
**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**


# AKTIVNÍ UČENÍ PRO ROZPOZNÁVÁNÍ TEXTU

**ACTIVE LEARNING FOR OCR**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** — **Bc. JAN KOHÚT**
**AUTHOR**

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



**BRNO 2019**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

# Zadání diplomové práce


Student: **Kohút Jan, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Aktivní učení pro rozpoznávání textu**
**Active Learning for OCR**
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy konvolučních neuronových sítí a aktivního učení.
2. Vytvořte si přehled o současných metodách využívajích aktivní učení a neuronové sítě se zaměřením na rozpoznávání textu a řeči.
3. Vyberte konkrétní metody a aplikace vhodné pro experimenty.
4. Implementujte navržené metody a proveďte experimenty nad vhodnou datovou sadou nejlépe historických dokumentů.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Štěpán Beneš: Aktivní učení s neuronovými sítěmi. Brno, 2018. Diplomová práce. VUT, FIT.
- Liwicki et al: A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In Proceedings of ICDAR, 2007.
- Graves et al.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. ICML, 2006.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 6. listopadu 2018

# Abstrakt

Cílem této práce je navrhnout metody aktivního učení a provést experimenty nad datovou sadou historických dokumentů. Pro experimenty využívám rozsáhlý a rozmanitý dataset IMPACT o více než jednom milionu řádků. Pomocí neuronových sítí provádím kontrolu vhodnosti řádků, tzn. čitelnosti a správnosti přepisů. Nejprve srovnávám architektury neuronových sítí, a to jak sítě čistě konvoluční, tak sítě obsahující obousměrnou rekurentní vrstvu LSTM. Dále se zabývám přístupy k učení neuronových sítí pomocí aktivního učení a samotnými metodami aktivního učení. Aktivní učení využívám zejména pro adaptaci neuronových sítí na jiné textové dokumenty, než na kterých byla původní síť učena. Aktivní učení tedy slouží k výběru vhodných adaptačních dat. Čistě konvoluční neuronové sítě dosahují úspěšnosti 98.6 %, rekurentní sítě pak 99.5 %. Chyba při adaptaci s využitím aktivního učení je o 26 % nižší než chyba při náhodném výběru dat.



# Abstract

The aim of this Master's thesis is to design methods of active learning and to experiment with datasets of historical documents. A large and diverse dataset IMPACT of more than one million lines is used for experiments. I am using neural networks to check the readability of lines and correctness of their annotations. Firstly, I compare architectures of convolutional and recurrent neural networks with bidirectional LSTM layer. Next, I study different ways of learning neural networks using methods of active learning. Mainly I use active learning to adapt neural networks to documents that the neural networks do not have in the original training dataset. Active learning is thus used for picking appropriate adaptation data. Convolutional neural networks achieve 98.6% accuracy, recurrent neural networks achieve 99.5% accuracy. Active learning decreases error by 26% compared to random pick of adaptations data.



# Klíčová slova

Aktivní učení, rozpoznávání textu, neuronové sítě, konvoluční neuronové sítě, rekurentní neuronové sítě, dataset IMPACT

# Keywords

Active learning, text recognition, neural networks, convolutional neural networks, recurrent neural networks, dataset IMPACT


# Citace

KOHÚT, Jan. *Aktivní učení pro rozpoznávání textu.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

# Aktivní učení pro rozpoznávání textu

## Prohlášení

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

.....
Jan Kohút
20. května 2019


## Poděkování

Především děkuji své rodině za podporu a pomoc při studiu. Dále děkuji členům týmu PERO, zejména za cenné rady a postřehy svému vedoucímu Ing. Michalu Hradišovi, Ph.D. Děkuji také Fakultě informačních technologií VUT za zpřístupnění výpočetního centra SGE.

# Obsah

**1 Úvod** **2**

**2 Neuronové sítě pro rozpoznávání textu** **3**
2.1 Rozpoznávání textu . . . . . 3
2.2 Neuronové sítě . . . . . 6
2.3 Shrnutí . . . . . 12

**3 Aktivní učení** **14**
3.1 Metody výběru vhodných dat . . . . . 15

**4 Aktivní učení neuronových sítí pro rozpoznávání textu** **21**
4.1 Architektury sítí . . . . . 21
4.2 Algoritmy aktivního učení . . . . . 22
4.3 Efektivní křížová validace . . . . . 26

**5 Dataset IMPACT** **28**
5.1 Zpracování datasetu . . . . . 28
5.2 Mapování datasetu . . . . . 31

**6 Experimenty** **33**
6.1 Architektury sítí . . . . . 33
6.2 Datové sady pro aktivní učení . . . . . 36
6.3 Přístupy učení . . . . . 37
6.4 Aktivní učení . . . . . 40
6.5 Shrnutí . . . . . 45

**7 Závěr** **47**

**Literatura** **48**

**A Obsah SD** **51**

# Kapitola 1

# Úvod

Rozpoznávání textu je problém převodu obrazu tištěného nebo psaného textu na text, se kterým lze pracovat v rámci počítače. Převodu je typicky dosaženo pomocí algoritmů pro zpracování obrazu a strojového učení. Nejlepších výsledků v této oblasti dosahují neuronové sítě, jejichž kvalita je závislá na množství dat, které lze použít pro učení. Získávání takovýchto dat je časově a finančně náročný proces. Je proto žádoucí naučit dostatečně kvalitní neuronové sítě pomocí co nejmenšího množství dat. Vybraná data tudíž musí být dostatečně reprezentativní vzhledem k danému problému. Následující text se zabývá tím, jak takováto data získat a co nejlépe je využít pro dosažení požadovaných výsledků, zejména pak pro problém rozpoznávání textu. Jelikož jsou neuronové sítě učeny průběžně s tím, jak jsou získávána nová data, nazývá se tento přístup aktivní učení. Ačkoliv tento přístup není nový, dosavadní výzkum se věnoval především jednoduchým metodám, které byly testovány na jednodušších klasifikátorech. Rovněž zájem o aktivní učení pro rozpoznávání textu není velký.

Tato práce vznikla v rámci projektu PERO[1], jehož cílem je mimo jiné kvalitně rozpoznávat tištěný a psaný historický text v českém jazyce. K dispozici je velké množství obrázků textů, ale chybí potřebné přepisy pro učení. Není jasné, jakým způsobem vybírat vhodné obrázky pro anotaci. Cílem je tedy navrhnout a vyzkoušet metody aktivního učení pro rozpoznávání textu, které tento výběr provádějí na základě neuronové sítě.

Nejprve se věnuji problému rozpoznávání textu a popisu neuronových sítí, které řeší samotnou klasifikaci. Tyto neuronové sítě využívají konvoluční a rekurentní vrstvy. Dále se zabývám popisem již existujících metod aktivního učení, které využívají modelů strojového učení pro výběr vhodných dat pro anotaci. Představuji dva algoritmy aktivního učení, které proces učení do značné míry automatizují. Pro kvalitnější výběr vhodných dat využívám síť, která má více výstupních vrstev. Tuto síť trénuji pomocí efektivní křížové validace. Experimenty provádím s rozsáhlým datasetem IMPACT, který obsahuje více než 1.2 milionu řádků. Tento dataset jsem pomocí neuronové sítě zpracoval tak, aby obsahoval co nejméně chybných dat. Závěrem provádím experimenty jak s klasickým trénováním neuronových sítí, tak s aktivním učením. Rekurentní neuronové sítě dosahují větší úspěšnosti než sítě čistě konvoluční. Metody aktivního učení vybírají vhodnější data pro učení než výběr náhodný. Díky vhodnému výběru pak sítě dosahují větší úspěšnosti v rozpoznávání textu.

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

# Kapitola 2

# Neuronové sítě pro rozpoznávání textu

Programy umožňující rozpoznávání textu jsou obecně známe pod pojmem OCR[1]. Samotné rozpoznávání se skládá z více částí, kterým se podrobněji věnuji v podkapitole 2.1. Jednotlivé OCR se liší přístupem k těmto částem, např. algoritmy pro nalezení textu v obrazu a algoritmy pro následnou klasifikaci mohou být různé. Přístup k dílčím částem může mít různý vliv na kvalitu výsledného rozpoznání. Je tedy vhodné mít alespoň základní představu o těchto částech vzhledem k aktivnímu učení. V této práci se především zaměřuji na samotnou klasifikaci tzn. nástroje na získání přesných výřezů textu z obrazu jsou k dispozici. Klasifikaci je možné provádět mnoha způsoby. Od naivního srovnávání sady vzorových obrázků znaků až po skryté Markovovi modely (HMM) a neuronové sítě [5]. Právě neuronové sítě dosahují v dnešní době nejlepších výsledků v této oblasti [18, 4, 26, 24].

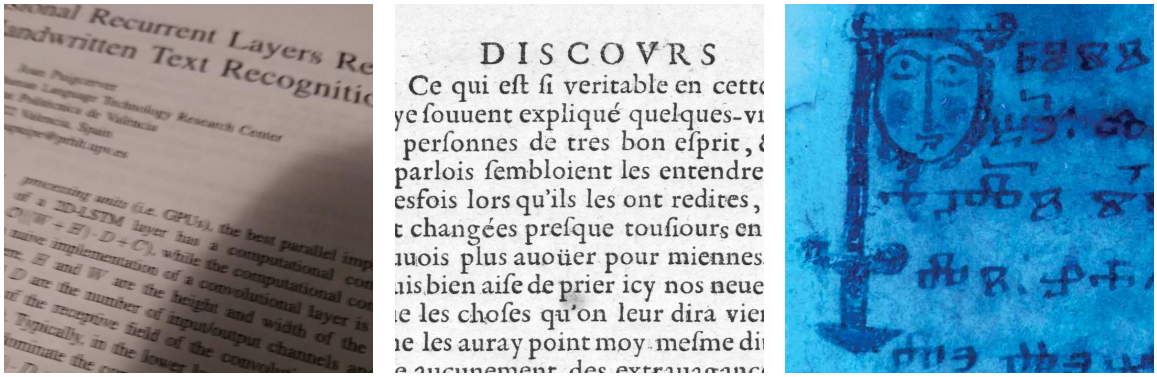
## 2.1 Rozpoznávání textu

V této podkapitole popisuji rozpoznávání textu podle částí, na které je tento problém rozdělen v rámci projektu PERO. Toto rozdělení dodržuje standardní postupy při zpracování obrazu, neliší se tedy nijak zásadně od rozdělení jiných [5]. K jednotlivým částem přistupuji z hlediska jejich významu v rámci rozpoznávání textu. Kromě části věnující se klasifikaci, které se budu samostatně věnovat v podkapitole 2.2, nepopisuji metody pomocí nichž se dosahuje daných výsledků. Těchto metod je velké množství a jejich podrobný popis nemá z hlediska této práce velký význam.

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 apod. 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.

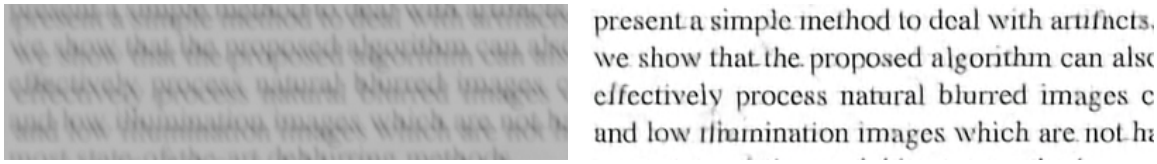
**Získávání obrazu.** První část spočívá v samotném zachycení obrazu textu, který má být rozpoznán. Z hlediska úspěšné klasifikace je důležité, aby bylo toto zachycení prováděno pokud možno stejným způsobem, jak při získávání dat pro učení klasifikátoru, tak

[1] Optical character recognition - optické rozpoznávání znaků




(a) (b) (c)

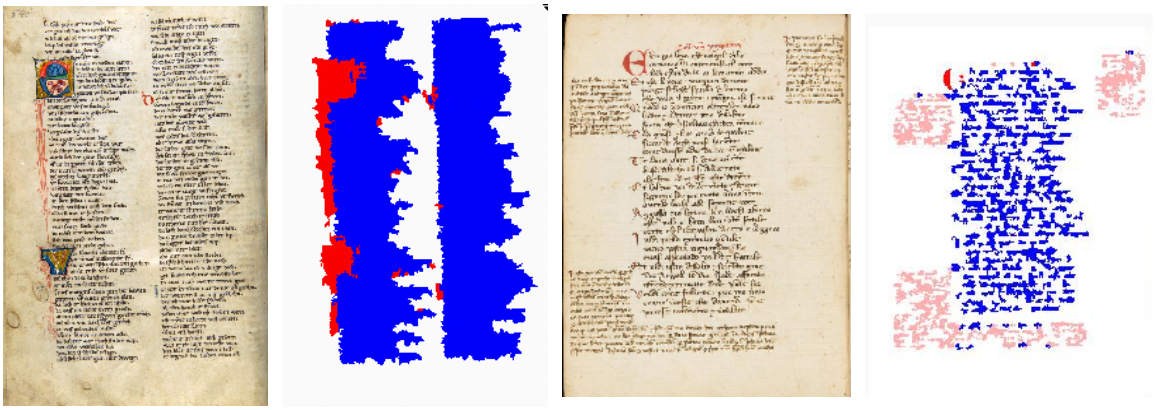
Obrázek 2.1: Výřezy obrazu textu pořízené různým způsobem. Fotografie 2.1a je pořízena mobilním telefonem, je znatelné nerovnoměrné nasvícení, nevhodné natočení a rozmazaní. Sken 2.1b je naopak rovnoměrně nasvícený, vhodně natočený a ostrý. Poslední snímek 2.1c byl získán multispektrálním snímkováním původně nečitelného dokumentu (převzato z [2]).




Obrázek 2.2: Doostření rozmazaného textu pomocí neuronové sítě. Původní rozmazaný text vlevo, výsledek vpravo (převzato z [14]).

při samotném používání v praxi. Pokud má být výsledné OCR používáno v mobilním zařízení, je vhodné použít při jeho navrhování fotografie z takovýchto zařízení. Tyto fotografie budou pravděpodobně do značné míry odlišné od obrazu získaného profesionálním skenerem. Nejedná se pouze o problém různých kvalit pořízených snímku, ale taky o možnost různého natočení, velikosti písma, rozmazání, nasvícení apod. V rámci historických dokumentů, které byly výrazně poškozeny a staly se tak nečitelnými, lze použít multispektrální snímkování. Příklady snímků získaných různým způsobem jsou na obrázku 2.1. Na fotografii 2.1a pořízené mobilním telefonem je znatelné nerovnoměrné nasvícení, nevhodné natočení a rozmazaní. Sken 2.1b je naopak rovnoměrně nasvícený, vhodně natočený a ostrý. Poslední snímek 2.1c byl získán multispektrálním snímkováním původně nečitelného dokumentu. Jednotlivé snímky jsou do značné míry vizuálně odlišné. Při vývoji OCR je tedy důležité zvážit jejich specifika a následující části rozpoznávání jim přizpůsobit.

**Předzpracování.** Předzpracování obrazu je taková změna obrazu, kdy výsledný obraz je jednodušší pro zpracování z hlediska metod, které se používají v dalších částech rozpoznávání. Např. pokud je v daném obraze text jenom v nějaké části, je vhodné vyříznout pouze tuto část. Další metody pak mohou spoléhat na tento fakt a fungovat díky tomu lépe. Rovněž natočení textu, nasvícení, rozmazání, šum apod. mohou být do jisté míry opraveny. Obrázek 2.2 znázorňuje doostření textu pomocí neuronové sítě. Původní rozmazaný text vlevo nevypadá z čistě vizuálního hlediska jako text, ale spíše jako rozmazané čáry, které text připomínají. Nelze tedy očekávat, že algoritmy pracující s obrazem textu budou kvalitně fungovat. Úkolem této části rozpoznávání by tedy ideálně měla být úprava obrazu do



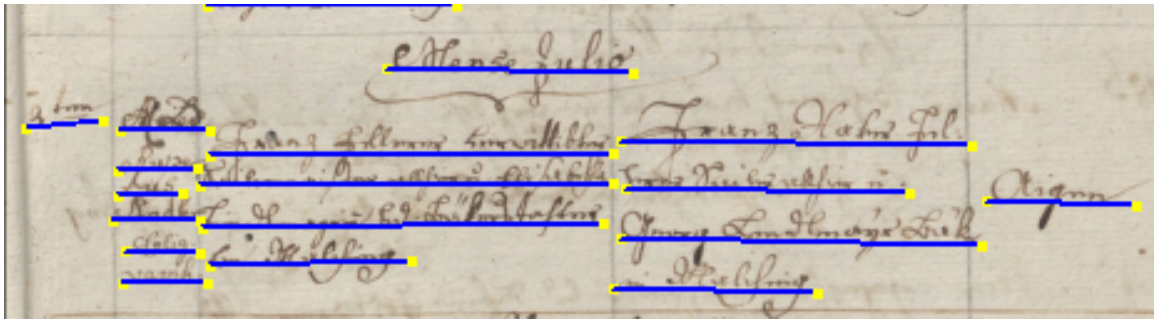
Obrázek 2.3: Ukázka segmentace stránek. Levý obrázek z dvojice je původní strana, pravý pak segmentace provedená pomocí neuronové sítě. Bílá značí stránku, červená dekorace a růžová komentáře (převzato z [6]).

takové podoby, aby byl dobře čitelný, stejně natočený a rovnoměrně nasvícený. Na druhou stranu, pokud jsou k dispozici kvalitní skeny kde je toto zaručeno, mohou vést zbytečné úpravy ke zhoršení původního obrazu.

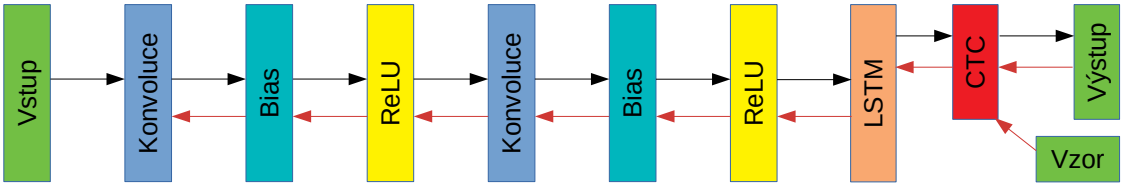
**Segmentace.** Tato část se zabývá nalezením zajímavých oblastí v rámci obrazu textu jako jsou odstavce, nadpisy, sloupce, číslování apod. Toto zpracování je typické pro skeny stránek knih. Po úspěšné detekci oblastí jsou k dispozici souřadnice určující jejich pozice v rámci stránky. Oblast odstavce pak slouží k detekování samotných řádků textu. Pokud by segmentace nebyla provedena, mohly by nastat nežádoucí detekce. Např. pokud má strana textu dva sloupce, byl by jako jeden řádek detekován řádek jdoucí skrz celou stranu, tedy přes oba sloupce (volné místo mezi sloupci by bylo považováno za oddělovač). Výsledný přepis stránky by tak nedával smysl. Obrázek 2.3 ukazuje stránky s příslušnou segmentací provedenou pomocí neuronové sítě. Bílá značí stránku, červená dekorace a růžová komentáře.

**Detekce řádků.** Závěrečnou částí z hlediska získání potřebného obrazu pro klasifikaci je detekce a vyřezání řádků. Výsledkem správné detekce je informace o pozici řádku na stránce. Pozice se zpravidla uvádí pomocí linky, na které leží řádek. Tato linka může být detekována s různou přesností, od jednoduché přímky až po křivku. Dále je nutné určit výšku řádků. Z linky a výšky řádku lze odhadnout velmi přesný polygon, který tento řádek obaluje. Část snímku uvnitř polygonu je použita pro klasifikaci. Klasifikátor vyžaduje tuto formu vstupu, tzn. obrázek pouze jednoho řádku, kde jsou přípustné libovolné horizontální okraje a co nejmenší vertikální. Obrázek 2.4 ukazuje příklady detekce řádků.

**Klasifikace.** Rozpoznávání textu je tedy problém, který se skládá z mnoha dílčích podproblémů. Cílem většiny metod je připravit vstup pro klasifikátor tak, aby byl s velkou pravděpodobností kvalitně klasifikován. Klasifikace obrazu textu znamená jeho převedení na text, se kterým se dá dále pracovat v počítači. Vstupem klasifikátoru je obraz a výstupem posloupnost znaků, podrobněji viz podkapitola 2.2. Tato příprava je klíčová pro správně fungující OCR a musí odpovídat specifikům zachycených snímků. V následujícím textu se věnuji především klasifikaci, a tudíž předpokládám více méně kvalitní výřezy řádků a pří-



Obrázek 2.4: Ukázka detekce řádků (převzato z [10]).



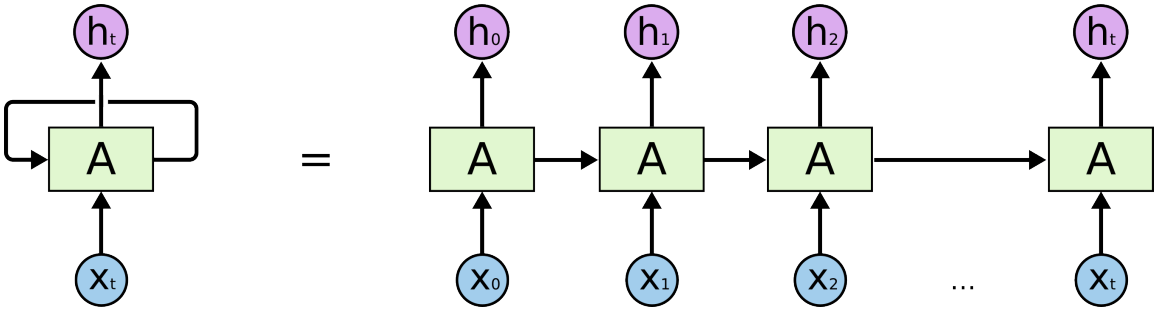

Obrázek 2.5: Výpočetní graf neuronové sítě. Černé šipky značí směr dopředného výpočtu, červené zpětnou propagaci chyby. Jednotlivé uzly představují operátory, které provádějí různé matematické operace nad svými vstupy.

slušné anotace. Z jednotlivých částí přípravy těchto výřezů ovšem plyne řada problému, které se ve výsledných výřezech projevují. Je tedy nutné si uvědomovat původ těchto problémů a vývoj, respektive učení klasifikátoru tomu přizpůsobit, tzn. aktivní učení.

## 2.2 Neuronové sítě

Neuronová síť [17] je matematický model sloužící pro aproximaci funkcí. Tento model je inspirován biologickými neuronovými sítěmi, které se nachází v mozku (odtud pochází název tohoto modelu). Model lze reprezentovat pomocí výpočetního grafu, kde jednotlivé uzly představují operátory. Operátor je funkce, která může mít více vstupů i výstupů. Obecně se jedná o tensory o požadované dimenzionalitě a rozměrech. Většina operátorů má tzv. váhy, které se spolu se vstupem podílejí na výpočtu výstupů. Vhodnou změnou vah lze dosáhnout požadovaného chování sítě, tzn. naučit (natrénovat) síť. Nejpoužívanějším přístupem k učení je učení s učitelem, které se provádí pomocí některé varianty algoritmu zpětného šíření chyby (*Backpropagation*).

