Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

NEURONOVÉ SÍTĚ PRO KLASIFIKACI AUDIO SIGNÁLU

Bakalářská práce

Martin Stránský

Studijní program: Aplikované vědy a informatika
Obor: Kybernetika a řídící technika
Vedoucí práce: Ing. Jan Švec, Ph.D.

Plzeň 2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin STRÁNSKÝ**
Osobní číslo:	**A17B0577P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Neuronové sítě pro klasifikaci audio signálu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Nastudujte problematiku hlubokých a rekurentních neuronových sítí.
2. Popište jednotlivé typy neuronových sítí a jejch použití v různých úlohách zaměřených na zpracování a klasifikaci audio signálu.
3. Pro vybranou úlohu navrhněte a realizujte celý experimentální postup: sběr trénovacích dat, návrh struktury modelu, trénování modelu a jeho evaluaci.
4. Dosažené výsledky analyzujte a navrhněte případné vylepšení.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce

Vedoucí bakalářské práce: **Ing. Jan Švec, Ph.D.**
Katedra kybernetiky

Datum zadání bakalářské práce: **15. února 2022**
Termín odevzdání bakalářské práce: **15. srpna 2022**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 15. února 2022

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 15. 8. 2022

.....
Martin Stránský

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Janu Švecovi, Ph.D., za podporu, rady a trpělivost při vedení bakalářské práce. Stejně tak bych rád poděkoval rodině taktéž za trpělivost a podporu při studiu.

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Abstrakt

Tématem bakalářské práce je seznámení se a následná aplikace neuronových sítí v oblasti audiosignálů. Jejím obsahem je problematika využití dostupných architektur neuronových sítí ve spojitosti s audiosignály a jejich popis. Následně je popsána úloha, jejímž cílem bylo zjistit dopad využití různých druhů textové reprezentace anotace (ortografické versus normované) trénovacích audionahrávek pro dotrénování předtrénovaného modelu. Model je následně otestován na testovacím balíku dat a jsou vyhodnoceny důsledky zvoleného trénovacího postupu na jeho výslednou přesnost, diskutován je i dopad dodatečné úpravy výstupu modelu na celkový výsledek. Bylo provedeno několik experimentů, jejichž výsledky jsou následně okomentovány a vyhodnoceny. Na získaných výsledcích je zformulována hypotéza o možnostech využití různých forem dat v závislosti na jejich dostupnosti.

Klíčová slova: Wav2vec2.0, Transformer, Automatické Rozpoznávání Řeči, Neuronové Sítě, Transkripce Textu, Transfer Learning.

Abstract

The aim of this thesis is to cover both the necessary fundamentals of the neural networks and how can they can be used in the real application in the field of audio signal processing. In the first part, the fundamentals of deep learning are covered, the terms explained and the development of selected neural network models and architectures briefly described, pointing out some of the caveats of the not-state-of-the-arts methods as these have led to the currently used methods. The significance of the current extensive development in the whole field of machine learning, mainly the emergence of transfer learning, the shift of the paradigm it means and possible social consequences are also stressed. The fundamentals provided are then used to explain the currently developed tools and lastly exploited in a real task. The following part describes the pre-trained model used which served as the basis for the fine-tuning process for the selected task and evaluates the results obtained though this process. In the conclusion a hypothesis about possible approaches varying on the available annotated data in order to obtain best results in similar applications is formulated upon these results.

Keywords: Wav2vec2.0, CTC, Transformers, Automatic Speech Recognition, Neural Networks, Text Transcription, Transfer Learning.

Obsah

Seznam Obrázků

Seznam Tabulek

Seznam zkratek

API Application Programming Interface. 40

CNN Convolutional Neural Network. 14

CTC Connectionist Temporal Classification. 29

ESN Echo State Networks. 11

GRU Gated Recurrent Units. 10

HHTT Human-Human Train Timetable. 46

HMM Hidden Markovov Model. 33

LSTM Long Short Term Memory. 7, 10

MLP Multi Layer Perceptrons. 5

RNN Reccurent Neural Network. 6, 7, 11

TPU Tensor Processing Unit. 1

WER Word Error Rate. 43

WIL Word Information Lost. 44

Kapitola 1

Neuronové sítě

Neuronové sítě konceptuálně vycházejí, i přes stále přetrvávající mezery v pochopení fungování nervové soustavy v mnoha ohledech, z chování biologických neuronů. Základním prvkem je, stejně jako u jejich biologického vzoru, umělý neuron. Aplikují se často v oblastech, kde je potřeba rozpoznávat nějaké vzory. Samotný příběh rozvoje neuronových sítí je v mnohém poučný – trvalo poměrně významnou dobu bylo přitom potřeba učinit (zpětně viděno) jen poměrně malý krůček (nabízí se paralela „malý krůček pro člověka, velký skok pro lidstvo“) — neurony spojit do vrstev.

Vývoj počítačového hardwaru zejména v posledním desetiletí, nejvýrazněji pak v segmentu grafických karet a vznik specializovaného hardwaru přímo specializovaného na výpočty úloh z oblasti strojového učení, například tensorových akcelerátorů (TPU[1]), výrazně ovlivnil možnosti uplatnění metod strojového učení. Tyto hardwarové prostředky umožňují trénovat i složitější neuronové sítě v akceptovatelném čase, což dává prostor pro neustále se rozšiřující oblasti možné aplikace.

Zároveň se v zejména posledních zhruba 2 letech posouvá i technická úroveň i velmi malých, levných, úsporných a zároveň paměťově až extrémně limitovaných čipů, či spíše přesněji System-On-Chip (například typu ARM-M4F). Tím je umožněno[2] implementovat například jednoduché porozumění řeči nebo potenciálu strojového učení využít jiným vhodným způsobem. Nabízí se například uplatnění v oblasti inteligentní regulace a optimalizace užití topení a dalších energetických zdrojů dle reálného používání (s ohledem na současné dění zejména toto bude dle mého mínění nabývat na významu).

Proč tato zdánlivě nesouvislá odbočka? Činím ji proto, že se čím dál častěji se používají předtrénované (pre-trained) modely na velkých datových sadách, přičemž vysoká výpočetní náročnost trénování téměř určuje, že se toto děje v nějakém velkém datacentru s potřebnými zdroji.

Takto vytvořené modely se poté přizpůsobují (s ohledem na další vlastnosti a potřeby celé odvětví často původně vznikají zejména s důrazem na aplikaci v prostředí sociálních

sítí – což je při uvážení jejich potřeb pochopitelné a samozřejmě přínosné, zároveň je ale při uvědomění si jejich významu pro společenské dění určitě na místě obezřetnost.

Výsledné modely lze poté po přizpůsobení specifickým potřebám využívat i na běžných spotřebitelských zařízeních. Drtivá většina dnešních aplikací neuronových sítí obecně využívá nějaký druh hluboké neuronové sítě.

1.1 Hluboké neuronové sítě

Úplnými počátky neuronových sítí byly jednoduché sítě o jedné vrstvě založené na perceptronech — pojmenování odvozeno od slova *percepce*, česky vnímání, přičemž cílem úsilí bylo napodobit vnímání lidské, vytvořit stroj, který umí „pochopit“ význam vstupu bez toho, aby bylo nutné všechno potřebné kódovat člověkem.

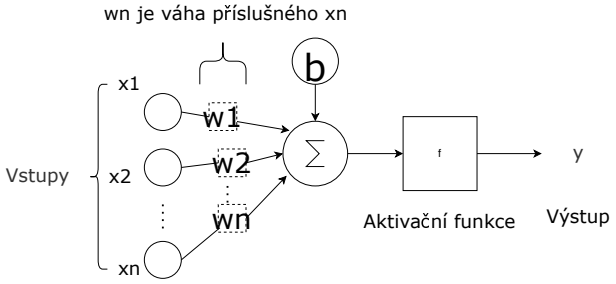
Souslovím „hluboká neuronová síť“ je obvykle označována taková neuronová síť, jež má mezi vstupní a výstupní vrstvou neuronů více než jednu mezivrstvu, v reálných aplikacích jsou těchto vrstev i desítky. Takové vrstvy jsou označovány jako skryté. Každá z těchto vrstev se určitým způsobem řízeně nebo samovolně organizuje podle druhu sítě, dodávaného vstupu a požadovaného výstupu.

1.1.1 Vrstva

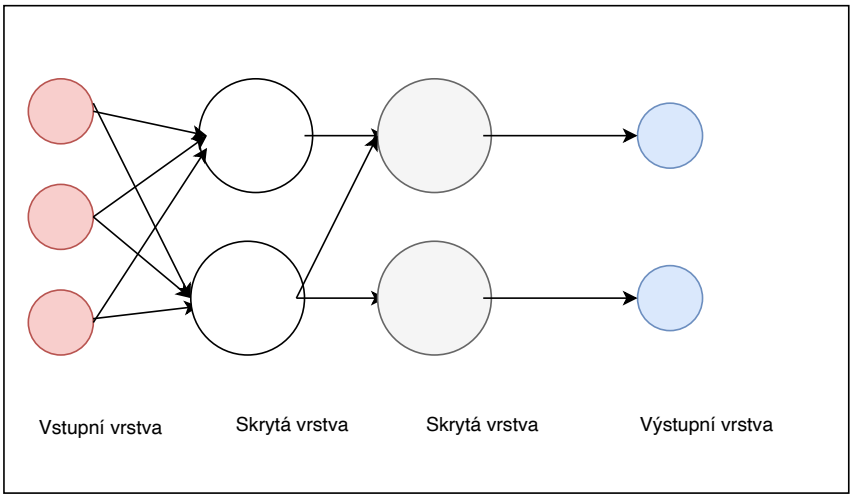
Vrstvou je míněna „řada“ neuronů ve stejné „hloubce“ ve vztahu k výstupu/vstupu.

Jednoduše řečeno: Vstupní vrstva slouží jako vstupní bod do sítě. Přes tyto uzly se do sítě vkládají data. Skryté vrstvy jsou ty, kde se děje „magie“. Výstupní vrstva slouží k vyčítání výsledků.

Každý neuron má obecně weight (váhovou) matici, bias a aktivační funkci. Neurony v rámci vrstvy typicky využívají stejnou aktivační funkci.



Obrázek 1.1: Vysvětlení atributů na perceptronu



Obrázek 1.2: Ilustrativní příklad hluboké NN složené ze vstupní, 2 skrytých vrstvami a výstupní vrstvy

1.1.2 Aktivační funkce

Každý neuron má aktivační funkci, jež určuje (výslednou) odezvu neuronu na dané vážené vstupy (po zohlednění prahu). Jejímž účelem je umožnit správnou reakci na vstup ve spolupráci s vahami a biasem – neuron se aktivuje až po dosažení daného prahu.

Obecně je úkolem aktivační funkce rozdělit prostor na 2 poloprostory a případné zavést do modelu nelinearitu.

Běžně užívanými funkcemi jsou lineární funkce:

$$f(x) = x \tag{1.1}$$

a zástupci nelineárních funkcí: sigmoid, tanh, ReLU.

$$Relu(z) = max(0, z) \tag{1.2}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1.3}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.4}$$

1.1.3 Hluboké učení

Považuji za vhodné vyjasnit i další pojmy s tímto tématem související, ve veřejném prostoru se často užívají, nejspíše protože znějí zajímavě, ale mnohdy bez vymezení jejich obsahu.

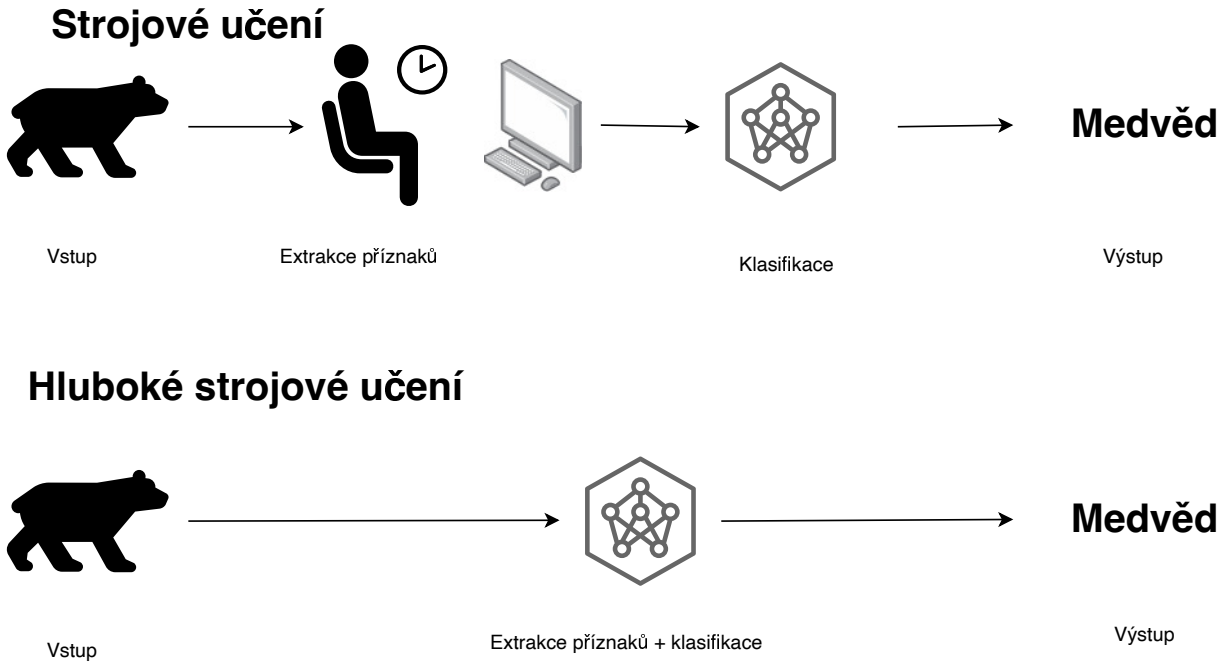
Hluboké neuronové sítě jsou součástí „balíčku“ technologií označovaných jako *strojové učení*, přesněji *hluboké* strojové učení.

Jaký je rozdíl mezi strojovým učením“ a *hlubokým.* strojovým učením“?

Strojové učení obecně je v současné době aplikací souboru znalostí zejména v oblasti lineární algebry a různých statistických metod.

Stěžejní rozdíl v přístupech a důvod užívání přívlastku „hluboké“ stručně tkví v tom, že hluboké strojové učení spojuje dvě vnitřní fáze procesu vstup –> výstup, extrakci příznaků ze vstupu a následnou klasifikaci, do jednoho celku. Extrakce příznaků je v prosté řeči převod vlastností (například, kdyby se jednalo o systém pro určení barvy nějakého předmětu) z „lidského“ (barva) popisu do numerického (vhodnou klasifikací), člověkem určenou.

Názorně lze rozdíly ilustrovat například hypotetickém klasifikátoru zvířat.



Obrázek 1.3: Ilustrace konceptuálního rozdílu v přístupech

Pro praktické použití je nejdůležitější praktických rozdílem hardwarová náročnost. Jednoduchý lineární klasifikátor lze natrénovat maximálně v řádu minut na běžném hardwaru, pro hluboké učení je typicky třeba výrazně výkonnější hardware, typicky jsou využívané grafické procesory. Příčinou je datová struktura reprezentující jednotlivé komponenty modelů: matice/tenzory. Ty odpovídají tomu, na co jsou navrhovány právě grafické karty: na práci s maticemi.

Navíc umožňují významnou míru paralelizace výpočtů proti běžným CPU, neboť ty mají typicky jednotky až nízké desítky jader. V porovnání grafické procesory disponují stovkami

až tisíci specializovaných jednotek a k tomu disponují vysokou paměťová propustností i velkým množstvím paměti. Případně specializované jednotky přímo na různé druhy strojového učení.

1.1.4 Multilayer Perceptrons (MLP)

Vícevrstvé perceptrony (MLP) jsou dopřednou architekturou neuronových sítí, u níž lze bez zaváhání prohlásit, že otevřela dveře všem moderním architekturám. Je plně propojená. Umožnily vyřešit zásadní překážku pro řešení mnohých problémů pomocí neuronových sítí: Umožnily namapovat vstup a výstup s nelineárním stavem (lineárně neseparabilní). Učení probíhá za pomoci algoritmu backpropagation. [3] Principiálně jde o základní stavební kameny potřebné a v nějakém rozsahu využívané pro veškeré složitější modely či druhy sítí.

1.2 Trénování hlubokých neuronových sítí

Trénování hlubokých neuronových sítí je poměrně komplexním procesem, nejdůležitějším nástrojem je backpropgation pro nastavování vah.

1.2.1 Backpropagation

Backpropagation (česky *Algoritmus zpětného šíření chyby)* je algoritmem, bez něhož by využití hlubokých neuronových nebylo možné. Umožňuje aktualizovat váhy v neuronové síti na základě žádoucího a skutečného výstupu sítě. Využívá takzvaného *řetězového pravidla* o derivaci složené funkce.

Průběh aktualizace vah v síti lze principiálně popsat ve 4 krocích:

1. Dávka dat je postoupena skrze vstupní vrstvu.
2. Dopředný průchod (forward propagation), čímž jsou získány hodnoty ztrátové funkce.
3. Zpětný průchod, výpočet gradientů.
4. Gradienty jsou použity k aktualizaci vah.

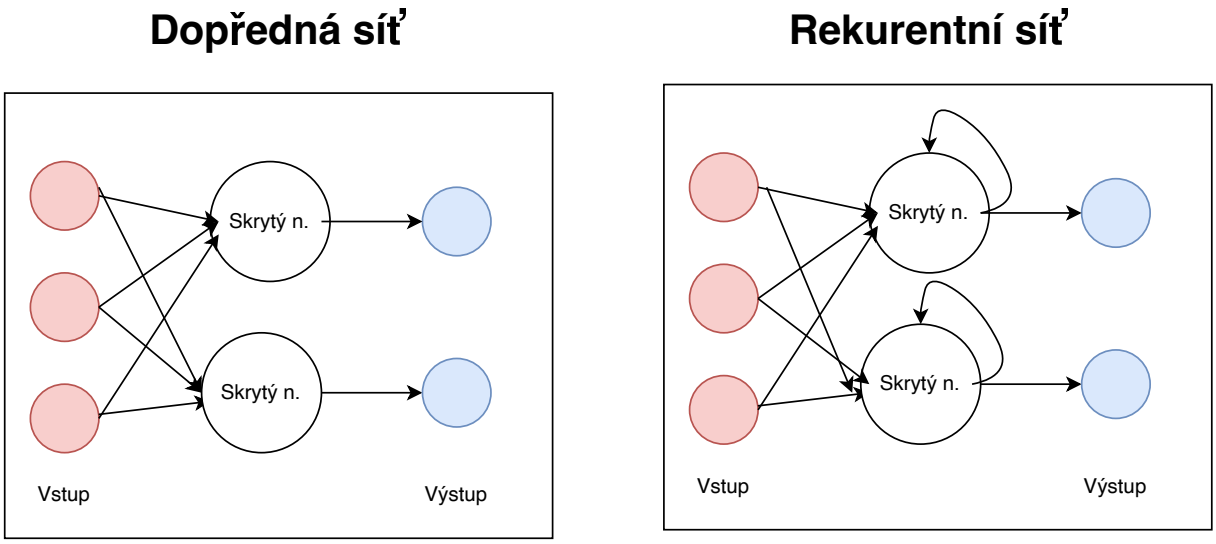
1.2.2 Problém mizejícího gradientu

Anglicky *Vanishing gradient problem* je problém vyskytující se při trénování hlubokých neuronových sítí právě algoritmem backpropagation.[4] Příčinou je velikost derivací aktivačních funkcí, například sigmoidální funkce. Vlivem uplatňování řetězového pravidla se

malé přírůstky násobí mezi sebou, čímž se ještě zmenšují. V nejhorším případě se může stát, že se tyto derivace stanou tak malými, že se trénování úplně zastaví.

