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

VYUŽITÍ NEANOTOVANÝCH DAT PRO TRÉNOVÁNÍ OCR

OCR TRAINED WITH UNANOTATED DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PETR BUCHAL**
AUTHOR

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

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Buchal Petr, Bc.**
Program: Informační technologie
Obor: Počítačová grafika a multimédia
Název: **Využití neanotovaných dat pro trénování OCR**
OCR Trained with Unanotated Data
Kategorie: Zpracování obrazu
Zadání:

1. Prostudujte základy konvolučních sítí a rozpoznávání ručně psaného textu a jazykových modelů.
2. Vytvořte si přehled o současných metodách rozpoznávání ručně psaného textu pomocí konvolučních sítí se zaměřením na možnosti využití neanotovaných dat.
3. Navrhněte metodu využívající neanotované digitalizované dokumenty a jazykové modely pro trénování neuronových sítí pro 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:

- Podle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Vytvoření kvalitního systému rozpoznání textu (OCR) vyžaduje velké množství anotovaných dat. Získání, potažmo vytvoření anotací je nákladný proces. Tato práce se zabývá několika způsoby efektivního využití neanotovaných dat pro trénování OCR neuronové sítě. Navržené metody využívající neanotovaná data spadají do kategorie self-training algoritmů. Obecný postup navržených metod se dá sumarizovat tak, že nejprve je na omezeném množství anotovaných dat natrénován počáteční model neuronové sítě. Ten je následně spolu s jazykovým modelem použit k vygenerování pseudo-štítků neanotovaných dat. Takto strojově anotovaná data jsou zkombinována s trénovacími daty, která byla použita k vytvoření počátečního modelu a následně jsou využita k natrénování cílového modelu. Úspěšnost jednotlivých metod je měřena na ručně psaném ICFHR 2014 Bentham datasetu. Experimenty byly provedeny na dvou datových sadách, které reprezentují různou míru dostupnosti anotovaných dat. Nejlepší model trénovaný na malé datové sadě dosahuje 3.70 CER [%], což je relativní zlepšení o 42 % oproti počátečnímu modelu trénovanému pouze na anotovaných datech a nejlepší model trénovaný na velké datové sadě dosahuje 1.90 CER [%], což je relativní zlepšení o 26 % oproti počátečnímu modelu. Za pomocí navržených metod lze efektivně zvýšit úspěšnost OCR s využitím neanotovaných dat.

Abstract

The creation of a high-quality optical character recognition system (OCR) requires a large amount of labeled data. Obtaining, or in other words creating, such a quantity of labeled data is a costly process. This thesis focuses on several methods which efficiently use unlabeled data for the training of an OCR neural network. The proposed methods fall into the category of self-training algorithms. The general approach of all proposed methods can be summarized as follows. Firstly, the seed model is trained on a limited amount of labeled data. Then, the seed model in combination with the language model is used for producing pseudo-labels for unlabeled data. Machine-labeled data are then combined with the training data used for the creation of the seed model and they are used again for the creation of the target model. The successfulness of individual methods is measured on the handwritten ICFHR 2014 Bentham dataset. Experiments were conducted on two datasets which represented different degrees of labeled data availability. The best model trained on the smaller dataset achieved 3.70 CER [%], which is a relative improvement of 42 % in comparison with the seed model, and the best model trained on the bigger dataset achieved 1.90 CER [%], which is a relative improvement of 26 % in comparison with the seed model. This thesis shows that the proposed methods can be efficiently used to improve the OCR error rate by means of unlabeled data.

Klíčová slova

neuronová síť, rozpoznání textu, self-training, neanotovaná data, jazykový model

Keywords

neural network, text recognition, self-training, unlabeled data, language model

Citace

BUCHAL, Petr. *Využití neanotovaných dat pro trénování OCR*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Využití neanotovaných dat pro trénování OCR

Prohlášení

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

.....
Petr Buchal
17. května 2021

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce Ing. Michalu Hradišovi, Ph.D. za odborné vedení, za rady a za pomoc při zpracování této práce.

Tato práce vznikla za podpory Ministerstva kultury České republiky v rámci projektu NAKI II PERO (DG18P02OVV055).

Obsah

Kapitola 1

Úvod

Dnešní doba je bezpochyby dobou digitalizace. Instituce po celém světě se snaží digitalizovat své dokumenty, aby k nim umožnily přístup budoucím generacím. Toto platí obzvláště pak o historických dokumentech, které jsou náročné na údržbu a do budoucna hrozí ztráta cenných informací. Digitalizace starých archivů vytvořila poptávku po vysoce spolehlivých OCR systémech, které jsou potřebné k indexování textu např. pro možnost vyhledávání. Vytváření OCR systémů ovšem vyžaduje velké množství ručně anotovaných dokumentů, což je finančně nákladné. U mnohých dokumentů s málo užívanými jazyky jsou pak k přepisování potřeba dokonce odborníci, kteří konečnou cenu ještě zvyšují.

Tato práce se zabývá vytvořením kvalitního OCR systému za pomocí malého množství anotovaných dat, výrazně většího množství neanotovaných dat a textového korpusu. Navrhl jsem několik metod, které spadají do kategorie algoritmů self-trainingu. Jeho základní myšlenkou je generování anotací pro neanotovaná data modelem, který byl trénován na dostupných anotovaných datech a využití těchto strojově anotovaných dat ke zvýšení úspěšnosti modelu samotného. Důležitým prvkem všech navržených metod je korekce strojových anotací jazykovým modelem, který byl vytvořen z dostupného textového korpusu. Druhým klíčovým prvkem navržených přístupů je využití několika nejpravděpodobnějších variant strojového přepisu pro trénování namísto jednoho nejpravděpodobnějšího.

Tři ze čtyř navržených přístupů pro trénování neuronové sítě pomocí strojově anotovaných dat nezanedbatelně zvýšily úspěšnost OCR systému. Výsledky této práce mohou do budoucna znamenat zlevnění celého proces vytváření kvalitních OCR systémů.

Kapitola 2

Optické rozpoznávání znaků

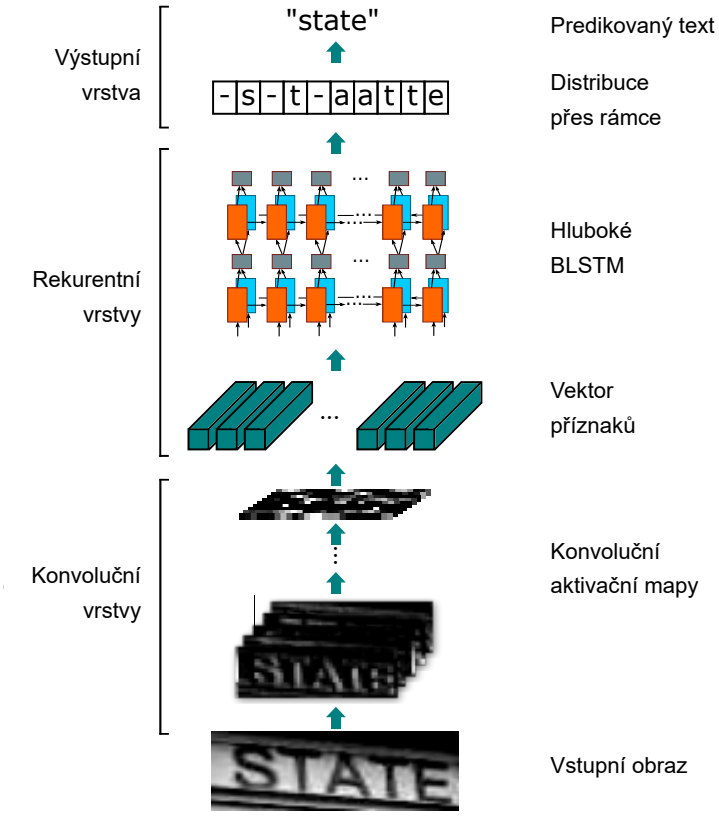
Optické rozpoznávání znaků (dále jen OCR) je systém, který převádí text z obrazové podoby do podoby textové, která je indexovatelná. OCR tedy umožňuje rychlé vyhledávání v digitalizovných a indexovatelných dokumentech [32].

OCR systémy mohou být na základě vstupních dat dále děleny na online a offline systémy. Offline systém je statický systém, který zpracovává data ve formě naskenovaných dokumentů. Online systém je více dynamický a je založen na pohybu pera, jeho rychlosti, úhlu a pozici [32]. Tato práce se zabývá offline systémy OCR.

Počátek vývoje OCR systémů se datuje do 40. let 20. století. Z počátku byly tyto systémy vyvíjeny na pomoc slepým pro převod textu na syntetickou řeč. V 60. letech byl představen první OCR systém schopný rozpoznat ručně psané číslice. V 70. letech byla vylepšována odezva a přesnost. V 80. letech byla založena firma ABBYY, která dnes provozuje jeden z nejznámějších softwarových OCR systémů ABBYY FineReader [57]. S nástupem stolních počítačů se vývoj OCR technologie zrychloval a v 10. letech 21. století docházelo k vývoji dalších komerčních i volně dostupných softwarových OCR systémů jako Tesseract [46] nebo OCRopus [7]. V poslední dekádě pak došlo ke skokovému zlepšení technologie v důsledku zvětšení výpočetních kapacit a výzkumu strojového učení. Využívány byly přístupy založené na využití náhodného lesa (RF), metody podpůrných vektorů (SVM), metody k nejbližších sousedů (kNN) nebo metody rozhodovacího stromu (DT). Tyto metody však byly postupně nahrazeny neuronovými sítěmi, které jsou aktuálně nejmodernějším přístupem [32].

OCR systémy tvořené neuronovými sítěmi se mohou lišit v použitých architekturách, obecně ale daný systém musí nejprve lokalizovat řádek, následně ho zpracovat a poté může dodatečně zlepšit kvalitu přepisu například pomocí jazykového modelu. OCR systém zkoumaný v této práci se skládá právě ze tří komponent. Řádek je lokalizován pomocí separátní konvoluční neuronové sítě, která detekuje linku, na které řádek sedí, polygon, ve kterém se řádek nachází a odstavec, do kterého náleží [26]. Druhou komponentou je konvoluční neuronová síť s rekurentními vrstvami [45] trénovaná CTC chybovou funkcí [19], viz obrázek 2.1. Tato neuronová síť na vstupu zpracovává detekovaný vyřezaný řádek a na výstupu vrací pravděpodobnosti symbolů na jednotlivých pozicích. Tento výstup se následně dekóduje, k čemuž lze použít několik různých algoritmů (např. hladové dekódování, prefixové paprskové dekódování...), které produkují samotný text přepisu. Tato práce se zabývá zvýšením úspěšnosti právě neuronové sítě zpracovávající detekovaný řádek.

Čas od času se ve výsledném přepisu mohou vyskytnout drobné chyby (např. písmeno i může být chybně rozpoznáno jako j), zde bývá použita třetí komponenta - jazykový model, který může být implementovaný například jako LSTM neuronová síť [51]. Ten v sobě udržuje přehled o slovech a skladbě vět daného jazyka. Jazykový model slova s chybami opravuje



Obrázek 2.1: Architektura CRNN použitá v OCR systému této práce. Převzato z práce [45].

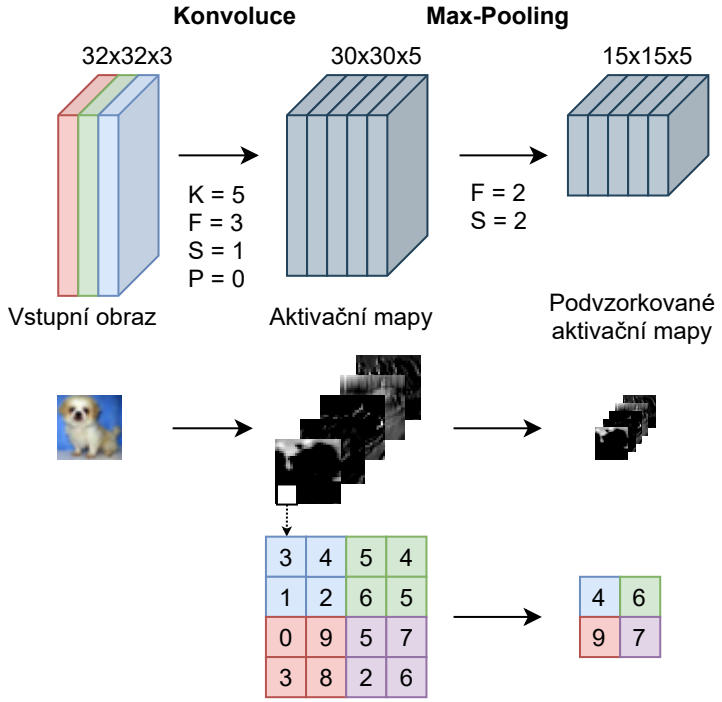
na slova, která v daném kontextu dávají smysl, jsou gramaticky podobná původním slovům a jsou bez chyb. Následující text rozebírá několik typů neuronových sítí a logiku za jejich použitím v OCR systémech.

2.1 Konvoluční neuronové sítě

V nejmodernějších OCR systémech hrají konvoluční neuronové sítě (dále jen CNN) důležitou roli při počátečním zpracování obrazu. Tradiční neuronové sítě s plně propojenými vrstvami jsou pro práci s obrazem nevhodné kvůli velkému počtu neuronů potřebnému pro jeho zpracování. S obrazem velikosti 1024×768 pixelů by neuronová síť na vstupu musela zpracovat 2 359 296 čísel (jedná-li se o RGB obraz se třemi kanály). Následně by pak každému neuronu v první vrstvě neuronové sítě příslušel stejný počet vah [48]. Tento přístup by v praxi byl složitě realizovatelný. Pro zpracování obrazu se standardně využívají CNN, které vstupní obraz zpracovávají pomocí naučených konvolučních filtrů, které v něm detekují příznaky využívající se v dalších výpočtech. Základem CNN jsou konvoluční a pooling vrstvy. Vizualizace operací, které tyto vrstvy provádí, je k dispozici na obrázku 2.2.

Konvoluční vrstvy

Konvoluční vrstva se skládá z množiny filtrů, jejichž hodnoty se CNN učí. Každý filtr má určité receptivní pole, to je ekvivalentní s velikostí filtru [1]. Během dopředného průchodu je každý filtr konvolvován se vstupem napříč šířkou, výškou a vytváří 2-dimenzionální ak-



Obrázek 2.2: Vizualizace příkladu zpracování obrazu jedním blokem CNN. CNN na vstupu dostane obraz o rozměrech 32×32 pixelů se 3 kanály. Ten je zpracován konvoluční vrstvou, která obsahuje K=5 filtrů o velikosti F=3 s krokem S=1 a šířkou okraje P=0. Konvoluční vrstva na výstupu vyprodukuje objem o rozměrech 30×30 pixelů s hloubkou 5, která je rovná počtu filtrů. Objem dále zpracuje Max-pooling vrstva s prostorovým obsahem F=2 a krokem S=2, která objem 2× podvzorkuje.

tivační mapu vstupního obrazu [54]. Během trénování si neuronová síť upravuje tyto filtry tak, aby dokázaly vidět unikátní příznaky objektů nacházejících se ve vstupním obraze. Filtry natrénované CNN jsou tak schopny detekovat specifické hrany konkrétních objektů, oblasti určité barvy nebo celé vzory. V kontextu OCR konvoluční vrstvy detekují hrany jednotlivých znaků. Výstup CNN je pak objem o hloubce rovné počtu filtrů aplikovaných na vstupní obraz [1], tedy počtu aktivačních map.

Konvoluční vrstva na vstup přijímá objem rozměrů $W_1 \times H_1 \times D_1$ a vyžaduje 4 hyperparametry - počet filtrů K, jejich velikost F, krok S a šířku okraje P. Na výstupu produkuje nový objem $W_2 \times H_2 \times D_2$, kde

$$W_2 = (W_1 - F + 2P)/S + 1, \tag{2.1}$$

$$H_2 = (H_1 - F + 2P)/S + 1, \tag{2.2}$$

$$D_2 = K. \tag{2.3}$$

Konvoluční vrstva tak obsahuje $F \cdot F \cdot D_1$ vah na jeden filtr a dohromady tedy $(F \cdot F \cdot D_1) \cdot K$ vah a K biasů. V novém objemu vyprodukovaném konvoluční vrstvou je d-tý výřez (velikosti $W_2 \times H_2$) výsledek konvoluce d-tého filtru nad vstupním objemem s krokem S a offsetem d-tého biasu [1].

Pooling vrstvy

Pooling vrstva se v CNN obvykle nachází mezi jednotlivými konvolučními vrstvami. Její funkcí je zmenšení velikosti jejího vstupu pro další zpracování. Pooling vrstvy tak pomáhají snížit výpočetní náročnost trénování neuronové sítě a zároveň předejít jejímu přetrénování. Pooling vrstva zmenšuje rozměry aktivační mapy spojením hodnot sousedících aktivací do jedné hodnoty vstupující do další vrstvy [54]. Pooling vrstvy mohou sdružovat hodnoty aktivací například výběrem největší hodnoty tzv. Max-Pooling nebo výpočtem průměrné hodnoty tzv. Average-Pooling. Pooling vrstva na vstupu přijímá objem rozměrů $W_1 \times H_1 \times D_1$ a vyžaduje dva hyperparametry - prostorový obsah F a krok S. Na výstupu produkuje nový objem $W_2 \times H_2 \times D_2$, kde

$$W_2 = (W_1 - F)/S + 1, \tag{2.4}$$

$$H_2 = (H_1 - F)/S + 1, \tag{2.5}$$

$$D_2 = D_1. \tag{2.6}$$

Pooling vrstva nemá žádné učící parametry, protože počítá pevně danou funkci pro vstupní hodnoty [1].