Jednoduchý výpočetní graf neuronové sítě je na obrázku 2.5. Výpočet se provádí zleva doprava (naznačeno černými šipkami), kdy vstupem sítě, respektive výpočetního grafu, je zelený blok „Vstup“, což může být obecně libovolný objekt reprezentovatelný v počítači, a výstupem zelený blok „Výstup“. Poté co neuronová síť vypočítá výstup, srovná jej se vzorem. Vzor představuje požadovaný výstup pro daný vstup. Výsledkem porovnání je chyba a na základě jejího gradientu se aktualizují váhy jednotlivých operátorů. Zpětné šíření gradientu chyby je naznačeno červenými šipkami. Každý různě barevný (pojmenovaný) blok značí jinou matematickou operaci, kterou provede nad svým vstupem. Při zpětném šíření gradientu chyby je počítána derivace této operace nad tímto gradientem, a to jak podle vah, aby je bylo možné aktualizovat, tak podle vstupů, aby došlo k propagaci.




Obrázek 2.6: Reprezentace rekuretní vrstvy. Vlevo je nákres principu. Výstup $h_t$ je vypočítán na základě vstupu $x_t$ a předchozího výstupu $h_{t-1}$. Princip praktické implementace pro délku vstupu $t+1$ je vyobrazen vpravo (převzato z [22]).

Aby síť dobře aproximovala příslušnou funkci musí být dostatečně velká, tzn. mít dostatečný počet operátorů a vah vzhledem k řešenému problému. Také je důležitá trénovací sada s velkým množstvím dvojic vstup, vzor. Strukturu samotné sítě nelze exaktně navrhnout tak, aby fungovala pro daný problém optimálně. Návrh je většinou založen na zkušenostech a experimentech s různými architekturami.

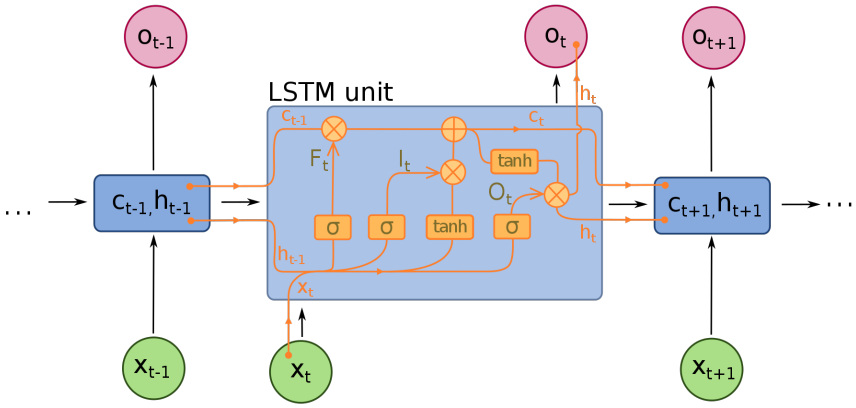
Existuje mnoho problému, které nelze řešit exaktně, tzn. neexistuje obecný a správný postup jak tyto problémy optimálně vyřešit a nebo je řešení příliš náročné. V rámci rozpoznávání obrazu mezi tyto problémy patří např. rozpoznávání obličejů, doostření textu nebo popis scény. Z příchodem konvolučních vrstev (operátorů), které jsou velmi dobré v extrakci příznaků z obrazu, jako jsou hrany, struktury apod., začaly neuronové sítě (konvoluční neuronové sítě) dosahovat v této oblasti nejlepších výsledků [29]. Při rozpoznávání prvku v rámci sekvence se může hodit informace o předcházejících prvcích/následujících prvcích. Mimo konvoluční vrstvy umí tuto informaci využít i vrstvy rekurentní.

**Rekurentní neuronové sítě.** Rekurentní neuronové sítě (RNN) [17] obsahují alespoň jednu rekurentní vrstvu. Tyto vrstvy jsou při zpracovávání svého vstupu schopny využít kontext. Např. pokud se rozpoznává věta „Může se stát cokoliv.“, kde znak „e“ byl špatně vytištěn a vypadá skoro jako znak „c“, síť i přesto rozpozná znak správně, neboť výskyt znaku „c“ za znakem „s“ nebo „ž“ v českém jazyce je téměř nemožný. 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.

Na obrázku 2.6 jsou dvě reprezentace rekurentní vrstvy. Reprezentace vlevo je nákres principu. Výstup $h_t$ je vypočítán na základě vstupu $x_t$ a předchozího výstupu $h_{t-1}$. Index $t$ značí čas, což je v rámci rozpoznávání textu z obrazu pozice v horizontálním směru. Pokud máme tedy výřez řádku textu, na vstup této vrstvy by v čase $t = 0$ vstoupila reprezentace (příznaky, typicky vektor) prvních několika daných pixelů zleva, v čase $t = 1$ stejné reprezentace stejného množství následujících pixelů atd. V určitém čase (délka vstupu) je cyklus zastaven. Princip praktické implementace pro délku vstupu $t+1$ je vyobrazen vpravo. Blok pro vstup $t + 1$ musí čekat na dokončení výpočtu bloku pro vstup $t$. Výpočet tedy nelze z pohledu času paralelizovat, což vede k značnému nevyužití zdrojů a čekání. Formálně je tedy výstup v čase $t$ dán vztahy:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{2.1}$$

$$y_t = W_{hy}h_t + b_y, \tag{2.2}$$



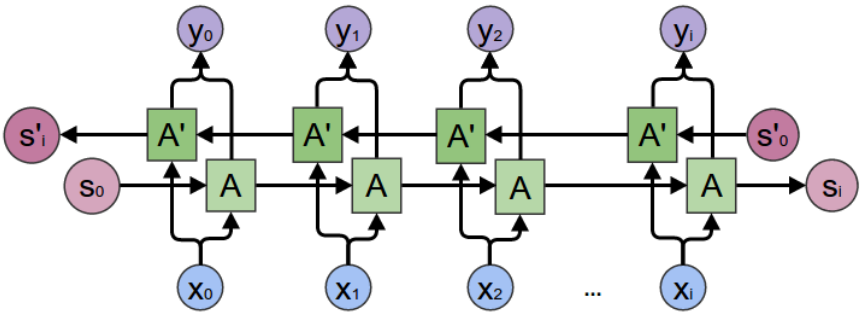

Obrázek 2.7: Schéma LSTM bloku. Znaménko × a + v kroužku značí násobení a sčítání po prvcích. Vstupy, výstupy bloků $x_t$, $o_t$. Předávané výstupy a zapamatované informace $h_t$ a $c_t$. Blok je rozdělen na několik částí tzv. brány a to zapomínající, vstupní a výstupní značené $F_t$, $I_t$, $O_t$. Tyto brány využívají příslušné váhové matice a aktivační funkce (převzato z [1]).

kde $W$ jsou váhové matice vstupu, vnitřního stavu a výstupu, $b$ jsou biasy a $f$ je aktivační funkce jako tangens, sigmoida apod.

Nevýhodou těchto vrstev je problematické učení pomocí algoritmu zpětného šíření chyby [3]. Může docházet jak k vyhasínání (*vanishing*) tak velkému nárůstu (*exploding*) gradientů, což zpravidla vede k znehodnocení průběhu učení.[2] Důvodem těchto problémů je, že při výpočtu gradientů pro jednotlivé bloky vrstvy se postupuje podle řetězového pravidla (*chain rule*), dochází k mnohonásobnému násobení gradientů předchozích bloků. Zjednodušeně si lze představit, že gradient každého bloku vrstvy „A“ na obrázku 2.6 se spočítá na základě bloků ležících za ním a to jako produkt všech jejich gradientů. Pokud jsou všechny tyto gradienty velmi blízké nule dochází k vyhasínání, a to tím více, čím je $t$ bloku bližší 0. Důsledkem je zejména zmenšení kontextu, který je síť schopna využívat pro klasifikaci. Se stoupajícím počtem bloků se problém zhoršuje.

Negativní dopady vyhasínání lze výrazně omezit pomocí tzv. LSTM (*Long Short-Term Memory*) bloků respektive vrstev [13]. Blok LSTM obsahuje aktivační funkce a využívá několik váhových matic. Příklad takového bloku je na obrázku 2.7. Znaménko × a + v kroužku značí násobení a sčítání po prvcích. Jako aktivační funkce jsou použity tangens a sigmoidy. Výstup bloku $o_t$ se liší od předávaného výstupu $h_t$, pro klasickou rekurentní vrstvu jsou tyto výstupy totožné. Novým vstupem, respektive výstupem, je zapamatovaná informace značená symbolem $c_t$. Blok je rozdělen na několik částí, tzv. brány, a to zapomínající, vstupní a výstupní, značené $F_t$, $I_t$, $O_t$. Implementace těchto bran, stejně jako propojení v rámci bloku, se mohou lišit. Typicky brány využívají již zmíněné váhové matice. Brány jednoho typu ve všech blocích využívají stejnou matici. Princip zapomínající brány spočívá v rozhodování o tom, která informace se zapomene z paměti předcházejícího bloku. Vstupní brána rozhoduje o uchování/ignorování vstupů, výstupní brána se stejným způsobem stará o výstup. Rozhodování se provádí na základě příslušných matic, které se učí stejně jako ostatní vrstvy sítě. Dále se také může použít matice při rozhodování o tom, co se má zapamatovat.

[2] Většina vah sítí se nachází v rozmezí ±1. Pokud je gradient malý, váha se téměř nezmění a síť se neučí. Pokud je gradient naopak extrémně velký, váha se změní na extrémně velkou a znehodnotí podíl ostatních „správných“ vah na výpočtu.




Obrázek 2.8: Schéma obousměrné rekurentní vrstvy. Jedná se v podstatě o dvě rekurentní vrstvy, kdy jedna se počítá zleva doprava a druhá ve směru opačném. Výsledek $y_t$ je konkatenací výsledků jednotlivých bloků pro daný vstup $x_t$ (převzato z [21]).

Výpočet zapomínající brány je vyjádřen vztahem:

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

kde $W_f$ značí zapomínající váhovou matici, $b_f$ zapomínající bias, tzn. hodnoty, které jsou přičteny po prvcích, $\sigma$ aktivační funkci sigmoida a hranaté závorky konkatenaci. Ostatní brány se počítají obdobně.

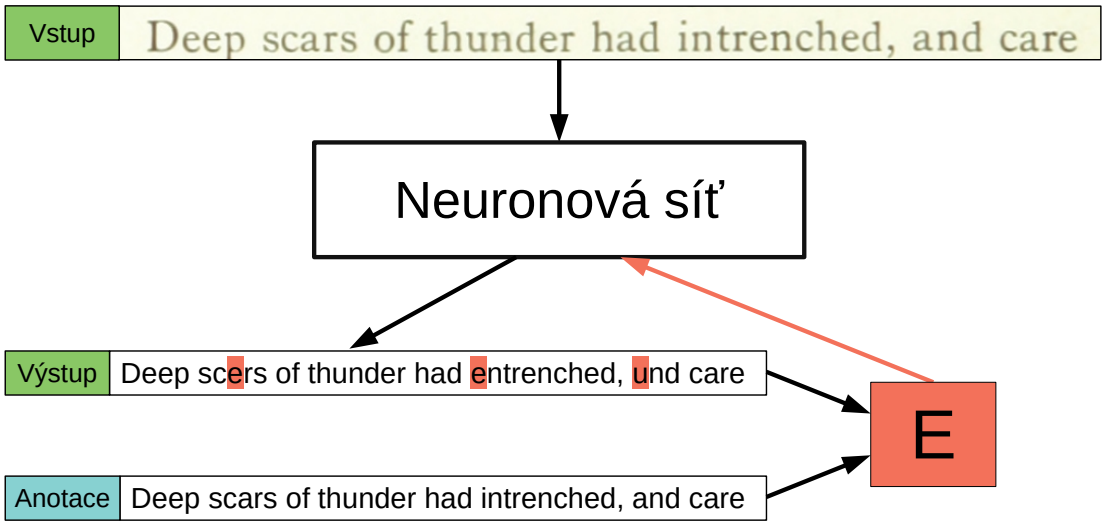
Implementace jednotlivých bran může být složitější, např. může záviset na zapamatované informaci $c_t$, nebo může používat více váhových matic [11]. Principiálně se ovšem jedná o podobný přístup. Pomocí těchto mechanizmů může síť pro klasifikaci využívat větší kontext. Na druhou stranu výpočet LSTM vrstvy je značně výpočetně náročnější než výpočet klasické rekurentní vrstvy. Klasická vrstva ovšem nemá dostatečně dobré vlastnosti, a proto je vrstva LSTM upřednostňována.

Doposud popsané vrstvy umožňují využití kontextu pouze v jednom směru. Pro využití obou směrů lze použít tzv. obousměrnou rekurentní vrstvu. Schéma této vrstvy je na obrázku 2.8. Jedná se v podstatě o dvě rekurentní vrstvy, kdy jedna se počítá zleva doprava a druhá ve směru opačném. Výsledek $y_t$ je konkatenací výsledků jednotlivých bloků pro daný vstup $x_t$. Bloky mohou být jak klasické tak LSTM.

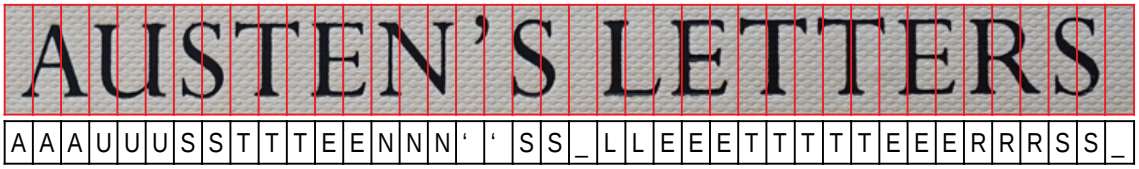
Rekurentní vrstvy mohou být použity samostatně, ale také v kombinaci s vrstvami jinými. Vzhledem ke zpracování obrazu jsou výhodné především již zmíněné konvoluční vrstvy. Síť pro rozpoznávání textu je pak tvořena sadou konvolučních vrstev spolu s vrstvami typu *pooling*, které umožňují redukci dimenzionality a vrstvami rekurentními (v tomto pořadí). Princip spočívá v dobré extrakci příznaků a následném využití kontextu.

**Chybová funkce pro rozpoznávání textu.** Aby bylo možné učit neuronové sítě, je nutné nějakým způsobem počítat chybu. Počítání chyby zajišťuje tzv. chybová funkce (*loss function*), která srovnává výstup sítě se vzorem. Anotace, respektive posloupnost znaků vyskytujících se v obraze, je vzorem pro rozpoznávání textu. Princip učení je zobrazen na obrázku 2.9. Síť na základě vstupního výřezu řádku vygeneruje jeho přepis a ten je pomocí chybové funkce $E$ srovnán s anotací. Výsledkem srovnání je chyba, které se následně propaguje sítí a dochází k učení. Špatně klasifikované pozice, které určí chybová funkce, jsou podbarveny červeně. Chybová funkce je standardně považována za součást neuronové sítě.

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




Obrázek 2.9: Princip učení neuronové sítě pro rozpoznávání textu. Síť na základě vstupního výřezu řádku vygeneruje jeho přepis, ten je pomocí chybové funkce $E$ srovnán s anotací. Výsledkem srovnání je chyba, které se následně propaguje sítí a dochází k učení. Špatně klasifikované pozice, které určí chybová funkce jsou podbarveny červeně.




Obrázek 2.10: Ukázka naivní anotace řádku textu, podtržítko značí mezerník.

obsahuje) různý počet znaků. To je způsobeno různou šířkou znaků, různým fontem a také nepřesnými výřezy, tzn. znaky stejného fontu mohou být různě veliké vlivem rozdílného odhadu výšky řádku, apod. Naivní řešení tohoto problému je horizontálně rozdělit řádek na konečný počet částí a tyto části anotovat podle výskytu jednotlivých znaků. Síť poté negeneruje přímo přepis, ale znaky v těchto částech. Přepis by se z těchto částí získal sloučením stejných znaků, které se nacházejí na sousedních pozicích. Ukázka možného výstupu sítě, respektive anotace pro daný výřez, je na obrázku 2.10, podtržítko značí mezerník. Anotovat řádek tímto způsobem je náročné z hlediska lidské práce a pro některé pozice navíc nelze určit správný znak. Rovněž by docházelo k problémům pokud by se ve výřezu vyskytovaly dva totožné znaky na sousedních pozicích. Příkladem je znak „T“ u kterého nelze z výsledné posloupnosti znaků určit, že se daný znak vyskytoval v původním výřezu dvakrát a byl by sloučen do znaku jednoho.

Je tedy výhodnější nechat samotnou síť, aby se tuto anotaci naučila na základě standardního přepisu. Toto chování umožňuje tzv. CTC (*Connectionist temporal classification*) vrstva [12]. Jejím vstupem je matice $T \times C$, kde $T$ značí počet částí na který je rozdělen vstupní výřez a $C$ počet znaků v abecedě rozšířené o prázdný znak (*blank*). Abeceda je

množina znaků, kterou dokáže neuronová síť rozpoznat. Na vstupní matici je aplikována funkce *softmax*, která jednotlivé hodnoty převede do rozsahu 0–1 a to tak, že suma hodnot každé části výřezu je 1. Každá část je tedy reprezentována vektorem hodnot, kde jednotlivé hodnoty přísluší jednotlivým znakům. Hodnoty představují pravděpodobnost s jakou se daný znak vyskytuje ve výřezu. Jedna znamená naprostou jistotu, nula naopak nejistotu.

Z výstupní pravděpodobnostní matice lze vypočítat pravděpodobnost všech možných průchodů pro daný výřez. Průchod je posloupnost znaků o délce $T$. Pravděpodobnost průchodu je dána vztahem

$$p(\pi|x) = \prod_{t=0}^{T-1} P_{t,i(\pi_t)}, \tag{2.4}$$

kde $P$ je výstupní matice pravděpodobností, $\pi$ průchod, $x$ vstupní výřez a $i(\pi_t)$ udává index znaku průchodu pro pozici $t$. Jedná se tedy o součin pravděpodobností všech znaků průchodu.

Z hlediska klasifikace je žádoucí zjistit nejpravděpodobnější přepis. Nejjednodušší a zároveň rychlý způsob jak najít přijatelně dobrý přepis, je zvolit nejpravděpodobnější průchod a ten upravit na přepis. Úprava se opět provádí slučováním sousedních znaků. Znaky představující anotaci více sousedních znaků ve výřezu nejsou sloučeny, protože se mezi nimi nachází prázdné znaky. Tyto znaky slouží pouze jako oddělovače a ve výsledném přepisu se nevyskytují. Formálně je tento postup vyjádřen vztahem:

$$y = G(\pi)|max(p(\pi|x)), \pi \in L, \tag{2.5}$$

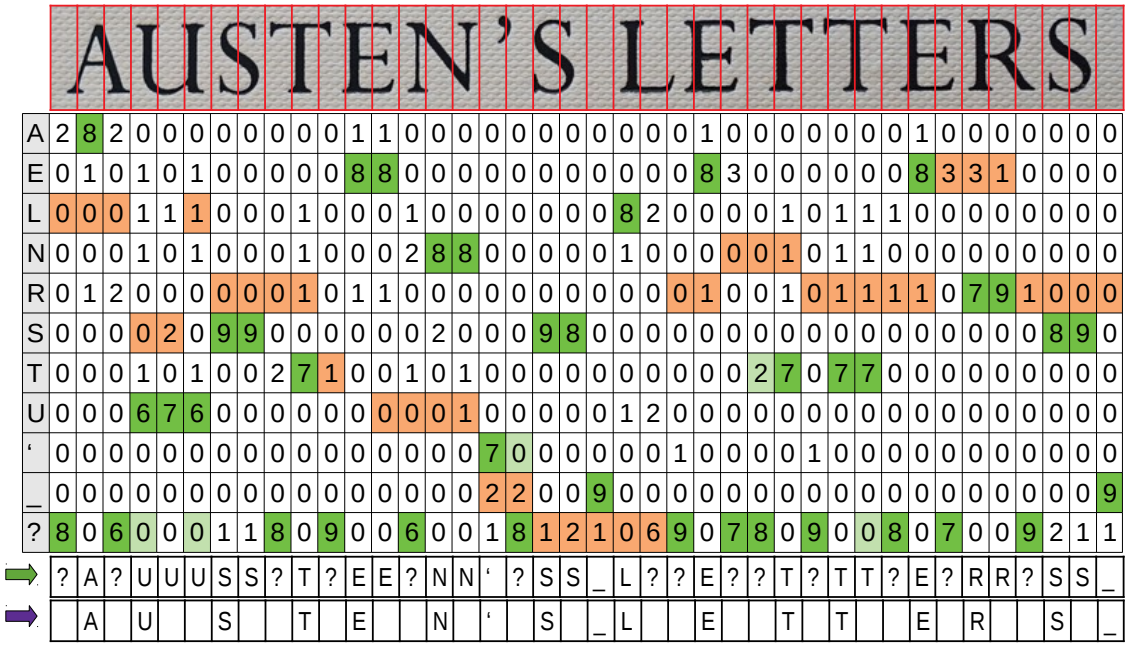
$y$ značí přepis, $G$ funkci provádějící slučování znaku a $L$ množinu všech průchodů. Takto získaný přepis ovšem nemusí být optimální. Metody poskytující lepší řešení jsou časově mnohem náročnější.

Výše popsaný princip dopředného průchodu CTC vrstvy je pro ujasnění znázorněn na obrázku 2.11. 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 vyskytující se ve vstupu a pravděpodobnost je vynásobena deseti. Sytě zelená barva značí nejpravděpodobnější průchod, oranžová průchod jiný, méně pravděpodobný. V některých sloupcích je světle zelenou barvou naznačena možnost záměny ze sytě zelenou. Po provedení libovolných záměn získáme jiný průchod, který bude méně pravděpodobný, ale jehož přepis bude stále správný. Řádek označený zelenou šipkou představuje znaky nejpravděpodobnějšího průchodu, tzn. $\pi$ pro které platí $max(p(\pi|x))$. Pro získání výsledného přepisu je následně použita funkce $G$, výsledek je na posledním řádku (fialová šipka).

Pro učení sítě je nutné určit jakým způsobem se počítá chyba pro daný vstup, výstup a vzor (anotaci). Chyba se spočítá podle vztahu:

$$E = -ln(\sum_{\pi \in B} p(\pi|x)) \tag{2.6}$$

jako negace logaritmu sumy všech pravděpodobností průchodů pro danou anotaci, kde $B$ je množina všech možných průchodů pro danou anotaci. Pokud se síť naučí jediný správný průchod, bude pravděpodobnost tohoto průchodu 1 a pravděpodobnost ostatních možností 0, tzn. nulovou chybu. Síť se učí zpětnou propagací gradientu této chyby. Problém spočívá ve výpočtu pravděpodobností všech možných průchodu, jelikož těchto průchodů bývá zpravidla velké množství. Jako řešení se využívá dynamického programování, čímž se celý výpočet značně zkomplikuje.