1.3 Rekurentní neuronové sítě

Neuronové sítě jsou či bývaly většinou navrhovány dopředně. Signál prostupuje sítí jedním směrem: od vstupní vrstvy k výstupní. Tento přístup je ale pro některé aplikace omezující, neboť někdy nestačí pouhá znalost nynějšího stavu. Řešení se snaží nabídnout rekurentní neuronové sítě (**RNN** – Reccurent Neural Network).



Obrázek 1.4: Schematická ilustrace konceptuálního rozdílu v návrhu sítí

O jaké příklady jde?

Kupříkladu při čtení psaného textu nepracuje člověk s každou větou izolovaně. Naopak text chápe v závislosti na tom, co bylo napsáno dříve a co již víte z dřívějšího kontextu. Myšlenky a výsledný vjem nevznikají zcela izolovaně po slovech či větách, nově získané informace se promítají do předchozích.

Poměrně intuitivní vysvětlení poskytuje následující příklad: Pokud bychom například chtěli běžnou dopřednou neuronovou síť využít třeba k tomu, aby nám řekla, co se děje v určitém okamžiku ve filmu, neměli bychom příliš úspěch. Síť není schopna využít předchozí stavy, aby si věc „zasadila do kontextu“, protože tato „běžná“ síť pracuje pouze s aktuálním stavem. Dozvěděli bychom se třeba, že se na plátně probíhá zatčení podezřelého zásahovkou, ale už bychom se nedozvěděli, že zatčený je bossem drogového gangu, který se chystal zavraždit nebohou bankovní úřednici – síť to není schopna odvodit, předchozí stav si nepamatuje.

RNN toto řeší zavedením smyček, v zásadě se nejedná o nic jiného než o řetězení kopií téže sítě, přičemž každá kopie předává informaci následující. Rekurentní sítě tím umožňují zachycovat/modelovat i nelineární procesy a časově závislé vzory – tedy například různé časové řady či sekvence (signály).

Rekurentní neuronové sítě jsou tedy takto nazývány kvůli tomu, že provádějí operace pro každý prvek v řadě, s tím, že výstup je závislý na předchozích výpočtech (rekurze). Jinou možností jak o RNN přemýšlet je představovat si je jako druh paměti, jež zachycuje, co zatím bylo vypočteno.

Problémem RNN jsou komplikované algoritmy pro jejich trénování vedoucí k časově náročnému procesu využití, přičemž výsledná řešení obvykle nejsou optimální. Teoretická délka zapamatovatelných sekvencí je nekonečná, ale v praxi se obvykle využívá pouze několik kroků zpět. Teoreticky nekonečnou „kapacitu“ pro pamatování omezují běžně užívané aktivační funkce typu *tanh* nebo *relu* a problémy s gradientem. Tím, že vstupem sítě jsou její vlastní výstupy, hrozí například různé vnitřní smyčky a zesilování šumu.

S rostoucím výpočetním výkonem a dostupností různých nástrojů pro práci s těmito sítěmi zažívaly a stále zažívají různé podtypy RNN a z nich vycházející nebo částečně jich využívající architektury rozkvět. Širokou paletu přístupů dobře ilustruje [5]. Následuje výčet architektur z obecných RNN vycházejících. Architektury jsou to v dnešní době již „oborovým standardem“ a proto si zasluhují stručný popis, zejména kvůli důvodům, jejichž následkem byla motivována snaha vyvinout „lepší“ přístupy.

Výpočetní náročnost rekurentních sítí pro představu závisí takto: O(délka ·o rozměr) Je to právě i důsledkem vnitřního fungování – vlivem závislosti na předchozích vstupech se celý proces poměrně obtížně paralelizuje.

1.3.1 Long Short Term Memory sítě (LSTM)

Long Short Term Memory speciálním druhem[6] RNN, který se umí učit dlouhodobé závislosti a zároveň odstraňuje potenciálně zásadní potíž standardních RNN – u nich je každá informace brána jako stejně důležitá, což přispívá k projevování efektu vanishing-gradientu. K potlačení tohoto efektu se u LSTM zavádějí brány (gates), které řídí míru zachování informace v daném časově odpovídajícím modulu.

Příslušné rovnice popisující jednotlivé stavy mají následující podobu:

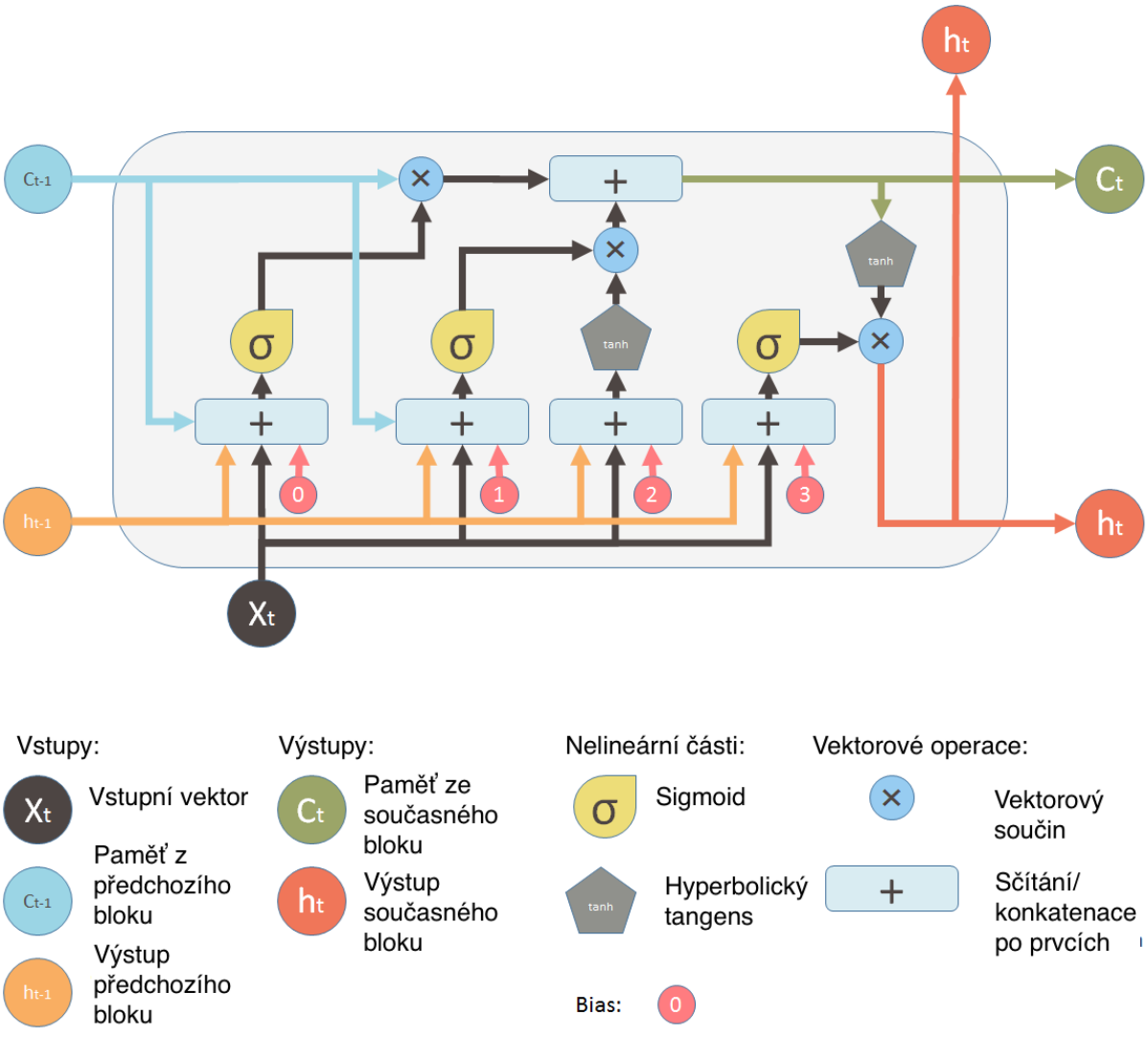
$$i_t = \sigma(W_{xi)}x_t + W_{hi)}x_{t-1} + W_{xi)}x_t \tag{1.5}$$

$$f_t = x^n + y^n = z^n \tag{1.6}$$

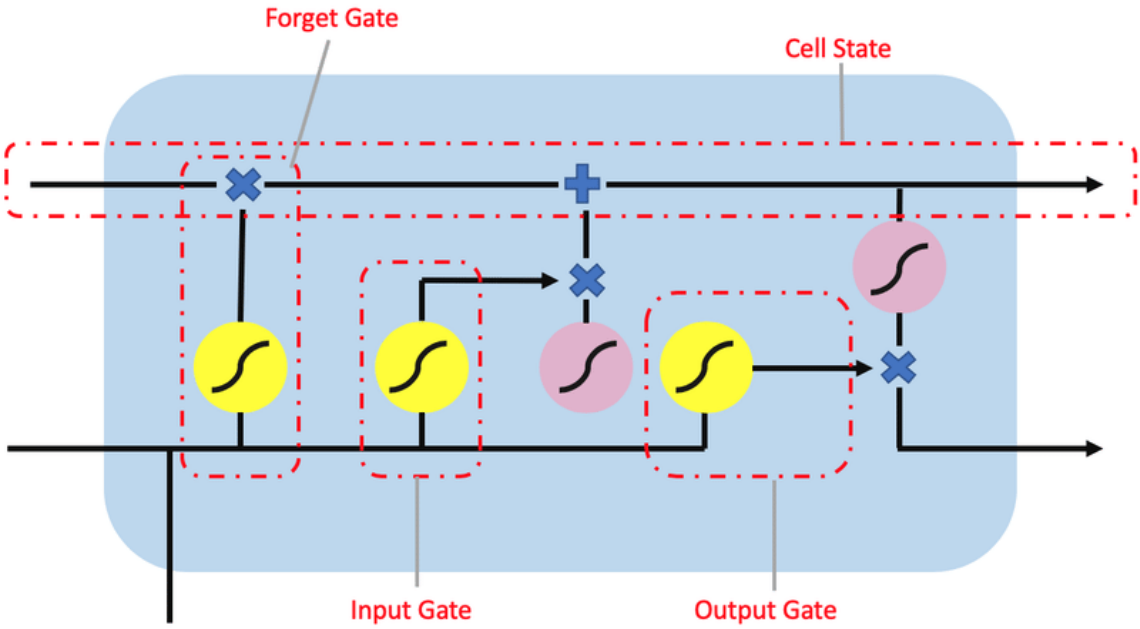
$$c_t = f_t c_{t-1} + i_t \tanh W_{xc}x_t + W_{hc}h_{t-1} + b_c \tag{1.7}$$

$$o_t = x^n + y^n = z^n \tag{1.8}$$

$$h_t = o_t \tanh(c_t) \tag{1.9}$$



Obrázek 1.5: Ilustrace bloku LTSM sítě; překlad z anglického originálu autorem[7]



Obrázek 1.6: LSTM brány, přejato z[8]

Forget gate

Určuje množství předchozí informace, které se zapomene (odtud forget gate). Používá simoidální funkci. Porovnává aktuální vstup se stavem v čase *t-1* a dává výstup v intervalu $(0; 1)$, kde hodnota odpovídá míře uchování informace.

Input gate

Tato brána je aktivní pouze v časovém okamžiku t. Uzel sítě rozhoduje, které vstupy budou mít vliv na aktualizaci nynějšího stavu buňky a jaké případné další vstupy budou vzaty v úvahu.

Output gate

Propouští ven výsledný stav jako výstup dané paměťové buňky.

Nejedná se o bránu, ale k doplnění schématu a fungování jsou nezbytné ještě další části: **Cell-State (stav buňky):** Nejlépe je význam vcelku intuitivně popsatelný na pásovém dopravníku.

Hidden-State (skrytý stav): Vlastní výstup buňky pro nynější vstup – jako skrytý se označuje kvůli tomu, že tento výstup se používá pouze jako vstup pro další časový okamžik, pro výstup z buňky slouží výstup z výstupní brány. Brány mají sigmoidální aktivační funkci.

Stručný principiální popis fungování:

1. Do uzlu je načten stav *t*-1 a vstup v čase *t*.

2. Zapomínací (forget) brána s pomocí informací ze vstupu odstraní nežádoucí informace ze stavu.

3. Vstupní (Input) brána rozhodne, jaká (část) informace z předchozích stavů se zachová a jaká nová uloží do stavu.

4. Stav se uloží do skrytého stavu.

5. Výstupní (Output) brána vpustí na výstup vypustí „filtrovanou“ verzi.

Filtrování výstupu probíhá tak, že nejdříve se použije sigmoidální vrstva, která rozhodne o tom, jaké informace ze stavu se využijí, tento výstup je následně přenásoben *tanh* aktuálního stavu, čímž je dosažen správný výstup.

1.3.2 Gated Recurrent Units

Gated Recurrent Units (GRU) jsou svým způsobem pokusem o zjednodušení LSTM – a to jak v oblasti trénování, tak strukturálně.

Důvodem jejich užívání je primárně možnost řídit, jakým způsobem je užitý nynější vstup a předchozí paměťové stavy pro aktualizaci nynější aktivace a vytvoření současného stavu pomocí dvou „bran“ (*gates*). Je také sdružen cell-state a hidden-state.

Tyto brány (označované jako „update“ a „reset“) mají vlastní sadu vah, která je adaptivně upravována v průběhu učící fáze, což má za cíl zamezit výskytu vanishing gradientu.

Činnost těchto bran je stručně nejlépe přiblížitelná pojmy[1] „pomněnka“ a „zapomněnka“, kdy jedna z těchto bran (*update*) určuje, jaké množství informací si model smí zapamatovat a druhá (*reset*) kolik musí zapomenout.

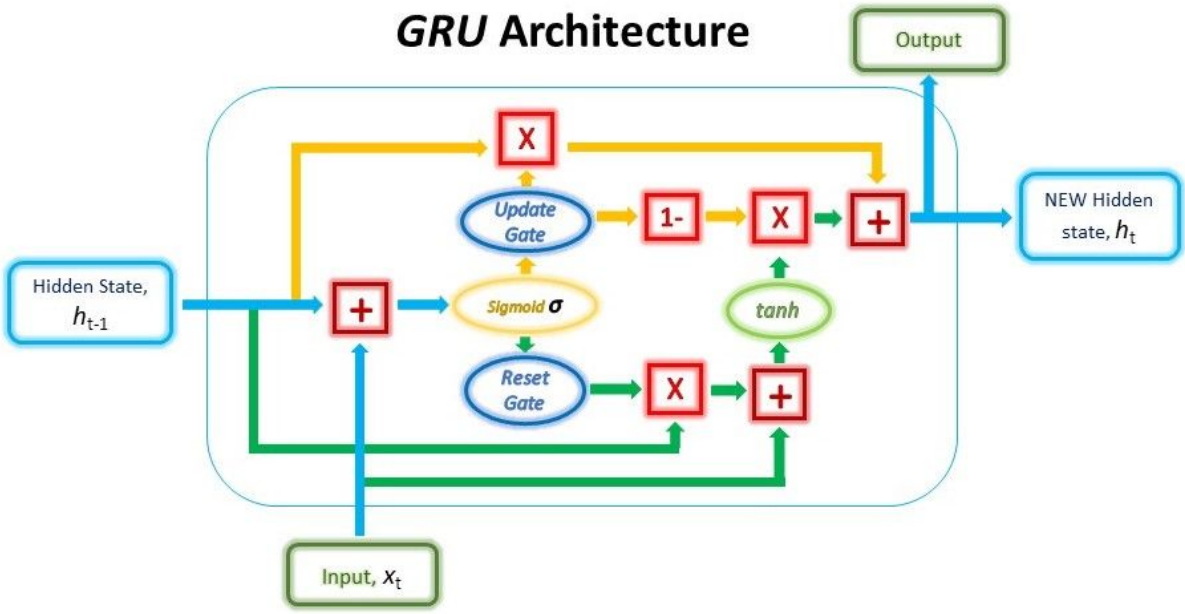
Podrobnější vysvětlení:

Update gate: Pomáhá v rámci modelu určit, jaké množství informace z dřívějších stavů má být předáváno do budoucnosti. V některých případech může model dojít k tomu, že se přenese celý předchozí stav, čímž se obejde riziko vanishing gradientu.

Reset gate:

Úkolem této brány je přesný opak. Určuje, jaké množství z předchozího stavu je potřeba zapomenout.

[1] Jára Cimrman během svého pedagogického působení rozděloval probírané učivo na dvě části, kterým přiřadil dva názvy: pomněnka a zapomněnka. Žáci se učili obojímu, ovšem jen pomněnku si směli pamatovat.



Obrázek 1.7: Schéma GRU bloku (buňky), převzato z [9]

1.3.3 GRU nebo LTSM

Oba typu sítí, jak GRU, tak LTSM, dosahují podle experimentů podobných výsledků a nelze jednoznačně prohlásit jednu nebo druhou variantu za lepší. Autoři se přiklánějí k názoru, že pro menší datové soubory je vhodnější GRU, pro větší LTSM. Pro modely sequence-to-sequence se experimentální výsledky přiklánějí spíše na stranu LTSM, doslovně ze závěru autorů: *„LSTM buňky konzistentně [výkonem] překonávaly GRU buňky“*. [10]

1.3.4 Echo State Networks (ESN)

Echo State Networks jsou dalším druhem rekurentní neuronové sítě, užívaným i v oblasti audiosignálů.[11] Tento druh sítí patří do oblasti Reservoir computingu (doslovně nepříliš čtivě přeložitelné jako „přehradové počítání“). [12]

Principem využití ESN je rozšíření modelu o neuronovou síť, jejíž vstup je připojen na „pevnou“ (netrénovatelný) a náhodný dynamický systém, v němž se využije embedding k tvorbě vnitřní reprezentace. Přes další již trénovatelné jednotky je výstup potřebným způsobem využit.

Základní myšlenkou toho přístupu je úvaha, že pro využití v praxi často dostačuje číst výstup poslední (výstupní) lineární vrstvy.

