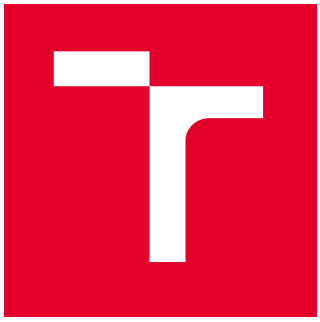


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

INTERPRETACE ZNAKOVÉ ŘEČI

INTERPRETATION OF SIGN LANGUAGE

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Dominik Ficek**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Miloslav Richter, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Dominik Ficek ***ID:*** 203219

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Interpretace znakové řeči

POKYNY PRO VYPRACOVÁNÍ:

1) Nastudujte základy znakového jazyka. Dále nastudujte možnosti sledování a interpretace tohoto jazyka pomocí prostředků počítačového vidění - HW a SW prostředky. Popište současný stav řešení a vlastnosti existujících aplikací řešících tuto problematiku.
2) Zvolte vhodný HW, definujte pracovní scénu (rozměry, vlastnosti, osvětlení ...) a nasnímejte základní sady pro testování algoritmů.
3) Navrhněte postup řešení, zvolte vhodné metody/algoritmy pro řešení, stanovte datové formáty pro přenos dat.
4) Navržený postup realizujte a zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

Gonzalez R.C., Woods R.E.: Digital Image Processing, 4th edition, Pearson, 2017, ISBN 978-0133356724

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 18.5.2022

Vedoucí práce: Ing. Miloslav Richter, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce se zabývá metodami zpracování obrazové informace člověka interpretující znakovou řeč s cílem rozpoznávání této řeči a následné automatické interpretace počítačovým modelem v jiném znakovém jazyce. V návaznosti na teoretický úvod je provedena rešerše stávajících metod a aplikací zabývající se touto problematikou. Na základě provedené rešerše je navržen řetězec zpracování obrazové informace pro účely rozpoznávání znakového jazyka sledováním rukou a obličeje interpreta. Navržený řetězec je realizován a evaluován a je sestavena aplikace rozpoznávající americkou znakovou řeč, která je automaticky interpretována v českém znakovém jazyce.

KLÍČOVÁ SLOVA

Rozpoznávání znakového jazyka, počítačové vidění, konvoluční neuronové sítě, objektové detektory, estimace pózy rukou, estimace klíčových bodů obličeje, sledování objektů, automatická interpretace

ABSTRACT

This work deals with image processing methods in persuit of sign language recognition and automatic interpretation in target sign language by artificial avatar. Following the theorethical introduction to the topic is a brief survey of current state of the art in sign language recognition. Proposed pipeline for the purposes of sign language recognition is based on the survey and latest state of the art hand and face landmark estimation methods. The designed pipeline is implemented and evaluated on suited metrics. As an application of this pipeline a system recognizing american sign langauge with automatic interpretation in czech sign language is built.

KEYWORDS

Sign language recognition, computer vision, convolutional neural networks, object detectors, hand pose estimation, face keypoint estimation, object tracking, automatic interpretation

FICEK, Dominik. *Interpretace znakové řeči*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 102 s. Diplomová práce. Vedoucí práce: Ing. Richter Miloslav, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Dominik Ficek
VUT ID autora:	203219
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Interpretace znakové řeči

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 16. května 2022 .....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Miloslavu Richterovi, Ph.D. za odborné vedení během studia, konzultace, vstřícnost, podnětné návrhy k práci a trpělivost.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Jedním ze zdravotních problémů lidstva je sluchové postižení jedince, kterým je zasaženo okolo 5% celkové lidské populace. Mezi komunikační nástroje takto zasažených spadá, také mimo jiné, znaková forma jazyka. Přirozeně pak vzniká jazyková bariéra mezi znakující minoritou a hlasově se dorozumívající majoritou. Dále pak vzniká jazyková bariéra mezi uživateli znakových řečí různých národů z důvodu rozdílnosti jednotlivých národních znakových řečí. Jedním ze způsobů možné redukce těchto jazykových a kulturních bariér je strojové porozumění nosičům znakových řečí a jejich automatické zpracování pro účely překladu do mluvené nebo psané formy nebo pro účely překladu mezi rozdílnými znakovými jazyky.

V rámci této práce jsou prozkoumány možnosti využití algoritmů z oblasti strojového učení pro potřeby porozumění nosičům znakových jazyků z vizuální informace. Dále je tato informace analyzována z dynamického hlediska, dané nosiče jsou sledovány v čase a interpretace je klasifikována do znaků slovní zásoby. Jako aplikace tohoto systému je zvolena automatická interpretace sledovaného jazyka v jiném znakovém jazyce za pomocí počítačového modelu.

V první části práce je teoreticky rozebrána znaková forma jazyka a kulturní rozdíly mezi jednotlivými znakovými jazyky. Dále jsou pak popsány vybrané aspekty z oblasti teorie umělých neuronových sítí a jejich aplikace pro účely zpracování vizuální a sekvenční informace. V navazující kapitole jsou rozebrány stávající metody zpracovávající obrazovou informaci pro účely porozumění znakujícímu člověku. Následně je na základě průzkumu navržen řetězec zpracování vizuální informace pro účely rozpoznávání interpretovaného jazyka. Ve třetí kapitole je popsána realizace jednotlivých bloků navrženého řetězce a navržené aplikace automatické interpretace. V poslední kapitole jsou algoritmy zhodnoceny a jsou diskutovány možnosti návaznosti práce.

1 Teoretický úvod

V rámci této kapitoly jsou popsány základy znakového jazyka a je stručně popsán kulturní aspekt jednotlivých národních znakových řečí. Následně jsou popsány vybrané části teorie hlubokého učení pro potřeby porozumění znakové řeči ze sekvenční obrazové informace.

1.1 Znakový jazyk

Znakový jazyk [1] je obdobně jako jazyk mluvený, systémem jednotek a pravidel jejich použití. Od jazyka mluveného se liší hlavně způsobem interpretace, znakové jazyky jsou nevokální a jsou vnímané zrakem, oproti jazykům mluveným, které jsou vokální a jsou vnímány převážně sluchem. Můžeme tedy říci, že jazyky znakové jsou založeny na tvarech, pozicích a pohybu, jinými slovy jsou to jazyky vizuálněmotorické.

V jazyce mluveném jsou jazykové jednotky (hlásky, slova, věty) skládány sekvenčně, v jazycích znakových tomu může být i jinak. Některé výrazy jsou stejně jako u jazyka mluveného skládány sekvenčně, avšak jiné výrazy mohou být produkovány současně. V komunikaci znakovou formou jsou k dispozici dva nosiče významu. Prvním nosičem je pohyb, tvar a pozice rukou, druhým nosičem je pak mimika, pohyb a pozice hlavy a horní části trupu. Oba nosiče mohou produkovat nebo být vnímány současně.

Základnou pro interpretaci znakového jazyka je trojrozměrný prostor. V prostoru jsou rozvrženy subjekty komunikace, pro potřeby odkazu k přímým účastníkům komunikace nebo k subjektům, které figurují jako předmět sdělování. Dále prostor figuruje jako základna textové soudružnosti nebo základna pro vyjádření časoprostorových vztahů mezi předměty. Pomocí prostoru jsou také ohýbána slovesa nebo také slouží pro vyjádření věcněobsahových vztahů.

1.1.1 Rodiny znakových řečí

Z uvedených ustanovení je zjevné, že různé znakové jazyky sdílejí specifické základní rysy [2]. Co je však potřeba zdůraznit je, že tyto rysy nevytváří univerzální mezinárodní znakový jazyk. V různých zemích existují specifické národní znakové jazyky, které se od ostatních liší. Znakové jazyky [3], na rozdíl od jazyků mluvených jsou velmi mladé. První zmínky o formálním popisu znakové řeči jsou z konce 18. století, kdy L'Abbé de L'Épée založil první školu znakové řeči pro neslyšící, L'Épée vycházel z již existujících znaků používaných v Paříži a složil první formální popis francouzské znakové řeči. Zakladatel první americké školy znakové řeči Edward Miner Gallaudet,

L'Épéeho školu navštívil a najal jednoho z předních studentů francouzské školy, aby vyučoval na jeho škole v Hartfordu v Connecticutu. Tento příklad ukazuje, že existují jednotlivé soubory znakových řečí, označovány jako rodiny znakových řečí, které sdílí kořeny řeči, obdobně jako jsou si podobné mluvené jazyky např. slovanského původu. Rozšíření jednotlivých rodin je dáno tedy převážně pohybem jejich učitelů po světě. Z tohoto také vyplývá, že národní znakový jazyk nemusí přímo odpovídat národnímu mluvenému jazyku. Můžou pak nastat případy kdy dva národy používají stejný mluvený jazyk, ale používají jiný znakový jazyk nebo případ, kdy dva národy používají jiný mluvený jazyk, ale používají stejný znakový jazyk. Příkladem může být Taiwan, kde je národní mluvený jazyk čínština a taiwanský znakový jazyk pochází z japonské rodiny, do které čínský znakový jazyk nespadá.

L'Épéeho škola dala základy dnešnímu francouzskému znakovému jazyku (zkráceně LSF, *Langue de Signe Française*), který se rozšířil do části Evropy, Jižní Ameriky a také do Spojených Států Amerických. Dalších rodin znakových řečí je mnoho a původ jednotlivých národních znakových řečí je často sporný. Původ české znaková řeči je připisován francouzské rodině a rodině pocházející z Rakouska-Uherska.

Nutno také zmínit fakt, že mluvený jazyk má velký vliv na znakový jazyk používaný konkrétním národem. Tyto vlivy, které často tvoří hlavní rozdíly mezi znakovými jazyky ze stejné rodiny jsou popsány v další sekci. Není tedy faktem, že všechny znakové jazyky ze stejné rodiny jsou identické.

1.1.2 Vliv mluveného jazyka na jazyk znakový

Ve světě koexistence znakového jazyka a mluveného jazyka je zjevné, že dominantní formou komunikace je komunikace jazykem mluveným. Lidé komunikující řečí znakovou tedy nezbytně musejí ovládat formu mluveného jazyka ve své psané formě. Tímto způsobem je pak znaková řeč ovlivněna řečí mluvenou, zejména v oblastech používání prstové abecedy, inicializace, mluvních komponentů a kalkování [4].

Užitím prstové abecedy se myslí manuální znakování grafémů psané formy příslušného mluveného jazyka. Její použití je ovlivněno používaným jazykem mluveným a může nabývat v jazyce unikátní role, specificky pro americký znakový jazyk se hláskovaný znak spojuje se znakem blízkého významu výsledného významu kombinovaného znaku. Obecně je prstová abeceda používána v případě, kdy znakující nezná specifický znak, který pak vyhláskuje a při dalších výskytech je proces opakován nebo může být použita pouze první hláska sekvence v kombinaci s bezhlasou artikulací.

Inicializace znaku je technika změny významu výsledného znaku použitím inicializačního znaku. Často je využívána první psaná hláska výsledného znaku jako incializační znak transformující význam doprovodného znaku. V praxi se pak může

několik znaků lišit pouze svým inicializačním znakem.

Mluvní komponenty znakového jazyka jsou taktéž ovlivněny jazykem mluveným, jedná se o bezhlasou artikulaci interpretovaného slova nebo jeho části. Rozlišuje se tedy mezi přímou artikulací znakovaného slova v mluvené formě a mezi ústním gestem, které nemá přímou artikulační vazbu k mluvenému slovu (tyto gesta jsou označovány jako orální komponenty). Významem mluvených komponentů je rozlišit význam znaku a význam znaku specifikovat, jsou tedy často používány společně se znaky, které nesou pouze obecný význam.

Kalkování je přejatí slova nebo slovního spojení doslovným překladem z mluveného jazyka. Tento mechanismus je obdobně používán také v jazyce mluveném.

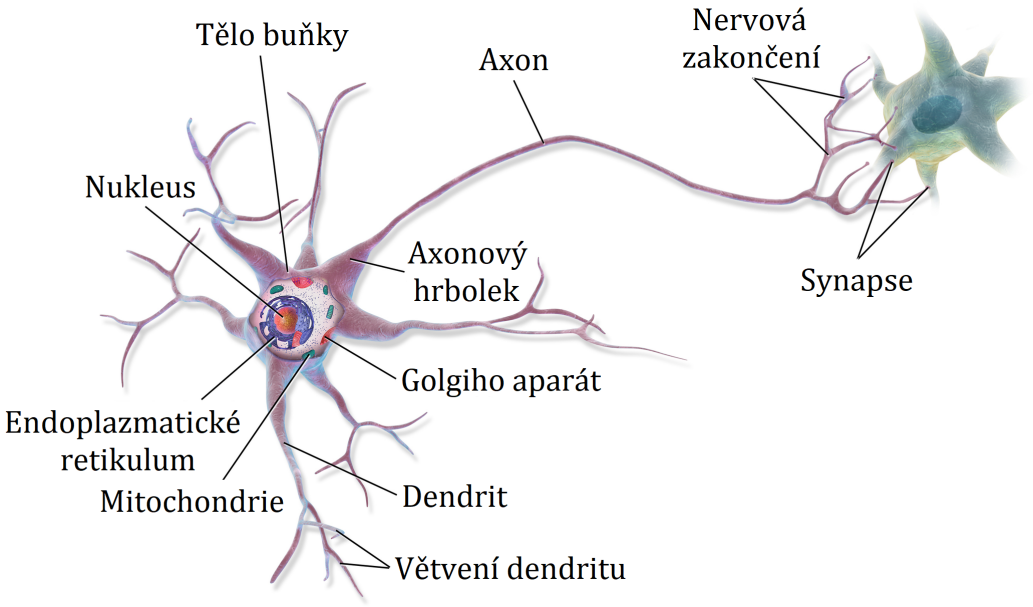
1.2 Umělé neuronové sítě

První zmínky o umělých neuronových sítích [5] jsou z roku 1943 kdy neurofyzik Warren McCulloch a matematik Walter Pitts vydali publikaci *A Logical Calculus of Idea Immanent in Nervous Activity* [6], kde popisují první zjednodušený výpočetní model biologického neuronu mozku zvířete. McCulloch a Pitts tehdy zavedli první architekturu umělé neuronové sítě. Tyto úspěchy vedly k velkým očekáváním z této oblasti, k zásadnímu pokroku došlo v roce 1958 kdy psycholog Frank Rosenblatt definoval model *Perceptronu* [7], modelující jednoduchý rozhodovací systém s více vstupy. V reakci na Resonblattovu práci vydali v roce 1969 tehdejší vědci z laboratoře umělé inteligence MIT Marvin Minsky a Seymout Papert knihu *Perceptrons* [8], ve které kritizují model percepronu a možnosti jeho použití ve větším měřítku. Minsky a Papert byli dostatečně vlivní, aby touto knihou téměř pozastavili jakýkoliv výzkum umělých neuronových sítí. V 80. letech můžeme sledovat obnovení popularity umělých neuronových sítích. V roce 1982 John Hopfield ve své práci *Neural Networks and Physical Systems with Emergent Collective Computational Abilities* [9] definoval novou architekturu umělé neuronové sítě (dnes známá jako *Hopfieldova neuronová síť*), prozkoumávající paměťové vlastnosti umělých neuronových sítí. Ke konci 80. let byla pro použití umělých neuronových sítí přeformulována metoda zpětné propagace chyby [10], techniky používající se dodnes pro optimalizaci parametrů umělých neuronových sítí. V návaznosti na reformulaci zpětné propagace Yann LeCun a ostatní publikovali první konvoluční neuronovou síť [11], druh architektury široce využívaný v dnešní době pro potřeby zpracování více-dimenzionálních dat. V 90. letech byla představena publikace *Support Vector Machines* [12] dnes velmi známá technika strojového učení, která se tehdy zdála jako v mnoha ohledech lepší technikou, než se zdálo použití umělých neuronových sítí a použití umělých neuronových sítí opět přešlo do pozadí. Později v roce 2012 Alex Krizhevsky a ostatní [13] dosáhli pomocí konvoluční neuronové sítě významně lepších výsledků na klasifikační

databázi *ImageNet* [14], než bylo do té doby dosaženo pomocí ostatních technik. Tyto výsledky společně s ostatními vlivy, jako např. výrazně vyšší výpočetní možnosti a přístup k výrazně většímu množství dat, způsobily významný růst popularity umělých neuronových sítí v posledních letech a tím i významný pokrok z hlediska výzkumu.

1.2.1 Biologický neuron

Biologický neuron [5] (obrázek 1.1), nebo také buňka nervové tkáně se vyskytuje převážně v mozcích živočichů. Skládá se z těla buňky, které obsahuje nukleus a většinu svých komplexních částí, spousty větvících se dendrit a axonu, jedné dlouhé rozšiřující linky. Délka axonu se mezi jednotlivými neurony výrazně liší, může být několikrát nebo až deseti-tisíckrát delší než velikost těla buňky. Ke konci své délky se axon dělí na mnoho nervových zakončení a na konci těchto zakončení se nacházejí synapse, které jsou pak umístěny velmi blízko dendritů nebo těl jiných neuronů. Neurony produkují krátké elektrické impulsy, které jsou vedeny po axonu. Synapse



Obr. 1.1: Biologický neuron [15]

po kontaktu s dostatečně silným impulsem uvolní chemický signál nazývaný neurotransmiter. Pokud neuron přijme dostatečné množství neurotransmiteru v řádově milisekundách, tak v závislosti na druhu neurotransmiteru excituje nebo neexcituje signál. Tímto způsobem jsou neurony uspořádány do sítě o několika miliard neuronů, kde jsou jednotlivé neurony propojeny s několika tisíci jinými neurony. Specifická architektura biologické neuronové sítě je stále předmětem výzkumu, avšak některé

výzkumy naznačují, že neurony jsou organizovány do jednotlivých vrstev, obzvláště v mozkové kůře.

V rámci pokračování této práce je při zmínění o neuronové síti myšlena umělá neuronová síť a obdobně i neuronem je myšlen umělý neuron.

1.2.2 Perceptron

Perceptron [5] je jedna z nejjednodušších architektur neuronových sítí. Perceptron je založen na jednoduché formě umělého neuronu, nazývaný lineární prahovaná jednotka (zkráceně *LTU* z angličtiny - *Linear Threshold Unit*). Model lineární prahované jednotky je vyobrazen na obrázku 1.2 vlevo, skládá se z vektoru vstupů $\mathbf{x} \in \mathbb{R}^{\mathrm{K}}$, vektoru vstupních vah $\mathbf{w} \in \mathbb{R}^{\mathrm{K}}$, biasu $b \in \mathbb{R}$, přechodové funkce f a výstupu y. Pro výstup lineární prahované jednotky pak platí rovnice

$$y = f(\mathbf{x}^T\mathbf{w} + b) \tag{1.1}$$

Nejčastější přechodové funkce používané LTU jsou Heavisidova přechodová funkce nebo funkce signum.

$$heaviside(z) = \begin{cases} 0, & \text{pro } z \ < 0 \\ 1, & \text{pro } z \ \geq 0 \end{cases} \tag{1.2}$$

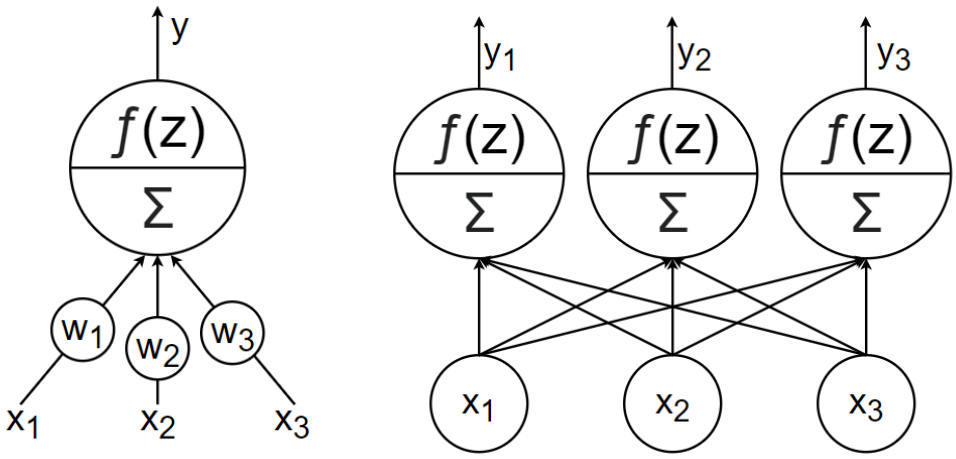
$$sign(z) = \begin{cases} -1, & \text{pro } z \ < 0 \\ 0, & \text{pro } z \ = 0 \\ +1, & \text{pro } z \ > 0 \end{cases} \tag{1.3}$$

Jedna LTU může být použita jako lineární binární klasifikátor. Perceptron je pak jedna vrstva lineárních prahovaných jednotek, kde každá LTU je propojená s každým vstupem. Vrstva, jejíž neurony jsou všechny propojeny se všemi vstupy se nazývá plně propojená, v angličtině označována jako *dense*. Model perceptronu se třemi LTU a třemi vstupy je ilustrován na obrázku 1.2 vpravo. V některých případech je do každé LTU přiveden konstantní vstup, označovaný jako *bias*. Počet LTU v perceptronu udává počet výstupů, model s N výstupy pak může klasifikovat až do N binárních tříd. Pro výstup perceptronu platí rovnice

$$\mathbf{y} = f(\mathbf{x}\mathbf{W} + \mathbf{b}) \tag{1.4}$$

kde $\mathbf{y} \in \{-1, 0, +1\}^M$ je vektor výstupu, M je počet LTU v perceptronu, f je přechodová funkce, $\mathbf{x} \in \mathbb{R}^{\mathrm{L}}$ je vektor vstupů, L je počet vstupů, $\mathbf{W} \in \mathbb{R}^{\mathrm{L}\times\mathrm{M}}$ je matice vah perceptronu a $\mathbf{b}^M$ je vektor biasu.

