{t,j})$. Nevýhodou tohto prístupu spočíva v tom, podobné elementy výstupnej sekvencie enkódera budú mať rovnaké skóre podobnosti [20]. Autori *Kang a spol.* vo svojom modeli využili *attention* funkciu, ktorá berie do úvahy aj pozíciu elementov výstupu enkódera [16]. Táto funkcia je rozšírením definície 2.17 o váhy z predchádzajúceho časového kroku, ktoré dostaneme aplikáciou operácie konvolúcie. Najprv musia byť extrahované vektory $f_{t,j} \in \mathbb{R}^k$ pre každú pozíciu j váh α_{t-1} pomocou aplikácie operácie konvolúcie na maticu $F \in \mathbb{R}^{k \times r}$ [3]:

$$f_t = F * \alpha_{t-1} \tag{2.19}$$

Funkcia je teda definovaná následovne [3]:

$$e_{t,j} = v^T \tanh(W_s s_t + W_h h_j + U f_{t,j} + b) \tag{2.20}$$

kde U sú trénovateľné parametre.

Dekóder je zvyčajne jednosmerná LSTM sieť, ktorá na vstupe v každom časovom kroku má kontextové vektory vyprodukované *attention* funkciou a klasifikáciu kontextového vektora z predchádzajúceho časového kroku. Takýto dekóder teda modeluje distribúciu cieľovej sekvencie $\mathbf{y} = (y_1, ..., y_T)$ ako $p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | y_1, ..., y_{t-1}, c_t)$, kde T je dĺžka výstupnej sekvencie dekódera [20, 4, 16, 28]. Pravdepodobnosti jednotlivých prvkov v časových krokoch sa teda počítajú ako $softmax(f(y_{t-1}, s_{t-1}, c_t))$, kde f je LSTM bunka.

Existujú prístupy, ktoré nepoužívajú rekurentné neurónové siete, ale používajú model Transformer. Motiváciou použitia modelu Transformer je zrušenie rekurencií z dôvodu nájdenia minimálnych ciest medzi doprednými a spätnými dlhodobými závislosťami medzi črtami vstupného obrázka. Dôvod použitia je aj rýchlosť trénovania, tým že model Transformer obsahuje vrstvy *Multi-Head Attention*, tak výpočet je paralelizovaný. Tieto vrstvy taktiež sú schopné sa naučiť širší kontext pri konštantnom počte operácií [2]. Autori *Barrere a spol.* navrhli model používajúci časti modelu Transformer, kde sa snažili znížiť počet parametrov a dosiahli to svojou vlastnou konvolučnou sieťou v enkóderi [2]. Vďaka modelu Transformer dokáže byť trénovanie rýchlejšie, počas trénovania je možné využiť metódu *teacher forcing* [2]. Počas inferencie je stále nutné iteratívne klasifikovať ako pri vyššie spomenutých štandardných modeloch [2, 15].

2.5 Autoregresívne vs. Non-autoregresívne dekódery

V predchádzajúcej sekcii boli popísané všeobecné seq2seq modely, konkrétne modely s enkóder-dekóder architektúrou a rozdiely medzi vybranými modelmi. Z predchádzajúcej sekcie sa dá všimnúť, že spomenuté modely boli autoregresívne, mali autoregresívne dekódery, ktoré generovali prvok cieľovej sekvencie na základe prvkov v predchádzajúcich časových krokov. V tejto sekcii budú formálne definované všeobecné autoregresívne a non-autoregresívne dekódery a zároveň budú popísané výhody a nevýhody oboch prístupov. Potom budú popísané dva non-autoregresívne dekódery, ktoré boli navrhnuté pre rozpoznávanie rečových nahrávok.

Autoregresívne dekódery

Autori *Gu a spol.* [12] definovali úlohu strojového prekladu ako problém generovania sekvencie zo sekvencie následujúcim spôsobom. S danou vstupnou sekvenciou $X = \{x_1, x_2, ..., x_N\}$,

model na strojový preklad faktorizuje distribúciu nad výstupnou sekvenciou $Y = \{y_1, y_2, ..., y_T\}$ do reťazca podmienených pravdepodobností s kauzálnou štruktúrou zľava doprava:

$$p_{AR}(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:N};\theta) \tag{2.21}$$

kde y_0 je špeciálny symbol $[SOS]$, ktorý značí začiatok sekvencie, y_{T+1} je špeciálny symbol $[EOS]$, ktorý reprezentuje koniec sekvencie a θ reprezentuje parametre modelu. Táto faktorizácia umožnuje priamy tréning modelu s maximálnou vierohodnosťou s chybovou funkciou krížovej entropie aplikovanou v každom dekódovacom kroku [12]:

$$L_{ML} = \log p_{AR}(Y|X;\theta) = \sum_{t=1}^{T+1} \log p(y_t|y_{0:t-1}, x_{1:N};\theta) \tag{2.22}$$

Autori *Gu a spol.* [12] tvrdia, že autoregresívna faktorizácia vykonaná konvenčnými prístupmi má svoje výhody. Zodpovedá slovnému charakteru produkcie ľudského jazyka a efektívne zachytáva distribúciu skutočných prekladov. Autoregresívne modely dosahujú *state-of-the-art* výsledky na veľkorozmerných korpusoch a sú ľahko trénovateľné, pokým metóda *beam search* poskytuje efektívnu metódu lokálneho vyhľadávania na nájdenie optimálneho prekladu. Autori *Gu a spol.* [12] ale zároveň tvrdia, že má to aj svoje nevýhody. Každý krok dekódovania musí byť vykonaný sekvenčne, čo zabraňuje architektúram inšpirovanými modelom Transformer vo vykonávaní týchto krokov paralelne. Metóda vyhľadávania *beam search* trpí klesajúcou návratnosťou vzhľadom na veľkosť lúča a vykazuje limitované paralélne vyhľadávanie, pretože zavádza výpočtovú závislosť medzi lúčami.

Non-autoregresívne dekódery

Naivné riešenie ako z autoregresívneho dekóderu vytvoriť non-autoregresívny dekóder je priamo odstrániť autoregresívne prepojenie z existujúceho enkóder-dekóder modelu [12]. V predpoklade, že dĺžka cieľovej sekvencie T môže byť modelovaná so samostatnou podmienenou distribúciou p_L je faktorizacia následovná:

$$p_{NA}(Y|X;\theta) = p_L(T|x_{1:N};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_{1:N};\theta) \tag{2.23}$$

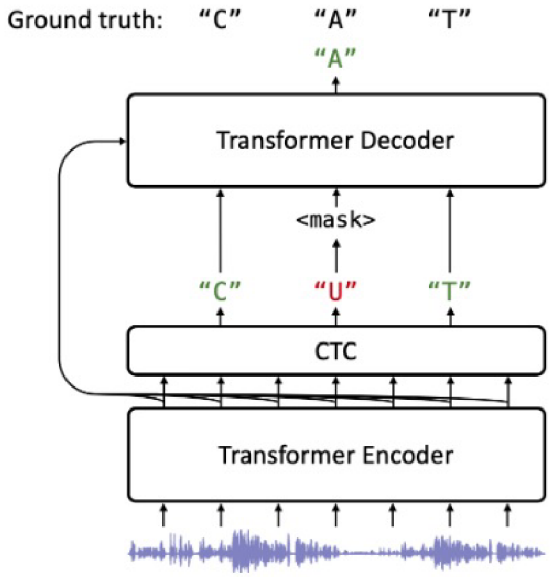
Zo vzorca 2.23 je možno vyčítať, že model stále má explicitne definovanú funkciu vierohodnosti a stále môže byť trénovaný s použitím nezávislých chybových funkcií krížovej entropie na každú výstupnú distribúciu. S týmto nastavením je možno výpočet výstupných distribúcií vykonať paralelne počas inferencie modelu.

