VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Ondřej Talár

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

REDUKCE ŠUMU AUDIONAHRÁVEK POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍTÍ
AUDIO NOISE REDUCTION USING DEEP NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Ondřej Talár
AUTHOR

VEDOUCÍ PRÁCE — Ing. Pavol Harár
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

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

Student: Bc. Ondřej Talár ***ID:*** 148143
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Redukce šumu audionahrávek pomocí hlubokých neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a natrénovat vhodný model hluboké neuronové sítě ve frameworku KERAS (Python), který bude schopen úspěšně odstraňovat šum ze vstupních zašuměných hlasových audio nahrávek. Dosažené výsledky vhodně statisticky vyhodnoťte a komentujte.

DOPORUČENÁ LITERATURA:

[1] NIELSEN, M. Neural Networks and Deep Learning [online]. 2015 [cit. 2016-09-10]. Dostupné z: http://neuralnetworksanddeeplearning.com/

[2] CHOLLET, F. Keras [online]. 2015 [cit. 2016-09-10]. Dostupné z: http://github.com/fchollet/keras

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 6.8.2017

Vedoucí práce: Ing. Pavol Harár
Konzultant:

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

ABSTRAKT

Práce se zabývá možností použití hluboké rekurentní neuronové sítě typu Long Short-Term Memory pro robustní odšumování zarušeného signálu. LSTM je v současnosti velice lákavá architektura díky své vlastnosti pamatovat si předchozí váhy, a nebo je upravovat nejen dle použitých algoritmů, ale také zkoumáním změn v sousedních buňkách. V práci je popsán výběr výchozího datasetu a použitých šumů spolu s vytvořením optimálních testovacích dat. Pro trénování sítě je zvolen framework KERAS pro jazyk Python. Jsou prozkoumány a popsány kandidátní sítě pro možné řešení, následně je provedeno několik experimentů pro zjištění skutečného chování neuronové sítě.

KLÍČOVÁ SLOVA

Long Short-Term Memory (LSTM), hluboké učení, tvorba datasetu, KERAS, odšumování, rekurentní neuronová síť

ABSTRACT

The thesis focuses on the use of deep recurrent neural network, architecture Long Short-Term Memory for robust denoising of audio signal. LSTM is currently very attractive due to its characteristics to remember previous weights, or edit them not only according to the used algorithms, but also by examining changes in neighboring cells. The work describes the selection of the initial dataset and used noise along with the creation of optimal test data. For network training, the KERAS framework for Python is selected. Candidate networks for possible solutions are explored and described, followed by several experiments to determine the true behavior of the neural network.

KEYWORDS

Long Short-Term Memory (LSTM), deep learning, created dataset, KERAS, denoising, reccurent neural network

TALÁR, Ondřej *Redukce šumu audionahrávek pomocí hlubokých neuronových sítí*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 32 s. Vedoucí práce byl Ing. Pavol Harár

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Redukce šumu audionahrávek pomocí hlubokých neuronových sítí“ jsem vypracoval 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 uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom 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 ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Pavlu Harárovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

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 . . . . . .

. . . . . .
(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Práce se zabývá možnostmi potlačení šumu v audionahrávkách pomocí hlubokých neuronových sítí. Jedná se konkrétně o vyčištění čteného textu, který je znehodnocen různými šumy, vyskytujícími se v běžném prostředí. Je popsán význam neuronových sítí, jejich vrstev, použití s důrazem na moderní rekurentní struktury. Také jsou uvedeny nedostatky neuronových sítí a jsou nastíněni kandidáti na řešení problému.

Důvod ke zpracování tématu je neustále se rozvíjející odvětví zahrnující strojové učení pomocí neuronových sítí. Metody za pomoci hlubokého učení představují velice atraktivní alternativu ke „klasickým“ metodám, například k různým druhům transformací. Neuronové sítě nepotřebují tak složitý matematický aparát pro svou funkci, omezují se na správné uchopení problému a vlastní konstrukci. Navíc nejsou limitovány lineárními transformacemi, které jsou podstatné u jiných metod odstraňování šumu. Není potřeba žádného zpětného kroku z oblasti, ve které je šum lépe viditelný či odstranitelný bez větší degradace signálu. Když neuronové síti předložíme problém, tak v případě správného nastavení je schopna naučit se správný postup při řešení našeho problému. Toto však není tak jednoduché, jak se zdá. Je zapotřebí velké databáze příkladů, taktéž čas učení je poměrně dlouhý. Proto se učení provádí na grafikých kartách, které tento čas dovedou velice znatelně zkrátit. Další problémy představují pak chybně sestavené sítě, které se naučí rozeznat šum místo užitečného signálu, nebo kupříkladu přetrénované sítě. Ty jsou nepoužitelné pro jiná řešení, než ty, na kterých jsou trénované.

V první části práce bude čtenář seznámen s neuronovými sítěmi, jejich vznikem, popisem struktur, vrstvami, nejpoužívanějšími algoritmy a nejobvyklejšími problémy při trénování. Též budou vysvětleny základní pojmy ve všech jednotlivých částech. Druhá část bude již zaměřena na popis kandidátních sítí a důvodů jejich možného použití. Taktéž popíšeme důvod a výběr datasetu pro testování, včetně srovnání s datasety použitými u učení kandidátních sítí. Zmíníme i framework KERAS, ve kterém bude probíhat návrh a učení sítě. V poslední kapitole bude popsán postup při vytváření podmínek pro experiment, včetně popisu částí, které bude třeba měnit pro optimální výsledky, a nakonec výsledky experimentů.

1 NEURONOVÉ SÍTĚ

1.1 Základní popis

Myšlenka umělých neuronových sítí je založena na fungování reálných neuronů. Kupříkladu člověk je schopen mozkem analyzovat a řešit problémy automaticky, pokud je k tomu veden. Učením se stává velice složitý problém snažším. Taktéž je možné využít pouze pozorovací schopnosti bez vnějšího učení. Stejně tak se dovede chovat umělá neuronová síť. Řečeno jinak: neuronová síť je masivně paralelizovaný distribuovaný procesor, který má přirozenou schopnost pamatovat si příklady a poté je použít. Důležité jsou pro síť dvě věci. První, vědomosti jsou získávány skrze učící proces. Druhá, spojení mezi neurony je váhované a používá se pro uchování znalosti. [1] [2]

Z důvodů výše uvedených vyplývá výpočetní síla neuronové sítě. Díky paralelní struktuře a především schopnosti se učit dovede problémy generalizovat. Generalizace spočívá v produkování uspokojivých výstupů i pro vstupy, které nebyly součástí trénování. Tím je možné řešit i komplexní problémy, které by se složitě popisovaly. Prakticky ale samotná síť obvykle nebývá samostatným řešením, ale součástí většího celku. Problém je rozložen na snažší přístupy, například rozeznání vzoru nebo třídění, a zde neuronová síť najde uplatnění.

