TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# Detekce překrývajících se úseků řeči v audiozáznamech s využitím hlubokých neuronových sítí

## Diplomová práce

*Studijní program:* N2612 – Elektrotechnika a informatika

*Studijní obor:* 1802T007 – Informační technologie

*Autor práce:* **Bc. Jakub Kovář**

*Vedoucí práce:* Ing. Lukáš Matějů, Ph.D.

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

**Zadání diplomové práce**

# Detekce překrývajících se úseků řeči v audiozáznamech s využitím hlubokých neuronových sítí

| | |
|---|---|
| *Jméno a příjmení:* | **Bc. Jakub Kovář** |
| *Osobní číslo:* | M20000168 |
| *Studijní program:* | N2612 Elektrotechnika a informatika |
| *Studijní obor:* | Informační technologie |
| *Zadávající katedra:* | Ústav informačních technologií a elektroniky |
| *Akademický rok:* | **2021/2022** |

**Zásady pro vypracování:**

1. Seznamte se s problematikou detekce překrývajících se úseků řeči v audio nahrávkách.
2. Připravte a anotujte datové sady pro trénování, validaci a testování neuronových sítí pro účely detekce překrývajících se úseků řeči. Pro anotaci reálných nahrávek vytvořte podpůrnou aplikaci.
3. Navrhněte a natrénujte vybrané architektury neuronových sítí pro detekci překrývajících se úseků řeči.
4. Natrénované detektory experimentálně vyhodnoťte a porovnejte všechny dosažené výsledky.

*Rozsah grafických prací:* dle potřeby dokumentace
*Rozsah pracovní zprávy:* 40-50
*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] MÁLEK, Jiří a Jindřich ŽĎÁNSKÝ. Voice-Activity and Overlapped Speech Detection Using x-Vectors. Text, Speech, and Dialogue. Cham: Springer International Publishing, 2020, s. 366-376. Lecture Notes in Computer Science. ISBN 978-3-030-58322-4.


*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 diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou 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 diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou 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 diplomová 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.

16. 5. 2022 Bc. Jakub Kovář

# Detekce překrývajících se úseků řeči v audiozáznamech s využitím hlubokých neuronových sítí

## Abstrakt

Tato práce se zabývá úlohou klasifikace neznámých audio nahrávek pro účely detekce překrývajících se úseků řeči. Práce zahrnuje popis překrývajících se úseků řeči a současných řešení její detekce v souvislosti především s neuronovými sítěmi. Jsou zde popsány metodiky a principy neuronových sítí a technologie použité pro účely vytvoření detektoru překrývajících se úseků řeči pomocí neuronových sítí. Pro trénování, validaci a vyhodnocení jsou vytvořeny umělé a ručně anotované datové sady. Reálné datové sady vznikly ruční anotací audio nahrávek společně s použitím již existujících reálných datových sad. K vytváření umělých dat, trénování a testování sítě byly vytvořeny programy v jazyce Python. Pro anotaci reálných dat vznikla aplikace v MATLAB-u. Jsou zde navrženy lineární, konvoluční a time delay neuronové sítě, které jsou experimentálně laděny. Nejlepší z nich jsou trénovány na úplné trénovací datové sadě. Natrénované sítě dokázaly dosáhnout velmi vysoké přesnosti detekce na umělých datech. Z provedených experimentů lze usoudit, že konvoluční nebo time delay architektury by měly na reálných datech dosáhnout stejně dobrých výsledků. V případě, že by měli k dispozici větší množství reálných trénovacích dat.

**Klíčová slova:** překrývající se řeč, MFCC, lineární neuronové sítě, time delay neuronové sítě, konvoluční neuronové sítě

# Crosstalk Detection in Audio Recordings Using Deep Neural Networks

## Abstract

This thesis describes clasification task of overlapping speech detection with neural networks. It includes what is an overlapped speech and what are the types of neural networks used for detecting it. There are explanations of neural network principles and their training methodologies. Also there are synthetic, real training, validation and testing datasets created. The real dataset consists of two parts. One part was created by manual anotation of audio recordings. The other part was taken from manually anotated existing datasets. Python script generating synthetic overlapping speech was made for creating synthetic datasets. MATLAB application was created for annotating real audio recordings. Linear, convolutional and time delay neural networks were used for detecting overlapping speech. These networks were experimentally tuned at first. Then the best of them were evaluated on test datasets.Neural networks managed to achieve very high accuracy on synthetic testing dataset. Based on experimental results, it is feasible that the convolutional or time delay neural networks would achieve similar results if they would be given more real training data.

**Keywords:** overlapping speech, MFCC, linear neural networks, convolutional neural networks, time delay neural networks

# Obsah

Seznam zkratek . . . . . 14

**1 Úvod** **15**

**2 Úvod do problematiky** **17**

2.1 Překrývající se řeč . . . . . 18

2.2 Stav poznání . . . . . 20

**3 Použité technologie** **23**

3.1 Příznaky audio signálu . . . . . 23

3.2 Neuronové sítě . . . . . 24

3.3 Princip fungování neuronových sítí . . . . . 25

3.4 Způsoby učení neuronových sítí . . . . . 25

3.5 Aktivační funkce . . . . . 26

3.5.1 Sigmoid . . . . . 27

3.5.2 Hyperbolický tangens . . . . . 27

3.5.3 Lineární jednotky . . . . . 27

3.6 Princip učení neuronových sítí . . . . . 28

3.6.1 Chybová funkce binární křížová entropie . . . . . 28

3.6.2 Optimalizační algoritmus Adam . . . . . 28

3.7 Pokročilé metody trénování neuronových sítí . . . . . 29
3.7.1 Dropout . . . . . 29
3.7.2 Normalizace . . . . . 30
3.8 Vybrané architektury neuronových sítí . . . . . 32
3.8.1 Lineární neuronové sítě . . . . . 32
3.8.2 Konvoluční neuronové sítě . . . . . 33
3.8.3 Time delay neuronové sítě . . . . . 35

**4 Vytvořené programy** **37**
4.1 GenData . . . . . 37
4.2 TrainNet . . . . . 38
4.3 TestNet . . . . . 39
4.4 Program na anotaci dat . . . . . 40

**5 Použité datové sady** **42**
5.1 Příprava audio souborů . . . . . 43
5.2 Trénovací a validační . . . . . 44
5.3 Vývojová . . . . . 45
5.4 Testovací . . . . . 45

**6 Experimentální ladění neuronových sítí** **47**
6.1 Způsob vyhodnocení . . . . . 47
6.1.1 Zpracování výstupu sítě . . . . . 47
6.1.2 Metriky . . . . . 47
6.2 Navržený přístup řešení . . . . . 48

6.3 Lineární neuronová síť . . . . . 49
6.3.1 Experiment s šířkou kontextu . . . . . 49
6.3.2 Experiment s hloubkou a šířkou . . . . . 50
6.3.3 Experiment s aktivačními funkcemi . . . . . 51
6.4 Konvoluční neuronová síť . . . . . 52
6.4.1 Experiment s počtem kanálů konvolučních vrstev . . . . . 52
6.4.2 Experiment s počtem konvolučních a redukčních vrstev . . . . 53
6.4.3 Rozšíření trénovací datové sady . . . . . 54
6.4.4 Experiment trénování ve dvou fázích . . . . . 54
6.4.5 Experiment trénování v jedné fázi . . . . . 55
6.4.6 Experiment s aktivační funkcí ELU . . . . . 57
6.5 Time delay neuronová síť . . . . . 57
6.5.1 Experiment s hloubkou a šířkou sítě . . . . . 57
6.5.2 Experiment s rozšířeným kontextem . . . . . 58
6.5.3 Experiment s plným kontextem . . . . . 59
6.6 Testování finálních neuronových sítí . . . . . 60
6.6.1 Finální architektura lineární neuronové sítě . . . . . 60
6.6.2 Finální architektura konvoluční neuronové sítě . . . . . 61
6.6.3 Finální architektura time delay neuronové sítě . . . . . 62
6.6.4 Výsledky . . . . . 62
6.7 Poznámky k trénování . . . . . 64

**7 Závěr** **65**

**Použitá literatura** **70**

**Přílohy** **71**

# Seznam obrázků


2.1 Druhy překrývající se řeči[6] . . . . . 19

3.1 Diagram získání MFCC koeficientů [18] . . . . . 24

3.2 Umělý neuron . . . . . 25

3.3 Ukázka postupu optimalizačního algoritmu [23] . . . . . 29

3.4 Obyčejná síť a síť s dropout [25] . . . . . 30

3.5 Druhy normalizace [27] . . . . . 31

3.6 Dopředná plně spojitá lineární sít [30] . . . . . 32

3.7 Vztah vstupních a výstupních kanálů konvoluční vrstvy [31] . . . . . 34

3.8 Ukázka fungování konvolučního jádra [32] . . . . . 35

3.9 Ukázka architektury time delay [34] . . . . . 36

4.1 Diagram vytváření umělých dat . . . . . 38

4.2 Vzhled vniklého anotačního programu . . . . . 40

6.1 Architektura lineární neuronové sítě . . . . . 60

6.2 Architektura konvoluční neuronové sítě . . . . . 61

# Seznam tabulek

5.1 Přehled použitých dat . . . . . 43

6.1 *F-value* výsledků experimentu s vstupním kontextem . . . . . 49

6.2 *F-value* výsledků experimentu s hloubkou a šířkou sítě na validační sadě . . . . . 50

6.3 *F-value* výsledků experimentu s hloubkou a šířkou sítě na vývojové sadě . . . . . 50

6.4 *F-value* výsledků experiment s aktivačními funkcemi . . . . . 51

6.5 *F-value* výsledků experimentu s kanály konvolučních vrstev . . . . . 52

6.6 *F-value* výsledků experimentu s počtem konvolučních a redukčních vrstev . . . . . 53

6.7 *F-value* výsledků experimentu trénování ve dvou fázích s počtem vrstev bez normalizace . . . . . 54

6.8 *F-value* výsledků experimentu trénování ve dvou fázích s počtem vrstev s normalizací . . . . . 55

6.9 *F-value* výsledků experimentu trénování v jedné fázi s počtem vrstev s normalizací . . . . . 56

6.10 *F-value* výsledků experimentu trénování v jedné fázi s počtem vrstev bez normalizace . . . . . 56

6.11 *F-value* výsledků experimentu s aktivační funkcí ELU . . . . . 57

6.12 *F-value* výsledků experimentu s šířkou a hloubkou sítě na validačních datech . . . 58

6.13 *F-value* výsledků experimentu s šířkou a hloubkou sítě na vývojových datech . . . 58

6.14 *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným kontextem na validačních datech . . . 59

6.15 *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným kontextem na vývojových datech . . . 59

6.16 *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným plným kontextem na validačních datech . . . 59

6.17 *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným plným kontextem na vývojových datech . . . 59

6.18 Parametry vrstev finální lineární neuronové sítě . . . 60

6.19 Parametry vrstev výsledné konvoluční neuronové sítě . . . 61

6.20 Parametry vrstev výsledné time delay neuronové sítě . . . 62

6.21 *F-value* výsledků finálních sítí . . . 62

6.22 Výsledky finálních sítí na umělé testovací sadě . . . 63

6.23 Výsledky finálních sítí na reálné testovací sadě . . . 63

# Seznam zkratek

**MFCC** Mel-frequency cepstral coefficients, Mel-frekvenční kepstrální koeficienty

**HMM** Hidden Markov model, Skrytý Markovův model

**LSTM** Long short-term memory, typ článku používaného v rekurentních neuronových sítích

**CRNN** Convolutional recurrent neural network, Konvolučně rekurentní neuronová síť

**ReLU** Rectified Linear Unit, Usměrněná lineární jednotka

**ELU** Exponential linear unit, Exponenciální lineární jednotka

**RGB** Red Green Blue, aditivní způsob míchání barev používaný pro obrázky

**CUDA** Compute Unified Device Architecture, architektura grafických karet

**HTK** The Hidden Markov Model toolkit, Sada nástrojů pro skryté Markovovi modely

**CSV** Comma separated values, hodnoty oddělené čárkami

**XML** Extensible Markup Language, rozšiřitelný značkovací jazyk

**ITE** Ústav informačních technologií a elektroniky Technické university v Liberci

# 1 Úvod

V oblasti zpracování řeči už dlouhou dobu dominuje strojové učení, dříve to byly skryté Markovovi modely, dnes to jsou neuronové sítě. Ty jsou s dostatečným množstvím dat schopny se naučit přímo ze zvukových nahrávek rozpoznávat řeč. Nebo také zvládají rozpoznat mluvčího. [1] Neuronové sítě se dají aplikovat na různé druhy zpracování řečí. Pomocí nich je možné provádět rozpoznání řeči na úrovni člověka [2] nebo také umožňují určit, kdo v daný okamžik mluví. [3] Dále jsou schopny se naučit například rozpoznávat emoce mluvčího. [4]

Neuronové sítě bývají robustnější oproti starším přístupům, avšak stále předpokládají vcelku čisté prostředí bez výrazného rušení a pouze jediného řečníka. Právě v případech, kde mluví více lidí zároveň, mají tyto sítě největší problém. K vyřešení tohoto problému v oblasti strojového učení je potřeba nejdříve dokázat určit zda dochází k překrývající se řeči či nikoliv. K tomuto slouží detektory překrývajících se úseků řeči. Překrývající se řeč je poměrně významně zastoupena v normální řeči, proto čím se systémy zpracování řeči zlepšují, tím je takováto detekce důležitější pro možné další zlepšení. Proto je použití takovéhoto detektoru vhodné prakticky v každé aplikaci.

Hlavním problémem vývoje detektoru překrývajících se úseků řeči je, že je třeba brát ohled i na neřečové zvuky, které mohou vnášet do procesu učení neuronových sítí chyby. Také je pro úspěšné natrénování potřeba velké množství anotovaných dat, kterých je velmi málo.

Samotný detektor se definuje podle použitých příznaků a použitého klasifikátoru, který podle daných příznaků určuje zda se jedná o překrývající se řeč. Nejpoužívanější příznaky pro úlohu detekce překrývající se řeči jsou Mel-frekvenční kepstrální

koeficienty (MFCC). Nejčastěji se klasifikátor v detektoru překrývající se řeči realizuje pomocí konvolučních sítí, rekurentních sítí nebo jejich kombinací. Případně neuronové sítě používající bloky s překračujícím spojením jako sítě typu ResNet. Dále se používají tzv. time delay architektury.

Tato práce slouží k vytvoření vhodného datového balíčku, tvořeného českým a slovenským jazykem, určeného na trénování a testování neuronových sítí jakožto detektorů překrývajících se úseků řeči. Práce také poslouží k průzkumu a porovnání architektur neuronových sítí pro tuto aplikaci.

Cílem této práce je provést průzkum a seznámení s uvedenou problematikou spolu s jejím aktuálním výzkumem. Dále je cílem vytvořit datovou sadu pro trénování, validaci a testování neuronových sítí, jejíž součástí budou ručně anotovaná reálná data. Poté je nutné experimentálně natrénovat a vyhodnotit vybrané architektury neuronových sítí.

