TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Identifikace indických jazyků z audio nahrávky s využitím hlubokých neuronových sítí

Bakalářská práce

Studijní program: B0613A140005 Informační technologie
Studijní obor: Aplikovaná informatika

Autor práce: **Martin Hájek**
Vedoucí práce: Ing. Lukáš Matějů, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Identifikace indických jazyků z audio nahrávky s využitím hlubokých neuronových sítí

Jméno a příjmení: **Martin Hájek**
Osobní číslo: M19000014
Studijní program: B0613A140005 Informační technologie
Specializace: Aplikovaná informatika
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Seznamte se s problematikou identifikace jazyka z audio nahrávky.
2. Poskytnutou datovou sadu indických jazyků vhodně předpřipravte pro trénování, validaci a testování neuronových sítí pro účely identifikace mluveného jazyka.
3. Navrhněte a natrénujte vybrané architektury neuronových sítí pro identifikaci indických jazyků z audio nahrávky.
4. Natrénované klasifikátory experimentálně vyhodnoťte a porovnejte všechny dosažené výsledky.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 30-40
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] BISHOP, Christopher M. Pattern recognition and machine learning. [New York]: Springer, c2006. Information science and statistics. ISBN 978-0-387-31073-2.
[2] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning. Cambridge, MA: MIT press, [2016]. Adaptive computation and machine learning series. ISBN 978-0-262-03561-3.
[3] DIWAN, Anuj, Rakesh VAIDEESWARAN, Sanket SHAH, et al. MUCS 2021: Multilingual and Code-Switching ASR Challenges for Low Resource Indian Languages. In: Interspeech 2021. ISCA: ISCA, 2021, s. 2446-2450.

Vedoucí práce: Ing. Lukáš Matějů, Ph.D.
Ústav informačních technologií a elektroniky

Datum zadání práce: 12. října 2021
Předpokládaný termín odevzdání: 16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 19. října 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

10. května 2022 Martin Hájek

Identifikace indických jazyků z audio nahrávky s využitím hlubokých neuronových sítí

Abstrakt

Identifikace jazyka je disciplína, ve které je snaha co nejpřesněji klasifikovat jazyk z promluvy. Tato práce se věnuje identifikaci jazyka z audio nahrávek. Pro klasifikaci bylo vybráno šest indických jazyků. Použité nahrávky pocházely z Multilingual and Code-Switching 2021, kde se jedna úloha zabývala touto problematikou.

Úloha je řešena ve dvou po sobě jdoucích krocích. V prvním kroku byly extrahovány nízkoúrovňové vlastnosti jazyka (akustická stránka). Pro extrakci vlastností zde byly zvoleny dva přístupy. První zvolený způsob reprezentace řeči jsou Mel-frekvenční kepstrální koeficienty (MFCC). Tyto příznaky jsou velmi využívané a ukazují se jako velmi vhodné. Jako další způsob reprezentace řeči, byly vybrány bottleneck příznaky. Tyto příznaky se generují pomocí předtrénované neuronové sítě. Jejich obliba a použití roste zejména v posledních letech. V dalším kroku je nutné příznaky klasifikovat. Pro klasifikaci zde bylo zvoleno strojové učení, konkrétně hluboké neuronové sítě. Jedná se o velmi užívanou metodu pro řešení této problematiky. Pro otestování byly vybrány dvě architektury, a to dopředné a konvoluční neuronové sítě.

Se vstupy v podobě MFCC příznaků se u dopředných sítí podařilo dosáhnout úspěšnosti 73 % a u konvolučních 71 %. U obou architektur byly provedeny rozsáhlé testy, které měly za účel zlepšení její úspěšnosti. Po otestování obou architektur na MFCC příznacích byly vstupy vyměněny za bottleneck příznaky. S využitím těchto příznaků se podařilo zvýšit úspěšnost u obou architektur o 10 %. Pro porovnání byly v práci natrénovány převzaté návrhy sítí. Tyto návrhy nepřekonaly svou úspěšností návrhy vytvořené během práce. Jako nejlepší systém byly zvoleny dopředné neuronové sítě ve spojení s bottleneck příznaky s celkovou úspěšností 83 %.

Na závěr práce byla vytvořena aplikace, která obsahuje modely natrénované během práce. Aplikace byla vytvořena pro Python pomocí Tkinter modulu. Aplikace má za úkol demonstrovat funkčnost jednotlivých řešení.

Klíčová slova: Identifikace jazyka, DNN, CNN, MFCC, Bottleneck, Indické jazyky

Spoken Language Identification of Indian Languages Using Deep Neural Networks

Abstract

Language identification is a discipline in which the effort is made to classify language from speech as accurately as possible. This thesis is concerned with language identification from audio recordings. Six Indian languages were selected for classification. The recordings used came from Multilingual and Code-Switching 2021, where one task dealt with this issue.

The issue is solved in two consecutive steps. In the first step, the low-level properties of the language were extracted (the acoustic side). Two approaches were chosen to extract the properties. The first chosen method of speech representation are Mel-frequency cepstral coefficients (MFCC). These features are widely used and are proven to be very suitable. As another way of speech representation, bottleneck features were chosen. These features are generated using a pre-trained neural network. Their popularity and use have been growing, especially in recent years. In the next step, the features are to be classified. Machine learning, specifically deep neural networks, was chosen for classification. This is a widely used method for solving this issue. Two architectures were selected for testing, namely feedforward and convolutional neural networks.

With inputs in the form of MFCC features, a success rate of 73 % was achieved for feedforward networks and 71 % for convolutional networks. Extensive tests have been performed on both architectures to improve its success. After testing both architectures for MFCC features, the inputs were exchanged for bottleneck features. Using these features, the success rate of both architectures managed to increase by 10 %. For comparison, adopted network designs were trained in the work. These suggestions did not surpass the suggestions created during the work. Feedforward neural networks in combination with bottleneck features were chosen as the best system with an overall success rate of 83 %.

Finally, a application was created, which contains models trained during the work. The application was created for Python using the Tkinter module and aims to demonstrate the functionality of individual solutions.

Keywords: Language identification, DNN, CNN, MFCC, Bottleneck, Indian languages

Obsah

Seznam tabulek

Seznam obrázků

Seznam zkratek

MFCC Mel-frekvenční kepstrální koeficienty

LI Identifikace jazyka (language identification)

DNN Hluboké neuronové sítě (deep neural network)

CNN Konvoluční neuronové sítě (convolutional neural network)

FT Fourierova transformace

FFT Rychlá Fourierova transformace (fast Fourier transform)

DFT Diskrétní Fourierova transformace (discrete Fourier transform)

MUCS Multilingual and Code-Switching

Úvod

Identifikace jazyka z promluvy je v informatice disciplína, která se snaží co nejpřesněji identifikovat jazyk mluvčího. Identifikace jazyka má široké využití. Nejčastější využití je směřováno na vícejazyčné telefonní linky. Další uplatnění by mohlo být při přepisu vícejazyčných pořadů. Zde by došlo k identifikaci jazyka a následně by byl zvolen systém pro přepis řeči na text pro identifikovaný jazyk. Díky rostoucímu výpočetnímu výkonu a rozsáhlému zkoumání se systémy umožňující identifikaci jazyka rychle zdokonalují. Problémy v této oblasti přichází u jazyků, které si jsou velmi podobné. U těchto jazyků je velmi těžké rozeznat jejich drobné rozdíly. Další problém může nastat u mluvčího, který mluví s jistým přízvukem, v tomto případě je identifikace jazyka velmi složitá.

Pro práci bylo vybráno šest indických jazyků a to: hindština, maráthština, urijština, tamilština, telugština a gudžárátština. Data, která byla pro práci použita pochází z MUCS 2021. [1] Tato soutěž obsahovala celou řadu úloh, přičemž jedna z nich se zabývala právě touto problematikou.

Úlohu identifikace jazyka lze rozdělit do více fází. V první fázi je zapotřebí extrahovat vlastnosti jazyka tzv. příznaky, díky kterým bude systém schopen identifikovat jazyk. Vlastnosti jazyka lze rozdělit do dvou kategorií: vysokoúrovňové a nízkoúrovňové. Mezi vysokoúrovňové vlastnosti patří zkoumání skladby vět či syntax jazyka. Nízkoúrovňové vlastnosti obsahují akustickou stránku jazyka. Další fáze obsahuje samotnou klasifikaci jazyků. Zde je možné zvolit více způsobů. Velmi využívané je strojové učení. Dříve se využívaly alternativní metody jako např. skryté Markovovy modely, ale s příchodem a rozvojem strojového učení se od těchto metod spíše upouští. Strojové učení velmi často dosahuje lepších výsledků než alternativní metody.

V práci budou extrahovány pouze nízkoúrovňové vlastnosti jazyka. Jejich využití je v této oblasti mnohem častější a často podávají lepší výsledky než vysokoúrovňové vlastnosti. Jejich výhoda oproti vysokoúrovňovým vlastnostem je v tom, že je lze extrahovat i z velmi krátkých nahrávek. V práci budou využity dva druhy příznaků, a to MFCC a bottleneck příznaky. Jejich využití je v této oblasti velmi časté a ukazují se jako velmi efektivní způsob reprezentace řeči. Pro klasifikaci je zde použito strojové učení, konkrétně hluboké neuronové sítě. Pro otestování byly vybrány dvě architektury hlubokých neuronových sítí, a to dopředné a konvoluční.

V současné době neexistuje příliš mnoho systémů, které jsou schopny identifikovat jazyk z audia. Je zde velký prostor pro vytvoření systému, který toho bude schopen. Ačkoliv se v posledních letech poznání v této oblasti rozšiřuje, pro vytvoření více univerzálního řešení zahrnující všechny jazyky je stále nedostatečné. Je zde

velká motivace k rozšíření poznání v této oblasti.

Práce má za cíl navrhnout systém, který bude schopen rozpoznat indické jazyky. V práci bude provedena celá řada pokusů o dosažení co nejlepších výsledků. Pokusy se budou nejčastěji věnovat úpravě hyper-parametrů sítí. V práci bude navrženo více systémů, jejich výsledky budou mezi sebou porovnávány.

1 Identifikace jazyka

Identifikace jazyka (anglicky language identification – LI) je široce se rozvíjející disciplína v oblasti zpracování řeči, kam např. spadá i rozpoznání řeči. Rozdíl mezi rozpoznáváním řeči a LI je v tom, že u rozpoznávání řeči záleží na obsahu či mluvčím, u LI na obsahu nezáleží. LI by se dala rozdělit na dva druhy, identifikace jazyka z textu nebo z audia. Mluvená řeč obsahuje akustické, fonetické a další rysy, které je možné analyzovat a využít u identifikace jazyků. V psané formě lze analyzovat rozdílnost slov, kořeny slov a lexikony. U mluvené formy se většina LI systémů zaměřuje na akustické a fonetické vlastnosti. [2] Práce se bude zabývat pouze identifikací jazyka z nahrávky.

LI má širokou škálu využití. Využití je směřováno např. na vícejazyčné linky. Pokud v současnosti volající volá na vícejazyčnou linku, přesměrování na správného tlumočníka probíhá obvykle pomocí metody pokus omyl. Tyto hovory mohou být velmi důležité, jelikož se velmi často může jednat o tísňové linky. Další využití přichází s vyšším využíváním virtuálních asistentů. Tito asistenti jsou většinou nastaveni na jeden jazyk. Zde by bylo možné přepínat jazyk virtuálního asistenta pro každého uživatele. Díky velkému pokroku v oblasti strojového učení se v posledních letech tyto systémy rychle zdokonalují. Hlavní problém u identifikace jazyka jsou podobnosti jazyků. Pokud jsou si jazyky velmi podobné není, jednoduché mezi nimi rozeznat malé rozdíly. Další problémy mohou nastat, pokud mluvčí hovoří s přízvukem, či v rámci promluvy změní jazyk, kterým hovoří.

1.1 Vlastnosti jazyka

Lidská mluva obsahuje vlastnosti, které je možné analyzovat. Jelikož si jazyky mohou být velice podobné, je zapotřebí analyzovat takové vlastnosti, v kterých bude systém schopen identifikovat rozdílnost jazyků. Tyto vlastnosti jazyka lze rozdělit do nízké a vysoké úrovně.

Nízkoúrovňové řečové rysy jsou takové, které se zaměřují na akustiku, fonologii a prozódii. Oproti tomu rysy vysoké úrovně se zaměřují na morfologii či syntax jazyka. [2]

1.1.1 Nízkoúrovňové vlastnosti

V poslední době se k LI více využívá akustických vlastností jazyka. K nízkoúrovňovým vlastnostem řeči patří akustické, prozódické či fonetické informace. Tyto

informace jsou obvykle získávané z mluvené řeči v původní podobě. Nízkoúrovňové vlastnosti řeči se velmi často reprezentují pomocí LPC (lineární prediktivní kódování), LPCC (lineární prediktivní kepstrální koeficienty), MFCC (Mel-frekvenční kepstrální koeficienty) atd. Výhoda nízkoúrovňových vlastností tkví v tom, že je lze extrahovat i z krátkého časového úseku. [2]

1.1.2 Vysokoúrovňové vlastnosti

Mezi vysokoúrovňové vlastnosti řadíme např. morfologii či syntax jazyka. Morfologie (tvarosloví) je věda, která studuje strukturu slov. Ve většině jazyků je práce se slovy odlišná. Syntax (skladba) je oblast, která se zabývá vztahy slov ve větě.

Výhodou vysokoúrovňových vlastností je, že jsou více odolné vůči šumu či jiné degradaci nahrávky. [2]

1.2 Současné poznání

Většina prací zabývajících se problematikou LI se zaměřuje na extrakci vlastností či navržením vhodného klasifikátoru. Práce zabývající se extrakcí vlastností nejčastěji volí nízkoúrovňové vlastnosti jazyka, jako vhodnější reprezentaci řeči. V pracích, které se snaží navrhnout klasifikátor, je nejčastěji využito strojové učení, zejména neuronové sítě. V dřívějších dobách se pro klasifikaci používaly například skryté Markovovy modely, které jsou využity např. v práci [3] z roku 1993.

Tento trend využívání strojového učení, zejména neuronových sítí, je patrný i v publikacích. To samé platí i u využití a zpracování nízkoúrovňových vlastností jazyka.

1.2.1 Související práce

V [4] se autoři snaží o porovnání starší metody podpůrných vektorů s neuronovými sítěmi. Pro porovnání zvolili konvoluční a dopředné neuronové sítě. Z výsledků jejich práce vychází, že neuronové sítě podávají lepší výsledky než alternativní metoda podpůrných vektorů. V práci [5] už k identifikaci jazyka využívají složitější architektury, a to neuronovou síť s dlouho-krátkodobou pamětí (Long short-term memory – LSTM). Tato architektura je už považována za složitější oproti např. dopředným sítím. Práce publikovaná na latinsko-americkém kongresu [6] se zabývá identifikací jazyka pomocí konvolučních neuronových sítí. Zde konvoluční neuronová síť má za vstup spektrogramy vytvořené z audio nahrávek. Touto metodou to řeší např. i v článku [7].

V [8] používají pro extrakci vlastností předtrénovanou neuronovou síť (bottleneck příznaky). Zde je snaha zvýšit úspěšnost dosavadních metod pomocí tohoto přístupu. V práci [9] z roku 2019 se autoři zabývají vytvořením vhodnějším extraktorem vlastností. Z jejich práce plyne, že použití bottleneck příznaků přineslo vyšší úspěšnost celého systému. V práci [10] porovnávají úspěšnost MFCC a LPC příznaků. Z jejich práce vyplývá, že MFCC příznaky jsou vhodnější pro reprezentaci řeči oproti LPC.