Avšak autori *Gu a spol.* [12] tvrdia, že tento naivný prístup neprináša dobré výsledky, pretože takýto model vykazuje úplnú podmienenú nezávislosť. Distribúcia každého symbolu $p(y_t)$ závisí len na vstupnej sekvencii X. Tento fakt zo spomínanej distribúcii robí nedostatočnú aproximáciu k skutočnej cieľovej distribúcii, čo vykazuje silnú koreláciu medzi symbolmi v čase. Autori uviedli následovný príklad prekladu z anglického jazyka do nemeckého jazyka ako interpretáciu tohto problému [12]. Je daná anglická veta "Thank you". Táto veta môže byť preložená do nemeckého jazyka ako "Danke", "Danke schön", "Viele Dank" a všetky tieto preklady sa môžu nachádzať v tréningovom korpuse. Táto cieľová distribúcia nemôže byť reprezentovaná produktom nezávislých pravdepodobnostných distribúcií pre každý uvedený príklad, pretože podmienene nezávislá distribúcia nedovoľuje

výskytu príkladov “Danke schön” a “Viele Dank” bez toho, aby sa v distribúcii vyskytovali výrazy “Danke Dank” a “Vielen schön”. Z tohto vyplýva, že podmienená nezávislosť zabraňuje modelu zachytávať vysoko multimodálnej distribúcie cieľových prekladov. Autori *Gu a spol.* tento problém nazvali *problém multimodality* [12].

Mask CTC

Autori *Higuchi a spol.* [13] navrhli model *Mask CTC* inšpirovaný modelom Transformer za účelom vytvorenia non-autoregresívneho dekóderu. Vstupom do tohto dekóderu sú maskované CTC predikcie, ktoré boli vypočítané pomocou CTC dekódera využívajúceho *best path decoding* metódu. Symboly v tejto predikcii boli maskované na základe pravdepodobností výskytov symbolov, t. j. ak mal symbol pravdepodobnosť menšiu ako bol zvolený prah, tak sa nahradil špeciálnym symbolom $[MASK]$. Takto upravené CTC predikcie potom slúžili ako vstup do non-autoregresívneho dekódera a výstupom je sekvencia symbolov s novými symbolmi namiesto $[MASK]$ symbolov. Prehľad tohto modelu je znázornený na obrázku 2.6. *Mask CTC* model sa samotnou technikou maskovania inšpiroval od podmieneného mas-



Obr. 2.6: Prehľad modelu. Model je trénovaný so spoločnou objektívnou funkciou zloženou z CTC objektívnej funkcie a *mask-predict* objektívnej funkcie. Počas inferencie je sekvencia inicializovaná pomocou *best path decoding* funkcie a následne symboly s nízkou pravdepodobnosťou sú maskované. Tieto maskované symboly sú následne predikované non-autoregresívnym dekóderom na základe ostatných symbolov. Prevzaté z [13].

kovaného jazykového modelu (ang. *conditional masked language model*, v skratke CMLM) od autorov *Ghazvininejad a spol.* [8], ktorý predikuje maskované symboly v cieľovej sekvencii. Vďaka výhode paralelného výpočtu modelu Transformer, CMLM dokáže predikovať akúkoľvek ľubovoľnú podmnožinu maskovaných symbolov v cieľovej sekvencií pomocou ostatných symbolov z predchádzajúcich a následujúcich časových krokov. Toto tvrdenie je formalizované následovne [13]:

$$P_{cmlm}(Y_{mask}|Y_{obs}, X) = \prod_{y \in Y_{mask}} P_{cmlm}(y|Y_{obs}, X) \tag{2.24}$$

kde $Y_{obs} = Y - Y_{mask}$ a X je vstupná sekvencia. Počas trénovania, sa maskujú náhodné symboly sekvencie a ich počet je vybraný z uniformnej distribúcie [13]. Počas inferencie je výstupná sekvencia je postupne generovaná v konštantnom počte iterácií, kde sa opakovane maskujú a predikujú maskované symboly [13]. Autori *Higuchi a spol.* zistili, že trénovanie so spoločnou objektívnou funkciou zloženou z CTC objektívnej funkcie a objektívnej funkcie 2.24 explicitne poskytuje modelu absolútnu pozičnú informáciu, čo zvyšuje úspešnosť modelu. Jej formalizácia je následovná [13]:

$$L_{NAR} = \gamma \log P_{ctc}(Y|X) + (1-\gamma) \log P_{cmlm}(Y_{mask}|Y_{obs}, X) \tag{2.25}$$

kde γ je laditeľný parameter.

Non-autoregresívne dekódery musia poznať dĺžku výstupnej sekvencie, aby boli schopné paralélne vypočítať celú výstupnú sekvenciu. Na vyriešenie tohto problému, autori *Higuchi a spol.* [13] berú do úvahy CTC výstupy ako iniciálnu sekvenciu na dekódovanie. CTC výstupy sú vypočítané pomocou enkódera a úlohou dekódera je ich spresnenie na základe celej tejto sekvencie. Tieto CTC výstupy boli spracované pomocou algoritmu *best path decoding* zmieneným v sekcii 2.3 v podsekcii *CTC dekóder*, aby autori zachovávali non-autoregresívnu inferenciu. Autori definovali výpočet pravdepodobností agregovaných symbolov CTC dekóderom pomocou použitia CTC pravdepodobností, ktoré sú na úrovni rámcov následovne [13]:

$$\hat{P}(\hat{y}_l|X) = \max_j \left(P(a_j \in A_l|X)\right) \tag{2.26}$$

kde $A_l = \{a_j\}_j$ sú po sebe idúce rovnaké symboly, ktoré zodpovedajú agregovanému symbolu $\hat{y}_l$. Maskovaná podmnožina symbolov z množiny $\hat{Y}$ na základe ich pravdepodobností pomocou prahu P_{thres} je definovaná následovne [13]:

$$\hat{Y}_{mask} = \{y_l \in \hat{Y} | \hat{P}(y_l|X) < P_{thres}\} \tag{2.27}$$

V poslednom rade predikcia $\hat{Y}_{mask}$ je vypočítaná na základe spozorovaných symbolov $\hat{Y}_{obs} = \hat{Y} - \hat{Y}_{mask}$

Autori *Higuchi a spol.* [13] experimentovali s iteratívnou dekódovacou metódou *easy first*, v ktorej sú postupne vypočítané predikcie symbolov podľa pravdepodobnosti vypočítanej CMLM dekóderom. Aktualizácia agregovaného symbolu v n-tej iterácii je definovaná následovne:

$$\hat{y}_l = \arg\max_w P_{cmlm}(\hat{y}_l = w|Y_{obs}^{(n)}, X) \tag{2.28}$$

V každej iterácii prebehne výpočet predikcií pre C maskovaných symbolov s najvyššími pravdepodobnosťami, kde $C = L/K$. L je definovaná ako počet maskovaných symbolov a K je zvolený počet iterácií.

CTC Enhanced

Autori *Song a spol.* [27] navrhli model, ktorý bol taktiež inšpirovaný modelom Transformer. Tento model ale nemaskuje symboly CTC predikcií, generuje výstupné sekvencie spresňovaním predikcií CTC dekódera. Motiváciou tohto návrhu bolo, že väčšina nonautoregresívnych dekóderov majúce na vstupe sekvenciu s maskovanými symbolmi, nedokážu efektívne využiť kontext cieľovej sekvencie, čo vedie k degradácii presnosti modelu. Autori tvrdia, že takéto modely trpia dvoma nevýhodami [27]:

1. Spozorované symboly nie sú vždy správne a maskované symboly nemusia byť vždy zlé, čo môže viesť k ďalším nesprávne rozpoznaným znakom

2. Dekóder je viac citlivý na kontext cieľovej sekvencie ako na kontext vstupnej sekvencie. Dáta v úlohe rozpoznania reči sú viac striktné ako dáta v úlohe strojového prekladu vďaka rozličným charakteristikám medzi akustickými signálmi a lingvistickými symbolmi.