Neuronová síť přináší následující užitečné vlastnosti [1]:

- Nelinearitu – neurony jsou nelineární konstrukce a jelikož jsou provázány, je i celá síť nelineární. Navíc je tato nelinearita velmi dobře distribuována v celé síti.
- Vstupně výstupní mapování – důležitá vlastnost u učení s učitelem. Vyznačuje se modifikací vah jednotlivých spojení v závislosti na trénovací sekvenci. Síti je předložen vstupní vektor, a také požadovaný výstup. Následně je na základě těchto kritérií měněna váha neurálních spojení dokud není dosaženo minimálních (nejlépe žádných) změn. Trénování probíhá tak dlouho, dokud se nedosáhne namapování a tím pádem k vyřešení problému. Trénovací sekvence se mohou i opakovat, ale nejlépe v náhodném pořadí, jinak mohou vést k chybným výsledkům (dostaneme se k nim později).
- Přizpůsobivost – souvisí s uvedeným mapováním. Neuronová síť má vestavěnou funkci upravovat své váhy v závislosti na datech. Síť trénována na specifickou funkci může být velmi snadno přetrénována, aby si poradila s menšími odchylkami ve vstupech. Navíc, pokud operuje v nestatickém prostředí, dovede upravovat své váhy v reálném čase. Kombinace architektury a přizpůsobivosti dělá z neuronové sítě skvělý nástroj pro klasifikaci a zpracování signálů.

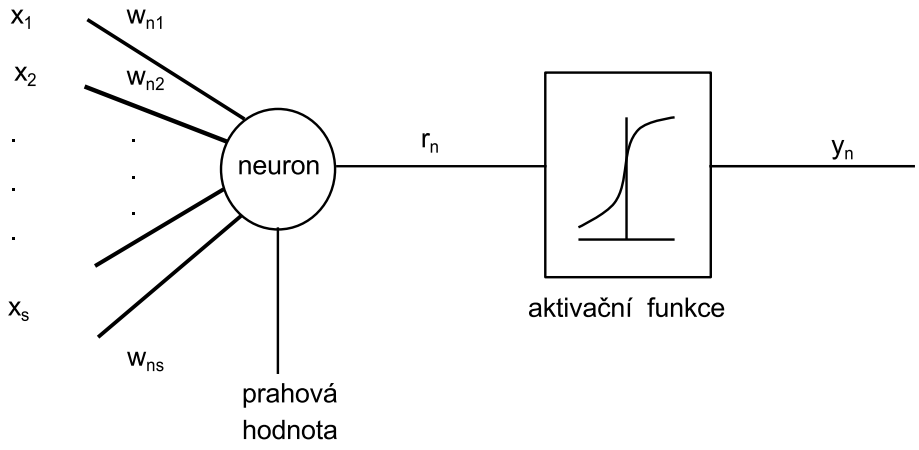
- Kontext – protože je znalost reprezentována celou strukturou neuronové sítě a každý neuron je potencionálně ovlivněn aktivitou ostatních neuronů, vzniká přirozeně schopnost zjistit a poradit si s kontextem dat.
- Uniformovanost – neurální síť je velice univerzální. Vždy máme stavební prvky v podobě neuronů, což umožňuje sdílet teorie a algoritmy v různých aplikacích.

1.2 Model neuronu

Nyní se podíváme a popíšeme stavební prvek neuronové sítě. Každý neuron 1.1 si můžeme představit jako biologickou součástku. Obsahuje zpravidla více vstupů, následuje vlastní vyhodnocovací jednotka, a pak obvykle jeden výstup (ale může mít i více).

Vstupy jsou charakterizovány svou vlastní váhou. Jedná se o to, že signál x_s vstupního spoje s je připojen k neuronu n a je násoben váhou spojení w_{ns}. Tato váha pak dělá z celého jednoho spojení buď nabuzující (pokud je kladná) či utlumující (pokud je záporná). Všechna spojení jsou přivedena do sčítací jednotky, která sečte všechny váhované vstupní signály a přidá vlastní prahovou hodnotu θ_n. Výstup z neuronu se dá matematicky zapsat takto

$$r_n = \sum_j (w_{ns} x_s) + \theta_n.$$



Obr. 1.1: Model neuronu.

Na výstupu ze sčítací jednotky musí být navíc ještě aktivační funkce, která původní hodnotu dostane do přijatelných mezí pro další zpracování. Bavíme se v užším pojetí o mezi $0 - 1$, případně se hodí rozpětí $-1 - 1$. Tento výstup pak slouží buď jako vstup dalším neuronům nebo jako finální výstup.

Zmínili jsme aktivační funkci. Jedná se o několik různých možností, jak upravit výsledek vstupních vah pro další zpracování. Aktivační funkce se liší především ve strmosti přechodu mezi stavy. Nepoužívanější aktivační funkce bývají sigmoidální, hyperbolická tangenta, softmax a ReLu. Sigmoidální funkce byla nejpoužívanější v neurálních sítích. Hlavním důvodem je možnost její diferencovatelnosti, což v neurálních sítích představuje výhodnou vlastnost. Používá rozsah hodnot od nuly do jedničky. Hyperbolická tangenta je v podstatě upravená sigmoidální funkce, jen hranice není nula, ale -1. U některých sítí je tato hranice výhodnější. Pro úplnost zde uvedeme rovnice pro sigmoidální a hyperbolickou aktivační funkci

$$\varphi(r_n) = \frac{1}{1+\exp(-r_n)},\ \varphi(r_n) = \frac{1-\exp(-r_n)}{1+\exp(-r_n)}.$$

V současnosti je mnohem oblíbenější a efektivnější ReLu (Rectified Linear Unit) funkce. Narozdíl od sigmoidální funkce nezavádí do sítě určité problémy, například problém mizícího gradientu (změny ve váhách mohou být příliš nevýrazné). Rovnice pro ReLu vypadá následovně:

$$f\left(x\right) = max\left(0, x\right)$$

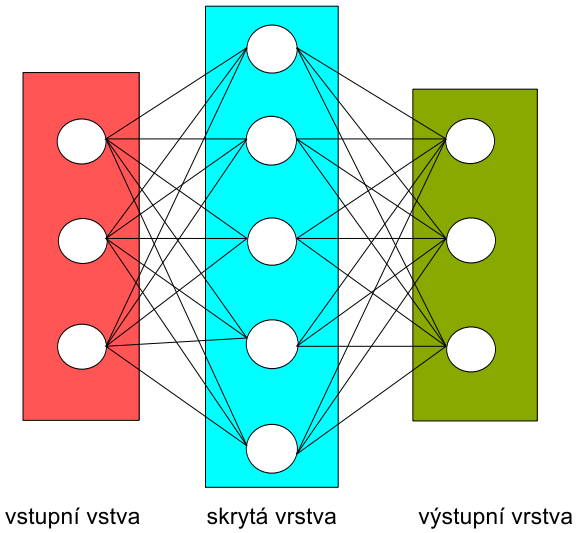
kde x je vstup neuronu. Čili pokud je vstup menší jak 0, bude vyrovnán na 0 a pokud je větší, zůstane zachován. To však může vést k neustále rostoucímu gradientu, protože ReLu nemá druhou hranici jako třeba sigmoidální funkce. Taktéž můžeme vidět, že se jedná o lineární funkci, je tedy objektivně rychlejší pro výpočet než exponenciální sigmoida.

Speciálním případem je softmax funkce. Souvisí s úpravou výstupu neuronu spíše exponenciálně než sigmoidálně. Stručně řešeno funguje tak, že maximální výstup je transformován na jedničku a všechny ostatní jsou sníženy na nulu. Použití takové funkce je vhodné, pokud síť slouží k určování pravděpodobností. Softmax je navíc velice oblíbená funkce pro výstupní vrstvu neuronové sítě. [1] [2]

1.3 Hluboké učení a architektury

V následující sekci se zaměříme na dvě základní architektury neuronových sítí. Než ale začneme, vysvětleme si pojem hluboké učení. Neuronová síť se v nejprostší formě ze tří vrstev, jako je vidět na 1.2: vstupní vrstvy, jedné skryté vrstvy a výstupní vrstvy.