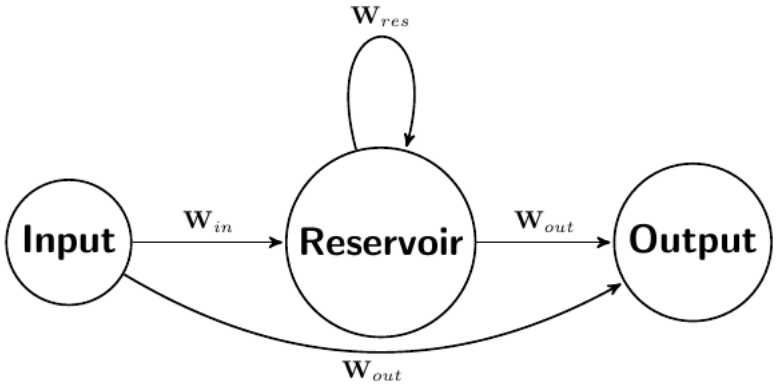
Cílem je získat výhody, které poskytují RNN (zpracovávat sekvenci na sobě vzájemně závislých vstupů, například časově – signál) bez obtíží, kterými jsou zatíženy tyto klasické

rekurentní sítě, jako například problém mizejícího gradientu *(the vanishing gradient). problem).*[13]

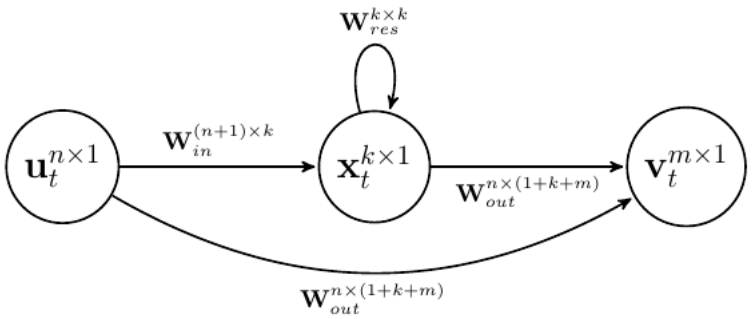
Základní charakteristiky ESN by se daly shrnout takto:

- Velké množství neuronů.
- Váhy mezi vstupními a skrytými vrstvami jsou náhodně přiřazované a nejsou trénovatelné.
- Míra propojení mezi vstupními a skrytými vrstvami je nízká (typicky do 10 %).
- Rekurentní nelineární embedding.
- Mají jednoduchý lineární algoritmus pro trénování.
- Výpočetně nenáročné (nepoužívá se backpropagation).
- Trénování s učitelem probíhá za využití lineární regrese.

Hlavní výhody ESN byly do značné míry převáženy vyřešením problémů s trénováním jiných druhů rekurentních sítí a dnes se příliš nevyužívají. Některé metody, nebo i přímo ESN samotné, se ale využívají v oblasti zpracování signálů nebo například byly využity při simulaci mechanických nano-oscilátorů.[14]



Obrázek 1.8: Ilustrativní princip ESN vstup-rezerovoár-výstup, převzato z [15]



Obrázek 1.9: Schematický výpočet stav, převzato z [16]

1.3.5 Obousměrné RNN (Bidirectional RNN)

Jedná se o druh RNN, obdobně případně LTSM a existují i různé další modifikace obousměrně zřetězených sítí, jenž je zřetězený nejen směrem zpět (do minulých pozorování), ale i dopředu. Zásadním omezení je, že je nutné znát celý vstup (včetně „budoucích“, vůči času T) a nelze je tak typicky využít pro aplikace v reálné čase. Principiálně je ideálním využitím například rozpoznávání ručně psaného textu – znalost předchozího a následujícího písmene ve slově může zvýšit výslednou přesnost klasifikace.[17]

1.3.6 Shrnutí problematických aspektů rekurentních sítí

Zásadním problémem rekurentních sítí, ať již „obecných“ nebo architektur typu LSTM, je při trénování modelů na nich postavených právě provázanost jejich vnitřních stavů a sekvenční zpracování – pro výpočet dalšího stavu je nutné znát stav předcházející, což vede na neuspokojitelné paměťové nároky[18] a navíc výrazně omezuje možnosti paralelizace a v důsledku prodlužují trénování modelu. Dalším nedostatkem je, že vzdálenost mezi jednotlivými časovými kroky je vždy stejná a tím i jejich důležitost.

Toto vedlo k rozvoji technik, jakými je například *attention* (pozornost). Tato tento přístup umožňuje[19] více se věnovat určité části vstupní sekvence. V odkazovaném dokumentu, v němž se autoři věnují mimo dalšího vodním ptákům, to znamená pozornost věnovat primárně buď zvířeti, nebo vodě v pozadí v jednotlivých fotografiích. Postupným dalším rozvojem byly objeveny jiné přístupy a zkonstruovány jiné architektury a modely sítí, sice již rekurentních sítí třeba ani nevyužívající, ovšem bez nich by pravděpodobně nikdy nebyly objeveny.

1.4 Konvoluční neuronové sítě

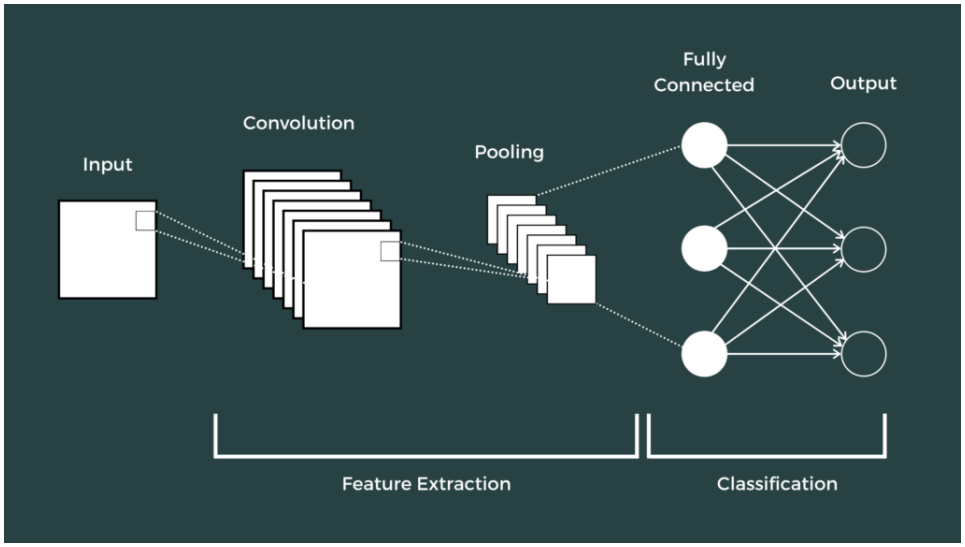
Existují i další důležité druhy neuronových sítí, mezi ně patří kupříkladu ty konvoluční (CNN).[20] Přestože byly původně konvoluční sítě používány zejména v úlohách zabývajícími se obrazem, některé projekty, jako například *Google WaveNet*, je postupně využily i v oblastech souvisejícími se audiosignály a u současných architektur modelů pracujícím i s daty reprezentující informace předávané skrze jiné médium, například právě zvukové/audio aplikace, jsou již zcela standardně užívány jako jedna z vrstev. Existují a využívají se[21] také snahy propojit rozpoznávání obrazu a zvuku a pro některé úlohy využít ke klasifikaci audiosignálů nebo přesnějšího získání jejich charakteristik obrazová data v podobě spektrogramů.

Tento přístup má podle některých autorů ale poměrně citelné nedostatky, například [22] Názorně ilustrovatelný je zásadní rozdíl v charakteru vizuálních a audio-dat na příkladu pozastaveného filmu, vezmeme-li jeden snímek: Zatímco ze „statického“ obrazu (typická snímkové frekvence je cca 24-25 Hz) stále lze získat nějaké informace vyčtete, zvukových je v takovémto okamžiku, pokud jsou, jen velmi poskrovnu.

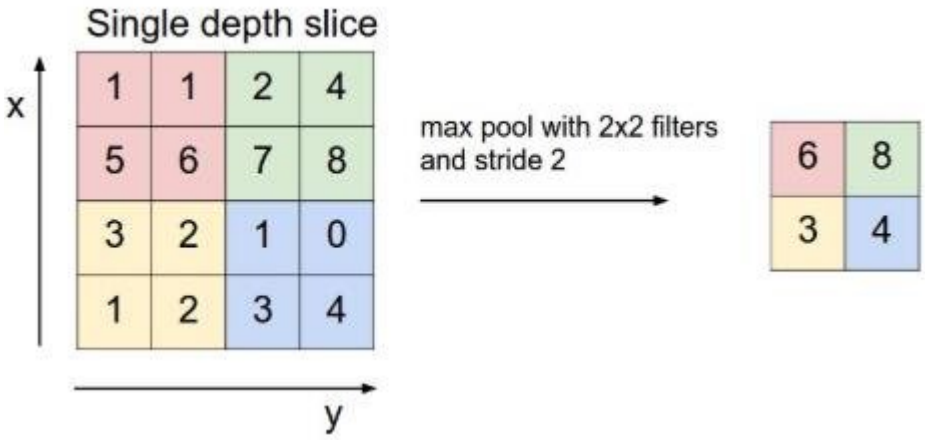
Tento problém vychází z fyzikální podstaty zvuku, kdy jde o řadu změn tlaků vzduchu a význam lze určit až jejich analýzou v čase, k čemuž se perfektně hodí dle výše uvedeného popisu právě konvoluční sítě. Jak ale ukazuje praxe, žádné z předkládaných problémů kritiků nejsou tak závažnými, aby bránily jejich využití pro různé podúlohy v rámci různých potřebných procesů v reálných aplikacích.

Důležitými složkami při užití těchto sítě jsou konvoluční vrstva, která aplikuje na sérii konvolucí a funguje tedy jako filtr. Cílem vrstvy je vytvořit ze vstupu vektor, který obsahuje důležité příznaky.

Další přítomnou vrstvou je pooling-vrstva. Ta má za úkol snížit rozměr (dimenzi) pro snížení výpočetní náročnosti.



Obrázek 1.10: Schéma konvoluční sítě, přejato z[23]



Obrázek 1.11: Úloha pooling vrstvy konvoluční sítě, přejato z[24]

Přínosem konvolučních sítí proti vícevrstvým perceptronovým sítím je též vyšší efektivita, protože přes překryvy v rámci okénka si část parametrů sdílejí.

Případné zájemce o získání zevrubného pochopení fungování konvolučních sítí je výrazně efektivnější cestou než číst pouze text využít interaktivních prostředků jako je například CNN Explainer.[2] Jedná se nástroj vytvořený univerzitou Georgia Tech s podporou velkých technologických společností jako je Google, Amazon a další[25].

Uživatele názorně provede všemi kroky trénování konvoluční sítě pro rozpoznávání obrázků. Míra interakce a tím náročnost pochopení je výrazně odlišná a pro „pasivní“ médium v podobě (digitální reprezentace) papíru porovnání nevychází příznivě.

Architektura sítí ale není příliš vhodná pro řešení některých problémů a řešení vzájemných závislostí mezi vstupem a výstupem, což se přímo dotýká například strojového

[2]Dostupný zde: **https://poloclub.github.io/cnn-explainer/**

překladu. Pro tyto účely jsou tedy stále rozvíjeny další techniky, i v kombinaci s těmito sítěmi.

Kapitola 2

Neuronové sítě a audiosignály

Jednou z oblastí, kde neuronové sítě našly uplatnění (ostatně plně to souzní s prapůvodní myšlenkou — i u člověka zpracovává zvukové vjemy nervová soustava), je řešení úloh, v nichž je nějakým způsobem potřeba pracovat s audiosignály – ty jsou nalezitelné ve všemožných oblastech života, od ryze účelů vědeckých, zdravotních či vzdělávacích až po zábavu.

2.1 Oblasti užití

Příklady různých druhů s potenciálem pro aplikaci neuronových sítí v oblasti zpracování (nejčastější je klasifikace) audiosignálů jsou například:

- Určení různých s lidskou řečí spjatých charakteristik, typu národnost, emocionální rozpoložení, pohlaví a podobně,
- Detekce přehrávané hudby (aplikace typu Shazam) nebo určování hudebních žánrů pro další použití např. pro doporučování hudby,
- Odšumování nebo odstraňování specifických zvuků (např. restaurování gramofonových desek atd.),
- Vylepšování hlasu v telefonu (zlepšení kompresních technik),
- Řečová syntéza – Různé systémy typu **Text-To–Speech** (text na řeč) nebo různé informační systémy, například rozhlas na nádražích (metro v Praze) nebo pro nevidomé uživatele PC,
- Pomocný nástroj pro vědeckou komunitu, například přírodovědce – klasifikace druhů živočichů, ve zcela konkrétní aplikaci určování ptáků dle jejich zvukového projevu a případně dalších chakteristik, [26]

- Pro využití řeči (syntéza i zpracování) pro ovládání zařízení, což v praxi znamená využívání hlasových asistentů jako je Apple Siri, Amazon Alexa, Google Assistant a další,
- Výuka jazyků (aplikace typu *Duolingo*),
- I v souvislosti s pandemií covidu-19 se rozmáhají taktéž diagnostické aplikace pro detekci nemocí z řečového signálu. [27]

2.2 Další vybrané architektury sítí

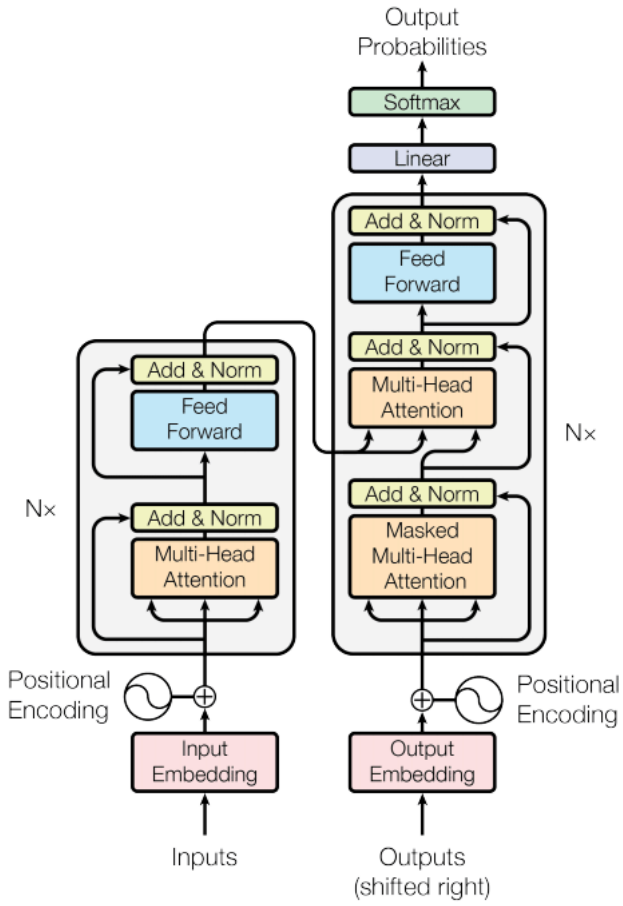
Mimo diskutované rekurentní sítě jsou pro zpracování audiosignálů využívány i další druhy sítí.

2.2.1 Transformer

Jedná se zástupce architektur typu sequence-to-sequence, obvykle slouží jako stavební prvek pro ještě komplexnější modely než jsou doteď uvedené druhy sítí. Zástupcem takových architektur je například wav2vece. Tuto síť si popíšeme hned po vysvětlení principu transformerů Sequence-to-sequence znamená, že pro vstupní sekvenci na výstupu obdržíme rovnou výstupní sekvenci.

Tento přístup byl obecně velkým technologickým průlomem, umožňujícím realizovat do té doby nevídané výkony.

Modely typu Trasformer řeší mnoho z popsaných problémů rekurentních sítí. Jejich struktura umožňuje řešit celou posloupnost vstupních vektorů paralelně, což umožňuje trénování výrazně urychlit, i přes přítomnost výpočetně náročných komponent.



Obrázek 2.1: Schéma transfomeru, součást [18]

Jak konkrétně „transformeři“ fungují? Na vstupu i výstupu se nachází shodný počet enkodérů (vstup) a dekodérů (výstup) – autoři architektury využívají v původním paperu konkrétně 6, ale není to nutností a jedná se o jeden z parametrů, s nímž lze případně experimentovat. [28]

Enkodéry jsou všechny stejné, co se týče struktury a vlastností, ale liší se jejich váhy. Každý enkodér se skládá ze dvou „submodulů/podvrstev“ – na vstupu se nachází tzv. self-attention (pod)vrstva, což je rozšíření u rekurentních sítí stručně nastíněné attention. Její fungování bude rozvedeno vzápětí. Výstup této vrstvy je následně předán do dopředné sítě/vrstvy.

Dekodér obsahuje „(pod)vrstvy“ tři. Mezi self-attention a dopřednou vrstvou je vložena ještě *encoder-decoder attention*, jejímž účelem je pomáhat „zaměřit pozornost“ na určité části vstupní sekvence (například relevantní věty, kupříkladu pro účely slovosledu), což bude přesněji vysvětleno dále.

Na vstupu je každé slovo pomocí *embeddingu* slovo převedeno na vektory. U embeddingu se na chvíli zastavíme vzápětí.

2.3 Důležité koncepty a pojmy

2.3.1 Self-supervised learning

Jedná se o koncept, který má za úkol vytvořit druh učení spojující tradiční učení bez učitele a učení s učitelem, což umožňuje odstranit jednu z největších překážek vyvstávajících při aplikaci většiny metod strojového učení s učitelem: potřeba anotace (labelování) dat, což je činnost velmi náročná na čas, potažmo lidskou práci, a kromě nákladů tedy i náchylná na vznik chyb.

Nejzásadnějším dlouhodobým přínosem je, že přestává být pro drtivou většinu aplikací „kde sehnat vhodná data v dostatečné kvantitě".

Neanotovaných dat je k dispozici neuvěřitelné množství – pro letošní rok se odhaduje, že ve světě je uloženo 100 ZB (předpona zetta má v systému SI význam 10^{21} – tedy v běžněji užívaných jednotkách 1 miliarda (10^9) terabajtů) a problematika trénování a aplikace modelů se tak dostává do roviny omezenosti ryze výpočetním výkonem (což je ale s ohledem na jeho dostupnost opět spjato s dalšími možnými důsledky zmíněnými v úvodu).

2.3.2 Contrastive learning

V českém překladu *kontrastní učení*. Princip tohoto druhu učení tkví v tom, že na základě předchozích pozorování se model snaží určit, jestli jsou si vzorky podobné, či nikoli. Cílem je, aby v lineární prostoru měly podobné objekty malou vzdálenost, rozdílné co největší, je tedy přímo provázané s embeddingem a jde o jeden z nejsilnějších dostupných nástrojů pro self-supervised learning. [29]

2.3.3 Transfer learning

Jde o obecný specifický přístup ke tvorbě modelu využívajících strojové učení: V první fázi je vytvořen obecný model, z obrovského balíku dat ze všech možných druhů dle povahy úlohy, přičemž tímto je vytvořen „obecný základ“ pro selekci relevantních příznaků v rámci modelu. Tato fáze se nazývá pre-training (využije se typicky self-supervised learning2.3.1).