Druhý bod naznačuje, že ak je daný plný kontext z čŕt audia a plný kontext predikovaných znakov, tak maskované symboly sa nahradia iba s pomocou cieľového kontextu. Z tohto vyplýva, že ak sa budú používať anotácie počas trénovania a CTC predikcie počas inferencie, tak to ovplyvní úspešnosť trénovania.

Autori *Song a spol.* navrhli model, kde dekóder bude na generovanie cieľovej sekvencie brať do úvahy iba kontext v predchádzajúcich časových krokoch, čím nútia model brať do úvahy aj kontext vstupnej sekvencie X [27]:

$$y_t = D(\hat{y}_{1:t-1}, E(X)) \tag{2.29}$$

kde $\hat{y}$ je CTC predikcia, $D(\cdot)$ je dekóder a $E(\cdot)$ je enkóder. Počas trénovania sa buď využije anotácia alebo CTC predikcia. Autori *Song a spol.* [27] navrhli dve metódy na trénovanie modelu s dekóderom formálne zadefinovaným vo vzorci 2.29:

1. *Teacher forcing* – počas trénovania je na vstupe dekódera anotácia (prepis reči), ktorá patrí k príslušnej nahrávke a počas inferencie je na vstupe postupnosť CTC predikcií. Používa sa aj kauzálna maska pomocou ktorej sa zahodí kontext CTC predikcií alebo anotácie z budúcich časových krokov, aby sa do úvahy bral len kontext z predchádzajúcich časových krokov

2. *CTC Sampling* – počas trénovania je počítaná postupnosť CTC predikcií, aby bola známa jeho dĺžka T'. Ak $|T - T'| \leq 2$, kde T je dĺžka anotácie, tak použijeme na vstup dekódera postupnosť CTC predikcií, inak anotáciu.

2.6 Levenštajnová vzdialenosť

Všetky zmienené modely rozpoznávania rukou písaného textu a rozpoznávania reči boli vyhodnotené pomocou metrík *Character Error Rate* (v skratke CER) a *Word Error Rate* (v skratke WER). Tieto dve metriky sú odvodené od Levenštajnovej vzdialenosti, ktorú si teraz popíšeme a potom sa predstavia CER a WER.

Levenštajnová vzdialenosť (alebo editačná vzdialenosť) je metrika, ktorá meria rozdielnosť dvoch reťazcov pomocou transformovania jedného reťazca na druhý [25]. Reťazec sa transformuje pomocou aplikovaním postupnosti editačných operácií na jednotlivé znaky ako sú napr. vloženie, vymazanie a nahradenie. Editačná vzdialenosť medzi dvoma reťazcami je teda minimálny počet takýchto operácií, ktoré sú potrebné na pretvorenie jedného reťazca na druhý reťazec. Editačná vzdialenosť medzi dvoma definovanými reťazcami S_1 a S_2 sa dá vypočítať pomocou dynamického programovania a môže byť formálne opísaná pomocou následovnej rekurzie [25]:

$$D(i,j) = min[D(i-1,j) + w_d, D(i,j-1) + w_i, D(i-1,j-1) + w_r], \forall i,j > 0 \tag{2.30}$$

$$D(i,0) = D(i-1,0) + w_d, \forall i > 0 \tag{2.31}$$

$$D(0,j) = D(0,j-1) + w_i, \forall j > 0 \tag{2.32}$$

$$D(0,0) = 0 \tag{2.33}$$

kde i a j sú dĺžky reťazcov, w_i, w_d a w_r je počet operácie vloženia, vymazania a nahradenia. Hodnota D bude vo výsledku menšia, ak sa reťazce budú viac podobať.

CER a WER

CER metrika je vypočítaná ako vyššie popísaná editačná vzdialenosť, čo je počet minimálneho počtu operácií vloženia, vymazania a substitúcie potrebných na prevedenie z jedného reťazca na druhé. Táto editačná vzdialenosť je potom podelená celkovým počtom znakov v anotácii, čím dostaneme hodnotu CER. Formalizácia je následovná [16]:

$$CER = \frac{S + I + D}{N} \tag{2.34}$$

kde S je počet substitúcií, I je počet vložení, D je počet vymazaní a N je celkový počet znakov v anotácii.

WER metrika je výpočtovo podobná ako CER metrika. WER hodnota je počítaná ako minimálny počet vložení, vymazaní a substitúcií slov v reťazci a následne táto hodnota je podelená celkovým počtom slov v anotácii. Tento výpočet je formalizovaný následovne [16]:

$$WER = \frac{S_w + I_w + D_w}{N_w} \tag{2.35}$$

kde S_w, I_w a D_w sú počty vložení, vymazaní a substitúcií slov v reťazci a N_w je celkový počet slov v reťazci.

Kapitola 3

Návrh riešenia

V kapitole 2 v sekcii 2.4 boli popísané enkóder-dekóder modely, ktoré boli navrhnuté na rozpoznanie ručne písaných textov. Tieto modely používali autoregresívne dekódery, ktoré generovali prvok cieľovej sekvencie na základe vygenerovaných prvkov v predchádzajúcich časových prvkov. Tieto prístupy zvyčajne používajú v enkóderi a dekóderi rekurentné neurónové siete. Tieto siete vedia modelovať dlhodobé závislosti resp. časový kontext sekvencie a zároveň vedia pracovať so vstupmi s premenlivou dĺžkou vďaka zdieľaným parametrom v čase. Táto myšlienka bola popísaná v kapitole 2 v sekcii 2.1. Toto je dôležité, pretože na to, aby model vedel korektne zakódovať vstupný obrázok resp. generovať výstupnú sekvenciu, musí poznať jazykové pravidlá daného jazyka. Súčasťou týchto enkóder-dekóder modelov používajúce rekurentné siete sú aj *attention* siete. Tieto siete počítajú váhy pre prvky vstupnej sekvencie a reprezentujú dôležitosť hodnôt v prvkoch. Vďaka týmto váham, dekóder bude brať do výpočtu iba hodnoty prvkov vstupnej sekvencie, ktoré sú pre ne najviac charakteristické. Ako bolo popísané v sekcii 2.4, štandardné *attention* mechanizmy do výpočtu váh zahŕňajú skrytý stav dekódera a počítajú skóre podobnosti s výstupom enkódera. Novšie prístupy enkóder-dekóder modelov v rozpoznávaní ručne písaného textu sú inšpirované modelom Transformer, ktorý bol popísaný v kapitole 2 v sekcii 2.2. Motivácia tejto inšpirácie bola v tom, že zrušením rekurencie sa model naučí minimálnu cestu medzi dlhodobými závislosťami medzi prvkami vstupnej sekvencie ako bolo popísané v sekcii 2.2. Tréning modelu Transformer určeného na rozpoznanie ručne písaného textu môže byť rýchly vďaka metóde *teacher forcing*, ale inferencia je rovnaká ako pri modeloch používajúce rekurentnú neurónovú sieť.