Obrázek 2.11: 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).

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.

## 2.3 Shrnutí

Cílem této kapitoly bylo objasnit základy rozpoznávání textu a neuronových sítí s ohledem na aktivní učení. Vynechal jsem tedy velké množství detailních popisů jednotlivých problémů, jelikož z pohledu této práce je důležitá zejména jejich podstata a princip jakým se řeší. Pochopení těchto problémů a jejich řešení má ovšem pro správný přístup k aktivnímu učení zásadní význam. Např. pokud klasifikátor nefunguje správně pro určitou skupinu řádků, je užitečné znát přesný důvod a jeho potenciální řešení. Ideálně se tak daná problematická množina řádků a příslušná data předá týmu, který tento problém může přímo vyřešit.

V první podkapitole jsem se zabýval samotným problémem rozpoznávání textu, který se dělí na několik částí, z nichž většina má na starosti předzpracování vstupu pro klasifikátor. Jedná se o získání obrazu, předzpracování, segmentaci, výřezy řádků a klasifikaci. Výhodou tohoto členění je schopnost řešit každý podproblém nezávisle a celkově tak dosáhnout větší přesnosti ve výsledné klasifikaci. Neuronovým sítím, které řeší samotnou klasifikaci, jsem se věnoval v druhé podkapitole. Konvoluční vrstvy extrahují kvalitní příznaky, které

jsou důležité pro správnou funkci vrstev následujících. Rekurentní vrstvy umožňují využití kontextu při klasifikaci, zvláště pak upravené vrstvy typu LSTM. Nejdůležitější vrstvou z hlediska aktivního učení je CTC, jejíž výstupem je pravděpodobnostní matice udávající pravděpodobnost všech možných průchodů, respektive přepisů pro daný vstup. Toho lze do značné míry využít a získat tak poměrně přesnou představu o tom, jaký má síť „názor“ na daný řádek textu.

# Kapitola 3

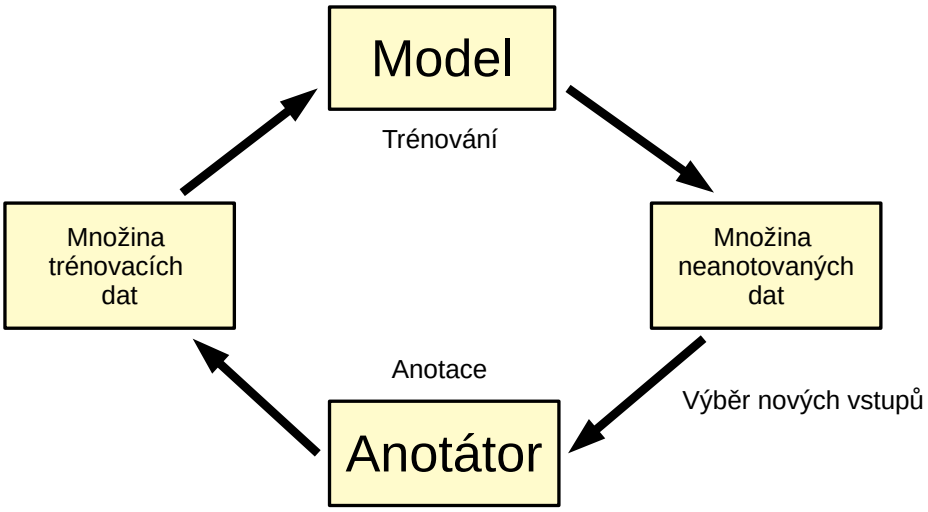
# Aktivní učení

Strojové učení je oblast umělé inteligence zabývající se matematickými modely, které mohou na základě učení měnit svůj vnitřní stav tak, aby se jejich chování jevilo jako inteligentní. Jedním se základních přístupů k učení je učení s učitelem (*supervised learning*), který zpravidla vyžaduje velké množství trénovacích dat. Tyto data obsahují dvojice vstup a požadovaný výstup (vzor). Např. pro učení kvalitního modelu rozpoznávajícího zvířata je zpravidla potřeba deseti tisíce obrázků, kde pro každý obrázek musí být k dispozici informace o jaké zvíře se jedná. Při vývoji těchto modelů je tudíž získání dostatečného množství trénovacích dat často největší problém. Je potřeba spousta lidského úsilí, tzn. finanční náklady a čas. Získání některých dat může být velmi náročné nebo nemožné a tím je nemožné i zlepšení modelu na základě přidávání nových dat. Často je k dispozici velké množství vstupů, ale chybí požadované vzory, tzn. informace o tom, co daný vstup představuje. Vzniká problém jak vybrat nové vstupy pro které se vytvoří vzory (na které fotografie zvířat se má člověk podívat a určit o jaké zvíře se jedná).

Aktivní učení se zabývá tím, jak maximálně využít lidské úsilí při učení modelů strojového učení [31, 15, 30]. Zejména se jedná o metody umožňující výběr nových vstupů a vyhodnocení stávajících trénovacích dat. Tyto metody zpravidla využívají model, který už byl naučen na nějaké množině dat. Tato množina je zpravidla vybrána intuitivně na základě zkušeností s daným modelem. Nejjednodušší je náhodný výběr nebo převzetí již připravených dat. Cílem výběru nových vstupů pomocí modelu je dosažení stejné kvality s menším úsilím než při výběru intuitivním, respektive získání kvalitnějšího modelu při stejném úsilí.

Výběr takových vstupů je náročný z hlediska vhodnosti jak pro model, tak pro lidi. Současné znalosti pokročilejších modelů strojového učení jsou do značné míry omezené a není tedy obecně známo, které vstupy jsou optimální pro zvýšení jejich kvality. Nelze ani předpokládat, že nalezení takovýchto vstupů nutně znamená vytvoření optimálního vzoru. Vzor zpravidla vytváří člověk a může tedy docházet k omylům. Pro výběr vhodného vstupu je tedy nutné zvážit mnoho různých hledisek. Vyhodnocení stávajících dat je důležité z hlediska možného výskytu chyb, které vznikly zpravidla špatným vytvořením vzoru. Rovněž výskyt příliš nejasných a matoucích vstupů může být nežádoucí. Konečně je také důležité zvážit čas nutný k vytváření jednotlivých vzorů. Lze očekávat, že komplikovanější vstupy budou mít větší přínos pro učení modelu. Zároveň je však pravděpodobné, že vytváření příslušných vzorů bude časově náročnější.

Tato kapitola slouží především k představení základních principů, které se při aktivním učení využívají. Vycházím zejména z disertační práce *Curious Machines: Active Learning with Structured Instances* [31] jejímž autorem je Burr Settles. Tato práce poskytuje rozsáhlý




Obrázek 3.1: Cyklus aktivního učení. Model předloží neanotovaný vstup k anotaci. Anotátor provede anotaci a vloží novou dvojici do trénovací množiny. Na této množině je model znovu trénován a cyklus se opakuje.

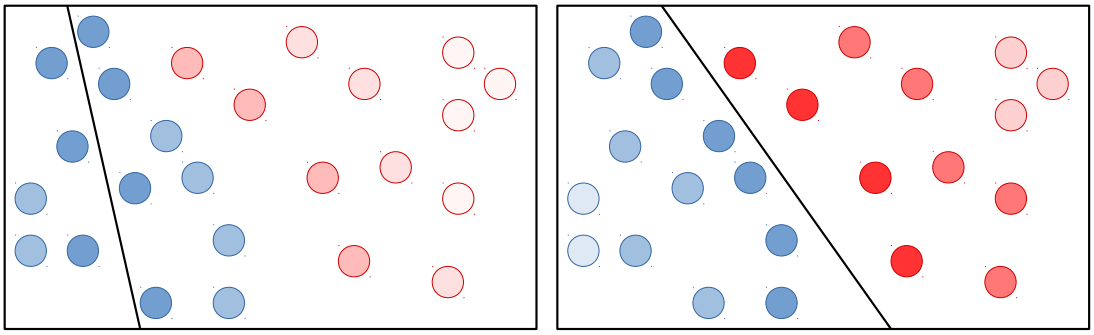
přehled výzkumu v rámci aktivního učení. Současný výzkum v tomto odvětví není příliš velký, jedná se spíše o výjimečné případy, které zmiňuji v jednotlivých podkapitolách.

**Aktivní učení založené na výběru neanotovaných vstupů.** Proces vytvoření vzoru k danému vstupu se nazývá anotování a tudíž anotace je synonymum pro vzor. Model vybírá vstup z množiny neanotovaných vstupů a předkládá jej anotátorovi. Ten tento vstup anotuje a přidá novou dvojici do trénovacích dat. Model je následně znovu učen s těmito novými vstupy. Poté se opět přistoupí k výběru neanotovaného vstupu atd. Tento základní přístup utváří cyklus znázorněný na Obrázku 3.1. Jelikož analýza trénovacích dat je do značné míry závislá na konkrétním problému, věnuji se v této kapitole zejména výběru vhodných vstupů (*select queries*).

## 3.1 Metody výběru vhodných dat

Výběr vhodných dat pro anotaci závisí jak na modelu, tak na konkrétním řešeném problému. Následující metody se zabývají především využitím modelu a mají tedy rozsáhlé uplatnění. Cílem je uvést princip jednotlivých metod, jejich výhody a nevýhody. Na základě těchto znalostí lze implementovat pokročilejší metody využívající znalosti řešeného problému. Funkce $\Phi_m(x)$ vrací vhodnost vstupu k anotaci podle metody $m$, čím větší je její výstup, tím je vstup vhodnější. Tato funkce vychází z vyhodnocení modelu nad daným vstupem, případně nad množinou možných vzorů. Při vyhodnocení se sleduje chování modelu, zejména pak jeho výstup a potenciální změny vyvolané přidáním vstupu do trénovací sady.

**Nejistota.** Základní metoda Aktivního učení je nejistota (*uncertainty sampling*) [19]. Tuto metodu lze využít pro učení modelů, které jsou schopny pro daný vstup určit jistotu svého rozhodnutí, tzn. jistotu výstupu. Neanotované vstupy tedy lze ohodnotit jednoduše



Obrázek 3.2: Prostor vstupních dat a dělící hranice binárního klasifikátoru. Modrá a červená barva odlišuje jednotlivé třídy, průhlednost barev značí nejistotu, respektive informativnost jednotlivých vstupů podle modelu. Čím je barva sytější, tím je vstup nejistější. Vlevo je situace po inicializaci modelu náhodnými vahami. Stav po dotrénování klasifikátoru je znázorněn vpravo.

pomocí výstupu modelu. Výstup může být obecně tensor, a tudíž toto ohodnocení vytváří n-rozměrný prostor vstupů. Tento prostor reprezentuje vztah modelu k neanotovaným datům. Nejjednodušším přístupem jak vybrat vhodný vstup je vybrat ten nejméně jistý.

Jako jednoduchý příklad použití tohoto přístupu uvádím binární klasifikátor 3.2 (logistický model, perceptron) s aktivační funkcí sigmoida 3.1. Logistický model je schopný lineárně rozdělit prostor pomocí přímky, respektive n-rozměrné plochy. Příslušnost k jedné ze dvou tříd je dána výstupem, který je díky funkci sigmoida v rozmezí 0–1. Pokud je $B(x) >= 0.5$ jedná se o jednu třídu, jinak o třídu druhou. Model je tím nejistější, čím je hodnota bližší 0.5. Hodnotící funkce $\Phi_U(x)$ je dána vztahem 3.3 kde funkce $G$ je Gaussova funkce se středem v 0.5.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

$$B(x) = \sigma(w^T x) \tag{3.2}$$

$$\Phi_U(x) = G(B(x)) \tag{3.3}$$

Obrázek 3.2 graficky znázorňuje prostor vstupních dat a dělící hranici binárního klasifikátoru tzn. $B(x) = 0.5$. Modrá a červená barva odlišuje jednotlivé třídy, průhlednost barev značí nejistotu, respektive informativnost jednotlivých vstupů podle $\Phi_U(x)$ modelu. Čím je barva sytější, tím je vstup nejistější. Vlevo je situace po inicializaci modelu náhodnými vahami. Na začátku není anotován žádný vstup (barvy zde představují budoucí anotace). Podle sytosti barev je patrné, že klasifikátor si je nejméně jistý vstupy v blízkosti rozhodovací hranice. Podle aktivního učení vybereme k anotaci právě jeden z těchto vstupů. Smysluplným výběrem pro trénování jsou modré vstupy napravo od rozhodovací hranice, s aktivním učením budou vybrány s mnohem větší pravděpodobností, než při náhodném výběru (kdy může dojít i k výběru libovolného červeného vstupu). Aktivní učení v tomto případě zajišťuje jistou formu kontroly rozhodovací hranice. Stav po dotrénování klasifikátoru je znázorněn vpravo.

Uvedený příklad ukazuje potenciál aktivního učení za pomoci jednoduchého binárního klasifikátoru. Lze předpokládat, že pro složitější modely bude tento přístup fungovat obdobně. Pro reálné modely a data ovšem typicky platí, že rozhodovací hranice mezi třídami nejsou pevně dány. Stále je však velmi pravděpodobné, že vstupy v blízkosti těchto hranic

jsou pro klasifikátor velmi přínosné, jelikož umožňují jejich upřesnění a tím pádem zlepšení kvality modelu.

Pro klasifikátory o více než dvou třídách lze použít entropii 3.4 [31, 25], kde $y_i$ je konkrétní třída a $P(y_i|x)$ pravděpodobnost, že daný vstup $x$ patří do této třídy. Pro správně klasifikované vstupy se výstup blíží 0. Často používanou metrikou je také rozdíl pravděpodobností pro dvě nejpravděpodobnější třídy 3.5 [31, 34, 25], kde $y_n^*$ značí n-tou nejpravděpodobnější třídu pro vstup $x$. Pro neuronové sítě a rozpoznávání textu lze využít výstupní pravděpodobnostní matice a algoritmu paprskového prohledávání (*beam search*). Tento algoritmus je schopný odhadnout požadované množství nejpravděpodobnějších průchodů touto maticí, $y_n^*$ pak značí n-tý nejpravděpodobnější průchod.

$$\Phi_{UE}(x) = -\sum_i P(y_i|x) log P(y_i|x) \tag{3.4}$$

$$\Phi_{UB}(x) = -abs(P(y_1^*|x) - P(y_2^*|x)) \tag{3.5}$$

**Neshoda.** Pomocí více různých modelů, které jsou natrénovány na stejných/podobných datech lze vybrat vstup na základě neshody mezi jednotlivými modely. To znamená výběr toho vstupu, pro který se jednotlivé výstupy modelů liší nejvíce. Samotné měření neshody záleží na konkrétní řešené úloze. Pro pravděpodobnostní modely lze použít hlasování pomocí entropie podle vztahu [8]:

$$\Phi_D(x) = -\sum_c^C \frac{V(x,c)}{k} log \frac{V(x,c)}{k}, \tag{3.6}$$

kde $k$ je počet modelů, $C$ množina tříd a $V(c,x)$ udává počet hlasů všech modelů pro daný vstup $x$ a třídu $c$. Funkce $\Phi_D$ nabývá hodnoty 0 pro shodu všech modelů.

Článek [27] uvádí použití Levenštejnovi vzdálenosti pro výstupy obsahující textové řetězce. Tato vzdálenost udává minimální počet záměn, které je nutné provést v rámci jednoho textového řetězce, aby byl totožný s řetězcem druhým. Výstupy modelů jsou porovnány principem každý s každým. Vzdálenosti v rámci jednoho řádku jsou sečteny a normalizovány jeho délkou. Řádek s největší celkovou vzdáleností je vybrán k anotaci.

V rámci neuronových sítí je získání několika natrénovaných modelů zpravidla velmi časově náročné. Alternativu k trénování několika neuronových sítí nabízí modifikace trénovaných neuronových sítí [9]. Nejprve je trénována jedna síť. Pokud jsou požadovány další modely pro výběr vhodných vstupů je stávající síť několikrát modifikována, tzn. různé váhy v rámci sítě se nastaví na 0. Takto modifikované sítě se musí následně trénovat po určitou dobu, aby se váhy přizpůsobily zavedeným změnám. Při trénování je třeba zamezit natrénování sítě do původní podoby, proto se trénuje jenom poslední vrstva. Tímto způsobem lze získat odlišné modely za výrazně nižší dobu než při standardním trénování od začátku.

Při výběru pomocí neshody je důležité, aby jednotlivé modely byly rozdílné a nechovali se stejně pro všechny vstupy. Toho lze dosáhnout jak výběrem různých modelů, tak trénováním stejných modelů na různých, ale podobných datech (případně kombinací obojího). Čím jsou rozdílné modely kvalitnější, tím je pravděpodobnost omylu při výběr vhodného vzoru menší.

**Odhad změny modelu.** Přidání nového vstupu do trénovací množiny a následné trénování modelu způsobí určitou změnu modelu. Velikost této změny nemusí nutně záviset na míře nejistoty daného vstupu. Výběr vstupů, které nijak neovlivní stávající model není

žádoucí, protože model se na těchto vstupech není schopen naučit nic nového (přinejmenším v daném stavu). Práce *Multiple-Instance Active Learning* a *Curious machines: Active learning with structured instances* [32, 31] předpokládají pravděpodobnostní modely učené na základě metody gradientního sestupu (*gradient descent*). Je tedy žádoucí vybrat vstupy, které mají největší gradient. Jelikož je anotace neznámá, nelze vstupy jednoduše ohodnotit. Řešením je odhadnout gradient pomocí modelu a všech možných anotací. Formálně vyjádřeno vztahem:

$$\Phi_G(x) = \sum_i P(y_i|x;\Theta)\|\nabla l(< x, y_i >;\Theta)\|_e, \tag{3.7}$$

$\nabla l$ značí gradient chyby pro vstup $x$, danou anotaci $y_i$ a model s vahami $\Theta$, euklidovská norma $e$ tohoto gradientu je váhována pomocí modelu tzn. pravděpodobnosti, že model bude mít na výstupu právě danou anotaci. Předpokládá se, že model je dostatečně natrénován na aktuálních vstupech, tzn. gradienty těchto vstupů lze zanedbat a uvažovat pouze gradient nově přidaného vstupu.

Pro neuronové sítě a rozpoznávání textu je nemožné postupovat výše uvedeným způsobem, jelikož množina možných anotací je potenciálně nekonečná. Jedním z řešení tohoto problému je určit gradient chyby pro $K$ nejpravděpodobnějších průchodů pravděpodobnostní maticí pro daný vstup [15]. Pro nalezení těchto průchodů se používá již zmíněný algoritmus paprskového prohledávání (*beam search*).

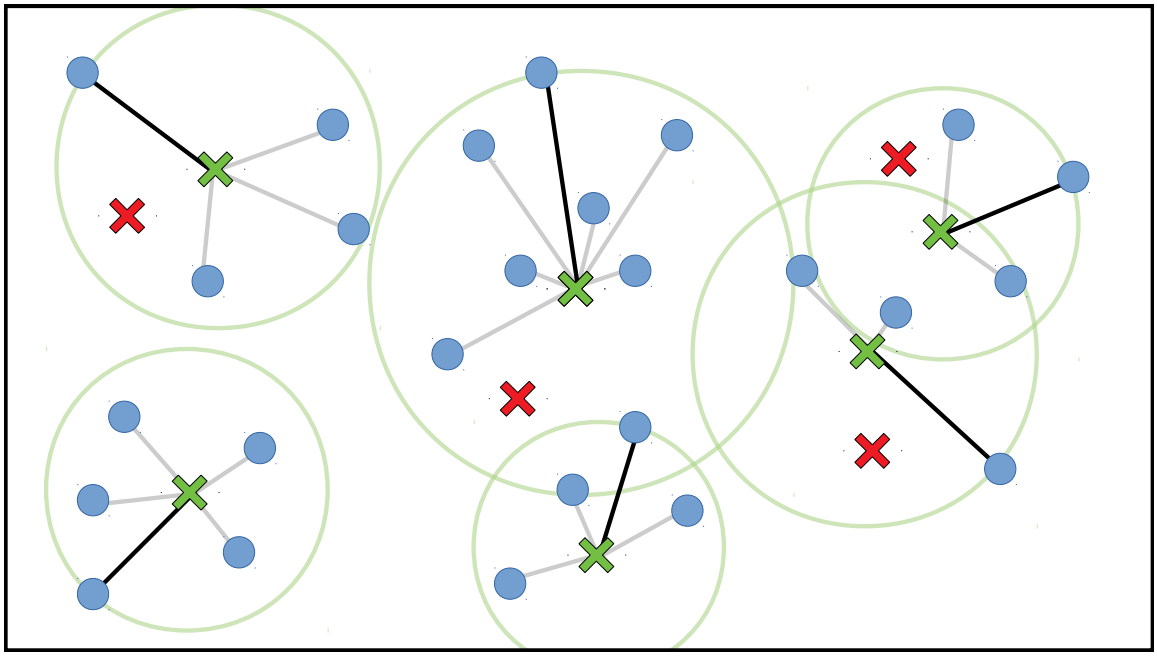
**Odhad snížení chyby.** Změna modelu na základě nového vstupu nemusí nutně znamenat změnu k lepšímu, tzn. nejistý vstup nemusí být nutně vhodným kandidátem pro zlepšení modelu. Možností je vybrat takový vstup, který maximálně sníží chybu současného modelu. Ačkoliv se tento přístup jeví jako optimální, pro velké množství modelů je z hlediska náročnosti výpočtu prakticky nepoužitelný. Např. pro neuronové sítě by bylo třeba trénovat síť pro každý neanotovaný vstup zvlášť jen pro to, aby se vybral jeden vhodný. Model tedy musí umožňovat rychlé přeučení při přidání nového vstupu, např. naivní Bayes [28] nebo SVM [20].

**Prohledávání prostoru vstupů.** Cílem Aktivního učení je natrénovat co nejkvalitnější model. Předchozí metody jsou založeny především na výběru vstupů z hlediska největšího aktuálního významu pro model. Tímto výběrem dochází k upřednostnění vstupů, které jsou nejméně jisté, nejvíce ovlivní model nebo vedou k největšímu snížení chyby. Model se tak snaží naučit na potenciálně problematických vstupech. Takovéto vstupy ovšem nemusí být dostatečně reprezentativní pro problém, který se model snaží řešit. Dalším přístupem pro výběr vzorů je tedy vybrat takové, které kvalitně reprezentují řešený problém.

