TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automatické odezírání ze rtů pomocí neuronových sítí

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Jakub Kovář**
Vedoucí práce: Ing. Karel Paleček, Ph.D.

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Automatické odezírání ze rtů pomocí neuronových sítí

Jméno a příjmení: **Jakub Kovář**
Osobní číslo: M17000022
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s problematikou automatického odezírání ze rtů pomocí umělých neuronových sítí.
2. Sestavte dataset audiovizuálních nahrávek a vhodně upravte pro snadné zpracování neuronovými sítěmi.
3. Navrhněte vhodnou architekturu neuronové sítě pro automatické odezírání ze rtů z obrazu s příp. využitím zvukopé stopy.
4. Porovnejte navržený systém s volně dostupnými řešeními pomocí standardně užívaných metrik.

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

Seznam odborné literatury:

[1] Goodfellow, I., Bengio, Y., Courville, A. Deep learning. MIT Press, 2016
[2] Bishop, C. Pattern Recognition and Machine Learning. 2006. ISBN 13: 978-038731073
[3] Karpathy, A., Johnson, J., Li, F. Convolutional neural neworks for visual recognition. dostupné online: http://cs231n.stanford.edu/

Vedoucí práce:	Ing. Karel Paleček, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 18. října 2019

Prohlášení

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

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

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

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

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

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

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

27. 5. 2020 Jakub Kovář

Poděkování

Rád bych poděkoval všem za podporu při zpracovávání této práce. Zejména mému vedoucímu panu Ing. Karelu Palečkovi, Ph.D. za odborné vedení, trpělivost a ochotu.

Automatické odezírání ze rtů pomocí neuronových sítí

Abstrakt

V této práci je popsána problematika odezírání ze rtů, stručně uvedené práce zabývající se touto problematikou a seznámení se s vhodnými architekturami neuronových sítí. Je zde proveden pokus o návrh modelu schopného odezírání ze rtů, zběžný popis postupu návrhu a optimalizace modelů využívající neuronové sítě. Jsou zde použité námi upravené datasety *OuluVS* a *OuluVS2* k natrénování našich modelů. V poslední kapitole jsou komentovány dosažené výsledky konvolučního a rekurentního modelu pro izolované fráze a sekvence získané křížovou validací.

Klíčová slova: Odezírání ze rtů, extrakce vizuálních charakteristik, konvoluční neuronové sítě, rekurentní neuronové sítě, hluboké učení

Neural networks for automatic lip reading

Abstract

This work describes problems with lip reading, briefly mentions related works and introduces us to viable neural network architectures. It contains attempt to design model capable of lip reading, perfunctory process of desing and optimalization of models using neural networks. Datasets *OuluVS* and *OuluVS2* are used for training of our models. Results of convolutional and reucurrent model are obtained using cross-validation and are discussed in the last chapter.

Keywords: lip reading, visual feature extracion, convolutional neural networks, recurrent neural networks, deep learning

Obsah

Seznam zkratek 12

1 Úvod **13**

2 Problematika odezírání ze rtů **14**

2.1 Přehled prací a metod zabývající se problematikou odezírání ze rtů . 14

2.2 Vrstvená neuronová síť 16

2.3 Konvoluční sítě 16

2.4 Rekurentní sítě 20

2.4.1 Elman recurrent neural network 21

2.4.2 Long short-term memory 21

2.4.3 Gated Recurrent unit 21

3 Trénování neuronových sítí v úloze odezíraní ze rtů **23**

3.1 Dropout 23

3.2 Overfitting 23

3.2.1 Early stopping 24

3.2.2 Batch normalization 24

3.3 Učení s učitelem 25

3.3.1 Cross-entropy loss 25

3.3.2 Connectionist temporal classification loss . . . 26
3.4 Křížová validace . . . 26

4 Použitá sada dat **27**
4.1 Práce s daty . . . 29
4.1.1 Rozšíření dat . . . 29
4.2 Rozdělení dat . . . 31

5 Experimenty **32**
5.1 Konvoluční model . . . 32
5.2 Rekurentní model . . . 37
5.3 Rekurentní model pro rozpoznávání sekvencí . . . 38
5.4 Hardwarové limitace . . . 39
5.5 Výsledky . . . 40
5.6 Dosavadní nejlepší výsledky . . . 42

6 Závěr **43**

Použitá literatura **44**

A Obsah přiloženého DVD **48**

Seznam obrázků

2.1 Vztah vstupních a výstupních kanálů konvoluční vrstvy [9] 18

2.2 *Vrstvy VGG-19 architektury* [10] 19

2.3 Základní blok ResNet-34 [11] 19

2.4 Vnitřní struktura LSTM [13] 22

2.5 Vnitřní struktura GRU [14] 22

3.1 Ukázka *Overfitting* [16] 24

4.1 Ukázka snímků z datasetu 28

4.2 Ukázka rozšíření dat 30

5.1 Architektura konvolučního modelu 33

5.2 Architektura finálního konvolučního modelu 35

5.3 Ukázka průběhu trénování sítě 36

5.4 Architektura rekurentního modelu 37

5.5 Ukázka trénování rekurentního modelu 38

5.6 Architektura rekurentního modelu pro sekvence 39

5.7 Ukázka trénování rekurentního modelu pro sekvence 39

Seznam tabulek

4.1 Přehled dat 27

4.2 Přehled dat 28

5.1 Experimenty s vrstvami 34

5.2 parametry konvolučních vrstev modelu 34

5.3 Experimenty s rozšířením dat 35

5.4 Výsledky na datasetu OuluVS 40

5.5 Výsledky na datasetu OuluVS2 41

Seznam zkratek

LSTM	Long Short Term Memory, architektura rekurentní neuronové sítě
CNN	Convolutional Neural Network, konvoluční neuronová sít
CTC	Connectionist Temporal Classification, typ výstupu a bodování neuronové sítě
GRU	Gated Recurrent Unit, architektura rekurentní neuronové sítě
MFCC	Mel-Frequency Cepstrum Coefficient, koeficienty reprezentující hustotu spektra zvuku
MLP	Multi-Layer Perceptron, třída neuronové sítě
RGB	Red Green Blue, aditivní způsob míchání barev používaný pro obrázky
ReLU	Rectified Linear Unit, aktivační funkce
PReLU	Parametric ReLU, aktivační funkce
SGD	stochastic gradient descent, optimalizační algoritmus
CUDA	Compute Unified Device Architecture, architektura grafických karet
GPU	Graphics Processing Unit, grafická karta
WER	Word Error Rate, míra chyb na úrovní slov

1 Úvod

Neuronové sítě se obecně za posledních několik let začaly využívat pro spoustu účelů. Snad jeden z největších účelů je zpracování obrazových informací jako je rozeznávání textu a klasifikace objektů, k tomu patří i odezírání ze rtů. V této práci se seznámíme s problematikou a uvedeme používané metody pro její řešení. Navrhneme a zrealizujeme architektury neuronových sítí, které budou schopné se naučit odezírat ze rtů. Budeme je realizovat v programovacím jazyce Python, s využitím knihoven Pytorch, které obsahují všechny nástroje na realizaci neuronových sítí. Popíšeme použitá data a jejich úpravu pro zpracování. A zhodnotíme dosažené výsledky.

2 Problematika odezírání ze rtů

Problematika odezírání ze rtů je jeden z mnoha oborů, kde se za posledních pár let osvědčilo použít neuronové sítě. Sice nebylo napsáno tolik prací ohledně hlubokého učení zaměřených na toto téma, ale i tak bylo dosaženo obdivuhodných výsledků. A to také díky možnosti využití pokroků a metod využívaných v ostatních oblastech. Motivací k vyřešení problému odezírání ze rtů je zejména využití společně s dalšími systémy, například k diktování textu v rušném prostředí, při poslechu více řečníků. A k obecnému zlepšení těchto systémů.
Hlavní obtíž způsobuje fakt, že některé fonémy sdílejí stejné postavení rtů a jazyka. Tudíž zde existuje nejasnost, které se nelze zbavit. Velká část komunikace závisí na zvuku a kvůli zmíněným nejasnostem je velmi těžké dosáhnout dobrých výsledků bez přidaného kontextu.

2.1 Přehled prací a metod zabývající se problematikou odezírání ze rtů

Existuje několik prací zabývající se touto problematikou. Dají se zhruba rozdělit podle úrovně rozeznávání a to od jednotlivých fonémů až po celé fráze.
V práci *Lipreading using Convolutional Neural Network* [1] autoři porovnávají využití konvolučních sítí oproti ručně vytvořeným metodám k získání vizuálních příznaků. Konvoluční síť trénovali rozeznávat fonémy ze snímků a pomocí skrytého Markova modelu skládali izolovaná slova. Na sadě dat 300 japonských slov s šesti mluvčími dosáhli lepších výsledků než ručně vytvořené metody.

Práce *Lip reading using CNN and LSTM* [2] se zabývá využitím konvoluce a rekurentních modelů k rozpoznání slov a frází. Porovnává zřetězení snímků do jednoho pro různý počet použitých snímků a použití normalizace. Dále využívají předtrénovanou dvourozměrnou konvoluční síť *VGGNet* (architektura hluboké konvoluční sítě podle [3]) na extrakci charakteristik a LSTM (Long short-term memory) k extrakci časové informace z posloupnosti snímků. Autoři se pokoušeli se využít konvoluční síť trénovanou od nuly, ale nepodařilo se jim dosáhnout uspokojivých výsledků. Nejlepších výsledků dosáhli před-trénovanou *VGGNet* sítí s použitím největšího počtu znormalizovaných snímků. LSTM dosáhlo uspokojivé, ale nižší přesnosti.