Proces učení perceptronu je adaptace, optimalizace matice vah. Algoritmus optimalizace vah byl navržen v původní publikaci Franka Rosenblatta [7]. Rovnice



(a) Model LTU (b) Model perceptronu

Obr. 1.2: LTU a Peceptron

popisující adaptaci vah je ve tvaru

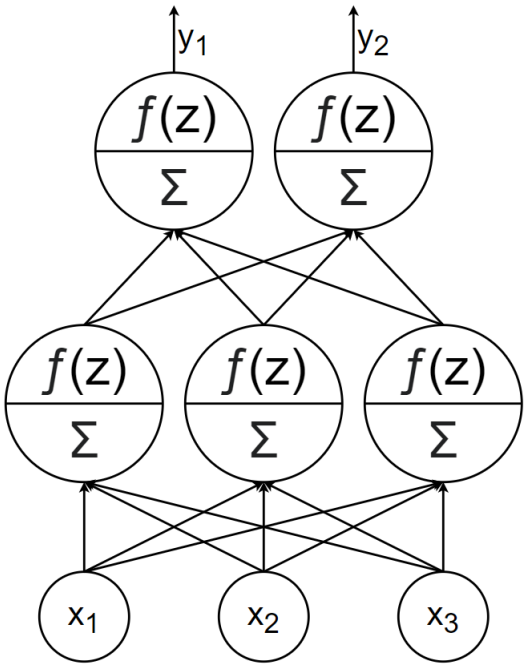
$$w_{i,j}(k+1) = w_{i,j}(k) + \eta(y_j - \hat{y}_j)x_i \tag{1.5}$$

kde $w_{i,j}(k)$ je váha j-tého neuronu napojeného na i-tý vstup v kroku k, x_i je i-tý vstup, $\hat{y}_j$ je predikovaný výstup z j-tého neuronu, y_j je požadovaný výstup z j-tého neuronu a η je koeficient učení.

Z rovnice 1.4 vyplývá, že rozhodovací rovina perceptronu je čistě lineární. Algoritmus optimalizace dokonverguje k optimálním vahám v případě, kdy je datový set lineárně separabilní. Konvergence pro lineárně separabilní datový set je dokázaná v Rosenblattově originální publikaci a je známá jako teorém konvergence perceptronu. Jedním z problémům perceptronu, které kritizovali Minsky a Papert [8], je inabilita perceptronu vyřešit některé jednoduché úlohy, specificky logickou funkci XOR, které jiné lineární klasifikátory vyřešit mohou. Tento problém je možné vyřešit složením více perceptronů nad sebe. Výsledkem je vícevrstvý perceptron, v angličtině *Multilayer Perceptron - MLP*. V případech, kdy neuronová síť obsahuje více vrstev, jsou označovány jako hluboké neuronové sítě.

1.2.3 Hluboké neuronové sítě

Ukázka architektury MLP je ilustrována na obrázku 1.3, jedná se tedy o hlubokou neuronovou síť o dvou vrstvách, první vrstva je plně propojená a složena ze tří neuronů, druhá vrstva je taktéž plně propojená a obsahuje dva neurony. Vrstvy, které nejsou výstupními vrstvami nazýváme jako vrstvy skryté, ukázka MLP obsahuje tedy jednu skrytou a jednu výstupní vrstvu. Algoritmus učení hlubokých



Obr. 1.3: Vícevrstvý perceptron

neuronových sítí byl předmětem tehdy průkopové publikace *Learning Internal Representations by Error Propagation* [10]. Jedná se o iterativní algoritmus výpočtu příspěvků jednotlivých vah k chybě zpětnou propagací sítí (v angličtině označováno jako *error backpropagation*). Algoritmus je složen z kroku dopředné propagace, tedy výpočet výstupu sítě a následně výpočet gradientů chybové funkce podle jednotlivých vah sítě $\frac{\partial L}{\partial w}$. Algoritmus je ukončen konvergencí chyby k optimu. Podmínkou použití tohoto algoritmu je, aby všechny části výpočtu výstupu neuronové sítě byly diferencovatelné.

Způsob aplikace vypočtených gradientů pro úpravu vah neuronových sítí se nazývá krok optimalizace. V originální publikaci zpětné propagace [10] byla použita optimalizace pomocí techniky nazývané *gradient descent*. *Gradient descent* bere v potaz aktuální hodnoty gradientů, pomocí kterých jsou aktualizovány hodnoty vah podle rovnice

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \tag{1.6}$$

kde $\mathbf{w}(k)$ jsou váhy sítě v kroku k, η je učící koeficient a L je chybová funkce. Variantou tohoto algoritmu je stochastický *gradient descent* (zkráceně SGD) [16, 17]. Standardní *gradient descent* je obecně počítán po jednotlivých sadách dat, neuronová síť tedy není datům vystavená jedno po druhém, ale po sadách o B vzorků z datového setu. Stochastický *gradient descent* je varianta, kdy jsou data náhodně (stochasticky) podávána neuronové síti, vždy pouze jeden vzorek najednou. Tento

algoritmus je i v dnešní době velmi populární, jelikož SGD téměř vždy spolehlivě dokonverguje k optimu. V návaznosti na *gradient descent* vydal v roce 1964 Boris Polyak studii *Some methods of speeding up the convergence of iteration methods* [18], kde představuje princip použití momenta, které výrazně urychluje proces učení sítě. Standardní *gradient descent* nebere v potaz předchozí hodnoty gradientu, algoritmus využívající momentum využívá i předchozí hodnoty gradientu pro aktualizaci vah a simuluje jev akcelerace. Rovnice popisující optimalizaci vah pomocí momenta jsou ve tvaru

$$\begin{aligned} \mathbf{m}(k+1) &= \beta \mathbf{m}(k) + \eta \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \mathbf{m}(k+1) \end{aligned} \tag{1.7}$$

kde $\mathbf{m}(k)$ je momentum gradientů v kroku k a β je koeficient simulující frikci akcelerace. V průběhu let došlo k výraznému pokroku metod aplikující adaptivní koeficient učení pro jednotlivé parametry sítě. Specificky metoda *adagrad* [19] adaptivně snižuje učící koeficient η na základě velikosti gradientů pro jednotlivé parametry. Tento přístup funguje jako korekce směru gradientů, a tím zrychluje dobu učení neuronové sítě. Matematické vyjádření *adagrad* algoritmu je pak ve tvaru

$$\begin{aligned} \mathbf{s}(k+1) &= \mathbf{s}(k) + \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \otimes \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \eta \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \oslash \sqrt{\mathbf{s}(k+1) + \epsilon} \end{aligned} \tag{1.8}$$

kde $\mathbf{s}(k)$ je koeficient redukce koeficientu učení v kroku k a ϵ je koeficient zaručující numerickou stabilitu [5]. Symboly $\otimes$ a $\oslash$ reprezentují násobení a dělení matic po prvcích. Z rovnice lze pozorovat, že redukce koeficientu učení je úměrná součtu kvadrátů všech předchozích gradientů a jedním z problémů *adagrad* algoritmu je nebezpečí rapidní redukce koeficientů učení, která vede k nedokončení konvergence. Následná modifikace algoritmu *adagrad* je algoritmus *adadelta* [20], která řeší problém rapidního zpomalování konvergence, tím že postupně snižuje váhu předchozím gradientům. Výsledný tvar rovnice je

$$\begin{aligned} \mathbf{s}(k+1) &= \beta \mathbf{s}(k) + (1-\beta) \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \otimes \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \eta \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \oslash \sqrt{\mathbf{s}(k+1) + \epsilon} \end{aligned} \tag{1.9}$$

kde je přidán koeficient β, váha předchozích gradientů parametrů. Jeden z dnes nejrozšířenějších optimalizačních algoritmů je *adam* [21], kombinující vlastnosti *adadelta* a momenta. Matematická formulace [5] algoritmu *adam* je ve tvaru

$$\begin{aligned} \mathbf{m}(k+1) &= \beta_1 \mathbf{m}(k) + (1-\beta_1) \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \\ \mathbf{s}(k+1) &= \beta_2 \mathbf{s}(k) + (1-\beta_2) \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \otimes \nabla_{\mathbf{w}} L(\mathbf{w}(k)) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \eta \frac{\mathbf{m}(k+1)}{1-\beta_1^k} \oslash \sqrt{\frac{\mathbf{s}(k+1)}{1-\beta_2^k} + \epsilon} \end{aligned} \tag{1.10}$$

kde β_1 je frikce momenta a β_2 je váha předchozích gradientů parametrů pro redukci učících koeficientů parametrů.

Perceptron s prahovanou přechodovou funkcí 1.2 1.3 není příliš vhodný pro aplikaci těchto algoritmů, pro prahované aktivační funkce totiž platí, že $\frac{\partial f}{\partial z}$ pro $z = 0$ je rovno ∞ a pro $z \neq 0$ je parciální derivace rovna nule. Proto jsou v případě hlubokých neuronových sítí používány jiné přechodové funkce (také označovány jako aktivační funkce). Zde jsou uvedeny některé základní nebo v rámci práce použité aktivační funkce [22]:

- Funkce sigmoid, funkce často používaná pro případy normalizace výstupů neuronové sítě na rozsah $\langle 0, 1 \rangle$. Často se vyskytuje jako aktivační funkce výstupních vrstev, zejména pro úlohy binární klasifikace. Použití této funkce pro aktivaci skrytých vrstev není doporučeno z důvodu téměř nulového gradientu v oblastech, které nejsou blízké nule a také z důvodu asymetrie funkce kolem osy y.
$$f(z) = \frac{1}{1 + e^{-z}} \tag{1.11}$$
- Funkce hyperbolického tangentu řeší jednu z kritik funkce sigmoidu, absence symetrie kolem osy y. Avšak stejně jako funkce sigmoidu má téměř nulové hodnoty gradientu v oblastech ne-blízké nule. Stejně jako sigmoid je funkce hyperbolického tangentu používaná pro potřeby binární klasifikace.
$$f(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{1.12}$$
- Rektifikovaná lineární jednotka (z angličtiny *Rectified Linear Unit*, *ReLU*) je široce využívaná aktivační funkce ve skrytých vrstvách neuronových sítí. Její předností je rychlý a jednoduchý výpočet gradientu, který navzdory nediferencovatelnému bodu $z = 0$ definujeme jako $\frac{\partial f}{\partial z} = 0$ pro $z < 0$ a $\frac{\partial f}{\partial z} = 1$ pro $z \geq 0$. Další výhodou funkce je, že nedochází k saturaci výstupu, tento fakt pak v praxi vede k rychlejší konvergenci parametrů neuronové sítě ve fázi učení.
$$f(z) = max(0, z) \tag{1.13}$$
- Rektifikovaná lineární jednotka s *únikem* (z angličtiny *Leaky ReLU*) je modifikace ReLU aktivační funkce. Nevýhodou ReLU funkce je nulová hodnota gradientu pro $z < 0$, což může vést ke stavu, kdy některé neurony vykazují ve většině fáze učení nulové gradienty. Tento jev se nazývá *umírání gradientu* a může být jeden z faktorů, které brání síti naučit se komplexní funkce. Leaky ReLU tento problém řeší záměnou problematické části funkce pro $z < 0$ za lineární funkci s nenulovým gradientem $y(x) = \alpha x$, kde α je voleno v rozsahu $\langle 10^{-3}, 10^{-1} \rangle$.
$$f(z) = max(-\alpha z, z) \tag{1.14}$$

- Funkce swish kombinuje lineární jednotku s funkcí sigmoidu. Hlavním faktorem, kterým se liší od ostatních funkcích, které byly zmíněny, je že swish funkce je monotonní. Navzdory tomu, že tato funkce dává dobré výsledky v úlohách klasifikace, není přesně znám důvod výhod této funkce.

$$f(z) = \frac{x}{1 + e^{-\beta z}} \tag{1.15}$$

- Funkce softmax je používána pro výstupní vrstvy neuronových sítí řešící úlohy klasifikace do jedné třídy. Funkce převede všechny výstupní hodnoty do pravděpodobnostního rozdělení, tak že platí $\sum y_i = 1$.

$$f(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1.16}$$

Uvedené aktivační funkce, krom funkce softmax, včetně jejich derivací jsou zobrazeny na obrázku 1.4.

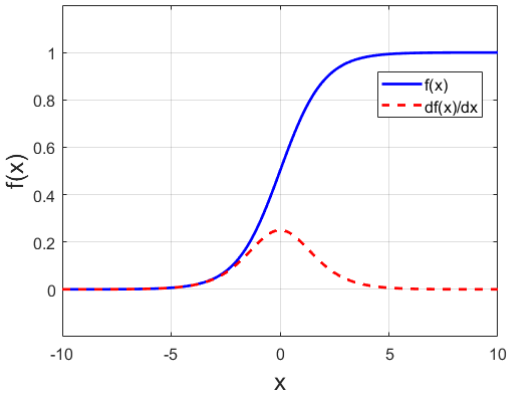
1.2.4 Konvoluční neuronové sítě

V letech 1958, 1959 a 1968 publikovali neurofyziologové David Hubel a Torston Wiesel studie o sérii experimentů na kočkách [23, 24] a opicích [25], kde předmětem studií byla struktura vizuálního kortexu. Autoři ukázali [5], že mnoho neuronů ve vizuálním kortexu má malé lokální receptivní pole, reagují tedy pouze na stimulus v limitovaném poli celého vizuálního pole. Jednotlivá receptivní pole se pak překrývají a tvoří kompletní vizuální pole. Dalším výstupem zmíněných experimentů je zjištění, že část neuronů reaguje na určitá geometrická primitiva, zatímco jiné neurony mají podstatně širší receptivní oblast a reagují na komplexnější vzory. Tato zjištění vedou k myšlence, že neurony reagující na komplexnější vzory reagují na výstupy neuronů, které jsou soustředěny na geometrická primitiva. Tyto myšlenky postupem času vedly k dnešním konvolučním neuronovým sítím, které jsou primárně používány ke zpracování obrazové informace.

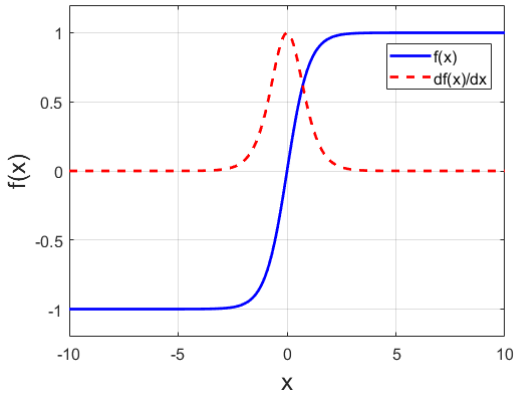
Prozatím byly neuronové sítě uvažovány jako čistě plně propojené, v praxi jsou používány vrstvy různých druhů. V případě konvolučních neuronových sítí je základním stavebním kamenem sítě konvoluční vrstva [5]. Navzdory svému názvu, konvoluční vrstva neprovádí operaci konvoluce, ale operaci křížové korelace. Rovnice definující výpočet výstupu konvoluční vrstvy je ve tvaru

$$\mathbf{y} = f(\mathbf{w} * \mathbf{x} + \mathbf{b}) \tag{1.17}$$

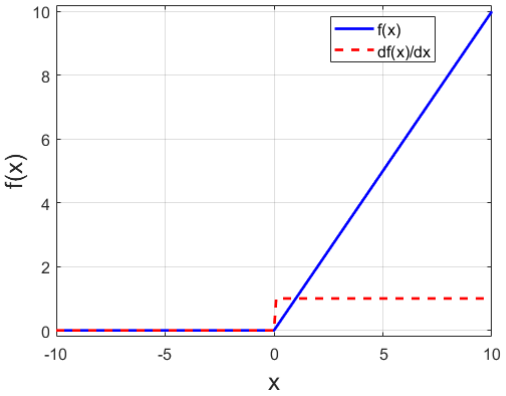
kde $\mathbf{y} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}\times\mathrm{F_{out}}}$ je výstupní tensor vrstvy, $\mathbf{w} \in \mathbb{R}^{\mathrm{F_{out}}\times\mathrm{K_1}\times\mathrm{K_2}\times\mathrm{F_{in}}}$ jsou váhy konvoluční vrstvy, $\mathbf{x} \in \mathbb{R}^{\mathrm{S}\times\mathrm{R}\times\mathrm{F_{in}}}$ je vstup do vrstvy, $\mathbf{b} \in \mathbb{R}^{\mathrm{F_{out}}}$ je bias vrstvy a f je aktivační funkce vrstvy. O vstupní informaci $\mathbf{x}$ je uvažováno jako o 3D informaci složené z F_{in} kanálů o rozměrech $S \times R$. Aplikací křížové korelace vstupu $\mathbf{x}$ s F_{out} filtry



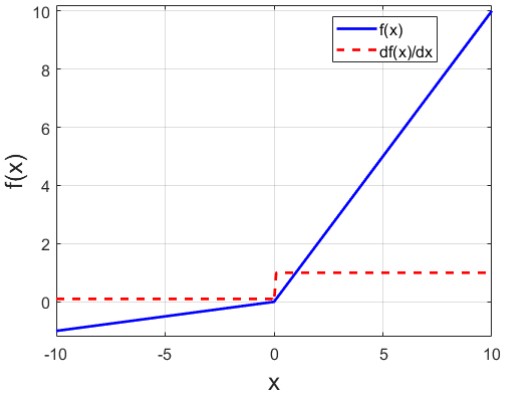
(a) Sigmoid funkce



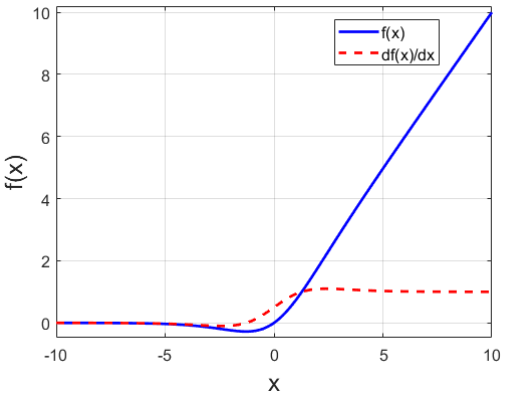
(b) Hyperbolický tangent



(c) ReLU funkce



(d) Leaky ReLU funkce pro $\alpha = 0.1$



(e) Swish funkce pro $\beta = 1$

Obr. 1.4: Aktivační funkce a jejich derivace

o rozměrech $K_1 \times K_2 \times F_{in}$, přičtení biasu (skalár pro každý výstupní kanál) a aktivací přechodovou funkcí je získán výstup $\mathbf{y}$, opět se jedná o 3D informaci složenou z F_{out} kanálů o rozměrech $N \times M$. Operace korelace je počítána pouze pro oblasti,

kde jsou všechna data validní, proto přirozeně pro velikosti filtrů $K_1 \geq 1$ a $K_2 \geq 1$ platí $N \leq S$ a $M \leq R$. Velikost N a K pak záleží na velikostech K_1, K_2, použité šířce doplněných okrajů vstupního tensoru P (označováno jako *padding*) a kroku korelace (v angličtině *stride*) T podle vztahů

$$\begin{aligned} N &= \lfloor \frac{S + 2P_n - K_1}{T_n} + 1 \rfloor \\ M &= \lfloor \frac{R + 2P_m - K_2}{T_m} + 1 \rfloor \end{aligned} \tag{1.18}$$

kde P_n, T_n jsou velikost *paddingu* a krok korelace ve směru dimenze N a P_m, T_m jsou velikost *paddingu* a krok korelace ve směru dimenze M.

