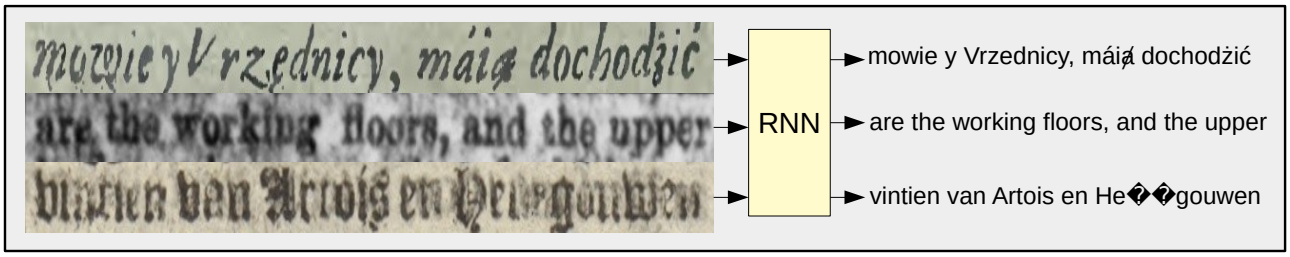
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2019
http://excel.fit.vutbr.cz

Učení a adaptace neuronových sítí pro rozpoznávání textu

Jan Kohút*



Abstrakt

Cílem této práce je srovnání architektur neuronových sítí pro rozpoznávání textu. Dále pak adaptace neuronových sítí na jiné texty, než na kterých byly učeny. Pro tyto experimenty využívám rozsáhlý a rozmanitý dataset IMPACT o více než jednom milionu řádků. Pomocí neuronových sítí provádím kontrolu vhodnosti řádků tzn. čitelnost a správnost výřezů řádků. Celkem srovnávám 6 čistě konvolučních sítí a 9 rekurentních sítí. Adaptace provádím na polských historických textech s tím, že trénovací data adaptovaných sítí neobsahovaly texty ve slovanských jazycích. Adaptace využívají přístupy aktivního učení pro výběr nových adaptačních dat. Čistě konvoluční sítě dosahují úspěšnosti 98.6 %, rekurentní sítě pak 99.5 %. Úspěšnost sítí před adaptací se pohybuje kolem 79%, po postupné adaptaci na 2500 řádcích stoupne úspěšnost na 97 %. Přístupy aktivního učení dosahují lepší úspěšnosti než náhodný výběr. Pro zpracování datasetů je vhodné používat již natrénované neuronové sítě tak, aby se odstranilo co možná nejvíce chybných dat. Rekurentní vrstvy znatelně zvyšují úspěšnost sítí. Při adaptaci je výhodné využívat přístupů aktivního učení.

Klíčová slova: rozpoznávání textu — rekurentní neuronové sítě — konvoluční neuronové sítě — adaptace — aktivní učení — dataset IMPACT

Přiložené materiály: *N/A*

*xkohut08@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Rozpoznávání textu v různých jazycích, fontech a kvalitě je náročný problém, který nemá optimální řešení. Cílem je převod obrazu tištěného nebo psaného textu na text, se kterým lze pracovat v rámci počítače. Hlavními body rozpoznání je detekce řádků textu v obrazu a jejich klasifikace. Jednou z možností, jak dosáhnout kvalitních výsledků jsou neuronové sítě a rozsáhlé datové sady.

Cílem této práce je připravit dataset IMPACT [1] obsahující obrázky stránek textu tak, aby jej bylo možné použít pro trénování neuronových sítí. Dále pak experimenty s různými architekturami sítí a jejich adaptacemi. Kvalitní příprava zahrnuje zejména filtraci datasetu tak, aby došlo k vyloučení co nejvíce chybných řádků tzn. řádků, které mají špatné anotace nebo jsou špatně vyřezány z původního obrázku stránky. v rámci experimentů s architekturami srovnávám zejména smysluplnost zvětšování jejich velikostí vzhledem k dosažené přesnosti a význam rekurentní

vrstvy. Závěrem uvádím experimenty s adaptacemi neuronových sítí na polské historické písmo s tím, že trénovací datasety daných sítí neobsahovaly při trénování slovanské jazyky. Cílem je ukázat které vrstvy sítě je potřeba adaptovat, aby bylo dosaženo co nejlepších výsledku s využitím co nejmenšího počtu dat.

Klasifikaci je možné provádět mnoha způsoby od naivního srovnávání sady vzorových obrázků znaků až po skryté Markovovi modely (HMM) a neuronové sítě [2]. Právě neuronové sítě dosahují v dnešní době nejlepších výsledků v této oblasti. Existující řešení využívají zejména konvolučních a rekurentních vrstev [3][4]. Konvoluční vrstvy slouží k extrakci kvalitních příznaků z obrazu, jako jsou hrany, struktury apod. Rekurentní vrstvy se používají kvůli schopnosti pracovat s kontextem, tuto schopnost ovšem mají i konvoluční vrstvy [5]. Novější přístup představuje tzv. *attention přístup*, který síti umožňuje výběr důležitých příznaků na základě vstupního obrázku [6]. Adaptace již naučené sítě probíhá pomocí učení na nových datech [7].