# 2 Úvod do problematiky

V běžné řeči je časté, že nastávají okamžiky, kdy přes sebe mluví více lidí najednou. Překrývající se úseky řeči jsou velkým zdrojem chyb v systémech zpracování řeči. Pokud se v audio záznamu nachází překrývající se řeč, není rozpoznávací systém bez detekce překryvů schopný přesně určit, co bylo řečeno, případně kým. Rozpoznávače typicky potřebují nalézt začátky a konce slov, což se v překrývající řeči značně komplikuje. Zkomplikuje se, protože se celá doba, kdy došlo k překryvu, označí jako jedno slovo. Systém, který se z takto označeného úseku pokusí dekódovat vyslovené slovo selže, protože nebude tento úsek odpovídat ani jednomu z vyslovených slov. Toto platí i v případě, kdy systém nalezne hranice slov korektně. Případně systémy, které fungují na úrovni fonémů, mají podobný problém, protože i kdyby detekovaly správně jednotlivé fonémy, systém neví, že z nich má poskládat dvě a více slov. Detekce překrývajících se úseků řeči se užívá k odstranění těchto chyb, a tím dojde ke zlepšení jejich funkčnosti.

Samotná úloha detekce překrývajících se úseků řeči se dá rozdělit na dvě části. První řeší způsob získání příznaků a jejich druh z neznámých audio záznamů. Druhá provádí klasifikaci nahrávek na základě těchto příznaků.

Detektor typicky slouží k předzpracování zvukového signálu, kde najde případné překryvy, které jsou následně zpracovány podle potřeby. Mezi hlavní aplikace patří přepis řeči na text, kde pokud systém ví, že mluví více lidí naráz, může se pokusit signály nějakým způsobem rozdělit, aby je byl schopný rozpoznávat samostatně. Nebo přestane po tuto dobu přepisovat. Další aplikace je rozeznávání identity mluvčích, u které je možné použít obdobné vylepšení. [5] Hlavní obtíž je, že neexistuje jednoznačný způsob jak určit překrývající se úsek řeči. Ve smyslu, že určení přesné

hranice je velice subjektivní a není jednotný a jednoznačný způsob jak jí určit. Což do jisté míry platí i u samotné řeči, že není jednoznačně dané kdy přesně začíná a končí promluva. Musí se třeba určit nějaká hranice síly signálů například na základě energie. Což v případě ruční anotace není možné, také z důvodu, že je těžší pro člověka rozeznat slabý signál za přítomnosti silnějšího.

## 2.1 Překrývající se řeč

Lidská řeč se skládá z několika vrstev, mimo samotnou informaci také zahrnuje citový obsah. Důležitý předpoklad pro plynulou řeč je předávání slova v konverzaci, které je řízené podvědomými faktory jako je intonace, sémantické náznaky, výraz obličeje, oční kontakt, dech a gestikulace. V rámci předávání slova v konverzaci hraje překryv řeči důležitou roli. Překryv často vzniká v místě potencionální výměny slova jako důsledek fungování předávání slova. Překrývající se řeč se rozlišuje [6] [7] na 4 druhy:

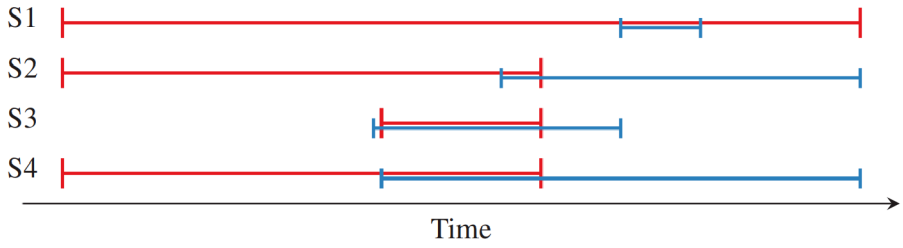
**S1** krátká zpětná vazba bez přerušení mluvčího

**S2** předčasné převzání slova při výměně řečníků

**S3** současný začátek promluvy po delším tichu

**S4** skočení do řeči za účelem převzání slova

V první situaci S1 chce posluchač pouze poskytnout zpětnou vazbu řečníkovi. To může být doplněno dalším způsobem jako je kývání hlavou, nebo jiné gestikulace. Neproběhne výměna mluvčích. S2 představuje situaci, kdy posluchač ví, že aktuální řečník ukončí promluvu. Avšak kvůli nedostatečné zpětné vazbě začne svoji promluvu příliš brzy. Třetí situace S3 vznikne, když po delším tichu začnou náhodně mluvit oba mluvčí. Situace S4 ukazuje narušení klasického předávání slova. Posluchač začne mluvit během promluvy někoho jiného za účelem stát se řečníkem. Na obrázku 2.1 jsou znázorněny všechny tyto druhy. Kde červeně a modře je označena řeč mluvčích.




Obrázek 2.1: Druhy překrývající se řeči[6]

V této práci se především pro zkrácení výrazu „překrývající se úsek řeči“ bude používat anglická fráze „crosstalk“, případně „crossnoise“. Crosstalk je definován jako překrývající se úsek řeči další řečí. A crossnoise je překrývající se řeč s citoslovci jako je *hmm*, *haha* a podobné. Kašel byl také uvažován jako crossnoise, i když není považován za řeč. Je nutné podotknout, že výraz „crossnoise“ je výraz používaný pouze v této práci, a to zejména pro snadnější porozumění. Obecně se takto překrývající řeč nerozděluje, protože se v obou případech jedná o překryv a není až tolik důležité, jestli šlo o citoslovce či nikoliv. V této práci a v použitých datech bylo pro toto rozdělení rozhodnuto na základě možnosti širšího zkoumání této problematiky. I když si většina lidí dokáže dobře představit, jak taková překrývající se řeč vypadá, je důležité ji definovat přesně, alespoň v rámci této práce především pro účely anotování vytvářené datové sady a následné vyhodnocení funkčnosti neuronové sítě. V rámci této práce se pro účely ruční anotace crosstalk-u a crossnoise překryv definuje následovně. Překryv je označen od doby, kdy bylo možné rozeznat promluvu dvou a více mluvčích naráz. Při poslechové hlasitosti na úrovní normální konverzace. Až do doby, kdy není možné rozeznat promluvu více lidí naráz. To je důležité, protože může nastat situace, kdy technicky dojde k překryvu, ale kvůli velkým rozdílům hlasitostí je lidským uchem slyšet pouze jeden mluvčí. Mezery jsou uvažovány jen mezi slovy, nikoliv v rámci slov. To znamená, že krátké mezery mezi slabikami se neuvažují jako mezery v překryvu řeči. Také stojí za zmínku to, že pokud se těsně za sebou prakticky bez mezery vymění řečníci, tak to lidskému uchu může znít jako crosstalk, i když se o crosstalk samozřejmě nejedná.

## 2.2 Stav poznání

Do doby něž se začali používat neuronové sítě, se používali především skryté Markovovy modely (HMM). Tato práce [8] byla mezi prvními, kde uvažovali využití detekce překrývajících se úseků řeči pro zlepšení systému zpracování řeči. V této době začaly být tyto systémy natolik přesné, že překrývající se úseky řeči začali způsobovat výrazné chyby. V této práci autoři představili systém, jehož součástí je detektor překryvů. S jeho pomocí dosáhli zlepšení. Tyto výsledky jasně ukázaly, že existuje dobrý důvod proč je důležité detekovat překrývající se úseky řeči.

Další práce, která stavěla na [8] je [9]. V ní autoři využívají skryté Markovovy modely na detekci překrývající se řeči. K tomu využívají kepstrální koeficienty a nově prozodické složky jako je výška hlasu, intenzita a formanty. Za pomocí inkluze těchto dodatečných příznaků autoři dosáhli lepších výsledků.

Jedna z prvních prací, která k detekování překrývajících se úseků řeči používala nějaký druh neuronových sítí je [10]. Autoři se v této práci zabývali použitím LSTM (*long short-term memory*) rekurentní neuronové sítě pro účely samotné detekce, ale i jako přídavný prvek pro získání více příznaků pro vstup do HMM systému. Na trénování používali datovou sadu AMI Corpus, který obsahuje záznamy z konferenčních hovorů. V jejich práci samotná neuronová síť dosáhla výsledků srovnatelných s HMM a při spojení sítě s HMM dosáhli autoři nejlepších výsledků.

V práci [11] autoři zkoumali využití konvolučních neuronových sítí na vlastních syntetických datech. Zabývali se především průzkumem vlivu velikosti vstupu do neuronové sítě a použití množství různých vstupních příznaků najednou. Cílem bylo dosáhnout co nejlepších výsledků při velikosti vstupního rámce pouhých 25 milisekund. Nejlepších výsledků dosáhli za použití 500 milisekundového vstupního rámce. S pomocí většího množství příznaků se jim podařilo dosáhnout uspokojivých výsledků i pro 25 milisekundové rámce. S výhodou velmi rychlého běhu neuronové sítě, kvůli malému množství vnitřních parametrů.

Práce [12] porovnává schopnost rozpoznání crosstalk-u lidmi oproti rozpoznání pomocí hlubokých neuronových sítí. Dobrovolníci se snaží na nahrávkách určit počet

mluvčích, který se může pohybovat od 1 do 4. Nahrávky jsou zároveň dále rozděleny podle délky v rozsahu půl až 5 sekund. Pro trénování neuronové sítě autoři použili datovou sadu LibriSpeech, ze kterého mixováním vytvořili crosstalk. U neuronových sítí podobně jako u lidí zkoumali vliv délky vstupních záznamů. Zde autoři potvrdili, že zvětšení kontextu má velký vliv zejména v rozsahu do 500 milisekund. Použili konvoluční architekturu sítě s velkým *dropout*-em. Ve výsledku byli lidé schopni poznat jednoho mluvčího s 94 % přesností, dva s 84 % přesností, v 1000 milisekundové nahrávce bez opakování. Navržená neuronová síť dosáhla 86 % přesnosti pro jednoho mluvčího a 78 % přesnosti pro dva mluvčí. Avšak síť dokázala s podobnými přesnostmi správně ohodnotit nahrávky s více mluvčími, kde lidé dosahovali 30 % a méně.

Další prací zabývající se použitím neuronových sítí je [13]. Pro experimenty využili datovou sadu GRID corpus. Jejich zaměření bylo prozkoumat architektury konvolučních neuronových sítí, přesnosti a výpočetní náročnost při použití různých příznaků. Používali krátkodobou Fourierovu transformaci, Mel filtr banky, MFCC a *Teafer--Kaiser energy-based pyknogram*-y jako příznaky. Z nichž nejlepších výsledků dosáhli za použití *Teafer-Kaiser energy-based pyknogram*-ů, avšak s MFCC dosáhli téměř stejných výsledků za méně než poloviční čas. Navrhli tři architektury konvolučních neuronových sítí. První se skládala z konvolučních vrstev a dvou lineárních. Druhá měla za každou konvoluční vrstvou přidanou normalizační a maximálovou redukční vrstvu. Třetí byla inspirovaná neuronovou sítí ResNet[14] a skládala se z bloků obsahující spojení umožňující přeskočit daný blok. Tohle dovoluje vstupním datům dostat se hlouběji do sítě. Nejlepších výsledků autoři dosáhli s použitím druhé architektury.

Práce [15] představuje užití kombinace konvoluční a rekurentní sítě (CRNN) na klasifikaci audio záznamu do tří skupin, a to na ticho, jeden mluvčí a crosstalk. Autoři zkoumali, zda má využití tří stavů, namísto dvou, pozitivní efekt na trénování sítě. Také vyhodnocovali použití nové architektury CRNN, tedy spojení konvoluční sítě s rekurentní. Použitím těchto dvou faktorů autoři dosáhli lepší přesnosti, a překonali tím dosavadní nejlepší výsledky na datové sadě DIHARD II.

Nejlepších výsledků v úloze detekce crosstalk-u dosahují diarizační systémy, ty určují kdo kdy mluví. V současné době to jsou *end-to-end* diarizační systémy. Namísto toho, aby byli složeny z více bloků, kde každý dělá část úlohy, jako je detekce aktivity, detekce překryvů řeči a samotné rozpoznání, má systém na vstupu pouze příznaky nebo samotný zvukový signál, ze kterého nám rovnou dává výstup. A rovnou také dokáží rozpoznat, kdy mluví více lidí, a každého označí. Příklady takových prací jsou [1] a [16].

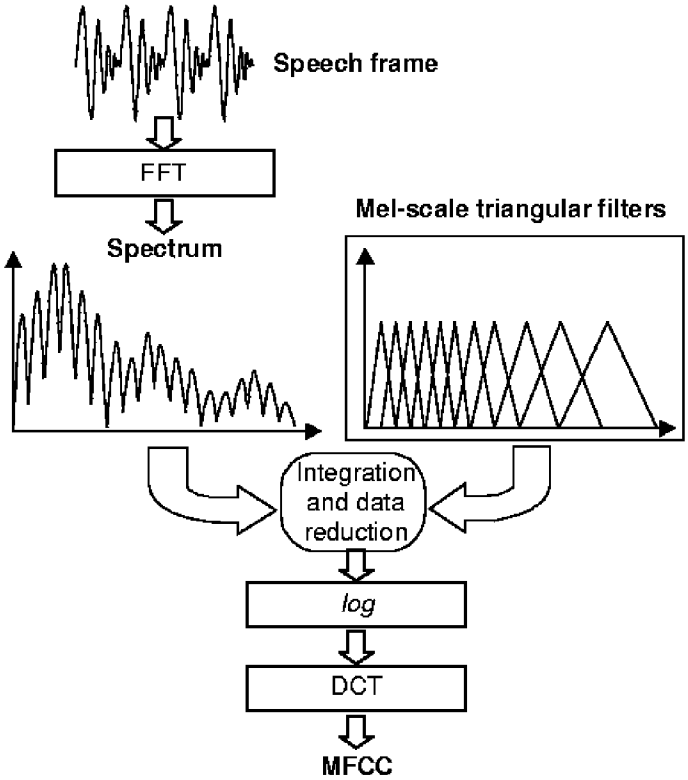
# 3 Použité technologie

V této kapitole budou popsány vybrané technologie používané pro detekci překrývajících se úseků řeči. Budou uvedeny především technologie použité v rámci této práce.

## 3.1 Příznaky audio signálu

Samotné hodnoty audio signálu nejsou příliš vypovídající o jejich obsahu. Jako relevantní informace se používají různé příznaky audio signálů. Kde mezi nejjednodušší patří energie nebo počet průchodů nulou za stanovenou dobu. Velice užitečné jsou spektrální koeficienty, které nám určují amplitudu jednotlivých frekvencí. Nejpoužívanější z nich jsou MFCC. MFCC používá nelineárně rozdělená frekvenční pásma, která dobře odpovídají lidské řeči.