2.2 Rekurentní neuronové sítě

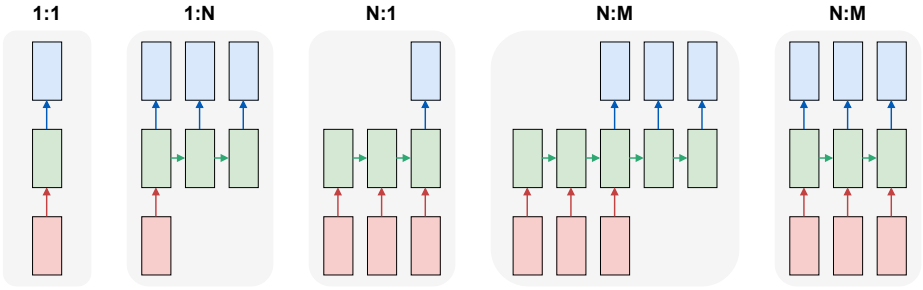
Klasické neuronové sítě na vstup přijímají vektor fixní délky a produkují vektor fixní délky na výstupu. Rekurentní neuronové sítě (dále jen RNN) umožňují provádět výpočet neuronové sítě nad sekvencemi vektorů a produkovat sekvenci vektorů [25]. Spojení mezi buňkami RNN tvoří orientovaný graf podél časové posloupnosti [55]. Tedy každý výstup RNN závisí na předchozích vstupech, které neuronovou sítí prošly. Díky této vlastnosti se RNN skvěle hodí ke zpracování časových řad. V kontextu OCR jsou rekurentní vrstvy využívány k modelování určité implicitní jazykové znalosti trénovaného modelu. Neuronová síť si tuto znalost vytváří při trénování díky tomu, že rekurentní vrstvy se dívají na řádek textu jako na sekvenci. Na obrázku 2.3 je vizualizován rozdíl mezi RNN a klasickými neuronovými sítěmi.

Diagram buňky RNN je k dispozici na obrázku 2.5a. Informace o předchozích vstupech, které neuronovou síti prošly jsou uložené ve skrytém stavu. Hodnotu skrytého stavu h_t v čase t lze získat rovnicí

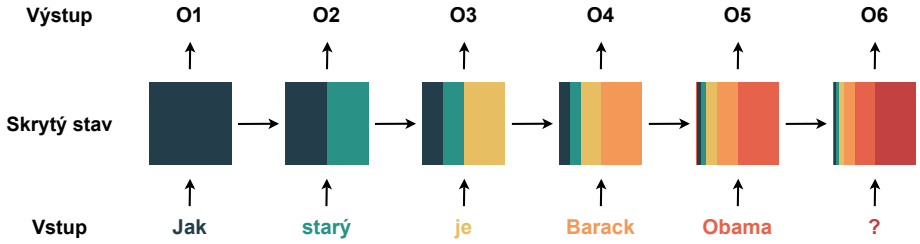
$$\mathrm{h}_t = \phi(W\mathrm{x}_t + U\mathrm{h}_{t-1}), \tag{2.7}$$

kde x_t je vstup v čase t násobený váhovou maticí W, h_{t-1} je skrytý stav v čase $t-1$ násobený maticí skrytého stavu U, která je stochastickou maticí. Váhová matice rozhoduje o poměru důležitosti aktuálního vstupu a skrytého stavu. Následně je na součet zpracovaného vstupu a skrytého stavu aplikována aktivační funkce ϕ (obvykle hyperbolický tangens) [36].

Díky nepřetržitému propagování předchozí hodnoty skrytého stavu do aktuálního, jsou ve skrytém stavu informace o všech předchozích vstupech. S každým dalším vstupem se ale zmenšuje objem informací o časově více vzdálených vstupech. Na obrázeku 2.4 je problém demonstrován na zpracování sekvence slov. Tuto potíž trápící RNN lze zjednodušeně pojmenovat jako krátkodobou paměť. Vzniká jako důsledek problému mizejícího gradientu [39]. K tomu dochází při trénování neuronových sítí algoritmem Backpropagation. Cílem tohoto



Obrázek 2.3: Každý obdélník reprezentuje vektor a šipka reprezentuje operaci. Vstupní vektory jsou červené, výstupní vektory jsou modré a zelené vektory reprezentují stav RNN. První diagram zleva zobrazuje princip klasické neuronové sítě se vstupem fixní délky a výstupem fixní délky. Tyto neuronové sítě jsou vhodné například pro klasifikaci nebo segmentaci. Ostatní diagramy zobrazují různé využití RNN. Příkladem úlohy odpovídající druhému diagramu je generování slovního popisu obrázku. Příkladem úloh dalších diagramů zleva může být analýza sentimentu z textu, překlad věty z jednoho jazyka do druhého a klasifikace jednotlivých snímků videa [25]. Převzato z práce [25].

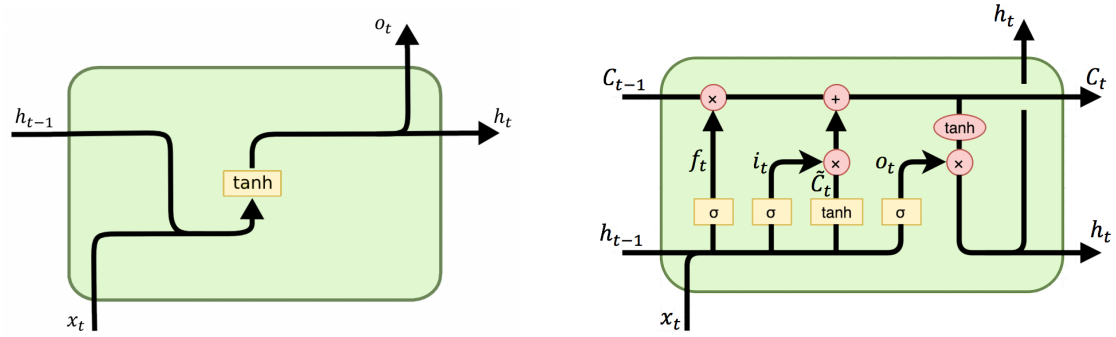


Obrázek 2.4: Vizualizace problému krátkodobé paměti RNN. V průběhu zpracování pozdějších slov se informace o slovech ze začátku věty začínají postupně vytrácet a pro neuronovou síť se stává interpretace věty jako celku problematická. Převzato z práce [39].

algoritmu je minimalizovat chybovou funkci L, která je funkcí všech vah neuronové sítě a vyjadřuje odchylku výstupu neuronové sítě od požadovaných hodnot [14]. Algoritmus nejprve provede dopředný průchod vstupních dat neuronovou sítí. Následně porovná výstup sítě s odpovídající referencí pomocí chybové funkce L. Ta vrátí chybovou hodnotu, která určuje, jak moc se od sebe výstup a reference liší. Algoritmus tuto hodnotu využije pro výpočet gradientů jednotlivých neuronů v síti. Gradienty jsou následně využity k úpravě vah neuronové sítě při zpětném průchodu neuronovou sítí. Problémem je skutečnost, že každý neuron bere při výpočtu svého gradientu v potaz efekt gradientů předchozí vrstvy, což má za následek exponenciální zmenšování vlivu gradientů na počáteční vrstvy neuronové sítě, které se učí jako poslední. V případě RNN se gradient exponenciálně zmenšuje s každým časovým krokem a neuronová síť se tak není schopná naučit dlouhodobé závislosti [39], viz obrázek 2.4.

Architektura LSTM

LSTM (Long short-term memory) [22] je architektura RNN, která je schopná naučit se dlouhodobé závislosti [39]. Oproti obyčejným RNN se LSTM liší buňkami, které zpracovávají přicházející data.



(a) RNN buňka

(b) LSTM buňka

Obrázek 2.5: Vizualizace paměťových buněk rekurentních neuronových sítí. Převzato z práce [11].

LSTM buňka je k dispozici na obrázku 2.5b. Vodorovná linka v horní části LSTM buňky reprezentuje stav buňky, který obsahuje informaci o předchozích vstupech neuronové sítě a dal by se přirovnat k dlouhodobé paměti. LSTM má speciální mechanismus, kterým může do stavu buňky přidávat nebo odebírat informace. Činí tak pomocí struktur, které se jmenují brány. Ty jsou složeny z aktivační funkce sigmoida a maticového násobení. Aktivační funkce sigmoida

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

vrací hodnotu mezi 0 a 1, která v případě LSTM slouží ke kontrole množství informace, která poputuje dál v buňce. V LSTM buňce jsou takové brány tři - zapomnětlivá, vstupní a výstupní [36]. Zapomnětlivá brána reguluje jaké informace by měly být zachovány a jaké by měly být zahozeny ze stavu buňky C v čase $t-1$. Zapomnětlivá brána tak činí na základě skrytého stavu buňky h v čase $t-1$ a vstupu x v čase t pomocí rovnice

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{2.9}$$

kde W_f jsou váhy vrstvy a b_f je bias. Vstupní brána rozhoduje o nových informacích, které budou propagovány do stavu buňky. Rovnice výpočtu výstupu vstupní brány i_t je stejná jako výstupu zapomnětlivé brány i_f jen s rozdílem, že vstupní brána má své vlastní váhy W_f a bias b_f. Další zastávkou vstupu x_t a skrytého stavu h_{t-1} je aktivační funkce hyperbolický tangens, která vytvoří vektor kandidátních hodnot $\tilde{C}$ následovně

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{2.10}$$

Hodnoty vektoru kandidátních hodnot $\tilde{C}$ jsou v následujícím kroku vynásobeny výstupem vstupní brány i_t a přidány do stavu buňky C_t i se změnami f_t způsobenými zapomnětlivou bránou podle vztahu

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{2.11}$$

Poslední bránou je výstupní brána. Výpočet jejího výstupu probíhá opět stejně jako u dvou předchozích bran jen s vlastními váhami W_o a biasem b_o. Výstup výstupní brány o_t je pak

vynásoben stavem buňky C_t, na který je ještě před tím aplikován hyperbolický tangens následovně

$$h_t = o_t * \tanh(C_t), \tag{2.12}$$

kde h_t je nový skrytý stav buňky [37].

2.3 Connectionist temporal classification

Výstupem rekurentních vrstev neuronové sítě v OCR úloze je pravděpodobnost jednotlivých znaků přes pozice ve vstupní sekvenci, viz obrázek 2.6. Connectionist temporal classification [19] (dále jen CTC) je algoritmus, který je možné ve spojení s výstupem RNN využít k dosažení dvou cílů - trénování neuronové sítě a dekódování textu z pravděpodobností jednotlivých znaků. Použití algoritmu CTC při trénování OCR neuronové sítě je výhodné, protože tato chybová funkce potřebuje pouze textový přepis řádku a nikoliv přesné pozice jednotlivých znaků [43].

Mějme vstupní sekvenci $X = [x_1, x_2, ..., x_t]$ (např. pixely řádku) a výstupní sekvenci $Y = [y_1, y_2, ..., y_t]$ (např. přepis řádku), algoritmus CTC tyto sekvence mapuje na sebe podle pravděpodobnosti jednotlivých znaků na výstupu RNN. Při mapování algoritmu nevadí různě dlouhé sekvence X a Y, různý poměr sekvencí X a Y, ani nekorespondování sekvencí X a Y. Zarovnání mezi vstupem X a výstupem Y je monotónní, rovněž se jedná zarovnání se vztahem N:1, což implikuje, že délka výstupu Y nemůže být delší než délka vstupu X [20]. Aby se byl CTC algoritmus schopný vypořádat se zdvojenými písmeny při dekódování, je představen speciální prázdný znak ϵ, který se mezi stejnými po sobě jdoucími písmeny musí nacházet vždy minimálně jednou.

Pravděpodobnost výstupu Y pro vstup X se získá sečtením pravděpodobností všech validních zarovnání mezi vstupem X a výstupem Y podle následujícího vztahu

$$p(Y \mid X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X). \tag{2.13}$$

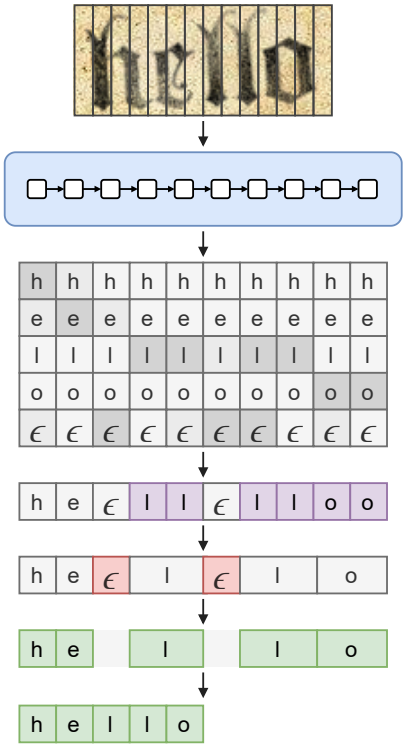
Při trénování neuronové sítě reprezentuje Y referenční přepis a výsledek vztahu 2.13 se používá pro úpravu vah v síti. Text řádku se z výstupu RNN, obsahujícího pravděpodobnosti jednotlivých znaků, získává následujícím vztahem

$$Y^* = \underset{Y}{\operatorname{argmax}}\, p(Y \mid X). \tag{2.14}$$

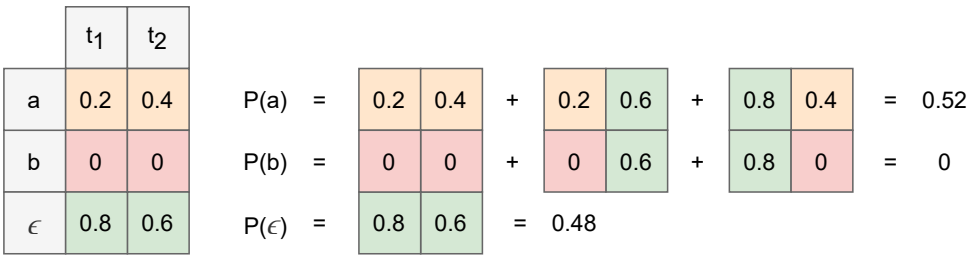
K dekódování nicméně můžou být použity různé heuristiky [20]. Nejjednodušší a nejrychlejší heuristika je hladové dekódování, kde se nejprve získá sekvence znaků s největšími pravděpodobnostmi, z té se následně odstraní stejné znaky jdoucí po ihned sobě a poté symbol prázdna ϵ [43]. Vizualizace procesu hladového dekódování je k dispozici na obrázku 2.6.

2.4 Prefixové paprskové dekódování

Hladové dekódování vždy nemusí vést k nejlepšímu možnému výsledku, příklad takové situace je k dispozici na obrázku 2.7. Alternativou k hladovému dekódování, která má ovšem značně větší časovou složitost, je prefixové paprskové dekódování [19]. Pseudokód prefixového paprskového dekódování je k dipozici v algoritmu 1.



Obrázek 2.6: Vizualizace transformace vstupní sekvence X na výstupní sekvenci Y. Vstupní sekvence je detekovaný a vyříznutý řádek s textem "hello". Ten je nejprve zpracován pomocí RNN (modrý blok), která pro jednotlivé úseky řádku spočítá pravděpodobnosti znaků. Čím tmavší je pozadí znaku, tím pravděpodobnější je jeho výskyt v daném úseku. CTC algoritmus pravděpodobnosti znaků prochází a z každého úseku vybere ten s největší pravděpodobností. Ze získané sekvence jsou sloučeny stejné znaky jdoucí ihned po sobě, které jsou vyznačeny fialově. Následně jsou odstraněny symboly ϵ vyznačené červeně a tím je získána výstupní sekvence Y - textový přepis obrázku. Převzato z práce [20].



Obrázek 2.7: Příklad situace, kdy hladové dekódování nevrátí nejpravděpodobnější řetězec. Z matice pravděpodobností mohou vzniknout 3 různé řetězce - a, b, ϵ. Hladové dekódování vybírá nejpravděpodobnější symboly na pozicích t_1 a t_2, přičemž v tomto případě je jeho výsledkem prázdný řetězec (nejpravděpodobnější symboly jsou ϵ a ϵ) s pravděpodobností $0.8 \cdot 0.6 = 0.48$. Pravděpodobnost řetězce b je 0, nicméně pravděpodobnost řetězce a odpovídá $(0.2 \cdot 0.4) + (0.2 \cdot 0.6) + (0.8 \cdot 0.4) = 0.52$ a řetězec a je tak nejpravděpodobnější variantou, přičemž ale hladové dekódování vrátilo jiný řetězec. Převzato z práce [42].

Algoritmus 1 Pseudokód prefixového paprskového dekódování. Převzato z práce [42].

```
Vstupem je matice logits, počet ponechávaných paprsků k a množina symbolů alphabet
Inicializuj množinu beams = {Ø}
Inicializuj score(Ø, 0) = 1
for t=1...T do
    bestB = bestBeams(beams, score, k)
    beams = {}
    for b in bestB do
        beams = beams ∪ b
        score(b, t) = calculateScore(logits, b, t)
        for c in alphabet do
            b' = b + c
            score(b', t) = calculateScore(logits, b', t)
            beams = beams ∪ b';
        end for
    end for
end for
```

Prefixové paprskové dekódování iteruje přes jednotlivé pozice v matici pravděpodobnosti *logits* získané z výstupu neuronové sítě. V každém kroku ponechává k nejpravděpodobnějších hypotéz (paprsků) z kroku předchozího. Pro tyto hypotézy počítá pravděpodobnost v kroku aktuálním. Zároveň každou z těchto hypotéz rozšiřuje o všechny možné znaky z výstupu neuronové sítě a pro nově vzniklé hypotézy počítá jejich pravděpodobnost. Z této množiny hypotéz je v dalším kroku algoritmu opět ponecháno jen k nejpravděpodobnějších, které jsou dále prozkoumávány. Jazykový model může být požitý jako součást výpočtu skóre jednotlivých hypotéz.

2.5 Jazykové modely

Jazykový model je funkce, která zachycuje pravděpodobnostní rozložení sekvence slov v přirozeném jazyce [3]. V zásadě lze jazykové modely rozdělit na dva typy. Prvním typem jsou modely operující nad statistickými daty jako N-gramy nebo Skryté Markovovy modely a druhým typem jsou neuronové sítě, které využívají distribuovaných reprezentací [41].

N-gramy jsou neparametrické modely, jejichž princip je založený na učení sekvencí slov a jejich frekvenci, přičemž se může jednat o sekvence různých délek (např. 2 - bigram, 3 - trigram) [3]. Parametrické modely se od neparametrických liší v počtu parametrů, zatímco parametrické modely mají předem daný počet parametrů, počet parametrů neparametrických modelů se zvětšuje s množstvím trénovacích dat [40]. N-gram model predikuje pravděpodobnost výskytu slova w v kontextu historie h (předchozích slov) $p(w|h)$, kde historie h n-gramového modelu má délku $n-1$ slov [41]. Řetízkové pravidlo pravděpodobnosti umožňuje spočítat společnou pravděpodobnost sekvence slov za pomocí podmíněných pravděpodobností předchozích slov podle vztahu

$$p(w_1...w_n) = p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)...p(w_n|w_1...w_{n-1}). \tag{2.15}$$

Problémem je přístup k podmíněným pravděpodobnostem až k $n-1$ slovu a proto se zavádí Markovův předpoklad

$$p(w_n|w_1...w_{n-1}) = p(w_n|w_{n-1}), \tag{2.16}$$

že výskyt slova w_n je závislý pouze na předchozím slovu w_{n-1} [41].

