VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ODHAD OBLIČEJE Z ŘEČOVÉHO SIGNÁLU

LEARNING THE FACE BEHIND A VOICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOSEF KRUŠINA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. OLDŘICH PLCHOT, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Krušina Josef**
Program: Informační technologie
Název: **Odhad obličeje z řečového signálu**
Learning the Face Behind a Voice
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Prostudujte statistické techniky pro modelování řeči a řečníka, soustřeďte se na neuronové sítě a extrakci embedingů.
2. Prostudujte techniky pro zpracování obrazu (obličejů), seznamte se s dostupnými modely, které umožňují extrakci embedingů z obrazových dat.
3. Proveďte rešerši problematiky odhadu obličeje z řeči a prostudujte současné přístupy a architektury používaných modelů.
4. Seznamte se s databázemi Voxceleb a AVS Speech a proveďte rešerši dalších dostupných databází vhodných k řešení problematiky odhadu obličeje z řečové nahrávky.
5. Navrhněte a natrénujte systém, který bude generovat obličej na základě vstupní řeči.
6. Vyhodnoťte úspěšnost a robustnost natrénovaného modelu a analyzujte jeho případné nedostatky.

Literatura:

- Tae-Hyun Oh, Tali Dekel, Changil Kim, Inbar Mosseri, William T. Freeman, Michael Rubinstein, Wojciech Matusik, "Speech2Face: Learning the Face Behind a Voice", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019

- A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation", ACM Transactions on Graphics (SIG-GRAPH), 2018

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Plchot Oldřich, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 2. listopadu 2021

Abstrakt

Tato práce řeší problém mapování fixních reprezentací (embeddingů) řečového signálu na embeddingy obličejů a následné generování obličeje z namapovaného embeddingu pomocí generativní adverzní sítě (GAN) naučené na generování obličejů. GAN jsou druhem neuronových sítí, které umí generovat data podobná těm, na kterých se trénovala. Architektura navrženého systému je založena na čtyřech komponentách: na extraktoru embeddingů obličeje, na extraktoru embeddingů hlasu, na algoritmu nad GAN, který umí generovat obličej z embeddingu obličeje a na mnou implementované mapovací síti určené k mapování embeddingu hlasu na embedding obličeje. Jako extraktory embeddingů jsou převzaty předtrénované neuronové sítě FaceNet a SpeechBrain. Pro zpětné generování obličeje je převzatý model používající předtrénovaný StyleGAN2. Přínos této práce je ten, že dovoluje extrapolovat obličej pouze z audio signálu.

Abstract

This work addresses the problem of mapping fixed representations (embeddings) of a speech signal to face embeddings and then generating a face from the mapped embedding using a generative adversarial network (GAN) that was trained for face generation. GANs are a type of neural networks that can generate data similar to the data they were trained on. The architecture of the proposed system is based on four components: a face embedding extractor, a voice embedding extractor, an algorithm on top of a GAN that can generate a face from a face embedding, and my mapping network used to map a voice embedding to a face embedding. The pre-trained neural networks FaceNet and SpeechBrain are adopted as embedding extractors. A model that uses a pre-trained StyleGAN2 is adopted for backward face generation. The contribution of this work is that it allows the extrapolation of a face from audio signal only.

Klíčová slova

Extrakce příznaků, Mapování, Embedding, FaceNet, SpeechBrain, StyleGAN2

Keywords

Feature extraction, Mapping, Embedding, FaceNet, SpeechBrain, StyleGAN2

Citace

KRUŠINA, Josef. *Odhad obličeje z řečového signálu.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Oldřich Plchot, Ph.D.

Odhad obličeje z řečového signálu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Oldřicha Plchota, Ph.D.. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Josef Krušina
10. května 2022

Poděkování

Tímto bych chtěl poděkovat Ing. Oldřichu Plchotovi, Ph.D. za vedení, ochotu a pomoc při vypracovávání této práce.

Obsah

Kapitola 1

Úvod

Neustále pokračující nárůst výpočetní síly počítačů dovoluje vznik složitějších algoritmů a programů. Velice dobře se tento fakt dá pozorovat v oblasti umělé inteligence, kde můžeme každodenně pozorovat stoupající přesnost nabízených reklam, které často nabízejí služby podle našich současných potřeb, kvalitu vícejazyčných překladačů a nebo detektorů obličejů. Detektory obličejů nepřímo souvisí s tématem této práce, která se zabývá odhadem obličeje z řečového signálu. V současné době se této problematice věnovala řada prací například [29] nebo předchozí bakalářská práce z minulého roku na stejné téma [23].

Hlas v sobě uchovává různé informace o samotném vzhledu řečníka, podle kterých může člověk i stroj odhadnout vzhled řečníka, kterého nikdy neviděli. Tato práce je unikátní v přístupu ke vstupním datům (hlasu a obličeji). Architektura výsledného systému je založena na třech předtrénovaných modelech. Princip spočívá v tom, že převzaté předtrénované neuronové sítě jsou jako nezávislé prvky systému dále obohaceny o můj model mapovací neuronové sítě, která mapuje výstup modelu natrénovaného na získávání hlasových příznaků na výstup modelu získávající příznaky obličejové. Z těchto získaných obličejových příznaků je poté už vytvořený odhad obličeje. Tato implementace slouží jako prvotní model.

Následující kapitoly slouží jako zasvěcení do problematiky, od teoretického úvodu přes vlastní zhodnocení současného stavu, až po popis vlastní implementace/práce a závěr. V následujících kapitolách 2 3 4 se pokusím blíže popsat a celkově shrnout dřívější a současné technologie úzce spjaty s problematikou tématu. Po teoretičtější části práce bude následovat kapitola 5, která se věnuje praktické části práce, ve které popisuji postup konkrétního řešení problému a způsob trénování modelu pro mapování hlasu na obličej. V kapitole 6 demonstruji výsledky, vyhodnocuji je a navrhuji možná rozšíření. Poslední kapitola 7 slouží ke konečnému celkovému shrnutí celé práce.

Kapitola 2

Základy strojového učení

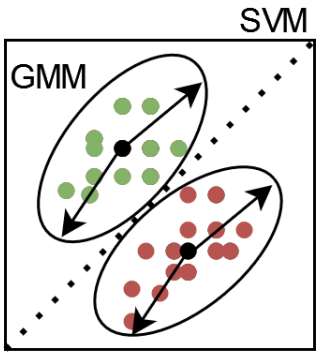
Následujících pár kapitol slouží jako průpravná část, která přiblíží čtenáři, u kterého se předpokládá úroveň vzdělání studenta bakalářského studia informačních technologií, základní teorii potřebnou k bezproblémovému pochopení praktické části této práce. Mimo jiné tyto kapitoly obsahují vysvětlená klíčová slova, která jsou nutná pro vysvětlení terminologie, první zmínky těchto slov jsou tučně zvýrazněny.

Strojové učení se dá definovat jako výpočetní metody, které se postupným nabýváním zkušeností samy zlepšují nebo dovolují přesné predikce na základě těchto zkušeností. Zkušenostmi jsou myšleny informace, které nabývají většinou elektronické formy. Informace, uchovávané ve formě dat, mohou být shlukovány jako například **datové sady** (datasets), které bývají podle patřičného případu užití **anotovány** (labeled). Hlavním kritickým faktorem, který určuje celkovou kvalitu kolekcí dat, a tím přímo úspěch samotných predikcí, je množství dat, kdy data zůstávají stále dostatečně kvalitní a nefungují kontraproduktivním způsobem pro proces trénování. Kvalita dat se snižuje pomocí negativních vlastností jako je špatná anotace dat, moc velký šum a data nevhodná pro daný úkol. [25]

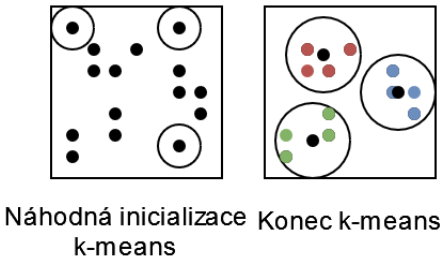
2.1 Generativní a diskriminativní modely

Rozdíl mezi **generativním** a **diskriminativním** modelem je ten, že diskriminativní model se učí hranici mezi třídami datové sady, zatímco generativní model se soustředí na rozdělení pravděpodobnosti datové sady a proto může vracet hodnotu pravděpodobnosti pro konkrétní vzorek. Dokáže tím pádem generovat nové vzorky dat. Neuronová síť, kterou implementuji v této práci a dále popisuji v kapitole 5 je diskriminativním modelem. Diskriminativní model se používá v úlohách s **učením s učitelem** 2.2.1. Generativní model se používá v úlohách s **učením bez učitele** 2.2.2. Mezi příklady diskriminativního modelu patří **Support Vector Machine – SVM** a **umělé neuronové sítě** 2.3. Mezi příklady generativního modelu patří **generativní adverzní síť** 2.7.3 a **Gaussian Mixture Model – GMM**. [45]

Na obrázku 2.1 je vidět příklad klasifikačního problému, který je řešený pomocí SVM a pomocí GMM zároveň. SVM dělí prostor příznaků na dva poloprostory, kde data, která leží v jiných polorovinách, patří do jiných tříd (znázorněno tečkovanou čarou) [22]. GMM modeluje Gaussovo rozdělení pro různé shluky dat, kde všechna data mají určitou pravděpodobnost, že spadají do určitého shluku (znázorněno elipsami) [45].



Obrázek 2.1: Příklad klasifikace pomocí SVM a GMM. SVM dělí prostor na dva poloprostory a GMM modeluje Gaussovo rozdělení pro oba shluky dat.



Obrázek 2.2: Příklad shlukové analýzy

2.2 Metody strojového učení

Strojové učení se dále dělí na základě toho, jakým stylem se dané stroje učí. Dělí se na tyto hlavní typy [8]:

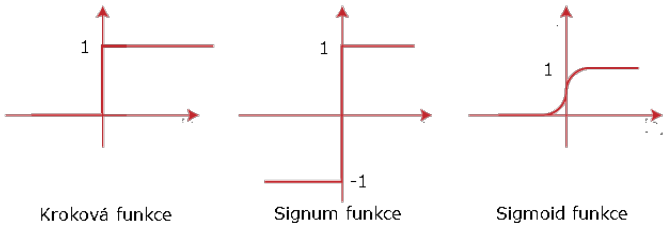
2.2.1 Učení s učitelem

Využívá anotovaných trénovacích dat k odvození funkce, která mapuje vstup na výstup. Mezi nejčastější úlohy učení s učitelem patří **klasifikace** a **regrese** [39]. Jako příklad klasifikace může sloužit případ, kdy se systém učí rozeznat z řady fotek druh zvířete (například kočka). Naopak regrese může například sloužit k odhadu hojnosti úrody pole (reálné číslo) na základě venkovní teploty, vlhkosti vzduchu, půdy a jiných faktorů. Detailnější popis klasifikace a regrese dále popsány v sekcích 2.9 a 2.10. Funkce pro klasifikaci rozděluje kolekci dat na různé třídy. Regrese hledá souvislosti mezi všemi proměnnými a vytváří funkci, která popisuje vztah mezi vstupními a výstupními hodnotami.

2.2.2 Učení bez učitele

Pomocí neanotovaných dat se stroj sám učí detekovat vzory a strukturu v datech bez dozoru učitele. Učení bez učitele se dělí na podúkoly jako:

Shluková analýza (clustering) – rozdělení prvků do skupin na základě vlastností, které mají jednotlivé prvky mezi sebou společné, prvky v jednotlivých skupinách si jsou mezi sebou podobnější než prvky ve skupinách jiných. Příkladem speciálního druhu shlukové



Obrázek 2.3: Aktivační funkce používané v perceptronech. Upravený obrázek, převzatý z internetu[2]

analýzy je **k-means clustering**. Číslo k určuje počet shluků. Centroid je bod, který tvoří střed shluku. Bod patří do některého shluku podle toho, ke kterému centroidu leží nejblíže. Na začátku se náhodně inicializuje velikost shluků a pozice shluků. Pozice centroidů se optimalizuje pomocí algoritmu. Během každé iterace se přepočítají aritmetické průměry mezi všemi body, čímž se získá nová pozice centroidu. Pokračuje se, dokud se nedosáhne kritéria konvergence, například předem definovaného počtu iterací. [20]

Redukce dimenzionality – zobrazení prvku z vysokodimenzionálního prostoru do prostoru s méně dimenzemi. Příkladem využití učení bez učitele na elektronických trzích pomocí shlukové analýzy je sdružování zákazníků nebo trhů do segmentů pro účely lepší komunikace s cílovými skupinami [19]. Na obrázku 2.2 lze vidět příklad shlukové analýzy. Architektury neuronových sítí se mohou skládat z více různých metod, příkladem neuronových sítí, které využívají učení bez učitele i s učitelem jsou generativní adverzní sítě 2.7.3.

2.3 Co to je neuron a neuronová síť

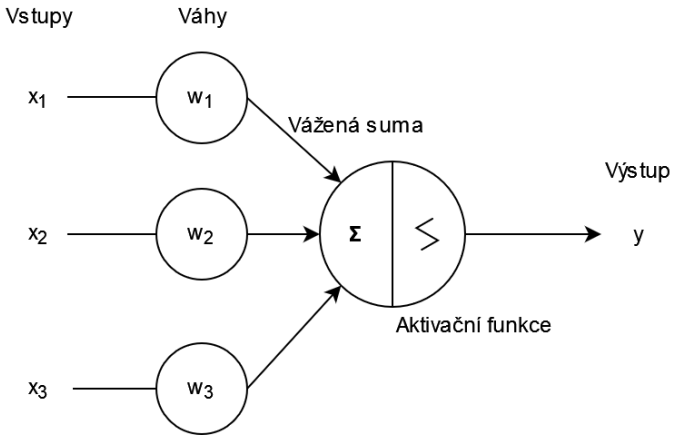
Jak může název vypovídat, **umělý neuron** funguje jako abstrakce živočišné nervové buňky. Umělé neurony se společně pojí do shromáždění, které se nazývá **neuronová síť**. Schopnost sítě zpracovávat informace tkví ve váhách jednotlivých propojení, které se optimalizují postupnou adaptací a učením se na kolekci trénovacích vzorů.

Na obrázku 2.4 můžeme vidět jednotlivé části umělého neuronu (perceptronu). Jednotlivá spojení sítě jsou modelována pomocí vstupů, které jsou násobeny váhami předtím, než jsou poslány do těla buňky. Vážené vstupy jsou sečteny, výsledkem je vážená suma, čímž se spustí aktivace uzlu. Perceptron plní funkci binárního klasifikátoru. Perceptron většinou obsahuje jednu ze tří **aktivačních funkcí** 2.5. Mezi příklady aktivačních funkcí patří **kroková funkce**, **sigmoid funkce** a **signum funkce**. Jednotlivé aktivační funkce jsou vidět na obrázku 2.3. [15]

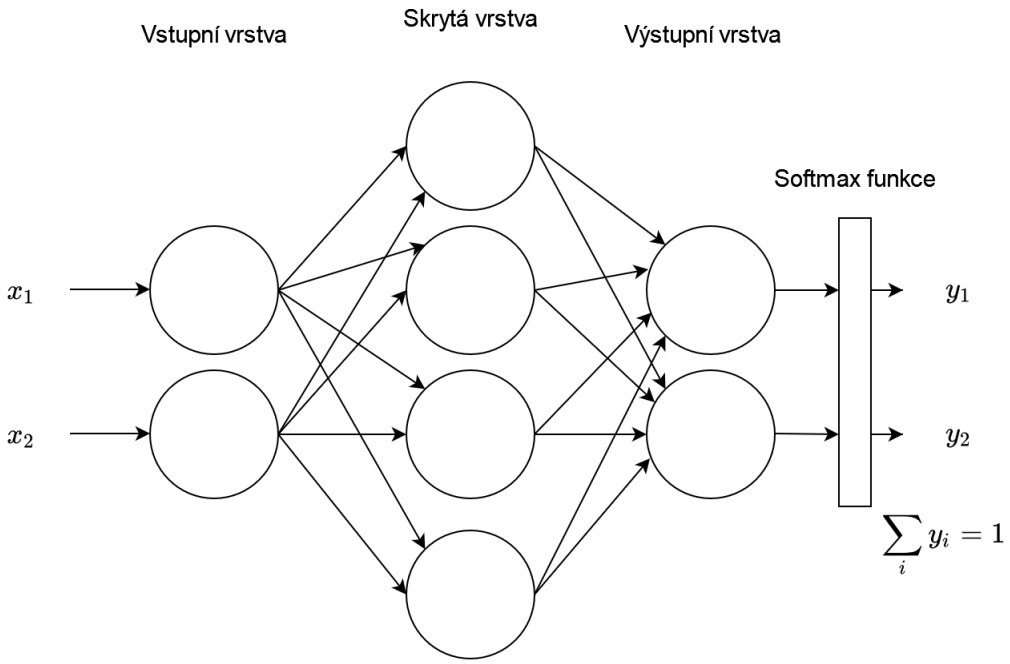
Neuronová síť slouží jako pojem pro označení systému vzájemně propojených umělých neuronů, který může nabývat velikosti od jednoho uzlu až po množinu uzlů. Neurony jsou uspořádány ve formě vrstev. Pokud je každý jeden uzel propojený s každým jiným uzlem v další vrstvě, nazývá se vrstva **plně propojená** (fully connected) [15].