Architektury neuronových sítí v této práci se skládají jak čistě z konvolučních vrstev, tak i z kombinace konvolučních a rekurentních vrstev. Jako rekurentní vrstva je použita obousměrná LSTM [8]. Jako chybovou funkci jsem zvolil CTC [9]. Adaptace se provádí učením, již naučených sítí a to tak, že se adaptují pouze některé vrstvy. Poslední vrstva je navíc namnožena. To znamená možnost provádět křížovou validaci (*cross-validation*) bez nutnosti současného učení mnoha sítí. Při adaptaci se začíná s malou množinou řádků, která je rozšiřována pouze pokud je to nutné tzn. není možné dosáhnout zlepšení s aktuálním počtem řádků. Výběr nových řádků je založen na aktivním učení.

Pro zpracování datasetu IMPACT využívám již natrénované neuronové sítě tak, abych odhalil chybné anotace a nečitelné řádky. Celkem jsem tímto způsobem získal 1217000 řádků. Nejlepších úspěšnosti dosahují rekurentní neuronové sítě a to 99.5 %. Adaptaci jsem provedl na polských historických textech, původní úspěšnost 79 % byla zvýšena na 97 %.

2. Rozpoznávání textu

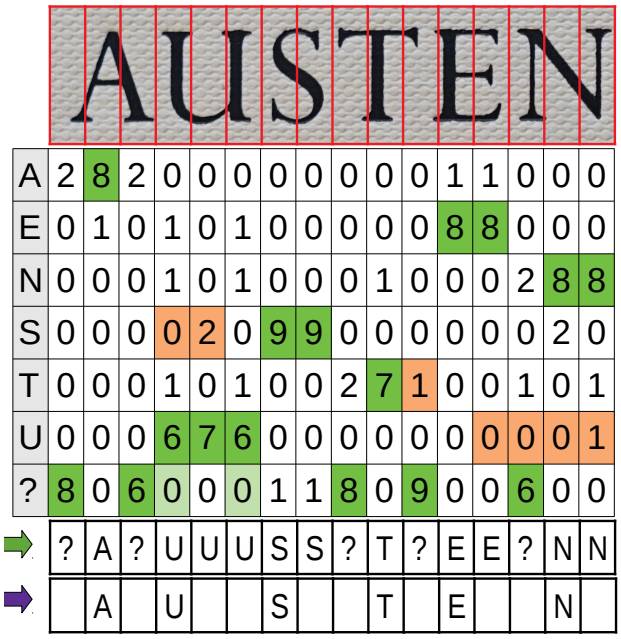
Poté co se pomocí skeneru nebo fotoaparátu získá obraz s textem, je tento obraz upraven pomocí různých transformačních a filtračních funkcí tak, aby byl text co nejlépe čitelný pro další zpracování. v takto upraveném obraze se detekují důležitá místa jako jsou odstavce, nadpisy, okraje atd. v rámci vybraných důležitých míst se pak detekují řádky textu a jejich výšky. Posledním krokem je pak samotná klasifikace takto zjištěných řádků tzn. strojový přepis na text, se kterým lze pracovat v rámci počítače. Vstupem neuronové sítě je tedy výřez řádku a výstupem jeho přepis. Neuronová síť pro rozpoznávání se typicky skládá z konvolučních a rekurentních vrstev. Jedna z chybových funkcí, kterou lze použít je CTC.

RNN Rekurentní neuronové sítě (RNN) [10] obsahují alespoň jednu rekurentní vrstvu. Tyto vrstvy jsou při zpracovávání svého vstupu schopny využít kontext. Kontext, který síť dokáže využít není neomezený. Záleží taky na směru zpracování vstupu, při zpracovávání zleva doprava se při rozpoznávání využívá znalost o znacích předchozích a naopak. Pro využití obou směrů lze použít tzv. obousměrnou rekurentní vrstvu. Pokročilejší rekurentní vrstvy využívají tzv. LSTM [8] (*Long Short-Term Memory*) bloky, které výrazně omezují dopady vyhasínání gradientů [11].

CTC Jelikož síť pro vstup o daném rozměru generuje vždy výstup o stejném rozměru, nelze generovat přepis obrázku přímo. Výřez o stejné šířce a výšce může obsahovat (a zpravidla obsahuje) různý počet znaků. Využívá se rozdělení vstupního obrázku na úseky, které jsou anotovány podle výskytu znaků. Je výhodné nechat samotnou síť, aby se anotaci naučila na základě standardního přepisu. Toto chování umožňuje tzv. CTC (*Connectionist temporal classification*) vrstva [9]. Jejím výstupem je matice $T \times C$, kde T značí počet částí na který je rozdělen vstupní výřez a C počet znaků v abecedě rozšířenou o prázdný znak (*blank*). Abeceda je množina znaků, kterou dokáže neuronová síť rozpoznat. Každá část je reprezentována vektorem hodnot, kde jednotlivé hodnoty představují pravděpodobnost jednotlivých znaků.

Z výstupní pravděpodobnostní matice lze vypočítat pravděpodobnost všech možných průchodů pro daný výřez. Princip CTC vrstvy je znázorněn na Obrázku 1. Pravděpodobnostní matice P pro vstupní výřez má dimenzi T jako sloupce. Každý sloupec obsahuje pravděpodobnost výskytu jednotlivých znaků v daném místě výřezu. Pro jednoduchost je abeceda omezena pouze na znaky vyskytujíce se ve vstupu a pravděpodobnost je vynásobena deseti. Sytě zelená barva značí nejpravděpodobnější průchod, oranžová průchod jiný, byť málo pravděpodobný. v některých sloupcích je světle zelenou barvou naznačena možnost záměny ze sytě zelenou. Řádek označený zelenou šipkou představuje znaky nejpravděpodobnějšího průchodu, výsledek je na posledním řádku (fialová šipka).