Nejmodernější jazykové modely jsou tvořené LSTM neuronovými sítěmi [51], které využívají schopnosti naučit se distribuované reprezentace a redukovat tak prokletí dimenzionality. Prokletí dimenzionality je jev, kdy obtížnost učení roste exponenciálně s počtem dimenzí [62]. V kontextu jazykových modelů je problémem velké množství možných sekvencí slov. Například pro sekvenci 10 slov a slovníku obsahujícího 100 000 slov existuje 10^{50} možných kombinací. Distribuovaná reprezentace slova je vektor hodnot reprezentující vlastnosti, které charakterizují význam slova a vzájemně se nevylučují. Neuronová síť se tyto vlastnosti učí v podobě spojitých hodnot [3].

Základní myšlenkou jazykových modelů tvořených neuronovými sítěmi je asociovat každé slovo ve slovníku s jeho reprezentací vektorem vlastností. Každé slovo tak odpovídá bodu ve vektorovém prostoru, přičemž se podobná slova nacházejí blíže k sobě. Výhoda využití distribuovaných reprezentací spočívá v tom, že model dokáže dobře generalizovat sekvence, které nejsou v trénovací sadě, ale jsou podobné vlastnostmi, tedy jejich distribuovanou reprezentací. Díky množství kombinací hodnot vektoru vlastností může být i velký dataset reprezentován kompaktně modelem s relativně malým množstvím trénovacích parametrů [3]. Jazykové modely tvořené LSTM neuronovými sítěmi pak mohou být trénovány kromě sekvence slov i na sekvenci znaků [18]. To poskytuje určitou výhodu ve zmenšení vektorového prostoru, jelikož znaků je mnohem méně než slov.

Kvalitu jazykového modelu je možné hodnotit vnitřními a vnějšími metrikami [8]. Vnitřní metrika je taková, která měří kvalitu modelu nezávisle na aplikaci. Příkladem vnitřní metriky je perplexita jazykového modelu. Ta ukazuje, jak přesně model predikuje model slova v sekvenci, které dříve neviděl. Perplexita jazykového modelu se odvíjí od korpusu, na kterém je počítána, tedy na různých korpusech mohou modely dosahovat odlišných nekorespondujících výsledků [8]. Vnější metrika slouží k vyhodnocení kvality modelu na konkrétní úloze pro níž byl vytvořen [28]. Příkladem v oblasti OCR je spuštění neuronové sítě se dvěma různými jazykovými modely a porovnání přesnosti přepisů upravených jazykovým modelem s referenčními přepisy.

Kapitola 3

Využití neanotovaných dat

V literatuře se objevuje několik metod využívající neanotovaná data pro trénování OCR. Většina metod spadá do kategorie algoritmů kombinující učení s učitelem a učení bez učitele (semi-supervised learning). Obecně tyto metody iterativně trénují OCR neuronovou síť na strojových přepisech neanotovaných dat vytvořených stejnou OCR neuronovou sítí/sítěmi v předchozí iteraci. Jednotlivé metody se liší hlavně v procesu generování strojových přepisů a způsobu výběru konkrétních neanotovaných dat, ze kterých se OCR neuronová síť má učit.

Následující kapitola popisuje dva nejvýznamnější směry ve využití neanotovaných dat pro trénování OCR a to co-training [6] a self-training [44]. V co-trainingu jsou trénovány dva modely, přičemž první model vytváří strojové anotace pro druhý model a ten vytváří strojové anotace pro první model. V případě self-trainingu je trénován pouze jeden model, který vytváří strojové anotace sám pro sebe.

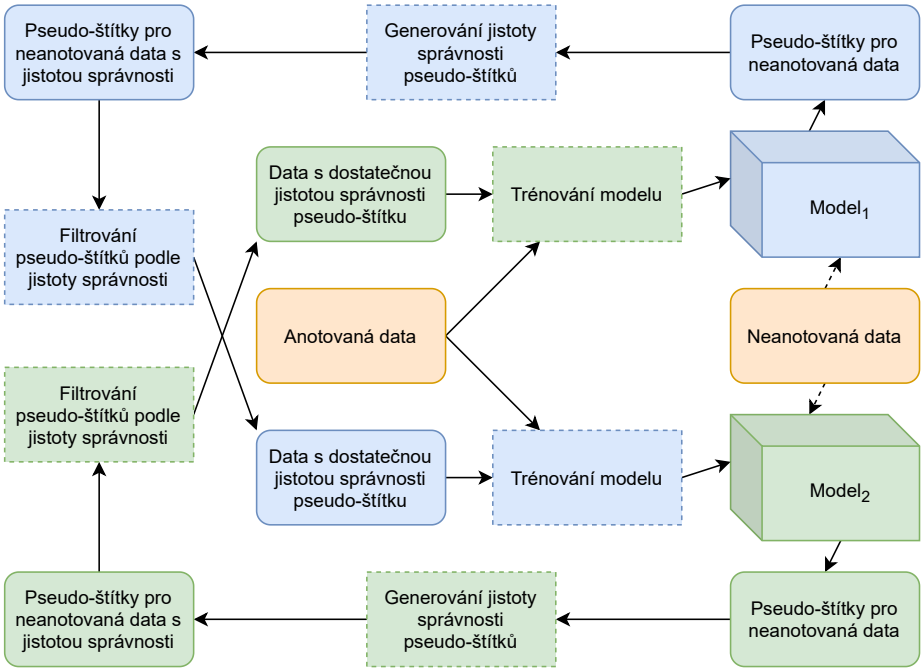
3.1 Co-training

Co-training [6, 16] je metoda, která k učení umožňuje využít velké množství neanotovaných dat bez potřeby lidského vstupu. Metoda využívá dva modely, které mají na začátku trénování nízkou úspěšnost, přičemž první model anotuje data pro druhý model a naopak. Vizualizace procesu co-trainingu je k dispozici na obrázku 3.1.

Prerekvizita úspěšného co-trainingu je podmíněně nezávislý pohled na data, tedy znaky pro určitou třídu prvního modelu nesmějí korelovat se znaky druhého modelu pro tutéž třídu. Pokud je tato podmínka splněna, může být pro trénování použito neomezené množství neanotovaných dat, což vede k postupnému snižování klasifikační chyby až k Bayesovu riziku [6]. V praxi je ovšem nezávislý pohled na data nepravděpodobný a při co-trainingu se používají jen rozdílné modely s vlastním a odlišným biasem [16]. V práci [16] je pak k vytvoření OCR systému pomocí co-trainingu použita BLSTM neuronová síť (vylepšená obousměrná varianta LSTM) jako první model a Skrytý Markovův model (dále jen HMM) jako druhý model.

Výpočet jistoty pseudo-štítku

Co-training pro trénování modelů využívá pseudo-štítky s největší jistotou. V práci [16], zabývající se trénováním OCR, se počítá jistota výstupů HMM odlišně od jistoty výstupů neuronové sítě. Důležité je pak zmínit, že experimenty v této práci byly prováděny na slovech. Stanovení jistoty pseudo-štítku se dělí na dvě fáze. Právě v první fázi se výpočet



Obrázek 3.1: Vizualizace trénování OCR pomocí co-trainingu. Oranžovou barvou jsou označena dostupná data. Modrou barvou je označen první model a strojové anotace, které produkuje. Zelenou barvou je označen druhý model a strojové anotace, které produkuje. Převzato z práce [16].

jistoty výstupu HMM liší od výpočtu jistoty výstupu neuronové sítě, v druhé fázi je výpočet jistoty stejný. U HMM je předběžná hodnota jistoty pseudo-štítku spočtena jako součet pravděpodobností dvou nejpravděpodobnějších strojových přepisů x namapovaný mezi 0 a 1 podle monotónní funkce [5]

$$conf_{prel} = 1 - (1 + \frac{x}{f})^{-1}, \tag{3.1}$$

kde $conf_{prel}$ je předběžná jistota pseudo-štítku (predikovaného slova) w v aktuální fázi výpočtu a f je normalizační konstanta (celkový počet kandidátů). V případě neuronové sítě je pro výpočet jistoty využita variabilita jejích vah způsobená náhodnou inicializací sítě. V první fázi výpočtu jistoty výstupu neuronové sítě se postupuje tak, že je současně trénováno n neuronových sítí, nejlepší síť je určena podle úspěšnosti na validační datové sadě a podíl neuronových sítí, které mají shodný výstup s nejlepším modelem, se bere jako předběžná jistota výstupu v první fázi výpočtu.

Druhá fáze výpočtu je stejná pro oba modely a obohacuje hodnotu jistoty o informace z fungování modelů na validační datové sadě. Nechť $c \in \{0, 1\}$ je správnost rozpoznání (0 reprezentuje nesprávnost, 1 reprezentuje správnost), pak $p_1(c|conf_{prel})$ reprezentuje pravděpodobnost rozpoznání slova w za předpokladu, že odhad jistoty $conf_{prel}$ byl správný. Dále pak $p_2(c|conf_{prel}, w)$ bere v potaz ještě pravděpodobnost správného rozpoznání slova w na validační datové sadě. Aby byl výpočet p_2 dostatečně robustní, je zapotřebí minimální počet výskytů slova w ve validační datové sadě θ_v. Pokud není tento požadavek splněn, je jistota určena jako p_1. Pokud nelze jistotu určit ani jako p_1, tak je jako hodnota jistoty použita její předběžná hodnota $conf_{prel}$ [16].

V práci [16] autoři využívají 3 různé prahy jistot pseudo-štítků, při jejichž překročení se data používají pro trénování. Při použití vysokého prahu byla do trénovací sady přidána strojově anotovaná data s jistotou 1. Pro střední práh byla do trénovací sady přidána data, které měla větší pravděpodobnost správnosti než výskytu chyby a pro nízký práh s hodnotou 0 a byla pro trénování použita všechna data. Experimentálně pak bylo autory dosaženo statisticky významného zlepšení úspěšnosti modelů při použití neanotovaných dat. Za klíčový parametr co-trainingu označili autoři práh jistot. Pokud byl nastavený příliš vysoko, modely se trénovaly s minimem přidaných neanotovaných dat a při jeho nastavení příliš nízko se modely učily na strojově anotovaných datech s chybami. První případ pak samozřejmě ke zlepšení úspěšnosti trénovaných modelů nevedl a druhý případ vedl v důsledku vzniku šumu dokonce ke zhoršení jejich úspěšnosti.

3.2 Self-training

Self-training [9, 44] se od co-trainingu liší trénováním jediného modelu, který má za úkol zlepšit svoji úspěšnost, na rozdíl od dvojice modelů trénovaných v co-trainingu, které se zlepšují navzájem. Kvůli své přímočarosti je využíván častěji než co-training, díky čemuž lze jako self-training označit vícero metod. Ty se od sebe více či méně odlišují způsobem generování pseudo-štítků nebo výběrem strojově anotovaných dat do trénovací sady. Práce [9] definuje self-training následovně. Mějme m anotovaných dat (L) a n neanotovaných dat (U) tak, že $n \gg m$. Model M_0 je natrénován na anotovaných datech L. Následně je M_0 využit ke generování pseudo-štítků pro neanotované data U. Neanotovaná data U s nejjistějšími pseudo-štítky, jejichž počet je p, jsou přimíchány k anotovaným datům L. Následně je na $m + p$ trénován model M_1. Tento proces je opakován tak, že na konci každého cyklu vznikne model M_{i+1}, který je pak využit k vygenerování $n - p$ pseudo-štítků dosud neanotovaných dat U. Self-training končí v okamžiku, kdy byla k trénování použita všechna neanotovaná data nebo se model trénováním dále nezlepšuje.

V následujícím textu se věnuji různým přístupům self-trainingu k výběru strojově anotovaných dat pro trénování. Rozebírám neduhy, kterými mohou natrénované modely trpět a řešení těchto problémů. Podrobněji se věnuji dvěma konkrétním self-training algoritmům zabývajících se trénováním OCR - anyOCR a AT-ST. Metoda anyOCR [24] je zajímavá naprostou minimalizací lidského vstupu pro natrénování OCR neuronové sítě. Metoda AT-ST [2] je jedním z nejmodernějších přístupů využívající neanotovaná data pro trénování OCR a v kapitole 6 věnující se experimentům využívám její úspěšnost jako referenční hodnotu. V závěru kapitoly se věnuji přehledu nejmodernějších self-training přístupů v oblasti klasifikace obrazu, které ještě nebyly řádně aplikovány na trénování OCR.

Výběr pseudo-štítků pro trénování

Pseudo-štítky přirozeně obsahují chyby, tato skutečnost je důvodem přidávání jen určitého podílu nejjistějších strojově anotovaných dat do trénovací datové sady. V případě co-trainingu je odhad jistoty pseudo-štítku generovaného neuronovou sítí časově náročný proces, viz kapitola 3.1. Self-training metody zpravidla používají pro odhad jistoty pseudo-štítku, potažmo výběru strojově anotovaných dat k trénování, rychlejší přístupy.

Autoři [50] pro výběr konkrétních strojově anotovaných dat do trénovací datové sady využívají slovníkovou verifikaci. Ta akceptuje pseudo-štítek, pokud jeho obsah náleží do slovníku, jinak ho odmítne. Použití slovníkové verifikace má za následek odmítnutí většího množství pseudo-štítků, na druhou stranu má výhodu jejich nízké chybovosti.

V práci [15] využívají autoři kromě tří statických prahů jistoty, které jsou stejné s těmi popsanými v kapitole 3.1, čtvrtý dynamický. Ten v průběhu trénování přidává strojově anotovaná do trénovacích dat podle pravděpodobnosti určené jistotou pseudo-štítku. Modely trénované s použitím tohoto prahu dosahovaly vyšší úspěšnosti než modely trénované s požitím vysokého a středního prahu. Autoři rovněž uvádí, že experimenty na datech, která měli k dispozici, ukazují, že pro úspěšnost cílových modelů byla důležitější kvantita strojově anotovaných dat než jejich kvalita. Nejlepších výsledků totiž dosahovalo trénování s nízkým prahem, který do trénovací sady přidával všechny strojově anotovaná data.

Práce [29] rozebírá možnost získání strojových anotací z datových sad, jejichž součástí jsou přepsané dokumenty na úrovni stránek. OCR neuronová síť je v této práci použita ke zpracování obrazové části datové sady. Právě na její výstup je poté mapován text z úrovně stránek a takto strojově anotovaná data jsou použita k přetrénování modelu. Rovněž autoři této práce prozkoumávají možnost korekce pseudo-štítků jazykovým modelem.

Konfirmační zkreslení a přetrénování modelů

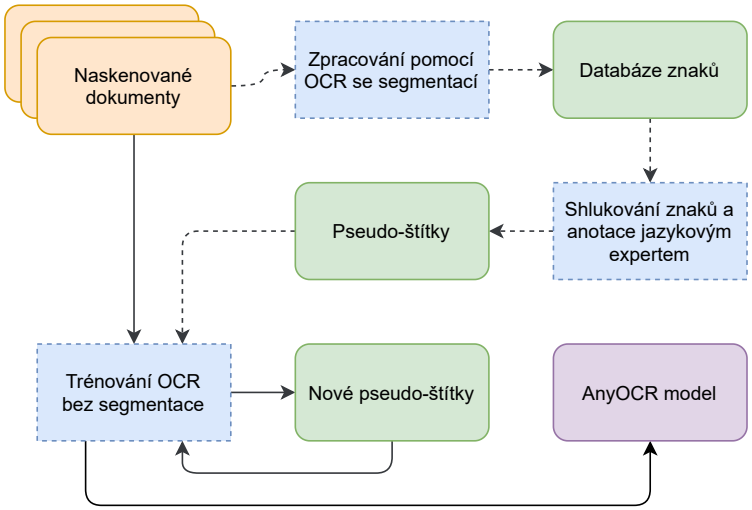
Jednou z nevýhod modelů trénovaných self-trainingem je jejich konfirmační zkreslení. Tento jev nastává při trénování na pseudo-štítcích, které obsahují podobné chyby. Díky konzistenci v chybách model získává nežádoucí jistotu na chybných datech a tím ztrácí schopnost učit se a opravovat své chyby [9]. Modely trénované self-training přístupem jsou rovněž náchylnější k přetrénování [58, 61]. Přetrénovaný model je příliš přizpůsobený trénovacím datům a při použití na jiné datové sadě dosahuje nižší úspěšnosti kvůli neschopnosti generalizace.

K zabránění konfirmačního zkreslení a přetrénování modelů se do procesu trénování přidávají různé druhy regularizace. Základním regularizačním prvkem trénování je augmentace dat. Obraz vstupních řádků může být upraven pomocí geometrických transformací, změny barev, rozostření nebo přidáním šumu [58]. Relativně novými metodami augmentací dat jsou maskovací augmentace [2] podrobněji popsána v kapitole 3.2.2, regularizace konzistencí [4, 47] podrobněji popsána v kapitole 3.2.3 a technika MixUp [59] podrobněji popsána v kapitole 3.2.3. Dalším způsobem regularizace je použití dropout vrstev v architektuře neuronové sítě [9]. Dropout vrstva [49] při trénování neuronové sítě náhodně přerušuje přenos informací některými neurony a snižuje tak riziko přetrénování. Jiným způsobem regularizace přímo v architektuře neuronové sítě je stochastická hloubka [23]. Ta během trénování vypouští určité vrstvy neuronové sítě a jejich místo nahrazuje funkcí identity. Použití stochastické hloubky díky snížení počtu trénovaných vrstev snižuje i celkový čas potřebný k natrénování sítě.

3.2.1 Algoritmus anyOCR

Algoritmus anyOCR [24] je příkladem konkrétní metody self-trainingu, která se snaží o největší možnou minimalizaci lidské práce v rámci anotace dat. Lidský vstup je zapotřebí jen pro přiřazení Unicode znaků shlukům symbolů. Vizualizace procesu trénování je k dispozici na obrázku 3.2. Základním prvkem metody je využití kombinace segmentačního OCR systému s OCR systémem bez segmentace (LSTM neuronová síť).