V druhé fázi je již pro určitou specifickou úlohu dodáno i řádově menší množství dat přímo související s touto konkrétní aplikací, přičemž toto způsobí „vyladění“ vah na výstupu modelu – od tohoto se tato část označuje jako fine-tuning, doslovně „jemné ladění“.

2.3.4 Embedding

V tomto kontextu jde do češtiny obtížně přeložitelný termín, nejpřesněji asi jako *zapuštění*, *zasazení* či *vnoření* (tento poslední termín jsem v českých zdrojích dohledal jako užívaný).

Oč jde: Metody (hlubokého) strojového učení jsou i přes někdy až vytvářený dojem magie souborem (mimo dalších) aplikovaných znalostí zejména z oblasti lineární algebry a statistiky. Je tedy potřeba kupříkladu z námi (lidmi) užívané reprezentace (například textové u jazyka) vytvořit vhodnou numerickou reprezentaci, protože všechny užívané metody pracují v nějaké podobě s vektory tvořenými číselnými hodnotami. Metoda má 2 zásadní vlastnosti:

- Prostorově (dimenzemi) efektivní reprezentace
- Významově blízké entity (slova) jsou si blízko

Co to konkrétně znamená, zejména druhá vlastnost?

Že v rámci vytvořeného prostoru jsou si entity významově blízké (slova příbuzná či ze stejné skupiny) blíže i ve smyslu geometrické vzdálenosti. Například různé druhy zeleniny budou blízko sebe, zatímco slovo vlak od nich bude výrazně dále.

Pro aplikaci například u překladačů toto ale nestačí – jazyky nejsou tvořeny jenom ze slov, významnou roli hrají i další vlastnosti, příkladně slovosled.

I tyto dodatečné informace se při užití embeddingu dají do modelu vložit, kdy pro například u transformerů je u získaného vektoru pro embedding ještě přičten vektor kódující informaci o pořadí slov.

Stručně shrnuto, embedding je metoda, jež umožňuje vytvořit nízkodimenzionální reprezentaci daných entit z korpusu ve vektorovém prostoru, přičemž vzdálenost v prostoru má vztah k míře podobnosti (typicky významová blízkost) jednotlivých entit. Přesněji se jedná o projekci řídkých one-hot vektorů do „hustého“ prostoru $\mathbb{R}^n$.

Teď již tedy víme, co se stane s daty na vstupu nejspodnějšího enkodéru. Pro konkrétnější představu se typicky jedná o vektor dimenze v řádu stovek, „binární“ hodnoty jako 256, 512, 768.

Takto *vnořená* vstupní sekvence následně postoupí do self-attention vrstvy, odkud pokračuje do dopředné *Feed-Forward sítě* a stejným způsobem prochází dalšími enkodéry (mimo embeddingu, ten, jak jsme si již uvedli, nastává, z též vysvětlených příčin, pouze na počátku celého děje).

2.3.5 Self-attention

Důležité je nepochybně taktéž pochopit koncept a fungování self-attention.

Zásadním je sada tří vektorů označovaná jako **Q**-**K**-**V** – opět z počátečních písmen anglických slov **Q**uery (dotaz), **K**ey (klíč) a **V**alue (hodnota).

Samotné vektory slouží jako hlavně jako člověku-přátelská abstrakce pro přemýšlení a výpočty pozornosti. Vnitřně systém pracuje s jejich (částečnými) lineárními reprezentacemi. Jeví se to poměrně zvláštně, ovšem nějakou dobu mi trvalo, než jsem přesně pochopil zejména rozdíl mezi vektory V a K v rámci transformerů, neboť ve zdrojích toto není (dle mého názoru) dostatečně rozvedené. Opět je to ryze osobní dojem, celkově nabývám dlouhodobě dojmu, že poměrně hojně existující komplikací u strojového učení je popis věcí „lidsky uchopitelně“ a nalézat vhodné analogie výrazně urychlující pochopení problematiky či nového (ať již zcela, či pro zatím nezasvěceného čtenáře) přístupu/postupu.

Koncept vychází z databázového přístupu při vyhledávání, nejsnazším vysvětlením je nejspíše analogie s hledáním ve vyhledávači:

- **Q** odpovídá požadovanému – pro ilustraci v tomto případě, obdržený textový vstup vyhledávače),
- **K** by byly hledaná klíčová slova – například shoda s titulkem či textem článku,
- **V** by potom byly vhodné takové články.

Pro další práci systému je nejprve vstup (typicky značený **I** – od anglického **I**nput) upraven. Vstup (**I**) je převeden následovně (do podoby tří uvedených vektorů Q, K,V):

$$\mathbf{Q} = \mathbf{I} * W^Q \tag{2.1}$$

$$\mathbf{K} = \mathbf{I} * W^K \tag{2.2}$$

$$\mathbf{V} = \mathbf{I} * W^V \tag{2.3}$$

Kde W_x jsou váhové matice pro získání příslušného vektoru. Matice, respektive jejich hodnoty, jsou důsledkem trénovacího procesu celého bloku sítě.

Celkově vypadá činnost celé sítě následovně:

(1) První krokem je sestavení těchto vektorů ze vstupních *vnořeností* (embeddingů).

(2) Následuje výpočet hodnocení, skóre. Ten vzniká jako *skalární* součin vektorů **Q** a **K**. Čím větší skóre je, tím významnější je role slova pro ostatní součásti věty.

(3) Dalším krokem je vydělení výsledných skóre druhou odmocninou velikosti vektoru **K**, výzkum totiž ukázal, že takovýto postup vede ke stabilnějšímu gradientu[28].

(4) Získané skóre je poté „znormalizováno“ skrz operaci *softmax* – tímto je v důsledku zajištěno, že všechna skóre jsou nezáporná a jejich celkový součet je 1.

Softmax je rozšířením logistické regrese do více tříd, stručně vyjádřeno, přiřazuje pravděpodobnosti ke každé třídě.

Třídami jsou v tomto případě jednotlivé entity (zde slova). Vlastností softmaxu je, že každá entita je vždy součástí právě jedné třídy.

Matematicky vyjádřeno proběhl tento výpočet:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{2.4}$$

Pro případ, že je čtenáři softmax neznámý, podrobnější vysvětlení:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad pro\ i = 1, 2, \ldots, K \tag{2.5}$$

kde x je výstupní vektor z neuronové sítě (vstup do softmaxu). Z arbitrární hodnoty v reálných číslech je získána pravděpodobnost v intervalu mezi 0 a 1.

Toto výsledné skóre říká udává míru důležitosti pro slovo na dané pozici – logicky je tedy nejdůležitější samotné slovo. Hodilo by se ovšem mít nástroj, pokud jde kupříkladu o vztažné zájmeno, který by zaručil, že velmi velkou váhu má navzájem právě i též podstatné jméno, na nějž je zájmenem odkazováno což je řešeno ještě dále dalším rozšířením popsaným dále.

(5) Posléze je každý **V** vektor přenásobený získanou hodnotou přes softmax. Tímto „zvážením“ důležitosti jsou zvýrazněna slova, kterými má smysl se cíleně, ze strany sítě „zabývat“ a která lze „vypustit“ (marginální hodnoty softmaxu blízké nule).

(6) Posledním krokem je sečtení těchto získaných vah pro všechna slova, čehož výsledkem je výstup self-attention vrstvy, což je zároveň vstup do dopředné vrstvy.

Toto je teoretický popis celého procesu zpracování vstupu v self-attention vrstvě. Ve skutečnosti se v praxi pro zrychlení výpočtů v transformerech používá maticová implementace téhož procesu.

Její výhoda je v tom, že umožňuje spočítat kroky 2 až 6 v jednom výpočtu. Embeddingy jsou „sdruženy“ do jedné matice X a tyto jsou přenásobeny natrénovanými maticemi W^Q, W^K, W^V.

Uvedený postup se ukázal jako příhodný, ale nedostatečný pro uváděný příklad zájmen a jeho vztahu k podstatnému jménu ve větě.[28]

Toto je řešeno zavedením „více hlav“ pro pozornost, přičemž každá má svůj vlastní soubor matic W^Q,w^K,W^V). To rozšiřuje schopnosti attention vrstvy o potřebnou „soustředit se“ skutečně na více věcí najednou.

Konečný výpočet pak dostává následující podobu:

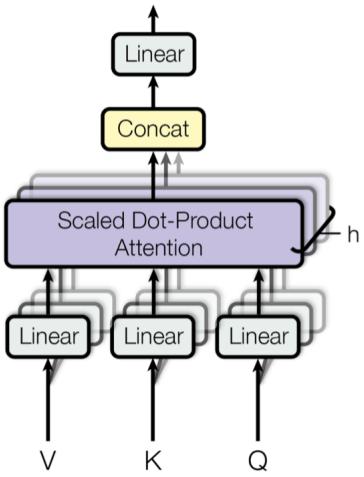
$$\begin{aligned} MHA(Q,K,V) &= Concat(h_1,...,h_h)W^O \\ h_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.6}$$

kde i odpovídá dané „hlavě“.
Zkratka MHA odpovídá MultiHeadAttention („vícehlavá pozornost“).

Podle aplikace je obvykle vyvíjeno úsilí k nalezení vhodného počtu hlav. Například v modelu wav2vec2 je konkrétně v každé attention vrstvě ve výchozích stavu hlav 12. Jsou zkoumány i vlivy počtů typu hlav v různých architekturách modelů a dopady na jejich přesnost. Ukazuje se, že volba a přínos počtu více hlav je samo o sobě „alchymií"a často se využívá jen malá část celkového potenciálu.

Proces probíhá stejně jak u dříve popsaného postupu (s jednou hlavou), takže na výstupu z vrstvy je 12 matic, z každé hlavy 1. To ovšem odporuje popsanému návrhu: další vrstva (dopředná) očekává jednu matici.

Tento problém je řešen zavedením další váhové matice W_o, která informace z těchto matic vhodně transformuje na jednu výslednou matici, značenou $\mathbf{Z}$, odpovídající východiskům pro návrh. Tato je postoupena do další vrstvy.



Obrázek 2.2: Více hlav — multihead attention

Vstupem do dekodéru je výstup enkodéru a předchozí výstup dekodéru. Dekodér vnitřně funguje obdobně jako enkodér, s jednou výjimkou:

Počítá se attention ve vztahu zdroj-cíl. Rozdílem v chování self-attention je ten, že dekodér bere v úvahu pouze (časově) předcházející tokeny, nebere narozdíl od dekodéru v úvahu všechny tokeny ve větě. Toto je dosaženo maskováním – efektivně mají takové tokeny pravděpodobnost v softmaxu 0 a tedy nemohou být vybrány.

Výstupem poslední vrstvy je opět výstupní matice, tedy čísla. Tuto matici ještě převést na cílovou reprezentaci (kupříkladu věty v cílovém jazyce pro překlad). Toto mají za úkol dvě vrstvy – výstupní lineární vrstva a úplně poslední je softmax.

Lineární vrstva je plně propojená a jejím účelem je mapovat poskytnutý výstup z dekodérů do takzvaného vektoru logitů.

Samotný termín logit představuje, programátorským slovníkem, poněkud přetížený termím. V matematice se jedná o funkci mapující pravděpodobnosti z rozsahu <0;1> do prostoru reálných čísel. V oblasti strojového učení se typicky jedná o označení pro ještě nenormalizovaný vstup modelu - tyto „surové"hodnoty se obvykle obtížně interpretují, proto jsou většinou převádějí na jinou reprezentaci.

Velikost vektoru odpovídá velikosti slovníku. Každý jeden člen vektoru odpovídá jedné entitě ve slovníku. Tento vektor je postoupen do výstupního softmaxu (představující onu změnu reprezentace), jehož výstupem jsou pravděpodobnosti. Konečným výstupem po aplikaci *argmax* na výstup *softmaxu* je index nejpravděpodobnější entity ve slovníku. Přistoupením na tento index ve slovníku je již získána kýžená výstupní entita, respektive token.

2.3.6 Stručné shrnutí procesu od vstupu k výstupu

Celý proces se dá schematicky shrnout do několika bodů.

Enkodér:

1. Vstup (token) je převeden do vytvořené lineární reprezentace (*embbedding*).
2. Tato reprezentace vstupu je rozšířena o poziční informaci (*positional embedding*).
3. Tento převedený a rozšířený vektor reprezentující původní vstupní vektor je předán enkodéru.

Dekodér:

1. Vstupem je předchozí výstup dekodéru (rekurentní chování).
2. Tento je doplněn o poziční informaci (*positional encoding*).
3. V první následné vrstvě je aplikováno maskování následných vstupů.
4. V druhé úrovni je k této informaci přidán odpovídající výstup z enkodéru – tím je doplněna informace o pozici a vztahu k vstupnímu celé větě/tokenu.
5. Výstup je popsaným postupem přes dopřednou síť a softmax zpracován a výstupem je již predikovaný token.

   Zásadním praktickým limitem transformerů je výpočetní složitost $O(n^2)$.

2.4 wav2vec 2.0

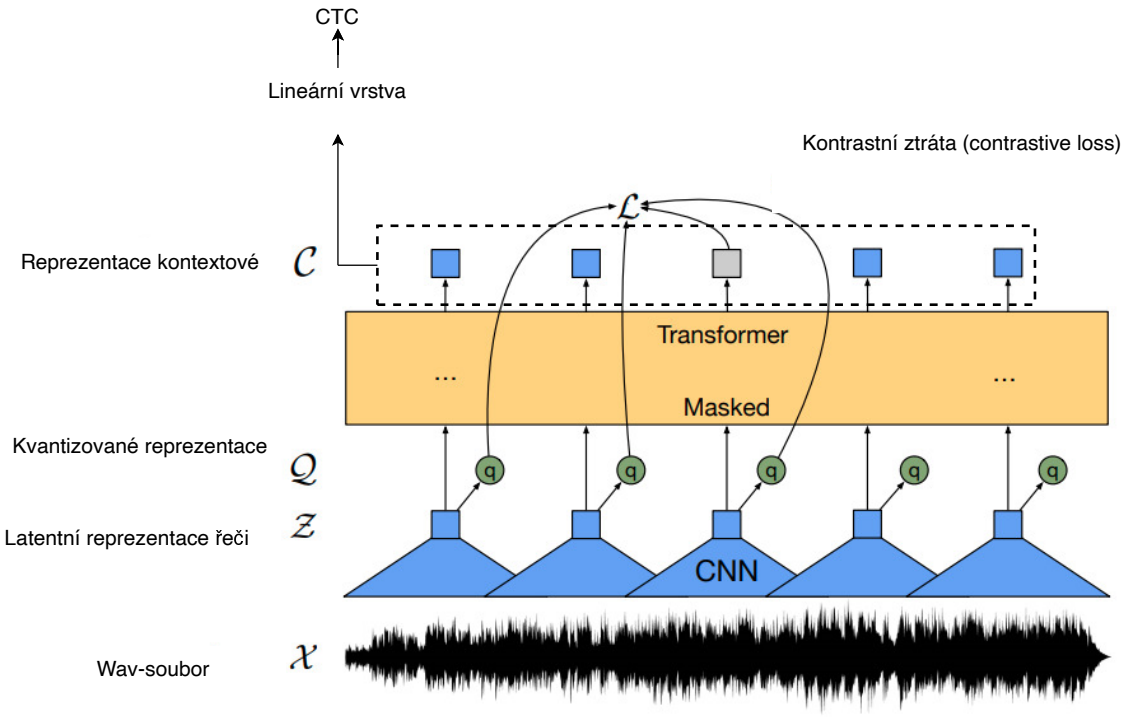
Zástupce nejmodernějších technologií pro účely automatického rozpoznávání řeči. Opravdu aktuální výzkum dokonce uvádí, že způsob zpracování řeči v rámci modelu wav2vec 2.0 je principiálně podobný tomu, jakým se učí lidskou řeč mozek, alespoň na základě pozorování pomocí funkční magnetické resonance (fMRI).[30]

Autoři došli ve studii k závěru, že stačí 600 hodin neanotovaných nahrávek ke zcela dostatečné reprezentaci jazyka, což není zásadně vzdálené od množství hodin, kterou potřebují k „pochycení“ jazyka kojenci a i korelace reprezentace řeči wav2vec2 a skutečným mozkem je poměrně vysoká.[30]

Jedná se o model s možností využít self-supervised learning a další metody, které jsme si bezprostředně popsali. Využívá se transformerů popsaných výše, s jedním rozdílem, týkající se způsobu, jak je dodána poziční informace: Byla přidána *grouped convolution layer*, která má za úkol kompenzovat fakt, že konvenční Transformer v rámci procesu self-attention nutně nezachovává pořadí prvků ve vstupním vektoru a tato vrstva předává relativní embeddingy s pozici.

Trénování obecně probíhá ve dvou fázích, i s ohledem na to, že jde o model využívající transfer learning, označované v první fázi jako *pre-training (předtrénování)*, kdy nejdříve se model natrénuje na obrovském korpusu audionahrávek bez potřeby anotace. Pre-training je časově velmi náročný, řádově trvá v případě tohoto typu modelu i desetitisíce hodin. V rámci tohoto procesu je v modelu s užitím self-supervised učení vytvořena reprezentace řeči. Zároveň se jedná o největší přínos architektury proti předchozím snahám o reprezentaci řeči v této podobě.

Ve druhé fázi se dotrénuje (*fine-tuning*) na (stačí i rozsahově malý) na anotované sadě dat, typicky vybrané tak, aby váhy v modelu lépe odpovídaly řešené úloze. Fine-tuning trvá řádově desítky až stovky hodin. Model je již „tradičně“ učen s učitelem rozpoznávat konkrétní řeči jednotky — znaky, případně slova. V řešeném případě jde o řečové jednotky odpovídající znakům.



Obrázek 2.3: Wav2vec2, včetně ztrátových funkcí, přejato a upraveno[31]

Architektura modelu vypadá podrobněji následujícím způsobem:
Celý model **po natrénování** využívá tří vrstev/bloků:

- Konvoluční vrstvy
- Transformerové vrstvy
- Výstupní vrstva — lineární

Pro pre-training fázi se navíc používá ještě blok řešící kvantizaci (reprezentováno v diagramu zeleným blokem **q**), kterou si více popíšeme dále.

Úlohou konvoluční vrstvy (konkrétně sedmivrstvé) je převod waveform (audiosignálu zakódovaného ve formátu *WAV*) do takzvaných *latentních reprezentací řeči.*

Latentní reprezentace a embedding je popis téhož principu: Efektivní reprezentace objektů v „rozumně dimenzionálním"prostoru. Příznaky jsou konkrétně určovány pro každých 20 ms, vstup je požadován se snímkovací frekvencí 16 000 Hz.

Autoři architektury vyšli z principů architektury modelů BERT [32], reprezentace řeči pro neexistenci jasných základních jednotek (její segmentace) z pohledu signálu (pproti textu) vzniká vytvořením základních jednotek o délce 25 ms. K trénování se též využívá maskování.

Tedy část vstupu transformeru (přibližně polovina) je zakrytá a cílem je, aby model dokázal „uhodnout“ tuto zakrytou část vstupu (podobně se někteří lidé učí slovíčka v jazycích ze slovníku, zakrývají si zdrojové nebo cílové slovo a hádají to či ono, je to principiálně velmi podobné).