$$p(\pi|x) = \prod_{t=0}^{T-1} P_{t,i(\pi_t)} \tag{1}$$



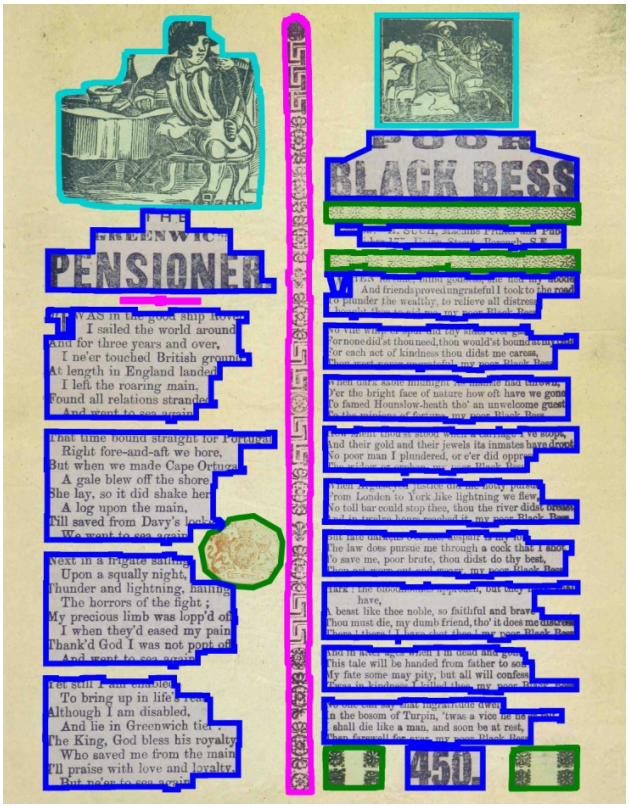
Obrázek 1. Získání přepisu při použití CTC vrstvy. Pravděpodobnostní matice P pro vstupní výřez má dimenzi T jako sloupce. Každý sloupec obsahuje pravděpodobnost výskytu (vynásobenou deseti) jednotlivých znaků v daném místě výřezu. Sytě zelená barva značí nejpravděpodobnější průchod, oranžová průchod jiný, byť málo pravděpodobný. Světle zelená naznačuje možnost záměny se sytě zelenou při zachování správného přepisu. Řádek označený zelenou šipkou představuje znaky nejpravděpodobnějšího průchodu. Výsledný přepis je na posledním řádku (fialová šipka).

Pravděpodobnost průchodu je dána vztahem 1, kde P je výstupní matice pravděpodobností, π průchod, x vstupní výřez a $i(\pi_t)$ udává index znaku průchodu pro pozici t. Nejjednodušší a zároveň rychlý způsob, jak najít přijatelně dobrý přepis je zvolit nejpravděpodobnější průchod a ten upravit na přepis.

Výhoda této vrstvy spočívá v možnosti využití anotací v podobě, která je přirozená člověku. Zároveň je díky pravděpodobnostní matici schopna podávat poměrně detailní přehled o kvalitě klasifikace.

3. Dataset IMPACT

Dataset IMPACT [1] obsahuje historické dokumenty z deseti evropských knihoven různých zemí. Každá knihovna měla za úkol poskytnout 50000 obrázků stránek knih, novin nebo jiných dokumentů. Texty obsažené ve vybraných dokumentech reprezentují danou knihovnu tzn. obsahují texty různých typů. Texty pochází z období od patnáctého do začátku dvacátého století, většina textů pochází z devatenáctého a začátku dvacátého století. Rozmanitost je také dána 9 jazyky a 10 fonty. Dataset tedy realisticky pokrývá všechny knihovny a tím pádem historické texty dostupné v rámci Evropy.



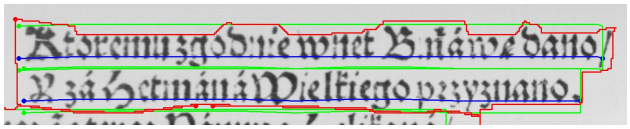
Obrázek 2. Ukázka stránky datasetu IMPACT s vyznačenými polygony obalujícími jednotlivé objekty stránky (převzato z [1]).

Anotována byla pouze podmnožina obrázků stránek o velikosti 45000. Anotace obsahují zejména polygony těsně obalující jednotlivé textové objekty stránek (odstavce, nadpisy atd.) s příslušným přepisem textu. Obrázek 2 ukazuje stránku datasetu IMPACT s vyznačenými polygony obalujícími jednotlivé objekty.

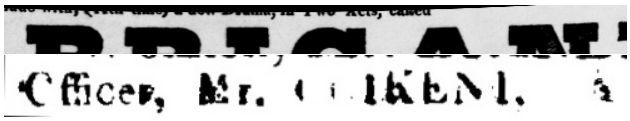
3.1 Zpracování datasetu

Jako první jsem se zaměřil na analýzu anotací v rámci datasetu. Anotace využívají celkem 636 znaků z nichž mnohé mají velmi malý výskyt tzn. jednotky. Pro urychlení trénování neuronových sítí jsem ponechal znaky, jejichž výskyt je větší nebo rovný 100. Dále se v datasetu vyskytují ligatury, které jsem se z hlediska nejednoznačnosti rozhodl rozložit na jednotlivé znaky. Nejednoznačnost spočívá v nemožnosti vizuálně rozlišit, jestli se jedná o tři následující znaky „ffi“ nebo jeden unicode znak představující trojici „ffi“. Dále zde byl problém různých druhů pomlček, kdy pomlčka vypadající vizuálně stejně byla anotována několika různými unicode znaky. Stejně vypadající pomlčky jsem sloučil do jednoho znaku. Těmito úpravami jsem počet znaků zmenšil na 263.