Speciální variantou konvoluční vrstvy je dekonvoluční vrstva [26, 27] (pojmenovaná podle operace dekonvoluce, bývá také označovaná jako zpětná nebo transponovaná konvoluční vrstva). Motivací za touto vrstvou je rekonstrukce snímku na vstupu konvoluční vrstvy. Na rozdíl od konvoluční vrstvy, která může redukovat velikost vstupních map, dekonvoluční vrstva je používaná pro nadvzorkování vstupní informace. Rovnice popisující tuto vrstvu je ve tvaru

$$\mathbf{y} = f(\mathbf{b} + \sum_{i=1}^{F_{in}} \mathbf{w}_i * \mathbf{x}_i) \tag{1.19}$$

Tento vztah je ve skutečnosti ekvivalentní s rovnicí příspěvku konvoluční vrstvy k chybě při výpočtu zpětné propagace chyby. Obdobně jako v rovnici 1.17 uvažujeme vstupní mapy do vrstvy $\mathbf{x} \in \mathbb{R}^{\mathrm{S} \times \mathrm{R} \times \mathrm{F_{in}}}$, váhy vrstvy $\mathbf{w} \in \mathbb{R}^{\mathrm{F_{out}} \times \mathrm{K_1} \times \mathrm{K_2} \times \mathrm{F_{in}}}$, bias vrstvy $\mathbf{b} \in \mathbb{R}^{\mathrm{F_{out}}}$, aktivační funkci f a výstup z vrstvy $\mathbf{y} \in \mathbb{R}^{\mathrm{N} \times \mathrm{M} \times \mathrm{F_{out}}}$. Velikost dimenzí N a M výstupních map je pak definován jako

$$\begin{aligned} N &= T_s(S - 1) + K_1 - 2P_s \\ M &= T_r(R - 1) + K_2 - 2P_r \end{aligned} \tag{1.20}$$

kde krok T_s a T_r nedefinuje krok korelace, ale krok rozmístění bodů vstupní mapy v dimenzích S a R, mezi kterými jsou umístěny nuly. P_s a P_r definují uvažovaný vstupní padding invertované konvoluční vrstvy v dimenzích S a R.

Mezi další vrstvy používané v konvolučních neuronových sítích patří *pooling* vrstvy [5]. Jedná se o vrstvy, které cíleně podvzorkují vstupní informaci, převážně pro účely redukce výpočetní náročnosti následujících vrstev v síti a redukce paměťových požadavků sítě. Obdobně jako u konvolučních vrstev, *pooling* vrstvy pracují s kernelem definované velikosti a definovaném kroku. Kernel vrstvy aplikuje redukční funkci nad vstupní informací, výstupem jedné kernelové redukce je pak jedna hodnota. Mezi redukující funkce, které jsou používány jsou primárně používány funkce maxima a průměru. Platí tedy, že *pooling* vrstva nemá žádné váhy a transformace, kterou provádí je stejná nezávisle na parametrech sítě. Klíčovým poznatkem je, že

redukce probíhá pouze ve 2D oblasti vstupních dat, redukce probíhá tedy nezávisle mezi kanály vstupních dat.

Další vrstvou, která je široce využívaná v architekturách konvolučních neuronových sítí je normalizační vrstva (z angličtiny *Batch Normalization Layer*) [28]. Motivací za použitím této vrstvy je častá prezence vysokého rozptylu aktivací ve skrytých vrstvách neuronových sítí ve fázi učení mezi různými vzorky dat, což zpomaluje rychlost učení. Vrstva obsahuje dva statistické parametry a dva optimalizovatelné parametry:

- Klouzavý průměr vstupních aktivací $\mu_R \in \mathbb{R}^{\mathrm{F_{in}}}$
- Klouzavý rozptyl vstupních aktivací $\sigma_R^2 \in \mathbb{R}^{\mathrm{F_{in}}}$
- Normalizační koeficient střední hodnoty $\beta \in \mathbb{R}^{\mathrm{F_{in}}}$
- Normalizační koeficient rozptylu $\gamma \in \mathbb{R}^{\mathrm{F_{in}}}$

Ve fázi učení vrstvy je pro každý vstupní subset dat vypočten nezávisle pro každý vstupní kanál průměr $\mu_B \in \mathbb{R}^{\mathrm{F_{in}}}$ a rozptyl $\sigma_B^2 \in \mathbb{R}^{\mathrm{F_{in}}}$, ze kterých je současně počítán klouzavý průměr μ_R a klouzavý rozptyl σ_R^2. Každý kanál subsetu dat je pak normalizován na nulovou střední hodnotu a na rozptyl roven jedné a následně transformován na rozptyl γ a střední hodnotu β. Rovnice definující tuto transformaci vstupních dat jsou ve tvaru

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_B^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \\ \hat{x_i} &= \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\ y_i &= \gamma \hat{x_i} + \beta \end{aligned} \tag{1.21}$$

kde m je počet datových vzorků v subsetu. Při použití vrstvy v inferenci, jsou namísto střední hodnoty a rozptylu vstupních dat použity pro normalizaci hodnoty klouzavého průměru μ_R a klouzavého rozptylu σ_R^2.

$$\begin{aligned} \hat{x_i} &= \frac{x_i - \mu_R}{\sqrt{\sigma_R^2 + \epsilon}} \\ y_i &= \gamma \hat{x_i} + \beta \end{aligned} \tag{1.22}$$

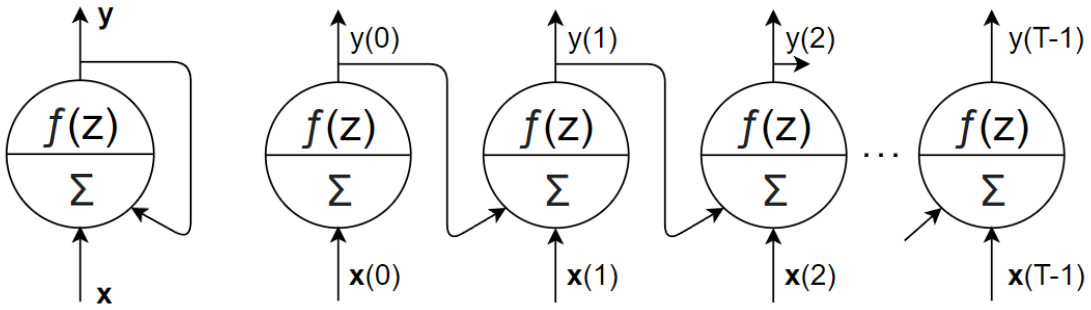
Jednou z dalších výhod této vrstvy jsou její regularizační účinky (snižuje možnosti přeučení) pro vrstvu nacházející se za normalizační vrstvou.

Regularizační nástroje jsou důležitou částí učení obecně neuronových sítí, které mívají obrovské množství parametrů a problém přeučení sítě je do jisté míry nevyhnutelný. Mezi regularizační techniky patří také využití *dropout* vrstvy [5]. Algoritmus vrstvy je jednoduchý, každá výstupní jednotka předešlé vrstvy je při procesu

učení s pravděpodobností p ignorována. Myšlenkou za tímto algoritmem je zamezení přílišné závislosti sítě na malém množství vybraných neuronů. Tímto vrstva podporuje zapojení většího množství neuronů ke správnému výpočtu výstupu. Důležitým faktem je, že vyřazování neuronů probíhá pouze při procesu učení, nikoliv při inferenci. Při inferenci, ale vzniká problém, že vrstva nacházející se za *dropout* vrstvou při inferenci dostává větší magnitudu signálu ze vstupní vrstvy. Tento jev je kompenzován faktorem $1 - p$, kterým je při inferenci násoben každý výstup *dropout* vrstvy.

1.2.5 Rekurentní neuronové sítě

Rekurentní neuronové sítě [5] jsou sítě navržené, pro zpracování sekvenční informace obecné délky. Základním stavebním blokem těchto sítí je rekurentní neuron, který na rozdíl od dosud zmíněných neuronů nebo vrstev neobsahuje pouze dopředné konexe, ale obsahuje i zpětnou vazbu. Pro každý časový krok vstupní sekvence $\mathbf{x}$ je počítána výstupní hodnota $y(t)$ ze vstupu $\mathbf{x}(t)$ a výstupu předešlého časového kroku $y(t-1)$. Výstupní hodnoty y jsou, obdobně jako u neuronů již zmíněných plně propojených vrstev, rovny nelineárně transformovanému váženému součtu vstupního vektoru $\mathbf{x}$. Na obrázku 1.5 vlevo je ilustrovaná obecná struktura rekurentního neuronu, vpravo je pak ilustrován rekurentní neuron rozvinutý v časové ose. Stejně jako

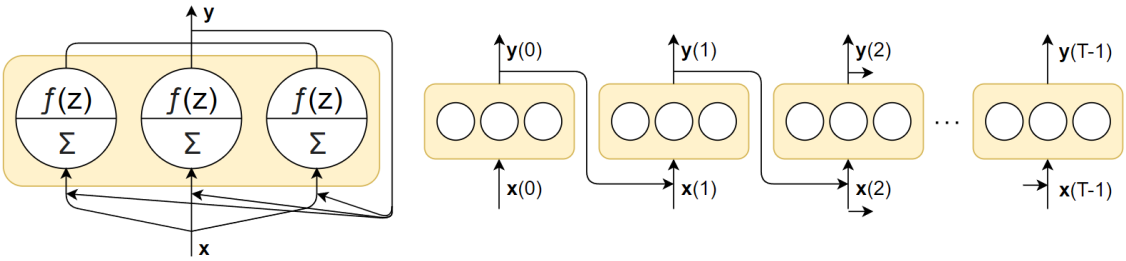


(a) Rek. neuron (b) Rekurentní neuron rozvinutý v čase

Obr. 1.5: Rekurentní neuron

standardní dopředné neurony, můžeme rekurentní neurony řadit do vrstev. V tomto případě každý neuron vrstvy dostává vstupní vektor $\mathbf{x}$ i výstupní vektor v předešlém časovém kroku $\mathbf{y}(t-1)$. Ilustrace rekurentní vrstvy se třemi rekurentními neurony je na obrázku 1.6 vlevo, vpravo je pak ilustrována stejná vrstva rozvinutá v čase. Pro rekurentní vrstvu o N neuronech pak platí rovnice udávající výstup

$$\mathbf{y}(t) = f\left(\mathbf{x}(t)\mathbf{W}_x + \mathbf{y}(t-1)\,\mathbf{W}_y + \mathbf{b}\right) = f\left(\begin{bmatrix}\mathbf{x}(t) & \mathbf{y}(t-1)\end{bmatrix}\begin{bmatrix}\mathbf{W}_x \\ \mathbf{W}_y\end{bmatrix} + \mathbf{b}\right) \quad (1.23)$$

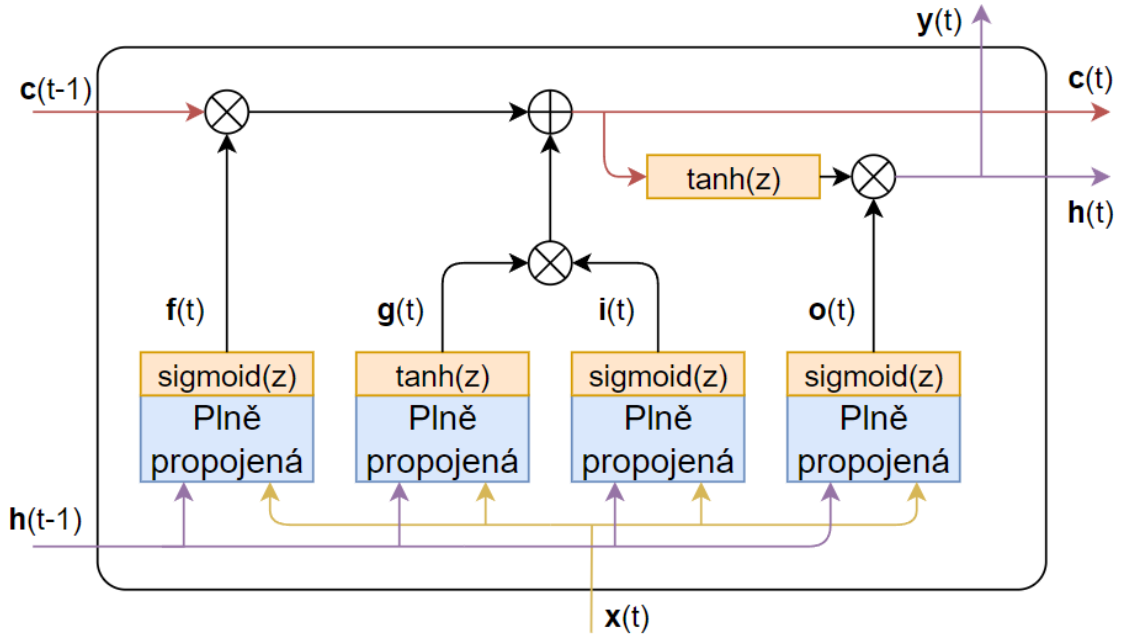


(a) Rekurentní vrstva (b) Rekurentní vrstva rozvinutá v čase

Obr. 1.6: Rekurentní vrstva

kde $\mathbf{y}(t) \in \mathbb{R}^{\mathrm{N}}$ je výstup pro časový krok t, $\mathbf{x}(t) \in \mathbb{R}^{\mathrm{M}}$ je vstupní vektor o M prvcích v časovém kroku t, $\mathbf{W}_x \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ a $\mathbf{W}_y \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ jsou matice vah vrstvy, $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$ je bias vrstvy a f je aktivační funkce vrstvy. Tím, že v tomto vztahu nefiguruje nijak délka sekvence vstupních vektorů, je potvrzeno tvrzení, že rekurentní vrstvy mohou zpracovat sekvence různých délek.

Z rovnice definující základní rekurentní vrstvu 1.23 je zjevné, že s každým zpracovaným časovým krokem dojde ke ztrátě informace z dřívějších časových kroků. Z tohoto důvodu jsou tyto jednoduché rekurentní vrstvy vhodné pouze pro zpracování velmi krátkých sekvencí. Tento problém řeší použití *Long Short-Term Memory* (LSTM) [29] buněk namísto rekurentního neuronu. Ilustrace LSTM struktury je na obrázku 1.7. Z vnějšího pohledu oproti rekurentnímu neuronu přibývá k výstupu



Obr. 1.7: LSTM buňka

buňky z minulého časového kroku $\mathbf{h}(t)$, také další stavový výstup $\mathbf{c}(t)$. Tyto stavy reprezentují informace uchovávané v krátkodobé paměti $\mathbf{h}(t)$ a dlouhodobé paměti $\mathbf{c}(t)$. Hlavní ideou funkcionality buňky je její schopnost naučit se, jakou informaci uchovat v dlouhodobé paměti, kterou informaci z paměti odstranit a jak informaci transformovat. Ve struktuře buňky lze vidět čtyři plně propojené vrstvy, ato vrstvy transformující vektor $\begin{bmatrix}\mathbf{x}(t) & \mathbf{h}(t-1)\end{bmatrix}$ na vektory:

- $\mathbf{f}(t)$ - vrstva určující, kterou informaci dlouhodobé paměti ponechat a kterou zapomenout.
- $\mathbf{g}(t)$ - vrstva transformující informaci z krátkodobé paměti a vstupního vektoru do informace dlouhodobé paměti.
- $\mathbf{i}(t)$ - vrstva určující, kterou informaci z krátkodobé paměti a vstupního vektoru zakomponovat do dlouhodobé paměti a kterou ignorovat.
- $\mathbf{o}(t)$ - vrstva určující, kterou informaci z dlouhodobé paměti ponechat v krátkodobé paměti a přenést na výstup.

V každém časovém kroku je na základě informace ze vstupního vektoru a krátkodobé paměti $\mathbf{f}(t)$ odstraněna z dlouhodobé paměti její nepodstatná část, dále je do dlouhodobé paměti přidána podstatná část (rozhodnuto podle $\mathbf{i}(t)$) informace z $\mathbf{g}(t)$. Z dlouhodobé paměti je pak její důležitá část (rozhodnuto podle $\mathbf{o}(t)$) transformovaná na výstup a uložena v krátkodobé paměti. Jinými slovy, buňka je schopná naučit se, který vstup je důležitý, uložit jej v dlouhodobé paměti, uchovat jej v paměti, dokud je potřeba a v případě potřeby jej promítnout na výstup. Matematicky lze buňku popsat několika rovnicemi definující vztah mezi vstupy a výstupy buňky

$$\begin{aligned}
\mathbf{f}(t) &= \sigma(\mathbf{x}(t)\cdot\mathbf{W}_{xf} + \mathbf{h}(t-1)\cdot\mathbf{W}_{hf} + \mathbf{b}_f)\\
\mathbf{g}(t) &= \tanh(\mathbf{x}(t)\cdot\mathbf{W}_{xg} + \mathbf{h}(t-1)\cdot\mathbf{W}_{hg} + \mathbf{b}_g)\\
\mathbf{i}(t) &= \sigma(\mathbf{x}(t)\cdot\mathbf{W}_{xi} + \mathbf{h}(t-1)\cdot\mathbf{W}_{hi} + \mathbf{b}_i)\\
\mathbf{o}(t) &= \sigma(\mathbf{x}(t)\cdot\mathbf{W}_{xo} + \mathbf{h}(t-1)\cdot\mathbf{W}_{ho} + \mathbf{b}_o)\\
\mathbf{c}(t) &= \mathbf{f}(t)\otimes\mathbf{c}(t-1) + \mathbf{i}(t)\otimes\mathbf{g}(t)\\
\mathbf{y}(t) &= \mathbf{h}(t) = \mathbf{o}(t)\otimes\tanh(\mathbf{c}(t))
\end{aligned} \tag{1.24}$$

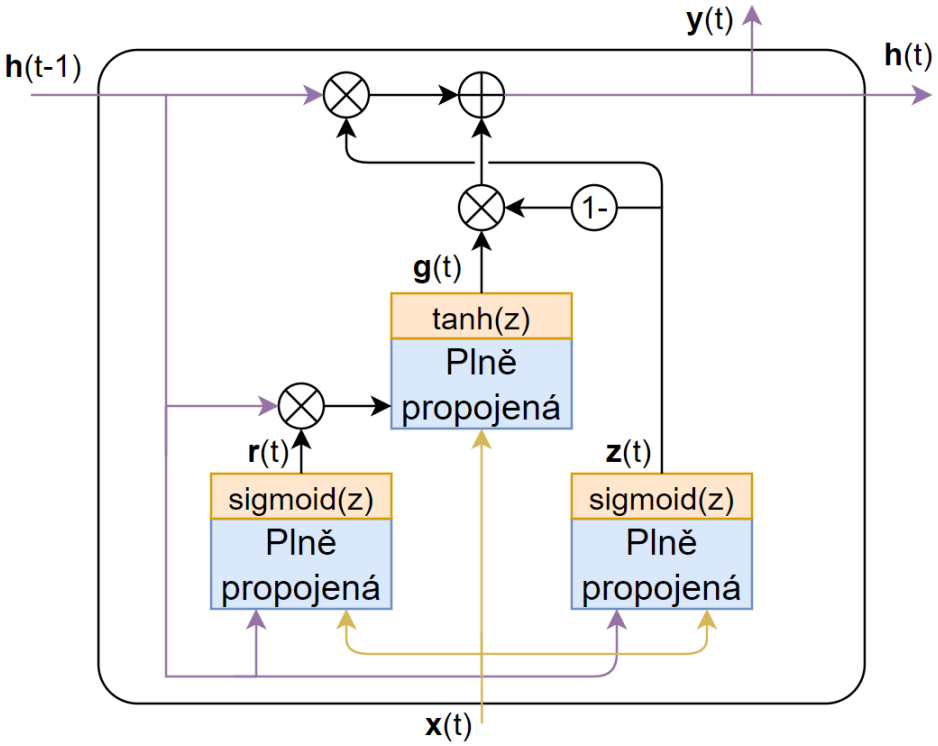
kde $\mathbf{x}(t) \in \mathbb{R}^{\mathrm{M}}$ je vstupní vektor sekvence o délce M, $\mathbf{W}_{xf}, \mathbf{W}_{xg}, \mathbf{W}_{xi}, \mathbf{W}_{xo} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ jsou váhy jednotlivých plně propojených vrstev o N neuronech napojených na vstupní vektor $\mathbf{x}(t)$, $\mathbf{W}_{hf}, \mathbf{W}_{hg}, \mathbf{W}_{hi}, \mathbf{W}_{ho} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ jsou váhy jednotlivých plně propojených vrstev neuronech napojených na vektor krátkodobé paměti z předešlého časového kroku $\mathbf{h}(t-1)$ a $\mathbf{b}_f, \mathbf{b}_g, \mathbf{b}_i, \mathbf{b}_o \in \mathbb{R}^{\mathrm{N}}$ jsou biasy jednotlivých vrstev.