Jak přesně je toto učení realizováno v případě pre-trainingu modelu s architekturou wav2vec2.0? Vstup je převeden dvěma odlišnými způsoby na 2 různé reprezentace. Model se následně trénuje na rozpoznání toho, zda-li se tyto reprezentace představují skutečně stejnou řečovou jednotku.

Ve wav2vec2.0 se jedna reprezentace tvoří pomocí transformerů a druhá se tvoří kvantizací. Pojem kvantizace označuje mapování spojitého prostoru (například reálná čísla) na konečný počet hodnot v diskrétním prostoru (kupříkladu celá čísla určitého rozsahu).

Myšlenka využití vychází z toho, že každý jazyk je tvořen konečným počtem fonů. Úkolem modelu je naučit se pro zamaskovanou kvantizovanou reprezentaci najít tu správnou.

Jedním z přínosů wav2vec2 je vyřešení úlohy automatického naučení se „základních“ řečových jednotek, neboť ty narozdíl od písmen latinky nejsou jasně dané (textová reprezentace zcela neodpovídá, jeden textový znak mnohdy reprezentuje více fonů, konkrétní závisí na okolním kontextu). Použití lidmi rozlišovaných fonémů by znemožňovalo použít self-supervised learning, neboť tyto model nezná. Maximálně je schopna se síť naučit až 102 400 řečových jednotek.

Zkoumány byly i možnosti, jak model využít pro ohrožené jazyky, kde je k dispozici příliš málo dat pro aplikaci self-supervised learning. Ukazuje [33] se, že nejen pro takovéto jazyky dobře funguje použít data z více jazyků, což vede k vytvoření lepší reprezentace jednotek v modelu.

Kvantizace zvukových jednotek probíhá přenásobením získaných latentních reprezentací přes váhovou matici pro kvantizaci, čímž je získáno výsledné skóre pro spojení řešené jednotky se slovy v kódové knize.

Trénování probíhá tak, že se zhruba polovina příznaků vektoru příznaků zamaskuje a výstup transformeru se porovná s druhou reprezentací. Výsledná chyba z tohoto procesu při porovnání se skutečností je označována jako constrastive loss.

Kromě contrastive loss je v modelu využívána ještě další ztrátová funkce, diversity loss, jejímž cílem je zajistit, aby byly využívány jednotlivé záznamy v kódové knize využívány co nejrovnoměrněji. [34]

Celková hodnota ztrátové funkce je potom rovna součtu contrastive a diversity loss.

Při druhé fázi trénování, fine-tuningu, se kvantizace již nepoužívá, místo toho se využívá lineární vrstva s použitím CTC-loss — ztrátové funkce, kterou jsme nyní popíšeme.

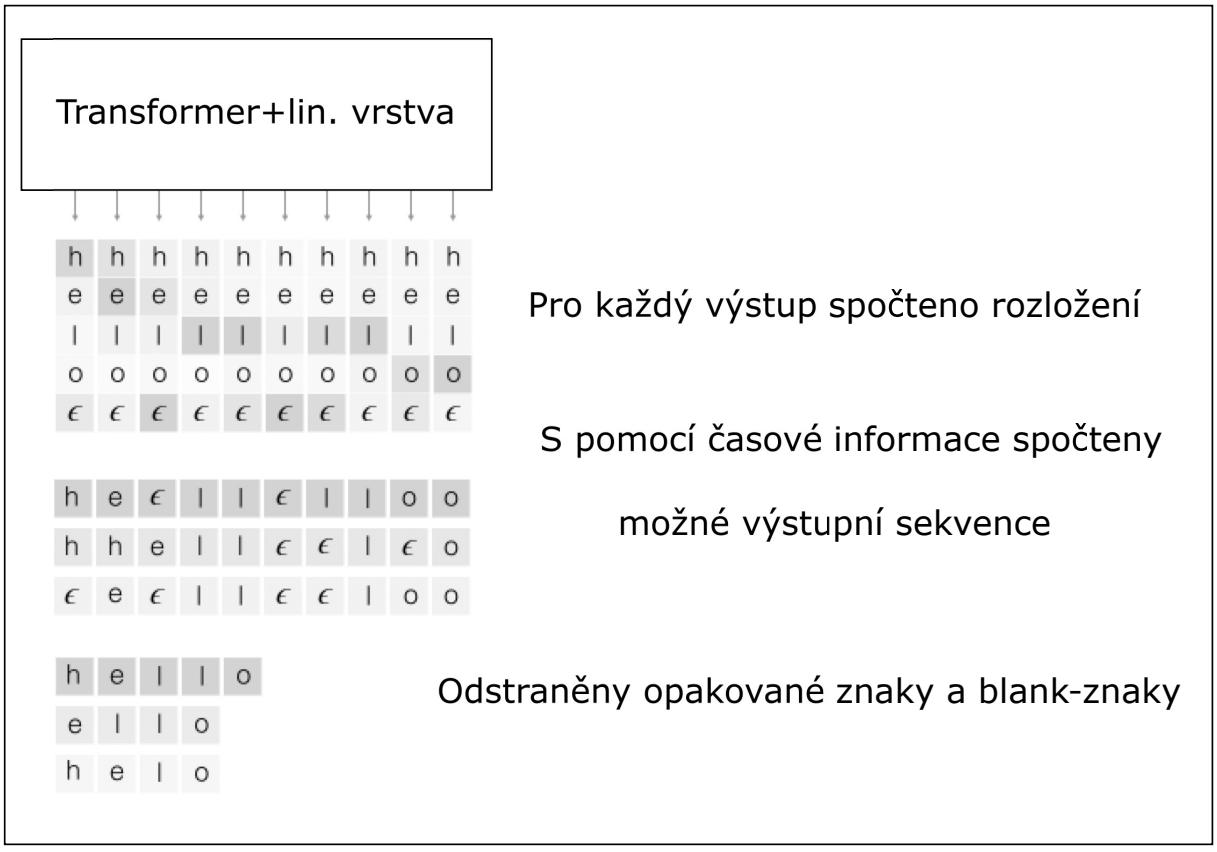
2.5 CTC

Pod zkratkou CTC se skrývá název „Connectionist Temporal Classification ". Název techniky se nejspíše nepřekládá (minimálně jsem v rámci své rešerše nenalezl etablovaný standardní termín), bude to pravděpodobně neexistencí termínu již pro samotný „Connectionism“, což je obor kongnitivních věd snažící se popsat mentální děje právě s užitím umělých neuronových sítí. [35]

Pokus o překlad by mohl znít „konektionistická časově závislá klasifikace“, což stále není příliš vysvětlující.

Velmi stručně, ne zcela přesně řečeno, CTC se zaobírá úkolem, jakým způsobem přejít z predikce jednotlivých rámců (*frames*) zvukového souboru (přičemž tyto jsou za sebou uspořádány v čase závisle za sebou), pomocí nichž (v tomto případě) je zvuk reprezentován (kdy jeden frame má při vzorkovací frekvenci 16000 Hz trvání $\frac{1}{16000}$ sekundy) na predikci (v tomto případě) cílových jednotek – fonů (různě dlouhých) a to nejvýhodnější cestou. [36]

Pro hrubou názornou představu o funkci CTC je nejlepší vizualizace v podobě obrázku:



Obrázek 2.4: Ilustrace CTC, převzato a upraveno z [36]

Příčinou generování násobných výstupních symbolů je, že každý jednotlivý fon je

rozložen přes více rámců (například každý řečník mluví jinou rychlostí).

2.5.1 Motivace

Rozeberme si výše uvedenou intuitivní motivaci podrobněji: Důležitým faktorem a současně prvním problémem k řešení při rozpoznávání časově závislých sekvencí obecně je mapování jednotlivých tokenů. Tokeny ve vstupní sekvenci (například úseku zvuku uchovaného ve *.wav souboru) jsou mapovány na odpovídající výstupní (například textovou) reprezentaci (tj. znak, písmeno).

Další příkladem je namapování rukou psaného písma na strojovou reprezentaci daných znaků. CTC umožňuje přímo neřešit zarovnávání (například délka jednotlivých fonů se liší) vstupu na výstup. [36]

Kdybychom si měli shrnout, jaké problémy se konkrétně při mapování sekvence $\mathbf{X}$ na $\mathbf{Y}$ vyskytují a výrazně komplikují využití jednodušších přístupů, dojdeme k následujícímu:

- Délka vektorů $\mathbf{X}$ a $\mathbf{Y}$ se může různým způsobem lišit.
- Poměry délek vektorů $\mathbf{X}$ a $\mathbf{Y}$ se mohou také lišit.
- Důsledek: Nemáme jednoznačné zarovnání/mapování (vizte výše).

Algoritmus CTC tyto potíže umožňuje překonat. Výstupem pro vstup $\mathbf{X}$ je rozložení přes všechna možná $\mathbf{Y}$. Tato rozložení je možné využít ke dvěma věcem:

- Buď: Lze odvodit (dedukovat) pravděpodobný žádoucí výstup.
- Nebo: Zhodnotit pravděpodobnost daného výstupu pro užitý vstup.

2.5.2 Popis algoritmu

Jak je z popsaného již nejspíše zřejmé, algoritmus CTC nevyžaduje zarovnání vstupu a výstupu. K získání výstupu pro daný vstup CTC provádí sumaci pravděpodobností všech možných zarovnání.

V rámci reálného použití typicky existuje několik různých variant, jak na sebe navzájem vstupní a výstupní vektor namapovat.

Kupříkladu při rozpoznání řeči je nutné se vypořádat se skutečností, že se (míněno časově, dobou trvání) délka jednotlivých hlásek liší, tedy tím i množství množství rámců, které se z řečového signálu mapují na výslednou ortografickou (písmennou) reprezentaci.

Nabízí se využít jednoduché řešení. Redukce všech těchto rámců, příslušných k danému písmenu, na onu příslušnou výstupní reprezentaci. To dobře funguje pro běžná slova.

Řeč se ovšem neskládá pouze z fónů různých druhů přímo reprezentujících jejich psanou podobu: V řeči jsou například pauzy v promluvě. Tyto mapovat mnohdy není žádoucí, nenesou obsah, v psané podobě je obvykle přímo nereprezentujeme[1] a jazyk se neskládá pouze ze slov, jež mají za sebou vždy různé hlásky. Mnohá slova mají za sebou dvě stejná písmena (pro názorný příklad se podívejme na slovo *proměnná*), tedy užití popsaného způsobu by vedlo k získávání nesprávných výsledků.

Tento jev se v rámci CTC řeší zavedením speciálního **blank** (prázdného) **tokenu** značeného symbolem ϵ. Ten neodpovídá žádnému reálnému znaku a na výstupu je vypuštěn. U slov s výskytem stejného znaku za sebou je nutné, aby jednotlivé výskyty byly právě tímto tokenem odděleny.

2.5.3 Důležité vlastnosti CTC

Řešení úloh za použití CTC je ovlivněno několika je vlastnostmi této metody. Ty hlavní jsou shrnuty v následujících odstavcích.

2.5.4 Zarovnávání (alignment) – shrnutí, omezení

Alignment v CTC má několik zásadních vlastností:

- Zarovnání jsou monotónní – to znamená, že když přejdeme k dalšímu vstupu, odpovídající výstup bude buď shodný jako předchozí, nebo bezprostředně další. CTC tedy nelze užít k řešení úloh, kde není garantována souslednost (příkladem budiž strojový překlad a odlišný slovosled).
- Vztah mezi zarovnáními X -> Y je několik na jednoho (*many-to-one*). Polopaticky: Jeden prvek vstupního vektoru X může odpovídat jednomu nebo několika.
- Poslední vlastnost vyplývá z předchozího: Délka Y <= X.

Tato poslední jmenovaná vlastnost může být rozhodující o tom, zda-li lze pro danou aplikaci CTC využít: Výstup nemůže mít více kroků než vstup.

Pro mnohé aplikace je praktickou komplikací, že může existovat velmi velké množství možných zarovnání. Pro výpočet ztrátové funkce se proto často používají techniky dynamického programování.

Nezbytnou součástí k úplnému pochopení fungování CTC je ještě potřebné vědět, jak se konkrétně získá výsledná hodnota ztrátové funkce.

[1] Pro upřesnění: některé pauzy v jazyce sledujeme, například čárky a tečky ve větách, mají ale i jiné významy, důležité například pro kontext či vyznění promluvy — celkově je problematika pauz v jazyce celkově překvapivě rozsáhlým tématem, např.: `http://sas.ujc.cas.cz/archiv.php?lang=en&art=3096`

2.5.5 Ztrátová funkce

Pro pár vstup:výstup (**X**,**Y**) je obecně potřeba najít řešení s největší pravděpodobností, platí rovnice[36]:

$$p(Y \mid X) \tag{2.7}$$

$$\prod_{t=1}^{T} p_t\left(a_t \mid X\right) \tag{2.8}$$

Pro každý alignment je spočítána pravděpodobnost po jednotlivých krocích.

A následně je spočítán nejpravděpodobnější alignment:

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \tag{2.9}$$

kde A jsou možné alignmenty (zarovnání) a poskytovaný výstup odpovídá využití nalezeného nejvhodnějšího kandidáta:

$$Y^* = \underset{Y}{\operatorname{argmax}}\, p(Y \mid X) \tag{2.10}$$

což obvykle, včetně popisované aplikace, odpovídá zarovnání (alignmentu) s největší pravděpodobností:

$$A^* = \underset{A}{\operatorname{argmax}} \prod_{t=1}^{T} p_t\left(a_t \mid X\right) \tag{2.11}$$

Z tohoto mezivýsledku jsou pak odstraněny blank tokeny ϵ a vyřešena opakování, čímž je získán cílový výstup **Y**.

Rovnice 3.1–3.5 jsou přejaty z [36].

2.5.6 Podmíněná nezávislost

U CTC se implicitně předpokládá, že každý výstup je nezávislý na ostatních výstupech pro daný vstup. Jasnější je význam a důsledky po vysvětlení na příkladu z oblasti rozpoznávání řeči: Vezměme si promluvu „tři A“. Korektní transkripcí by mohlo být *tři A* i *AAA* . Jevilo by se tedy logickým, že pokud začíná predikovaná transkripce písmenem A, budou na následující pozici s velmi vysokou pravděpodobností CTC „uvažovat“ další *A*. Takto tomu ale není, právě kvůli principu vzájemné nezávislosti [37]. Tato vlastnost může být ale i výhodnou. Podrobněji je toto popsáno v odkazovaném článku.

2.5.6.1 Vztah s HMM

Na první pohled se může zdát, že se jedná o značně odlišný přístup výrazně se lišící od skrytých Markovových modelů (HMM). Ve skutečnosti jde o podobné procesy, existují práce ukazující, že trénování s použitím algoritmu CTC je speciálním případem trénování HMM [38].

Hlavním rozdílem je ve výsledku skutečnost, že CTC je diskriminativní, tedy přímo rozděluje do tříd.

Softwarová implementace CTC je poměrně náročná, například výpočet ztráty způsobem je numericky nestabilním; tento a další problémy jsou řešeny různými optimalizacemi v jednotlivých časových krocích.

Kapitola 3

Řešená úloha

Vynořivší se vhodnou aplikací neuronových sítí k širšímu prověření byl vliv druhu a reprezentace užitých dat pro fine-tuning na výslednou přesnost predikce modelu pro automatické rozpoznávání mluveného jazyka, kdy natrénovaný model je součástí širšího bloku – výstup slouží jako vstup pro modul SLU (*S*poken *L*anguage *U*nderstanding) – porozumění mluvenému jazyku.

Pro porovnání vlivu využitých dat v rámci fine-tuningu na finální výsledky modelu v popsané aplikaci byly vytvořeny trénovací balíčky pro fairseq (bude popsáno podrobněji dále) s užitím tří základních typů dat:

- CommonVoice data (wavy + přepisy)
- Ortografické přepisy + HHTT wavy
- Normované (spisovné) přepisy + HHTT wavy

Mezi datasety jsou dva základní principiální rozdíly: CommonVoice je reprezentací obecného jazyka, HHTT data jsou reprezentací výrazně specifického jazyka (podrobněji bude popsáno v další části). Intuitivně lze i na základě předchozího předpokládat, že model laděný na specializovaném jazyku z téhož prostředí by mohl být, v rámci klasifikace promluv, model přesnější. O jak výrazný rozdíl se bude jednat? Toť otázka k zodpovězení.

Zároveň český jazyk není jednolitým „zmraženým“ komunikačním prostředkem – existuje sice formalizovaný spisovný jazyk, ale většina komunikace probíhá v obecné češtině (i vlivem historicky podmíněných rozhodnutí kupříkladu obrozenců, kteří se rozhodli kodifikovat jazyk v podobě již v době kodifikace lehce zastaralé (bible kralická) [39] a jde v nemálo aspektech odchylný jazyk od běžně užívaného.

I proto je nutné mu (nejen) na základní škole věnovat ono vysoké množství vyučovacích hodin, které mu je věnováno)[40]. Další otázkou k zodpovězení: Je lepší model trénovat přímo z obecného (v ortografické reprezentaci) jazyka, nebo ze spisovného?

A případně pomůže k lepším výsledkům vůči referencím kombinace obou druhů dat a jaký bude posléze dopad „normování“ výstupu z rozpoznávače na zvýšení přesnosti?

V jednom z modelů k porovnání byla využita i kombinace přístupů a zdrojů anotovaných dat – model nejprve natrénovaný na obecném jazyce z CommonVoice dat byl znovu finetunován ještě s užitím spisovné reprezentace dat nahrávek z korpusu HHTT.

Cílem bylo ověřit předpoklady na různých modelech a zjistit, jak závisí výsledná přesnost rozpoznávače na datech, na nichž byl wav2vec2 model trénován a na formě jejich reprezentace.

Motivací k experimentu byla právě dříve popsaná skutečnost, že spisovná a obecná čeština jsou do značné míry odlišnými „podjazyky“ — strukturou vět jsou stejné, ale slovní zásoba a jejích tvary se někdy i citelně liší, přitom systém by měl být schopen zvládnout obě.

Hlavními složkami experimentů bylo ověřit, zda-li model natrénovaný z obou typů reprezentace dat současně bude dosahovat lepších výsledků při evaluaci, další bylo vyzkoušet, jaká reprezentace dat je pro tento účel příhodnější.

3.1 Lidská řeč

Dovolím si malou lingvistickou odbočku do společenských věd se technickými detaily. Lidská řeč a jazyky jsou samy o sobě mimořádně zajímavými – jak ukazuje výzkum, i přes značné rozdíly v jazycích všechny sdílejí, i z ryze technického pohledu, některé charakteristiky, které jsou v technickým aplikacích minimálně pozornosti hodnými a někdy dost možná i podstatnými pro různé konkrétní technické aplikace.