V kapitole 2 v sekcii 2.5 boli formalizované autoregresívne a non-autoregresívne modely, aby bolo vidieť rozdiely medzi nimi. Autoregresívne dekódery majú svoje výhody aj nevýhody. Medzi výhodami patrí napr. fakt, že autoregresívna faktorizácia zodpovedá slovnému charakteru produkcie a efektívne zachytáva distribúciu skutočných prekladov. Nevýhodou autoregresie je, že každý prvok je generovaný sekvenčne na základe predchádzajúcich prvkov v predchádzajúcich časových krokoch. Non-autoregresívne dekódery majú výhodu v tom, že odstránením autoregresívnych prepojení je generovanie prvkov cieľovej sekvencie paralélne. Nevýhodou ale je, že takýto model trpí problémom multimodality. V posledom rade boli popísané dva non-autoregresívne modely. Prvý model bol inšpirovaný modelom Transformer a do jeho dekódera vstupuje sekvencia vygenerovaná CTC dekóderom, kde symboly s nízkymi pravdepodobnosťami sú maskované. Z definície objektívnej funkcie 2.24 je zrejmé, že maskované symboly pomáhajú dekóderu v učení sa časového kontextu cieľovej sekvencie. Druhý model, ktorý je taktiež inšpirovaný modelom Transformer, nepoužíva maskované symboly. Tento model používa CTC predikcie ale s tým, že dekóder počas vý-

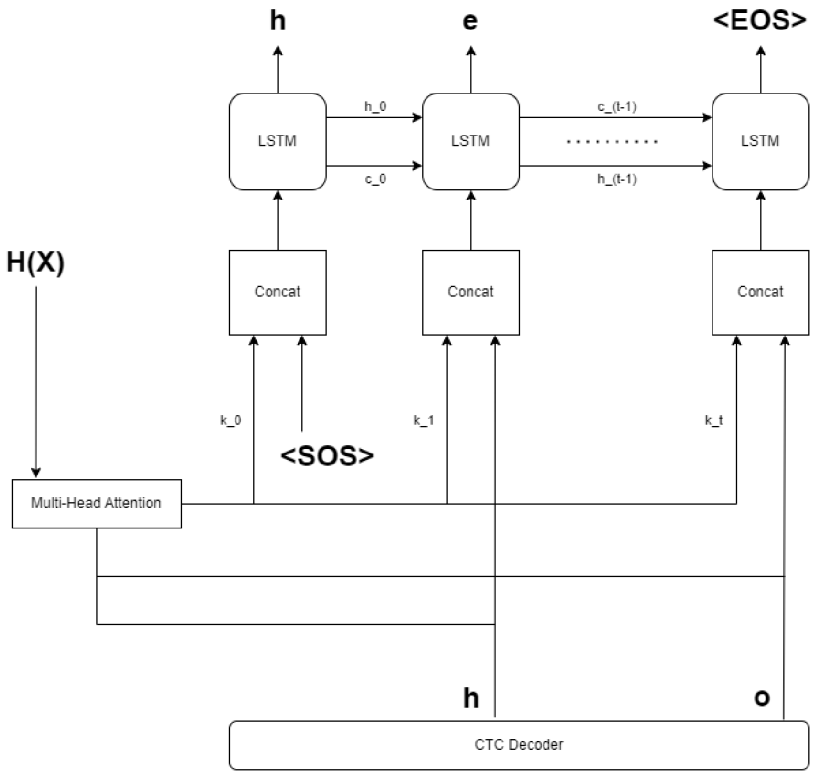
počtu berie do úvahy iba kontext v predchádzajúcich časových krokov, aby výpočet závisel aj od kontextu vstupnej sekvencie.

Na rozpoznanie ručne písaného textu nebol nájdený model, ktorý využíva non-autoregresívny dekóder. Počas štúdovania výskumných článkov bolo zistené, že non-autoregresívny prístup využívajú modely, ktoré sa zameriavajú na rozpoznávanie reči [8, 13, 27] alebo na strojový preklad textu [21, 12, 17]. Ako dekóder v enkóder-dekóder modeli je možné použiť dekódery modelov rozpoznávania reči zo sekcie 2.5, pretože obrázky obsahujúce ručne písané texty môžu mať podobné črty ako audio nahrávky obsahujúce hovorený text, aj keď sú rozdiely medzi dátovou reprezentáciou a ich spracovaním. Za varianty môžeme považovať aj dekódery, ktoré budú využívať rovnakú stratégiu ako zmienené modely v sekcii 2.5, ale budú nahradené sieťou LSTM s attention mechanizmom. V následujúcej sekcii sa podrobne popíšu navrhované riešenia, ktoré boli implementované.

3.1 Navrhované dekódery

CTC Self/Cross-Attention LSTM Toto riešenie sa skladá z dvoch variant, kde jedna varianta má *self-attention* mechanizmus a druhá varianta má *cross-attention* mechanizmus. Dôvod prečo je zahrnutý *self-attention* mechanizmus je, že má viacero výhod. Ako bolo zmienené v kapitole 2 v sekcii 2.2 tento mechanizmus môže zachytávať dlhodobé závislosti medzi vektormi alebo pozíciami vo vstupnej sekvencii, ktoré sú od seba v časových krokoch ďaleko. Toto môže pomôcť, pretože LSTM sieť nezachytáva minimálne dlhodobé závislosti medzi ďalekými vektormi v čase ako model Transformer. *Cross-attention* mechanizmus má výhodu v tom, že umožňuje zdieľanie informácií z rôznych vektorov, čím sa uľahčuje tok informácií medzi enkóderom a dekóderom, tak ako v modeli Transformer (na obrázku 2.3 je to stredná podvrstva v dekóderi). Tento mechanizmus môže byť užitočný z toho dôvodu, že na výpočet váh pre sekvenciu z LSTM siete enkódera zoberie do úvahy aj informáciu o CTC predikciách, čím sa môžu spresniť predikcie tohto riešenia. Obidve varianty okrem *Multi-head Attention* mechanizmu obsahujú aj jednosmernú sieť LSTM a CTC dekóder. Voľba jednosmernej LSTM siete bola inšpirovaná modelom zmieneným v podsekcii *CTC Enchanced*, kde brali do úvahy len kontext v predchádzajúcich časových krokoch. Výstup z enkódera obsahuje surové hodnoty a tie slúžia ako vstup do CTC dekódera, ktorý počíta CTC predikcie pomocou metódy *best path decoding* zmienenej v sekcii 2.3. Tieto CTC predikcie sú prehodené do embeddingov, aby mali vektorovú reprezentáciu. Následne je z nich vypočítané pozičné kódovanie [29] a sčítajú sa dokopy, aby bola do vektorových reprezentácií vnesená aj pozičná informácia. Vstup *attention* mechanizmu závisí od spomínaných variánt. Ak sa jedná o *self-attention* mechanizmus vstupom je sekvencia z LSTM siete enkódera, v prípade *cross-attention* mechanizmu je to sekvencia aj CTC predikcie z CTC dekódera. Výstupom sú kontextové vektory, ktoré sa konkatenujú s CTC predikciami v poslednej dimenzii. Takáto sekvencia je potom vstupom do jednosmernej LSTM siete. Výstup LSTM siete je posunutý do lineárnej vrstvy, ktorá produkuje surové hodnoty. Z týchto surových hodnôt sú počítané pravdepodobnosti pomocou funkcie softmax. V poslednom rade sú vybrané najväčšie pravdepodobnosti symbolov a prehodia sa na znaky. Prehľad varianty s *cross-attention* mechanizmu je na obrázku 3.1.

CTC Transformer Decoder je priamo prebraté z modelu, ktorý bol popísaný v podsekcii *CTC Enhanced* a bolo implementované. To znamená, že ako dekóder sa použije dekóder z modelu Transformer. Dôvod použitia je zmienený v sekcii 2.5 v podsekcii *CTC Enhanced*, tento dekóder limituje používanie kontextu z následujúcich časových krokov, čím sa



Obr. 3.1: Prehľad varianty s *cross-attention* mechanizmom. *H(X)* je výstup z LSTM siete enkódera, *k* je kontextový vektor, *c* je *cell state* a *h* je skrytý stav LSTM siete. CTC dekóder vypočíta CTC predikcie na základe surových hodnôt enkódera.

donucuje použitie kontextu z enkódera. Keď sa používa celý kontext CTC dekóderu, tak to podľa autorov *Song a spol.* potláča kontext z enkódera počas výpočtu predikcii dekódera.

