VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

REKURENTNÍ NEURONOVÉ SÍTĚ PRO KLASIFIKACI TEXTŮ
RECURRENT NEURAL NETWORK FOR TEXT CLASSIFICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Vojtěch Myška

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Lukáš Povoda

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Vojtěch Myška ***ID:*** 164611
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Rekurentní neuronové sítě pro klasifikaci textů

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte existující rámce hlubokého učení založené na rekurentních sítích, které se v současné době využívají na analýzu textů. Získané poznatky aplikujte při návrhu vlastní struktury hluboké neuronové sítě dle zadání vedoucího. Funkčnost vhodně prezentujte pomocí dosažených výsledků.

DOPORUČENÁ LITERATURA:

[1] ZHANG, Xiang a LECUN, Yann. Text understanding from scratch. arXiv preprint arXiv:1502.01710, 2015.

[2] ZHANG, Xiang a Junbo ZHAO. Character-level convolutional networks for text classification. Advances in neural information processing systems. 2015, 2015(28), 649-657.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Lukáš Povoda
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Diplomová práce se zabývá návrhem neuronových sítí pro klasifikaci pozitivních a negativních textů. Vývoj probíhal v programovacím jazyce Python. Návrh modelů hlubokých neuronových sítí byl proveden pomocí vysokoúrovňového API Keras využívající knihovnu pro numerické výpočty TensorFlow. Výpočetní operace byly provedeny pomocí GPU využívající CUDA architekturu. Výstupem práce je jazykově nezávislý model neuronových sítí umožňující klasifikaci textů na úrovni znaků. Vzorky byly úspěšně klasifikovány až v 93,64% případů. Trénovací a testovací data byla poskytnuta vícejazyčnou a Yelp databází. Simulace byly provedeny na 1 200 000 anglických, 12 000 českých, německých a španělských textů.

KLÍČOVÁ SLOVA

neuronové sítě, klasifikace textů, hluboké učení, rekurentní neuronové sítě, Keras, TensorFlow, CUDA, Kex

ABSTRACT

Thesis deals with the proposal of the neural networks for classification of positive and negative texts. Development took place in the Python programming language. Design of deep neural network models was performed using the Keras high-level API and the TensorFlow numerical computation library. The computations were performed using GPU with support of the CUDA architecture. The final outcome of the thesis is linguistically independent neural network model for classifying texts at character level reaching up to 93,64% accuracy. Training and testing data were provided by multilingual and Yelp databases. The simulations were performed on 1 200 000 English, 12 000 Czech, German and Spanish texts.

KEYWORDS

neural networks, texts classification, deep learning, recurrent neural networks, Keras, TensorFlow, CUDA, Kex

MYŠKA, Vojtěch. *Rekurentní neuronové sítě pro klasifikaci textů*. Brno, 2018, 71 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Lukáš Povoda

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Rekurentní neuronové sítě pro klasifikaci textů“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

Bc. Vojtěch Myška

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Lukáši Povodovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

Bc. Vojtěch Myška

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

Bc. Vojtěch Myška

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V informačním věku se setkáváme s velkým počtem nejrůznějších digitálních zařízení, počínaje od inteligentních kontejnerů přes chytré telefony až po superpočítače. Neustále rostoucí počet uživatelů využívajících tato zařízení stojí za trvale zvyšujícím se objemem dat. Jeho růst je exponenciální a proto vzniká zásadní otázka, jak takový objem data zpracovat. Byl dosažen bod, kdy již nelze uvažovat o zpracování dat lidskou silou.

Zásluhou vývoje nových technologií a výrobních procesů dochází k nárůstu výpočetního výkonu čipů při zachování energetické náročnosti. Tato skutečnost otevírá cestu k trvale se zdokonalujícímu strojovému zpracování dat, které se stává dostupnější. Mezi jedny z nejpoužívanějších výpočetních modelů v umělé inteligenci patří neuronové sítě. Nejsou zatím schopny řešit komplexní úlohy jako například člověk, ale při řešení jedné konkrétní funkce často dosahují daleko lepších výsledků. Zpracování přirozeného jazyka řešící například klasifikaci textu, kde je hlavním cílem třídit dokumenty, odstavce nebo jednotlivé věty do předem definovaných tříd, patří mezi typické úkoly podléhající strojovému zpracování dat. Zpracování přirozeného jazyka vzbuzuje velký zájem akademických obcí i komerční sféry. Klasifikací textu mohou být získané informace využity společnostmi k určení správné marketingové strategie, získání nových zákazníků nebo pro zlepšení služeb zákaznického servisu.

Tato diplomová práce se zabývá řešením problému klasifikace textů dle jeho pozitivního nebo negativního významu. Cílem je návrh modelů využívající konvoluční i rekurentní neuronové vrstvy pro účely klasifikace textů do pozitivní a negativní třídy, nikoliv na základě slov, ale pouze na základě znaků. Navržená neuronová síť představuje jazykově nezávislý klasifikátor textů. Trénovací a testovací proces je proveden na anglicky, česky, německy a španělsky psaných textech.

První kapitola se zabývá historií neuronových sítí, popisem základního modelu umělého neuronu a jeho aktivační funkce. Je zde věnován prostor některým vrstvám neuronových sítí, dělení neuronových sítí, základním technikám učení a LSTM vrstvám. Další stěžejní kapitola je věnována problematice klasifikace textu. Jsou popsány tradiční modely, modely shlukování slov a hlavně nedávno vydané práce prezentující výsledky klasifikace textu dosažené pomocí konvolučních neuronových sítí. Práce dokazuje možnost vytvoření jazykově nezávislého textového klasifikátoru. Třetí kapitola popisuje použité vývojové technologie. Příkladem je popis paralelního zpracování dat pomocí grafického procesoru, knihoven využívaných v rámci praktické řešení a hardwarové složení výpočetních zařízeních. Poslední kapitola se zabývá realizací praktické části této práce. Jde o popis postupu zpracování databází obsahujících textové vzorky, představení navržených modelů a diskuzi dosažených výsledků.

1 NEURONOVÉ SÍTĚ

Kapitola je rozdělena na tři části. První z nich je věnována úvodu do neuronových sítí. Druhá část se zabývá jejich dělením, typy učení a algoritmem Backpropagation. Poslední část popisuje LSTM sítě.

1.1 Úvod do umělých neuronových sítí

V rámci této podkapitoly je rozebrána historie neuronových sítí, poté popsán model formálního neuronu. Zbylý prostor je věnován některým vrstvám, aktivačním funkcím a algoritmu zpětné propagace chyby. Čerpáno z [3], [12], [21], [20], [21].

1.1.1 Historie neuronových sítí

Počátek historie je datován do roku 1943, kdy byl vytvořen první počítačový model umělého neuronu. Za jeho vznikem stojí dvojice amerických vědců, neurofyziolog Walter Pittse a matematik Warren McCulloch. V roce 1949 Donald Hebb publikoval Hebbio zákon, který v obecnější podobě představuje základ téměř všech procedur neurálního učení. Perceptron byl vynalezen americkým psychologem Frankem Rosenblattem v roce 1957. Jedná se o zjednodušený model umělého neuronu. Později také sestrojil Mark I Perceptrom, první počítač pracující na principu umělých neuronů.

Výzkum v oblasti umělé inteligence a neuronových sítí byl utlumen v průběhu dvou období nazývaných AI Winter, během kterých došlo k významnému poklesu financování zmíněných oborů.

Nenaplněné ambiciózní sliby vědců měly za následek znevážení oboru v průběhu 70. let 20. století a jeho první úpadek vývoje (AI Winter). Naštěstí se našli vědci pokračující ve výzkumu.

První konvoluční neuronová síť byla navržena v roce 1979 japonským vědcem Kunihiko Fukushimou. Obsahovala vícenásobné konvoluční a pooling[1] vrstvy. Fukushima stojí taktéž za vznikem neuronové sítě Neocognitron. Jedná se o hierarchickou vícevrstvou síť, umožňující rozpoznání vizuálních vzorů. Tento model navíc umožňuje změnu vah některých synaptických spojů. V roce 1986 počítačoví odborníci Williams, Hinton a psycholog Remelhart dokázali, že s pomocí algoritmu BackPropagation vzniknou ve skrytých vrstvách užitečné vnitřní reprezentace. Základy algoritmu položil Henry J. Kelley a sahají až do roku 1960. V roce 1989 byla provedena první praktická ukázka algoritmu BackPropagation v laboratořích Bell Labs francouzským vědcem Yannem LeCunem. Jednalo se o jeho kombinaci s konvoluční

[1] Vrstva, jejíž účelem je podvzorkování dat.

neuronovou sítí určenou pro čtení ručně psaných číslic. Následně byla použita při zpracovávání ručně psaných šeků.

V letech 1985–1990 nastalo druhé těžké období pro obor umělé inteligence, kdy byl dokonce považován za pseudovědecký obor. Podobně, jako v prvním období AI Winter, se našlo pár vědců, kteří se nepřestali danou problematikou zabývat. Díky nim došlo i v této době k několika důležitým objevům, např. v roce 1995 sovětský vědec Vladimir Vapnik a jeho dánský kolega Dana Cortes vytvořili SVM. O dva roky později přišli němečtí počítačoví experti Sepp Hochreiterem a Juergen Schmidhuber s LSTM modelem rekurentní neuronové sítě. Významný evoluční skok následoval v roce 1999, kdy došlo ke zrychlení počítačového zpracování dat.

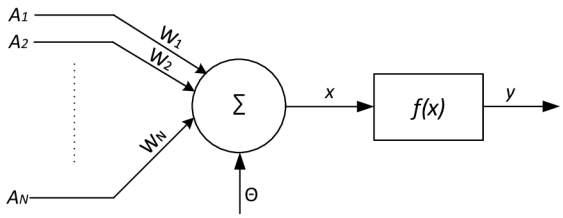
Fei-Fei Li, stanfordský profesor, zabývající se umělou inteligencí, vytvořil v roce 2009 projekt nazvaný ImageNet. Jedná se o volně dostupnou databázi obsahující více než 14 milionů označených obrázků využívaných pro trénování neuronových sítí. V roce 2011 došlo k dalšímu významnému zvýšení výpočetního výkonu grafických akcelerátorů. Důsledkem toho bylo možné používat konvoluční sítě bez předtrénovaných vrstev. V září roku 2017 představila společnost Intel neuromorfní čip Loihi. Jde o mikroprocesor s architekturou napodobující biologickou strukturu mozku. Obsahuje 130 000 neuronů se 130 miliony synaptických spojů a má být až tisíckrát energeticky efektivnější, než současné způsoby práce s AI. Společnost Nvidia vydala v roce 2018 primárně pro vědecké účely grafickou kartu Titan V založenou na architektuře Volta. Zavádí tzv. tenzor jádra poskytující více jak 100 TFLOPS, to je více než pětinásobný nárůst oproti předešlé architektuře Pascal. U některých kusů karet Titan V byly zjištěny chybné výpočty během vědeckých simulací. Na vině je dle odhadů u pamětí absence ECC.

1.1.2 Umělý neuron

Model umělého neuronu, obdoba biologického neuronu, je základní výpočetní jednotkou všech neuronových sítí. Formální neuron je zobrazen na obrázku 1.1. Úkolem neuronu je vytvářet výstupní signál na základě vnitřního potencionálu, který je určen sumou ohodnocených příchozích signálů a prahem. Výstupní signál je předán dalším neuronům, které jsou mezi sebou propojeny ohodnocenými spoji, tzv. synapsemi.

Neuron má zpravidla n vstupů a jeden výstup. Vstupy tvoří vstupní vektor $\vec{A} = (A_1, A_2 \ldots, A_N)$. Každý ze vstupů přispívá do vnitřního potenciálu určitou vahou. Ta je dána příslušnou složkou vektoru synaptických vah $\vec{W} = (W_1, W_2 \ldots, W_N)$. Dalším možným vstupem sumační funkce je prahová hodnota θ. Vnitřní potencionál x uvedeného neuronu je dán sumační funkcí dle rovnice 1.1.

$$x = \sum_{i=1}^{N} A_i \times W_i + \theta \tag{1.1}$$



Obr. 1.1: Model umělého neuronu

Vnitřní potencionál je následně přiveden na vstup aktivační funkce $f(x)$, která definuje způsob, jakým je transformován. Výstup neuronu $y = f(x)$ je tedy závislý na použitém druhu aktivační funkce. Přehled některých aktivačních funkcí je uvedený v tabulce 1.1.

1.1.3 Vrstvy neuronových sítí

V předchozí části byl popsán model neuronu. Ten není sám o sobě schopný řešit složitější úlohy. Proto jsou neurony uspořádány do vrstev, které jsou nejvyšším stavebním blokem neuronových sítích. Jde o kolekci neuronů transformující vstupní signály nejčastěji pomocí některé nelineární aktivační funkce. Zpracované signály jsou předány další vrstvě. Vrstvy obvykle obsahují pouze jeden typ aktivační funkce, nebo jde o vrstvu konvoluční, pooling apod.

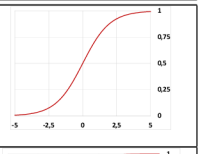
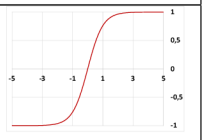
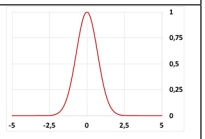
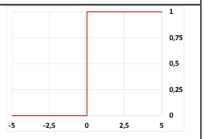
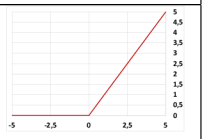
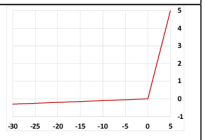
Neuronové sítě mohou být složeny z následujících tří typů vrstev, které jsou známy jako:

1. **Vstupní vrstva** je první vrstvou neuronové sítě zodpovídající za příjem informací (dat), signálů, funkcí nebo měření veličin z externího prostředí. Pro dosažení vyšší přesnosti matematických operací prováděných neuronovou sítí jsou vstupy obvykle normalizovány v mezích limitních hodnot aktivačních funkcí.
2. **Skryté vrstvy** leží mezi vstupní a výstupní vrstvou. Skládají se z neuronů zodpovědných za získání informací spojených s analyzovaným procesem, nebo systémem.
3. **Výstupní vrstva** je zodpovědná za vytvoření finálních výstupů sítě, které jsou výsledkem zpracovávání vstupních dat všemi předešlými vrstvami.

Plně propojená vrstva

Z názvu lze odvodit, že se jedná se o vrstvu, ve které jsou všechny neurony propojeny se všemi neurony vrstvy předešlé. Každý spoj má opět přidělenou určitou váhu. Je paměťově a výpočetně velmi náročná.

Tab. 1.1: Některé aktivační funkce

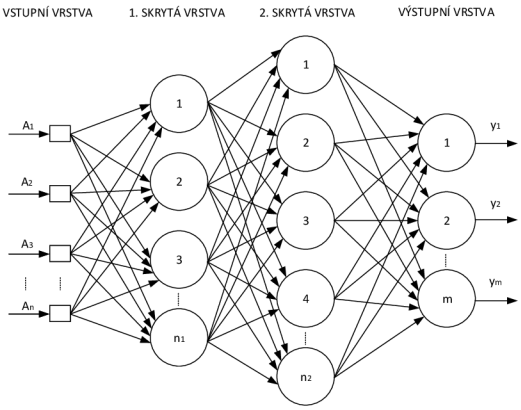
Funkce	**Matematické vyjádření**	**Graf**
Sigmoidní	$f(x) = \frac{1}{1+e^{-x}}$	
Hyperbolický tangens	$f(x) = \tanh(x)$	
Gaussova	$f(x) = e^{-x^2}$	
Skoková	$f(x) = \begin{cases} 1, \text{ pro } x \geq 0 \\ 0, \text{ pro } x < 0 \end{cases}$	
ReLu	$f(x) = \begin{cases} x, \text{ pro } x \geq 0 \\ 0, \text{ pro } x < 0 \end{cases}$	
Leaky ReLu	$f(x) = \begin{cases} x, \text{ pro } x \geq 0 \\ 0,01x, \text{ pro } x < 0 \end{cases}$	

Konvoluční vrstva

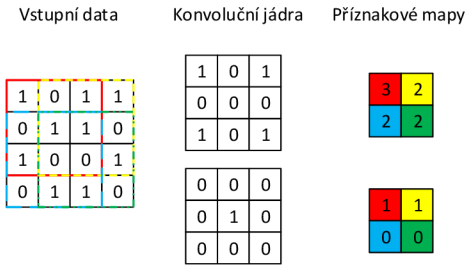
Hlavním úkolem konvoluční vrstvy je získání informací o části vstupních dat pomocí operace konvoluce. Principem je posouvání jádra po vstupních datech. Výstupními daty jsou množiny příznakových map. Velikost této množiny je dána počtem konvolučních jader. Pro příklad je na obrázku 1.3 uvedena konvoluce vstupních dat o rozměrech 4×4 s dvěma konvolučími jádry velikosti 3×3 a výslednými příznakovými mapami rozměrů 2×2.

Pooling vrstva

Jejím hlavním úkolem je slučování několika hodnot do jedné. Tím dochází k jejich redukci, tzv. k podvzorkování a rychlejší konvergenci neuronové sítě. Podle použitého typu seskupení hodnot můžeme dělit pooling vrstvu na max-pooling, avg-pooling, sum-pooling apod. Na obrázku 1.4 je ukázán princip max-poolingu. Zde dochází

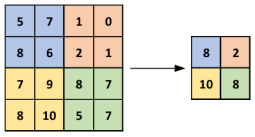


Obr. 1.2: Model vícevrstvé dopředné neuronové sítě



Obr. 1.3: 2D konvoluce

k dvojnásobnému podvzorkování v horizontálním i ve vertikálním směru vybráním vždy maximální hodnoty z dané oblasti.



Obr. 1.4: Max-pooling

Dropout vrstva

Neuronové sítě jsou během procesu učení náchylné k přetrénování. Dropout vrstva zabraňuje tomuto nežádoucímu jevu tím, že během procesu učení náhodně vybere množinu spojů a deaktivuje ji. Tím se jednotlivé neurony stávají na sobě více nezávislé.

SoftMax vrstva

Jde obvykle o poslední vrstvu neuronové sítě, tedy výstupní vrstvu. Její hlavní funkcí je přeměna libovolných reálných hodnot N–rozměrného vektoru $\vec{x}$ na jiný N–rozměrný vektor reálných hodnot $\vec{y}$, které leží v intervalu $\langle 0, 1 \rangle$. Součet všech složek vektoru $\vec{y}$ je poté roven právě 1. Tyto vlastnosti předurčují její využití jako výstupní vrstvy v klasifikačních úlohách (určuje pravděpodobnosti, že vstup náleží do výstupních tříd). Převod jednotlivých složek vstupního vektoru $\vec{x}$ je realizován pomocí matematického vztahu 1.2.

$$y_i = \frac{e^{x_j}}{\sum_{k=1}^{N} e^{x_k}}, \text{pro } i \in 1, 2, \ldots, K \tag{1.2}$$

Mezi další používané vrstvy se řadí například flatten, noise, embedding apod.