Každá tato vrstva má jeden nebo více neuronů, obecně má vstupní a výstupní vrstva méně bloků, než skrytá. Hluboké učení nastává ve chvíli, kdy používáme víc jak právě jednu skrytou vrstvu. Je to logický vývoj. Čím více vrstev do sítě přidáme, tím větší dataset můžeme použít, zpracovat komplexnější funkci a získat lepší



Obr. 1.2: Základní neuronová síť.

výsledky. Bavíme-li se o hlubokém učením, jsou nejpoužívanější dvě architektury – konvoluční a rekurentní. Každá z nich má spoustu variací. My si popíšeme pouze jejich základní vlastnosti a jelikož v práci budeme používat určitý typ rekurentní sítě, zaměříme se následně na ni.

1.3.1 Konvoluční sítě

Konvoluční sítě jsou první ze dvou uváděných architektur. Abychom měli jasnější představu, připomeneme si, jak funguje konvoluce

$$y\left[n\right] = x\left[n\right] * h\left[n\right] = \sum_{k} x\left[k\right] * h\left[n-k\right], \ kde\ k \in (-\infty, +\infty)\,.$$

Sítě jsou pojmenovány jako konvoluční, ale operace prováděná na vstupech není přímo konvoluce, ale spíše modifikovaná varianta. Říká se jí korelace, a rovnice je velice podobná

$$y\left[n\right] = x\left[n\right] * h\left[n\right] = \sum_{k} x\left[k\right] * h\left[n+k\right], \ kde\ k \in (-\infty, +\infty)\,.$$

Protože se jedná pouze o faktické znalosti, použijme pro vysvětlení příměru. Představme si $x[n]$ jako vstupní obrázek a $h[n]$ jako filtr. Pak nám rovnice v podstatě říkají, že výstupní obrázek $y[n]$ vznikne postupnou aplikací filtru na původní data. Dostaneme tak obvykle něják zvýrazněné rysy, dle nastavení filtru. [7]

Konvoluční sítě dosahují velice dobrých výsledků při zpracování obrazů. Dovedou s vysokou přeností identifikovat obličeje, objekty či ručně psaný text. Každá síť se dělí na určité části. Jedná se o konvoluci, nelinearitu (ReLu), Pooling nebo podvzorkování a klasifikaci. [7] [8]

Konvoluční krok především slouží k získání důležitých znaků ze vstupního obrazu. V podstatě se tato operace provádí pomocí masky, která se přeloží přes vstupní matici obrazu. Tato maska je menšího rozměru než vstup. Postupováním a násobením matic obrazu a masky získáme menší matici, ve které jsou vyjádřeny nejpodstatnější rysy vstupu. Masce se správně v terminologii říká filtr nebo detektor rysů. Výstupní matici se pak říká aktivační nebo mapa rysů. V praxi se síť učí, jakou hodnotu matice pro filtr nastavit, aby získala co nejvýraznější rysy obrazu. Taktéž může mít více těchto filtrů. [7] [8]

Velikost mapy rysů kontrolují tři parametry, které musí být nastaveny, než je proveden konvoluční krok:

- Hloubka – tento parametr určuje počet filtrů, které užijeme pro konvoluci. Použijeme-li tedy tři filtry, získáme tři různé mapy rysů.
- Krok – značí po kolika pixelech budeme posouvat naši filtrovou matici po matici vstupu. Pokud je krok 1, posune se filtr vždy o jeden pixel a tak dále. Čím větší zvolíme krok, tím menší mapu rysů dostaneme.
- Výplň nul – občas je vhodné ohraničit vstupní matici nulami, abychom mohli aplikovat filtr na krajní elementy. To nám umožňuje také kontrolovat velikost mapy rysů.

Nelinearita nebo ReLu operace se aplikuje na každý pixel. Následuje po konvoluční operaci a postupně takto přemění všechny negativní pixely na pozitivní. Účelem této operace je vložení nelinearity do naší sítě, protože data, na kterých se učí, jsou taktéž nelineární (konvoluce sama je operací lineární). Místo ReLu lze použít třeba sigmoidální fukci, ale ReLu se ukázala jako efektivnější ve většině případů. [7] [8]

Pooling neboli podvzorkování redukuje dimenzi každé mapy rysů, ale ponechává nejdůležitější informaci. Může mít několik typů: maximum, průměr či sumu. V případě maximálního poolingu vytvoříme malou matici jako okno. Prvky v tomto okně se pak zahodí, kromě toho s nejvyšší hodnotou informace. Takto projde okno dle nastavení kroku celou mapu rysů a vytvoří novou matici. Obdobně funguje průměrování nebo součet, ale maximum funguje často nejlépe. V zásadě pooling pomáhá s velikostí matice, redukuje počet parametrů a výpočtů v síti, dělá síť odolnější k malým změnám vstupu (bereme maximum výrazného rysu, takže malá změna vstupu nemá vliv na výsledek), a nakonec pomáhá určit objekt bez ohledu na jeho velikost (i velký objekt je zredukován do malé matice). [7] [8]

Tyto tři důležité kroky se mohou opakovat několikrát po sobě. Není nic neobvyklého mít sérii konvoluce a ReLu následovanou poolingem a poté ještě jednou. Vytvoří se tak více map rysů, což je pro síť vhodné. Výstup poslední poolingové vrstvy je pak vstupem pro plně propojenou vrstvu.

Ta je složená z tradičních neuronů a používá funkci softmax ve výstupní vrstvě.

Plně propojená vrstva znamená, že každý neuron předchozí vrstvy předává svůj výstup každému neuronu následující vrstvy. Toto je výhodné pro učení nelineárních kombinací rysů.

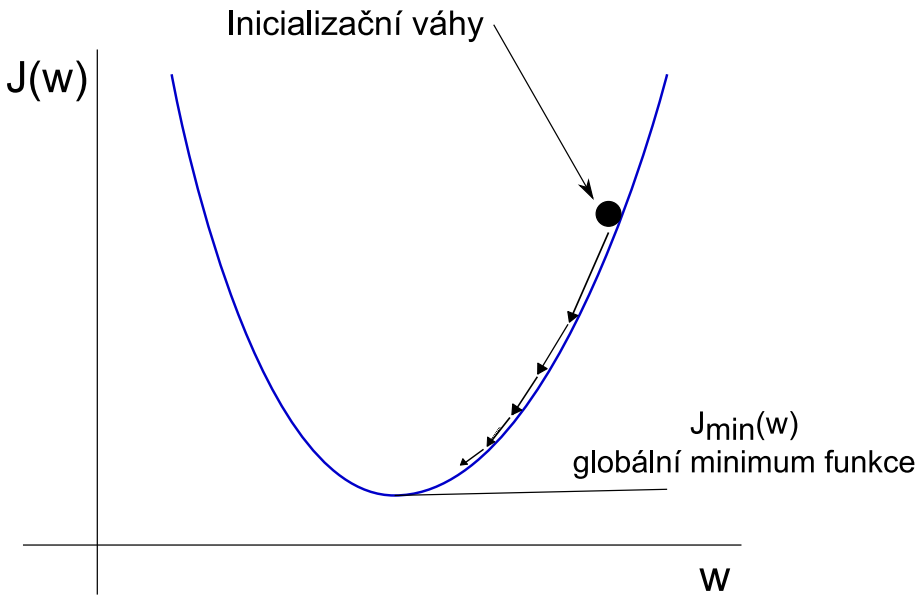
Konvoluční neuronová síť je obvykle (ne ale nutně vždy) stavěná jako feedforward, to znamená, že postupuje pouze směrem od vstupu k výstupu a neexistuje zde žádná vazba mezi neurony na jedné vrstvě. [7] [8]