Jako klasifikátory zde využili hluboké neuronové sítě. Další práce [11] využívající MFCC příznaky, je používá pro identifikaci patnácti indických jazyků. Práce [12] se zabývá porovnání MFCC a bottleneck příznaků s využitím dvou klasifikátorů (hluboké neuronové sítě a Smíšené Gaussovy modely – GMM). Z výsledků vychází, že hluboké neuronové sítě podávají lepší výsledky pro oba druhy příznaků než GMM. Práce [13] se zabývala porovnáním MFCC, MFCC s dynamickými koeficienty a SDC (shifted delta cepstra) příznaky. Z jejich pokusů vyšlo, že nejlepších výsledků bylo dosaženo využitím MFCC příznaků s dynamickými koeficienty.

Práce publikovaná na webu Stringer začátkem roku 2022 se zabývá problematikou identifikace indických jazyků. Jejich přístup k LI využívá metod extrakce nízkoúrovňových vlastností řeči. Pro extrakci vlastností zvolili autoři MFCC příznaky. Pro klasifikaci jazyka zvolili neuronové sítě, konkrétně umělou neuronovou síť (tato síť obsahuje pouze jednu skrytou vrstvu). Ve své práci se snaží o identifikaci indických jazyků společně s angličtinou. Z jejich práce vyplývá, že využití neuronové sítě přineslo velmi dobré výsledky. Jejich úspěšnost klasifikace se pohybuje nad 99 %. [14]

Z výše zmíněných prací vyplývá, že velmi často jsou za nejlepší klasifikátory považovány hluboké neuronové sítě. Z jednodušších architektur se nejvíce používají dopředné neuronové sítě. U složitějších architektur se nejčastěji používají rekurentní neuronové sítě, konkrétně LSTM. V některých pracích jsou provedené pokusy a porovnání s alternativními přístupy. Ve většině prací jsou neuronové sítě zvoleny jako vhodnější klasifikátor než alternativní metody. Pro reprezentaci řeči se velmi často využívají MFCC příznaky. Studie porovnávající příznaky velmi často dochází k závěru, že MFCC příznaky dosahují lepších výsledků než jiné druhy příznaků. Poměrně nový způsob reprezentace řeči jsou bottleneck příznaky. Tyto příznaky jsou v dnešní době již zavedeným standardem. Ukazují se jako minimálně stejně vhodné jako např. MFCC příznaky, velmi často tyto standardní příznaky překonávají.

1.3 Extistující řešení

V současné době není mnoho systémů, které by byly schopny rozpoznat jazyk z audio nahrávky. Rozpoznání jazyka z psané formy je velmi častější. Toto umí většina překladačů, např. Google překladač, Microsoft překladač atd. Většina překladačů od velkých technologických firem umí rozpoznat jazyk z textu.

U rozpoznání jazyka z audia je výběr znatelně menší. Většinou se jedná o práce, které se touto problematikou zabývají, ale hotových produktů je poměrně málo. Na internetu se nachází pár webových služeb, jako např. [15], která umí identifikovat 8 jazyků. Zde uvedená služba byla v době psaní práce bohužel nedostupná.

1.3.1 Phonexia Language Identification

Tento program je schopen identifikovat z audio nahrávky až 78 jazyků. Systém je schopen rozpoznat jazyk z audia o snímkovací frekvenci 8kHz či 16kHz. Doporučovaná délka nahrávky je minimálně 5 sekund.

Tento systém obsahuje možnost přidání vlastních jazyků. Pro přidání jazyka doporučuje Phonexia použít minimálně 20 hodin nahrávek z daného jazyka.

Při vložení nahrávky je program schopen analyzovat nahrávku a vrátit soubor ve formátu XML/JSON s procentuálním vyhodnocení pravděpodobnosti, o jaký jazyk se jedná. Na jejich webových stránkách se nepíše o tom, jaké přístupy používají. [16]

2 Vybrané principy

Vybrané principy pro tuto práci vychází z moderních trendů, které se v oblasti identifikace jazyků využívají. Zde vybrané principy jsou výsledkem provedeného průzkumu s přihlédnutím na možnosti, které byly v rámci práce k dispozici.

Pro extrakci vlastností byly zvoleny metody, které extrahují nízkoúrovňové vlastnosti řeči. Zde byly zvoleny dva přístupy, které se v posledních letech řadí k velmi často využívaným. U klasifikace jazyka bylo využito strojového učení, konkrétně neuronových sítí. Zde bylo využito jednodušších architektur, které nejsou tak náročné na výpočetní výkon.

2.1 Extrakce vlastností

Pro extrakci nízkoúrovňových vlastností řeči byly zvoleny dvě metody. První metoda převádí audio na Mel-frekvenční kepstrální koeficienty. Tyto příznaky jsou velmi často využívanou variantou pro reprezentaci řeči. Jejich výhoda oproti ostatním metodám je v tom, že při svém výpočtu se snaží aproximovat lidské vnímání sluchu.

Druhou zvolenou metodou pro extrakci vlastností jsou bottleneck příznaky. Tento přístup je poměrně nový a řadí se mezi velmi efektivní metody. Mezi výhody tohoto přístupu patří velká odolnost vůči šumu v pozadí či jiné degradace signálu. Dále jsou zde popsány postupy, které se buď v práci využívají, či jsou pro pochopení celkového systému relevantní.

2.1.1 Fourierova transformace

Nejjednodušší spektrální zobrazení je pomocí Fourierovi transformace (FT). Fourierova transformace slouží pro převod signálů z časové oblasti do oblasti frekvenční. Pomocí jejího použití lze získat frekvenční spektrum signálu. FT se dá aplikovat pouze na spojitý signál, ale audio je v počítači reprezentováno jako diskrétní signál. V informatice se proto využívá diskrétní Fourierova transformace (DFT). DFT je schopna vytvořit z diskrétního systému jeho frekvenční spektrum.

V práci sice není výstup z FT (respektive DFT) nijak použit, její využití je i v prostředcích, které byly v rámci práce využity. Jedná se o velmi často využívanou metodu, doplněnou o další operace. Je zde uvedena z důvodu velkého významu pro zvolenou extrakci vlastností.

Diskrétní Fourierova transformace

DFT je pro každý bod definováno jako:

$$X_{\mathbf{k}} = \sum_{n=0}^{N-1} x_n e^{(-j2\pi nk/N)} \tag{2.1}$$

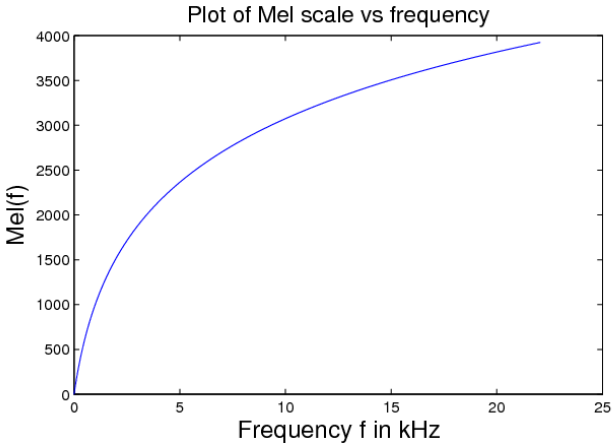
Vstupem je N hodnot diskrétního signálu. Výstupem je N hodnot komplexních koeficientů spektra na normovaných frekvencích k/N. Hodnota N představuje délku vstupního (i výstupního) signálu, x vstupní diskrétní signál, X komplexní hodnoty spektra.

Pro zpracování řeči se využívá rychlá Fourierova transformace (fast Fourier transformation – FFT). FFT rozdělí diskrétní signál na okna (často se překrývají), pro které se vypočítá DFT. Z výstupu FFT jsme již schopni určit charakteristiku signálu. Z výstupu FFT lze vytvořit tzv. spektrogram. Jedná se o vizuální reprezentaci spektra signálu. [17]

2.1.2 Mel-frekvenční kepstrální koeficienty

Hodně využívanou metodou zpracování řeči je převedení signálu na Mel-frekvenční kepstrální koeficienty. Tyto příznaky jsou často využívány pro žánrové zařazení hudby, identifikace mluvčích, identifikaci jazyka a obecně v úlohách zabývajících se zpracováním řeči či audia. Mezi jejich nevýhody patří vyšší citlivost na šum.

MFCC příznaky představují krátkodobé výkonové spektrum a jsou odvozeny od kepstrální reprezentace zvuku. Kepstrum signálu je výsledkem výpočtu inverzní FT z logaritmu spektra signálu (FFT). U MFCC se využívá Mel-frekvenční kepstrum, které používá Mel křivku. Mel křivka aproximuje lidské vnímání frekvencí. [2]



Obrázek 2.1: Mel křivka [18]

Výpočet MFCC

Před samotným výpočtem je signál upraven HP (horní propust) filtrem. Upravení signálu HP filtrem má za účel zesílení vyšších frekvencí. Jako další se vypočte FFT.

V následujícím kroku se provede pásmová filtrace frekvenčního spektra signálu (z výsledku FFT). Zde se pomocí trojúhelníkových oken definují pásma. Výkony jednotlivých složek z FFT se vynásobí příslušným koeficientem z Mel křivky. V dalších několika krocích dochází k výpočtu kepstrálních koeficientů tzn. zlogaritmování výsledného spektra signálu a aplikací inverzní FT. Kepstrální koeficienty se následně vynásobí speciální okénkovou funkcí. Tento krok má za následek vyrovnání rozptylů hodnot. K takto vytvořeným koeficientům se mohou přidat ještě tzv. dynamické koeficienty (delta a delta-delta koeficienty). Jelikož ostatní koeficienty jsou statické, tyto koeficienty mají za úkol přinést dynamickou informaci o signálu. Delta koeficient je definován jako první derivace okolí a delta-delta koeficienty jsou definovány jako druhá derivace okolí. K výpočtu delta koeficientů slouží vzorec:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{2.2}$$

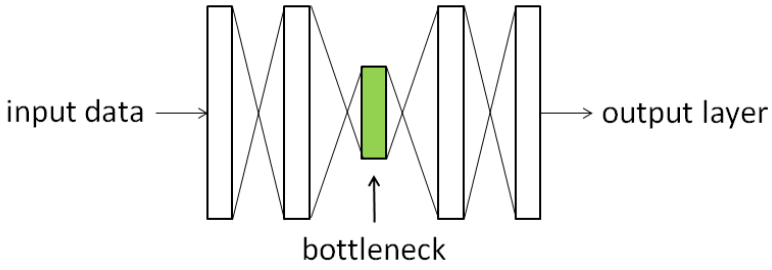
Kde d_t je delta koeficient pro daný vzorek t vypočítaný ze statických koeficientů c v okolí $t+n$ do $t-n$. Hodnota N definuje velikost okolí, její obvyklá hodnota je 2. [19, 20, 21]

2.1.3 Bottleneck

Příznaky vytvořené pomocí neuronové sítě se v posledních letech ukázaly jako vhodný nástroj pro extrakci vlastností řeči. Tyto příznaky jsou generovány z vícevrstvé předtrénované neuronové sítě. Tato neuronová síť se vyznačuje tím, že jedna vrstva má značně menší dimenzionalitu než ostatní. Tato menší vrstva vytváří v síti omezení, které nutí informace týkající se klasifikace v této vrstvě reprezentovat. K bottleneck příznakům se mohou připojit i jiné příznaky a vytvořit tím ještě komplexnější strukturu vlastností řeči. [22]

Generování bottleneck příznaků

Příznaky jsou generovány pomocí průchodu signálu předtrénovanou neuronovou sítí. Nákres takové sítě je na obrázku 2.2. Tyto neuronové sítě se trénují na určitý druh rozpoznávání řeči. Velmi často se trénují pro rozpoznání fonémů.



Obrázek 2.2: Neuronová síť s vrstvou s menší dimenzionalitou [23]

Po natrénování sítě se všechny vrstvy za vrstvou s menší dimenzionalitou odeberou. Vrstva s menší dimenzionalitou se poté stane výstupní vrstvou. Výstup z této vrstvy jsou příznaky, které lze použít pro klasifikaci.

2.2 Klasifikace jazyka

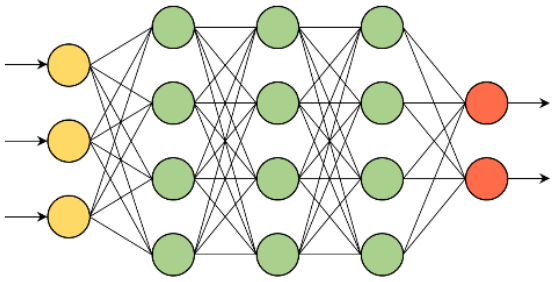
Pro klasifikaci jazyka bylo zvoleno strojové učení, konkrétně hluboké neuronové sítě. Neuronové sítě patří mezi velmi často využívaný nástroj nejen u LI. Pro potřeby práce byly vybrány dvě architektury. První architekturou jsou dopředné neuronové sítě. Jejich výhoda je v jednoduchém použití a konzistentnímu chování.

Druhá zvolená architektura byla konvoluční. Konvoluční neuronové sítě se nejčastěji využívají pro počítačové vidění, to ale neznamená, že nemohou být efektivní nástroj i pro identifikaci jazyků. Zde byly zvoleny, jelikož se jedná o architekturu s přijatelnými nároky na výpočetní výkon.

2.2.1 Hluboké neuronové sítě

Hluboká neuronová síť (deep neural network – DNN) je neuronová síť s více než dvěma skrytými vrstvami. Skryté vrstvy bývají doplněné o aktivační funkce. Aktivační funkce se používají pro vnesení nelinearity do modelu.

Na obrázku 2.3 je žlutě označena vstupní vrstva, červeně výstupní vrstva a zeleně jsou vyznačené skryté vrstvy. Neuronová síť na obrázku obsahuje 3 skryté vrstvy.

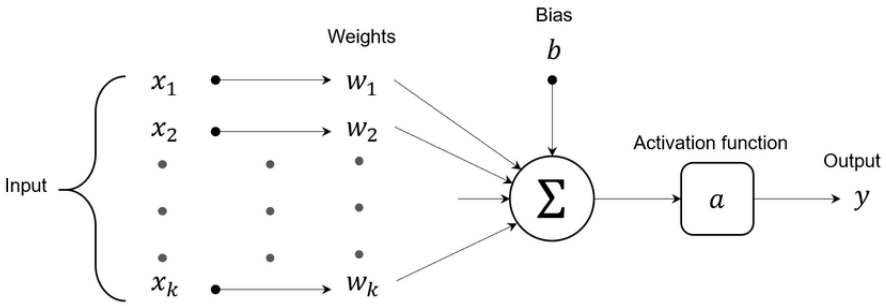


Obrázek 2.3: Hluboká neuronová síť [24]

Hluboké neuronové sítě se skládají z neuronů a vah. Spojení mezi neurony je definováno vahou, s kterou přechází výstup z jednoho neuronu jako vstup do druhého. K jednotlivým neuronům se přičítají ještě bias hodnoty. Bias má za úkol posunutí aktivačních funkcí po ose x. Všechny vstupy se v neuronu sečtou (včetně hodnoty bias) a jsou předány dále, nejčastěji aktivační funkci. Znázornění neuronu s více vstupy, bias hodnotou a aktivační funkcí je na obrázku 2.4. Průchod neuronem lze vyjádřit i vzorcem:

$$y = b + \sum_{i}^{k} x_i w_i \tag{2.3}$$

kde y odpovídá výsledné hodnotě neuronu před upravením aktivační funkcí. Hodnoty x jsou vstupy neuronu vynásobené vahami w. Hodnota k definuje počet vstupů neuronu.