1.1.4 Backpropagation

Algoritmus zpětného šíření chyby, nebo-li BackPropagation, byl jedním z nejčastěji používaných algoritmů pro učení neuronových sítí. Jde o algoritmus založený na gradientní metodě (změna gradientu udává, jak se mění chybovost neuronové sítě se změnou vah synaptických spojů). Složitost výpočtu gradientu je vysoká díky velkému počtu synaptických spojů i prvků trénovací množiny. Cílem je změna vah tak, aby docházelo ke klesání gradientu a tím minimalizaci chyby neuronové sítě. Proces algoritmu BackPropagation je složen ze dvou fází definovaných pomocí pravidel.

Dopředné šíření můžeme definovat následně:

1. je provedeno maticové násobení, pokud dojde k setkání matice dat a matice vah,
2. pokud dojde k setkání matice dat s aktivační funkcí, tak je na data daná funkce aplikována,
3. výstup současné vrstvy slouží jako vstup vrstvy následující,
4. pokud dojde k setkání matice dat s chybovou funkcí, jsou vygenerována data pro zpětný průchod.

Zpětné šíření chyby je podobné dopřednému šíření, avšak opačným směrem (výstupní → vstupní vrstva). Jako vstupní data slouží vygenerovaná data dle 4. pravidla dopředného šíření. Pravidla jsou následující:

1. je provedeno maticové násobení transponovanou maticí vah, pokud dojde k jejímu setkání s maticí dat,
2. pokud dojde k setkání matice dat s aktivační funkcí, tak je na data aplikována derivace dané funkce,
3. chybová matice aktuální vrstvy slouží jako vstupní data vrstvy předešlé.

Výpočet gradientu je proveden pomocí maticového násobení mezivýsledku získaného po aplikaci 2. pravidla zpětného šíření chyby a mezivýsledkem získaným po aplikaci 2. pravidla dopředného šíření.

1.2 Rozdělení umělých neuronových sítí

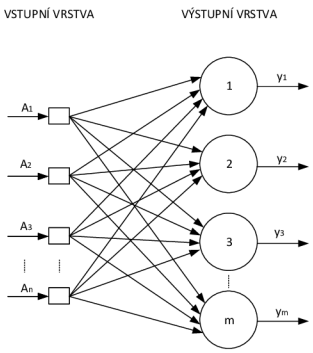
Umělé neuronové sítě mohou být děleny dle určitých kritérií popsaných v této podkapitole. Jde například o členění dle architektury, nebo typu učení.

1.2.1 Dle architektury

Jedno z možných dělení neuronových sítí je dle jejich architektury, tedy zda se jedná o jedno či vícevrstvou dopřednou, nebo zpětnovazební síť.

Jednovrstvá dopředná síť

Tento typ umělé neuronové sítě obsahuje pouze vstupní vrstvu a jedinou vrstvu, která je tvořena neurony a je zároveň vrstvou výstupní. Informační tok je jednosměrný (vstup → výstup). Z obrázku 1.5 je patrné, že počet výstupů jednovrstvé dopředné sítě je totožný s počtem neuronů výstupní vrstvy. Tyto sítě jsou využívány pro klasifikaci vzorků a lineární filtrování.



Obr. 1.5: Jednovrstvá dopředná síť

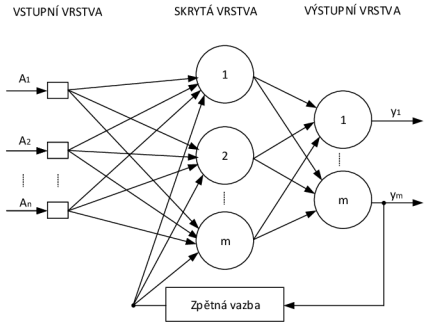
Vícevrstvá dopředná síť

Na rozdíl od předešlé architektury obsahuje více skrytých vrstev viz obrázek 1.2. Využití nacházejí při řešení různých problémů, např. klasifikace vzorků, identifikace systémů, optimalizace, robotika a podobně. Počet skrytých vrstev a jejich neuronů je závislý nejen na charakteru a složitosti řešeného problému, ale i na množství a kvalitě

dostupných trénovacích dat. Opět je počet výstupních signálů shodný s počtem neuronů vrstvy výstupní.

Zpětnovazební síť

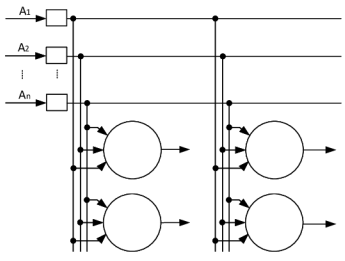
V případě zpětnovazební architektury mohou být použity výstupy neuronů jako vstupy do nich samotných nebo jiných neuronů z některé předešlé vrstvy viz obr. 1.6. Zpětnovazební funkce umožňuje tyto sítě použít pro dynamické zpracování informací. Díky tomu naleznou využití při zpracování časově proměnných systémů, např. predikce časových řad, identifikace systémů a optimalizace, řízení procesů atd.



Obr. 1.6: Zpětnovazební síť

Samoorganizující síť

Její architektura je založena na prostorovém uspořádání neuronů viz obr. 1.7. Nalezne využití v široké škále úloh, např. shlukování dat, rozpoznání vzorů a podobně.



Obr. 1.7: Samoorganizující síť

1.2.2 Dle typu učení

Další možné dělení neuronových sítí je podle přístupu k procesu učení. V rámci tohoto rozdělení jsou popsány čtyři hlavní typy učení.

Učení s učitelem

V případě učení s učitelem je k dispozici množina trénovacích dat. Každý prvek x z této množiny obsahuje i odpovídající výstup Y. Výstupem může být spojitá hodnota, nebo třída vstupní hodnoty, např. určení zda vstup je pozitivním, nebo negativním textem. Dále je k dispozici funkce $Y = f(x)$ jejíž výstup závisí na vstupních datech. Cílem je na základě trénovací množiny upravit parametry této funkce tak, aby co nejpřesněji dokázala přiřadit správnou výstupní hodnotu novým, neznámým vstupním datům.

Učení bez učitele

Učení bez učitele má k dispozici pouze vstupní data x a žádné k nim odpovídající výstupní hodnoty. Cílem je najít vztah nebo strukturu mezi rozdílnými vstupními daty.

Částečně řízené učení

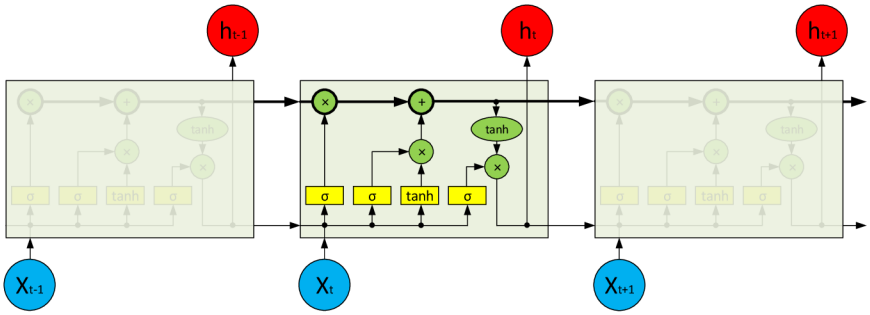
Je směsicí obou výše zmíněných typů učení. Má k dispozici obsahově velkou množinu trénovacích dat x, avšak pouze některé prvky z množiny obsahují odpovídající výstup Y. Na základě označených dat probíhá změna parametrů funkce obdobným způsobem, jako v případě učení s učitelem.

Zpětnovazební učení

Jde o učení, které umožňuje strojům automaticky určit ideální chování a maximalizovat jejich výkon. Učení probíhá pomocí pokusů a omylů.

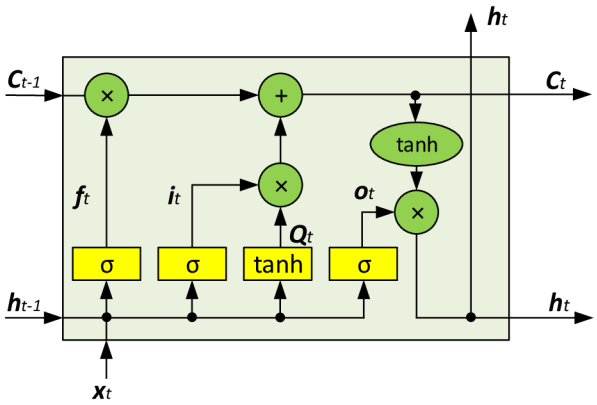
1.3 LSTM sítě

Long Short Term Memory, dále jen LSTM, je speciální druh zpětnovazebních neuronových sítí. LSTM je schopna se učit dlouhodobé závislosti, tedy případy, kdy je aktuální výstup závislý na velmi vzdáleném dřívějším vstupu. Klasické rekurentní neuronové sítě toho byly schopny pouze v teoretické rovině. Díky tomu našly LSTM sítě uplatnění na širokou škálu problému. Například predikce časových řad, rozpoznání řeči a ručně psaných textů, skládání hudby a z hlediska této práce klasifikace textů.



Obr. 1.8: Opakující se modul LSTM

Všechny rekurentní neuronové sítě mají strukturu zřetězených, opakujících se modulů, výjimkou není ani LSTM viz obr. 1.8. Klíčovou vlastností LSTM sítí je vnitřní stav buňky, který prochází celým řetězcem a je ovlivňován pouze lineárními interakcemi (vyznačen tlustou čarou na obrázku 1.8). Díky tomu mohou informace procházet napříč řetězcem beze změn. LSTM mohou upravovat informace v jejich vnitřním stavu pomocí prvků nazývaných brány. Ty se skládají z operace násobení a nelineární sigmoidní funkce. Popisovaná varianta LSTM obsahuje vstupní, výstupní a zapomínající bránu. Samostatná LSTM buňka je na obrázku 1.9.



Obr. 1.9: Buňka LSTM

Zapomínající brána rozhoduje jakým způsobem se bude zacházet s vnitřním stavem buňky, který reprezentuje vektor $\vec{C}_{t-1}$. Na základě současného vstupu a předešlého výstupu je vytvořen za pomoci rovnice 1.3 vektor $\vec{f}_t$. Jeho složky nabývají hodnot od 0 (zahození dané složky vektoru předchozího stavu) do 1 (nezměněná podoba dané složky vektoru předchozího stavu).

Vstupní brána určuje, které nové informace budou přidány do vnitřního stavu buňky. Tento úkol se skládá ze dvou částí. V první části je rozhodnuto pomocí

rovnice 1.4, které složky vektoru $\vec{Q}_t$ budou přidány. Dále je nutné vytvořit vektor samotný viz rov. 1.5.

Aktualizace vnitřního stavu ze staré hodnoty $\vec{C}_{t-1}$ na novou $\vec{C}_t$. Výše zmíněné brány již určily, které informace budou ponechány, respektive zahozeny a jaké nové informace mají být uloženy. Aktualizace probíhá dle rovnice 1.6, kdy nejdříve dojde k vektorovému součinu výstupu zapomínající brány a předchozího vnitřního stavu buňky a následně k vektorovému součtu s výstupem vstupní brány.

Výstupní brána je zodpovědná za vytvoření výstupní hodnoty. Ta je závislá na upraveném aktuálním vnitřním stavu buňky funkcí hyperbolického tangensu. Opět je na základě vstupu a předchozího výstupu rozhodnuto, které složky vnitřního stavu chceme přivést na výstup viz rovnice 1.7. Výstupní hodnota je výsledkem vektorového součinu upraveného vnitřního stavu buňky a výstupu výstupní brány (rovnice 1.8). Čerpáno z [14].

$$\vec{f}_t = \sigma(\vec{W}_f \times [\vec{h}_{t-1}, \vec{x}_t] + b_f) \tag{1.3}$$

$$\vec{i}_t = \sigma(\vec{W}_i \times [\vec{h}_{t-1}, \vec{x}_t] + b_i) \tag{1.4}$$

$$\vec{Q}_t = \tanh(\vec{W}_C \times [\vec{h}_{t-1}, \vec{x}_t] + b_c) \tag{1.5}$$

$$\vec{C}_t = \vec{f}_t \times \vec{C}_{t-1} + \vec{i}_t \times \vec{Q}_t \tag{1.6}$$

$$\vec{o}_t = \sigma(\vec{W}_o \times [\vec{h}_{t-1}, \vec{x}_t] + b_o) \tag{1.7}$$

$$\vec{h}_t = \vec{o}_t \times \tanh(\vec{C}_t) \tag{1.8}$$

2 KLASIFIKACE TEXTŮ

Počítačové zpracování přirozeného jazyka je jedním z oborů, ve kterých dosáhly metody založené na hlubokém učení významného pokroku. Velký zájem akademických obcí i komerční sféry vzbuzuje oblast klasifikace textu, ve které je hlavním cílem třídit dokumenty, odstavce, nebo jednotlivé věty do předem definovaných kategorií. Příkladem může být zařazení věty do určité kategorie dle jejího obsahu, nebo určení sentimentu zkoumaného textu (pozitivní, negativní, nebo neutrální). Následně mohou být takto získané informace různými společnostmi využity k určení marketingové strategie, získání nových potencionálních zákazníků, nebo za účelem zlepšení kvality zákaznického servisu.

Využití rekurentních neuronových sítí vedl k významnému pokroku počítačového zpracování přirozeného jazyka. Tyto typy sítí, zejména pak LSTM, dosáhly slibných výsledků v úlohách pracujících s časovými řadami (rozpoznání řeči, strojový překlad, klasifikace textu, apod.). V současné době je klasifikace textu nejčastěji řešena pomocí modelů využívajících funkcí, které pracují se slovními prvky. Mezi ně patří raneček slov (Bag-of-words), n-gramy (n-grams) a shlukování slov (Word embedding). Mezi metody klasifikace textu lze dále zařadit raneček slov s TF-IDF, K-průměry (K-means) s word2vec, konvoluční i LSTM neuronové sítě se shukováním slov a raneček n-gramů (Bag-of-n-grams) s lineárním klasifikátorem.

Vyjmenované přístupy klasifikace textu vyžadují jeho určité předzpracování, například lemmatizaci[1]. Právě z nutnosti předzpracování textu, které je často závislé na konkrétním jazyku, nelze vytvořit univerzální klasifikátor. V případě změny příslušného jazyka (nová slova, fráze, apod.) je nutné dřívěji získané znalosti upravit.

Popsané nevýhody se rozhodli odstranit počítačoví experti z New Yorské univerzity. Inspirovali se úspěchem konvolučních neuronových sítí v oblasti počítačového vidění, kde dochází k učení pouze na základě surových pixelů. Původní předpoklad úspěšné klasifikace textu pouze na úrovni surových znaků byl správný. Tato diplomová práce se také zabývá klasifikací textu na úrovni znaků, ale s využitím kombinace konvolučních a rekurentních neuronových sítí.

2.1 Tradiční metody

V rámci dané části jsou popsány některé z tradičních metod pro klasifikaci textů. Jde zejména o raneček slov, raneček n-gramů, jejich modifikaci s TF-IDF a metodu podpůrných vektorových strojů.

[1] Vytvoření základního tvaru slova (barvě → barva)

2.1.1 Raneček slov

Raneček slov, (dále jen BoW), je algoritmus používaný v počítačovém zpracování přirozeného jazyka. Jedním z jeho hlavních cílů je klasifikace textu na základě analýzy ranečku slov. BoW si lze představit jako stroj zpracovávající množinu textu, jehož výstupem je tabulka. Její jednotlivé řádky představují vstupní texty a sloupce četnost daného slova v odpovídajících textech. Příkladem je tabulka 2.1 získaná po analýze tří vět uvedených v seznamu níže.

1. Líbí se mi tenis a hokej.
2. Nelíbí se mi golf.
3. Rád hraji tenis a rád spím.

Tab. 2.1: Vektory reprezentující počet slov výše vypsaných textů

Text	**Slova**										
–	líbí	se	mi	tenis	a	hokej	nelíbí	golf	rád	hraji	spím
1.	1	1	1	1	1	1	0	0	0	0	0
2.	0	1	1	0	0	0	1	1	0	0	0
3.	0	0	0	1	1	0	0	0	2	1	1

Lze pozorovat, že slova **líbí** a **golf** se vyskytují pouze v první, respektive ve druhé větě. Slova **rád**, **hraji**, **spím** se objevují výhradně v poslední, třetí větě. Získané vektory bývají obvykle normalizovány celkovým počtem slov dle následujícího vztahu:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \,, \tag{2.1}$$

kde $tf_{i,j}$ je složka normalizovaného vektoru (příslušné slovo), $n_{i,j}$ představuje četnost výskytu daného slova v textu t_j a $\sum_k n_{k,j}$ reprezentuje počet slov v textu t_j. Příkladem je normalizovaný vektor třetí věty $\vec{t_3} = (0, 0, 0, \frac{1}{6}, \frac{1}{6}, 0, 0, 0, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$. Algoritmus pracuje pouze s četností termů (Term frequency). V některých případech je velmi žádoucí nějakým způsobem vyzdvihnout významná slova analyzovaných textů (viz výše tučně zvýrazněná). Toho lze docílit pomocí níže popsané TF-IDF metodiky.

Klasifikátor využívající BoW byl testován v práci [23]. Vykazoval chybovost 9,60% při klasifikaci textů do dvou kategorií (pozitivní/negativní) dle sentimentu. Při stejném typu klasifikace, tentokrát do pěti kategorií, chybně klasifikoval 45,36% vzorků. Model BoW s TF-IDF vykazoval o něco nižší chybovost, 9%, respektive 44,74%. Nevýhodou ranečku slov je nerespektování pořadí slov, čímž dojde ke ztrátě významu textu (různé věty obsahující stejná slova, ale v jiném pořadí, mají jiný význam, avšak stejnou reprezentaci).

2.1.2 TF-IDF

TF-IDF je metodika kombinující četnost výskytu slov s inverzní četností slov ve všech textech (dokumentech). První složkou je četnost výskytu slov (výstup modelu BoW) určující významnost daného slova v textu (častěji se vyskytující slovo bude pravděpodobně velmi důležité). Druhou složkou je inverzní četnost slov ve všech dokumentech. Inverzní čestnost říká, že aby bylo možné považovat slovo z dokumentu za významné, nemělo by se často vyskytovat v ostatních dokumentech. Jinak řečeno, četnost výskytu významného slova by měla být v ostatních dokumentech nízká, respektive inverze této četnosti vysoká. IDF je vypočtena dle následujícího vzorce:

$$idf_i = \frac{|T|}{|\{f : t_i \in t_j\}|}\ , \tag{2.2}$$

kde $|T|$ představuje celkový počet textů (dokumentů) a $|\{f : t_i \in t_j\}|$ je počet textů, ve kterých se vyskytuje dané slovo i. TF-IDF hodnoty jsou výsledkem vektorového násobení složek TF a IDF.