1.3.2 Rekurentní sítě

Nyní se zaměřme na rekurentní sítě. Tyto architektury získavají na popularitě především v posledních letech díky pokroku v návrhu sítí, a také díky možnosti využívat procesní výkon moderních grafických karet. Jsou velice užitečné pro řešení sekvenčních nebo časově závislých problémů. To je možné hlavně díky tomu, že každý neuron obsahuje vnitřní paměť, s pomocí které může porovnávat aktuální vstup informací z předchozího vstupu. To umožňuje síti hlubší pochopení problému a naučení se správného řešení. Rekurentní neuronová síť obsahuje navíc smyčky, které dovolují informaci být distribuována do všech neuronů při čtení vstupu. Specifická rekurentní síť je Long-Short Term Memory, o které pohovoříme v další kapitole. Na tomto místě ale musíme vysvětlit dva důležité pojmy, užívané v těchto sítích. Jedná se o backpropagation a optimalizační metody. [9] [10]

Optimalizační metody jsou základem pro efektivitu učícího procesu. Pro neuronové sítě se nejčastěji používá některá z metod Gradient descent. Jedná se o optimalizační algoritmus, který hledá lokální minimum funkce, jak je vidět na 1.3. Nicméně, pro správné fungování tohoto algoritmu je třeba celkového řešení, v našem případě kompletního výstupu vstupní sekvence. Jak již bylo zmíněno, rekurentní sítě se používají pro řešení časově závislých problémů. Proto se používá modifikovaná metoda, Stochastic gradient descent (SGD). Narozdíl od předchůdce tato metoda používá aproximaci klasické gradient descent optimalizace, čili hledá minimum (nebo maximum) pro každou iteraci. Kvůli neustálému tvoření se ale u SGD tvoří časté variace výsledků, které mohou vést ke zhoršení sítě. Na druhou stranu, právě toto občasné zhoršení vede nakonec k nacházení stále lepšího minima po dostatečném počtu iterací, a tím pádem ke kýženému řešení. [10]

Backpropagation (zpětná propagace) [10] je velice důležitý nástroj při učení neuronových sítí. Používá se především ve spojení s optimalizačními metodami. Pokud řekneme, že používáme SGD, pak po výpočtu minima je třeba zpětně upravit váhy mezi neurony podle toho, který jak přispěl ke správnému výsledku. K tomu slouží právě algoritmus zpětné propagace. Nejprve musí dojít k průchodu trénovací sekvence celou sítí a vytvoření výsledku. Každý výsledek je složen z příspěvku neuronů, které byly aktivovány a především z váhy příslušné dráhy mezi nimi. Pomocí SGD



Obr. 1.3: Princip Gradient descent algoritmu.

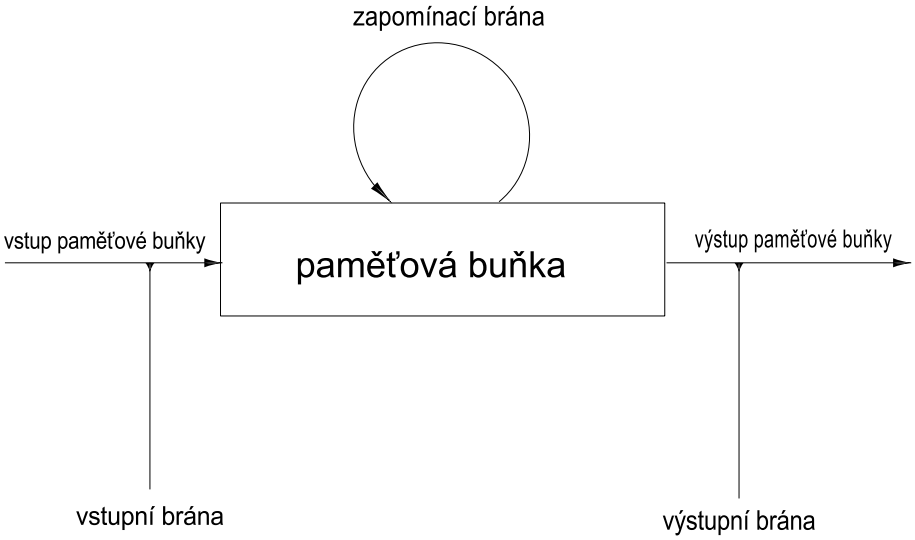
se vypočítá minimalizace dané chyby vzhledem k očekávanému výsledku, a ta je následně distribuována zpětně sítí. To se děje pomocí parciální derivace vstupů jednotlivých neuronů ve vrstvě. Zjistíme tím, která váha nejvíce přispěla k výsledku, a podle toho ji upravíme (chybové dráhy mají váhu sníženou a naopak). Po upravení jednotlivých vah se postoupí o vrstvu zpět, a opět se pokračuje s již upravenými vahami. Takto projde algoritmus všechny váhy až ke vstupu do sítě. Pokud je neuronová síť správně nastavena, je zaručena postupná konvergence k co nejmenší chybě.

1.4 Long-Short Term Memory

Z mnoha architektur rekurentních neuronových sítí se blíže zaměříme na Long-Short Term Memory (LSTM) [3]. Tento přístup se ukázal jako efektivní při zpracování řeči nebo hudby, nicméně jeho využití zde nekončí. Jeho výhodou je velice rychlá schopnost učení, a to především díky paměťovému bloku (odtud název). Ten si dovede uchovávat mnohem delší dobu předchozí váhy a pomocí toho se učit.

Problémem většiny rekurentních sítí se stává fakt, že ve zpětnovazební smyčce uchovávají informaci o předchozí aktivaci. Algoritmy pro určení informace pro zapamatování ale zpravidla trvají dlouho, což je nechtěné hlavně v případě, že mezi vstupním a učícím (zpětnovazebním) signálem není dostatečné zpoždění. Nastávají prakticky dva případy. Buď se začnou učící signály překrývat se vstupními. To zapříčiní nestálost vah a nestabilní učení. Druhá možnost nastane ve chvíli, kdy se zpětný signál postupně stane tak minimálním, že se síť nedovede nic naučit v přija-

telném čase (mluvíme o mizící chybě). Možností by byla úprava parametrů, nicméně algoritmy jsou na takové změny citlivé, a v konečném důsledku se nic nemění. Ideální by bylo, abychom mohli zabránit konfliktům mezi vstupní a výstupní váhou a ochránit správně naučenou hodnotu proti těmto vlivům. Zde přichází místo pro LSTM architekturu.



Obr. 1.4: Model paměťové buňky LSTM.

V terminologii se neuron LSTM sítě nazývá paměťová buňka 1.4. Obsahuje vnitřní smyčku (zapomínací bránu) a vstupy i výstupy jsou multiplikativní. V podstatě to znamená, že buňka může použít i vstupy a výstupy ostatních buněk (vstupní a výstupní bránu) v síti pro zjištění, zda se momentální úprava má uložit, nebo raději uchová svou vlastní. Navíc se toto propojení dá používat pro zamezení ovlivnění ostatních buněk v síti, pokud nastane konflikt vstupního a výstupního signálu [3].

Díky těmto vlastnostem dosahuje LSTM architektura mnohem lepších výsledků při použití menšího počtu bloků a taktéž kratší doby učení. Důležité ale je, aby se s ní zacházelo správně. LSTM se učí lépe z kontextu, a proto potřebuje širší okno pro vstupní data. Kupříkladu, chceme naučit neuronovou síť abecedu tak, aby ji dovedla napsat jako spojení dvou po sobě následujících písmen. Pokud použijeme malé okno a budeme se učit jedno písmeno po druhém, dosáhneme určitého výsledku, ale nebude ideální. Na druhou stranu, pokud síti umožníme učit se kupříkladu z trojice písmen, úspěšnost vzroste na maximum.