Práce *Lip Reading Using Committee Networks With Two Different Types of Concatenated Frame Images* [4] používá k rozeznání frází a číslic komisní síť, která posuzuje výsledky dvou konvolučních sítí. Každá konvoluční síť, založená na *VGGNet*, je trénována na odlišně zřetězených snímcích. Jedna snímky pouze poskládá podle pořadí do jednoho. Druhá rozdělí snímky podle rysů rtů, což zajistí, že jednotlivé rysy budou vždy na stejném místě. Dosáhli výsledků potvrzujících zlepšení při využití kombinace více sítí. Dále docílili lepších výsledků než před-trénovaná síť na velké sadě dat.

V *LipNet: End-to-End Sentence-level Lipreading* [5] autoři vytvářeli síť schopnou mapovat různě dlouhé sekvence video snímků na text, a to na úrovni vět. V práci využívali trojrozměrnou konvoluční sít, rekurentní sít a kriteriální funkci *CTC (connectionist temporal classification) Loss*. Jako rekurentní síť používali dvojici obousměrných GRU (gated recurrent unit). *LipNet* byla první *end-to-end* síť schopná se zároveň učit prostoro-temporální vizuální příznaky a sekvenční model na úrovní vět. Při testování na sadě dat *GRID* [6] dosáhli mimořádné přesnosti a překonali tím i dosavadně nejlepší práce. Dosáhli téměř třikrát menší *word error rate* oproti dosavadním nejlepším výsledkům.

Práce *Lip Reading Sentences in the Wild* [7] navrhuje *WLAS* (watch, listen, attend and spell) síť, která využívá k rozpoznání videa s případnou pomocí audia. *Watch* část obsahuje konvoluční síť a LSTM.

Listen část používá MFCC (Mel frequency cepstral coefficients) na získání charakteristik audia a LSTM. Obě části vytvářejí vektor pro *attend* část, která spolu se *spell* částí, obsahující LSTM a MLP (Multi-layer perceptron), vytváří výstup. Tento výstup je použit jako vstup pro další krok v LSTM *spell* části. *Attend* část umožnila modelu soustředit se na podstatné části vstupů a v tomto modelu zajistila výrazně lepší korelaci vstupů a výstupů. Zároveň využívá dva nezávislé *attention* mechanismy pro audio a pro video. A díky tomu nemusí být vstupy synchronizované nebo může jeden chybět. V porovnání s předchozí prací *LIPNET: END-TO-END SENTENCE-LEVEL LIPREADING* na *GRID* sadě dat dosáhli nižší chybovosti.

2.2 Vrstvená neuronová síť

Samostatná vrstvená neuronová síť není vhodná ke zpracování obrazových informací, zejména z důvodu velkého množství potřebných parametrů na obrázky s většími rozměry. Uvádíme ji zde hlavně pro úplnost, a také z důvodu, že se většinou používá společně nebo jako součást ostatních sítí.

V modelech zpracovávajících obraz slouží jako poslední vrstva redukující rozměr výstupu. Zároveň se učí mapovat vstupní charakteristiky odpovídajícím výstupům. Málokdy mívají více něž dvě vrstvy, jelikož v těchto modelech neslouží k extrahování informací z vstupu, ale pouze k mapování vstupních charakteristik na výstupy. Jednotlivé vrstvy jsou realizovány takto:

$$y = xA^T + b \tag{2.1}$$

Kde y je výstup, x jsou vstupy, A jsou učitelné váhy a b je *bias*.

2.3 Konvoluční sítě

Konvoluční sítě jsou vhodné zejména ke zpracování vizuálních informací, například k extrakci charakteristik jako jsou hrany a rohy. Hlubší sítě s více konvolučními

vrstvami jsou schopné rozpoznávat i komplexnější tvary, jako jsou třeba číslice. Konvoluční sítě jsou vhodné pro zpracování obrazových informací, protože z principu fungování berou ohled na prostorovou souvislost jednotlivých vstupních bodů, např. pixelů v obrázku, oproti vrstvené síti, která považuje všechny vstupy za nezávislé. Toto se nemusí týkat jen prostorové závislosti, ale i temporální (časové) závislosti, proto lze použít čistě konvoluční síť na extrakci charakteristik z videa. Stačí nám trojrozměrná konvoluce k získání prostorových i časových charakteristik.

Oproti vrstvené neuronové síti mají konvoluční vrstvy výhodu v tom, že mají počet volných parametrů daný podle velikosti jádra vrstvy. Tudíž pro dvourozměrné jádro o velikosti 3×3 by tato vrstva měla devět váhových parametrů a jeden *bias* parametr pro každý výstupní kanál, což odpovídá relativně malému počtu parametrů modelu, který je možné použít na rozměrná vstupní data.

Konvoluční síť se skládá z konvolučních a sdružovacích vrstev. Sdružovací vrstvy slouží k redukci rozměrů dat. Používá se sdružování průměrové nebo nejvyšší hodnoty. Nejvíce se užívá sdružování podle nejvyšší hodnoty, protože zároveň potlačuje šum.[8] Konvoluční vrstvy mohou částečně nahradit sdružovací vrstvy tím, že také mohou zmenšovat rozměry. Kromě jádra mají tyto vrstvy ještě další parametry:

Stride určuje velikost posunu jádra.

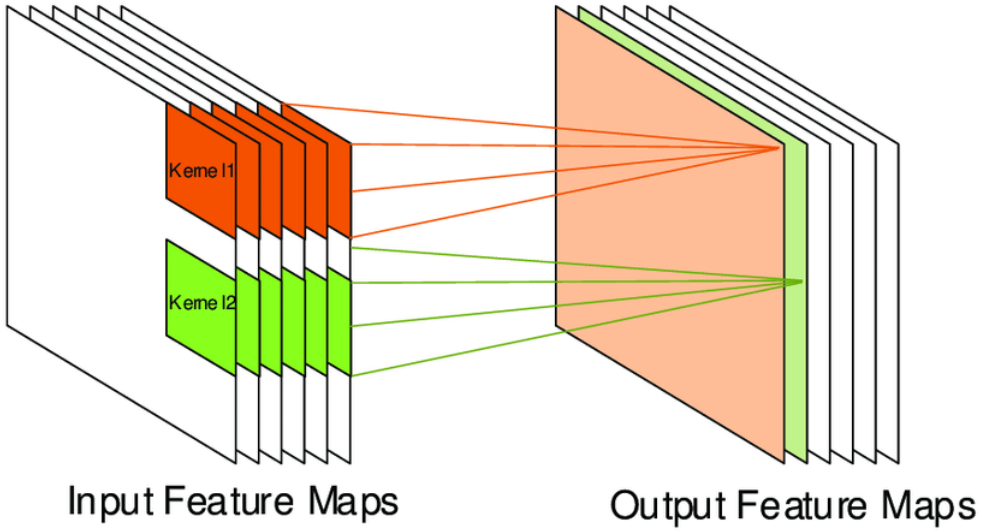
Padding určuje přítomnost, velikost a hodnoty obalu dat. Používá se k zachování rozměrů

Dilation určuje dilataci jádra.

Jádro a *stride* nemusí mít symetrický tvar, i když se nesymetrický tvar zřídka používá. Výstupní tvar se pro každý rozměr počítá následovně:

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor \tag{2.2}$$

Konvoluční vrstvy mohou být vícekanálové, ať už třeba pro použití tří kanálů RGB obrázku nebo pro zvýšení kapacity sítě. Všechny kanály mají stejný rozměr, vlastní váhy a *bias*. Výstup každého kanálu je roven součtu konvolucí ze všech vstupních vrstev a *bias*. Obrázek 2.1 vyobrazuje vztah vstupních a výstupních kanálů.



Obrázek 2.1: Vztah vstupních a výstupních kanálů konvoluční vrstvy [9]

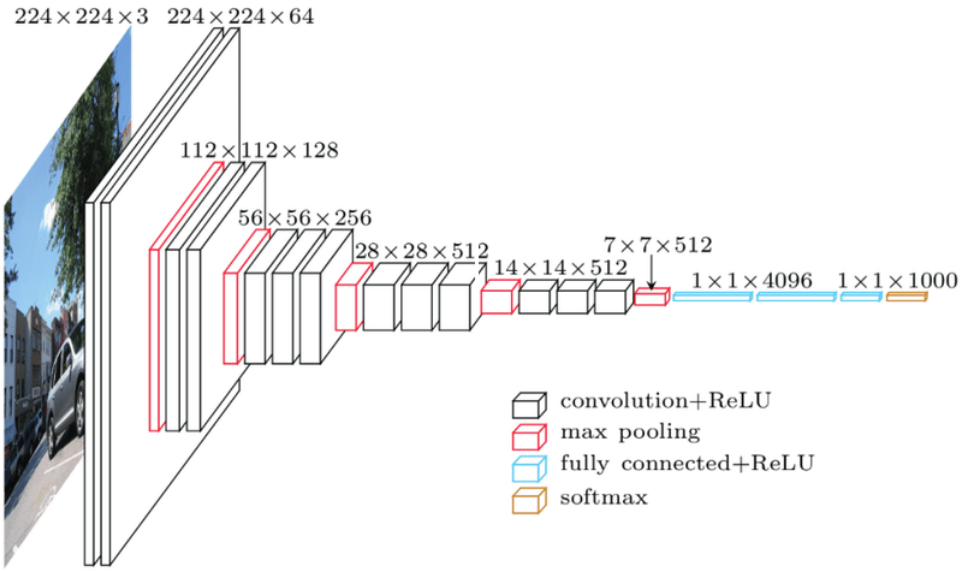
Rovnice popisující výstup konvoluční vrstvy:

$$\text{out}(C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{in}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(k) \tag{2.3}$$

C označuje kanál.