Na obrázku 2.5 lze vidět neuronová síť, která obsahuje tři vrstvy. Informace v ní postupně prochází neuronovou sítí od **vstupní vrstvy** přes **skrytou vrstvu** až nakonec jako poslední projdou **vrstvou výstupní**. Název skrytá vrstva odkazuje na skutečnost, že skryté vrstvy nejsou přímo vidět ze vstupu ani výstupu sítě, tudíž neznáme z vnějšího pohledu jejich pravé hodnoty. Umožňují extrahovat ze svého vstupu statistiky vyššího řádu.

[2] https://www.javatpoint.com/perceptron-in-machine-learning



Obrázek 2.4: Jednoduchý umělý neuron (perceptron)



Obrázek 2.5: Jednoduchá dopředná, plně propojená neuronová síť, za výstupní vrstvou následuje softmax funkce 2.5.5, která výsledky výstupní vrstvy přemění na pravděpodobnosti, kdy se suma všech softmax výstupů rovná 1.

Obecně se síť s více vrstvami nazývá **vícevrstvá neuronová síť**. Neuronovou síť skládající se z jedné výstupní vrstvy, nazveme **jednovrstvá neuronová síť**. **Dopředná síť** (feed-forward network) disponuje propojeními směřujícími vždy směrem k výstupní vrstvě [17].

2.4 Rozdíl mezi hlubokým a nehlubokým učením

Velké množství technologie se inspirovalo přírodou. Je tedy přirozené, že se člověk dále inspiroval lidským mozkem a jeho architekturou k vytvoření inteligentního stroje, který nazýváme **umělá neuronová síť**. Umělé neuronové sítě tvoří jádro hlubokého učení. Jejich vlastnosti a schopnost se efektivně dynamicky přizpůsobovat řešeným úlohám z nich dělá ideální nástroj k řešení problémů strojového učení. [14]

Hluboké učení je charakteristické tím, že přidává více vrstev mezi vstup a výstup. Parametry modelu se určí z výstupů předchozích vrstev, nikoliv přímo z příznaků trénovacích dat [6]. Z tohoto důvodu se hluboké neuronové sítě lépe vyrovnávají s daty trpícími šumem a nestrukturovanými daty. V závislosti na typu dat a výběru architektury existují různé mechanismy pro učení příznaků. Nehluboké učení se dá představit jako síť s jednou nebo málo vrstvami mezi vstupem a výstupem neuronové sítě. Příkladem hlubokého učení mohou být **autoenkodéry** 2.7.4 nebo **konvoluční neuronové sítě** 2.7.1. Příkladem nehlubokého učení může být již dříve zmíněný SVM. Neuronová síť, kterou v této práci implementuji je taky příkladem nehlubokého učení.

2.5 Aktivační funkce

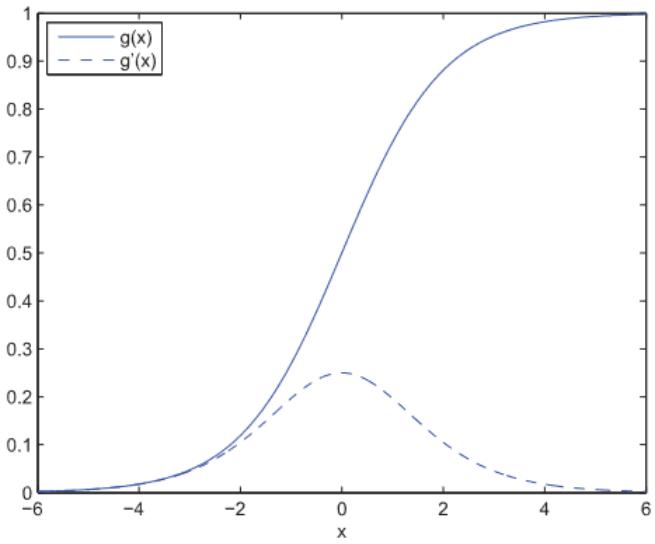
Mapování vstupu na výstup je základní funkcí všech aktivačních funkcí ve všech typech neuronových sítí. Vstupní hodnota se určí výpočtem vážené sumy vstupů neuronů násobených jejich vahami. Výstupní hodnota se určí podle konkrétní vybrané aktivační funkce. To znamená, že aktivační funkce rozhoduje o tom, zda má neuron vyslat impuls s ohledem na určitý vstup vytvořením odpovídajícího výstupu. [3] Mezi příklady aktivačních funkcí patří:

2.5.1 Sigmoid funkce

Na vstupu **sigmoidní funkce** jsou reálná čísla, zatímco výstup je omezený na hodnoty mezi jedničkou a nulou [3].

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

V metodách učení založených na gradientu se při zpětném šíření váhy sítě aktualizují úměrně hodnotě gradientu (parciální derivace **objektivní funkce** 2.6 vzhledem k aktuálním vahám) po každé trénovací **epoše** [5]. Sigmoid funkci lze použít v nehlubokém učení, v učení s omezeným počtem vrstev a nebo v poslední výstupní vrstvě, nelze ji používat v sítích s velkým množstvím vrstev, protože poté nastává problém s gradienty. Tento problém spočívá v tom, že velké změny na vstupu způsobují malé změny na výstupu, jak jde vidět podle její derivace na obrázku 2.6. Postupně se začne derivace snižovat, dokud nedosáhne nuly. Při hodnotách blízkých nule se zabrání tomu, aby síť aktualizovala svoje váhy. Tento problém se nazývá **problém mizejícího gradientu** (vanishing gradient problem) [13].



Obrázek 2.6: Aktivační funkce sigmoid, převzato z [13], upraveno

2.5.2 Tanh funkce

Funkce s hyperbolickým tangens se dá jednoduše zadefinovat jako poměr mezi hyperbolickým sinem a cosinem.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

Hyperbolický tangens vrací výstup v rozsahu (-1)–1. Neuronové sítě využívající tanh oproti sigmoid funkci konvergují dříve a klasifikační úlohy dosahují vyšších úspěšností. Problém s mizejícím gradientem se u tanh funkce vyskytuje dále. [13]

2.5.3 ReLU funkce

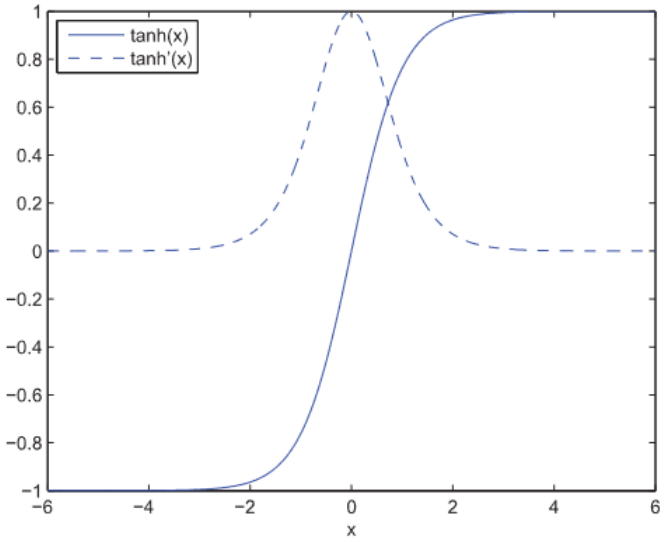
ReLU funkce (Rectified Linear Unit) a její variace jsou jedny z nejoblíbenějších aktivačních funkcí využívaných v hlubokých neuronových sítích [13]. Převádí celé hodnoty vstupu na nezáporná čísla.

$$g(x) = max(0, x) = \begin{cases} x, & \text{pokud } x \geq 0. \\ 0, & \text{jinak.} \end{cases} \tag{2.3}$$

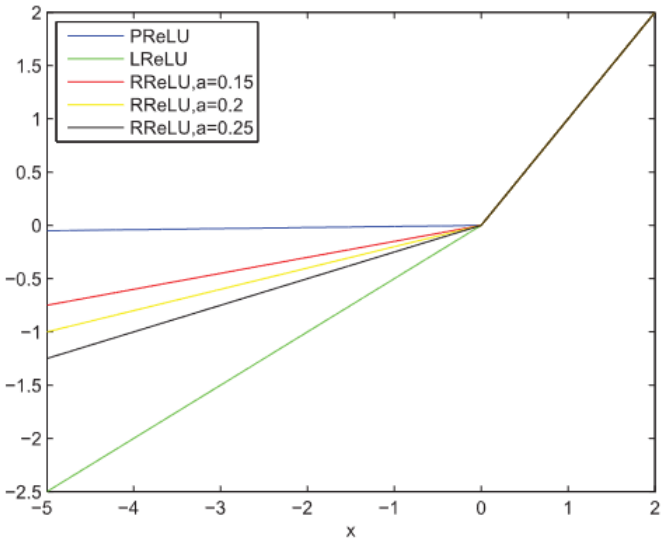
Mezi výhody ReLU patří nízká výpočetní náročnost, rychlejší konvergence než u dříve zmíněných aktivačních funkcí a zamezení možnosti propuknutí mizejícího gradientu. Potýká se také se závažnými problémy, například při předávání vysokých gradientů se mohou v rámci ReLU zaktualizovat váhy takovým způsobem, že neuron se již znovu nezaktivuje a dojde k jeho „smrti“. [13]

2.5.4 Leaky ReLU funkce

Leaky ReLU se vyhýbá problému s mrtvými neurony tím, že záporné hodnoty vynásobí drobnou, nenulovou hodnotou, které se říká **leak factor** (volně přeloženo jako faktor úniku).



Obrázek 2.7: Aktivační funkce tanh, převzato z [13], upraveno



Obrázek 2.8: Různé variace ReLU lišící se v parametru leak factor, převzato z [13], upraveno

V rovnici 2.4 se leak factor značí s písmenem m, na obrázku 2.8 se značí s písmenem a. Pro Leaky ReLU se obvykle používá hodnota v řádech okolo 0.001. Tím se zbaví možnosti nulových gradientů v záporných hodnotách a žádný neuron se nemůže stát mrtvým. [3]

$$g(x) = max(0, x) = \begin{cases} x, & \text{pokud } x \geq 0. \\ mx, & \text{jinak.} \end{cases} \tag{2.4}$$

2.5.5 Softmax funkce

Softmax aktivační funkce se často používá v klasifikačních problémech jako poslední funkce v neuronové síti. Na vstup přijímá všechny výstupy vrstvy před ní a na výstup vrací čísla v rozmezí 0–1. Suma všech čísel na výstupu dává hodnotu 1, to znamená, že softmax vrací pravděpodobnosti pro jednotlivé hodnoty. Větší čísla mají vyšší hodnotu pravděpodobnosti než čísla menší. Tím, že převádí libovolné rozdělení na pravděpodobnostní distribuci šetří práci a dovoluje jednodušší práci s hodnotami všech vstupů. [3]

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \tag{2.5}$$

2.6 Objektivní funkce

Objektivní funkce (loss functions) jsou penalizační funkce, které objektivně hodnotí správnost výstupu neuronové sítě vůči správnému výsledku pro daný vzorek, čili jak dobrý model vytváří pro danou kolekci dat. Cílem může být buď minimalizovat nebo maximalizovat hodnotu funkce. Objektivní funkce se musí vybírat podle řešeného problému. [3]

2.6.1 Střední kvadratická chyba

Střední kvadratická chyba (mean square error – MSE) je druh objektivní funkce, která se často používá u regresních modelů. Střední kvadratická chyba modelu, vzhledem k testovací množině, je průměrem kvadratických chyb předpovědi všech instancí v testovací množině. Chyba předpovědi je rozdíl mezi skutečnou hodnotou a předpovězenou hodnotou pro danou instanci. [11]

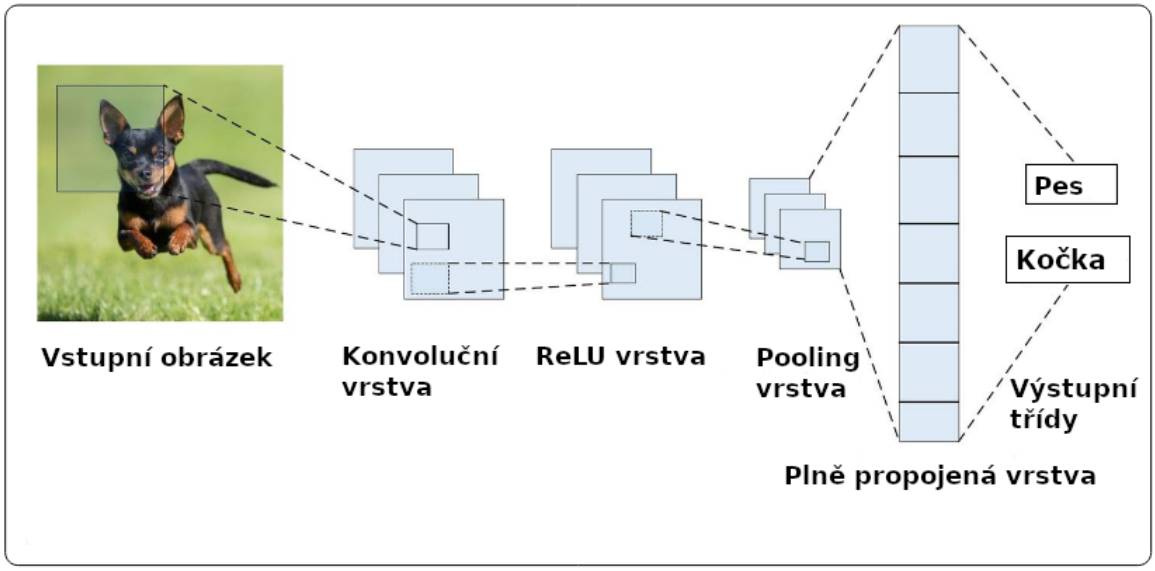
$$MSE = \frac{\sum_{i=1}^{n}(y_i - \lambda(x_i))^2}{n} \tag{2.6}$$

kde y_i je skutečná cílová hodnota pro testovací instanci x_i, $\lambda(x_i)$ je předpovězená cílová hodnota pro testovací instanci x_i a n je počet testovacích instancí.