1.5 Proces učení a problémy

Probrali jsme základy neuronových sítí a určitých architektur, je ale třeba na konec zmínit, jak funguje proces vlastního učení a kontroly správnosti naučených schopností. Taktéž uvedeme nejčastější potíže při učení, jejich důvody, a v neposlední řadě i rady, jak se jich vyvarovat.

Učící proces neuronových sítí se bez ohledu na architekturu skládá ze tří fází: trénování, validace a testování. V trénovací fázi předkládáme síti zástupce takových dat, která představují typické či lehce idealizované zástupce problému. Kupříkladu, pokud se jedná o rozpoznávání objektu, pak tento objekt musí být dostatečně dobře rozeznatelný. Pokud tomu tak není (třeba u zarušených dat, která chceme vyčistit), je třeba mít trénovací sekvenci dostatečně rozmanitou.

Trénování je první fází učícího procesu. V ní probíhá vlastní úprava vah neuronové sítě, aby byla schopná správně řešit předložený problém. Důležité je vysvětlit si nyní dva pojmy, které se zde vyskytují: batch a epocha. Pro lepší pochopení si představme množinu trénovacích dat, řekněme 1000 prvků. Můžeme sice do sítě přivést všechny prvky zaráz, ale lepší je rozdělit je na menší položky. V našem případě tedy rozdělíme celý balík dat na 10 batchů, každý o velikosti 100 prvků. Nyní dochází k iteracím. Každý batch projde sítí a podle architektury dojde k úpravám vah (pro zjednodušení budou třeba úpravy probíhat po každé iteraci). Takto projde všech 10 batchů, a jakmile se tak stane, mluvíme o jedné epoše. Těchto epoch je při trénování mnoho, prakticky tolik, kolik si nastavíme. Není ale příliš žádoucí používat jich nepřiměřeně mnoho, jak uvidíme dále. Po každém provedené epoše nastává validace. Spočívá v předkládání nových dat, obvykle podobného rázu jako trénovacích. Nicméně síť k nim neměla přístup a proto se na nich ověří její funkčnost. Validace je tedy ukazatel správnosti našeho přístupu. Jestliže se trénovací a validační výsledky příliš neliší, postupujeme správným směrem. Pokračujeme tedy další epochou a další validací a tak dále, až ke konečné přesnosti. Poslední zkouškou neuronové sítě je testování. Natrénovaná a validovaná síť dostane naprosto nová data (odlišná od trénovacích a validačních, která bývají obvykle z jednoho specificky připraveného datasetu). Pokud i testování dosahuje uspokojivých výsledků, můžeme říct, že je neuronová síť funkční a připravena na reálnou práci.

Nabízí se otázka, co dělat, pokud validace selhává. Menší odchylky v přesnosti se dají očekávat, a nejspíše souvisí s nepříliš dobrou natrénovaností či špatnou formulací problému. Pokud ale síť vykazuje znatelně horší výsledky, pak se s největší pravděpodobností jedná o overfitting. Je to nežádoucí jev a vzniká v důsledku přetrénovanosti sítě nebo málo rozmanitému datasetu. Pokud se vrátíme k teorii neuronových sítí, pak víme, že při dostatečně dlouhém trénování bude správnost řešení konvergovat. Jestliže se ale zaměříme příliš na vlastní trénování, pak budeme zákonitě dostávat

stále lepší výsledky, čím déle bude pokračovat v učení. Síti tím ale dáme možnost naučit se i ty vlastnosti, které nepožadujeme. V důsledku toho bude reagovat velkou chybovostí i na nepatrné odchylky od trénovacích dat.

Řešení naštěstí bývá relativně snadné. Lepší je zvolit rovnou jinou architekturu, nebo jiné aktivační funkce, které budou obecnější. Další možností je sledování trénování a validace. Můžeme si pak povšimnout, po kolika epochách dochází k overfittingu a předčasně zde program zastavit. Obecným řešením je ale hlavně výběr dostatečně různorodého datasetu, který obsahuje množství rozdílných příkladů jak pro trénování, tak pro validaci.

2 TEORETICKÉ ŘEŠENÍ

2.1 Kandidátní sítě pro trénování

2.1.1 Popis problému

V naší práci se budeme snažit zajistit co nejlepší možné vyčištění nahrávek pomocí hlubokého učení. Zaměříme se na Long Short-Term Memory architekturu, která není v současné době stále dostatečně prozkoumána či využívána. Slibujeme si od ní minimálně „state of the art“ výsledky, samozřejmě budeme cílit i na lepší. Velkou výhodou právě tohoto přístupu oproti klasickým odšumovacím metodám je vynikající vlastnost neuronových sítí, a to jejich nelinearita. Klasické odšumovací metody fungují hlavně na převedení znehodnocených dat do nějáké jiné podoby, ve které je možné aplikovat matematický model. Nevýhodou je ale potřeba zpětné transformace, čili všechny operace musí být lineární. Neuronové sítě se závislosti učí samy, tudíž se na ně nevztahují tato omezení. Díky tomu dosahují lepších výsledků. Oproti jiným rekurentním metodám má právě LSTM nejvýraznější schopnost učit se z kontextu díky své vnitřní struktuře, k čemuž nám dopomůže rozsáhlý dataset.

Pro řešení našeho problému s odšumováním jsme našli několik vhodných kandidátních sítí, které se pokouší o podobnou úlohu právě pomocí hlubokého učení. Díky tomu máme představu, jakým směrem se bude naše praktická práce ubírat, a jaké můžeme zhruba očekávat výsledky. Nutno říci, že všichni autoři používají různé post procesy pro vylepšení výsledků nebo je odšumování jen bonus k jejich výzkumu. Nás bude zajímat pouze čistá výkonnost neuronové sítě.

2.1.2 Spektrální mapování

Prvním kandidátem je síť použitá k naučení spektrálního mapování pro potlačení odrazů a šumu v řeči z článku [4]. Jedná se o metodu učení s učitelem, při které je jako vstupní signál použita spektrální reprezentace zarušené řeči. Signál je přiveden na vstup nejen samotný, ale taktéž se přidruží dalších 5 vzorků před i za současných signálem. Následuje hluboká neuronová síť, obsahující tři skryté vrstvy. Vstupní vektor má 161 prvků, spolu s pěti sousedními vzorky po stranách se dostaneme na $16 \times 111 = 1771$ prvků pro vstupní vrstvu. Zmíněné tři skryté vrstvy obsahují 1600 prvků každá. Výstup je pak veden jako signál o 161 prvcích (tedy bez potřebné reference, která byla u vstupu). Pro vývoj je použita metoda křížové validace (validační technika, která podle statistické analýzy odhaduje míru úspěšnosti na nezávislém datasetu). Funkce pro porovnání výsledků je směrodatná odchylka (mean squared

error). Jako aktivační funkce vrstev jsou zvoleny rektifikované lineární funkce (rectified linear function - ReLu), a ve výstupní vrstvě je použita funkce sigmoidální. Váhy jsou před trénováním nastaveny náhodně. Algoritmus zpětné propagace užívá SGD s adaptivními učícími hodnotami a momentovým termem (adaptive learning rates a momentum term - obojí slouží k vyvažování možných chyb a stabilním učení). Výsledky jsou dle autorů velice uspokojivé (zvýšení SNR v průměru o 3 dB pro neznámý signál) a sami doporučují další průzkum směrem k LSTM.