Kupříkladu i přesto, že existují obrovské rozdíly mezi tím, jaké prostředky jsou ke kódování informací použity, od věcí typu slovosled, počty rodů a další, využívání vlastností typu tónovost a obdobně, všechny jazyky dospívají k podobnému průměrnému datovému toku – zhruba 39 bitů za sekundu. [41] To je necelých 5 bajtů za sekund, což není mnoho, ale očividně to je prostředek postačující na mnohé (pro představu, uvádí se [42], že celková kapacita lidského vědomého vnímání je zhruba 120 bitů za sekundu, tedy řeč a její zpracovávání je kognitivně velmi náročný proces).

Řečníci v jednodušších jazycích (s menší informační hustotou) hovoří typicky rychleji a naopak, ale nejsou přítomny žádné extrémy – jak uvádějí autoři v diskusi, příliš vysoké tempo řeči je fyzicky velmi náročné pro řečníka (udržení artikulace a srozumitelnosti současně s promýšlením samotného obsahu promluvy), tak pro posluchače (přehlcení kapacity vnímání celkově nebo alespoň zpracování obsahu). Na druhou stranu pomalá řeč s vysokou (z výzkumu vyplývá například důležitá informace pro návrháře hlasových dialogových systémů, totiž že v typické přirozené mezilidské neformální konverzaci trvá

obrátka 2 sekundy[43][44].

Informačně hustější jazyk navíc generuje větší náročnost pro krátkodobou/pracovní paměť (na zpracování aktuálního kontextu), což je nejspíše důvod, proč se tomuto druhému extrému lidé také vyhýbají a podobný jazyk se neužívá.

Každý jazyk je tedy v nějakém aspektu unikátní systémem (mezi)lidské komunikaci, který se po dlouhou řadu let samovolně vyvíjí, přesto ve výsledku jsou všechny v úhrnu poměrně podobně efektivní, z čehož vyplývá, že jazyk je formován mnoha faktory, například prostředím, společenskými potřebami, ale ve výsledku je též omezen biologicko–technickými faktory.

Je svým způsobem fascinující, že kvůli takto zdánlivě nízko-datový přenos informací je třeba tvořit takto komplexní modely a postupy pro aspoň částečné porozumění.

3.2 Důležité pojmy a nástroje

3.2.1 fairseq

Fairseq je sada nástrojů založená nad frameworkem Pytorch, jejímž cílem je usnadnit vývojářům a vědeckým pracovníkům trénování modelů sloužícím zejména pro práci s (textově vyjádřeným) jazykem v potřebných/řešených úlohách.[45]

Poskytuje předtrénované modely založené na různých architekturách a přístupech, zejména na dříve popsaných CNN, LTSM sítích a Transformerech určených k další práci a umožňuje začleňovat i vlastní modely, zpřístupňuje též různé pomocné nástroje, například pro výpočty ztrátových funkcí a datové sady.

Věcí mající potenciál stát se značnou komplikací, kterážto se obecněji týká všeobecněji většiny nástrojů pro strojové učení a je dána i rychlým vývojem, se ukázala kompatibilita mezi jednotlivými verzemi (dochází ke změnám definic či názvů parametrů a podobně), přesněji nepřítomnost kompatibility zpětné. Je tedy nezbytné například vědět, jaká verze byla využívána například pro trénování před-trénovaného modelu.

3.2.2 HuggingFace

HuggingFace je společnost, jež původně začala jako vývojář a poskytovatel chatovací aplikace a služeb. Po čase existence se rozhodla, že chce vyvíjet nástroje pro zpracování přirozeného jazyka a to modelem „open-source, open-science“.

Za tímto účelem se ji podařilo vybrat 15 milionů dolarů a svou vizi plní, je jakýmsi „uzlem“ pro pestrou škálu nástrojů pro strojové učení a různé s ním spjaté aplikace.

„Horkou novinkou“ s (domnívám se) významnými dopady do budoucna je oznámená spolupáce společností Microsoft a HuggingFace, kdy Microsoft již nyní nabízí možnost

využít na cloudové platformě Azure hostované modely, což mnohým výrazně usnadní integraci takovýchto modelů do svých tam provozovaných softwarových řešení a zároveň umožní v případě potřeby snadno škálovat dostupný výkon.[46]

3.2.3 Pretrained model ClTRUS

Výchozím základem je předtrénovaný model CITRUS (**C**zech **l**anguage **TR**ransformer from **U**nlabeled **S**peech), tedy v českém jazyce „Českojazyčný transformer z neatonované řeči“ [47]. Jde o model natrénovaný na datech o rozsahu zhruba 80 tisíc hodin řeči, získaných z rozličných dostupných zdrojů. Největší podíl tvoří nahrávky z rádia a datasetu VoxPopuli (dohromady přibližně $\frac{1}{2}$ rozsahu).

3.3 HHTT (Human-Human Train Timetable) korpus

Datovou sadu využitou pro vyhodnocování je testovací část (*test-split*) korpusu Human–Human Train Timetable, což je datový soubor dotazů a odpovědí vztahující se k vlakovým spojením. Jedná se konkrétně pro test část o 1439 vět/souvětích. Korpus HHTT se skládá[48] z 7249 vět rozdělených následujícím způsobem:

- train: 5240
- dev: 570
- test: 1439

Jde o poměr rozdělení 72:8:20 (train:dev:test).

Na okraj lze pro zajímavost uvést, že pročítání/poslech dat z korpusu HHTT je současně i jistým výletem do (železniční) historie – obsahuje totiž některé prvky a konverzace o různých aspektech spjatými s cestováním vlakem, na které v současnosti cestující již nenarazí – kupříkladu veškeré příplatky za vlaky vyšší kvality (rychlíkové, InterCity/EuroCity), o nichž část promluv pojednává, byly s platností od prosince 2007 definitivně zrušeny. [49]

3.3.1 Trénovací data – audio

Trénovací data byla více způsoby „pročištěna“ a upravena tak, aby bylo možné zrealizovat fine–tuning: Konkrétními zásahy bylo zejména:

- Sjednocení parametrů nahrávek na úroveň očekávanou modelem (zejména původní vzorkovací frekvence souborů byla z 8000 Hz navýšena na 16000 Hz, což je technickým

požadavkem daným návrhem modelu C1TRUS [47], respektive samotného wav2vec2). Tento proces, nazývaný upscaling, je korektní, neboť zvýšením vzorkovací frekvence není ztracena žádná informace oproti původní podobě dat.

- Datový balík byl nejprve pročištěn o vzorky delší než 30s, což by měla být maximální podporovaná délka s ohledem na dostupnou paměť GPU; občasně jsem ovšem i tak narazil na chybu přetečení během průchodu backward-forward algoritmem při trénování, po zkoušení různých pokusných úprav jsem nakonec přikročil ještě ke konečnému protřídění na délku do 16s (zásah v poměru k celkové velikosti datasetu není ta citelný, do 2 %). Tímto se problémy přestaly vyskytovat.

Po těchto zásazích tedy trénovací data sestávají z 70473 audio-souborů nahrávek o vzorkovací frekvenci 16000 kHz. K nim jsou příslušné jejich anotace:

3.3.2 Ortografická reprezentace dat

Jedná se o přepisy nahrávek tak, „jak jsou“, tedy obecnou češtinou (například s koncovkami -ej, příkladně slovo „*kterej*“) s doplňkovými informaci o některých dějích (nádechy, neindentifikovaný hluk a další, zejména spjatými s řečí – nádech, vložené „hmm, ehm“ a další bezobsažné „promluvy“.).

Příkladové řádky takovéhoto souboru:

```
1 CD03~Part1~04-000427-103656-1_008    padesát tři _noise_ a je to na domažli
2 CD03~Part1~04-000427-161350-1_003    _spk_inhale_ v kolipa tak hodin
```

S ohledem na to, že tato doplňková metadata nereprezentují řečové promluvy a tedy vedou následně k chybám při aplikaci CTC (reprezentace neodpovídají skutečným a žádoucím hláskám na výstupu), jsem z tohoto souboru vytvořil ještě jede, z něhož je tento druh dat odstraněn. Příkladové řádky v souboru po provedení popsané úpravy vypadají následovně:

```
1 CD03~Part1~04-000427-103656-1_008    padesát tři a je to na domažli
2 CD03~Part1~04-000427-161350-1_003    v kolipa tak hodin
```

3.3.3 Spisovná (normalizovaná) reprezentace dat

Jedná se o přepisy týchž nahrávek, ovšem ve zcela spisovné podobě, bez chyb (chybějící slabika) a bez pomocných (meta)informacích o dějích uvedených výše. Soubory vypadají konkrétně takto:

```
1 CD03~Part1~04-000427-103656-1_008    padesát tři a je to na domažlice
2 CD03~Part1~04-000427-161350-1_003    v kolikpak tak hodin
```

3.3.4 CommonVoice data

CommonVoice je bezplatná databáze (a zároveň název pro dataset) řečových vzorků, jejíž vznik byl iniciovaný organizaci Mozzila (stojící mimojiné například za prohlížečem Firefox).

Je snahou, minimálně v angličtině explicitní, aby obsahovala širší spektrum řečníků – většina projektů dle správců nadreprezentová řečníky bez výrazného akcentu a muže – dáno je to nejspíše nejčastějším zdrojem nahrávek, jímž jsou často rádio, televize a podobně, kde jsou tyto jevy obvykle potlačovány pro srozumitelnost (minimálně u moderátorů); ostatně za sebe hovoří samotná čísla uváděná u jednotlivých datasetů i právě v rámci CommonVoice. V současné době jde o nejobsáhlejší projekt takového typu. [50]

Součástí každého souboru v datesetu jsou proto i metadata, specifikující přízvuk, věk, pohlaví, pro případ, kdy by s těmito příznaky chtěl vývojář dále pracovat.

V mém případě má užitý dataset z CommonVoice rozsah 34875 nahrávek.

3.4 Trénování

3.4.1 Fine-tuning — příprava dat

Pro účely fine-tuningu je z textových a audiodat mimo dalších kroků sestavit korektní manifest, což je sada souborů popisující zejména rozdělení balíku dat na trénovací a validační části (tyto soubory jsou ve formátu TSV – Tab Separated Values) a k nim příslušné textové reprezentace audio-souborů (textové přepisy, uloženo jako běžné prosté textové soubory formátu *.txt*), kdy index řádku v manifestu (název souboru) číselnou hodnotou odpovídá příslušné nahrávce.

Pro rozdělení (split) datasetu je nejlepší použít skript dostupný v rámci repositářů projektu.

Příklad řádku v souboru typu **.wrd* (word):

```
buďte tak moc hodná chtěla sem se zeptat ...
... jesli jede ten ex rychlík třináct nula osum na prahu
```

Příklad řádku v souboru typu **.ltr* (letter):

```
v e | č t r n á c t | d e s e t | z | p l z n ě | d o | s t ř í b r a |
```

Poslední nezbytnou komponentou je slovník (dict). Ten obsahuje počty znaků v trénovací (train) části trénovacího datasetu. Znak *svislice* (tj. |) plní roli mezery.

Příkladové řádky takovéhoto souboru:

```
| 910120
e 407320
t 321698
```

3.5 Rozpoznávač na bázi Wav2Vec2

Pro účely hlasových dialogových systémů je nezbytnou fundamentální součástí rozpoznávač mluvené řeči — pro jakoukoli další činnost v dalších úrovních dialogového systému je nezbytností dosahovat co nejpřesnějšího rozpoznávání. Dnešní počítačové systémy pracují s informací v binární podobě, přičemž pro člověka jde nejčastěji o řetězce či čísla – kupříkladu pro volání webového API vyplývajícího z uživatelova požadavku nelze využít přímo soubor se záznamem *.wav promluvy. Tvořený wav2vec2 model slouží právě tomuto účelu, přičemž obecně prozatím není s ohledem na výpočetní náročnost možné nasadit takovéto rozpoznávače za účelem jejich využití při komunikaci v reálném čase (i na výkonných výpočetních uzlech Metacentra trvá čistě rozpoznání jedné promluvy více jak jednu sekundu).

3.5.1 Normování výstupu rozpoznávače

U dat získaných z rozpoznávače je při vyhodnocení porovnávána „surová“ verze výstupu a normovaná. Účelem normování výstupu je [48] omezit míru zkreslení při vyhodnocování, přítomnost tohoto zkreslení je dána variabilitou vyjadřování mluvčími jazyka. Autoři rozpoznávače[48] totiž během jeho vlastní tvorby a evaluace zjistili, že výrazné množství chybných (přesněji takto označených) rozpoznání bylo ryze na úrovni záměny slova – často šlo totiž o výrazy v obecné místo spisovné češtině, kdy se slova sice liší na úrovni „vzhledu“ (formou), významem se ovšem neliší. Účelem je toto narovnat. Příkladem takovéto záměny je například slovní spojení *na shledanou*, přičemž v mluvené řeči tomuto odpovídá (pravděpodobně nikoli zcela vyčerpávající) výčet slov: *naschledanou, nashledanou, na schledanou, nashle, naschle, nashledle, nshledanou*. Vlivem normalizace se ze všech těchto sémanticky ekvivalentních variant stane jediné spisovné „na shledanou“ a je tak získána užitečnější informace o přesnosti rozpoznávače pro použití v reálném prostředí.

3.6 Fine-tuning modelů

Všechny modely (s výjimkou tak označeného) zde vyhodnocované byly natrénovány za užití stejných parametrů trénování, pouze z odlišné reprezentace dat (dle popisu uvedeného

u popisu datasetů).

3.6.1 Fine-tuning na CommonVoice–datech

Model natrénovaný z obecných Commonvoice dat.

Hlavní parametry modelů pro trénovací proces pomocí fairseq-train, pokud není uvedeno jinak, byly následující:

- max-tokens: *1600000* - tato hodnota určuje, kolik tokenů je načteno v jedné testovací dávce.
- update-freq: *16* - tento parametr určuje mění práci s gradientem mezi jednotlivými dávkam.
- max-update: *80000* - Počet trénovacích cyklů.
- best-checkpoint-metric: *wer* metrika po hodnocení trénování.

3.6.2 Fine-tuning na ortografických HHTT–datech se zachovanými pomocnými daty

Jde o model trénovaný z HHTT-dat v transkripční podobě, včetně zachovaných pomocných dat o nádeších a podobně.

3.6.3 Fine-tuning na ortografických HHTT–datech

Tentýž model jako předchozí, pouze s odstraněnými pomocnými daty.

3.6.4 Fine-tuning na spisovných HHTT–datech

Zvuková data jsou stejná, jejich textová reprezentace je spisovná a úplná (bez chybějících slabik a podobně) a bez metadat.

3.6.5 Fine-tuning na spisovných HHTT–datech – odlišné parametry trénování

Pro porovnání jsem vyzkoušel i vliv změny training rate.

Očekávaným důsledkem bylo výrazně delší trénování, přínosem mohou a nemusejí být lepší výsledky. S ohledem na výpočetní náročnost trénování modelů považuji ale problematiku „jakým způsobem a jak dlouho trénovat“ za zajímavou (a i s ohledem na vývoj ceny energií tato problematika bude možná nabývat na významu) a je dlouhodobě

zkoumána, například v [51]. V případě jednoho z testovaných modelů s velkými transformery se podařilo zkrátit dobu trénování o 80 %. Konkrétně došlo ke zvýšení *–update-freq* z hodnoty *16* na *22*.

3.6.6 Opakovaný fine-tuning CommonVoice na spisovných HHTT–datech

Původní myšlenkou bylo otestovat následující: Model vzniklý dvojnásobným fine-tunováním – výchozím je model popsaný v části *Fine-tuning na spisovných HHTT–datech*, přičemž tento model byl stejným procesem fine-tunován ještě jednou, tentokrát na spisovných datech HHTT. Intuitivně by se dle předchozích výsledků modelů založených na „specializovanějších“ datech dalo očekávat, že tento model bude dosahovat lepších výsledků než čistý CommonVoice.

U tohoto modelu jsem využil strukturu s modelem novější verze fairseq, u níž jsem narazil na problém s nekompatibilními balíky. Poučení z toho pro mne plyne takové, že součástí archivů při sdílení rozpracovaných modelů by mělo uvážit přiložit soubor popisující verze použitých balíků a to nejen verzí, ale mnohdy, zejména u fairseq, přesné verze užitých verzí (definované přes commit-hash z repozitáře projektu GitHubu). Vhodným téměř garantovaně funkčním řešením je nejspíš obraz kompletního prostředí (kupříkladu Docker), je otázkou, zda-li je toto při zvážení veškerých důsledků, nároků na úložiště počínaje, opravdu ideálním postupem, ve výsledku bude záležet na zhodnocení dle okolností a dispozic uživatele.

3.6.7 Sdružený fine-tuning na HHTT-datech v obou reprezentacích

Posledním modelem vznikl provedením fine-tuningu z dat o obou formách jazyka současně, tedy s přepisy ortografickými i spisovnými současně. Předpokladem je, že tento model bude nejlepší. Je tomu skutečně tak?

Kapitola 4

Evaluace výsledků

Získané výsledky různých modelů je nutné nějakým způsobem mezi sebou moci porovnat. K tomuto lze použít různé metriky. V rámci celého procesu přípravy modelu je různé hodnocení prováděno na různých místech. Významným je zejména vyhodnocování při trénování (na nějž reaguje toolkit fairseq automaticky) a poté vyhodnocení výsledků různých přístupů pro řešenou úlohu.

4.1 Metrika vyhodnocování trénování (fine-tuning)

V rámci finetuningu se vyhodnocují výsledky jednotlivých epoch trénování za pomoci metriky WER.

4.1.1 Word Error Rate (WER)

Word Error Rate (*WER*) (česky nejlépe přeložitelné nejspíš jako „míra chybovosti slov") je v oblasti rozpoznávání řeči běžně užívaná metrika, odvozená od Levenshteinovy vzdálenosti. Levensteinova vzdálenost je metrikou míry odlišnosti mezi 2 řetězci. Algoritmus k jejímu výpočtu je mimochodem dokázán jako nejefektivnější možné řešení tohoto problému. [52]

Odlišností WER proti Levenshteinově vzdálenosti je, že základní hodnocenou jednotkou je **slovo** namísto fonému. Velmi užitečná je zejména pro výsledné hodnocení inkrementálních vylepšení v rámci jednoho systému nebo pro porovnání mezi dvěma různými systémy.

Hodnocení se skládá ze tří hodnocených druhů chyb, ty jsou označovány typicky jako chyby typu **S**, **I** a **D**, přičemž pod písmeny se skrývá následující:

1. Substituce (**S**ubstitution): Záměna slova za jiné, například „umyl“ a „ubil“
2. Vložení (**I**nsertion): ve větě je vloženo slovo, které nebylo řečeno.

3. Smazání: (**D**eletion): ve větě je vynecháno slovo. Příkladem je například rozpoznání „Byl lese"namísto „Byl v lese"(vynechané „v“).

Zásadním faktorem, jenž je nutné mít při vyhodnocování dle WER metodiky neustále na paměti, je skutečnost, že sice můžeme sledovat, jaká je celková úspěšnost rozpoznání a jaký druh chyb se děje (pokud je nějaký jev systematicky se pravidelně objevující, lze s tímto dále pracovat), ale neznáme příčinu, zdroj chyby, je nutné použít další nástroje.