Příkladem pro neuronové sítě je metoda výběru pomocí aproximačního řešení problému *k-center* [30]. Cílem je anotovat takové vstupy, aby výsledný trénovací dataset byl dostatečně reprezentativní vůči řešenému problému, tzn. model by se na tomto datasetu měl být schopný naučit stejně kvalitně jako na datasetu se všemi anotovanými vstupy. Formálně je žádoucí výběr $s^1$ vyjádřen vztahem:

$$\min_{s^1:|s^1|\leq b} \left| \frac{1}{|N|} \sum_{i\in N} l(x_i, y_i; A_{s^0\cup s^1}) - \frac{1}{|s^0 + s^1|} \sum_{j\in s^0\cup s^1} l(x_j, y_j; A_{s^0\cup s^1}) \right|, \tag{3.8}$$

kde $b$ je maximální počet vstupů které lze vybrat, $N$ množina všech anotovaných vstupů, $s^0$ aktuálně anotované vstupy, $A_D$ váhy natrénovaného modelu na datasetu $D$ a $l$ je chyba



Obrázek 3.3: Příklad možného pokrytí prostoru problému *k-Center*. Kolečka značí anotované vstupy a křížky neanotované. Přiřazení anotovaných vstupů k jednotlivým vybraným centrům (zelené křížky) je naznačeno úsečkou, přitom černá úsečka značí vstup o maximální vzdálenosti od příslušného centra. Zelený kruh značí pokrytí daného centra. Jedná se pouze o ilustraci problému tzn. toto pokrytí nemusí být optimální.

modelu o daných vahách a vstupech. Preferuje se takový výběr, který minimalizuje rozdíl v chybě modelu nad částečnou a celkovou trénovací sadou.

Jelikož anotace pro data $N - s^0 \cup s^1$ nejsou známy, provádí se odhad takového výběru pomocí aktuálního modelu. Všechny vstupy jsou reprezentovány vektorem o velikosti $n$, tento vektor se získá z výstupu poslední vrstvy neuronové sítě (případně jiné vrstvy). Každý vstup má tak určitou pozici v n-dimenzionálním prostoru vstupů. Vektory neanotovaných vstupů (index $i$) tvoří potenciální centra pro vektory anotovaných vstupů (index $j$), které jsou jim nejblíže (na základě nějaké metriky např. euklidovské). Odhad lze získat na základě vyřešení problému *k-Center*:

$$\min_{s^1:|s^1|\leq b} \max_{i\in s^1} \min_{j\in s^1\cup s^0} \Delta(x_i, x_j), \tag{3.9}$$

nejvíce zanořené minimum značí přiřazení anotovaného vstupu nejbližšímu neanotovanému, maximum značí vzdálenost nejvíce vzdáleného anotovaného vstupu v rámci centra tvořeného neanotovaným vstupem, konečně poslední minimum vyžaduje, aby tyto maximální vzdálenosti byly v součtu co nejmenší. Tento princip je znázorněn na Obrázku 3.3. Kolečka značí anotované vstupy a křížky neanotované. Přiřazení anotovaných vstupů k jednotlivým centrům je naznačeno úsečkou, přitom černá úsečka značí vstup o maximální vzdálenosti od příslušného centra. Zelený kruh značí pokrytí daného centra. Cílem je pokrýt celý prostor pomocí center o počtu $b$ a to tak, aby poloměry kruhů byly co nejmenší. Neanotované vstupy, které jsou reprezentovány těmito centry jsou následně vybrány k anotaci. V případě obrázku je $b$ rovno 6 a zelené křížky značí vybraná centra. Jedná se však pouze o ilustraci problému a toto pokrytí proto nemusí být optimální.

Tento problém je NP-těžký, prakticky se proto řeší pomocí aproximačních algoritmů. Jedním z těchto algoritmů je hladový přístup (*greedy*). Iterativně je vybráno $b$ vstupů pro anotaci na základě vztahu

$$\Phi_S(x) = \max_{i \in N-s} \min_{j \in s} \Delta(x_i, x_j), \tag{3.10}$$

kde $s$ značí aktuální množinu anotovaných vstupů sjednocenou s množinou již vybraných vstupů pro anotaci. Je vybrán ten vstup, pro který je výstup funkce $\Phi_S(x)$ největší. V každé iteraci je tedy vybrán neanotovaný vstup jehož centrum pokryje největší část prostoru.

**Pseudo anotace.** Dalším principiálně odlišným přístupem, jak vybrat vstupy, je výběr pomocí největší jistoty [35] (vybírá se vstup s nejmenší entropií). Tyto vstupy nemají na klasifikátor s velkou pravděpodobností žádný větší vliv, neboť gradient chyby takových vstupů je velmi malý. Výstup modelu pro tyto vstupy je však velmi jistý a tudíž jej lze využít jako anotace, tzv. pseudo anotace. Takto lze získat velké množství pseudo anotovaných vstupů a přidat je do trénovací sady. Následkem může být kvalitnější reprezentace známých/jistých vzorů v síti. V průběhu trénování se počet jistých vstupů zvětšuje. Je tedy potřeba průběžně měnit práh podle kterého se vybírají jisté vzorky tak, aby tento počet nebyl příliš velký ,tzn. jistota na začátku trénování není dostačující pro pozdní fáze trénování. Tento přístup lze kombinovat s přístupy předchozími, tzn. přidávat jak anotované nejisté vstupy, tak pseudo anotované jisté vstupy.

**Cena anotace.** Velikost ceny anotace určuje, jak náročné je anotaci získat. Cena může záviset na mnoha faktorech, např. čas potřebný k anotaci, finance, požadavky na anotátora atd. Pokud je nějaký vstup těžký z hlediska modelu je velmi pravděpodobné, že bude těžký i z hlediska lidského. Např. anotovat krátký čitelný řádek bude mnohem méně náročné než anotovat špatně čitelný dlouhý řádek. Při předpokladu, že máme k dispozici kvalitní anotátory, je cena zejména otázkou času [31]. Doba anotace závisí jak na konkrétním vstupu, tak na konkrétním anotátorovi. Jelikož je anotátor typicky člověk, tak doba anotování vstupu se může lišit v závislosti na soustředění, náladě apod. Čas potřebný k anotaci lze predikovat na základě vstupů, tzn. lze naučit model na základě vstupů a příslušné doby potřebné k anotaci. Výběr vhodného vstupu pak závisí na nejistotě, která je ale navíc normalizována předpokládanou dobou anotace.

Vzhledem k různým vlivům na dobu anotace je její predikace značně problematická. Tato práce se zaměřuje na výběr nejinformativnějších vstupů a tudíž čas potřebný k anotaci těchto vstupů neuvažuji. V praxi je ovšem možné, že hlavním limitem pro získání co nejvíce vhodných nových vstupů do trénovací sady bude čas a ne počet. Problém pak není anotovat co nejinformativnější vstupy, ale anotatovat takovou množinu vstupů, pomocí které lze dosáhnout co nejlepších výsledků za daný čas. Tyto množiny mohou být stejné, ale taky různé. Např. pokud by se anotovaly striktně nejtěžší vstupy, může být počet výsledných anotací dvakrát menší, než pokud by se anotovaly vstupy o něco jistější. Ve výsledku pak může model lépe fungovat na větší datové množině. V extrémním případě by mohlo dojít k tomu, že náhodné anotování bude stejně přínosné jako anotování nejméně jistých řádků. V pozdějších fázích trénování je však tato situace značně nepravděpodobná, neboť model zvládá naprostou většinu vstupů. Pro zlepšení výsledků je tedy nutné vybírat nejisté vstupy.

# Kapitola 4

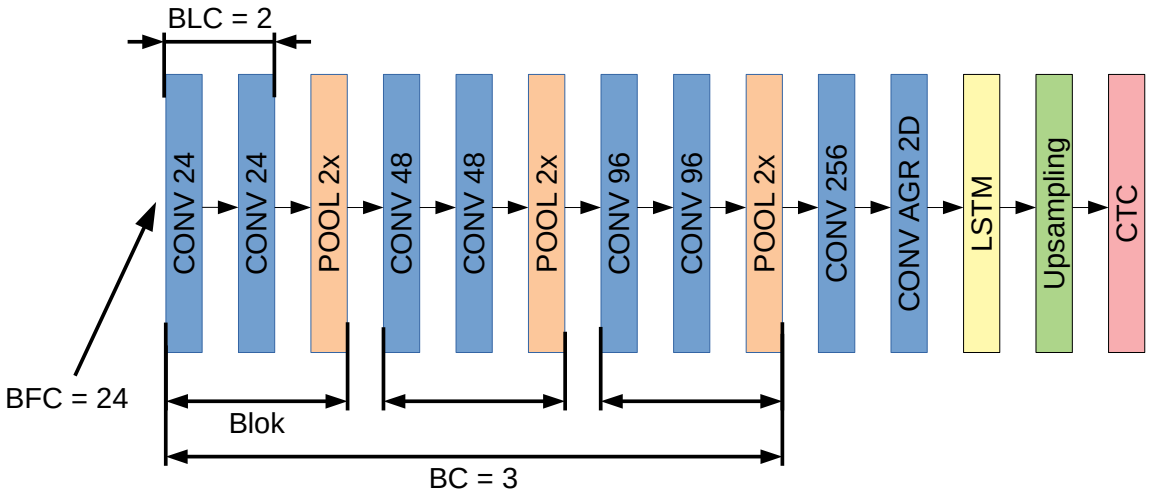
# Aktivní učení neuronových sítí pro rozpoznávání textu

V této kapitole nejprve popisuji architektury sítí, se kterými provádím experimenty. Jedná se o architektury sestávající se zejména z konvolučních a rekurentních vrstev. Dále se zabývám popisem algoritmů aktivního učení, které jsem navrhl za účelem zjednodušení procesu aktivního učení. První algoritmus provádí samotný výběr vhodných dat pro trénování na základě některé z metod aktivního učení. Na takto vybraných datech je pak možné síť přetrénovat pomocí algoritmu druhého. Dále popisuji efektivní způsob provádění křížové validace (*cross-validation*), který umožňuje maximální využití dat, jež jsou dostupné pro učení neuronových sítí. Metoda je založena na vícenásobném využití posledních vrstev neuronové sítě. Nakonec popisuji metody aktivního učení založené na pravděpodobnostní matici výstupní vrstvy CTC.

## 4.1 Architektury sítí

Architektury sítí se kterými experimentuji v této práci se skládají z vrstev konvolučních, *pooling*, ReLU, *batch* normalizace, LSTM a CTC. Konvoluční vrstvy extrahují kvalitní příznaky ze vstupního výřezu řádku jako jsou různé hrany a vzory objevující se v rámci tištěného textu. Rovněž jsou spolu s rekurentní vrstvou LSTM schopny reagovat na kontext.

Konvence pojmenování architektur je zachycena na Obrázku 4.1 za použití diagramu představujícího síť LSTM, tzn. síť která obsahuje rekurentní vrstvu. BC (*block count*) značí počet bloků sítě. Blok jsou počáteční skupiny vrstev, které začínají konvolucemi a končí vrstvami typu *pooling*. Konvoluce mají velikost jádra 3 a *pooling* 2. Jelikož je velikost kroku vrstev *pooling* 2, každý blok zmenší svůj vstup dvakrát. Každá konvoluční vrstva je následována *batch* normalizací a aktivační funkcí ReLU. Počet konvolučních vrstev v rámci bloku udává BLC (*block layer count*). Počet jader konvolučních vrstev udává vztah $BFC * (2^i)$ (*base filter count*), kde $i$ je index bloku (první blok má index 0). Za bloky následují dvě konvoluční vrstvy. Druhá z nich má na výstupu 2D příznakové mapy z důvodu následující LSTM a CTC vrstvy. LSTM je obousměrná rekurentní vrstva s LSTM bloky. Daná síť zmenší svůj vstup osmkrát. Jelikož se síť učí klasifikovat znak každé 4 pixely je nutné požít jednu zvětšovací vrstvu (*upsampling*), která zvětší šířku vstupu dvakrát. Pro čtyři bloky je nutné použít dvě takové vrstvy, pro dva bloky žádnou. Jednodušší sítě neobsahují LSTM vrstvu. Pojmenování jednotlivých architektur má formát BC_BLC_BFC.

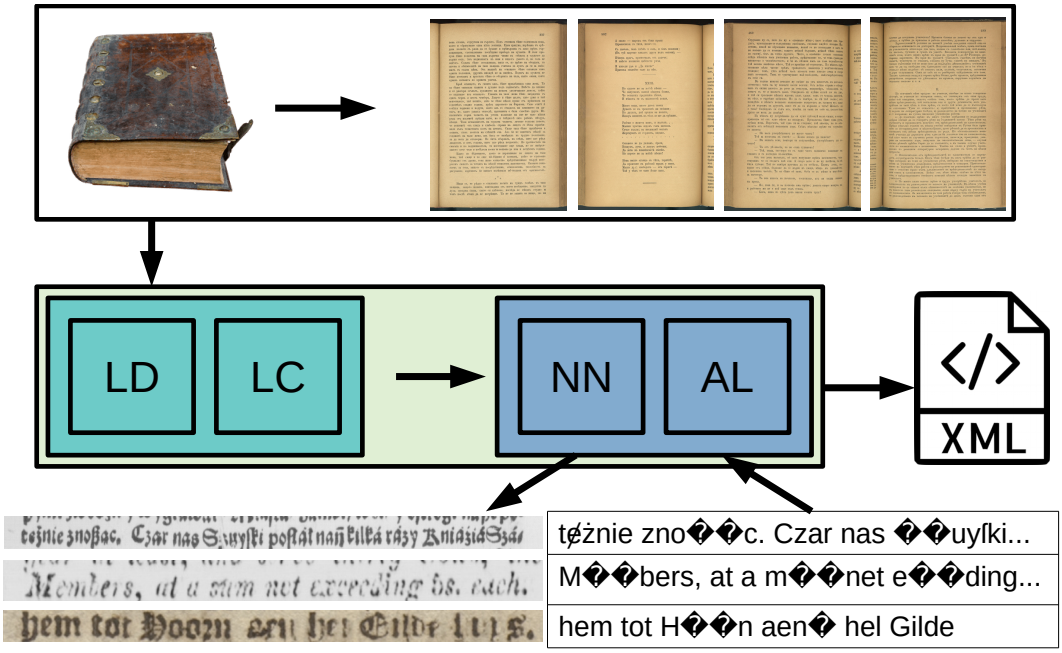



Obrázek 4.1: BC (*block count*) značí počet bloků sítě. Blok jsou počáteční skupiny vrstev, které začínají konvolucemi a končí vrstvami typu *pooling*. Počet konvolučních vrstev v rámci bloku udává BLC (*block layer count*). Počet jader konvolučních vrstev udává vztah $BFC * (2^i)$ (*base filter count*), kde $i$ je index bloku (první blok má index 0). Za bloky následují dvě konvoluční vrstvy, z nichž druhá má na výstupu 2D příznakové mapy, z důvodu následující LSTM a CTC vrstvy. Zvětšovací vrstva (*upsampling*) zvětší šířku vstupu dvakrát.

## 4.2 Algoritmy aktivního učení

Učení neuronových sítí je zpravidla prováděno na velké datové množině, která zaručuje kvalitní generalizaci sítí, tzn. chyba na trénovacích a testovacích datech je prakticky totožná. Jelikož je anotovaní řádků textu časově náročný proces je žádoucí, aby se neuronové sítě trénovaly na pokud možno co nejmenší datové množině. Požadavky na minimální množství dat a kvalitní generalizaci tedy jdou proti sobě. Jedním z řešení jak zvýšit úspěšnost neuronových sítí, pokud není k dispozici dostatek dat, je využít již natrénovanou neuronovou síť. Váhy této sítě se použijí při inicializaci vah nové sítě. Tento scénář je typický pro rozpoznávání textu, jelikož je zpravidla k dispozici velké množství řádků, které byly získány z nejrůznějších dokumentů. Naproti tomu uživatel může požadovat přepis dokumentu, jehož řádky se typově liší od řádků v původních dokumentech a tudíž musí poskytnout anotovanou množinu řádků z tohoto dokumentu (s uživatelského hlediska je žádoucí poskytnout co nejméně řádků). Neuronová síť je poté adaptována na nový typ řádků s využitím této množiny.

Malá množina dat způsobuje přetrénování neuronových sítí, tzn. chyba na trénovací množině je znatelně menší než na množině testovací. Rovněž dochází ke ztrátě schopnosti sítě kvalitně rozpoznávat řádky na kterých byla původně natrénována. Tyto nevýhody se v rámci algoritmů aktivního učení snažím minimalizovat tak, že trénuji neuronové sítě na různých poměrech původních a adaptačních dat. Původní data jsou data na kterých byla natrénována původní neuronová síť, tzn. neuronová síť, která se využije pro inicializaci vah sítě nové. Adaptační data představují malou množinu dat, která je typově odlišná od původních dat.




Obrázek 4.2: Schéma serveru pro automatickou anotaci dokumentů.

**Server pro automatickou anotaci dokumentů.** V rámci experimentů se zabývám především adaptací neuronové sítě na menší (adaptační) datovou sadu za pomoci metod aktivního učení. Pro ujasnění smyslu adaptace uvádím schéma serveru pro automatickou anotaci dokumentů, viz Obrázek 4.2. Uživatel chce přepsat (rozpoznat) knihu, která ale není přítomna v datasetu na kterém se učila neuronová síť serveru. Po naskenování jednotlivých stran knihy jsou výsledné obrázku stránek odeslány serveru. Server nejprve provede detekci a vyřezání řádků, část LD a LC. Poté dojde k rozpoznání řádků pomocí neuronové sítě (NN). Uživateli jsou zobrazeny výsledné přepisy řádků. Pokud je uživatel spokojen, server vrátí přepisy dokumentu a proces rozpoznání/přepisu je ukončen. Pokud uživatel není spokojen s kvalitou přepisů, je využito některé z metod aktivního učení k tomu, aby byly vybrány řádky z uživatelem nahrané knihy (AL). Tyto řádky musí uživatel následně anotovat, aby mohly být využity pro adaptaci. Anotace nových řádků, adaptace a rozpoznání se provádí v cyklu tak dlouho, dokud uživatel není spokojen. V experimentech v rámci této práce se věnuji částem NN a AL.

Aby bylo možné provádět smysluplné experimenty, vytvořil jsem dva algoritmy aktivního učení. Tyto algoritmy do značné míry automatizují proces aktivního učení za pomoci několika málo parametrů. Tyto parametry definují množství dat, které budou vybrány metodou aktivního učení, míru přetrénování sítě, poměr původních a adaptačních dat a také horní limity při kterých je algoritmus ukončen (např. bylo přesaženo určité množství adaptačních řádků nebo se síť příliš přetrénovala).

Oba algoritmy využívají dataset, který umožňuje míchání původních a adaptačních dat v daném poměru. Např. při poměru 0.01 je pravděpodobnost, že se vybere řádek z původní datové množiny 0.99 zatímco k výběru řádku z adaptačního datasetu dojde s pravděpodobností 0.01. Toto míchání má za úkol zamezit přetrénování neuronových sítí na adaptační data kterých je málo.

**Algorithm 1** Algoritmus pro postupné učení neuronové sítě

```
1: function AdaptNet(net, origData, newData, ALMethod, baseRatio, seen)
2:     pickedAdaptData = []
3:     while size(pickedAdaptData) < maxPickedAdaptData do
4:         pickedAdaptData.append(PickAdaptData(net, ALMethod, newData))
5:         SaveDataset(datasetName, pickedAdaptData)
6:         RestoreOrigNet(net)
7:         ratio = baseRatio
8:         numberOfIter = 0
9:         adaptAttempts = 0
10:        while numberOfIter < maxIter do
11:            if NotOverfitted(net, pickedAdaptData) then
12:                if adaptAttempts > maxAdaptAttempts then
13:                    ratio = IncreaseRatio(ratio)
14:                    adaptAttempts = 0
15:                adaptIter = TrainNet(pickedAdaptData, origData, ratio, seen)
16:                numberOfIter += adaptIter
17:                adaptAttempts += 1
18:            else
19:                break
```

Dále oba algoritmy umožňují výběr vrstev, které se mají v průběhu trénovat. Je možné vybrat vrstvy všechny nebo libovolný počet vrstev jak od začátku, tak od konce sítě. Možnost volby vrstev, které se mají trénovat, je důležitá z hlediska zjištění, které vrstvy je nutné trénovat pro to, aby bylo dosaženo kvalitní adaptace. Výběr pouze posledních několika vrstev také zrychluje samotný průběh adaptace, jelikož není nutné pro aktualizaci vah sítě počítat všechny zpětné průchody vrstev.

**Algoritmus pro postupné učení neuronové sítě.** Prvním algoritmem je algoritmus pro postupné učení neuronové sítě, viz Algoritmus 1. Cílem algoritmu je postupné zvětšování množiny adaptačních dat a adaptace neuronové sítě na tuto postupně se zvětšující množinu.

Následující popis se týká jednoho provedení hlavní (nejvíce vnější) smyčky algoritmu. Algoritmus nejprve vybere určitý počet dat (výřezů řádků a přepisů) pro adaptaci, tzn. simulace anotování uživatelem. Tento výběr je prováděn na základě některé z metod aktivního učení nebo náhodně. Následně dojde k uložení vybraného datasetu pro pozdější přeučení neuronové sítě. Neuronová síť se obnoví do svého původního stavu, tzn. stavu kdy je naučena pouze na původním datasetu. Vnitřní smyčka zajišťuje trénování neuronové sítě na původních a adaptačních datech. Poměr dat je určen pomocí parametru `ratio`. Pokud nedochází k přetrénování sítě na adaptačních datech, je poměr dat adaptačních ku původním zvětšován. Jakmile dojde k přetrénování, začne se provádět další iterace smyčky, tzn. jsou přidány nové anotace a síť je opět obnovena do původního stavu.

Funkce `NotOverfitted` kontroluje přetrénování na základě validační a trénovací chyby podle vztahu:

$$E = \frac{E_{val} - E_{trn}}{E_{val}}, \tag{4.1}$$

kde $E_{val}$ je validační chyba a $E_{trn}$ chyba trénovací. Kontrola přetrénování je prováděna na základě prahu. Aby byla síť považována za nepřetrénovanou, musí hodnota $E$ být menší

**Algorithm 2** Algoritmus pro přeučení neuronové sítě

```
1: function OvertrainNet(net, origData, datasetName, ratio, numberOfIter)
2:     dataset = LoadDataset(datasetName)
3:     RestoreOrigNet(net)
4:     for i in range(numberOfIter) do
5:         OvertrainNet(dataset, origData, ratio)
```

nebo rovna prahu. Vztah vyjadřuje o kolik procent validační chyby je větší validační chyba oproti trénovací. Intuitivně je tento vztah závislý na velikostech chyb a to tak, že pokud jsou obě chyby velké je zde větší tolerance chyby ,tzn. chyby se můžou lišit o poměrně velkou hodnotu. Naopak pokud jsou obě chyby malé, způsobí i menší rozdíl překročení prahu.

Funkce `PickAdaptData` představuje nejdůležitější část algoritmu, jelikož zajišťuje výběr nových adaptačních dat pro anotaci z množiny všech potenciálních adaptačních dat (`newData`). Nová adaptační data jsou tedy vybrány na základě neuronové sítě z předchozí iterace smyčky.