2.6.2 Objektivní funkce křížové entropie

Křížová entropie (cross entropy function) se také nazývá logaritmickou ztrátovou funkcí. Měří výkonnost klasifikačního modelu, jehož výstupem je hodnota pravděpodobnosti mezi 0 a 1. Proto se požívá často ve spolupráci se softmax aktivační funkcí. Ztráta křížové entropie roste s tím, jak se předpovězená pravděpodobnost vychyluje od skutečné hodnoty značky. Používá se při úpravě vah modelu během trénování. [3]

$$H(p, y) = -\sum_{i=1}^{N} y_i \log(p_i) \tag{2.7}$$



Obrázek 2.9: Příklad konvoluční neuronové sítě, která funguje jako klasifikátor, převzato z [3], upraveno

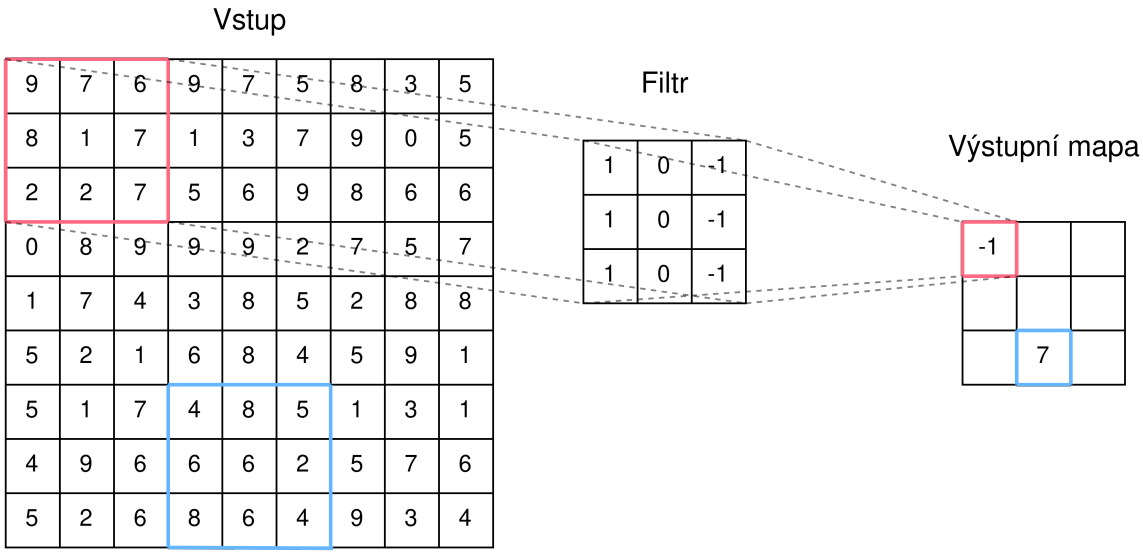
2.7 Příklady architektur neuronových sítí

Každý druh architektury neuronových sítí je specifický svým typem vrstev, neurony a spojeními, které jednotlivé neurony používají. I přes teoretickou možnost využití každé architektury pro každý úkol jsou některé druhy architektur vhodnější pro určité specifické úkoly, například pokud se jedná o časovou řadu nebo práci s obrázky [19]. V sekci 2.3 jsem již zmínil jednoduchý perceptron a jednoduché feedforward neuronové sítě jakožto možné příklady. V dalších podsekcích se věnuji dalším příkladům skupin architektur.

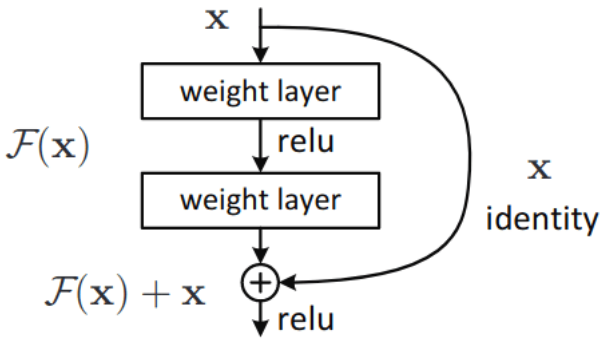
2.7.1 Konvoluční neuronové sítě

Konvoluční neuronové sítě – CNN jsou architekturou, která se využívá především pro úlohy, které souvisejí s počítačovým viděním a rozpoznáváním řeči. Jsou schopny řešit úlohy nad soubory dat u kterých nejsou sloupce ani řádky zaměnitelné – existují prostorové vztahy u těchto dat (například obrazová data). Skládají se z řad seskupení, které umožňují hierarchické učení se příznaků podle příslušné úlohy modelování. Při úloze rozpoznávání objektů na obraze je několik prvních vrstev zodpovědných za extrakci základních příznaků ve formě hran a rohů. Tyto příznaky jsou v posledních vrstvách agregovány do příznaků složitějších, které již připomínají objekty (například druhy zvířat) [19].

CNN na obrázku 2.9 využívá hned několika vrstev. **Konvoluční vrstva** je nejdůležitější komponentou z celé CNN architektury. Skládá se z kolekce **konvolučních filtrů** (takzvaných jader). Vstupní obraz se konvolvuje pomocí těchto filtrů a vytváří se výstupní mapa příznaků. Tyto filtry jsou složené z polí/mřížek, které obsahují určité množství vah. Na začátku trénovacího procesu jsou tyto váhy náhodně nebo pomocí některé inicializační metody inicializovány, během trénování se tyto váhy upravují, čímž se filtr učí extrahovat významné příznaky.[3]



Obrázek 2.10: Znázornění operace konvoluce

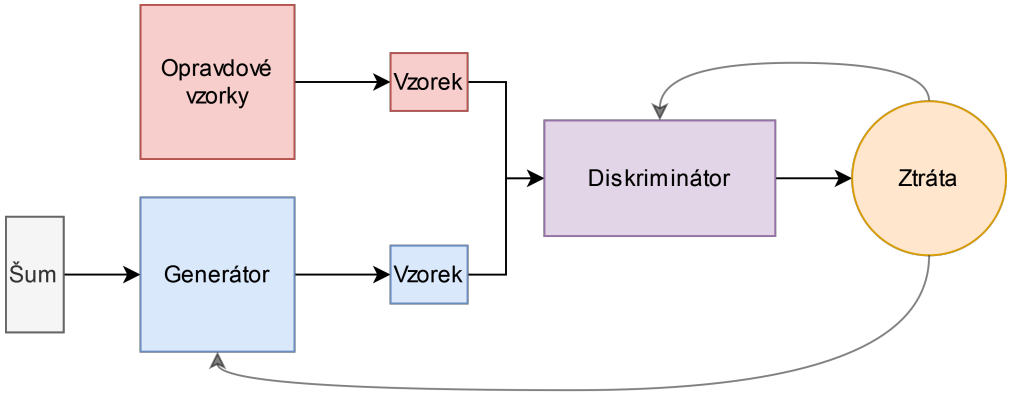


Obrázek 2.11: Jeden reziduální blok s ResNet spojením, převzato z [18]

Na obrázku 2.10 je znázorněná konvoluce. Princip výpočtu spočívá v pronásobení vstupních hodnot s hodnotami filtru na stejných pozicích. Tyto vynásobené hodnoty se následně sečtou a tím se určí výsledná hodnota pro jednu buňku výstupní mapy. Velikost výstupní mapy záleží na velikosti vstupu, na velikosti filtru a na délce **kroku** (stride). V tomto případě se krok rovná 3, krok určuje o kolik posouváme vstupní okénko mezi výpočty. Nejdříve se posouvá horizonálně dokud to je možné, poté se posune vertikálně. Dále je **nulová výplň** (zero–padding) nastavena na nulu. Pokud by byla nastavena na 1, byla by okolo vstupu přidána vrstva nul.

Další vrstvou je **pooling vrstva**, která provádí podvzorkování z příznakových map, které bývají výstupem konvolučních vrstev. Během procesu zmenšování map příznaků zachovává dominantní příznaky. Existují různé metody podvzorkování. Mezi nejznámější a nejčastěji používané patří max, min a global average pooling (GAP).[3]

Objektivní a aktivační funkce jsou vybírány podle konkrétní potřeby modelu pro hledané řešení. ReLU je nejčastěji používanou aktivační funkcí v kontextu CNN.[3]



Obrázek 2.12: Příklad generativní adverzní sítě

2.7.2 Reziduální neuronové sítě

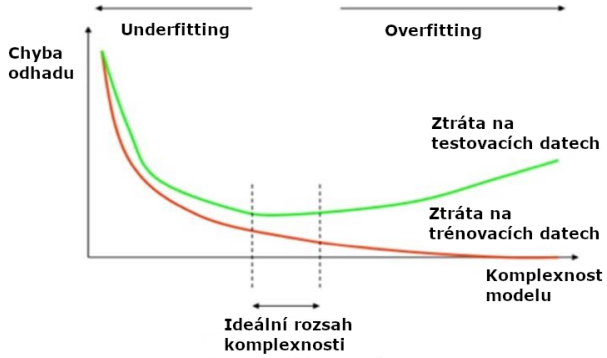
Reziduální sítě – ResNet vznikly jako odlišný přístup k trénování hlubokého učení, protože trénování hlubokého učení obecně trvá poměrně dlouho a je omezeno na určitý počet vrstev. ResNet mohou provést přeskočení spojení nebo mohou využít zkratek. Výhodou modelu ResNet ve srovnání s jinými architektonickými modely je, že výkon této architektury neklesá, i když se architektura „prohlubuje“. Využitím ResNet se stávají výpočty lehčí a schopnost trénovat sítě se zlepšuje. Architektura ResNet je realizována vynecháním spojení na dvou až třech vrstvách obsahujících ReLU a dávkovou normalizaci (**batch normalizaci**). ResNet dosahuje lepších výsledků při klasifikaci obrazu než jiné modely, z čehož vyplývá, že kvalitně extrahuje obrazové příznaky.[40]

Obrázek 2.11 ukazuje příklad reziduálního bloku s ResNet spojením. Při normálním průchodu jednotlivými vrstvami získáme ze vstupu x výstup F(x). Pokud se využije ResNet spojení dojde k součtu vstupu x s F(x). Aby šlo přičíst x k F(x), je potřebné implementovat 1x1 konvoluční vrstvu, která x převede na stejnou dimenzi, jako je F(x). ResNet spojení pomáhá s problémem mizejícího gradientu, protože pomocí ResNet spojení, přeskočením vrstev, se obnovuje gradient. [18]

2.7.3 Generativní adverzní sítě

Generativní adverzní neuronové sítě – GAN jsou soustavou dvou neuronových sítí, jejichž cílem je naučit se pravděpodobnostní rozdělení nad sadou trénovacích dat, aby síť mohla náhodně generovat nové vzorky dat s určitou variabilitou. Za tímto účelem se GAN skládají ze dvou konkurenčních sítí. První sítí je **generátor**, která zachycuje pravděpodobnostní rozdělení vstupních dat a generuje nové příklady. Druhou sítí je **diskriminátor**, která se snaží rozlišit skutečné příklady od uměle generovaných. Obě sítě jsou společně trénovány v nekooperativní hře s nulovým součtem (zero–sum game), kdy zisk pro jednu síť znamená ztrátu pro druhou, dokud diskriminátor již nedokáže rozlišit oba typy vzorků. [19]

Architektura GAN je znázorněna na obrázku 2.12. Generátor přijímá na vstup náhodný šum, pomocí kterého se snaží generovat obraz. Obraz je předán diskriminátoru, ten se rozhodne, zda je vstup falešný nebo ne. Generátor si upraví parametry vzhledem k výsledku. Diskriminátor si vytváří pravděpodobnostní funkce podle toho, jak přijímá falešné a pravé vzorky. Parametry obou sítí jsou aktualizovány pomocí **zpětného šíření chyby**. [19]



Obrázek 2.13: Vliv komplexnosti modelu na ztráty modelu, převzato z [2], upraveno

2.7.4 Autoenkodéry

Autoenkodéry se obvykle skládají z fáze kódování, v níž je vstup komprimován do nízkodimenzionální reprezentace, a fáze dekódování, v níž se síť snaží rekonstruovat původní vstup z naučených rysů. Tímto způsobem je síť nucena zachovat smysluplné informace v latentní reprezentaci redukováním dimenzionality a zároveň ignorovat irelevantní šum. Autoenkodéry se běžně používají pro učení příznaků učením bez učitele. Autoenkodéry se mohou například používat pro generování nových dat na základě enkódovaných dat. [19]

2.8 Trénování neuronových sítí

Celý proces trénování neuronových sítí je založený na experimentování. **Hyperparametry**, architektura a celkový přístup se mohou měnit v rámci hledání nejlepšího řešení. Jedním z nejdůležitějších předpokladů pro úspěšný proces trénování je mít kvalitní a hojnou kolekci dat, nad kterou bude síť trénována. Datová sada se dále dělí na trénovací, na validační a na testovací sadu. Síť je trénována na trénovací sadě a testovací sada se používá jako měřítko kvality trénování, je během trénovacího procesu neuronové síti skryta. Dále je důležitá volba správné architektury neuronové sítě a objektivní funkce, která se vybírá podle charakteristik řešeného problému.

Jeden průchod celou trénovací sadou, se nazývá **epocha**. Trénování sítě založené na **gradientním sestupu** 2.11.2 se dělí podle toho, jestli se gradienty počítají po průchodu celou trénovací sadou – **dávkový gradientní sestup** (batch gradient descent) nebo jestli se gradienty počítají po průchodu částí trénovací sady – **mini-dávkový gradientní sestup** (mini-batch gradient descent). Čili během batch gradientního sestupu se gradienty aktualizují jenom jednou během epochy a počet aktualizací gradientů u mini-batch gradientního sestupu závisí na velikosti trénovací sady a **velikosti dávek** (batch size). Efektivita trénovaného modelu se vyhodnocuje pomocí **trénovací ztráty** (training loss) a **testovací ztráty** (test loss). Trénovací ztráta se měří nad jednotlivými dávkami a po každém výpočtu se propaguje dále do sítě a způsobuje úpravu parametrů jednotlivých neuronů. Validační sada se používá pro vypilování jednotlivých hyperparametrů modelu. Testovací ztráta se měří na konci trénování modelu, aby se ověřila schopnost modelu generalizovat. Mezi hyperparametry sítě patří parametry jako jsou **koeficient učení** (learning rate), **velikost dávek** (batch size) a **míra penalizace objektivní funkce**.

Proces trénování je složitý, příklad problému komplexnosti systému a jeho vliv na kvalitu trénování je vidět na obrázku 2.13. **Underfitting** je jev, který se děje, pokud model není dostatečně natrénovaný a není schopný zachytit závislosti jednotlivých příznaků trénovacích ani nezávislých dat. **Overfitting** se děje, když je model přetrénovaný na trénovací sadě a není schopný dobře generalizovat, čili se naučil závislosti specifické pro trénovací data, které u nezávislých už neexistují. Proti underfitting se dá bojovat přidáním na komplexnosti modelu, navýšením počtu trénovacích epoch nebo odstraněním šumu z dat. Proti overfittingu se zase dá bojovat redukováním komplexnosti modelu, navýšením počtu trénovacích dat, augmentací dat (představením šumu, zrcadlením obrazových dat atd.) nebo různými metodami regularizace.

2.9 Klasifikace

Klasifikace je problém, který se zabývá automatickým přiřazením značky k neanotovaným datům. Klasifikace je řešená klasifikačním algoritmem. Na počátku přijímá na vstup kolekce anotovaných dat, vytvoří model, který je schopný přijmout neanotovaná data jako vstup a jim vyhodnotit značku.

Klasifikace se dělí na **binární klasifikaci**, **vícetřídní klasifikaci** a **víceznačkovou klasifikaci**. Binární klasifikace se vyznačuje tím, že se množina tříd skládá ze dvou tříd. Příklad pro rozeznávání nevyžádaných emailových zpráv: je spam/není spam. Na rozdíl od binární klasifikace se vícetřídní klasifikace, podle názvu, skládá z více tříd. Víceznačková klasifikace se používá v situacích, kdy lze různá slova asociovat s různými značkami nebo třídami. Spolu s vícetřídní klasifikací se od sebe liší v tom, že víceznačková klasifikace předvídá více různých vzájemně se nevylučujících značek nebo tříd. Třídy ve vícetřídní klasifikaci se navzájem vylučují, tudíž jeden prvek může spadat pouze do jedné třídy [39].