Řádky byly detekovány v rámci odstavců strá-



Obrázek 3. Ukázka detekce řádků na stránce datasetu IMPACT. Kde červený polygon značí odstavec, modrá linka základní linku textu (*baseline*) a zelené linky horizontální ohraničení řádku.



Obrázek 4. Ukázka špatných řádků datasetu IMPACT detekovaných neuronovou sítí. Horní řádek byl špatně vyřezán, dolní je částečně nečitelný.

nek za pomoci neuronových sítí. Výstupem těchto sítí jsou souřadnice bodů linky, na které leží řádek a linek horizontálně ohraničujících řádek viz 3. Kde červený polygon značí odstavec, modrá linka základní linku textu (*baseline*) a zelené linky horizontální ohraničení řádku (odhad výšky). Celkem bylo získáno 1301026 řádků. Tato detekce byla provedena Ing. Oldřichem Kodymem v rámci týmu pracujícího na projektu PERO[1].

Detekce linek řádků nemusí být zaručeně správná, a tudíž můžou vznikat výřezy, které obsahují různě deformované řádky. Dále mohou výřezy obsahovat zcela nečitelné posloupnosti znaků. Také anotace mohou být nepřesné nebo neúplné. Takovéto výřezy (viz. 4) značně komplikují průběh trénování sítí a znemožňují dosažení větších přesností. Po dosažení úspěšnosti 98.5 % na testovací i trénovací sadě jsem tuto síť použil pro odstranění řádků s největší chybovostí tzn. výstup chybové funkce (CTC) byl největší. Řádky byly sestupně seřazeny jak podle chyby, tak podle chyby normalizované délkou řádku. Vizuálně jsem odhadnul míru odstranění špatných výřezů na 20000 nejhorších řádků v rámci obou seřazení. Dále jsem odstranil všechny řádky, které měly chybu větší než 0.25 tzn. více než čtvrtina znaků byla špatně rozpoznána. Častou chybou byly výřezy, kterým chyběl začátek či konec, a tudíž anotace nebyla přesná. Takové řádky jsem detekoval pomocí Levenshteinovi vzdálenosti anotace a přepisu generovaného sítí. Celkem bylo těmito přístupy odstraněno více než 60000 podezřelých řádků.

Negativní dopad takovéto filtrace spočívá v odstranění části správně anotovaných a vyřezaných řádků, které jsou těžší na rozpoznání. Cílem je však získat co nejkvalitnější množinu trénovacích dat. Případná složitost vlivem různých poškození může být simulována

[1] Pokročilá extrakce a rozpoznávání obsahu tištěných a rukou psaných digitalizátů pro zvýšení jejich přístupnosti a využitelnosti

	es	nl	en	sl	pl	fr	bg	it	cs
D	14	38	43	29	29	12	5	1	1
Ř	353k	249k	176k	166k	99k	81k	79k	11k	2k
Z	20M	17M	11M	10M	7M	5M	5M	0.6M	0.2M
P	58	67	62	62	70	65	65	53	68

Tabulka 1. Přehled mapování jazyků dokumentů datasetu IMPACT na řádky textu. D je počet dokumentů, Ř počet řádků, z počet znaků a P je poměr počtu znaků ku počtu řádků.

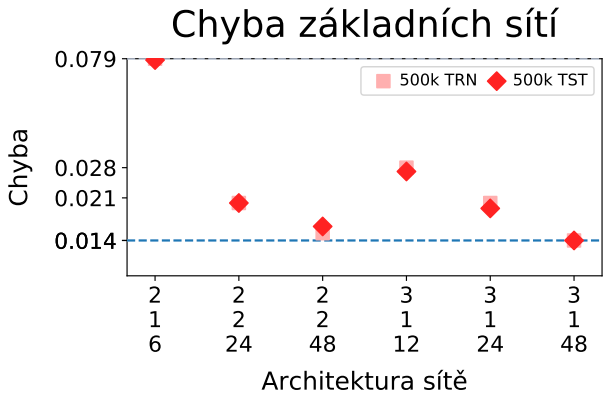
uměle, tím pádem je dosaženo větší kontroly, která je důležitá z hlediska experimentů.

Filtrace a odhalování chyb není jednoprůchodová záležitost tzn. při získání nové, lépe fungující sítě se provádí opětovné vyhodnocení datasetu a zkoumají se problémy. Tímto postupem je získán dataset, který má minimální počet špatně anotovaných řádků. Výřezů, které splňují všechny dané omezení je celkem 1217000. Poměr velikosti trénovací a testovací sady je 99/1.