2.1.3 N-gramy

Metoda pracuje na obdobném principu jako klasický BoW. Rozdílem je seskupování n slov. Příkladem může být vytvoření následujících dvojic (Líbí–se , se–mi, mi–tenis, tenis–a, a–hokej) z první věty obsažené v tabulce 2.1. Model n-gram s modifikací TF-IDF byl také testován v práci počítačových expertů Xianga Zhanga a Yanna LeCuna [23]. Pro testy bylo zvoleno seskupení pěti slov, tzn. $n = 5$. Model n-gram i n-gram využívající TF-IDF vykazoval chybovost 7,98%, respektive 8,46%. Bylo tedy dosaženo lepších výsledků, než vykazoval raneček slov. Model n-gram s TF-IDF vykazoval vetší chybovost oproti n-gramu bez této modifikace. Výhodou n-gramů ve srovnání s ranečkem slov je do jisté míry zachování pořadí slov. V případě vhodně zvoleného počtu seskupovaných slov je možné zachytit fráze. Tím lze dosáhnout menší chybovosti.

2.1.4 SVM klasifikátor

V roce 2016 byla představena Lukášem Povodou, Radimem Burgetem, Janem Maškem, Václavem Uherem a Malayem Duttou metoda automatického rozpoznání emocí s využitím SVM klasifikátoru [16]. Vstupní data nejprve procházejí procesem předzpracování. Prvním krokem je rozdělení textu na slova, tzv. tokenizace. Takto získané tokeny projdou kontrolou pravopisu, případně jsou detekované chyby opraveny. Tokeny následně projdou procesem lemmatizace. Jsou odstraněny předložky a pomocná slovesa, totéž platí o množině nevýznamných slov. Výsledkem je množina pouze významných slov, často nesoucích emoční náboj. Nakonec jsou zbylá slova nahrazena

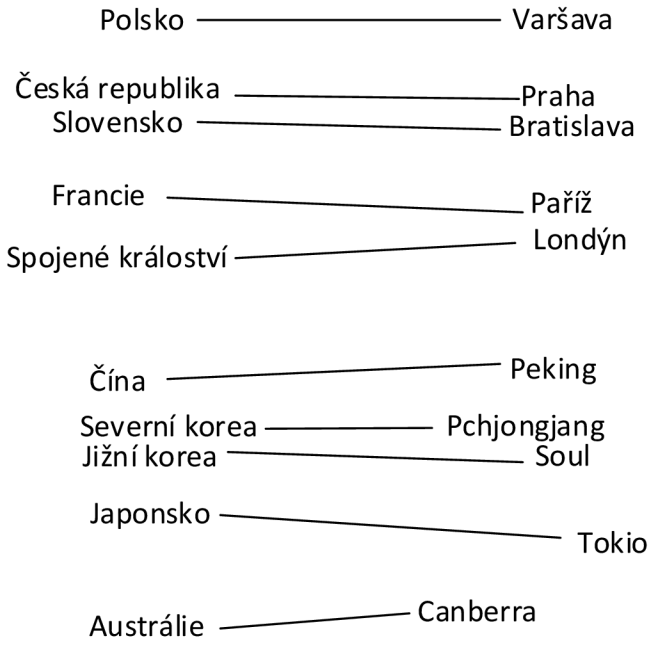
identifikátorem skupiny slov (hněv, vulgarismus apod.). Prezentovaný model dosáhl vysoké úspěšnosti – 86,89% při klasifikaci do pěti tříd. Nevýhodou zůstává jazyková závislost způsobená předzpracováním textu (lemmatizace, definované skupiny slov).

2.2 Shlukování slov, vět a dokumentů

Shlukování slov je funkce $W : \textit{slovo} \rightarrow \mathbb{R}^n$ převádějící slova do N-rozměrného (např. 100 rozměrů) vektorového prostoru. Každý z možných rozměrů může odpovídat určitému tématu, nebo vlastnosti daného slova. Shlukování slov se používá v mnoha úlohách počítačového zpracování přirozeného jazyka, zejména pokud jsou tyto úlohy vykonávány pomocí hlubokého učení. Vektorová reprezentace slov může být přímo použita jako vstup do neuronové sítě.

$$W(\text{"neuron"}) = (0.9, 0.7, -0.4, \dots)$$
$$W(\text{"pes"}) = (0.1, -0.4, 0.7, \dots)$$

Často se vyskytující slova leží ve vektorovém prostoru ve své blízkosti. S vektory lze provádět aritmetické operace. Na obrázku 2.1 je dvourozměrný vektorový prostor přiřazující hlavní města k danému státu. Dochází také k zohlednění geografické pozice států.



Obr. 2.1: Vztah stát - hlavní město

Existuje několik algoritmů vytvářejících vektorový prostor, jde například o Word2vec, GloVe, Word2vecf apod.

2.2.1 Word2vec

Patrně nejpopulárnější modelová architektura pro výpočet vektorových reprezentací slov. Je považována za počátek využití hlubokého učení při zpracování přirozeného jazyka. Účelem Word2vec je seskupení slov sémanticky si podobných do vektorového prostoru. Tím vytváří bohatou distibuovanou reprezentaci slov zachovávající sémantické vztahy mezi nimi samotnými. Výsledkem je schopnost nalézt podobná slova, nebo vztahy mezi jednotlivými slovy. V rámci Word2vec existují dva základní přístupy.

Kontinuální raneček slov

Dále označovaný jen CBOG (Continous Bag-of-Words) je architektura pro predikci slov. Predikce slova w_i probíhá na základě kontextu dvou minulých a dvou následujících slov w_{i-2} , w_{i-1} , w_{i+1} a w_{i+2}. Výsledek nezáleží na pořadí slov použitých pro predikci, proto název architektury obsahuje „raneček slov“.

Kontinuální Skipgramový model

Tato architektura je podobná CBOG, ale místo predikce aktuálního slova na základě kontextu se snaží předpovědět slova w_{i-2} , w_{i-1} , w_{i+1} , w_{i+2} ležící v určité vzdálenosti od slova aktuálního w_i.

2.2.2 GloVe

Algoritmus [18] s učením bez učitele sloužící pro získání vektorové reprezentace slov. Trénování probíhá na základě statistiky společného výskytu slov v korpusu. GloVe využívá pro měření lingvistické nebo sémantické podobnosti slov Euklidovskou vzdálenost. Díky této metrice někdy odhalí vzácná, ale relevantní slova ležící mimo průměrnou lidskou slovní zásobu (například nejbližší slova ke slovu žába jsou: Litoria, Eleutherodactylus – jde o rody žab). V některých testech syntaktických a sémantických úloh dosahuje lepších výsledků ve srovnání s Word2vec viz [23]. Nevýhodou je větší paměťová náročnost.

2.2.3 Word2vecf

Model byl publikován v roce 2014 Omerem Levy a Yoavem Goldbergem [8], zaměstnanci izraelské univerzity Bar-Ilan. Základem je upravený skip-gramový model

s negativním vzorkováním[2], který umožňuje zahrnout kontexty slov. Experimentují s kontextem založeným na závislosti zajišťující menší tématickou závislost, než původní skip-gramový model. Během nutného předzpracování surového textu dojde k vytvoření dvojice slovo–kontext, která je následně použita jako vstup modelu.

2.2.4 Adaptivní skip-gramový model

Za adaptivním skip-gramovým modelem, nebo-li AdaGramem, stojí trojice ruských výzkumníků a jeden francouzský odborník. Jde o model, který je neparametrickým Bayesovským rozšířením skip-gramového modelu. Je schopen automatického učení určitého počtu reprezentací slov při požadovaném sémantickém rozlišení. Díky tomu je vyřešen problém vícevýznamových slov. Podrobnější informace lze nalézt v [1].

Ke shlukování vět nebo dokumentů by mohlo být přistupováno pouze sčítáním vektorů jednotlivých slov vytvořených za pomocí výše zmíněných modelů. Lepší řešení poskytují následující modely pracující na principu upraveného sčítání vektorů.

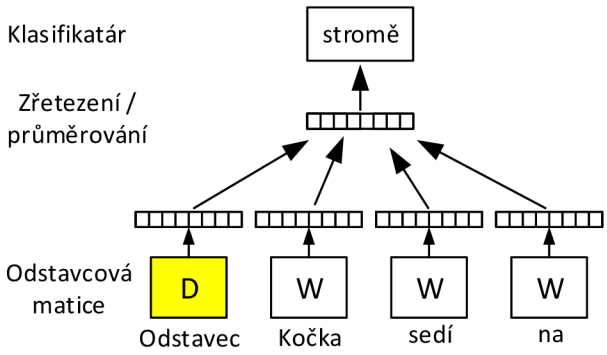
2.2.5 Paragraph2vec

Model byl navržený v práci publikované v roce 2014 Tomášem Mikolovem a Quoc Le [7]. Jde o model s učením bez učitele, který je schopný učit se kontinuální distribuované vektorové reprezentace textu mající délku v rozsahu od vět až po celé dokumenty. Pro predikci slova v odstavci se využívá spojení odstavcového vektoru s několika slovními vektory viz obr. 2.2. V práci prezentované výsledky experimentu přisuzují Paragraph2vec nejvyšší přesnost, respektive nejnižší chybovost v úloze klasifikace sentimentu a to 12,2%. Pokud by bylo užito techniky prostého průměrování slovních vektorů, dojde k chybovosti 19,9%. Kompletní výsledky experimentu lze nalézt v [7]. Hlavní výhodou Paragraph2vec je možnost učit se z nepopsaných dat a také úvaha pořadí slov. Dále přebírá klíčovou vlastnost BoW – sémantiku slov.

2.2.6 Skip-through

Zde dochází ke změně principu shlukování. Přechází se z úrovně slov na úroveň vět. Jednoduše řečeno, místo využití slova k predikci okolního kontextu dochází ke kódování celé věty. Následně jsou pomocí této věty předpovídány věty sousední. Model vyžaduje korpus kontinuálního textu (vstup musí být souvislý, sousední věty jsou si sémanticky blízké) [6].

[2]Trénovací vzorek aktualizuje pouze malé procento vah modelu



Obr. 2.2: Paragraph2vec

2.3 Související práce

Tato práce je inspirována již potvrzenou myšlenkou počítačových expertů z New Yorské univerzity. Ve svých pracích uvedli, že text je ve skutečnosti časová řada a lze jej klasifikovat na základě surových znaků bez nutných jazykových znalostí. Zpracování časových řad může být provedeno s využitím rekurentních neuronových sítí. Rekurentní neuronové vrstvy jsou nedílnou součástí modelů navržených v rámci této diplomové práce.

2.3.1 Porozumění textu od nuly

Práce[22] (Text understanding from scratch) byla publikovaná počítačovými experty Xiangem Zhangem a Yannem LeCunem, pracujícími na ústavu informačních technologií New Yorské univerzity, v únoru roku 2015. Zabývá se aplikací hlubokého učení s cílem porozumění textů nikoliv na základě slov, ale znaků. Prezentované výsledky úspěšně potvrdily možnost strojového porozumění textů bez nutné znalosti slov, frází, vět či některých jiných syntaktických, nebo sémantických struktur spjatých s určitým jazykem.

Aplikovali model konvoluční neuronové sítě na různorodé úkoly týkající se klasifikace textů. Vstupními daty neuronové sítě jsou sekvence kódovaných znaků a výstupními daty jsou abstraktní vlastnosti textů. Přístup lze shrnout do následujících dvou vět.

1. Místo práce se slovy dochází pouze k práci se znaky. Odpadá nutnost využití shlukování slov.
2. Není nutná znalost sentimentální nebo sémantické struktury.

Model

Byl navržen velký a malý model konvoluční sítě. Oba jsou složeny celkem z devíti vrstev – šesti konvolučních a tří plně propojených vrstev lišících se velikostí, respektive počtem neuronů. Každý model také obsahuje tři Max-Pooling moduly a dva dropout moduly vložené mezi plně propojené vrstvy mající pravděpodobnost vynechání spojů 0,5. Počáteční váhy byly inicializovány pomocí Gaussova rozdělení se střední hodnotou 0 a směrodatnou odchylkou 0,02 pro velký model, respektive s hodnotami (0, 0,05) pro malý model. Vstupní a výstupní dílka dat se liší dle daného klasifikačního problému.

Zvětšení dat

Vhodný způsob zvětšení dat je užitečný při kontrolování chyby modelů hlubokých neuronových sítí. Tyto techniky obvykle dobře fungují v případě možnosti najití invariantních vlastností, které by klasifikátor měl mít. Například model pro rozpoznání obrazu by měl být nezávislý na případné rotaci, převrácení nebo změně velikosti vstupního obrazu. V případě textového klasifikátoru představuje invarianci nahrazení slov nebo frází synonymy, čímž nedojde ke změně významu textu.

Pro zvětšení dat byl použit anglický slovník obsahující synonyma, která byla ohodnocena na základě sémantické blízkosti k nahrazovanému slovu. Prvním krokem zvětšení dat bylo extrahování všech nahraditelných slov. Následně došlo k vybrání synonyma, kterým bylo dané slovo nahrazeno.

Analýza sentimentu

Výzkum se zabýval klasifikací dle ontologie, tématu, zpravodajských témat a analýzou sentimentu. Pro účel analýzy sentimentu byla použita databáze hodnocení produktů na Amazonu. Systém hodnocení obsahuje 5 kategorií pro subjektivní hodnocení produktu uživatelem. Databáze obsahuje více než 34,5 milionů hodnocení od více jak 6,5 milionů uživatelů, kteří hodnotili na 2,4 milionů produktů viz tab. 2.2. Pro účely analýzy sentimentu byly náhodně vybrány texty s rozsahem 100 až 1014 znaků. Počet jednotlivých vzorků pro klasifikaci do dvou, respektive pěti kategorií je uveden také v tabulce 2.2. V případě klasifikace na pozitivní, či negativní text sloužilo 1 800 000 vzorků jako testovací data a 200 000 jako trénovací data pro obě kategorie. Pro druhý případ sloužilo z každé kategorie 600 000 vzorků jako testovací data a 130 000 jako data trénovací. Z výsledků vyplývá, že při klasifikaci všech kategorií dosahuje nejlepších výsledků malý model konvoluční neuronové sítě s 59,57% úspěšností a v případě klasifikace na pozitivní/negativní text nejvyšší úspěšnosti

(95,07%) dosahuje velký model konvoluční sítě. V obou případech je trénovací úspěšnost vyšší bez využití zvětšení dat, ale s jeho využitím roste testovací úspěšnost viz tab. 3.1.

Tab. 2.2: Struktura databáze a počty vybraných vzorků pro daný typ klasifikace

Známka	**Celkem vzorků**	**Vzorky – 5 kategorií**	**Vzorky – 2 kategorie**
1	2 746 559	730 000	1 275 000
2	1 791 219	730 000	725 000
3	2 892 566	730 000	0
4	6 551 166	730 000	725 000
5	20 705 260	730 000	1 275 000

Tab. 2.3: Výsledky klasifikace do dvou i pěti kategorií

Model	**Zvětšení dat**	**Klasifikace – 5 kategorií**		**Klasifikace – 2 kategorie**	
–	–	Trénovací	Testovací	Trénovací	Testovací
Velký	Ne	62,96%	58,69%	**97,57**%	94,49%
Velký	Ano	68,90%	59,55%	96,82%	**95,07**%
Malý	Ne	**62,24**%	59,47%	96,03%	94,50%
Malý	Ano	62,11%	**59,57**%	95,44%	94,33%
BoW	Ne	54,45%	54,17%	89,96%	89,86%
W2V	Ne	36,56%	36,50%	72,95%	72,86%

2.3.2 Klasifikace textu pomocí konvolučních neuronových sítí na úrovni znaků.

Práce (Character-level convolutional networks for text classification) byla publikována experty Xiangem Zhangem, Yannem LeCunem, Junbem Zhaoem a vznikla rozšířením původního, výše popsaného výzkumu. Rozšířením je značné zvětšení počtu experimentů a srovnání výsledků s více modely (n-grams, LSTM s embeddingem apod). Podrobné výsledky testů lze najít v [23].

2.3.3 LSTM rekurentní neuronové sítě pro krátké textové a sentimentální klasifikace

Článek [11] (LSTM Recurrent Neural Networks for Short Text and Sentiment Classification) se zabývá využitím LSTM sítí a jejich modifikacemi (obousměrné LSTM

a GRU) pro klasifikaci textů. Data použitá pro kategorizaci dle sentimentu byla získána z Amazon databáze. Jednalo se o recenze knih, které byly rozděleny do pozitivní, negativní, nebo neutrální kategorie. Texty byly normalizovány tak, aby gramatika ani interpunkce neměla vliv na výsledek (použití pouze malých písmen, odstranění speciálních znaků, nahrazení číslic speciální značkou). Došlo k vytvoření slovníku unikátních slov, kde každé z nich je reprezentováno jedním vektorem sloužícím jako vstup do neuronové sítě. Nejvyšší úspěšnost klasifikace do tří kategorií byla dosažena pomocí obousměrné LSTM (86,40%). Podrobné výsledky a popis průběhu experimentu lze nalézt v [11].

2.3.4 Sentimentální analýza filmových recenzí založená na CNN–BLSTM neuronových sítích

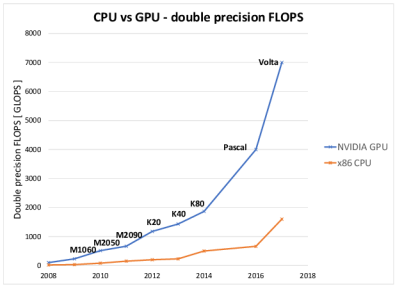
Článek [19] (Sentiment Analysis of Movie Reviews Based on CNN-BLSTM) publikovaný v září roku 2017 se zabývá kombinací konvolučních a rekurentních neuronových vrstev pro klasifikaci textů do pozitivní, nebo negativní kategorie. Vstupní data byla převedena do padesáti dimenzionálního vektorového prostoru pomocí GloVe algoritmu. Datová množina obsahovala 50 000 uživatelských filmových recenzí z IMDb databáze, 40 000 vzorků bylo použito při trénovacím procesu a zbytek byl použit během testování natrénovaných sítí. V prvním experimentu byl zkoumán rozdíl úspěšnosti obousměrné LSTM sítě a kombinace této sítě s konvoluční sítí. Ta vykazovala o 2,8% vyšší úspěšnost klasifikace. Druhý experiment se zabýval vlivem počtu konvolučních sítí a použitím předtrénovaného vektorového prostoru na klasifikační úspěšnost. Kompletní výsledky lze nalézt v [19].

3 POUŽITÉ TECHNOLOGIE

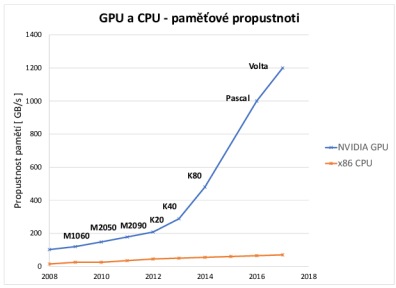
Obsahem kapitoly je popis využitých technologií a knihoven, s jejichž pomocí byla řešena praktická část této diplomové práce.