Algoritmus z naskenovaných dokumentů nejdříve extrahuje řádky, z nich následně segmentací získá jednotlivé symboly. Ty jsou shlukovány do stejných tříd pomocí modifikované varianty algoritmu K-means, kterou autoři označují jako Iterativní K-means. Původní algoritmus K-means iterativně shlukuje množinu N vektorů do k tříd [56]. V případě symbolů Iterativní K-means shlukuje vektory obsahující binarizovaná obrazová data jednotlivých detekovaných znaků o velikosti 32x32 pixelů, které jsou vycentrované na střed. Iterativní K-



Obrázek 3.2: Vizualizace self-training algoritmu AnyOCR. Přerušované šipky znázorňují inicializační část trénování, která se provádí pouze jednou. Celé šipky znázorňují cyklus trénování, který se opakuje po dobu zlepšování trénovaného modelu. Převzato z práce [24].

means je plně automatizovaná varianta původního algoritmu, která dosahuje při shlukování písmen z historických dokumentů lepších výsledků [24]. Každá iterace Iterativního K-means algoritmu je zakončena výpočtem rozmazanosti průměrného obrazu každého shluku. Pokud je shluk příliš rozmazaný, je v další iteraci shlukování provedeno znovu. Nové k je určeno jako počet překrývajících se tvarů.

Po úspěšném vytvoření shluků jednotlivých znaků je zapotřebí jediný lidský vstup v kontextu anotování dat. Jazykový expert je přizván pro přiřazení Unicode znaků jednotlivým shlukům. Díky pozicím známým ze segmentace jednotlivých písmen je nyní algoritmus schopný generovat pseudo-štítky a od této chvíle už se jedná o klasičtější self-training metodu, kde OCR tvořené LSTM neuronovou sítí iterativně generuje pseudo-štítky samo pro sebe do dalších iterací trénování.

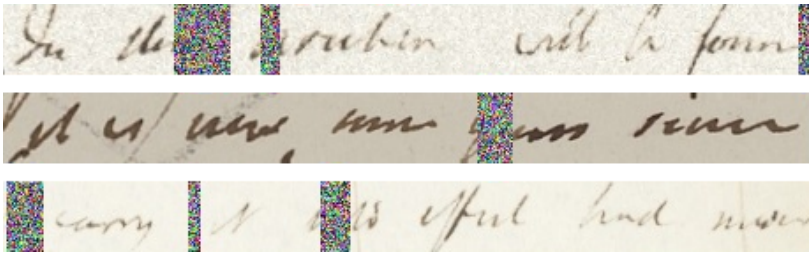
3.2.2 Algoritmus AT-ST

Autoři AT-ST [2] se kromě zvýšení úspěšnosti OCR neuronové sítě věnují také zvýšení úspěšnosti LSTM jazykového modelu a následně je využívají v kombinaci v rámci jednoho OCR systému k dosažení, co možná nejlepších výsledků.

Autoři experimentují s několika metrikami výpočtu jistoty pseudo-štítku, přičemž nejlepších výsledků dosáhli s jeho posteriorní pravděpodobností. Tu určují pomocí prefixového paprskového dekódování tak, že pro každý řádek l vygenerují k hypotéz $H = \{h_1, h_2 ... h_k\}$ spolu s jejich nenormalizovanými logaritmickými pravděpodobnostmi $C(h_i)$ a tyto hodnoty normalizují pomocí vzorce

$$M_{posterior}(l) = P(h_g|l) = \frac{exp(C(h_g))}{\sum_{i=1} exp(C(h_i))}, \tag{3.2}$$

kterým získají posteriorní pravděpodobnosti jednotlivých hypotéz H řádku l. Do trénovací sady poté přidávají 1 %, 3 %, 10 %, 32 %, 56 % a 100 % nejjistějších strojově anotovaných dat a nejúspěšnější model na validační datové sadě považují za cílový.



Obrázek 3.3: Ukázky maskovací augmentace.

Autoři rovněž experimentují s regularizační technikou maskovací augmentace, která vychází z metody Cutout [10]. Ta se používá ke zvýšení úspěšnosti konvolučních neuronových sítí zejména na klasifikačních úlohách a to pomocí odstranění kontinuálních částí obrazu. Samotný Cutout už byl ale použit i při trénování OCR v práci [12], kde rovněž zvýšil úspěšnost trénovaných modelů. Maskovací augmentace místo odstranění části vstupního obrazu nahrazuje vybranou oblast řádku šumem generovaným z rovnoměrného rozložení pravděpodobnosti, zatímco referenční přepis zůstává stejný. Autoři [2] uvádějí, že použití šumu místo konstantní barvy při maskování pomáhá neuronové síti odlišit prázdné místa od maskovaných míst a tím snížit komplexitu trénování. Ukázka maskovací augmentace je k dispozici na obrázku 3.3.

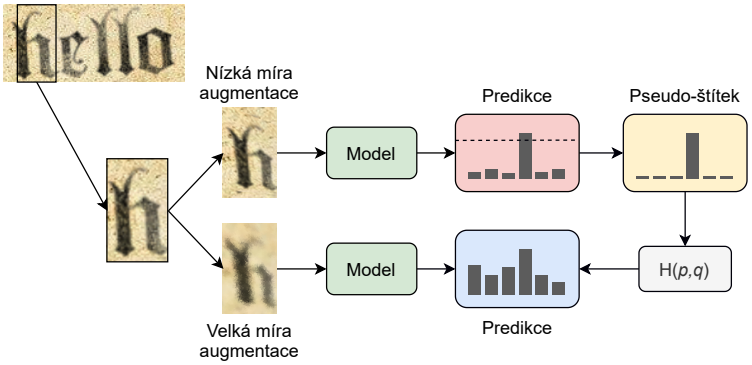
3.2.3 Self-training algoritmy používané pro klasifikaci

Použití self-training algoritmů ve spojení s OCR je v literatuře zatím relativně vzácné. Většina stávajících self-training algoritmů demonstruje svůj přínos na klasifikačních úlohách. Koncepty generování pseudo-štítků a určování jejich jistoty jsou ale přenositelné napříč různými úlohami a mohou tedy být použity i pro trénování OCR. Obrázky 3.4 a 3.5 demonstrují právě způsoby, kterými by popisované algoritmy mohly být využity pro trénování OCR pomocí chybové funkce rámcové křížové entropie. Tento způsob trénování je popsán podrobněji v kapitole 4.3.

Algoritmus FixMatch

FixMatch [47] je algoritmus spadající do kategorie self-trainingu, mezi jehož nejvýznamnější prvky patří augmentace dat při vytváření pseudo-štítků. Vizualizace návrhu použití algoritmu FixMatch pro trénování OCR je k dispozici na obrázku 3.4. Postup vytváření pseudo-štítků neanotovaných dat algoritmem FixMatch je následující. Algoritmus nejprve provede slabou augmentaci vstupního obrazu a následně ho zpracuje modelem. Výstup modelu tvoří pravděpodobnostní rozložení, které charakterizuje náležitost vzorku x do jednotlivých tříd. Pokud je pravděpodobnost určité třídy větší než stanovený práh, tak je z výstupu modelu vytvořen pseudo-štítek v podobě one-hot kódu. Následně algoritmus aplikuje silnou augmentaci na stejný řádek a spolu s vygenerovaným pseudo-štítkem ho zařadí do trénovací datové sady.

Algoritmus při procesu vytváření pseudo-štítků používá regularizaci konzistencí. Jedná se o regularizační techniku augmentace dat, která je využívána v nejmodernějších semi-supervised algoritmech. Cílem regularizace konzistencí je stejná klasifikace neaugmentova-



Obrázek 3.4: Vizualizace návrhu trénování OCR pomocí algoritmu FixMatch a rámcové křížové entropie. $H(p, q)$ je značení křížové entropie mezi pravděpodobnostními rozloženími p a q, které reprezentují výstup neuronové sítě a náležející pseudo-štítek. Převzato z práce [47].

ného neanotovaného vzorku x a augmentovaného neanotovaného vzorku $\alpha(x)$ [4] následovně

$$\sum_{i=1}^{n} \| \, p_m(y \mid \alpha(x_i)) - p_m(y \mid \alpha(x_i)) \, \|_2^2, \tag{3.3}$$

kde n je počet vzorků v trénovacím vzorku, x_i je i-tý vzorek v trénovacím vzorku, α je funkce slabé augmentace a p_m je pravděpodobnostní příslušnost vzorku x_i k jednotlivým třídám modelu m. Jak α, tak p_m jsou stochastické funkce, z čehož plyne, že se od sebe výrazy budou lišit. Autoři [47] implementovali algoritmus FixMatch pro klasifikační úlohu, při trénovaní se tak kromě chybové funkce 3.3 využívá křížová entropie

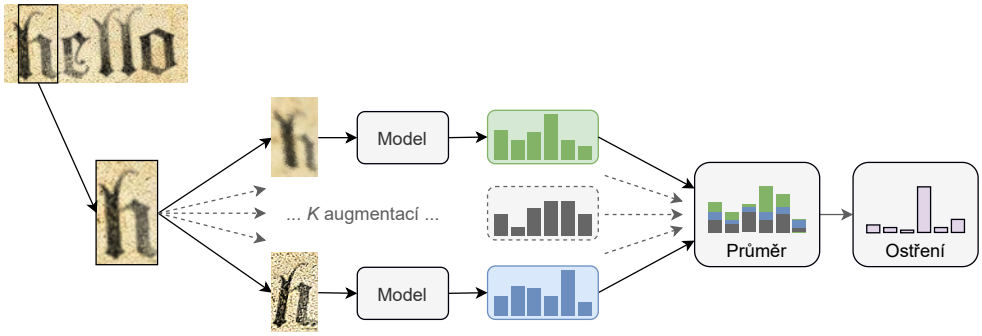
$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i), \tag{3.4}$$

kde n je počet tříd, t_i je pseudo-štítek a p_i je normalizovaná pravděpodobnost i-té třídy [27]. Využití křížové entropie není nutně překážkou pro použití algoritmu FixMatch k trénování OCR. I přes dnes standardní používání chybové funkce CTC pro trénování OCR neuronové sítě, autoři [30] použili rámcovou křížovou entropii bez většího poklesu úspěšnosti trénovaného modelu.

Originální FixMatch algoritmus při trénování využívá tvrdé pseudo-štítky (one hot kód). V oblasti rozpoznání řeči ale autoři [38] přišli i s verzí algoritmu, která pracuje s měkkými pseudo-štítky. Experimentálně pak ukázali, že tento přístup je výhodnější než původní přístup s tvrdými štítky, jelikož je schopný eliminovat chyby v pseudo-štítcích vzniklé na nejednoznačných místech.

Algoritmus MixMatch

Algoritmus MixMatch [4] má některé prvky společné s algoritmem FixMatch, zásadně se ale liší ve způsobu generování pseudo-štítků. Vizualizace generování pseudo-štítků je na obrázku 3.5. Pro každý neanotovaný vzorek x se vygeneruje K jeho augmentovaných verzí. Tyto verze jsou následně zpracovány modelem, z jehož výstupu algoritmus dostane pravděpodobnostní rozložení příslušnosti vzorku do jednotlivých tříd. Pravděpodobnostní roz-



Obrázek 3.5: Vizualizace generování pseudo-štítků algoritmem MixMatch pro trénování rámcovou křížovou entropií. Převzato z práce [4].

ložení příslušnosti jednotlivých augmentací vzorku se následně průměrují. Průměr pravděpodobnostního rozložení je dále zaostřen pomocí vztahu

$$S(p,T)_i = p_i^{\frac{1}{T}} / \sum_{j=1}^{L} p_j^{\frac{1}{T}}, \tag{3.5}$$

kde p je průměrné pravděpodobnostní rozložení všech augmentací vzorku x a T je hyperparametr teploty. Teplota ostření udává, jak moc budou štítky tvrdé. Při teplotě $T \to 0$ se pseudo-štítky blíží tvrdé variantě [4].

Dále při trénovaní využívá algoritmus MixMatch regularizační techniku učení MixUp [59], kterou v následujících řádcích demonstruji na datové sadě pro trénování OCR. MixUp z trénovací datové sady vezme dva náhodné řádky x_i, x_j a jejich přepisy y_i, y_j a z jejich kombinace vytvoří nový řádek a jemu odpovídající přepis, který se v datové sadě nenachází, následovně

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \tag{3.6}$$

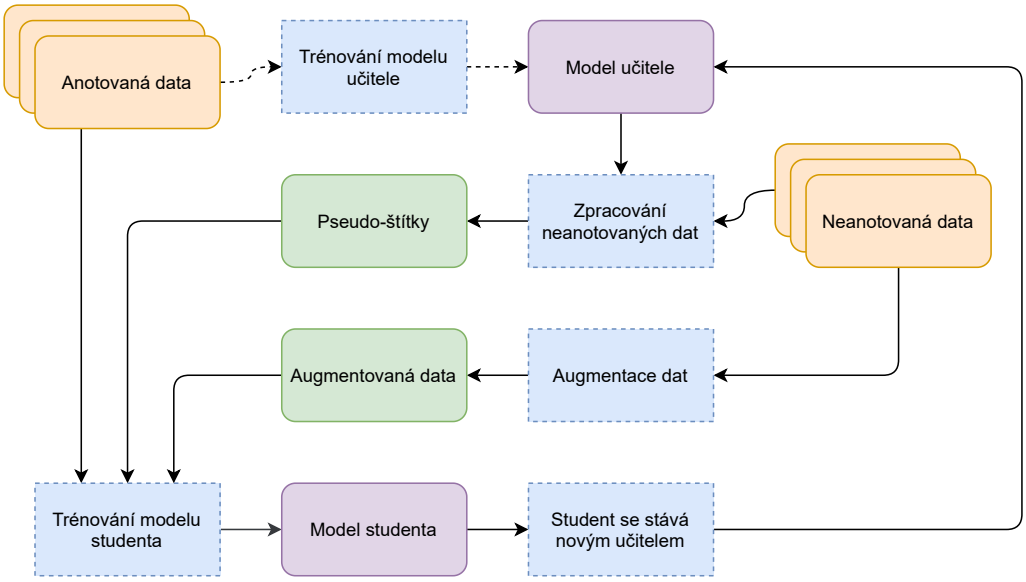
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j. \tag{3.7}$$

Parametr λ určuje poměr původních řádků při tvorbě nového a získává se z beta rozložení $Beta(\alpha, \alpha)$, kde $\alpha \in (0, \infty)$. MixUp přispívá k regularizaci neuronové sítě a její odolnosti vůči učení se z chybných štítků [59]. Technika MixUp byla originálně představena pro trénování klasifikačních neuronových sítí, autoři [35] ale techniku úspěšně adaptovali i na učení OCR s chybovou funkcí CTC.

Algoritmus Noisy Student

Algoritmus Noisy Student [58] je zajímavý kombinací dvou věcí - používáním několika zdrojů regularizace a faktu, že model studenta je větší než model učitele. Trénování modelu studenta natrénovaným modelem učitele je relativně běžná záležitost. Většinou je ale model studenta menší než model učitele a trénování probíhá s cílem vytvoření rychlejšího modelu, jedná se o koncentrování znalostí. Algoritmus Noisy Student cílí naopak na expanzi znalostí, a to pomocí úspěšného modelu učitele, který vytváří pseudo-štítky neanotovaných dat, jež jsou při trénování zároveň augmentována.

Vizualizace průběhu algoritmu je k dispozici na obrázku 3.6. Algoritmus začíná trénováním prvního modelu učitele na anotovaných datech L. Natrénovaný model učitele je následně použit k vygenerování pseudo-štítků neanotovaných dat U. Algoritmus je při tom



Obrázek 3.6: Vizualizace trénování neuronové sítě algoritmem Noisy Student. Přerušované linky znázorňují kroky algoritmu, které probíhají pouze během první iterace algoritmu.

nezávislý na konkrétní podobě pseudo-štítků, ty mohou být tvrdé i měkké. Následně jsou sloučena data L s daty U a trénuje se na nich nový model studenta, který je buď stejně velký nebo větší než model učitele. Natrénovaný student se stává novým učitelem a generuje nové pseudo-štítky pro nového studenta [58].

Regularizace je v průběhu trénování aplikována jak na trénovací data, tak na model samotný. Trénovací data jsou augmentována geometrickými transformacemi a přidáním šumu. Model je regularizován pomocí dropout vrstev a stochastické hloubky.

Kapitola 4

Návrh řešení

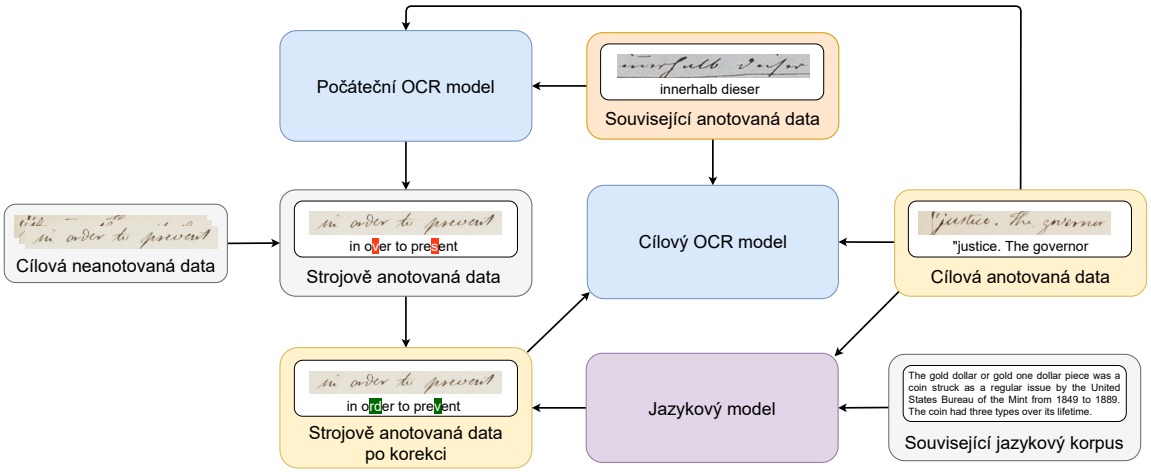
Ve své diplomové práci jsem se v rámci zaměření na využití neanotovaných dat pro trénování OCR rozhodl orientovat na výzkum oblasti self-training algoritmů. Mnou navržené přístupy vychází z existujících algoritmů popsaných v kapitole 3 a pevně věřím, že je obohacují o nové myšlenky. Za klíčový prvek, který tato práce přináší, a který většina algoritmů v kapitole 3 postrádá, považuji využití jazykového modelu ke korekci pseudo-štítků. Druhým klíčovým prvkem této práce je pak využití měkkých pseudo-štítků, jejichž použití se v několika případech ukázalo jako benefitující úspěšnosti cílových modelů [38, 58]. V těchto případech se ale nejednalo o použití pro trénování OCR neuronové sítě.