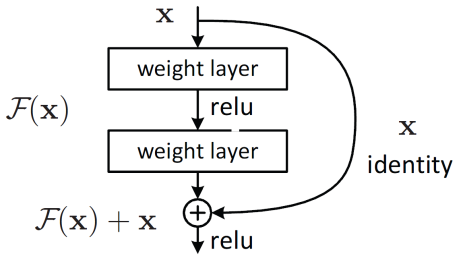
Samotné konvoluční vrstvy nevyžadují, aby vstupy byly znormalizované na jednotný rozměr, ale to, že při průchodu dat skrze ně se mění jejich rozměry. Při následném využití je potřeba znát jejich rozměr dopředu. U neznormalizovaných dat by to nebylo možné, nebo by se rozměr měnil podle vstupních dat. Proto je potřeba data před použitím v konvolučních sítích znormalizovat.

V kontextu odezíraní ze rtů je používáno několik architektur konvolučních sítí. Populární je *VGGNet*, protože se osvědčili na extrakci obrazových informací, a také je možné použít před-trénovanou síť na obličejích. Tato architektura ukázala, že je výhodnější použít větší množství menších konvolucí oproti méně větších, protože se tím sníží celkový počet parametrů. Existuje několik variant, a to od *VGG-11* do *VGG-19*, kde číslo určuje počet konvolučních vrstev. Na obrázku 2.2 je příklad modelu typu VGGNet. Při přidávání dalších vrstev dochází k nasycení, a poté se přesnost modelu rychle zhoršuje. Po dosažení potřebného počtu vrstev další přidané fungují jako identita, proto pro zlepšení nelze pouze přidávat další vrstvy. Dalším faktorem je, že při zvyšování počtu vrstev je obtížnější síť trénovat.



Obrázek 2.2: *Vrstvy VGG-19 architektury* [10]

Další užívaná architektura je *ResNet* používající vrstvy učící se residuální funkce. Tím se zamezí problému, aby se nějaká vrstva naučila být jen identitou. Zároveň v práci [11] autoři empiricky dokázali, že je snadnější ji optimalizovat, je možné vytvořit model s mnohem více vrstvami a dosáhnout zlepšení. Používají se verze od 34 až po 152 vrstev. Na obrázku 2.3 je základní blok sítě typu *ResNet*.



Obrázek 2.3: Základní blok ResNet-34 [11]

Pro porovnání, *VGG-16* má 138 milionů parametrů a *ResNet-34* má 21 milionů parametrů a zároveň dosahuje lepších výsledků.[12] Existují i další architektury, ale ty nejsou zde uvedeny. Některé práce nevyužívají žádnou z těchto architektur a na místo toho používají vlastní jednoduchou síť tvořenou kombinací konvolučních a sdružovacích vrstev, kterou používají jako součást modelu. Například *LipNet* používá tři konvoluční vrstvy se sdružovací vrstvou za každou konvoluční.

2.4 Rekurentní sítě

Rekurentní sítě jsou vhodné ke zpracování sekvenčních informací, kvůli jejich schopnosti uchovat informaci o dřívějších vstupech, a oproti ostatním sítím nepotřebují, aby vstupní data měla danou stejnou délku, jako jsou v našem případě různě dlouhé záběry obličejů.

Je možné využít vícevrstvé rekurentní sítě, pokud je síť vícevrstvá, tak výstup z jedné vrstvy je přiveden na vstup vrstvy následující. Vícevrstvé sítě mají větší kapacitu, tudíž schopnost se naučit víc. Také je možné použít obousměrnou síť. Díky tomu je síť schopná reagovat jak na minulost, tak na budoucnost, ve smyslu kontextu. Tedy pro každý vstupní časový okamžik bere v potaz všechny předchozí a budoucí vstupy a čím blíže jsou našemu časovému okamžiku, tím větší mají vliv, protože síť postupně „zapomíná“.

Při aplikaci rekurentních sítí na úlohy strojového překladu, rozpoznání a syntézy řeči, je možné použít rekurentní sítě samostatně nebo s vrstvenou neuronovou sítí. Pro aplikaci jako je popis obrázku, se k získání charakteristik používají konvoluční sítě. Tyto charakteristiky se použijí jako vstupy do rekurentní sítě, která určí, co se v obrázku vyskytuje. Existuje zde možnost k tomuto přidat další rekurentní síť, která vygeneruje souvislý text popisující obrázek namísto prostého výpisu vyskytujících se objektů.

Zde jsou uvedeny příklady použitých konfigurací rekurentních sítí ve vybraných pracích. V modelu *LipNet* [5] , po získání příznaků konvoluční sítí autoři používali dvě obousměrné GRU vrstvy za sebou. V práci *Lip Reading Sentences in the Wild* [7] v části *watch* a *listen* použili tři jednosměrné LSTM za sebou. V části *spell* použili tři jednosměrné LSTM za sebou, kde jako počáteční *cell* stav použili spojený stav posledních LSTM z *watch* a *listen*, také jako vstup do používali výsledek z předchozího časového úseku.

2.4.1 Elman recurrent neural network

Jednoduchá rekurentní sít skládající se ze čtyř vrstev, vstupní, kontextové, skryté a výstupní. V knihovně Pytorch je implementována následovně.

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}) \tag{2.4}$$

Symbol h je skrytá vrstva, x je vstup, W jsou váhy a b je *bias*. Výstup je roven skryté vrstvě po průchodu celé sekvence. Při zpětné propagaci gradientu během trénování nastává problém mizejícího gradientu, čím hlouběji se dostaneme v síti, tím více se zpomaluje trénování.

2.4.2 Long short-term memory

LSTM je architektura navržená k eliminování problému s mizejícím gradientem. Skládá se z pěti prvků. *Cell* představuje aktuální stav, *input gate*, *forget gate*, *cell gate* a *output gate* slouží k regulaci informací v a z *cell*.

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned} \tag{2.5}$$

Kde x je vstup, W je daná váha, b je *bias*, i je *input gate*,f je *forget gate*, g je *cell gate*, o je *output gate*, c je *cell*, h je skrytá vrstva a zároveň výstup v daném okamžiku sekvence. Funkce je vyobrazena na obrázku 2.4 pro lepší názornost.

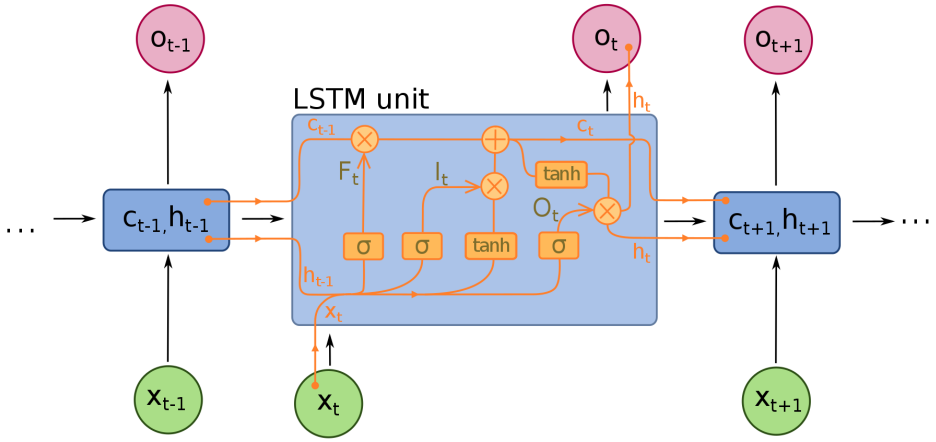
2.4.3 Gated Recurrent unit

GRU je zjednodušená verze LSTM, které chybí *output gate*. Má méně parametrů něž LSTM, proto je obecně horší. I když v některých případech užití dosahuje lepších

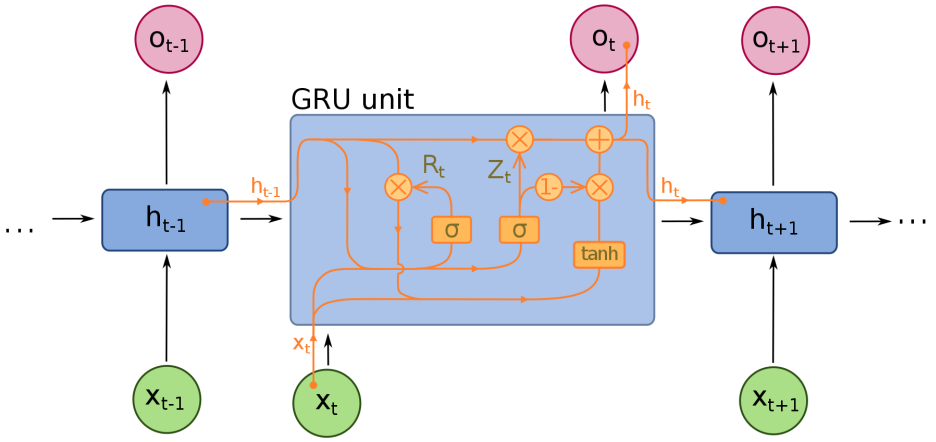
výsledků než LSTM. Používá se i pro odezírání ze rtů, i když LSTM je častěji užívané. Součástí GRU je *hidden state*, *reset gate*, *update gate* a *new gate*. Podle *new gate* a *update gate* se určuje nová informace v *hidden state*. [14]

$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn})) \\
h_t &= (1 - z_t) * n_t + z_t * h_{(t-1)}
\end{aligned}
\tag{2.6}
$$

Ve vzorci je x vstup, W je daná váha, b je *bias*, r je *reset gate*,z je *update gate*, n je *new gate*, h je skrytá vrstva. Na obrázku 2.5 je vyobrazena vnitřní struktura.