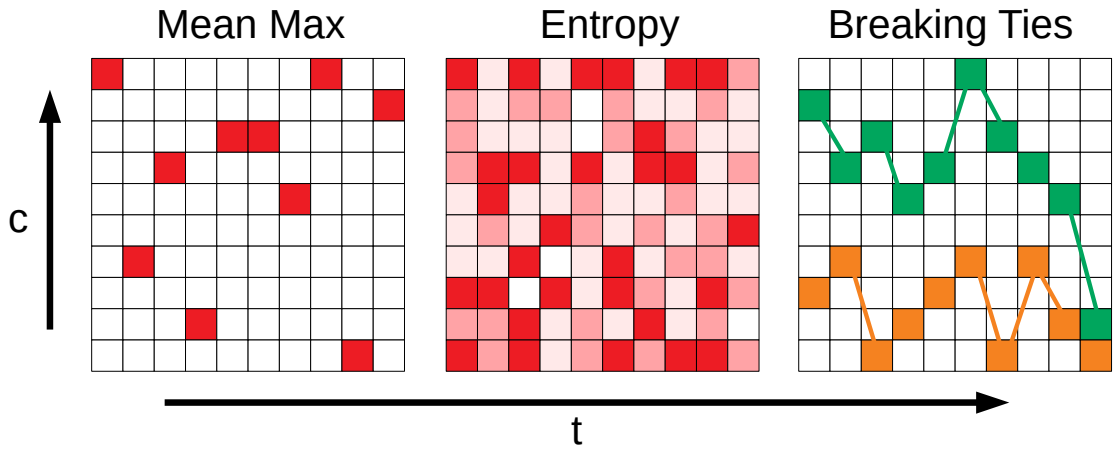
Na přetrénování má vliv funkce `TrainNet`. Parametr `seen` udává, kolikrát mají být „viděna“ adaptační data neuronovou sítí, tzn. přetrénování není kontrolováno dokud nejsou všechna adaptační data viděna alespoň jednou. Čím menší je poměr adaptačních a původních dat, tím trvá volání trénovací funkce déle, jelikož musí být provedeno více iterací. Adaptační data jsou míchána s původními tak, že adaptační data jsou vybírána postupně na základě permutace jejich indexů. Tento přístup je zvolen na základě předpokladu, že před zvětšením počtu adaptačních dat by měla síť vidět všechna současná adaptační data alespoň jednou.

**Algoritmus pro přeučení neuronové sítě.** Druhým algoritmem je algoritmus pro přeučení neuronové sítě, viz Algoritmus 2. Tento algoritmus využívá datasety uložené prvním algoritmem a na těchto datasetech přeučuje neuronovou síť. Smysl tohoto algoritmu spočívá v ověření kvality vybraných dat prvním algoritmem. Čím větší úspěšnosti se podaří dosáhnout na nezávislé testovací sadě, tím kvalitnější data jsou.

**Metody aktivního učení pro rozpoznávání textu.** Hlavním parametrem funkce Algoritmu 1 `PickAdaptData` je parametr `ALMethod`, který udává, jaká metoda aktivního učení se použije pro výběr nových dat. Využívám celkem čtyři metody z nichž tři jsou založeny na pravděpodobnostní matici a jedna na samotných výstupech sítě tzn. přepisech. Metody ohodnocují řádky příslušnou hodnotou. Řádky jsou poté řazeny podle hodnoty vzestupně nebo sestupně a ze seřazené posloupnosti je vybrán požadovaný počet řádků, respektive dvojic výřez řádku a jeho přepis.

Obrázek 4.3 ukazuje princip jednotlivých metod aktivního učení za pomocí pravděpodobnostní matice. Jednotlivé matice mají ve směru osy $y$ čas, respektive pozici, ve vstupním výřezu, ve směru $x$ pak pravděpodobnost výskytu pro daný znak. První metoda *Mean Max* (MM) ohodnocuje řádek jako průměr všech maximální pravděpodobností skrz čas, výběr jednotlivých pravděpodobností je označen červenými čtverečky. Řádky ohodnocené touto metodou jsou řazeny vzestupně, jelikož řádek, kterým si je síť méně jistá, dostane nižší skóre. Ohodnocení touto metodou představuje jistou míru nejistoty.

Druhá metoda využívající pravděpodobnostní matici *Entropy* (EN) počítá entropii v rámci každé pozice výřezu, tzn. sloupce matice. Výsledné ohodnocení je průměrem všech entropií pozic, řádky jsou řazeny sestupně, tudíž řádky s větší entropií jsou z hlediska trénování po-




Obrázek 4.3: Princip metod aktivního učení využívající pravděpodobnostní matici. Červené čtverečky metody *Mean Max* značí výběr maximální pravděpodobnosti v rámci daného časového místa. Různé odstíny červené u metody *Entropy* značí velikost jednotlivých pravděpodobností se kterými metoda pracuje. Zelený a oranžový průchod pravděpodobnostní maticí u metody *Breaking Ties* značí dva nejpravděpodobnější průchody.

važovány za zajímavější/informativnější. Odstíny červených čtverečků představují velikosti pravděpodobností v daných místech nad kterou se počítá entropie podle vztahu:

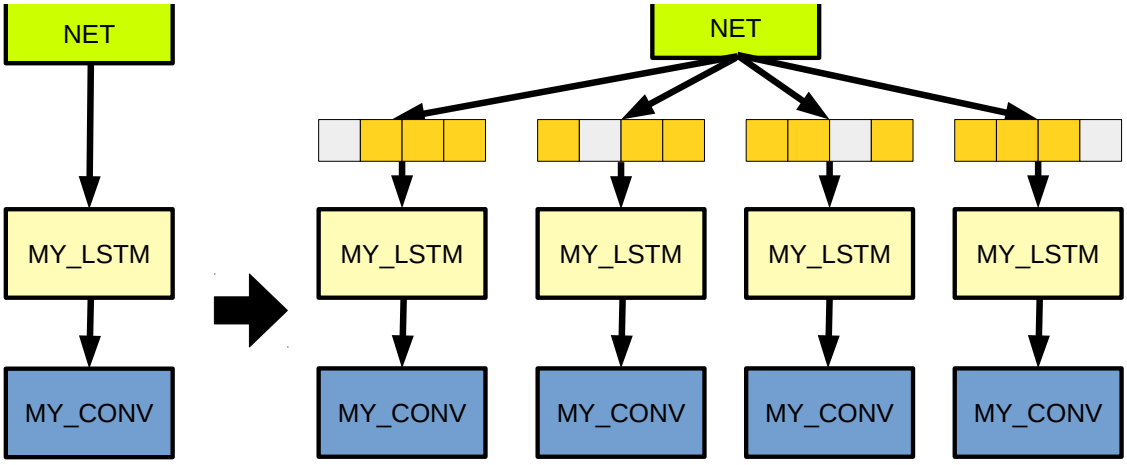
$$E = -\sum_{i=0}^{N} P_i * log(P_i), \tag{4.2}$$

kde $i$ značí pozici ve sloupci matice. Metody MM a EN neuvažují při výpočtu sloupce, kde jistota prázdného symbolu (*blank*) je větší nebo rovna 0.95. Pokud by se uvažovaly sloupce, které mají velmi jistý výskyt prázdného symbolu, výsledná pravděpodobnost/entropie by byla do značné míry ovlivněna. Řádky s velkým výskytem prázdných symbolů by pak byly velmi jisté a to i navzdory skutečnosti, že pozice představující skutečné znaky ve výřezu by byly velmi nejisté.

Poslední metoda *Committee* (CO) je založena na využití několikanásobné výstupní vrstvy (popisuji v následující podkapitole). Neuronová síť je trénována tak, že má více výstupních vrstev. Každá z těchto vrstev má mírně odlišné chování, což má za následek generování různých přepisů. Každá vrstva z posledních vrstev tedy vygeneruje mírně odlišný přepis od přepisu vrstev ostatních (může vygenerovat i stejný přepis). Přístupem každý s každým je následně vypočítána Levenštejnova vzdálenost. Výsledné skóre pro daný řádek je spočteno jako suma dílčích vzdáleností, řádky jsou řazeny sestupně.

## 4.3 Efektivní křížová validace

Jelikož algoritmy popsané v předchozí podkapitole typicky pracují s velmi malým množstvím adaptačních dat, je žádoucí tato data co nejlépe využít. Jeden ze způsobů představuje křížová validace, při které jsou data rozdělena na několik stejně velkých množin. Poté je trénováno tolik modelů, kolik množin je k dispozici s tím, že jedna množina je použita pro testování a zbytek pro trénování (každý model využívá jinou množinu pro testování a tím pádem se liší i množina pro trénování).




Obrázek 4.4: Princip namnožení posledních vrstev neuronové sítě. Vlevo před šipkou původní neuronová síť, vpravo za šipkou neuronová síť s více koncovými vrstvami a maskováním.

Trénování několika neuronových sítí by bylo značně zdlouhavé a komplikované z hlediska vyhodnocování výsledků. Z tohoto důvodu jsem se rozhodl namnožit poslední vrstvy sítě tak, jak znázorňuje Obrázek 4.4. Namnožení provádím tak, že vytvořím nový model, který má několik koncových vrstev, dále větví. Váhy původních koncových vrstev poté nakopíruji do koncových vrstev nových.

Trénování sítě, respektive křížovou validaci s několika větvemi, provádím pomocí maskování. Princip maskování je znázorněn na Obrázku 4.4 jako obdélníky s šedými a oranžovými čtverečky. Datový vzorek (*batch*) o velikosti 32 je vytvořen tak, že prvních 8 výřezů řádků patří do testovací množiny první větve. Dalších 8 výřezů patří do testovací množiny druhé větve, atd. Maskování má za úkol vyfiltrovat testovací řádky pro danou větev sítě (šedé čtverečky). Testovací řádky ostatních větví tvoří trénovací řádky (pro danou větev) a tudíž jsou ponechány (oranžové čtverečky).

Testování datové sady na libovolné větvi sítě se provádí tak, že je zrušeno maskování a výpočet probíhá pouze pro danou větev (ostatní větve jsou ignorovány a výpočet zde vůbec neprobíhá). Při testování celé sítě na nezávislé datové množině je opět zrušeno maskování, ale výpočet probíhá ve všech větvích. Vyhodnocování pomocí metod aktivního učení MM, EN a BT probíhá tak, že se vyhodnotí všechny větve a výsledné skóre řádku je dáno jako suma ohodnocení v rámci jednotlivých větví.

# Kapitola 5

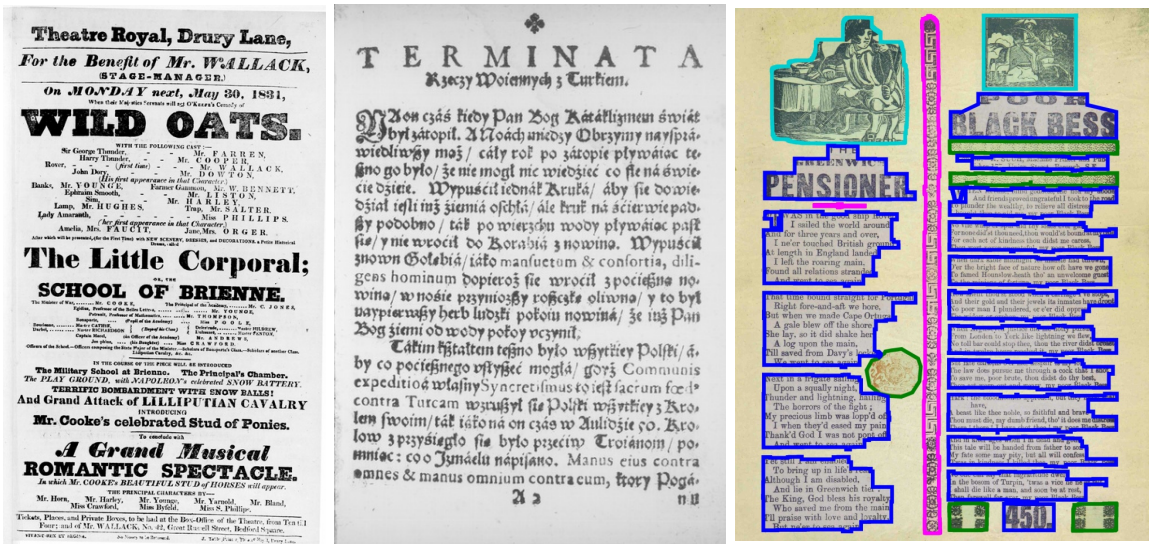
# Dataset IMPACT

Pro experimenty s aktivním učením je potřebný velký dataset anotovaných vstupů. Pomocí tohoto datasetu lze efektivně simulovat anotaci jednotlivých vstupů a tím průběh samotné metody aktivního učení. V rámci této práce využívám dataset IMPACT [23]. Tento dataset 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 a různých období. Rozmanitost je také dána 9 jazyky a 10 fonty. Dataset tedy realisticky pokrývá všechny knihovny a tím pádem historické texty dostupné v rámci Evropy.

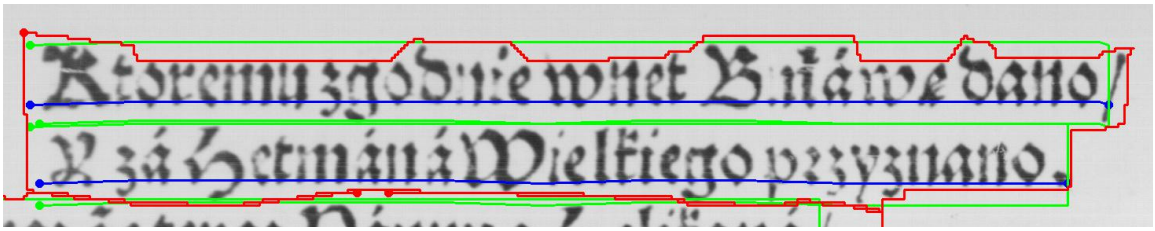
Celkem bylo získáno přes 600000 obrázků. Nebylo reálné anotovat takové množství textu a tak byla anotováno pouze podmnožina stránek. Podmnožina je rozdělena rovnoměrně mezi jednotlivé knihovny tak, aby reprezentovala celkový vzorek dat a obsahuje okolo 45000 obrázků stran. Anotace obsahují zejména polygony těsně obalující jednotlivé textové objekty stránek (odstavce, nadpisy atd.) s příslušným přepisem textu. Obrázek 5.1 ukazuje tři rozdílné stránky datasetu, kdy na poslední straně jsou vyznačeny polygony obalujícími jednotlivé objekty. Dále jsou k dispozici informace o objektech, které nepředstavují text, jako jsou obrázky a oddělovače. Samotná anotace textu je provedena partnery projektu IMPACT (různí pracovníci a knihovny) za pomocí znaků podle normy unicode. Jednotlivé speciální znaky a ligatury (více znaků je spojeno/svázáno do jednoho) tedy nejsou nahrazovány současnými znaky, ale je použit jejich přesný ekvivalent v rámci unicode. Pokud nějaký znak neexistoval v rámci unicode, byl definován znak nový pomocí soukromé uživatelské oblasti. Anotátoři mohli při anotaci postupovat samostatně nebo využít pomoci již existujícího OCR Aletheia [7] a opravit výsledky. Kontrola kvality textových anotací je provedena samotnými knihovnami. Při práci s takto rozsáhlým datasetem nelze zaručit absolutní přesnost, proto byla stanovena na 99.95 %.

## 5.1 Zpracování datasetu

Jako první jsem se zaměřil na analýzu anotací v rámci datasetu. Anotace obsahují celkem 636 znaků s 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 již zmíněné 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“.




Obrázek 5.1: Ukázky různých stránek datasetu IMPACT. Vpravo stránka s vyznačenými polygony obalujícími jednotlivé objekty stránky (převzato z [23]).



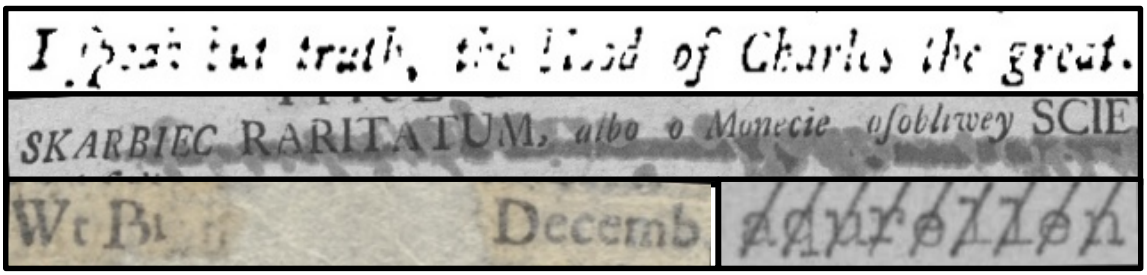
Obrázek 5.2: Ukázka detekce řádků na stránce datasetu IMPACT. Kde červený polygon značí odstavec, modrá linka základní linku textu (*baseline*) a zelené linky horizontální ohraničení řádku.

Dále zde byl problém různých druhů pomlček, kdy krátká pomlčka vypadající vizuálně stejně byla anotována několika různými unicode znaky, které představovali „krátkou čárku“ (stejný problém i pro dlouhé pomlčky). Tyto různé pomlčky jsem sloučil do dvou znaků, kdy jeden představuje všechny krátké pomlčky a druhý všechny dlouhé. Těmito úpravami jsem počet znaků zmenšil na 263.

Jelikož tento dataset neobsahuje informace o pozicích řádků, prvním krokem při zpracování byla detekce řádků. Řádky byly detekovány v rámci odstavců stránek za pomoci neuronových sítí. Výstupem těchto sítí jsou souřadnice bodů linky, na které leží řádek a linek horizontálně ohraničujících řádek, viz 5.2. Kde červený polygon značí odstavec, modrá linka základní linku textu (*baseline*) a zelené linky horizontální ohraničení řádku (odhad výšky). Celkem bylo získáno 1.3 milionu řádků. Tato detekce byla provedena Ing. Oldřichem Kodymem v rámci týmu pracujícího na projektu PERO.

Detekce linek řádků nemusí být zaručeně správná a tudíž mohou vznikat výřezy, které obsahují různě deformované řádky. Dále mohou výřezy obsahovat zcela nečitelné posloupnosti znaků. Takovéto výřezy 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 byl největší. Řádky byly sestupně seřazeny jak podle chyby, tak podle chyby normalizované délkou řádku


Obrázek 5.3: Ukázka špatně vyřezaných řádků datasetu IMPACT. Nahoře řádek obsahující velká písmena, uprostřed řádky s méně kvalitním textem (první dva) a textem netypickým (poslední). Dole řádek, který nesedí na anotaci, která je zvýrazněna růžovým podbarvením.




Obrázek 5.4: Ukázka špatně čitelných řádků datasetu IMPACT.

(chyba závisí na délce řádku, normalizací vzniká jistá míra chybovosti vztažená na znak). Vizuálně jsem odhadl 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 Levenštejnovi vzdálenosti anotace a přepisu generovaného sítí. Oba řetězce na sebe byly zarovnány. Pokud se začátek nebo konec řetězce nepodařilo zarovnat, tzn. počáteční/koncové znaky řetězců na sebe neseděly, došlo k odstranění takové anotace. Pro velmi krátké řetězce (délka kratší než 10 znaků) jsem rozdíl zarovnání stanovil na 1, pro delší řetězce na 2. Celkem bylo těmito přístupy odstraněno více než 60000 podezřelých řádků.

Ukázky špatných výřezů jsou na Obrázku 5.3 a ukázky špatně čitelných výřezů na Obrázku 5.4. Špatné výřezy vznikají důsledkem špatného odhadu výšky řádku, což je způsobeno řádky obsahujícími pouze velká písmena a řádky s méně kvalitním či netypickým textem. Dále jsou za špatné výřezy považovány ty, které nejsou zarovnány (nesedí) na svoji anotaci. Dolnímu řádku na Obrázku 5.3 chybí začátek a konec (růžově podbarvené části). Nastat může i opačná situace, kde naopak výřez řádku bude přesahovat jeho anotaci. Špatně čitelné řádky vznikají kvůli poškozeným dokumentům nebo nekvalitním zdrojovým obrázkům.

Negativní dopad takovéto filtrace je 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 uměle, tím pádem je dosaženo větší kontroly, která je důležitá z hlediska experimentů v rámci aktivního učení. Filtrace a odhalování chyb není jednoprůchodová záležitost. Při získání

| | es | nl | en | sl | pl | fr | bg | it | cs | k | N | Ú |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D | 14 | 38 | 43 | 29 | 29 | 12 | 5 | 1 | 1 | 131 | 37 | 4 |
| Ř | 353k | 249k | 176k | 166k | 99k | 81k | 79k | 11k | 2k | 845k | 299k | 72k |
| Z | 20M | 17M | 11M | 10M | 7M | 5M | 5M | 0.6M | 0.2M | 51M | 20M | 5M |
| P | 58 | 67 | 62 | 62 | 70 | 65 | 65 | 53 | 68 | 61 | 66 | 68 |

Tabulka 5.1: Levá část tabulky: 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. Pravá část tabulky: mapování typu dokumentu. K značí knihy, N noviny a Ú úřední dokumenty.

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.

Pro rychlejší trénování jsem neuronové sítě trénoval pouze na výřezech s výškou 32px a šířkou do 1024px, přitom všechny výřezy jsou upraveny tak, aby měly právě tuto velikost. Nejprve je řádek škálován na požadovanou výšku. Úpravu kratších výřezů jsem provedl přidáním rovnoměrných černých okrajů z každé strany. Delší řádky jsem do výsledné množiny neuvažoval. Tímto způsobem lze efektivně trénovat síť na více řádcích zároveň. Výřezů, které splňují dané omezení (i co se týče zmiňované chybovosti), je celkem 1217000. Poměr velikosti trénovací a testovací sady je 99/1. Dataset je dále uměle zvětšován pomocí různých transformačních operací nad řádky, tzn. posun, rotace, zvětšení. Dále je uměle přidáván šum a měněna saturace a barva.

## 5.2 Mapování datasetu

Pro experimenty jsem vytvořil mapování různých vlastností dokumentů na řádky textu. Tato mapování (až na mapování jazyků) jsem vytvořil na základě několika stránek daného dokumentu manuálně. Přehled mapování jazyků dokumentů a typů dokumentů je v Tabulce 5.1. Levá část tabulky obsahuje 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. Pro určení jazyka dokumentů jsem využil knihovnu pro detekci jazyků *langdetect* [33]. V pravé části tabulky je mapování typu dokumentu. K značí knihy, N noviny a Ú úřední dokumenty. Úředními dokumenty jsou například zápisy v matrice apod. Dalšími mapováními jsou typ písma a kvalita, viz Tabulka 5.2. 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). Dokument může mít více typů písma. Kvalita je hodnocena na základě čitelnosti dokumentu. Hodnocení je od 1 do 5, kdy 5 je nejhorší. Nejvíce zastoupeným jazykem je španělština následovaná nizozemštinou a angličtinou. Dataset je ze tří čtvrtin tvořen knihami. Nejvíce je zastoupený normální typ písma a většina dokumentů je bez problémů čitelná (kvalita 1 nebo 2). Důvodem horší čitelnosti bývá poškození dokumentu nebo špatná kvalita skenů.