3.1 Výpočty s využitím grafického procesoru

Zvyšující se tlak pro zpracovávání 3D grafiky ve vysokém rozlišení a v reálném čase vedlo k vývoji mnohojádrových grafických procesorů, které disponují obrovským výpočetním výkonem a vysokou paměťovou propustností viz obr. 3.1, respektive 3.2. Vzhledem k výkonovému rozdílu mezi CPU a GPU je snahou využít grafický procesor pro zrychlení obecných výpočtů. GPGPU je metoda paralelního zpracování dat grafickým procesorem.

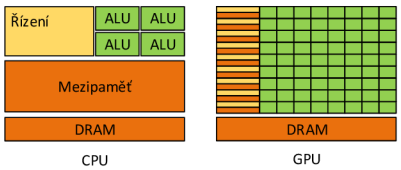


Obr. 3.1: GPU a CPU - porovnání DP FLOPS



Obr. 3.2: GPU a CPU - porovnání paměťové propustnosti

Velký výkonový rozdíl v FLOPS mezi CPU a GPU je dán optimalizací grafického procesoru pro účely paralelních výpočtů. Čip GPU je navržen tak, aby co nejvíce tranzistorů bylo určeno pro početní operace, zatímco pro účely řízení toku dat nebo ukládání dat do mezipaměti jich bylo nezbytné minimum viz obr. 3.3. GPU

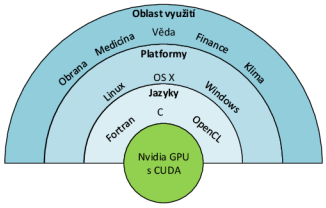


Obr. 3.3: Tranzistory pro různé účely na čipu CPU a GPU

je vhodný využít při řešení úkolů, které mohou být paralelizovány s vysokou výpočetní intenzitou[1]. Využitím paralelismu dojde ke snížení požadavků na řízení toku dat. Latence přístupu k paměti může být skryta (odpadá nutnost velké vyrovnávací paměti) provedením mnoha výpočtů s vysokou výpočetní intenzitou. Čerpáno z [2].

3.1.1 CUDA

CUDA byla představena společností Nvidia v listopadu roku 2006. Jedná se o paralelní výpočetní a programovací model. Umožňuje efektivnější řešení výpočetních úloh s využitím GPU, než pomocí klasických procesorů. CUDA podporuje různé programovací jazyky a API, např.: C, C++, Fortran, Java, Python, OpenCL. Je multiplatformní (Windows, Linux a OS X) a využívá se v mnoha odvětvích, například v medicíně, zpracování videa, klimatických výpočetních modelech apod.



Obr. 3.4: CUDA architektura

Škálovatelný programovací model

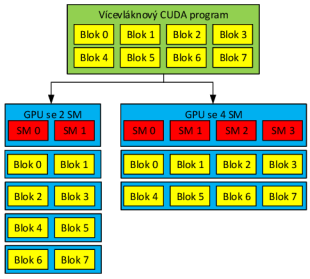
CUDA model je navržen tak, aby byl se zvyšujícím se počtem výpočetních jader udržen vysoký stupeň paralelizace a nebyly neúměrně zvyšovány požadavky na programátorské schopnosti. Toho je docíleno následujícími třemi klíčovými abstrakcemi:

- hierarchie vláken,
- sdílená paměť,
- bariérový synchronizační vzor.[2]

Zmíněné abstrakce vedou programátora k dělení řešeného problému na dílčí úkoly. Ty mohou být paralelně a nezávisle zpracovávány pomocí bloků vláken. Dále jsou dílčí úkoly členěny na jemnější výpočty, které mohou být zpracovávány souběžně všemi vlákny v rámci daného bloku. Každý blok vláken může být spuštěn na libovolném, dostupném multiprocesoru GPU. O počet fyzicky přítomných multiprocesorů na GPU se stará běhové prostředí viz 3.5.

[1] Poměr výpočetních ku paměťovým operacím.

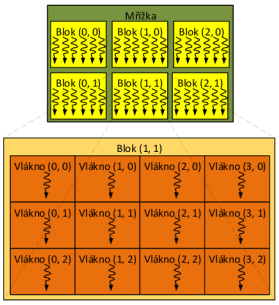
[2] Všechna vlákna čekají v určitém bodě na poslední



Obr. 3.5: Automatická škálovatelnost

Hierarchie vláken a pamětí

Základní jednotkou je vlákno. Každé z nich může přistupovat k vlastní i globální paměti a je identifikováno pomocí jedno, dvou, nebo trojrozměrného vektoru. Vlákna jsou shlukována do bloků (jedno/dvou/trojrozměrného). Ty mají k dispozici rychlou sdílenou paměť s nízkou latencí přístupu. Vzhledem ke konečné velikosti této paměti je počet vláken na blok limitován. Současné GPU nabízejí nejčastěji 1024 vláken/blok (Compute capability 2.x a vyšší). Multiprocesor přepíná mezi jednotlivými vlákny, čímž dochází k zakrytí latence globální paměti. Kernel je obvykle tvořen více bloky, které jsou uspořádány do jedno, dvou, nebo trojrozměrné mřížky viz obr. 3.6. K dispozici na GPU jsou ještě paměti konstant a textur umožňující pouze čtecí operace. Čerpáno z [10], [13].



Obr. 3.6: Hierarchie vláken

3.2 TensorFlow

TensorFlow [17] byl vyvinut týmem inženýrů a vědců pracujícími pro společnost Google v rámci organizace zabývající se umělou inteligencí. Jedná se o knihovnu s otevřeným zdrojovým kódem pro numerické výpočty s použitím diagramu datových toků. Uzly v diagramu představují jednotlivé matematické operace. Komuni-

kace mezi nimi je zprostředkována pomocí hran. Ty reprezentují multidimensionální datová pole (tenzory). Adaptabilní architektura TensorFlowu a jednotné API umožňuje vykonávat výpočty na jednom, nebo více CPU či GPU mobilního, stolního, případně serverového zařízení.

3.3 Keras

Keras [5] byl vyvinut zaměstnancem Googlu Françoisem Cholletem za účelem usnadnění experimentů s neuronovými sítěmi. Jde o vysokoúrovňové API napsané v jazyce Python. Pracuje nad některou z knihoven pro numerické výpočty, např. TensorFlow, Theano, Deeplearning4j, CNTK, nebo MXNet. Umožňuje snadné a rychlé prototypování díky modularitě, rozšiřitelnosti a uživatelské přívětivosti. Bezproblémově pracuje na CPU nebo GPU a nabízí podporu pro konvoluční, rekurentní sítě i jejich kombinaci.

3.4 KEX

Jde o knihovnu [4] uživateli zjednodušující proces trénování hlubokých neuronových sítí s využitím nástrojů Keras, Theano, nebo TensorFlow. Kex vytvoří jeden experiment ze všech definovaných modelů neuronových sítí. Následně dojde k procesu trénování těchto modelů a uložení všech konečných výsledků i konfiguračních souborů do příslušných složek daného experimentu. Díky tomu lze jednoduše porovnávat úspěšnost a výkonnost jednotlivých definovaných modelů.

3.5 Výpočetní zařízení

Výpočty probíhaly na třech zařízeních. Jejich parametry jsou uvedeny v tab. 3.1.

Tab. 3.1: Parametry zařízení

–	**Notebook**	**PC**	**Server**
OS	Windows 10	Windows 10	Ubuntu 4.4.0-97.120
CPU	Athlon II X4 630	Core i5-3230M	2x Xeon E5-2640 v2
GPU	GTX 750 1GB	GT 740M 2 GB	GTX 1080 Ti 11 GB
RAM	8 GB	8 GB	64 GB
Keras	2.1.5	2.1.5	2.1.5
TensorFlow	1.40	1.40	1.40
cuDNN	v6.0	v6.0	v6.0

4 ŘEŠENÍ

Obsahem této kapitoly je řešení cíle diplomové práce. Popisuje přístup a způsob zpracování textových databází. Dále je zde věnován prostor vytvořeným modelům neuronových sítí spolu s prezentací dosažených výsledků.

4.1 Soubor dat

Navržené neuronové sítě používají vstupní data, kterými jsou pozitivní a negativní texty poskytované dvojicí databází. Navržené modely neuronové sítě byly trénovány především na anglicky psaných textech, jelikož množství těchto dat bylo několikanásobně větší, než v případě ostatních tří jazyků. Souborem dat anglicky psaných textů byla využita volně dostupná Yelp databáze obsahující více jak 5 000 000 uživatelských recenzí, stažená jako záloha SQL databáze. Pro účely této práce byla relevantní pouze data obsahující pozitivní a negativní recenze. Pro snadnější manipulaci s relevantními daty bylo provedeno extrahování jednotlivých textů. Následně byly vloženy do nové databáze. Každý databázový záznam obsahuje pouze nezbytné informace pro trénování a testování neuronových sítí. Úprava ulehčila manipulaci s daty a zmenšila jejich původní velikost z lehce překračujících 7 GB na přibližně 1,5 GB. Během procesu trénování a testování navržených modelů neuronové sítě bylo využito 600 000 anglicky psaných recenzí z každé klasifikované kategorie.

Druhá vícejazyčná databáze zahrnuje česky, anglicky, německy a španělsky psané pozitivní a negativní texty. Celkem zahrnuje 48 000 recenzí. Každá množina dat jednoho ze čtyř jazyků představuje 12 000 textů. Dále se dělí na dvě podmnožiny představující klasifikované kategorie, každá z nich disponuje 6 000 texty. Vzhledem k využití Yelp databáze je množina anglicky psaných textů irelevantní. Bylo vhodné sjednotit práci s oběma databázemi, a proto byla data vícejazyčné databáze vložena do tabulek relační databáze.

Všechny databáze byly nahrány na open source databázový MySQL server. Databázové schéma obsahuje osm tabulek. Pro každý jazyk jsou negativní a pozitivní texty uloženy v samostatných tabulkách.

4.2 Zpracování jednotlivých textů

Před vstupem do neuronové sítě musí být jednotlivé surové texty upraveny. Z důvodu nedostatečného výpočetního výkonu je nutné každý text nejprve omezit na předem definovaný počet znaků. Poté jsou všechna velká písmena převedena na malá. Úpravou je docíleno zmenšení rozměrů vstupní matice do neuronové sítě a tím urychlení

trénovacího procesu. Nicméně by bylo zajímavé pozorovat změnu úspěšnosti klasifikace, pokud by tato informace byla brána v úvahu (využití velkých písmen se často používá k zdůraznění nějakého pocitu nebo informace). Vypuštěním sledování velkých písmen dojde k ustálení počtu sledovaných znaků na 88 (a-z , 0-9, znaky češtiny, španělštiny a němčiny, znak pro americký dolar, euro a libru, mezera a následující znaky: !#%&<>*,-./:;=?@[]_^°~'\).

4.2.1 Převod surového textu do maticové podoby

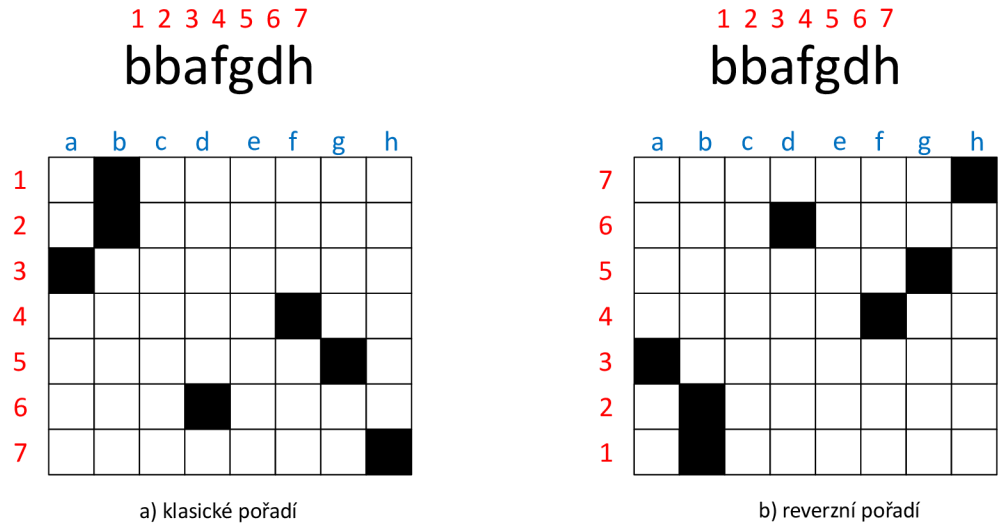
Dříve již bylo zmíněno, že nelze použít surový text jako přímý vstup do neuronové sítě, proto musí být nejprve převeden do vhodné reprezentace. Tou je matice typu $m \times n$. V takovéto matici představuje každý řádek jeden textový znak. Výpočetní náročnost na zpracování vstupních matic roste s počtem řádků (uvažovaných znaků textu) m. Z tohoto důvodu je jejich počet omezen. Druhý rozměr matice n udává počet sloupců a je dán velikostí abecedy znaků. Každý sloupec reprezentuje jeden sledovaný znak. V případě praktických výpočtů této práce je velikost abecedy znaků 88 a počet uvažovaných znaků 256, nebo 512.

Konverze surového textu do maticové podoby probíhá sekvenčně, znak po znaku buď v nezměněném, nebo reverzním pořadí. Sledované znaky jsou uloženy v jednom řetězci znaků, který je nazýván abecedou znaků. Pokud se aktuálně převáděný znak vyskytuje v abecedě znaků, je zjištěna jeho číselná pozice v této abecedě. Ta představuje číslo sloupce, do kterého má být zapsána číselná hodnota 1. Číslo řádku je dáno pozicí aktuálně převáděného znaku v textu. Stejný způsob konverze surového textu byl použit také v práci [22]. Příklad konverze je znázorněn na obrázku 4.1. Zde je abeceda sledovaných znaků (znázorněna modrými písmeny) `a, b, c, d, e, f, g, h`. Probíhá převod slova `bbafgdh` do maticové podoby. Nad každým znakem převáděného slova je červeně vyznačeno jeho pořadí a dále je toto číslo přiřazeno vedle řádků matice. V matici je zapsání číselné hodnoty 1 znázorněno černě vybarveným prvkem. V levé části obrázku je zobrazen převod probíhající v klasickém pořadí a vpravo převod probíhající reverzně.

4.2.2 Příprava textových dat pro vstup do neuronové sítě

Efektivní proces trénování neuronové sítě je docílen dávkovým zpracováním vstupních dat. Příprava vstupních textových dat probíhá následně:

1. SQL dotaz požadující pozitivní texty,
2. SQL dotaz požadující negativní texty,
3. promíchání obou množin,
4. převod do textové podoby,
5. vytvoření pole reprezentující kategorie daných textů.



Obr. 4.1: Matice vytvořená z textu

Vzhledem k obsáhlosti Yelp databáze není možné provést konverzi všech textových vzorků do maticové podoby najednou. Řešením bylo vytvoření mechanizmu pro postupný převod celé množiny dat. Zkrácený postup přípravy dat je popsán výše. Praktické výpočty byly provedeny na třech zařízeních s odlišnými hardwarovými parametry. Z tohoto důvodu byl mechanismus pro přípravu dat navržen s ohledem na jeho snadnou konfigurovatelnost. Bylo možné nastavit množství vzorků obsažených v dávce dat (nutno volit dle komplexnosti modelu neuronové sítě a dle velikosti paměti grafické karty). Počet vzorků v dávce dat je dvojnásobný oproti danému parametru. Je to dáno tím, že udává počet načítaných vzorků pro každou klasifikovanou třídu. S tím souvisí možnost volby počtu najednou v operační paměti načtených dávek dat (nutno volit dle dostupné velikosti operační paměti). Další konfigurovatelný parametr definuje maximální délku zpracovávaného textu. Mezi poslední parametry patří definice počtu vzorků použitých pro trénování, validaci či testování neuronové sítě. Přehled používaného počtu vzorků v praktických výpočtech je zobrazen v tab. 4.1. Druhý konfigurační soubor obsahuje nastavitelné parametry nutné pro přístup do MySQL databází.

4.2.3 Vrstvy LSTM / CuDNNLSTM

Keras verze 2.0.9 vydaný začátkem listopadu roku 2017 přidal podporu CuDNNLSTM vrstvy. Jedná se o cuDNN knihovnou akcelerovaný protějšek klasické LSTM vrstvy. Hlavní výhodou CuDNNLSTM vrstvy je několikanásobné zrychlení učení sítě. Mezi nejzásadnější nevýhodu patří chybějící zpětnovazební dropout a také

Tab. 4.1: Počet vzorků v množině pro každou kategorii

	YELP databáze	**Vícejazyčná databáze**		
Množina	**Anglický jazyk**	**Český jazyk**	**Německý jazyk**	**Španělský jazyk**
Trénovací	240 000	3 600		
Validační	180 000	1 200		
Testovací	180 000	1 200		

omezenější možnosti nastavení vrstvy. Dle zveřejněných specifikací by mělo použití CuDNNLSTM vrstvy znamenat tří až šestinásobné zrychlení trénovacího procesu. Tabulka 4.2 rychlostní rozdíl při zpracování 6 000 vzorků z každé klasifikované kategorie pomocí LSTM a CuDNNLSTM vrstvy. Navržený model využívající cuDNN knihovnou akcelerovanou LSTM vrstvu byl více než dvakrát rychlejší oproti modelu využívajícího klasickou LSTM vrstvu. Simulace probíhala na serverovém zařízení viz tab.3.1.

Tab. 4.2: Čas potřebný pro zpracování 12 000 vzorků

Vrstva	**Čas [s]**
LSTM	695,5
CuDNNLSTM	343,7

4.3 Výsledky klasifikace pozitivních a negativních textů

V této části jsou prezentovány dosažené výsledky klasifikace textu pomocí navržených modelů rekurentních neuronových sítí.

4.3.1 Dosažené výsledky při omezení textů na 256 znaků

Prezentovaní výsledky byly dosaženy pomocí modelu A.1.

Během trénování modelů obsahující LSTM vrstvy docházelo k výskytu tzv. jevu vymývání, nebo explodování gradientu. Jedná se o problém postihující zejména modely obsahující LSTM vrstvy. Jeho eliminace byla docílena adaptací parametrů Adam optimalizátoru, především změnou míry učení a parametru ϵ.

Trénovací a testovací texty byly omezeny na 256 znaků. Model je složen z rekurentních i konvolučních vrstev. Konvoluční vrstvy typu 2D vyžadují přidání nové

dimenze, tzv. kanálu k vstupní textové matici. Dále dochází k rozvětvení modelu. Každá větev je složena z konvolučních vrstev následovaných tzv. Reshape vrstvou, kde dochází k přeuspořádání tenzoru. Změna dimenze, respektive vypuštění rozměru představující kanál, je nutná z hlediska očekávaného tvaru vstupního tenzoru pro CuDNNLSTM vrstvy. Mezi nimi se nachází tzv. dropout vrstva sloužící k omezení přetrénování sítě. Druhá větev se oproti první liší pouze v počtu hledaných příznaků konvoluční vrstvou. Ke spojení větví dochází zprůměrováním jejich výstupních hodnot. Následuje trojice plně propojených vrstev doplněná dropout vrstvami. Poslední z trojice plně propojených vrstev je tzv. softmax vrstva, jejíž výstupem je dvojice čísel reprezentující pravděpodobnost příslušnosti vstupních dat do klasifikovaných kategorií.