MFCC jsou podobné spektrogramu tak, že rozdělují nahrávky na rámce a následně je zpracovávají Fourierovou transformací. Na obrázku 3.1 je vyobrazen celý postup. Hodnoty ze vzniklého spektra a filtrů se zlogaritmují a provede se na nich diskrétní Kosinova transformace. Takto vznikne 12 příznaků, k nimž se často přidává další, tzv. nultý, který odpovídá energii rámce. K těmto příznakům se mnohdy přidávají jejich první a druhé derivace na získání 39 koeficientů. [17]



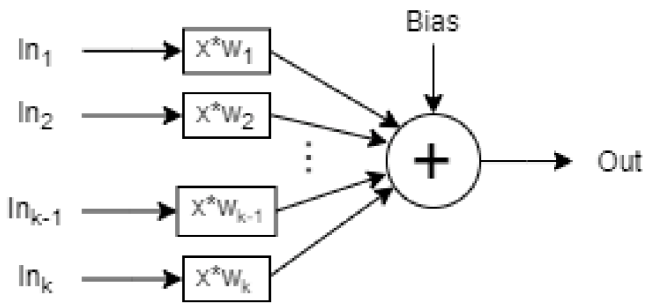

Obrázek 3.1: Diagram získání MFCC koeficientů [18]

## 3.2 Neuronové sítě

Neuronové sítě nebo také umělé neuronové sítě jsou výpočetní modely inspirované fungováním lidského mozku. Skládají se z tzv. neuronů, což jsou jednoduché celky, které dostávají signál od dalších neuronů. Tento signál zpracují a pokud se nejedná o výstupní neuron, pošlou ho dalším neuronům. Neurony se obecně řadí do vrstev, kde vrstva dostává signály z vrstvy předchozí a výstupní signál se předá do vrstvy další. Samotné fungování neuronů v takových vrstvách záleží na konkrétním typu a parametrech vrstvy. Neuronová síť se skládá ze vstupní vrstvy, na jejímž vstupu je vstup do sítě, ze skrytých vrstev, se kterými se z vnějšku přímo neinteraguje, a z vrstvy výstupní, jejíž výstup je výstup celé sítě. Za hluboké neuronové sítě se považují všechny takové, které mají více než jednu skrytou vrstvu. [19]

## 3.3 Princip fungování neuronových sítí

Neuronová síť dosáhne výsledku pro určitý vstup postupným průchodem jednotlivých vrstev. Ve většině případů se jedná pouze o dopředný průchod hodnot z jedné vrstvy do následující. Ale v některých architekturách prochází hodnoty jednotlivými vrstvami vícekrát, nebo se rozdělují do více větví, které se poté znovu spojí. Umělý neuron pouze násobí a sčítá vstupní hodnoty, k tomu navíc přičítá takzvaný *bias*. Na obrázku 3.2 je vyobrazen takovýto umělý neuron.




Obrázek 3.2: Umělý neuron

*In* s indexy jsou vstupní hodnoty, $k$ je počet vstupních hodnot, $w$ s indexy jsou váhové parametry neuronu a *out* je výstup z neuronu. Tento výstup je dále vstupem pro aktivační funkci, jejíž výstup je přiveden na vstup dalších neuronů. Váhy a *bias* jsou nastavitelné parametry. Při jejich vhodném nastavení síť dává na výstupu neuronu požadovaný výstup. Co je to požadovaný výstup je určeno pouze na neuronech ve výstupní vrstvě. Parametry ostatních vrstev jsou určeny potřebou sítě pro zadaný problém.

## 3.4 Způsoby učení neuronových sítí

Učení sítě může probíhat různými způsoby, tady však bude zmíněno pouze učení s učitelem, protože to je způsob využívaný v této práci. To probíhá tak, že při trénování se do sítě dodají data společně s očekávanými výstupy sítě. Po průchodu sítí se tyto výstupy následně porovnávají pomocí chybové funkce s referenčními

výstupy. Podle získané chyby se upraví vnitřní parametry sítě. Tento proces probíhá v takzvaných epochách. V každé epoše se použijí všechna data určená na trénování. Pro trénování neuronových sítí je potřeba mít dostatek dat. Čím je problém složitější, tím více je potřeba dat. A čím víc má být síť robustnější, tím víc je potřeba, aby data zahrnovaly všechny možné vstupní data. Celková data se obecně rozdělují na tři části. Jedna část je trénovací, na ní se síť trénuje. Další část je validační, na které se během trénování ověřuje schopnost sítě zobecňovat problém. A poslední testovací, na které se změří výsledná přesnost po natrénování.
Na testovací sadě by se nemělo ladit trénování sítě. Protože testovací data slouží na otestování sítě na obecně neznámých datech. Ale i to může být výhodné, takže se v takovém případě používá ještě datová sada vývojová, která během vývoje funguje jako testovací. Ale ladí se podle ní parametry sítě a trénování.
Při nadměrném trénování může docházet k takzvanému přetrénování. To je stav, kdy se síť naučí přesně vystihnout pouze trénovací data, avšak už nedokáže správně vyhodnotit podobná data, které síť dřív neviděla. Tomuto se dá předejít zastavením trénování ve vhodnou dobu, například podle pozorování výsledků na validační sadě. Nebo se přetrénování dá omezit pomocí metod popsaných v 3.7.

## 3.5 Aktivační funkce

Aktivační funkce (nebo též přenosové funkce) se používají k zavedení nelinearity do výstupu jednotlivých neuronových vrstev. Tím se dosáhne možnosti řešit komplexnější nelineární problémy. Aktivační funkce slouží k určení výstupního průběhu a také mohou sloužit k omezení výstupu. Ve většině případech jsou aktivační funkce použity na výstupní hodnoty jednotlivých vrstev před vstupem do následující vrstvy. V kontextu neuronových sítí je důležité, aby funkce měly definovanou derivaci, která se užívá během trénování. Vhodná volba aktivační funkce může být stejně důležitá jako výběr architektury. [20]

### 3.5.1 Sigmoid

Funkce *sigmoid* je omezená funkce a jejím příkladem je například logistická křivka. Pro neuronové sítě je vhodná, protože je omezená, tím normalizuje vstupy pro další neurony. Avšak toto nemusí být nejideálnější pro velké vstupní hodnoty, u kterých může zmenšit schopnost a rychlost učení.

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

Kde $x$ je vstupní hodnota.

### 3.5.2 Hyperbolický tangens

Hyperbolický tangens podobně jako *sigmoid* normalizuje výstupy do malého rozsahu. Oproti *sigmoid*-u však produkuje i záporné hodnoty. Proto bývá vhodnější na použití uvnitř neuronových sítí.

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.2}$$

Kde $x$ je vstupní hodnota.

### 3.5.3 Lineární jednotky

ReLU neboli *rectified linear unit*, je nejpoužívanější aktivační funkce. Je výpočetně nenáročná a dovoluje dosažení rychlé konvergence. Pokud se vstupy blíží nule nebo jsou záporné, tak se neuronová síť není schopná učit. Proto se v některých případech používají verze s nenulovým výstupem v záporné části jako je ELU. ELU je exponenciální lineární jednotka, má zápornou část definovanou exponenciálou.

$$ReLU(x) = max(0, x) \tag{3.3}$$

$$ELU(x) = \begin{cases} x, & \text{pokud } x \geq 0. \\ \alpha(e^x - 1), & \text{jinak.} \end{cases} \tag{3.4}$$

Kde $x$ je vstupní hodnota a $\alpha$ v rovnici 3.4 je nastavitelný parametr.

## 3.6 Princip učení neuronových sítí

Trénování probíhá postupnou úpravou parametrů sítě podle optimalizačního algoritmu. Existuje celá řada takových algoritmů, všechny však fungují na principu výpočtu gradientu kriteriální, nebo také chybové, funkce pomocí zpětného průchodu sítí. Pomocí tohoto gradientu se snaží optimalizační algoritmus minimalizovat chybu.
V této práci je využita binární křížová entropie jako kriteriální funkce na úlohu klasifikace. A jako optimalizační algoritmus je použit Adam. [21]

### 3.6.1 Chybová funkce binární křížová entropie

Pomocí chybové funkce je počítána chyba výstupu neuronové sítě během trénování. Tato chyba se poté minimalizuje a tím učí síť. Volba chybové funkce je velmi závislá na daném problému. Binární křížová entropie je logaritmická chybová funkce. Používá se při trénování klasifikátoru dvou tříd. V knihovně PyTorch [22] je implementována následovně:

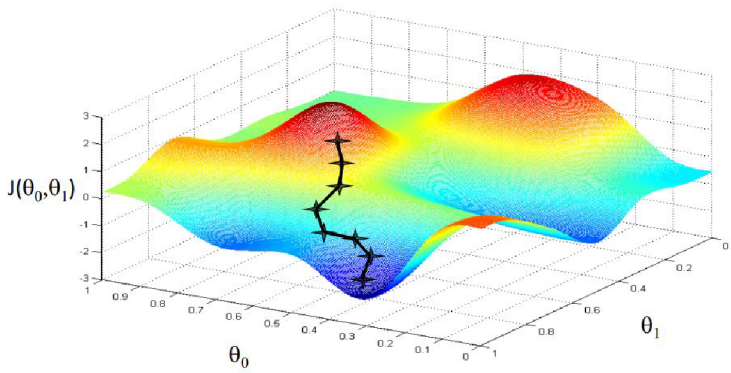
$$chyba(x) = -[y \cdot log(x) + (1 - y) \cdot log(1 - x)] \tag{3.5}$$

Kde $x$ je výstup modelu a $y$ je skutečná třída vstupních dat.

### 3.6.2 Optimalizační algoritmus Adam

Adam [21] neboli *adaptive moment estimation* je optimalizační algoritmus, který umožňuje neuronové síti optimalizovat parametry a tím se učit. Je to algoritmus

založený na metodě *stochastic gradient descent*, která podle záporné hodnoty gradientu postupnými kroky minimalizuje chybu. Jelikož je přesný výpočet gradientu výpočetně náročný, využívá tento algoritmus pouze jeho odhad, čímž se výrazně zrychlí. Na obrázku 3.3 je tento postup vyobrazen pro dva parametry.




Obrázek 3.3: Ukázka postupu optimalizačního algoritmu [23]

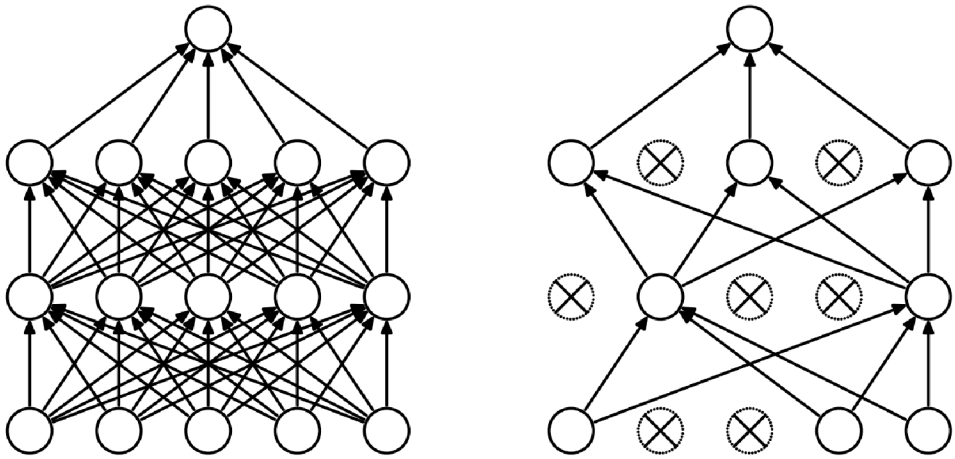
Základem jeho fungování je suma čtverců gradientů počítána pomocí exponenciálního klouzavého průměru, a navíc k tomu ještě používá exponenciální klouzavý průměr posledních gradientů podobně jako u *Momentum*[24]. Ten podle velikosti gradientu zvětšuje, či zmenšuje velikost kroku, a tím výrazně urychluje učení. Jak název napovídá, Adam se automaticky adaptuje, a proto je tento algoritmus vhodný na široké spektrum problémů. Zároveň není ve většině případů třeba ladit jeho parametry.

## 3.7 Pokročilé metody trénování neuronových sítí

### 3.7.1 Dropout

Pro zvýšení robustnosti sítě se v mnoha případech využívá takzvaný *dropout*. To je metoda, která napodobuje trénování většího množství sítí naráz. Náhodně vynechává výstupy na vybraných vrstvách z výpočtů, tím efektivně neexistují.

Na obrázku 3.4 je zobrazena vlevo obyčejná síť a vpravo síť s *dropout*-em. Tímto vypouštěním neuronů je síť nucena naučit se několik na sobě nezávislých způsobů jak přijít ke správnému výsledku. To zároveň pomáhá předcházet přeučení sítě. *Dropout* je aktivní pouze při trénování, čímž se výsledná síť stává robustnější a potenciálně přesnější. [25]



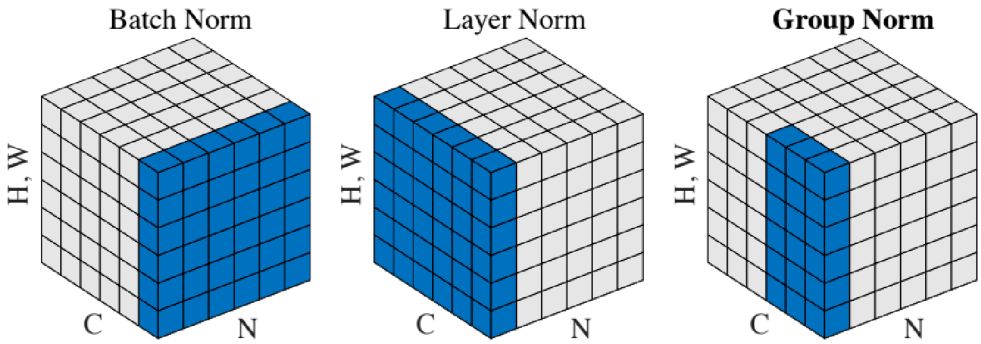
Obrázek 3.4: Obyčejná síť a síť s dropout [25]

### 3.7.2 Normalizace

Normalizační vrstvy nám slouží k normalizaci vah vnitřních parametrů, a tím zamezují, aby se dostaly nad určitou hranici. Naopak zesilují váhy, které se blíží nule. Během trénování neuronových sítí se mění vnitřních parametry, a tím se mění rozdělení výstupů do následující vrstvy. To znamená, že každá vnitřní vrstva se musí pokaždé optimalizovat vzhledem k novým rozdělením. Tento jev se nazývá *internal covariate shift* a způsobuje zpomalení trénování, kvůli nutnosti použití menšího kroku pro úpravu parametrů. Také je kvůli tomu síť citlivější na počáteční parametry. Normalizační vrstvy redukují *internal covariate shift*, tím zamezují tomuto problému a umožňují využít větší krok při učení. [26]

Samotná normalizace nám zároveň urychluje učení, protože stabilizuje velikost gradientu pro všechny parametry, a tím umožňuje rychlejší konvergenci ke globálnímu minimu chybové funkce. Také nepatrným způsobem zabraňuje přetrénování.

Normalizační vrstvy fungují na principu výpočtu střední hodnoty a směrodatné odchylky. Pomocí nich následně změní měřítko výstupní vrstvy, aby byl průměr nula a směrodatná odchylka jedna. Existuje několik variant normalizačních vrstev, liší se v tom jakou část dat uvažují pro normalizaci. Na obrázku 3.5 jsou vyobrazeny různé druhy normalizačních vrstev, $N$ představuje dávku, $C$ kanál, $H$, $W$ rozměry obrázku nebo obecněji počet příznaků.