Mask CTC Self/Cross-attention LSTM Ďalšia riešenia v tomto odstavci sú inšpirované modelom z podsekcie *Mask CTC*, kde sa maskujú symboly s nízkou pravdepodobnosťou. Majú skoro rovnakú štruktúru ako hore spomenuté predchádzajúce riešenia. Obsahujú obojsmernú LSTM sieť, *Multi-Head Attention* funkciu a CTC dekóder. Obojsmerná LSTM sieť bola zvolená z toho dôvodu, aby sa predikcia maskovaných symbolov vypočítala na základe spozorovaných symbolov. Prah maskovania je daný a maskujú sa CTC predikcie na základe neho. Tak ako v predchádzajúcich riešeniach, je tu varianta so *self-attention* a *cross-attention*. Tieto dve riešenia sa môžu trénovať dvomi spôsobmi. Prvý spôsob je zmienený v podsekcii *Mask CTC* a ten spočíva v náhodnom výbere symbolov, ktoré sa budú maskovať, čím budú modelu poskytnuté ťažké a ľahké príklady na trénovanie. Druhý spôsob môže byť pomocou zvoleného prahu, symboly sa budú maskovať na základe prahu a tým bude tréning kontrolovateľnejší. Pri prvom spôsobe je inferencia taká, ako je zmienená v sekcii 2.5 v podsekcii *Mask CTC*. V každej iterácii budú vypočítané predikcie pre C maskovaných symbolov, ktoré majú vypočítanú najvyššiu pravdepodobnosť na základe

Mask CTC Transformer Decoder je prebraté z podsekcie *Mask CTC* z dôvodu maskovania symbolov s nízkymi pravdepodobnosťami. Dekóder z modelu Transformer môže lepšie zachytávať dlhodobé závislosti medzi vektormi a CTC predikciami vďaka *attention*

mechanizmom, ktoré rušia rekurenciu. Maskované symboly môžu byť vďaka týmto funkciám korektne predikované na základe spozorovaných symbolov, ktoré sú ďaleko od maskovaného symbolu. Takisto ako *Mask CTC Self/Cross-attention LSTM* má dva spôsoby trénovania.

3.2 Návrh enkóderu a tréning siete

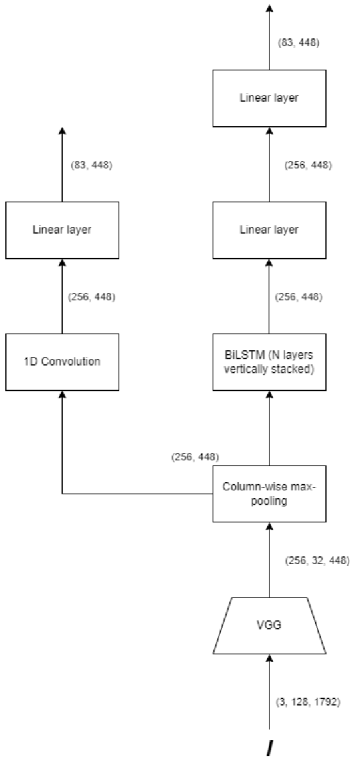
Implementácia neurónovej siete pozostáva z enkódera a všetkých navrhovaných riešení, ktoré boli spomenuté v predchádzajúcej časti. Enkóder je zložený z konvolučnej neurónovej siete VGG16BN, ktorá berie na vstup obrázky v rozmeroch (3, 128, 1792) a výstupom sú *feature* mapy v rozmeroch (256, 32, 448), kde uvedené rozmery predstavujú kanály, výška a šírka tenzoru. Z uvedených rozmerov je jasné, že VGG len podvzorkuje vstupný obrázok do 1/4 jeho veľkosti, aby sa nestratili dôležité črty obrázka. Z tejto konvolučnej neurónovej siete sa používa 22 vrstiev, ktoré pozostávajú z konvolučnej vrstvy s filtrom 3x3, *batch* normalizáciou, aktivačnou funkciou *ReLU* a *max-pooling* funkciou vo veľkosti 2x2 [26]. Dôvod výberu VGG siete bol taký, že obrázky *IAM Handwriting Database* datasetu neobsahujú komplikované pozadie [16]. Následne výstupný tenzor VGG-16-BN pôjde do *max-pooling* mechanizmu, kde sa vyberú maximálne hodnoty v každom stĺpci. Touto funkciou vymažeme dimenziu predstavujúcu výšku. Typicky sa vykonáva konkatenácia dimenzie výšky a kanálu a tým vznikne tenzor v rozmeroch (ch, w), kde c je počet kanálov, h je výška tenzoru a w je šírka tenzoru, ale spomenutou *max-pooling* operáciou dostaneme tenzor v rozmeroch (c, w). Táto *max-pooling* operácia zabezpečuje aj modelovú translačnú invariantnosť vo vertikálnom smere. Hlavnou myšlienkou tohto max-poolingu je to, že sa vyhadzujú ostatné črty v stĺpci, pretože sú dôležité črty súvisiace so znakom a nie ich priestorová pozícia [23]. Následne za týmto max-poolingom sa nachádzajú dve vetvy. Pomocná vetva sa používa len počas trénovania enkódera a obsahuje len jednorozmernú konvolučnú vrstvu a lineárnu vrstvu produkujúcu surové hodnoty. Táto vetva má za úlohu priame kódovanie kontextovej informácie a poskytovanie alternatívnej dekódovacej cesty [23]. Počas trénovania slúži pomocná vetva kvôli trénovaniu hlavnej vetvy enkódera, takže nie je nutné, aby počítala presné výsledky. Hlavná vetva sa skladá z obojsmernej LSTM siete, ktorá sa skladá z N vrstiev, ktoré sú vertikálne naskladané jedna na druhej. Ďalej hlavná vetva obsahuje lineárnu vrstvu na preškálovanie vektorov výstupu LSTM siete. Obsahuje aj lineárnu vrstvu, ktorá produkuje surové hodnoty. Výstupom hlavnej a pomocnej vetvy je tenzor v rozmeroch (a, w), kde a je veľkosť abecedy (v konkrétnych hodnotách je to tenzor v rozmeroch $(83, 448)$). Ak sa trénoval celý model s dekóderom, tak výstupom je tenzor z hlavnej vetvy a tenzor z obojsmernej LSTM siete, ktorý má veľkosť $(256, 448)$, kde prvá hodnota je veľkosť skrytej vektorovej dimenzie a druhá hodnota je dĺžka sekvencie. Výstup z obojsmernej LSTM siete predstavuje mapy čŕt obrázka. Oba výstupy potom slúžili ako vstup do ľubovoľnéj varianty dekódera, ktoré sú popísané vyššie.

Enkóder bol trénovaný samostatne a používal následujúcu chybovú funkciu [23]:

$$L_{CTC}(f_{rec}(f_{cnn}(I)); s) + 0.1 L_{CTC}(f_{shortcut}(f_{cnn}(I)); s) \tag{3.1}$$

kde f_{rec} predstavuje hlavnú vetvu, $f_{shortcut}$ reprezentuje pomocnú vetvu, f_{cnn} reprezentuje VGG s max-poolingom, I je vstupný obrázok a s je anotácia. Na obrázku 3.2 je vidieť prehľad enkódera.