Pro získání uživatelem specifikovaných typů řádku jsem vytvořil skript, který na základě mapování vrátí příslušnou množinu řádků. Např. uživatel může požadovat pouze řádky z knih v anglickém a nizozemském jazyce o obtížnosti 2 a typem písma *normal*. Pro zjednodušení práce jsem také implementoval možnost reverzního výběru, tzn. uživatel může požadovat všechny řádky, které nesplňují nějaké kritérium výběru.

| | NC | N | H | Y | HC | NH | NHC | 2 | 1 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D | 66 | 66 | 26 | 5 | 3 | 2 | 4 | 113 | 27 | 31 | 1 |
| Ř | 589k | 446k | 87k | 79k | 8k | 5k | 6k | 889k | 181k | 146k | 1k |
| Z | 35M | 29M | 5M | 5M | 0.5M | 0.4M | 0.3M | 55M | 12M | 9M | 0.07M |
| P | 60 | 66 | 63 | 65 | 66 | 75 | 53 | 62 | 67 | 64 | 78 |

Tabulka 5.2: Levá část tabulky: mapování typu písma dokumentů datasetu IMPACT na řádky textu. N je normální text podobný soudobému tištěnému textu, H je historický text jako gotika nebo švabach, C značí krasopisné písmo a Y cyrilici. Pravá část tabulky: mapování obtížností dokumentů. 1 nejjednodušší pěkně čitelný text až 4 špatně čitelný, poškozený text.

# Kapitola 6

# Experimenty

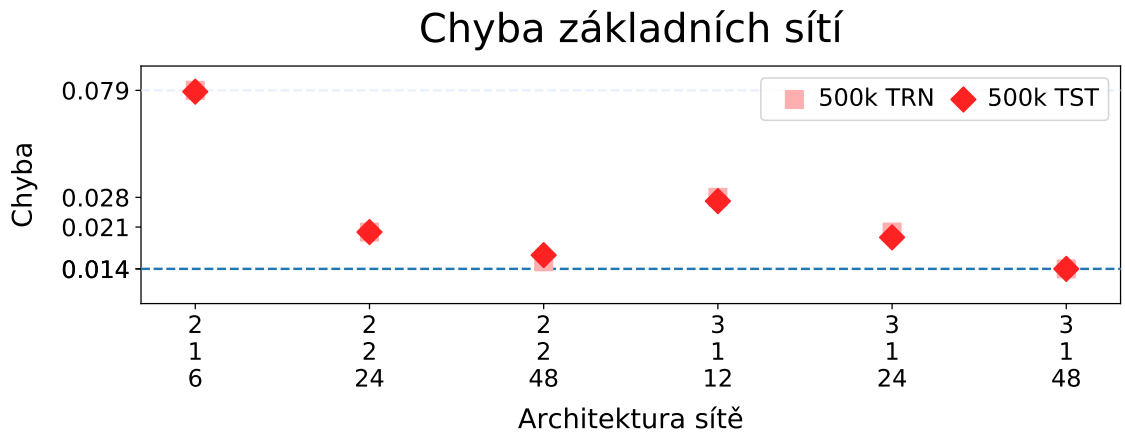
V této kapitole se nejprve zabývám klasickým trénováním neuronových sítí, kde porovnávám úspěšnost čistě konvolučních neuronových sítí a sítí s rekurentní vrstvou LSTM. Dále popisuji experimenty, které jsem provedl jak v rámci přístupů k učení neuronových sítí, tak v rámci metod aktivního učení. Experimenty s přístupy učení zkoumají především vliv různých parametrů algoritmů aktivního učení na chováni/vývoj neuronových sítí v průběhu učení a kvalitu vybraných dat. Experimenty s metodami aktivního učení srovnávají více metod aktivního učení.

## 6.1 Architektury sítí

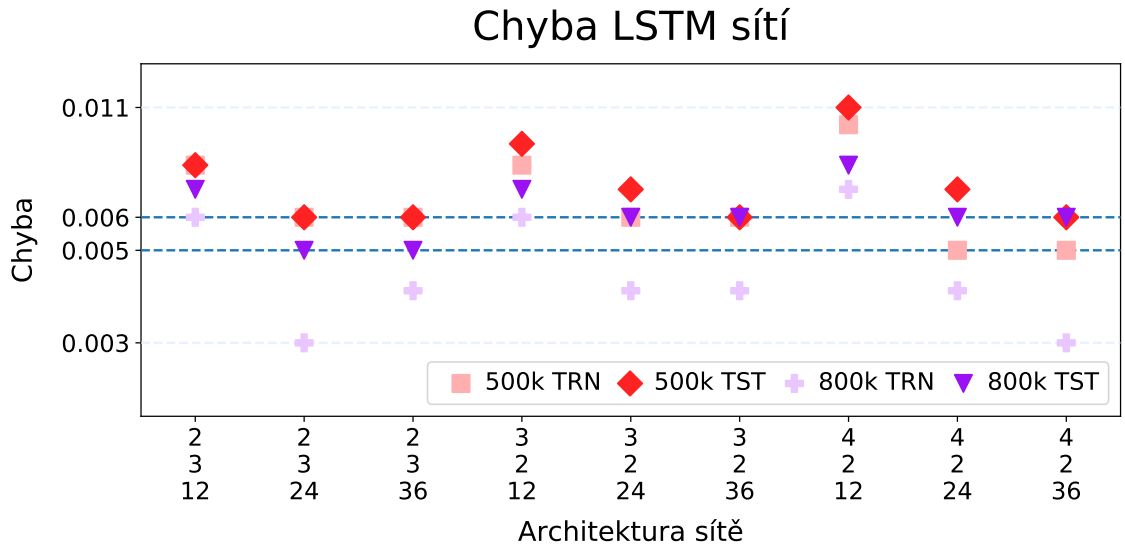
Cílem experimentů je ověřit kvalitu datasetu respektive anotací a daných výřezů. Dále pak srovnat účinost sítí bez a s rekurentní vrstvou. Experimenty s klasickým trénováním neuronových sítí využívají celý dataset IMPACT tzn. 1.2M řádků, kde poměr trénovací a testovací sady je 99/1.

Nejprve jsem natrénoval jednodušší sítě bez LSTM vrstvy, viz graf na Obrázku 6.1. 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[1] a optimalizační metodu založenou na gradientním sestupu *Adam* [16]. Výsledky se zlepšují s rostoucí velikostí sítí, tzn. počtem trénovaných vah. Chyba na trénovací i testovací sadě je srovnatelná, sítě úspěšně generalizují a nedochází k přetrénování. Nejlepší síť má průměrnou chybu 1 znak na 70 znaků, 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). Např. LSTM síť 3_2_24 je z hlediska výpočetní náročnosti srovnatelná s jednodušími sítěmi 2_2_24 a 3_1_48.

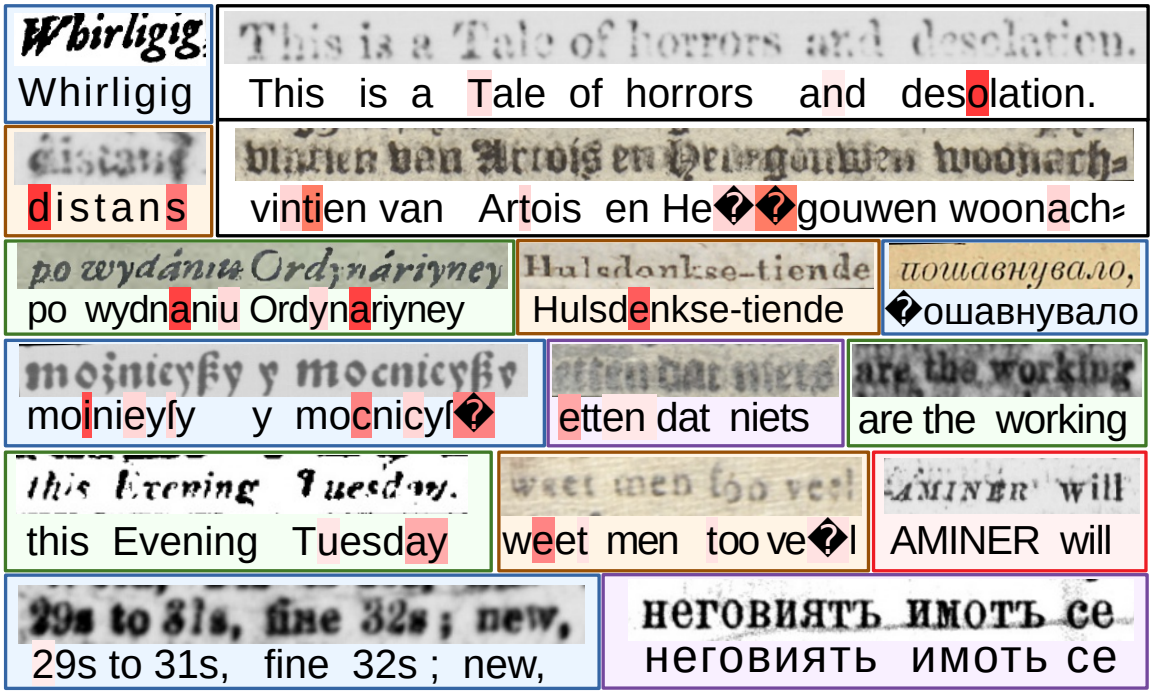
[1] TensorFlow: Large-Scale Machine Learningon Heterogeneous Systems (http://tensorflow.org/).




Obrázek 6.1: Výsledky experimentů základních sítí na datasetu IMPACT. Světle červené čtverečky představují chybu na trénovací sadě, červené kosočtverce pak na sadě testovací. Sytá přerušovaná čára značí nejlepší dosažený výsledek na testovací sadě. Osa představující chybu je v logaritmickém měřítku.



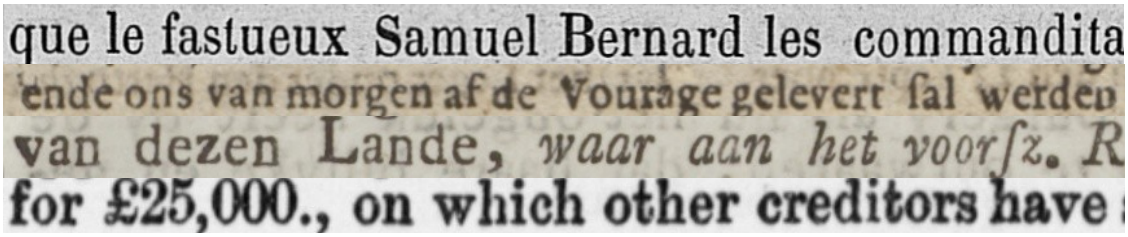

Obrázek 6.2: Výsledky experimentů LSTM sítí na datasetu IMPACT. Světlé symboly představují chybu na trénovací sadě, tmavé pak na sadě testovací. Červená barva představuje počátečních 500000 iterací, fialová dotrénování pomocí dalších 300000 iterací. Sytě přerušované čáry značí nejlepší dosažený výsledek na testovací sadě pro počáteční iterace a dotrénování. Osa představující chybu je v logaritmickém měřítku.



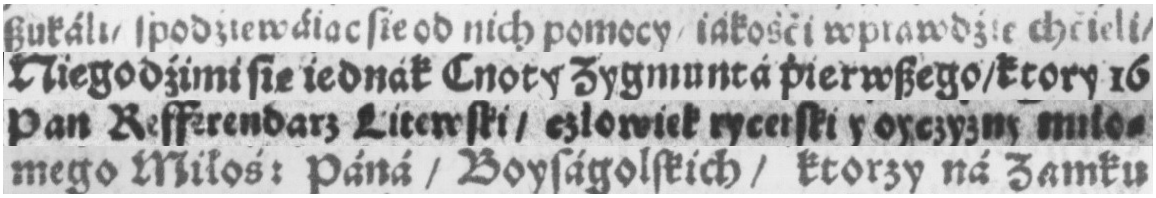

Obrázek 6.3: Ukázka složitějších řádků datasetu IMPACT, které neuronová síť ještě dokáže správně přepsat tzn. přepisy sítě jsou totožné s anotacemi. Červeně podbarvené znaky značí pravděpodobnost nejjistějšího znaku, čím je tato pravděpodobnost nižší, tím je červená barva tmavější.

Graf na Obrázku 6.2 prezentuje výsledky pro sítě s vrstvou LSTM. Světle červené čtverečky opět představují úspěšnost na trénovací sadě a červené kosočtverce na testovací. Parametry trénování jsou stejné jako u jednodušších sítí. Je patrné značné zlepšení, a to i pro sítě výpočetně srovnatelně, tzn. jednoduchá síť 2_2_24 a LSTM síť 3_2_24. 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, učící konstanta zmenšena na polovinu a velikost batch zvětšena dvojnásobně. Výsledky jsou znázorněny pomocí fialových symbolů, kde světlý opět představuje úspěšnost na trénovací sadě a tmavý na sadě 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. Je tedy žádoucí ověřit, na kterých řádcích síť chybuje, pokud by např. některé řádky byly špatně anotovány nebo nebylo možné je přečíst, nemá cenu se snažit dosáhnout větších úspěšností.

Složitější řádky, které neuronová síť ještě dokáže správně přepsat tzn. přepisy sítě jsou totožné s anotacemi, jsou na Obrázku 6.3. Červeně podbarvené znaky značí pravděpodobnost nejjistějšího znaku. Čím je tato pravděpodobnost nižší, tím je červená barva tmavější. Neuronová síť je schopna rozpoznat velké množství typově odlišných řádků textu a to z hlediska jazyku, textury, míry eroze a dilatace. Chyby se vyskytují v místech, kde je text špatně čitelný, a to i z lidského hlediska. Při vizuálním pozorování chybových řádků šlo



Obrázek 6.4: Ukázka řádků původní množiny, na které byla netrénována síť pro adaptaci. Písmo je typu *normal* a *calligraphy*.



Obrázek 6.5: Ukázka řádků polské adaptační množiny. Písmo je typu *historic*.

většinou o chyby tohoto typu. Dalším problém jsou již zmiňované špatně ořezané řádky a to jak z hlediska špatně odhadnuté výšky (ve výřezu je vidět značná část jiného řádku nebo dokonce celý, případně je řádek příliš úzce vyřezán), tak z hlediska šířky (začátek nebo konec řádku je ořezán a anotace tudíž nesedí). Vzhledem k charakteru chyb lze potvrdit kvalitní anotace tohoto datasetu a tudíž i jeho vhodnost pro experimenty s aktivním učením.

## 6.2 Datové sady pro aktivní učení

Experimenty s aktivním učení využívají dvě datové sady. První datová sada jsou původní data na kterých byla natrénována adaptovaná síť a v druhé datové sadě jsou samotná adaptační data, ze kterých se vybírají data pro adaptaci. Jelikož v rámci experimentů nemám k dispozici anotátory a navíc by případné anotace byly časově náročné, provádím simulaci anotace pomocí „odhalování“ anotací datasetu IMPACT. Anotace se poté provede prostým přiřazením již existující anotace v rámci datasetu IMPACT.

Celkem jsem zvolil dva scénáře adaptace. První scénář (PLH) uvažuje jako původní množinu dat dokumenty obsahující pouze typ písma *normal* a *calligraphy*, tzn. písmo podobné soudobému tištěnému textu s možností výskytu ozdobného písma. Původní množina navíc neobsahuje žádné slovanské jazyky. Celkový počet řádků je 770k. Ukázka typických řádků původní množiny je na Obrázku 6.4. Jako adaptační data jsem zvolil polské historické dokumenty. Tyto dokumenty jsou pouze v polském jazyce a typ písma je *historic*, tzn. staré gotické písmo apod. Celkový počet řádků je 12k. Ukázka typických řádků polské adaptační množiny je na Obrázku 6.5. Jako druhý scénář (NPN) jsem zvolil adaptaci neuronové sítě natrénované pouze na knihách s typem písma *normal* a *calligraphy* na noviny s typem písma *normal*. Množina původních dat má velikost 790k řádků, adaptační pak 85k řádků.

První scénář jsem zvolil z důvodu značné odlišnosti původních a adaptačních datových sad. Odlišnost zaručuje, že chyba adaptované neuronové sítě bude před adaptací vysoká a tudíž bude porovnání jednotlivých přístupů pro adaptaci snazší. Druhý scénář slouží

k potvrzení/vyvrácení domněnky, že aktivní učení lépe funguje pro adaptační sadu, která je větší a jednodušší tzn. původní data jsou do značné míry podobné datům adaptačním.

## 6.3 Přístupy učení

V této podkapitole se zabývám experimenty s přístupy učení neuronových sítí, tzn. jaké parametry algoritmů aktivního učení zvolit, aby bylo dosaženo co nejlepších výsledků, respektive co nejlepší generalizace. Nejprve experimentuji s učením neuronových sítí při různých poměrech původních a adaptačních dat. Dále se věnuji účinnosti křížové validace při různém počtu trénovaných vrstev. Jako původní a adaptační data jsem zvolil datové sady popsané pro první scénář, tzn. normální písmo a polské historické texty.

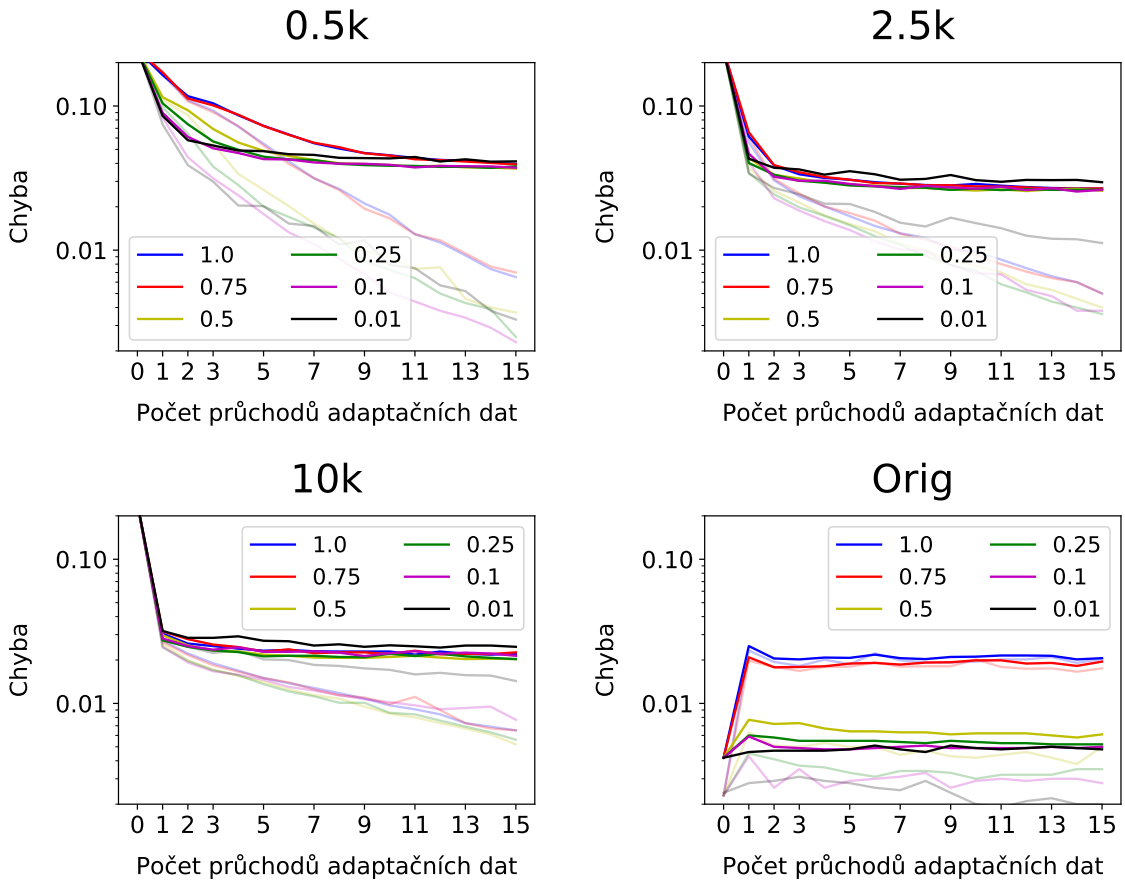
Pro ujasnění definuji pět datových množin, které dále zmiňuji v jednotlivých experimentech: původní trénovací sada, původní testovací sada, adaptační trénovací sada, adaptační validační sada a adaptační testovací sada. Původní sady jsou sady, na kterých byla natrénována neuronová síť, která bude adaptována. Adaptační trénovací sadu využívám pro adaptaci na danou množinu a spolu s adaptační validační sadou pro kontrolu přetrénování v rámci Algoritmu 1. Adaptační testovací sada je nezávislá testovací sada, tzn. na rozdíl od validační sady se žádným způsobem nepodílí na trénování sítě. Adaptační validační sada se na trénování sítě podílí pokud se síť trénuje pomocí křížové validace.

**Regularizace pomocí míchání původních a adaptačních dat.** Důvodem míchání původních a adaptačních trénovacích dat v rámci algoritmů aktivního učení je zamezení přetrénování na malé množině adaptačních dat tím, že se tato data při trénování budou objevovat jen velmi zřídka. Dalším přínosem je zachování vysoké úspěšnosti sítě na původních datech.

Aby bylo možné porovnávat různé poměry původních a adaptačních dat, je kritérium pro porovnání počet průchodů adaptačními trénovacími daty, tzn. při trénování musí být neuronovou sítí viděna všechna adaptační trénovací data např. desetkrát. Tento přístup má za následek to, že s klesajícím poměrem adaptačních a původních dat stoupá počet iterací nutných k tomu, aby byla viděna všechna adaptační data. Např. pokud se trénuje na množině dat o velikosti 32000 s poměrem 0.01 a velikost datového vzorku (*batch*) je 32, počet iterací nutný k průchodu všech adaptačních dat je 100k (při poměru 1.0 1k).

Výsledky experimentů s různými poměry původních a adaptačních dat jsou na Obrázku 6.6. Osa představující chybu je v logaritmickém měřítku. První tři grafy zobrazují chybu adaptované sítě na nezávislé adaptační testovací sadě (syté linky) a chybu na adaptační trénovací sadě (průhledné linky) v závislosti na počtu průchodů adaptačního trénovacího datasetu. *n*k v názvu grafu značí počet řádků v adaptační trénovací množině. Poslední graf zobrazuje závislost chyby sítě na původní trénovací (průhledné linky) a testovací (syté linky) sadě a na počtu průchodů adaptační trénovací sady. Síť byla trénována s *dropout* 2 % a učící konstantou o velikosti 0.0003.