Obrázek 3.5: Druhy normalizace [27]

*Batch* neboli dávková normalizace vznikla jako první, aby vyřešila problém *internal covariate shift*. Ale její výpočet je závislý na velikosti dávky,a proto není vhodná při použití menší dávky. Také není vhodná pro rekurentní neuronové sítě.[28] *Layer* neboli vrstvová normalizace je právě vhodná pro rekurentní sítě, kde dosahuje lepších výsledků. [29] Skupinová normalizace je velice podobná vrstvové, ale oproti ní je flexibilnější ve vstupním rozsahu a v některých případech dosahuje lepších výsledků. [27]

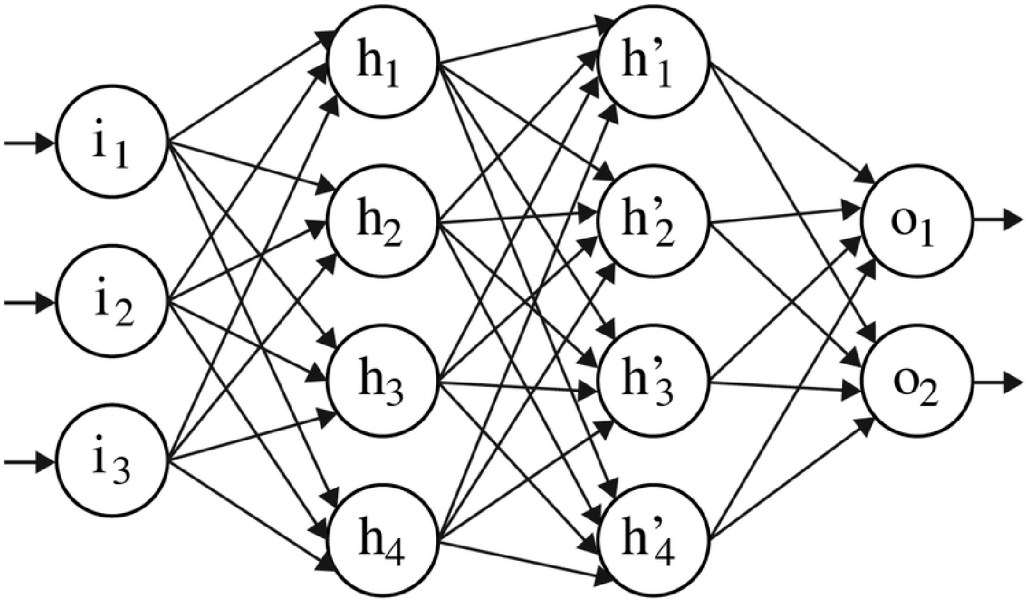
## 3.8 Vybrané architektury neuronových sítí

### 3.8.1 Lineární neuronové sítě

Lineární neuronové sítě jsou nejjednodušším typem neuronových sítí. Skládají se pouze z lineárních vrstev obsahující neurony, které jsou definované váhami vstupních signálů, *bias*-em (pevným posunem) a aktivační funkcí. Síť jako taková je schopná se s omezenou přesností naučit většině problémů. Samotný výpočet výstupu pro jednotlivé neurony vypadá takto:

$$y = \phi(xW^T + b) \tag{3.6}$$

Kde $y$ je výstup, $\phi$ je aktivační funkce, $x$ jsou vstupy, $W$ jsou váhy a $b$ je *bias*. Na




Obrázek 3.6: Dopředná plně spojitá lineární sít [30]

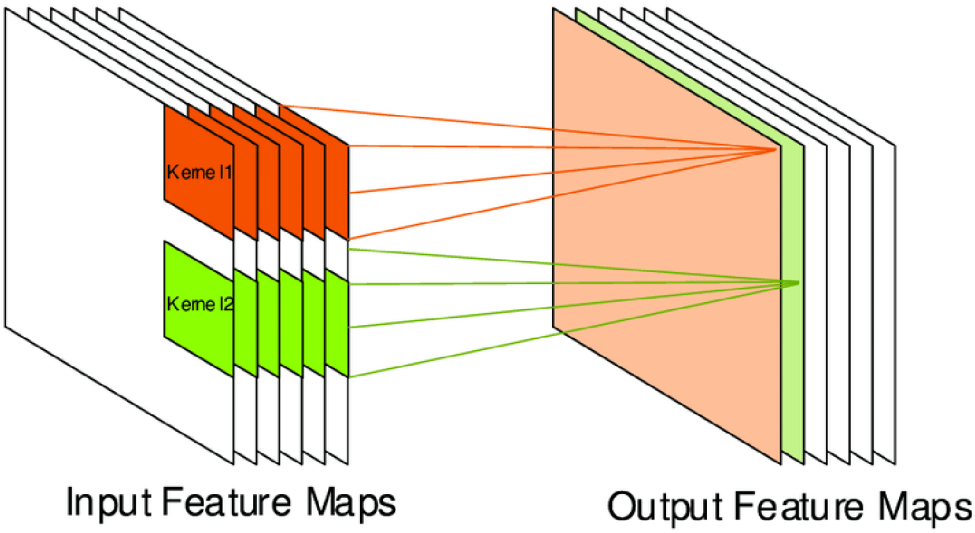
obrázku 3.6 je vyobrazena jednoduchá plně dopředná neuronová síť. $i$ značí vstupní vrstvu, $h$ značí skryté vrstvy a $o$ značí výstupní vrstvu. Plně dopředná znamená, že každý neuron má jako vstup výstup z každého neuronu předchozí vrstvy. V takovém typu sítě jsou nejdůležitějšími parametry hloubka a šířka. Šířka udává počet neuronů ve vrstvě a zvyšuje kapacitu učení sítě. Hloubka představuje počet skrytých vrstev.

Hloubka pomáhá síti učit se abstraktnější pojetí dat a zvyšuje její kapacitu, která je závislá primárně na počtu neuronů. To znamená, že pro určitou úlohu je třeba mít určitou kapacitu pro rozumné natrénování. Nadměrné zvyšování hloubky a šířky nemusí vést ke zlepšení výkonu sítě.

### 3.8.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě se zpravidla skládají z konvolučních, redukčních a lineárních vrstev. V zásadě fungují tak, že konvoluční vrstvy se naučí extrahovat různé rysy trénovacích dat a lineární vrstvy se naučí vybírat třídu na jejich základě.
Konvoluční vrstvy oproti lineárním mají velkou výhodu v tom, že z principu jejich fungování berou ohled na prostorové rozložení vstupních dat. To je výhodné například pro zpracování obrazu, kde konvoluční vrstva podle jádra bere pro každý bod v potaz nějaké okolí. Naproti tomu lineární vrstva považuje všechny vstupy za nezávislé. Stejně tak u audio záznamu, pokud se jako vstup používají spektrální příznaky, tak nám téměř vzniká obraz, na který se aplikuje konvoluční jádro. Další výhodou konvoluční oproti lineární vrstvě je poměrně malý počet vnitřních parametrů, v lineární vrstvě má každý neuron parametr pro každý vstup. Počet parametrů u konvoluční vrstvy je závislý pouze na velikosti konvolučního jádra a na počtu vstupních a výstupních kanálů. Tedy nezáleží na vstupních datech, což umožňuje použít konvoluční vrstvy na rozměrnější data.
Konvoluční vrstvy jsou kromě velikosti jádra, anglicky *kernel*, určeny ještě parametry *stride*, obal, dilatace, počet vstupních a výstupních kanálů. Parametr *stride* nám udává velikost posunu jádra na vstupních datech. Obalem se určuje jestli a jakým způsobem mají být vstupní data obalena. Tohoto se používá zejména pro zachování rozměrů. Parametr dilatace určuje dilataci vstupních hodnot, která se užívá při takzvané dekonvoluci, kdy výstupní data mají větší rozměr než vstupní. Počet kanálů funguje například jako u RGB obrázků, kde každý ze tří kanálů nese nezávislou informaci. Stejně tak se používají tady. Jsou tedy nezávislé vrstvy, které mají vlastní parametry konvolučního jádra, které používají na každý vstupní kanál. Obrázek 3.7

zobrazuje tento vztah ve dvou rozměrech.



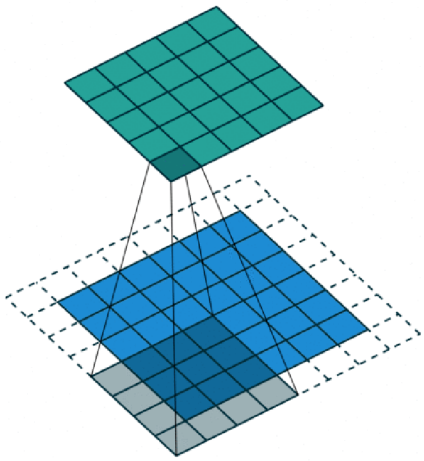

Obrázek 3.7: Vztah vstupních a výstupních kanálů konvoluční vrstvy [31]

Hodnoty konvolučního jádra společně s *bias*-em, který se staticky přičítá stejně jako u obyčejných neuronů, jsou učitelné parametry konvoluční vrstvy. Zbytek parametrů určuje návrhář neuronové sítě. Rovnice 3.7 popisuje výpočet jednoho kanálu konvoluční vrstvy.

$$y = b + \sum_{k=0}^{C-1} kernel_k \star x_k \tag{3.7}$$

Kde $y$ označuje výstup, $b$ je *bias*, $C$ označuje počet kanálů, *kernel* je jádro k-tého kanálu, operace $\star$ je konvoluce a $x_k$ je vstup k-tého kanálu. Na obrázku 3.8 je znázorněno fungování konvoluční vrstvy s jádrem o velikosti 4x4, *stride* 0 a obalem 1. Kde modrá označuje vstupní data a tyrkysová označuje výstupní data.

Redukční vrstvy slouží pro redukci rozměrů dat uvnitř sítě, fungují podobně jako konvoluční. Jejich fungování je definováno jejich druhem a velikostí jádra. Velikost jádra funguje v zásadě stejně jako velikost konvolučního jádra u konvolučních vrstev. Druh určuje jakou operaci bude vrstva provádět. V praxi se používají v zásadě pouze dva druhy. Maximálové, které vybírají z vstupu maximální hodnotu a průměrové, které počítají průměrnou hodnotu vstupu. Z toho se nejvíce používá maximálové, protože zároveň potlačují šum v datech.[33]

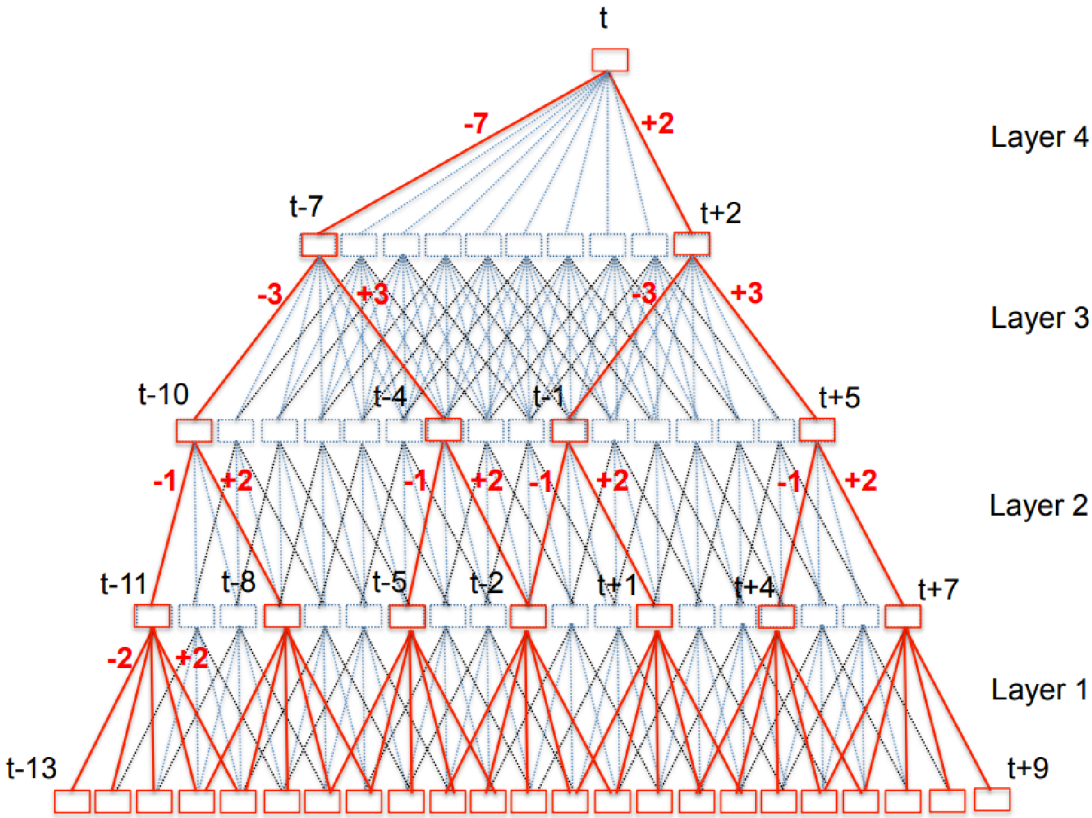


Obrázek 3.8: Ukázka fungování konvolučního jádra [32]

### 3.8.3 Time delay neuronové sítě

Time delay neuronové sítě vznikly jako alternativa rekurentních sítí, jejichž trénování oproti dopředným sítím trvá déle. Struktura neuronových sítí time delay jsou podobné lineárním sítím v tom, že jednotlivé vrstvy sčítají vstupy z předchozích vrstev. Nicméně zde základní jednotky obsahují nastavitelnou prodlevu. Tato prodleva určuje vstupní kontext. Na obrázku 3.9 je znázorněn tento koncept. Pomocí toho dokážou stejně jako rekurentní sítě modelovat dlouhodobé časové závislosti. [34] Na obrázku 3.9 je vidět, že každá vrstva má svoji danou prodlevu v obou směrech. Tento kontext se postupně sčítá, takže konečný výstup má v tomto případě 23 vstupů. Také lze zvolit, jestli bude vrstva brát v potaz celý rozsah kontextu, na obrázku znázorněno modře, nebo jen vybrané, či krajní. Vrstva číslo 1 má kontext plný, zbytek vrstev využívá pouze krajní vstupy.

Tuto funkcionalitu je možné realizovat pomocí dopředné lineární vrstvy s pomocí vhodné manipulace vstupních dat. Za předpokladu použití pouze symetrického a rovnoměrně rozloženého kontextu, je možné využít jednorozměrné konvoluce a parametru *stride*. Toto nám umožní mnohem rychlejší trénování, za cenu tohoto omezení.