Implementácia spomenutých riešení resp. z čoho by sa skladalo riešenie a ako by prebiehal výpočet je uvedená v predchádzajúcej sekcii. Na trénovanie celého modelu s ktoroukoľvek variantou dekóderu bola použitá objektívna funkcia 2.25. Pri trénovaní enkóderu alebo



Obr. 3.2: Prehľad enkódera. Naľavo je vidieť pomocnú vetvu a napravo od nej je hlavná vetva. Pri šípkach sú znázornené rozmery tenzoru. I je vstupný obrázok.

celého modelu na vstupných obrázkoch sa upravuje kontrast a jas obrázka, vykonáva sa Gaussovské rozmazanie aj so šumom a vykonáva sa rotácia obrázka.

Kapitola 4

Dátové sady

V tejto kapitole budú popísané dátové sady, ktoré by boli vhodné na natrénovanie siete.

IAM Handwriting Database [19] je datová sada, ktorá je tvorená z ručne písaných anglických viet, ktoré môžu byť použité na trénovanie a testovanie neurónových sietí určené na rozpoznanie ručne písaných textov a na vykonanie experimentov spojenými s identifikáciou a verifikáciou spisovateľa. Databáza obsahuje formuláre ručne písaného textu, ktoré boli naskenované v rozlíšení 300 dpi a uložené ako obrázky PNG s 256 úrovňami šedej. Samotný text pochádza z LOB (Lancaster-Oslo-Bergen) korpusu, ktorý obsahuje anglické texty rôznych žánrov napr. sci-fi, dobrodružné romány ale aj novinové články. Táto databáza sa skladá z následujúcich počtov a typov obrázkov, ktoré sú od 657 rôznych spisovateľov:

- 1 539 stránok skenovaného textu
- 5 685 izolovaných a anotovaných viet
- 13 353 izolovaných a anotovaných textových riadkov
- 115 320 izolovaných a anotovaných slov

Textové riadky boli vyrezané zo skenovaných formulárov a následne z nich boli vyrezané slová pomocou použitia automatickej segmentácie.

RIMES [11] je datová sada, ktorá je tvorená z ručne písaných listov vo francúzskom jazyku adresovaných firmám alebo administratívam, ktoré boli poslané poštou alebo faxom. Z právnych dôvodov, databáza neobsahuje pravé listy, takže každému dobrovoľníkovi pridelili fiktívnu identitu. Počet dobrovoľníkov bol 1 300. Táto databáza obsahuje 12 723 skenovaných stránok v rozlíšení 300 dpi. Z týchto stránok bolo extrahovaných 100 00 ručne písaných číslic a písmen z dotazníkov, 500 log z faxov a 300 000 ručne písaných slov z listov. Táto dátová sada slúži na úlohy analýzy rozloženia textu, rozpoznávania rukou písaného textu, identifikácie a overenia autora, identifikácie loga a extrakcie informácií.

Bentham Dataset [6] je dátová sada, ktorá obsahuje kolekciu dokumentov napísaných anglickým filozofom Jeremy Benthamom o rôznych témach. Táto kolekcia má viac ako 80 000 dokumentov a viacero z nich sú digitalizované. Z digitalizovaných dokumentov bolo anotovaných viac ako 6 000 dokumentov pomocou crowd-source platformy. Oficiálne je dostupná experimentálna partícia, ktorá obsahuje 11 479 textových riadkov.

Kapitola 5

Experimenty

V tejto kapitole sa nachádza stručný popis experimentov a výsledky s hodnotami. V prvom experimente sa zameriame na modely bez maskovaných CTC predikcií a na vplyv kontextu výstupu enkódera na spresnenie CTC predikcií. V druhom experimente sa zameriame na vplyv iteratívneho dekódovania na modely z predchádzajúceho experimentu. V treťom experimente sa zameriame na modely s maskovanými CTC predikciami. Vo štvrtom experimente budeme skúmať vplyv nastavenia jednotlivých prahov maskovaných symbolov počas tréningu a inferencie modelov. V piatom experimente sú vyskúšané modely, kde sa maskujú náhodné symboly počas tréningu a počas inferencie sa počítajú predikcie pre maskované symboly s najväčšou pravdepodobnosťou. V poslednom experimente je skúmaný vplyv iteratívneho dekódovania na modely z predchádzajúceho experimentu.

Ako dátová sada na trénovanie a testovanie bola vybraná dátová sada textových riadkov z *IAM Handwriting Database*. Trénovacia množina obsahovala 6161 obrázkov, validačná sada obsahovala 900 obrázkov a testovacia sada obsahovala 1861 obrázkov. Výber *IAM Handwriting Database* je odôvodnený tým, že obsahuje rôzne žánre textu v korpuse LOB a veľký počet autorov, čo zaisťuje veľkú variabilitu štýlov písania. Dátová sada *RIMES* obsahuje viac autorov ako *IAM Handwriting Database*, ale obsahuje len formuláre, čo znamená že nemusí tam byť zabezpečená variabilita štýlov písania z hľadiska žánru textu. *RIMES* nebol ani nájdený na Internete. Dátová sada *Bentham Dataset* má síce veľa obrázkov, ale obsahuje rukou písané texty jedného autora, čo nie je dobrá voľba.

5.1 Experimenty modelov bez maskovania

V tomto experimente sme trénovali modely, kde na vstupe dekódera boli CTC predikcie bez maskovaných symbolov. Boli vyskúšané varianty, ktoré boli popísané v sekcii 3.1, t. j. model s LSTM sieťou a *self-attention* funkciou, model s LSTM sieťou a *cross-attention* funkciou a model s dekóderom modelu Transformer. Vo všetkých variantách sa vykonala len jedna iterácia výpočtu dekóderu, čo znamená, že jeho výstup neslúžil opätovne ako vstup do CTC dekóderu. Podľa výsledkov v tabuľke 5.1 môžeme vidieť, že tieto návrhy moc nepomohli. CER hodnoty medzi navrhovanými riešeniami a enkóderom majú rozdiel medzi 7-9 %. Dôvodom môže byť málo vykonaných iterácii, čím sa CTC predikcie nemajú šancu zlepšiť. Podľa výsledkov v tabuľke varianta s *cross-attention* funkciou má okolo 2 % lepší výsledok ako varianta so *self-attention* mechanizmom. Z tohto môžme usúdiť, že informácia o CTC predikciach vypočítaných pomocou CTC dekóderov pomáha v počítaní váh pre vektory enkódera v spomenutej *cross-attention* funkcii. Varianta **CTC Transformer Decoder** má

lepšiu úspešnosť ako varianta **CTC Cross-Attention LSTM**, rozdiel medzi hodnotami je okolo 2 % ako pri oboch variantách s rôznymi *attention* funkciami. Varianta používajúca dekóder z modelu Transformer môže mať lepší výsledok kvôli tomu, že vďaka *self-attention* mechanizmom hľadá minimálnu cestu dlhodobých závislostí medzi vektormi, ktoré sú v rozmedzí časových krokov ďaleko od seba. Avšak, má horšiu znakovú chybovosť ako samotne trénovaný enkóder, takže nespresňuje CTC predikcie.

Model	CER (%) (T=1)	WER (%) (T=1)
CTC Self-Attention LSTM	15.47	42.51
CTC Cross-Attention LSTM	13.56	38.17
CTC Transformer Decoder	11.69	34.53
Encoder	6.38	20.70

Tabuľka 5.1: Tabuľka s hodnotami CER a WER metrík modelov bez maskovania. T je počet iterácii dekódera.

V ďalšom experimente bude vykonané iteratívne dekódovanie vo variante **CTC Transformer Decoder**. Iteratívne dekódovanie môže spresniť CTC predikcie z predchádzajúcich iterácií počas dekódovania. Počet iterácií sa zvolil 3 a 5.