Zjednodušenou variantou LSTM buňky je *Gated Recurrent Unit* [30] (GRU) buňka, která dosahuje přibližně stejných výsledků jako LSTM vrstvy. Mezi hlavní simplifikace patří

- sjednocení obou LSTM stavových vektorů $\mathbf{c}(t), \mathbf{h}(t)$ do jednoho vektoru $\mathbf{h}(t)$

- sjednocení dvou kontrolních hradel $\mathbf{f}(t)$ a $\mathbf{i}(t)$ do jednoho hradla $\mathbf{z}(t)$
- odstranění výstupního hradla $\mathbf{o}(t)$

Ilustrace GRU struktury je na obrázku 1.8. Ze struktury lze pozorovat hlavně změněnou konstrukci kontrolních vrstev. Vrstva formující kontrolní vektor $\mathbf{z}(t)$ kontroluje zapomínání informace ze stavového vektoru a načítání nové informace ze vstupního vektoru $\mathbf{x}(t)$ v případě, kdy $\mathbf{z}(t) \to 0$. Opačných výsledků pak dosahuje při $\mathbf{z}(t) \to 1$. Dále lze pozorovat, že i v případě, kdy $\mathbf{z}(t) \to 0$, mohou nově načítané informace do stavového vektoru obsahovat informaci ze stavového vektoru předchozího časového kroku $\mathbf{h}(t-1)$, ato na základě kontrolního vektoru $\mathbf{r}(t)$. Tím je zajištěna možnost uchování informace i zapamatování nové informace v jednom časovém kroku. Rovnice definující tuto buňku jsou ve tvaru



Obr. 1.8: GRU buňka

$$\begin{aligned}
\mathbf{r}(t) &= \sigma(\mathbf{x}(t) \cdot \mathbf{W}_{xr} + \mathbf{h}(t-1) \cdot \mathbf{W}_{hr} + \mathbf{b}_r) \\
\mathbf{z}(t) &= \sigma(\mathbf{x}(t) \cdot \mathbf{W}_{xz} + \mathbf{h}(t-1) \cdot \mathbf{W}_{hz} + \mathbf{b}_z) \\
\mathbf{g}(t) &= \tanh(\mathbf{x}(t) \cdot \mathbf{W}_{xg} + (\mathbf{r}(t) \otimes \mathbf{h}(t-1)) \cdot \mathbf{W}_{hg} + \mathbf{b}_g) \\
\mathbf{y}(t) &= \mathbf{h}(t) = \mathbf{z}(t) \otimes \mathbf{h}(t-1) + (1 - \mathbf{z}(t)) \otimes \mathbf{g}(t)
\end{aligned} \tag{1.25}$$

kde $\mathbf{x}(t) \in \mathbb{R}^{\mathrm{M}}$ je vstupní vektor sekvence o délce M, $\mathbf{W}_{xr}, \mathbf{W}_{xz}, \mathbf{W}_{xg} \in \mathbb{R}^{\mathrm{M} \times \mathrm{N}}$ jsou váhy jednotlivých plně propojených vrstev o N neuronech napojených na vstupní

vektor $\mathbf{x}(t)$, $\mathbf{W}_{hr}, \mathbf{W}_{hz}, \mathbf{W}_{hg} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ jsou váhy jednotlivých plně propojených vrstev neuronech napojených na stavový vektor z předešlého časového kroku $\mathbf{h}(t-1)$ a $\mathbf{b}_r, \mathbf{b}_z, \mathbf{b}_g \in \mathbb{R}^{\mathrm{N}}$ jsou biasy jednotlivých vrstev.

2 Návrh řešení

V rámci této kapitoly je proveden průzkum současného stavu metod rozpoznávání znakové řeči v oblasti výzkumu i v oblasti trhu. Dále je navržen způsob řešení úlohy a jsou stanoveny základní podmínky použití, ze kterých návrh vychází.

2.1 Metody rozpoznávání znakové řeči

Pro potřeby rozpoznávání znakové řeči je potřeba vycházet z nosičů informace znakové řeči, tedy pohybu, tvaru a pozice rukou a mimiky, pohybu a pozice obličeje. Z hlediska zpracování pomocí metod počítačového vidění, se bavíme zejména o problematice lokalizace a popisu rukou, obličeje a jejich pohybu a následné klasifikace. S vývojem výzkumu hlubokého učení v oblasti počítačového vidění v posledních letech bylo pomocí těchto technik dosaženo výrazně lepších výsledků [31], než použitím jiných metod z oblasti strojového učení, a to v různých oblastech počítačového vidění. Tyto výsledky se promítly také do oblastí související s problematikou rozpoznávání znakové řeči, jedná se především o úlohy detekce objektů, klasifikace, zpracování sekvenční informace a estimace pózy člověka. Při řešení úlohy rozpoznávání znakové řeči pouze z obrazové informace jsou v dnešní době používány převážně metody z oblasti hlubokého učení.

2.1.1 Dělení metod rozpoznávání znakové řeči

Metody, které jsou používány pro potřeby rozpoznávání obecných gest lze rozdělit podle různých kritérií [31]. Jako první rozdělení uveďme rozdělení podle druhu vstupní informace:

- RGB snímky
- Hloubkové snímky
- Kombinace RGB a hloubkových snímků
- IR snímky

Jedná se v podstatě o rozdělení podle použitého snímače, kde v případě kombinace RGB a hloubkových snímků se může jednat o kombinaci RGB a hloubkového snímače nebo o stereovidění kombinací více RGB snímačů. Ačkoliv jsou metody využívající IR informaci s výhodou použity v oblastech detekce člověka nebo rozpoznávání obličejů, nebylo použití tohoto druhu informace v této doméně příliš prozkoumáno a většina výzkumu v této oblasti se zaměřuje na použití RGB a hloubkové informace. Dále můžeme vstupní informace dělit na dvě podoblasti:

- Statická informace
- Dynamická informace

V případě statické informace se bavíme o statickém gestu nebo znaku, který není proměnlivý v čase, pro jehož rozpoznání je teoreticky potřeba pouze jeden snímek. Naopak dynamická vstupní informace nese informaci o dynamickém gestu, proměnlivém v čase. Pro rozpoznání je většinou potřeba dynamiku gesta dostatečně navzorkovat $n > 1$ snímky. Navzdory tomu, že za nositele znakové řeči jsou obecně označovány ruce a obličej, ne všechny metody se zaměřují na analýzu obou nosičů. Dělíme tyto metody na:

- Metody rozpoznávání znakové řeči analýzou rukou
- Metody rozpoznávání znakové řeči analýzou rukou a obličeje
- Metody rozpoznávání znakové řeči analýzou rukou, obličeje a lidského těla

2.1.2 Metody rozpoznávání znakové řeči analýzou rukou

Tyto metody využívají pouze analýzu rukou pro rozpoznání jednotlivých znaků, využívají pak poznatky z jednotlivých podoblastí [31] analýzy rukou, jedná se o detekci rukou v obraze, estimace pózy rukou, a rozpoznávání gest rukou.

Úloha lokalizace rukou v obraze je v dnešní době pojata jako úloha detekce objektů. Jedním z prvních objektových detektorů je dvoufázová architektura konvoluční neuronové sítě *Region-based Convolutional Neural Network - R-CNN* [32], která v první fázi ve snímku navrhne oblastí s potenciálním výskytem objektu a v druhé fázi jsou tyto oblasti klasifikovány. Další práce na této architektuře (*Fast-RCNN* a *Faster-RCNN*) [33, 34] byly zaměřeny primárně na její optimalizaci a redukci výpočetní náročnosti. Metodickým problémem dvoufázových detektorů je primárně v principu klasifikace navržených regionů, kde klasifikační algoritmus nemá k dispozici kontext okolo navrženého regionu, který může uchovávat důležité informace pro potřeby klasifikace. Pro tyto potřeby byly navrženy jednofázové detektory, které provádí obě fáze detekce a klasifikace najednou. Mezi první jednofázové detektory patří architektura *Single Shot Detector - SSD* [35] a také architektura *You Only Look Once - YOLO* [36] a její pozdější modifikace *YOLO9000* [37], *YOLOv3* [38], *YOLOv4* [39] a nepublikovaná verze *YOLOv5* [40]. V návaznosti na tyto architektury byla vyvinuta architektura *EfficientDet* [41], jejíž cílem bylo především dosažení efektivního využití parametrů architektury.

Úloha estimace pózy rukou může být pojata různými způsoby, může se jednat o estimaci 2D pózy, kde jsou estimovány klíčové body rukou pouze v oblasti snímkových souřadnic nebo 3D pózy, kde jsou dodatečně ke 2D souřadnicím klíčových bodů přidány estimace ve třetí, hloubkové dimenzi. Problémem estimace 3D pózy spočívá ve tvorbě datasetu. Anotace ve třetí dimenzi je náročná úloha i pro člověka, proto jsou často využívány synteticky vytvořené datasety [42, 43, 44, 45, 46], které ovšem reflektují realitu s určitým omezením. Alternativou je automatická anotace

ve scénách, kde je k dispozici více úhlů pohledu a výsledná anotace je pak získána triangulací 2D estimací [47, 48, 49], ovšem obdobně jako u syntetických snímků, snímky z předem připravených scén často představují pouze omezenou distribuci reálného světa. V této úloze je také s výhodou využívaná informace z hloubkového senzoru, buďto samostatně [50, 51, 52, 53, 54] nebo i v kombinaci s informací z RGB senzoru [55, 56, 57, 58, 59]. Výhodou těchto konfigurací je přímý přístup k hloubkové informaci přímo ze snímače a v případě použití kombinace RGB i hloubkového senzoru, je zvýšena celková informační hodnota vstupních dat do neuronové sítě. Použitím hloubkového senzoru jsou tedy eliminovány problémy s anotací hloubkových souřadnic v datasetech. Z pohledu architektury neuronové sítě řešící úlohu estimace pózy [60] může být úloha převedena na úlohu regrese, kde výstupem jsou 2D souřadnice, případně i hloubková informace, klíčových bodů rukou. Alternativou je generování teplotních map reprezentující pravděpodobnostní rozložení klíčových bodů. Nezávisle na použité metodě je úloha estimace pózy rukou hluboce prozkoumaná, ovšem v případech, kdy vstupní data vybočují z distribuce datasetu, na kterém je daný model naučen, modely často nevykazují nejlepší výsledky. Typicky snímek, ve kterém ruce interagují nebo jsou částečně v okluzi jsou pro modely velmi náročné [31].

Rozpoznávání obecných gest [31] přímo z RGB snímků je obecně realizováno pomocí konvolučních neuronových sítí. V některých případech jsou použity segmentační kroky před finálním klasifikačním krokem, a to buď Gaussovým barevným modelem lidské pleti [61] nebo další konvoluční neuronovou sítí, řešící úlohu sémantické segmentace [62]. Pro úlohy dynamického rozpoznávání gest [63] jsou s výhodou použity rekurentní neuronové sítě. Posledními trendy v oblasti hlubokého učení jsou mechanismy pozornosti, které byly také použity pro potřeby rozpoznávání gest rukou [64]. Možnost rozpoznávání gest v hloubkových snímcích konvolučními neuronovými sítěmi prozkoumali Kang a ostatní [65] na úloze klasifikace hláskovacích znaků. Nabízí se také metoda rozpoznávání gest z extrahované pózy rukou. Tuto možnost prozkoumali Isaacs a Foo [66], kde extrahovali 2D pózu rukou pro klasifikaci 24 statických znaků amerického znakového jazyka. V práci od Moryossefa a ostatních [67] se autoři zabývají detekcí znakového jazyka, tedy binární klasifikací zda-li uživatel znakuje. Autoři použili model estimující pózu člověka jako popis scény a použitím lineárního klasifikátoru a výpočtem optického toku z jednotlivých detekovaných klíčových bodů dosáhli vysoké přesnosti na testovacím subsetu.

2.1.3 Metody rozpoznávání znakové řeči analýzou rukou a obličeje

Navzdory tomu, že znaková řeč je převážně komunikována pomocí rukou a obličeje, není samostatná analýza obličeje a rukou následovaná fúzí příliš prozkoumaná. Obdobně jako v úloze analýzy rukou, můžeme přistoupit k lokalizaci rukou a obličeje jako k úloze detekce objektů. Dále stejně, jako můžeme estimovat klíčové body rukou, je možné estimovat klíčové body obličeje [68, 69, 70, 71]. Oproti estimaci klíčových bodů rukou, jsou datasety pro estimaci klíčových bodů obličeje pouze anotovány ve 2D souřadnicích. Alternativou je automatický popis scény konvoluční neuronovou sítí, kterou Rao a ostatní [72] použili k rozpoznávání indického znakového jazyka. V publikaci *Deep Learning of Mouth Shapes for Sign Language* [73] studují autoři nepříliš prozkoumanou oblast porozumění tvarům lidských úst pro účely rozpoznávání znakové řeči. Autoři použili konvoluční neuronovou sít učenou se slabým učitelem, bez explicitního definování třídy datových vzorků, použitím skrytých markovových modelů. Automatické učení znakové řeči byla představena také v práci od Pfistera a ostatních [74], kde ukázali, že tvar lidských úst je informačně bohatý, zejména pro potřeby oddělení znaků od sebe.

2.1.4 Metody rozpoznávání znakové řeči analýzou rukou, obličeje a lidského těla

Metody využívající informaci o celém lidském těle jsou z uvedených metod nejvíce sofistikované. Populární technikou je estimace pózy člověka, často se skládající z hrubé estimace pózy, následované zaměřenou estimací pózy rukou a klíčových bodů obličeje. Použití hrubé estimace pózy může přinést informační hodnotu klasifikačnímu modelu v případech kdy jsou hlavní nosiče informace zakryté nebo mezi sebou interagují [31].

Úlohu estimace pózy člověka [75] můžeme pojat jako úlohu estimace 2D nebo 3D souřadnic jednotlivých klíčových bodů člověka. Dále můžeme úlohu rozdělit podle způsobu zpracování na metody, kde jsou použity detektory člověka a pózy jsou estimovány pro každého detekovaného člověka samostatně (označovány jako *top-down* metody), a metody kde je estimace pózy provedena najednou pro všechny osoby na snímku (označovány jako *bottom-up* metody). Výhodou *bottom-up* metod je menší výpočetní náročnost pro scény, kde je estimována póza více osob, avšak oproti *top-down* metod ztrácí na přesnosti. Z hlediska dostupných datasetů, se pro 3D estimaci pózy člověka uplatňuje stejný problém jako při anotaci datasetu pro estimaci 3D pózy lidských rukou, manuální anotace je obdobně velmi náročná. Obdobně jako datasety pro 3D estimaci pózy rukou je k dispozici synteticky vytvořený dataset

[76] a dataset z přizpůsobených scén [77]. Dalším datasetem, vydaným v rámci práce *Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera* [78], je dataset obsahující video sekvence z reálného světa. Autoři pro anotaci těchto sekvencí využili 2D estimátor pózy, pohybující se kamery a informace z interciální jednotky. V rámci estimace 2D pózy člověka je k dispozici větší množství datasetů, např. [79, 80, 81, 82, 83].

Estimaci pózy člověka využili De Coster a ostatní [84] pro rozpoznávání znakové řeči. Autoři použili pro klasifikaci model z oblasti zpracování přirozené řeči, architekturu transformátoru s moduly pozornosti. Konstantinidis a ostatní v práci *A Deep Learning Approach for Analyzing Video and Skeletal Features in Sign Language Recognition* [85] používají fúzi pózy člověka, včetně obličeje a rukou pro účely rozpoznávání znakové řeči. Autoři v práci prozkoumávají různé kombinace fúzí dat z estimátorů póz a příznaků z konvoluční neuronové sítě pro dosažení vysoké přesnosti na testovacích subsetech. V další práci Li a ostatní [86] představují dataset pro rozpoznávání slov z amerického znakového jazyka a porovnávají výsledky různých metod rozpoznávání na svém testovacím subsetu. Autoři porovnávají výsledky z konvoluční neuronové sítě, 3D konvoluční neuronové sítě, rekurentní neuronové sítě s estimátory pózy člověka a grafovou neuronovou síť s estimátory pózy člověka. Autoři modely evaluovali na několika konfiguracích testovacího setu, nejlepších výsledků dosáhl model 3D konvoluční neuronové sítě. Gattupalli a ostatní [87] prozkoumali pomocí hrubé estimace pózy člověka rozptyl pohybu jednotlivých částí lidského těla při znakování. Z experimentů v této práci vyplývá podle očekávání, že při znakování lidé pohybují nejvíce dlaněmi, a naopak nejméně hlavou a rameny. Parelli a ostatní [88] prozkoumali ve své práci možnosti použití fúze pózy člověka, klíčových bodů obličeje, 3D projekcí pózy rukou a konvolučních příznaků z oblastí obličeje a rukou jako vstup pro pyramidovou architekturu typu enkóder-dekóder s moduly pozornosti pro potřeby rozpoznávání znakového jazyka. Autoři dosáhli SOTA výsledků evaluací na datasetu řecké znakové řeči a datasetu amerického znakového jazyka.