2.1.3 Speech enhancement

Další možné provedení je použito jako součást většího celku v článku [5]. Provádí posílení řeči (speech enhancement), což je také technika potlačení nežáducího šumu pomocí neuronových sítí. Tento kandidát byl testován jako plně propojená architektura s 32 vstupy, 120 skrytými prvky a 11 výstupními jednotkami. Váhy jsou aktualizovány pomocí zpětná propagace v čase (backpropagation trough time - BPTT). Chyby jsou vyhodnocovány pomocí sumy kvadratických chyb (sum squared error). Jde především o předpovídání čistého signálu ze zašuměného. Oproti spektrálnímu mapování je zde používána větší škála upravených nahrávek (SNR bylo měněno od 20 dB do -5 dB po kroku 5 dB). Porovnání s ostatními přístupy je ale velmi obtížné, neboť autorům článku nešlo v hodnocení ani tak o zjištění míry odšumování, jako spíše o správné identifikování odšuměné řeči.

2.1.4 Komplexní rekurentní sítě

Třetí a poslední možnost se na problém dívá z jiného pohledu. V článku [6] autoři spekulují, že samotné odstranění šumu pomocí posílení řeči je problémem. Jako důvod uvádí zajímavý poznatek, že i šum je „znehodnocen“ užitečným signálem. Taktéž se zaměřují na fakt, že vstup může a nemusí nutně obsahovat smíšený signál (někdy je v řeči odmlka a vynikne šum). Pak jsou žádoucí jen dva výsledky: buď čistá řeč nebo ticho. Ve zpracování se dále předpokládá, že v mezerách mezi řečí je pouze šum, a že šum má víceméně stálé spektrální rozpětí. Výhodné je, že použitá architektura se velice podobá principiálnímu fungování LSTM. Klíčové části sítě jsou experimentální: 3 až 4 vrstvy, každá má 32 až 258 jednotek. Aktivační funkce jsou nastaveny sigmoidální. Ve výsledcích je použita míra signal to distortion ratio (SDR), která je v decibelech. Taktéž je použita různá škála nahrávek (SNR od -5 dB do 15 dB po 5 dB).

2.1.5 Porovnání kandidátů

Všechny tři kandidátní sítě, které se zabývají nějákým způsobem odstraňování šumu, používají rozdílné míry pro představení výsledků. Kvůli tomu je zde porovnáme z jediného objektivního hlediska, tedy jak jsou sítě postaveny a jaké používají nastavení. Následující tabulka může být tedy brána jako rekapitulace parametrů kandidátních sítí:

Tab. 2.1: Srovnání kandidátních sítí.

	Kandidát 1	Kandidát 2	Kandidát 3
Počet vstupů	1771	32	516
Skryté vrstvy	3	1? (nespec.)	3-4
Jednotek ve skr. vrstvě	1600	30-120	32-258
Počet výstupů	161	11	129
Aktivační funkce	ReLu + sigmoid.	nespec.	sigmoid.
Propojení vrstev	plné propojení		

Naše výsledná síť na rozdíl od kandidátů bude používat LSTM architekturu, ale předpokládáme, že první i třetí kandidátní síť by mohla problém odšumování řešit. Samozřejmě se jedná pouze o pokusy, ze kterých budeme vycházet. Naše vlastní může mít jinou skladbu parametrů, už kvůli odlišné architektuře. Navíc se LSTM lépe učí na velkých datasetech i při menším počtu jednotek, oproti klasickým rekurentním sítím.

2.2 Dataset a KERAS

V rámci semestrální části práce jsme hledali dostatečně velký dataset zašuměných dat. To se ale ukázalo jako zbytečné, neboť takový dataset nejspíš neexistuje a pokud ano, není veřejně dostupný. Proto jsme přistoupili ke tvorbě vlastního datasetu. Nejprve předvedeme náš sestavený dataset, a poté ho porovnáme s datasety užitými v kandidátních sítích.

Základ tvoří archiv [11], obsahující 28539 nahrávek čteného textu o velikosti 6,21 GB. Texty jsou dostatečně rozdílné (jedná se o výňatky z volně dostupných knih) a délka čteného segmentu kolísá mezi 1s až 26s. Čtenáři jsou muži i ženy, všichni anglicky mluvící s různými přízvuky. Stažený dataset taktéž obsahuje dostatečný počet popisků, přiložených v textových souborech. Každá nahrávka je ve svém základu ve formátu FLAC, jednokanálová (mono), vzorkována na 16000 kHz s 16 bity na vzorek. Toto nastavení je pro nás velice příhodné, pouze formát byl změněn na WAV, což změnilo jeho velikost na 10,8 GB. Pro změny v nahrávkách byl používán nástroj

SoX. Vzhledem k velikosti datasetu a faktu, že každý set nahrávek je pečlivě roztříděn do podsložek, byl místo skriptu použit open-source nástroj foobar2000, který dovede snadno ignorovat složkovou strukturu, a zároveň je do něj možné přidat SoX jako knihovnu.

Další část tvorby vhodného datasetu je volba šumu. Logickou úvahou se došlo k závěru, že bude nejlepší vybrat nějáký častý hluk, který se může objevit například při telefonování. Pro testovací dataset byly vybrány tři: hluk v kanceláři [12], na ulici [13] a foukání větru [14]. K testovací fázi budou vybrány ještě další vzorky hluků. Každý ze šumů byl následně opět pomocí SoXu upraven do stejného formátu a podoby, jakou mají nahrávky řeči. Pro lepší rozprostření byla navíc každá hluková nahrávka zrychlena a zpomalena o 10%, následně byly tyto tři segmenty spojeny v jeden. Poté došlo k mixování nahrávek řeči s hlukem. Užitečný signál byl překryt signálem šumu, jehož začátek byl volen pro každou nahrávku náhodně. Celkem tedy náš dataset obsahuje 85617 znehodnocených nahrávek o celkové velikosti 32,4 GB.

Datasety použité pro kandidátní sítě jsou TIMIT a AURORA (TIDigits), následně kombinované s hluky. Náš dataset možná není tak rozmanitý, jako druhé dva, nicméně je volně dostupný pro jakékoliv použití pod licencí CC BY 4.0 (druhé dva vyžadují platbu). Abychom ukázali rozdíl mezi nimi, podívejme se na následující tabulku, která obsahuje vlastnosti surových datasetů:

Tab. 2.2: Porovnání používaných datasetů.

	Náš dataset	TIMIT	AURORA
Muži	126	438	161
Ženy	125	192	165
Velikost[GB]	10,8	0,6	nedostupné
Počet nahrávek	28539	25200	25000
Licence	CC BY 4.0	LDC	LDC
Zdarma	Ano	Ne	Ne
Forma	Ukázky z knih	Věty	Čísla

Nástrojem pro vlastní tvorbu neuronové sítě a její testování byl zvolen framework KERAS, speciálně vytvořený pro tvorbu a testování neuronových sítí. Je psaný pro Python a jeho výhodou je velice dobrá spolupráce s knihovnami pro specifickou práci s maticemi (numpy a scipy), stejně jako dostupné backendy TensorFlow a Theano. Taktéž má dobré vlastnosti co se rychlosti zpracování týče. Vlastní trénování pak bude navíc prováděno na grafické kartě, abychom dosáhli výsledků v rozumném čase. Přívětivost jazyka Python je pouze subjektivní názor autora.