O jaké jevy konkrétně může příkladem jít:

- Technické: Problém s mikrofonem a podobně.
- Prostředí: Přílišný hluk, šum.
- Lidský faktor: Řečník špatně vyslovuje.
- Jazykové: Nezvyklé jméno.

Výsledné skóre se vypočten následujícím způsobem:

$$\text{WER} = \frac{S + D + I}{N} \tag{4.1}$$

kde: S – počet substitucí (substitution);
D – počet smazání/vynechání (deletion);
I - vložení (Insertion);
C/H – počet správně určených slov (Correct/Hit);
N je součtem všech slov ve vzorku.
Tedy pro lepší přehlednost:

$$\text{WER} = \frac{S + D + I}{S + D + H}, \tag{4.2}$$

Výsledkem je nezáporné číslo, které se pohybuje od 0 výše, kdy 0 je nejlepší možné – jedná se o poměr chybných slov vůči celku. Často se proto vyjadřuje v %. Důvodem, proč je možné dosáhnout více než 100 % je situace, kdy systém tvoří velké množství chyb typu *I*.

4.1.2 Kritika WER, alternativní metriky

Myšlenka z předchozího bodu ohledně zdrojů chyb si zaslouží rozvést – i v důsledku toho, že chyby mohou vznikat důsledkem mnoha různých jevů a mnoha místech celého procesu, přičemž WER je nepostihuje, vede k návrhům na metriky jiné, například WIL (Word Information Lost).[53] Autoři v tomto dokumentu mimo další aspekty shrnují základní problematické charakteristiky WER:

- Nejedná se o skutečně jasnou metriku (dávající výsledky 0-100 %) – již výše zmíněným mechanismem přes chyby typu I se dá dostat nad 100 %, z čehož mimochodem vyplývá, že váhy chyby D a I nejsou vždy stejně velké (typ I „umí“ být „větší“).
- Tím, že nemá horní mez, se jedná pouze o relativní porovnání mezi systémy (kdy menší je lepší).

WER je vyhodnocen jako dostačující metrika pro přepisy (transkripci), ale pro jiné účely jsou navrhovány jiné metriky. Z nich se jako nejpraktičtější jeví právě WIL (Word Information Lost).

Tato metrika vychází z jiné metriky, RIL (Relative Information Lost – relativní ztráta informace), založené na závislosti vstupního slova X a výstupního slova Y.

$$\mathrm{RIL} = \frac{H(Y|X)}{H(Y)} \tag{4.3}$$

Problematické je namapování vstupu a výstupu na metriku při reálném použití, autoři RIL tedy navrhli jeho aproximaci, zmiňovaný WIL, která využívá pouze hodnoty využívané již ve WER – hodnoty H, S, D a I.

Hodnota je získána následovně:

$$\mathrm{WIL} = \frac{H^2}{(H+S+D)+(H+S+I)} \tag{4.4}$$

Jeho výhodou je, že je výsledek skutečně shora omezen 100 %.

Pro vyhodnocování v rámci výsledků trénování v rámci fine-tuning procesu u modelu wav2vec2 je využita standardní metrika WER.

4.2 Vyhodnocení modelů

Následují podrobné charakteristiky popisující výsledky pro každý model, hlavní výsledky poté sdruženě porovnává poslední shrnující tabulka.

Pro přehlednost a srozumitelnost: Pod *CommonVoice* je myšleno užití příslušného datasetu s anotací, pod ortografický x spisovný ve spojitosti HHTT příslušná transkripce užitá spolu s audio-soubory. Normováním je případně myšleno normování výstupu modelu dle mechanismu uvedeného v předchozí kapitole.

4.2.1 Metrika vyhodnocení

Pro vyhodnocení jsou použity hodnoty word-level-accuracy (*přesnost na úrovni slov*). Accuracy se vypočte jako doplněk do 100 % k WER popsanému výše. Příkladně: Při WER rovné 20 % je reportovaná a vyhodnocovaná přesnost 100-20, tedy 80 %.

Použitý vyhodnocovací skript z [48] kromě této hodnoty poskytuje i další charakteristiku, correctness (*správnost*). Ta se vypočte jako $\frac{H}{N}$. Kromě samotných hodnot jsou ještě spočteny intervaly spolehlivosti pro $p = 0.001$, $p = 0.01$, $p = 0.05$ Tyto jsou získány v rámci zpracování spočtených hodnot, z nichž je sestavena kovarianční matice a provedeny potřebné výpočty.

4.2.2 Postup vyhodnocování

Natrénovaný model je využit k rozpoznání popsané testovací sady z HHTT. Pro porovnatelnost a jednotnost postupu s jinými výstupy je výstup z modelu převeden do formátu **.MLF*[1].

Pro porovnání například s výsledky v odkazovaném dokumentu s HHTT vyhodnocen stejným způsobem jako při porovnání "standardních"přístup a wav2vec2.0 modelu, předpokladem čehož je právě předpokládaný formát výstupu.

Vyhodnocování probíhá proti 2 referencím (dvěma reprezentacím):

- Ortografické přepisy referenční sady, představované souborem ref.txt
- Ortografické přepisy, na nichž byla aplikována táž pravidla, s jejichž užitím je normován výstup z modelu, představováno souborem ref_norm.txt

Sada pravidel má rozsah 79 výrazů, vychází ze sady v [48].

[1] Master Label Files, formát pro použití v sadě nástrojů (toolkitu) HTK. Více se lze o formátu dozvědět kupříkladu zde: http://www.seas.ucla.edu/spapl/weichu/htkbook/node117_mn.html

4.2.3 Fine-tuning na CommonVoice–datech

První model je do-natrénovaný z balíku obecných Commonvoice dat. Pro testovací sadu HHTT vychází následující hodnocení:

Commonvoice, bez normování

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	49.27	–	709	–	730	–	1439
WORD	79.51	73.66	4642	303	893	342	5838
CHAR	94.63	88.09	25220	762	668	1745	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	77.40	81.62	70.67	76.59
0.990	77.86	81.16	71.33	75.95
0.950	78.26	80.77	71.89	75.41

Tabulka 4.1: Vyhodnocení fine-tuningu s užitím CommonVoice dat

Commonvoice, s normováním

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	59.07	—	850	–	850	–	1439
WORD	85.32	79.48	4982	303	554	341	5839
CHAR	95.60	89.60	25416	669	500	1597	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	83.40	87.24	76.64	82.29
0.990	83.82	86.82	77.26	81.68
0.950	84.18	86.46	77.79	81.16

Tabulka 4.2: Vyhodnocení fine-tuningu s užitím CommonVoice dat s užitím normovaného výstupu

Právě prezentované výsledky lze pro další porovnávání požadovat za referenční – nejhorší možný akceptovatelný výsledek, přičemž laděný model by měl dle předpokladů

dosáhnout až výrazně lepších než těchto výsledků. Jak je vidět, normování výstupu má poměrně značný vliv na výslednou přesnost rozpoznávání.

4.2.4 Fine-tuning na ortografických datech

ortografická data

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	61.92	—	891	–	548	–	1439
WORD	86.26	80.99	5036	228	574	308	5838
CHAR	96.11	89.34	25612	740	298	1802	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	84.14	88.36	78.27	83.67
0.990	84.60	87.91	78.86	83.09
0.950	85.00	87.51	79.37	82.59

Tabulka 4.3: Vyhodnocení fine-tuningu s užitím ortografických dat s metadaty

ortografická data s normalizací

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	67.34	—	969	–	470	–	1439
WORD	90.19	84.50	5266	185	388	332	5839
CHAR	97.04	90.33	25797	533	255	1782	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	88.36	92.01	81.98	86.99
0.990	88.76	91.61	82.54	86.45
0.950	89.10	91.27	83.01	85.98

Tabulka 4.4: Vyhodnocení fine-tuningu s užitím ortografických dat s metadaty s užitím normovaného výstupu

Lze vidět, že model trénovaný z ortografických dat naplňuje předpoklad, že bude lepším než z obecných dat. Rozdíl není enormní, leč je jasně viditelný.

4.2.5 Fine-tuning na ortografických datech bez pomocných dat

ortografická data bez metadat

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	66.30	—	954	–	485	–	1439
WORD	88.71	84.19	5179	186	473	264	5838
CHAR	96.60	92.36	25745	639	266	1130	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	86.86	90.55	81.61	86.72
0.990	87.27	90.15	82.18	86.17
0.950	87.61	89.80	82.66	85.70

Tabulka 4.5: Vyhodnocení fine-tuningu s užitím ortografických dat bez metadat

ortografická data bez metadat, s normováním výstupu

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	71.30	—	1026	–	413	–	1439
WORD	90.79	86.26	5301	187	351	264	5839
CHAR	96.93	92.79	25768	578	239	1101	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	89.02	92.54	83.77	86.72
0.990	89.41	92.16	84.32	88.19
0.950	89.74	91.83	84.79	87.73

Tabulka 4.6: Vyhodnocení fine-tuningu s užitím ortografických dat bez metadat s užitím normalizovaného výstupu

Poměrně zajímavý výsledek. Odstraněním pomocných dat o dalších akustických jevech (proč je toto správně je rozvedeno dále) v promluvě dochází k vylepšení přesnosti modelu.

4.2.6 Fine-tuning na spisovných datech

Spisovná data, bez normování výstupu

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	59.49	—	856	–	583	–	1439
WORD	84.93	80.58	4958	176	704	254	5838
CHAR	95.19	90.58	25368	803	479	1229	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	82.95	86.90	77.85	83.27
0.990	83.38	86.47	78.44	82.69
0.950	83.75	86.10	78.96	82.18

Tabulka 4.7: Vyhodnocení fine-tuningu s užitím spisovných dat

Spisovná data, s normováním výstupu

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	68.31	—	983	–	456	–	1439
WORD	89.60	85.25	5232	228	574	308	5839
CHAR	96.48	92.22	25648	613	324	1130	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	87.84	91.37	82.71	87.76
0.990	88.22	90.98	83.27	87.22
0.950	88.55	90.65	83.75	86.75

Vyhodnocení fine-tuningu s užitím spisovných dat s užitím normalizovaného výstupu Rozdíly jsou sice poměrně malé, leč opravdu zajímavé: Model natrénovaný z ortograficky reprezentovaného jazyka má vyšší přesnost než model z normalizovaného jazyka. Rozdíl zůstává ve všech testovaných variantách (ne)úpravy výstupu.

4.2.7 Fine-tuning na spisovných datech s odlišnými parametry

Výrazně delší trénování přináší zlepšení, ovšem velmi malé, v řádu desetin procenta. Na základě tohoto experimentu lze říci, že standardní parametry jsou zvoleny vhodně, vynaložení více zdrojů nepřináší žádný výrazný přínos.

Spisovná data

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	60.11	—	865	–	574	–	1439
WORD	85.05	80.88	4965	188	685	243	5838
CHAR	95.28	90.58	25392	777	481	1253	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	83.03	87.06	78.19	83.54
0.990	83.4	86.62	78.78	82.97
0.950	83.85	86.24	79.29	82.47

Tabulka 4.8: Vyhodnocení fine-tuningu s užitím spisovných dat, prodloužené trénování

Spisovná data, s normalizací

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	69.01	—	993	–	446	–	1439
WORD	89.71	85.55	5238	189	412	243	5839
CHAR	96.57	92.22	25674	583	328	1157	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	87.90	91.51	83.02	88.03
0.990	88.29	91.12	83.58	87.49
0.950	84.05	87.03	84.05	87.03

Tabulka 4.9: Vyhodnocení fine-tuningu s užitím spisovných dat, prodloužené trénování, s užitím normalizovaného výstupu

4.2.8 Opakovaný fine-tuning na spisovných datech

Pokus s opakovaným fine-tuningem. Prvním byl CommonVoice, druhé opakování jsou spisovná HHTT data.

CommonVoice + spisovná, bez normalizace výstupu

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	61.64	—	887	–	552	–	1439
WORD	86.11	81.83	5027	161	650	250	5838
CHAR	95.80	91.19	25532	687	431	1229	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	84.21	88.00	79.17	84.45
0.990	84.63	87.59	79.75	83.88
0.950	84.98	87.23	80.25	83.39

Tabulka 4.10: Vyhodnocení opakovaného fine-tuningu s užitím CommonVoice a HHTT spisovných dat

CommonVoice + spisovná data, s normováním výstupu

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	70.54	—	1015	–	424	–	1439
WORD	90.80	86.54	5302	161	376	249	5839
CHAR	97.10	92.83	25814	496	275	1136	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	89.15	92.45	84.08	88.96
0.990	89.51	92.09	84.62	88.44
0.950	89.82	91.78	85.08	87.98

Tabulka 4.11: Vyhodnocení opakovaného fine-tuningu s užitím CommonVoice a HHTT spisovných dat, normování výstupu

4.2.9 Fine-tuning ortografická+spisovná data

Vyhodnocení modelů natrénovaných na obou reprezentacích současně (ortografická i spisovná).

Zdvojená reprezentace dat

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	60.46	—	870	–	569	–	1439
WORD	84.91	80.73	4957	189	692	244	5838
CHAR	95.42	91.14	25429	831	390	1140	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	82.88	86.93	78.02	83.40
0.990	83.32	86.49	78.62	82.82
0.950	83.70	86.11	79.12	82.32

Zdvojená reprezentace dat, s normováním

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	67.69	—	974	–	465	–	1439
WORD	88.66	84.48	5177	190	472	244	5839
CHAR	96.25	92.15	25589	674	322	1092	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	86.75	90.57	81.89	87.04
0.990	87.17	90.15	82.46	86.49
0.950	87.53	89.80	82.94	86.01

Tabulka 4.12: Vyhodnocení modelu se sdruženými reprezentacemi, s metadaty

Zdvojená reprezentace bez metadat

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	60.46	—	870	–	569	–	1439
WORD	85.35	81.31	4983	198	657	236	5838
CHAR	95.24	91.15	25381	880	389	1090	26650

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	83.35	87.35	78.63	83.95
0.990	83.78	86.92	79.22	83.38
0.950	84.16	86.54	79.72	82.89

Zdvojená reprezentace bez metadat, s normováním

	Correct [%]	Accuracy [%]	H	D	S	I	N
SENT	67.89	—	977	–	462	–	1439
WORD	88.99	85.55	5196	198	445	235	5839
CHAR	96.19	92.33	25571	713	301	1025	26585

Intervaly spolehlivosti (p)

	Correct[%] (min)	Correct[%] (max)	Accuracy[%] (min)	Accuracy[%] (max)
0.999	87.12	90.85	82.42	87.48
0.990	87.52	90.45	82.97	86.93
0.950	87.88	90.10	83.45	86.46

Tabulka 4.13: Vyhodnocení modelu se sdruženými reprezentacemi, bez metadat

4.2.10 Shrnující tabulka

Ze shrnující tabulky jsou již vynechány ty modely, u jejichž trénování byla využita anotace s metadaty, neboť jsou principiálně (nedodržení podmínky pro CTC) nekorektní.

Shrnující tabulka

	CommonVoice	orto	spis	CV+HHTT-N	orto+spis
WORD[%]	73.66	84.19	80.58	81.83	81.31
Po normování	CommonVoice	orto	spis	CV+HHTT-N	orto+spis
WORD[%]	79.48	86.26	85.25	86.54	85.55

Tabulka 4.14: Shrnující tabulka

Orto – model, u nějž proběhl fine-tuning s pomocí ortografické anotace.
Spis - model, u nějž proběhl fine-tuning s pomocí spisovné anotace.

4.3 Shrnutí

Ukazuje se, že alespoň za daných podmínek, tedy s daným pre-trained modelem, sadou audio-souborů a s užitými druhy anotace těchto dat je dosažitelná přesnost na úrovni rozpoznání slov přes 86 %. K této metě se dá z popsaných zdrojů dostat 2 způsoby:

- Dotrénováním modelu z ortografických dat a normováním výstupu s aplikací normování výstupu.
- Dotrénováním na CommonVoice a druhým dotrénováním na spisovných datech s aplikací normování výstupu.

Ostatní modely se této hodnotě blíží, ovšem nedosahují ji. Model s dvojitou reprezentací dat je o necelý procentní bod horší než model z čistě ortografických dat, což je zajímavé zjištění: Více učících dat modelu nepomohlo, naopak.

Z toho plyne několik možných závěrů pro aplikace, při nichž nemusí být dostupné všechny typy dat.

Mezi možnostmi reprezentace dat je mezi nejlepšími „skutečná“ reprezentace, která je následně korigována na výstupu modelu normalizací.

Pokud by z nějakého důvodu ovšem nebyly dostupné doslovné přepisy nahrávek, je ekvivalentní náhradou s takřka identickými výsledky použití obecného CommonVoice a doplněním spisovných přepisů cílové aplikace opakovaným fine-tuningem.

Toto se může zdát jako zvláštní kombinace, ovšem potenciál pro využití této kombinace bych viděl zcela konkrétní a reálný: Toto rozložení výhradně podle mého pozorování, neboť podrobnou analýzou nedisponuji, poměrně dobře odpovídá například rozhovorům vedeným

v Českém rozhlase nebo některých podcastech, kde autoři poměrně často poskytují textový přepis ve spisovném jazyce.

Tato data jsou obvykle poměrně dobře dostupná, leč nejsem si jist, zda-li jsou vždy mapována přesně 1:1 a nedochází k reformulacím.

Dále jsem si ověřil, že skutečně platí dříve uvedené předpoklady o délkách vektorů CTC (musí platit vstupní $X <=$ výstupní Y) – v ortografické anotaci přítomná pomocná metadata jako například *_inhale_* (reprezentující nádech) jsou delší než výstup (nádech), který by měly reprezentovat (z nádechu hlásku „*a*“ nelze získat), což v důsledku vede k chybám a zhoršení dosahovaných výsledků v porovnání s daty tyto podmínky plnícími.

Věci mi komplikovaly a ve výsledku zdržely i tyto 2 faktory:

- Lidská chyba – například použitý špatný/špatně vytvořený manifest pro model (zejména když se toto stane opakovaně a projevení se trvá několik hodin/dnů trénování, jde o nepříjemnou věc).
- Softwarová nekompatibilita/rozdílnost užitých verzí použitých nástrojů i v rámci „stejné“ verze (rozdílné commity, ale stejná verze).

Věcí k diskusi je samotné normování a jeho realizace. V současném provedení je nutné (vy)tvořit ony sady pravidel, kterými jsou postižené pojmy převedeny na ekvivalentní reprezentace. U aplikací na konkrétní problém s omezenou slovní zásobou se jedná o řešitelný problém.

Myšlenkou, která mne u formulace tohoto shrnutí napadla, zda-li by nešlo na výstup modelu doplnit ještě dodatečnou vrstvu obsahující znalost „zaměnitelnosti“, čímž by normalizaci prováděl přímo model.

Závěr

Cílem práce bylo seznámit se, jak popisované a užité technologie fungují a jakým způsobem se aplikují, a posléze prověřit důsledky voleb různých typů vstupních dat v kombinaci s různými dílčími úpravami nebo změnou interpretace vstupů či výstupů.