2.10 Regrese

Regrese je problém, který se zabývá předvídáním určité reálné hodnoty pro neanotovaná data předávaná na vstup. Nejjednodušším druhem regrese je **lineární regrese**, která je zároveň obecně jedna z nejoblíbenějších modelovacích technik ve světě strojového učení [39]. Rovnice **jednoduché lineární regrese** lze napsat jako

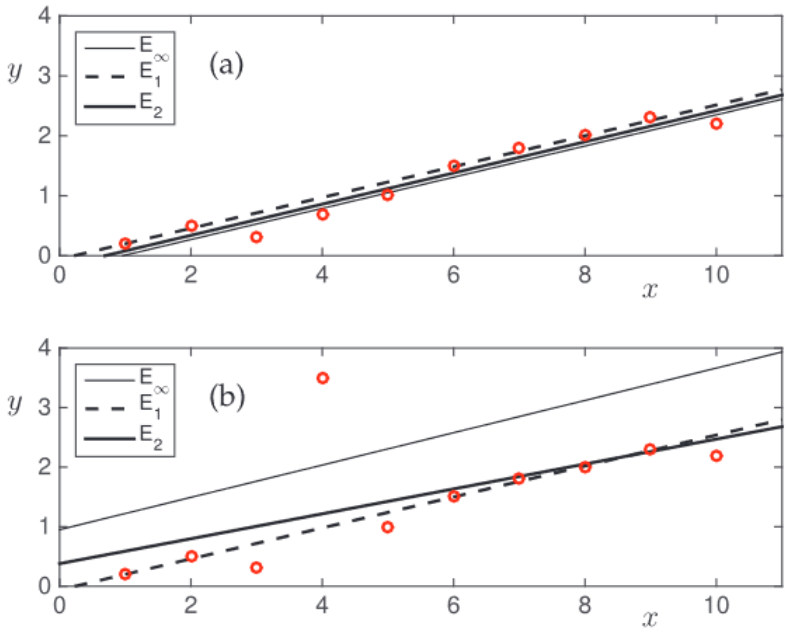
$$y = a + bx \tag{2.8}$$

kde y je **závislá proměnná**, a je **posun**/konstantní člen, b je koeficient sklonu přímky a x je **nezávislá proměnná**. Pokud existuje na vstupu více nezávislých proměnných, tak se jedná o **vícenásobnou lineární regresi**.

$$y = a + b_1x_1 + b_2x_2 + b_3x_3 \dots \tag{2.9}$$

Zde má každá nezávislá proměnná x_1, x_2 a x_3 vlastní koeficient b_1, b_2 a b_3. Příkladem vícenásobné lineární regrese je odhadnout cenu domu podle vlastností/charakteristik domu jako jsou počet místností, rozloha a lokace. Vlastnosti domu fungují jako nezávislé proměnné a výsledná cena domu jako závislá proměnná. K predikci správné hodnoty závislé proměnné se musí určit vhodné gradienty a posuv.

Na obrázku 2.14 jsou na grafu a vidět různé příklady regresních funkcí, kdy všechny dobře **fitují** (vytvoření co nejlépe odpovídající funkce datům) vzorky. Na grafu b je vidět,



Obrázek 2.14: Regrese, převzato z [7]

jak z jednotlivých funkcí jediná E_1 dobře fituje vzorky. Funkce s plnou čarou má zvýšenou hodnotu posuvu kvůli odlehlému vzorku. Funkce s plnou tučnou čarou má vychýlený gradient a ve výsledku je celá vychýlená z optimální pozice.

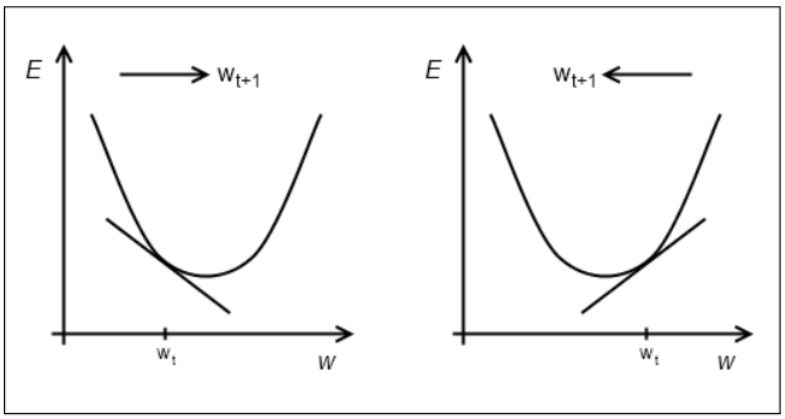
2.11 Optimalizační metody

Optimalizační problémy jsou běžné v mnoha oborech a různých oblastech. U optimalizačních problémů hledáme řešení, která jsou optimální nebo téměř optimální vzhledem k určitým cílům. Často takovým cílem je nalezení hodnot, které jsou pro řešené funkce minima nebo maxima. Tento postup se dá nazvat procesem hledání extrémů řešených reálných funkcí. Obvykle nejsme schopni řešit problémy jedním krokem, ale postupujeme podle určitého procesu, který nás při řešení problému vede. Často je proces řešení rozdělen do různých kroků, které se postupně provádějí [38].

Cílem optimalizačních metod je nalézt optimální nebo téměř optimální řešení optimalizačních problémů s malou výpočetní náročností. Náročnost optimalizační metody lze určit podle výpočetního času a spotřebované paměti počítače během řešení optimalizační metody. U mnoha optimalizačních metod, zejména u moderních **heuristických metod**, musí existovat kompromis mezi kvalitou řešení a náročností, protože kvalita řešení se navyšuje s rostoucí náročností řešení [37].

2.11.1 Metoda nejmenších čtverců

Metoda nejmenších čtverců minimalizuje sumu čtverců zbytků mezi daným modelem a daty. Lineární metoda nejmenších čtverců s lineárními zbytky má řešení v uzavřeném tvaru, které lze spočítat pomocí derivace zbytku vzhledem ke každé neznámé. Je běžně používaná



Obrázek 2.15: Algoritmus zpětného šíření chyby, převzato [16]

v technických a aplikovaných vědách pro fitování polynomických funkcí. Nelineární metoda nejmenších čtverců obvykle vyžaduje iterační upřesnění založené na aproximaci nelineárních nejmenších čtverců lineárními nejmenšími čtverci v každé iteraci [7].

2.11.2 Gradientní sestup a algoritmus zpětného šíření

Gradientní sestup je optimalizační metoda pro systémy s vysokou dimenzionalitou. Minimalizuje zbytky výpočtem gradientu dané fitovací funkce. Protože se často používá ve vysokodimenzionálních prostředích, je náchylná k nalezení pouze lokálního minima. Mezi zásadní inovace pro aplikace s velkými objemy dat využívající gradientní sestup patří **stochastický gradientní sestup** (Stochastic Gradient Descent – SGD) a **algoritmus zpětného šíření chyby** (Backpropagation – BP). Tyto dva přístupy se často používají společně k úspěšnému natrénování neuronové sítě. [7]

Stochastický gradientní sestup – SGD je jedním z nejběžnějších algoritmů, který se používá k optimalizaci logické sítě. Stochastický gradientní sestup vyžaduje opakovaný odhad chyby na základě současného stavu modelu, proto se používá objektivní funkce, aby se opakovaně obnovovala hodnota vah a snižovala chyba příštích odhadů. Normální gradientní sestup musí počítat gradient pro všechny vstupní vzorky, při velkých kolekcích dat se tato operace stává velice výpočetně náročnou. Stochastický gradientní sestup razantně snižuje výpočetní náročnost tím, že náhodně vybere vzorek, u kterého vypočítá gradient. Tímto dojde k celkovému urychlení trénovacího procesu. [40]

Algoritmus zpětného šíření – BP modifikuje váhy neuronové sítě s cílem najít lokální minimum objektivní funkce. Váhy se mění v opačném směru než je gradient, dokud se nedosáhne lokálního minima. Je-li parciální derivace záporná, váha se zvýší, je-li parciální derivace kladná, váha se sníží, jak je vidět na obrázku 2.15. Tím je zajištěno dosažení lokálního minima. Všechny parciální derivace se počítají pomocí řetězového pravidla, to znamená, že se počítá postupně po vrstvách. [16]

2.11.3 Simulované žíhání

Simulované žíhání je pravděpodobnostní heuristické vyhledávání pro řešení optimalizačních problémů, používané s velkým úspěchem na reálných problémech. Ve své standardní

podobě simulované žíhání má dva parametry: **počáteční teplotu** a **faktor chlazení**. Algoritmus založený na tomto konceptu najde globální minima, pokud se nechá běžet dostatečně dlouho, pro problémy nelineárního programování se spojitými, diskrétními a celočíselnými proměnnými. [44]

Základním mechanismem implementace procesu žíhání je generování náhodných bodů v okolí aktuálního nejlepšího bodu a vyhodnocení problémových funkcí v těchto bodech. Pokud je hodnota **objektivní funkce** menší než její aktuální nejlepší hodnota, je bod přijat a hodnota nejlepší funkce je aktualizována. Pokud je hodnota funkce vyšší než dosud známá nejlepší hodnota, pak je bod někdy přijat a někdy zamítnut. Zda je bod přijat závisí na hodnotě funkce hustoty pravděpodobnosti **Boltzmannova rozdělení** [4].

Pokud má tato funkce hustoty pravděpodobnosti větší hodnotu než náhodné číslo, je zkušební bod přijat jako nejlepší řešení, i když je hodnota jeho funkce vyšší než známá nejlepší hodnota. Při výpočtu funkce hustoty pravděpodobnosti se používá teplota. Pro optimalizační problém může být tato teplota cílem pro **objektivní funkci** jako optimální hodnota. [4]

Zpočátku je zvolena větší cílová hodnota. S navyšujícím počtem pokusů se cílová hodnota (teplota) snižuje, tento jev se nazývá chlazení, a proces se ukončí po velkém počtu pokusů. Pravděpodobnost přijetí se postupně blíží k nule, podle toho jak se snižuje teplota. V počátečních fázích tedy metoda někdy přijímá horší návrhy, zatímco v závěrečných fázích jsou horší návrhy téměř vždy zamítnuty. Tato optimalizační metoda zabraňuje uvíznutí v bodě lokálního minima. [4]

2.12 Regularizace

Bez ohledu na mnohé úspěchy v aplikacích neuronových sítí zůstává overfitting stále jedním z hlavních problémů, které je třeba ve výzkumné komunitě strojového učení překonat. Ke snížení vlivu tohoto problému bylo navrženo několik přístupů. Široce používanou metodou regularizace je přidání regularizačního členu k chybové funkci tak, aby byla minimalizována celková chybová funkce.

Příkladem takového typu regularizace jsou **L1 a L2 regularizace**. Obě tyto metody penalizují výsledky objektivní funkce. Rozdíl mezi L1 a L2 regularizací je ten, že při výpočtu konečné objektivní funkce k původní objektivní funkci L2 přičítá druhou mocninu hodnot vah a L1 přičítá absolutní hodnotu vah. Toto znamená, že ve výsledku L1 může vynulovat některé výsledky, mezitím L2 více penalizuje vysoké výsledky, ale nikdy je nevynuluje. [34]

Dalším druhem regularizace je **dropout**. Dropout je druh regularizace, který se inspiroval evoluční genetikou. Nový potomek přijme polovinu genů od jednoho a druhou polovinu od druhého rodiče. Aby byly geny odolnější, musí se naučit dobře spolupracovat s jinou sadou genů. Myšlenka dropout spočívá ve vypínání neuronů v neuronové síti a to s určitou pravděpodobností. Když je některý neuron vyřazen, všechny jeho příchozí a odchozí spojení budou zanedbána. Cílem náhodného vypínání neuronů je umožnit každému neuronu, aby se sám naučil něco, co za něj uměl jiný neuron. [34]

Kapitola 3

Data, práce nad nimi a jejich uchování

Význam kolekcí dat pro výzkum v oblasti strojového učení nelze podceňovat. Datové sady jsou považovány za limitující faktor vývoje algoritmů a vědeckého pokroku. K efektivnímu trénování je potřeba správné množství kvalitních dat. Abychom si mohli vytvořit kolekci dat, musíme samotná data nějakým způsobem pořídit, dále je potřeba tato data nějakým vhodným, často individuálním, způsobem zpracovat a upravit do podoby, ve které se urychlí proces trénování a samotná data se zkvalitní. [33]

V sekci 3.2 jsou podrobněji popsány zvolené datové sady.

3.1 Způsoby získání dat

Cílem získávání dat je najít soubory dat, které lze použít k trénování modelů strojového učení. V literatuře se objevují převážně tři přístupy: objevování dat, rozšiřování dat a generování dat. Objevování dat je nezbytné, když chceme sdílet nebo vyhledávat nové datové sady, a postupně nabývá na významu, protože je k dispozici stále více datových sad. Rozšiřování dat doplňuje objevování dat, kdy se stávající datové sady rozšiřují přidáním dalších externích dat. Generování dat lze použít v případě, že není k dispozici žádná externí datová sada, ale místo toho je možné generovat syntetické datové sady. [36]

3.2 Datové sady

Pro tuto práci jsou relevantní datové sady, které obsahují anotované identity řečníků, pro které máme dostatečný počet audio nahrávek a zároveň obrazů obličeje. Tyto datové sady jsem zvolil, protože převzaté modely z kapitoly 4 jsou na těchto datových sadách předtrénovány.

3.2.1 Audio datové sady

VoxCeleb [26] a **VoxCeleb2** [10] jsou datové sady, které obsahují nahrávky různých celebrit získané z platformy YouTube. Statistiky jednotlivých datových sad jsou porovnány v tabulce 3.1. Vývojová datová sada VoxCeleb2 se nepřekrývá s identitami z VoxCeleb. Nahrávky jsou pořízeny z videí z velkého množství náročných prostředí, proto jsou degradovány povídáním na pozadí a dalšími jinými rušivými faktory. Komplexní proces získávání

Dataset	VoxCeleb	VoxCeleb2
Počet identit	1251	6112
Poměr muž/žena	690/561	3761/2351
Počet různých národností	36	145
Počet nahrávek	153 516	1 128 246
Průměrná délka nahrávek (s)	8.2	7.8

Tabulka 3.1: Srovnání Voxceleb a Voxceleb2

Dataset	Dev	Test	Celkem
Počet identit	5994	118	6112
Počet nahrávek	1 092 009	36 237	1 128 246

Tabulka 3.2: Rozdělení Voxceleb2 na vývojovou a testovací sadu

dat byl složený z několika různých fází. Identity se čerpaly z datových sad VGGFace pro VoxCeleb a z VGGFace 2 pro VoxCeleb2. Nejdříve bylo vyhledáno a staženo 50 nejsledovanějších videí pro všechny VoxCeleb identity a 100 nejsledovanějších videí pro všechny VoxCeleb2 identity. Pomocí specifických detektorů obličejů byly detekovány obličeje, které byly pomocí jiných modelů na rozeznávání obličejů klasifikovány, jestli patří hledané identitě. Dále bylo detekováno, zda je tato identita řečníkem. Protože více různých videí může mít stejný obsah, je nutné zkontrolovat a promazat duplikovaná videa. Identity obdržely své národnosti podle článků z Wikipiedie. [10]

3.2.2 Obrazové datové sady

Jako obrazové datové sady jsem zvolil **VGGFace** [32] a **VGGFace2** [9]. Byly vybrány pro svůj vysoký počet obrazů a společné identity s audio datovými sadami. Proces získání dat pro VGGFace a VGGFace2 datové sady má podobné kroky jako VoxCeleb a VoxCeleb2. Nejdříve bylo získáno 5000000 identit z databáze s věřejně činnými osobami pro VGGFace2, ze stejné databáze byl vybrát subset 5000 identit pro VGGFace, 2500 byli muži a 2500 byly ženy. Bylo vyhledáno a staženo 200 obrazů v Google Image hledání, které byly dále lidmi zkontrolovány, zda jsou unikátní a zda-li identity mají tolik fotek na Google Images. Pro VGGFace2 se zredukoval počet identit na 9244 a pro VGGFace se zredukoval na 2622. Poté byly hledány a staženy další fotky. U VGGFace se pomocí filtru vylepšila čistota datasetu pomocí klasifikátoru, smazaly se duplikáty a datová sada prošla finální manuální filtrací. U VGGFace2 se použilo detekce obličeje, automatické filtrace klasifikátorem, poté se smazaly duplikáty a dataset prošel finální automatickou a manuální filtrací. [9]

Při procházení datových sad byla okamžitě viditelná odlišná kvalita dat. Některé obrazy ani nepatřily správným identitám ve VGGFace. Dále se VGGFace data musela ještě pomocí skriptu oříznout a upravit.