Obrázek 3.9: Ukázka architektury time delay [34]

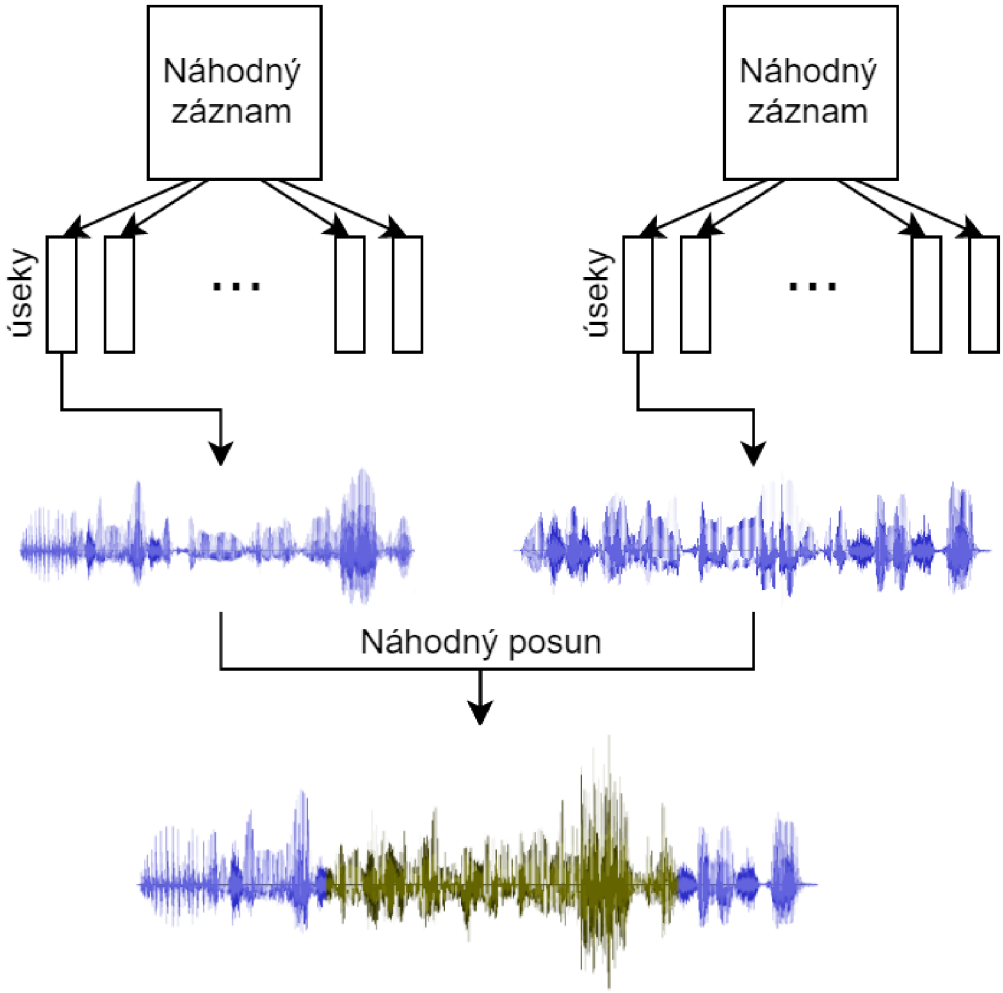
# 4 Vytvořené programy

Všechny programy byli vytvářeny v jazyce Python. Neuronové sítě byly vytvářeny a trénovány pomocí knihovny PyTorch [35]. Verze Python-u byla 3.9, knihovny PyTorch 1.10.1 se sadou nástrojů CUDA verze 11.3. Dále byl použit program HCopy z HTK [36] verze 3.4.1, MATLAB verze 2020b a Audacity verze 2.4.2.

## 4.1 GenData

Tento program slouží ke generování umělé části trénovacích dat. Nejprve vytvoří seznam všech souborů určených pro generování dat. Postupně prochází tento seznam ve dvou vnořených cyklech, v náhodném pořadí vybere dvojici nahrávek a k tomu příslušné soubory s anotací. Pomocí souborů s anotací nalezne a oddělí úseky řeči. Pro zmenšení pravděpodobnosti výskytu neoznačeného crosstalk-u je začátek a konec každého úseku zkrácen o půl sekundy. Vybíráme úseky dlouhé 1 až 10 sekund, pokud je úsek delší, je dělen tak dlouho než tuto podmínku splní. Jelikož by použitím všech možných kombinací těchto úseků mohlo vzniknout ohromné množství dat, tak je každý 1 z 10 000 úseků náhodně ponechán. Potom se dva různé úseky překryjí s náhodným posunem a uloží do nového audio souboru. V programu je zadaná minimální délka překrytí 40 milisekund, stejně jako v ostatních datech. Horní limit není daný, ale největší možný překryv vznikne plným překrytím dvou deseti sekundových nahrávek. K novému audio souboru přísluší textový soubor ve kterém je uložena informace o začátku a konci překrývající se části. Program se zastaví po vygenerování požadovaného množství dat nebo po vyčerpání vstupních dat.

Na obrázku 4.1 je znázorněno generování dat, a je zde také žlutě zvýrazněna část, kde dochází k překryvu řeči.




Obrázek 4.1: Diagram vytváření umělých dat

## 4.2 TrainNet

Tento program slouží k samotnému trénování neuronových sítí a provádění experimentů. Knihovna PyTorch poskytuje implementaci všech potřebných funkcí pro trénování neuronových sítí.

Před spuštěním jakéhokoliv trénování se zafixuje generování náhodných čísel použitím jednorázového a náhodně zvoleného *seed*-u. Tohle je provedeno pro eliminaci

vlivu náhody na výsledky jednotlivých experimentů. Jsou tím zafixovány zejména inicializační hodnoty jednotlivých vrstev. Knihovna PyTorch využívá pro některé operace nedeterministické algoritmy, které jsou rychlejší. Naštěstí velká část těchto algoritmů má implementovanou i deterministickou verzi. Pomocí přepínače *torch.use_deterministic_algorithms(True)* program nastaví použití deterministických verzí. Nicméně lineární vrstva, která je součástí každé sítě využívá interpolace, která nemá deterministickou implementaci, takže stále využívá nedeterministickou verzi. Dále při použití knihovny CUDA ve verzi $\geq 10.2$ je potřeba nastavit použití deterministických algoritmů i v ní. To je provedeno pomocí proměnného prostředí *CUBLAS_WORKSPACE_CONFIG* na hodnotu „:4096:8“.

Poté se inicializuje síť, chybová funkce a optimalizátor. Načtou se trénovací a validační data do paměti. Z trénovacích umělých dat se pro experimenty načítá pouze část záznamů. Pro konečné vyhodnocení se načítají všechny data umělé trénovací sady, ale kvůli jejich velikosti se načítají po částech.

Pak probíhá samotné trénovaní sítě. Trénování může probíhat ve dvou fázích. Kdy nejprve se síť trénuje na umělých datech a na datové sadě s nádechy. V druhé fázi se trénuje na reálné trénovací sadě. Případně se síť trénuje v jedné fázi, kdy se trénuje na všech trénovacích datech zároveň.

Po proběhnutí trénovací epochy se provede validace, jejíž výsledky jsou dodatečně upraveny podle kapitoly 6.1.1. Výsledky z trénování a validace jsou zaznamenány do souboru protokolu. Udržuje se kopie dosavadní nejlepší sítě. Na ní je následně provedeno vyhodnocení na testovací sadě s a bez crossnoise. Výsledky se uloží společně s výslednou natrénovanou sítí.

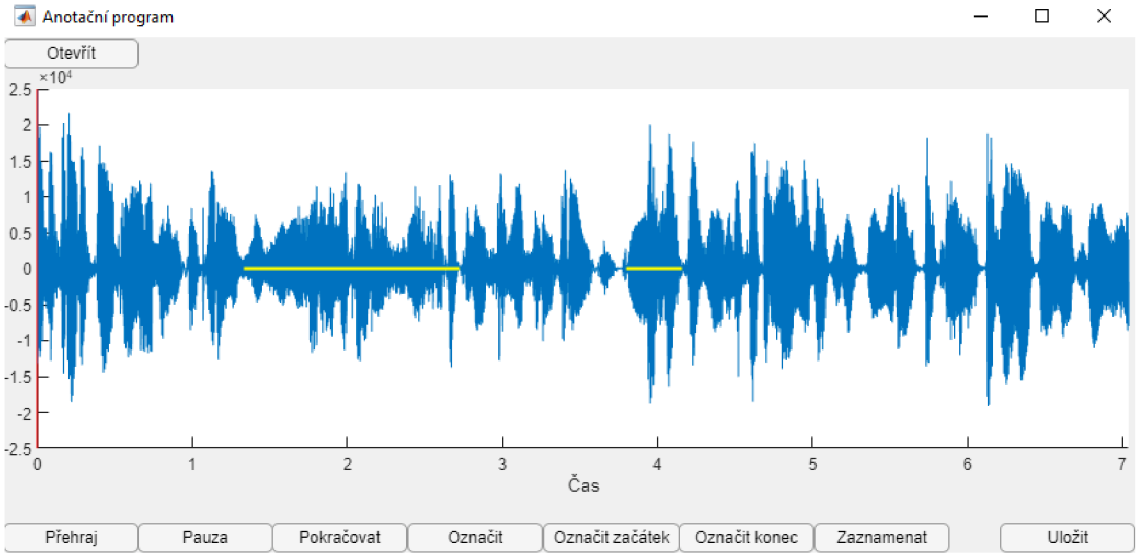
## 4.3 TestNet

Program TestNet slouží k otestování již natrénované neuronové sítě. Je třeba nastavit stejné parametry sítě, protože se ukládají pouze parametry sítě, nikoliv informace o typu a nastavení jednotlivých vrstev. Načtou se testovací data, poté se načte natrénovaný model a provede se vyhodnocení. Je také možné z výstupu sítě vytvořit

anotace k jednotlivým souborům. Pro soubory s již existující anotací je možné ji zahrnout jako referenci.

## 4.4 Program na anotaci dat

Anotační program vznikl za účelem zjednodušení ruční anotace dat. Vznikl v návrhovém prostředí MATLAB App Designer. Na obrázku 4.2 je vyobrazeno uživatelské rozhraní.




Obrázek 4.2: Vzhled vniklého anotačního programu

Po otevření audio souboru se zobrazí audio signál. Po stisku tlačítka „Přehraj“ se začne audio přehrávat od začátku. Červený ukazatel značí aktuálně přehrávaný signál. Tlačítko „Pauza“ a „Pokračovat“ pozastaví a obnoví přehrávání audia. Po stisku tlačítka „Označ“ uživatel vybere dva body signálu, které se následně označí jako crosstalk a v audio signálu se označí žlutě. Tlačítka „Označ začátek“, „Označ konec“ a „Zaznamenat“ slouží také k označení crosstalk-u, ale je možné začátek a konec označeného úseku upravovat, než dojde ke stisku tlačítka „Zaznamenat“. Po stisku tlačítka „Uložit“ se dle označených úseků vytvoří jednotlivé audio soubory obsahující crosstalk. Kromě crosstalk-u se na obou stranách nahrávky nechá půl sekundy bez crosstalk-u. Společně s audio souborem vznikne referenční soubor

s informací o poloze crosstalk-u.

Program je použitelný pro svůj účel, ale chybí mu několik důležitých vlastností. Jednou z nich je možnost přehrávat jen daný úsek záznamu. Dále se ukázalo, že je třeba anotovat dlouhé nahrávky, na které nebyl program uzpůsobený. Nakonec se ukázalo, že vývoj dodatečných vlastností by zabral příliš mnoho času, a proto byl po dohodě s vedoucím vývoj zastaven, a program ponechán v tomto základním stavu.

Ručně anotovaná data byla vytvářena s pomocí programu Audacity, protože nabízí všechny potřebné funkce k rychlému nalezení počátku a konce crosstalk-u, za cenu složitější extrakce této informace.

# 5 Použité datové sady

Použitá datová sada se skládá ze čtyř částí, trénovací, validační, vývojová a testovací. Velká část trénovací sady byla vytvářena uměle, stejně jako celá validační sada a část testovací. Trénovací sada dále obsahuje reálnou část a její poslední část se skládá z nahrávek nádechů bez jakýchkoliv překryvů.

Značná část této práce se zaměřuje na vytváření použitých dat. Umělá data byla vytvořena za pomocí vytvořeného skriptu popsaného v kapitole 4.1 z již anotovaných záznamů. Vývojová sada byla vytvořena zcela nově ruční anotací pro účely této práce. Tyto data v budoucnu poslouží jako rozšíření již existující datové sady, jež je zde použita na reálnou trénovací a testovací sadu.

Za normálních okolností se využívá validační sada k ladění parametrů. Jenže je trénovací a validační datová sada složená z umělých dat a testovací zcela není, byla pro ladění parametrů použita vývojová datová sada, která lépe odpovídá skutečné testovací sadě. Validační sada nám zde ukazuje především schopnosti sítě naučit se zobecnit problém. Proto je stále cenný ukazatel výkonu sítě. Tady je použita pro vyhodnocení úspěšnosti na umělých datech. Výsledky na validačních datech jsou také použity během trénování pro volbu sítě pro trénování na skutečných datech. Ve validačních a v trénovacích datech není rozlišováno mezi crosstalk-em a crossnoise-m.

Ve vývojové a skutečné testovací sadě byl crosstalk a crossnoise označován zvlášť, díky tomu bylo možné následně porovnávat výsledky v případě, kde je uvažován pouze crosstalk, nebo při zahrnutí crossnoise. Což pro tuto práci nemá velký význam, ale pro budoucí práce s těmito daty může hrát větší roli. Například pokud bude vznikat detektor, který má rozlišovat mezi těmito druhy překrývající

se řeči. Protože je pro tuto práci důležitý jakýkoliv druh překryvu řeči a jelikož trénovací data tyto druhy nerozlišují, nedává smysl mezi těmito druhy rozlišovat při vyhodnocování. Takže neuronové sítě rozpoznávají pouze to, zda se jedná nebo nejedná o překrývající se řeč.

Tabulka 5.1 obsahuje přehled všech použitých dat. U všech sad byla zvolena minimální doba crosstalk-u a crossnoise 40 milisekund. Kratší úseky nebyly anotovány, a případné kratší mezery v překrývajících se úsecích jsou odstraněny.

| | celkový čas | crosstalk | crossnoise |
|---|---|---|---|
| Umělá trénovací sada | 90 h | 24 h 19 m 40 s | - |
| Trénovací sada s nádechy | 1 h 27 m 2 s | - | - |
| Reálná trénovací sada | 5h 2 m 5 s | 12 m 29 s | 1 m 26 s |
| Validační sada | 1 h | 15 m 54 s | - |
| Vývojová sada | 5 h 1 m 37 s | 5 m | 1 m 56 s |
| Umělá testovací sada | 10 h | 2 h 41 m 50 s | - |
| Reální testovací sada | 5 h 10 s | 6 m 51 s | 1 m 34 s |

Tabulka 5.1: Přehled použitých dat

## 5.1 Příprava audio souborů

Veškeré použité audio záznamy byly převedeny do jednotného formátu a to jednokanálového *wav* souboru s vzorkovací frekvencí 16 kHz, na kterém byli vypočítány MFCC příznaky. MFCC byly vypočteny včetně nultého koeficientu, první a druhé derivace. Takže na každý rámec vychází 39 příznaků. K výpočtu byl použit program HCopy, který je součástí sady nástrojů HTK[36]. Rámce byly vytvářené každých 10 milisekund a délka rámce je určena Hammingovým oknem dlouhým 25 milisekund. K výpočtu 12 kepstrálních koeficientů bylo použito 26 Mel filtrů. A na signál byla

předtím použita preemfáze prvního řádu s koeficientem 0,97. Normalizace energie nebyla použita.

## 5.2 Trénovací a validační

Trénovací a validační datová sada byla vytvářena uměle za pomocí již anotovaných audio záznamů z rádií, televizního vysílání a on-line rozhovorů. K záznamům byly dostupné anotace ve formátu založeného na XML, kde jsou zaznamenány časy od kdy do kdy mluvil jaký mluvčí, a co řekl. Pokud se jedná o normální řeč, je zde fonetický a textový přepis. Soubory dále obsahují značky pro různé neřečové události jako je kašel, smích, nádech, hluk, hudba a podobné. Dohromady bylo k dispozici 98 hodin 9 minut a 7 sekund záznamu. Z audio dat byly používány pouze části označené jako řeč, kašlání a nádech. V použitých zdrojových datech je možné, že se vyskytuje crosstalk nebo crossnoise, který tam však není nijak označen. Proto byly ze záznamu vyloučeny krátké části v blízkosti jiného mluvčího. Bylo vyloučeno 500 milisekund před a za úsekem, kde se mluví.