Navrhovaná vylepšení nejsou závislá na konkrétní self-training metodě. Ve své práci jsem se je rozhodl integrovat do řešení podobného práci AT-ST [2], která patří k nejmodernějším přístupům využití neanotovaných dat pro trénování OCR. Za klíčovou součást, kterou mé řešení přebírá z AT-ST, považuji bezpochyby využití maskovací augmentace. Ta je důležitým regularizačním prvkem, který by teoreticky měl zabraňovat přetrénování sítě, což je jeden z problémů využití self-training metod [58, 61]. Použití maskovací augmentace by zároveň mělo vést ke zlepšení implicitně modelované jazykové znalosti OCR neuronové sítě. V článku AT-ST pak byla maskovací augmentace schopná zvýšit úspěšnost cílového modelu až o 10 %.

Navrhovaný přístup trénování je pak následující, mějme anotovaná data L_c a L_s. L_c jsou data cílová, na kterých chceme, aby model dosahoval optimální úspěšnosti a L_s jsou data související, která jsou do trénování přidávána proti přetrénování sítě na malém množství dat cílových. Dále mějme počáteční model M_p, který je trénován na datech $L_c \cup L_s$. Natrénovaný počáteční model M_p je použit pro vygenerování strojových anotací neanotovaných dat U. Na $L_c \cup L_s \cup U$ je pak natrénován cílový model M_c. Vizualizace navrhované metody, už s korekcí pseudo-štítků jazykovým modelem, je k dispozici na obrázku 4.1.

4.1 Korekce pseudo-štítků jazykovým modelem

Strojové anotace získané z počátečního modelu M_p přirozeně obsahují chyby. S nimi se v zásadě lze vyrovnat dvěma způsoby - použitím pouze nejjistějších strojově anotovaných dat s nejméně chybami nebo jejich korekce. Vybírání nejjistějších strojově anotovaných dat má hned několik úskalí. Prvním úskalím je netriviální výběr metriky pro výpočet jistoty. V předchozí kapitole bylo představeno hned několik takových metrik. Druhým úskalím je výběr podílu nejjistějších strojově anotovaných dat, protože i s ideální metrikou jistoty bude pro různé podíly dosahovat cílový model M_c různých úspěšností. V případě podílu obsahujícího jen nejjistější data se úspěšnost cílového modelu nemusí nijak výrazně zlepšit



Obrázek 4.1: Navržená metoda trénování s korekcí pseudo-štítků jazykovým modelem. Modrou barvou jsou označeny OCR modely, fialovou barvou je označený jazykový model.

a v případě příliš velkého podílu obsahujícího velké množství chyb se dokonce úspěšnost cílového modelu může zhoršit [16].

K dosažení ideálních výsledků navrhuji využít jazykový model při dekódování výstupu OCR neuronové sítě pomocí prefixového paprskového dekódování. Jazykový model umožňuje spočítat pravděpodobnost dané sekvence písmen v jazyce a právě tato pravděpodobnost může být využita při výpočtu pravděpodobnostního skóre jednotlivých paprsků (hypotéz) podle následujícího vzorce

$$\log S(a) = \log S_O(a) + \alpha \log S_L(a) + \beta|a|, \tag{4.1}$$

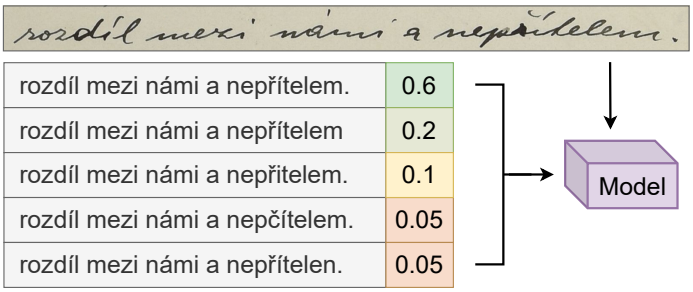
kde S_O je skóre počátečního modelu, S_L je skóre jazykového modelu, α je váha jazykového modelu, β je vkladový bonus a $|a|$ je délka hypotézy a v počtu znaků [2]. Vizualizace procesu trénování obohaceného o korekci pseudo-štítků jazykovým modelem je k dispozici na obrázku 4.1.

Hodnoty získané ze vztahu 4.1 pak mohou být použity pro výpočet posteriorních pravděpodobností jednotlivých hypotéz podle vztahu 3.2. Jazykový model tímto způsobem může být použit ke korekci pseudo-štítku, výpočtu jeho jistoty a tedy i rozhodnutí o jeho zařazení či nezařazení do trénovací sady.

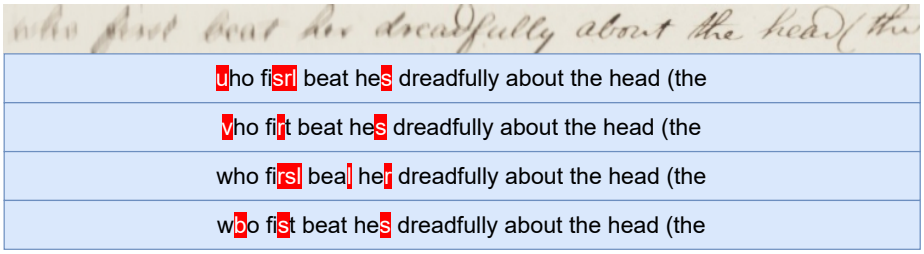
Práce [15] trénuje cílový model na strojově anotovaných řádcích, které jsou do trénovacích vzorků přidávány s pravděpodobností rovné jejich jistotě. Toto dynamické zařazování strojově anotovaných dat do trénovací datové sady je výhodné, protože není třeba stanovit pevný podíl strojově anotovaných dat, která se mají použít pro trénování. Zde bych chtěl navrhnout podobný přístup k trénování nevyžadující nastavení konkrétního prahu a to váhování gradientů podle jistoty odpovídajících pseudo-štítků. V praxi to znamená trénování cílového modelu na všech strojově anotovaných datech a násobení gradientů těchto dat jejich jistotou tak, že méně jisté pseudo-štítky budou mít menší vliv na úpravu vah neuronové sítě v průběhu trénování.

4.2 CTC a váhované varianty pseudo-štítků

Self-training metody obvykle při vytváření trénovací datové sady kladou důraz na výběr strojově anotovaných dat s nejjistějšími pseudo-štítky. Pseudo-štítky s nižší jistotou obsahují



Obrázek 4.2: Vizualizace trénování neuronové sítě chybovou funkcí CTC na měkkých pseudo-štítcích.



Obrázek 4.3: Ukázka řádku a jeho měkkého pseudo-štítku jehož chyby jsou lokalizované pouze v určité oblasti.

častěji chyby a jejich přítomnost v trénovací sadě může snížit úspěšnost trénované neuronové sítě [9]. Standardně jsou poté při trénování OCR neuronových sítí používány tvrdé pseudo-štítky. Jedná se pseudo-štítky s pravděpodobností 1, tedy případ kdy je vstupnímu řádku přiřazen jeden konkrétní přepis považovaný za správný [17]. Použití tvrdých pseudo-štítků u strojově anotovaných dat, která obsahují chyby, ale může snížit úspěšnost neuronové sítě, protože pseudo-štítky s nízkým počtem chyb mají při trénování stejnou váhu jako nejisté pseudo-štítky s vyšším počtem chyb [34].

Tento jev může pomoci odstranit trénování neuronové sítě s měkkými pseudo-štítky, viz obrázek 4.2. K jednomu trénovacímu řádku v tomto případě náleží několik nejpravděpodobnějších variant pseudo-štítku namísto původní jedné varianty s pravděpodobností 1 v případě tvrdých pseudo-štítků. Při trénování je pak gradient jednotlivých variant pseudo-štítku vynásobený jistotou varianty. Práce [38, 58] dokládají, že model trénovaný tímto způsobem může dosáhnout lepší úspěšnosti než při trénování na tvrdých pseudo-štítcích.

Benefit použití měkkých pseudo-štítků je rovněž demonstrován na obrázku 4.3. V okamžiku, kdy se neuronová síť učí na k variantách pseudo-štítku, kde je suma jejich pravděpodobností rovna 1, zůstává přepis v jistějších oblastech řádků stejný napříč variantami a jeho kumulativní pravděpodobnost je blízká 1. Zatímco v méně jistých oblastech řádku se varianty pseudo-štítku liší, tím vzniká šum a neuronová síť se tak primárně učí z jistějších oblastí. Trénováním na měkkých pseudo-štítcích ale model získává i relevantní informace o možném výskytu jiného než nejpravděpodobnějšího znaku. Příkladem jsou znaky, které jsou si podobné např. i, í.

V praxi navrhuji pro neanotovaný řádek generovat k nejpravděpodobnějších hypotéz pomocí prefixového paprskového dekódování využívajícího jazykový model a tyto hypotézy použít jako varianty měkkého pseudo-štítku. Posteriorní pravděpodobnosti jednotlivých variant pseudo-štítku pak navrhuji využít jako jejich jistotu (váhu), viz obrázek 4.2.

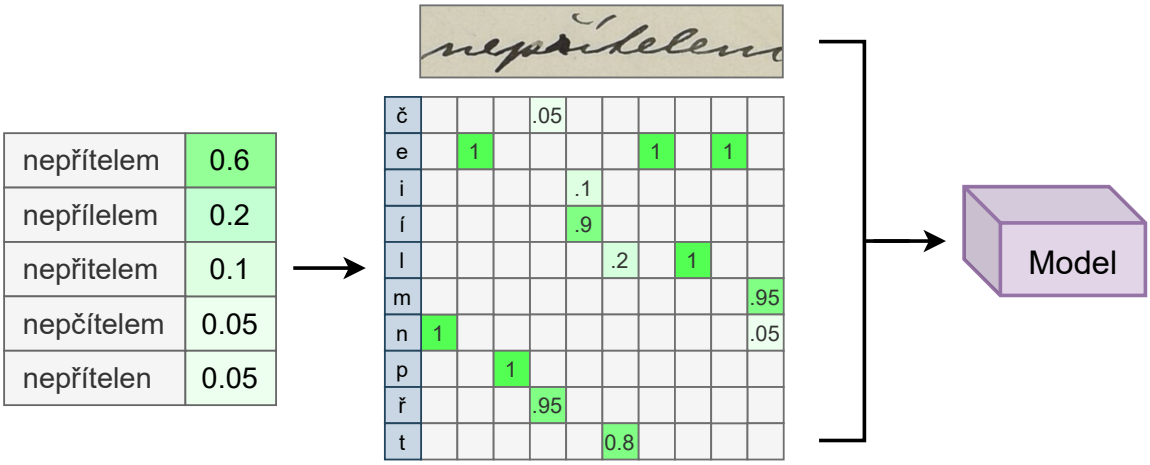
4.3 Rámcová křížová entropie

Alternativním přístupem k trénování neuronové sítě pomocí chybové funkce CTC je trénování pomocí rámcové křížové entropie [30]. Autoři [30] byli schopní experimentálně dosáhnout podobných výsledků jako při použití chybové funkce CTC, ale jejich práce se nezabývala self-training algoritmy a využitím měkkých pseudo-štítků. Trénování rámcovou křížovou entropií má nevýhodu nutné znalosti pozic jednotlivých znaků ve vstupním řádku. To je jedním z důvodů, proč se standardně používá chybová funkce CTC, která tuto znalost nevyžaduje. Varianty měkkého pseudo-štítku, vygenerované algoritmem popsaným v kapitole 4.2, navrhuji zarovnat na pozice ve vstupním řádku pomocí následujícího algoritmu.

Pseudo-štítek pro trénování s rámcovou křížovou entropií má podobu matice, kde osa y reprezentuje pravděpodobnost symbolu a a osa x představuje časovou osu vstupního obrazu diskretizovanou na rámce, viz obrázek 4.4. Matice tak obsahuje pro každý rámec pravděpodobnosti všech symbolů nacházejících se na výstupu OCR neuronové sítě. Pomocí prefixového paprskového dekódování a jazykového modelu je na základě výstupu OCR neuronové sítě vygenerováno k hypotéz. Každá hypotéza je zarovnána na výstup OCR neuronové sítě pomocí Viterbiho algoritmu [13]. To znamená, že každému symbolu hypotézy je přiřazen unikátní index popisující jeho pozici na ose x, tedy pozici na které se symbol nejpravděpodobněji nachází na základě výstupu OCR neuronové sítě. Pozice, kterým není v rámci jedné hypotézy přiřazen žádný symbol, je přiřazen prázdný symbol ϵ. Všechny zarovnané symboly (včetně ϵ) v rámci jedné hypotézy mají pravděpodobnost 1, ostatní symboly mají pravděpodobnost 0. Pravděpodobnost p_{xy} symbolu s indexem y pro rámec x v pseudo-štítku se počítá následovně

$$p_{xy} = \sum_{i=0}^{n} q_i * r_{ixy}, \tag{4.2}$$

kde q_i je posteriorní pravděpodobnost i-té hypotézy a r_{ixy} je pravděpodobnost y-tého znaku na pozici x-tého rámce i-té hypotézy. Příklad zarovnání hypotéz do matice pseudo-štítku je k dispozici na obrázku 4.4.



Obrázek 4.4: Vizualizace zarovnání hypotéz generovaných prefixovým paprskovým dekódováním a jazykovým modelem do matice pravděpodobností symbol-rámec, která se při trénování rámcovou křížovou entropií používá jako pseudo-štítek. V levé části obrázku jsou zobrazeny hypotézy a jejich posteriorní pravděpodobnosti, uprostřed je pak zobrazená matice v níž hypotézy zarovnané. Prázdné políčka v matici mají pravděpodobnost 0.

Kapitola 5

Implementace

Tématu diplomové práce jsem se věnoval v rámci spolupráce na projektu PERO[1]. Kód všech experimentů je psaný v programovacím jazyku Python 3. K jejich provedení jsem využil dva repozitáře spadající pod projekt PERO, jejichž jsem spoluautorem. Repozitář pero-ocr[2] je veřejně dostupný na webové platformě Github, která slouží k verzování kódu a zároveň je dostupný v rámci Pythonovského správce balíčků pip. Druhý použitý repozitář se jménem pero je soukromý a slouží pro interní výzkum a vývoj OCR softwaru před jeho nasazením do produkce. Neuronové sítě jsou implementovány v knihovně Pytorch[3], přičemž k trénování LSTM jazykového modelu jsem použil knihovnu BrnoLM[4].

Veškerá implementovaná funkcionalita v rámci diplomové práce byla přidána do repozitáře pero. Mnou implementovaný kód pak v zásadě lze rozdělit na dvě části - kód sloužící ke zpracování datových sad a kód zabývající se samotným trénováním. Skriptů sloužících k vytváření datových sad je několik a podporují tvorbu datové sady s vícero referenčními přepisy pro jeden řádek, kde každý přepis má svoji váhu a tvorbu datové sady s referenčními přepisy ve formě pravděpodobnostní matice pro trénování rámcovou křížovou entropií. Rovněž tyto skripty umožňují spojování existujících datových sad, duplikaci řádků anebo řazení a výběr řádků podle jejich váhy (jistoty).

V kontextu kódu týkajícího se samotného trénování jsem implementoval váhovanou CTC chybovou funkci a chybovou funkci rámcové křížové entropie, které v knihovně Pytorch chybí. Při trénování standardních CTC neuronových sítí pomocí nástrojů v repozitáři pero jsou důležitou součástí kódu třídy pracující s datovými sadami. Do těchto tříd jsem implementoval podporu pro práci s váhovanými referenčními přepisy. Tuto funkcionalitu rovněž využívá nový trénovací skript, který byl použit v experimentech s váhovanými pseudo-štítky a variantami pseudo-štítků. Kromě experimentů provedených v této práci je možné využít tuto novou funkcionalitu k doladění vah neuronových sítí na specifická data. Příkladem takového doladění je adaptace OCR na nová data dříve neviděného stylu. Stávající OCR neuronová síť může být přetrénována na původní trénovací datové sadě sloučené s nově dostupnými daty, kterým ale bude zvýšena váha (důležitost). Do tříd pracujících s datovými sadami v průběhu trénování jsem rovněž implementoval podporu pro práci s referenční přepisy ve formě matice pravděpodobnosti, viz kapitola 4.3. Pro práci s tímto formátem anotací jsem rovněž vytvořil samostatný trénovací skript. I přes nepřesvědčivé

[1] https://pero.fit.vutbr.cz/
[2] https://github.com/DCGM/pero-ocr
[3] https://pytorch.org/
[4] https://github.com/BUTSpeechFIT/BrnoLM

výsledky trénování rámcovou křížovou entropií dosažených v této práci, práce [30] dokládá její využitelnost v budoucím výzkumu.

Kapitola 6

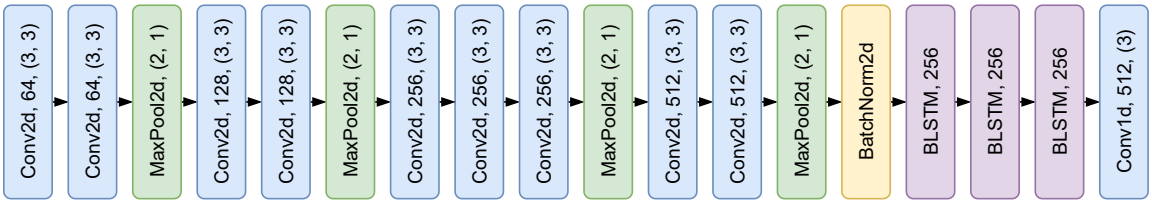
Experimenty

Všechny nově navržené metody využití neanotovaných dat pro trénování OCR jsem postupně experimentálně otestoval. Nejprve jsem se zabýval vlivem množství cílových anotovaných dat na úspěšnost modelu pro konkrétní datovou sadu, která byla použita ve všech experimentech. Díky tomu je v dalších experimentech zkoumána úspěšnost modelů na dvou trénovacích datových sadách, kde každá obsahuje různé množství cílových anotovaných dat. Následně jsem zkoumal vliv korekce pseudo-štítků jazykovým modelem na úspěšnost cílového modelu. Tento experiment jsem provedl pro různé podíly nejjistějších strojově anotovaných dat a z výsledků vyplývá, že korekce pseudo-štítků jazykovým modelem skutečně benefituje úspěšnosti cílových modelů. K dosažení nejlepších výsledků je ale třeba zvolit ideální podíl nejjistějších strojově anotovaných dat pro trénování, což je obtížné. Další experiment zkoumá přínos trénování neuronové sítě pomocí strojově anotovaných dat, která mají přiřazenou váhu podle jejich jistoty. Tento přístup je výhodný, protože není zapotřebí trénovat neuronovou síť na určitém idélním podílu nejjistějších strojově anotovaných dat. Další experimenty se pak věnují trénování neuronové sítě pomocí měkkých pseudo-štítků a to pomocí váhované CTC chybové funkce a rámcové křížové entropie. Byť většina cílových modelů trénovaných na měkkých pseudo-štítcích dosahuje značného zlepšení oproti počátečním modelům, nedosahují tyto modely takového zlepšení jako modely trénované na tvrdých pseudo-štítcích.