Dataset	VGGFace	VGGFace2
Počet identit	2622	9131
Poměr muž/žena	-	59.3%/40.7%
Počet obrazů	2 600 000+	3 000 000+

Tabulka 3.3: Srovnání VGGFace a VGGFace2

3.3 Normalizace dat

Účinnost a rychlost každého algoritmu učení je do značné míry závislá na metodě normalizace. Hlavním cílem normalizace dat je vytvořit vysokou kvalitu dat, která mohou být dále použita v jakémkoli algoritmu pro učení. Nezpracovaná data jsou náchylná na šum, nekonzistenci a mohou nabývat velkého rozsahu hodnot, proto je třeba je škálovat na jednotný rozsah hodnot, aby se mohl urychlit proces učení. Normalizaci v této práci nevyužívám, ale může být použita ke škálování dat na stejný rozsah hodnot pro každý vstupní příznak, aby se dosáhlo minimalizace zkreslení mezi jednotlivými příznaky mezi sebou. Mezi příklady různých metod normalizace patří: [27]

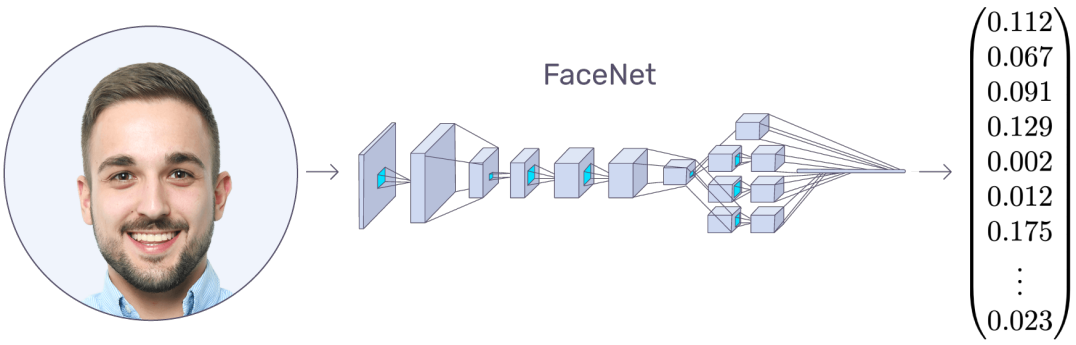
Min-max normalizace je jedna z nejpopulárnějších a nejpoužívanějších normalizačních metod. Tato metoda přeškáluje příznaky nebo výstupy z libovolného rozsahu na nový rozsah. Obvykle jsou příznaky škálovány na rozsah 0–1 nebo (-1)–1. Při použití metody min-max zůstává efektivně každý příznak stejný, v relaci s ostatními, přitom patří do nového rozsahu. Tato metoda zachovává všechny relační vlastnosti dat. [1]

Z-skóre normalizace nastaví průměrnou hodnotu každého prvku v souboru dat na nulu a směrodatnou odchylku na jedničku. Součástí postupu je nejprve normalizovat vektory příznaků v souboru dat. Pro každý příznak v trénovacích datech se vypočítá průměr a směrodatná odchylka, které se použijí jako váha při konečném návrhu systému. [1]

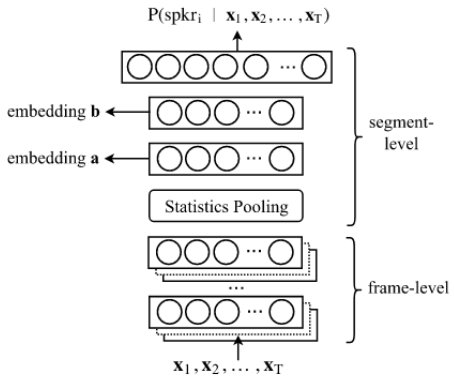
3.4 Vektory příznaků, embeddingy

Uchovávání dat v jejich přirozené formě je náročné na uložiště a výpočetně náročné na zpracování trénovanou sítí. Data se musí převést do vektorového prostoru, aby je počítač mohl efektivně zpracovávat. **Embeddingy** jsou nízkodimenzionální spojitou reprezentací vysokodimenzionálních proměnných. Proces trénování se pomocí embeddingů stává efektivnější. Většinou jsou extrahovány pomocí neuronových sítí specializovaných na extrakci embeddigů. Embeddingy se pomocí číselné reprezentace jednotlivých příznaků (dimenzí) rozprostřou po příznakovém prostoru. Embeddingy s podobnými příznaky se v prostoru nachází k sobě blíže než embeddingy s odlišnými příznaky. [28]

Jedním z příkladů reprezentace embeddingů, které v této práci pomocí převzatých a předtrénovaných sítí extrahuji je obrázek 3.1. Na obrázku je vidět, jak se data z vysokodimenzionálního obrázku po vyextrahování příznaků pomocí **FaceNet** [41] přemění na fixní nízkodimenzionální embedding. Tento embedding má formu vektoru čísel, které reprezentují hodnoty příznaků. Obdobný embedding s rozdílnou dimenzionalitou dostanu extrakcí hlasu z audio nahrávky pomocí **SpeechBrain** [35]. Oba tyto modely jsou rozepsané do podrobna v příští kapitole 4. Mým úkolem v této práci je namapovat hlasový na obrazový embedding.



Obrázek 3.1: Znázornění extrakce embeddingu pomocí FaceNet. Obrázek převzatý z internetu[2]



Obrázek 3.2: Architektura neuronové sítě, sloužící pro verifikaci mluvčího, lze použít pro extrakci x-vektorů, převzato z [42]

V oblasti verifikace a rozpoznávání řečníků se v posledním desetiletí rozmohly koncepty i-vektorů a x-vektorů. I-vektory a x-vektory jsou vektorové reprezentace fixní délky audio nahrávek libovolné délky. I-vektory jsou založeny na generativním modelu, zatímco x-vektory jsou založeny na diskriminativním modelu. Extraktory i-vektorů jsou trénovány za účelem zachycení nejlepší reprezentace celého akustického prostoru. Narozdíl od toho jsou x-vektor extraktory trénovány za účelem diskriminace řečníka, tudíž se snaží eliminovat jakoukoliv nechtěnou variabilitu. Dále se jednotlivé vektory liší podle způsobu jejich trénování, i-vektor extraktory se učí bez učitele a x-vektor extraktory s učitelem. [28]

Diagram x-vektor extraktoru je vidět na obrázku 3.2. Tato síť se skládá z plně propojených vrstev, které tvoří **úroveň snímků** (frame-level). Další vrstva, patřící do **segmentové úrovně** (segment-layer), jménem **statistics pooling layer** reprezentuje statistiky vstupních akustických příznaků. Další dvě schované vrstvy o různých dimenzionalitách se mohou použít pro vytvoření x-vektorů. Výstupní vrstvu tvoří softmax funkce.

[2] https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/

Kapitola 4

Zpracování řeči, obličeje a generování obličeje

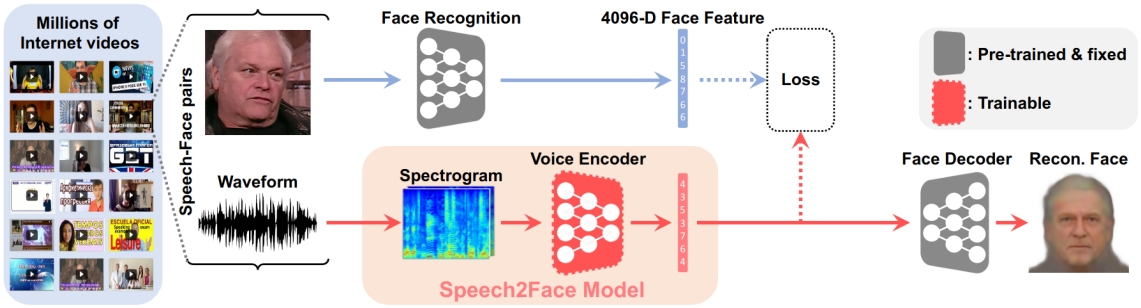
Architektura systému této práce je založena na třech převzatých a předem předtrénovaných modelech. Tyto modely byly vybrány, protože v současné době jsou považovány jako state-of-the-art (nejlepší technologie pro daný účel v dané době). Každý model se věnuje specifickému účelu potřebnému pro funkcionalitu výsledného systému. Je potřeba zajistit extrakce embeddingů z datových sad a generování obličeje z obrázkového embeddingu, moje práce bude tyto modely propojit. Nyní představím podobné práce, které se věnují stejné problematice.

4.1 Příklad jiných přístupů řešících odhad obličeje z řeči

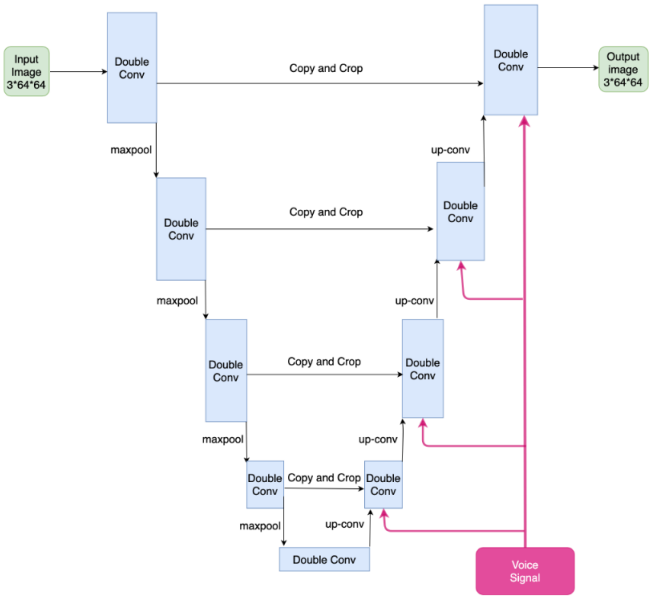
Existuje řada prací, které se věnují problému generování obličeje na základu mluvené řeči. Uvádím zde dva příklady využívající autoenkodérových architektur.

4.1.1 Speech2Face

Práce **Speech2Face** se skládá ze **Speech2Face modelu** a **trénovací pipeline**. Vstupem Speech2Face modelu je komplexní spektrogram vypočítaný z krátké nahrávky mluvící osoby. Výstupem je 4096 dimenzionální příznak obličeje, který je následně dekódován do obrazu obličeje pomocí předem natrénované sítě dekodéru obličeje. Modul, který se trénuje, zvýrazněný oranžovým rámečkem. Síť se trénovala, aby regresovala na skutečný příznak ob-



Obrázek 4.1: Speech2Face model a trénovací pipeline, převzato z [30]



Obrázek 4.2: Architektura řízeného autoenkodéru. Levá část značí enkodér, který pomocí konvolučních sítí získá nízkodimenzionální reprezentaci vstupu, druhá část tvoří dekodér, který pomocí transpozičních konvolučních vrstev zrekonstruuje vstupní obrázek. Tato architektura také obsahuje přeskoková spojení mezi jednotlivými vrstvami enkodéru k odpovídajícím vrstvám dekodéru. Převzato z [24].

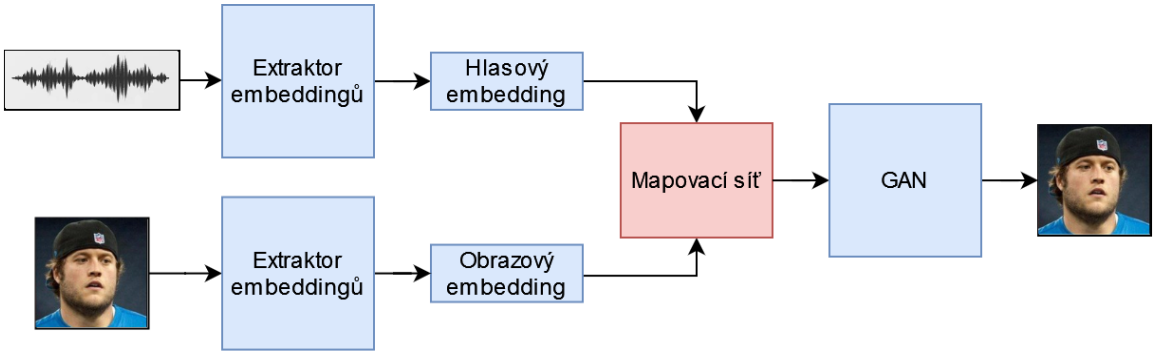
ličeje. Tohoto se dosáhlo vložením obrazu osoby do sítě pro rozpoznávání obličeje a extrakcí rysu z její předposlední vrstvy. Obličejové příznaky se porovnávaly objektivní funkcí. [30]

4.1.2 Controlled AutoEncoders to Generate Faces from Voices

Model **Controlled AutoEncoders to Generate Faces from Voices** [24] vychází z hypotézy, že může být snazší generovat realistické napodobeniny obličeje mluvčího, pokud se začne s výchozí šablonou skutečného obličeje a povedou se v ní minimální změny, za účelem se zachování co nejvíce rysů obličeje, aby odpovídala hlasu. Model začíná s počáteční šablonou obličeje a pomocí vyextrahovaných informací z hlasu modifikuje tuto šablonu. Model používaný v tomto článku se nazývá **řízený autoenkodér** (controlled autoencoder – **CAE**). Skládá se ze dvou částí. Architektura je znázorněna na obrázku 4.2. [24]

Hlavní část funguje jako lešení, přebírá návrh obličeje a na výstup vrací překreslenou tvář. V případě nepřítomnosti jiných signálů musí síť přegenerovat vstupní šablonu obličeje.

Druhá část, která přijímá hlasový vstup, upravuje parametry dekodéru na základě informací získaných z hlasu, aby vygenerovaná tvář odpovídala hlasu. Model při trénování obsahuje také **diskriminační složku**, která se používá k optimalizaci parametrů sítě. Celý model se učí pomocí sady ztrát, které současně zachycují míru shody s cílovou (mluvčího) tváří a odchylku od navrhované tváře. [24]



Obrázek 4.3: Schéma složené z jednotlivých stavebních komponent, červeně zvýrazněná část je mojí vlastní implementací potřebného mezidílu, aby tento systém mohl fungovat.

4.2 Základní stavební komponenty této práce

Tato práce se vydává jiným, trochu experimentálním směrem, pokouší se využít toho, jakého pokroku dosáhly generativní adverzní sítě v posledních letech. Na obrázku 4.3 jsou vidět jednotlivé komponenty této práce.