Python s KERASem byl uveden do chodu na OS Lubuntu a byla na něm pro ověření funkčnosti natrénována LSTM síť pro určení následujícího písmena abecedy.

3 PRAKTICKÉ ŘEŠENÍ

3.1 Příprava experimentu

Pro objektivní zhodnocení možností LSTM sítí je nutno sestavit několik modelů a otestovat je. Při vytváření neuronové sítě pomocí KERASu je ve hře velmi mnoho hyperparametrů, které mohou, ale také nemusí ovlivnit výsledky. Model může fungovat rozdílně při použití odlišného počtu neuronů na vrstvě či s rozdílnou hloubkou. Obecně je sice pravda, že oba tyto parametry fungují víceméně lineárně, nicméně je nutno vzít v potaz, že čím větší budeme mít síť, tím déle bude trvat trénování. Taktéž je třeba experimentovat s odlišným počtem epoch. Zde opět více epoch neznamená vždy automaticky lepší výsledky, neboť může dojít k přetrénování. Specificky LSTM problémem může být správné vymodelování tenzoru vstupních dat.

Dataset je pro účely testování v programu předem rozdělen. Pro lepší čtení v průběhu testování jsou audiostopy seřazeny, takže zašuměné a čisté sekvence, podle kterých se bude provádět predikce, jsou vždy srovnány a vzájemně si odpovídají. Vstupní data sítě jsou načtena pomocí pythonovské knihovny scipy, která umožňuje přímou reprezentaci audio dat v podobě vektoru čísel. Rozsah těchto čísel je dán kódováním, v našem případě 16 bitů (tedy 1 prvek vektoru může nabývat hodnot od -32768 do 32767). Problém nastává ve chvíli, kdy jsou data předávána do vstupní vrstvy. Neuronová síť totiž očekává na vstupu tenzor stejné délky pro všechna data. Pro jednoduchost a dosažení tohoto cíle jsou proto vstupní vektory nastaveny na jednotnou velikost, která vznikne buď doplněním nul do požadované délky, či oříznutím. Jinou možností by bylo kupříkladu rozdělení původních dat na specifickou délku a tím pádem rozšíření datasetu. Hlavní délkou vektoru je tedy zvoleno číslo 64000, což v našem případě odpovídá segmentu audia dlouhého 4 vteřiny.

Vstupní segment neuronové sítě však očekává tenzor, proto je nutné data přemodelovat. V LSTM modelech se používá systém: (počet_sekvencí, počet_timestepů, počet_znaků). Jak víme, LSTM se učí dle kontextu, což je v případě dat reprezentováno počtem timestepů v jedné sekvenci. Taktéž se zdá, že LSTM síť pravděpodobně příliš nehledí na počet znaků (features) v jednom timestepu [15], což jasně udává, že se budeme snažit najít optimální poměr mezi sekvencí a timestepy. To vše pochopitelně za předpokladu, že tyto parametry budou mít vliv, poněvadž není jasná odpověď, která by hovořila pro nebo proti.

3.2 Experiment 1

První experiment je založen na nejjednodušším předpokladu, tedy že budeme po neuronové síti vyžadovat na přímý vstup přímý výstup. Tento požadavek se dá zapsat formálně jako $X(t) \rightarrow Y(t)$, kde X odpovídá zašuměnému vstupu a Y je upravený (odšumený) výstup. Přístup sice netěží z hlavních výhod LSTM sítě, nicméně je velice rychlý a snadný na přípravu. Taktéž je výhodné, že můžeme libovolvě tvarovat vstupní a tím pádem i výstupní (či trénovací) signál. Počet vrstev je zvolen na 4, všechny typu LSTM s aktivačními funkcemi ReLu, kromě poslední, kde je funkce softmax. Počet neuronů je 64 na prvních třech vrstvách, poslední je neuron jeden a vrací sekvenci. Optimizér je zvolen klasický SGD, bez jakýchkoliv úprav. Data mohou být libovolně tvarována, dokud budou mít celkem 64 000 prvků, pro tento první případ bylo po několika pokusech ustaveno sestavení (1000, 64, 1). Počet epoch byl zvolen na 10, což by mělo být dostačující pro směrodatné výsledky, vzhledem k velikému počtu nahrávek.

Tento pokus skončil neúspěchem. Ukázalo se, že ačkoliv se jedná o nejrychlejší experiment, vzhledem k jiným možnostem), výsledné řešení je nepodstatné. Loss se totiž pohybuje v hodnotách NaN, což značí, že je něco v nepořádku se sítí nebo vstupními daty. Tato chyba byla odhalena jako přesné určení aktivačních funkcí, LSTM vrstvy fungují lépe při použití předem určených vnitřních aktivací. Nicméně ani po této opravě nenastala změna ve výsledcích, bylo to však dobré pro další experiment. Závěrem je tedy tato forma nedostačující, už kvůli zmíněnému nevyužití plného potenciálu LSTM.

3.3 Experiment 2

Další fáze experimentování opustila myšlenku jednoduchosti a zaměřila se více na největší sílu LSTM sítí, tedy pochopení problému z kontextu. Problém můžeme opět zapsat formálně jako $X(t-n,\ldots,t) \rightarrow Y(t)$. Tím pádem je zaručeno, že LSTM síť dostane předchozích n vzorků pro kontext. Nicméně toto nás staví před závažný problém. Jelikož výsledný vektor chceme v určitém tvaru, konkrétně (64000, 1), musíme vstupní tenzor trénovací sekvence připravit ve tvaru (64 000, n, 1). V původním návrhu bylo určeno, že se použije zpětný pohled $n = 20ms$, tedy o 320 vzorků. Z tohoto poměru je zřejmé, že nastanou problémy z pamětí. Ačkoliv byla k dispozici grafická výpočetní karta s 10 GB RAM, nebylo to dostatečné pro tento tvar dat. Nejsnažší možností se jevilo zmenšit velikost vektoru vstupních dat. Místo 64 000 vzorků se tedy začalo používat 32 000. To nebylo dost, tak se snížil i počet vzorků na zpětný pohled až na $n = 5ms$, tedy 80 vzorků. To nebylo stále dostatečné, nicméně dále ubírat na počtu vzorků není žádoucí. Proto se postupně

začala zmenšovat velikost sítě a bylo nalezeno řešení, které s tenzorem (32 000, 80, 1) funguje a je proveditelné na grafické kartě. Neuronová síť byla stále ve tvaru 4 vrstev LSTM, ale velikost musela být snížena na 16, 12 a 8 neuronů, výstupní neuron je stále jeden. Aktivační funkce jsou ponechány vnitřně nastavené pro LSTM vrstvy, optimizér však byl použit ADAM, což je adaptabilní SGD algoritmus, nastavený v KERASu. Počet epoch byl ponechán na 10, v následných obměnách hyperparametrů snížen na 5.

Tyto experimenty se setkaly taktéž s neúspěchem, na rozdíl od prvních pokusů zde však bylo dosaženo alespoň číselných výsledků. Loss se pohybuje v řádu 10^6 a spíše kolísá, než by klesal. Taktéž se jedná o mnohem časově náročnější experiment, než byl předchozí. Proto byl pro změny zmenšen počet epoch na polovinu. Bylo důvodné podezření, že lepších výsledků se dosáhne použitím jiné výstupní vrstvy s nějakou aktivační funkcí. K tomu byla zvolena vrstva Dense, za použití wrapper TimeDistributed, pro jeho vlastnost vracení sekvencí. Aktivace byla zvolena ReLu, Tanh a poté ještě softmax. Ani jedna ze změn neměla na výsledný loss vliv.

