FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Syntéza řeči s animací obličeje

Autor:
Karel Müller

Vedoucí práce:
Ing. Jan Zelinka, Ph.D.

15. srpna 2022

Prohlášení

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

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

V Plzni dne 15. srpna 2022 _____

ZÁPADOČESKÁ UNIVERZITA

Fakulta aplikovaných věd

Katedra kybernetiky

Syntéza řeči s animací obličeje

Karel Müller

Abstrakt

V poslední době se rozmohl trend řešit vše vzdáleně, audiovizuální syntéza může být využita pro tvorbu avatarů nebo vytvoření virtuálního studia. S rozvojem hlubokého strojového učení roste i jeho využití v metodách syntézy tváře. Z evolučních důvodů je člověk vysoce citlivý na nedokonalosti v pohybu obličeje, zejména rtů. Proto je na současných řešeních stále co zdokonalovat. V této práci rozeberu základní pojmy z oblasti neuronových sítí, podám soupis metod syntézy audiovizuální hlavy, navrhnu perspektivní metodu syntézy audiovizuální hlavy, připravím data pro experimenty, experimentálně ověřím základní funkčnost metody, zhodnotím výsledky a doporučím další směr budoucí práce.

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Janu Zelinkovi, Ph.D., za cenné rady a konzultace při opravě skriptů, dokumentace a pomoc při tvorbě datasetu. Dále děkuji Ing. Zdeňku Krňoulovi Ph.D., za pomoc a cenné rady při nahrávání videa pro experiment.

Obsah

1 Úvod

Audiovizuální syntéza má v dnešním světě mnoho zajímavých využití, např. virtuální studio, herní průmysl, konzervace podoby a hlasu člověka nebo třeba nahrazení tváře člověka, který nechce být zobrazen, avatarem. Navíc bychom s tímto systémem mohli i detekovat řeč z obrazu.

S nedávnými pokroky v hlubokém učení lidé v tomto dlouhodobém problému udělali velký pokrok. Nicméně realistická animace mluvící hlavy zůstává stále výzvou. Tato úloha je složitá, k čemuž přispívá několik faktorů: (i) generování synchronizovaných a perzonalizovaných pohybů rtů. (i) kvůli obtížnosti mapování ze zvuku na pohyb obličeje. Dokonalé ztvárnění přirozeného pohybu rtů a obličeje je velmi důležité, neboť lidé jsou extrémně citliví na artefakty v obličeji, protože je obličej pro člověka a jeho vnímání evolučně důležitý. To je také jeden z důvodů, proč je dosažení realistické mluvící hlavy výzvou. V současnosti se většina audiovizuální syntézy neprovádí pomocí neuronových sítí, nýbrž s pomocí živého herce. S nynějším rozvojem neuronových sítí a velkým množstvím dat je řešení pomocí nich cestou do budoucna.

S rozvojem audiovizuální syntézy přichází i nebezpečí jejího zneužití. Lze takto s tváří známé osobnosti např. vyvolat poplašné zprávy nebo třeba poškodit dotyčnou osobu nějakou promluvou s její tváří. Tento tzv. deep fake je na světě již delší dobu, avšak s rozvojem strojového učení je stále složitější ho rozpoznat. Tento problém vytváří, ale může i řešit "Machine Learning".

2 Cíle práce

V této práci jsem si dal za cíl:

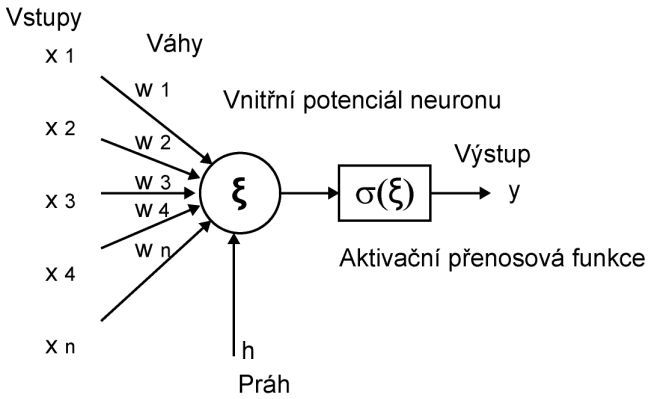
1. seznámit se se základními pojmy z oblasti neuronových sítí se zaměřením na hluboké učení,
2. podat přehled metod syntézy audiovizuální hlavy,
3. navrhnut perspektivní metodu syntézy audiovizuální hlavy,
4. připravit data pro experimenty,
5. experimentálně ověřit základní funkčnost metody,
6. zhodnotit dosažené výsledky a doporučit směr další práce.

3 Přehled důležitých pojmů z oblasti neuronových sítí

V práci se budu odkazovat na termíny, přístupy a metody používané k datu současného vědeckého poznání (state-of-the-art) v oblasti učení hlubokých neuronových sítí.

Neuronová síť (Neural network NN) je systém skládající se z výpočetních jednotek - neuronů, které jsou mezi sebou vzájemně propojeny spoji ohodnocenými váhami. Síť dále můžeme učit adaptací těchto vah. Existuje více architektur NN. Základní architekturou, která je základem tzv. hlubokého učení (deep learning) je vícevrstvá perceptronová síť.

Na obrázku 1 je znázorněna stavba neuronu.



Obrázek 1: Model neuronu

3.1 Vícevrstvá perceptronová síť

Díky vícevrstvé perceptronové síti bylo možno vyřešit problém XOR, který nelze vyřešit pomocí jedné vrstvy. Vícevrstvá perceptronová síť je standartně řešena metodou učení s učitelem, tj. trénovací vzory musí obsahovat i hodnoty odpovídajících výstupů. Existuje velká řada variant architektury modelů vícevrstvé perceptronové sítě, ale já zde ukážei jen základní model.

3.1.1 Struktura sítě

Tzv. perceptronová síť obsahuje neurony, které jsou mezi sebou propojeny tak, že vytvářejí graf. Kde vnitřní potenciál každého neuronu ξ je počítán jako vážený součet vstupů, dle vztahu (1).

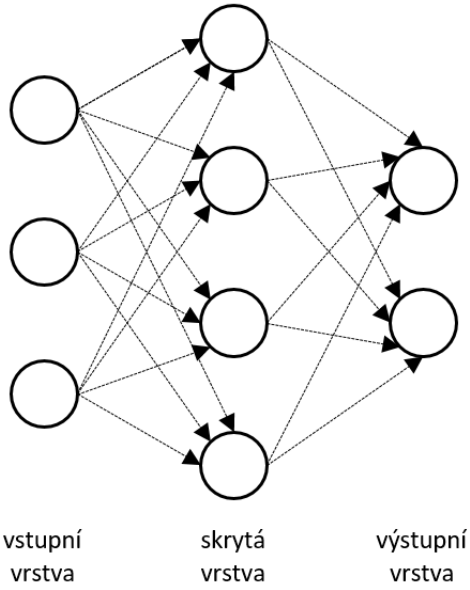
$$\xi = \sum_{i=1}^{n} w_i x_i - \theta = \sum_{i=0}^{n} w_i x_i, \tag{1}$$

kde θ je práh, $x_1, ..., x_n$ jseou vstupy $w_1, ..., w_n$ jsou váhy spojů, $x_0 = 1$ formální vstup, $w_0 = -\theta$. Výstupem je hodnota, tzv. aktivační funkce $f(\xi)$. Aktivační funkce může být různého druhu, nejzákladnější je např. sigmoida / sigmoidální funkce / logistická funkce.

$$f(\xi) = \frac{1}{1 + e^{-\lambda \xi}}, \tag{2}$$

kde λ je parametr strmosti (gain).

Pro neuronové sítě se sigmoda používá jako inspirace biologickým neuronem, jehož výstup je chápán jako frekvence tzv. pálení (firing) biologického neuronu. Hodnota 0 přirozeně představuje minimum výstupu biologického neuronu. Maximální frekvence pálení biologického neuronu je dána kapacitou neuronové buňky obnovovat nebo odstraňovat ionty draslíku, sodíku a případně i vápníku. Sigmoidální funkce je přirozeným modelem funkce biologického neuronu. Pro umělý neuron je dostatečná jakákoli (skoro všude) diferencovatelná funkce a proto je často volena ta nejjednuduší, a za tu je pokládána Rectified Linear Unit (ReLU) $f(x) = \begin{cases} x & \text{když } x > 0, \\ 0 & \text{jinak.} \end{cases}$



Obrázek 2: Uspořádání třívrstvé perceptronové sítě

Vícevrstvá NN je pak tvořena vrstvami neuronů viz obrázek (2). Perceptrony jednotlivých vrstev jsou uspořádány tak, že tvoří úplný bipartitní graf, tj. výstup jednoho neuronu vrstvy je poslán do vstupů všech perceptronů následující vrstvy. Poslední vrstva se nazývá výstupní vrstva, ostatní se nazývají skryté. Počet vrstev a počet neuronů jsou pak parametry architektury dané sítě a jejich volba záleží na povaze úlohy, hlavně s ohledem na množství dat. V praxi se stanovují tak, abychom dosáhli co nejlepších výsledků na zvolené "objektivní"funkci, která bývá často volena heuristicky.

Na výstup poslední vrstvy se může aplikovat např. funkce softmax, jejíž předpis je:

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ pro } i = 1, \ldots, K \text{ a } \boldsymbol{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K. \tag{3}$$

Každý perceptron používá vážený součet vstupů, tedy jejich lineární kombinaci, což odpovídá rovnici nadroviny [21], jejíž výstup je následně zpracován aktivační funkcí.

Vícevrstvá perceptronová síť pracuje s reálnými hodnotami, tedy všechny výstupy, vstupy, hodnoty vah i potenciály jsou většinou reálná čísla. Ke správnému fungování sítě, kromě návrhu její architektury, ještě musíme správně nastavit váhy, to je předmětem učení/trénování sítě.

3.1.2 Učení sítě

Snahou učení je dosáhnout takového nastavení vah, aby byla chyba E mezi skutečnými a požadovanými výstupy minimální. Proto musíme definovat chybu sítě E:

$$E = \sum_k E_k, \tag{4}$$

kde k probíhá přes všechna trénovací data a E_k je chybová, či též ztrátová funkce tj. chyba k-tého trénovacího vzoru. Neexistuje jedna ztrátová funkce pro všechny úlohy a je nutné volit každou z hlediska toho 1) co pro nás znamená ztráta a 2) jakou chybu jsme schopni použít v algoritmu učení. Základní je např. definována vztahem:

$$E_k = \sum_j (y_j - d_{kj})^2, \tag{5}$$

kde j probíhá přes všechny výstupy vrstvy a d_{kj} je j-tý element požadovaného výstupu k-tého vzoru. Tato funkce vychází z L^2 metriky ($E_k = (L^2)^2 = L2$). Zobecnění na L^p metriku je pak zřejmé. Např. pro L^1 metriku platí:

$$E_k = \sum_j |(y_j - d_{kj})|, \tag{6}$$

Pro optimalizaci úlohy se v základním modelu používá gradientní metoda. Na začátku učení se všechny váhy nastaví na nějaké počáteční hodnoty [9]. Poté se předkládají jednotlivé trénovací vzory, pro každý se spočítá chyba E_k a tato chyba se akumuluje, takže na konci dostaneme výslednou chybu E a na jejím základě upravíme hodnoty vah podle vztahu:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}, \tag{7}$$

kde změny vah získáme ze vztahu:

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}, \tag{8}$$

kde η je míra učení a platí $\eta > 0$ a w_{ij} váhu mezi i-tým a j-tým, kde i jsou indexy probíhající přes neurony nižší a j jsou indexy probíhající přes indexy vyšší ze dvou sousedních vrstev. Celý postup je opakován tak dlouho než je chyba menší než stanovená mez, popř. než algoritmus učení přeruší výpočet, aby zabránil přetrénování sítě, jako např. v algoritmu early stopping.

3.1.3 Metoda backpropagation

Jedná se o metodu učení parametrů vícevrstvé NN gradientní metodou. Postupujeme odzadu, tj. od poslední vrstvy, protože primární informaci o chybě máme až ve výstupní vrstvě a do ostatních ji musíme postupně odzadu převést. Parciální derivace chyby podle vah w_{ij} je

$$\frac{\partial E}{\partial w_{ij}} = \sum_k \frac{\partial E_k}{\partial w_{ij}} \tag{9}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E_k}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial \xi_j}{\partial w_{ij}}, \tag{10}$$

kde $y_j = f(\xi_j)$ a $\xi_j = \sum_i w_{ij} y_i$, aktivační funkce f je sigmoida, vztah 2. Parciální derivaci $\frac{\partial \xi_j}{\partial w_{ij}}$ pak získáme derivací vztahu $\xi_j = \sum_i w_{ij} y_i$.

$$\frac{\partial \xi_j}{\partial w_{ij}} = y_i$$

Parciální derivaci $\frac{\partial y_j}{\partial \xi_j}$ získáme derivací aktivační funkce (pro náš případ sigmoida).

$$\frac{\partial y_j}{\partial \xi_j} = \lambda y_j (1 - y_j)$$

Hodnotu $\frac{\partial E_k}{\partial y_j}$ získáme metodou zpětného šíření, tj postupným procházením sítě vrstvu po vrstvě směrem od poslední vrstvy zpět k předchozím. Parciální derivace chyby ve výstupní vrstvě je pro ztrátovou funkci (5) dána dvojnásobkem rozdílu mezi aktuálním a požadovaným výstupem.

$$\frac{\partial E_k}{\partial y_j} = 2(y_j - d_{kj}),$$

kde index j prochází přes neurony výstupní vrstvy. Parciální derivace chyby pro skryté vrstvy počítáme dle vztahu

$$\frac{\partial E_k}{\partial y_j} = \sum_r \frac{\partial E_k}{\partial y_r} \frac{\partial y_r}{\partial \xi_r} \frac{\partial \xi_r}{\partial y_j} = \sum \frac{\partial E_k}{\partial y_r} \lambda y_r (1 - y_r) w_{rj},$$

kde index r prochází přes všechny neurony do nichž vede výstup z neuronu j a $\frac{\partial E_k}{\partial y_r}$ jsou známé z předchozího výpočtu [21].

3.1.4 Metody strojového hlubokého učení NN

Při použití gradientní metody, se nám může stát, že se při hledání (ideálně globálního) minima chybové funkce může metody zastavit v nežádoucím lokálním minimu. Toto nastává poměrně často a proto byla nalezena řada metod pro umenšení tohoto problému.

Velikost míry učení η: Změna velikosti míry učení významně ovlivňuje učení sítě, zejména rychlost učení a konvergenci k řešení. Při malé hodnotě klesá chyba pomalu, což vede k dlouhému učení, naopak při velkém kroku se nám může stát, že síť bude divergovat. Na začátku se doporučuje začínat s nízkou hodnotou η, kterou budeme postupně zvyšovat.