Model (počet iterácií)	CER (%)	WER (%)
CTC Transformer Decoder (T=3)	12.20	35.67
CTC Transformer Decoder (T=5)	12.35	35.93

Tabuľka 5.2: Tabuľka s hodnotami CER a WER metrík modelov bez maskovania s iteratívnym dekódovaním. T je počet iterácii dekódera.

Ako vidíme v tabuľke 5.2, tak iteratívne dekódovanie prináša horšie výsledky ako výpočet len v jednej iterácii. Medzi dekódovaním v troch a piatich iterácii nie je skoro žiadny rozdiel.

5.2 Experimenty s modelmi s maskovaním

V ďalšom experimente sme trénovali modely, kde na vstupe dekódera sú maskované CTC predikcie na základe nízkej pravdepodobnosti daného maskovaného znaku. Boli vyskúšané varianty, ktoré boli popísané v sekcii 3.1. Tak ako v predchádzajúcej sekcií, aj tuto sa vykonala len jedna iterácia dekóderu. Prah pre maskované symboly bol nastavený na hodnotu 0.99. Ako je vidieť v tabuľke 5.3, tieto navrhované riešenia dosahujú horšej znakovej chybovosti ako natrénovaný enkóder. Avšak, keď sa porovnajú modely s podobnými architektúrami v tabuľkách 5.1 a 5.3), tak modely s maskovanými symbolmi dosahujú o 1-2 % nižšiu znakovú chybovosť ako modely bez masiek. Na základe tohto porovnania môžeme usúdiť, že maskovanie má o niečo lepší vplyv ako obmedzenie kontextu sekvencie CTC predikcií. Jedným z faktorov môže byť, že na výpočet finálnej sekvencie je použitý kontext z oboch strán, takýmto spôsobom sa model korektnejšie naučí jazykové pravidlá daného jazyka v ktorom bol textový riadok napísaný. Vďaka tomuto model vie potom vypočítať správne predikcie symbolov, ktoré sú maskované. Na základe výsledkov sa ukazuje, že dekóder z modelu Transformer má o niečo lepšie výsledky ako modely používajúce LSTM sieť.

Dôvod môže byť rovnaký ako experiment z predchádzajúcej sekcie, *attention* mechanizmy sa dokážu naučiť minimálnu cestu dlhodobých závislostí medzi prvkami sekvencie. Tieto modely môžu mať horšie výsledky od enkódera z toho dôvodu, že sa maskuje príliš moc symbolov vďaka nastavenému prahu, ktorý má vysokú hodnotu. Toto znamená, že na výpočet predikcií maskovaných symbolov je k dispozícií málo kontextu zo všetkých časových krokov.

Model (threshold=0.99)	CER (%) (T=1)	WER (%) (T=1)
Mask CTC Self-Attention LSTM	12.84	34.78
Mask CTC Cross-Attention LSTM	12.74	34.23
Mask CTC Transformer Decoder	11.59	32.57
Encoder	6.38	20.70

Tabuľka 5.3: Tabuľka s hodnotami CER a WER metrík modelov s maskovaním. T je počet iterácii dekódera.

V ďalšom experimente skúsime nastaviť nižšie prahy pre model *Mask CTC Transformer Decoder*, ktorý mal z navrhovaných riešení najmenšiu znakovú chybovosť v tabuľke 5.3. Boli zvolené následovné prahy: 0.95, 0.9, 0.8 a 0.7.

Prah	0.95	0.9	0.8	0.7
CER (%) (T=1)	9.72	9.77	9.86	8.92
WER (%) (T=1)	29.02	29.00	29.87	27.53

Tabuľka 5.4: Tabuľka s výsledkami trénovania modelu *Mask CTC Transformer Decoder* s rôznymi nastaveniami prahu. T je počet iteracií dekódera.

Podľa tabuľky 5.4 vidíme, že znižovanie prahu má pozitívny vplyv. Modely, ktoré majú prah medzi 0.8 - 0.95 majú približné rovnakú znakovú chybovosť, avšak oproti pôvodnému modelu z tabuľky 5.3 majú nižšiu znakovú chybovosť okolo 2 %. Z tohto môže vyplývať, že v predchádzajúcom experimente sa vyskytovalo príliš veľa maskovaných symbolov, čo môže znemožniť presnejšiemu výpočtu predikcií maskovaných symbolov.

V ďalšom experimente vyskúšame variantu modelov maskujúcich symboly, kde počas trénovania sa vyberie náhodný počet symbolov, ktoré sa zamaskujú a počas inferencie sa maskujú symboly, ktoré mali pravdepodobnosť výskytu nižšiu ako zvolený prah. Potom sa iteratívne dekódujú tieto symboly, v každej iterácii sa vypočíta predikcia pre C maskovaných symbolov, ktoré majú najväčšie pravdepodobnosti. Táto stratégia sa volá *easy first* a bola zmienená v sekcii 2.5 v podsekcii *Mask CTC*. Počas dekódovania vždy prebehla len jedna iterácia, takže všetky predikcie pre maskované symboly boli vypočítané naraz.

Ako vidíme v tabuľke 5.5 tento návrh riešenia je o kúsok lepší ako experimenty zaznamenané v tabuľke 5.3. Z tohto vyplýva, že náhodné maskovanie môže prispievať k učeniu sa kontextu cieľovej sekvencie, pretože model sa stretáva s ľahkými postupnosťami, kde je málo maskovaných symbolov a zároveň stretáva sa s ťažkými postupnosťami, kde je ich naopak veľa. Ďalším faktorom je spomenutý prah v inferencii, vďaka čomu je počas dekódovania poskytnuté väčšie množstvo informácií. V tomto experimente sa ale počas dekódovania vykonávala len jedna iterácia.

Model (prah=0.7)	CER (%) (T=1)	WER (%) (T=1)
Mask CTC Self-Attention LSTM	10.84	33.85
Mask CTC Cross-Attention LSTM	10.74	33.30
Mask CTC Transformer Decoder	10.52	32.90
Encoder	6.38	20.70

Tabuľka 5.5: Tabuľka s hodnotami CER a WER metrík modelov so stratégiou *easy first*. T je počet iterácii dekódera.

V následujúcom experimente sa vyberie model *Mask CTC Transformer Decoder* a vykoná sa podobný experiment, ale s vyšším počtom iterácií.

Model (počet iterácií)	CER (%)	WER (%)
Mask CTC Transformer Decoder (T=3)	12.13	36.89
Mask CTC Transformer Decoder (T=5)	11.99	36.78

Tabuľka 5.6: Tabuľka s hodnotami CER a WER metrík modelov s maskovaním a iteratívnym dekódovaním. T je počet iterácii dekódera v inferencii.

Z hodnôt tabuľky 5.6 možno usúdiť, že iteratívne dekódovanie zhoršuje výkon modelu.

5.3 Zhrnutie a návrh budúcej práce

Iteratívne dekódovanie zhoršuje znakovú chybovosť pri obidvoch typoch modelu, čo môže byť spôsobené nesprávnymi predikciami z predchádzajúcich iterácii a tým sa zanášajú chyby do následujúcich iterácii. Prahovanie maskovaných symbolov zlepšilo kvalitu predikcii modelu, pretože boli vypočítané na základe väčšieho počtu dostupných spozorovaných symbolov. Modely, ktoré používali dekóder modelu Transformer, sú zo všetkých modelov najúspešnejšie kvôli zrušenej rekurencii zmienenej v sekcii 2.2.