Obrázek 2.4: Neuron [25]

Aktivační funkce

Aktivační funkce upravuje výstupní hodnotu neuronu. Mezi nejčastěji použité aktivační funkce se řadí ReLU, sigmoid či softmax. Softmax aktivační funkce se zpravidla využívá pouze na výstupu sítě.

Propagace

U neuronových sítí máme dva druhy propagace. Díky dopředné propagaci získáme výsledek neuronové sítě. Jde o průchod signálu všemi vrstvami neuronové sítě, kdy se v každém neuronu sečtou jeho vstupy (vynásobené vahou) s bias hodnotou. Tento výsledek je dále upraven nejčastěji aktivační funkcí. Výsledek neuronové sítě je matice $1 \times N$, kde N je počet výstupních tříd. Tato matice udává, s jakou pravděpodobností je vstup zařazen do jednotlivých tříd podle daného modelu.

Při učení po dopředné propagaci následuje zpětná propagace. Zpětná propagace je nejčastěji používaná učící metoda neuronových sítí. Po dopředném průchodu vypočítáme ztrátovou funkci. Ztrátová funkce reprezentuje chybovost modelu vůči požadovanému výsledku. Následně zpětnou propagací upravíme pomocí optimalizačního algoritmu naučitelné parametry, tzn. váhy a bias hodnoty. Trénování je rozděleno na epochy. Jedna epocha obsahuje kompletní průchod trénovacích dat. [26]

Křížová entropie

Při trénování neuronové sítě je vhodné zvolit správnou ztrátovou funkci. Velmi často se využívá křížová entropie. Tato funkce má za úkol měřit rozdíl mezi výstupem z neuronové sítě a očekávaným výstupem. Křížová entropie je definována jako:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \tag{2.4}$$

Kde *Loss* je výsledný rozdíl, N je počet tříd, y_i je šance, s jakou má být třída klasifikována, a $\hat{y}_i$ je výstupní pravděpodobnost po dopředném průchodu sítě. Nyní po výpočtu ztrátové funkce je nutné upravit naučitelné parametry (váhy sítě, bias).

Pokud je v neuronové síti využita křížová entropie, je nutné, aby poslední výstupní vrstva byla logaritmická softmax funkce. [27]

SGD

Optimalizačních algoritmů je více. Zde byl zvolen jeden z nejpoužívanějších, a to SGD (stochastic gradient descent). SGD je iterativní metoda, která postupně aktualizuje naučitelné parametry tak, aby minimalizovala ztrátu (výsledek ztrátové funkce). SGD nám poskytuje směr, ve kterém má funkce (výsledky ztrátové funkce) nejstrmější rychlost sestupu (gradient).

U optimalizačního algoritmu lze nastavit parametry, jako je rychlost učení a momentum. Rychlost učení udává, jak moc se změní naučitelné parametry. Momentum je přídavný parametr (nemusí být použit u SGD), ale jeho použití je žádoucí. Vývoj ztrátové funkce může mít lokální minima. Momentum má zabránit zaseknutí se v lokálním minimu a dosáhnout tak absolutního minima ztrátové funkce. [28]

2.2.2 Dopředné neuronové sítě

Dopředné neuronové sítě (Feedforward neural network) jsou jednou ze základních architektur DNN. Jedná se o jednu z nejstarších architektur. Tyto sítě jsou velmi často používané u identifikace jazyka. Tato architektura obsahuje pouze lineární vrstvy ve spojení s aktivačními funkcemi. Hluboká neuronová síť na obrázku 2.3 je zároveň i vizualizací dopředné sítě. Hloubka této architektury udává počet lineárních vrstev v daném návrhu sítě. Skladba této architektury se obvykle skládá z opakujících se bloků. Tento blok tvoří lineární vrstva následovaná aktivační funkcí.

Lineární vrstvy

Lineární vrstvy (někdy taky nazývány plně propojené vrstvy) jsou v podstatě matice (reprezentují váhy), kterými je vstup vynásoben a předán dále. Díky maticovému násobení jsou schopny měnit rozměry vstupu. Výstup této vrstvy je propojen s každým vstupem další vrstvy. U této vrstvy lze nastavit hyper-parametr v podobě počtu výstupů. Tento hyper-parametr je v práci nazýván jako šířka sítě.

2.2.3 Konvoluční neuronové sítě

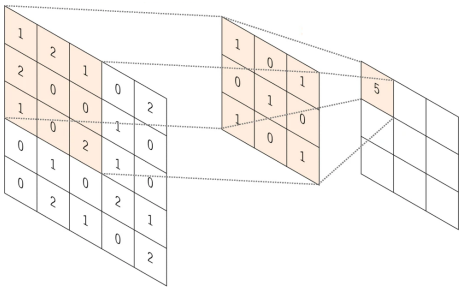
Konvoluční neuronové sítě (convolutional neural networks – CNN) jsou neuronové sítě, které se velmi často využívají k rozpoznání obrazů a obecně v počítačovém vidění. To neznamená že nemohou být použity i pro jiné využití.

Jak ze samotného názvu vyplývá, architektura této sítě využívá konvoluci. Architektura CNN se skládá nejčastěji z konvolučních vrstev, pool vrstev a lineárních vrstev. Lineární vrstvy se využívají nejčastěji na konci, kde určují velikost výstupu.

Hlavní část sítě se nejčastěji skládá z bloků, které tvoří konvoluční vrstva následovaná aktivační funkcí a poté pool vrstvou. Počet těchto bloků je v práci dále označován jako hloubka sítě.

Konvoluční vrstvy

Konvoluční vrstvy fungují na principu úpravy vstupu pomocí konvoluce. Principem je posouvání jádra (angl. kernel) po vstupních datech. Toto jádro sahá do okolí počítaného bodu. Po vypočtení hodnot jádra vynásobeného vstupem se hodnoty sečtou. U konvoluce lze zvolit parametry, jako je velikost jádra, přesah či posun jádra. Na obrázku 2.5 je konvoluční jádro o velikosti 3×3.



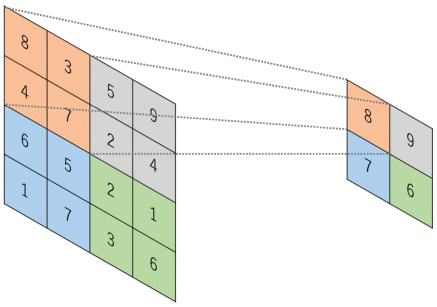
Obrázek 2.5: Konvoluce [29]

U konvoluce může být výsledný rozměr menší, stejný či větší (záleží na velikosti přesahu).

Konvoluce nemusí být jednodimenzionální. Pokud je vstup vícedimenzionální, lze na něj aplikovat konvoluci vícedimenzionální. U vícedimenzionálních konvoluce lze zvolit přístup více výstupních kanálů. Více výstupních kanálů znamená, že daný vstup bude procházen více jádry. [29]

Pool vrstvy

Pooling vrstvy nabízí pomocí matematických operací zmenšení vstupu vrstvy. Jejím úkolem je sloučení více hodnot do jedné. Tato operace neobsahuje žádné učící parametry, jedná se pouze o operaci zmenšení vstupu. Každý výstupní prvek je vypočten z části vstupu. Většinou se 4 původní prvky nahradí pouze jedním. Velmi často se využívá výběr maximálního prvku z okolí (alternativně k tomu výběr minimálního prvku) či průměr prvků. U této vrstvy na rozdíl od konvoluce musí vždy dojít ke zmenšení.



Obrázek 2.6: Maxpool vrstva [29]

Ostatní vrstvy

Dále se v konvolučních vrstvách velmi často využívá normalizační vrstva. Normalizační vrstva slouží ke snížení náročnosti na trénování, protože normalizuje vstup na menší hodnoty. Normalizační vrstva normalizuje vstup, tzn. jeho průměrná hodnota po průchodu bude 0 a směrodatná odchylka dat bude rovna 1.

Lineární vrstvy se u konvolučních architektur využívají většinou až na konci sítě. Poslední vrstva bývá většinou lineární a určuje velikost výstupu. [29]

2.2.4 Složitější neuronové sítě

Mezi složitější architektury patří například rekurentní neuronové sítě (recurrent neural network – RNN). RNN obsahují zpětnovazební smyčky, které výstup z jedné vrstvy předávají jako vstup předešlé vrstvě. Rekurentní neuronové sítě mohou mít podobu jak dopředných sítí, tak i např. konvolučních neuronových sítí.

Velmi častou architekturou pro řešení problematiky identifikace jazyka a obecně rozpoznání řeči se využívají LSTM neuronové sítě. Jedná se o druh rekurentních neuronových sítí.

Naučení složitějších architektur je značně složitější z důvodu jejich vyššího nároku na výpočetní výkon.

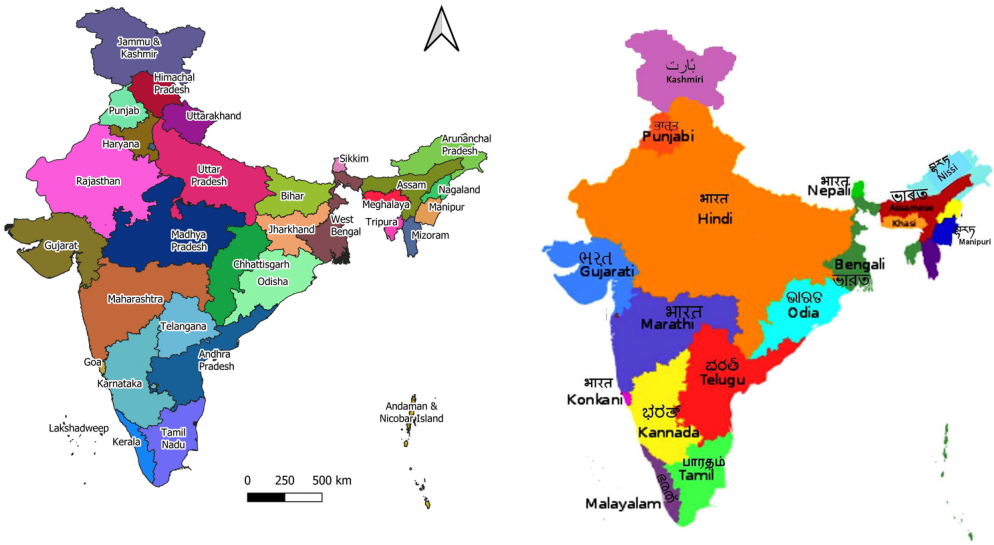
3 Indické jazyky

Indie (Indická republika) je sedmý největší stát na světě. Její počet obyvatel přesahuje 1,3 miliardy. V Indii je velká rozmanitost jazyků. Nejvíce používané jazyky spadají do indoevropské a drávidské jazykové rodiny [30].

I přes to, že se v Indii mluví stovkami jazyků, indická ústava jich jako oficiální uznává pouze 22 (včetně angličtiny). Z těchto 22 jazyků bylo pro práci vybráno šest, a to z důvodu že nahrávky, které byly použity pochází z MUCS 2021, pro kterou byl zveřejněn dataset nahrávek, a ten obsahoval pouze těchto šest vyjmenovaných jazyků.

3.1 Státy Indie

Indie se dělí na 28 států a 8 teritorií. Rozdělení jazyků napříč Indií do jisté míry kopíruje i rozložení států a teritorií. Přestože má Indie dva národní jazyky (hindštinu a angličtinu), mají některé státy další úřední jazyky.



(a) Státy indie [31] (b) Rozložení jazyků [32]

Obrázek 3.1: Rozložení jazyků podle států

Z porovnání mapy distribuce jazyků a map států (3.1) je patrné, že některé jazyky se používají na více územích. Prolínání těchto jazyků je v Indii normální.

3.2 Vybrané jazyky

Pro práci bylo přiděleno šest jazyků. Všechny tyto jazyky patří do 22 oficiálně uznávaných jazyků přičemž jeden (hindština) je i národním jazykem. Vybranými jazyky hovoří v Indii přes 75 % obyvatel.

Těchto šest jazyků bylo v MUCS zvoleno z důvodu, že pokrývají všechny čtyři demografické oblasti Indie (jih, sever, východ a západ). Dále tyto jazyky nejvíce ovlivňují i ostatní jazyky používané v Indii. Vybrané jazyky spadají do nejvíce zastoupených jazykových rodin v Indii. [1]

V následující kapitole jsou jazyky pojmenovávány jejich českými názvy. V dalších kapitolách budou jazyky nazývány jejich anglickými názvy.

3.2.1 Hindština

Hindština (angl. Hindi) patří do indoevropské jazykové rodiny. Tento jazyk je oficiální jazyk Indie společně s angličtinou. V Indii má tento jazyk jako mateřský 322 milionů obyvatel a jako vedlejší jazyk jím mluví 206 milionů obyvatel, což z něj dělá nejpoužívanější jazyk v celé Indii. [33] Díky takto vysokému zastoupení je to jazyk s třetím největším počtem mluvčích na světě. Podoba tohoto jazyka není v celé Indii stejná. Jazyk má přes 48 dialektů. [34]

V Indii se tímto jazykem mluví celkově v 8 státech (Bihar, Delhi, Haryana, Jharkhand, Madhya Pradesh, Rajasthan, Uttarakhand a Uttar Pradesh viz 3.1a) [34]. Tento jazyk se nepoužívá pouze v Indii. Silné zastoupení má taktéž v Jižní Africe, Bangladéši, Yemenu, Ugandě a na ostrově Mauricius. [35]

3.2.2 Maráthština

Maráthština (angl. Marathi) patří do indoevropské rodiny jazyků. Maráthština má 83 milionů mateřských mluvčích. Díky tomu je třetí nejpoužívanější jazyk v Indii a desátý jazyk s nejvíce mluvčími na světě. [33]

Maráthština je oficiálním jazykem státu Maharashtra. Dále se tímto jazykem mluví v Indii na územích Goa, Dadra. Maráthština a hindština jsou jazyky ze stejné jazykové rodiny, přičemž maráthština se nejvíce podobá východnímu dialektu hindštiny. [36]

3.2.3 Urijština

Urijština (angl. Odia nebo také Oriya) patří do indoevropské rodiny jazyků. Celkový počet mluvčích v Indii je přes 37 milionů, z toho pro více než 3 miliony lidí není mateřským jazykem. [33] V Indii je tento jazyk devátý nejpoužívanější.

Je oficiálním jazykem státu Odisha. Tento jazyk má více dialektů. Dialekt s názvem Mughalbandi je standardní podobou jazyka. Tento dialekt se používá např. ve školství. [37] Z vybraných jazyků má tento jazyk nejméně mluvčích.

3.2.4 Tamilština

Tamilština (angl. Tamil) spadá do drávidské jazykové rodiny. Počet mluvčích pro které je jazyk mateřský, je 69 milionů. Celkově tímto jazykem mluví v Indii 6,3 % obyvatelstva. [33]

Jazyk je uznán jako oficiální v indických státech Tamil Nadu a Poducherry. Dále je uznán jako oficiální na Srí Lance a v Singapuru. Na Srí Lance mluví tímto jazykem přes 18 % obyvatelstva, v ostatních státech tímto jazykem mluví maximálně jednotky procent. Značné množství mluvčích se nachází také v Malajsii, Mauriciu, Fidži a Jižní Africe. [38]

3.2.5 Telugština

Telugština (angl. Telugu) patří stejně jako jazyk tamilština do drávidské jazykové rodiny. Tímto jazykem mluví v Indii 81 milionů obyvatel. Tento jazyk v Indii ovládá 7,8 % obyvatel (čtvrtý nejpoužívanější jazyk v Indii). [33]

Tímto jazykem se nejvíce mluví na jihovýchodě Indie. Je to oficiální jazyk indických států Andhra Pradesh a Telangana. Na rozdíl od ostatních jazyků se tento jazyk používá výhradně v Indii. [39]

3.2.6 Gudžarátština

Gudžarátština (angl. Gujarati) patří do rodiny indoevropských jazyků. Celkově tento jazyk v Indii ovládá 55 milionů Indů. [33]

Jazyk se nejvíce používá ve státě Gujarat. Významné množství mluvčích je také v Severní Americe a Spojeném království. Jazyk se také používá v Tanzanii, Ugandě, Keni a Pákistánu.