V rámci první sady výsledků 4.3 byl zkoumán vliv pořadí textu na klasifikační úspěšnost. Parametry použitého modelu nalezneme v přehledu níže. Dle výsledků obsažených v tabulce lze pozorovat průměrově vyšší procentuální úspěšnost při zachování původního pořadí textu, než v případě reverzního pořadí. Průměrně byla úspěšnost klasifikace při zachování pořadí textů oproti obrácenému pořadí vyšší o necelá 4 procenta.

Anglicky psané texty dokázal model klasifikovat s poměrně vysokou úspěšností lehce převyšující 91%. Ostatní jazyky byly klasifikovány s nižší procentuální úspěšností. Je nutné podotknout, že tento jev je dán především nízkým počtem textových vzorků pro český, německý a španělský jazyk viz 4.1. Nicméně s ohledem na nízký počet vzorků ve zmíněných třech jazycích lze dosažené výsledky považovat za poměrně úspěšné. Výjimkou je klasifikace německy psaných textů, kde se úspěšnost pohybuje pouze kolem 51%. Výrazný pokles ve srovnání s ostatními jazyky může být dán charakterem textových dat. Dle obrázku 4.2 obsahujícího grafické průběhy úspěšnosti a chybovosti testovaných modelů pro nezměněné (1. model) a reverzní (2. model) pořadí anglicky psaných textových dat lze usoudit, že došlo k přetrénování sítě. Pro redukci nežádoucího jevu by bylo vhodné použít místo CuDNNLSTM vrstvy klasickou LSTM vrstvu umožnující aplikaci rekurentního dopoutu. Mezi další uvažované techniky prevence přetrénování by bylo možné použít augmentaci textových dat, zmenšení složitosti navržených modelů či zvýšení podílu vynechaných spojů mezi jednotlivými plně propojenými i LSTM vrstvami. Konfúzní matice pro všechny klasifikované jazyky lze nalézt v tabulkách 4.4, 4.5, 4.6 a 4.7. Grafické průběhy pro zbývající tři jazyky lze nalézt v přílohách B.1, B.2 a B.3.

Parametry společné pro všechny modely:

- Počet iterací: **30**
- Funkce reprezentující míru chyby sítě: **Křížová entropie**
- Optimalizační algoritmus: **Adam**
- Předčasné ukončení učení:

Tab. 4.3: Klasifikační úspěšnosti dle pořadí textů (výsledky jsou v %)

–	**AJ**		**ČJ**		**NJ**		**ŠJ**		**Průměr**	
Pořadí textu	V	T	V	T	V	T	V	T	V	T
Nezměněné	91,00	91,09	73,03	72,66	51,93	51,25	70,28	67,33	71,56	70,59
Reverzní	90,85	90,84	72,74	74,22	52,01	51,54	50,21	50,25	66,46	66,72

Tab. 4.4: Konfúzní matice testovacích 256 znakových anglických textů

		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	164 343	15 657	91,30	163 282	16 718	90,71
	Negativní	16 419	163 581	90,87	16 258	163 742	90,96
Preciznost (%)		**90,92**	**91,26**		**90,94**	**90,74**	
Úspěšnost (%)		**91,09**			**90,84**		

Tab. 4.5: Konfúzní matice testovacích 256 znakových českých textů

		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	894	306	74,50	945	255	78,75
	Negativní	350	850	70,83	364	836	69,67
Preciznost (%)		**71,86**	**73,53**		**72,19**	**76,63**	
Úspěšnost (%)		**72,66**			**74,22**		

Tab. 4.6: Konfúzní matice německých 256 znakových německých textů

		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	511	689	42,58	504	696	42,00
	Negativní	481	719	59,91	467	733	61,08
Preciznost (%)		**51,51**	**51,06**		**51,91**	**51,29**	
Úspěšnost (%)		**51,25**			**51,54**		

Tab. 4.7: Konfúzní matice testovacích 256 znakových španělských textů

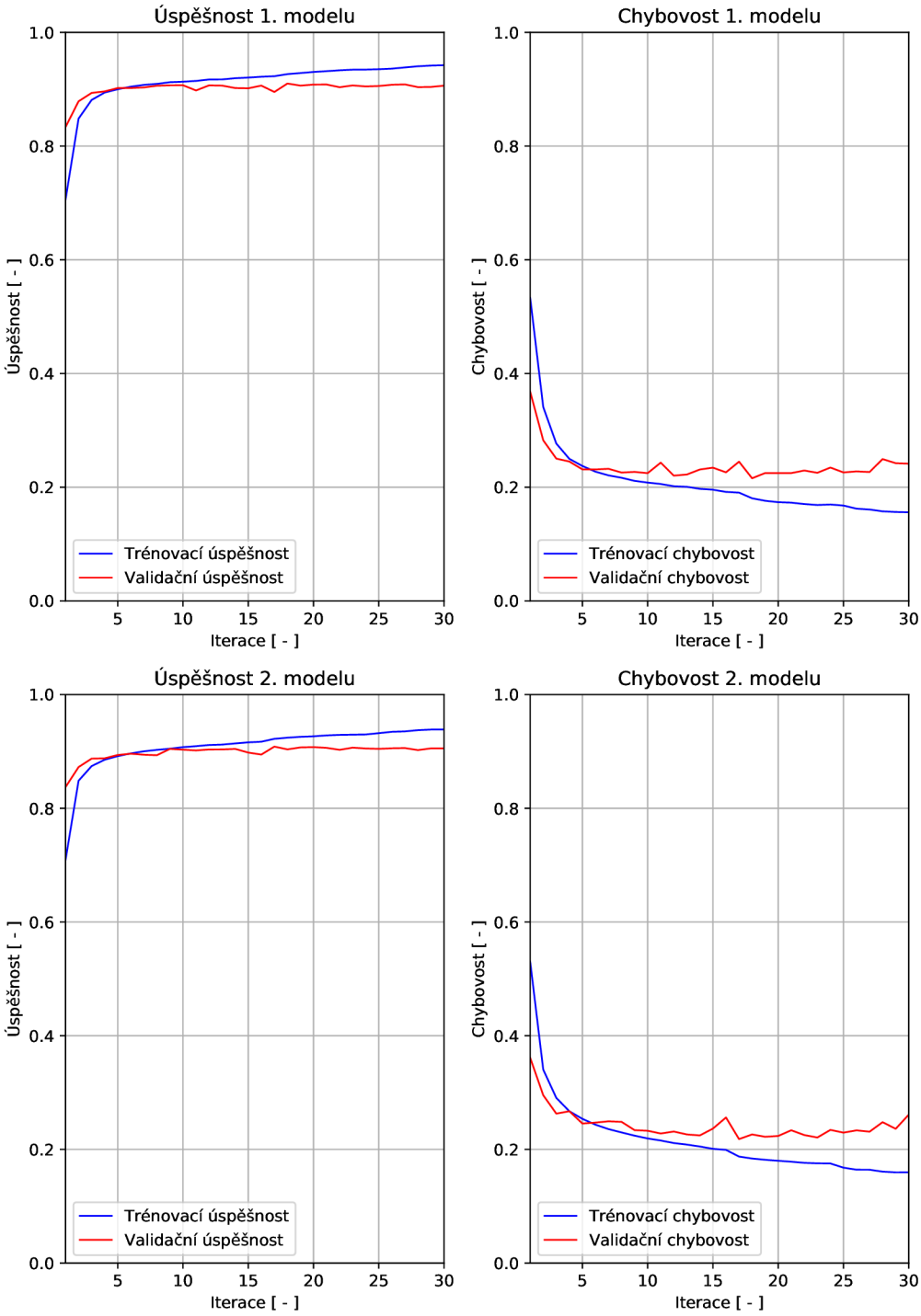
		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	863	337	71,92	33	1 167	2,75
	Negativní	447	753	62,75	27	1 173	97,75
Preciznost (%)		**65,88**	**69,08**		**55,00**	**50,13**	
Úspěšnost (%)		**67,33**			**50,25**		

  - Monitorovací funkce: **validační míra chyby sítě**
  - Doba vyčkávání: **33 iterací**
- Snížení míry učení:

  - Monitorovací funkce: **úspěšnost validace**
  - Doba vyčkávání: **6 iterací**
  - Faktor snížení: **0,333**
  - Režim: **automatický**

Přehled parametrů modelů:

- První větev:
  - Konvoluční vrstvy:
    - První konvoluční vrstva - počet hledaných příznaků: **400**, velikost jádra: **2, 88**
    - Druhá konvoluční vrstva - počet hledaných příznaků: **200**, velikost jádra: **2, 1**
  - CuDNNLSTM vrstvy:
    - První CuDNNLSTM vrstva - počet neuronů: **512**, vrácení celé sekvence **Ano**
    - Druhá CuDNNLSTM vrstva - počet neuronů: **512**, vrácení celé sekvence **Ne**
- Druhá větev:
  - Konvoluční vrstvy:
    - První konvoluční vrstva - počet hledaných příznaků: **600**, velikost jádra: **2, 88**
    - Druhá konvoluční vrstva - počet hledaných příznaků: **300**, velikost jádra: **2, 1**
  - CuDNNLSTM vrstvy:
    - První CuDNNLSTM vrstva - počet neuronů: **512**, vrácení celé sekvence **Ano**
    - Druhá CuDNNLSTM vrstva - počet neuronů: **512**, vrácení celé sekvence **Ne**
- Společná část:
  - Dropout1, Dropout2, Dropout3:
    - Podíl vynechaných spojení: **0,1**
  - Dense1, Dense2:
    - Počet neuronů: **256**
    - Aktivační funkce: **relu**
  - Dense3:
    - Počet neuronů: **2**
    - Aktivační funkce: **softmax**



Obr. 4.2: Průběh úspěšnosti a chybovosti modelů pro anglický jazyk v závislosti na iteraci sítě

4.4 Dosažené výsledky při omezení textů na 512 znaků

Prezentovaní výsledky byly dosaženy pomocí modelu A.2.

Dosažená úspěšnost klasifikace u anglicky psaných textů převyšující 91% by se dala považovat za úspěch. Pomocí úpravy modelu A.1 a odladěním parametrů by mohla být úspěšnost pravděpodobně navýšena i při zachování uvažované délky 256 znaků. Zpřesnění klasifikace lze snadněji docílit prodloužením uvažované délky vstupních textových dat na 512 znaků. Pro tyto potřeby došlo ke dvojnásobnému zvětšení počtu hledaných příznaků konvolučními vrstvami a také k dvojnásobnému nárůstu počtu neuronů rekurentních i plně propojených vrstev. Bez otestování původního modelu pro 512 znakový vstup nelze zpětně tento přístup vyhodnotit za správný, nebo nesprávný. Nicméně takovýto růst modelu a délky vstupních dat zapříčinil prodloužení celého procesu trénování i testování anglicky psaných testů přibližně na 110 hodin.

Získané výsledky pomocí modelu A.2 jsou zaznamenány v tabulce 4.8. S ohledem na vyšší klasifikační úspěšnost u nezměněného pořadí textu z první sady výsledků nebylo v této části obrácené pořadí testováno. Model s hodnotou dropout 0,1 klasifikoval anglický jazyk s vysokou úspěšností 93,63%. Úspěšnost klasifikace se oproti výsledkům z první sady zvýšila také u německého jazyka a to o 1,48% na 52,75%. Český jazyk zaznamenal v klasifikaci mírný pokles, španělský dokonce rapidní pokles o 13,42% na 53,90%. Dle grafů 4.3 lze pozorovat, že došlo k poměrně silnému přetrénování. Proto došlo ke zvýšení podílu vynechaných spojů z hodnoty 0,1 na 0,15 (2.model). Opatření mělo za cíl zmenšení přetrénování. Po jeho aplikaci byla síť stále silně přetrénována, nicméně došlo k mírnému zvýšení klasifikační úspěšnosti u anglického a českého jazyka. Místo stálého zvyšování hodnoty podílu vynechaných spojů se nabízí možnost nahradit CuDNNLSTM vrstvy klasickými LSTM vrstvami. Výsledkem by bylo citelné prodloužení trénovacího a testovacího procesu z 110 hodin na přibližně 222 hodin. Nízká úspěšnost klasifikování ostatních tří jazyků může být vysvětlena příliš malým počtem trénovacích vzorků a pro tak malý počet příliš komplexním modelem. Konfúzní matice pro všechny klasifikované jazyky lze nalézt v tabulkách 4.9, 4.10, 4.11 a 4.12. Grafické průběhy pro zbývající tři jazyky lze nalézt v přílohách B.4, B.5 a B.6.

Tab. 4.8: Klasifikační úspěšnost dle hodnoty dropout (výsledky jsou v %)

–	**AJ**		**ČJ**		**NJ**		**ŠJ**		**Průměr**	
Dropout	V	T	V	T	V	T	V	T	V	T
0,1	93,57	93,63	69,70	70,32	53,49	52,75	52,79	53,90	67,39	67,65
0,15	93,60	93,64	70,40	70,73	53,45	53,20	53,12	53,45	67,64	67,76

Tab. 4.9: Konfúzní matice testovacích 512 znakových anglických textů

		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	167 802	12 198	93,22	167 962	12 038	93,31
	Negativní	10 734	169 266	94,04	10 858	169 142	93,97
Preciznost (%)		**93,99**	**93,28**		**93,93**	**93,56**	
Úspěšnost (%)		**93,63**			**93,64**		

Tab. 4.10: Konfúzní matice testovacích 512 znakových českých textů

		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	743	447	61,91	859	341	71,59
	Negativní	255	945	78,73	362	838	69,85
Preciznost (%)		**74,45**	**67,89**		**70,35**	**71,08**	
Úspěšnost (%)		**70,32**			**70,73**		

Tab. 4.11: Konfúzní matice německých 512 znakových německých textů

		Predikovaná třída					
		Dropout 0,1			**Dropout 0,15**		
		Pozitivní	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
Skutečná třída	**Pozitivní**	714	488	59,50	732	468	61,00
	Negativní	648	552	46,00	655	545	45,41
Preciznost (%)		**52,42**	**53,08**		**52,78**	**53,80**	
Úspěšnost (%)		**52,75**			**53,21**		

Parametry společné pro všechny modely:

- Počet iterací: **30**
- Funkce reprezentující míru chyby sítě: **Křížová entropie**
- Optimalizační algoritmus: **Adam**
- Předčasné ukončení učení:
  - Monitorovací funkce: **validační míra chyby sítě**
  - Doba vyčkávání: **33 iterací**
- Snížení míry učení:

Tab. 4.12: Konfúzní matice testovacích 512 znakových španělských textů

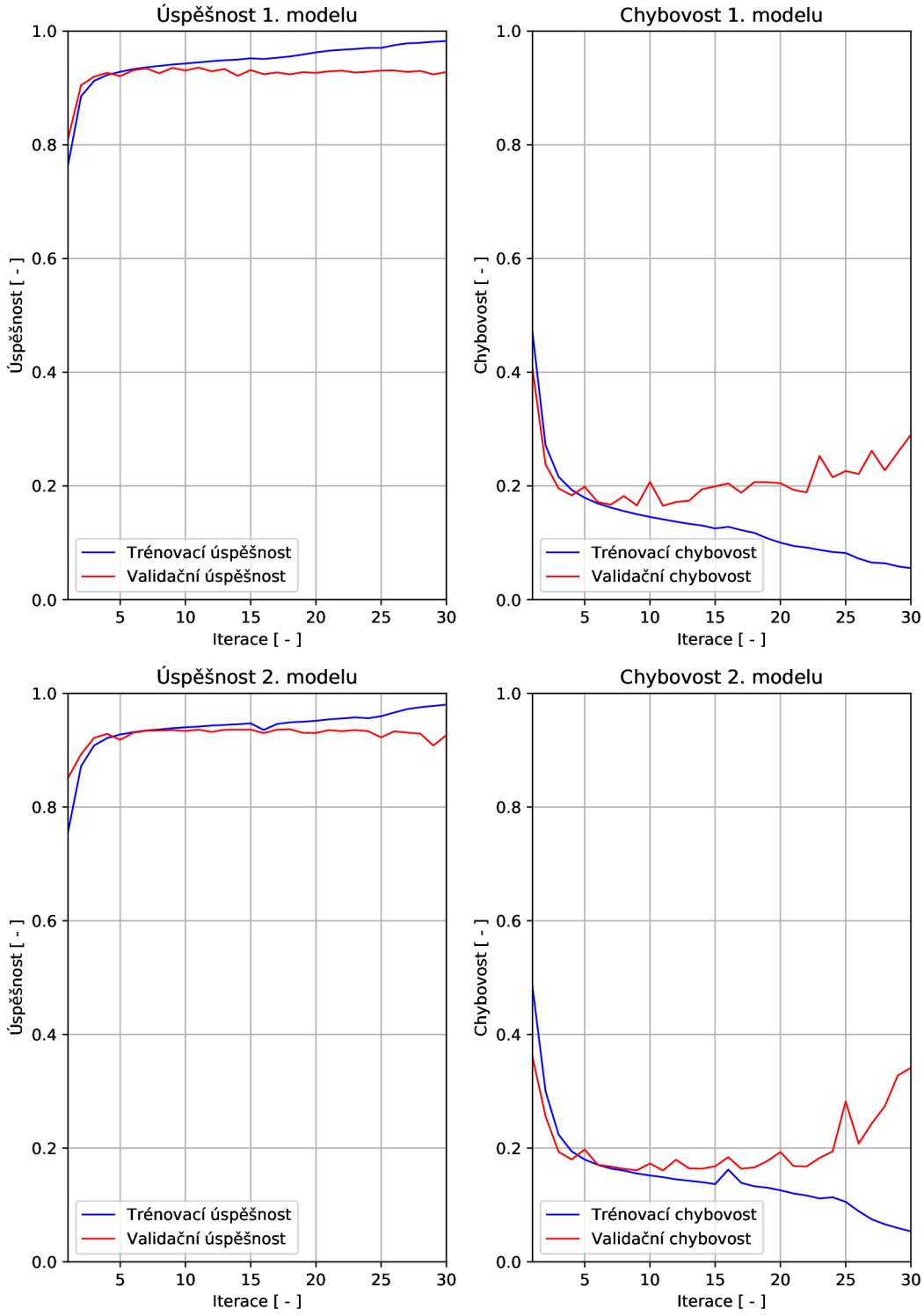
		Predikovaná třída					
		Dropout 0,1			Dropout 0,15		
Skutečná třída		**Pozitivní**	**Negativní**	**Senzitivita (%)**	**Pozitivní**	**Negativní**	**Senzitivita (%)**
	Pozitivní	430	770	35,82	391	809	32,58
	Negativní	336	864	72,00	308	892	74,33
Preciznost (%)		**56,14**	**52,88**		**55,94**	**52,43**	
Úspěšnost (%)		**53,90**			**53,45**		

  - Monitorovací funkce: **úspěšnost validace**
  - Doba vyčkávání: **6 iterací**
  - Faktor snížení: **0,333**
  - Režim: **automatický**

Přehled parametrů modelů:

- První větev:
  - Konvoluční vrstvy:
    - První konvoluční vrstva - počet hledaných příznaků: **800**, velikost jádra: **2, 88**
    - Druhá konvoluční vrstva - počet hledaných příznaků: **400**, velikost jádra: **2, 1**
  - CuDNNLSTM vrstvy:
    - První CuDNNLSTM vrstva - počet neuronů: **1 024**, vrácení celé sekvence **Ano**
    - Druhá CuDNNLSTM vrstva - počet neuronů: **1 024**, vrácení celé sekvence **Ne**
- Druhá větev:
  - Konvoluční vrstvy:
    - První konvoluční vrstva - počet hledaných příznaků: **1 200**, velikost jádra: **2, 88**
    - Druhá konvoluční vrstva - počet hledaných příznaků: **600**, velikost jádra: **2, 1**
  - CuDNNLSTM vrstvy:
    - První CuDNNLSTM vrstva - počet neuronů: **1 024**, vrácení celé sekvence **Ano**
    - Druhá CuDNNLSTM vrstva - počet neuronů: **1 024**, vrácení celé sekvence **Ne**
- Společná část:
  - Dropout1, Dropout2, Dropout3:
    - Podíl vynechaných spojení: **0,1** (1. model) , **0,15** (2. model)
  - Dense1, Dense2:

    - Počet neuronů: **512**
    - Aktivační funkce: **relu**
- Dense3:
    - Počet neuronů: **2**
    - Aktivační funkce: **softmax**



Obr. 4.3: Průběh úspěšnosti a chybovosti modelů pro anglický jazyk v závislosti na iteraci sítě

5 ZÁVĚR

Jedním z cílů diplomové práce bylo nastudování informací o neuronových sítích a současných metodách analýzy textů. Dalším cílem bylo nastudované poznatky aplikovat při návrhu vlastní struktury hluboké neuronové sítě dle zadání vedoucího a prezentovat dosažené výsledky pomocí navržených modelů neuronových sítí.