Obličejový embedding se vyextrahuje z obrazu řečníka pomocí **extraktoru obličejových embeddingů**, **hlasový embedding** se vyextrahuje z nahrávky pomocí **extraktoru hlasových embeddingů**. Existují **GANy**, které generují z obrazového embeddingu zpátky skoro totožný obrázek obličeje. Na základě předpokladu, že hlas nese informace o vzhledu řečníka, tudíž existuje souvislost mezi hlasem a vzhledem, se dá také předpokládat, že bude existovat souvislost mezi tím, jaké informace přenáší hlasové a obrazové embeddingy stejné identity. Tím pádem by mělo být možné naučit neuronovou síť tyto embeddingy na sebe mapovat a namapované embeddingy by se měly dát používat místo obrazových embeddingů pro generování obličeje pomocí GAN.

V následujících sekcích popisuji jednotlivé převzaté extraktory embeddingů a převzatý GAN.

4.3 Extrakce embeddingů z audionahrávek

SpeechBrain [35] je model použitý pro extrakci embeddingů z audionahrávek řečníků. Mezi příklady možných využití SpeechBrain patří **převod řeči na text** (speech recognition), **separace řeči** (speech separation) a **verifikace řečníka** (speaker recognition). Mnou využívaná verze SpeechBrain využívá model **ECAPA–TDNN** [12]. ECAPA se skládá z kombinace konvolučních a reziduálních sítí. Systém využívá aktivační funkce softmax–loss. K extrakci ECAPA embeddingů dochází pomocí **pozorného statistického sdružování** [31]. Různé architektury speechbrain vrací různé druhy embeddingů, například x-vektory nebo ResNet-34 embeddingy. ECAPA-TDNN model vrací 192 dimenzionální embeddingy, které výkonnostně překonávají ostatní druhy embeddingů. Tento model byl natrénovaný na datasetech VoxCeleb a VoxCeleb2. Ověřování mluvčích určuje kosinovou vzdáleností mezi embeddingy mluvčích. Pro moje účely mi stačí část pro extrakci embeddingů. [35]

4.4 Extrakce embeddingů z obrazu

FaceNet [41] je model, který jsem použil pro extrakci embeddingů z obrazů řečníků. FaceNet se skládá z hluboké konvoluční sítě následovanou L2 normalizací, jejíž výsledkem je obličejový embedding. Trénovací proces v této architektuře ještě utilizuje triplet loss. Triplet loss je objektivní funkce, která minimalizuje vzdálenost mezi kotvou (anchor) a pozitivním bodem. Dále maximalizuje vzdálenost mezi kotvou a negativním bodem. Kotva je v tomto případě nějaký referenční embedding, pozitivní bod je embedding, který patří stejné identitě a negativní bod je embedding patřící jiné identitě. V této práci využívám FaceNet natrénovaný na datové sadě VGGFace2. Extrahované embeddingy modelu FaceNet nabývají 512 dimenzí. FaceNet využívá **MTCNN** a **Inception-ResNetV1** jako součást svojí architektury. [41]

4.4.1 MTCNN

MTCNN je složena ze tří stádií. V prvním stádiu se vytvoří různé velikosti obrazu, aby se získaly různé velikosti obrázků s různými velikostmi obličeje. Vytvoří se ohraničená místa (bounding boxes), kde s určitou jistotou leží obličeje. Dále se musí projít a vymazat ohraničená místa, která mají nízkou jistotu. Tyto obrázky projdou ještě dvěma dalšími stádii, kde se postupnými iteracemi MTCNN zbavuje špatně ohraničených míst a jako výsledek vrací slovník s ohraničeními místy pro daný obraz. [46]

4.4.2 Inception-ResNetV1

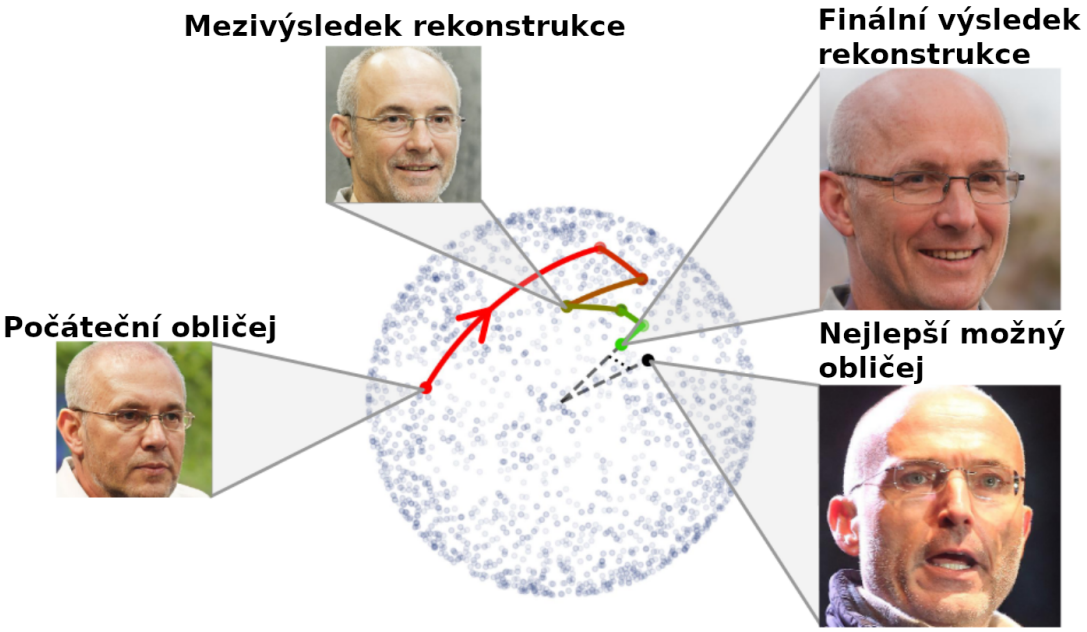
Inception modely využívají nahromadění více různých konvolučních filtrů do jedné vrstvy, z jejich výstupu se provede konkatenace a výsledek je předán další Inception vrstvě. Pomocí Inception architektury dojde k razantnímu snížení výpočetní náročnosti. **Inception-ResNetV1** ještě navíc implementuje nad Inception moduly ResNet spojení, které výstupy konvoluce předávají na vstup. [41]

4.5 Generování obličeje z embeddingů obrazu

Pro generování obličeje jsem převzal model z práce **Realistic Face Reconstruction from Deep Embeddings** [43], který extrahuje FaceNet embedding z obrázku a aplikuje nad ním důmyslný algoritmus, ten pomocí **StyleGAN2** [21] rekonstruuje původní obličej.

Rodina StyleGAN se liší od jiných GANů svojí architekturou. Ke konstantní matici se postupně přidávají příznaky pomocí stylových vektorů a přidaného šumu. Tyto stylové vektory jsou výstupem mapovací sítě, která dostává na vstup vektor získaný generátorem. Velikost generovaného obrazu se postupně navyšuje, z počátečních 4x4 na výstupních 1024x1024. [21]

Základní algoritmus z Realistic Face Reconstruction from Deep Embeddings pomocí hladového algoritmu prochází příznakovým prostorem StyleGAN2 k vygenerování obrazu. Z jednotlivých obrazů se získá FaceNet embedding a pomocí metriky L2 vzdálenosti se pokouší co nejpřesněji shodovat s FaceNet embeddingem na vstupu. Příznakové prostory StyleGAN2 a FaceNet jsou odlišné a proto se musí zpětně získávat FaceNet embedding. Další rozšíření algoritmu spočívají v možnosti nahrazení hladového algoritmu za simulované žíhání a předgenerování různých počátečních vektorů, ze kterých se vybere nejlepší. Využíval jsem simulované žíhání, protože zabránilo případnému uvíznutí na lokálním minimu. [43]



Obrázek 4.4: Algoritmus rekonstrukce obličeje, na kterém je vidět postupné procházení příznakovým prostorem StyleGAN2, převzato z [43], upraveno

Na začátku algoritmus začíná s nulovým vektorem, pokud není zvolená možnost předgenerování počátečních vektorů, a opakovaně generuje nový odhad přidáním malého náhodného šumu. Z těchto nových odhadů se pokaždé vygeneruje obraz, ze kterého se získá FaceNet embedding a znova se porovná. Pokud je nový odhad lepší než předchozí, našel se nový nejlepší vektor. Postupným snižováním směrodatné odchylky šumu se konverguje k řešení. Jak algoritmus postupně prochází příznakovým prostorem StyleGAN2 je vidět na obrázku 4.4. [43]

Kapitola 5

Návrh, implementace a trénování

V předchozí kapitole jsem popsal dopodrobna jednotlivé části systému, v této kapitole popíšu návrh architektury trénovacího modelu, popis jeho implementace a trénovacího procesu. Práce byla implementována v jazyce Python pomocí různých knihoven, mezi které patří PyTorch[1], Pandas[2], NumPy[3] a Pickle[4]. Všechny skripty jsem psal na Google Colab, protože jsem potřeboval dostatečně výkonnou grafickou kartu pro algoritmus nad StyleGAN2.

5.1 Připravení dat

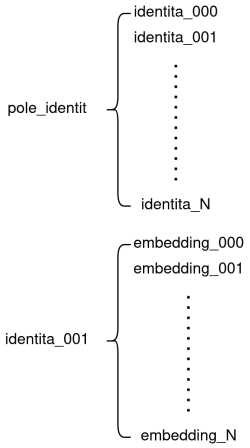
Jednotlivá data se musela předzpracovat. Z připravených dat se extrahovaly embeddingy, které se uchovávaly ve speciálních datových strukturách. Nakonec se ukládaly do pickle souborů, ze kterých byly jednotlivé datové sady načítány pro trénování mapovací sítě. Informace o jednotlivých identitách jsem uchovával v metasouborech zvlášť.

5.1.1 Příprava audionahrávek

V rámci přípravy audio dat byla potřeba jednotlivým audionahrávkám sjednotit délku, podle udávané průměrné délky nahrávek VoxCeleb a VoxCeleb2 jsem se rozhodl pro délku šesti vteřin. Nahrávky mají vzorkovací frekvenci 16kHz, tudíž nebyla potřeba vzorkovací frekvenci měnit, tato vzorkovací frekvence je nutná pro správnou funkci extrakce embeddingů pomocí SpeechBrain. Skript na úpravu délky audia a vzorkovací frekvence jsem převzal od kolegy Petra Zubalíka, který vypracovává jinou variaci stejného tématu jako diplomovou práci.

Po extrakci každého audio embeddingu pomocí SpeechBrain byla potřeba převést audio embedding z procesoru na grafickou kartu pomocí `embedding.to("cuda")`. Nakonec byla potřeba embeddingům zapnout možnost výpočtu derivace pro optimalizátor pomocí `embedding.requires_grad_()`.

[1] https://pytorch.org/
[2] https://pandas.pydata.org/
[3] https://numpy.org/
[4] https://docs.python.org/3/library/pickle.html



Obrázek 5.1: Struktura dat pro trénování mapovací sítě

5.1.2 Příprava obrazů

VGGFace obrázky potřebovaly vyříznout obličej na konkrétní pozici popsané v metadatech. VGGFace i VGGFace2 data se dále převedla na barevné obrázky, pokud doposud nebyly. Obrázky byly přeškálovány na velikost 160x160, na které byl FaceNet natrénován. Embedding se extrahoval pomocí Inception-ResNetV1 sítě, poté se též musel převést na grafickou kartu pomocí `embedding.to("cuda")`.

5.1.3 Následné skladování embeddingů

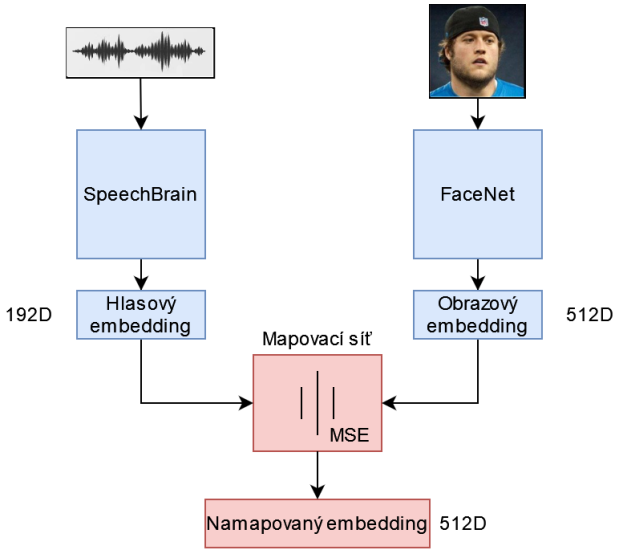
Embeddingy byly předem extrahovány pro zefektivnění trénování na systému, jehož architektura je popsána v následující sekci 5.2. Jednotlivé embeddingy jsem následně nahrál do polí zvlášť pro obraz a zvlášť pro nahrávky. Tímto způsobem mi vznikla dvě pole, která mají stejnou strukturu až na samotný obsah. Takto jsem získal pole jednotlivých identit, která jsou poli embeddingů. Struktura je vizualizována na obrázku 5.1. Tato pole jsem si pro budoucí využití ukládal do souborů **pickle**.

5.1.4 Dataset a DataLoader

Datovou sadu jsem načetl do Dataset třídy. **Dataset** a **DataLoader** jsou zprostředkovány knihovnou PyTorch. Výhodou načtení Dataset do DataLoader je možné nastavit velikost dávek a zamíchat data mezi každými epochami. Mapovací síť viděla během jedné trénovací epochy kartézský součin hlasů a obličejů všech jednotlivých identit. Optimální velikost dávek se lišila podle použitých datových sad. Pro VoxCeleb1 a VGGFace1 byla nejoptimálnější velikost dávek 128, pro VoxCeleb2 a VGGFace2 byla nejoptimálnější velikost dávek 256.

5.2 Architektura trénovacího systému a její implementace

Trénovací systém se skládá celkově ze tří částí. Ze dvou převzatých a předtrénovaných modelů pro účely extrakce embeddingů a z mé vlastní neuronové sítě, která plní funkci mapování 192 dimenzionálního hlasového embeddingu na 512 dimenzionální obrazový embedding. Podrobnosti implementace mapovací sítě jsou v sekci 5.4.



Obrázek 5.2: Trénovací architektura systému je založena na třech sítích, sítě znázorněné modrou barvou jsou již předtrénované. Moje červená mapovací síť je jedinou trénovanou komponentou tohoto systému, pomocí lineární regrese a objektivní funkce střední kvadratické chyby (MSE) se embeddingy na sebe mapují.

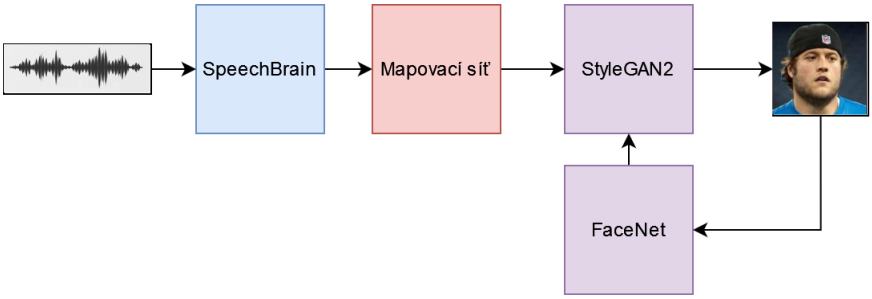
Na obrázku 5.2 je vidět schéma trénovacího modelu. V předchozí sekci 5.1 jsem popsal, jak se zapojí oba extraktory embeddingů do procesu trénování. Po extrakci embeddingů mi zbývá natrénovat mapovací síť.

Tato architektura byla zvolena jako prvotní plán, který by se dále mohl rozvinout. Nejdříve jsem potřeboval dosáhnout dostatečných výsledků na této jednoduché architektuře, abych si ověřil, jestli má cenu na tomto systému dále pracovat. Po experimentaci jsem došel k modelu, který vrací slibné výsledky popsané v kapitole 6.