4 Práce s daty

Tato část práce je věnována přípravě dat a práci s nimi. Data byla v rámci práce přidělena a nebylo je tedy nutné získávat samostatně. Data byla ve formě WAVE audio souborů, celková velikost byla 48 GB. Tato data byla již rozdělena na trénovací a testovací.

Na začátku práce bylo rozhodnuto, že surová data (nahrávky) nebudou vstupem do neuronové sítě. V případě že by byla surová data vstupem do neuronové sítě, trénování by se násobně prodloužilo, protože při extrakci vlastností z dat dochází zároveň k jejich zmenšení. Zpracováním se získá vhodnější vstup, jelikož budou obsahovat relevantní podobu pro identifikaci jazyka, proto budou v práci použity příznaky jako vstup.

Poskytnuté nahrávky pocházely z MUCS 2021 (MUltilingual and Code-Switching ASR Challenges for Low Resource Indian Languages). Vlastnosti nahrávek se lišily u většiny jazyků. Jazyky Hindi, Marathi a Odia měly pro trénování unikátních frází od 820 (Odia) až po 4506 (Hindi), zatímco jazyky Talugu, Tamil a Gujarati měly od 20 257 (Gujarati) až po 30 329 (Tamil). Počet mluvčích byl u všech jazyků menší než 100, kromě jazyků Telugu a Tamil, ty měly unikátních mluvčích přes 400. U testovacích dat jsou tyto trendy taktéž patrné. [1]

4.1 Data

K dispozici bylo přiděleno celkově 375 702 nahrávek (435 hodin). Nahrávky byly již rozděleny do podskupin podle jazyku. Zároveň byly již rozděleny na testovací a trénovací nahrávky. Počet nahrávek nebyl pro každý jazyk stejný. Jejich počet pro každý jazyk (včetně testovacích nahrávek) je uveden v tabulce 4.1.

Jazyk	Počet nahrávek	Čas
Hindi	103 763	100,5 h.
Marathi	84 097	99 h.
Odia	63 251	100 h.
Tamil	44 819	45 h.
Telugu	50 471	45 h.
Gujarati	29 301	45 h.

Tabulka 4.1: Celkový počet dat

Všechny nahrávky byly ve formátu WAVE, měly pouze jeden zvukový kanál a měly vzorkovací frekvenci 16 000 Hz. Vzorky měly velikost 16 bitů.

V těchto datech se zároveň nachází i tzv. measure data. Tyto nahrávky nebyly použity pro trénování ani testování. Jejich celkový počet byl 8 577 (nahrávky jsou započítány i v tab. 4.1). Tyto nahrávky se nacházely pouze u třech jazyků (Tamil, Telegu a Gujarati). Vzhledem k tomu, že dat byl dostatek, nebylo nutné je použít.

Práce s daty vyžadovala složitější přístup vzhledem k jejich objemu. Průzkumník souborů ve Windows si s tímto objemem dat nedokázal poradit, proto pro sčítání dat a jejich celkových délek byl vytvořen jednoduchý Python skript, který byl schopen pracovat se soubory a zjistit jejich celkovou dobu trvání.

4.1.1 Trénovací nahrávky

Trénovací nahrávky tvořily většinovou část z celkového objemu dat. Z trénovacích nahrávek byly použity téměř všechny nahrávky, až na pár jednotlivých souborů, které vykazovaly chybový formát (vzhledem k objemu dat nebylo nutné toto řešit). Distribuce dat a jejich celkový čas jsou zaznamenány v tabulce (viz 4.2).

Jazyk	Počet nahrávek	Čas
Hindi	99 920	95 h.
Marathi	79 422	94 h.
Odia	59 780	94,5 h.
Tamil	39 129	40 h.
Telugu	44 883	40 h.
Gujarati	22 807	40 h.

Tabulka 4.2: Trénovací data

Omezení trénovacích dat

Z důvodu velkého objemu a náročnosti na výpočetní výkon, byla vytvořena omezenější sada trénovacích dat. Tato sada omezovala první tři jazyky na cca 40 h. Data byla náhodně vybrána pomocí Python skriptu vytvořeného pro tento účel.

Jazyk	Počet nahrávek	Čas
Hindi	41 958	40 h.
Marathi	33 302	40 h.
Odia	25 277	40 h.
Tamil	39 129	40 h.
Telugu	44 883	40 h.
Gujarati	22 807	40 h.

Tabulka 4.3: Omezená sada trénovacích dat

V tabulce 4.3 je vidět míra omezení. Toto omezení bylo použito na trénování složitějších architektur sítí, zejména s časových důvodů. Další benefit zarovnání trénovacích dat na stejnou délku je, že síť se bude učit všechny jazyky rovnoměrně. Nemělo by tedy docházet k zvýšené klasifikaci tříd (jazyků) s vyšším počtem trénovacích dat (hodin nahrávek).

4.1.2 Testovací nahrávky

Testovací nahrávky byly již při přidělení dat vyčleněny ze všech ostatních nahrávek. Tato data nebyla nijak použita při trénování sítí.

Jazyk	Počet nahrávek	Čas	Prům. dél.
Hindi	3 843	5,5 h.	5,2 s.
Marathi	4 675	5 h.	3,9 s.
Odia	3 471	5,5 h.	5,7 s.
Tamil	3 081	5 h.	5,8 s.
Telugu	3 040	5 h.	5,9 s.
Gujarati	3 075	5 h.	5,9 s

Tabulka 4.4: Testovací data

Testovací data na rozdíl od dat trénovacích byla mnohem lépe distribuovaná. Jejich počty a rozdělení jsou zaznamenány v tabulce 4.4. U testovacích dat je velmi důležitá délka jednotlivých nahrávek, protože pokud je nahrávka delší, je k dispozici více dat, které je možné analyzovat. U všech jazyků (kromě Odia) byla průměrná délka nahrávky vyšší než pět sekund.

4.2 Zpracování dat

Data byla zpracována na příznaky. Příznaky jsou lepším vstupem neuronových sítí, protože obsahují klíčové informace pro identifikaci jazyka. Dále jsou mnohem menšími vstupy než samotné WAVE nahrávky, díky tomu je učení i rychlejší. Bylo proto rozhodnuto, že data budou převedena na MFCC příznaky. Tyto příznaky byly vybrány také z důvodu jejich rozsáhlého použití při řešení této problematiky. Tyto příznaky mají podobu matic o velikosti $N \times 39$, kde N je závislé na délce nahrávky.

Při generování MFCC příznaků bylo využito základních statických koeficientů. Základní statické koeficienty mají dimenzi o velikosti 13. Dále k nim byly vygenerovány delta a delta-delta koeficienty. Tyto dynamické koeficienty mají dimenzi 13 (každý). Celková dimenze koeficientů je proto 39.

4.2.1 Vygenerovaná data

Z poskytnutých nahrávek bylo vygenerováno 22,6 GB souborů, obsahující MFCC příznaky. Jejich celkový počet byl stejný jako počet nahrávek uvedených v tabulce 4.2 a 4.4. Na MFCC příznaky byly převedeny jak trénovací, tak i testovací nahrávky.

5 Použité prostředky

Tato kapitola se věnuje seznámení se softwarem a hardwarem, který byl u práce použit. Zvolení dobře optimalizovaného softwaru pro daný hardware může celý proces urychlit a zlepšit jeho fungování. K práci nebyl poskytnut žádný hardware, proto zde byla velká snaha udělat trénování sítí co nejefektivnější.

5.1 Software

Při práci bylo použito několik knihoven, které umožnily práci se sítěmi a doprovodnými funkcemi. Nejdůležitějšími softwarovými prostředky, které byly použity, byla knihovna PyTorch společně s HTK Toolkitem.

Celá práce byla vypracovaná v Pythnu (ve verzi 3.8). Při práci byly použity další externí knihovny, jako například NumPy, tqdm atd.

5.1.1 PyTorch

Pro vytváření neuronových sítí, testování i trénování byla použita knihovna PyTorch. Tato knihovna je založená na Torch knihovně. PyTorch nabízí celou sadu nástrojů pro práci s neuronovými sítěmi a byl proto zvolen jako nástroj pro práci s nimi.

PyTorch nabízí i knihovnu Torchvision, která obsahuje již hotové modely neuronových sítí.

5.1.2 HTK Toolkit

HTK Toolkit je software, využívaný zejména pro rozpoznávání řeči. Tento Toolkit je schopen z nahrávky vygenerovat soubory obsahující příznaky. Tento toolkit byl použit pouze jednou, a to při generování souborů obsahujících MFCC příznaky (viz 4.2).

HTK a Python

Pro Python není žádná oficiální knihovna pro práci se soubory vygenerovanými HTK Toolkitem. K jejich obsluze byla použita část z Github repozitáře od autora Danijel Korzinek [40]. Byl použit soubor s názvem HTK.py, který umožňuje načítání MFCC souborů. Tento soubor obsahuje i další funkce, ale ty nebyly pro práci klíčové.

5.2 Hardware

Důležité bylo i využití optimálních hardwarových prostředků pro urychlení trénování a celkovou práci se sítěmi. PyTorch nabízí možnost trénování neuronové sítě pomocí CUDA (Compute Unified Device Architecture) jader, které obsahují grafické karty od značky NVidia. Tato možnost byla využita ke zrychlení trénování.

5.2.1 Použitý stroj

Objem dat který je v práci použit bylo nutné někde uchovat. Byla proto vyčleněna část na SSD o velikosti 150 GB. Uložení na SSD je klíčové z důvodu častého načítání dat. Na začátku práce byla data uložena na externím HDD, které ovšem nezvládalo vysoké přenosové rychlosti, a celý proces se značně prodlužoval.

V práci se uvádí i časové údaje o délce trénování. Je proto vhodné uvést, na jakém stroji byly sítě trénovány. V případě, že by stroj disponoval vyšším výkonem, trénování by mohlo být znatelně rychlejší. Stroj, na kterém byly všechny neuronové sítě natrénovány, je uveden v tabulce 5.1. V tabulce jsou poznamenány pouze klíčové komponenty, které nejvíce ovlivňovaly dobu trénování. Při práci byl využit pouze tento stroj.

Tabulka 5.1: Parametry stroje

Komponenta	Název
CPU	Intel i5-9400F
GPU	NVIDIA GTX 1650 4 GB
RAM	16 GB
SSD	150 GB

5.2.2 Využití prostředků při trénování

Trénování probíhalo pomocí CUDA-Cores. NVIDIA GTX 1650 disponuje 896 CUDA-Cores. Vytížení grafické karty během trénování dosahovalo až 100 %. Tato grafická karta disponuje pamětí o velikosti 4 GB.

Využití RAM paměti bylo cca 80 %, záleželo na části dat, která byla načtená. Při trénování docházelo ze začátku k pádům z důvodu nedostatku RAM. Tento problém byl odstraněn vypnutím nepotřebných procesů běžících na pozadí. Podstatnou část RAM paměti si totiž nárokuje i operační systém.

Vytížení CPU se během trénování pohybovalo kolem 30 %. Jelikož se trénovalo na grafické kartě, CPU pouze pracoval s daty v RAM a neprováděl tak přímo výpočty trénování.

6 Trénování

Tato část práce se bude věnovat řešení trénování. Zde bylo nutné zvolit vhodný způsob práce s příznaky. Při trénování nastalo omezení v podobě nedostatku RAM paměti. Zde je vysvětlen přístup, který tento problém řeší. Dále zde byly zvoleny velmi často využívané prostředky pro trénování, včetně standardních parametrů trénování.

6.1 Vstup sítí

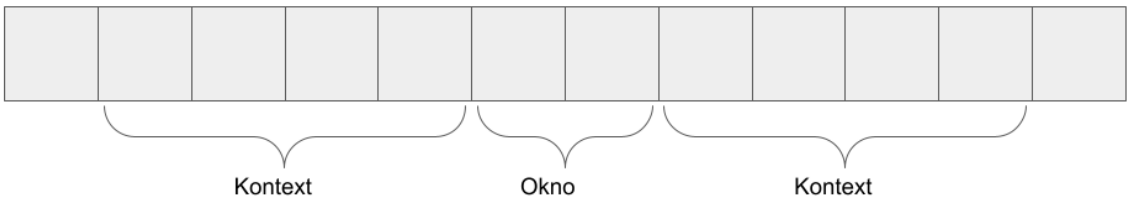
Vstupem do neuronové sítě jsou vygenerované příznaky z audio nahrávek. Tyto matice mají jasně danou první dimenzi, ovšem jejich druhá dimenze závisí vždy na délce nahrávky, z které je vygenerována. Bylo tedy nutné zvolit vhodnou velikost vstupu do neuronové sítě.

Vstup sítí je vždy matice o velikosti $N \times 39$, kde N je závislé na velikosti kontextu a velikosti okna.

6.1.1 Kontext a okno

Kontext je velikost okolí, které bude součástí matice. Části kontextu jsou společné pro více vstupů. Použití kontextu vede ke stabilizaci vstupů.

Během práce proběhly pokusy o zjištění co nejlepší velikosti kontextu. Pokud je velikost okna rovna 1, tak je k dispozici nejvyšší počet dat. Standardně je v práci použita právě hodnota 1. Velikost okna byla změněna pouze u vybraných sítí. Změnou velikosti okna dochází k snížení počtu trénovacích dat.



Obrázek 6.1: Kontext a okno

Na obrázku 6.1 je znázorněno na jednorozměrném poli vztah okna a kontextu. Při použití na matici příznaků se berou všechny prvky, které jsou na svislé ose. Výsledná šířka vstupu sítě je rovna $2 \times kontext + okno$.

6.1.2 Rozdělení do skupin

Jelikož načtení všech dat do paměti RAM nebylo možné z důvodu nedostatku RAM, bylo zavedeno rozdělení trénovacích dat. Data byla rozdělena na části, přičemž tyto části se postupně načítaly. Při velikosti 16 GB RAM bylo nejvhodnější rozdělit je do 6 skupin. Bylo žádoucí nechat v paměti nějaké volné místo, aby se paměť nepřetěžovala a zároveň ji nenechat zbytečně prázdnou. V případě, že by byla data rozdělena do velkého množství skupin trénování, by nemuselo být tak efektivní.

Každý soubor obsahuje matici příznaků. Části této matice se při trénování používají víckrát, jelikož kontext je pro více vstupů sdílený. Jelikož se při trénování data vybírají náhodně, není možné určit, která data budou potřeba, a načíst pouze určená data. To by znamenalo, že každý soubor se bude načítat víckrát než pouze jednou za celou epochu.

Rozdělení do skupin spočívalo v načtení vždy pouze určitého počtu souborů pro každý jazyk. Počet souborů, které se mají načíst, byl definován vydělením počtu souborů pro daný jazyk počtem skupin. Tím se docílilo, že každá skupina měla stejný počet souborů.