Moment: Přidání momentu do rovnice učení vah (8) je nejrozšířenější ochranou před zastavením v lokálním minimu a inflexním bodě.

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t-1),$$

kde α je parametrem momentu, nabývající hodnot $0 < \alpha < 1$, hodnotu parametru volíme blízko jedné. Moment představuje krokovou setrvačnost, míříme-li tedy k lokálnímu minimu, můžeme ho překročit. Existují různá další zlepšení založené na nějaké heuristice, jednou z nejpoužívanějších je např. metoda ADAM [18].

3.2 Sítě pamatující si historii

Jelikož animace mluvící hlavy z řeči vyžaduje jak extrakci výrazu tváře z řeči tak i extrakci parametrů pohyb hlavy z řeči, kdy jsou tyto úlohy silně závislé na historii, je vhodné použít síť pamatující si historii. Po následující sekci o rekurentních neuronových sítích si v dalších dvou sekcích představíme 2 architektury sítí, které se budou aplikovat v naší úloze mluvící hlavy.

3.2.1 Rekurentní neuronové sítě

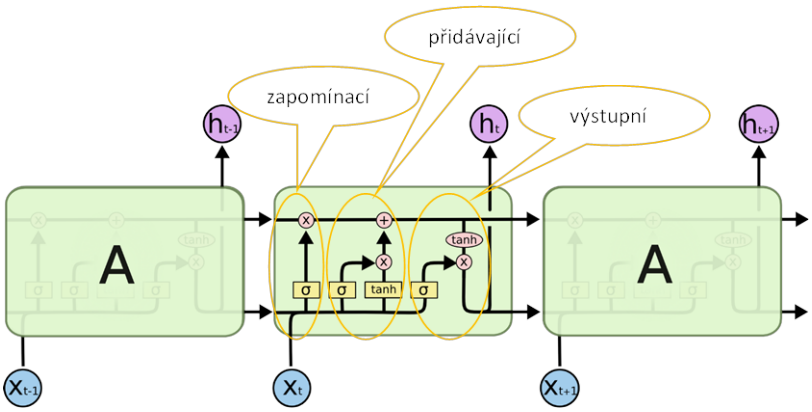
Rekurentní neuronové sítě jsou neuronové sítě s vnitřním stavem, jehož minulá hodnota (typicky v čase $t-1$) je (společně se standardním vstupem) použita pro určení vnitřního stavu (a výstupu) v čase t. Pokud označíme vnitřní stav v čase $t-1$ zápisem $\boldsymbol{h}_{t-1}$ a vstup v čase t jako $\boldsymbol{x}_t$, je vnitřní stav $\boldsymbol{h}_t$ v čase t možno vyjádřit vztahem:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

nebo po rozepsání funkce f vztahem

$$\boldsymbol{h}_t = g(\boldsymbol{W}_{\boldsymbol{h},\boldsymbol{h}} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{\boldsymbol{h},\boldsymbol{x}} \boldsymbol{x}_t),$$

kde g je aktivační funkce. Aktuální výstup $\boldsymbol{y}_t$ je pak odvozen od aktuálního stavu $\boldsymbol{h_t}$. Pokud v těchto vztazích nahradíme $\boldsymbol{h}_t$ proměnnou $\boldsymbol{x}_{t+1}$ a $\boldsymbol{x}_t$ nahradíme proměnnou $\boldsymbol{u}_t$, dostaneme stavový popis dynamického t-invariantního nelineárního systému používaného v teorii řízení. Typické RNN jsou náchylné k tzv.



Obrázek 3: LSTM (obr. převzat z [23])

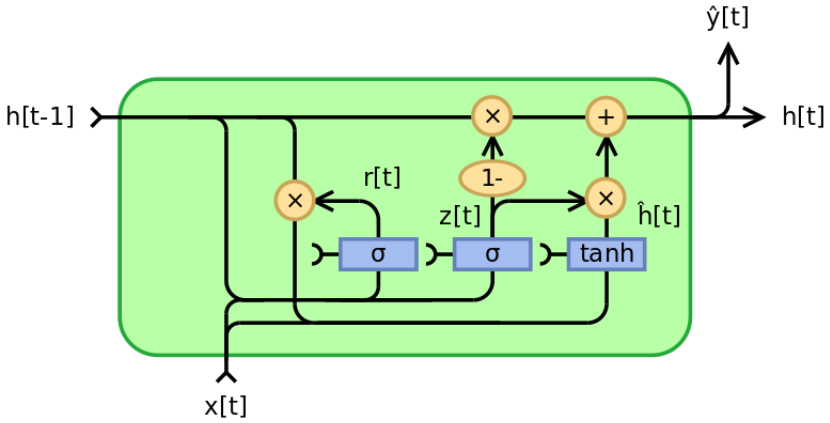
vanishing gradientu (vymizení gradientu) nebo naopak k tzv. exploding (přetečení) gradientu. Tento problém je řešen zavedením Long-Short-Term–Memory nebo Gated-Recurrent-Unit sítí.

3.2.2 Long-Short-Term memory

Long-Short-Term memory (LSTM) je síť pamatující si historii, která řeší problém vymizení gradientu pomocí své architektury bran (především zapomínací). Zobrazení struktury LSTM sítě je na obr. 3, kde síť A v každém časovém kroku x_t upgraduje svůj stav pomocí zapomínací, přidávací a výstupní brány. LSTM síť je speciálním typem rekurentní neuronové sítě (Recurrent Neural Network, RNN), tj. sítě, která si pamatuje svůj stav podobně jako je tomu v teorii (ne)lineárních dynamických časově invariantních systémů. LSTM sítě také můžeme považovat za RNN sítě druhého řádu (2nd order RNN), které používají více vah $w_{i,j,k}$ než jen váhy $w_{i,j}$ mezi sousedními vrstvami i a j. LSTMs jsou cenným prostředkem v úlohách hlubokého učení a to i přes některé své nevýhody, jako např. dlouhou dobu trénování, velké množství potřebné paměti, náchylnost k přetrénování, nesnadné aplikování drop-outu a relativně vysoký vliv počátečních hodnot vah na výsledné chování sítě.

3.2.3 Gated Reccurent Unit

Gated Reccurent Unit (GRU) je v podstatě zjednodušená LSTM síť. Pamatuje si historii a problém vymizení gradientu řeší stejně jako LSTM branami propouštějícími vstupy jen do určité vzdálenosti. Schéma sítě GRU viz obr. 4.



Obrázek 4: GRU (obr. převzat z https://commons.wikimedia.org/w/index.php?curid=66225938)

kde na rozdíl od sítě LSTM architektura GRU nemá výstupní bránu. Některé typy GRU jsou ještě jednodušší tím, že nemají bias nebo je každá brána počítána jen z biasu.

3.3 Heatmaps

Heatmaps, neboli tepelné mapy, jsou technikou vizualizace dat, která zobrazuje velikost jevu intenzitou barvy ve 2D prostoru. Existují 2 hlavní kategorie:

1. Shluková

   Shlukové heatmapy zobrazují velikost daného parametru jako barvu ve dvourozměrné matici, kdy každá dimenze představuje kategorii vlastností a barva velikost sledovaného parametru.

2. Prostorová

   Zobrazuje velikost sledovaného parametru jako barvu na nějakém prostoru, obvykle mapě. Příkladem prostorové heatmapy je např. zobrazení teploty na mapě pomocí překrytí barvami. (od modré - nejnižší po červenou- nejvyšší)

Tato technika se požívá zejména pro analýzu dat, ale lze ji použít i třeba pro odhad výrazu tváře [3].

3.4 Afinní pole částí

Zavedení afinního pole částí (Part Affinity Fields, PAF) pomáhá v úlohách strojového učení s lokalizací anatomické části člověka na 2D snímku. Lidský odhad je z velké části zaměřen na těla jednotlivců, ale odvození pózy více lidí na snímku představuje výzvu. Za prvé každý obrázek může obsahovat neznámý

počet lidí, za druhé interakce mezi lidmi vyvolává rušení. Problém rozlišení osob a jejich končetin se řeší různými způsoby, jedním z nich jsou právě afinní pole částí PAF. Na obrázku níže vidíme rozlišení končetin jednotlivců pomocí PAF. [4]

Obrázek 5: Zobrazení končetin afinním polem částí (obr. převzat z [4])

3.5 Konvoluční síť

Vzhledem k tomu, že dále v textu budeme odkazovat na konvoluční sítě, v krátkosti si je představíme.

Existence konvolučních sítí vychází z biologického uspořádání neuronů lidského mozku pro zpracování obrazu ze sítnice, kdy jsou všechny neurony stejné, místo toho, aby se na každou část obrazu jednotlivé neurony specializovaly. Přesně tento pohled se hodí pro strojové rozpoznávání obrazu, kdy chceme mít rozsáhlou síť s minimem parametrů, zároveň chceme zpracovávat každý pixel obrazu stejným způsobem.

Konvoluční síť se využívá primárně při práci s obrázky. Obraz si můžeme představit jako matici, jejíž hodnoty představují intenzitu barvy. Přesněji pak pro barevný obrázek máme 3 matice, každá z nich pro jednu z barev RGB. V případě, že máme velký obrázek, např. 7680x4320, by byl počet vstupů do neuronové sítě obrovský. Proto je třeba obraz upravit do podoby více vhodné pro následné zpracování. Představíme si sítě Convolutional Neural Network, dále jen CNN. Algoritmus využívá operace Convolution a Max-pooling.

3.5.1 Operace Convolution

Při této operaci máme na vstupu matici, kterou "proženeme"přes námi zvolený filtr, v případě, že máme filtrů n, pak na výstupu dostaneme n upravených ma-

tic. Postup je takový, že postupně přikládáme matici filtru F na matici obrázku I tak, že pod každým prvkem matice F leží nějaký prvek matice I. Začínáme v levém horním rohu (co nejvíce vlevo nahoře) matice I a postupně posunujeme (většinou) o jeden prvek (pixel) filtr F po matici I. Velikost posunu nemusí být 1 a obecně záleží na velikosti parametru, který se nazývá stride. Pro každé přiložení F na I se pro každou dvojici překrytých prvků (hodnot) spočítá součin těchto dvou hodnot, takto vzniklé součiny se následně sečtou pro všechny součiny vzniklé jedním přiložením filtru F na I a výsledek se zapíše do nové matice pro danou pozici. Pro první pozici se výsledek zapíše do levého horního rohu nové matice a pro všechny další posuny se jejich výsledky do nové matice zapisují ve směru těchto posunů.

Demonstrace na příkladu:

Vstupem je matice Image 4 x 4 a máme 2 filtry F (Filter I, Filter II) velikosti 2 x 2

$$Image = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$FilterI = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$FilterII = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Výsledkem budou 2 matice řádu 3 x 3.

$$Image * FilterI = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$Image * FilterII = \begin{bmatrix} 3 & 4 & 3 \\ 4 & 3 & 1 \\ 2 & 1 & 0 \end{bmatrix}$$

Tato metoda nám zkracuje dimenzi obrazu, kdybychom potřebovali dimenzi zachovat, můžeme vstupní matici "rozšířit"nulami, tzv. paddingem: pro náš případ např.

$$Image = I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.6 Max-pooling

Na vstupu máme matici I velikosti m x n, a velikostně námi zvolenou matici F. Postup je takový, že postupně přikládáme matici filtru F na matici obrázku I tak, že pod každým prvkem matice F leží nějaký prvek matice I. Začínáme v

levém horním rohu (co nejvíce vlevo nahoře) matice I a postupně posunujeme (většinou) o jeden prvek (pixel) filtr F po matici I. Velikost posunu nemusí být 1, a obecně se jedná o tzv. stride. Pro každé přiložení F na I se z prvků matice I, které leží pod maticí F vybere prvek s maximální hodnotou a výsledek se zapíše do nové matice pro danou pozici. Pro první pozici se zapisuje do levého horního rohu nové matice a pro všechny další posuny se do nové matice zapisuje ve směru těchto posunů. Výhodou metody max-pooling je, že po její aplikaci je zachována diferencovatelnost. Metoda Median-pooling se liší jen v tom, že vybíráme místo maximální hodnoty medián.

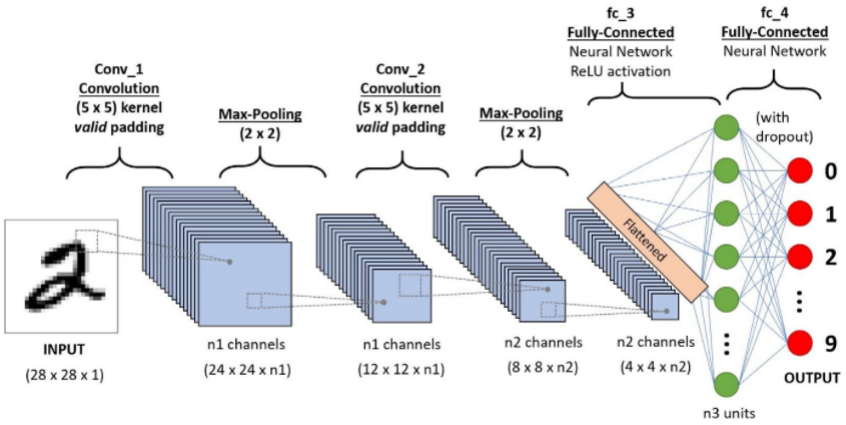
Operaci budeme opět demonstrovat na příkladu:

$$Image = I = \begin{bmatrix} 13 & 4 & 14 & 8 \\ 4 & 17 & 1 & 15 \\ 18 & 1 & 5 & 20 \\ 10 & 6 & 30 & 1 \end{bmatrix},$$

velikost okénka = 2 x 2, stride = 2;

$$Max - pooling2x2 = \begin{bmatrix} 17 & 15 \\ 18 & 30 \end{bmatrix}$$

Operace Max-pooling 2 x 2 v našem případě zmenšuje dimenzi výstupního obrazu na polovinu, avšak výsledná velikost dimenze závisí na velikosti kroku, stride.



Obrázek 6: CNN: sekvence pro určování ručně psaných písmen (obr. převzat z [27])

Na obr. 6 vysvětlíme strukturu sítě metody zpracování obrazu klasifikace do pevného počtu tříd.