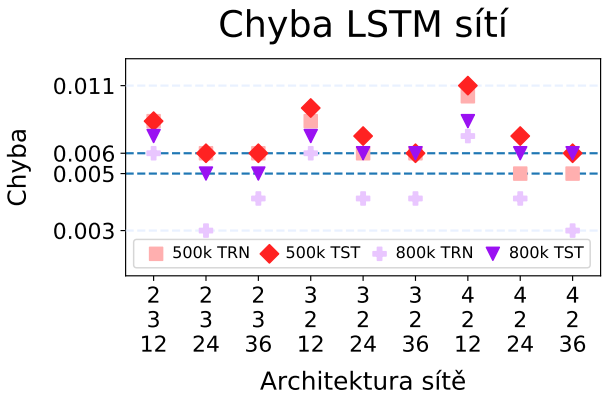
Pro experimenty s adaptacemi jsem vytvořil mapování různých vlastností dokumentů na řádky textu. Přehled mapování jazyků dokumentů je v Tabulce 1. D je počet dokumentů, Ř počet řádků, z počet znaků a P je poměr počtu znaků ku počtu řádků. Pro určení jazyku dokumentů jsem využil knihovnu pro detekci jazyků *langdetect* [12]. Dalšími mapování jsou typ dokumentu (kniha, noviny, úřední dokument), typ písma a kvalita. Typ písma může být *normal* (podobné současnému tištěnému písmu), *historic* (staré písmo jako je např. gotické písmo), *cyrillic* a *calligraphy* (písmo podobající se krasopisu, typicky použito v rámci jiného typu písma). Tyto mapovávní jsem vytvořil na základě několika stránek daného dokumentu manuálně. Dokument může mít více typů písma. Kvalita je hodnocena na základě čitelnosti dokumentu. Hodnocení je od 1 do 5, kdy 5 je nejhorší. Dataset je ze tří čtvrtin tvořen knihami, nejvíce je zastoupený normalní typ písma a většina dokumentů je bez problémů čitelná (kvalita 1 nebo 2).

4. Architektury

Architektury sítí dělím na dvě hlavní části, a to podle toho, zda obsahují obousměrnou rekurentní LSTM vrstvu nebo ne. Každá architektura má své značení v podobě tří čísel *BC* (počet bloků), *BLC* (počet konvolučních vrstev v rámci bloku) a *BFC* (počet jader první konvoluční vrstvy s indexem 0). Vztah $BFC * 2^i$ udává počet jader konvolučních vrstev v rámci dalších bloků, kde i je index bloku. Za bloky (počáteční část sítě) následují dvě konvoluční vrstvy, z nichž druhá má na výstupu 1D příznakové mapy, z důvodu následující LSTM (volitelné) a CTC vrstvy. Výsledná



Obrázek 5. Výsledky experimentů základních sítí na datasetu IMPACT. Světle červené čtverečky představují chybu na trénovací sadě, červené kosočtverce pak na sadě testovací. Osa představující chybu je v logaritmickém měřítku.



Obrázek 6. Výsledky experimentů LSTM sítí na datasetu IMPACT. Světlé symboly představují chybu na trénovací sadě, tmavé pak na sadě testovací. Červená barva představuje počátečních 500000 iterací, fialová dotrénování pomocí dalších 300000 iterací. Osa představující chybu je v logaritmickém měřítku.

pravděpodobnostní matice udává pravděpodobnost jednotlivých znaků každé 4px. z tohoto důvodu je někdy nutné přidat na konec sítě další konvoluční vrstvy, které zvětší (roztáhnou) výstup. v následujících experimentech porovnávám sítě pomocí chyby na trénovací a testovací sadě. Chyba značí procentuální chybu ve znacích podělenou 100. Chyba pro trénovací sadu je průměrem náhodných 10000 řádků z této sady, chyba pro testovací sadu je průměrem chyby všech řádků.

4.1 Experimenty

Nejprve jsem natrénoval základní sítě bez LSTM vrstvy viz graf na Obrázku 5. Chyba na trénovací sadě je znázorněna pomocí světle červených čtverečků, chyba na testovací pomocí červených kosočtverců. Celkem bylo provedeno 500000 iterací, kdy každá iterace znamenala vyhodnocení 16 výřezů (*batch size* 16). Učící konstanta (*learning rate*) byla nastavena na 0.0003 a dropout na 4 % (každý blok má dropout na svém konci). Pro samotné trénování jsem využil prostředí TensorFlow [13] a optimalizační metodu založenou na gradientním sestupu *Adam* [14]. Výsledky se zlepšují s rostoucí velikostí sítí tzn. počtem trénovaných vah. Úspěšnost na trénovací i testovací sadě je srovnatelná, sítě úspěšně generalizují a nedochází k přetrénování. Nejlepší síť má průměrnou chybu 1 znak na 70, tento výsledek je nedostačující. Vzhledem k velkému datasetu a dobré generalizaci by tedy bylo vhodné pokračovat se zvětšováním sítí. Největší sítě v rámci tohoto experimentu jsou však již tak náročné na výpočet, že se vyplatí přidání LSTM vrstvy (která díky sekvenčnímu vyhodnocení obecně zpomaluje vyhodnocení sítě).

Graf na Obrázku 6 prezentuje výsledky pro sítě s vrstvou LSTM. Světle červené čtverečky opět představují chybu na trénovací sadě a červené kosočtverce na testovací. Parametry trénování jsou stejné jako u základních sítí. Je patrné značné zlepšení, a to i pro sítě výpočetně srovnatelně. Sítě opět dobře generalizují, byť přetrénování je mírně znatelnější (ovšem prakticky stále zanedbatelné). Vzhledem k tomu, že provedení všech iterací pro největší sítě v této kategorii trvá přibližně jeden den, nepokračoval jsem s dalším zvětšováním sítí. Namísto toho jsem se rozhodl sítě dotrénovat (*finetuning*). Bylo provedeno dalších 300000 iterací, *dropout* byl nastaven na 0, *learning rate* zmenšen na polovinu a velikost datového vzorku (*batch size*) zvětšena dvojnásobně. Výsledky jsou znázorněny pomocí fialových symbolů, kde světlé plus představuje chybu na trénovací sadě a tmavý trojúhelník na testovací. Je patrné větší přetrénování a mírné zlepšení výsledků. Nejlepší síť chybuje v 1 znaku z 200. Tato chyba by mohla být způsobena pouze 60 absolutně nečitelnými řádky (při průměrné délce řádku 20 znaků, což je podhodnoceno) v 12000 testovacích řádcích. Při vizuálním pozorování chybových řádků, šlo většinou o chyby plynoucí ze špatně čitelných až nečitelných řádků, a to i z lidského hlediska. Dalším problém jsou již zmiňované špatně ořezané řádky nebo anotace. Pokud je řádek dobře čitelný, síť prakticky nechybuje. Příklady rozpoznání horších řádků jsou na úvodním obrázku této práce.