6.2 Parametry trénování

Při trénování je možné zvolit určité parametry, které mohou ovlivnit vývoj trénování, a tím ovlivnit i celkovou funkčnost sítě.

Jako kritérium při trénování neuronové sítě byla zvolena křížová entropie. Jedná se o standardní kritérium, které je velmi často využíváno. Toto kritérium vyžaduje, aby se na konci neuronové sítě nacházela logaritmická softmax aktivační funkce, proto je na konci každé neuronové sítě.

Jako optimalizační algoritmus byl zvolen SGD. Hodnota momentum byla nastavena na hodnotu 0,9 a pro rychlost učení byla zvolena hodnota 0,001. Bylo zde pár pokusů se změnou těchto parametrů, hlavně v případech, kdy trénování vybraných architektur neuronových sítí nabíralo na časové náročnosti.

7 Testování

U testování je velmi důležité zvolit vhodný způsob interpretace výsledků. Zde zvolené způsoby klasifikace jsou standardními postupy při vyhodnocování výsledků z neuronových sítí. Bylo zde definováno více typů úspěšnosti z důvodu lepšího porozumění výsledků a případného odhalení nefunkčnosti modelu.

7.1 Metodika testování

Při testování je pro nás důležité posoudit, zda síť funguje a pro jeden daný soubor vrátí jeho skutečné nářečí. Jestliže soubor obsahuje příznaky alespoň velikosti vstupu sítě, získáme minimálně jeden výstup ze sítě. Není důležitý výsledek jednotlivého vstupu, ale výsledek pro celý soubor a jeho správná klasifikace.

U testování je důležitá správná reprezentace výsledků. Při nesprávné interpretaci by se síť mohla jevit jako funkční ale pro určitou třídu by nemusela fungovat vůbec. Je proto vhodné vypočítat výsledky jak pro každou třídu (jazyk), tak i celkovou úspěšnost.

7.1.1 Vyhodnocení jednotlivého souboru

Každý soubor obsahující příznaky, za předpokladu, že je dlouhý alespoň jako vstup neuronové sítě, obsahuje jeden a více vstupů. Pro každý vstup je získán jeden výstup. Jelikož z jednoho souboru je získán minimálně jeden výstup, je nutné tyto výstupy vhodně interpretovat. Výstupy ze sítě jsou matice, které udávají, jak je podle daného modelu pravděpodobné, že daný vstup bude jednou ze tříd.

Zde byl zvolen způsob sčítání výstupů a následně zvolení třídy s nejvyšším součtem. Pro každý vstup sítě je získán jeden výstup v podobě matice o velikosti 1×6. Hodnoty matice udávají, s jakou pravděpodobností je vstup zařazen do jednotlivých tříd podle daného modelu. Tyto výsledné matice (pravděpodobnosti) se sčítají a po sečtení je brána třída s nejvyšším součtem jako klasifikovaná. Pokud by příznaky byly tak krátké, že by obsahovaly pouze jeden vstup, došlo by ke zvolení třídy s nejvyšší hodnotou. Pokud by byl kratší, než je vstup modelu, není možné z něj klasifikovat jazyk a není nijak započítán do výsledků.

7.1.2 Úspěšnosti

Úspěšnost je reprezentována jako podíl úspěšných klasifikací a počtem souborů. Lze také reprezentovat vzorcem, který úspěšnost reprezentuje v procentech:

$$\text{Úspěšnost} = \frac{\text{Úspěšné klasifikace}}{\text{Počet souborů}} \cdot 100 \tag{7.1}$$

Úspěšnost pro třídu

Ve vzorci úspěšnosti je za úspěšné klasifikace považován počet správných klasifikací pro danou třídu a za počet souborů je považován počet testovacích souborů pro danou třídu. Jedná se o poměr správných a celkových klasifikací pro danou třídu.

Celková úspěšnost

Pří výpočtu celkové úspěšnosti reprezentují úspěšné klasifikace celkový počet úspěšných klasifikací (součet správných klasifikací každé třídy) a počet souborů je celkový počet testovacích souborů, které jsou dlouhé natolik, aby byl získán minimálně jeden výstup. Nejedná se tedy o průměr úspěšností tříd. V tomto výsledku hraje velkou roli to, že pro každé nářečí není stejný počet testovacích souborů (viz 4.4).

8 Dopředné sítě

První typ architektury sítě, která byla zvolena k řešení problematiky, byly dopředné sítě. Zde bylo provedeno několik pokusů k zjištění, jak dosáhnout co nejlepších výsledků v rámci této problematiky. Poznatky, kterých bylo při vytváření a testování dopředných sítí dosaženo, byly použity i v následujících řešeních.

Dopředné sítě byly zvoleny z důvody jejich nižších časových nároků na trénování oproti např. rekurentním sítím.

Při trénování dopředných sítí byla zvolena délka trénování na šest epoch a bylo určeno, že se velikost vstupu bude měnit pouze na základě změny velikosti kontextu.

8.1 Architektura dopředných sítí

Dopředné sítě se obvykle skládají z opakujících se bloků. Jeden blok obsahuje lineární vrstvu a poté aktivační funkci. Příklad takového bloku je v tabulce 8.1. Byly zvoleny bloky, které mají stejnou velikost vstupu i výstupu. Lišil se poslední blok, u kterého byla výstupní velikost podle počtu tříd následně se softmax funkcí. Dále se lišil první blok, u kterého byla velikost upravena na velikost samotného vstupu.

Tabulka 8.1: Blok dopředné sítě

Vrstva	Vstupní velikost	Výstupní velikost	Aktivační funkce
Linear	$N \times 39$	$N \times 39$	ReLU

V návrzích dále bylo nutné přidat vrstvu, která z vstupní matice příznaků (2D pole) vytvoří 1D pole, jelikož lineární vrstva přijímá pouze 1D pole. Proto se vždy na začátku lineárních sítí vyskytuje vrstva, která převede 2D pole na 1D poskládáním dat za sebe.

8.2 Zjištění optimálních hyper-parametrů

Za účelem dosažení co nejlepších a konzistentních výsledků bylo rozhodnuto, že se dopředné sítě budou lišit především v jejich velikosti vstupu, šířce a hloubce.

Tyto experimenty probíhaly z části paralelně na sobě, protože na začátku práce nebylo jasné, zda mezi nimi nebude nějaká spojitost.

Pro natrénování této architektury bylo zvoleno trénování na 6 epoch a bylo využito maximálního počtu trénovacích dat.

8.2.1 Vstup a šířka sítě

První dva parametry, které byly zvoleny pro vyhodnocení, byly šířka a velikost vstupu. Tyto testy probíhaly paralelně. Paralelně probíhaly zejména z důvodu, že nebylo možné z něčeho vycházet a nebylo známo, zda mezi nimi není nějaká spojitost či závislost na sobě.

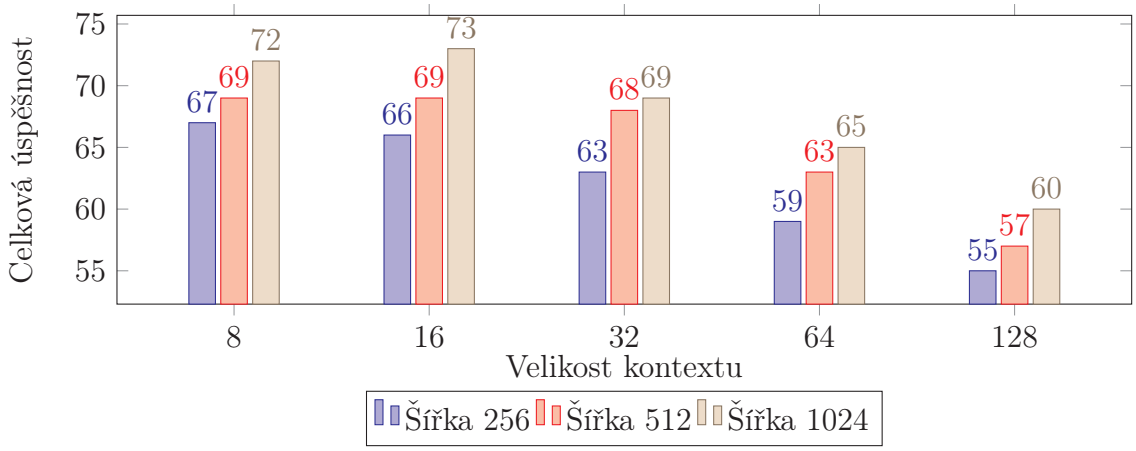
První sada návrhů

Pro účely otestování vlivu těchto dvou parametrů byla vytvořena sada návrhů, která měla pomoci určit optimální hodnotu těchto parametrů. Za tímto účelem byly vytvořeny tři návrhy sítí. Všechny návrhy obsahovaly pět lineárních bloků (viz 8.1). Tyto tři návrhy se lišily v jejich šířce. Šířka udává, jaký je počet vstupů N a počet výstupů N lineárních vrstev (počet neuronů). Jak již bylo zmíněno, vstupy i výstupy v rámci jednotlivých bloků byly stejné.

Jednalo se o základní návrhy, a tak byly zvoleny tři velikosti šířky (256, 512 a 1024).

Výsledky testů

Z grafu 8.1 je patrné, že úspěšnost se s narůstající šířkou sítě zvyšovala. Nejvyšší hodnota 1024 dosahovala z trojice šířek nejlepších výsledků.



Obrázek 8.1: Graf pokusu velikosti vstupu a šířky lineárních sítí

Postupný nárůst úspěšnosti je i patrný při postupném snižování velikosti kontextu. V tomto testu se měnila pouze velikost kontextu, nikoli okna.

Velikosti kontextu 8 a 16 vycházely velmi podobně. U šířky sítě 1024 byl rozdíl mezi kontextem 8 a 16 pouze jednoprocentní. Výsledky u menší velikosti kontextu byly mnohem méně konzistentní a nejmenší úspěšnost u jazyka Tamil byla pouhých 38 %. Úspěšnost u jazyka Tamil při velikosti kontextu 16 byla 50 % a zároveň to byla nejhorší úspěšnost ze všech jazyků. Za nejlepší byla vybrána velikost kontextu 16. Za nejlepší hodnotu šířky je považována hodnota 1024.

8.2.2 Hloubka sítě

Další parametr, u kterého byl otestován vliv na výsledky, byla hloubka sítě. Tento test probíhal po dokončení prvního testu s nejlepší šířkou a velikostí kontextu (viz 8.2.1).

Druhá sada návrhů

V druhé sadě návrhů šlo zejména o změnu hloubky sítě (počet opakování lineárních bloků).

Jak vyšlo z předešlého testování (viz 8.2.1), nejlépe dopadla šířka sítě 1024 a velikost kontextu 16. Tyto hyper-parametry zde byly použity. Základní hloubka byla 5 pro první sadu návrhů. Zde byly zvoleny další tři (3, 4 a 6).

Výsledky testu

Z tabulky 8.2 je vidět, že celková úspěšnost se nijak výrazně nelišila při změně hloubky. Hlavní rozdíl byl v úspěšnosti problematického jazyka Tamil. Ten měl nejvyšší úspěšnost u sítě s hloubkou 5 (původní hloubka). Ve třech zbylých hloubkách měl nejméně o 4 % a nejvíce až o 20 % horší výsledky.

	Hl. 3	Hl. 4	Hl. 5	Hl. 6
Hindi	74,5 %	72,0 %	72,5 %	68,9 %
Marathi	51,5 %	54,0 %	57,6 %	60,7 %
Odia	97,2 %	97,9 %	97,0 %	96,8 %
Tamil	46,0 %	30,0 %	50,0 %	41,5 %
Telugu	73,0 %	87,0 %	74,9 %	80,4 %
Gujarati	90,6 %	90,0 %	95,3 %	92,6 %
Celková	71 %	70 %	73 %	72 %

Tabulka 8.2: Výsledky testu hloubky sítě

Jazyk Marathi, který má druhou nejhorší úspěšnost, s přibývající hloubkou vycházel lépe. Rozdíl v úspěšnosti jazyka Tamil zde byl ale mnohem markantnější. Jako nejlepší hodnota hloubky byla vyhodnocena 5.

8.3 Zhodnocení dopředných sítí

Dopředné sítě, ačkoliv se jedná o jednu z nejjednodušších architektur, dosáhly dobrého výsledku a ukázalo se, že by mohly být vhodné i pro tyto problematiky. Průměrná celková úspěšnost byla 65 % a pohybovala se v rozmezí od 55 % do 73 %.

Pro všechny natrénované modely platilo, že jazyk Tamil dosahoval nejhorších výsledků. Tento jazyk působil největší problém při vyhodnocování.

Časová náročnost této architektury, ačkoliv se jedná o jednodušší architekturu, byla i přesto docela vysoká. Pro účely experimentů bylo vytvořeno 6 návrhů. Celkově bylo natrénováno 18 modelů. Časová náročnost na trénování jedné epochy se

pohybovala od 1,25 do 7 hodin, průměrný čas na epochu byl téměř 3 hodiny. Celkově bylo na této architektuře odtrénováno přes 300 hodin.

8.3.1 Nejlepší model

Jako nejlepší dopředný model byl vyhodnocen model se šířkou 1024, hloubkou 5 a velikostí kontextu 16 s oknem o velikosti 1 (celková velikost vstupu byla 33×39). V tabulce 8.3 je návrh sítě.

Tabulka 8.3: Nejlepší lineární model – návrh

Vrstva	Vstupní velikost	Výstupní velikost	Aktivační funkce
Vstup	33×39	33×39	-
Flatten	33×39	1×1287	-
Linear	1×1287	1×1024	ReLU
Linear	1×1024	1×1024	ReLU
Linear	1×1024	1×1024	ReLU
Linear	1×1024	1×1024	ReLU
Linear	1×1024	1×6	LogSoftmax

Výsledky

Výsledky nejlepšího modelu jsou zaznamenány v tabulce 8.4. Z výsledků je patrné, že jazyky Tamil a Marathi jsou nejproblematičtější. Tento deficit se u dopředných sítí nepodařilo odstranit, ale do jisté míry se ho podařilo zmírnit.

U tohoto modelu trvala doba trénování jedné epochy cca 3 hodiny, což je dobrá hodnota s přihlédnutím k možnostem hardwaru a dokonce je nižší než průměrná doba natrénovaných modelů v rámci dopředných sítí.

Hindi	72,5 %
Marathi	57,6 %
Odia	97,0 %
Tamil	50,0 %
Telugu	74,9 %
Gujarati	95,3 %
Celková	73 %

Tabulka 8.4: Nejlepší lineární model – výsledky

9 Konvoluční sítě

Další architektura zvolená k řešení této problematiky byla konvoluční. Konvoluční sítě v posledních letech nabývají na oblibě. Využívají se zejména u rozpoznání obrazů, mohou být ovšem i vhodné pro řešení jiných problematik.

Při práci s konvolučními sítěmi byly použity poznatky nabité již z předchozího pozorování, zejména velikost vstupů. Zde stejně jako u dopředných bylo zvoleno trénování na šest epoch.

9.1 Architektura konvolučních sítí

Konvoluční sítě se nejčastěji skládají z konvolučních vrstev, které jsou doplněny o pool vrstvy. Na konec sítě se navíc přidávají lineární vrstvy pro správný počet výstupů. Pro lepší funkčnost sítí je vhodné přidat také normalizační vrstvu. Normalizační vrstva normalizuje výstup předchozí vrstvy.

9.1.1 Zvolené vrstvy

Existuje více typů vrstev, které by mohly být použity. Zde byla vybrána pool vrstva, která vybírá maximální hodnotu z okolí. Jako aktivační funkce byla zvolena ReLU. U konvolučních vrstev je možné změnit počet ovlivňovaných dimenzí. Zde byla zvolena 2D konvoluce.