Jako vstup zde dostáváme obrázek (matice 28 x 28), na kterém provedeme konvoluční operaci [3.5.1] pro n_1 filtrů velikosti 5 x 5. Na vzniklé matice aplikujeme metodu Max-pooling [3.6] 2 x 2. Poté znovu použijeme konvoluční operaci (5 x 5) pro n_1 matic dostaneme n_2 ($n_2 > n_1$) nových, na výstup pak opět použijeme Max-pooling (2 x 2). Výslednou sadu matic pak připojíme do neuronové sítě, abychom tak mohli učinit, přepisujeme každou matici do sloupečku, dohodnuté pravidlo je, že začínáme v levém horním rohu a postupujeme po řádcích. Všechny takto vzniklé sloupce spojíme do jednoho vektoru. Hodnoty všech filtrů se počítají během učení sítě, neboť je lze chápat jako argumenty ztrátové funkce, pro kterou hledáme takový bod v prostoru jejích argumentů (tzv. parametrickém prostoru), pro nějž ztrátová funkce nabývá minima.

$$Image = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 9 & 8 & 7 \end{bmatrix} -- > \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 9 \\ 8 \\ 7 \end{bmatrix}$$

3.7 Drop-out

Neuronové sítě se mohou i snadno přetrénovat, tj. naučit se skvěle řešit úlohu, pokud na jejich vstup přivedeme trénovací data, ale na nových dosud neviděných datech jejich výkon (úspěšnost nalezení optimálního řešení úlohy) může prudce poklesnout. Z tohoto důvodu se zavádí tzv. regularizační techniky, jejichž cílem je donutit neuronovou síť se během trénování naučit zobecňovat schopnost řešit úlohu pro dosud (v tréninku sítě) neviděná data. Protože bylo zjištěno, že přetrénování sítě se dá bránit použitím sady více sítí s různou architekturou (ale za cenu nadměrné výpočetní náročnosti), byla snaha nahradit přístup použití více sítí výpočetně jednodušším mechanismem. Tento mechanismus se jmenuje drop-out a jeho hlavní myšlenka spočívá ve změně struktury jedné a té samé sítě během trénování. Struktura sítě se pak jednoduše mění tím, že se v jednotlivých krocích trénování sítě (kroku gradientní metody) ze sítě (náhodně) vypustí některé její neurony. Tím se mění struktura sítě a tento faktor podporuje síť v nalezení řešení pro zobecněná data.

3.8 Dilated Neural Network

Dilated Neural Network, česky možná roztažené sítě, zavádějí do konvolučních vrstev další parametr zvaný rychlost dilatace. Tato definuje mezeru mezi hodnotami v matici filtru (jádře, kernelu). Například filtr dimenze 3 x 3 s rychlostí dilatace 2 se bude chovat jako filtr dimenze 5 x 5 (bude mít stejné "zorné pole jako filtr 5 x 5). Můžeme si představit, že se 9 hodnot původního filtru roztáhne

rovnoměrně ve všech dimenzích (v našem případě dvou) do velikosti filtru 5 x 5, ale matice tohoto filtru velikosti 5 x 5 má jen 9 prvků, se kterými filtr počítá, a zbylých 25 prvků "jalových", se kterými se výpočet filtrace neprovádí. Matici vzniklého filtru 5 x 5 si můžeme představit tak, že z matice 5 x 5 odstraníme každý druhý sloupec a řádek.

Příklad:

Vstupem je matice Image 5 x 5 a Filter F 3 x 3 s dilation 2:

$$Image = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Dilated filtr DF:

$$\begin{bmatrix} 1 & \square & 2 & \square & 3 \\ \square & \square & \square & \square & \square \\ 4 & \square & 5 & \square & 6 \\ \square & \square & \square & \square & \square \\ 7 & \square & 8 & \square & 9 \end{bmatrix},$$

kde s DF se počítá jako s filtrem 5 x 5, ale jen s jeho prvky různými od prvku $\square$.

Výsledek bez paddingu:

$$Image * DF = \begin{bmatrix} 15 \end{bmatrix}.$$

3.9 Generative Adversarial Network

Generative Adversarial Network (GAN), česky snad síť trénování protivníků, je architektura umožňující generování "umělých"dat z náhodného vstupu ("bílého šumu"), kterou vytvořil kanadský tým z Universite de Montreal [13] v roce 2014. Cílem je, aby vygenerovaná data byla k nerozeznání podobná datům reálným. GAN je založena na dvou proti sobě soupeřících sítích: generátoru a diskriminátoru:

1. Generátor se snaží přelstít diskriminátor, tzn. vygenerovat co nejvěrohodnější data (např. obraz), tedy taková, aby je diskriminátor vyhodnotil jako data skutečná.

2. Diskriminátor se naopak co nejvíce snaží odhalovat umělá vygenerovaná data a rozlišovat je od skutečných.

Každá síť se musí trénovat zvlášť podle jiného kritéria, jelikož kdyby se trénovaly jako jedna síť, přestaly by soupeřit a diskriminátor by například mohl všechna data považovat za reálná.

Ztrátových funkcí pro nastavení parametrů diskriminátoru $\mathcal{L}_{GAN}(D)$ a generátoru $\mathcal{L}_{GAN}(G)$ (tj. pro natrénování sítě) může být více. Původní ztrátová funkce je min-max funkce tvaru:

$$\mathcal{L}_{GAN}(D) = log(D(\boldsymbol{x})) + log(1 - D(G(\boldsymbol{z}))), \tag{11}$$

kde $D(\boldsymbol{x})$ je výstup diskriminátoru D pro reálný obraz $\boldsymbol{x}$ a $D(G(\boldsymbol{z}))$ je výstup diskriminátoru pro obraz $G(\boldsymbol{z})$ syntetizovaný generátorem G. Jinou ztrátovou funkcí je LSGAN (Least Square GAN):

$$\mathcal{L}_{GAN}(D) = (D(\boldsymbol{x}) - 1)^2 + D(G(\boldsymbol{z}))^2, \tag{12}$$

$$\mathcal{L}_{GAN}(G) = (D(G(\boldsymbol{z})) - 1)^2. \tag{13}$$

Speciálními případy GAN jsou například sítě:

1. Podmíněný (conditional) GAN. Standardní GAN generuje data podle sdružené pravděpodobnosti $P(\boldsymbol{X}, \boldsymbol{Y})$, kde $\boldsymbol{X}$ jsou data (např. obrázek) a $\boldsymbol{Y}$ třída (např. informace, zda je na obrázku kočka nebo pes). Vstupem standardního GAN je šum. Conditional GAN generuje data podle podmíněné pravděpodobnosti $P(\boldsymbol{X}|\boldsymbol{Y})$. To znamená, že síti můžeme předepsat, jaký druh obrázku má vytvořit (zda kočku nebo psa). Vstupem Conditional GAN je šum a značky třídy, jejíž instance se má vygenerovat). [22]

2. CycleGAN modifikuje obrázky jedné třídy na jiné obrázky tak, aby mohly patřit k jiné třídě. Například může obrázek koně modifikovat na obrázek zebry. Přitom pro trénování nepotřebuje připravit páry, které by jej pomohly učit, jak modifikaci provádět. Tréninková data jsou tvořena dvěma množinami obrázků (např. množina obrázků koní a množina obrázků zeber). [36]

3. Image-to-Image Translation síť. Tento druh GAN vstupní obrázek převádí na výstupní obrázek s různými vlastnostmi. Např.obrázek skici tváře převede na fotorealistický obrázek tváře. Ztrátová funkce je vyjádřena jako vážený průměr standardní ztráty diskriminátoru a ztráty rozdílů pixelů (pixel-wise loss) mezi zdrojovým a generovaným obrázkem. Architektura Image-to-Image Translation sítě může být různá, ale může být realizována též jako U-Net. [14]

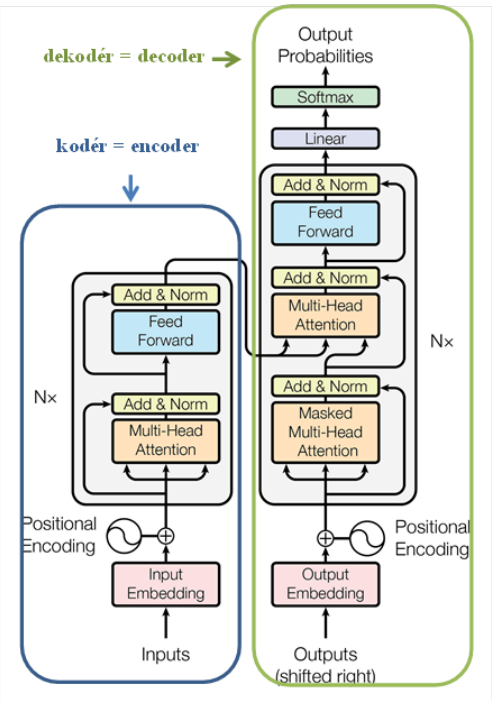
,

3.10 Associative-and-Adversarial Networks

Asociativní adversariální sít (AAN) je v zásadě GAN rozšířená o asociativní paměť, která je umístěna mezi generátorem a diskriminátorem a spojuje obě tyto sítě (tj. síť generátoru a síť diskriminátoru) standardního GAN dohromady. Asociativní paměť je stochastický generativní model v článku [1] realizovaný omezeným Boltzmannovým strojem (Restricted Boltzmann Machines (RBMs). Asociativní paměť se učí pravděpodobnostní rozložení vysokoúrovňových příznaků generovaných diskriminátorem. Toto rozdělení se pak použije ke generování vstupu generátoru (místo generátoru (bílého) šumu užívaného standardní (tj. bez neasociativní paměti) sítě GAN).

3.11 Transformery

Transformery jsou neuronové sítě spadající do oblasti hlubokého učení, které mají speciální strukturou a vlastnostmi. Transformery byly původně vyvinuté pro automatický překlad textu přirozeného jazyka, ale jejich použití je daleko širší. Základní vlastností transformerů je, že obsahují tzv. attention (pozornostní) vrstvy. Dokonce původní článek, který uvedl transformery a popsal jejich strukturu se nazývá „Attention Is All You Need“ [32]. Tyto vrstvy umožňují transformeru se soustředit na důležité události (vstupu i výstupu), které mohou nastat v širokém kontextu (např. pořadí slov je v různých jazycích různé a je třeba při překladu pracovat s velkým kontextem).



Obrázek 7: Struktura Transformeru převzato z Attention Is All You Need (obr. převzat z [32])

Na obr. 7 je obecná struktura transformeru. V některých případech je ale struktura jednodušší a záleží na typu úlohy, pro kterou se transformer použije:

1. Transformer obsahující jak kodér, tak i dekodér (encoder-decoder model, sequence-to-sequence model) – pro úlohu, jejímž cílem je vygenerovat posloupnost cílových symbolů na základě známé posloupnosti vstupních symbolů. Představiteli takové úlohy jsou např. strojový překlad, sumarizace, rekonstrukce, převod mezi modalitami (řeči do textu, textu do řeči, rozpoznávání či syntéza znakového jazyka). (Příkladem je T5.)

2. Transformer obsahující pouze kodér (encoder model) – pro úlohu, jejímž cílem je klasifikovat na základě známé posloupnosti vstupních symbolů. Představiteli takové úlohy jsou např. klasifikace vstupní věty, named entity recognition, automatická interpunkce. (Příkladem je BERT.)

3. Transformer obsahující pouze dekodér (decoder model) – pro úlohu, jejímž cílem je vygenerovat výstup, jako například generátor textu. (Příkladem je GPT3.)

Během trénování kodér (encoder) dostává vstup (v úloze překladu věty zdrojového jazyka posloupnost slov zdrojového jazyka) a dekodér (decoder) do-

stává informaci od učitele (větu cílového jazyka). Ale na rozdíl od kodéru, dekodér nevidí v případě úlohy strojového překladu celou větu cílového jazyka, ale jen část, která byla dosud přeložena – tj. minulost, nikoliv přítomnost a budoucnost.

Nejvýznamnější součástí architektury je vrstva pozornosti, které budeme dále říkat attention vrstva. Každá položka vstupní posloupnosti (např. slovo v NLP úloze) je reprezentovaná vektorem $\boldsymbol{w}$ (angl. input embedding), který se vytváří na nejnižších (vstupních) vrstvách transformeru ještě před příchodem do první attention vrstvy transformeru. Síť se ve fázi trénování pro každou svou attention vrstvu (pro zjednodušení ve výkladu se omezíme na případ jedné hlavy tzv, head) naučí tři matice: $\boldsymbol{Q}_W$ (queries), $\boldsymbol{K}_W$ (keys) a $\boldsymbol{V}_W$ (values). Pokud všechny (v rámci vyšetřovaného slovního kontextu, např. v rámci věty) řádkové vstupní vektory (vstupní slova) $\boldsymbol{w_i}$ zapíšeme po řádcích do matice $\boldsymbol{W}$ (tj. každý řádek matice $\boldsymbol{W}$ obsahuje jeden vektor $\boldsymbol{w_i}$) a za nutné podmínky, že počet řádek matic $\boldsymbol{Q}_W$ (queries), $\boldsymbol{K}_W$ (keys) a $\boldsymbol{V}_W$ (values) je roven dimenzi vektoru $\boldsymbol{w_i}$, a za předpokladu, že dimenze vektoru queries a vektoru keyes (tj. počet sloupců matice $\boldsymbol{Q}_W$ (queries) a počet sloupců matice $\boldsymbol{K}_W$ (keys)) jsou stejné, můžeme spočítat ke každé položce (slovu), jakou má výslednou attention (jakou pozornost si toto slovo zasluhuje s uvážením jeho kontextu). Attention $A(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ je pak možno spočítat maticovým násobením:

$$\boldsymbol{Q} = \boldsymbol{W}\boldsymbol{Q}_W$$
$$\boldsymbol{K} = \boldsymbol{W}\boldsymbol{K}_W$$
$$\boldsymbol{V} = \boldsymbol{W}\boldsymbol{V}_W$$
$$A(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}})\boldsymbol{V}.$$

Softmax se provádí pro každý řádek zvlášť podobně jako v Pythonu. Výsledkem je matice, jejíž každý řádek udává attention jednoho vstupního slova, a počet sloupců výsledné matice odpovídá počtu sloupců matice $\boldsymbol{V}$, tj. dimenzi vektoru values.

Poznámka: prakticky se kvůli numerické stabilitě součin $\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}$ před aplikací softmaxu násobí číslem $1/\sqrt{(d_k)}$, kde d_k je dimenze keys, tj. počet sloupců matice $\boldsymbol{K}_W$.

Je třeba upozornit, že model transformeru na rozdíl od rekurentních neuronových sítí nepoužívá rekurenci ani konvoluci a považuje každý datový údaj za nezávislý na druhém. Z důvodu odlišení různé pozice stejných symbolů na vstupu transformeru se do modelu přidává informace o poloze explicitně (aby se např. zachovala informace týkající se pořadí slov ve větě). Standardě se řeší přidání této informace tzv. pozičním kódováním (angl. positional encoding).

Je třeba ještě upozornit na další důležitou vlastnost transformerů, a to na využití kodéru transformeru pro klasifikační úlohu učení bez učitele. Toho se dociluje tak, že události, které má transformer rozpoznávat (identifikovat – např. při úloze rozpoznávat, kdy psát měkké „i“ a kdy tvrdé „y“) se v trénovacím textu vymaskují a transformer se pak ve fázi trénování učí tato písmena rozpoznávat na základě jejich (širokého) kontextu a známé informace, kde tato

písmena v původním textu leží. Přínosem je, že pro naučení kodéru je možno využít obrovské množství dat (data se nepotřebují značkovat informací od učitele, což by jinak byla časově velmi náročná úloha). Natrénovaný kodér se pak dá dotrénovat již jen na malé množině označených dat pro úlohy, které vyžadují informaci od učitele (např. rozpoznávání fonémů či celé řeči). Představitelem takového přístupu je např. model wav2vec 2.0 [2].