Obrázek 2.4: Vnitřní struktura LSTM [13]



Obrázek 2.5: Vnitřní struktura GRU [14]

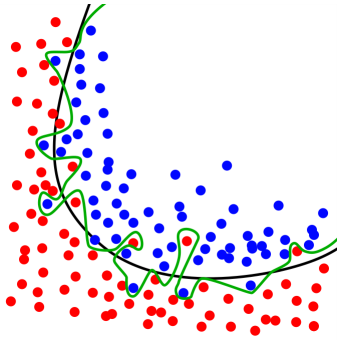
3 Trénování neuronových sítí v úloze odezíraní ze rtů

3.1 Dropout

Dropout je metoda, která napodobuje trénování velkého množství neuronových sítí s odlišnými strukturami paralelně. Během trénování jsou některé výstupy dané vrstvy náhodně vynechány. V průběhu trénování se tyto vrstvy jeví jako vrstvy s různým počtem uzlů a spojení s předchozí vrstvou. V každé iteraci může být vstup náhodně vynechán, tím je omezeno upřednostňování jakéhokoli prvku ve vrstvě. To má za důsledek, že matice vah je rovnoměrněji rozložena. *Dropout* je používán pouze při trénování. Při použití *dropout* jsou váhy sítě větší podle použité pravděpodobnosti, proto se před použitím sítě musí váhy škálovat zpět do normální velikosti. [15]

3.2 Overfitting

Overfitting modelu znamená, že se model příliš blízce přizpůsobil dané sadě dat do takové míry, že ztrácí schopnost generalizovat. Model se namísto generalizování naučí vystihnout šum v datech nebo dokonce si „zapamatovat“ konkrétní hodnoty z trénovací sady dat. K tomu může docházet v případě, že je model zbytečně komplexní a je trénován příliš dlouho. Na obrázku 3.1 je znázorněn princip, černá je dobrý model, zelená je model, kde došlo k *overfittingu*. [16]



Obrázek 3.1: Ukázka *Overfitting* [16]

3.2.1 Early stopping

Jednoduše určit, jak dlouho je potřeba neuronovou síť trénovat, nelze. Při nedostatečném trénování nedosáhneme maximálního potenciálu, a pokud bychom trénovali příliš dlouho, tak dojde k *overfittingu*. *Early stopping* je metoda sloužící k zastavení trénování v okamžiku, kdy se síť začne zhoršovat. Abychom poznali, že se síť začíná zhoršovat, je třeba měřit její průběžnou přesnost. K tomu lze použít testování sítě na validační sadě dat na konci každé epochy. Testování po každé epoše zvyšuje výpočetní náklady, proto je možné provádět měření méně často. Dále je potřeba zvolit podmínku pro zastavení trénování. Není vhodné zastavit trénování při první známce zhoršení, protože trénování je stochastické a může obsahovat šum. Je vhodnější čekat například na stagnaci nebo dlouhodobější zhoršení. Po ukončení trénování je třeba vybrat, který model budeme považovat za výsledek trénování. Nejlepším řešením je si vždy uchovávat model s nejlepšími výsledky. [17]

3.2.2 Batch normalization

Při trénování neuronových sítí se při změně vnitřních parametrů mění rozdělení výstupů do následující vrstvy. Což znamená, že následující vrstva se musí průběžně optimalizovat vzhledem k novým rozdělením. Tento jev se nazývá *internal covariate shift*. Tento jev způsobuje zpomalení trénování, kvůli nutnosti použití menšího kroku pro úpravu parametrů, a také je model citlivější na inicializaci parametrů.

K odstranění tohoto problému slouží *batch normalization*, který toho dosáhne změnou měřítka výstupní vrstvy tak, aby výstup měl průměr nula a směrodatnou odchylku jedna, čímž výrazně urychlí trénování a zároveň zvýší jeho stabilitu. Může mít navíc regulační efekt, který zamezuje *overfitting*. [18, 19]

3.3 Učení s učitelem

Pro samotné trénovaní úpravou parametrů sítě se využívá zpětné propagace chyby. Analyticky zjistíme gradient kriteriální funkce jednotlivých vrstev s ohledem na jednotlivé parametry. Následně upravíme tyto parametry podle použitého optimalizačního algoritmu.
Uvedeme vybrané kriteriální funkce, které nám určují chybu jednotlivých výstupů modelu během trénování. Anglicky *Loss function* je funkce, kterou se během trénování snažíme minimalizovat pomocí změny parametrů modelu.

3.3.1 Cross-entropy loss

Pro úlohu výběru jedné z několika možností je vhodné použít logaritmickou ztrátovou funkci jako je *Cross-entropy loss*. Tato funkce se používá při trénování rozpoznávat jednu z N tříd. V knihovně Pytorch je implementována následovně:

$$\text{loss(x, class)} = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) \tag{3.1}$$

Kde x je výstup modelu, *class* je skutečná třída vstupních dat.

3.3.2 Connectionist temporal classification loss

Pro určení chyby mezi sekvencemi lze použít *connectionist temporal classification loss*, zkráceně *CTC loss*. Dokáže určit chybu výstupní sekvence sítě oproti skutečné sekvenci. Vstup funkce očekává skutečnou sekvenci a vstupní sekvenci, která může obsahovat stejně nebo více prvků než skutečná. Vstupní sekvence může obsahovat navíc znak *blank* označující prázdno nebo mezeru, čímž se označují mezery mezi slovy nebo pokud se ve slově opakuje více znaků za sebou. Vstupem je pravděpodobnostní rozdělení jednotlivých znaků pro každý časový okamžik. Funkce vypočte pravděpodobnost každé možné sekvence znaků z dodaného rozdělení. Duplikátní znaky za sebou se sjednotí, pokud nejsou odděleny znakem *blank*, a pravděpodobnosti totožných sekvencí po zkrácení se sečtou. Výstupní chyba je záporný logaritmus pravděpodobnosti odpovídající skutečné sekvenci. Výhoda využití této funkce je v tom, že nemusíme mít v datech zaznamenané odpovídající znaky pro každý časový okamžik. [20]

3.4 Křížová validace

Křížová validace je metoda zjišťující schopnost generalizace statistické analýzy na nezávislých datech. Cílem je odhalit problémy jako je *overfitting* a výběrové zkreslení. Během křížové validace se dostupná data rozdělí na trénovací sadu a na validační sadu. Model se natrénuje pouze na trénovací sadě dat a otestuje se přesnost na validační sadě. K určení výsledku křížové validace se zprůměruje přesnost několika různých rozdělení dat. Následují vybrané typy křížové validace. *Leave-p-out* křížová validace použije p vzorků k validaci a zbytek dat k trénování, to se opakuje na všech možných kombinacích. Křížová validace *k-fold* dělí data na k částí, kde jedna část je použita na validaci a zbytek na trénování, to se opakuje pro všechny kombinace. [21]

4 Použitá sada dat

V této kapitole popisuji data set, který jsem použil v experimentální části této práce. Využívám data set *OuluVS* a *OuluVS2* z Finské univerzity *University of Oulu*. *OuluVS* obsahuje audiovizuální záznam dvaceti osob vyslovující deset anglických frází. *OuluVS2* obsahuje audiovizuální záznam 52 osob vyslovujících deset sekvencí číslic, deset anglických frází a deset náhodně vybraných vět ze sady dat *TIMIT* z několika úhlů pohledu. [22, 23] Já jsem využíval pouze video záznamy frází a sekvencí čísel z přímého pohledu. Tabulka 4.1 ukazuje přehled veškerých použitých dat.

	OuluVS	OuluVS2
řečníci	20	52
fráze	10	10
sekvence	-	10
opakování	5	3
promluvy	1000	3120

Tabulka 4.1: Přehled dat

V tabulce 4.2 jsou vypsány fráze a sekvence čísel v sadách dat. A na obrázku 4.1 je příklad snímků z obou datasetů.

Fráze	Sekvence
Excuse me	1 7 3 5 1 6 2 6 6 7
Goodbye	4 0 2 9 1 8 5 9 0 4
Hello	1 9 0 7 8 8 0 3 2 8
How are you	4 9 1 2 1 1 8 5 5 1
Nice to meet you	8 6 3 5 4 0 2 1 1 2
See you	2 3 9 0 0 1 6 7 6 4
I am sorry	5 2 7 1 6 1 3 6 7 0
Thank you	9 7 4 4 4 3 5 5 8 7
Have a good time	6 3 8 5 3 9 8 5 6 5
You are welcome	7 3 2 4 0 1 9 9 5 0

Tabulka 4.2: Přehled dat

(a) OuluVS

(b) OuluVS2

Obrázek 4.1: Ukázka snímků z datasetu

4.1 Práce s daty

K této práci jsem obdržel již částečně předzpracovaná data z data setu *OuluVS*. Z původního formátu 720×576 oříznuté na rty a změněné na velikost 64×64. Pro jednoduchost jsem snímky převedl do černobílého formátu a sjednotil počet snímků. Nahrazoval jsem snímky nejbližším snímkem co se týče pozice ve videu z průměrných zhruba 29 na 32. Takováto normalizace je nutná pro konvoluční sítě a zároveň nám zajistí částečnou nezávislost na rychlosti promluvy jednotlivých řečníků. Data set *OuluVS2* obsahuje pět pohledů na každou nahrávku, od čelního pohledu až po profil. V této práci jsem využíval pouze nahrávky z čelního pohledu. Video záznamy byly tvůrci oříznuty na oblast rtů z původního 1920×1080 formátu. Stejně jako u *OuluVS* jsem sjednotil počet snímků, tentokrát na 20, což zhruba odpovídá průměru. A navíc jsem sjednotil rozměr snímků na velikost 128×64, to zhruba odpovídá průměrnému poměru stran snímků. Takovéto sjednocení je nutné kvůli konvolučním sítím, ale jelikož jsou záběry z originálního formátu oříznuty na potřebné minimum, znamená to, že snímky s menším pohybem úst budou mít jiný poměr stran než snímky s velkým pohybem úst. Tudíž tímto znormalizováním dojde ke zkreslení některých snímků více a některých méně. Pro modely využívající rekurentní sítě není třeba sjednocovat délku, proto jsem pro ně data pouze převedl do černobílého formátu a v případě *OuluVS2* sjednotil rozměry.