Praktická část semestrální práce byla řešena pomocí programovacího jazyka Python. Pro návrh a práci s modely hlubokých neuronových sítí bylo využito vysokoúrovňového API Keras verze 2.1.5, které využívalo knihovnu pro numerické výpočty s použitím diagramu datových toků TensorFlow. Samotné výpočty byly provedeny na grafických procesorech. Usnadnění experimentů s neuronovými sítěmi zajišťovala knihovna Kex, která byla přizpůsobena pro individuální potřeby této práce.

Původní předpoklad možného využití čistě rekurentních neuronových sítí pro klasifikaci textu na úrovni po sobě jdoucích jednotlivých znaků zatím nebyl potvrzen. Během trénovacího procesu pravidelně docházelo k problému vymývání nebo explodování gradientu. Jeho odstranění bylo docíleno adaptací parametrů optimalizátoru. Výsledkem bylo stagnování průběžné klasifikační úspěšnosti během trénovacího procesu pohybující se okolo 60%. Řešením zmíněného problému bylo předřazení konvolučních vrstev před rekurentní vrstvy. Pro proces trénování a testování modelů byly k dispozici datové množiny obsahující anglicky, česky, německy a španělsky psané texty. Nejvíce vzorků, 600 000 z každé klasifikované kategorie, obsahovala množina anglicky psaných textů. Zdrojem dat byla Yelp databáze. Zbylé jazykové množiny obsahovaly pouze 6 000 vzorků.

V rámci prezentace dosažených klasifikačních úspěšností jsou představeny dvě sady výsledků. První z nich prezentuje dosažené úspěšnosti klasifikace pro textová data omezená na 256 znaků vstupující do modelu neuronové sítě v nezměněném nebo opačném pořadí. Výsledky přisuzují o necelá 4% vyšší průměrnou klasifikační úspěšnost vstupním datům bez změny pořadí textů. V případě rozsáhlé datové množiny dokáže navržená neuronová síť klasifikovat daný vstup do pozitivní nebo negativní kategorie s vysokou úspěšností lehce převyšující 91%, i při poměrně nízké délce vstupního textu. Česky a španělsky psané texty byly klasifikovány s úspěšností pohybující se kolem 70%. Neúspěchem skončila klasifikace německy psaných textů. Úspěšnost byla pouze 51,27%. Trénování i testování každého modelu na Yelp databázi trvalo více jak 26 hodin.

Druhá výsledková sada obsahuje klasifikační úspěšnosti při 512 znakové délce vstupního textu. Došlo také k dvojnásobnému „zvětšení“ navrženého modelu. Uvedené kroky vedly k prodloužení trénování i testování z 26 na 110 hodin, proto bylo testováno pouze úspěšněji klasifikované pořadí textů z první sady výsledků. Modely vykazovaly silné přetrénování, i přes to byly anglicky psané texty klasifikovány s vy-

sokou úspěšností činící 93,64%. Xiang Zhang a Yann LeCun dosháli v práci [22] 94,50% úspěšnosti při uvažované délce textů 1014 znaků. Česky a španělsky psané texty byly oproti první sadě výsledků klasifikovány s větší chybovostí. U německy psaných textů došlo k mírnému zvýšení klasifikační úspěšnosti.

Prezentované výsledky dokazují možnost klasifikace textů pouze na úrovni znaků bez jakékoli nutné znalosti příslušného jazyka. Pro účely klasifikace textů tedy lze vytvořit jazykově nezávislou strukturu neuronových sítí.

V budoucnosti je možné provést úpravu parametrů navržených modelů neuronových sítí pro odstranění nežádoucího přetrénování. Dále je možné otestovat vliv dalšího zvětšení počtu uvažovaných znaků jednotlivých textů na klasifikační úspěšnost a natrénování modelů na obsáhlejších datových množinách pro více jazyků.

LITERATURA

[1] BARTUNOV, Sergey, Dmitry KONDRASHKIN, Anton OSOKUB, Dmitry VETROV: *Breaking Sticks and Ambiguities with Adaptive Skip-gram* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://arxiv.org/abs/1502.07257>.

[2] BOGDAN Oancea, TUDOREL Andrei, RALUCA Mariana Dragoescu *GPGPU Computing. 2014* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://arxiv.org/abs/1408.6923>.

[3] Dataversity: *A Brief History of Deep Learning* [online]. [cit. 24. 11. 2017]. Dostupné z: <http://www.dataversity.net/brief-history-deep-learning/>.

[4] HARÁR, Pavol: *KEX - Library for easier management and result representation of Keras experiments.* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://gitlab.com/paloha/kex>.

[5] Keras: *Keras: The Python Deep Learning library.* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://keras.io/>.

[6] KIROS, Ryan , Yukun ZHU, Ruslan SALAKHUTDINOV, Richard S. ZEMEL, Antonio TORRALBA, Raquel URTASUB, Sanja FIDLER: *Skip-Thought Vectors* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://arxiv.org/abs/1506.06726>.

[7] LE, V. Quoc a MIKOLOV Tomas: *Distributed Representations of Sentences and Documents* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://arxiv.org/abs/1405.4053>.

[8] LEVY, Omer a GOLDBERG Yoav: *Dependency-Based Word Embeddings* [online]. [cit. 24. 11. 2017]. Dostupné z: <http://www.aclweb.org/anthology/P14-2050>.

[9] Matematická biologie: *Umělá inteligence* [online]. [cit. 24. 11. 2017]. Dostupné z: <http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence>.

[10] NICKOLLS, John, Ian BUCK, Michael GARLAND a Kevin SKADRON. Scalable parallel programming with CUDA. *Queue* [online]. 2008, 6(2), 40- [cit. 20. 11. 2017]. DOI: 10.1145/1365490.1365500. ISSN 15427730. Dostupné z: <http://portal.acm.org/citation.cfm?doid=1365490.1365500>

[11] NOWAK, Jakub, Ahmet TASPINAR a Rafał SCHERER. LSTM Recurrent Neural Networks for Short Text and Sentiment Classification. RUTKOWSKI, Leszek, Marcin KORYTKOWSKI, Rafał SCHERER, Ryszard TADEUSIEWICZ, Lotfi A. ZADEH a Jacek M. ZURADA, ed. *Artificial Intelligence and Soft Computing* [online]. Cham: Springer International Publishing, 2017, 2017-05-24, s. 553-562 [cit. 24. 11. 2017]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-59060-8_50. ISBN 978-3-319-59059-2. Dostupné z: <http://link.springer.com/10.1007/978-3-319-59060-8_50>

[12] Nvidia: *Deep Learning in a Nutshell* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts>.

[13] Nvidia: *CUDA C Programming Guide* [online]. [cit. 24. 11. 2017]. Dostupné z: <http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>.

[14] OLAH, Christopher: *Understanding LSTM Networks* [online]. [cit. 24. 11. 2017]. Dostupné z: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

[15] POVODA, Lukáš, Radim BURGET, Jan MAŠEK, Lubomír CVRK. Automatické rozpoznávanie emócií z českého textu pomocou umelej inteligencie: *Elektrorevue - Internetový časopis (http://www.elektrorevue.cz)*, 2015, roč. 17, č. 1, s. 15-18. ISSN: 1213-1539. [cit. 24. 11. 2017].

[16] POVODA, Lukáš, Radim BURGET , Jan MAŠEK , Václav UHER , Malay Kishore DUTTA. Optimization Methods in Emotion Recognition System. *Radioengineering, 2016,* 2016, roč. 25, č. 3, s. 565-572. ISSN: 1805-9600. [cit. 24. 11. 2017].

[17] Programmer's Guide | TensorFlow. *TensorFlow* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://www.tensorflow.org/programmers_guide/>

[18] The Stanford Natural Language Processing Group: *GloVe: Global Vectors for Word Representation* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://nlp.stanford.edu/pubs/glove.pdf>.

[19] SHEN, Qianzi, Zijian WANG a Yaoru SUN. Sentiment Analysis of Movie Reviews Based on CNN-BLSTM. SHI, Zhongzhi, Ben GOERTZEL a Jiali FENG, ed. *Intelligence Science I* [online]. Cham: Springer International Publishing, 2017, 2017-09-27, s. 164-171 [cit. 2018-05-12]. IFIP Advances in

Information and Communication Technology. DOI: 10.1007/978-3-319-68121-4_17. ISBN 978-3-319-68120-7. Dostupné z: <http://link.springer.com/10.1007/978-3-319-68121-4_17>

[20] Stanford University *Introduction to Convolutional Neural Networks. 2018* [online]. [cit. 10. 4. 2018]. Dostupné z: <http://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf>.

[21] YEUNG, Daniel S., Ian CLOETE, Daming SHI a Wing W.Y. NG. Introduction to Neural Networks. YEUNG, Daniel S., Ian CLOETE, Daming SHI a Wing W. Y. NG. *Sensitivity Analysis for Neural Networks* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 2009-10-14, s. 1-15 [cit. 24. 10. 2017]. Natural Computing Series. DOI: 10.1007/978-3-642-02532-7_1. ISBN 978-3-642-02531-0. Dostupné z: <http://link.springer.com/10.1007/978-3-642-02532-7_1>

[22] ZHANG, Xiang a LECUN, Yann: *Text understanding from scratch. arXiv preprint arXiv:1502.01710, 2015.* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://arxiv.org/abs/1502.01710>.

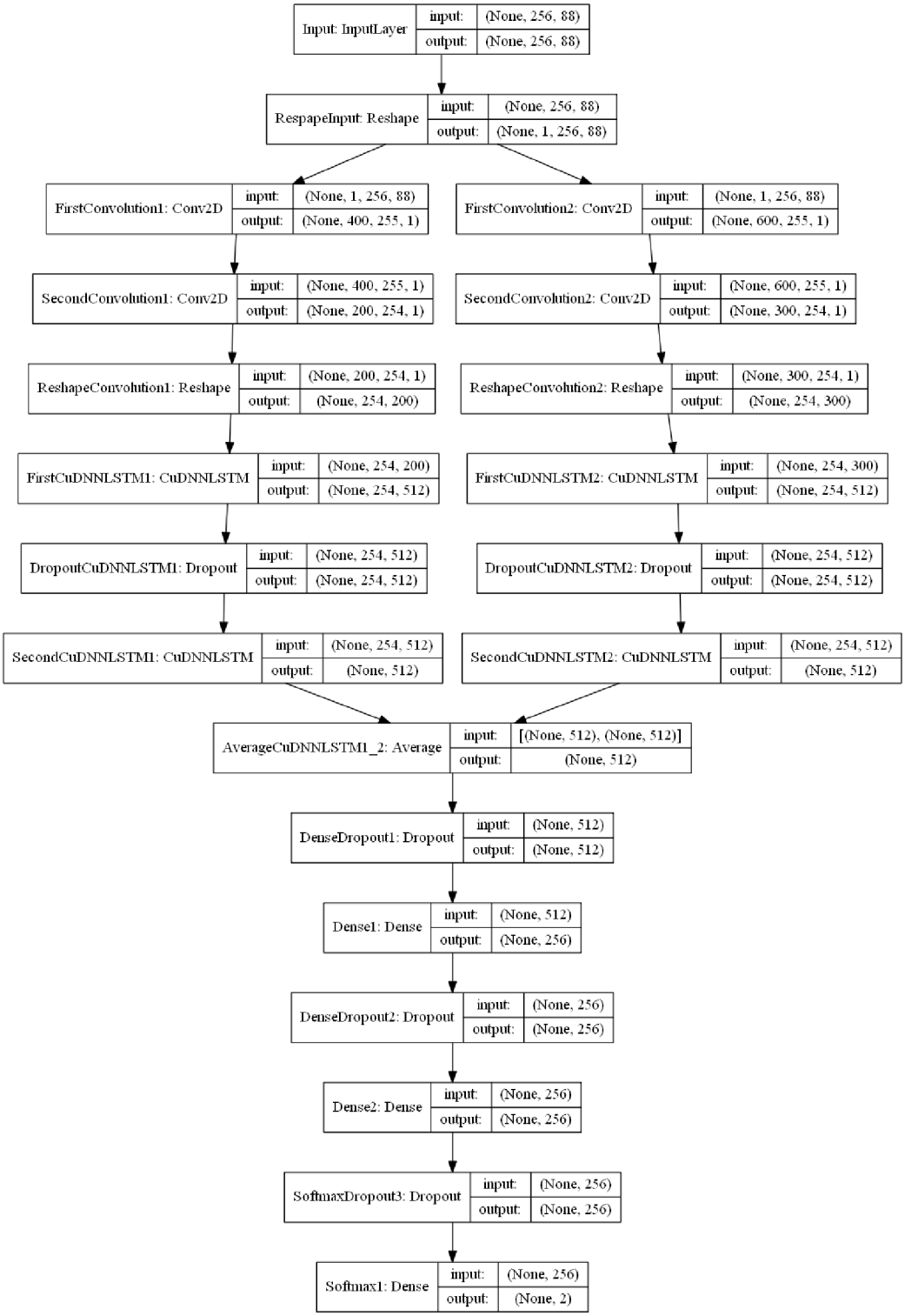
[23] ZHANG, Xiang a ZHAO, Junbo: *Character-level convolutional networks for text classification. Advances in neural information processing systems. 2015, 2015(28), 649-657.* [online]. [cit. 24. 11. 2017]. Dostupné z: <https://arxiv.org/abs/1509.01626>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

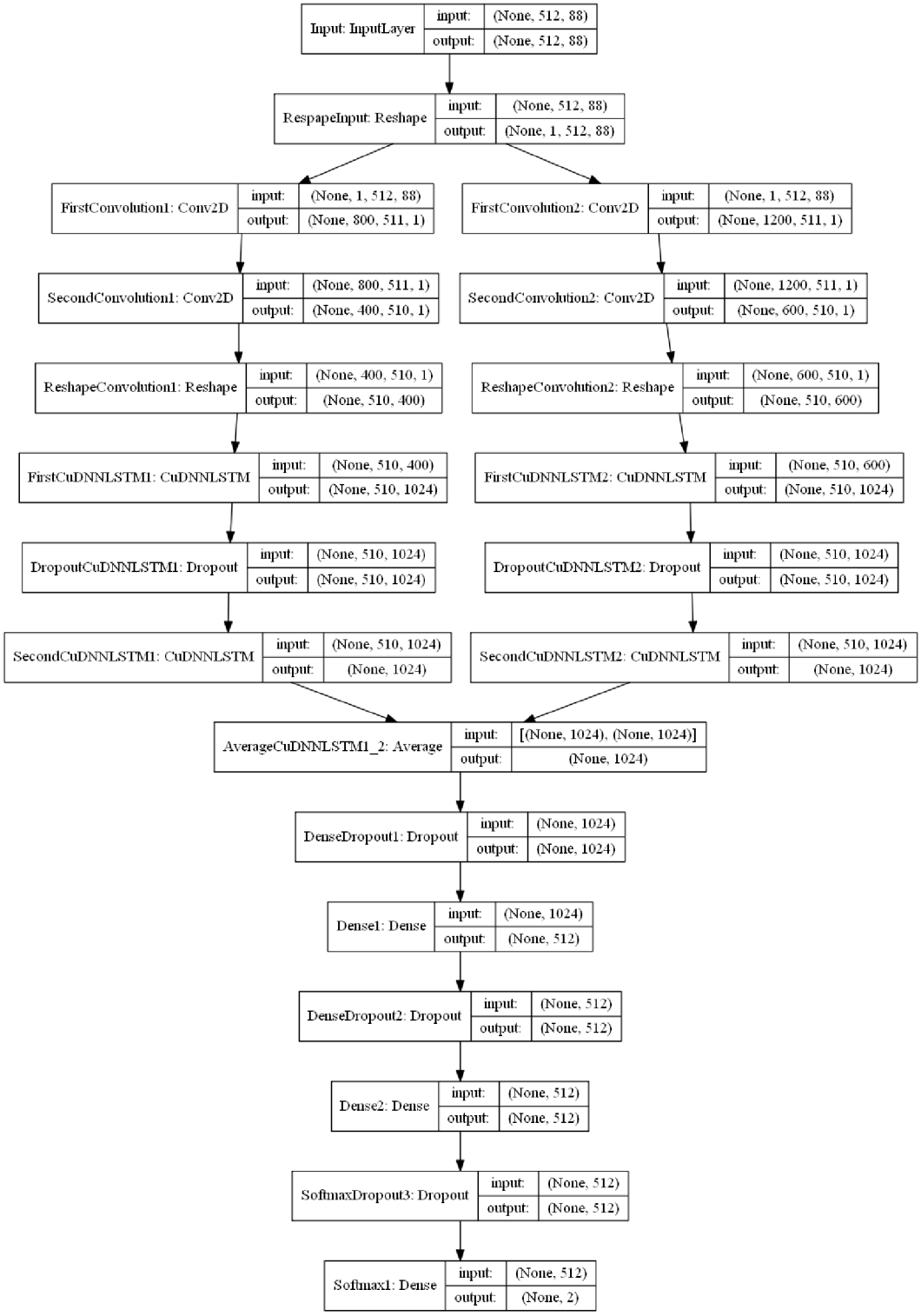
API	Application Programming Interface
CNTK	the microsoft CogNitive ToolKit
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
ECC	Error Checking and Correcting
GPGPU	General-Purpose computing on Graphics Processing Units
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
SVM	Support Vector Machines
TF-IDF	Term Frequency - Inverse Document Frequency
TFLOPS	Tera FLoating-point Operations Per Second