4 X2Face

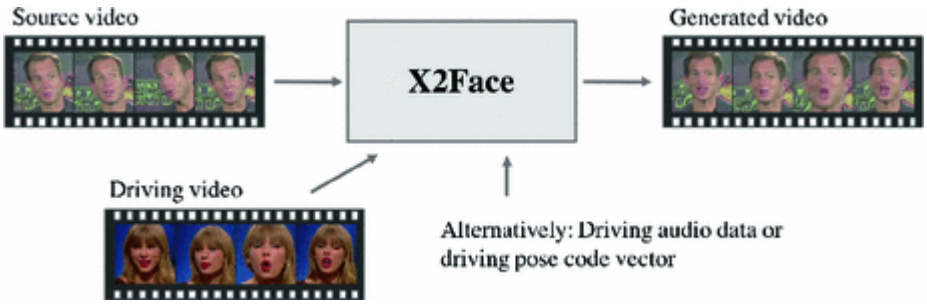
Dále popíšeme několik důležitých článků pojednávajících o metodách vizuální syntézy založených na neuronových sítích. Začneme s článkem popisujícím metodu X2Face [33]. X2Face je síť pro ovládání vytváření tváře pomocí obrázků, zvuku, kódů pozice a výrazu tváře.

4.1 Představení algoritmu X2Face

X2Face je síťová architektura s vlastním řízením, která umožňuje ovládání pózy (natočení tváře) a výrazu dané tváře řídicím vektorem, vizuální složkou jiné tváře (např. její pózou či výrazem) nebo složkou zvukovou (audiem řeči). Může být také použita pro lehké, sofistikované úpravy videa a obrázků.

Navrhovaný model přebírá v trénovacím čase dva vstupy: zdrojový frame (nebo posloupnost framů) videa (zdrojové video) a řídicí frame (nebo posloupnost framů) videa (řídicí video). Je natrénován k vytváření obličeje s identitou, pozadím a účesem zdrojového framu, kterému přiřadí pózu a výraz framu řídicího. Díky své architektuře se NN učí rozdělit problém na dva dílčí problémy zachycené ve dvou podsítích: síť pro vkládání a síť pro řízení (pózy a výrazu tváře). Síť pro vkládání identity tváře (tváře konkrétní osoby) se naučí reprezentaci vložené tváře - čelní frontalizaci tváře.

Řídicí síť se naučí namapovat vloženou reprezentaci tváře na cílovou pózu nebo výraz, které jsou určeny řídicím vektorem (kódem) nebo vizuální či zvukovou složkou videa stejné nebo i jiné osoby viz obr. 8.



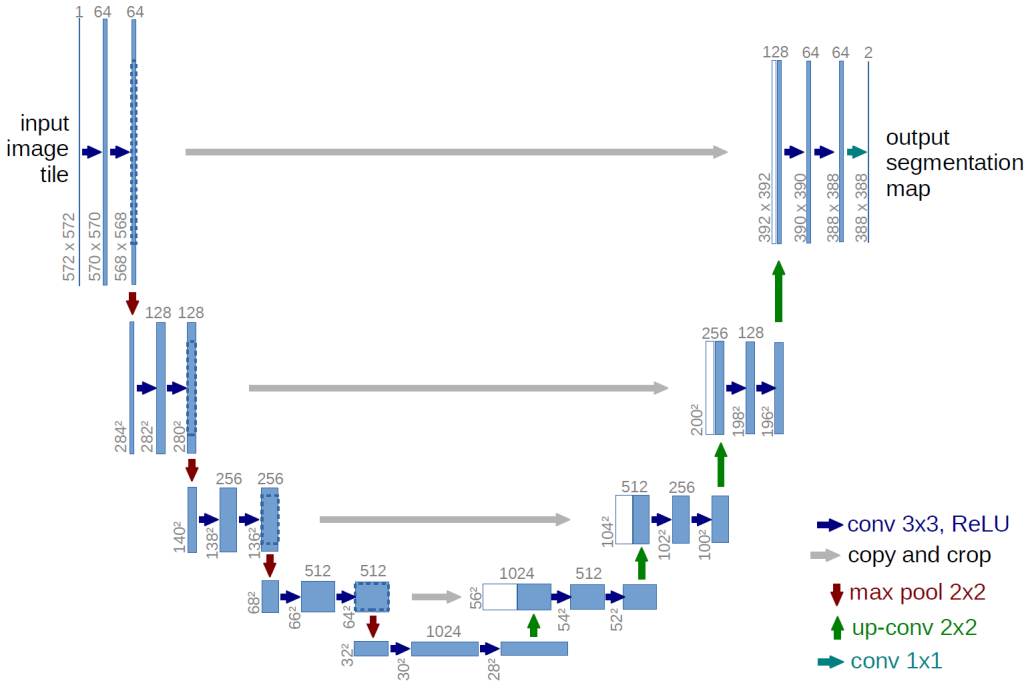
Obrázek 8: X2Face: model pro ovládání zdrojové tváře pomocí řídicího rámce (obr. převzat z [33])

4.2 Metoda natrénování sítě

Jak jsme již zmínili, síť během trénování přebírá dva vstupy: řídicí a zdrojový. Síť je dále rozložena na dvě podsítě, řídicí a vkládací s vlastním vstupem. Vstupem řídicí sítě je řídicí frame (posloupnost framů). Vstupem vkládací sítě je zdrojový frame (posloupnost framů).

4.2.1 Vkládací síť

Tato podsíť natrénuje bilineární vzorkovač, aby mapoval zdrojový vstup (video tváře) na reprezentaci tváře. Architektura sítě je založena na U-Net[24] a pix2pix[15]. Tato síť má pak tendenci vstupní obraz frontalizovat, ačkoliv k tomu není přímo nucena. Nicméně aby mohla generovat tvář aniž by znala pózu nebo výraz, musí mít společnou reprezentaci. Výstupem sítě je 2-kanálový obraz, kódující směr pohybu δx a δy každého pixelu. Architekturu U-net viz obr. 9.



Obrázek 9: U-net architektura (obr. převzat z [24])

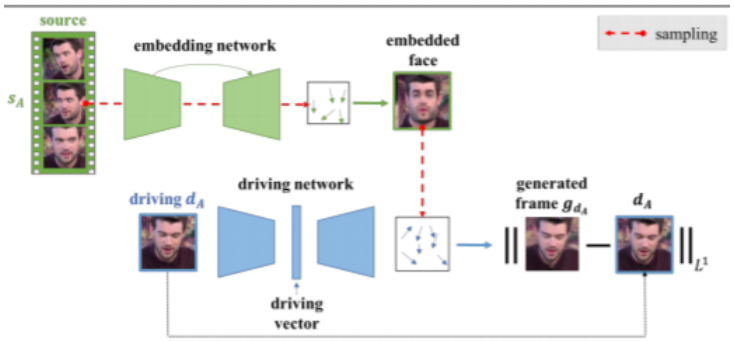
4.2.2 Řídicí síť

Řídicí síť bere řídicí frame (posloupnost framů) jako svůj vstup a naučí bilineární vzorkovač transformovat pixely vložené tváře do obrázku tváře pro generaci výsledného obrazu. Síť má strukturu kódér-dekodér. Aby mohl vzorkovat

správně vložený obraz a vyprodukovat výsledný, musí řídicí vektor zakódovat výraz/pózu atd.

4.2.3 Trénování sítě

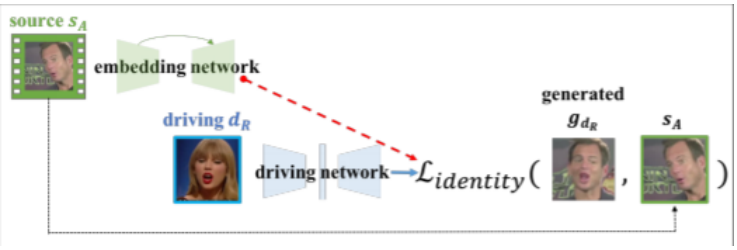
Síť je trénována ve dvou fázích, viz obr. 10.
I. V této fázi se počítá se stejnou identitou (v obou framech je stejná osoba) zdrojového a řídicího framu, proto dle článku [33] nám bude stačit ztrátová (pixel-wise) funkce $L1$ mezi generovaným a řídicím framem. Tato ztrátová funkce je dostačující pro řízení výrazu a pózy, avšak dochází při ní ke ztrátě informace o tvaru obličeje (prodlužují se hrany). Proto musíme vytvořit nové ztrátové funkce tzv. identitní ztrátové funkce $L_{identity}$, které použijeme ve druhé fázi učení. Učení v první fázi je v článku označeno jako učení bez učitele, nicméně to nejspíše znamená, že není potřeba žádná ruční anotace dat, ale že se ze zdrojového framu natočí tvář podle framu řídicího, ale jelikož se jedná o stejnou osobu, měl by být výsledek totžný s řídicím framem, takže řídicí frame slouží jako reference od učitele.



Obrázek 10: Síť X2Face během testovací fáze (obr. převzat z [33])

II. Ve druhé fázi se používají další 2 funkce ztráty identity (tj. identity osoby) k dosažení toho, aby identita mezi generovanými a zdrojovými snímky byla stejná bez ohledu na povahu identity řídicího framu. Ve druhé fázi se používá předtrénovaná CNN síť pro úlohu identifikaci řečníka. Jedná se o 11-vrstvou VGG síť (konfigurace A) [37] trénovanou na VGG-Face Datasetu [26]. Používají se dvě ztrátové funkce $L_{identity}$ (d_A, g_{dA}) a $L_{identity}$ (s_A, g_{dR}) založené na tzv. ztrátě obsahu (content loss, [13, 6]), kde d_A je řídicí frame identity A, g_{dA} je frame generovaný tímto řídícím framem d_A a zdrojovým framem s_A tytéž osoby A a g_{dR} je frame generovaný řídicím framem d_R osoby odlišné od A a a zdrojovým framem s_A osoby A. Pro $L_{identity}$ (d_A, g_{dA}) je užita fotometrická $L1$ ztráta a $L1$ ztráta obsahu na vrstvách Conv2-5 a Conv7, Pro $L_{identity}$ (s_A, g_{dR}) pak $L1$ ztráta obsahu na vrstvách Conv6-7 (tj. vrstev kódujících vysokoúrovňovou infromaci o identitě mezi g_{dA} and s_A. Příklad odlišné zdrojové a řídicí identity

je zobrazen na obr. 11.



Obrázek 11: $L_{identity}$ pro případ, kdy je zdrojová identita odlišná od řídicí identity (obr. převzat z [33])

4.3 Použití sítě pro jiné vstupy

Nejprve musíme síť přetrénovat pro určitý vstup, pro natrénovanou síť X2Face může řídicí vektor řídit daný obraz (tvář) i jinými vstupy než videoframem, například audiem nebo pózou.

5 Neural Voice Puppetry

V této kapitole si představíme další způsob audiovizuální syntézy a to konkrétně způsob popsaný v článku: "Neural Voice Puppetry: Audio-Driven Facial Reenactment"[29].

Pro danou zvukovou nahrávku tato metoda syntézy vyprodukuje vizuální realistický výstup určené osoby, který bude synchronizovaný se zvukovým vstupem. Neural Voice Puppetry má řadu případů použití, včetně zvukově řízených video avatarů, dabingu videa a textu. Mimo jiné lze tuto metodu použít i pro video syntézu mluvící hlavy. Demonstraci metody Neural Voice Puppetry vídíme na obr. 12.

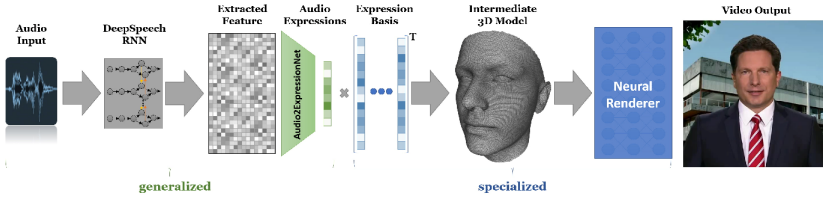
Obrázek 12: Demonstrace Neural Voice Puppetry (obr. převzat z [29])

5.1 Představení sítě

Neural Voice Puppetry se skládá ze dvou hlavních částí: obecné a specifické.

V obecné části se predikuje latentní výrazový vektor v prostoru zvukových výrazů. Tento prostor je sdílen mezi všemi osobami, umožňuje tedy přenos výrazu mezi osobami. Pro zajištění generalizace zvuku je použita předtrénovaná síť [11] převádějící řeč na text, tento text se následně použje jako vstup pro generaci výrazu.

Ve specifické části vytvoříme 3D model dané osoby, včetně specifických výrazů při mluvení jednotlivce. K vytvoření 3D modelu potřebujeme 2-3 minutovou videonahrávku dané osoby, ze které vytvoříme 3D model, na který budeme moci aplikovat zvukové výrazy. Na obr. 13 je zobrazeno schéma metody.



Obrázek 13: Neural Voice Puppetry (obr. převzat z [29])

5.2 Metoda generování modelu tváře

Aby tvůrci metody umožnili fotorealistickou rekonstrukci obličeje na základě zvukových signálů, použili 3D model obličeje jako přechodnou reprezentaci pohybu obličeje. Klíčovou součástí je odhad výrazu založený na zvuku. Jelikož se styl mluvy a tudíž i výraz obličeje jednotlivých lidí liší, stanovuje se pro každou osobu speciální výrazový prostor, který lze určit pro každou cílovou sekvenci. Abychom dokázali přiřadit k řeči obecně jakéhokoliv řečníka požadovaný pohyb tváře, musíme systém generalizovat. Pro generalizaci na více osob je použit latentní prostor zvukových výrazů, tento prostor sdílejí všechny osoby. Z tohoto zvukově výrazového prostoru lze mapovat do prostoru specifického. Vzhledem k odhadům výrazů a extrahované zvukové funkci autoři aplikovali speciální neurální vykreslovací techniku, která vygeneruje konečný výstupní obrázek.

5.3 Trénování sítě

Metoda vyžaduje trénování ve dvou fázích, nejprve se trénuje obecná a dále pak specifická část.

V první fázi se trénuje síť Audio2ExpressonNet pro všechny sekvence trénovacího datasetu, toto trénování je s učitelem. Z hlediska vizuální informace je znám 3D model obličeje konkrétní osoby pro každý frame. Během trénování

je vytvořen 3D model se zvukovým vstupem optimalizováním parametrů sítě a mapováním z prostoru zvukových výrazů do 3D prostoru.

Ve druhé fázi je natrénována vykreslovací síť pro specifickou cílovou sekvenci.