Výsledky všech experimentů porovnávám s prací AT-ST [2], která na totožných datech zkoumá využití tvrdých pseudo-štítků bez korekce jazykovým modelem. Díky provedení experimentů na stejných datech a se stejným nastavením lze výsledky přímo porovnat a úspěšnost metody AT-ST uvádím jako referenční výsledek, viz tabulka 6.4. Úspěšnost modelů pak udávám v metrice CER [31] (chybovost znaků), která se počítá následovně

$$\mathrm{CER} = \frac{S + I + D}{N} * 100, \tag{6.1}$$

kde N je počet znaků reference a S, I, D jsou počty znaků přepisu, které jsou změněné, vložené a smazané vůči referenci. Čím nižší je hodnota CER, tím úspěšnější je zkoumaný model. U všech modelů bylo k získání strojového přepisu pro výpočet chybovosti CER použito hladové dekódování. Kromě rychlostního benefitu je toto dekódování použito v AT-ST, a tak je vhodné jej rovněž využít kvůli přímé porovnatelnosti. Použitím prefixového paprskového dekódování, viz kapitola 2.4, by velmi pravděpodobně bylo dosaženo nižší chybovosti CER v obou pracích, ale vzhledem k nulovému efektu na trénování a tedy na kvalitu modelů samotných je volba dekódování pro porovnání nedůležitá. Opak platí pro generování pseudo-štítků, jejichž kvalita má na trénování a cílový model vliv a proto jsem pro jejich



Obrázek 6.1: Architektura OCR neuronové sítě. U konvolučních vrstev je vyznačen počet filtrů a jejich velikost, u Max-Pooling vrstev je vyznačen prostorový krok a u LSTM vrstev je vyznačena velikost skrytého stavu.

generování prefixové paprskové dekódování použil. V praxi se pak každý může rozhodnout sám, zdali je pro něj důležitější rychlejší hladové dekódování nebo přesnější prefixové paprskové dekódování.

Architektury neuronových sítí

Jak architektura OCR neuronové sítě, tak architektura LSTM jazykového modelu je shodná s architekturou referenčního řešení AT-ST [2] kvůli možnosti přímého porovnání. Architektura OCR neuronové sítě vychází z CRNN [45], kde jsou konvoluční bloky následované BLSTM rekurentními vrstvami, viz obrázek 6.1. Váhy konvolučních vrstev byly inicializovány pomocí vah z předtrénované neuronové sítě VGG16 [60]. Lokalizované a vyříznuté vstupní řádky mají konstantní výšku 40 pixelů a variabilní šířku W. Výstup neuronové sítě má pak rozměry $W/4 \times |V|$, kde $|V|$ je počet prvků množiny výstupních symbolů zahrnující i prázdný symbol ϵ. OCR modely byly trénovány 250 000 epizod optimalizačním algoritmem Adam a různými chybovými funkcemi odvíjejícími se od konkrétního experimentu - originální CTC, váhovaná CTC a rámcová křížová entropie. Na začátku trénování byl učící koeficient nastaven na hodnotu 2×10^{-4} a jeho hodnota se skokově snižovala na polovinu po 150 a 200 tisících epizodách trénování. Velikost trénovacího vzorku byla nastavena na 32.

Jazykový model je implementovaný jako LSTM neuronová síť [51] se dvěma vrstvami s velikostí skrytého stavu 1 500. Znaky na vstupu neuronové sítě jsou zakódovány formou embeddingu s dimenzí 50. Při trénování byl použit optimalizační algoritmus SGD bez momenta. Na začátku trénování byl učící koeficient nastaven na hodnotu 2.0 a jeho hodnota se skokově snižovala na polovinu pokaždé, když se přestala zlepšovat perplexita modelu na validačních datech. Při vytváření pseudo-štítků prefixovým paprskovým dekódováním byla váha jazykového modelu α pro výpočet jistoty pseudo-štítku podle vztahu 4.1 nastavena na hodnotu 0.4. Počet paprsků dekódování K byl nastaven na 16 a hodnota vkladového bonusu β na 1.0. AT-ST uvádí, že nízké hodnoty parametru β vedou k většímu počtu smazaných znaků ve výsledném přepisu a naopak vyšší hodnoty vedou k lepšímu poměru vložených a smazaných znaků, zároveň s vyššími hodnotami ale roste i chybovost.

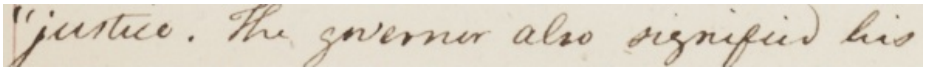
6.1 Datové sady

Experimenty jsem provedl na ručně psaných datech, která jsou shodná s daty použitými v AT-ST [2]. Použitá data jsou rozdělena do tří disjunktních skupin - cílová anotovaná data, cílová neanotovaná data a související anotovaná data, viz tabulka 6.1. Účelem self-trainingu je zvýšení úspěšnosti modelu na cílových datech. Související data jsou vizuálně podobná datům cílovým a do trénování se přidávají kvůli nízkému počtu anotovaných cílových dat

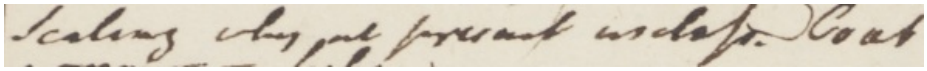
Tabulka 6.1: Počty řádků v použitých datových sadách.

	Trénovací	Validační	Testovací
Související anotované řádky	189 805	—	700
Cílové anotované řádky	9 194	1 415	860
Cílové neanotované řádky	1 141 566	—	—

(a) ICDAR 2017 READ dataset.



(b) ICFHR 2014 Bentham dataset.



(c) Dataset od Bentham projektu.

Obrázek 6.2: Ukázky řádků z použitých datových sad.

s cílem lepší generalizace modelu. Související anotovaná data obsahují řádky z ICDAR 2017 READ datasetu [53], cílová anotovaná data obsahují řádky z ICFHR 2014 Bentham datasetu [52]. Cílové neanotované řádky pocházejí z datasetu získaného od Bentham projektu[1]. Ukázky řádků z jednotlivých datových sad jsou k dispozici na obrázku 6.2. READ dataset obsahuje stránky psané v němčině, italštině a francouzštině. Bentham dataset a dataset získaný od Bentham projektu obsahuje stránky psané v angličtině raného 19. století.

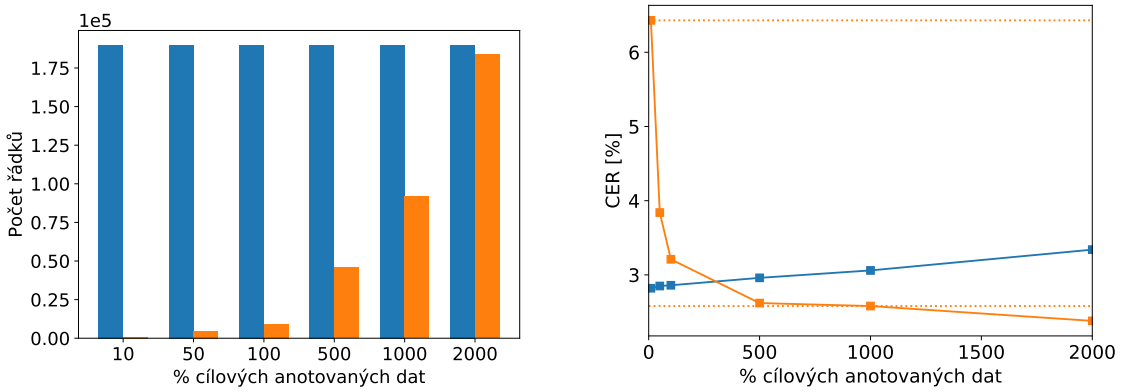
Jazykový model byl trénován na datasetu Wikitext-103 [33], který obsahuje přibližně 530M znaků (103 M anglických slov) a na anotovaných cílových datech pocházejících z ICFHR 2014 Bentham datasetu.

Většina experimentů byla provedena na dvou různě velkých datových sadách, které zkoumají vliv množství dostupných cílových anotovaných dat na úspěšnost OCR neuronové sítě. Malá datová sada obsahuje 10 % cílových anotovaných dat, která byla vybrána náhodně. Velká datová sada obsahuje 100 % cílových anotovaných dat, kterým byla při trénování zvednuta váha pomocí 10× častějšího výskytu v trénovacích datech. Toto množství anotovaných dat bylo použito jak k natrénování jednotlivých počátečních OCR neuronových sítí, tak k natrénování příslušných jazykových modelů.

Augmentace dat

Kromě maskovacích augmentací popsaných v sekci 3.2.2 byly na data při trénování aplikovány i klasičtější augmentace - geometrické transformace, změna barev, rozostření a přidání šumu. Geometrická transformace byla na řádek aplikována s pravděpodobností 0.66 a zahrnovala posunutí, zkosení a změnu měřítka řádku. S pravděpodobností 0.1 byl do vyříznutého řádku přidán Gaussův šum, jehož směrodatná odchylka byla generována z rovnoměrného rozložení pravděpodobnosti s minimem 1 a maximem 12. Rozostření vstupního obrazu probíhalo dvěma způsoby a to rozostřením pohybem s pravděpodobností 0.16 a defocus rozostřením s pravděpodobností 0.1, jehož jádro modeluje rozptylová funkce. Oba

[1] https://www.ucl.ac.uk/bentham-project



(a) Poměr řádků v trénovací sadě

(b) Chybovost na testovací sadě

Obrázek 6.3: V levém grafu je vizualizován poměr souvisejících a cílových anotovaných dat pro jednotlivé varianty trénovací datové sady. V pravém grafu jsou výsledky modelů trénovaných na jednotlivých variantách trénovací datové sady měřené v CER na testovací datové sadě. Modře jsou označeny související data a oranžově cílová data. V pravém grafu jsou tečkovanými linkami vyznačené úspěšnosti později používaných počátečních modelů.

způsoby rozostření při tom mohli být použity současně. Korekce barev vstupního řádku pak byla provedena s pravděpodobností 0.33, přičemž hodnota rozsahu aplikované operace byla generována z rovnoměrného rozložení pravděpodobnosti s maximem 0.3 pro jas, 0.4 pro kontrast, 0.8 pro saturaci a 0.3 pro odstín. Znaménko hodnoty rozsahu operace pak bylo zvoleno zcela náhodně a s ním tedy i směr barevné korekce. S pravděpodobností 0.25 pak byla u vstupního obrazu provedena gamma korekce s hodnotou γ generovanou z rovnoměrného rozložení pravděpodobnosti s minimem 0.5 a maximem 1.5. Maskovací augmentace byla aplikována s pravděpodobností 0.5. Počet maskovaných oblastí byl pak získán z binomického rozložení pravděpodobnosti, kde hodnota n byla rovna délce řádku v pixelech a $p = 0.005$. Výška maskované oblasti odpovídala výšce řádku a její délka byla získána z rovnoměrného rozložení pravděpodobnosti, kde minimum činilo 0.125 výšky řádku a maximum bylo rovno výšce řádku.

6.2 Vliv množství cílových anotovaných dat

Pokud trénovací datová sada obsahuje malé množství dat, může snadno dojít k přetrénování neuronové sítě. Self-training metody pak právě s malým množstvím dat počítají, ostatně nedostatek anotovaných dat je jedním z důvodů jejich využití. Do trénovací datové sady se kvůli nízkému počtu cílových anotovaných dat přidávají související anotovaná data s cílem lepší generalizace trénovaného modelu a menší pravděpodobnosti přetrénování. V tomto experimentu ověřuji, jaký vliv má různé množství cílových anotovaných dat v kombinaci se stabilním počtem souvisejících anotovaných dat na úspěšnost počátečního modelu. Zároveň se tento experiment snaží validovat výběr množství cílových anotovaných dat použitých k vytvoření počátečních modelů používaných v dalších experimentech pro generování strojových anotací.

V rámci experimentu jsem vytvořil několik trénovacích datových sad, které vždy obsahovaly 100 % souvisejících anotovaných dat a odlišné procento cílových anotovaných dat.

Jednotlivé varianty konkrétně obsahovaly 10 % a 50 % náhodně vybraných cílových dat, 100 % cílových dat a poté znásobené počty cílových dat, tedy konkrétně byly v trénovacích datových sadách cílová data 5× častěji, 10× častěji a 20× častěji. Vizualizace poměru cílových a souvisejících anotovaných dat v jednotlivých trénovacích datových sadách je k dispozici na obrázku 6.3a.

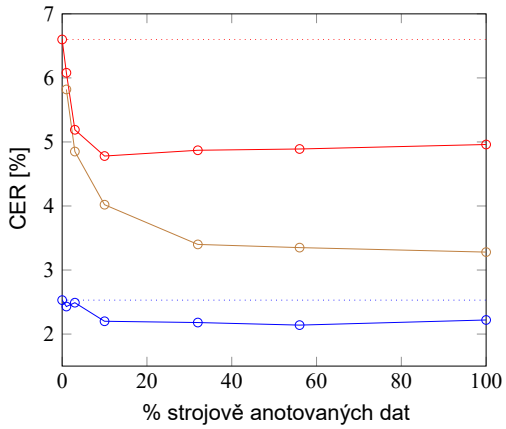
Výsledky experimentu jsou k dispozici na obrázku 6.3b. Zatímco při výběru počátečního modelu reprezentující malou dostupnost cílových anotovaných dat dává od začátku smysl vzít model trénovaný na 10 % cílových anotovaných dat, tak u počátečního modelu reprezentující větší dostupnost cílových anotovaných dat je výběr konkrétního modelu složitější. Z výsledků experimentu vyplývá, že násobný (5×, 10×, 20×) výskyt cílových dat v trénovací datové sadě dále zvyšuje úspěšnost trénovaného modelu. Tento jev je pravděpodobně způsoben snahou neuronové sítě adaptovat se zejména na související data, kterých je ve všech trénovacích datových sadách více a v některých výrazně více (varianty s 10 %, 50 % a 100 %). Přičemž u trénovacích datových sad s výrazně rostoucím poměrem cílových dat vůči souvisejícím datům přirozeně klesá chybovost na cílové testovací datové sadě a roste chybovost na související testovací datové sadě. Jako počáteční model modelující větší dostupnost cílových anotovaných dat je tak vhodné zvolit právě jeden z modelů s násobným výskytem cílových dat v trénovací sadě, protože u těchto modelů se neuronová síť už výrazněji nesnaží adaptovat na související data. Rozdíl ve vzájemné úspěšnosti těchto modelů tím pádem není tak markantní jako u modelů trénovaných s menšími podíly cílových anotovaných dat. Pro generování strojových anotací v dalších experimentech jsou použity vybrané modely právě z tohoto experimentu. Konkrétně tedy pro datovou sadu simulující malé množství dostupných cílových anotovaných dat je použit model trénovaný na 10 % cílových anotovaných dat a pro datovou sadu simulující větší množství dostupných cílových anotovaných dat je použit model trénovaný s 10× častějším výskytem cílových anotovaných dat, viz tabulka 6.3.

6.3 Korekce pseudo-štítků jazykovým modelem

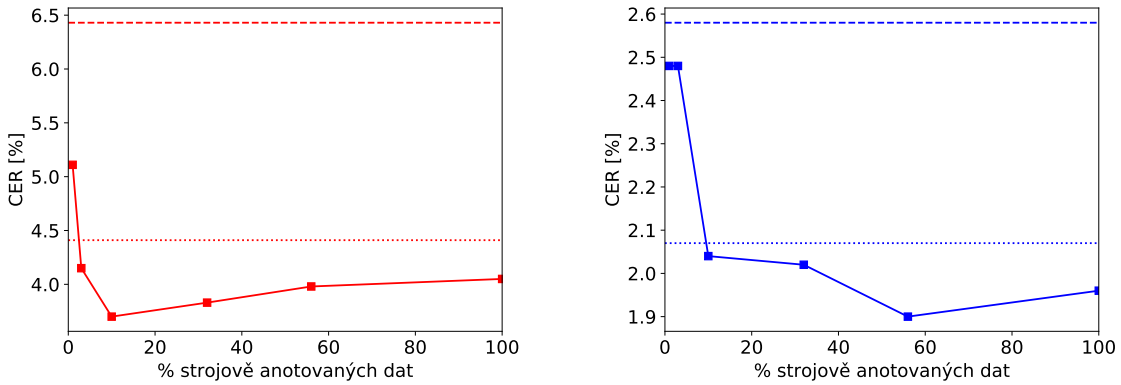
Použití pseudo-štítků s korekcí jazykovým modelem pro trénování má za cíl zvýšit úspěšnost cílového modelu více než použití stejných pseudo-štítků bez korekce. Důležitým prvkem self-trainingu je pak výběr nejjistějších strojově anotovaných dat. Selekce optimální metriky pro výpočet jistoty pseudo-štítku ani volba množství strojově anotovaných dat pro trénování ovšem nejsou triviální záležitostí. Tento experiment zkoumá zvýšení úspěšnosti cílového modelu při korekci pseudo-štítků jazykovým modelem a otázku, jaký je ideální podíl nejjistějších strojově anotovaných dat v trénovací sadě.

Výsledky experimentu jsou přímo porovnatelné s AT-ST [2], kde byly použity pseudo-štítky bez korekce jazykovým modelem. Stejně jako v AT-ST bylo v jednotlivých variantách experimentu přidáno do trénovací sady 1 %, 3 %, 10 %, 32 %, 56 % a 100 % nejjistějších strojově anotovaných dat. Jako metrika jistoty strojové anotace byla použita její posteriorní pravděpodobnost, viz kapitola 4.2. V případě dvou řádků se stejně pravděpodobným strojovým přepisem byl do trénovací sady přidán ten delší. Výsledky nejlepších AT-ST modelů jsou označeny jako referenční, obrázek 6.4 pak obsahuje graf úspěšností všech modelů AT-ST.

Na obrázku 6.5 jsou k dispozici grafy s vizualizací úspěšnosti cílových modelů, které byly trénovány na strojově anotovaných datech s korekcí jazykovým modelem. Při porovnání s úspěšností modelů trénovaných na datech bez korekce na obrázku 6.4, lze jasně pozorovat vyšší úspěšnost modelů trénovaných na datech s korekcí a to u všech variant obou datových



Obrázek 6.4: Úspěšnost cílových modelů trénovaných na určitém procentu nejjistějších strojově anotovaných dat, která neprošla korekcí jazykovým modelem. Červená barva reprezentuje modely trénované na malé datové sadě, modrá barva reprezentuje modely trénované na velké datové sadě. Hnědá barva není v kontextu tohoto experimentu podstatná. Tečkované linky znázorňují úspěšnost počátečních modelů. Převzato z AT-ST [2].