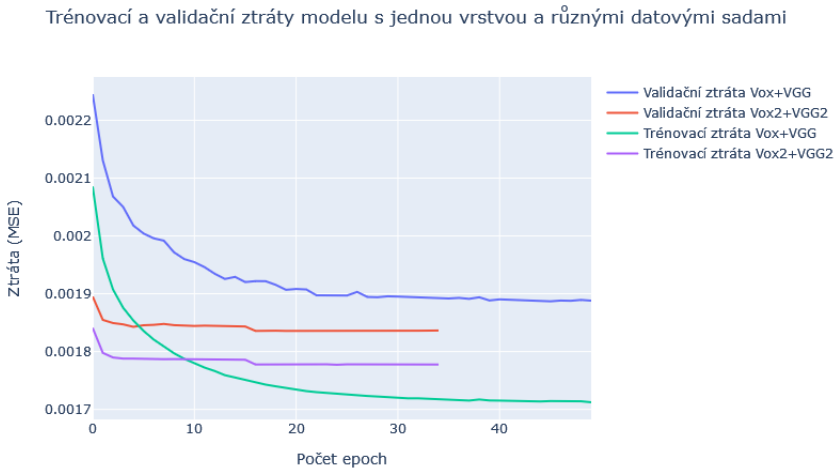
Jelikož mi trvalo dlouho, než jsem se dostal ke slibným výsledkům, nezbyl mi dále čas na implementaci a na trénování důmyslnější trénovací architektury, tak jsem se soustředil na vylepšování výsledků tohoto modelu. Návrhy na vylepšení trénovací architektury jsou popsány v sekci 6.3.

5.3 Architektura modelu pro generování obrazu z řečové nahrávky

Na obrázku 5.3 je vidět architektonické schéma systému, který generuje obraz z řečové nahrávky. SpeechBrain extrahuje hlasový embedding z hlasové nahrávky, který mapovací síť namapuje. Trénink mapovací sítě je popsaný v předchozí sekci, čili všechny části systému jsou natrénované a schopné fungovat. Algoritmus nad StyleGAN2 přijme výstup mapovací sítě a iteračně generuje obrázky. Všechny části tohoto systému, až na mapovací síť, jsou předtrénované, čili zbývá natrénovat mapovací síť pomocí architektury trénovacího systému.



Obrázek 5.3: Architektura modelu pro generování obrazu z řečové nahrávky.

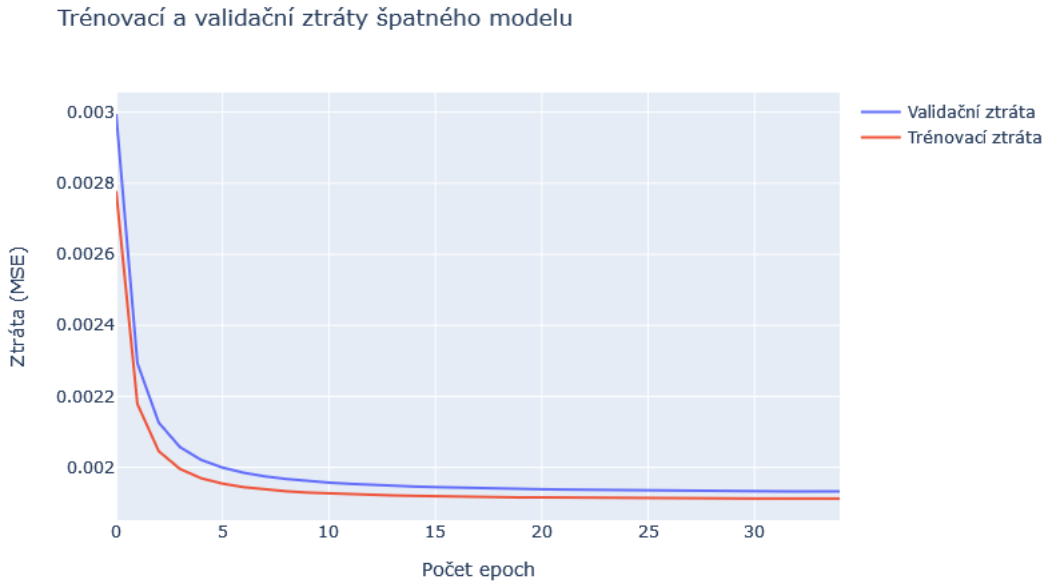


Obrázek 5.4: Rozdíl ve ztrátách, u stejného jednovrstvého modelu, mezi datovými sadami. Větší datové sady VoxCeleb2 a VGGFace2 překonaly VoxCeleb a VGGFace.

5.4 Mapovací síť a její trénovaní

Problém mapování embeddingů byl řešený pomocí linerání regrese. Během trénování jsem postupoval od nejjednoduššího modelu k modelu s nejlepšími výsledky. Používal jsem dvojí metriky měření výkonu modelu. Vývoj trénovacího procesu hodnotila objektivní funkce střední kvadratické chyby (MSE). Pokud se po trénovací epoše zlepšila ztráta, uložil jsem si model. Nad nejlepším modelem z trénovacího procesu jsem zkušebně mapoval pět nahrávek. Podle vzdálenosti mezi FaceNet embeddingy a podobnosti osoby na obrázku s identitou jsem usuzoval kvalitu modelu pro generování obrázků.

Nejlepších výsledků dosahovaly modely mapovací sítě s datovými sadami VGGFace2 a Voxceleb2, s datovými sadami VGGFace a Voxceleb byly nepoužitelný. Rozdíl ztrát podle datových sad je vidět na obrázku 5.4. Výsledné vygenerované obličeje se neshodovaly pohlavím, ani jinými rysy. Myslím si, že hlavním faktorem je rozdílný počet identit a rozdílný počet nahrávek a obrazů. U VoxCeleb a VGGFace dat nebyla mapovací síť schopná identifikovat důležité příznaky.



Obrázek 5.5: Ztráty modelu se dvěma vrstvami a s přidanou nelinearitou ve formě ReLU funkce, použité datové sady VGGFace2 a VoxCeleb2.

5.4.1 Problém přidávání hloubky modelu

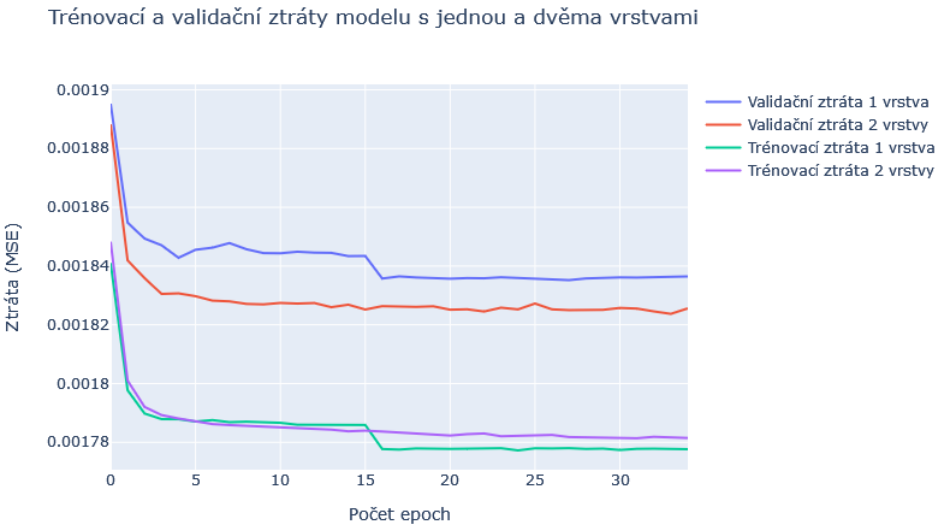
Pokud jsem do sítě zavedl více jak dvě vrstvy nebo jakoukoliv formu nelinearity, začala síť podávat daleko horší výkon. Pomocí objektivní funkce střední kvadratické chyby se rozdíl nezdá být tak markantní, ale při generování obrázků je ihned patrné, že síť nezvládá zachytit potřebné příznaky. Vygenerovaný obraz identity na první pohled naprosto neodpovídá identitě. Úprava hyperparametrů nijak razantně zlepšit výsledky nepomohla, způsobila jedině pomalejší/rychlejší konvergenci či miniaturní změny ve ztrátách. Bídnou úroveň generovaných obrazů to ale neovlivnilo.

Na obrázku 5.5 jsou vidět vysoké trénovací a validační ztráty modelu se dvěma vrstvami s ReLU funkcí. Tento model budu dále nazývat jako „model C“. Nemělo smysl se dále pokoušet experimentovat se složitějšími architekturami, protože síť fungovala nejlépe na jedné nebo na dvou plně propojených vrstvách.

5.4.2 Nejlepší modely

Na nejlepší architekturu pro mapování embeddingů pro tento systém jsem narazil už v podstatě na začátku. Modely podávaly nejlepší výkon s jednou nebo dvěma vrstvami. Postupnou experimentací s parametry a hyperparametry jsem došel ke dvěma nejlepším modelům, které tyto parametry sdílejí i přes rozdílný počet vrstev.

Na obrázku 5.6 jsou vidět ztráty obou modelů. Dvouvrstvý model měl lepší ztráty než model s jedinou vrstvou. I přes tento fakt jsou v kapitole 6 vidět obrázky, ze kterých je patrné, že se model s jedinou vrstvou, podle výsledků generování obrazů, ukázal být všeobecně lepší v zachycení jednotlivých důležitých rysů identit než dvouvrstvý model. Model s jednou vrstvou budu dále nazývat „model A“ a model s vrstvami dvěma budu nazývat „model B“.



Obrázek 5.6: Porovnání ztrát dvou nejlepších modelů, použité datové sady VGGFace2 a VoxCeleb2.

Model	Model A	Model B	Model C
Počet vrstev	1	2	2
Typy vrstev	Plněpropojené		
Nelinearita	-	-	ReLU
Validační ztráta	1,835e-3	1,825e-3	1,923e-3
Průměrná hodnota L2 vzdálenosti	0,7912	0,8174	0,9563

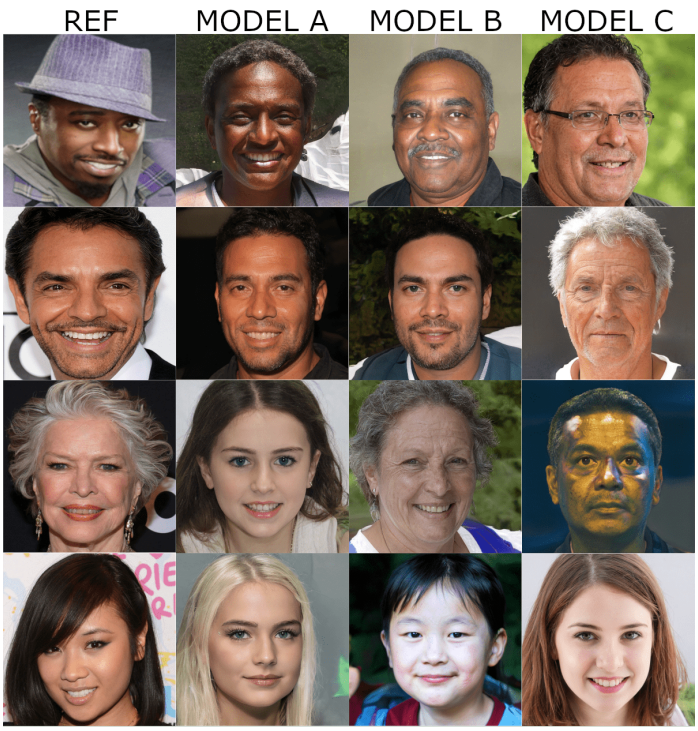
Tabulka 5.1: Porovnání jednotlivých modelů. Průměrná hodnota L2 vzdálenosti je vypočítaná z 10 vygenerovaných obrazů.

5.5 Porovnání jednotlivých modelů

Na obrázku 5.7 jsou vidět výsledky generování identit, které jsem vzal z testovací datové sady VoxCeleb, žádná komponenta ze systému této práce tato data nikdy neviděla. Model C nezvládá mapovat důležité příznaky a tak ve většině případů generuje nerelevantní obrazy identit. Model A a model B podávají slibné výsledky. Model A většinou nezvládá zaznamenat europoidní ženy nad 45 let, s čím model B problém nemá, avšak zbytek výsledků modelu A přesněji zachycují rysy jednotlivých identit.

V tabulce 5.1 je vidět srovnání všech modelů. Přes lepší trénovací a validační ztráty modelu B dosahuje model A lepších výsledků, co se týče přesnosti zachycení identity. Model A dosahuje nejnižších hodnot pro průměrnou hodnotu L2 vzdálenosti na deseti identitách. Spolu se samotnou objektivní kvalitou vygenerovaného obrazu je toto nejdůležitější metrika, která nepřímým způsobem určuje, jak moc jsou embeddingy podobné FaceNet embeddingům. U Modelu A se L2 vzdálenost pohybuje lehce pod 0,8.

Pro samotný algoritmus nad StyleGAN2 s extrahovaným embeddingem pomocí FaceNet platí, že dosáhne-li L2 vzdálenosti mezi embeddingy nižší než 0.6, klasifikace pro FaceNet akceptuje identitu z vygenerovaného obrazu jako stejnou té, která byla na vstupním obrazu.



Obrázek 5.7: Příklad vygenerovaných obličejů jednotlivých modelů nad nahrávkami identit z testovací sady VoxCeleb, které tento systém neviděl. Jako první je referenční fotka identity, další sloupce patří generovaným obrazům jednotlivých modelů.

Parametr	**Hodnota parametru**
Objektivní funkce	MSE
Druh optimalizace	SGD
Druh plánovače	ReduceLROnPlateau
Koeficient učení	0.01
Míra L2 regularizace	1e-5
Velikost dávek	256

Tabulka 5.2: Parametry a hyperparametry modelů A a B

Kapitola 6

Výsledky generování obrazů

Pokusil jsem se o zkušební generování 30 obličejů na konečném modelu A. Během tohoto generování systém v 27 z 30 příkladů trefil správné pohlaví řečníka, průměrná hodnota L2 vzdálenosti se rovnala 0,8105. V mnoha případech systém vygeneroval vzhled řečníka výtečně, vygenerovaná identita disponovala stejnou rasou, podobným věkem a dalšími přesnými rysy, včetně tvaru obličeje. Algoritmus generování obličeje s využitím simulovaného žíhání trvá zhruba 15-22 minut a zabírá 13.5GB GPU RAM.

6.1 Problémy generování obrazů

Problémy, kdy nešlo správně odhadnout pohlaví, se vyskytují u europoidních žen, které věkově spadají do kategorie nad 45 let. Pokud systém správně vygeneroval pohlaví, tak často obrázky trpěly různými artefakty, jak je vidět u třetího obličeje na obrázku 6.1. Nejspíš je tento jev způsobený nedostatečnou reprezentací této skupiny v trénovacích datech a neschopností modelu pochytit potřebné příznaky.

Často se vyskytují různé artefakty na obličejích, které při bližší inspekci prozrazují, že obrázky jsou uměle vytvořené. Mezi takové artefakty patří zvláštně zakalené oči a rozmazané části obličeje. Mezi další artefakty patří podivné módní příslušenství, například náušnice, které mají divný tvar a divné barvy nebo brýle s chybějícími obroučky.

Obrázek 6.1: Artefakty

Na stejném obrázku jsou vidět ještě další zvláštní artefakty. Na pozadí se někdy vytváří různé další zdeformované obličeje. Netuším, čím by to mohlo být, tento problém je unikátní pro obličeje generované tímto systémem.