Závěry jsou z uvedených dat poměrně jednoznačné. Modely založené na transformerech nelze nazvat jinak než jako obrovský posun na poli řešení rozmanitých problémů, v nichž hrají ústřední roli sekvence všemožných druhů. Pro řešenou úlohu se po prověření několika možných variant „doladění“ (*fine-tuningu*) daného předtrénovaného (*pre-trained*) modelu jeví jako nejlepším přístupem pro danou úlohu použít model vzniklý kombinací CommonVoice dat následně rozšířený dalším fine-tuningem ze spisovných anotovaných dat, těsně následovaným modelem z ortografický dat.

Zjištění lze shrnout do několika bodů a kategorií:

- Nastavení trénování transformerů (80 tisíc trénovacích cyklů a learning rate) je vhodně nastaveno. Rozšiřování délky trénování nemělo zásadní přínos (byť byl lehce kladný, tak v poměru k dodatečnému potřebnému výpočetnímu času se rozhodně nevyplatí).

- Správnou volbou dat použitých pro fine-tuning lze citelně ovlivnit výslednou přesnost.

- Model se umí naučit, alespoň co lze z vyhodnocení testovací sady říct, že jedna zvuková stopa může reprezentovat lehce odlišné textové podoby, ovšem přínos není nijak zásadní.

- Pomocné informace, které neprezentují přímo jazyk (pomocná metadata), ale různé nejazykové zvukové jevy (kašel, označení neidentifikovaného šumu/hluku a podobně, parazitická slova typu „hmmm“[2] „výplně“ nevedou ke zlepšení přesnosti rozpoznávání, naopak takovýto model vychází i hůře než jeho „sourozenec“ bez těchto dat, neboť takováto anotace dat odporuje pravidlům pro použití CTC.

[2]Existuje i „hmmm“ s významem souhlasu a toto je citoslovcem, přičemž „standardní“ vyjádření souhlasu, například slovo *ano*, je částicí; obecně hranice mezi těmito slovními druhy je někdy komplikovaně určitelná) http://nase-rec.ujc.cas.cz/archiv.php?art=7419

- Dodatečná normalizace (její provedení na výstupu souborem pravidel) zvyšuje přesnost vůči referenčnímu souboru ve všech případech.
- Model, jenž prošel fine-tuningem za užití ortografické reprezentace textu je přesnější než model z normalizované podoby téhož textu.
- Případné lidské chyby (například při tvorbě manifestu, výběru souborů a podobně) se při nepozornosti člověka dají z průběhu zjistit až po několika tisících iteracích. Jde o velmi nepříjemnou zkušenost.
- Užití transformerů v kombinaci s dalšími nástroji pro toto použití je určitě vhodné, dosahované výsledky jsou poměrně uspokojivé.

Z průzkumu aktuálního vývoje dostupných možností mi také vyšlo, že by mohlo být dále vhodné zkusit prověřit chování modelu využívajícího modifikované verze Wav2vec2, nazývaná Wav2Vec2-Conformer, jejíž odlišností je použití upravených transformerů. Tyto jednotky jsou nazývány Conformery, odlišností je využití i složky tvořené konvoluční neuronovou sít [54]. S ohledem na odkazové výsledky by možná takovýto model lepších výsledků. Negativem je, že k tomuto by bylo potřeba natrénovat nový *„podkladový“* model založený na této architektuře, tedy od počátku započít celý proces transfer learningu, konkrétně natrénovat *pre-trained* model, což je zejména z pohledu potřebných zdrojů komplikované.

Literatura a jiné zdroje

1. JOUPPI, Norman P.; YOUNG, Cliff; PATIL, Nishant; PATTERSON, David; AGRAWAL, Gaurav; BAJWA, Raminder; BATES, Sarah; BHATIA, Suresh; BODEN, Nan; BORCHERS, Al; BOYLE, Rick; CANTIN, Pierre-luc; CHAO, Clifford; CLARK, Chris; CORIELL, Jeremy; DALEY, Mike; DAU, Matt; DEAN, Jeffrey; GELB, Ben; GHAEMMAGHAMI, Tara Vazir; GOTTIPATI, Rajendra; GULLAND, William; HAGMANN, Robert; HO, C. Richard; HOGBERG, Doug; HU, John; HUNDT, Robert; HURT, Dan; IBARZ, Julian; JAFFEY, Aaron; JAWORSKI, Alek; KAPLAN, Alexander; KHAITAN, Harshit; KOCH, Andy; KUMAR, Naveen; LACY, Steve; LAUDON, James; LAW, James; LE, Diemthu; LEARY, Chris; LIU, Zhuyuan; LUCKE, Kyle; LUNDIN, Alan; MACKEAN, Gordon; MAGGIORE, Adriana; MAHONY, Maire; MILLER, Kieran; NAGARAJAN, Rahul; NARAYANASWAMI, Ravi; NI, Ray; NIX, Kathy; NORRIE, Thomas; OMERNICK, Mark; PENUKONDA, Narayana; PHELPS, Andy; ROSS, Jonathan; ROSS, Matt; SALEK, Amir; SAMADIANI, Emad; SEVERN, Chris; SIZIKOV, Gregory; SNELHAM, Matthew; SOUTER, Jed; STEINBERG, Dan; SWING, Andy; TAN, Mercedes; THORSON, Gregory; TIAN, Bo; TOMA, Horia; TUTTLE, Erick; VASUDEVAN, Vijay; WALTER, Richard; WANG, Walter; WILCOX, Eric; YOON, Doe Hyun. *In-Datacenter Performance Analysis of a Tensor Processing Unit*. arXiv, 2017. Dostupné z DOI: 10.48550/ARXIV.1704.04760.

2. *TensorFlow Lite for Microcontrollers* [online]. tensorflow, 2022 [cit. 2022-06-11]. Dostupné z: https://github.com/tensorflow/tflite-micro. original-date: 2021-04-08T21:40:50Z.

3. *1.17. Neural network models (supervised)* [online]. [B.r.] [cit. 2022-08-08]. Dostupné z: https://scikit-learn/stable/modules/neural_networks_supervised.html.

4. BASODI, Sunitha; JI, Chunyan; ZHANG, Haiping; PAN, Yi. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*. 2020, roč. 3, č. 3, s. 196–207. Dostupné z DOI: 10.26599/BDMA.2020.9020004.

5. OLAH, Chris; CARTER, Shan. Attention and Augmented Recurrent Neural Networks. *Distill*. 2016. Dostupné z DOI: 10.23915/distill.00001.

6. OLAH, Christopher. *Understanding LSTM Networks*. 2015-08. Dostupné také z: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

7. SHI YAN. *LSTM building block*. 2016. Dostupné také z: https://miro.medium.com/max/1400/1*laH0_xXEkFE0lKJu54gkFQ.png. [Online; accessed June 27, 2022].

8. DIVISH RENGASAMY. *LSTM unit*. 2019. Dostupné také z: https://www.researchgate.net/figure/The-LSTM-unit-contain-a-forget-gate-output-gate-and-input-gate-The-yellow-circle_fig2_338717757. [Online; accessed July 27, 2022].

9. GABRIEL LOYE. *Inner workings of the GRU cell*. 2019. Dostupné také z: http://areshenk-research-notes.com/wp-content/uploads/2015/02/esn_math.png. [Online; accessed July 27, 2022].

10. KLAUS GREFF, RUPESH K. SRIVASTAVA, JAN KOUTNÍK, BAS R. STEUNEBRINK, JÜRGEN SCHMIDHUBER. LSTM: A Search Space Odyssey. *TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*. [B.r.]. Dostupné z eprint: 503.04069.

11. HOLZMANN, Georg. *Echo State Networks with Filter Neurons and a Delay&Sum Readout with Applications in Audio Signal Processing*. 2008. Dostupné také z: http://grh.mur.at/sites/default/files/MasterThesis.pdf. Master's Thesis. Graz University of Technology.

12. KUPPUSWAMY, Naveen. *Reservoir Computing Introduction*. 2013-04-26. Dostupné také z: https://www.ifi.uzh.ch/dam/jcr:00000000-7f84-9c3b-ffff-ffffbda0da2e/zurich_26_04_13.pdf.

13. ADMIN. *The vanishing gradient problem and ReLUs – a TensorFlow investigation*. Dostupné také z: https://adventuresinmachinelearning.com/vanishing-gradient-problem-tensorflow/.

14. JAEGER, H. Echo state network. *Scholarpedia*. 2007, roč. 2, č. 9, s. 2330. Dostupné z DOI: 10.4249/scholarpedia.2330. revision #196567.

15. CORSON N. ARESHENKOFF. *ECHO-STATE NETWORKS IN R*. 2014. Dostupné také z: http://areshenk-research-notes.com/wp-content/uploads/2015/02/esn_architecture.png. [Online; accessed July 27, 2022].

16. CORSON N. ARESHENKOFF. *ECHO-STATE NETWORKS IN R*. 2014. Dostupné také z: http://areshenk-research-notes.com/wp-content/uploads/2015/02/esn_math.png. [Online; accessed July 27, 2022].

17. LIWICKI, Marcus; GRAVES, Alex; BUNKE, Horst; SCHMIDHUBER, Jürgen. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In: *In Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*. 2007.

18. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need*. arXiv, 2017. Dostupné z DOI: 10.48550/ARXIV.1706.03762.

19. XU, Kelvin; BA, Jimmy; KIROS, Ryan; CHO, Kyunghyun; COURVILLE, Aaron; SALAKHUTDINOV, Ruslan; ZEMEL, Richard; BENGIO, Yoshua. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. arXiv, 2015. Dostupné z DOI: 10.48550/ARXIV.1502.03044.

20. O'SHEA, Keiron; NASH, Ryan. An Introduction to Convolutional Neural Networks. *CoRR*. 2015, roč. abs/1511.08458. Dostupné z arXiv: 1511.08458.

21. NEUVEDEN. *Sound Classification Using Convolutional Neural Network and Tensor Deep Stacking Network* [online] [cit. 2020-05-29]. Dostupné z: https://ieeexplore.ieee.org/document/8605515.

22. ROTHMANN, Daniel. *What's wrong with CNNs and spectrograms for audio processing?* Dostupné také z: https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd.

23. THE CLICK READER. *Building a Convolutional Neural Network*. 2019. Dostupné také z: https://www.theclickreader.com/wp-content/uploads/2020/07/cnn-architecture-1024x576.png. [Online; accessed July 27, 2022].

24. RAGHUVEER, Lakkavaram V. S. [B.r.].

25. WANG, Zijie J.; TURKO, Robert; SHAIKH, Omar; PARK, Haekyu; DAS, Nilaksh; HOHMAN, Fred; KAHNG, Minsuk; CHAU, Duen Horng Polo. CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics*. 2021, roč. 27, č. 2, s. 1396–1406. Dostupné z DOI: 10.1109/tvcg.2020.3030418.

26. EICHINSKI, Philip; ALEXANDER, Callan; ROE, Paul; PARSONS, Stuart; FULLER, Susan. A Convolutional Neural Network Bird Species Recognizer Built From Little Data by Iteratively Training, Detecting, and Labeling. *Frontiers in Ecology and Evolution* [online]. 2022, roč. 10 [cit. 2022-06-23]. ISSN 2296-701X. Dostupné z: https://www.frontiersin.org/article/10.3389/fevo.2022.810330.

27. STASAK, Brian; HUANG, Zhaocheng; RAZAVI, Sabah; JOACHIM, Dale; EPPS, Julien. Automatic Detection of COVID-19 Based on Short-Duration Acoustic Smartphone Speech Analysis. *Journal of Healthcare Informatics Research* [online]. 2021, vol. 5, no. 2, s. 201–217 [cit. 2022-06-17]. ISSN 2509-498X. Dostupné z DOI: 10.1007/s41666-020-00090-4.

28. ALAMMAR, Jay. *The Illustrated Transformer* [online] [cit. 2022-08-10]. Dostupné z: https://jalammar.github.io/illustrated-transformer/.

29. WENG, Lilian. *Contrastive Representation Learning* [online]. 2021-05-31 [cit. 2022-08-04]. Dostupné z: https://lilianweng.github.io/posts/2021-05-31-contrastive/. Section: posts.

30. MILLET, Juliette; CAUCHETEUX, Charlotte; ORHAN, Pierre; BOUBENEC, Yves; GRAMFORT, Alexandre; DUNBAR, Ewan; PALLIER, Christophe; KING, Jean-Remi. *Toward a realistic model of speech processing in the brain with self-supervised learning* [online]. 2022-06-03 [cit. 2022-08-06]. arXiv:2206.01685. arXiv. Dostupné z DOI: 10.48550/arXiv.2206.01685. type: article.

31. GEKSUT. *How the model works.* 2019. Dostupné také z: https://humandata.io/wp-content/uploads/2021/08/image.png. [Online; accessed July 27, 2022].

32. AULI, Michael; CONNEAU, Alexis; BAEVSKI, Alexei. *WAV2VEC 2.0: Learning the structure of Speech from Raw Audio.* [B.r.]. Dostupné také z: https://ai.facebook.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw-audio/.

33. AULI, Michael; BAEVSKI, Alexei; SHAH, Siddharth; FUEGEN, Christian. *Wav2vec: State-of-the-art speech recognition through self-supervision — ai.facebook.com* [https://ai.facebook.com/blog/wav2vec-state-of-the-art-speech-recognition-through-self-supervision/]. 2019. [Accessed 1-Aug-2022].

34. BOIGNE, Jonathan. *An illustrated tour of wav2vec 2.0.* 2021. Dostupné také z: https://jonathanbgn.com/2021/09/30/illustrated-wav2vec-2.html.

35. GARSON, James. Connectionism [online]. 1997 [cit. 2022-08-02]. Dostupné z: https://plato.stanford.edu/archives/fall2018/entries/connectionism/. Last Modified: 2015-02-19.

36. HANNUN, Awni. Sequence Modeling with CTC. *Distill* [online]. 2017, vol. 2, no. 11, e8 [cit. 2022-05-22]. ISSN 2476-0757. Dostupné z DOI: 10.23915/distill.00008.

37. BATTENBERG, Eric; CHEN, Jitong; CHILD, Rewon; COATES, Adam; GAUR, Yashesh; LI, Yi; LIU, Hairong; SATHEESH, Sanjeev; SEETAPUN, David; SRIRAM, Anuroop; ZHU, Zhenyao. Exploring Neural Transducers for End-to-End Speech Recognition. *CoRR.* 2017, roč. abs/1707.07413. Dostupné z arXiv: 1707.07413.

38. ZEYER, Albert; BECK, Eugen; SCHLÜTER, Ralf; NEY, Hermann. CTC in the Context of Generalized Full-Sum HMM Training. In: 2017, s. 944–948. Dostupné z DOI: 10.21437/Interspeech.2017-1073.

39. *Čeština je krásná díky jazyku Bible kralické* [Vysočina] [online]. 2016-09-12 [cit. 2022-08-03]. Dostupné z: https://vysocina.rozhlas.cz/cestina-je-krasna-diky-jazyku-bible-kralicke-7123550. Section: Životní styl.

40. *Naše řeč – O jazyce Kralické bible* [online] [cit. 2022-08-03]. Dostupné z: http://nase-rec.ujc.cas.cz/archiv.php?art=6140.

41. COUPÉ, Christophe; OH, Yoon Mi; DEDIU, Dan; PELLEGRINO, François. Different languages, similar encoding efficiency: Comparable information rates across the human communicative niche. *Science Advances*. 2019, roč. 5, č. 9, eaaw2594. Dostupné z DOI: 10.1126/sciadv.aaw2594.

42. CSIKSZENTMIHALYI, Mihaly. *Flow and the Foundations of Positive Psychology* [online]. Dordrecht: Springer Netherlands, 2014 [cit. 2022-06-08]. ISBN 978-94-017-9087-1 978-94-017-9088-8. Dostupné z DOI: 10.1007/978-94-017-9088-8.

43. *CORDIS | European Commission — cordis.europa.eu* [https://cordis.europa.eu/article/id/191265-turntaking-in-human-communication]. [B.r.]. [Accessed 14-Jul-2022].

44. LEVINSON, Stephen C. Turn-taking in Human Communication – Origins and Implications for Language Processing. *Trends in Cognitive Sciences*. 2016, roč. 20, č. 1, s. 6–14. ISSN 1364-6613. Dostupné z DOI: https://doi.org/10.1016/j.tics.2015.10.010.

45. OTT, Myle; EDUNOV, Sergey; BAEVSKI, Alexei; FAN, Angela; GROSS, Sam; NG, Nathan; GRANGIER, David; AULI, Michael. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In: *Proceedings of NAACL-HLT 2019: Demonstrations*. 2019.

46. DICKSON, Ben. *What Hugging Face and Microsoft's collaboration means for applied AI* [TechTalks] [online]. 2022-05-31 [cit. 2022-06-14]. Dostupné z: https://bdtechtalks.com/2022/05/31/hugging-face-microsoft-machine-learning/.

47. LEHEČKA, Jan; ŠVEC, Jan; PRAŽÁK, Aleš; PSUTKA, Josef V. Exploring Capabilities of Monolingual Audio Transformers using Large Datasets in Automatic Speech Recognition of Czech. In: *Interspeech 2022*. ISCA, 2022. Dostupné také z: https://arxiv.org/abs/2206.07627. (in press).

48. ŠVEC, Jan; FRÉMUND, Adam; BULÍN, Martin; LEHAČKA, Jan. Transfer Learning of Transformers for Spoken Language Understanding. In: 2022.

49. HONZA BERÁNEK, Zdeněk Michl. *České dráhy budou vozit od prosince cestující podle zcela nového tarifu* [ŽelPage.cz] [online] [cit. 2022-07-13]. Dostupné z: `http://www.zelpage.cz/zpravy/5429`. ISSN: 1801-5425.

50. *Mozilla Common Voice* [online] [cit. 2022-08-03]. Dostupné z: `https://commonvoice.mozilla.org/`.

51. OTT, Myle; EDUNOV, Sergey; GRANGIER, David; AULI, Michael. Scaling Neural Machine Translation. *CoRR*. 2018, roč. abs/1806.00187. Dostupné z arXiv: `1806.00187`.

52. *Longstanding problem put to rest* [MIT News | Massachusetts Institute of Technology] [online] [cit. 2022-08-03]. Dostupné z: `https://news.mit.edu/2015/algorithm-genome-best-possible-0610`.

53. ERRATTAHI, Rahhal; EL HANNANI, Asmaa; OUAHMANE, Hassan. Automatic Speech Recognition Errors Detection and Correction: A Review. *Procedia Computer Science* [online]. 2018, vol. 128, s. 32–37 [cit. 2022-07-24]. ISSN 1877-0509. Dostupné z DOI: `10.1016/j.procs.2018.03.005`.

54. GULATI, Anmol; QIN, James; CHIU, Chung-Cheng; PARMAR, Niki; ZHANG, Yu; YU, Jiahui; HAN, Wei; WANG, Shibo; ZHANG, Zhengdong; WU, Yonghui; PANG, Ruoming. *Conformer: Convolution-augmented Transformer for Speech Recognition*. arXiv, 2020. Dostupné z DOI: `10.48550/ARXIV.2005.08100`.