6 Neural Voice Puppetry a Adversarially Disentangled Audio-Visual Representation

Další metodou audiovizuální syntézy je metoda popsaná v článku "Neural Voice Puppetry a Adversarially Disentangled Audio-Visual Representation"[34]. (Disentangle = oddělit, rozmotat věci, které se do sebe zapletly). Tento postup spojuje audio či video složkou řízený pohyb s modelem (libovolné jiné) konkrétní tváře během řeči. Výsledkem je mluvící tvář. Tato naučená audiovizuální reprezentace je i mimořádně užitečná pro úlohu automatického odezírání ze rtů.

6.1 Představení sítě

Porozumění obsahu projevu osoby z pohybu a výrazu její tváře a rtů má velkou váhu pro komunikaci člověk-stroj. Úloha mluvící tváře dle popisu metody dá rozdělit na tří podúlohy:

1. Síť se naučí společnou audiovizuální reprezentaci pomocí associative-and-adversarial tréningu 3.10.

2. Zároveň se užitím associative-and-adversarial tréningu vizuální reprezentace rozdělí na informaci o identitě osoby a na informaci o řeči.

3. Kombinací složek o identitě osoby, video složky a audio složky může mluvící tvář nabýt nové identity. Jako vstup je použito video nebo audio v end-to-end frameworku, který syntetizuje kvalitní časově synchronní mluvící tvář.

6.2 Architektura sítě

Na obr. 14 je zobrazena architektura sítě. Disentangled Audio-Visual System nadále DAVS je end-to-end trénovatelná síť pro generování mluvicí tváře. Pro učení v prostoru identity osoby (Person-ID) a prostoru slov (Word-ID) jsou použity 3 sítě:

1. Video to Word-ID

   Kodér ${E_w}^v$ se naučí vložit reprezentaci videoframu s^v do vizuální reprezentace ${f_w}^v$, která obsahuje pouze řečové informace (neobsahuje tedy informace o identitě osoby, ale jen informaci o vizuální složce řeči).

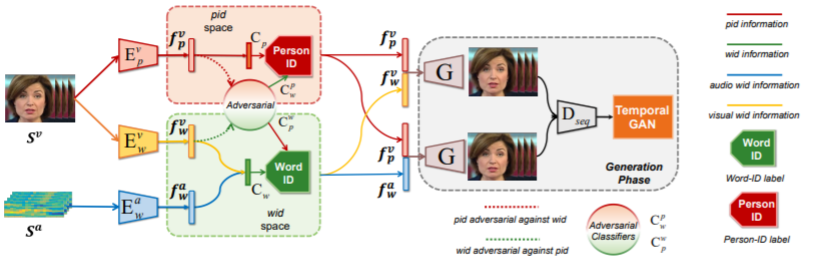


Figure 3: Illustration of our framework. E_p^v is the encoder that encodes Person-ID information from **visual** source to the *pid* space, E_w^v and E_w^a are the Word-ID encoders that extract speech content information to *wid* space from **video** and **audio**. Decoder G takes any combination of features in *pid* and *wid* space to generate faces. D_{seq} is a discriminator used for GAN loss. The adversarial training part contains two extra classifiers C_p^w and C_w^p. The details of embedding the *wid* space and adversarial training are shown in Fig 4 and 5.

Obrázek 14: DAVS (obr. převzat z [34])

2. Audio to Word-ID

   Kodér ${E_w}^a$ se naučí vložit řeč s^a do zvukové reprezentace ${f_w}^a$, která však je ve stejném prostoru jako ${f_w}^v$ výše.

3. Video to Person-ID

   Kodér ${E_p}^v$ se naučí vkládat video obraz s^v do reprezentace ${f_p}^v$, obsahující pouze informace o dané osobě.

6.3 Natrénování sítě

Trénování lze rozdělit do tří fází:

1. Učení společné audiovizuální representace:

   Zde dochází k natrénování sítě reprezentující vstupní video i audio ve společném audiovizuálním prostoru (dochází zde ke spojení reprezentace z obou modalit vizuální a zvukové).

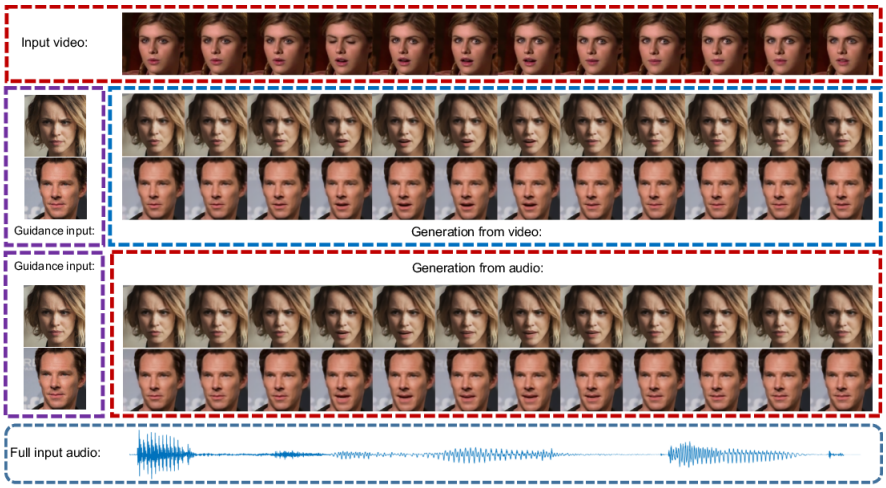
2. Adversarial trénování za účelem oddělení informace o identitě osoby od řečového obsahu:

   V této části pomocí adversariálního tréninku dochází k oddělení (rozmotání) informace o identitě osoby a informace o řeči.

3. Generace tváře libovolné osoby:

   V této části se vygeneruje libovolná mluvící tvář pomocí "rozmotaných"reprezentací naučených výše.

Výsledek generování umělé tváře ze zvukového/vizuálního signálu lze vidět na obr. 15



Obrázek 15: Talking Face Generování (obr. převzat z [34])

7 Live Speech Portraits (LSP)

Metoda LSP [Yuanxun Lu, Jinxiang Chai, Xun Cao: Live Speech Portraits: Real-Time Photorealistic Talking-Head Animation, SIGGRAPH Asia 2021, arXiv:2109.10595] [19] je použitelná i pro aplikace pracující v reálném čase generující personalizovanou fotorealistickou animaci vizuální mluvící hlavy čistě z audio vstupu (za předpokladu předem vytvořeného modelu vizuální složky hlavy z předložených video nahrávek hlavy reálné osoby). Tato metoda ve srovnání s ostatními dosahuje i velmi dobrých výsledků.

Nevýhoda metody je, že se jedná o poměrně komplexní systém vyžadující tvorbu relativně většího počtu NN modelů, a potřebuje označená vizuální data ve 3D prostoru pro robustní trénování.

Tuto metodu jsem zvolil jako vhodnou k implementaci a vyzkoušení a je proto podrobněji popsána v následující části.

8 Zvolený přístup generování vizuální syntézy

Po prostudování literatury, jsem zvolil přístup založený na metodě LSP, protože poskytuje oproti ostatním mnohé výhody, jako například fotorealistickou syntézu, rychlost a schopnost generování vizuální složky hlavy jen z řečového signálu, tj. bez nutnosti reprezentace vstupu textem.

Přístup k vizuální syntéze může být založen na modelech, jejichž vstup je buď text nebo jen odpovídající řečový signál.

Výhoda modelu, jehož vstupem je audio, je, že v případě, že při generaci (inferenci) máme k dispozici text a ne audio, můžeme použít TTS (syntéza

řeči z textu) a vygenerovat wav, který je následně zpracován audiovizuální metodou.

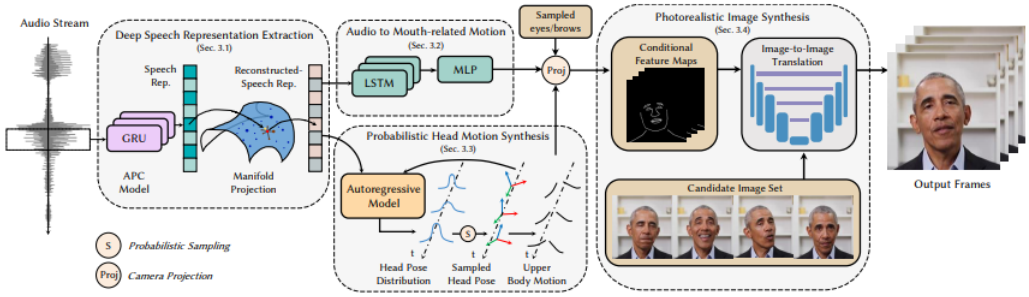
V případě modelu, jehož vstupem je text, je v případě, že máme pro syntézu k dispozici pouze audio, nutno použít systém rozpoznávání mluvené řeči, který bývá výpočetně náročnější a méně přesný než TTS.

Pro aplikace v reálném čase konvertující hlas a tvář na mluvícího avatara, je výhodnější použít model, jehož vstupem je hlas.

Metoda LSP je velmi komplexní a její implementace je časově náročná, přesto jsem ji zvolil, protože je v plánu na jejím základě vyvinout nový model audiovizuální syntézy v následujících letech. V této bakalářské práci bylo proto cílem implementace LSP dokázat generovat libovolnou českou promluvu předdefinovaným vizuálním modelem. Vlastní vizuální modely pak budou předmětem práce následujících období.

8.1 Detailní popis metody Live Speech Portraits

8.1.1 Základní přehled metody



Obrázek 16: LSP overview (obr. převzat z [19])

Na obr. 16 je znázorněn model metody LSP. Model je rozdělen na 3 hlavní bloky:

Zpracování audio vstupu a reprezentace řečového signálu. Vstupem audio složky je zvuk ve formátu wav o frekvenci 16 kHz, který je předzpracován, zparametrizován a přiveden na vstup hluboké neuronové sítě (Deep Neural Networkm, DNN), která vytváří latentní reprezentaci vstupního řečového signálu. Výstupem tohoto bloku je generalizovaný vektor $\hat{\boldsymbol{h}}$ reprezentující řeč v daném časovém okamžiku.

Modelování pohybu úst a hlavy. Vstupem této části modelu je vektor $\hat{\boldsymbol{h}}$ a výstupem je vektor $\Delta \boldsymbol{v}_{m,t}$ určující 3D rozmístění určených bodů kontury úst a vektor $\boldsymbol{P}_t$ určující pozici celé hlavy. Pozice hlavy je pouze částečně závislá na vstupním audiosignálu a na předcházejících pozicích hlavy a tato částečná závislost je modelována pravděpodobnostním modelem. Model

hlavy je rigidní a pohyb hlavy je dán jeho rotací a translací. Poté se tyto vektory promítnou do formy skici hlavy (feature map), připojí se k nim náhodný pohyb očí a obočí, čímž získáme skicu pohybu daných bodů hlavy a úst. Pohyby horní části těla jsou odvozeny z pozice hlavy. Metoda LSP tedy umožňuje kontrolu pozice hlavy a těla.

Projekce obrazu reálné tváře na vytvořený model. V této části se spojí vytvořená skica s (foto)obrazy cílové reálné hlavy a vytvoří se fotorealistický obraz hlavy.

8.1.2 Zpracování zvukového vstupu

Ze vstupního zvukového souboru se vytvoří spektrogram.

8.1.3 Vytvoření spektrogramu

Zvuk je vzorkován 16 kHz a každou 1/120 sec je zpracováno jeho jedno časové okénko délky 1/60 sec do 80-ti Melových spektrálních koeficientů. Okénka se tedy překrývají polovinou své délky. Použití 512-ti bodové krátkodobé Fourierovy transformace vyžaduje nejspíše doplnění vstupu této transformace nulami (vychází přibližně 267 vzorků na okénko nebo použité okénko má větší délku, než je v popisu metody reportováno).

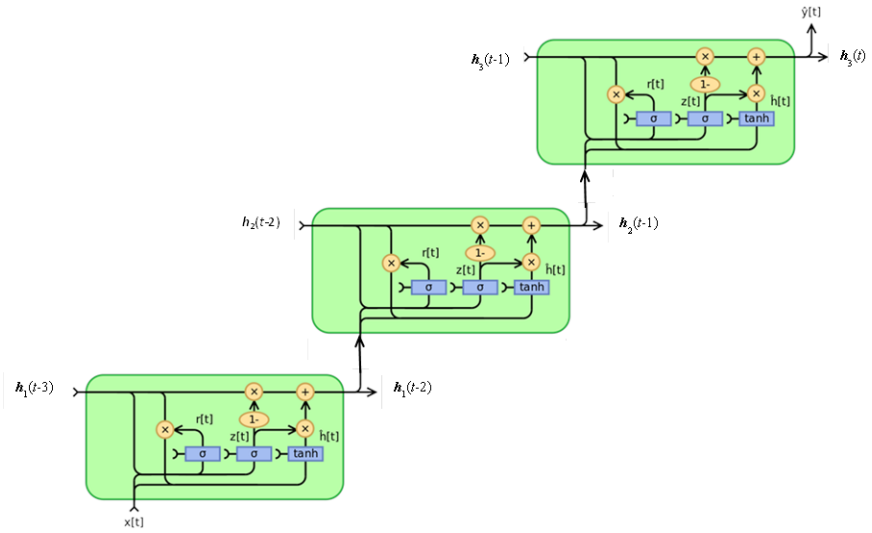
8.1.4 Reprezentace řečového signálu

Dobrá reprezentace vstupního řečového signálu hraje klíčovou roli, protože se od ní odvíjí celé výsledné video. Základní myšlenka reprezentace řečového signálu spočívá v reprezentaci řečových jednotek (např. fónů, fonémů, jejich stavů nebo jiných jednotek) příznakovými vektory v latentním prostoru. Transformace ze vstupního prostoru příznaků (tj. log Mel spektrogramu) do latentního prostoru je obecně vysoce nelineární a je realizována průchodem vstupního příznakového vektoru vrstvami hlubokých neuronových sítí. LSP používá tři vrstvy jednosměrného modelu GRU (Gated Recurrent Unit) viz obr. 17.

$$\boldsymbol{h}_l = GRU^{(l)}(\boldsymbol{h}_{l-1}), \forall l \in [1, L], \tag{14}$$

kde $L = 3$ a $\boldsymbol{h}_l \in \mathbb{R}^{512}$ jsou skryté stavy každé vrstvy GRU. Hodnoty poslední skryté vrstvy GRU v čase t jsou pak brány jako vektor reprezentují řeč v tomto časovém kroku t. Délka časového kroku odpovídá jednomu framu, kterých je 120/sec.