(a) Malá datová sada

(b) Velká datová sada

Obrázek 6.5: Úspěšnost cílových modelů trénovaných na určitém procentu nejjistějších strojově anotovaných dat, která prošla korekcí jazykovým modelem. Čárkovaná linka znázorňuje úspěšnost počátečního modelu, tečkovaná linka znázorňuje úspěšnost referenčního modelu AT-ST.

sad. Korekce pseudo-štítků jazykovým modelem tedy jasně benefituje úspěšnosti cílového modelu. Při srovnání křivek úspěšností na obrázcích 6.4 a 6.5 je pak patrná korelace mezi úspěšnostmi modelů trénovaných na datech s korekcí a modelů trénovaných na datech bez korekce a to u obou datových sad. Pro malou datovou sadu vychází v obou případech jako nejúspěšnější model ten, který byl trénovaný na 10 % nejjistějších strojově anotovaných dat. U velké datové sady pak byly nejúspěšnější modely trénované na 56 % nejjistějších strojově anotovaných dat. Nejúspěšnější model trénovaný na malé datové sadě dosahuje 3.70 CER, což je relativní zlepšení $\sim$ 42 % vůči počátečnímu modelu a $\sim$ 16 % relativní zlepšení vůči referenčnímu modelu. Nejúspěšnější model trénovaný na velké datové sadě pak dosahuje

Tabulka 6.2: Porovnání metody využívající váhované pseudo-štítky s metodou využívající podíl nejjistějších pseudo-štítků (bez váhy). Průměry jsou uvedeny v tabulce z důvodu neexistence heuristiky, podle které lze vybrat ideální podíl strojově anotovaných dat.

	Malá dat. sada	Velká dat. sada
S váhou (100 %)	**3.79**	2.00
Bez váhy (100 %)	4.05	**1.96**
Bez váhy (průměr 10 %, 56 %)	3.84	1.97
Bez váhy (průměr 10 %, 32 % a 56 %)	3.83	1.99
Bez váhy (průměr 1 %, 3 %, 10 %, 32 %, 56 %, 100%)	4.14	2.15

1.90 CER, což je relativní zlepšení $\sim$ 26 % vzhledem počátečnímu modelu a $\sim$ 8 % relativní zlepšení vzhledem referenčnímu modelu.

I přes velmi dobré výsledky zůstává značnou nevýhodou této metody parametr výběru podílu strojově anotovaných dat pro trénování. U obou datových sad byl nejúspěšnější model trénován na jiném podílu nejjistějších strojově anotovaných dat, a tak z provedených experimentů nelze dojít k obecně nejlepší hodnotě podílu. Z grafů na obrázku 6.5 lze nicméně vyčíst skutečnost, že od nalezeného globálního minima (nejúspěšnější model) chybovost sítě roste na obě strany a nevyskytuje se zde žádná forma oscilace. Výsledky experimentu jsou porovnány s ostatními metodami v tabulce 6.4.

Vliv jistoty pseudo-štítku

Alternativou k použití určitého počtu nejjistějších strojově anotovaných dat je použít všechna tato data a váhovat je podle jejich jistoty tak, že se neuronová síť bude učit méně z gradientů méně jistých pseudo-štítků. Tento přístup v kombinaci s korekcí pseudo-štítků jazykovým modelem by měl vést ke zvýšení úspěšnosti cílového modelu vzhledem k počátečnímu modelu a odstranění potřeby výběru určitého podílu nejjistějších strojově anotovaných dat.

Váha pseudo-štítku byla určena jako jeho posteriorní pravděpodobnost, viz kapitola 4.2. Výsledky experimentu jsou porovnány s ostatními metodami v tabulce 6.4. Cílový model trénovaný na malé datové sadě dosáhl 3.79 CER, což je relativní zlepšení o $\sim$ 41 % vůči počátečnímu modelu a o $\sim$ 14 % vůči referenčnímu modelu. Na velké datové sadě pak cílový model dosáhl 2.00 CER, což je relativní zlepšení o $\sim$ 23 % vůči počátečnímu modelu a o $\sim$ 3 % vůči referenčnímu modelu. Byť úspěšnost modelů trénovaných touto metodou nedosahuje lepších výsledků než nejúspěšnější modely trénované na určitém podílu nejjistějších strojově anotovaných dat, je tento přístup výhodný, protože u něj není zapotřebí zjišťovat velikost ideálního podílu nejjistějších strojově anotovaných dat, viz tabulka 6.2. Při malé dostupnosti cílových anotovaných dat pak může metoda využívající váhovaných pseudo-štítků dosahovat i lepší úspěšnosti než při zvolení většiny podílů nejjistějších strojově anotovaných dat. U velké datové sady pak metoda zaostává o setiny procenta, při jejím použití se tedy jedná o jakýsi kompromis mezi přímočarostí a nejlepším možným výsledkem. U malé datové sady byl model trénovaný touto metoda úspěšnější než všechny modely trénované na podílech strojově anotovaných dat kromě modelu trénovaného na 10 % podílu. U velké datové sady byl model trénovaný touto metodou úspěšnější než všechny modely trénované na podílech strojově anotovaných dat kromě modelů trénovaných na 56 % a 100 % podílu.

Tabulka 6.3: Počty trénovacích řádků v jednotlivých datových sadách. Ve velké datové sadě bylo použito 9 194 jedinečných cílových anotovaných řádků, které se při trénování vyskytovaly 10× častěji než ostatní řádky.

	Malá datová sada	Velká datová sada
Související anotované řádky	189 805	189 805
Cílové anotované řádky	894	9 194×10
Cílové strojově anotované řádky	1 002 351	1 022 199

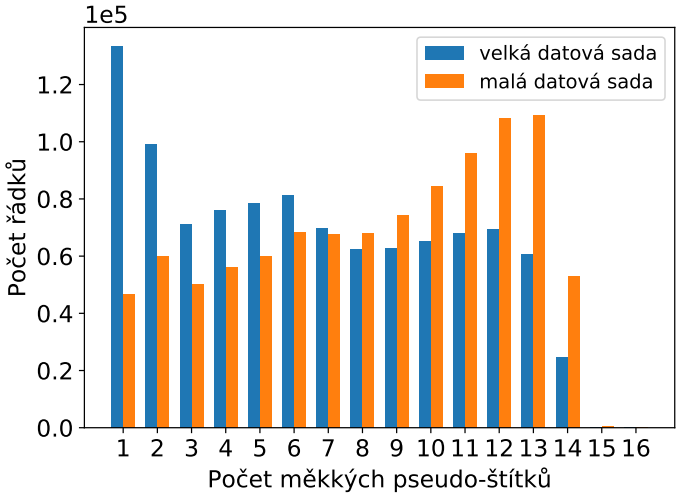
6.4 Trénování OCR variantami pseudo-štítků

Trénováním neuronové sítě s váhovanými pseudo-štítky podle jejich posteriorních pravděpodobností dochází ke snížení vlivu pseudo-štítků s větší chybovostí na neuronovou síť. Díky tomu se ale rovněž zmenšuje možnost neuronové sítě učit se z jistějších částí chybovějších řádků, viz obrázek 4.3, kde jsou chyby pouze v určité oblasti řádku. Při použití měkkých pseudo-štítků, tedy několika variant strojového přepisu s různou jistotou náležící k jednomu řádku, ale zůstávají jistější oblasti pseudo-štítku stejné a varianty se liší hlavně v nejistých místech. Díky tomu by se neuronová síť teoreticky měla učit převážně z jistých oblastí se stejnými přepisy, jejichž váhy se v součtu budou blížit hodnotě 1. Zatímco v nejistých oblastech by se neuronová síť měla učit z chyb méně v důsledku rozdílnosti variant pseudo-štítku a tím vzniklého šumu.

Pro každý zpracovaný řádek bylo pomocí prefixového paprskového dekódování a jazykového modelu vygenerováno 16 nejpravděpodobnějších hypotéz. Jako varianty pseudo-štítku byly použity nejjistější hypotézy s kumulativní pravděpodobností 90 %. Hodnota 90 % byla zvolena experimentálně tak, aby v případě jistých řádků nebyly do trénovací datové sady zbytečně přidávány hypotézy s pravděpodobností blížící se nule. Pokud 16 nejpravděpodobnějších hypotéz obsahovalo hypotézu prázdného řetězce, tak se do trénovací datové sady nepřidala žádná a řádek byl vyřazen. Přehled množství užitých trénovacích dat pro obě datové sady je k dispozici v tabulce 6.3.

Z obrázku 6.6 vyplývá, že s větším množstvím cílových anotovaných dat a stoupající úspěšností počátečního modelu jsou do trénovací datové sady zařazovány pseudo-štítky s menším počtem variant strojového přepisu. Z toho lze vyvodit, že varianty pseudo-štítků generované počátečním modelem trénovaném na velké datové sadě jsou jistější.

Úspěšnost cílových modelů trénovaných na měkkých pseudo-štítcích je v tabulce 6.4 porovnána s úspěšností ostatních metod trénování. Hypotéza se zlepšením modelu díky rozdílným variantám strojového přepisu na nejistých místech nebyla potvrzena. Model trénovaný na malé datové sadě dosáhl 3.98 CER, což je relativní zhoršení o $\sim$ 5 % vzhledem k modelu trénovanému na váhovaných pseudo-štítcích s jednou variantou. Model trénovaný na velké datové sadě pak dosáhl 1.99 CER, což je úspěšnost velmi blízká 2.00 CER, které dosáhl model trénovaný na váhovaných pseudo-štítcích s jednou variantou. Byť jde v případě trénování na velké datové sadě o $\sim$ 0.5 % relativní zlepšení, neukázala se navržená metoda jako efektivnější oproti předchozím a to zejména v kontextu úspěšnosti cílového modelu trénovaného na malé datové sadě. Obecně jsou ale cílové modely úspěšnější než počáteční i referenční modely. V případě modelu trénovaného na malé datové sadě jde o relativní zlepšení $\sim$ 38 % vůči počátečnímu modelu a o $\sim$ 10 % relativní zlepšení vůči referenčnímu modelu. V případě modelu trénovaného na velké datové sadě pak jde o relativní zlepšení $\sim$ 22 % vůči počátečnímu modelu a o $\sim$ 4 % proti referenčnímu modelu.



Obrázek 6.6: Počet variant pseudo-štítku na jeden neanotovaný řádek v trénovací datové sadě. Oranžová barva reprezentuje pseudo-štítky neanotovaných dat generovaných počátečním modelem trénovaným na malé datové sadě. Modrá barva reprezentuje pseudo-štítky generované počátečním modelem trénovaným na velké datové sadě.

Inicializace cílového modelu počátečním modelem

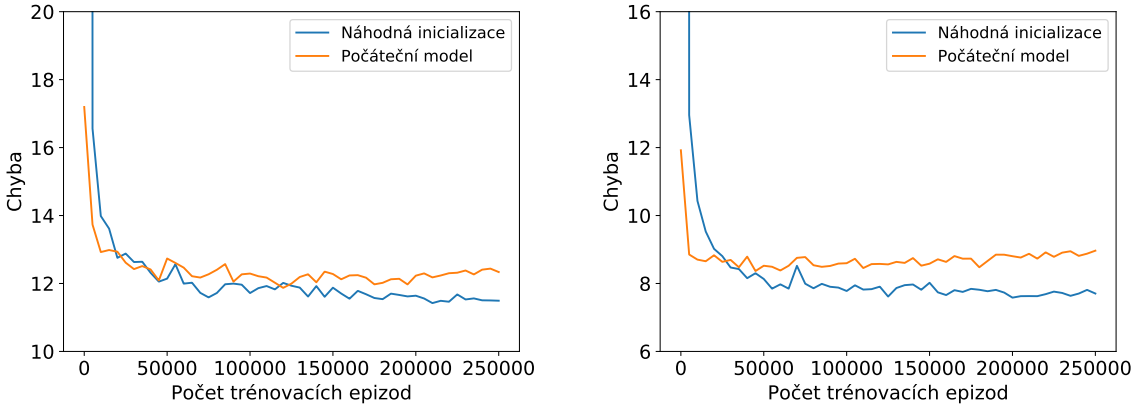
Navržená self-training metoda je časově relativně náročná skrze čas strávený trénováním počátečního modelu, generováním strojových anotací a trénováním cílového modelu. Čas strávený trénováním cílového modelu jsem se pokusil redukovat inicializací vah cílového modelu váhami z natrénovaného počátečního modelu. Cílem při tom bylo získat model se stejnou úspěšností za kratší počet epizod trénování. Všechny parametry trénování byly nastaveny stejně jako při trénování nového náhodně inicializovaného modelu a to včetně postupného poklesu učícího koeficientu.

Na obou datových sadách došlo ke zhoršení úspěšnosti výsledných modelů oproti původnímu trénování s náhodně inicializovanými modely. Konkrétněji na malé datové sadě dosáhl cílový model 4.30 CER a na velké datové sadě 2.10 CER. Práce [21] dokládá, že při využití předtrénované neuronové sítě může k mírnému zhoršení úspěšnosti dojít, nicméně z grafů na obrázku 6.7 vyplývá, že při inicializaci vah cílového modelu váhami počátečního modelu pravděpodobně dochází k přetrénování neuronové sítě. Tento jev dokládá rostoucí hodnota chybové funkce na validačních datech v průběhu trénování.

Benefit druhé iterace

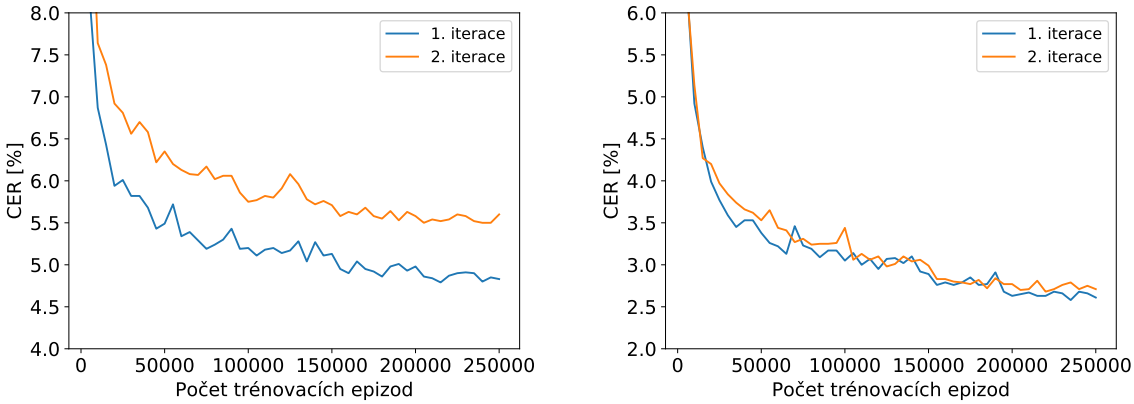
Self-training je relativně běžné provádět iterativně po dobu zlepšování úspěšnosti cílového modelu. Právě v rámci experimentu s variantami pseudo-štítků jsem se rozhodl experimentovat s druhou iterací cílového modelu. Experiment jsem provedl s cílem zjistit, jak se model při trénování bude chovat a zdalipak je možné zvýšit jeho úspěšnost oproti první iteraci.

Prvním krokem experimentu bylo vygenerování pseudo-štítků pro neanotovaná data k čemuž byly použity natrénované cílové modely z první iterace a původní jazykový model. Rozložení počtu variant pseudo-štítku na jeden neanotovaný řádek v nově vytvořených datových sadách bylo velmi podobné rozložení v první iteraci self-training metody, které



(a) Malá datová sada (b) Velká datová sada

Obrázek 6.7: Hodnota chybové funkce cílových modelů na validačních datech v průběhu trénování. Modrá křivka reprezentuje náhodně inicializovaný model a oranžová křivka reprezentuje model inicializovaný váhami z počátečního modelu.

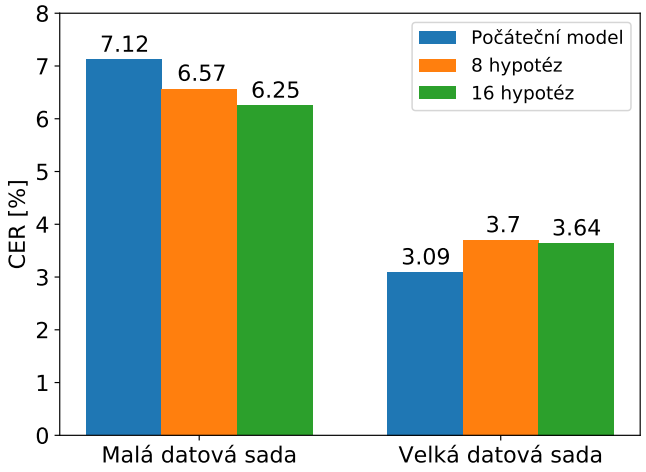


(a) Malá datová sada (b) Velká datová sada

Obrázek 6.8: Úspěšnost modelů měřená v CER [%] na validačních datech v průběhu trénování. Modrá křivka reprezentuje první iteraci trénování a oranžová křivka reprezentuje druhou iteraci trénování.

je k dispozici na obrázku 6.6. Následný proces trénování probíhal stejně jako ve všech předchozích experimentech.

Na obou datových sadách došlo ke zhoršení úspěšnosti cílových modelů oproti jejich první iteraci. Konkrétněji na malé datové sadě dosáhl cílový model 4.55 CER a na velké datové sadě 2.06 CER. Úspěšnosti modelů v průběhu trénování jsou k dispozici na obrázku 6.8. V tomto experimentu by horší úspěšnost cílových modelů neměla být způsobena přetrénováním, protože chyba na validační datové sadě v průběhu celého trénování klesá. Vzhledem ke statisticky podobnému rozložení dat jako v první iteraci bude zhoršená úspěšnost pravděpodobně souviset s kvalitou dat. Zejména u modelu trénovaného na malé datové sadě lze vidět propastný rozdíl v úspěšnosti mezi první a druhou iterací, viz obrázek 6.8a. Cílové modely první iterace teoreticky mohou trpět určitou formou konfirmačního zkreslení, které při dekódování jejich výstupů algoritmem prefixového paprskového dekódování může vy-



Obrázek 6.9: Graf porovnávající úspěšnost počátečního modelu a cílových modelů trénovaných rámcovou křížovou entropií. Úspěšnost modelů je měřena na validační datové sadě.

tvořit varianty pseudo-štítků s chybami. Tyto chyby při vyhodnocování úspěšnosti cílových modelů ale zůstanou skryty, protože se nevyskytují v nejpravděpodobnějších hypotézách, které se k vyhodnocování používají. Jedná se ovšem pouze o hypotézu. Identifikace a odstranění jevu způsobujícího zhoršenou úspěšnost v druhé iteraci trénování je do budoucna určitě jednou z oblastí výzkumu s vysokým přínosem.