Vytváření probíhalo pomocí programu na generování dat popsaného v kapitole 4.1. Z různých dvou nahrávek se částečně překryly úseky, které byly jednu až deset sekund dlouhé. U těchto dat je přesně daný interval překryvu řeči. Minimální doba překryvu byla zvolena 40 milisekund, stejně jako u testovacích dat. Tímto způsobem bylo celkem vygenerováno 101 hodin záznamu rozdělených v 58 991 souborech. Z toho prvních 90 hodin je umělá trénovací datová sada, dalších 10 hodin je umělá testovací sada a zbývající hodina je validační datová sada. Takto vygenerovaná data obsahují přibližně z 37 % překrývající se úseky řeči. Na obrázku 4.1 je znázorněno vytváření umělých dat.

Dále byla k trénovacím datům přidána zhruba hodina a půl dat obsahující nádechy, protože se nádechy ukázaly jako oblast s velkou chybovostí. Tato data obsahují pouze nádech jednoho mluvčího. Jejich přidáním by se měla síť lépe naučit rozlišovat nádechy a skutečný crossnoise. Tato data byla získána z části dat použitých pro vytváření umělého crossnois-u. Jelikož jsou použitá data buď označena stejně jako

již existující data, nebo v případě, že se liší, je jejich označení také odlišné. Proto není problém použít stejná data vícekrát v rámci trénování.
Jako poslední část trénovacích dat je 12 záznamů dlouhých zhruba 5 hodin obsahující ručně anotovaný crosstalk a crossnoise, který byl vytvořen týmem ze *SpeechLab*-u ITE v rámci dřívějších prací. Tato část dat slouží především k dodatečnému trénování sítě, protože je důležité trénovat na datech co nejbližších skutečnému použití a umělá data pouze simulují crosstalk a crossnoise.

## 5.3 Vývojová

Při experimentálním vyhodnocení funkčnosti různých architektur neuronových sítí je nutné vyzkoušet velké množství různých kombinací parametrů. Pokud bychom ladili parametry na konkrétní testovací data, je velice pravděpodobné, že by na jiných datech stejného charakteru síť dosáhla výrazně horších výsledků. Proto by tyto výsledky nebyly vypovídající o výsledcích v reálném použití. Z tohoto důvodu se normálně používá validační sada, ta je však v tomto případě vytvářena uměle, proto neodpovídá testovacím datům. Z tohoto důvodu se využívá vývojová sada, která se používá místo testovací během vývoje.
Vývojová sada vznikla ruční anotací audio záznamů z internetové televize DVTV. Jelikož záznamy obsahují úvodní znělku byly anotovány dohromady čtyři třídy, crosstalk, crossnoise, *non-speech* a jeden mluvčí/ticho. *Non-speech* označuje úseky hudby a podobné zvuky, ty se při vyhodnocování neberou v potaz.
Datová sada se skládá z 13 souborů o celkové délce 5 hodin 1 minuta 37 sekund, které obsahují 5 minut crosstalk-u a necelé 2 minuty crossnoise.

## 5.4 Testovací

Testovací datová sada nám slouží k vyhodnocení přesnosti finálních neuronových sítí. Tato sada byla použita pouze pro otestování výsledných architektur popsaných

v kapitole 6.6. Testovací datová sada byla stejně jako poslední část trénovací sady vytvořena týmem *SpeechLab* ITE v rámci dřívějších prací. Vznikla ruční anotací záznamů pořadů z internetu, televize a rádií. Dohromady má tato datová sada celkovou délku 5 hodin a 10 sekund. Obsahuje 6 minut a 51 sekund crosstalk-u a 1 minutu a 54 sekund crossnoise.

# 6 Experimentální ladění neuronových sítí

## 6.1 Způsob vyhodnocení

Ve výsledcích experimentů bude dána přednost výsledkům bez zpracování popsaným v kapitole 6.1.1 před zpracovanými. Protože tyto výsledky lépe odpovídají schopnostem samotné sítě. Také bude dána přednost výsledkům na vývojových datech před validačními. Protože ty lépe odpovídají očekávaným výsledků na skutečných datech.

### 6.1.1 Zpracování výstupu sítě

Výsledky sítě byly zpracovávány dvěma způsoby. Nejdříve byl na výstup aplikován klouzavý průměr o velikosti 51 vzorků, což odpovídá okénku 525 milisekund. Výstupy klouzavého průměru nad 0,5 byly považovány za crosstalk, zbytek za jednoho mluvčího nebo ticho. Následně byly odstraněny mezery mezi těmito dvěma třídami kratší než 100 milisekund.

### 6.1.2 Metriky

K vyhodnocování výsledků jsou používané standardní metriky jako preciznost, výtěžnost a *f-value*. Preciznost určuje poměr pravdivě pozitivních (TP) a označených pozitivních, to je pravdivě pozitivních s falešně pozitivními (FP).

$$preciznost = \frac{TP}{TP + FP} \tag{6.1}$$

Výtěžnost je poměr pravdivě pozitivních a skutečně pozitivních, to je pravdivě pozitivních s falešně negativními (FN).

$$\text{výtěžnost} = \frac{TP}{TP + FN} \tag{6.2}$$

*F-value*, někdy nazývané taky *f-score*, je kombinací obou. Preciznost a výtěžnost jdou ze svého principu proti sobě, takže pro dosažení vysoké hodnoty *f-value* je potřeba vysoká hodnota obou. Proto *f-value* funguje jako dobrý ukazatel celkové přesnosti výsledků a bude použit jako hlavní metrika při vyhodnocování.

$$F - value = \frac{2 \cdot preciznost \cdot \text{výtěžnost}}{preciznost + \text{výtěžnost}} \tag{6.3}$$

## 6.2 Navržený přístup řešení

K řešení úlohy překrývajících se úseků řeči budou použity neuronové sítě a jako vstupní příznaky budou použity MFCC. Experimenty se zaměří na výběr architektur a parametrů neuronových sítí jako klasifikátorů pro detekci překrývajících se úseků řeči.

Je řešen problém klasifikace neznámých dat. Tyto data rozdělujeme pouze na dvě třídy, což lze zjednodušit na jeden výstup v rozsahu 0 až 1, podle toho s jakou jistotou si síť myslí, že se v daném úseku nachází překrývající se řeč anebo jediný mluvčí/ticho. Pro omezení výstupu do tohoto rozsahu používáme jako finální aktivační funkci *signoid*.

Trénování probíhalo v 20 epochách, kde výsledná síť byla vybrána podle největší *f-value* na validační sadě dat. Bude použit optimalizátor Adam s nastavením $\beta_1 = 0,9$, $\beta_2 = 0,999$, $\epsilon = 1e-08$ a velikost kroku $\eta$, neboli *learn rate*, nastavený na 0,001. Jako chybová funkce bude použita binární křížová entropie. Případné změny trénování, či vyhodnocení budou zmíněny u příslušných experimentů.

## 6.3 Lineární neuronová síť

Základní klasifikátor bude lineární neuronová sít se čtyrmi skrytými vrstvami o šířce 128 neuronů, se vstupním kontextem 100 milisekund na obě strany. Jako aktivační funkce se kromě poslední používala ReLU. Tato síť dosáhla 82,49 % *f-value* na validačních datech a 11,01 % *f-value* na vývojových datech.

### 6.3.1 Experiment s šířkou kontextu

Jako první se experimentovalo s šířkou vstupního kontextu. Tedy kolik MFCC rámců bude použito jako vstup neuronové sítě. V tabulce 6.1 je výsledek tohoto experimentu.

| Kontext | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 50 ms | 81,16% | 84,41% | 12,26% | 12,97% |
| 100 ms | 82,51% | 84,92% | 13,30% | 13,90% |
| 150 ms | 82,94% | **85,05**% | 12,94% | 13,8 % |
| 200 ms | 82,31% | 83,50% | 12,05% | 12,71% |
| 250 ms | 82,27% | 83,21% | 11,38% | 12,05% |
| 300 ms | **83,07**% | 84,22% | **13,72**% | **14,32**% |
| 350 ms | 82,60% | 83,68% | 12,16% | 12,82% |
| 400 ms | 82,98% | 83,92% | 12,02% | 12,43% |
| 450 ms | 81,67% | 82,46% | 11,46% | 11,76% |
| 500 ms | 82,20% | 82,92% | 11,90% | 12,46% |

Tabulka 6.1: *F-value* výsledků experimentu s vstupním kontextem

Na výsledcích je možné vidět, že s malým a příliš velkým kontextem je těžší rozpoznat překryv. U malého kontextu to může být způsobeno nedostatkem dat, u velkého může být těžší se z dat naučit poznat překryv. Nejlepších výsledků bylo dosaženo

s kontextem 300 milisekund na obě strany, proto bude dále používán pro všechny ostatní experimenty. Takže vstup do sítě je 61 spektrálních rámců, každý s 39 koeficienty. Dohromady 2379 vstupních příznaků.

### 6.3.2 Experiment s hloubkou a šířkou

Dále byl proveden experiment s šířkou a hloubkou lineární sítě. Výsledky tohoto experimentu jsou v tabulkách 6.2 a 6.3, první zobrazuje výsledky na validační sadě a druhá na sadě vývojové.

| Šířka \ Hloubka | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 64 | 79,76% | 80,94% | 81,22% | 82,07% | 81,64% | 81,3% |
| 128 | 80,44% | 81,32% | 82,33% | 82,81% | 82,7% | 81,85% |
| 256 | 80,64% | 80,05% | 82,72% | **83,01**% | 82,07% | 82,45% |
| 512 | 81,42% | 80,16% | 82,22% | 82,61% | 82,5% | 82,12% |
| 1024 | 80,06% | 80% | 81,75% | 82,38% | 82,06% | 82,47% |

Tabulka 6.2: *F-value* výsledků experimentu s hloubkou a šířkou sítě na validační sadě

| Šířka \ Hloubka | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 64 | 12,15% | 13,95% | 12,94% | **14,43**% | 13,09% | 12,69% |
| 128 | 11,59% | 13,89% | 13,43% | 13,54% | 12,83% | 12,06% |
| 256 | 9,78% | 9,16% | 12,18% | 12,96% | 12,79% | 12,23% |
| 512 | 10,91% | 8,76% | 11,29% | 11,39% | 12,07% | 4,83% |
| 1024 | 9,22% | 8,77% | 10,68% | 12,74% | 12,07% | 11,94% |

Tabulka 6.3: *F-value* výsledků experimentu s hloubkou a šířkou sítě na vývojové sadě

Výsledky na validačních datech jsou poměrně konzistentní, oproti tomu výsledky na vývojových datech jsou relativně volatilní. Je vidět, že přidaná kapacita větší šířky má zanedbatelný, nebo v případě vývojových dat negativní efekt na výsledky. Naopak síť do určité míry těží z větší hloubky. Jak na validační tak na vývojové sadě, sítě s 5 skrytými vrstvami dosáhly nejlepších výsledků. U šířky považujeme za nejlepší síť s 64 neurony na vrstvu. Tyto parametry se budou dále používat.

### 6.3.3 Experiment s aktivačními funkcemi

Poté se provedl experiment v aktivačními funkcemi. ReLU se obecně považuje za nejlepší, ale to nemusí být pravda pro všechny problémy. Každý experiment s aktivační funkcí byl proveden třikrát a výsledky zprůměrovány. V tabulce 6.4 jsou výsledky pro jednotlivé aktivační funkce.

| Aktivační funkce | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| ReLU | **82,42%** | **85,09%** | **11,97%** | **13,33%** |
| ELU | 81,34% | 84,13% | 11,49% | 12,49% |
| Tanh | 65,04% | 69,10% | 7,85% | 9,09% |
| Sigmoid | 68,22% | 72,83% | 8,40% | 10,58% |

Tabulka 6.4: *F-value* výsledků experiment s aktivačními funkcemi

Z výsledků je vidět, že síti nepomáhá normalizační schopnost *tanh* nebo *sigmoid*-u, ani přidání negativního výstupu *ELU*. Nejlepších výsledků dosáhla aktivační funkce ReLU. Proto se bude nadále používat.

## 6.4 Konvoluční neuronová síť

Jako druhá architektura pro experimenty byla konvoluční. Jako vstup tato síť považuje stejný kontext jako síť lineární, avšak v dvourozměrném formátu 61x39. Síť se skládá z konvolučních vrstev s jádrem 3x3, stride 1, bez dilatace a s obalem 1. Za každou druhou konvoluční vrstvou se nachází maximálová redukční vrstva, která pomocí jádra 2x2 redukuje oba rozměry na polovinu. Na konci je jedna lineární vrstva, která dekóduje výstupy z poslední konvoluční vrstvy, respektive redukční vrstvy. Jako aktivační funkce je zde používaná ReLU až na poslední lineární vrstvu, kde se stále používá *sigmoid*.

### 6.4.1 Experiment s počtem kanálů konvolučních vrstev

Nejprve se experimentovalo s počtem kanálů konvolučních vrstev. V tabulce 6.5 jsou zobrazeny výsledky.

| Počet kanálů | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 8 | 86,48% | 88,48% | 20,66% | 22,31% |
| 16 | **88,15%** | **89,55%** | **24,71%** | **27,23%** |
| 32 | 87,28% | 89,27% | 20,01% | 21,85% |
| 64 | 87,15% | 89,45% | 21,48% | 23,64% |

Tabulka 6.5: *F-value* výsledků experimentu s kanály konvolučních vrstev

Z tabulky je vidět, že více kanálů, a tím různých konvolučních jader nemá pozitivní vliv na výsledky. V těchto experimentech vyšlo použití 16 kanálů jako nejlepší jak na validačních datech, tak na vývojové sadě. 16 kanálů se bude dál používat v ostatních experimentech.

### 6.4.2 Experiment s počtem konvolučních a redukčních vrstev

Potom se zkoumal vliv počtu konvolučních vrstev společně s počtem redukčních vrstev. V prvním případě jsou redukční vrstvy za každou druhou konvoluční vrstvou. V druhém případě jsou redukční vrstvy umisťovány stejným způsobem, ale jsou ponechány jen první dvě. V tabulce 6.6 jsou výsledky tohoto experimentu.

| Hloubka | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 4 | 88,15% | 89,55% | 24,71% | 27,23% |
| 5 | 88,5% | 89,99% | 24,74% | 24,71% |
| 6 | 89,01% | 90,73% | 23,7% | 23,18% |
| 7 | **90,05**% | **91,36**% | 25,62% | 26,04% |
| 8 | 89,95% | 91,5% | 25,38% | 26,26% |
| 9 | 89,57% | 90,88% | 22,53% | 22,85% |
| 6 (2 redukční) | 88,93% | 90,43% | 24,25% | 26,77% |
| 7 (2 redukční) | 89,52% | 90,71% | **32,96**% | **35,45**% |
| 8 (2 redukční) | 89,82% | 91,3% | 18,98% | 19,9% |
| 9 (2 redukční) | 89,66% | 91,09% | 29,23% | 31,02% |

Tabulka 6.6: *F-value* výsledků experimentu s počtem konvolučních a redukčních vrstev

Z tabulky je možné pozorovat, že na validační data nemá počet konvolučních ani redukčních vrstev velký vliv. To může být způsobeno tím, že umělá validační data jsou méně různorodá než reálná vývojová data. Použití pouze dvou redukčních vrstev s vyšším celkovým počtem znamená větší rozměry uvnitř sítě, to očividně pomáhá síti se lépe naučit vystihnout reálná data. Nejlepších výsledků dosáhla síť se sedmi skrytými vrstvami a dvěma redukčními. Použití dvou redukčních vrstev bude ponecháno pro další experimenty.