4.1.1 Rozšíření dat

Konvoluční sítě nejsou z principu invariantní na škálu. Větší počet kanálů jim ale umožňuje naučit se rozpoznávat obrazy s různými měřítky. Stejně tak nejsou invariantní vůči rotaci a dalším změnám, ale dokáží se je naučit. A jelikož je každý řečník alespoň nějak odlišný, je důležité síť naučit dobře generalizovat, na což nemusí sady dat v řádech tisíců videí stačit. Proto může být výhodné použít nějakou formu rozšíření dat ze stávajících dostupných dat. Anglicky *data augmentation* slouží k rozšíření obrazových trénovacích dat, a tím k možnému lepšímu natrénování neuronových sítí.

Výhody jsou nejvíce znatelné, pokud nemáme k dispozici velké množství dat. Pokud známe možné vstupy, dostupná data lze pomocí vybraných funkcí rozšířit, tak aby se podobala všem možným vstupům. Tím dosáhneme natrénování robustní sítě. Existují dvě základní metody aplikace rozšíření dat. *Offline augmentation* při které data upravíme a následně uložíme. To je vhodné u menšího množství dat, protože ušetříme výpočetní výkon při samotném trénování a nepotřebujeme velké úložiště. *Online augmentation* kde data upravujeme během trénování, většinou po dávkách těsně před jejich použitím při trénování. Mezi základní metody patří translace, převrácení, rotace, oříznutí a přidání šumu.

V sadě *OuluVS* je přibližně tisíc videí, což není příliš mnoho pro trénování sítě. Proto jsem použil náhodné horizontální převrácení, náhodnou rotaci, náhodnou perspektivu a afinní transformaci *shear mapping*, abych dosáhl co nejrobustnějšího natrénování. Jelikož používám trojrozměrnou konvoluci v čistě konvoluční síti, bylo potřeba zajistit identickou úpravu pro všechny snímky v jednom videu. Použil jsem proto k tomuto náhodnému generování vždy stejný *seed* pro všechny snímky v jednom videu. Pro takto relativně malou sadu dat je vhodné použít *offline augnentation*. Oproti tomu *online augnentaion* nám umožňuje snadněji měnit použité funkce rozšíření, proto jsem ji použil pro naše experimenty s rozšířením dat. Na Obrázku 4.2 je ukázka použitého rozšíření dat. [24]

(a) Originální snímek (b) Upravený snímek

Obrázek 4.2: Ukázka rozšíření dat

4.2 Rozdělení dat

K určení úspěšnosti trénování je třeba data rozdělit. Obecně se data rozdělují na tři části.

Trénovací část se používá na trénování sítě.

Validační část se užívá v průběhu trénování k vyhodnocení přesnost a podle úspěšnosti na ní se volí výsledný model.

Testovací část slouží k vyhodnocení úspěšnosti trénování.

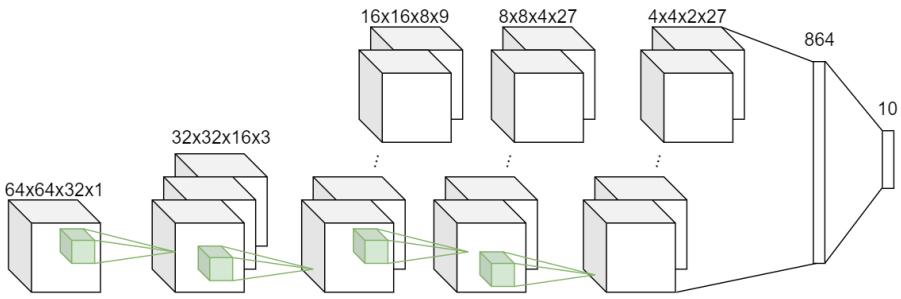
Model se učí pouze z trénovacích dat, proto validační a testovací data považujeme za neznámá data.
Pro vyhodnocení úspěšnosti modelu, jsem použil následující rozdělení dat. U data setu *OuluVS* jsem rozdělil data na trénovací a validační v poměru 19:1. Z důvodu malého celkového počtu dat v datasetu *OuluVs* jsem se rozhodl použít validační data zároveň i jako testovací data. Tím dochází ke zkreslení výsledků, protože za výsledný model je považován ten, který dosahuje nejlepších výsledků na validačních datech. Je možné, že by výsledky byly horší, kdyby byla použita separátní část dat jako testovací. Pro data set *OuluVS2* jsem rozdělil data na trénovací, validační a testovací v poměru 40:10:2.

5 Experimenty

Experimenty jsem prováděl převážně na datasetu *OuluVS*. Začal jsem s vývojem konvoluční sítě, protože se to zdálo být nejsnadnějším způsobem, a zároveň je součástí ostatních architektur. Takže když se podaří vytvořit vhodnou konvoluční síť, budu ji moci použít i v ostatních. Proto jsem nejdřív experimentoval s čistě konvoluční sítí k rozpoznávání frází, poté rekurentní sítí a nakonec jsem zkoumal využití této architektury na rozpoznávání sekvence čísel z datasetu *OuluVS2*.
Pro úlohu rozpoznávání frází jsem používal *cross-entropy loss* jako kriteriální funkci, pro sekvenci čísel jsem použil *CTC loss*. Jako aktivační funkce jednotlivých vrstev jsem použil *PReLU*. Původně jsem chtěl využívat *ReLu*, ale s ní se mi nedařilo úspěšně natrénovat síť.

5.1 Konvoluční model

První funkční verze konvolučního modelu je vyobrazena na obrázku 5.1, vyobrazené velikosti jsou ve formátu šířka × výška × počet snímků × počet kanálů. Síť obsahovala čtyři 3×3×3 konvoluční vrstvy a jednu lineární vrstvu. Navíc za každou konvoluční vrstvou následovala *batch normalization* vrstva, *PReLU* aktivační funkce a *maxpool* vrstva. Pro všechny tyto a následující konvoluční vrstvy jsem používal *padding* takový, aby byl zachován rozměr.
V několika prvních verzích sítě jsem experimentoval s parametrem *learn rate*, který udává optimalizačnímu algoritmu velikost kroku úpravy parametrů. Porovnával jsem vliv změn *learn ratu* společně s optimalizátorem *SGD*.



Obrázek 5.1: Architektura konvolučního modelu

Nejmenší chyby kriteriální funkce jsem dosáhl postupným zmenšováním *learn ratu* každé dvě epochy. Následně jsem však začal využívat optimalizační algoritmus *Adam*, který si z principu jeho fungování upravuje *learn rate* pro jednotlivé parametry sám, ale lze ho snižováním omezit. Vyzkoušel jsem jeho omezování, nicméně *Adam* dosáhl nejlepších výsledků s použitím výchozích parametrů. Ve všech dalších experimentech a výsledcích uvažujeme použití optimalizačního algoritmu *Adam* s výchozími parametry: learn rate $= 0.001$, $\beta 1 = 0.9$, $\beta 2 = 0.999$, $\epsilon = 1\mathrm{E} - 08$.

Potom jsem experimentoval s velikostí, počtem konvolučních vrstev a počtem kanálů. Snažil jsem se zvětšit *receptive field*, neboli oblast kterou zahrnují výstupní příznaky. Rozšířil jsem model o další konvoluční vrstvu, bez přidání *maxpoolu*, abych nesnižoval velikost výsledného počtu extrahovaných charakteristik. Zkoušel jsem použít konvoluční jádra o velikosti $7 \times 7 \times 7$ a $5 \times 5 \times 5$ v několika místech, také jsem zkoušel využít celkem osm konvolučních vrstev velikosti $3 \times 3 \times 3$ s *maxpoolem* po každých dvou vrstvách společně s většími konvolučními jádry, z čehož jsem dosahoval nejlepších výsledků při použití modelu s pěti konvolučními vrstvami a velikostí jádra $5 \times 5 \times 5$ v 3. a 4. vrstvě. V tabulce 5.1 je zkrácený přehled experimentů s vrstvami. Co se týče použití pro dataset *OuluVS2*, bylo potřeba síť trochu upravit. Jelikož data měla podstatně menší rozměr co se počtu snímků týče, tak jsem se rozhodl použít nesymetrický *maxpool* ve tvaru $2 \times 2 \times 1$ za čtvrtou konvoluční vrstvou, to znamená nezmenšovat počet snímků. Zároveň mají data dvakrát větší šířku, tak je rozměr za poslední sdružovací vrstvou větší, což znamená, že síť pro *OuluVS2* má více parametrů v lineární vrstvě. Takže konvoluční síť má celkem 529 tisíc parametrů, pro *OuluVS2* má 550 tisíc parametrů.

model	změněné vrstvy	jádro	přesnost [%]
s 5 konv. vrstvami	-		76
	3,4	5	**80**
	2,3,4,5	5	74
	3,4	7	76
	3,4,5	7	64
s 8 konv. vrstvami	-		78
	3,5	5	72
	4,6	5	78
	4,6,8	5	76

Tabulka 5.1: Experimenty s vrstvami

Na obrázku 5.2a je finální architektura konvoluční sítě s rozměry dat uvnitř modelu pro dataset *OuluVS* a na obrázku 5.2b pro *OuluVS2* a v tabulce 5.2 jsou popsány jednotlivé vrstvy.