6.5 Trénování OCR rámcovou křížovou entropií

Vytváření strojových anotací pro trénování s rámcovou křížovou entropií je časově o dost náročnější než vytváření strojových anotací pro trénování s variantami pseudo-štítků pomocí CTC. Časová náročnost je způsobená zarovnáváním hypotéz na výstup OCR neuronové sítě. V experimentu proto kromě úspěšnosti metody samotné zkoumám, jakým způsobem ovlivňuje počet zarovnávaných hypotéz úspěšnost cílového modelu. Poloviční počet zarovnávaných hypotéz totiž může zkrátit dobu vytváření trénovací datové sady téměř o polovinu. Vzhledem k dopřednému generování pseudo-štítků by augmentace trénovacích dat pomocí geometrických transformací znehodnotily přesné zarovnání, a tak v tomto experimentu nebyly na data aplikovány. Cílová anotovaná data byla pro trénování rovněž zarovnána pomocí příslušných počátečních modelů.

Na obrázku 6.9 jsou k dispozici výsledky experimentu. U obou datových sad je možné pozorovat zvýšení úspěšnosti cílového modelu při použití většího množství zarovnaných hypotéz. U malé datové sady se jedná o $\sim$ 5 % relativní zlepšení a u velké datové sady o $\sim$ 1.5 % relativní zlepšení. Cílový model nicméně překonal úspěšnost počátečního modelu jen na malé datové sadě. Na testovacích datech dosáhl 5.29 CER, což je relativní zlepšení o $\sim$ 18 % oproti počátečnímu modelu. V porovnání s referenčním modelem pak jde dokonce o $\sim$ 20 % relativní zhoršení. Model trénovaný na velké datové sadě dosáhl na testovacích datech 2.89 CER, což je relativní zhoršení o $\sim$ 12 % oproti počátečnímu modelu. V tomto experimentu je výrazné zhoršení úspěšnosti cílových modelů pravděpodobně způsobeno relativně odlišným přístupem k trénování (jiný formát dat a odlišná chybová funkce), který se ukázal jako ne příliš úspěšný. Je pravděpodobné, že pokud by byly počáteční modely trénované stejnou chybovou funkcí, došlo by ke zvýšení úspěšnosti cílových modelů v kontextu použití stejné chybové funkce.

Lepších výsledků při trénování rámcovou křížovou entropií by teoreticky mohlo být dosaženo použitím jiného algoritmu pro zarovnávání hypotéz jazykové modelu do matice pravděpodobností. Použitý algoritmus totiž mapuje symbol vždy jen na nejpravděpodobnější rámec a při tom se symbol může nacházet v několika rámcích najednou. Dalšího zlepšení by teoreticky mohlo být dosaženo využitím jen určitého počtu nejjistějších strojově anotovaných dat jako v experimentu 6.3. Výsledky metody trénování rámcovou křížovou entropií jsou v tabulce 6.4 porovnány s výsledky ostatních metod.

Tabulka 6.4: Výsledky jednotlivých metod na cílových testovacích datech v CER [%].

	Malá datová sada	Velká datová sada
Počáteční model	6.43	2.58
AT-ST	4.41	2.07
Cílový model - pseudo-štítek po korekci	**3.70**	**1.90**
Cílový model - pseudo-štítek s váhou	3.79	2.00
Cílový model - varianty pseudo-štítku	3.98	1.99
Cílový model - rámcová křížová entropie	5.29	2.89

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat efektivní způsob využití neanotovaných dat pro trénování OCR. Ke splnění tohoto cíle jsem nastudoval nejmodernější metody využívající neanotovaná data, jak v oblasti OCR, tak v oblastech dalších jako je zpracování řeči nebo klasifikační úlohy.

Experimentoval jsem s několika přístupy, které spadají do kategorie self-training algoritmů. Obecný postup všech navržených metod se dá sumarizovat následovně. Nejprve je na omezeném množství anotovaných dat natrénován počáteční model OCR neuronové sítě. Ten je následně spolu s LSTM jazykovým modelem použit k vygenerování pseudo-štítků neanotovaných dat. Takto strojově anotovaná data jsou zkombinována s trénovacími daty, která byla použita k vytvoření počátečního modelu a následně jsou využita k natrénování cílového modelu. Úspěšnost jednotlivých navržených postupů trénování jsem měřil na ručně psaném ICFHR 2014 Bentham datasetu. Přičemž jsem všechny experimenty provedl na dvou datových sadách, které simulovaly různou míru dostupnosti anotovaných dat.

Mezi klíčové, mnou navrhované, prvky trénování patří využití jazykového modelu ke korekci pseudo-štítků. Nejlepší modely trénované na strojově anotovaných datech, která prošla korekcí jazykovým modelem, dosahují na malé datové sadě 3.70 CER, což je relativní zlepšení o $\sim$ 42 % oproti počátečnímu modelu a 1.90 CER na velké datové sadě, což je relativní zlepšení o $\sim$ 26 % oproti počátečnímu modelu. Cílové modely trénované touto metodou rovněž překonávají metodu AT-ST [2] o $\sim$ 16 % na malé datové sadě a o $\sim$ 8 % na velké datové sadě. Tato procentuální zlepšení, i všechna dále zmíněná, jsou udávaná jako relativní.

Dalším navrženým prvkem trénování OCR neuronové sítě je váhování strojově anotovaných řádků podle jejich jistoty. Tento přístup nedosahoval takových výsledků jako první metoda, ale jeho použití je přímočařejší. Nevyžaduje totiž nastavení určitých parametrů klíčových pro dosažení optimální úspěšnosti OCR trénováním první metodou. Tento přístup rovněž překonal AT-ST a to o $\sim$ 14 % na malé datové sadě a o $\sim$ 3 % na velké datové sadě.

Mezi méně úspěšné přístupy trénování, které jsem navrhl, patří využití měkkých pseudo-štítků, tedy několika variant pseudo-štítku náležícího k jednomu strojově anotovanému řádku. Tato metoda překonala AT-ST o $\sim$ 10 % na malé datové sadě a o $\sim$ 4 % na velké datové sadě. Určitou formu měkkých pseudo-štítků pak využívá i poslední navržený přístup a to trénování OCR neuronové sítě rámcovou křížovou entropií. Tento přístup jako jediný nedosáhl relativního zlepšení oproti metodě AT-ST a na velké datové sadě kvalitu cílového modelu dokonce zhoršil.

Dalších pokroků v oblasti využití neanotovaných dat pro trénování OCR by mohlo být dosaženo zejména větším experimentováním s dalšími iteracemi self-training metod. Provedení všech experimentů zabralo celkem desítky dní GPU času. Téma této práce mi umožnilo získat jedinečný rozhled v oblasti self-training algoritmů, což vnímám jako velmi cenou znalost v kontextu míry využití semi-supervised algoritmů v současnosti.

Literatura

[1] *Convolutional Neural Networks (CNNs / ConvNets)* [Course materials (english)]. https://cs231n.github.io/convolutional-networks/.

[2] Beneš, K. a Kišš, M. AT-ST: Self-Training Adaptation Strategy for OCR in Domains with Limited Transcriptions. 2021.

[3] Bengio, Y. Neural net language models. *Scholarpedia*. 2008, sv. 3, č. 1, s. 3881. DOI: 10.4249/scholarpedia.3881. revision #140963.

[4] Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A. et al. MixMatch: A Holistic Approach to Semi-Supervised Learning. *CoRR*. 2019, abs/1905.02249. Dostupné z: http://arxiv.org/abs/1905.02249.

[5] Bertolami, R., Zimmermann, M. a Bunke, H. Rejection strategies for offline handwritten text line recognition. *Pattern Recognition Letters*. Prosinec 2006, sv. 27, s. 2005–2012. DOI: 10.1016/j.patrec.2006.06.002.

[6] Blum, A. a Mitchell, T. Combining Labeled and Unlabeled Data with Co-Training. *Proceedings of the Annual ACM Conference on Computational Learning Theory*. Říjen 2000. DOI: 10.1145/279943.279962.

[7] Breuel, T. The OCRopus open source OCR system. In:. Leden 2008, sv. 6815, s. 68150. DOI: 10.1117/12.783598.

[8] Chakravorty, S. *Language Models* [blogpost (english)]. June 2020. https://towardsdatascience.com/language-models-1a08779b8e12.

[9] Das, D. a Jawahar, C. Adapting OCR with Limited Supervision. In:. Srpen 2020, s. 30–44. DOI: 10.1007/978-3-030-57058-3_3. ISBN 978-3-030-57057-6.

[10] DeVries, T. a Taylor, G. W. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017.

[11] dprogrammer. *RNN, LSTM GRU* [blogpost (english)]. April 2019. http://dprogrammer.org/rnn-lstm-gru.

[12] Du, Y., Li, C., Guo, R., Yin, X., Liu, W. et al. *PP-OCR: A Practical Ultra Lightweight OCR System*. 2020.

[13] Forney, G. The viterbi algorithm. *Proceedings of the IEEE*. 1973, sv. 61, č. 3, s. 268–278. DOI: 10.1109/PROC.1973.9030.

[14] František, Z. *Acyklické a dopředné neuronové sítě. Algoritmus backpropagation.* [slides (czech)]. https://www.fit.vutbr.cz/study/courses/SFC/private/20sfc_2.pdf.

[15] Frinken, V. a Bunke, H. Evaluating Retraining Rules for Semi-Supervised Learning in Neural Network Based Cursive Word Recognition. In: *2009 10th International Conference on Document Analysis and Recognition*. 2009, s. 31–35. DOI: 10.1109/ICDAR.2009.18.

[16] Frinken, V., Fischer, A., Bunke, H. a Foornes, A. Co-Training for Handwritten Word Recognition. In:. říjen 2011, s. 314 – 318. DOI: 10.1109/ICDAR.2011.71.

[17] Galstyan, A. a Cohen, P. Empirical comparison of hard and soft label propagation for relational classification. In:. červen 2007, s. 98–111. DOI: 10.1007/978-3-540-78469-213.

[18] Graves, A. Generating Sequences With Recurrent Neural Networks. *CoRR*. 2013, abs/1308.0850. Dostupné z: http://arxiv.org/abs/1308.0850.

[19] Graves, A., Fernández, S., Gomez, F. a Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. In:. Leden 2006, sv. 2006, s. 369–376. DOI: 10.1145/1143844.1143891.

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

[21] He, K., Girshick, R. B. a Dollár, P. Rethinking ImageNet Pre-training. *CoRR*. 2018, abs/1811.08883. Dostupné z: http://arxiv.org/abs/1811.08883.

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

[23] Huang, G., Sun, Y., Liu, Z., Sedra, D. a Weinberger, K. Q. Deep Networks with Stochastic Depth. *CoRR*. 2016, abs/1603.09382. Dostupné z: http://arxiv.org/abs/1603.09382.

[24] Jenckel, M., Bukhari, S. a Dengel, A. AnyOCR: A sequence learning based OCR system for unlabeled historical documents. In:. Prosinec 2016, s. 4035–4040. DOI: 10.1109/ICPR.2016.7900265.

[25] Karpathy, A. *The Unreasonable Effectiveness of Recurrent Neural Networks* [blogpost (english)]. May 2015. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[26] Kodym, O. a Hradiš, M. *Page Layout Analysis System for Unconstrained Historic Documents*. 2021.

[27] Koech, K. E. *Cross-Entropy Loss Function* [blogpost (english)]. October 2020. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[28] Kulathilake, H. *Evaluating Language Model* [slides (english)]. May 2018. https://www.slideshare.net/shkulathilake/nlpkashkevaluating-language-model.

[29] Leifert, G., Labahn, R. a Sánchez, J. A. Two Semi-Supervised Training Approaches for Automated Text Recognition. In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, s. 145–150. DOI: 10.1109/ICFHR2020.2020.00036.

[30] Li, H. a Wang, W. A Novel Re-weighting Method for Connectionist Temporal Classification. *CoRR*. 2019, abs/1904.10619. Dostupné z: http://arxiv.org/abs/1904.10619.

[31] Matcha, A. C. N. *How to easily do Handwriting Recognition using Deep Learning* [blogpost (english)]. March 2021. https://nanonets.com/blog/handwritten-character-recognition/.

[32] Memon, J., Sami, M. a Khan, R. A. *Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)*. 2020.

[33] Merity, S., Xiong, C., Bradbury, J. a Socher, R. Pointer Sentinel Mixture Models. *CoRR*. 2016, abs/1609.07843. Dostupné z: http://arxiv.org/abs/1609.07843.

[34] Mey, A. a Loog, M. A soft-labeled self-training approach. In:. Prosinec 2016, s. 2604–2609. DOI: 10.1109/ICPR.2016.7900028.

[35] Moysset, B. a Messina, R. Manifold Mixup Improves Text Recognition with CTC Loss. In:. Září 2019, s. 799–804. DOI: 10.1109/ICDAR.2019.00133.

[36] Nicholson, C. *A Beginner's Guide to LSTMs and Recurrent Neural Networks* [blogpost (english)]. https://wiki.pathmind.com/lstm.

[37] Olah, C. *Understanding LSTM Networks* [blogpost (english)]. August 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[38] Pham, V. T., Xu, H., Khassanov, Y., Zeng, Z., Chng, E. S. et al. Independent language modeling architecture for end-to-end ASR. *CoRR*. 2019, abs/1912.00863. Dostupné z: http://arxiv.org/abs/1912.00863.

[39] Phi, M. *Illustrated Guide to Recurrent Neural Networks* [blogpost (english)]. September 2018. https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9.

[40] Raschka, S. *What is the difference between a parametric learning algorithm and a nonparametric learning algorithm?* [blogpost (english)]. https://sebastianraschka.com/faq/docs/parametric_vs_nonparametric.html.

[41] Rizvi, M. S. Z. *A Comprehensive Guide to Build your own Language Model in Python!* [blogpost (english)]. August 2019. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-language-model-nlp-python-code/.

[42] Scheidl, H. *Beam Search Decoding in CTC-trained Neural Networks* [blogpost (english)]. July 2018. https://towardsdatascience.com/beam-search-decoding-in-ctc-trained-neural-networks-5a889a3d85a7.

[43] Scheidl, H. *An Intuitive Explanation of Connectionist Temporal Classification* [blogpost (english)]. June 2018. https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c.

[44] Scudder, H. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*. 1965, sv. 11, č. 3, s. 363–371. DOI: 10.1109/TIT.1965.1053799.

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

[46] Smith, R. An Overview of the Tesseract OCR Engine. In:. říjen 2007, sv. 2, s. 629 – 633. DOI: 10.1109/ICDAR.2007.4376991. ISBN 978-0-7695-2822-9.

[47] Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N. et al. *FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence*. 2020.

[48] Solegaonkar, V. *Convolutional Neural Networks* [blogpost (english)]. February 2019. https://towardsdatascience.com/convolutional-neural-networks-e5a6745b2810.

[49] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014, sv. 15, č. 56, s. 1929–1958. Dostupné z: http://jmlr.org/papers/v15/srivastava14a.html.

[50] Stuner, B., Chatelain, C. a Paquet, T. Self-Training of BLSTM with Lexicon Verification for Handwriting Recognition. In:. Listopad 2017, s. 633–638. DOI: 10.1109/ICDAR.2017.109.

[51] Sundermeyer, M., Schlüter, R. a Ney, H. LSTM Neural Networks for Language Modeling. In:. Září 2012.

[52] Sánchez, J. A., Romero, V., Toselli, A. H. a Vidal, E. ICFHR2014 Competition on Handwritten Text Recognition on Transcriptorium Datasets (HTRtS). In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. 2014, s. 785–790. DOI: 10.1109/ICFHR.2014.137.

[53] Sánchez, J. A., Romero, V., Toselli, A. H., Villegas, M. a Vidal, E. Dataset for ICDAR2017 Competition on Handwritten Text Recognition on the READ Dataset (ICDAR2017 HTR). Zenodo. červenec 2017. DOI: 10.5281/zenodo.835489. Dostupné z: https://doi.org/10.5281/zenodo.835489.

[54] Wikipedia. *Convolutional neural network — Wikipedia, The Free Encyclopedia* [http://en.wikipedia.org/w/index.php?title=Convolutional%20neural%20network&oldid=993979296]. 2020. [Online; accessed 30-December-2020].

[55] Wikipedia. *Recurrent neural network — Wikipedia, The Free Encyclopedia* [http://en.wikipedia.org/w/index.php?title=Recurrent%20neural%20network&oldid=996339215]. 2020. [Online; accessed 29-December-2020].

[56] Wikipedia. *K-means clustering — Wikipedia, The Free Encyclopedia* [http://en.wikipedia.org/w/index.php?title=K-means%20clustering&oldid=1001316455]. 2021. [Online; accessed 22-January-2021].

[57] Wikipedia. *Timeline of optical character recognition — Wikipedia, The Free Encyclopedia* [http://en.wikipedia.org/w/index.php?title=Timeline%20of%20optical%20character%20recognition&oldid=986374960]. 2021. [Online; accessed 08-January-2021].

[58] Xie, Q., Hovy, E. H., Luong, M. a Le, Q. V. Self-training with Noisy Student improves ImageNet classification. *CoRR*. 2019, abs/1911.04252. Dostupné z: http://arxiv.org/abs/1911.04252.

[59] Zhang, H., Cissé, M., Dauphin, Y. N. a Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. *CoRR*. 2017, abs/1710.09412. Dostupné z: http://arxiv.org/abs/1710.09412.

[60] Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv 1409.1556*. Září 2014.

[61] Zoph, B., Ghiasi, G., Lin, T., Cui, Y., Liu, H. et al. Rethinking Pre-training and Self-training. *CoRR*. 2020, abs/2006.06882. Dostupné z: https://arxiv.org/abs/2006.06882.

[62] Čepek, M. *Vytěžování dat - Redukce dimenzionality* [slides (czech)]. https://cw.fel.cvut.cz/old/_media/courses/a7b36vyd/prednasky/13-dimred-print.pdf.

Příloha A

Obsah přiloženého CD

- **doc/** – adresář s technickou zprávou
- **src/** – adresář se zdrojovými kódy k experimentům
- **video/** – adresář s videem prezentující moji práci