### 6.4.3 Rozšíření trénovací datové sady

Po vyhodnocení průběžných výsledků bylo pozorováno, že síť chybuje v oblasti nádechů řečníků a nesprávně je označuje za crossnoise. Proto byla vytvořena a přidána datová sada s nádechy, dále byla přidána reálná trénovací sada za účelem dalšího zlepšení, obě jsou popsány v kapitole 5.2. Další experimenty zkoumaly nejlepší způsob trénování s rozšířenou datovou sadou.

### 6.4.4 Experiment trénování ve dvou fázích

Nejprve bylo trénování rozděleno do dvou fází. V první fázi se síť učila na datech s umělým crosstalk-em a nádechy. Tato fáze trvala 20 epoch. Na konci této fáze byla vybrána nejlepší síť podle *f-value* na validační sadě. V druhé fázi se tato nejlepší síť dále trénovala dalších 20 epoch na reálných trénovacích datech. Výstupní síť z této fáze byla ta, která během své epochy trénování dosáhne nejmenší trénovací chyby. Jako první proběhl experiment použití sítě bez normalizační vrstvy a s ní. Byla použita vrstvová normalizační vrstva. V tabulkách 6.7 a 6.8 jsou zobrazeny výsledky experimentu.

| Hloubka | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 5 | 69,82 % | 70,91 % | 19,82 % | 16,2 % |
| 6 | 60,43 % | 59,24 % | 16,82 % | 13,54 % |
| 7 | 66,6 % | 66,14 % | 19,43 % | 15,3 % |
| 8 | 69,34 % | 70,72 % | 21,1 % | 19,8 % |
| 9 | **72,88** % | **75,2** % | **22,41** % | **20,2** % |

Tabulka 6.7: *F-value* výsledků experimentu trénování ve dvou fázích s počtem vrstev bez normalizace

| Hloubka | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 5 | 61,94 % | 61,59 % | 24,27 % | 28,21 % |
| 6 | **71,16** % | **72,18** % | 10,55 % | 7,52 % |
| 7 | 68,51 % | 69,07 % | 23,68 % | 24,78 % |
| 8 | 69,62 % | 71,97 % | **26,77** % | **32,66** % |
| 9 | 68,57 % | 68,28 % | 26,49 % | 32,04 % |

Tabulka 6.8: *F-value* výsledků experimentu trénování ve dvou fázích s počtem vrstev s normalizací

Z tabulek je vidět, že výsledky na validační sadě jsou výrazně horší než u předešlých experimentů, ale to se očekávalo, protože se síť učí naposledy na reálných datech, která jsou poměrně odlišná od umělých. Mezi sítí s normalizační vrstvou a bez ní nejsou velké rozdíly u výsledků na validační sadě, avšak u reálných dat v tomto případě normalizace pomáhá síti dosáhnout lepších výsledků. U výsledků s normalizační vrstvou je zajímavé, že síť která dosáhla nejlepších výsledků na validační sadě dosáhla nejhorších výsledků na vývojové sadě. K tomuhle mohlo dojít tím, že se síť v první části naučila dobře umělá data, ale v druhé části se už nedokázala dobře naučit reálná data.

### 6.4.5 Experiment trénování v jedné fázi

Potom proběhl experiment pro trénování v jedné fázi. Opět s použitím vrstvové normalizační vrstvy a bez ní. Trénování v jedné fázi probíhalo najednou na umělých datech, datech s nádechy a reálnými daty. Výsledná síť byla vybrána podle největší *f-value* na validační sadě. V tabulkách 6.9 a 6.10 jsou výsledky tohoto experimentu.

| Hloubka | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 6 | 90,74 % | 92,2 % | 22,26 % | 21,55 % |
| 7 | 91,19 % | 92,44 % | 23,73 % | 23,92 % |
| 8 | 91,02 % | 92,55 % | **29,87** % | **30,12** % |
| 9 | **91,55** % | **92,95** % | 21,22 % | 20,59 % |

Tabulka 6.9: *F-value* výsledků experimentu trénování v jedné fázi s počtem vrstev s normalizací

Výsledky na validačních datech jsou opět vyšší a je vidět, že s normalizační vrstvou na nich síť dosahuje lepších výsledků.

| Hloubka | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 5 | 88,93 % | 90,16 % | 31,68 % | 29,98 % |
| 6 | 89,17 % | 90,91 % | 20,4 % | 19,39 % |
| 7 | 89,73 % | 91,16 % | **34,05** % | **36,36** % |
| 8 | 89,81 % | 91,31 % | 29,28 % | 29,51 % |
| 9 | **89,82** % | **91,41** % | 25,91 % | 24,91 % |

Tabulka 6.10: *F-value* výsledků experimentu trénování v jedné fázi s počtem vrstev bez normalizace

Síť bez normalizace dosahuje průměrně vyšší přesnosti na vývojových datech. Nejlepších výsledků dosáhla síť se sedmi skrytými vrstvami bez normalizace při trénování v jedné fázi. Dále budou experimenty probíhat v jedné fázi s rozšířenou datovou sadou, protože dosáhly lepších výsledků než na nerozšířené datové sadě. A nebudou se používat normalizační vrstvy, protože nedosáhly lepších výsledků.

### 6.4.6 Experiment s aktivační funkcí ELU

Dále byl proveden experiment s použitím aktivační funkce ELU namísto dosavadně používané ReLU v předchozím experimentu. V tabulce 6.11 jsou ukázány výsledky.

| Hloubka | Validační | Validační s PP | Vývojové | Vývojové s PP |
|---|---|---|---|---|
| 5 | 89,44 % | 90,86 % | 30,39 % | 28,46 % |
| 6 | 89,55 % | 91,02 % | 24,52 % | 26,52 % |
| 7 | 89,83 % | 91,36 % | 29,72 % | 32,4 % |
| 8 | 89,91 % | 91,36 % | 28,2 % | 30,17 % |
| 9 | **90,38** % | **91,61** % | **31,09** % | **32,79**% |

Tabulka 6.11: *F-value* výsledků experimentu s aktivační funkcí ELU

Síť dosáhla nepatrně lepších výsledků na validačních datech, ale horších na vývojových datech. Nejlepších výsledků dosáhla síť s devíti skrytými vrstvami. Nadále se bude používat ReLU, protože ELU nedosáhla lepších výsledků.

## 6.5 Time delay neuronová síť

Další experimenty probíhaly na architektuře time delay. Ta se skládá z jednorozměrných konvolučních vrstev s dilatací tak, aby simulovaly požadovaný kontext. Základní verze používá ReLU jako aktivační funkce a kontext [-1 0 +1].

### 6.5.1 Experiment s hloubkou a šířkou sítě

První proběhl experiment s šířkou a hloubkou sítě. V tabulkách 6.12 a 6.13 jsou zobrazeny výsledky na validační a vývojové sadě.

| Šířka \ Hloubka | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|
| 32 | 90,71 | 90,34 | 90,45 | 90,29 | 90,4 |
| 64 | 90,21 | 90,65 | 90,36 | 90,32 | 90,23 |
| 128 | 90,61 | 91,15 | 90,5 | 90,82 | **91,37** |
| 256 | 91,15 | 91,19 | 91,05 | 91,22 | 91,26 |

Tabulka 6.12: *F-value* výsledků experimentu s šířkou a hloubkou sítě na validačních datech

| Šířka \ Hloubka | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|
| 32 | 19,02 | 20,19 | 24,41 | 16,16 | 16,07 |
| 64 | 18,83 | 23,45 | 17,13 | 18,45 | **26,63** |
| 128 | 8,57 | 10,4 | 15,92 | 16,77 | 12,78 |
| 256 | 5,05 | 13,54 | 8,07 | 9,29 | 7,26 |

Tabulka 6.13: *F-value* výsledků experimentu s šířkou a hloubkou sítě na vývojových datech

Na výsledcích na vývojové sadě je vidět trend zhoršující se přesnosti s šířkou. Oproti tomu se přesnost na validačních datech nepatrně zlepšuje s šířkou. Podle výsledků nemá hloubka sítě velice výrazný vliv na dosaženou přesnost. Nejlepších výsledků dosáhla síť s devíti skrytými vrstvami a s šířkou 64.

### 6.5.2 Experiment s rozšířeným kontextem

Dále se experimentovalo s rozšířeným kontextem [-2 0 +2]. V tabulkách 6.14 a 6.15 jsou výsledky experimentu.

| Šířka \ Hloubka | 7 | 8 | 9 |
|---|---|---|---|
| 16 | 89,89 | 89,33 | 89,71 |
| 32 | 90,01 | 89,84 | **90,28** |
| 64 | 90,02 | 89,76 | 89,93 |

Tabulka 6.14: *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným kontextem na validačních datech

| Šířka \ Hloubka | 7 | 8 | 9 |
|---|---|---|---|
| 16 | 3,06 | 3,75 | 4,24 |
| 32 | 5,06 | **5,57** | 3,37 |
| 64 | 5,28 | 4,51 | 4,31 |

Tabulka 6.15: *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným kontextem na vývojových datech

Validační přesnost se zmenšila oproti menšímu kontextu, stejně tak i přesnost na vývojových datech. Na těch je pokles ještě větší.

### 6.5.3 Experiment s plným kontextem

Poté se experimentovalo s plným kontextem [-2 -1 0 +1 +2]. v tabulkách 6.16 a 6.17 jsou výsledky s rozšířeným plným kontextem.

| Šířka \ Hloubka | 7 | 8 | 9 |
|---|---|---|---|
| 16 | 89,84 | 89,91 | 89,44 |
| 32 | **90,22** | 89,91 | 89,14 |
| 64 | 89,98 | 89,73 | 89,77 |

Tabulka 6.16: *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným plným kontextem na validačních datech

| Šířka \ Hloubka | 7 | 8 | 9 |
|---|---|---|---|
| 16 | 3,44 | 6,27 | 2,94 |
| 32 | 1,42 | 2,85 | **6,84** |
| 64 | 2,85 | 4,66 | 1,08 |

Tabulka 6.17: *F-value* výsledků experimentu s šířkou a hloubkou sítě s rozšířeným plným kontextem na vývojových datech

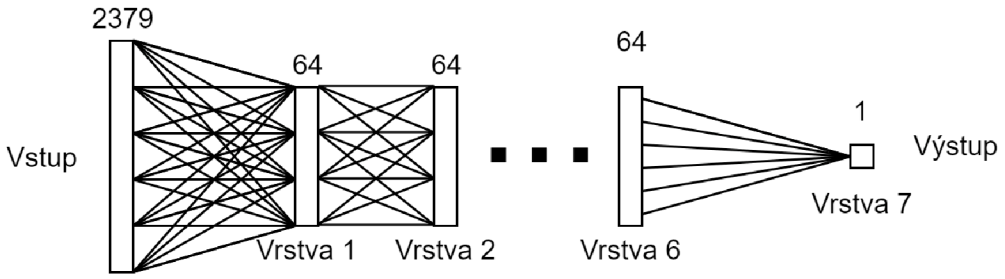
Validační přesnost se opět nepatrně zmenšila oproti menšímu kontextu i oproti necelému kontextu. Nejlepší přesnost na vývojových datech je vyšší než s necelým kontextem, ale stále menší než s původním kontextem [-1 0 +1].

## 6.6 Testování finálních neuronových sítí

Při trénování konečných sítí byl změněn parametr optimalizátoru $\eta$, neboli *learn rate* na 0,0001 pro zlepšení stability a omezení přetrénování s více daty. Trénování probíhalo v jedné fázi, tak že se trénuje po částech umělých dat velkých 5000 záznamů a náhodně se do toho jednou za epochu vloží nádechy a reálná data.

### 6.6.1 Finální architektura lineární neuronové sítě

Na obrázku 6.1 je vyobrazena konečná architektura lineární sítě sestavená z parametrů dosahující nejlepší výsledky v experimentech. V tabulce 6.18 jsou rozepsány parametry jednotlivých vrstev.



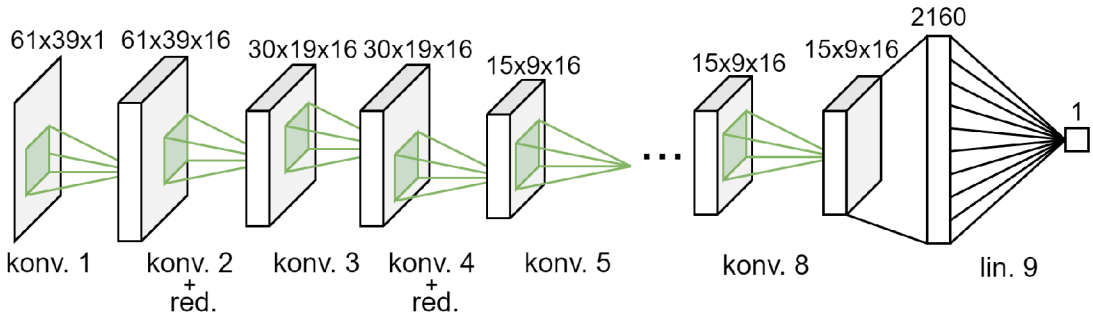

Obrázek 6.1: Architektura lineární neuronové sítě

| | Vstupní rozměry | Výstupní rozměry | Aktivační funkce |
|---|---|---|---|
| Lineární vrstva 1 | 2379 | 64 | ReLU |
| Lineární vrstva 2-6 | 64 | 64 | ReLU |
| Lineární vrstva 7 | 64 | 1 | Sigmoid |

Tabulka 6.18: Parametry vrstev finální lineární neuronové sítě

### 6.6.2 Finální architektura konvoluční neuronové sítě

Pro trénování v jedné fázi s ReLU z kapitoly 6.4.5 vyšlo použití 7 skrytých vrstev bez normalizace nejlépe. Výsledná architektura konvoluční sítě je na obrázku 6.2 a její parametry jsou v tabulce 6.19.