SEZNAM PŘÍLOH

A TESTOVANÉ MODELY



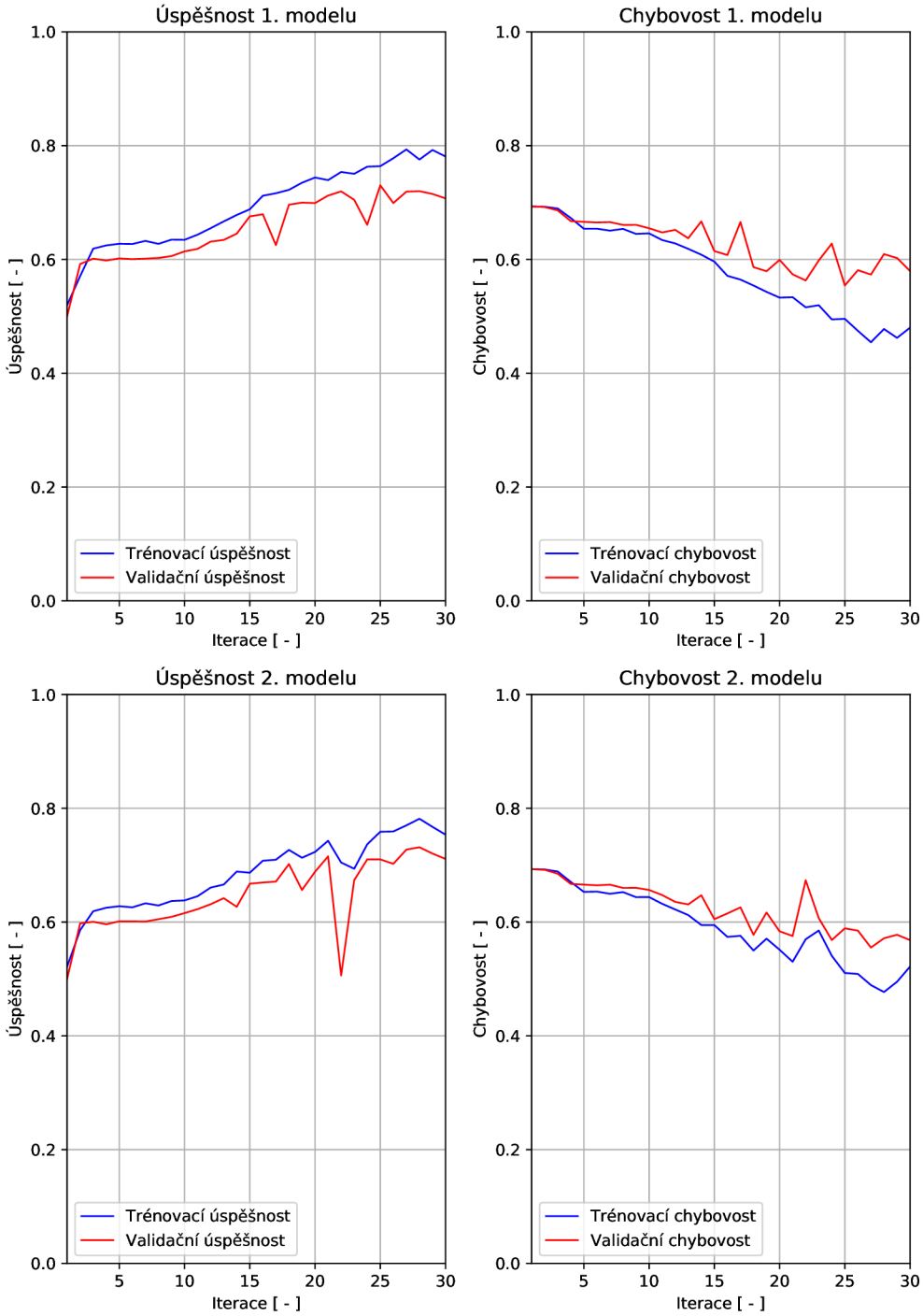
Obr. A.1: Model pro klasifikaci textů s 256 znaky



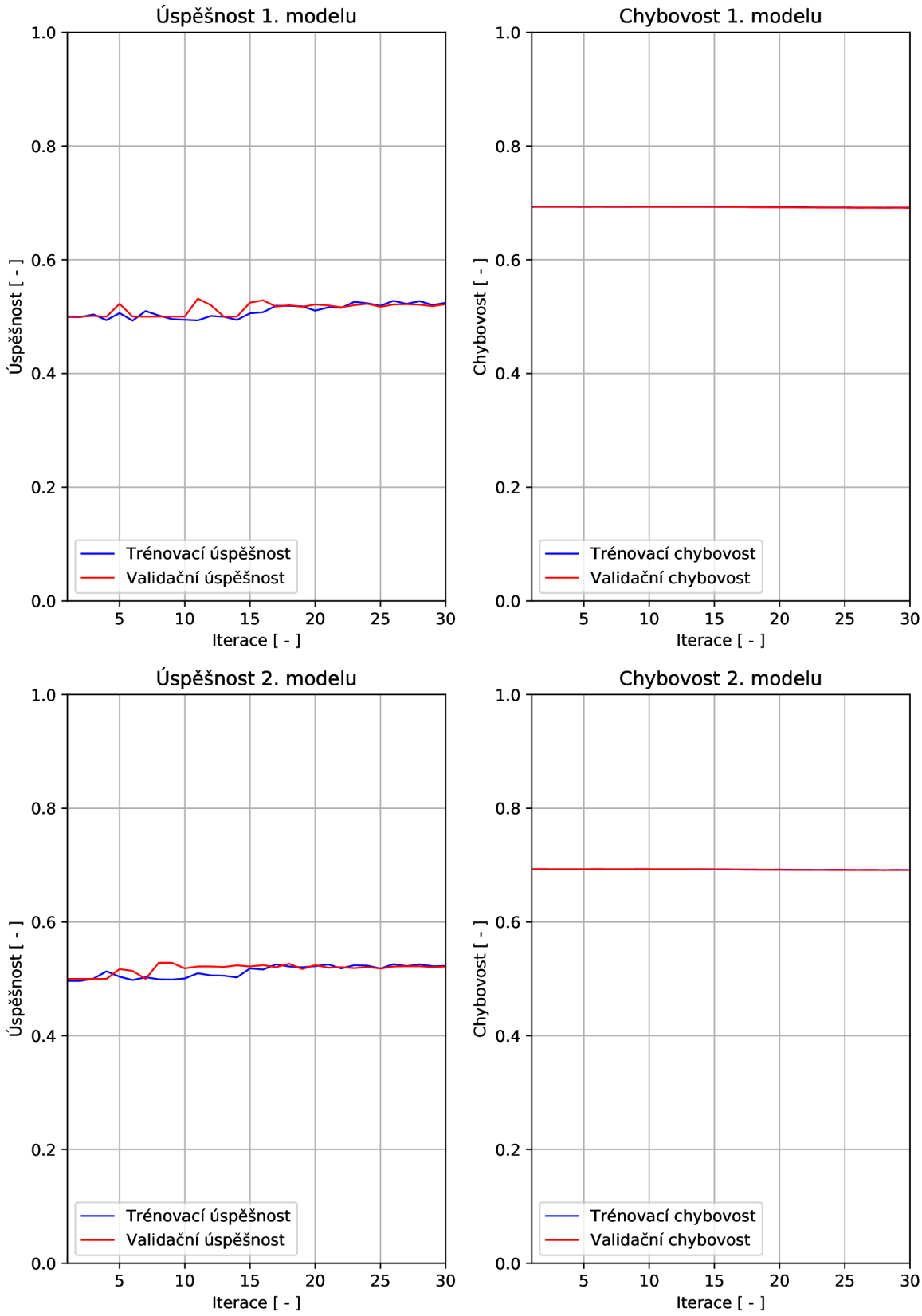
Obr. A.2: Model pro klasifikaci textů s 512 znaky

B GRAFICKÉ ZÁVISLOSTI

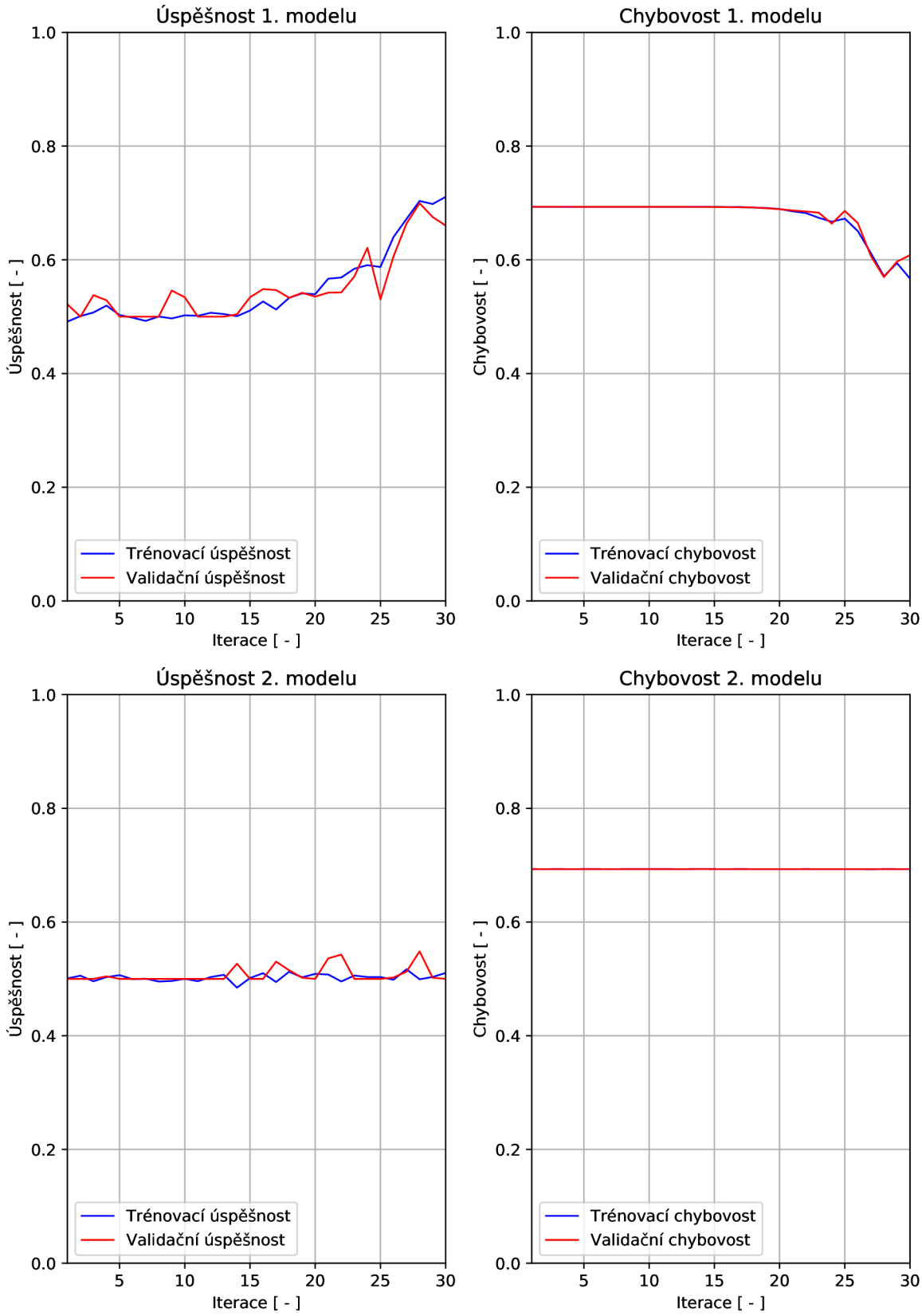
Číselné označení každého modelu odpovídá sestupnému pořadí výsledků prezentovaných v tabulce 4.3, respektive 4.8.



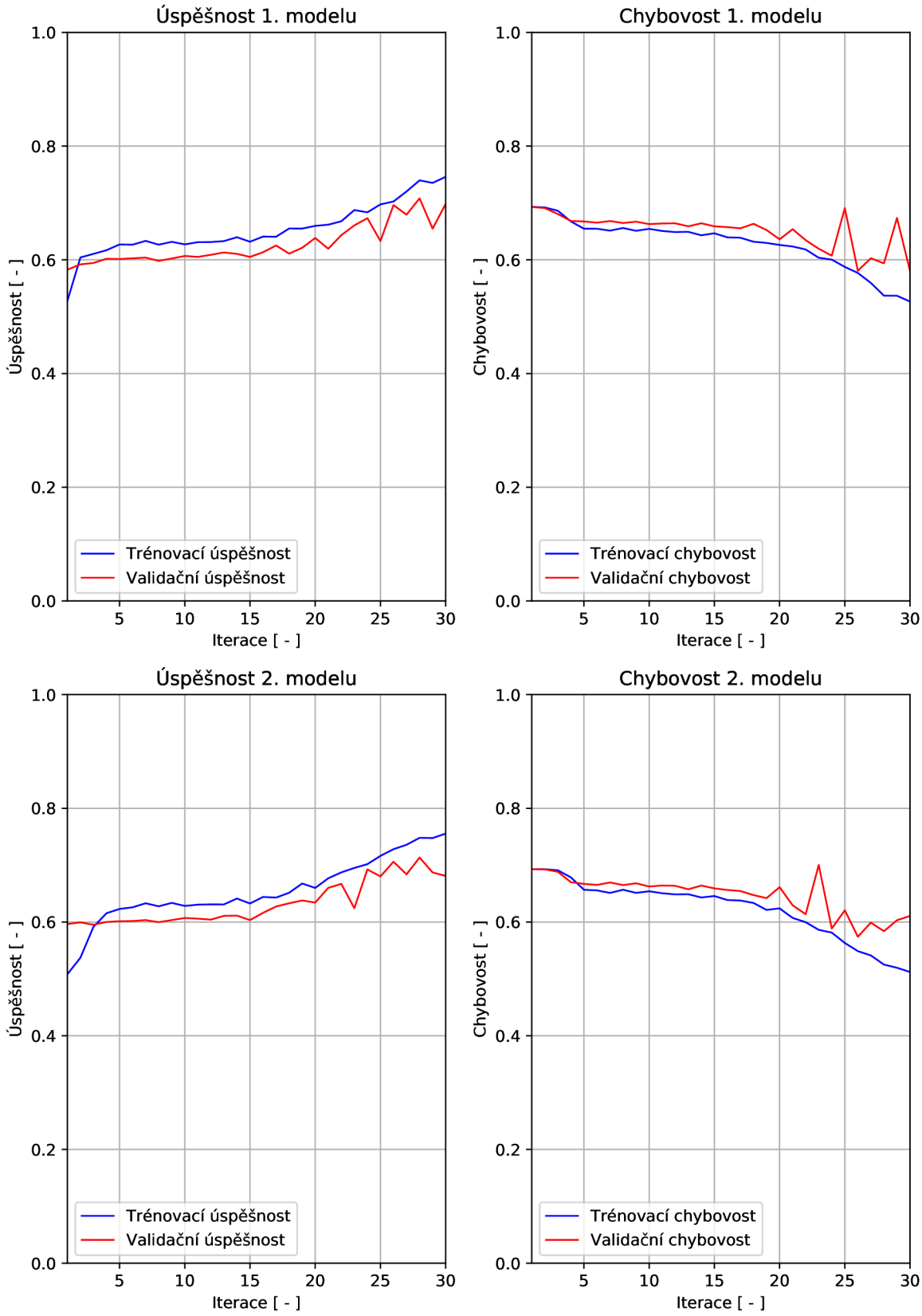
Obr. B.1: Průběh úspěšnosti a chybovosti modelů pro české texty dlouhé 256 znaků v závislosti na iteraci sítě



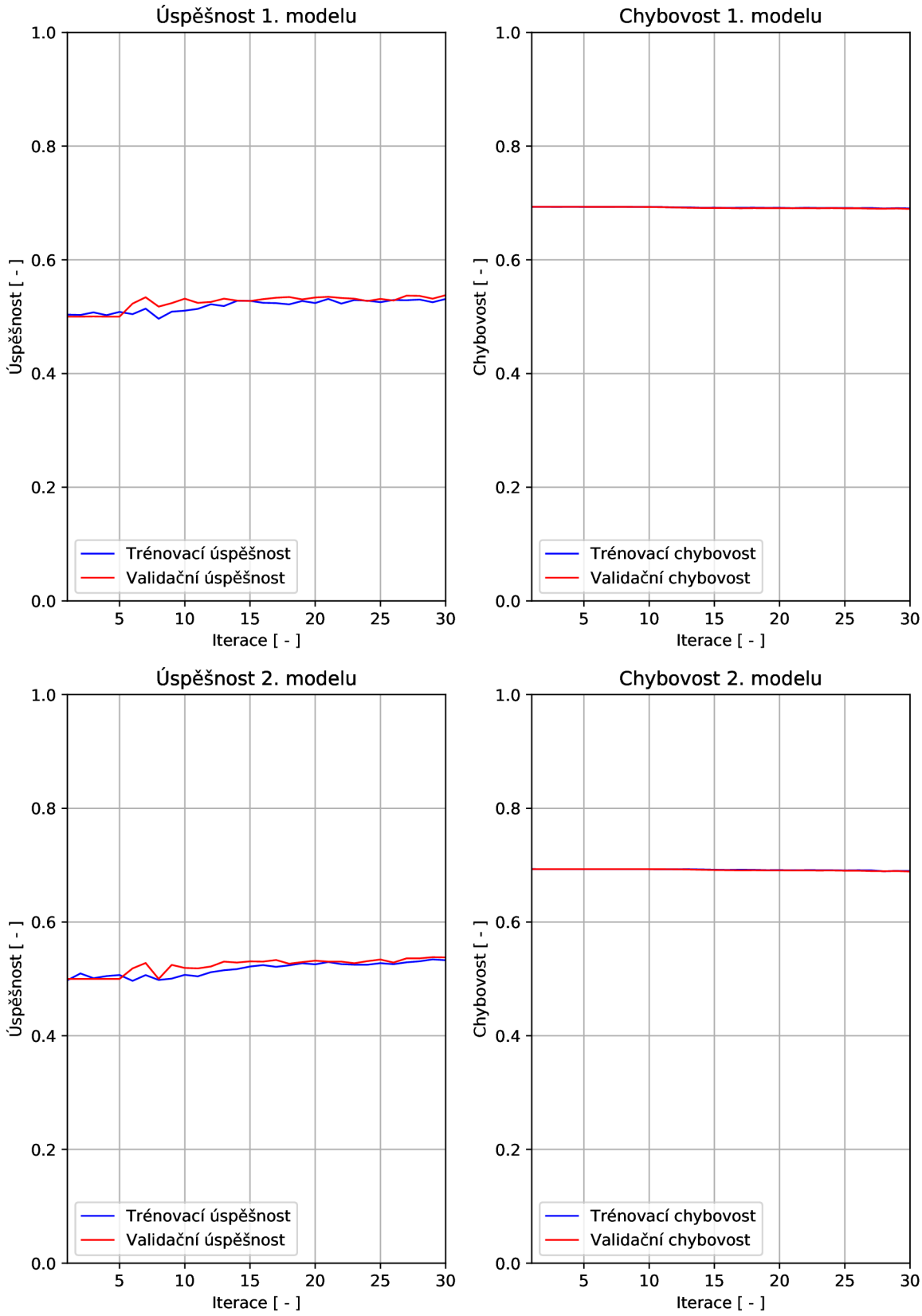
Obr. B.2: Průběh úspěšnosti a chybovosti modelů pro německé texty dlouhé 256 znaků v závislosti na iteraci sítě