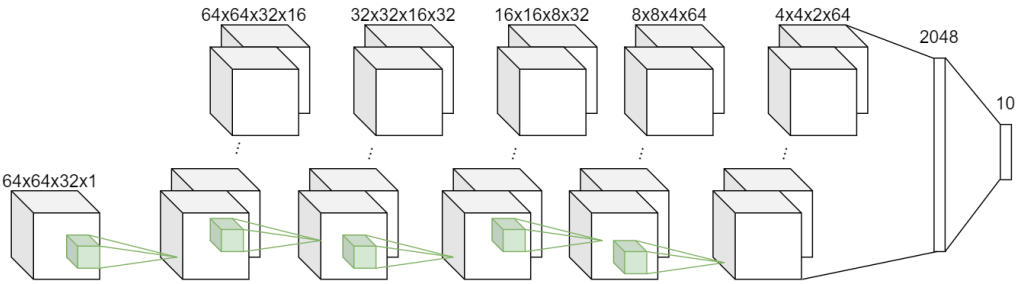
Vrsta	velikost	stride	padding
conv1	3x3x3	1x1x1	1
conv2	3x3x3	1x1x1	1
pool1	2x2x2	2x2x2	-
conv3	5x5x5	1x1x1	2
pool2	2x2x2	2x2x2	-
conv4	5x5x5	1x1x1	2
pool3	2x2x2	2x2x2	-
conv5	3x3x3	1x1x1	1
pool4	2x2x2	2x2x2	-

(a) *OuluVS*

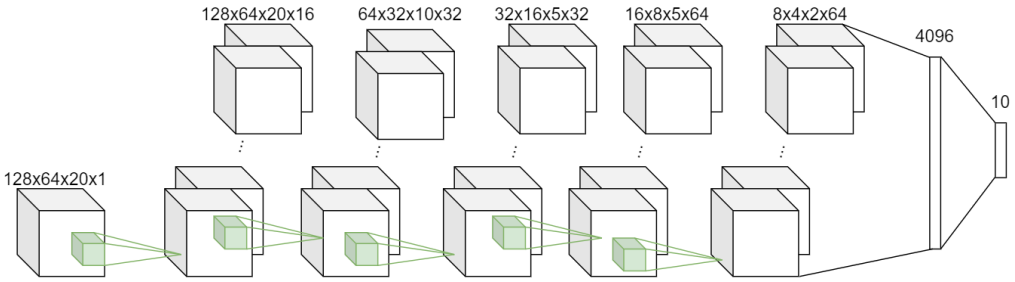
Vrsta	velikost	stride	padding
conv1	3x3x3	1x1x1	1
conv2	3x3x3	1x1x1	1
pool1	2x2x2	2x2x2	-
conv3	5x5x5	1x1x1	2
pool2	2x2x2	2x2x2	-
conv4	5x5x5	1x1x1	2
pool3	2x2x1	2x2x1	-
conv5	3x3x3	1x1x1	1
pool4	2x2x2	2x2x2	-

(b) *OuluVS2*

Tabulka 5.2: parametry konvolučních vrstev modelu



(a) *OuluVS*



(b) *OuluVS2*

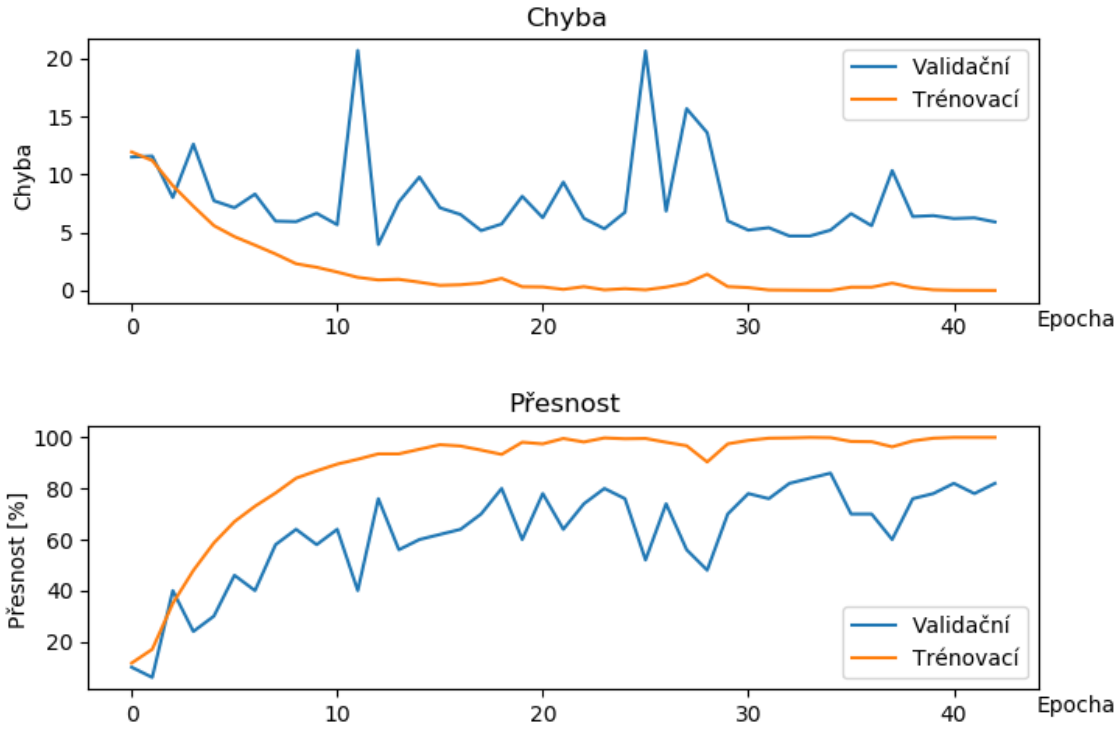
Obrázek 5.2: Architektura finálního konvolučního modelu

V dalším kroku jsem se zabýval využitím rozšíření dat pro zlepšení přesnosti sítě. Ze všech možných způsobů rozšíření dat jsem se rozhodl vyzkoušet použít zrcadlení, rotaci, *shear mapping* a změnu perspektivy. Z toho se mi nejlépe osvědčilo použití náhodného horizontálního odzrcadlení, náhodnou rotaci o ±10° a náhodný *shear mapping* s *shear* úhlem ±10°. Toto mi pomohlo zlepšit přesnost zhruba o několik jednotek procent. V tabulce 5.3 je zkrácený přehled experimentů s rozšířením dat.

	přesnost [%]
základ	80
shear ±10°, rot. ±10°	**84**
shear ±20°, rot. ±20°	76
shear ±20°, rot. ±10°	70
shear ±15°, rot. ±15°	80

Tabulka 5.3: Experimenty s rozšířením dat

Doposud jsem nechal síť trénovat pevný počet epoch, abych omezil zbytečné trénování nebo *overfitting*, implementoval jsem následujícím způsobem *early stopping*. Pomocí programu se vždy ukládala nejlepší síť. Pokud bylo dosaženo 100 % přesnosti na trénovacích datech alespoň tři epochy za sebou a síť se za posledních pět epoch nezlepšila, bylo ukončeno trénování. Trénování trvalo přibližně 60 sekund na epochu pro dataset *OuluVS* a 80 pro *OuluVS2*, takže jsem tímto vylepšením výrazně urychlil trénování.

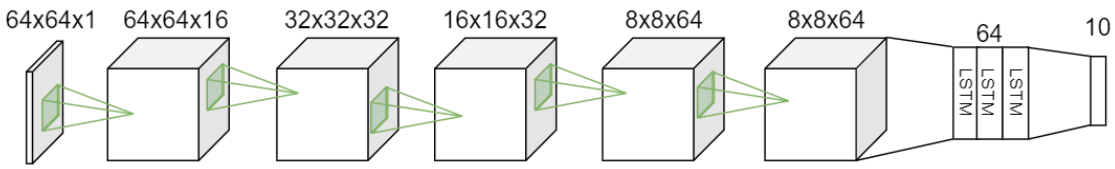


Obrázek 5.3: Ukázka průběhu trénování sítě

Na obrázku 5.3 je vyobrazen průběh trénování této architektury. Z grafu je možné vidět, že nejnižší chyba nemusí nutně odpovídat nejlepší přesnosti. Model měl nejnižší chybu v 13. a nejlepší přesnost v 34. epoše. Trénování je poměrně rychlé, model dosáhne podmínky zastavení výrazně dříve, než dosáhne námi nastavený limit šedesáti epoch.