5. Adaptace

Není možné natrénovat neuronovou síť, která pokryje veškerý současný a budoucí text. z tohoto důvodu je žádoucí mít mechanizmy, které umožní adaptovat stávající neuronovou síť na nové texty. Zpravidla existuje velmi rozsáhlý původní dataset, který ale neob-

sahuje texty, které je potřeba rozpoznat. Datová sada adaptačních textů je naopak velmi malá a musí být manuálně vytvořena.

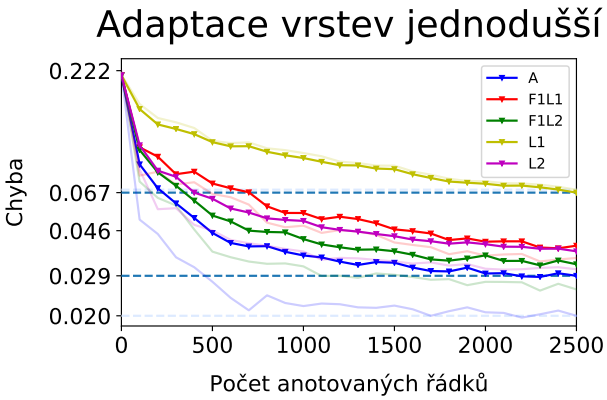
Trénovaní neuronové sítě na velmi malém datasetu (např. 100 řádků) je problematické z hlediska přetrénování. z tohoto důvodu jsem se rozhodl namnožit poslední vrstvu a každou z těchto vrstev trénovat a testovat na jiném vzorku dat tzv. křížová validace (*cross-validation*). Tímto způsobem je dosaženo větší kontroly přetrénování. Další kontrolu představuje míchání původních a adaptačních dat tzn. síť se trénuje na všech dostupných datech s tím, že zastoupení nových dat se postupně zvětšuje. Jestliže se síť začne přetrénovávat, dojde k přidání adaptačních dat do trénovací množiny (simulace nového anotování uživatelem). Poměr původních a adaptačních dat je obnoven na minulou hodnotu, rovněž jsou obnoveny váhy sítě do stavu, kdy ještě nebyla přetrénována. Výběr nových dat může být náhodný nebo založen na přístupech aktivního učení [15][16][17]. Aby nedošlo k nekonečnému čekání na přetrénování sítě, dojde po určitém počtu adaptivních iterací k rozšíření adaptačních dat. Podrobněji je tento přístup uveden v pseudo kódu 1. *AL* značí metodu aktivního učení. Většina volaných funkcí má mnoho parametrů, pro jednoduchost jsem ponechal pouze ty nejdůležitější.

Algorithm 1 Adaptation

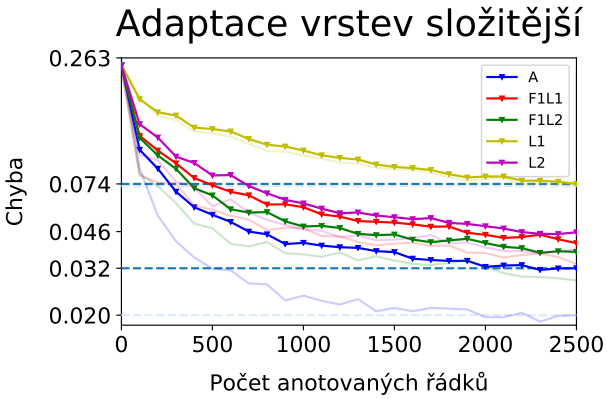
```
1:  function ADAPT(net, origData, adaptData, AL)
2:      pickedAdaptData = []
3:      while size(pickedAdaptData) < maxPAD do
4:          pickAdaptData(AL)
5:          numberOfIter = 0
6:          adaptAttempts = 0
7:          while numberOfIter < maxIter do
8:              if notOverfitted() then
9:                  if adaptAttempts > maxAA then
10:                     increaseRatio()
11:                     adaptAttempts = 0
12:                 TrainNet(pickedAdaptData)
13:                 numberOfIter+ = adaptIter
14:                 SaveNet()
15:                 adaptAttempts+ = 1
16:             else
17:                 restorePreviousRatio()
18:                 restorePreviousNet()
19:                 break
```

5.1 Experimenty

Pro experimenty s adaptací jsem zvolil LSTM síť s architekturou 2_3_24. Tuto síť jsem natrénoval na dvou odlišných datových sadách. První sada neobsahuje



Obrázek 7. Adaptace sítě trénované na neslovanských dokumentech na polské historické knihy. *A* představuje všechny vrstvy, *Fn* a *Ln* počet počátečních/koncových vrstev sítě, kde *n* je daný počet. Osa představující chybu je v logaritmickém měřítku.