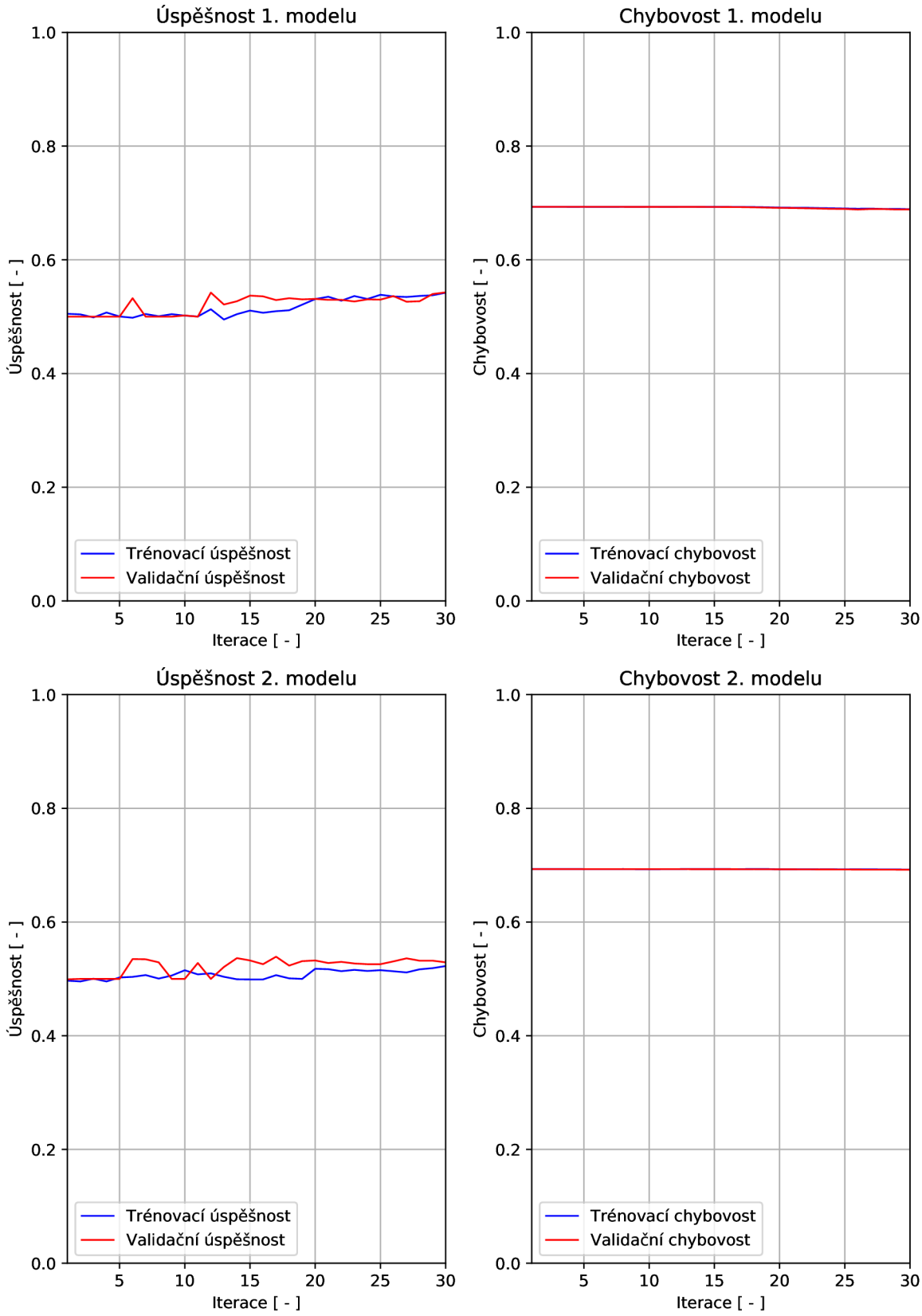
Obr. B.3: Průběh úspěšnosti a chybovosti modelů pro španělské texty dlouhé 256 znaků v závislosti na iteraci sítě



Obr. B.4: Průběh úspěšnosti a chybovosti modelů pro české texty dlouhé 512 znaků v závislosti na iteraci sítě



Obr. B.5: Průběh úspěšnosti a chybovosti modelů pro německé texty dlouhé 512 znaků v závislosti na iteraci sítě



Obr. B.6: Průběh úspěšnosti a chybovosti modelů pro španělské texty dlouhé 512 znaků v závislosti na iteraci sítě

C NÁVOD NA ZPROVOZNĚNÍ PRAKTICKÉ ČÁSTI

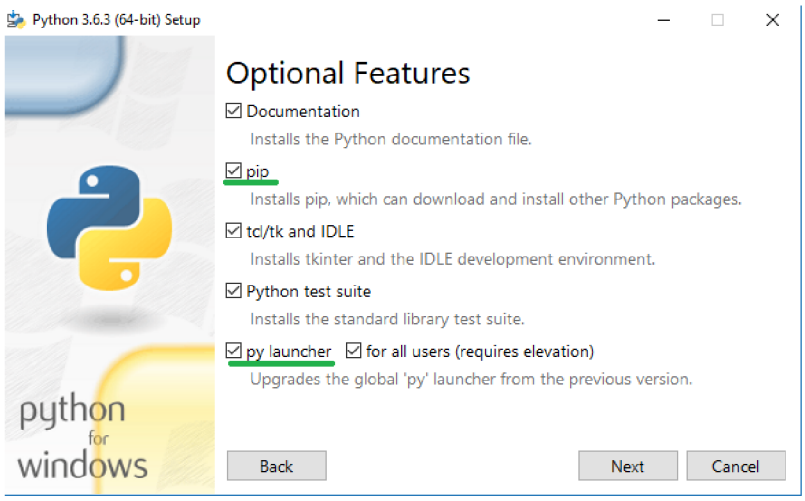
Uvedený postup pro zprovoznění trénování a testování navržených modelů rekurzivní neuronové sítě pro klasifikaci textů s použitím knihovny `KEX` je vyzkoušený na 64bitovém operačním systému Windows 10. S malými intuitivními změnami by však měl být funkční i pro ostatní bitové verze Windows 7, 8, 10. Před začátkem instalace si ověřte, zda vaše grafická karta splňuje kritéria pro práci s hlubokými neuronovými sítěmi, viz C.4.

C.1 Instalace Python

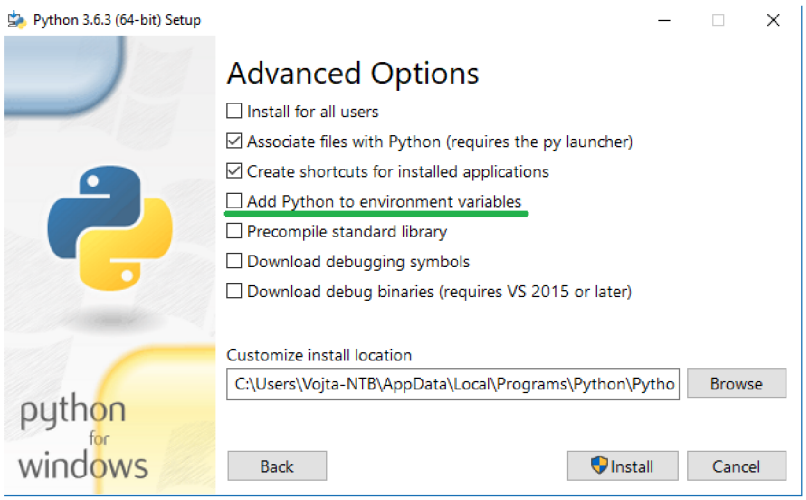
Prvním krokem je stažení instalátoru `Pythonu`, pro vaši verzi operačního systému, z jeho oficiální stránky: `https://www.python.org/downloads/windows/`. Doporučuji `Python 3.5.0 - 2015-09-13`. Funkčnost s jinými verzemi není testována.

Spusťte samotnou instalaci a v úvodním okně vyberte položku `Add Python 3.5 to PATH`, viz obr. C.1. Tím je zajištěno spouštění příkazů `python` a `pip` bez nutnosti zadání absolutní nebo relativní cesty k jejich spustitelným souborům. Zvolte některý ze způsobů instalace. Pokud se rozhodnete pro vlastní instalace, tak vyberte možnost `pip,py launcher` a v dalším kroku `Add Python to environment variables`, viz obr. C.2, respektive obr. C.3.

Obr. C.1: Úvodní okno instalátoru `Pythonu`



Obr. C.2: Vlastní instalace – volitelné funkce



Obr. C.3: Vlastní instalace – pokročilé možnosti

C.2 Instalace rozšiřujících balíčků Pythonu

Potřebné rozšiřující balíčky budou instalovány pomocí správce balíčků `pip`. Nejprve spusťte příkazový řádek s administrátorskými právy. Poté ověřte dostupnost aktualizací balíčkového manažeru pomocí příkazu:

```
python -m pip install --upgrade pip
```

V případě neaktuální verze dojde k jeho automatické aktualizaci. Okno s příkazovým řádkem nezavírejte.

Nyní stáhněte předkompilované balíčky z webové stránky: `http://www.lfd.uci.edu/~gohlke/pythonlibs/`. Pro 64bitový Windows a `Python 3.5.0` jsou to následující balíčky:

- numpy-1.13.3+mkl-cp35-cp35m-win_amd64.whl,

- scipy-0.19.1-cp35-cp35m-win_amd64.whl,
- h5py-2.7.1-cp35-cp35m-win_amd64.whl.

Instalace některých balíčků je podmíněna přítomnosti distribuovatelných součástí `Microsoft Visual C++ 2015`. Stáhnout `Visual C++ 2015` lze z: `https://www.microsoft.com/en-us/download/details.aspx?id=53587`.

Následujícím krokem je instalace stažených předkompilovaných balíčků. V otevřeném okně příkazového řádku změňte aktivní adresář na adresář obsahující stažené balíčky pomocí příkazu:

```
cd \%CESTA_K_ADRESÁŘI_SE_STAŽENÝMI_BALÍČKY\%.
```

Balíčky postupně nainstalujte následujícími příkazy:

```
pip install numpy-1.13.3+mkl-cp35-cp35m-win_amd64.whl,
pip install scipy-0.19.1-cp35-cp35m-win_amd64.whl,
pip install h5py-2.7.1-cp35-cp35m-win_amd64.whl.
```

Automaticky se nainstalují i některé jiné balíčky, které jsou nezbytné pro funkčnost instalovaných balíčků. Pokračujte v instalaci dalších potřebných balíčků. V instalačním příkazu `Tensorflow` nesmíte opomenout na specifikaci verze.

```
pip install tensorflow-gpu==1.4.0
pip install keras
pip install texttable
pip install pydot_ng
pip install matplotlib
pip install sklearn
```

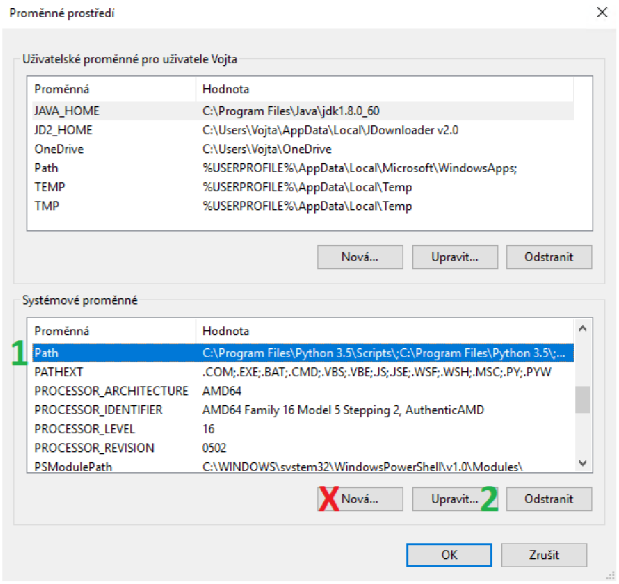
C.3 Graphviz

Prvním krokem zprovoznění je samotná instalace vizualizačního softwaru. Instalátor stáhněte z adresy: `http://www.graphviz.org/Download_windows.php`. Po dokončené instalaci je nutné přidat adresu složky `bin` softwaru `Graphviz` do systémové proměnné `Path`. Nastavení proměnného prostředí lze docílit stiskem kláves `Win+r`, spuštěním `SystemPropertiesAdvanced.exe` a následným stiskem klávesy `p`, nebo kliknutím na tlačítko `Proměnné prostředí`. V tabulce `Systémové proměnné` označte proměnnou `Path` a zvolte `Upravit`. **!Nepoužívejte možnost `Nová`. V případě duplicity jména proměnné dojde ke smazání všech současných hodnot dané proměnné!**, viz obr. C.4. Postup přidání nové hodnoty proměnné `Path` je schodný s obr. C.5.

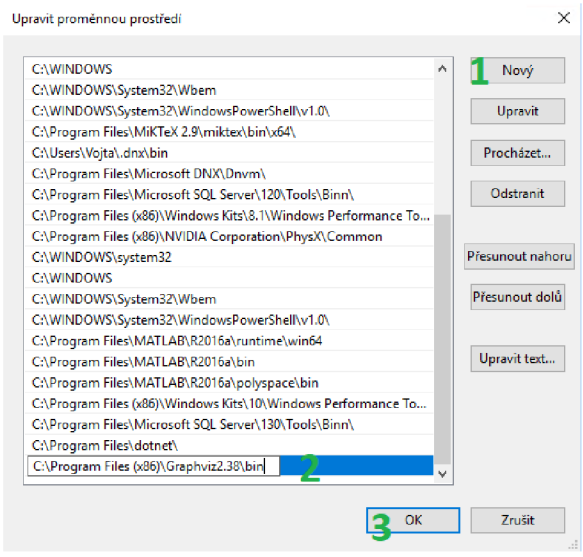
Správné nastavení je možné ověřit zadáním následujícího příkazu do příkazové řádky:

```
dot -v
```

!Při změně systémových proměnných je nutné spustit novou instanci příkazového řádku!



Obr. C.4: Tabulky `proměnného prostředí`



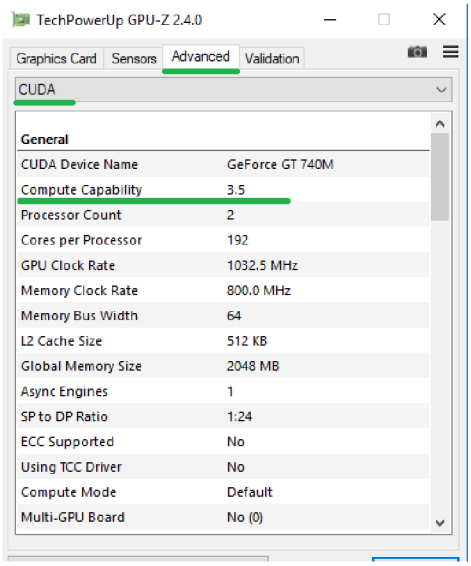
Obr. C.5: Hodnoty proměnné `Path`

C.4 Instalace Nvidia softwaru

Pro správnou funkčnost je vyžadována grafická karta od firmy Nvidia s podporou `CUDA Compute Capability 3.0` a vyšší. Zda vaše grafická karta má tuto vlastnost si lze ověřit na webu: `https://developer.nvidia.com/cuda-gpus`, nebo s pomocí programu `GPU-Z` v záložce `Advanced` → `CUDA` → `Compute Capability`, viz obr. C.6, nebo pomocí jiného diagnostického nástroje.

C.4.1 CUDA Toolkit 8.0

Instalační soubor aplikace lze stáhnout z následujících webových stránek společnosti nVidia: `https://developer.nvidia.com/cuda-80-ga2-download-archive`. Je doporučené zvolit expresní možnost instalace, neboť je bezproblémová.



Obr. C.6: Kontrola verze `Compute Capability`

C.4.2 Knihovny cuDNN v6

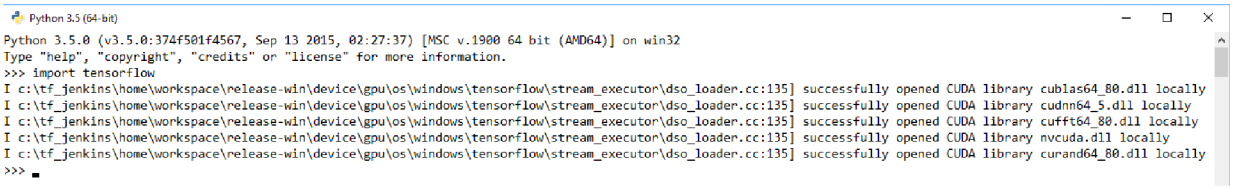
Stažení je podmíněno vytvořením nebo vlastněním `NVIDIA Developer Program` účtu. Knihovna je k dispozici po přihlášení na následující adrese: `https://developer.nvidia.com/rdp/cudnn-download`. Stáhněte verzi `cuDNN v6 pro CUDA 8.0`. Nevyžaduje instalaci, pouze je nutné stažený archiv extrahovat a následně přidat cestu k souboru `cudnn64_6.dll` do systémové proměnné `Path`. Postup je totožný jako v případě instalace softwaru `Graphviz`.

C.4.3 Ověření správnosti instalace CUDA a cuDNN

Spusťte aplikaci `Python` a zadejte příkaz:

```
import tensorflow.
```

V případě úspěšné instalace bude vypsána informace o úspěšném načtení `CUDA` knihoven, viz obr. C.7.



Obr. C.7: Kontrolní výpis načtení `CUDA` knihoven

C.5 Trénování a testování navržených modelů

Nyní zbývá pouze nainstalovat MySQL server a nahrát přiloženou vícejazyčnou databázi. Po zprovoznění MySQL serveru jste úspěšně dokončili instalaci a nastavení všech potřebných součástí pro spuštění trénování a testování navržených modelů pro klasifikaci textů s využitím knihovny `KEX`.

Spusťte příkazový řádek a změňte aktivní adresář na adresář obsahující knihovnu `KEX` a navržené modely neuronových sítí. Samotné spuštění proveďte příkazem:

```
python run.py.
```

D OBSAH PŘILOŽENÉHO CD