Ďalej by bolo vyskúšané predtrénovanie dekóderu na textovom korpuse, ktorého texty by mali rovnaký charakter ako *IAM Handwriting Database*. Vďaka tomuto predtrénovaniu by mal dekóder znalosť o jazykových pravidlách, čo by pomohlo k nižšej znakovej chybovosti non-autoregresívneho dekódera. Ďalší návrh by bol prebratie ľubovoľného autoregresívneho dekóderu, ktorý by mal nižšiu znakovú chybovosť ako enkóder a využitie tzv. *mimicking learning*-u ako navrhli autori *Qiao a spol.* [22]. Táto stratégia spočíva v tom, že oba dekódery by zdieľali rovnaký enkóder a na konci by sa počítala chyba podobnosti z výstupov oboch dekóderov. Takto sa non-autoregresívny dekóder naučí lepšie extrahovať presnejšie a diskriminačné informácie, ktoré autoregresívny dekóder produkuje na základe podmienenej závislosti zmienenej v sekcii 2.5.

Kapitola 6

Záver

Cieľom tejto práce je navrhnúť a naimplementovať non-autoregresívny dekóder, ktorého účelom je predikcia znakov nezávisle na ostatné predikované znaky v ostatných časových krokoch. Motiváciou tohto cieľa je spresnenie CTC predikcií enkódera, čiže korekcia nesprávnych predikcií, ktoré boli zanesené enkóderom.

V rámci tejto práce boli naštudované rekurentné neurónové siete, ktoré zdieľajú parametre naprieč časovými krokmi, aby mohli byť spracované sekvencie v ktorých majú prvky medzi sebou závislosť. Ďalej bol vysvetlený model Transformer, ktorý vďaka *attention* mechanizmom ruší rekurentné výpočty, čím umožňuje modelovanie závislostí medzi ďalekými prvkami. Ako ďalšie bolo vysvetlené *Connectionist Temporal Classification*, ktorého účelom je trénovanie neurónových sietí nesegmentujúce vstupné dáta. Ďalej boli predstavené seq2seq modely, ktoré sú používané na rozpoznanie rukou písaných textov a majú enkóder-dekóder architektúru s rekurentnými neurónovými sieťami. Po tomto boli vysvetlené rozdiely autoregresívnych a non-autoregresívnych modelov. Ako posledné boli predstavené metriky *Character Error Rate* a *Word Error Rate*.

Následne bol predstavený návrh rôznych riešení, ktoré používajú myšlienky z kapitoly 2. Tieto riešenia hlavne používajú myšlienky predstavených non-autoregresívnych dekóderov v sekcii 2.5, ktoré buď blokujú informácie z budúcich časových krokov kvôli donúteniu použitia kontextovej informácie z enkódera alebo maskujú symboly, ktoré majú nízku pravdepodobnosť výskytu a počítajú predikcie na základe spozorovaných symbolov. Tieto dekódery používali buď LSTM alebo Transformer model. Následne boli predstavené dátové sady *IAM Handwriting Database*, *Rimes* a *Bentham Dataset*, ktoré by boli vhodné na trénovanie modelu. Následne bol vybraný dataset *IAM Handwriting Database*, pretože obsahoval texty rôznych žánrov od rôznych autorov. V poslednom rade boli vykonané experimenty, ktorých hlavná myšlienka bolo skúmanie vplyvu nepoužívania budúcej informácie a maskovania predikovaných symbolov. Na základe experimentov sa ukázalo, že tieto koncepty nefungujú v prípade modelov na rozpoznanie ručne písaných textov. Ďalej experimenty ukázali, že iteratívne dekódovanie zhoršuje výpočet predikcií. Dôvodom môžu byť nesprávne vypočítané predikcie v predchádzajúcich iteráciach. Ako posledné, zníženie prahu sa ukázalo byť prospešné. Najlepší model má znakovú chybovosť 8.92 %, čo ale nespĺňa zadanie, pretože enkóder má chybovosť 6.38 %.

Literatúra

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

[2] Barrere, K., Soullard, Y., Lemaitre, A. a Coüasnon, B. A light transformer-based architecture for handwritten text recognition. In: Springer. *Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings*. 2022, s. 275–290.

[3] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K. a Bengio, Y. Attention-based models for speech recognition. *Advances in neural information processing systems*. 2015, zv. 28.

[4] Chowdhury, A. a Vig, L. *An Efficient End-to-End Neural Model for Handwritten Text Recognition*. 2018.

[5] Doetsch, P., Zeyer, A. a Ney, H. Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition. In: IEEE. *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, s. 361–366.

[6] Gatos, B., Louloudis, G., Causer, T., Grint, K., Romero, V. et al. Ground-Truth Production in the Transcriptorium Project. In: *2014 11th IAPR International Workshop on Document Analysis Systems*. 2014, s. 237–241. DOI: 10.1109/DAS.2014.23.

[7] Gers, F., Schmidhuber, J. a Cummins, F. Learning to forget: continual prediction with LSTM. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. 1999, sv. 2, s. 850–855 vol.2. DOI: 10.1049/cp:19991218.

[8] Ghazvininejad, M., Levy, O., Liu, Y. a Zettlemoyer, L. Mask-predict: Parallel decoding of conditional masked language models. *ArXiv preprint arXiv:1904.09324*. 2019.

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

[10] 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.

[11] Grosicki[1], E., Carre, M., Brodin, J.-M. a Geoffrois[1], E. RIMES evaluation campaign for handwritten mail processing. Citeseer. 2008.

[12] Gu, J., Bradbury, J., Xiong, C., Li, V. O. a Socher, R. Non-autoregressive neural machine translation. *ArXiv preprint arXiv:1711.02281*. 2017.

[13] Higuchi, Y., Watanabe, S., Chen, N., Ogawa, T. a Kobayashi, T. Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict. *ArXiv preprint arXiv:2005.08700*. 2020.

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

[15] Kang, L., Riba, P., Rusiñol, M., Fornés, A. a Villegas, M. Pay attention to what you read: Non-recurrent handwritten text-Line recognition. *Pattern Recognition*. 2022, zv. 129, s. 108766. DOI: https://doi.org/10.1016/j.patcog.2022.108766. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0031320322002473.

[16] 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. *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*. 2019, s. 459–472.

[17] Libovický, J. a Helcl, J. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. *ArXiv preprint arXiv:1811.04719*. 2018.

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

[19] 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, s. 39–46.

[20] Michael, J., Labahn, R., Grüning, T. a Zöllner, J. *Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition*. 2019.

[21] Qian, L., Zhou, H., Bao, Y., Wang, M., Qiu, L. et al. Glancing transformer for non-autoregressive neural machine translation. *ArXiv preprint arXiv:2008.07905*. 2020.

[22] Qiao, Z., Zhou, Y., Wei, J., Wang, W., Zhang, Y. et al. Pimnet: a parallel, iterative and mimicking network for scene text recognition. In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, s. 2046–2055.

[23] Retsinas, G., Sfikas, G., Gatos, B. a Nikou, C. Best practices for a handwritten text recognition system. In: Springer. *Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings*. 2022, s. 247–259.

[24] Sak, H., Senior, A. a Beaufays, F. *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. 2014.

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

[26] Simonyan, K. a Zisserman, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* 2015.

[27] Song, X., Wu, Z., Huang, Y., Weng, C., Su, D. et al. Non-autoregressive transformer asr with ctc-enhanced decoder input. In: IEEE. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2021, s. 5894–5898.

[28] Sueiras, J., Ruiz, V., Sanchez, A. a Velez, J. F. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing.* 2018, zv. 289, s. 119–128. DOI: https://doi.org/10.1016/j.neucom.2018.02.008. ISSN 0925-2312. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0925231218301371.

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention Is All You Need. *CoRR.* 2017, abs/1706.03762. Dostupné z: http://arxiv.org/abs/1706.03762.