Tabulka 9.1: Blok konvoluční sítě

Vrstva	Vstupní velikost	Výstupní velikost	Aktivační funkce
Conv2D	$N \times 39$	$M \times N \times 39$	-
BatchNorm	$M \times N \times 39$	$M \times N \times 39$	ReLU
MaxPool	$M \times N \times 39$	$M \times \frac{N}{2} \times 19$	-

V tabulce 9.1 je znázorněná architektura konvolučního bloku se zvolenými vrstvami. Rozměr N definuje velikost vstupu, M udává počet výstupních kanálů z 2D konvoluční vrstvy. Hodnota M se v každém následujícím bloku zvyšovala dvakrát.

1D konvoluce

Při volbě vrstev se místo 2D konvoluce nabízelo použít i 1D konvoluci. Byly proto při prvotním výběru vytvořeny i sítě s 1D konvolucí. Sítě s 1D konvolucí podávaly

mnohem horší výsledky oproti sítím s 2D konvolucí. Návrhy s 1D konvolucí byly na základě tohoto poznatku zavrženy pro testování.

9.2 Zjištění optimálních parametrů

Podobně jako u dopředných sítí (viz 8.2) byly i u konvolučních sítí provedeny experimenty pro zjištění optimálních parametrů sítí. Bylo zde provedeno více testů. Na některých parametrech nezáleželo, protože jejich změnou nedošlo k nárůstu či snížení úspěšnosti. V tom případě se vycházelo z obecných doporučených vlastností konvolučních sítí.

Všechny modely byly natrénovány na 6 epoch. Při trénování byl využit omezený počet trénovacích dat pro rychlejší trénování (viz 4.1.1).

9.2.1 Velikost vstupu

První provedený experiment bylo zjištění velikosti vstupu. Zde se stejně jako u dopředných sítí měnila velikost kontextu. Vycházelo se zde ze znalostí z předešlého pozorování. Byly tedy zvoleny tři velikosti kontextu (8, 16 a 32).

Prvotní návrh

První návrh, který byl vytvořen, obsahoval tři konvoluční bloky (viz 9.1). První blok měl hodnotu M (počet výstupních kanálů) nastavenou na 8, druhý na 16 a třetí na 32. Na konci se nacházela jedna lineární vrstva.

Výsledky

Pro všechny tři zvolené velikosti vstupu byla úspěšnost sítě velmi podobná. Kontexty o velikosti 8 a 16 dosáhly celkové úspěšnosti 61 % a pro kontext 32 byla úspěšnost 62 % (viz 9.2).

U konvolučních sítí na velikosti vstupu pravděpodobně nezáleželo tolik jako u dopředných. Bylo proto rozhodnuto pokračovat s velikostí kontextu 16, která v předchozím pozorování vycházela nejlépe.

	Kon. 8	Kon. 16	Kon. 32
Hindi	61,5 %	60,3 %	71,6 %
Marathi	56,6 %	49,3 %	39,6 %
Odia	84,8 %	85,5 %	95,0 %
Tamil	22,4 %	13,1 %	28,7 %
Telugu	52,5 %	69,8 %	53,4 %
Gujarati	93,2 %	93,6 %	92,1 %
Celková	61 %	61 %	62 %

Tabulka 9.2: Velikost vstupu u konvolučních sítí

9.2.2 Velikost konvolučního jádra

U konvolučních vrstev lze nastavit velmi důležitý hyper-parametr. Jedná se o velikost konvolučního jádra. Tento hyper-parametr udává, z jak velké oblasti bude vypočítána konvoluce. U prvního návrhu byl nastaven na velikost 3. Byl přidán další návrh, který se lišil pouze ve velikosti konvolučního jádra. Velikost jádra byla nastavená na 5.

Výsledky

U tohoto experimentu nedošlo k žádné změně celkové úspěšnosti. Původní celková úspěšnost byla 61 % a se zvětšením jádra se nijak nezměnila. Výrazné změny nastaly u jazyka Hidni. Tato změna snížila úspěšnost jazyka Hindi o skoro 30 %. Tento hyper-parametr neměl vliv na celkovou úspěšnost, měl vliv spíše na jednotlivé úspěšnosti jazyků. Porovnání úspěšnosti obou velikostí kernelu je v tab. 9.3.

Zde bylo rozhodnuto změnit velikost jádra na velikost 5. Bylo tak rozhodnuto zejména proto, že se tato hodnota používá u známých konvolučních sítích, jako např. u LeNet [41].

	Ker. 3	Ker. 5
Hindi	60,3 %	31,4 %
Marathi	49,3 %	65,6 %
Odia	85,5 %	93,9 %
Tamil	13,1 %	21,5 %
Telugu	69,8 %	71,9 %
Gujarati	93,6 %	89,0 %
Celková	61 %	61 %

Tabulka 9.3: Vliv změny velikosti kernelu u konvolučních sítí

9.2.3 Lineární vrstvy na konci

U konvolučních sítí se vždy na konci vyskytuje alespoň jedna lineární vrstva. V základním návrhu byla jedna. Zde byl tedy pokus o zjištění, zda bude mít nějaký vliv počet lineárních vrstev na úspěšnost. U obecných modelů (např. LeNet [41]) jsou obvykle dvě a více.

Výsledky

Zde byla celková úspěšnost pouze o 1 % vyšší. Před přidáním této vrstvy zde byly dva jazyky (Hindi, Tamil), které nedosahovaly úspěšnosti ani 50 %. Přidáním této vrstvy se u jazyka Hindi zvýšila úspěšnost. U jazyka Tamil to naopak snížilo úspěšnost o 6 %. Výsledky, kterých bylo dosaženo přidáním lineární vrstvy, jsou v tab. 9.4 ve třetím sloupci (hloubka 3). Před přidáním této vrstvy jsou výsledky v tab. 9.3 ve třetím sloupci (kernel 5).

Z předešlého pozorování vyplynulo, že jazyk Tamil působí větší problémy, proto přidání lineární vrstvy bylo zachováno pro další testování.

9.2.4 Hloubka sítě

Předešlé experimenty byly spíše ladící k ustálení hodnot výsledků. Zvýšením či snížením hloubky sítě se podařilo mnohem více ovlivnit výsledky sítě. Hloubku sítě zde chápeme jako počet konvolučních bloků. Původní návrh obsahoval 3 konvoluční bloky. U tohoto experimentu byly zvoleny tři další počty konvolučních bloků.

Sada návrhů

K otestování tohoto experimentu byly vytvořeny další tři návrhy s třemi hloubkami (2, 4 a 5). Tyto sítě vycházely z předchozího pozorování. Na konci měly 2 lineární vrstvy, velikost kernelu byla 5 a velikost kontextu byla 16. Počet výstupních kanálů

z první konvoluční vrstvy byl 8 a každá následující konvoluční vrstva zdvojnásobila počet výstupních kanálů.

Výsledky

Výsledky experimentu už podaly jasnější data než předchozí. Výsledky ukázaly, že základní hloubka 3 byla vhodná. Snížení hloubky sítě snížilo úspěšnost sítě. U zvýšení hloubky sítě na 4 dosahovala síť stejné celkové úspěšnosti jako u hloubky 3. Při ještě dalším navýšení hloubky na 5 došlo k výraznému poklesu funkčnosti sítě. Zde byla zvolena hloubka 4 jako nejlepší. Zvolena byla z důvodu nižší úspěšnosti než 50 % pouze u jednoho jazyku.

	Hl. 2	Hl. 3	Hl. 4	Hl. 5
Hindi	61,4 %	65,6 %	59,8 %	9,3 %
Marathi	54,9 %	52,6 %	52,9 %	0,4 %
Odia	90,5 %	95,9 %	89,9 %	86,4 %
Tamil	2,5 %	15,9 %	17,4 %	0,0 %
Telugu	38,4 %	54,4 %	86,8 %	96,6 %
Gujarati	99,2 %	91,3 %	68,5 %	2,9 %
Celková	58 %	62 %	62 %	30 %

Tabulka 9.4: Výsledky změny hloubky konvolučních sítí

9.2.5 Počet výstupních kanálů konvolučních vrstev

Poslední experiment, který proběhl u konvolučních sítí, se týkal změny počtu výstupních kanálů konvoluční vrstvy. U tohoto experimentu se již podařilo značně zvýšit úspěšnost sítí. Pro tento účel byly vytvořeny další 4 návrhy. Tyto návrhy se lišily v hyper-parametru počtu výstupních kanálů konvolučních vrstev.

Sada návrhů

Další 4 vytvořené modely vycházely z předešlého pozorování. Předešlý návrh měl u první konvoluční vrstvy 8 výstupních kanálů a každá následující konvoluční vrstva tento počet zdvojnásobila.

Nové návrhy zachovávaly vlastnost dvojnásobení počtu výstupních kanálů u další vrstvy. Zde byly zvoleny jak větší, tak i menší počty výstupních kanálů. Pro otestování menšího počtu výstupních kanálů obsahovaly dva návrhy u první konvoluční vrstvy 4 a 2 výstupní kanály. U zvýšení počtu výstupních kanálů byly zvoleny hodnoty 16 a 24. Nebyla zvolena hodnota 32, jelikož už se jedná o rychle rostoucí složitost sítě a její nároky na natrénování by byly už dost vysoké.

Výsledky

Výsledky posledního experimentu přinesly největší nárůst v úspěšnosti v rámci této architektury. S postupným snižováním výstupních kanálů se snižovala i úspěšnost sítě. Při snížení na 4 výstupní kanály se celková úspěšnost lišila o 4 %. Při dalším snížení na hodnotu 2 se snížila o dalších 9 %. Snižování počtu výstupních kanálů nepřineslo zvýšení úspěšnosti.

U zvýšení počtu výstupních kanálů na 16 došlo k velkému nárůstu úspěšnosti. Po dalším zvýšení na hodnotu 24 došlo k mírnému zhoršení. Na rozdíl od snižování, zvyšování počtu výstupních kanálů mělo pozitivní vliv na celkovou úspěšnost. Zde byla vybrána jako nejlepší hodnota počtu výstupních kanálu 16.

	VK 2	VK 4	VK 8	VK 16	VK 24
Hindi	36,6 %	31,6 %	59,8 %	78,4 %	69,5 %
Marathi	63,4 %	55,0 %	52,9 %	53,9 %	57,6 %
Odia	69,0 %	93,7 %	89,9 %	88,2 %	97,1 %
Tamil	10,1 %	59,8 %	17,4 %	55,2 %	65,2 %
Telugu	59,2 %	54,5 %	86,8 %	75,9 %	76,2 %
Gujarati	49,8 %	57,3 %	68,5 %	83,7 %	51,6 %
Celková	49 %	58 %	62 %	71 %	69 %

Tabulka 9.5: Výsledky změny hloubky konvolučních sítí

V tab. 9.5 jsou zaznamenány kompletní výsledky tohoto testu. V prvním řádku jsou uvdené počty výstupních kanálů (VK) první konvoluční vrstvy.

9.3 Zhodnocení konvolučních sítí

Konvoluční sítě nejsou velmi složitou architekturou. Jelikož se v práci používaly méně složité sítě, nebyla jejich trénovací doba vysoká. Jejich průměrná doba na natrénování jedné epochy byla téměř 2,5 hodiny, což je nižší než u dopředných. Nižšího trénovacího času zde bylo dosaženo zejména z důvodu omezení trénovacích dat (viz 4.1.1). Trénovací čas na epochu byl v rozmezí 1,5 až 5 hodin. V součtu bylo natrénováno skoro 200 hodin na této architektuře. Pro účely experimentů bylo vytvořeno 10 návrhů.

Celkově bylo natrénováno 12 modelů s průměrnou celkovou úspěšností přes 58 %. Rozmezí úspěšnosti bylo od 30 % do 71 %.

Jako u lineárních sítí zde působil největší problém jazyk Tamil. Tento jazyk míval nejčastěji nejhorší úspěšnost ze všech jazyků. Nejčastěji se jazyk Tamil klasifikoval jako Telugu.

9.3.1 Nejlepší konvoluční model

V tabulce 9.6 je znázorněn návrh modelu, který dosáhl nejlepších výsledků. Jak vyplývá z experimentů, nejlepší počet konvolučních bloků jsou 4. U konvolučních vrstev byla zvolena velikost jádra 5. Pro pool vrstvy byla zvolena funkce Max Pool. Aktivační funkce je vždy ReLU (s výjimkou výstupu, kde je logaritmická softmax funkce).

Tabulka 9.6: Nejlepší konvoluční model – návrh

Vrstva	Vstupní velikost	Výstupní velikost	Aktivační funkce
Vstup	33×39	$1 \times 33 \times 39$	-
Conv2D	$16 \times 33 \times 39$	$16 \times 33 \times 39$	-
BatchNorm	$16 \times 33 \times 39$	$16 \times 33 \times 39$	ReLU
Pool	$16 \times 33 \times 39$	$16 \times 16 \times 19$	-
Conv2D	$16 \times 16 \times 19$	$32 \times 16 \times 19$	-
BatchNorm	$16 \times 16 \times 19$	$16 \times 16 \times 19$	ReLU
Pool	$16 \times 16 \times 19$	$32 \times 8 \times 9$	-
Conv2D	$32 \times 8 \times 9$	$64 \times 8 \times 9$	-
BatchNorm	$64 \times 8 \times 9$	$64 \times 8 \times 9$	ReLU
Pool	$64 \times 8 \times 9$	$64 \times 4 \times 4$	-
Conv2D	$64 \times 4 \times 4$	$128 \times 4 \times 4$	-
BatchNorm	$128 \times 4 \times 4$	$128 \times 4 \times 4$	ReLU
Pool	$128 \times 4 \times 4$	$128 \times 2 \times 2$	-
Linear	1×512	1×256	-
Linear	1×256	1×6	LogSoftmax

Výsledky

Výsledky nejlepšího konvolučního modelu jsou zaznamenány v tabulce 9.7. Výsledky jsou do jisté míry podobné jako u dopředných sítí. Jazyk Tamil a Marathi působily největší problémy. Tamil byl nejčastěji klasifikován jako Telugu a jazyk Marathi jako Tamil. Tento deficit se nacházel i u dopředné architektury.

Hindi	78,4 %
Marathi	53,9 %
Odia	88,2 %
Tamil	55,2 %
Telugu	75,9 %
Gujarati	83,7 %
Celková	71 %

Tabulka 9.7: Nejlepší konvoluční model – výsledky

U tohoto modelu trvalo natrénovat jednu epochu 3,75 hodiny. Byl trénován na 6 epoch. I při trénování tohoto modelu bylo využito redukovaného počtu trénovacích dat (4.1.1).

Výsledky konvolučního návrhu v porovnání s dopředným si jsou velmi podobné. Celková úspěšnost se lišila pouze o 2 %. Problematický jazyk Tamil dosáhl u obou architektur podobných výsledků. Výsledky obou architektur ve spojení s MFCC příznaky jsou srovnatelné, i přesto zde byl zvolen dopředný návrh jako lepší. Zde by mohlo hrát roli i to, že dopředný návrh obsahoval více trénovacích dat, protože u konvoluční architektury se využíval omezený počet trénovacích dat.

10 Převzaté sítě

V průběhu práce byla snaha porovnat výsledky dosažené v rámci práce s jinými pracemi, které se zabývaly podobnou problematikou. Došlo také na porovnání výsledků vůči známým návrhům sítí.