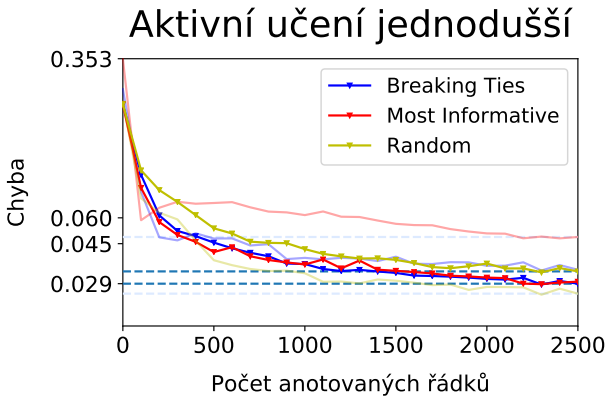


Obrázek 8. Adaptace sítě trénované na neslovanských dokumentech bez historického typu písma na polské historické knihy. *A* představuje všechny vrstvy, *Fn* a *Ln* počet počátečních/koncových vrstev sítě, kde *n* je daný počet. Osa představující chybu je v logaritmickém měřítku.

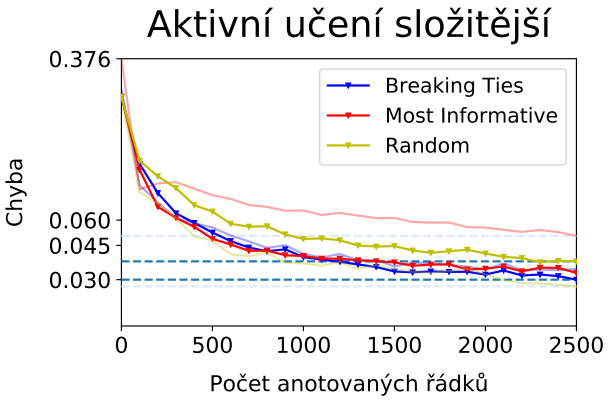
žádné slovanské jazyky tzn. polštinu, slovinštinu a češtinu. Druhá sada navíc neobsahuje žádné dokumenty s historickým typem písma. Trénovací a testovací chyba první sítě je 0.002 a 0.004, druhé sítě pak 0.003 a 0.005. Sada, na kterou adaptuji se skládá pouze s polských dokumentů, kde typ písma je pouze historický. Chyba na těchto datech je 0.21 pro první síť a 0.24 pro druhou.

Poměr původních a adaptačních dat a krok jakým se tento poměr zvyšuje jsem nastavil na 0.01. Přetrénování je definováno jako překročení velikosti validační chyby (*loss*) o 5 % vzhledem k trénovací chybě. Maximální počet anotovaných řádků je 2500 s tím, že nové řádky se přidávají po stovkách. Poslední vrstva sítě byla namnožena třikrát tzn. síť má 4 výstupní vrstvy.

Grafy na obrázcích 7 a 8 znázorňují výledky při adaptaci různého počtu vrstev. První graf je pro síť



Obrázek 9. Adaptace sítě trénované na neslovanských dokumentech na polské historické knihy. *Breaking Ties* vybírá takové řádky, které mají nejmenší absolutní rozdíl pravděpodobností prvního a druhého průchodu pravděpodobnostní maticí vrstvy CTC. *Most Informative* vybírá řádky, jejichž přepis je nejméně jistý. *Random* značí náhodný výběr. Osa představující chybu je v logaritmickém měřítku.



Obrázek 10. Adaptace sítě trénované na neslovanských dokumentech bez historického typu písma na polské historické knihy. *Breaking Ties* vybírá takové řádky, které mají nejmenší absolutní rozdíl pravděpodobností prvního a druhého průchodu pravděpodobnostní maticí vrstvy CTC. *Most Informative* vybírá řádky, jejichž přepis je nejméně jistý. *Random* značí náhodný výběr. Osa představující chybu je v logaritmickém měřítku.

trénovanou na první sadě, druhý graf pak pro síť trénovanou na sadě druhé. *A* představuje všechny vrstvy, *Fn* a *Ln* počet počátečních/koncových (namnožená poslední vrstva je brána jako jeden celek) vrstev sítě, kde *n* je daný počet. Tmavé linky značí chybu na nezávislé testovací sadě, příslušné světlé linky chybu na validačních datech (chyba určená pomocí křížové validace). Vykreslené hodnoty jsou průměrem výsledků 4 výstupních vrstev. Je patrné, že čím více vrstev se adaptuje, tím jsou výsledky na testovací sadě lepší. Na druhou stranu dochází ke větší ztrátě kontroly nad procesem adaptace, neboť výsledky na validačních datech se liší čím dál více (světlé linky).

Grafy na obrázcích 9 a 10 znázorňují výledky pro adaptaci s vrstvami *F*1*L*2 pro různý způsob výběru nových řádků pro anotaci. *Breaking Ties* [16][18] vybírá takové řádky, které mají nejmenší absolutní rozdíl pravděpodobností prvního a druhého průchodu pravděpodobnostní maticí vrstvy CTC. *Most Informative* [19] vybírá řádky, jejichž průchod pravděpodobnostní maticí je nejméně jistý. *Random* značí náhodný výběr. Náhodný výběr je v obou případech horší než přístupy aktivního učení. *Breaking Ties* má oproti ostatním přístupům výhodu v lepší kontrole průběhu trénování tzn. validační a testovací chyba jsou po celou dobu velmi podobné.

Jelikož je při adaptaci použit velmi malý poměr adaptačních a původních dat, úspěšnost sítě na původních datech zůstává stejná tzn. síť se pouze naučí nový typ písma a jazyk. Největším problém představuje přetrénování, ke kterému dochází už po malém počtu iterací s daným počtem řádků (řádově stovky iterací).

Dosažené chyby se pohybují kolem 0.03. Optimální výsledek by se blížil hodnotě 0.016, což je výsledek sítí z části 4.1.