2.1.5 Shrnutí

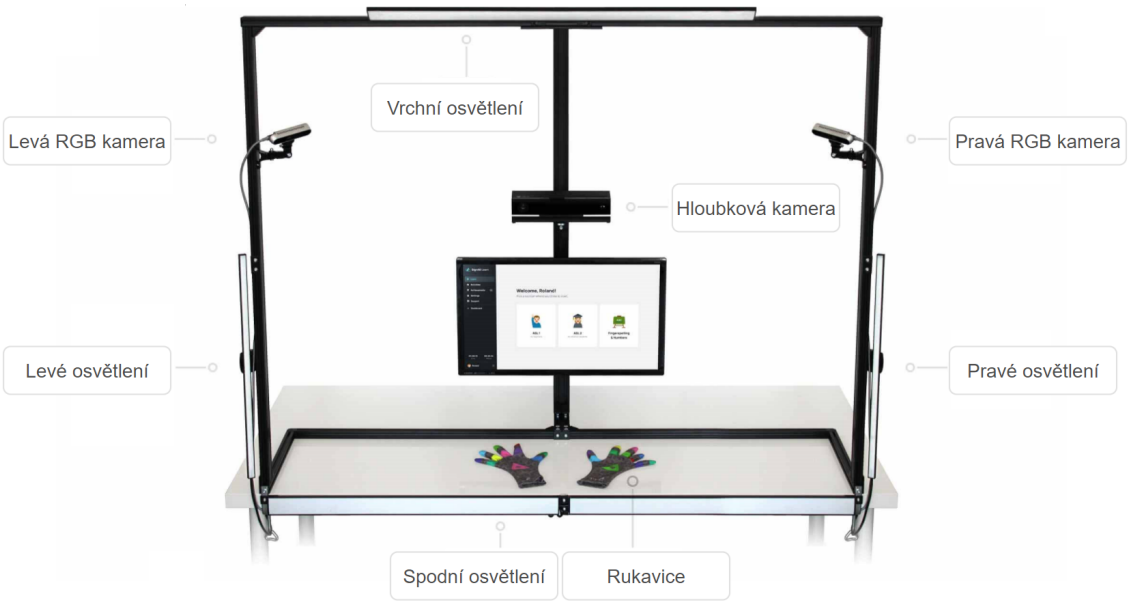
Úlohu rozpoznávání znakové řeči můžeme pojat jako úlohu sledování rukou, obličeje, lidského těla nebo kombinací těchto nosičů informace. S postupem výzkumu v oblasti estimace pózy člověka jsou tyto techniky aplikovány pro řešení úlohy rozpoznávání znakové řeči. Alternativní metodou je přímé použití konvolučních neuronových sítí na zpracovávaná data za účelem automatické extrakce příznaků. Z hlediska druhu použitého snímače, jsou z oblasti zpracovávání vizuální informace používány RGB a hloubkové senzory. Hloubkové senzory jsou s výhodou používány pro úlohy esti-

mace pózy člověka a speciálně rukou člověka. Výhodou použití hloubkových senzorů je i možnost estimovat pózy přímo ve 3D souřadnicích, použití RGB snímku pro 3D estimaci je stále předmětem výzkumu. Použití hloubkových senzorů pro zpracování vstupní informace, např. estimace pózy, se však potýká s nevýhodou nedostupnosti datasetů pro účely rozpoznávání znakové řeči, které jsou převážně distribuovány v RGB formátu. V rámci rozpoznávání dynamických znaků je potvrzena efektivnost rekurentních neuronových sítí a modulů pozornosti, které jsou široce využívány pro zpracování sekvenční informace v oblasti zpracování přirozeného jazyka.

2.2 Existující aplikace rozpoznávání znakové řeči

Existujících aplikací řešící úlohu automatického rozpoznávání znakové řeči, které se pohybují na trhu, není mnoho.

Společnost *SignAll Technologies* [89] nabízí produkty související se vzdálenou komunikací slyšícího a neslyšícího, včetně překladu znakové řeči do textové formy. Podmínkou použití je však sestavení definované pracovní stanice (obrázek 2.1), součástí které je hloubková kamera, dvě RGB kamery a osvětlení. Je také požadováno, aby interpret využíval uživatelské rukavice při interpretaci.



Obr. 2.1: Stanice pro použití *SignAll* systému [89]

Další společnosti zabývající se touto problematikou pravděpodobně ukončily činnost. Společnost *Evalk* pracovala na produktu *GnoSys* [90], který měl překládat znakový jazyk na mobilních zařízeních do mluveného jazyka, avšak krom novinářských

článků z roku 2018 není o společnosti zmínka a jejich webové stránky nejsou dostupné, předpokládáme tedy, že společnost není aktivní. Další takovou společností je *MotionSavvy* [91], která pracovala na stejnojmenné aplikaci pro vlastní hardware, ovšem obdobně jako webové stránky společnosti *Evalk*, webové stránky *MotionSavvy* nejsou aktivní a další informace nejsou dohledatelné, předpokládáme, že společnost ukončila činnost.

Úloha rozpoznávání znakové řeči není stále na trhu příliš rozšířená. Rozšířenější úlohou související s touto problematikou je vyhledávání znaků na základě vstupní textové nebo hlasové informace, kde drtivá většina produktů pracuje pouze s hláskovací abecedou daného znakového jazyka. Za zmínku stojí aplikace *HandTalk* [92], která na základě mluveného slova provádí interpretaci v různých znakových jazycích pomocí počítačového modelu.

2.3 Hardware a pracovní scéna

V návaznost na rozšiřující se vývoj metod estimace pózy z RGB snímků bude v rámci řešení práce použita jedna RGB kamera. Výhodou této konfigurace je nízká cena oproti zařízením s hloubkovým snímačem nebo větší potenciál pro použití v reálném čase oproti stereo zařízením. Další možností je použití různých optických soustav pro přizpůsobení vstupní informace RGB snímače. Uvažujeme-li ovšem o konečném zařízení jako o mobilním zařízení nebo zařízení s integrovanou kamerou, je úprava optické soustavy pro uživatele značně nepraktická, proto těchto možností prozatím nevyužíváme.

Použitím RGB snímače je scéna více náchylná na variabilitu oproti použití hloubkového snímače. Možností je tedy přizpůsobení scény pro potřeby zpracování výstupní informace ze senzoru. Opět v návaznost na postup výzkumu algoritmů hlubokého učení zpracovávající obrazovou informaci, implicitně neomezujeme scénu použití. Výjimkou je omezení výskytu lidí, kteří nejsou považováni jako subjekt zpracování. V praxi to pak znamená, že použití navržených algoritmů je vhodné ve scénách, ve kterých se pohybují pouze subjekty zájmu.

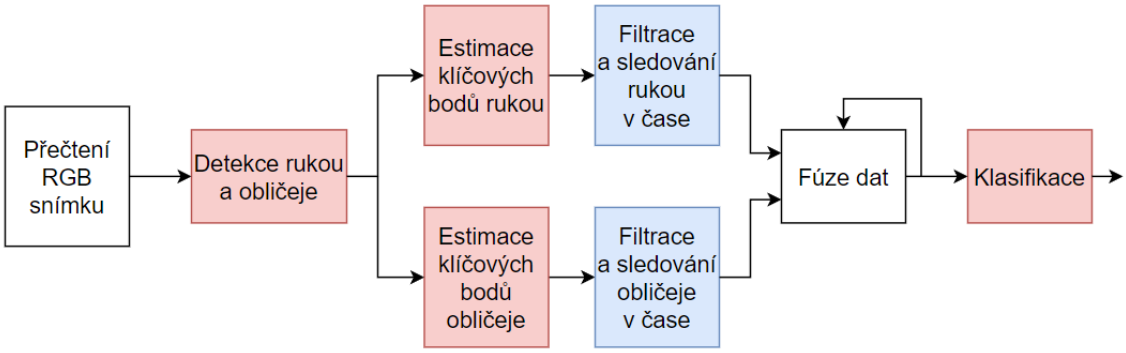
2.4 Řetězec pro rozpoznávání znakové řeči

Jednou z prerekvizit naučení robustní konvoluční neuronové sítě je dostatečně velká kapacita datasetu, na kterém je neuronová síť naučena. Tento fakt může být problematický pro naučení robustní konvoluční neuronové sítě pro přímou klasifikaci znaků v případě, kdy pro překládaný znakový jazyk nejsou data dostupná nebo nejsou dostupná v dostatečném měřítku. Úvahou je tedy vstupní snímky popsat,

v tomto případě popsat sledovaného interpreta estimací pózy, čímž silně zredukujeme vstupní dimenzionalitu klasifikátoru. Redukcí vstupní dimenzionality klasifikátoru také redukujeme potřebný počet parametrů klasifikátoru a také počet dat, potřebných k robustnímu naučení klasifikátoru. Navzdory tomu, že většina výzkumu zabývající se touto problematikou využívá estimaci pózy člověka jako celého, tedy estimace hrubé pózy člověka, rukou a klíčových bodů obličeje, v rámci této práce prozkoumáme použití pouze informací o obličeji a rukou interpreta. Jedním z důvodů nepoužití estimace hrubé pózy člověka je určitá redundance informace o pozici a pohybu ramen (podle [87]) a loktů, které jsou vázány na pohyb dlaní, které sledujeme. Další výhodou je eliminace kroku estimace hrubé pózy člověka, čímž je snížena výpočetní náročnost.

Navržený řetězec (pipeline) pro rozpoznávání interpretovaných znaků ze vstupních RGB snímků je ilustrován na obrázku 2.2. Prvním krokem zpracování je detekce rukou a obličeje v obraze, následuje estimace klíčových bodů detekovaných rukou a obličeje. Po obou krocích estimace následuje krok filtrace chyb detekce a estimace póz a sledování detekovaných objektů v čase. Dalším krokem je fúze dat, výstupem této fúze je popis vstupního snímku klíčovými body rukou a obličeje interpreta, pomocí kterého je provedena klasifikace, která je posledním krokem zpracování. K povšimnutí je zavedená zpětná vazba bloku fúze dat, tato vazba plyne z toho, že sledovaná scéna je dynamická a ke klasifikaci je vyžadována informace o sledovaném objektu z T_{steps} předchozích časových kroků. V algoritmu je zmíněn proces filtrace chyb, tímto je myšleno použití prediktivního modelu pro odfiltrování chybných detekcí nebo v případě chybějící detekce pro predikci stavu a pro filtraci estimovaných klíčových bodů. Nutnost použití tohoto kroku spočívá také ve faktu, že interpretace znakového jazyka je silně dynamická úloha a při použití komerčně dostupných kamer, které často snímají s frekvencí 30 snímků za vteřinu, dochází k distorzi dynamických částí snímků do stavu, kdy je úloha estimace pózy téměř neměřitelná. Součástí tohoto bloku je také sledování objektů v čase, tato úloha řeší problematiku asociace detekovaných objektů s objekty detekovanými v předchozích časových krocích. Nutnost použití takového algoritmu spočívá hlavně v zamezení logickému prohazování pořadí příznaků jednotlivých objektů ve finálním příznakovém vektoru mezi časovými kroky pro potřeby klasifikace.

Pro detekci rukou a obličeje obraze je použit objektový detektor na bázi konvoluční neuronové sítě. Obecná architektura neuronových sítí pro estimaci klíčových bodů rukou a obličeje je přejata z publikace *InterHand2.6M: A Dataset and Baseline for 3D Interacting Hand Pose Estimation from a Single RGB Image* [47]. Jedná se o pyramidovou architekturu typu enkóder-dekóder, skládající se z konvoluční části (enkóder) a následně dekonvoluční části (dekóder). Výstupem těchto sítí jsou teplotní mapy pozic jednotlivých klíčových bodů ve vstupním snímku. Pro potřeby



Obr. 2.2: Navržený řetězec pro rozpoznávání znakové řeči

sledování detekovaných rukou a obličejů v čase a filtrace chyb detekcí a estimací klíčových bodů je použit algoritmus *Deep SORT* [93], využívající kombinaci maďarského algoritmu a lineárního Kalmanova filtru pro úlohu asociace a filtraci chyb. Pro potřeby klasifikace je využita jednoduchá rekurentní neuronová síť klasifikující sekvenci příznakových vektorů mezi rozpoznávané znaky daného znakového jazyka.

2.5 Aplikace navrženého systému

Aplikací navrženého systému rozpoznávání znakové řeči interpreta je mnoho, očividnou aplikací je např. automatický překlad znakovaného jazyka do textového formátu, který řeší problematiku jazykové bariéry mezi znakujícím a osobou, která není zdatná ve specifickém znakovém jazyce, ve kterém interpret znakuje.

V rámci této práce je pozornost nasměrována na problematiku jazykové bariéry mezi dvěma znakujícími, kteří jsou zdatní v různých znakových jazycích. Jedním z řešení je automatická simulace interpretace znaků pomocí počítačové grafiky. Výsledkem je počítačový avatar, který rozpoznané znaky v jazyce původního interpreta znakuje v cílovém znakovém jazyce. V rámci práce je tedy namodelován počítačový avatar, pro který jsou animovány interpretace vybraných znaků.

Konečná vize této aplikace, je taková, že uživatel sledující interpretaci počítačového modelu může interpretaci přepnout do jakéhokoliv znakového jazyka. Nutno však dodat, že tato finální forma je silně omezena nutností animovat jednotlivé znaky v různých znakových jazycích.

2.6 Vzdálená komunikace a přenos dat

Jednou z možných aplikací navrženého řetězce zpracování dat je vzdálená komunikace slyšícího s neslyšícím nebo komunikace neslyšícího s neslyšícím, kde oba zna-

kující komunikují znakovými jazyky. V obou případech je při úspěšném rozpoznání znaků možné přenášet mezi komunikujícími pouze informaci o aktuálním znaku a přenos obrazové informace se stává do jisté míry redundantním. Tento přístup ovšem zanedbává fundamentální výhodu znakového jazyka oproti psané formě jazyka, a to možnost mimikou nebo obecně pohybem interpreta přidat do komunikace emocionální prvky. Odstraněním obrazové informace z komunikace se pak komunikace přibližuje komunikaci psané, ovšem faktem zůstává, že uživateli znakového jazyka je často bližší komunikace znakovou formou než psanou formou.

Navzdory tomu, že existují systémy konvertující znaky do tištěné podoby [94], nejsou tyto systémy příliš rozšířené a jsou definované pouze pro některé znakové jazyky. V oblasti znakových jazyků, tedy chybí standardizovaná digitální forma jednotlivých znaků typu *Unicode* [95]. V rámci přenosu informace o znaku je možné využít vlastnosti rozpoznávacího systému, na jehož výstupu je klasifikační algoritmus. V tomto případě se nabízí přiřazení každé výstupní třídy k identifikačnímu číslu, které pak může být přeneseno komunikačním protokolem. V případě komunikace slyšícího s neslyšícím by bylo potřeba na konečném zařízení dekódovat třídu pomocí vyhledávací tabulky, která by odpovídala výstupu z klasifikátoru. V případě komunikace neslyšícího s neslyšícím, kde oba znakující komunikují znakovými jazyky je možné opět dekódovat třídu pomocí vyhledávací tabulky nebo automaticky simulovat interpretaci znaku pomocí počítačové grafiky obdobně, jako bylo již popsáno v podkapitole 2.5.

Při použití enkódování znaku do identifikačního čísla je formát enkódování a následného dekódování závislý na formě datasetu, na kterém je naučen klasifikátor. Obecný datový formát by pak byl ve tvaru

$$\langle header \rangle \langle data \rangle \langle footer \rangle$$

kde $\langle header \rangle$ a $\langle footer \rangle$ jsou hlavička a patička daného komunikačního protokolu (např. TCP) a blok $\langle data \rangle$ obsahuje informaci o identifikačním čísle právě rozpoznaného znaku.

3 Řešení práce

V této kapitole jsou rozebrány podrobnosti ohledně realizace algoritmů detekce rukou a obličeje v obraze, estimace klíčových bodů rukou a obličeje člověka, filtrace chyb a sledování rukou a obličejů v čase a klasifikace znakového jazyka. V rámci řešení jsou použity algoritmy hlubokého učení, pro implementaci je použit programovací jazyk *Python 3* [96] a *opensource* knihovna *TensorFlow 2* [97]. Dále jsou popsány podrobnosti ohledně aplikace automatického překladu z interpretovaného znakového jazyka do cílového znakového jazyka počítačovým modelem.

3.1 Detekce rukou a obličeje

Úloha lokalizace rukou a obličeje v obraze je pojata jako úloha detekce objektů. Pro vyřešení této úlohy byly implementovány objektové detektory *YOLOv5* [40] a *EfficientDet* [41]. Z hlediska dostupných datasetů pro řešení úlohy jsou k dispozici pouze datasety se samostatnými anotacemi rukou a datasety se samostatnými anotacemi obličejů. V rámci datasetů pro estimaci pózy člověka jsou v některých případech přidány anotace polohy a lokalizace rukou i obličeje, ale často nejsou anotovány všechny lidské subjekty na snímku.

Řešením může být sekvenční použití detektoru rukou a následně detektoru obličeje. Tato varianta je ovšem výpočetně neefektivní, obzvláště přihlédneme-li k faktu, že objektové detektory jsou navrženy pro detekci objektů různých tříd. Alternativou je naučení samostatných detektorů, pomocí kterých budou křížově doplněny chybějící anotace.

3.1.1 Objektové detektory

Před popisem jednotlivých úloh detekce je potřeba popsat funkci použitých detektorů *YOLOv5* a *EfficientDet*. Nutno dodat, že detektory *YOLOv5* a *EfficientDet* nejsou přímo detektory, ale rodiny detektorů obsahující několik konfigurací, převážně pro účely ladění velikosti modelu. V rámci detektorů *YOLOv5* jsou implementovány konfigurace *YOLOv5-S*, *YOLOv5-M* a *YOLOv5-L*. Z rodiny detektorů *EfficientDet* jsou implementovány konfigurace *EfficientDet-D0*, *EfficientDet-D1* a *EfficientDet-D2*.

Z *black-box* pohledu, fungují oba detektory na stejném principu. Vstupní informací je RGB snímek, který je rozdělen podle N mřížek. Pro detektory *YOLOv5* platí $N = 3$ a pro detektory *EfficientDet* platí $N = 5$. Velikost jednotlivých mřížek je pak závislá na velikosti vstupního snímku. V rámci práce jsou vstupní snímky voleny s velikostí 512×512 pixelů, pro tento případ platí velikosti mřížek *YOLOv5*

detektorů 32×32, 64×64 a 128×128 pixelů, pro *EfficientDet* detektory pak 4×4, 8×8, 16×16, 32×32 a 64×64 pixelů. Velikost mřížky souvisí s velikostí detekovaných objektů, kde největší objekty jsou detekovány na největší mřížce. V každém poli každé mřížky je detekováno až M objektů, kde M značí počet *anchor boxů*, které zajišťují detekci objektů s různými poměry stran. Výstupem neuronové sítě je pak N mřížek, kde v každém poli mřížky se nachází M potenciálních detekcí. Každá potenciální detekce je reprezentována vektorem

$$\hat{\mathbf{y}}_{n_i,m} = [x, y, w, h, obj, \mathbf{class}] \tag{3.1}$$

kde $\hat{\mathbf{y}}_{n_i m}$ je predikovaný vektor pro pole i mřížky n a *anchor boxu* m, $x \in \langle 0, 1\rangle$ a $y \in \langle 0, 1\rangle$ jsou souřadnice středu predikovaného objektu, $w \in \langle 0, 1\rangle$ je šířka predikovaného objektu a h je výška predikovaného objektu, $obj \in \langle 0, 1\rangle$ je příznak pravděpodobnosti prezence objektu a $\mathbf{class} \in \langle 0, 1\rangle^C$ je vektor pravděpodobnosti příslušnosti objektu do jednotlivých tříd, kde C je celkový počet tříd. Výstupem lokalizace objektů v obraze jsou tedy obdélníky definované středovými kartézskými souřadnicemi v obraze a jejich šířkou a výškou, kde všechny hodnoty jsou normalizovány na rozsah $\langle 0, 1\rangle$ a původní rozsah je definován velikostí vstupního snímku, pro tento případ $\langle 0, 512\rangle$. Z důvodů stability sítí při učení nejsou hodnoty x, y, w a h predikovány přímo, souřadnice x a y jsou predikovány jako *offset* oproti svému poli v mřížce a výška a šířka predikovaného obdélníku jsou predikované v logaritmickém měřítku v rozsahu velikosti příslušného *anchor boxu*, ke kterému je objekt přiřazen. Transformace predikovaných hodnot x, y, w a h do formátu, který očekáváme je dán rovnicemi:

$$\begin{aligned} \hat{x} &= \sigma(x) + c_x \\ \hat{y} &= \sigma(y) + c_y \\ \hat{w} &= a_w e^w \\ \hat{h} &= a_h e^h \end{aligned} \tag{3.2}$$

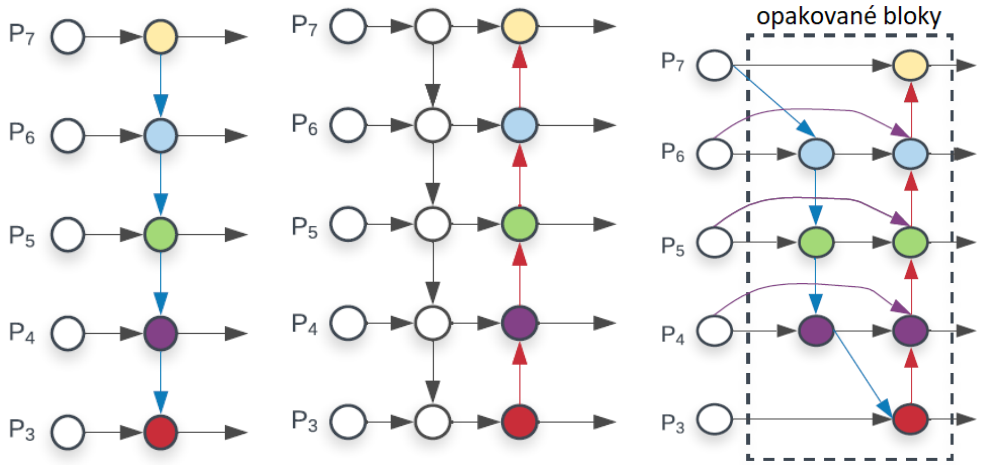
kde σ 1.11 je funkce sigmoidu, c_x a c_y jsou souřadnice levého horního rohu pole mřížky objektu a a_w a a_h jsou velikosti *anchor boxu* objektu. Chybová funkce sítě je pak definovaná jako chyba čtverců pro lokalizaci a velikost detekcí, entropie pro predikci příznaku prezence objektu a entropie pro klasifikaci objektu. Důležitým poznatkem je, že chyba není počítána z lokalizace, velikosti a třídy, pokud v daném zkoumaném poli není přítomen objekt.