10.1 Arzo Mahmood, Utku Köse

Práce od autorů Arzo Mahmood a Utku Köse [42], se zabývá docela odlišným problémem. Rozdílnost prací je v tom že místo identifikace jazyků se jedná o klasifikaci hlasových příkazů. V práci klasifikují příkazy taktéž pomocí MFCC příznaků. Tyto příznaky mají menší velikost, než jaké byly použity během této práce. V jejich práci jsou uvedené dvě architektury neuronových sítí.

Všechny natrénované modely podle těchto architektur byly natrénovány s počtem 6 epoch. Oba jejich návrhy byly do jisté míry upraveny vzhledem k závěrům, k jakým se v práci došlo. Největší změnou byla velikost samotného vstupu a výstupu. Velikost vstupu byla zvolena ta, která ze zde provedených testů vycházela nejlépe (33×39).

10.1.1 Návrhy

Jejich první návrh obsahoval jednodimenzionální konvoluční vrstvy, dropout vrstvy a jednu průměrovací pooling vrstvu. Vstup byl nejprve upraven pomocí konvoluční vrstvy s ReLU aktivační funkcí následované dropout vrstvou. Tato operace zde byla provedena dvakrát. Poté následovala jedna průměrovací pooling vrstva následovaná jednou lineární vrstvou jako výstup.

Druhý návrh se od prvního liší jen velmi málo. V tomto návrhu bylo využito více konvolučních vrstev (celkově 5) a již se zde nenacházela jakákoliv pool vrstva. Za každou konvoluční vrstvou se nacházela ReLU aktivační funkce následována dropout vrstvou. Na konci návrhu se nacházela jedna lineární vrstva pro výstup.

10.1.2 Výsledky

Pro problematiku identifikace jazyka se návrhy z této práce ukázaly jako nevhodné. Ve své práce uváděli i změny hyper-parametrů, i přes tyto změny se nepodařilo dosáhnout přijatelných výsledků. Zde je nutné zdůraznit, že Arzo Mahmood a Utku Köse v práci řeší odlišnou problematiku, a to rozpoznání povelů. Tento výsledek poukazuje taktéž na rozdílnost těchto problematik.

Pravděpodobná příčina nefunkčnosti těchto modelů je využívaní 1D konvolučních vrstev. V rámci práce byl proveden pokus s 1D konvolučními sítěmi, tyto sítě však vykazovaly stejné problémy jako tyto 9.1.1.

10.2 LeNet

LeNet je konvoluční neuronová síť, která byla představena roku 1989 Yann LeCunem [41]. Zde byl zreplikován návrh LeNet-5, pozměněn pouze o velikosti vstupu a výstupu jednotlivých vrstev. Síť byla pro potřeby práce lehce upravena na velikost vstupu, který v práci vycházel jako nejlepší.

Tabulka 10.1: LeNet návrh

Vrstva	Vstupní velikost	Výstupní velikost	Aktivační funkce
Vstup	33×39	33×39	-
Conv2D	33×39	$6 \times 33 \times 39$	Sigmoid
AveragePool	$6 \times 33 \times 39$	$6 \times 16 \times 19$	-
Conv2D	$6 \times 16 \times 19$	$16 \times 16 \times 19$	Sigmoid
AveragePool	$16 \times 16 \times 19$	$16 \times 8 \times 9$	-
Linear	1×1152	1×216	Sigmoid
Linear	1×216	1×6	LogSoftmax

10.2.1 Výsledky

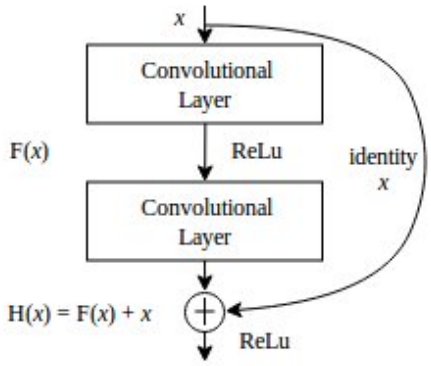
Výsledky byly do značné míry podobné jako výsledky konvolučních návrhů vytvořených v této práci. V tabulce 10.2 je vidět, že problematický jazyk Tamil je s úspěšností 26,6 % nejnižší, přičemž všechny ostatní jazyky dosahují úspěšnosti minimálně 49 %. Celková úspěšnost 68 % je velmi dobrá a dosahuje skoro hodnoty nejlepšího konvolučního návrhu vytvořeného v rámci práce.

Hindi	74,4 %
Marathi	49,4 %
Odia	99,2 %
Tamil	26,6 %
Telugu	70,3 %
Gujarati	97,9 %
Celková	68 %

Tabulka 10.2: LeNet výsledky

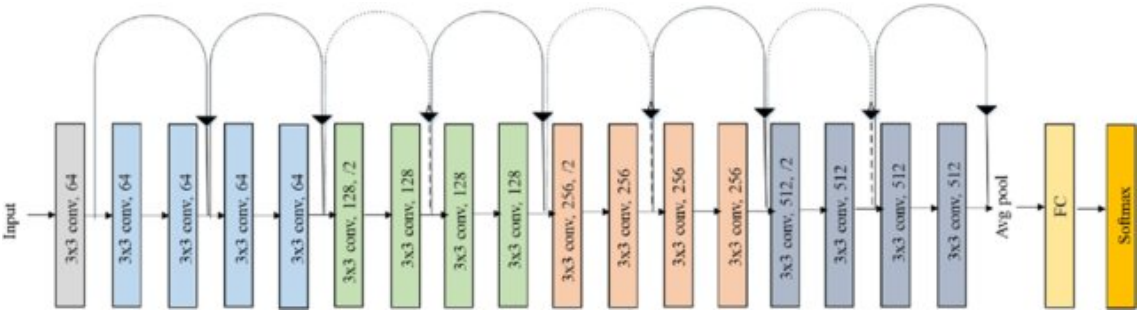
10.3 ResNet

ResNet neboli reziduální neuronová síť, představena roku 2016. [43] Tato síť se skládá z konvolučních bloků s ReLU aktivační a normalizační funkcí. Rozdílem oproti ostatním architekturám je, že vstupy jednotlivých bloků nejsou samotné výstupy předchozího bloku, ale taktéž jsou sečteny z výstupem předešlých bloků. Na obrázku 10.1 je nákres fungování a propojení jednotlivých bloků.



Obrázek 10.1: ResNet blok [44]

Je více typů ResNet sítí, zejména se liší v jejich hloubce (počet bloků). Pro trénování byla zvolena ta nejmenší síť ResNet18 (viz 10.2). Nejmenší síť byla zvolena hlavně z důvodu časových nároků na trénování. ResNet architektura už je komplexní a hluboká neuronová síť oproti modelům vytvořeným v rámci práce. I přestože trénování sítě probíhalo na třetině vstupů (omezené byly na základě velikosti okna), tak natrénování jedné epochy trvalo téměř sedm hodin.



Obrázek 10.2: ResNet18 návrh [45]

10.3.1 ResNet výsledky

Tabulka 10.3: ResNet výsledky

	Okno 15	Okno 7	Okno 3
Hindi	24,8 %	35,9 %	68,6 %
Marathi	0,0 %	5,6 %	49,2 %
Odia	5,4 %	28,1 %	66,2 %
Tamil	0,1 %	14,4 %	35,9 %
Telugu	33,4 %	19,0 %	32,6 %
Gujarati	99,9 %	100 %	86,0 %
Celková	24 %	31 %	56 %

ResNet byla natrénovaná třikrát. Každý natrénovaný model měl jiný počet trénovacích dat. Trénovací data byla omezena na základě změny velikosti okna. Velikost vstupu všech modelů byla vždy 33 (vycházelo se z poznatků u lineárních sítí viz 8.1). Všechny tři modely byly dále trénovány na 6 epoch.

Ze začátku nebyla známá časová náročnost sítí. První velikost okna byla zvolena 15. Takto omezený počet dat podal velmi nízké výsledky (viz 10.3), trénovací čas byl ovšem menší než 2 hodiny na epochu.

Další velikost okna byla menší než polovina (více jak dvojnásobek dat). Nárůst úspěšnosti oproti velikosti okna 15 nebyl nijak veliký, ale ani zanedbatelný. Nárůst v úspěšnosti byl o 7 % viz 10.3. Trénovací čas na epochu se zvýšil cca na dvojnásobek (3 hodiny na epochu).

Z předešlých dvou výsledků bylo zřejmé, že s rostoucím počtem dat roste i úspěšnost sítě. Velikost okna byla znovu snížena, tentokrát na velikost 3. U tohoto modelu trvalo natrénovat epochu skoro 7 hodin. U tohoto počtu dat už byl nárůst úspěšnosti oproti velikosti okna 7 téměř dvojnásobný. Podařilo se dosáhnout úspěšnosti 56 % (viz 10.3), kde jazyk Telugu dosahuje nejmenších výsledků.

Zde vznikl předpoklad, pokud by se síť natrénovala s plným počtem dat, mohla by podat ještě lepší výsledky, než kterých bylo dosaženo s omezenými daty. Další zvýšení počtu trénovacích dat, respektive snížení velikosti okna již nebylo možné, jelikož trénovací čas na daném stroji by byl už příliš vysoký.

11 Bottleneck příznaky

Příznaků z audio nahrávek existuje více druhů. V předchozí části byly využity MFCC příznaky. MFCC příznaky fungovaly poměrně dobře, nicméně se u nich nepodařilo přesáhnout úspěšnost 73 %.

Byl zde důvod pro změnu příznaků a zjištění, zda jiné příznaky mohou podat lepší výsledky. Zde byly využity bottleneck příznaky. Pro jejich vytvoření byl použit software od firmy Phonexia ve spojení s Vysokým učením technickým v Brně (BUT) [46]. Tento software generuje příznaky pomocí předtrénované neuronové sítě.

11.1 Generování příznaků

Pro generování byl využit program vytvořený v Pythnu, vytvořený spoluprací Phonexie a BUT. Tento program obsahuje tři předtrénované neuronové sítě. Každá ze sítí je naučená na jiné datové sadě. Datové sady se liší zejména počtem jazyků a jejich mluvčími. Pro účely práce byla zvolena síť BabelMulti. Tato síť byla trénována na datové sadě, která obsahovala 17 jazyků. Generování příznaků probíhá zpracováním WAVE audio souboru a zpracováním neuronovou sítí.

11.1.1 Software

Software pro generování bottleneck příznaků, byl zřejmě napsaný pro starší verzi Pythnu. Zde bylo nutné pro spuštění softwaru přepsat určité části kódu, které obsahovaly staré příkazy, které se již v Pythnu (ve verzi 3.8) nepoužívají. Některé funkce externích knihoven, které zde byly použity, již měly jinou podobu, či se musely použít novější alternativy funkcí.

Software je schopen generovat příznaky pouze z nahrávek s vzorkovací frekvencí 8 kHz. Bylo tedy nutné všechny nahrávky převést na 8 kHz. Pro změnu vzorkovací frekvence byl vytvořen jednoduchý skript, který všechny nahrávky převedl.

11.1.2 Příznaky

Příznaky jsou generovány z předtrénované neuronové sítě. Vstup této sítě je o velikosti 222. Vstupem není samotná nahrávka, ale již extrahované vlastnosti z audia. Tato síť obsahuje 4 skryté vrstvy, z toho jedna je vrstva s nižší dimenzionalitou (bottleneck vrstva). Šířka bottleneck vrstvy byla fixně nastavená na 80. Hodnota 80 způsobuje, že systém je robustnější a dosahuje tak lepších výsledku, oproti použití

nižší hodnoty. Toto číslo vychází z předešlého pozorování autorů. Ostatní lineární vrstvy měly šířku nastavenou fixně na 1 500. Aktivační funkce použité v síti byly sigmoid a na konci sítě byla použita softmax. [46]

Tato síť byla natrénována za účelem identifikace fonémů jazyků. Použité jazyky byly např. turečtina, vietnamština, bengálština, tamilština atd.

Vygenerovaná data

Vygenerovaná data byla stejného formátu jako MFCC příznaky. Rozdíl mezi MFCC a bottleneck příznaky byl ve výšce matice. MFCC příznaky měly rozměr $N \times 39$, bottleneck příznaky měly rozměr $N \times 80$. Jejich celková velikost byla 46,4 GB.

11.2 Použité návrhy

Pro změnu příznaků byly vybrány návrhy, které byly odladěny pro MFCC příznaky. Konkrétně se jednalo o návrhy, které byly v práci vybrány jako nejlepší. Jelikož se příznaky liší pouze v jejich dimenzi, nebylo nutné návrhy složitě upravovat.

Modely byly trénovány na 6 epoch. Počet dat byl zkrácen na podobné hodnoty jako v tabulce 4.1.1. Jelikož tyto příznaky mají větší rozměr, zabírají i větší počet v RAM paměti. Byl proto zvýšen počet skupin, do kterých se data rozdělují (viz 6.1.2).

11.2.1 Dopředný návrh

Při úpravě lineárního návrhu se lišila velikost vstupu a s tím navazující velikost výstupu z flatten vrstvy (viz 8.3). Šířka i hloubka návrhu zůstala zachována.

Výsledky po změně příznaků byly značně lepší než u MFCC příznaků. Celková úspěšnost vzrostla o 10 %. Úspěšnost u problematického jazyka Tamil vzrostla o 13 %, na 63 %. Ke zhoršení došlo u jazyků Gujarati a Telugu. Zhoršení bylo v jednotkách procent či pouze v setinách procent. Tento model s těmito příznaky dosáhl nejlepších výsledků z celé práce.

Hindi	98,1 %
Marathi	76,0 %
Odia	99,2 %
Tamil	61,0 %
Telugu	74,6 %
Gujarati	92,9 %
Celková	83 %

Tabulka 11.1: Lineární model s bottleneck příznaky

Matice zmatku

Jelikož tento model dosáhl v rámci práce nejlepšího výsledku, byla pro něj vytvořená tzv. matice zmatku. Tato matice ukazuje, jaká třída se pletla s jakou a kolikrát.

Matice zmatku pro tento model je zaznamenána v tabulce 11.2. Zeleně označená políčka ukazují správné klasifikace. Svislá osa ukazuje jak daný model vyhodnocoval testovací soubory. Jazyky Hindi a Odia se téměř nepletly s ostatními.

Problematický jazyk Tamil se nejvíce pletl s Telugu. To by mohlo být z důvodu, že spadají do stejné jazykové rodiny. Tamil se výrazně zaměňoval s téměř všemi jazyky. To by mohlo být způsobeno i tím, že trénovací data tohoto jazyku, pocházela

od velkého množství mluvčích. Jak vyplývá i z tabulky úspěšnosti, jazyky Hindi, Odia a Gujarati se nepletly téměř vůbec.

	Hindi	Marathi	Odia	Tamil	Telugu	Gujarati
Hindi	3769	721	6	129	28	0
Marathi	57	3553	16	34	40	0
Odia	13	64	3442	245	329	4.
Tamil	0	3	0	1880	1	12
Telugu	1	67	0	669	2267	202
Gujarati	3	267	7	124	375	2857

Tabulka 11.2: Matice zmatku

11.2.2 Konvoluční návrh

U konvolučního návrhu muselo dojít ke změně v lineárních vrstvách. Výstup z posledního konvolučního bloku (viz 9.1) měl vyšší rozměry, kvůli vyšším rozměrům příznaků. Zde byly upraveny pouze velikosti vstupů a výstupů dvou lineárních vrstev na konci.