Obecně je při hledání vhodných reprezentantů tříd (shluků) výhodné použít metody učení bez učitele. Pro tento případ LSP pro zpracování řeči používá model autoregresivního prediktivního kódování (Autoregressive Predictive Coding, APC model) [6]. APC model se snaží na základě minulých hodnot 80-ti dimenzionálních vektorů log Mel spekter určit následných n (80-ti dimenzionálních) vektorů log Mel spektra. Trénování je tedy zcela samoučící se. Označíme-li posloupnost 80-ti dimenzionálních vektorů log Mel spektrogramů zápisem



Obrázek 17: 3 vrstvy GRU, $\boldsymbol{x}(t)$ je vektor log Mel spektra (obr. převzat z https://commons.wikimedia.org/w/index.php?curid=66225938))

$(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ a zápisem $(\boldsymbol{y}_{1+n}, \boldsymbol{y}_{2+n}, ..., \boldsymbol{y}_{T+n})$ odhadovanou posloupnost log Mel spektrogramů metodou APC, je ztrátová funkce $(L1)$ určena výrazem

$$\sum_{t=1}^{T-n} |\boldsymbol{x}_{i+n} - \boldsymbol{y}_i|, \tag{15}$$

kde $n = 3$ dle [Chung and Glass 2020][6].

Při trénování sítě GRU je oproti obrázku navíc přidána lineární vrstva na výstup poslední vrstvy, která předpovídá budoucí log Mel spektrogram. Tato vrstva se v inferenčním běhu nepoužije.

8.1.5 Locally-linear embedding

Jelikož každý řečník vlastní sobě specifický řečnický styl, musí se řečová reprezentace (zde představená vektorem $\boldsymbol{h}$) generalizovat. Bez této generalizace by se s přímým použitím získané řečové reprezentace dosahovalo slabých výsledků v případech, kdy by řečník vlastnil výrazně odlišný řečnický styl oproti cílové osobě (osobě, jejíž hlava bude použita pro získání obrazu).

Zde přichází na řadu projekce pomocí variety. Tato operace je inspirována syntézou tváře ze skic [5]. Je zde použito lokální lineární vkládání (anglicky Locally-linear embedding, LLE), vychází se z předpokladu, že každý datový bod a jeho sousedi leží v lokálně lineární části prostoru globálně nelineárního mnohodimenzionálního manifoldu [26]. K dosud spočítanému vektoru řečové reprezentace $\boldsymbol{h}$ se spočítá jeho LLE rekonstruovaná reprezentace $\hat{\boldsymbol{h}}$ v každé dimenzi.

Nejprve se pomocí Eukleidovské vzdálenosti nalezne K nejbližších vektorů - sousedů vektoru $\boldsymbol{h}$. Potom se hledá taková lineární kombinace K nalezených sousedů, aby co nejlépe rekonstruovala $\boldsymbol{h}$. Pro nalezení vah lineární kombinace w_k použijeme vztah pro výpočet barycentrických souřadnic vektoru $\boldsymbol{h}$ na základě jeho sousedů (K sousedních vektorů):

$$min||\boldsymbol{h} - \sum_{k=1}^{K} w_k \cdot \boldsymbol{f}_k||_2^2, \qquad \sum_{k=1}^{K} w_k = 1, \tag{16}$$

kde w_k je barycentrická váha $k-$nejbližšího souseda $\boldsymbol{f}_k$, který je spočten metodou nejmenších čtverců. Optimální hodnotu K autoři LSP určili experimentálně jako 10. Výsledkem je získaná generalizovaná řečová reprezentace $\hat{\boldsymbol{h}}$:

$$\hat{\boldsymbol{h}} = \sum_{k=1}^{K} w_k \cdot \boldsymbol{f}_k \tag{17}$$

Tato generalizovaná reprezentace řeči $\hat{\boldsymbol{h}}$ je pak předána dále jako vstup dalších modulů.

8.1.6 Syntéza pohybu úst

Metody predikce pohybu úst na základě informace ze zvukového řečového signálu byly v poslední době dost zkoumány. Opět je hlavní pozornost upřena k metodám hlubokého učení např. [35] či [29].

Metoda LSP pracuje s 25 body ležícími na kontuře rtů, pozice těchto bodů je trackována ve 3D prostoru, tzn., že výsledná dimenze vektoru popisujícího pohyb úst (spíše ale rtů) je $\mathbb{R}^{25x3}$. Úlohou je k vektoru generalizované reprezentace části řečového signálu $\hat{\boldsymbol{h}}_t$ v čase t přiřadit informaci dobře popisující pozici úst (kontury rtů) odpovídající skutečné poloze úst v čase t. LSP řeší tuto úlohu pomocí neuronové sítě architektury Long-Short-Term Memory (LSTM), jejíž výstupem je "mezivýsledný"vektor $\boldsymbol{m}_t$ relativně vyšší dimenze velikosti 256. přesněji je tato část realizována třemi (stack) LSTM vrstvami, každá z nich o velikosti 256 skrytých stavů.

Aby se získal popis pozice bodů kontury rtů pomocí vektoru popisujícího pohyb této kontury, je vektor $\boldsymbol{m}_t$ následně zpracován ještě další sítí (MultiLayer Perceptron) MLP, která zobrazuje $\boldsymbol{m}_t$ na výsledný 75-ti dimenzionální vektor $\Delta v_{m,t}$. Vektor $\Delta v_{m,t}$ obsahuje informace o relativní poloze kontury rtů vzhledem ke střední hodnotě poloh.

Dále bylo zjištěno, že metoda dosahuje výrazně lepších výsledků, pokud se přidá pravý časový kontext. Přidání pravého kontextu vede ke zpoždění o d framů. Toto zpoždění ale umožňuje síti přesněji odhadnout popis pohybu rtů a podle toho upravit svůj výstup. Proces zpracování je názorně popsán následujícími rovnicemi:

$$\boldsymbol{m}_0, \boldsymbol{m}_1, ..., \boldsymbol{m}_t = LSTM(\hat{\boldsymbol{h}}_0, \hat{\boldsymbol{h}}_1, ..., \hat{\boldsymbol{h}}_{t+d}), \tag{18}$$

$$\Delta \boldsymbol{v}_{m,t} = MLP(\boldsymbol{m}_t) \tag{19}$$

Parametr zpoždění d byl experimentálně nastaven na hodnotu 18. To při frekvenci 60 FPS znamená zpoždění 300 ms, které se zdá akceptovatelné i při on-line provozu.

Pro naučení správného zobrazení z generalizované reprezentace řeči na pohyb úst se minimalizuje ztrátová funkce $L2$ mezi odhadovaným pohybem úst $\Delta\hat{\boldsymbol{v}}_{m,t}$ a informací od učitele $\Delta\boldsymbol{v}_{m,t}$. Ztrátová funkce se dá zapsat následovně:

$$\sum_{t=1}^{T}\sum_{i=1}^{N} ||\Delta \boldsymbol{v}_{m,t,i} - \Delta\hat{\boldsymbol{v}}_{m,t,i}||_2^2, \tag{20}$$

kde $\boldsymbol{v}_{m,t,i}$ je $\boldsymbol{v}_{m,t}$ jen pro i-tý bod z celkového počtu $N = 25$ předem definovaných 3D bodů na kontuře rtů a $T = 240$ je počet za sebou jdoucích framů v jedné iteraci.

8.1.7 Syntéza pohybu hlavy

Pohyb hlavy se syntetizuje pravděpodobnostním modelem. Pro více realistickou vizuální syntézu přispívá i pohyb hlavy a ramen (viz další sekce). Pro lidi je totiž normální při mluveném projevu hýbat hlavou a tělem. Tyto pohyby dodávají projevu emoce a mohou ho zdůraznit. Pro dobrý subjektivní dojem je třeba, aby pohyby byly do jisté míry náhodné, avšak aby splňovaly určitá omezení, např. neškubat hlavou ze strany na stranu.

Odhadování pohybu hlavy a ramen ze zvukového signálu není triviální úlohou, neboť mezi nimi je malá spojitost. Model funguje na základě 2 předpokladů:

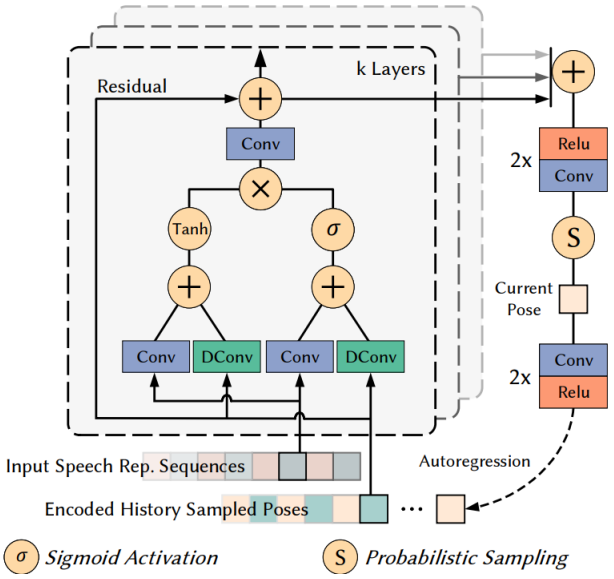
Předpoklad I Pohyb hlavy částečně závisí na mluveném projevu, zejména na výrazu a intonaci. Např. lidé přikyvují při souhlasu nebo zdvihají hlavu při zvyšování intonace.

Předpoklad II Současná pozice hlavy závisí na minulých pozicích. Např. je velmi pravděpodobné, že lidé po výrazném naklonění hlavy ji naklání zpět.

Pro splnění těchto předpokladů by navrhovaná architektura měla mít schopnost vidět minulé stavy polohy hlavy a současný vstup audio signálu (generalizované reprezentace řeči $\hat{\boldsymbol{h}}_t$). Pro zajištění jisté míry náhodnosti pohybů je u LSP použit pravděpodobnostní model syntézy pohybu, který předčí výsledky deterministického modelování [12].

Sdružená hustota pravděpodobnosti pohybu hlavy $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$ podmíněná řečovou reprezentací $\hat{\boldsymbol{h}} = (\hat{\boldsymbol{h}_1}, \hat{\boldsymbol{h}}_2, ..., \hat{\boldsymbol{h}_T})$ může být zapsána jako:

$$p(\boldsymbol{x}|\hat{\boldsymbol{h}}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{t-1}, \hat{\boldsymbol{h}}_t). \tag{21}$$



Obrázek 18: Struktura pravděpodobnostní sítě pro odhad pozice hlavy (obr. převzat z [19])

$\boldsymbol{x}$ je tedy posloupnost vektorů popisující pohyb hlavy a $\hat{\boldsymbol{h}}$ je posloupnost vektorů generalizované reprezentace řeči. Model využívá multidimenzionálního Gaussova rozdělení. Architektura sítě byla inspirována úspěchem v podmíněném pravděpodobnostním generativním modelování [31] a je ilustrována na obr. 18.

Model se skládá ze dvou za sebou jdoucích (stack) reziduálních bloků, každý o sedmi vrstvách. Pro modelování časově dlouhých závislostí pohybu hlavy jak na minulém audiosignálu tak i na její předchozí poloze, používají oba bloky časově rozšířené (dilated) konvoluční sítě. Rychlost dilatace je dvojnásobná s každou další vrstvou, tj. nabývá hodnot 2, 4, 8, 16, 32, 64 a je provedena v každém bloku. Výstup ze sítě tedy zachycuje historii 255 framů, což při 60 FPS odpovídá 4,25 sec. Tato doba se jeví jako přijatelná pro rozhodnutí pozice hlavy v daném čase v závislosti na její minulé poloze a minulého řečového projevu. Výstupy každé vrstvy jsou u každého bloku sečteny a součet zpracován postprocessingem, který má formu 2 za sebou spojených (stack) RELU konvolučních sítí - viz obr. 18. Výstup tohoto postprocessingu pak udává odhad parametrů normálního rozdělení, ze kterého je následně provedeno vzorkování pro obdržení výsledné generované pozice hlavy $\boldsymbol{p}$. Použité normální rozdělení autory LSP je nejspíše dimenze 6 (3 rotace a 3 translace ve 3D) s diagonální kovarianční maticí, což je ekvivalentní 6-ti jednorozměrným normálním rozdělením. Po provedení odhadu pozice hlavy vzorkováním z normálního rozdělení, je vektor aktuální pozice hlavy $\boldsymbol{p}$ přiveden po transformaci konvoluční sítí s RELU zpět na vstup celé sítě jako informace o poslední pozici hlavy pro její odhad v následujícím

časovém kroku. Celkově tedy ve shodě s článkem [19]:

$$\boldsymbol{p}_{para,t} = \boldsymbol{\Phi}(\boldsymbol{p}_{t-F}, \boldsymbol{p}_{t-F+1}, ..., \boldsymbol{p}_{t-1}, \hat{\boldsymbol{h}}_t), \tag{22}$$

$$\Delta\boldsymbol{p}_{m,t} = Sample(\boldsymbol{p}_{para,t}) \tag{23}$$

Během trénování je třeba nastavit váhy sítě generující pozici hlavy tak, aby poskytovala dobrý odhad parametrů normálního rozdělení popisujícího aktuální polohu hlavy. Ztrátová funkce je tvaru:

$$-\ln(\mathcal{N}(\hat{\boldsymbol{p}}_t, \hat{\boldsymbol{h}}_t|\hat{\boldsymbol{\mu}}), \hat{\boldsymbol{\sigma}}), \tag{24}$$

kde $\hat{\boldsymbol{p}}_t$ a $\hat{\boldsymbol{h}}_t$ jsou po řadě pozice hlavy a generalizovaná řečová reprezentace v čase t. Minimalizace ztrátové funkce nutí model ke správnému odhadu parametrů $\hat{\boldsymbol{\mu}}$ a $\hat{\boldsymbol{\sigma}}$. Přitom $\hat{\boldsymbol{p}}_t$ je vektor obsahující nejen 6-ti dimenzionální vektor polohy hlavy $\boldsymbol{p}_t$ (3 rotace a 3 translace ve 3D), ale i člen rychlosti uvažovaný jako změna polohy $\Delta\boldsymbol{p}_t$. Toto rozšíření vektoru polohy přináší zlepšení v plynulosti pohybu hlavy.

8.1.8 Pravděpodobnostní syntéza pohybu horní části těla

LSP využívá jednoduché závislosti pohybu horní poloviny těla na pohybu hlavy. Obrázek horní poloviny těla je realizován statickým obrazem horní poloviny těla, který se pohybuje v závislosti na posunu hlavy a to vždy o jednu polovinu posunu hlavy v každém směru.