Rozdíl mezi architekturami *YOLOv5* a *EfficientDet* spočívá hlavně ve vnitřním uspořádání vrstev. Porovnání těchto uspořádání je uvedeno v *EfficientDet* publikaci [41]. Architektury jednofázových objektových detektorů obecně obsahují dopřednou konvoluční část, pomocí které jsou extrahovány příznaky a následně na konvoluční část navazuje část, která je obecně nazývaná *Feature Pyramid Network* -

FPN. Klasická *FPN* je ilustrovaná na obrázku 3.1a, kde P_1 až P_7 reprezentují výstupy z konvoluční části. Vzhledem k tomu, že pro úlohu detekce objektů jsou obecně užitečné příznaky extrahované z různých částí sítě, byly prozkoumávané varianty *FPN* sloučení příznaků, např. obrázek 3.1b. Detektory architektury *YOLOv5* používají modifikovanou formu *FPN*. Autoři publikace *EfficientDet* pak představili obousměrnou architekturu *FPN* (obrázek 3.1c), kde jsou příznaky z jednotlivých vrstev kombinovány častěji. V publikaci také prozkoumávají různé metody sloučení příznaků, jelikož klasická a modifikovaná *FPN* příznaky pouze sčítá. Podle autorů nemají všechny příznaky stejnou váhu a jednou z variant, kterou navrhují je normalizovaný váhový součet podle rovnice

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i \tag{3.3}$$

kde $w_i \in \mathbb{R} \geq 0$ je váha mapy příznaků I_i a ϵ je koeficient zaručující numerickou stabilitu. Tato forma fúze příznaků je použita v rámci práce v implementaci architektur typu *EfficientDet*.



(a) *FPN* (b) Modifikovaná *FPN* (c) Obousměrná *FPN*

Obr. 3.1: Varianty *FPN* [41]

Vzhledem k nezávislé predikci prezence objektu mezi jednotlivými poli mřížky, dochází k současné detekci jednoho objektu na snímku více výstupními jednotkami. Důležitým krokem predikce ve fázi inference je proces označovaný jako *Non-Maximum Suppression - NMS*, jedná se o proces vyloučení predikcí, které detekují stejný objekt. Během tohoto procesu je počítána hodnota *Intersection Over Union - IOU*, která je definována vztahem

$$IOU = \frac{A \cap B}{A \cup B} \tag{3.4}$$

kde A a B jsou detekované objekty reprezentované ohraničujícími boxy. V případě, kdy hodnota IOU přesáhne hraniční hodnotu $t_{IOU} \in \langle 0, 1 \rangle$, je ponechána pouze detekce s větší hodnotou *score*, která je definovaná jako součin příznaku přítomnosti objektu a nejvyšší hodnoty pravděpodobnosti příslušnosti k třídě.

$$score = obj \cdot max(\mathbf{class}) \tag{3.5}$$

3.1.2 Detekce rukou v obraze

Cílem úlohy detekce rukou je naučení modelu, který bude použit pro anotaci výsledného datasetu detekce rukou a obličejů v obraze. Jako dataset řešící tuto úlohu byl zvolen dataset 100 Days of Hands [98]. V rámci tvorby datasetu autoři anotovali 27.3 tisíc YouTube videí běžných interakcí z 11 kategorií. Výsledným datasetem je zvolených 99 899 snímků s celkem 189 426 anotacemi polohy a velikostmi rukou. Autoři také přikládají k datasetu rozdělení mezi subsety pro učení, validaci a testování (tabulka 3.1). Dále z datasetu odstraníme anotace, které jsou menší než 0.16% celkového počtu pixelů snímku, vzhledem k tomu, že takové detekce nejsou pro řešenou úlohu podstatné.

	Učení	**Validace**	**Testování**
#Snímků	79 921	9 995	9 983

Tab. 3.1: Distribuce snímků z datasetu mezi subsety pro učení, validaci a testování

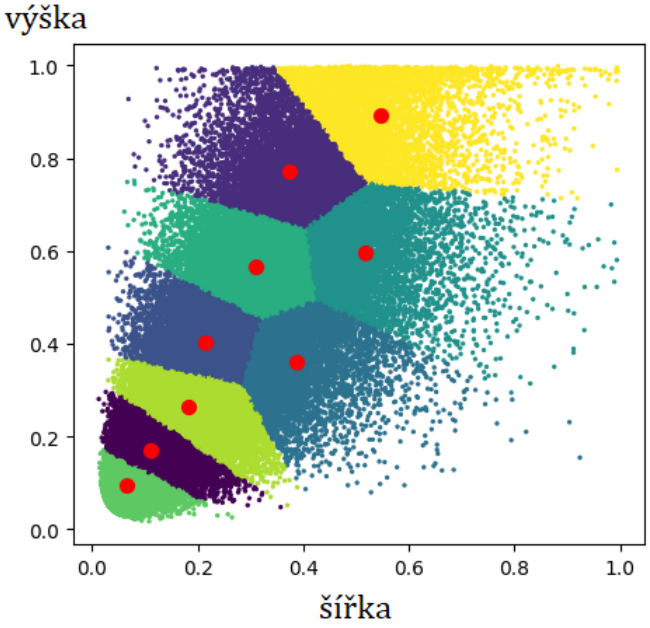
Jako architektura detektoru rukou byla zvolena architektura *YOLOv5-S* se třemi výstupy (počet mřížek $N = 3$) a třemi *anchor boxy* pro každý výstup ($M = 3$, $N = 3$, celkem tedy 9 unikátních *anchor boxů*). Stanovení velikostí anchor boxů je obecně uskutečněno pomocí *K-means* shlukovacího algoritmu. Jako shlukovaná data jsou zvoleny hodnoty výšky a šířky všech objektů v subsetu pro učení.

$$\mathbf{x}_{kmeans} = [\mathbf{x}_h^T, \mathbf{x}_w^T] \tag{3.6}$$

Hledáme 9 *anchor boxů*, tedy algoritmu *K-means* je stanoveno $k = 9$. Výsledné středy shluků pak odpovídají stanoveným *anchor boxům* (tabulka 3.2). Vizualizace výsledků *K-means* algoritmu je ilustrovaná na obrázku 3.2.

3.1.3 Detekce obličejů v obraze

Podobně jako detektor rukou, je detektor obličejů použit pro anotaci výsledného datasetu pro detekci rukou a obličeje člověka. K naučení takového detektoru je použit Wider Face dataset [99]. Dataset obsahuje 32 203 snímků s 393 703 anotacemi



Obr. 3.2: Vizualizace výsledných centroidů velikostí rukou *K-means* algoritmu

Mřížka #1		**Mřížka #2**		**Mřížka #3**	
Šířka	Výška	Šířka	Výška	Šířka	Výška
0.54	0.89	0.30	0.56	0.17	0.25
0.51	0.58	0.37	0.34	0.10	0.16
0.37	0.76	0.21	0.39	0.06	0.09

Tab. 3.2: Výsledné velikosti *anchor boxů* detektoru rukou

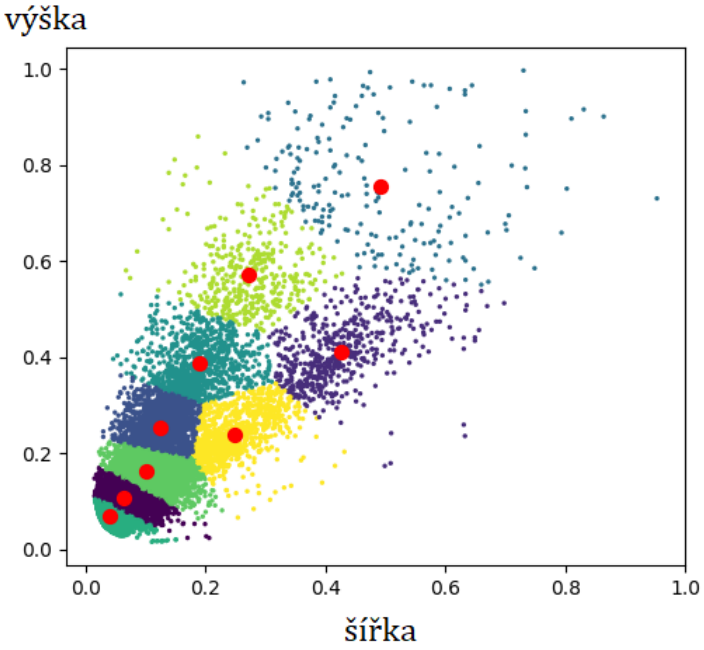
při různých příležitostech a aktivitách. Na rozdíl od 100 Days of Hands datasetu, autoři nezveřejnili anotace testovacího subsetu, k dispozici jsou tedy pouze anotace subsetů pro učení a validaci. Obdobně, jako pro detekci rukou, jsou odstraněny anotace, které jsou menší než 0.16% celkového počtu pixelů daného snímku.

	Učení	**Validace**
#Snímků	12 880	3 226

Tab. 3.3: Distribuce snímků z datasetu mezi subsety pro učení a validaci

Architektura modelu je zvolena stejná jako architektura modelu detekce rukou, tedy konfigurace *YOLOv5-S*. Obdobně algoritmem *K-means* dataset analyzujeme pro potřeby zjištění velikostí *anchor boxů* pro $k = 9$. Vizualizace výsledků *K-means*

algoritmu je ilustrovaná na obrázku 3.3 a výsledné velikosti *anchor boxů* jsou zaneseny do tabulky 3.4.



Obr. 3.3: Vizualizace výsledných centroidů velikostí obličejů *K-means* algoritmu

Mřížka #1		**Mřížka #2**		**Mřížka #3**	
Šířka	Výška	Šířka	Výška	Šířka	Výška
0.49	0.75	0.18	0.38	0.10	0.16
0.42	0.41	0.24	0.23	0.06	0.10
0.27	0.57	0.12	0.25	0.04	0.06

Tab. 3.4: Výsledné velikosti *anchor boxů* detektoru obličejů

3.1.4 Kombinovaná detekce rukou a obličeje

Pro potřeby současné detekce rukou i obličeje je dataset 100 Days of Hands doplněn o anotace obličejů pomocí naučeného detektoru a naopak dataset Wider Face je doplněn o anotace rukou pomocí naučeného detektoru rukou. Tímto způsobem neanotujeme testovací subset datasetu 100 Days of Hands a validační subset datasetu Wider Face. Tyto dva subsety jsou použity pro testovací účely a z důvodu zachování maximální kvality subsetu jsou ponechány pouze původní anotace. Jako subset pro učení je použita kombinace subsetů pro učení obou datasetů a pro validaci

je použit pouze validační set datasetu 100 Days of Hands. Počet snímků v jednotlivých subsetech je zanesen do tabulky 3.5 a zastoupení instancí rukou a obličejů v datasetu je zaneseno do tabulky 3.6.

	Učení	**Validace**	**Testování**
#Snímků	90 126	9 873	12 712

Tab. 3.5: Distribuce snímků z datasetu mezi subsety pro učení, validaci a testování

	Učení	**Validace**	**Testování**
#Rukou	163 585	18 249	17 858
#Obličejů	88 328	6 748	10 090

Tab. 3.6: Distribuce instancí tříd datasetu mezi subsety

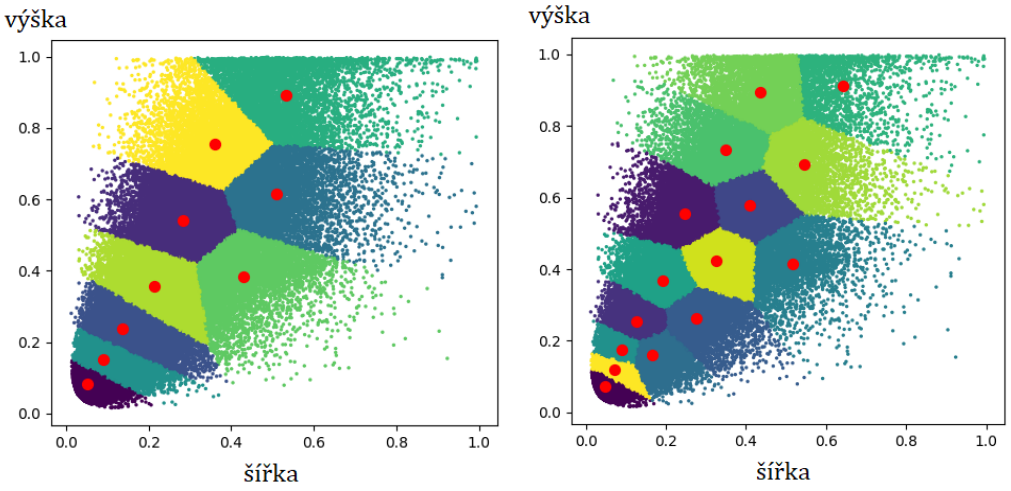
Pro účely detekce rukou a obličeje jsou naučeny detektory *YOLOv5-S*, *YOLOv5-M*, *YOLOv5-L*, *EfficientDet-D0*, *EfficientDet-D1* a *EfficientDet-D2*. Na rozdíl od jednotlivých detektorů samostatných rukou nebo obličejů, je potřeba provést dvě analýzy anotací pro stanovení *anchor boxů*. Pro *YOLOv5* architektury se jedná opět o *K-means* analýzu pro $k = 9$, pro *EfficientDet* architektury je potřeba stanovit 15 *anchor boxů* (model má 5 výstupů, je sestaveno 5 mřížek, 3 *anchor boxy* pro každé pole mřížek), tedy *K-means* pro $k = 15$. Výsledné velikosti *anchor boxů* jsou zaneseny do tabulek 3.7 a 3.8. Vizualizace *K-means* centroidů jsou ilustrovány na obrázcích 3.4.

Mřížka #1		**Mřížka #2**		**Mřížka #3**	
Šířka	Výška	Šířka	Výška	Šířka	Výška
0.53	0.89	0.42	0.38	0.13	0.23
0.50	0.61	0.28	0.54	0.09	0.15
0.36	0.75	0.21	0.35	0.05	0.08

Tab. 3.7: Výsledné velikosti *anchor boxů* architektur *YOLOv5*

3.1.5 Detaily procesu učení detektorů

Všechny uvedené detektory jsou optimalizovány algoritmem *adam* 1.10 s parametry $\beta_1 = 0.9$ a $\beta_2 = 0.999$. Koeficient učení v procesu učení je konstantní pro prvních



(a) *YOLOv5* (b) *EfficientDet*

Obr. 3.4: Vizualizace výsledných centroidů velikostí anotací *K-means* algoritmu

Mřížka #1		**Mřížka #2**		**Mřížka #3**		**Mřížka #4**		**Mřížka #5**	
Šířka	Výška	Šířka	Výška	Šířka	Výška	Šířka	Výška	Šířka	Výška
0.64	0.91	0.34	0.73	0.32	0.42	0.19	0.36	0.08	0.17
0.43	0.89	0.40	0.57	0.24	0.55	0.12	0.25	0.07	0.11
0.54	0.69	0.51	0.41	0.27	0.26	0.16	0.15	0.04	0.07

Tab. 3.8: Výsledné velikosti *anchor boxů* architektur *EfficientDet*

pět optimalizačních iterací a exponenciálně klesající pro iterace $k \geq 5$.

$$\eta(k) = \begin{cases} 10^{-3}, & \text{pro } k < 5 \\ 10^{-3}e^{-0.1k}, & \text{pro } k \geq 5 \end{cases}$$

Ukončovací podmínka pro optimalizaci je divergence validační chyby po dobu 5-ti optimalizačních iterací. Váhy všech modelů jsou iniciovány vahami předem naučenými na COCO [100] datasetu (přenos vah z jedné úlohy pro účely inicializace vah je v angličtině označován jako *transfer learning*).

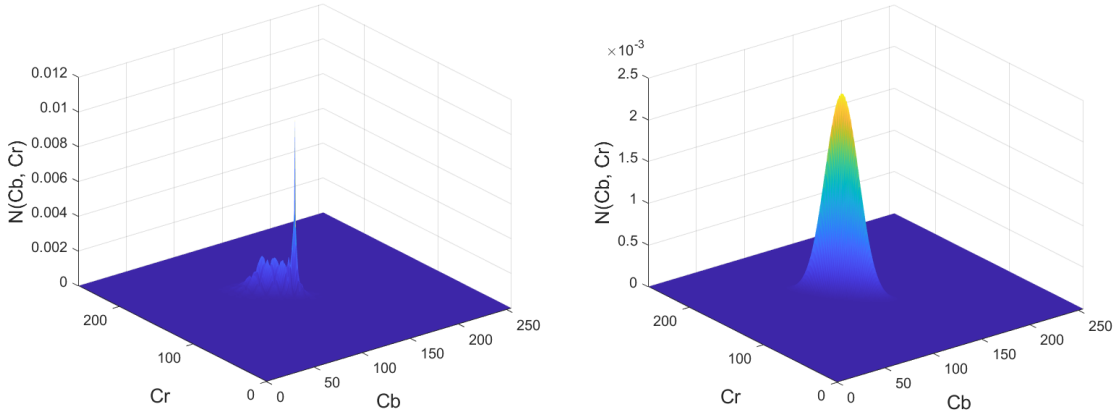
3.1.6 Limitace použitých dat

V posledních letech je vedeno více diskuzí na téma inkluzity dostupných datasetů pracující s obrazovou informací. V závislosti na použitých datasetech pro detekci částí lidského těla, tedy definujeme Gaussovu distribuci zastoupení barvy kůže na zá-

kladě informace o červené a modré chromacitě z formátu barev $YCbCr$.

$$\begin{gathered}\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \boldsymbol{\mu} = [\mu_{Cb}, \mu_{Cr}] = [116.42, 145.11] \\ \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{Cb^2} & \sigma_{CbCr} \\ \sigma_{CrCb} & \sigma_{Cr^2} \end{bmatrix} = \begin{bmatrix} 84.96 & -88.22 \\ -88.22 & 142.76 \end{bmatrix}\end{gathered} \tag{3.7}$$

Použití algoritmů realizovaných v rámci této práce je pak omezeno touto distribucí. Rozložení vzorků z datasetu v $CbCr$ doméně je ilustrováno na obrázku 3.5 vlevo, model gaussova rozložení na obrázku 3.5 vpravo.



(a) Distribuce barvy pleti datasetu

(b) Model barvy pleti

Obr. 3.5: Distribuce barvy pleti datasetu v $CbCr$ doméně a její model

Použité datasety omezují také rozměry detekovaných objektů. V rámci zpracování anotací a přiložených snímků datasetů byly odstraněny anotace, jejichž obsah je menší než 0.16% celkového obsahu snímku v prvních dvou dimenzích. Tento práh je naprosté minimum velikosti rukou nebo obličeje sledovaného subjektu v obraze.