6. Závěr

Cílem této práce bylo experimentálně porovnat různé architektury neuronových sítí pro rozpoznávání textu, dále pak srovnání různých přístupů adaptace těchto sítí. Za tímto účelem byl zpracován rozmanitý a rozsáhlý dataset IMPACT. Rekurentní sítě dosahují lepších výsledků, a to i vzhledem k rychlosti než čistě konvoluční sítě. Čím více vrstev sítě je adaptováno na adaptační data, tím dochází k většímu přetrénování a tím pádem není adaptace pod kontrolou. Nadruhou stranu takto přetrénované sítě fungují o něco lépe než sítě, které se nepřetrénují. Na adaptaci má také vliv volba nových adaptačních řádků. Pokud je pro výběr použita stávající neuronová síť, je zpravidla dosaženo lepších výsledků než pro výběr náhodný.

Zpracováním datasetu IMPACT bylo celkem získáno 1217000 řádků. Úspěšnost na testovacích datech dosahuje 98.6 % pro čistě konvoluční sítě, pro rekurentní sítě pak 99.5 %. Úspěšnost na testovacích a trénovacích datech je velmi podobná tzn. sítě dobře generalizují. Adaptace sítí učených na neslovanských dokumentech na sadu polských historických textů dosahuje úspěšnosti 97 % z původní úspěšnosti 79 %. Pro adaptaci bylo postupně použito 2500 řádků. Úspěšnost sítě natrénované na všech datech na adaptačních datech je 98.4 %.

Pro zpracování velkého datasetu je vhodné průběžně využívat již natrénované neuronové sítě tak, aby bylo odhaleno co možná nejvíce chyb v trénovacích datech. Jedna neuronová síť je schopna zároveň rozpoznávat tištěné texty v mnoha jazycích a fontech. Také tolerance na kvalitu výtisku je poměrně veliká tzn. chyby se vyskytují výhradně u nečitelných řádků. Rekurentní vrstvy značně zvyšují úspěšnost sítě. Úspěšnost sítí na nových datech lze výrazně zvýšit za použití poměrně malého množství dat (stovky anotovaných řádků). Efektivnost křížové validace klesá s navyšováním počtu vrstev, které se podílí na adaptaci (důvodem je pravděpodobně sdílení většiny vrstev). Při adaptaci je vhodné využívat přístupů aktivního učení.

Tento článek shrnuje dosavadní práci na mé diplomové práci. Hlavním cílem dalšího výzkumu je adaptační mechanizmus, který dosáhne co nejlepších výsledků pro co nejmenší počet řádků, jelikož anotování řádků je nejdražší. v současném stavu je největší problém kontrola přetrénování. Sítě se přetrénovávají velmi rychle tzn. potřeba nových řádků je velká. Vhodnými experimenty k provedení jsou: různé poměry původních a adaptačních dat a práh přetrénování. Rovněž sdílení vrstev v rámci křížové validace snižuje její efektivitu, protože se sdílená část sítě učí na všech validačních datech. Řešením by bylo trénování několika samostatných sítí, což by ale bylo časově velmi náročné. Jiným řešením by bylo namnožení více vrstev na konci sítě. Velmi důležitý je také výběr nových adaptačních dat pomocí metod aktivního učení tak, aby se na těchto datech byla síť schopna naučit kvalitně řešit nový problém.

Poděkování

Děkuji členům týmu PERO za ochotu a pomoc, zejména pak Ing. Michalu Hradišovi Ph.D. a Ing. Oldřichu Kodymovi.

Literatura

[1] Christos Papadopoulos, Stefan Pletschacher, Christian Clausner, and Apostolos Antonacopoulos. The impact dataset of historical document images. In *Proceedings of the 2Nd International Workshop on Historical Document Imaging and Processing*, HIP '13, pages 123–130, New York, NY, USA, 2013. ACM.

[2] Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, and Soumya K. Ghosh. *Optical Character Recognition Systems for Different Languages with Soft Computing*, volume 352 of *Studies in Fuzziness and Soft Computing*. Springer, 2017.

[3] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait. High-performance OCR for printed english and fraktur using LSTM networks. In *2013 12th International Conference on Document Analysis and Recognition*, pages 683–687, Aug 2013.

[4] Anupama Ray, Sai Rajeswar, and Santanu Chaudhury. Text recognition using deep BLSTM networks. 01 2015.

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

[6] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for OCR in the wild. *CoRR*, abs/1603.03101, 2016.

[7] Rathin Radhakrishnan Nair, Nishant Sankaran, Bharagava Urala Kota, Sergey Tulyakov, Srirangaraj Setlur, and Venu Govindaraju. Knowledge transfer using neural network based approach for handwritten text recognition. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 441–446. IEEE, 2018.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[11] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994.

[12] Nakatani Shuyo. Language detection library for java, 2010.

[13] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz

Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] Burr Settles. *Curious machines: Active learning with structured instances*. PhD thesis, University of Wisconsin–Madison, 2008.

[16] Fabian Stark, Caner Hazırbaş, Rudolph Triebel, and Daniel Cremers. Captcha recognition with active deep learning. 09 2015.

[17] Melanie Ducoffe and Frédéric Precioso. QBDC: query by dropout committee for training deep supervised architecture. *CoRR*, abs/1511.06412, 2015.

[18] M.M. Al Rahhal, Yakoub Bazi, Haikel AlHichri, Naif Alajlan, Farid Melgani, and R.R. Yager. Deep learning approach for active classification of electrocardiogram signals. *Information Sciences*, 345:340 – 354, 2016.

[19] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. pages 3–12, 1994.