Je patrné, že i pro relativně malé poměry stále dochází k výraznému přetrénování. Výjimku tvoří až nejmenší poměr 0.01 ve třetím grafu, kde adaptační trénovací množina obsahuje 10000 řádků. Tento výsledek je poměrně překvapivý. Síť má velkou paměť a dokáže si zapamatovat řádky s adaptační množiny i v případě, že se vyskytují v trénovací množině velmi zřídka. Např. pro 2500 řádků v adaptační trénovací sadě a poměr 0.01 to znamená, že síť vidí ten stejný řádek jednou za 250000 řádků ostatních a i přes to se stále dokáže znatelně přetrénovat. Dalším zajímavým výsledkem je skutečnost, že dva největší poměry

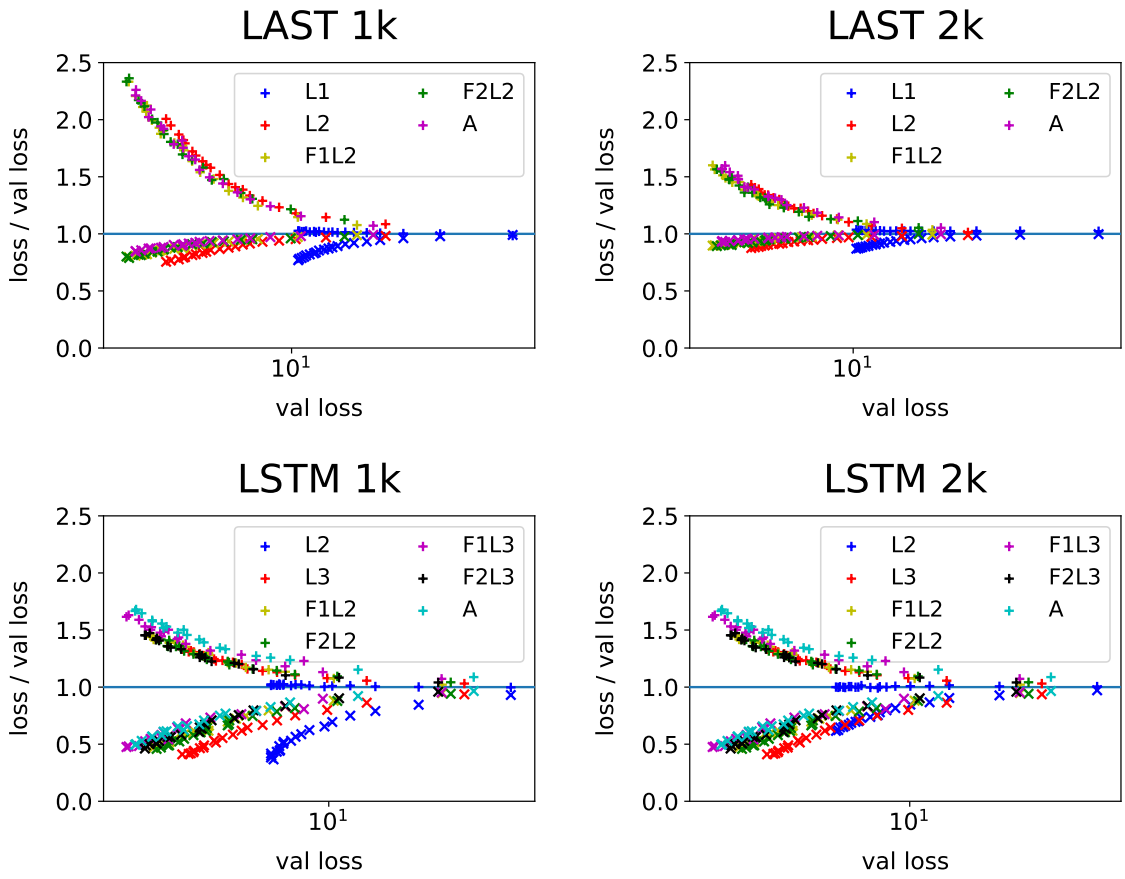



Obrázek 6.6: Výsledky experimentů s různými poměry původních a adaptačních dat. Osa představující chybu je v logaritmickém měřítku. První tři grafy zobrazují chybu adaptované sítě na nezávislé adaptační testovací sadě (syté linky) a chybu na adaptační trénovací sadě (průhledné linky) v závislosti na počtu průchodů adaptačního trénovacího datasetu. $n$k v názvu grafu značí počet řádků v adaptační trénovací množině. Poslední graf zobrazuje závislost chyby sítě na původní trénovací (průhledné linky) a testovací (syté linky) sadě na počtu průchodů adaptační trénovací sady.

dosahují nejmenšího přetrénování v rámci grafu o počtu řádků 500. Tato skutečnost je pravděpodobně spojena s tím, že pro velké poměry a malý počet řádků je celkový počet iterací sítě poměrně malý a tudíž síť nemá dostatek času na přetrénování. Síť se přeučuje tím méně, čím větší je adaptační dataset.

Celkově očekávaný výsledek představuje poslední graf, kde při menších poměrech původních a adaptačních dat nedochází ke zvětšení chyby na původním datasetu. Naopak pro velké poměry neuronová síť zapomíná schopnost dobře rozpoznávat data z původní datové množiny.

Regularizace přetrénování pomocí míchání původní a adaptační sady tedy funguje jenom pro dostatečně velkou adaptační sadu. Pravděpodobně bude fungovat i pro sady menší, ale poměr adaptačních a původních dat bude muset být tak malý, že učení prováděné tímto způsobem bude velmi zdlouhavé. Z hlediska aktivního učení může být výhodné mít k dispozici síť, která co nejlépe generalizuje. Z tohoto důvodu v dalších experimentech




Obrázek 6.7: Grafy udávají vztah mezi validační, testovací (znaky plus) a trénovací chybou (znaky krát). Osa $x$ udává validační chybu, osa $y$ pak poměr testovací či trénovací chyby ku validační chybě. Název grafu udává typ namnožení posledních vrstev a počet řádků, které byly použity pro trénování. Jednotlivé barvy udávají, které vrstvy sítě byly trénovány. Význam značení je následující: L$n$ značí počet posledních vrstev, které jsou trénovány a F$n$ značí počet prvních vrstev, které jsou trénovány, a značí trénování všech vrstev sítě.

používám nižší poměry jako 0.01, jelikož tyto poměry zaručují kvalitní fungování sítě na původní datové sadě.

**Účinnost křížové validace v závislosti na počtu trénovaných vrstev.** Jak jsem již uvedl v podkapitole 4.3, trénuji mimo sítě standardní i sítě s využitím křížové validace, a to tak, že namnožím poslední vrstvy neuronové sítě. Přitom můžu namnožit jenom poslední konvoluční vrstvu nebo poslední konvoluční vrstvu spolu s rekurentní vrstvou LSTM.

Při křížové validaci je žádoucí aby validační chyba byla co nejblíže nezávislé testovací chybě. Validační chyba neuronové sítě s namnoženými posledními vrstvami je průměrná chyba jednotlivých větví na jejich validačních datech. Pokud se trénují pouze vrstvy, které byly namnoženy, tzn. neexistuje vrstva, která by byla sdílená pro všechny větve, je validační a testovací chyba totožná (až na malé odchylky způsobené rozdílností v datech). Otázkou je co nastane pokud se budou trénovat i vrstvy které jsou větvemi sdíleny, případně úplně všechny vrstvy sítě.

Grafy na Obrázku 6.7 udávají vztah mezi validační, testovací (znaky plus) a trénovací chybou (znaky krát). Osa $x$ udává validační chybu (jedná se o chybu *loss* nikoliv chybu znakovou), osa $y$ pak poměr testovací či trénovací chyby ku validační chybě. Název grafu udává typ namnožení posledních vrstev a počet řádků, které byly použity pro trénování. Typ namnožení může být buď pouze poslední konvoluční vrstva (LAST) nebo poslední konvoluční vrstva spolu s vrstvou rekurentní (LSTM). Síť se trénovala na daném počtu adaptačních řádků po 20000 iterací (výsledné chyby byly vypisovány každých 1000 iterací). Jednotlivé barvy udávají, které vrstvy sítě byly trénovány. Význam značení je následující: L$n$ značí počet posledních vrstev, které jsou trénovány a F$n$ značí počet prvních vrstev, které jsou trénovány, a značí trénování všech vrstev sítě. Namnožené poslední vrstvy se počítají jako vrstva jedna, tzn. pro LSTM grafy L2 znamená trénování všech osmi posledních vrstev (při čtyřech větvích).

Ideálně by všechny poměry chyb měly být jedna, tzn. značky plus a krát by ležely na modré přímce. Jak již ukázal předchozí experiment, neuronové sítě se na menším počtu dat značně přeučují. Graficky je tato skutečnost patrná na značkách krát, které leží pod modrou přímkou. Zajímavější skutečnost prezentují značky plus. Již při sdílení jedné vrstvy se tyto značky odchylují od modré přímky směrem vzhůru, tzn. L2 při namnožení pouze poslední vrstvy a L3 při namnožení poslední konvoluční vrstvy a vrstvy rekurentní. Toto odchýlení znamená, že se při sdílení vrstev nedá řídit podle validační chyby. Důvodem odchýlení je pravděpodobně to, že síť si prostřednictvím sdílených vrstev předává znalosti o jednotlivých řádcích, tzn. větve se částečně učí i na svých validačních datech. Trénování více vrstev má také dopad na celkovou chybu, která se při zvětšování počtu trénovaných vrstev zmenšuje. Na druhou stranu s rostoucím počtem trénovaných vrstev roste přetrénování, které je nejvíce patrné při trénování všech vrstev. Křížová validace je tím méně efektivní, čím více vrstev je sdíleno jednotlivými větvemi.

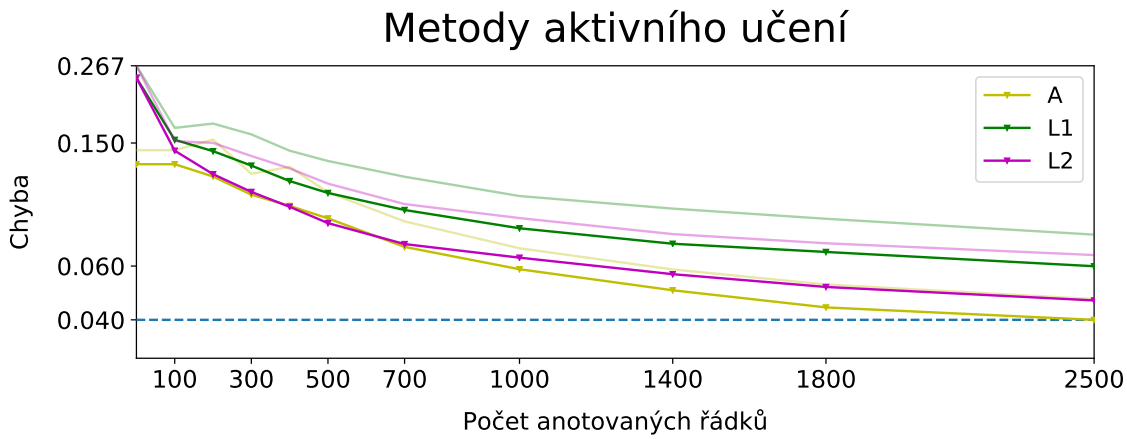
Křížová validace má tedy význam pouze v případě, že nejsou sdíleny žádné větve, tzn. L1 pro namnožení pouze poslední konvoluční vrstvy a L2 pro namnožení poslední konvoluční vrstvy a vrstvy LSTM.

## 6.4 Aktivní učení

V této podkapitole srovnávám metody aktivního učení, které byly popsány v podkapitole 4.2. Tyto metody jsou převážně založeny na pravděpodobnostní matici vrstvy CTC. Výjimku tvoří metoda založená na výstupních přepisech jednotlivých větví sítě.

Pro srovnání metod využívám dva popsané algoritmy aktivního učení. Nejdříve je proveden algoritmus postupného učení neuronové sítě, který postupně ukládá vytvořené datasety, tzn. výběry dat podle metody aktivního učení. Poté je nad vybranými datasety proveden algoritmus druhý, který na těchto datasetech síť přetrénuje. Tento přístup je zvolen z toho důvodu, že úspěšnost v průběhu algoritmu postupného učení neuronové sítě ještě nemusí vypovídat o celkové kvalitě vybraných dat. Proto je síť na vybraných datech přeučena a až poté je rozhodnuto o účinnosti vybraných dat.

**Účinnost výběru adaptačních dat.** První algoritmus tedy vybírá data na základě metody aktivního učení. Kromě metody aktivního učení je výběr závislý i na počtu vrstev, které se v průběhu algoritmu trénují, což zahrnuje i využití/nevyužití křížové validace. Před srovnáním samotných metod aktivního učení se věnuji srovnání účinnosti vybraných datasetů na základě vrstev, které byly při výběru učeny. Účinnost výběru je tím větší, čím



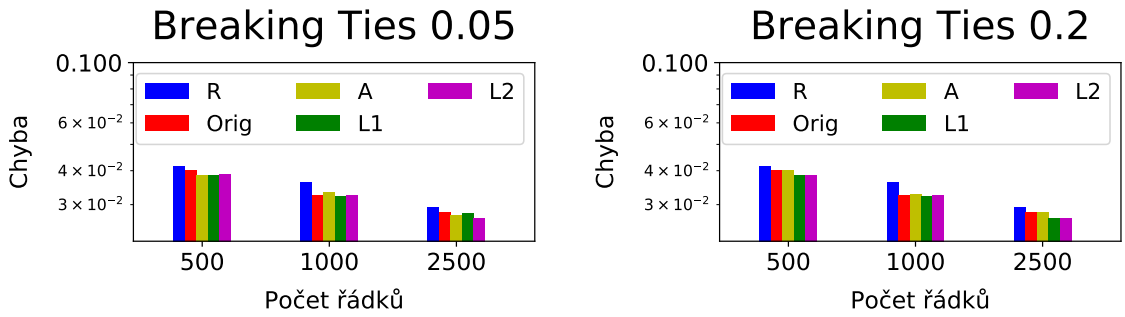

Obrázek 6.8: Průběh prvního algoritmu pro trénování všech vrstev bez křížové validace (A), jenom poslední vrstvy s využitím křížové validace (L1) a posledních dvou vrstev s využitím křížové validace (L2). Počet aktuálně anotovaných dat je na ose $x$, osa $y$ je v logaritmickém měřítku. Metoda aktivního učení je BT (*Breaking Ties*).

větší je úspěšnost neuronové sítě, která je na tomto výběru trénována/přeučena. Cílem je rozhodnout, zda prosté trénování celé sítě nedosahuje lepších výsledků než trénování sítě s více větvemi a křížovou validací.

Obrázek 6.8 představuje průběh prvního algoritmu pro trénování všech vrstev bez křížové validace (A), jenom poslední vrstvy s využitím křížové validace (L1) a posledních dvou vrstev s využitím křížové validace (L2). Počet aktuálně anotovaných dat je na ose $x$, osa $y$ je v logaritmickém měřítku. Metoda aktivního učení je BT (*Breaking Ties*). Práh přetrénování byl nastaven na 5 %. Kontrola přetrénování se prováděla vždy po jednom vidění adaptačního datasetu. Nejméně úspěšný průběh má přístup L1 následovaný přístupem L2. Nejlepší průběh má trénování všech vrstev sítě. Samotný průběh prvního algoritmu ovšem nezaručuje, že přístupy s lepším průběhem budou mít také lepší respektive účinnější výběr dat.

Obrázek 6.9 představuje výsledky přeučení druhého algoritmu na určitém počtu anotovaných řádků, které byly vybrány náhodně (R), na základě původní sítě bez adaptace (Orig), na základě trénování všech vrstev sítě (A) a na základě trénování posledních vrstev s využitím křížové validace (L1 a L2). Graf vlevo je pro práh přetrénování Algoritmu 1 5 %, graf vpravo pro 20 %. Metoda aktivního učení byla BT (*Breaking Ties*). Přeučení se provádělo pro 500, 1000 a 2500 řádků po 20k, 30k a 50k iterací s poměrem adaptačních a původních dat 0.02. Nejhorší přístup k výběru je výběr náhodný a výběr zakládající se na původní ještě neadaptované síti. Naopak nejlepší jsou výběry na základě adaptovaných sítí s tím, že nejlépe se chová výběr založený na křížové validaci při namnožení poslední konvoluční a rekurentní vrstvy.

Patrný je rozdíl zejména mezi přístupy využívajícími pro výběr adaptačních dat neuronovou síť a přístupem náhodným. Rozdíly mezi přístupy, které využívají neuronovou síť jsou velmi malé. Přístup využívající křížovou validaci při namnožení posledních dvou vrstev je však průměrně nejlepší a nabízí mimo samotnou úspěšnost i jiné výhody. Zejména se jedná o možnost využití čtyř výstupů pro aktivní učení, tzn. rozhodnutí o vhodnosti řádků pro anotaci se provádí na základě všech čtyřech výstupů. Další výhodou je rychlost adaptace, tzn. učení pouze posledních vrstev je rychlejší než učení celé neuronové sítě.




Obrázek 6.9: Výsledky přeučení druhého algoritmu na určitém počtu anotovaných řádků, které byly vybrány náhodně (R), na základě původní sítě bez adaptace (Orig), na základě trénování všech vrstev sítě (A) a na základě trénování posledních vrstev s využitím křížové validace (L1 a L2). Graf vlevo je pro limit přetrénování algoritmu 1 5 %, graf vpravo pro 20 %.

| | Orig TRN | Orig TST | Adapt TRN | Adapt TST |
|---|---|---|---|---|
| Orig | 0.0025 | 0.0042 | 0.2447 | 0.2447 |
| IMPACT | 0.0043 | 0.0052 | 0.0158 | 0.0208 |

Tabulka 6.1: Chybovost původní neadaptované sítě (Orig) a sítě natrénované na celém datasetu IMPACT na původních (Orig) a adaptačních (Adap) datech. TRN značí trénovací sadu a TST sadu testovací. Původními daty jsou dokumenty s typem písma *normal* a *calligraphy* a adaptační data jsou historické polské texty tzn. první scénář adaptace PLH.

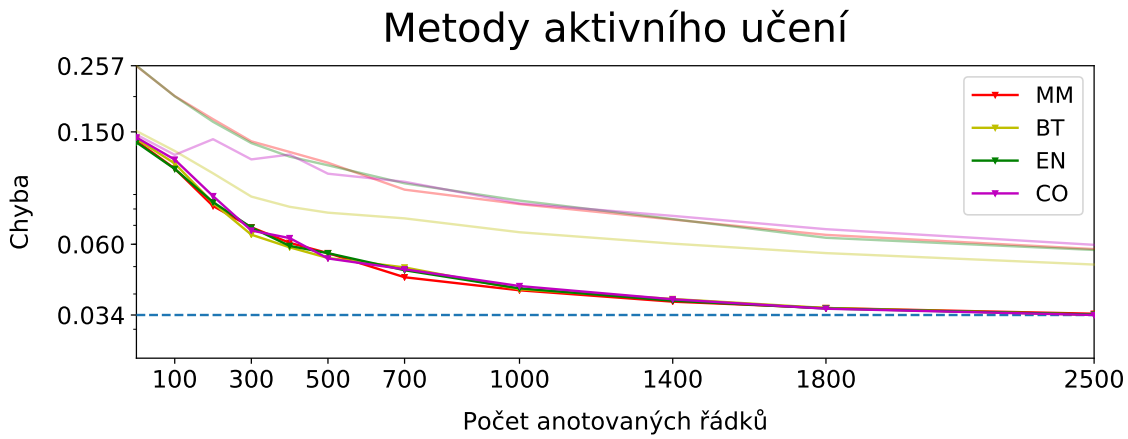
**Metody aktivního učení.** Metody aktivního učení porovnávám na základě provedení prvního algoritmu a následném provedení algoritmu druhého. Jedná se o stejný postup jako při srovnání různých přístupů pro výběr adaptačních dat, respektive srovnání účinnosti adaptačních dat. Jelikož nepředpokládám vliv parametrů jako jsou učící konstanta a velikost *dropout* na výsledné výběry řádků pro adaptaci, zkoumám dopad míry přetrénování sítí, tzn. velikost prahu pro přetrénování prvního algoritmu na tento výběr.

Před samotnými experimenty s adaptací pomocí aktivního učení uvádím tabulky chybovosti původních sítí bez adaptace (Orig) a sítě natrénované na celém datasetu IMPACT. Chybovost uvádím pro trénovací a testovací dataset jak původní (Orig), tak adaptační (Adapt). Tabulka 6.1 udává chybovost pro první scénář (PLH), tzn. původní dokumenty mají styl písma *normal* a *calligraphy* a adaptační dokumenty jsou historické polské texty. Tabulka 6.2 udává chybovost pro scénář druhý (NPN), tzn. původní dokumenty jsou knihy se stylem písma *normal* a *calligraphy* a adaptační dokumenty jsou noviny se stylem písma *normal*. Původní síť pro scénář PLH má značnou chybu na adaptačních datech. Tato skutečnost plyne ze značné odlišnosti původních a adaptačních dat. Naopak původní síť pro scénář NPN má na adaptačních datech chybu mnohem menší. Opět tato skutečnost plyne z toho, jak jsou si původní a adaptační data podobné (mezi řádky z knih a novin není velký rozdíl).

Nejdůležitějšími údaji z těchto tabulek jsou chybovosti původních neadaptovaných sítí na adaptačních datech, tzn. chyba, kterou se adaptací snažím co nejvíce snížit a chybovost sítě natrénované na celém datasetu IMPACT na adaptačních datech, tzn. ideální chyba, které se snažím adaptací co nejvíce přiblížit.

| | Orig TRN | Orig TST | Adapt TRN | Adapt TST |
|---|---|---|---|---|
| Orig | 0.0038 | 0.0052 | 0.0268 | 0.0268 |
| IMPACT | 0.0044 | 0.0050 | 0.0058 | 0.0074 |

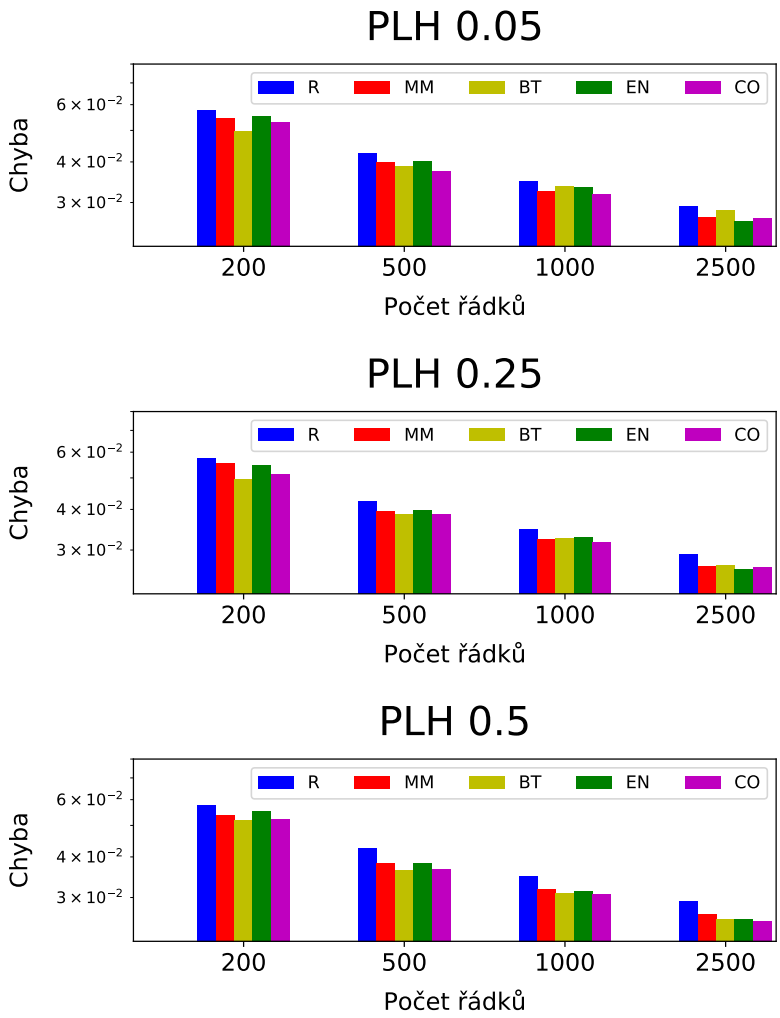
Tabulka 6.2: Chybovost původní neadaptované sítě (Orig) a sítě natrénované na celém datasetu IMPACT na původních (Orig) a adaptačních (Adap) datech. TRN značí trénovací sadu a TST sadu testovací. Původními daty jsou knihy s typem písma *normal* a *calligraphy* a adaptační data jsou noviny s typem písma *normal* tzn. druhý scénář adaptace NPN.