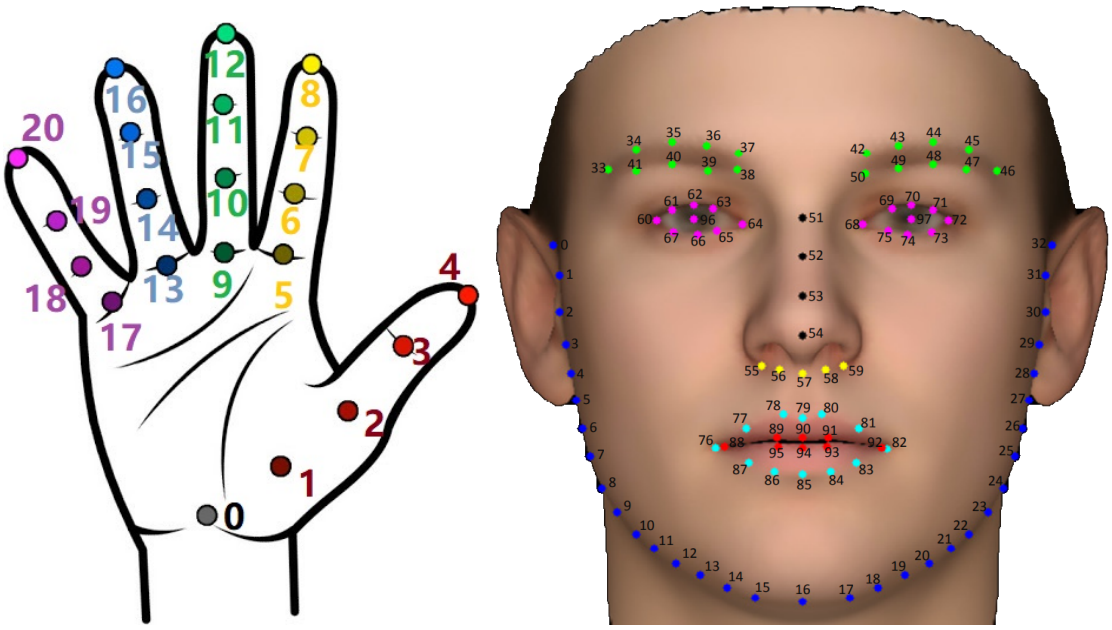
3.2 Estimace klíčových bodů

V rámci této úlohy jsou estimovány klíčové body rukou a obličeje. Klíčové body rukou i obličeje jsou definovány svým kinematickým modelem. Model rukou (obrázek 3.6 vlevo) obsahuje 21 klíčových bodů pro levou ruku a 21 bodů pro pravou ruku. Model obličeje (obrázek 3.6 vpravo) je pak definován 98 body. Modely neuronových sítí jsou konstruovány, tak že jejich výstupem jsou teplotní mapy znázorňující mapu *sebejistoty* estimace polohy bodů. V rámci procesu učení jsou anotované body transformovány na teplotní mapy s Gaussovým rozložením $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, kde $\boldsymbol{\mu}$ jsou

anotované klíčové body a $\mathbf{\Sigma}$ je kovarianční matice ve tvaru

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_c & 0 \\ 0 & \sigma_c \end{bmatrix}$$

kde za σ_c je zvolena konstanta 2.5.



Obr. 3.6: Kinematické modely pravé ruky [102] a obličeje [69]

3.2.1 Detaily procesu učení estimátorů

Naučené modely estimující klíčové body jsou naučené pomocí *SGD* optimalizátoru 1.6 s konstantním koeficientem učení v prvních pěti iteracích a exponenciálně klesajícím koeficientem v iteracích $k \geq 5$.

$$\eta(k) = \begin{cases} 10^{-3}, & \text{pro } k < 5 \\ 10^{-3}e^{-0.1k}, & \text{pro } k \geq 5 \end{cases}$$

Ukončovací podmínka pro optimalizaci je divergence validační chyby po dobu 10-ti optimalizačních iterací. Váhy *EfficientNet* extraktorů jsou iniciovány vahami předem naučenými na ImageNet [14] datasetu. Váhy dalších vrstev jsou iniciovány náhodně podle uniformního rozložení Glorota a Bengia [110]

$$\mathbf{w_i}(0) \sim U\left(-\sqrt{\frac{6}{P_{i,in} + P_{i,out}}}, \sqrt{\frac{6}{P_{i,in} + P_{i,out}}}\right) \tag{3.8}$$

kde $\mathbf{w_i}$ je tensor vah vrstvy i a $P_{i,in}$, $P_{i,out}$ jsou počty vstupních a výstupních jednotek tensoru vah $\mathbf{w_i}$.

3.2.2 Estimace pózy rukou

Prvním experimentem pro řešení této úlohy bylo použití předem naučeného modelu z publikace *InterHand2.6M: A Dataset and Baseline for 3D Interacting Hand Pose Estimation from a Single RGB Image* [47]. Jedná se o model estimující 3D pózu rukou z jednoho RGB snímku. Bohužel dataset, na kterém je model naučen, byl sestaven ve studiu a neodpovídá příliš dobře reálnému světu. Byly provedeny experimenty pro otestování modelu v nedefinovné scéně a následně i v upravené scéně, více odpovídající scéně z datasetu. Z výsledků těchto experimentů vyplynulo, že model není příliš vhodný pro reálné použití.

Vzhledem k tomu, že cílem estimace by měla být estimace pózy obou rukou i v případě, kdy spolu interagují, je velmi limitovaný výběr datasetů s anotacemi póz rukou. V rámci 3D anotací, jsou krom InterHand2.6M datasetu k dispozici pouze syntetické datasety. Z tohoto důvodu, estimujeme pouze 2D polohy klíčových bodů. V doméně datasetů obsahující 2D anotace klíčových bodů rukou, je stále omezené množství datasetů obsahující anotace pro případy kdy ruce interagují. V rámci řešení práce jsou použity datasety COCO-WholeBody [101] (dále označován jako COCO dataset), Halpe Full-Body Human Keypoints and HOI-Det [103, 104] (dále označován jako Halpe dataset), manuálně anotovaná část datasetu Hand Keypoint Detection in Single Images using Multiview Bootstrapping [105] (dále označován jako MWB dataset) a jeden syntetický dataset Rendered Handpose Dataset [43] (dále označován jako RHD dataset). Datasety COCO, MWB a RHD obsahují rozdělení datasetu mezi subsety pro učení a pro testování, tyto subsety pro testování jsou sloučeny do jednoho testovacího subsetu. Ze zbytku datasetů, tedy subsetů pro učení datasetů COCO a MWB, Halpe a RHD, je náhodně vybráno 10% datových vzorků, které tvoří subset pro validaci. Zbývající vzorky z datasetu jsou součástí subsetu pro učení. Výsledná distribuce snímků do jednotlivých subsetů je vynesena do tabulky 3.9. Dále můžeme prozkoumat zastoupenost interagujících rukou v jednotlivých subsetech (tabulka 3.10).

	Učení	**Validace**	**Testování**
#Snímků	105 931	11 830	6 581

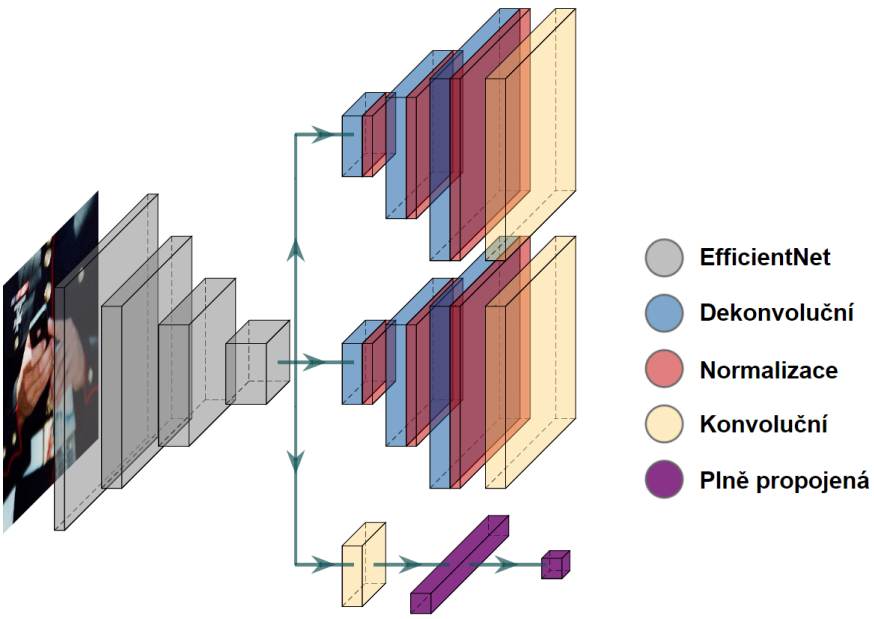
Tab. 3.9: Distribuce snímků z datasetu mezi subsety pro učení, validaci a testování

Navržená architektura modelu je inspirovaná podle modelu navrženém v publikaci k datasetu *Interhand2.6*. Jedná se o kombinaci extraktoru příznaků, dekonvolučních vrstev pro generaci teplotních map a plně propojených vrstev pro klasifikaci

	Učení	Validace	Testování
#Samostatných rukou	93 162	10 382	5 799
#Interagujících rukou	12 769	1 448	782

Tab. 3.10: Distribuce instancí tříd datasetu mezi subsety

mezi případy, kdy se jedná o samostatnou ruku nebo interagující ruce. Ilustrace architektury je na obrázku 3.7. Jako extraktory příznaků jsou zvoleny konvoluční neuronové sítě typu *EfficientNet* [106]. Klasifikační část sítě je složena z jedné konvoluční vrstvy o 256 filtrech, jedné *pooling* vrstvy, jedné plně propojené vrstvy o 128 neuronech a výstupní vrstvy o jednom neuronu. Dekonvoluční části jsou pak složené ze třech bloků dekonvoluční a normalizační vrstvy a následné konvoluční výstupní vrstvy. Dekonvoluční vrstvy větve estimující pózu jedné ruky obsahují 64 filtrů a dekonvoluční vrstvy větve estimující pózu interagujících rukou obsahují 128 filtrů.



Obr. 3.7: Model estimace pózy rukou

Chybová funkce je definovaná jako entropie pro klasifikační výstup a chyba čtverců pro výstupy teplotních map. Důležitým poznatkem je, že chyba je nulovaná pro výstup, který neodpovídá třídě daného vzorku, tedy pro vzorek samostatné ruky není počítána chyba z teplotních map interagujících rukou a naopak. V tabulce 3.10 lze pozorovat nevyváženost tříd v datasetu, kde třída *Samostatné ruce* má téměř 8× větší zastoupení v učícím subsetu, z tohoto důvodu jsou do výpočtu klasifikační chyby entropie přidány váhy jednotlivým třídám. Hodnoty vah jsou odvozené

ze zastoupení jednotlivých tříd v datasetu podle vztahu

$$\begin{aligned} w_i &= \frac{\sum_j N_j}{2 \cdot N_i} \\ w_{samostatne} &= 0.56 \\ w_{interagujici} &= 4.14 \end{aligned} \tag{3.9}$$

kde w_i je váha třídy i a N_i je počet vzorků v datasetu z třídy i. Výsledná rovnice pro chybu klasifikace je pak ve tvaru

$$L_{class} = y \cdot w_{interagujici} \cdot \log(\hat{y}) + (1 - y) \cdot w_{samostatne} \cdot \log(1 - \hat{y}) \tag{3.10}$$

kde $y \in \{0, 1\}$ je binární příznak interagujících rukou, $\hat{y}$ je predikce modelu a w jsou jednotlivé váhy tříd.

3.2.3 Estimace klíčových bodů obličeje

Úloha estimace klíčových bodů obličeje je jednodušší variantou estimace klíčových bodů rukou. Jako dataset pro naučení modelu je zvolen Wider Facial Landmarks in-the-wild - WFLW dataset [69]. Dataset obsahuje celkem 10 000 anotací za různých podmínek. Autoři rozdělili dataset do částí pro učení a testování. Subset pro validaci byl sestrojen náhodným výběrem 10% anotací ze subsetu pro učení. Počet snímků v jednotlivých subsetech jsou zaneseny do tabulky 3.11.

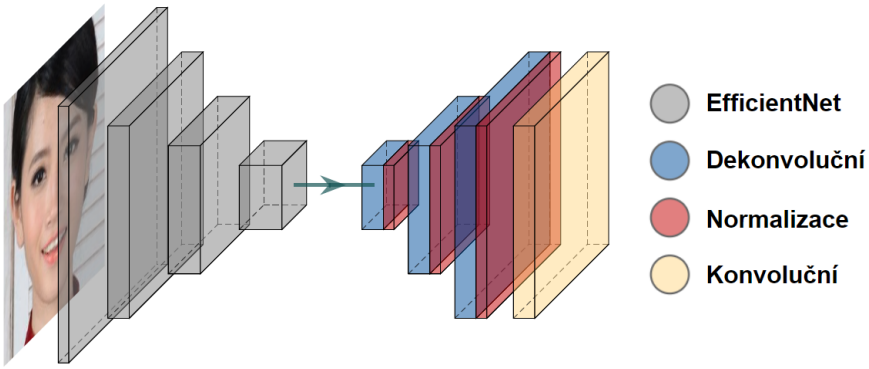
	Učení	**Validace**	**Testování**
#Snímků	6 750	750	2 500

Tab. 3.11: Distribuce snímků z datasetu mezi subsety pro učení, validaci a testování

Architektura modelu je obdobná modelu estimace pózy rukou. Jedná se o sekvenční model extraktoru příznaků a dekonvoluční části generující teplotní mapy. Ilustrace modelu je na obrázku 3.8. Jako extraktor příznaků jsou opět použity architektury typu *EfficientNet* a dekonvoluční část se skládá ze tří bloků dekonvoluční vrstvy o 64 filtrech a normalizační vrstvy a následné konvoluční výstupní vrstvy. Jako chybová funkce je zvolena chyba čtverců.

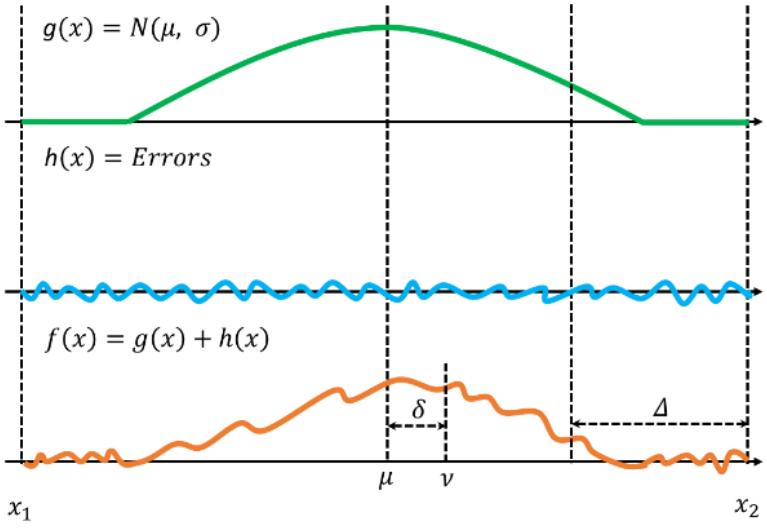
3.2.4 Dekódování teplotních map

Při použití enkódování výstupu modelů estimace pózy do teplotních map, je potřeba definovat metody dekódování této informace do souřadnicového systému snímku. Klasickým přístupem je výběr pixelu teplotní mapy s maximální hodnotou, který je



Obr. 3.8: Model estimace klíčových bodů obličeje

prohlášen jako poloha zkoumaného bodu. Problémem tohoto přístupu je diskrétní stanovení pozice, čímž jsou způsobeny nepřesnosti v případech, kdy teplotní mapy jsou menší v rozměrech než vstupní snímek. V rámci řešení práce jsou vstupní snímky modelů estimující klíčové body transformovány na velikost $256 \times 256 \times 3$ a výstupní teplotní mapy jsou ve tvaru $64 \times 64 \times K$, kde K je počet estimovaných bodů. V případě diskrétního stanovení polohy výsledných bodů, může výsledný bod odpovídat až čtyřem bodům ve vstupním snímku modelu, čímž vzniká určitá úroveň nejistoty.



Obr. 3.9: Reprezentace šumu v teplotních mapách na výstupu modelu [107]

Touto problematikou se mimo jiných zabývá publikace *Error Compensation Heatmap Decoding for Human Pose Estimation* [107], kde autoři představují metodu dekódování teplotních map pro použití v reálném čase. Autoři říkají, že pro distri-

buci s Gaussovým rozložením na výstupu modelu estimující teplotní mapy platí, že je ovlivněn uniformním šumem. Vyřazením části estimované teplotní mapy Δ, je pak kompenzována celková úroveň šumu. Na obrázku 3.9 je ilustrace úlohy převedené do jedné dimenze, kde funkce $g(x) \sim \mathcal{N}(\mu, \sigma)$ je ovlivněna neznámým aditivním šumem $h(x)$. Výsledný signál je pak ve tvaru funkce $f(x) = g(x) + h(h)$. Autoři uvádějí, že stanovení střední hodnoty je ovlivněno o chybu δ, která je úměrná úrovni šumu v oblasti Δ. Výsledný vztah pro stanovení μ je ve tvaru

$$\mu \approx \frac{\int_{x_1}^{x2-\Delta} x f(x)\, dx}{\int_{x_1}^{x2-\Delta} f(x)\, dx} \tag{3.11}$$

kde μ je výsledná poloha zkoumaného bodu, x_1 a x_2 jsou souřadnice ohraničující distribuci a Δ je velikost oblasti map definující úroveň šumu v distribuci. Autoři uvádějí optimální $\Delta = 3$ pro optimální výsledky na COCO datasetu.

3.3 Filtrace chyb a sledování objektů v čase

V rámci zpracování dynamické informace je vhodné krom statické analýzy každého snímku, také analyzovat dynamické vlastnosti scény. Pro tyto potřeby je použit algoritmus *Deep SORT* [93], rozšíření autory původně publikovaného algoritmu *SORT* [108].

3.3.1 Kalmanův filtr

Před diskuzí použití algoritmů pro řešení problematiky této práce definujeme použitý Kalmanův filtr [109] pro filtraci měření a estimace nejistoty měření. Obecně předpokládáme lineární procesní model definovaným jako

$$\mathbf{x}(k) = \mathbf{F}\mathbf{x}(k-1) + \mathbf{B}\mathbf{u}(k-1) + \mathbf{w}(k-1) \tag{3.12}$$

kde $\mathbf{x}$ je stavový vektor, $\mathbf{F}$ je matice přechodů, $\mathbf{B}$ je vstupně-kontrolní matice, $\mathbf{u}$ je vstupní vektor a $\mathbf{w}$ je procesní šum, o kterém předpokládáme, že pochází z Gaussovy distribuce o kovarianci $\mathbf{Q}$: $\mathbf{w} \sim \mathcal{N}(0, \mathbf{Q})$. Vazba stavového vektoru na prostor měření je ve tvaru

$$\hat{\mathbf{y}}(k) = \mathbf{H}\mathbf{x}(k-1) + \mathbf{v}(k-1) \tag{3.13}$$

kde $\mathbf{y}$ je stavový vektor transformovaný do měřené oblasti, $\mathbf{H}$ je matice měření a $\mathbf{v}$ je šum měření, o kterém opět předpokládáme, že pochází z Gaussovy distribuce o kovarianci $\mathbf{R}$: $\mathbf{v} \sim \mathcal{N}(0, \mathbf{R})$. Takto definovaný filtr je použit pro filtraci detekcí a filtraci estimovaných klíčových bodů. V obou případech se jedná o model uvažující

konstantní rychlost, kde stavové vektory jsou ve tvaru

$$\begin{aligned} \mathbf{x}_d &= \begin{bmatrix} c_x & c_y & a & \dot{c_x} & \dot{c_y} & \dot{a} \end{bmatrix}^T \\ \mathbf{x}_p &= \begin{bmatrix} \mathbf{p} & \dot{\mathbf{p}} \end{bmatrix}^T \end{aligned} \tag{3.14}$$

kde $\mathbf{x}_d$ je stavový vektor polohy a velikosti objektu, c_x a c_y jsou středové souřadnice středu ohraničujícího boxu objektu, a je jeho velikost (celkový obsah boxu), $\mathbf{x}_p$ je stavový vektor klíčových bodů objektu a $\mathbf{p}$ je vektor polohy jednotlivých klíčových bodů. Stavové vektory se v obecném smyslu liší pouze dimenzionalitou měřeného stavu, proto budou popsány pouze matice modelu pro filtraci detekcí, matice modelů pro filtraci klíčových bodů jsou sestaveny analogicky. Navržené matice pro filtraci detektoru jsou ve tvaru

$$\begin{aligned} \mathbf{F}_d &= \begin{bmatrix} 1 & 0 & 0 & dT & 0 & 0 \\ 0 & 1 & 0 & 0 & dT & 0 \\ 0 & 0 & 1 & 0 & 0 & dT \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{Q}_d = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25 \end{bmatrix} \\ \mathbf{H}_d &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \mathbf{R}_d = \begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 2.5 \end{bmatrix} \end{aligned} \tag{3.15}$$

Hodnoty kovariancí procesního a měřícího šumu jsou zvoleny empiricky na základě experimentů. Vzhledem k tomu, že použitý model je model uvažující konstantní rychlost, což v realitě není pravda, sledované objekty (lidské ruce a obličej) svou rychlost mění, jsou kovariance rychlostních složek procesního šumu nastaveny na vyšší hodnotu. V rámci této filtrace není využíván vstupní vektor $\mathbf{u}$, nestanovujeme tedy matice $\mathbf{B}$.