Obrázek 6.2: Architektura konvoluční neuronové sítě

| | Vstupní rozměry | Výstupní rozměry | Jádro | Aktivační fn. |
|---|---|---|---|---|
| Konvoluční vrstva 1 | 61x39x1 | 61x39x16 | 3x3 | ReLU |
| Konvoluční vrstva 2 | 61x39x16 | 61x39x16 | 3x3 | - |
| Redukční vrstva | 61x39x16 | 30x19x16 | 2x2 | ReLU |
| Konvoluční vrstva 3 | 30x19x16 | 30x19x16 | 3x3 | ReLU |
| Konvoluční vrstva 4 | 30x19x16 | 30x19x16 | 3x3 | - |
| Redukční vrstva | 30x19x16 | 15x9x16 | 2x2 | ReLU |
| Konvoluční vrstva 5-8 | 15x9x16 | 15x9x16 | 3x3 | ReLU |
| Lineární vrstva 9 | 2160 | 1 | - | Sigmoid |

Tabulka 6.19: Parametry vrstev výsledné konvoluční neuronové sítě

### 6.6.3 Finální architektura time delay neuronové sítě

Výsledná architektura sítě typu time delay je s kontextem [-1 0 +1], hloubkou 9 a šířkou 64, v tabulce 6.20 jsou její další parametry.

| | Vstupní rozměry | Výstupní rozměry | Aktivační funkce |
|---|---|---|---|
| Time delay vrstva 1 | 61x39 | 59x39 | ReLU |
| Time delay vrstva 2 | 59x39 | 57x39 | ReLU |
| Time delay vrstva 3 | 57x39 | 55x39 | ReLU |
| Time delay vrstva 4 | 55x39 | 53x39 | ReLU |
| Time delay vrstva 5 | 53x39 | 51x39 | ReLU |
| Time delay vrstva 6 | 51x39 | 49x39 | ReLU |
| Time delay vrstva 7 | 49x39 | 47x39 | ReLU |
| Time delay vrstva 8 | 47x39 | 45x39 | ReLU |
| Time delay vrstva 9 | 45x39 | 43x39 | ReLU |
| Time delay vrstva 10 | 43x39 | 41x39 | ReLU |
| Linární vrstva 11 | 1599 | 1 | Sigmoid |

Tabulka 6.20: Parametry vrstev výsledné time delay neuronové sítě

### 6.6.4 Výsledky

V tabulce 6.21 jsou výsledky finálních sítí jednotlivých architektur.

| Architektura | Val. | Val. s PP | Umělá | Umělá s PP | reálná | reálná s PP |
|---|---|---|---|---|---|---|
| Lineární | 80,88 | 82,34 | 81,22 | 82,48 | 4,91 | 4,46 |
| Konvoluční | 90,91 | 91,45 | 91,59 | 92,25 | **5,02** | **5,07** |
| Time Delay | **91,83** | **92,21** | **92,78** | **93,15** | 4,89 | 4,60 |

Tabulka 6.21: *F-value* výsledků finálních sítí

Výsledky na umělých datech lehce předčily očekávané výsledky podle experimentů, protože jsou vyšší než výsledky na validačních datech. Všechny tři architektury dosáhli velmi špatných výsledků na reálné testovací sadě. Jsou výrazně horší než se očekávalo podle výsledků na vývojové sadě v experimentální části. V tabulkách 6.22 a 6.23 jsou detailněji uvedeny výsledky na testovacích sadách bez zpracování výstupu.

| Architektura | Preciznost | Výtěžnost | *F-value* |
|---|---|---|---|
| Lineární | 86,74 | 76,37 | 81,22 |
| Konvoluční | 93,94 | 89,35 | 91,59 |
| Time Delay | **95,82** | **89,93** | **92,78** |

Tabulka 6.22: Výsledky finálních sítí na umělé testovací sadě

| Architektura | Preciznost | Výtěžnost | *F-value* |
|---|---|---|---|
| Lineární | 2,759 | **22,191** | 4,91 |
| Konvoluční | **2,944** | 17,094 | **5,02** |
| Time Delay | 2,912 | 15,214 | 4,89 |

Tabulka 6.23: Výsledky finálních sítí na reálné testovací sadě

Experimenty dokázaly, že sítě dokáží dosáhnout mnohem vyšších výsledků na reálných datech. Z výsledků na umělých datech je vidět, že se síť dokáže dobře naučit modelovat data, ale je zřejmé, že umělá data nebyla dostatečně přesnou simulací reálné překrývající se řeči. Je pravděpodobné, že by stejné architektury dosáhly dobrých výsledků na reálných datech pokud by větší část trénování probíhala na skutečných datech. Protože reálná data obsahují poměrně málo překrývající se řeči, bylo by nejspíše potřeba více dat, než je v této době k dispozici.

Z výsledků v tabulce 6.23 je vidět, že síť mnohem častěji chybovala tak, že označovala úseky jako překryv, když se tam ve skutečnosti nenacházel. Reálná testovací sada obsahuje více překryvů než vývojová sada, což by mohlo způsobovat tyto problémy. Také kompletní trénovací data obsahují větší poměr překryvů, než

omezená trénovací sada pro experimenty.

Výsledky na reálné testovací sadě jsou podobné pro všechny tři architektury. Nepatrně lepšího výsledků dosáhla konvoluční síť stejně jako nejvyšší přesnosti v průběhu experimentů. Proto ji považujeme za nejúspěšnější síť na reálných datech. Na umělých datech dosáhla nejlepších výsledků jak v experimentální části, tak v této výsledné části síť architektury time delay.

## 6.7 Poznámky k trénování

Počítač na němž byly sítě trénovány byl vybaven čtyř jádrovým procesorem Intel i7-6700K, grafickou kartou NVIDIA GTX 1070 Ti s 8 GB pamětí a 16 GB RAM paměti. Dohromady experimenty trvaly zhruba 22 dní 5 hodin čistého výpočetního času a trénování výsledných sítí trvalo 3 dny 2 hodiny čistého výpočetního času. Epocha trénování finální lineární sítě trvala přibližně 17 minut, konvoluční sítě 94 minut a time delay sítě 114 minut.

# 7 Závěr

Pro úlohu detekce překrývacích se úseků řeči byly použity MFCC příznaky a jako klasifikátory byly navrženy neuronové sítě různých architektur.

Byly vytvořeny datové sady pro trénování, validaci, vývoj a testování neuronových sítí. Pro trénování bylo uměle vytvořeno velké množství trénovacích dat simulující překrývající se řeč, k tomu byla přidána část ručně anotovaných dat. Ručně anotovaná data byla dále použita ve vývojových a testovacích datových sadách. Pro tento účel bylo ručně anotováno pět hodit audio nahrávek. Pro potřeby ruční anotace vznikla základní verze podpůrného programu.

Pro detekci překrývajících se úseků řeči byly navrženy tři architektury neuronových sítí. Byly to lineární, kovnoluční a time delay architektury. Pomocí experimentů byly parametry těchto architektur postupně laděny pro co nejlepší výsledky na vývojové datové sadě. Nakonec byly nejlepší z nich vyhodnoceny na umělých datech a na ručně anotovaných reálných datech.

Přidání reálných dat do trénovací sady nepomohlo znatelně zlepšit výsledky experimentů. To bylo nejspíš způsobeno tím, že reálná data mají velký nepoměr mezi překrývající se řečí a jediným mluvčím nebo tichem. Taky těchto reálných dat bylo oproti umělým datům málo.

Všechny sítě dosáhly očekávaných výsledků na umělých datech. Avšak na reálných dopadly všechny výrazně hůře než se očekávalo. Výsledky experimentů na vývojových datech ukazovaly schopnost sítí se alespoň do určité míry naučit vystihnout skutečná data. A to i při trénování pouze na umělých datech. Tohle se pravděpodobně stalo kvůli rozdílům mezi reálnými daty používanými na trénování, ladění a testování sítě. I přes špatné výsledky se konvoluční a time delay architektury zdají

jako vhodné pro detekci překrývajících se úseků řeči.

Ke zlepšení výsledků na reálných datech by mělo pomoci použít větší množství skutečných dat na trénování a vytvářet umělá data tak, aby se lépe shodovala se skutečnými daty. Takové zlepšení by mohlo přinést mnohem komplexnější vytváření umělých dat. Třeba tak, že by se z anotovaných dat extrahovaly úseky nádechů, přitakávání a podobné, a to by se kombinovalo s promluvou jednoho mluvčího. Tím by se přesněji simuloval jeden z druhů překryvů řeči definovaný v kapitole 2.1. Podobným způsobem by bylo možné vytvářet i ostatní druhy tak, aby i poměr více odpovídal skutečným datům. Případně by mohlo být výhodné provádět trénování na více fází. Například v první fázi by se sít učila s umělými daty, které byly podobné umělým datům v této práci. To by mohla být vhodná první fáze trénování, protože nepoměr tříd v reálných datech by mohl způsobovat obtíže s trénováním. Potom by se síť učila na vylepšené umělé sadě. Do toho by se mohly přidat reálná data, nebo by se na nich trénovalo v další fázi. Aby se síť mohla efektivně učit pouze na reálných datech, bylo by potřeba na nich zároveň validovat trénování, podle kterého by se následně vybírala nejlepší epocha a výsledná síť.

Tato práce pomohla rozšířit datovou sadu obsahující reálná ručně anotovaná data, které poslouží v dalších pracích zaměřených na strojové učení v oblasti zpracování zvuku. Také provedla průzkum použití různých architektur neuronových sítí pro detekci překrývajících se úseků řeči pro český a slovenský jazyk.

# Použitá literatura


1. FUJITA, Yusuke et al. *End-to-End Neural Speaker Diarization with Self-attention*. arXiv, 2019. Dostupné z DOI: 10.48550/ARXIV.1909.06247.
2. ALAM, Mahbubul et al. *Survey on Deep Neural Networks in Speech and Vision Systems*. arXiv, 2019. Dostupné z DOI: 10.48550/ARXIV.1908.07656.
3. PARK, Tae Jin et al. *A Review of Speaker Diarization: Recent Advances with Deep Learning*. arXiv, 2021. Dostupné z DOI: 10.48550/ARXIV.2101.09624.
4. ABDUL QAYYUM, Alif Bin, Asiful AREFEEN a Celia SHAHNAZ. Convolutional Neural Network (CNN) Based Speech-Emotion Recognition. In: *2019 IEEE International Conference on Signal Processing, Information, Communication Systems (SPICSCON)*. 2019, s. 122–125. Dostupné z DOI: 10.1109/SPICSCON48833.2019.9065172.
5. GEIGER, J.T. et al. Detecting overlapping speech with long short-term memory recurrent neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2013, s. 1668–1672.
6. SIEGERT, Ingo et al. Overlapping speech, utterance duration and affective content in HHI and HCI - An comparison. In: *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. 2015, s. 83–88. Dostupné z DOI: 10.1109/CogInfoCom.2015.7390569.
7. SACKS, Harvey, Emanuel A. SCHEGLOFF a Gail JEFFERSON. A Simplest Systematics for the Organization of Turn-Taking for Conversation. *Language*

[online]. 1974, roč. 50, č. 4, s. 696–735 [cit. 2022-05-03]. ISSN 00978507, ISSN 15350665. Dostupné z: http://www.jstor.org/stable/412243.

8. BOAKYE, Kofi et al. Overlapped speech detection for improved speaker diarization in multiparty meetings. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2008, s. 4353–4356. Dostupné z DOI: 10.1109/ICASSP.2008.4518619.

9. ZELENÁK, Martin a Javier HERNANDO. The Detection of Overlapping Speech with Prosodic Features for Speaker Diarization. In: 2011, s. 1041–1044.

10. GEIGER, J.T. et al. Detecting overlapping speech with long short-term memory recurrent neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2013, s. 1668–1672.

11. ANDREI, Valentin, Horia CUCU a Corneliu BURILEANU. Detecting Overlapped Speech on Short Timeframes Using Deep Learning. In: 2017, s. 1198–1202. Dostupné z DOI: 10.21437/Interspeech.2017-188.

12. ANDREI, Valentin, Horia CUCU a Corneliu BURILEANU. Overlapped Speech Detection and Competing Speaker Counting–-Humans Versus Deep Learning. *IEEE Journal of Selected Topics in Signal Processing*. 2019, roč. 13, č. 4, s. 850–862. Dostupné z DOI: 10.1109/JSTSP.2019.2910759.

13. YOUSEFI, Midia a John H. L. HANSEN. Block-Based High Performance CNN Architectures for Frame-Level Overlapping Speech Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2021, roč. 29, s. 28–40. Dostupné z DOI: 10.1109/TASLP.2020.3036237.

14. HE, Kaiming et al. *Deep Residual Learning for Image Recognition*. arXiv, 2015. Dostupné z DOI: 10.48550/ARXIV.1512.03385.

15. JUNG, Jee-weon et al. *Three-class Overlapped Speech Detection using a Convolutional Recurrent Neural Network*. arXiv, 2021. Dostupné z DOI: 10.48550/ARXIV.2104.02878.

16. LAI, Yongquan et al. *End-to-end speaker diarization with transformer*. arXiv, 2021. Dostupné z DOI: 10.48550/ARXIV.2112.07463.

17. *Mel-frequency cepstrum*. San Francisco (CA): Wikimedia Foundation, 2001--2022. Dostupné také z: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum.

18. CHIBELUSHI, Claude, Farzin DERAVI a John MASON. A review of speech--based bimodal recognition. *Multimedia, IEEE Transactions on*. 2002, roč. 4, s. 23–37. Dostupné z DOI: 10.1109/6046.985551.

19. *Artificial neural network*. San Francisco (CA): Wikimedia Foundation, 2001--2022. Dostupné také z: https://en.wikipedia.org/wiki/Artificial_neural_network.

20. *7 Types of Activation Functions in Neural Network*. Uttar Pradesh, 2021. Dostupné také z: https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network.

21. KINGMA, Diederik P. a Jimmy BA. *Adam: A Method for Stochastic Optimization*. arXiv, 2014. Dostupné z DOI: 10.48550/ARXIV.1412.6980.

22. *BCELOSS*. 2022. Dostupné také z: https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html.

23. NABIL, Madali. *Linear Regression With One Variable | Gradient descent*. 2022. Dostupné také z: https://medium.com/@madali.nabil97/linear-regression-with-one-variable-gradient-descent-1dea65fef0a8.

24. QIAN, Ning. On the momentum term in gradient descent learning algorithms. *Neural Networks*. 1999, roč. 12, č. 1, s. 145–151. ISSN 0893-6080. Dostupné z DOI: https://doi.org/10.1016/S0893-6080(98)00116-6.

25. SRIVASTAVA, Nitish et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014, roč. 15, s. 1929–1958.

26. BINDAL, Aakash. *Normalization Techniques in Deep Neural Networks*. 2019. Dostupné také z: https://medium.com/techspace-usict/normalization-techniques-in-deep-neural-networks-9121bf100d8.

27. WU, Yuxin a Kaiming HE. *Group Normalization*. arXiv, 2018. Dostupné z DOI: 10.48550/ARXIV.1803.08494.

28. IOFFE, Sergey a Christian SZEGEDY. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv, 2015. Dostupné z DOI: 10.48550/ARXIV.1502.03167.

29. BA, Jimmy Lei, Jamie Ryan KIROS a Geoffrey E. HINTON. *Layer Normalization*. arXiv, 2016. Dostupné z DOI: 10.48550/ARXIV.1607.06450.

30. BELTRAMELLI, Tony a Sebastian RISI. Deep-Spying: Spying using Smartwatch and Deep Learning. 2015.

31. VÉSTIAS, Mário. A Survey of Convolutional Neural Networks on Edge with Reconfigurable Computing. *Algorithms*. 2019, roč. 12, s. 154. Dostupné z DOI: 10.3390/a12080154.

32. *Convolution arithmetic*. [B.r.]. Dostupné také z: https://github.com/vdumoulin/conv_arithmetic.

33. GIRGIN, Samet. *Day-41 Deep Learning-6 (CNN-2)*. 2019. Dostupné také z: https://medium.com/pursuitnotes/day-41-deep-learning-6-cnn-2-d38b355bd0ba.

34. PEDDINTI, Vijayaditya, Daniel POVEY a Sanjeev KHUDANPUR. A time delay neural network architecture for efficient modeling of long temporal contexts. In: 2015, s. 3214–3218. Dostupné z DOI: 10.21437/Interspeech.2015-647.

35. *PyTorch*. 2022. Dostupné také z: https://pytorch.org/.

36. *HTK Speech Recognition Toolkit*. [B.r.]. Dostupné také z: https://htk.eng.cam.ac.uk/.

# A Přílohy

- Zdrojové kódy
- Vytvořené anotace reálných dat