6.2 Výsledky nejlepšího modelu

Výsledný mapovací model, natrénovaný na datových sadách VGGFace2 a VoxCeleb2, vygeneroval tyto obličeje 6.2 z nahrávek z VoxCeleb trénovací sady. Mapovací síti byly VoxCeleb identity schované. SpeechBrain byl předtrénovaný na kombinaci VoxCeleb a VoxCeleb2 dat. Vygenerované obrázky dosahují poměrně vysoké kvality díky použité metodě generování pomocí StyleGAN2. Hladový algoritmus pro průchod příznakovým prostorem StyleGAN2 se často zastavoval na lokálním minimu a vygeneroval obrázky identit, které neodpovídaly identitám. Proto jsem používal simulované žíhání, které v těchto situacích dovolilo vyhnout se uvíznutí v lokálním minimu. Dále je zde pozorovatelný vliv simulovaného žíhání na změnu pohlaví a rasy v průběhu generování. V některých případech simulované žíhání zhoršilo výsledek, ale na základě zhodnocení všech výsledků, jakožto celku, je simulované žíhání nezbytné pro generování obrazů tímto systémem.

Vypozoroval jsem silnou koorelaci mezi výsledky a sílou přízvuku řečníka. Zdá se, že podle přízvuku algoritmus vyhodnotí etnickou skupinu, od které převezme určité charakteristické rysy. Tyto rysy dále již v menším měřítku upravuje. Pokud neuhodne etnickou skupinu, vychýlí se od výsledku. Pokud je přízvuk hlasu neutrální, preferuje si vybrat europoidní vzhled, který je nejhojněji zastoupený v datových sadách VoxCeleb2 a VGGFace2.

6.3 Možná vylepšení

V této práci jsem prováděl jednoduché mapování dvou embeddingů na sebe. I přes tuto jednoduchou metodu dosáhly generované obrázky obličejů identit z audio nahrávek slibných výsledků, co se týče kvality a podobnosti obličejových rysů. Jako další možná vylepšení a rozšíření systému, v rámci navýšení kvality a konzistence generovaných obrázků, navrhuji dva způsoby. Oba způsoby jsou časově a výpočetně náročné, druhý řádově více než ten první. Druhý způsob trénovaní by odhadem trval několik dní.

6.3.1 Přetrénování SpeechBrain na mapování FaceNet výsledků

Prvním způsobem by bylo rozdělit SpeechBrain na jednotlivé části, z vrstev předcházejících poslední aktivační funkci a objektivní funkci vyvést embedding. Tento embedding by se porovnával s FaceNet embeddingy, dále by se využily funkce soft-max a objektivní funkce střední kvadratické chyby. Speechbrain by se přeučil na generování obrazových embeddingů z hlasu a nebyl by nutný mezikrok mapovací sítí. Trénování modelu by bylo ale daleko časově náročnější, hlasové embeddingy by se nemohly předvytvořit. Extrahování hlasových embeddingů je pro velké datové sady časově náročné.

6.3.2 Využití metriky StyleGAN2

Druhý způsobem by bylo využít L2 vzdálenosti embeddingů získané algoritmem generování obličeje. Tuto vzdálenost jsem používal jenom jako orientační měřítko kvality modelu, ale tímto by se tato metrika dala využít během samotného trénování. Tento způsob by byl velice časově náročný, čistě kvůli délce generování obrazů.

Obrázek 6.2: Výsledky generování obličejů z trénovací sady VoxCeleb nahrávek, na kterých byl SpeechBrain natrénovaný, mapovací síť tyto identity nikdy neviděla. První obrázek je referenční fotka identity, druhý obrázek je první vygenerovaný obličej z náhodného šumu. Na dalších obrázcích je vidět, jak algoritmus nad StyleGAN2 prochází StyleGAN2 příznakový prostor, dokud nezkonverguje u řešení. SG2 a SG3 byly vybrány z mnoha mezivýsledků rekonstrukce obličeje.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout a natrénovat systém, který bude generovat obličej na základě vstupní řeči a následně vyhodnotit výsledný natrénovaný model.

Podařilo se mi implementovat systém, který poměrně úspěšně generuje obličej ze vstupní nahrávky řečníka. Použil jsem tři převzaté a předtrénované state-of-the-art neuronové sítě, ke kterým jsem vytvořil vlastní neuronovou síť, která plnila funkci mapování výstupu dvou sítí a umožňovala svým výstupem generování obličeje z řeči. Tuto síť jsem nejdříve navrhl a postupným experimentováním jsem našel nejlepší model, který jsem natrénoval. Nejlepší model jsem porovnal s dalšími natrénovanými modely, získanými během experimentování.

Na základě výsledků se dá potvrdit, že existuje souvislost mezi obličejem a hlasem, když lze pomocí jedné vrstvy neuronové sítě namapovat hlasový embedding na embedding obličejový, ze kterého se dají vygenerovat reálně vypadající obličeje. Výsledky této práce se dají považovat za slibné díky úspěšné implementaci unikátního způsobu řešení zadání. Kvalita výsledných obličejů je též slibná, vygenerované obličeje v mnoha případech silně připomínají řečníka. Bohužel existuje určitá variabilita ve výsledcích, i přes schopnost generovat kvalitní vygenerované obličeje není vzácné, že by výsledný obraz naprosto neodpovídal vzhledu identity řečníka.

Slibný výsledek této práce by se dále ještě mohl vylepšit jednou z navržených robustnějších trénovacích architektur. Trénování tímto způsobem by bylo velice časově a výpočetně náročné. Pokud bych měl na vylepšení této práce další půlrok, tak si myslím, že bych systém mohl natrénovat jednou z těchto architektur.

Věřím, že by při dostatečném množství potřebné práce nad architekturou tento systém mohl dosáhnout výsledků, které předčí leckteré jiné systémy, které se považují za state-of-the-art, co se týče generování obličeje ze vstupní řeči.

Literatura

[1] Aksu, G., Güzeller, C. O. a Eser, M. T. The Effect of the Normalization Method Used in Different Sample Sizes on the Success of Artificial Neural Network Model. *International Journal of Assessment Tools in Education*. červenec 2019, sv. 6, č. 2, s. 170–192. DOI: 10.21449/ijate.479404. Dostupné z: https://dergipark.org.tr/en/pub/ijate/issue/44255/479404.

[2] Al Behadili, H., Ku Mahamud, K. a Sagban, R. Rule pruning techniques in the ant-miner classification algorithm and its variants: A review. In:. Duben 2018. DOI: 10.1109/ISCAIE.2018.8405448.

[3] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al Dujaili, A., Duan, Y. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*. March 2021, sv. 8, č. 1, s. 53. DOI: 10.1186/s40537-021-00444-8. ISSN 2196-1115. Dostupné z: https://doi.org/10.1186/s40537-021-00444-8.

[4] Arora, J. S. *15 - Discrete Variable Optimum Design Concepts and Methods*. 2. vyd. San Diego: Academic Press, 2004. 513-530 s. ISBN 978-0-12-064155-0. Dostupné z: https://www.sciencedirect.com/science/article/pii/B978012064155050015X.

[5] Basodi, S., Ji, C., Zhang, H. a Pan, Y. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*. 2020, sv. 3, č. 3, s. 196–207. DOI: 10.26599/BDMA.2020.9020004.

[6] Bronwlee, J. *What is the Difference Between a Parameter and a Hyperparameter?* [online]. červen 2019. Vid. 15. 1. 2021. Dostupné z: https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/.

[7] Brunton, S. a Kutz, J. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2022. ISBN 9781009098489.

[8] Burkov, A. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019. ISBN 9781999579517.

[9] Cao, Q., Shen, L., Xie, W., Parkhi, O. M. a Zisserman, A. VGGFace2: A dataset for recognising faces across pose and age. *CoRR*. 2017, abs/1710.08092. Dostupné z: http://arxiv.org/abs/1710.08092.

[10] Chung, J. S., Nagrani, A. a Zisserman, A. VoxCeleb2: Deep Speaker Recognition. In: *INTERSPEECH*. 2018.

[11] Claude Sammut, G. I. W. *Encyclopedia of Machine Learning*. 1. vyd. Springer, 2011. ISBN 0387307680,9780387307688.

[12] Desplanques, B., Thienpondt, J. a Demuynck, K. ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification. In: Meng, H., Xu, B. a Zheng, T. F., ed. *Interspeech 2020*. ISCA, 2020, s. 3830–3834.

[13] Ding, B., Qian, H. a Zhou, J. Activation functions and their characteristics in deep neural networks. In: *2018 Chinese Control And Decision Conference (CCDC)*. 2018, s. 1836–1841. DOI: 10.1109/CCDC.2018.8407425.

[14] Géron, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019. ISBN 9781492032649.

[15] Gurney, K. *An introduction to neural networks*. 1. vyd. UCL Press, 1997. ISBN 9781857286731,1857286731,1857285034.

[16] Günther, F. a Fritsch, S. neuralnet: Training of Neural Networks. *The R Journal*. 2010, sv. 2, č. 1, s. 30–38. DOI: 10.32614/RJ-2010-006. Dostupné z: https://doi.org/10.32614/RJ-2010-006.

[17] Haykin, S. *Neural Networks and Learning Machines*. 3. vyd. Prentice Hall, 2009. Neural networks and learning machines. ISBN 9780131471399.

[18] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. *CoRR*. 2015, abs/1512.03385. Dostupné z: http://arxiv.org/abs/1512.03385.

[19] Janiesch, C., Zschech, P. a Heinrich, K. Machine learning and deep learning. *Electronic Markets*. Springer. 2021, sv. 31, č. 3, s. 685–695.

[20] Jin, X. a Han, J. K-Means Clustering. In: Sammut, C. a Webb, G. I., ed. *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, s. 563–564. DOI: 10.1007/978-0-387-30164-8_425. ISBN 978-0-387-30164-8. Dostupné z: https://doi.org/10.1007/978-0-387-30164-8_425.

[21] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. et al. Training Generative Adversarial Networks with Limited Data. *CoRR*. 2020, abs/2006.06676. Dostupné z: https://arxiv.org/abs/2006.06676.

[22] Kecman, V. Support Vector Machines – An Introduction. In:. Květen 2005, sv. 177, s. 605–605. DOI: 10.1007/10984697_1. ISBN 978-3-540-24388-5.

[23] Kyjonka, M. *Odhad obličeje z řečového signálu*. Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/24006/.

[24] Liang, H., Yu, L., Xu, G., Raj, B. a Singh, R. Controlled AutoEncoders to Generate Faces from Voices. *CoRR*. 2021, abs/2107.07988. Dostupné z: https://arxiv.org/abs/2107.07988.

[25] Mohri, M., Rostamizadeh, A. a Talwalkar, A. *Foundations of Machine Learning, second edition*. 2. vyd. MIT Press, 2018. Adaptive Computation and Machine Learning series. ISBN 9780262351362.

[26] Nagrani, A., Chung, J. S. a Zisserman, A. VoxCeleb: a large-scale speaker identification dataset. In: *INTERSPEECH*. 2017.

[27] Nayak, S., Misra, B. a Behera, D. H. Impact of Data Normalization on Stock Index Forecasting. *International Journal of Computer Information Systems and Industrial Management Applications*. Prosinec 2014, sv. 6, s. 357–369.

[28] Novotný, O. *Improving Robustness of Speaker Recognition using Discriminative Techniques*. Brno, CZ, 2021. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/phd-thesis/1033/.

[29] Oh, T., Dekel, T., Kim, C., Mosseri, I., Freeman, W. T. et al. Speech2Face: Learning the Face Behind a Voice. *CoRR*. 2019, abs/1905.09773. Dostupné z: http://arxiv.org/abs/1905.09773.

[30] Oh, T., Dekel, T., Kim, C., Mosseri, I., Freeman, W. T. et al. Speech2Face: Learning the Face Behind a Voice. *CoRR*. 2019, abs/1905.09773. Dostupné z: http://arxiv.org/abs/1905.09773.

[31] Okabe, K., Koshinaka, T. a Shinoda, K. Attentive Statistics Pooling for Deep Speaker Embedding. In: *Interspeech 2018*. ISCA, Sep 2018. DOI: 10.21437/interspeech.2018-993. Dostupné z: https://doi.org/10.21437%2Finterspeech.2018-993.

[32] Parkhi, O. M., Vedaldi, A. a Zisserman, A. Deep Face Recognition. In: Xianghua Xie, M. W. J. a Tam, G. K. L., ed. *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2015, s. 41.1–41.12. DOI: 10.5244/C.29.41. ISBN 1-901725-53-7. Dostupné z: https://dx.doi.org/10.5244/C.29.41.

[33] Paullada, A., Raji, I. D., Bender, E. M., Denton, E. a Hanna, A. Data and its (dis)contents: A survey of dataset development and use in machine learning research. *Patterns*. 2021, sv. 2, č. 11, s. 100336. DOI: https://doi.org/10.1016/j.patter.2021.100336. ISSN 2666-3899. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2666389921001847.

[34] Phaisangittisagul, E. An Analysis of the Regularization Between L2 and Dropout in Single Hidden Layer Neural Network. In: *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*. 2016, s. 174–179. DOI: 10.1109/ISMS.2016.14.

[35] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S. et al. *SpeechBrain: A General-Purpose Speech Toolkit*. 2021. ArXiv:2106.04624.

[36] Roh, Y., Heo, G. a Whang, S. E. A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering*. 2021, sv. 33, č. 4, s. 1328–1347. DOI: 10.1109/TKDE.2019.2946162.

[37] Rothlauf, F. *Optimization Methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. 45–102 s. ISBN 978-3-540-72962-4. Dostupné z: https://doi.org/10.1007/978-3-540-72962-4_3.

[38] Rothlauf, F. *Optimization Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. 7–44 s. ISBN 978-3-540-72962-4. Dostupné z: https://doi.org/10.1007/978-3-540-72962-4_2.

[39] Sarker, I. H. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*. Springer. 2021, sv. 2, č. 3, s. 1–21.

[40] Sarwinda, D., Paradisa, R. H., Bustamam, A. a Anggia, P. Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer. *Procedia Computer Science*. 2021, sv. 179, s. 423–431. DOI: https://doi.org/10.1016/j.procs.2021.01.025. ISSN 1877-0509. 5th International Conference on Computer Science and Computational Intelligence 2020. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877050921000284.

[41] Schroff, F., Kalenichenko, D. a Philbin, J. FaceNet: A unified embedding for face recognition and clustering. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun 2015. DOI: 10.1109/cvpr.2015.7298682. Dostupné z: https://doi.org/10.1109%2Fcvpr.2015.7298682.

[42] Snyder, D., Garcia Romero, D., Povey, D. a Khudanpur, S. Deep Neural Network Embeddings for Text-Independent Speaker Verification. In: *Proc. Interspeech 2017*. 2017, s. 999–1003. DOI: 10.21437/Interspeech.2017-620.

[43] Vendrow, E. a Vendrow, J. Realistic Face Reconstruction from Deep Embeddings. In: *NeurIPS 2021 Workshop Privacy in Machine Learning*. 2021. Dostupné z: https://openreview.net/forum?id=-WsBmzWwPee.

[44] Weyland, D. Simulated Annealing, Its Parameter Settings and the Longest Common Subsequence Problem. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2008, s. 803–810. GECCO '08. DOI: 10.1145/1389095.1389253. ISBN 9781605581309. Dostupné z: https://doi.org/10.1145/1389095.1389253.

[45] Yu, G., Sapiro, G. a Mallat, S. Solving Inverse Problems with Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity. *CoRR*. 2010, abs/1006.3056. Dostupné z: http://arxiv.org/abs/1006.3056.

[46] Zhang, K., Zhang, Z., Li, Z. a Qiao, Y. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR*. 2016, abs/1604.02878. Dostupné z: http://arxiv.org/abs/1604.02878.

Příloha A

Obsah přiložené SD karty

Přiložená SD karta obsahuje:

- MappingNetwork.py – mapovací síť
- train.py – skript pro trénování mapovací sítě
- generate.py – skript pro generování obrázku z řečové nahrávky
- training_data – složka obsahující trénovací data
- models – natrénované modely mapovací sítě
- stylegan2-ada-pytorch – složka obsahující předtrénovaný model StyleGAN2
- pretrained_models – složka obsahující předtrénovaný model SpeechBrain
- requirements.txt – potřebné knihovny
- README.md – manuál ke struktuře a jednotlivým součástem