Zde došlo taktéž k nárůstu celkové úspěšnosti o 10 %. Jazyky Tamil a Telugu dosáhly značného zhoršení. U jazyka Tamil došlo k zhoršení úspěšnosti o více než 15 % a jazyka Telugu o skoro 10 %. Zde je patrné, že síť funguje velmi dobře pro všechny jazyky, až na jazyk Tamil.

Hindi	98,3 %
Marathi	80,8 %
Odia	99,3 %
Tamil	39,8 %
Telugu	66,1 %
Gujarati	98,2 %
Celková	81 %

Tabulka 11.3: Konvoluční model s bottleneck příznaky

Konvoluční návrh dosáhl o 2 % horší celkové úspěšnosti než dopředný návrh. Díky tomu byl dopředný návrh zvolen jako lepší. Dále se výsledky obou návrhů nejvíce lišily v úspěšnosti jazyků Tamil a Telugu. Dopředný návrh zde podal pro tyto dva problematické jazyky značně lepší výsledky.

Závěr

Práce se zabývala identifikací šesti indických jazyků z audio nahrávky. Audio nahrávky pocházely ze soutěže MUCS 2021, která se zabývala příbuznou problematikou. V práci proběhlo navržení celé řady systémů, které měly tuto problematiku řešit. Pro identifikaci jazyka byly vybrány nízkoúrovňové vlastnosti jazyka (akustická stránka jazyka). Pro klasifikaci jazyků bylo zvoleno strojové učení, konkrétně neuronové sítě.

První přístup zvolený pro extrakci nízkoúrovňových vlastností byly Mel-frekvenční kepstrální koeficienty (MFCC). Tyto příznaky jsou standardně využívané pro extrakci nízkoúrovňových vlastností pro identifikaci jazyka. Jako další bylo zvoleno použití příznaků vygenerovaných z předtrénované neuronové sítě (bottleneck příznaky). Tento druhý typ příznaků byl využit až po odladění návrhů sítí na MFCC příznacích. Z architektur neuronových sítí byly vybrány dvě, a to dopředné a konvoluční neuronové sítě. Dopředné neuronové sítě jsou velmi často využívané pro své univerzální použití. Konvoluční neuronové sítě se velmi často využívají pro rozpoznání obrazů, ale nacházejí své uplatnění i v jiných oblastech. Zde byly zvoleny z důvodu poměrně nízkého výpočetního nároku.

U dopředných sítí bylo provedeno několik testů. První test se věnoval ověření závislosti velikosti vstupu na úspěšnost. Další test, který probíhal s tímto testem paralelně, byl test šířky dopředných sítí. Toto byl prvotní test. V tomto testu se ukázalo, že vyšší šířka sítě s menším kontextem podávají nejlepší výsledky. Následoval test změny hloubky sítě. U tohoto testu se s přibývající či klesající hloubkou sítě snižovala úspěšnost.

U konvoluční architektury se vycházelo z některých poznatků z předchozího pozorování, zejména velikosti vstupu. I přesto zde byl zopakován test s velikostí vstupu, ovšem v menší míře. Následoval test s velikostí konvolučního jádra a počtu lineárních vrstev na konci. Tyto dva testy neměly velký vliv na úspěšnost. Další test se věnoval hloubce konvoluční sítě. Z tohoto testu vzešlo, že drobné zvýšení hloubky oproti původnímu návrhu může vylepšit úspěšnost. Poslední parametr k otestování byl počet výstupních kanálů z konvolučních vrstev. Zde došlo k výraznému navýšení úspěšnosti. Zvýšení počtu výstupních kanálů razantně zvýšilo úspěšnost. Při snižování naopak docházelo ke značnému zhoršování funkčnosti systému.

Nejlepší návrhy odladěné pro MFCC příznaky byly následně velmi minimálně pozměněny pro bottleneck příznaky. Po výměně příznaků došlo ke znatelnému nárůstu úspěšnosti u obou architektur. Bottleneck příznaky se ukázaly jako vhodnější způsob reprezentace než MFCC příznaky. Obě architektury podaly pro zvolené příznaky uspokojivé výsledky. Výsledky nejlepších modelů se lišily v jednotkách procent. Do-

předné sítě však podávaly o trochu lepší výsledky než konvoluční. Za nejlepší systém byla vybrána dopředná síť se vstupem v podobě bottleneck příznaků, s celkovou úspěšností 83 %.

Cíle práce byly splněny a nad rámec zadání byla vytvořena aplikace pro lepší demonstraci funkčnosti systémů. Tato aplikace zahrnuje většinu natrénovaných modelů, které jsou v práci zmíněné. Aplikace byla vytvořená v Pythnu pomocí modulu Tkinter. Aplikace umí načítat natrénované modely, zobrazit si jejich podrobnosti a otestovat jednotlivé modely na vygenerovaných souborech s příznaky.

Existuje více možností rozšíření práce. První rozšíření práce by mohlo nastat v podobě použití vlastních bottleneck příznaků. Toto vlastní řešení by mohlo přispět k vysokému nárůstu úspěšnosti. Další rozšíření, které se nabízí, je přidání všech oficiálně uznaných indických jazyků. Práce by se dala případně rozšířit o složitější architektury neuronových sítí.

Seznam použité literatury

1. DIWAN, Anuj; VAIDEESWARAN, Rakesh; SHAH, Sanket; SINGH, Ankita; RAGHAVAN, Srinivasa; KHARE, Shreya; UNNI, Vinit; VYAS, Saurabh; RAJPURIA, Akash; YARRA, Chiranjeevi et al. Multilingual and code-switching ASR challenges for low resource Indian languages. *arXiv preprint arXiv:2104.00235*. 2021.
2. DESHWAL, Deepti; SANGWAN, Pardeep; KUMAR, Divya. Feature extraction methods in language identification: a survey. *Wireless Personal Communications*. 2019, roč. 107, č. 4, s. 2071–2103.
3. ZISSMAN, Marc A. Automatic language identification using Gaussian mixture and hidden Markov models. In: *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. 1993, sv. 2, s. 399–402.
4. HERACLEOUS, Panikos; TAKAI, Kohichi; YASUDA, Keiji; MOHAMMAD, Yasser; YONEYAMA, Akio. Comparative Study on Spoken Language Identification Based on Deep Learning. In: *2018 26th European Signal Processing Conference (EUSIPCO)*. 2018, s. 2265–2269. Dostupné z DOI: 10.23919/EUSIPCO.2018.8553347.
5. ZAZO, Ruben; LOZANO-DIEZ, Alicia; GONZALEZ-DOMINGUEZ, Javier; T. TOLEDANO, Doroteo; GONZALEZ-RODRIGUEZ, Joaquin. Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks. *PLOS ONE*. 2016, roč. 11, č. 1, s. 1–17. Dostupné z DOI: 10.1371/journal.pone.0146917.
6. GRIS, Lucas Rafael Stefanel; JUNIOR, Arnaldo Candido. Automatic Spoken Language Identification using Convolutional Neural Networks. In: *Anais do XVII Congresso Latino-Americano de Software Livre e Tecnologias Abertas*. 2020, s. 16–20.
7. MONTAVON, Gregoire. Deep learning for spoken language identification. In: *NIPS Workshop on deep learning for speech recognition and related applications*. 2009, s. 1–4.
8. JIANG, Bing; SONG, Yan; WEI, Si; LIU, Jun-Hua; MCLOUGHLIN, Ian Vince; DAI, Li-Rong. Deep Bottleneck Features for Spoken Language Identification. *PLOS ONE*. 2014, roč. 9, č. 7, s. 1–11. Dostupné z DOI: 10.1371/journal.pone.0100795.

9. MALO, Grisard; MOTLICEK, Petr; ALLOUCHI, Wissem; BAERISWYL, Michael; LAZARIDIS, Alexandros; ZHAN, Qingran. SPOKEN LANGUAGE IDENTIFICATION USING LANGUAGE BOTTLENECK FEATURES. 2019. Dostupné také z: http://infoscience.epfl.ch/record/270140.

10. MOSELHY, Abdallaa Mohammed; ABDELNAIEM, Abdelaziz Alsayed. LPC and MFCC performance evaluation with artificial neural network for spoken language identification. *International Journal of Signal Processing, Image Processing and Pattern Recognition*. 2013, roč. 6, č. 3, s. 55.

11. KOOLAGUDI, Shashidhar G.; RASTOGI, Deepika; RAO, K. Sreenivasa. Identification of Language using Mel-Frequency Cepstral Coefficients (MFCC). *Procedia Engineering*. 2012, roč. 38, s. 3391–3398. ISSN 1877-7058. Dostupné z DOI: https://doi.org/10.1016/j.proeng.2012.06.392. INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING.

12. RICHARDSON, Fred; REYNOLDS, Douglas; DEHAK, Najim. Deep Neural Network Approaches to Speaker and Language Recognition. *IEEE Signal Processing Letters*. 2015, roč. 22, č. 10, s. 1671–1675. Dostupné z DOI: 10.1109/LSP.2015.2420092.

13. JOTHILAKSHMI, S.; RAMALINGAM, V.; PALANIVEL, S. A hierarchical language identification system for Indian languages. *Digital Signal Processing*. 2012, roč. 22, č. 3, s. 544–553. ISSN 1051-2004. Dostupné z DOI: https://doi.org/10.1016/j.dsp.2011.11.008.

14. BISWAS, Mainak; RAHAMAN, Saif; AHMADIAN, Ali; SUBARI, Kamalularifin; SINGH, Pawan Kumar. Automatic spoken language identification using MFCC based time series features. *Multimedia Tools and Applications*. 2022, s. 1–31.

15. Dostupné také z: https://translatedlabs.com/spoken-language-identifier.

16. Dostupné také z: https://www.phonexia.com/en/product/language-identification/.

17. SUNDARARAJAN, Duraisamy. *The discrete Fourier transform: theory, algorithms and applications*. World Scientific, 2001.

18. RAMASESHAN, Ajay. *Application of Multiway Methods for Dimensionality Reduction to Music*. 2013. Dis. pr.

19. ALIM, Sabur Ajibola; RASHID, Nahrul Khair Alang. Some Commonly Used Speech Feature Extraction Algorithms. In: LOPEZ-RUIZ, Ricardo (ed.). *From Natural to Artificial Intelligence*. Rijeka: IntechOpen, 2018, kap. 1. Dostupné z DOI: 10.5772/intechopen.80419.

20. JAROLÍN, Milan. Optimalizace rozmístění pásmových filtrů v MFCC s ohledem na zpracovávanou množinu řečníků. 2012.

21. AKPUDO, Ugochukwu; HUR, Jang-Wook. A Cost-Efficient MFCC-Based Fault Detection and Isolation Technology for Electromagnetic Pumps. *Electronics*. 2021, roč. 10, s. 439. Dostupné z DOI: 10.3390/electronics10040439.

22. YU, Dong; SELTZER, Michael L. Improved bottleneck features using pretrained deep neural networks. In: *Twelfth annual conference of the international speech communication association*. 2011.

23. WU, Yuzhong; LEE, Tan. Reducing Model Complexity for DNN Based Large-Scale Audio Classification. 2017.

24. BAEK, Jieun; CHOI, Yosoon. Deep Neural Network for Predicting Ore Production by Truck-Haulage Systems in Open-Pit Mines. *Applied Sciences*. 2020, roč. 10, s. 1657. Dostupné z DOI: 10.3390/app10051657.

25. ATAEI, Mehdi; BUSSMANN, Markus; SHAAYEGAN, Vahid; COSTA, Franco; HAN, Sejin; PARK, C.B. NPLIC: A Machine Learning Approach to Piecewise Linear Interface Construction. 2020.

26. LI, Jing; CHENG, Ji-hang; SHI, Jing-yuan; HUANG, Fei. Brief introduction of back propagation (BP) neural network algorithm and its improvement. In: *Advances in computer science and information engineering*. Springer, 2012, s. 553–558.

27. ZHANG, Zhilu; SABUNCU, Mert. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In: BENGIO, S.; WALLACH, H.; LAROCHELLE, H.; GRAUMAN, K.; CESA-BIANCHI, N.; GARNETT, R. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018, sv. 31. Dostupné také z: https://proceedings.neurips.cc/paper/2018/file/f2925f97bc13ad2852a7a551802feea0-Paper.pdf.

28. BOTTOU, Léon et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*. 1991, roč. 91, č. 8, s. 12.

29. YAMASHITA, Rikiya; NISHIO, Mizuho; DO, Richard Kinh Gian; TOGASHI, Kaori. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*. 2018, roč. 9, č. 4, s. 611–629.

30. BRITANNICA. Indian languages. 2019. Dostupné také z: https://www.britannica.com/topic/Indian-languages.

31. HSU, Fengchi; ZHIZHIN, Mikhail; TILOTTAMA, Ghosh; ELVIDGE, Christopher; TANEJA, Jay. The Annual Cycling of Nighttime Lights in India. *Remote Sensing*. 2021, roč. 13, s. 1199. Dostupné z DOI: 10.3390/rs13061199.

32. JOSHI, R Malt; NAKAMURA, Pooja; CHATTERJEE, Nandini. Reading Research and Practice: Indian Perspective. *New Directions for Child and Adolescent Development*. 2017, roč. 2017, s. 43–53. Dostupné z DOI: 10.1002/cad.20222.

33. FAZAL, SHAHAB; ALI, Md Kaikubad. Some Insight into the Language Census of India 2011. 2019, roč. 66.

34. ILLINOIS, Universiti of. About Hindi. [B.r.]. Dostupné také z: https://linguistics.illinois.edu/hindi/about-hindi.

35. BRITANNICA. Hindi language. 2019. Dostupné také z: https://www.britannica.com/topic/Hindi-language.

36. BRITANNICA. Marathi language. 2021. Dostupné také z: https://www.britannica.com/topic/Marathi-language.

37. BRITANNICA. Odia language. 2020. Dostupné také z: https://www.britannica.com/topic/Odia-language.

38. BRITANNICA. Tamil language. 2022. Dostupné také z: https://www.britannica.com/topic/Odia-language.

39. BRITANNICA. Telugu language. 2019. Dostupné také z: https://www.britannica.com/topic/Telugu-language.

40. KORZINEK, DANIEL. *PyHTK*. 2022-01-22. Dostupné také z: https://github.com/danijel3/PyHTK.

41. LECUN, Yann. Generalization and network design strategies. In: *Connectionism in perspective*. Ed. PFEIFER, R.; SCHRETER, Z.; FOGELMAN, F.; STEELS, L. Elsevier, 1989.

42. MAHMOOD, Arzo; KÖSE, Utku. Speech recognition based on Convolutional neural networks and MFCC algorithm. 2021, roč. 1, s. 6–12.

43. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. In: 2016, s. 770–778. Dostupné z DOI: 10.1109/CVPR.2016.90.

44. SURINTA, Olarik; KHAMKET, Thananchai. Recognizing Pornographic Images using Deep Convolutional Neural Networks. In: 2019. Dostupné z DOI: 10.1109/ECTI-NCON.2019.8692296.

45. RAMZAN, Farheen; KHAN, Muhammad Usman; REHMAT, Asim; IQBAL, Sajid; SABA, Tanzila; REHMAN, Amjad; MEHMOOD, Zahid. A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks. *Journal of Medical Systems*. 2019, roč. 44. Dostupné z DOI: 10.1007/s10916-019-1475-2.

46. FÉR, Radek; MATĚJKA, Pavel; GRÉZL, František; PLCHOT, Oldřich; VESELÝ, Karel; ČERNOCKÝ, Jan Honza. Multilingually trained bottleneck features in spoken language recognition. *Computer Speech & Language*. 2017, roč. 46, s. 252–267. ISSN 0885-2308. Dostupné z DOI: https://doi.org/10.1016/j.csl.2017.06.008.