Obrázek 6.10: Průběh prvního algoritmu pro metody aktivního učení MM (Mean Max), BT (Breaking Ties), EN (Entropy) a CO (Committee). Počet aktuálně anotovaných dat je na ose $x$, osa $y$ je v logaritmickém měřítku.

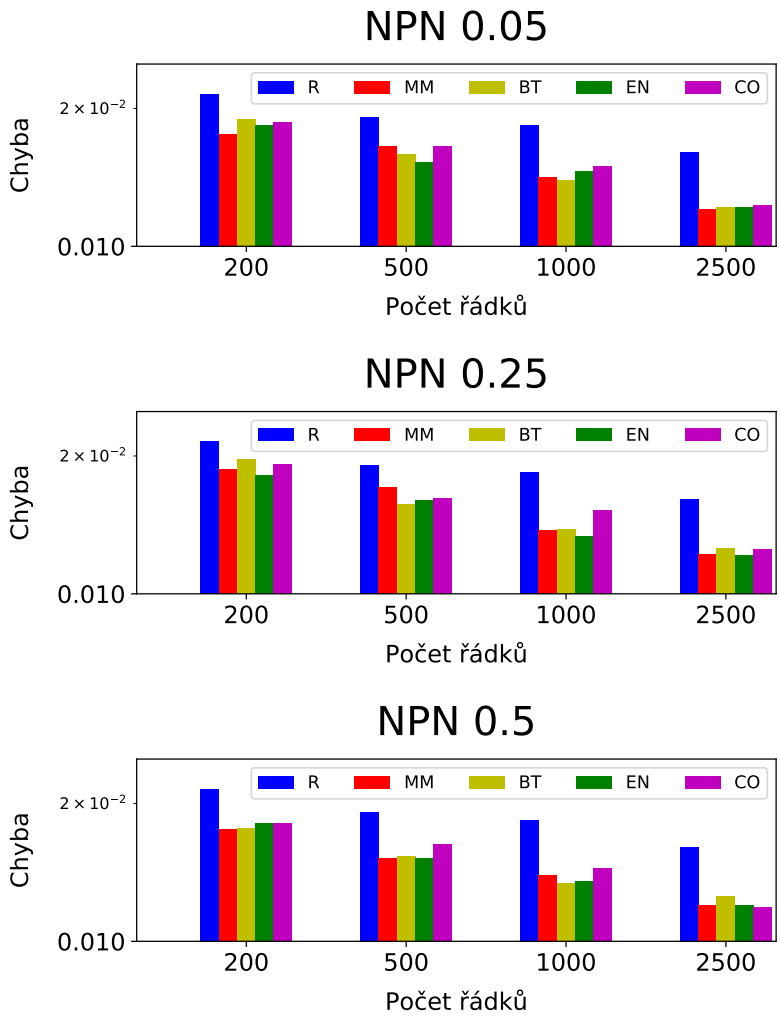
Průběh prvního algoritmu pro metody aktivního učení MM (Mean Max), BT (Breaking Ties), EN (Entropy) a CO (Committee) je na Obrázku 6.10. Původní a adaptační data odpovídají prvnímu scénáři, tzn. PLH. Počet anotovaných řádků je postupně zvětšován. Syté linky představují chybu na nezávislé testovací sadě, linky průhledné chybu na trénovací sadě. Je patrné, že všechny metody fungují prakticky totožně. Chyba na trénovacích datech je u všech metod znatelně horší. To je způsobeno tím, že všechny přístupy nějakým způsobem vybírají těžké řádky, tzn. řádky, které mají velkou chybovost. Jak již jsem zmínil průběh samotného prvního algoritmu není vzhledem k porovnání metod/přístupů zcela vypovídající, neboť neuronová síť není dostatečně trénována na daném počtu řádků.

Výsledky přetrénování na vybraných sadách o velikostech 200, 500, 1000 a 2500 řádků pro scénář PLH jsou na Obrázku 6.11. Modré sloupečky přestavují náhodný výběr řádků. Síť byla trénována na daném množství řádků po 10k, 20k, 30k a 50k iterací s poměrem adaptačních a původních dat 0.02. Míra přetrénování respektive velikost prahu přetrénování pro první algoritmus je uvedena v názvech jednotlivých grafů. Je patrné, že míra přetrénování při výběru adaptačních řádků nehraje velkou roli. To znamená, že i přetrénovaná síť dokáže vybírat kvalitní řádky z hlediska aktivního učení. Výsledky stejného experimentu pro druhý scénář NPN jsou na Obrázku 6.12. Je patrný znatelnější rozdíl mezi náhodným výběrem a výběrem podle metod aktivního učení. Metody aktivního učení jsou při všech experimentech prakticky totožné, tzn. žádná metoda nemá výrazně lepší výsledky než metody ostatní.




Obrázek 6.11: Výsledky přetrénování na vybraných sadách o velikostech 200, 500, 1000 a 2500 řádků pro scénář PLH. Modré sloupečky přestavují náhodný výběr řádků, ostatní pak výběr příslušnou metodu aktivního učení. Velikost prahu přetrénování pro první algoritmus je uvedena v názvech jednotlivých grafů.

Pro srovnání úspěšnosti adaptace s metodami aktivního učení uvádím chybovost v Tabulce 6.3 pro oba scénáře s počtem řádků 2500 a přetrénovacím prahem prvního algoritmu 0.5. Tato chyba byla spočtena jako průměrná chyba posledních 3000 iterací při přeučování sítě pomocí druhého algoritmu. Pro první scénář PLH došlo při použití aktivního učení ke zmenšení chyby oproti náhodnému výběru o 13 %. Pro druhý scénář NPN pak o 26 %. Dosažené chyby se také přibližují (úměrně k počtu využitých řádků) chybám sítě natrénované na celém datasetu IMPACT (sloupec IMPACT). Tato síť měla při trénování k dispozici všechna adaptační data, tzn. v případě prvního scénáře měla čtyřikrát více dat a v případě druhého scénáře asi třicetkrát více dat.




Obrázek 6.12: Výsledky přetrénování na vybraných sadách o velikostech 200, 500, 1000 a 2500 řádků pro scénář NPN. Modré sloupečky přestavují náhodný výběr řádků, ostatní pak výběr příslušnou metodu aktivního učení. Velikost prahu přetrénování pro první algoritmus je uvedena v názvech jednotlivých grafů.

## 6.5 Shrnutí

Neuronové sítě obsahující konvoluční a rekurentní vrstvy jsou schopny kvalitně rozpoznávat text. Jedná se o řádky textů, které jsou mnohdy do značné míry poškozené, mají různou barvu a texturu pozadí a mnoho odlišných fontů. Čistě konvoluční neuronové sítě nedosahují tak kvalitních výsledků, jako sítě které využívají rekurentní vrstvy. Rekurentní vrstvy naopak přináší mírné zpoždění díky sekvenčnímu vyhodnocování rekurentních vrstev. Chyba čistě konvolučních sítí je 1.4 % a chyba sítí rekurentních je 0.5 %.

Aktivní učení probíhá se dvěma datovými sadami. První sada je původní, tzn. data na kterých byla natrénována neuronová síť, která se bude adaptovat a druhá je adaptační, tzn. data, na která se neuronová síť bude adaptovat. Pro kontrolu přetrénování jsem zvolil míchání původních a adaptačních dat s tím, že adaptační data se v trénovací množině objevují jen velmi zřídka. Tato regularizace se ukázala jako nefunkční, respektive poměr

| | RA | MM | BT | EN | CO | IMPACT |
|---|---|---|---|---|---|---|
| PLH | 0.0292 | 0.0265 | 0.0256 | 0.0257 | **0.0253** | 0.0208 |
| NPN | 0.0160 | 0.0119 | 0.0125 | 0.0120 | **0.0118** | 0.0074 |

Tabulka 6.3: Chybovost metod aktivního učení pro scénáře PLH a NPN. RA náhodný výběr, MM (Mean Ma), BT (Breaking Ties), EN (Entropy), CO (Committee) a IMPACT je chyba sítě natrénované na celém datasetu IMPACT.

adaptačních dat by musel být neúnosně malý, aby nedocházelo k přetrénování. Na druhou stranu si síť díky míchání dat uchovala schopnost velmi dobře rozpoznávat data z původní datové množiny s tím, že úspěšnost na adaptační množině to nijak negativně neovlivnilo.

Účinnost, respektive použitelnost efektivní křížové validace, která je založena na namnožení posledních vrstev neuronové sítě klesá tím víc, čím více vrstev mají jednotlivé větve společných. Negativní dopad společných vrstev je patrný již při jedné sdílené vrstvě. Efektivní křížovou validaci tak má smysl používat pouze pokud větve nesdílejí žádné vrstvy. Rovněž je patrný menší dopad sdílení vrstev pokud je namnoženo více posledních vrstev sítě.

Adaptační data jsou tím účinější, čím úspěšnější neuronovou síť na nich lze natrénovat. Výběr adaptačních dat do adaptační trénovací množiny lze provádět za pomoci neuronové sítě. Náhodný výběr a výběr na základě původní neadaptované sítě je nejméně kvalitní. Naopak výběry prováděné na základě adaptované sítě jsou účinnější. Nejvýhodnější se přitom jeví využít efektivní křížové validace, která má namnožené dvě poslední vrstvy, a to vrstvu výstupní konvoluční a vrstvu rekurentní. Výhoda oproti klasické síti spočívá ve více rozdílných výstupech, které lze využívat pro aktivní učení, respektive výběr dat.

Metody aktivního učení dosahují zlepšení vůči prostému náhodnému výběru dat pro adaptaci. Výsledky samotných metod se prakticky neliší. Všechny metody jsou nějakým způsobem založeny na výběru potenciálně chybových řádků. Z tohoto důvodu je pak chyba na trénovacích datech značně vyšší než na datech validačních nebo testovacích. Představuji dva scénáře adaptace. Prvním je adaptace neuronové sítě naučená na dokumentech se stylem písma podobnému dnešnímu tištěnému písmu na dokumenty obsahující historické polské písmo. Zmenšení chyby díky aktivnímu učení je 13 %. Druhým scénářem je adaptace neuronové sítě naučené na knihách, jejichž styl písma je opět podobný tomu soudobému, na noviny o stejném stylu písma. Zmenšení chyby díky aktivnímu učení je 26 %.

Otázkou je, co je na datových sadách jiného, že druhý scénář dosáhne s využitím aktivního učení dvojnásobného zlepšení. Důležitou roli hrají pravděpodobně následující dva faktory. Prvním je velikost množiny, ze které jsou vybírány řádky pro adaptaci. Zatímco v případě prvního scénáře má tato množina velikost asi 10k řádků, v případě druhém je tato množina více než osmkrát větší. Čím je množina větší, tím je pravděpodobnější, že náhodný výběr dat bude nevhodný. Naopak čím je množina menší, tím je pravděpodobnější, že výběr dat bude bližší výběru ideálnímu. Druhým faktorem je obtížnost samotných dat. V prvním scénáři jsou adaptační data vůči původním velmi složitá. Naopak ve scénáři druhém jsou adaptační data vůči datům původním značně jednoduchá. Čím jsou data obtížnější, tím je větší množina potenciálně chybových, respektive informativních řádků. To znamená, že aktivní učení je méně účinné, jelikož volba kteréhokoli řádku je pro síť prospěšná. Naopak čím jsou data jednodušší, tím je důležitější vybrat právě ta, na kterých síť nejvíce chybuje, neboť většina dat je tak jednoduchých, že se na nich síť nic nového nenaučí. Aktivní učení je tedy tím účinnější, čím je datová sada větší a jednodušší.

# Kapitola 7

# Závěr

Cílem této práce bylo navrhnout metody aktivního učení a provést experimenty nad datovou sadou historických dokumentů. Aktivní učení provádím pomocí dvou algoritmů, které byly navrženy tak, aby co nejvíce automatizovaly proces aktivního učení. Jelikož se učení provádí za pomoci velmi malého množství dat, využívám při trénování efektivní křížovou validaci. Tři metody aktivního učení jsou založeny na pravděpodobnostní matici výstupní vrstvy CTC. Čtvrtá, poslední metoda, je založena na Levenštejnově vzdálenosti přepisů jednotlivých větví sítě.

Dataset IMPACT představuje rozsáhlou sadu historických dokumentů. Na základě chybovosti neuronové sítě a Levenštejnovi vzdálenosti přepisů sítě a anotací filtruji podezřelé řádky tak, aby negativně neovlivňovaly proces učení sítě. V rámci datasetu IMPACT jsem vytvořil mapování jednotlivých dokumentů a to na styl písma, typ dokumentu, jazyk a kvalitu dokumentu. Mapování lze využít pro experimenty s adaptací pomocí aktivního učení.

Čistě konvoluční neuronové sítě dosahují chybovosti 1.4 %, sítě využívající obousměrnou vrstvu LSTM dosahují chybovosti 0.5 %. Mícháním původních a adaptačních dat je dosaženo zachování schopnosti neuronové sítě dobře rozpoznávat řádky z původní datové množiny. Neuronové sítě mají dlouhodobou paměť, neboť i při malém poměru adaptačních a původních data (0.01) dochází ke znatelnému přetrénování. Efektivní křížová validace má význam pouze tehdy, pokud jednotlivé větve sítě nesdílý žádnou vrstvu neuronové sítě. Už při sdílení jediné vrstvy dochází k výměně informace mezi větvemi. Při adaptaci sítě učené na textech podobných soudobému tištěnému textu na historické polské texty jsem díky aktivnímu učení dosáhl zlepšení o 13 % oproti náhodnému výběru dat. Při adaptaci sítě učené na knihách s textem podobnému soudobému tištěnému textu na novinové texty jsem s pomocí aktivního učení dosáhl zlepšení 23 %. Aktivní učení je tím účinnější, čím je adaptační datová sada rozsáhlejší a jednodušší.

Další vývoj z hlediska filtrování chybových řádků by se mohl zaměřit na metodu, která by neodstraňovala těžké, ale správné přepisy. Tato metoda by byla pravděpodobně založena na pravděpodobnostní matici výstupní vrstvy CTC. Jinou možností by bylo natrénovat neuronovou síť pro klasifikaci podezřelých řádků, případně kombinace obojího. Co se týče samotného aktivního učení, je zde prostor pro implementaci metod, které nejsou založené na chybovosti řádků, ale na jejich reprezentativnosti vzhledem k problému který se síť učí. Dále by bylo vhodné experimentovat s více scénáři adaptace, které mají odlišná jak data, tak samotný průběh aktivního učení. Obecným a důležitým problémem aktivního učení je cena anotace, kterou v rámci této práce neuvažuji.

# Literatura


[1] *Recurrent neural network*. [Online; navštíveno 29.12.2018].
URL https://en.wikipedia.org/wiki/Recurrent_neural_network

[2] Ana Čamba, F. H. S. F. R. S., Melanie Gau: Multispectral Imaging, Image Enhancement, and Automated Writer Identification in Historical Manuscripts. 2014.

[3] Bengio, Y.; Simard, P.; Frasconi, P.: Learning Long-term Dependencies with Gradient Descent is Difficult. *Trans. Neur. Netw.*, ročník 5, č. 2, Březen 1994: s. 157–166, ISSN 1045-9227.

[4] Breuel, T. M.; Ul-Hasan, A.; Al-Azawi, M. A.; aj.: High-Performance OCR for Printed English and Fraktur Using LSTM Networks. In *2013 12th International Conference on Document Analysis and Recognition*, Aug 2013, ISSN 1520-5363, s. 683–687, doi:10.1109/ICDAR.2013.140.

[5] Chaudhuri, A.; Mandaviya, K.; Badelia, P.; aj.: *Optical Character Recognition Systems for Different Languages with Soft Computing*, *Studies in Fuzziness and Soft Computing*, ročník 352. Springer, 2017, ISBN 978-3-319-50251-9.

[6] Chen, K.; Seuret, M.; Hennebert, J.; aj.: Convolutional Neural Networks for Page Segmentation of Historical Document Images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, ročník 01, Nov 2017, ISSN 2379-2140, s. 965–970.

[7] Clausner, C.; Pletschacher, S.; Antonacopoulos, A.: Aletheia - An Advanced Document Layout and Text Ground-Truthing System for Production Environments. In *2011 International Conference on Document Analysis and Recognition*, Sep. 2011, ISSN 2379-2140, s. 48–52, doi:10.1109/ICDAR.2011.19.

[8] Dagan, I.; Engelson, S. P.: Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, Elsevier, 1995, s. 150–157.

[9] Ducoffe, M.; Precioso, F.: QBDC: Query by dropout committee for training deep supervised architecture. *CoRR*, ročník abs/1511.06412, 2015, 1511.06412.

[10] Fink, M.; Layer, T.; Mackenbrock, G.; aj.: Baseline Detection in Historical Documents using Convolutional U-Nets. *CoRR*, ročník abs/1810.09343, 2018, 1810.09343.

[11] Graves, A.: Generating Sequences With Recurrent Neural Networks. *CoRR*, ročník abs/1308.0850, 2013, 1308.0850.

[12] Graves, A.; Fernández, S.; Gomez, F.; aj.: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, New York, NY, USA: ACM, 2006, ISBN 1-59593-383-2, s. 369–376, doi:10.1145/1143844.1143891.

[13] Hochreiter, S.; Schmidhuber, J.: Long short-term memory. *Neural computation*, ročník 9, č. 8, 1997: s. 1735–1780.

[14] Hradiš, M.; Kotera, J.; Zemčík, P.; aj.: Convolutional Neural Networks for Direct Text Deblurring. In *Proceedings of BMVC 2015*, The British Machine Vision Association and Society for Pattern Recognition, 2015, ISBN 1-901725-53-7.

[15] Huang, J.; Child, R.; Rao, V.; aj.: Active Learning for Speech Recognition: the Power of Gradients. *CoRR*, ročník abs/1612.03226, 2016, 1612.03226.

[16] Kingma, D. P.; Ba, J.: Adam: A Method for Stochastic Optimization. *CoRR*, ročník abs/1412.6980, 2014, 1412.6980.

[17] LeCun, Y.; Bengio, Y.; Hinton, G.: Deep learning. *nature*, ročník 521, č. 7553, 2015: str. 436.

[18] Lee, C.; Osindero, S.: Recursive Recurrent Nets with Attention Modeling for OCR in the Wild. *CoRR*, ročník abs/1603.03101, 2016, 1603.03101.

[19] Lewis, D. D.; Gale, W. A.: A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, Springer-Verlag New York, Inc., 1994, s. 3–12.

[20] Nissim, N.; Moskovitch, R.; Rokach, L.; aj.: Detecting Unknown Computer Worm Activity via Support Vector Machines and Active Learning. *Pattern Anal. Appl.*, ročník 15, č. 4, Listopad 2012: s. 459–475, ISSN 1433-7541, doi:10.1007/s10044-012-0296-4.

[21] Olah, C.: *Neural Networks, Types, and Functional Programming*. Září 2015, [Online; navštíveno 29.12.2018].
URL http://colah.github.io/posts/2015-09-NN-Types-FP/

[22] Olah, C.: *Understanding LSTM Networks*. Srpen 2015, [Online; navštíveno 29.12.2018].
URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[23] Papadopoulos, C.; Pletschacher, S.; Clausner, C.; aj.: The IMPACT Dataset of Historical Document Images. In *Proceedings of the 2Nd International Workshop on Historical Document Imaging and Processing*, HIP '13, New York, NY, USA: ACM, 2013, ISBN 978-1-4503-2115-0, s. 123–130, doi:10.1145/2501115.2501130.

[24] Puigcerver, J.: Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, ročník 1, IEEE, 2017, s. 67–72.

[25] Rahhal, M. A.; Bazi, Y.; AlHichri, H.; aj.: Deep learning approach for active classification of electrocardiogram signals. *Information Sciences*, ročník 345, 2016: s. 340 – 354, ISSN 0020-0255, doi:https://doi.org/10.1016/j.ins.2016.01.082.

[26] Ray, A.; Rajeswar, S.; Chaudhury, S.: Text recognition using deep BLSTM networks. 01 2015, doi:10.1109/ICAPR.2015.7050699.

[27] Reul, C.; Springmann, U.; Wick, C.; aj.: Improving OCR Accuracy on Early Printed Books by combining Pretraining, Voting, and Active Learning. *CoRR*, ročník abs/1802.10038, 2018, 1802.10038.

[28] Roy, N.; McCallum, A.: Toward Optimal Active Learning through Sampling Estimation of Error Reduction. In *ICML*, 2001.

[29] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, ročník 115, 09 2014.

[30] Sener, O.; Savarese, S.: Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[31] Settles, B.: *Curious machines: Active learning with structured instances*. Dizertační práce, University of Wisconsin–Madison, 2008.

[32] Settles, B.; Craven, M.; Ray, S.: Multiple-Instance Active Learning. In *Advances in Neural Information Processing Systems 20*, editace J. C. Platt; D. Koller; Y. Singer; S. T. Roweis, Curran Associates, Inc., 2008, s. 1289–1296.

[33] Shuyo, N.: Language Detection Library for Java. 2010.
URL http://code.google.com/p/language-detection/

[34] Stark, F.; Hazırbaş, C.; Triebel, R.; aj.: CAPTCHA Recognition with Active Deep Learning. 09 2015.

[35] Wang, K.; Zhang, D.; Li, Y.; aj.: Cost-Effective Active Learning for Deep Image Classification. *CoRR*, ročník abs/1701.03551, 2017, 1701.03551.

# Příloha A

# Obsah SD

| | |
|---|---|
| latex/ | Zdrojový tvar diplomové práce |
| nets/ | Soubory představující sítě ve frameworku Tensorflow |
| datasets/ | Datasety pro aktivní učení |
| transcriptions/ | Ukázky přepisů řádků |
| pero/ | Skripty aktivního učení |
| DP.pdf | Diplomová práce ve formátu PDF |
| video.mp4 | Video prezentující diplomovou práci |