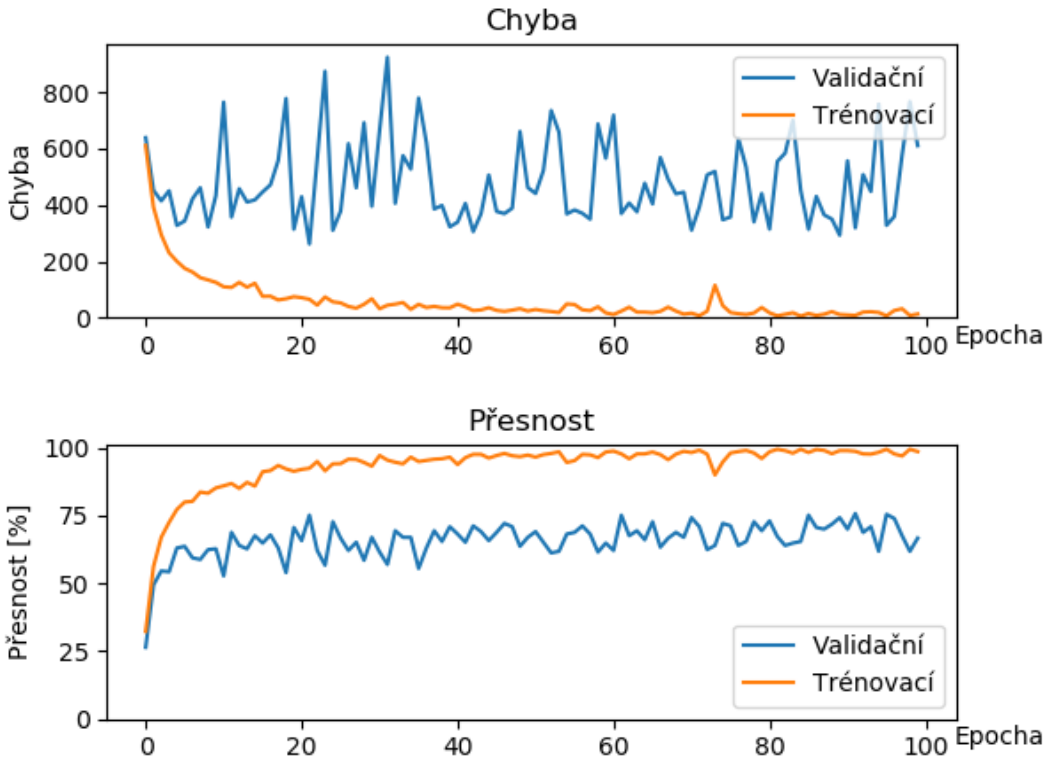
5.2 Rekurentní model

V této části popisuji svou implementaci rekurentního modelu k rozpoznávání frází. Rozhodl jsem se použít architekturu s dvourozměrnou konvoluční sítí na extrakci charakteristik z jednotlivých snímků, rekurentní LSTM vrstvou a lineární vrstvou. Soustředil jsem se především na nastavení parametrů LSTM vrstvy. Lineární vrstva dostává jako vstup výstup z LSTM pro poslední časový okamžik. Jako konvoluční síť jsem použil dvourozměrnou variantu sítě použité v konvolučním modelu. Jen namísto trojrozměrných konvolucí, sdružování a *batch normalizace* používám jejich dvourozměrné varianty. Model se trénoval výrazně pomaleji oproti konvolučnímu modelu co se týče počtu epoch, proto jsem se snažil urychlit trénování zvýšením *learn ratu*, čehož se mi nepodařilo dosáhnout. Musel jsem tedy zvýšit limit epoch pro trénování. Nejprve jsem testoval použití *dropoutu* před LSTM vrstou nebo před lineární vrstvu, jenže *Dropout* zpomaluje trénování a v mých experimentech nedosáhl zlepšení modelu, z tohoto důvodu jsem jej přestal používat. Následně jsem experimentoval s použitím jednosměrných nebo obousměrných vrstev, velikostí skrytých vrstev a počtem vrstev. Nejlepších výsledků jsem dosáhl s jednosměrnou LSTM se třemi vrstvami a velikostí skrytých vrstev 64. Na obrázku 5.4 je vyobrazena konečná architektura.



Obrázek 5.4: Architektura rekurentního modelu

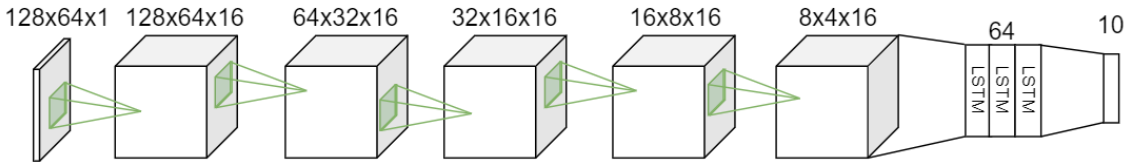
Na grafu 5.5 je vyobrazen průběh trénování rekurentního modelu. Z grafu je vidět, že trénovací chyba je výrazně vyšší než u konvolučního modelu, a také se pomaleji snižuje. Validační chyba je mnohem nestabilnější oproti konvolučnímu modelu.



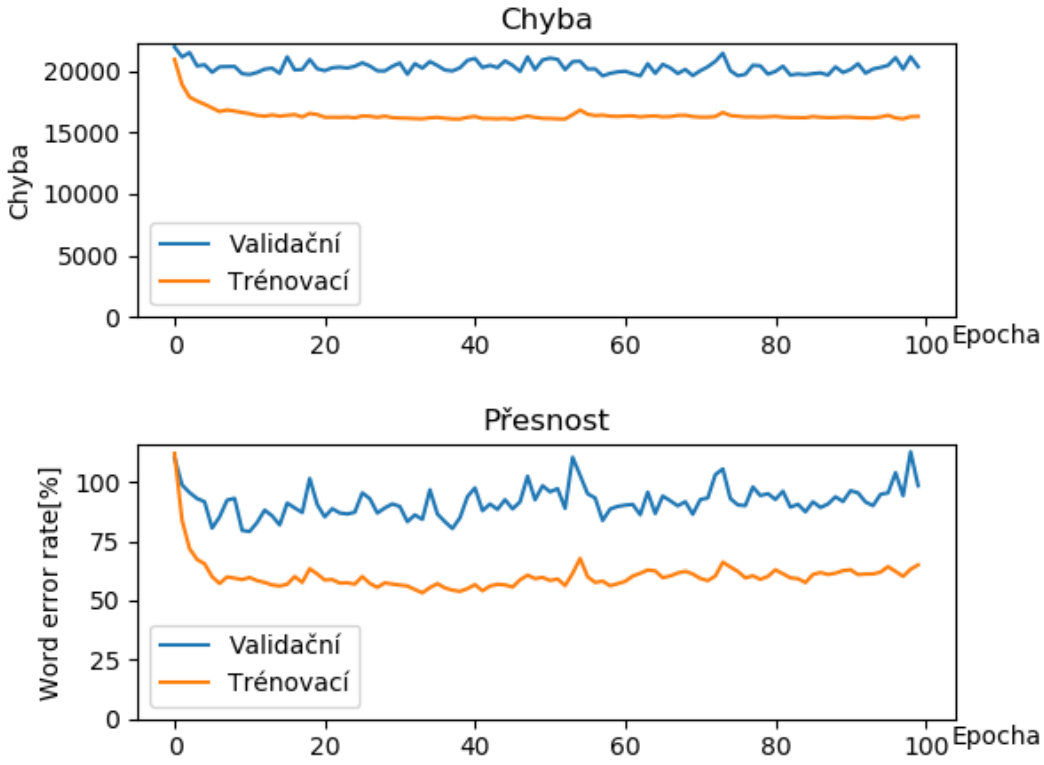
Obrázek 5.5: Ukázka trénování rekurentního modelu

5.3 Rekurentní model pro rozpoznávání sekvencí

Pro rozpoznávání jsem použil finální verzi rekurentní sítě. Jako kriteriální funkci jsem použil *CTC loss* a jako vstup do poslední lineární vrstvy uvažujeme výstupy za každý časový okamžik. Konečný výstup sítě je pravděpodobnostní rozdělení číslic v každém časovém okamžiku. Výsledná sekvence pro vyhodnocení přesnosti byla zkracována vypuštěním za sebou opakujících se čísel a vypuštěním znaku prázdna. Na obrázku 5.6 je výsledná architektura modelu a na grafu 5.7 je vidět příklad trénování modelu.



Obrázek 5.6: Architektura rekurentního modelu pro sekvence



Obrázek 5.7: Ukázka trénování rekurentního modelu pro sekvence

5.4 Hardwarové limitace

Knihovna *Pytorch* dovoluje použití softwarové a hardwarové architektury *CUDA* pro paralelizaci trénování a užívání neuronových sítí pomocí GPU, této skutečnosti jsem se snažil co nejvíce využít. Jelikož jsem měl k dispozici počítač s omezenou GPU pamětí, musel jsem v průběhu vývoje postupně zmenšovat počet dat pro paralelní zpracování.

S komplexnějším modelem jsem snižoval *batch size* z původního 100 na 5 k použití s posledním konvolučním modelem pro sadu dat *OuluVS2*. Pro rekurentní sítě se mi nepodařilo implementovat paralelní zpracování, i když knihovna nabízí metody pro to určené, avšak mě se s nimi nepodařilo síť úspěšně natrénovat. Také jsem s cílem urychlení trénování načítal veškerá trénovací data do paměti, namísto opakovaného načítání z disku, nicméně nám pro sekvence číslic z data setu *OuluVS2* nestačila operační paměť. Pro tuto úlohu jsem opět začal načítat data těsně před jejich zpracováním. Také jsem musel výrazně zjednodušit rekurentní model pro sekvenci, kvůli nedostatku GPU paměti. Zmenšil jsem tedy počet kanálů konvolucí v modelu.

5.5 Výsledky

Přesnost výsledků je pro modely k rozpoznávaní frází prostý poměr shody výstupu ze sítě s pravdou. Pro model rozpoznávání sekvence používám *WER* (word error rate) výsledné sekvence vůči skutečné. K určení konečného výsledku jsem použil křížovou validaci. Pro data z datasetu *OuluVS* jsem použil *leave-one-out* metodu. Trénink probíhal na 19 osobách a na jedné probíhalo testování. Pro *OuluVS2* jsem použil *5-fold* metodu podle rozdělení uvedené v kapitole 4.2, kde součástí testovací části byli vždy poslední dva řečníci. V tabulce 5.4 jsou výsledky jednotlivých podmnožin křížového rozdělení pro dataset *OuluVS*.

podmnožina	1	2	3	4	5	6	7	8	9	10
přesnost CNN [%]	80	96	94	64	74	98	73,47	92	100	80
přesnost RNN [%]	84	86	56	66	92	70	87,76	100	100	64
podmnožina	11	12	13	14	15	16	17	18	19	20
přesnost CNN [%]	82	94	82	90	82	89,58	80	84	86	96
přesnost RNN [%]	50	98	60	90	84	91,67	76	82	82	62

Tabulka 5.4: Výsledky na datasetu OuluVS

Na *OuluVS* jsem s CNN modelem dosáhl celkové přesnosti 85,85 % a s RNN modelem 79,07 %. CNN model se také mnohem rychleji trénoval, průměrně trvalo 30 epoch k dosažení nejlepšího výsledku a 36 epoch k dosažení podmínky ukončení trénování. RNN model potřeboval průměrně 56 epoch k dosažení nejlepšího výsledku a 92 epoch k dosažení podmínky ukončení trénování s tím, že u většiny došlo k dosažení limitu 100 epoch. V tabulce 5.5 jsou výsledky pro dataset *OuluVS2*.

	přesnost CNN [%]		přesnost RNN [%]	
podmnožina	test.	val.	test.	val.
1	90	88,18	75	76,36
2	85	87,88	86,67	75,76
3	88,33	87	65	66,67
4	85	90,91	60	72,73
5	81,67	82,12	78,33	70
průměr	86	87,22	73	72

Tabulka 5.5: Výsledky na datasetu OuluVS2

Na *OuluVS2* jsme dosáhli celkové přesnosti 86 % s CNN modelem a 73 % s RNN modelem. Oba modely měly téměř stejnou validační i testovací přesnost. CNN model průměrně dosáhl podmínky konce trénování rychleji na datasetu *OuluVS2* než na *OuluVS* datasetu, i když potřeboval správně předpovědět výrazně větší počet dat. K ukončení trénování potřeboval průměrně 33 epoch a nejlepších výsledků dosáhl v průměru za 27 epoch.