8.1.9 Fotorealistická syntéza obrazu

Poslední fází přístupu LSP je vygenerování fotorealistických obrazů obličeje z předchozích predikcí. Vykreslovací síť je inspirována nedávným pokrokem v syntéze fotorealistických a ovladatelných obličejových videí [7][16][17][30]. LSP používá k vygenerování fotorealistické syntézy přístupu trénování protivníků (adversarial training) a neuronové sítě architektury U-Net. Výsledkem práce modelů popsaných v předchozích sekcích je pozice rtů a úst, hlavy a horní poloviny těla v každém časovém kroku t. Z pozic těchto částí těla ve 3D je dále vykreslena skica tváře (zahrnující obrys tváře, kresbu kontury rtů, očí, obočí) s obrysem ramen (horní poloviny těla) a to projekcí do 2D. Na tuto skicu se následně vyrenderuje obraz tváře cílové osoby. Obrazy použité pro tuto úlohu jsou automaticky vybírány z množiny kandidátů-obrazů, kterou může tvořit např. video obsahující nahrávku hlavy cílové osoby o délce řádově několika minut. Z množiny obrázků-kandidátů jsou vybrány 4 obrázky. První dva jsou vybrány jako obrázek stý v pořadí řazených podle nejmenší a největší velikosti oblasti úst, zbylé dva se pak vybírají jako dva nejbližší obrázky z množiny vzniklé rovnoměrným navzorkováním videa dle rotace podle osy x a podle osy y. Použitá neuronová síť je 8-vrstvá podobná U-Net [8][10][25]. Jedná se o konvoluční neuronovou síť s přeskokem

spojení (skip connection) v každé vrstvě rozlišení. Rozlišení jednotlivých vrstev je: 2562, 1282, 642, 322, 162, 82, 42, 22 a odpovídající počty kanálů jsou: 64, 128, 256, 512, 512, 512, 512, 512). Každá vrstva kodéru se skládá z jednoho konvolučního (se stride 2) a jednoho residuálního bloku. Vrstvy symetrického dekodéru jsou téměř stejné kromě toho, že první konvoluční blok je nahrazen upsamplingem s faktorem 2.

Image-to-Image Traslation síť je trénována "adversariálně", tj. snaží se generovat obrázky co nejvíce realistické tak, aby oklamala diskriminátor D, zatímco diskriminátor D je trénován, aby odhaloval generované obrázky od skutečných. Použité ztrátové funkce pro nastavení diskriminátoru $\mathcal{L}_{GAN}(D)$ a generátoru $\mathcal{L}_{GAN}(G)$ jsou LSGAN (Least Square GAN) [20]:

$$\mathcal{L}_{GAN}(D) = (\hat{r} - 1)^2 + r^2, \tag{25}$$

$$\mathcal{L}_{GAN}(G) = (r - 1)^2, \tag{26}$$

kde $\hat{r} = D(\boldsymbol{x})$ je výstup diskriminátoru D pro reálný obraz $\boldsymbol{x}$ a $r = D(G(\boldsymbol{z}))$ je výstup diskriminátoru pro obraz $G(\boldsymbol{z})$ syntetizovaný generátorem G. Ztrátová funkce generátoru $\mathcal{L}_{GAN}(G)$ je doplněná o ztrátovou funkci barvy $\mathcal{L}_C$, ztrátovou percepční funkci $\mathcal{L}_P$ a ztrátovou funkci feature matching loss $\mathcal{L}_{FM}$ (pro urychlení a větší stabilitu trénovacího procesu):

$$\mathcal{L}_G = \mathcal{L}_{GAN}(G) + \lambda_C \mathcal{L}_C + \lambda_P \mathcal{L}_P + \lambda_{FM} \mathcal{L}_{FM}, \tag{27}$$

kde $\lambda_C = 100$, $\lambda_P = 10$, $\lambda_{FM} = 1$ a

$$\mathcal{L}_C = ||y - \hat{y}||_1, \tag{28}$$

kde y je obraz syntetizovaný generátorem a $\hat{y}$ je skutečný obraz, přičemž ztrátová funkce počítá s každým pixelem (per pixel loss),

$$\mathcal{L}_P = \sum_{i \in \mathcal{S}} ||\Phi^{(i)}(y) - \Phi^{(i)}(\hat{y})||_1, \tag{29}$$

kde ztrátová funkce $\mathcal{L}_P$ je aplikovaná při trénování sítě VGG19 [28],
$\mathcal{S} = \{1, 6, 11, 20, 29\}$ je množina indexů vrstev sítě a $\Phi^{(i)}$ je i-tá vrstva sítě a

$$\mathcal{L}_{FM} = \sum_{i=1}^{L} ||r - \hat{r}||_1, \tag{30}$$

kde L je počet vrstev diskriminátoru D.

8.2 Analýza implementovatelnosti LSP

Pro řešení úlohy audiovizuální syntézy metodou LSP je potřeba řada jednotlivých různých architektur neuronových sítí. V tomto ohledu se LSP jeví jako poměrně komplikovaná a časově náročná na vlastní implementaci. Na druhou

stranu variabilita použitých modelů umožňuje specifické nastavení každého jednotlivého modelu pro úspěšnější zvládnutí celé úlohy. Jednotlivé modely mají různé nároky na své trénování. Např. reprezentace řečového signálu modelem APC je dostatečně univerzální, tj. model APC je nezávislý na konkrétním hlasu a je obecný a společný pro všechny řečníky. Reprezentace řečového signálu se tedy nemusí modelovat pro každého řečníka zvlášť. Toto však neplatí pro modely vizuální složky, které jsou specifické pro daného mluvčího, jsou závislé na řečníkovi a vyžadují tvorbu modelu pro každou jednotlivou osobu. K tvorbě modelu vizuální složky jsou vyžadována trénovací data (cca 5 minut audiovizuálního záznamu) od konkrétní osoby, jejíž vizuální složku má model generovat. Tato trénovací data se musí předzpracovat doplněním audiovizuálního záznamu o informaci od učitele, a to o poloze předem vybraných bodů tváře, zejména pak bodů ležících na kontuře rtů. Tyto body je potřeba sledovat (trackovat) po celou dobu záznamu trénovacího videa a to ve 3D. Pro zvládnutí této úlohy je potřeba 3D tracker. V době práce jsme na našem pracovišti měli k dispozici jen systém pro trackování bodů v prostoru 2D (konkrétně systém OpenPose). Na základě výše popsané analýzy byl navržen následující postup. Pokusit se zprovoznit část LSP metody reprezentace řečového signálu. Pokud budou k dispozici hotové modely vizuální složky určitých řečníků, použít je. Výsledkem by tedy měla být audiovizuální hlava jedné z předem daných osob, jejichž vizuální složku nelze měnit, ale jejichž model audiovizuální hlavy může pronášet libovolný řečový obsah. Dále vytvořit databázi nahrávek konkrétní osoby pro odzkoušení trackování bodů tváře systémem OpenPose (tj. ve 2D). Poté odzkoušet úspěšnost trackeru pro úlohu generování skici tváře ve 2D. V budoucí práci (v dalších letech) analyzovat možnosti dostupnosti (popř. vývoje) 3D trackeru a v případě pozitivní odpovědi se zabývat tvorbou 3D modelů. V případě, že 3D tracker nebude k dispozici nebo jeho přesnost nebude dostatečná, analyzovat možnosti 2D modelů.

9 Realizace audiovizuální syntézy

Součástí zadání této práce bylo vytvoření systému audiovizuální syntézy řeči pomocí dostupných metod. Jako jedno řešení jsem použil metodu LSP, dále jsem navrhl experiment syntézy skici tváře z videa. V obou případech potřebujeme pro jejich realizaci referenční data o pohybu klíčových bodů tváře, které musíme nějak získat. Pro získání dat se využívají dostupné trackery snímající pozici klíčových bodů tváře a to i během jejich pohybu v době promluvy.

9.1 Použité keypoints tváře

Pro detekci pohybu člověka, ať už těla, ruky, či obličeje, se může proces detekce zjednodušit na detekci pohybu několika bodů charakterizujících pohyb celého sledovaného objektu. Těmto bodům se říká klíčové (keypoints).

V případě sledování obličeje máme několik oblastí, které se při mluvení pohybují do jisté míry nezávisle na ostatních (nejsou s nimi zcela rigidně svá-

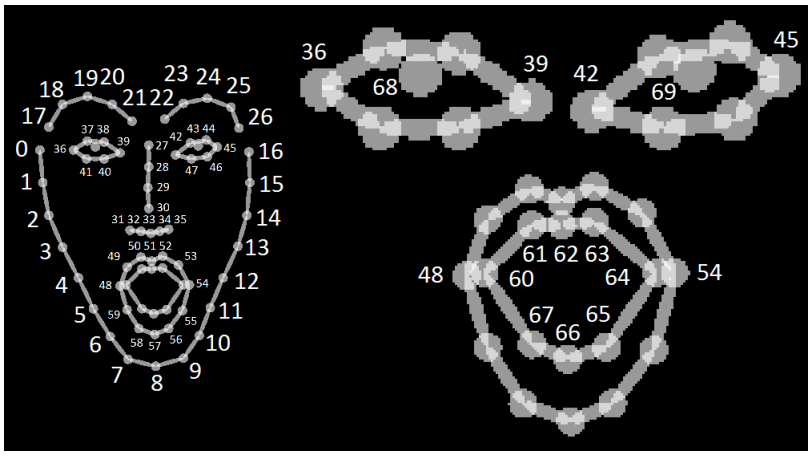
zány), proto je třeba určit klíčové body pro všechny tyto oblasti, tj. oči, ústa, tvář, nos. Pro audiovizuální syntézu řeči nás zajímá převážně pohyb úst, dále nás zajímá také pohyb obrysu tváře. Sledování pohybu očí je sice také vhodné, nicméně jelikož prakticky lze předpokládat téměř nulovou závislost mezi pohybem úst a očí při promluvě, není třeba přikládat sledování pohybu očí v našem experimentu tak velkou váhu. Pro naše účely není třeba sledovat natočení oka a jeho mrkání. Známá poloha očí však může pomoci zpřesnit určení pozici tváře. Nos se při promluvě většinou hýbe minimálně.

Je výhodné použít standardizovanou sadu klíčových bodů. Výhodou výběru standardizované sady je především možnost porovnání výsledků mezi výzkumnými týmy napříč světem. Další výhodou je použití dostupných trackerů, které jsou vyvíjeny s ohledem na tyto sady.

Nevýhodou může být nepřesné sledování těchto bodů pro specifické obličeje.

9.1.1 OpenPose

Pro řešení úlohy trackování jsem použil systém OpenPose. OpenPose je systém pro detekci klíčových bodů těla, ruky, obličeje a nohou. Pro naše účely jsem použil sadu klíčových bodů, kterou používá systém OpenPose pro detekci obličeje. Tato sada se skládá ze 70-ti bodů sledujících obrys obličeje, tvar obočí, polohu nosu, polohu očí a polohu úst (okraje rtů). Poloha těchto bodů na tváři je zobrazena na obr. 19.



Obrázek 19: Využívané keypoints na tváři v programu OpenPose (obr. převzat z OpenPose dokumentace(https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_02_output.html))

9.2 Realizace audiovizuální syntézy metodou LSP s předdefinovaným vizuálním modelem

Pro realizaci audiovizuální syntézy metodou LSP se mi podařilo vyhledat a zprovoznit kód demo verze poskytnuté autory článku [Live Speech Portraits: Real-Time Photorealistic Talking-Head Animation][19].

Tento kód je napsaný v jazyce Python a využívá mnoho balíčků. Zprovoznění demo kódu nebylo úplně jednoduché, jelikož z důvodu verzování nefungovalo několik metod, jejichž kód jsem musel mírně upravit, dále se musely najít kompatibilní verze balíčků a doinstalovat je do vybraného prostředí. Dostupný kód neobsahoval žádný kód pro trénování modelů. Ale i v případě jeho dostupnosti není možno připravit žádný model vizuální složky hlavy bez použití (relativně přesného) 3D trackeru, protože při trénování modelů je potřeba dodat informaci o 3D poloze klíčových bodů tváře po celou dobu trénovací nahrávky. Ke kódu demoverze se ale daly stáhnout hotové modely vizuální složky hlavy, které jsem použil pro zprovoznění syntézy.

Výsledná aplikace má na vstupu audiosoubor ve formátu wav a na výstupu sekvenci obrázků. Pro vytvoření výsledného videa jsem použil nástroj FFmpeg, který skládá obrázky a přikládá k nim zvuk, čímž vytvoří video.

Aplikaci jsem odladil v prostředí Visual Studio Code, ve kterém ji lze také spustit (nejprve je třeba aktivovat vybrané Python prostředí a poté je třeba spustit Python skript demo.py z adresáře, ve kterém je uložen. V tomtéž adresáři se nachází i příslušné složky obsahující natrénované modely hlavy a vstupní data (audio ve formátu wav). V terminálu se skript spouští příkazem:

```
python demo.py --id May --driving_audio ./data/Input/XXX.wav
--device cuda
```

kde parametr `driving_audio` rozhoduje o tváři řečníka (na výběr je: Obama, May, Nadella, McStay) a `XXX.wav` je jméno vstupního audio souboru. Vstupní audio je tedy třeba nahrát do adresáře `data/Input` ve formátu wav, audio obsah souboru může být libovolný, pro účely demonstrace jsem nahrál několik záznamů svého i jiných hlasů a vyzkoušel jejich audiovizuální syntézu.

Syntéza pomocí LSP funguje podle očekávání a všechny čtyři předdefinované vizuální modely mohou mluvit hlasem člověka vstupního audiosouboru.

9.3 Syntéza skici tváře z videa

Jako jednoduchou verzi syntézy jsem navrhl vytvoření mluvící skici tváře hovořící osoby založené na vhodném vybrání (ve správném pořadí) snímků (framů videa neboli videoframů) z předem připravené sady (databáze) nahrávek této osoby.

9.3.1 Vytvoření databáze

Cílem této části práce bylo vyzkoušet možnost tvorby videa skici hlavy prostým poskládáním vhodných videoframů z předem připravené databáze videí

mluvící osoby. Vstupem metody je nějaké dosud neviděné video mluvící hlavy. Úkolem je nalézt v databázi videí vhodnou množinu framů a poskládat z nich výsledné syntetizované video tak, aby bylo v určitém smyslu co nejpodobnější videu vstupnímu podobně jako je tomu v konkatenační syntéze. Jedná se jen o proof-of-concept pro získání představy, jaké výsledky může tento přístup přinést, a tomu odpovídá jednoduchost použité metody. Ta ke každému framu vstupního videa automatickým trackováním (OpenPose) nalezne polohu 70-ti standardních klíčových bodů a v databázi videí nalezne frame, jehož klíčové body mají od vstupního framu (snímku) nejmenší vzdálenost vyjádřenou jako sumu vzdáleností všech 70 klíčových bodů framu databáze od odpovídajících bodů vstupního framu. Vzdálenosti mezi sousedními videoframy tvořící syntetizované video nejsou zohledněny, tudíž není potřeba dekodéru, který by hledal posloupnost framů podle kritéria nejmenší kumulativní vzdálenosti počítané jako sumu přes všechny časy součtu dvou vzdáleností: vzdáleností videoframu syntetizovaného od originálního (videoframu vstupního videa) v čase t a vzdálenosti mezi sousedními dvěma syntetizovanými videoframy časů t a $t+1$. (Poznámka: Hodnota poslední vzdálenosti mezi sousedními dvěma syntetizovanými videoframy časů T a $T+1$, kde T je délka posloupnosti framů videa, je definovaná jako 0).