3.4 Další možnosti

Kvůli časové náročnosti experimentů nebylo možno provádět veškeré úpravy, které by byly potřeba. Proto zde nastíníme další možnosti, které by mohly eventuálně vést k řešení, jedná se ale o čistou teorii.

Jako hlavní neuskutečněná změna je použití spektrální reprezentace signálu místo raw dat. Tyto spektrální sekvence jsou použity i v teoretickém řešení a je prokázáno, že k řešení vedou. Tato spektrální transformace je tedy možná adice do programu a dovede síti najít vhodnější řešení.

Jinou možností je předřadit LSTM vrstvám jinou vrstvu, která připraví audio data. Nejspíše by se mělo jednat o konvoluční vrstvu, která se pro tyto účely prý používá. V tomto případě je ale vliv na řešení pouhou spekulací, která může a nemusí přinášet užitek. V potaz přichází také zvýšení paměťové náročnosti.

Problém je především v tom, že state-of-the-art řešení počítají vždycky s jedním šumem v tréninkové sekvenci, což je možné ověřit v příslušných článcích diskutovaných v teoretickém řešení. V našem problému se snažíme odstranit postupně tři druhy šumů za účelem eliminace nejlépe jakéhokoliv šumu. Tato podmínka může hrát v experimentu značnou roli, neboť jde proti základním vlastnostem neuronových sítí, které mají být využity hlavně pro relativně snadná řešení. Je tedy velice pravděpodobné, že lepší výsledky lze získat pouze mnohem delším trénováním (v řádu měsíců) za předpokladu, že je to možné. Provedené experimenty však naznačují, že je v současné době problém příliš komplexní a je výhodnější natrénovat

několik neuronových sítí, každou pro specifický problém.

4 ZÁVĚR

V práci jsme teoreticky ukázali vlastnosti neuronových sítí a jejich možné aplikace, s důrazem na rekurentní architekturu. Uvedení kandidáti představují příslib kvalitních výsledků, které dávají naději na efektivní řešení problému odšumování. Nicméně se jedná pouze o návrh řešení, naše vlastní struktura se může lišit pro lepší výsledky. LSTM sítě totiž velmi často vyžadují jiný přístup, než klasické rekurentní sítě. Díky našemu řešení by bylo možné pohodlně odstraňovat šum z nahrávek bez potřeby složité a hlavně lineární matematické reprezentace, nebo znehodnocení původního signálu.

Téma LSTM sítí je taktéž velice aktuální, zvláště kvůli současným výpočetním možnostem. Díky nim se dají datasety a trénování provádět v přijatelných časových intervalech. Jak se ovšem prokázalo, ve vlastním trénování je velmi mnoho parametrů, které mohou ovlivnit výsledek či rychlost trénování neuronové sítě. Experimenty nebylo dosaženo vhodného řešení a dalšími možnostmi může být tedy předřazení konvoluční vrstvy pro lepší reprezentaci audio dat pro následné učení. Dalším přístupem může být použití reprezentace ve spektrální podobě či výrazně delší doba trénování. Tak či onak, výsledky pokusů a též teoretické přípravy momentálně naznačují, že problém odstraňování několika šumových signálů zároveň je příliš komplexní pro neuronovou síť a jde přímo proti jejím principům. Jako výhodnější se jeví natrénovat několik sítí, každou specificky na jiný šum a používat je dle potřeby.

LITERATURA

[1] Hajek, Milan: *Neural Networks.*, University of KwaZulu-Natal, 2005: 114 s. Dostupné z URL k: http://www.cs.ukzn.ac.za/notes/NeuralNetworks2005.pdf.

[2] Kröse, Ben; Smagdt, Patrick van der.: *An introduction to Neural Networks* The University of Amsterdam, 1996: 135 s.
Dostupné z URL: http://www.lia.univ-avignon.fr/chercheurs/torres/livres/book-neuro-intro.pdf.

[3] Hochreiter, S.; Schmidhuber, J.: *Long Short-Term Memory.* Neural Computation, 1997: 32 s.
Dostupné z URL: http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf.

[4] Han, K.; Wang, D.; Wang Y. aj.: *Learning spectral mapping for speech dereverberation and denoising.* Journal IEEE/ACM Transactions on Audio, Speech and Language Processing, 2015: 11 s.
Dostupné z URL: http://ieeexplore.ieee.org/iel7/6570655/6633080/07067387.pdf.

[5] Parveen, Shahla; Green, Phil: *Speech enhancement with missing data techniques using recurrent neural networks.* Acoustics, Speech, and Signal Processing, 2004: 4 s. ISBN 0-7803-8484-9.
Dostupné z URL: http://ieeexplore.ieee.org/iel5/9248/29343/01326090.pdf.

[6] Osako, K.; Singh, R.; Raj, B.: *Complex recurrent neural networks for denoising speech signals.* IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2015: 5 s. ISBN 978-1-4799-7450-4.
Dostupné z URL: http://mlsp.cs.cmu.edu/people/rsingh/docs/waspaa2015.pdf.

[7] Karn, Ujjwal.: *An Intuitive Explanation of Convolutional Neural Networks.* www.ujjwalkarn.me [online], 11.8.2016 [cit. 2016-12-11].
Dostupné z URL: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

[8] Saxena, Abhineet: *Convolutional Neural Networks (CNNs): An Illustrated Explanation.* www.xrds.acm.org [online], 29.6.2016 [cit. 2016-12-11].
Dostupné z URL: http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/.

[9] Godbout, Camron: *Recurrent Neural Networks for Beginners.* www.medium.com [online], 12.8.2016 [cit. 2016-12-11].
Dostupné z URL: `https://medium.com/@camrongodbout/recurrent-neural-networks-for-beginners-7aca4e933b82#.86glga7wp`.

[10] Nielsen, A., Michael: *Neural Networks and Deep Learning.* Determination Press [online], 2015 [cit. 2016-12-11].
Dostupné z URL: `http://neuralnetworksanddeeplearning.com/`.

[11] Dataset LibriSpeech corpus pod licencí CC BY 4.0
Dostupné z URL: `http://www.openslr.org/resources/12/train-clean-100.tar.gz`.

[12] Hluk-kancelář pod licencí CC BY-NC 3.0 (NonCommercial)
Dostupné z URL: `https://www.freesound.org/people/urupin/sounds/92032/`.

[13] Hluk-ulice pod licencí CC BY 3.0
Dostupné z URL: `https://www.freesound.org/people/Zabuhailo/sounds/167235/`.

[14] Hluk-vítr pod pod licencí CC BY-NC 3.0 (NonCommercial)
Dostupné z URL: `https://www.freesound.org/people/Bosk1/sounds/144083/`.

[15] Brownlee, Jason: *How to Use Features in LSTM Networks for Time Series Forecasting.* www.machinelearningmastery.com [online],
19.4.2017 [cit. 2017-05-22].
Dostupné z URL: `http://machinelearningmastery.com/use-features-lstm-networks-time-series-forecasting/`.

SEZNAM PŘÍLOH

A PŘÍLOHY NA CD

Na CD se nachází elektronická verze diplomové práce a používané programy.

Programy jsou spustitelné v jakémkoliv prostředí, které umožňuje využívat jazyk Python, verze 3 a vyšší. Dále jsou nutné knihovny numpy, scipy, KERAS, tensorflow a k nim připojené balíčky. Archiv obsahuje tyto složky:

1. Základní program (pokus.py) – obsahuje základní tělo programu s potřebnou dokumentací k proměnným a používaným funkcím