Na druhou stranu RNN model ani v jednom případě nedosáhl podmínky k ukončení trénování, tedy se trénoval vždy 100 epoch, a k dosažení nejlepších výsledků potřeboval průměrně 40 epoch.

Co se týče modelu pro rozpoznávání sekvencí, jak lze vidět z grafu 5.7, model se trénuje správně, ale postrádá dostatečnou kapacitu k naučení se problematiky. Z důvodu hardwarové limitace jsem nemohl navrhnout a natrénovat model s dostatečnou kapacitou.

V experimentech jsem nedosáhl uspokojivých výsledků a trénování trvalo výrazně delší dobu oproti ostatním modelům, proto model nepovažuji za dostačující. Rozhodl jsem se proto model natrénovat pouze na prvním rozdělení. Model dosáhl testovací WER 60,67 %, validační WER 79,21 % a nejlepší trénovací WER 53,24 %.

5.6 Dosavadní nejlepší výsledky

V literatuře jsou uvedeny dosavadní nejlepší výsledky na mnou použitých datasetech, která uvádím pro porovnání. Na datasetu *OuluVS* bylo v práci [25] dosaženo přesnosti 92,1 %. Na sadě dat *OuluVS2* autoři v [26] dosáhli přesnosti 95,6 % na frázích. V práci [27] autoři dosáhli WER 7,2 % na sekvenci čísel.

6 Závěr

Při porovnání výsledků navržených architektur jsem došel k závěru, že lepších výsledků dosahuje konvoluční model než rekurentní model na obou datasetech. Pro většinu podmnožin konvoluční model dosáhl lepších a stabilnějších výsledků než rekurentní. Model rozpoznávající sekvenci čísel se mi nepodařilo úspěšně zrealizovat z důvodu hardwarového omezení. Domnívám se, že komplexnější model se stejnou architekturou by měl být schopný dosáhnout přijatelných výsledků.

V porovnání s profesionály na odezírání ze rtů modely dosáhly mnohem lepší přesnosti. Je důležité brát ohled na to, že tyto výsledky jsou trénované a testované na poměrně malém množství různých frází.

Je obtížné natrénovat velmi přesný model s malým počtem dat, z důvodu nejistot spojených s odezíráním ze rtů, a to i na datasetech jako jsou *OuluVS* a *OuluVS2*, které obsahují malý počet různých frází. Ostatní modely využívají další mechanismy společně s komplexnějšími sítěmi k zlepšení přesnosti, což v mém případě nebylo z důvodu nedostatku prostředků možné. Proto považujeme dosažené výsledky s našimi relativně jednoduchými modely za úspěch.

Dostupná řešení využívající neuronové sítě bývají trénována na výrazně větším množství slov, což je výhodné pro potřeby dobré generalizace. Zároveň jsou výrazně komplexnější, a proto dosahují lepších výsledků.

Použitá literatura

[1] NODA, Kuniaki, Yuki YAMAGUCHI, Kazuhiro NAKADAI, Hiroshi G. OKUNO a Tetsuya OGATA. Lipreading using Convolutional Neural Network. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2014, 1149-1153.

[2] GARG, Amit, Jonathan NOYOLA a Sameep BAGADIA. *Lip reading using CNN and LSTM*. Stanford University 450 Jane Stanford Way Stanford, CA 94305—2004, 2016. Dostupné také z: http://cs231n.stanford.edu/reports/2016/pdfs/217_Report.pdf

[3] SIMONYAN, Karen a Andrew ZISSERMAN. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv.org* [online]. 2015 [cit. 2020-05-19]. Dostupné z: https://arxiv.org/abs/1409.1556

[4] JANG, Dong-Won, Hong-In KIM, Changsoo JE, Rae-Hong PARK a Hyung-Min PARK. Lip Reading Using Committee Networks With Two Different Types of Concatenated Frame Images. *IEEE Access*. 2019, **7**, 90125-90131. DOI: 10.1109/ACCESS.2019.2927166. ISSN 2169-3536. Dostupné také z: https://ieeexplore.ieee.org/document/8756156/

[5] M. ASSAEL, Yannis, Brendan SHILLINGFORD, Shimon WHITESON a Nando de FREITAS. LipNet: End-to-End Sentence-level Lipreading. *ArXiv.org* [online]. 2016 [cit. 2020-05-19]. Dostupné z: https://arxiv.org/abs/1611.01599v2

[6] BARKER, Jon, Martin COOKE, Stuart CUNNINGHAM a Xu SHAO. The GRID audiovisual sentence corpus. *The University of Sheffield* [online]. Sheffield [cit. 2020-05-19]. Dostupné z: http://spandh.dcs.shef.ac.uk/gridcorpus/

[7] CHUNG, Joon Son, Andrew SENIOR, Oriol VINYALS a Andrew ZISSERMAN. Lip Reading Sentences in the Wild. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, 2017, , 3444-3453. DOI: 10.1109/CVPR.2017.367. ISBN 978-1-5386-0457-1. Dostupné také z: http://ieeexplore.ieee.org/document/8099850/

[8] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way. *Toward Data Science* [online]. 15 Dec 2018 [cit. 2020-05-19]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[9] VÉSTIAS, Mário P. A Survey of Convolutional Neural Networks on Edge with Reconfigurable Computing. *Algorithms*. 2019, **12**(8). DOI: 10.3390/a12080154. ISSN 1999-4893. Dostupné také z: https://www.mdpi.com/1999-4893/12/8/154

[10] BEZDAN, Timea a Nebojša BAČANIN DŽAKULA. Convolutional Neural Network Layers and Architectures. *Proceedings of the International Scientific Conference - Sinteza 2019*. Novi Sad, Serbia: Singidunum University, 2019, 2019, , 445-451. DOI: 10.15308/Sinteza-2019-445-451. ISBN 978-86-7912-703-7. Dostupné také z: http://portal.sinteza.singidunum.ac.rs/paper/700

[11] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. *ArXiv* [online]. 10 Dec 2015 [cit. 2020-05-19]. Dostupné z: https://arxiv.org/abs/1512.03385

[12] RUIZ, Pablo. Understanding and visualizing ResNets. *Toward Data Science* [online]. 8 Oct 2018 [cit. 2020-05-19]. Dostupné z: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

[13] Long short-term memory unit. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-05-21]. Dostupné z: https://en.wikipedia.org/wiki/Long_short-term_memory

[14] Gated recurrent unit. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-05-21]. Dostupné z: https://en.wikipedia.org/wiki/Gated_recurrent_unit

[15] BROWNLEE, Jason. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. *Machine Learning Mastery* [online]. 3 Dec 2018 [cit. 2020-05-19]. Dostupné z: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks

[16] Overfitting. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-05-19]. Dostupné z: https://en.wikipedia.org/wiki/Overfitting

[17] BROWNLEE, James. A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks. *Machine Learning Mastery* [online]. 7 Dec 2018 [cit. 2020-05-20]. Dostupné z: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

[18] IOFFE, Sergey a Christian SZEGEDY. Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift. *ArXiv* [online]. [cit. 2020-05-19]. Dostupné z: https://arxiv.org/abs/1502.03167

[19] BROWNLEE, Jason. A Gentle Introduction to Batch Normalization for Deep Neural Networks. *Machine Learning Mastery* [online]. 16 Jan 2019 [cit. 2020-05-19]. Dostupné z: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[20] SCHEIDL, Harald. An Intuitive Explanation of Connectionist Temporal Classification. *Toward Data Science* [online]. 10 Jun 2018 [cit. 2020-05-19]. Dostupné z: https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c

[21] Cross-validation (statistics). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-05-23]. Dostupné z: https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[22] OuluVS database. *University of Oulu* [online]. [cit. 2020-05-19]. Dostupné z: https://www.oulu.fi/cmvs/node/41315

[23] OULUVS2: A MULTI-VIEW AUDIOVISUAL DATABASE. *University of Oulu* [online]. [cit. 2020-05-19]. Dostupné z: http://www.ee.oulu.fi/research/imag/OuluVS2/

[24] GANDHI, Arun. Data Augmentation: How to use Deep Learning when you have Limited Data. *NanoNets* [online]. 2018 [cit. 2020-05-19]. Dostupné z: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

[25] HANDA, Anand, Rashi AGARWAL a Narendra KOHLI. A multimodel keyword spotting system based on lip movement and speech features. *Multimedia Tools and Applications* [online]. [cit. 2020-05-23]. DOI: 10.1007/s11042-020-08837-2. ISSN 1380-7501. Dostupné z: http://link.springer.com/10.1007/s11042-020-08837-2

[26] PETRIDIS, Stavros, Yujiang WANG, Pingchuan MA, Zuwei LI a Maja PANTIC. End-to-End Visual Speech Recognition for Small-Scale Datasets. *ArXiv* [online]. [cit. 2020-05-23]. Dostupné z: https://arxiv.org/abs/1904.01954

[27] CHUNG, Joon Son a Andrew ZISSERMAN. Out of Time: Automated Lip Sync in the Wild. *Computer Vision — ACCV 2016 Workshops*. Cham: Springer International Publishing, 2017, 2017-03-16, , 251-263. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-54427-4_19. ISBN 978-3-319-54426-7. ISSN 0302-9743. Dostupné také z: http://link.springer.com/10.1007/978-3-319-54427-4_19

A Obsah přiloženého DVD

- Zdrojové kódy