Pro konstrukci databáze videí je nutné získat nejprve vhodné videonahrávky. Ve videu nahrávek je nutné, aby byla vidět dobře tvář při promluvě všech základních hlásek. Nejprve jsem se pokusil najít nějaké dostupné nahrávky na internetu, avšak kvůli zvukovému šumu na pozadí v pořadech, kde je dobře viditelná tvář, nebo kvůli častým střihům během rozhovorů nebo špatnému rozlišení nahrávek se mi nepovedlo najít vhodnou sadu nahrávek jednoho řečníka. Proto jsem se rozhodl vytvořit vlastní sadu nahrávek.

Na univerzitě jsem s pomocí svého vedoucího Ing. Jana Zelinky, Ph.D. a pana Ing. Zdeňka Krňoula, Ph.D. vytvořil několik nahrávek svého obličeje při promluvě-četby několika stran z knihy vybraných tak, aby byly obsahově (co se týče promluvy) pestré, tj. obsahovaly všechny hlásky. K dispozici jsme měli dvě kamery, z čehož jedna snímala stereoobraz a druhá pouze standardní 2D obraz. Kvůli technické poruše stereokamery se bohužel musel použít 2D obraz. Celkově pořízená databáze obsahuje 75 120 snímků s informací o souřadnicích každého ze 70-ti klíčových bodů každého snímku.

Pořízená datová sada nahrávek byla při každém experimentu rozdělena na dvě části. První část pořízené datové sady posloužila jako databáze videí, z nichž se vybírají framy pro syntézu. Druhá část slouží jako vstupní video, jež se má z framů první části syntetizovat.

9.3.2 Zpracování dat

Nyní jsem již disponoval dostatečným množstvím nahrávek, avšak k dalšímu postupu bylo nutné nějak zachytit pohyb tváře, pro tento účel jsem použil systém OpenPose, který detekoval klíčové body tváře z nahrávky a poskytl ke každé nahrávce posloupnost JSON souborů. Video bylo navzorkované na sekvenci obrázků (50 obrázků za sekundu) a každý z nich měl vlastní JSON soubor a v něm uložené jednotlivé souřadnice X a Y všech 70-ti klíčových bodů. Díky tomuto

postupu jsem získal reprezentaci tváře pro každý obrázek, nicméně jelikož souřadnice X a Y byly absolutní, nedala se tato reprezentace použít k porovnávání jednotlivých snímků (framů databáze s framy vstupního videa).

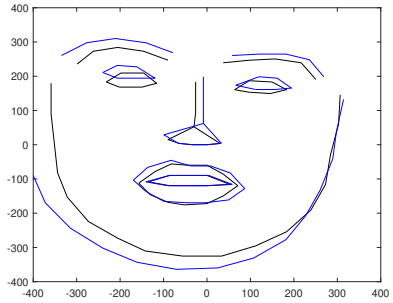
Pro možnost porovnání snímků bylo třeba souřadnice převést na relativní. Rozhodl jsem se, že nejlepším bodem, ke kterému souřadnice vztáhnu, bude nějaký z klíčových bodů, který se ve srovnání s ostatními nejméně pohybuje. Teoreticky by právě takovým bodem mohl být kořen nosu, avšak bylo třeba napsat algoritmus, který nám tento bod vybere. Pro výběr nejstabilnějšího bodu (tj. takového, jehož pohyb vůči všem ostatním bodům v čase je nejmenší) jsem použil program Matlab, ve kterém jsem napsal skript porovnávající vzdálenosti všech bodů od všech ostatních v průběhu času všech nahrávek. Výpočet tohoto algoritmu byl sice náročný, nicméně jej stačilo spočítat jedenkrát. Výsledek potvrdil teorii, že nejstabilnějším místem je nos, konkrétně bod číslo 35.

Poté jsem převedl všechny souřadnice všech bodů každého snímku na relativní souřadnice vzhledem k určenému bodu nosu (č. 35), a to translací (posunem) ve směru osy x i ve směru osy y.

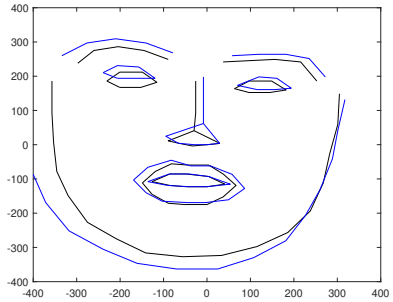
9.3.3 Vytvoření skici

Vykreslením jednotlivých bodů na plátno a jejich následným propojením jsem získal hrubou skicu každého obrázku, nyní již jen stačilo provést experiment.

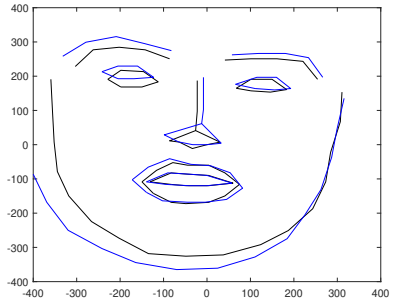
Vstupem byla krátká sekvence framů a výstup stejně dlouhá sekvence jim nejpodobnějších framů z vytvořené databáze. Podobnost mezi framy byla určena na základě nejmenší vážené vzdálenosti klíčových bodů srovnávaných framů s takovými vahami, aby byl kladený největší důraz na klíčové body ležící na kontuře rtů. Konkrétně byly váhy experimentálně nastaveny na hodnotu 10 klíčových bodů kontury rtů, na hodnotu 2 klíčových bodů očí a na hodnotu 1 ostatních klíčových bodů. Na obr. 22 vidíme příklady výsledných nalezených obrazů skic tváře - skicu originálního vstupního snímku (černá barva) a k ní nejpodobnější skicu snímku z databáze snímků poskládaných do syntetizovaného videa (modrá barva).



Obrázek 20: Skica referenčního snímku v čase $t-1$ a k němu nejpodobnější skica z databáze videí



Obrázek 21: Skica referenčního snímku v čase t a k němu nejpodobnější skica z databáze videí



Obrázek 22: Skica referenčního snímku v čase $t+1$ a k němu nejpodobnější skica z databáze videí

9.3.4 Implementace v Pythonu

Matlab je komerčním programem s vysokou cenou licence, oproti tomu Python je zcela zdarma a poskytuje obrovské množství knihoven pro práci s obrazem

i zvukem a v neposlední řadě se stal i hlavním programovacím prostředkem v oblasti hlubokého učení. Proto jsem výsledný program sestavující syntézu mluvící skici realizoval v Pythonu. Tento program navíc před výpočtem vzdáleností jednorázově provádí normalizaci každého framu databáze (výsledek normalizace je permanentně uložen a použit při novém spuštění programu) a normalizaci každého framu vstupního videa. Normalizace se provádí aplikováním lineární transformace na každý snímek (videoframe) tak, aby každý ze 3 klíčových bodů (klíčový bod nosu (klíčový bod 35), krajní bod levého (46) a krajní bod pravého oka (40)) měl po této transformaci na všech snímcích stejné souřadnice. Klíčové body byly vybrány jako body nosu, levého a pravého oka, které minimálně mění svou polohu během promluvy. Tyto cílové souřadnice všech tří bodů byly získány jako souřadnice zmíněných tří bodů ze snímku etalonu, který je definován jako nejvíce neutrální snímek (tvář na něm má co nejvíce přímý pohled) ze všech snímků databáze videí. Snímek etalon byl vybrán jako snímek databáze videí, který má největší součet vzdáleností mezi těmito třemi body ze všech snímků databáze videí.

Z důvodů zmíněných výše se v budoucím výzkumu budu orientovat na implementaci v jazyce Python.

10 Zhodnocení a závěr

Během své bakalářské práce jsem se seznámil se základními pojmy z oblasti neuronových sítí se zaměřením na hluboké učení, připravil jsem základní přehled metod syntézy audiovizuální hlavy, navrhl perspektivní metodu syntézy audiovizuální hlavy opírající se o přístup audiovizuální syntézy řeči metodou LSP, která je první prezentovanou demoverzí řečí řízené fotorealistické perzonalizované animace mluvící hlavy v reálném čase, připravil počáteční databázi pro experimenty a experimentálně ověřil základní funkčnost metody audiovizuální syntézy hlavy.

Pokud mohu na základě výše uvedených činností zhodnotit dosažené výsledky a doporučit směr další práce, pak na základě prostudování recentní literatury a provedení experimentů spatřuji jako nejvýhodnější směr dalšího výzkumu a vývoje jít cestou modelování úlohy syntézy z řeči množinou neuronových sítí, každou z nich s různou architekturou specializovanou na určitou podoblast celé úlohy. Podobně i struktura lidského mozku není homogenní, ale skládá se ze specializovaných oblastí odpovědných za určitou část úkolu, který mozek jako celek musí řešit. Konkrétně předkládám návrh modelovat zvlášť reprezentaci řečového signálu posloupností vektorů v latentním prostoru příznaků, zvlášť pohyb úst a tváře, zvlášť pohyb hlavy a těla a zvlášť fotorealistické ztvárnění audiovizuální hlavy. Na rozdíl od metody LSP navrhuji nahradit metodu autoregresivního prediktivního kódování (APC) modelem wav2vec 2.0, který by mohl poskytnout možná lepší výsledky reprezentace řečového signálu. Též je možné zvážit se orientovat na reprezentaci vícemodálního prostoru, tj. prostoru příznaků nesoucích informaci o řeči i pozici kontury rtů, popř. jazyka. V modelování pohybu úst v prostoru 3D je pak důležitou otázkou dostupnost a využití

3D trackeru k poskytnutí informace o reálném pohybu klíčových bodů v prostoru. Po experimentu s 2D trackerem, který jsem provedl, vyvstává též otázka vyhlazení údajů (souřadnic klíčových bodů v prostoru 2D) poskytovaných trackerem, neboť tyto hodnoty někdy "nadměrně kolísají". Mezivýsledek v syntéze v podobě mluvící skici tváře se mi jeví jako dobrý přístup, možný směr dalšího výzkumu by mohl zahrnovat vývoj obecného nepersonalizovaného modelu skici tváře s následnou adaptací na konkrétní osobu, možná s využitím architektur transformerů. Pro fotorealistickou syntézu je pravděpodobně vhodným přístupem použití Image–to-Image Translation sítě se vstupem skici tváře a několika snímků hlavy osoby, jejíž vizuální složka hlavy se má ztvárnit v podobě mluvící hlavy trénované jako (podmíněné) sítě GAN.

Audiovizuální ztvárnění syntézy řeči mluvící hlavou má v budoucnosti dalekosáhlé využití. Nicméně, současný stav „state of the art“ vyvíjených modelů má k cíli ještě daleko, zejména v souvislosti s faktem, že každý člověk je evolučně velmi citlivý k „falešnému“ výrazu tváře, tedy především pohybu úst, obličeje a horní části trupu. Proto budoucí využití v mluvících digitálních avatarech, vizuálním dabingu, při videokonferencích, při ztvárňování virtuální reality, počítačových hrách a syntéze řeči z mnoha dalších důvodů vyžaduje od reality nerozeznatelné ztvárnění syntézy řeči mluvící hlavou, tj. především věrohodné synchronizování pohybu rtů s perzonalizovanou obličejovou dynamikou a s nimi souvisejícími pohyby horních částí trupu.

Výzkum a vývoj však v posledních letech znatelně pokročil. Např. v oblasti audiosyntézy jsou dnes některé syntetické hlasy hodnoceny jako více přirozené než lidské a podobný vývoj se dá očekávat i v oblasti audiovizuální syntézy. Závěrem bych rád znovu poděkoval vedoucímu své diplomové práce a jeho spolupracovníkům za podporu, kterou mi věnovali. Pracoviště Katedry kybernetiky FAV ZČU se dlouhodobě zabývá výzkumem syntézy znakového jazyka. Byl bych rád, kdyby tato bakalářská práce mohla přispět i k řešení této úlohy.

Reference

[1] Tarik Arici and Asli Celikyilmaz. Associative adversarial networks. *ArXiv*, abs/1611.06953, 2016.

[2] Alexei Baevski, Henry Zhou, Abdel rahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *ArXiv*, abs/2006.11477, 2020.

[3] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. *ArXiv*, abs/1609.01743, 2016.

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, 2017.

[5] Lele Chen, Guofeng Cui, Celong Liu, Zhong Li, Ziyi Kou, Yi Xu, and Chenliang Xu. Talking-head generation with rhythmic head motion. In *ECCV*, 2020.

[6] Yu-An Chung and James Glass. Improved speech representations with multi-target autoregressive predictive coding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2353–2358, Online, July 2020. Association for Computational Linguistics.

[7] Mohamed A. Elgharib, Mohit Mendiratta, Justus Thies, Matthias Nießner, Hans-Peter Seidel, Ayush Tewari, Vladislav Golyanik, and Christian Theobalt. Egocentric videoconferencing. *ACM Transactions on Graphics (TOG)*, 39:1 – 16, 2020.

[8] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[10] Xintong Han, Zuxuan Wu, Weilin Huang, Matthew R. Scott, and Larry S. Davis. Finet: Compatible and diverse fashion image inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[11] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Gregory Frederick Diamos, Erich Elsen, Ryan J. Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and A. Ng. Deep speech: Scaling up end-to-end speech recognition. *ArXiv*, abs/1412.5567, 2014.

[12] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Trans. Graph.*, 39:236:1–236:14, 2020.

[13] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial networks. jun 2014.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[17] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics*, 37(4):163:1–14, August 2018.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[19] Yuanxun Lu, Jinxiang Chai, and Xun Cao. Live speech portraits: Real-time photorealistic talking-head animation. *ArXiv*, abs/2109.10595, 2021.

[20] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, 2017.

[21] Ležanský a kol Mařík, Štěpánková. *Umělá inteligence*. Academia, Praha, 4 edition, 2003.

[22] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

[23] Christopher Olah. Understanding LSTM Networks, August 2015.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

[26] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. In *ECCV*, page 2323–2326, 2000.

[27] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. dec 2018.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[29] Justus Thies, Mohamed Elgharib, Ayush Tewari, Christian Theobalt, and Matthias Nießner. Neural voice puppetry: Audio-driven facial reenactment. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 716–731, Cham, 2020. Springer International Publishing.

[30] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019.

[31] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[32] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[33] Olivia Wiles, A. Sophia Koepke, and Andrew Zisserman. X2face: A network for controlling face generation by using images, audio, and pose codes. In *ECCV*, 2018.

[34] Hang Zhou, Yu Liu, Ziwei Liu, Ping Luo, and Xiaogang Wang. Talking face generation by adversarially disentangled audio-visual representation. *ArXiv*, abs/1807.07860, 2019.

[35] Yang Zhou, Xintong Han, Eli Shechtman, Jose Echevarria, Evangelos Kalogerakis, and Dingzeyu Li. Makelttalk: Speaker-aware talking-head animation. *ACM Trans. Graph.*, 39(6), nov 2020.

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.