Jak bylo zmíněno, stavový vektor pro filtraci detekcí $\mathbf{x}_d$ obsahuje pouze obsah korespondujícího boxu objektu a, navzdory tomu, že modely detektorů predikují výšku h i šířku w objektu. Predikované boxy jsou v rámci zpracování transformovány na čtvercové, převážně z důvodů zjednodušení problému, dochází tedy k redukci informace o obou stranách predikovaných boxů na informaci o velikosti jedné strany a jelikož sledované objekty jsou obecně modelovatelné čtvercovým ohraničujícím boxem, nedochází ke snížení kvality zpracování. Definované Kalmanovy filtry jsou pak použity v rámci implementace algoritmu *Deep SORT*.

3.3.2 Algoritmy pro sledování objektů

Algoritmus *SORT* řeší problematiku udržování informace o sledovaných objektech a asociaci nových detekcí s těmito objekty. V původní publikaci algoritmu *SORT*,

autoři použili maďarský algoritmus s *IOU* metrikou 3.4 pro řešení asociačního problému v kombinaci s lineárním Kalmanovým filtrem pro filtraci detekce. Hlavním problémem této konfigurace je, že *IOU* metrika neobsahuje jinou informaci kromě polohy objektu, proto objekty, které nejsou detektorem zachycené v jakémkoliv časovém kroku, nemohou být re-identifikovány v dalších časových krocích, kdy detektor tyto objekty zaznamená. Z této nevýhody vyplývá, že algoritmu není příliš vhodný pro sledování objektů, u kterých dochází k okluzi.

V rozšíření algoritmu *Deep SORT* autoři *IOU* metriku používají pouze pro asociaci nepotvrzených sledovaných objektů, které jsou v inicializační fázi sledování. Detekované objekty jsou označeny jako nepotvrzené, pokud jsou detektorem zaznamenány po dobu časových kroků menší než inicializační práh T_{init}. Pro konstrukci matice nákladů pro asociaci potvrzených objektů autoři využívají kombinaci dvou metrik. První metrikou je čtvercová Mahalanobisova vzdálenost mezi predikovaným stavem Kalmanova filtru a měřeným stavem

$$\begin{aligned} d_1(i,j) &= (\mathbf{y}_j - \hat{\mathbf{y}}_i)^T \mathbf{S}_i^{-1} (\mathbf{y}_j - \hat{\mathbf{y}}_i) \\ \mathbf{S}_i &= \mathbf{H}_i \mathbf{P}_i \mathbf{H}_i^T + \mathbf{R}_i \end{aligned} \tag{3.16}$$

kde $d_1(i,j)$ je vzdálenost mezi i-tým sledovaným objektem popsaným stavovým vektorem příslušného Kalmanova filtru objektu projektovaném do měřícího prostoru $\hat{\mathbf{y}}_i$ a j-tou detekcí $\mathbf{y}_j$ a $\mathbf{S}_i$ je stavová kovariance $\mathbf{P}_i$ příslušného Kalmanova filtru projektovaná do měřícího prostoru. Tato metrika je pak dále rozšířená prahováním na 95% intervalu získaného z inverzní distribuce χ^2, čímž je dosažena filtrace nepravděpodobných asociací. Indikátor tohoto prahování je definován jako

$$b_1(i,j) = \mathbb{1}[d_1(i,j) \leq t_1] \tag{3.17}$$

kde indikátor $b_1(i,j)$ je roven jedné v případě, kdy asociace podle $d_1(i,j)$ vzdálenosti je přijatelná. Rozhodovací práh je pro tří-dimenzionální prostor měření detektoru roven $t_1 = 7.8147$. Takto definovaná vzdálenost je dobře informovaná metrika na základě pozice a pohybu objektů, ale neobsahuje vizuální informaci o jednotlivých objektech. Pro tyto účely autoři využívají i druhou metriku. Metrika spočívá v kosinově vzdálenosti mezi detekovanými a sledovanými objekty v oblasti hlubokých příznaků získaných z externí konvoluční neuronové sítě. Nevýhodou tohoto přístupu je použití dalšího extraktoru příznaků, čímž by se také zvýšila výpočetní náročnost celého systému. Namísto tohoto přístupu je využita informace o estimovaných klíčových bodech, které jsou použity jako vizuální deskriptor objektů. Metrika vzdálenosti je pak čtvercová vzdálenost jednotlivých klíčových bodů detekovaného objektu a sledovaného objektu.

$$d_2(i,j) = \sum_{n=1}^{N} \left(\mathbf{p}_i^{(n)} - \mathbf{p}_j^{(n)} \right)^2 \tag{3.18}$$

kde $d_2(i,j)$ je vzdálenost mezi i-tým sledovaným objektem a j-tým detekovaným objektem a N je počet klíčových bodů objektů. Tato vzdálenost je opět prahovaná pro získání indikátoru validity asociace, v tomto případě je prah empiricky zvolen.

$$b_2(i,j) = \mathbb{1}[d_2(i,j) \leq t_2] \tag{3.19}$$

Z těchto metrik je sestrojena matice nákladů, podle které je pomocí maďarského algoritmu řešen asociační problém. Matice nákladů je vypočtena jako váhový součet obou metrik

$$C(i,j) = \lambda d_1(i,j) + (1-\lambda) d_2(i,j) \tag{3.20}$$

kde $\lambda \in \langle 0,1 \rangle$ je váha jednotlivých metrik a asociace $C(i,j)$ je přístupná pokud jsou splněny podmínky obou indikátorů validity

$$b(i,j) = \prod_{m=1}^{2} b_m(i,j) \tag{3.21}$$

Autoři v publikaci také uvádí, že v případech, kdy kamera není staticky uložená, ale pohybuje se společně se sledovanými objekty, je dosaženo nejlepších výsledků při $\lambda = 0$. Navzdory tomu, že v rámci práce je kamera umístěna staticky, bylo experimentálně zjištěno, že metrika d_1 přináší chybnou informaci do asociační matice. Tento problém pramení v nízké stavové kovarianci Kalmanova filtru v případě, kdy sledovaný objekt je téměř statický a náhlá akcelerace objektu prudce vzdálenost d_1 zvedne. Tento jev je sledován převážně při nízké vzorkovací frekvenci, kde snímky jsou zpracovávány s frekvencí menší než 10 snímků za vteřinu. Z tohoto důvodu je v rámci práce zvoleno taktéž $\lambda = 0$. Z dalších experimentů také vyplynulo, že d_2 je dostatečně informovaná metrika, pomocí níž je algoritmus dostatečně robustní.

Tímto je sestavena matice nákladů asociačního problému, výstupem asociace je seznam detekovaných objektů, které jsou přiřazeny ke sledovaným objektům, seznam detekovaných objektů, které jsou označeny za nové a seznam sledovaných objektů, které nebyly asociovány. Objekty detekované, ale nepřiřazené jsou zařazeny do inicializačního stavu a jsou označeny jako nepotvrzené. Objekty, které jsou sledovány, ale detektorem nejsou zaznamenány, jsou stále aktivní, dokud nejsou přiřazeny k detekci po dobu T_{max} časových kroků, v tomto případě je objekt vyřazen ze seznamu sledovaných objektů. Všechny objekty, které jsou sledovány mají přiřazený Kalmanův filtr K_d pro filtraci měření polohy a velikosti objektu. Kalmanův filtr K_p pro filtraci měření polohy jednotlivých klíčových bodů je přiřazen pouze objektům rukou převážně z důvodů výpočetní náročnosti. Součástí výpočtu aktualizace Kalmanova filtru je výpočet inverzní matice stavové kovariance v měřící rovině $(R + HP^{-}H^{T})^{-1}$, která je počítána dekompozicí singulárních hodnot (*Singular Value Decomposition - SVD*) o výpočetní komplexitě $O(N^3)$. Experimentálně bylo zjištěno, že tento výpočet je počitatelný v reálném čase v případě, kdy se jedná o stav polohy klíčových bodů

lidské ruky o délce $21 \cdot 2 = 42$, ale výpočet je příliš dlouhý pro výpočet aktualizace stavu polohy klíčových bodů obličeje o délce $98 \cdot 2 = 196$. Z poměru výpočetních náročností lze také experimenty ověřit.

$$\frac{N_O^3}{N_R^3} = \frac{196^3}{42^3} = 101.62 \tag{3.22}$$

Výpočet aktualizace Kalmanova filtru pro filtraci polohy klíčových bodů obličeje je tedy přibližně 100× výpočetně náročnější než aktualizace Kalmanova filtru filtrující polohu klíčových bodů lidské ruky.

3.4 Klasifikace znakového jazyka

V rámci rozpoznávání znakového jazyka je využit volně přístupný dataset z publikace Word-Level American Sign Language [86] (WLASL), který obsahuje 2000 různých dynamických znaků z amerického znakového jazyka interpretovaných různými interprety včetně dialektů. Součástí datasetu je rozdělení datových vzorků mezi trénovací, validační a testovací subsety. Autoři také v publikaci dataset rozdělují podle velikosti slovní zásoby na datasety o 100, 300, 1000 a 2000 slovech (označovány jako WLASL-100, WLASL-300, WLASL-1000 a WLASL-2000). Distribuce datových vzorků jednotlivých datasetů mezi subsety je vypsána v tabulce 3.12.

	Učení	**Validace**	**Testování**
WLASL-2000	14 289	3 916	2 878
WLASL-1000	8 974	2 318	1 876
WLASL-300	3 549	900	668
WLASL-100	1 442	338	258

Tab. 3.12: Distribuce vzorků WLASL datasetů mezi subsety pro učení, validaci a testování

Pro potřeby klasifikace, je potřeba mít k datovým vzorkům anotaci klíčových bodů obličeje a rukou odpovídající navrženým modelům estimátorů v rámci této práce. Součástí WLASL datasetu jsou anotace pózy člověka, ovšem klíčové body obličeje v anotaci neodpovídají klíčovým bodům estimovanými navrženým modelem. Dále také inspekcí kvality anotací klíčových bodů rukou lze pozorovat kvalitativní nedostatky anotací. Z těchto důvodů dodané anotace v rámci datasetu nejsou využity a anotace jsou získány automatickou anotací navrženými modely. Každá video

sekvence datasetu je tedy anotována pomocí modelů detektorů a estimátorů v kombinaci s algoritmem pro dynamické sledování objektů a filtrace chyb. Pro maximální přesnost výsledných anotací jsou video sekvence analyzovány dopředně i zpětně (video je analyzováno pozpátku), tím jsou získány dvě anotační sekvence a výsledná anotační sekvence je získána aritmetickým průměrem jednotlivých klíčových bodů.

$$\mathbf{k}^{(i)}(t) = \frac{1}{2}(\mathbf{k}^{(i)}_{forward}(t) + \mathbf{k}^{(i)}_{backward}(t)) \tag{3.23}$$

Tyto body nesou informaci o své absolutní poloze v původním snímku, je tedy potřeba body transformovat do univerzálního formátu, který nebude závislý na velikosti zpracovávaného snímku nebo velikosti a pozici člověka promítnutého na snímky. Pro potřebu takové normalizace stanovíme normalizovanou vzdálenost mezi středy lidských očí β. Normalizační faktor je pak počítán pro každý snímek sekvence a výsledný faktor je roven jejich aritmetickému průměru podle následujícího vztahu

$$\gamma = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{\beta}\sqrt{(\mathbf{k}^{(lo)}(t) - \mathbf{k}^{(po)}(t))^2} \tag{3.24}$$

kde γ je výsledný normalizační faktor, β je stanovená normalizovaná vzdálenost, $\mathbf{k}^{(lo)}$ a $\mathbf{k}^{(po)}$ jsou klíčové body levého a pravého oka a N je počet snímků sekvence. V rámci práce je stanoveno $\beta = 0.1$. Dále je potřeba klíčové body centralizovat vůči subjektu člověka, jako středový bod transformovaných bodů je zvolena špička nosu člověka. Výsledná transformace je ve tvaru

$$\widetilde{\mathbf{k}}^{(i)}(t) = \frac{\mathbf{k}^{(i)}(t) - \mathbf{k}^{(n)}(t)}{\gamma} \tag{3.25}$$

kde $\mathbf{k}^{(n)}$ je poloha klíčového bodu špičky nosu. Výsledkem je anotace pro každý snímek každého datového vzorku datasetu, která obsahuje informaci o 2D poloze 98 klíčových bodů obličeje a 2×21 klíčových bodů rukou. Zřetězením těchto bodů získáme pro každý snímek příznakový vektor o 280 prvcích, pro každou video sekvenci datasetu je pak k dispozici sekvence příznakových vektorů $S \in \mathbb{R}^{\mathrm{N}\times 280}$, kterou při učení nebo inferenci klasifikátoru transformujeme podle rovnice 3.25. Pro paralelizaci procesu učení je v rámci učení modelů v každé iteraci z každé sekvence o délce N náhodně vybrána sekvence $N_S = 50$ snímků, na kterých je model naučen. Touto úpravou je umožněna propagace B datových vzorků modelem najednou $S_B \in \mathbb{R}^{\mathrm{B}\times \mathrm{N_S}\times 280}$. V rámci testování k uvedenému vzorkování neprobíhá, propagace datových vzorků probíhá po jednom, pro získání co nejpřesnějších výsledků. V rámci inference, je pak parametrem na kolika snímcích je propagace modelem počítána. Vhodným základním nastavením je stejný počet snímků, na kterých byly modely naučeny, tedy $N_I = 50$. Avšak optimální velikost této sekvence je laděna, tak aby co nejlépe izolovala interpretovaný znak. Závisí tedy na frekvenci zpracovávání snímků

a rychlosti interpretace. V rámci práce byla pro inferenci v reálném čase použita délka sekvence $N_I = 20$, ale jedná se o parametr, který by mohl být uživatelsky nastavitelný.

Na uvedené vzorkování v rámci učení modelu lze také pohlížet jako na formu datové augmentace, jako další forma datové augmentace je využita náhodná poměrná translace rukou vzhledem k obličeji a také náhodné zašumění polohy klíčových bodů podle Gaussovy distribuce.

Jako klasifikační modely jsou navrženy čtyři jednoduché rekurentní neuronové sítě, každá navržená pro naučení na jednom z WLASL datasetu. Navržená architektura je složena z plně propojené vrstvy o 128 neuronech, bloků GRU vrstev a výstupní plně propojené vrstvy. Počty GRU vrstev a počty neuronů v GRU vrstvách a ve výstupní vrstvě závisí na použitém datasetu, tato informace je uvedena v tabulce 3.13. Výstupem všech modelů je pravděpodobnostní rozdělení klasifikace sekvence mezi jednotlivé třídy. Při procesu učení byla jedním z hlavních problémů

	#Neuronů v GRU vrstvách	**#Neuronů ve výstupní vrstvě**
WLASL-2000	48, 32 a 32	2000
WLASL-1000	40 a 40	1000
WLASL-300	32 a 24	300
WLASL-100	24 a 16	100

Tab. 3.13: Konfigurace klasifikátorů pro jednotlivé datasety

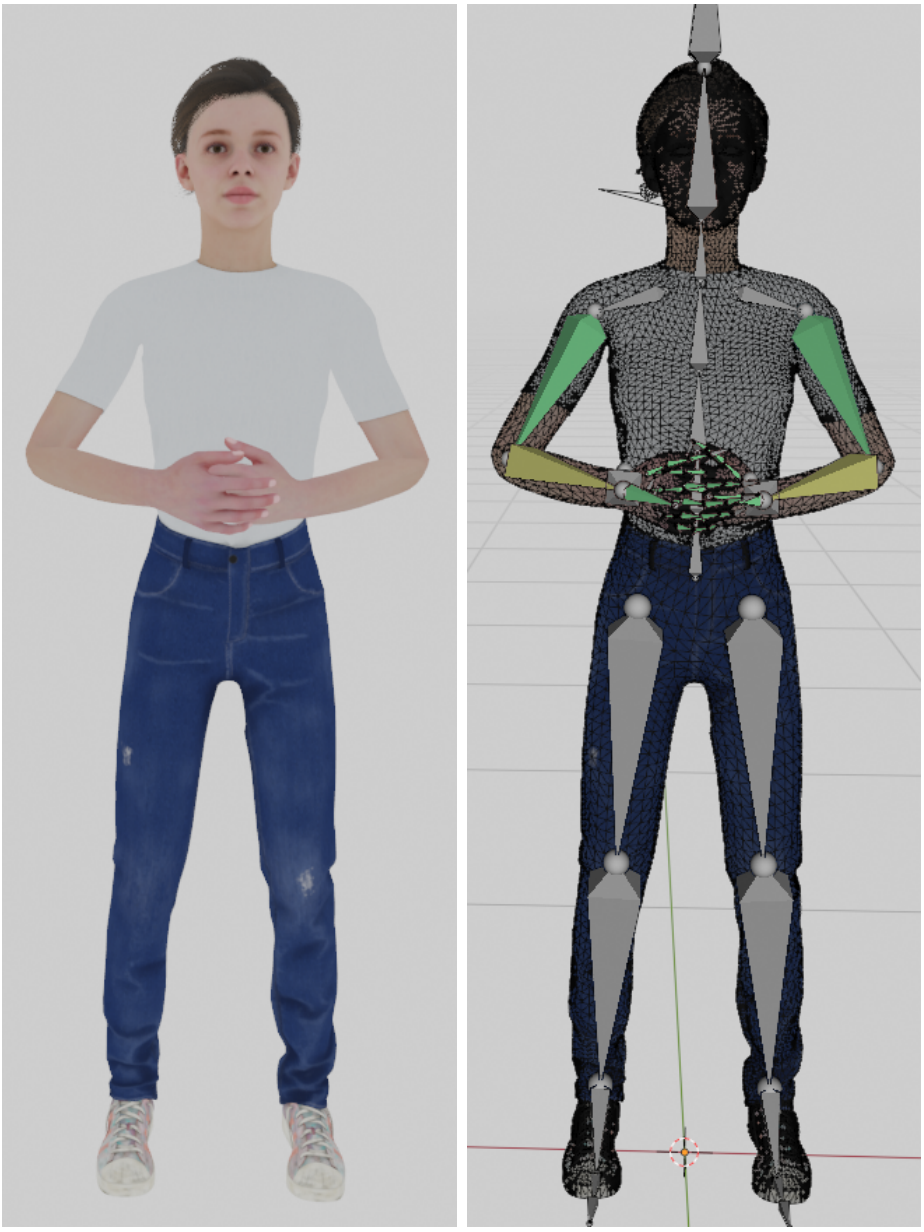
tendence modelů k přeučení na trénovacím subsetu, kdy validační chyba diverguje. Na základě tohoto jevu byly empiricky navrženy architektury s cílem minimalizovat validační chybu. Jako regularizační jednotky jsou použity normalizační vrstvy, které jsou vloženy za každou GRU vrstvu a *dropout* vrstvy, které jsou vloženy za všechny normalizační vrstvy a za úvodní plně propojenou vrstvu.

Váhy modelů jsou inicializovány náhodně podle uniformního rozložení Glorota a Begnia 3.8. Modely jsou naučeny pomocí *adam* optimalizátoru 1.10, jako chybová funkce je zvolena chyba entropie a koeficient učení je inicializován na hodnotu 10^{-3} a snižován v případě, kdy při 100 iteracích nedojde ke snížení validační chyby. Ukončovací podmínka učení je divergence validační chyby po dobu 200 iterací.

3.5 Automatický překlad počítačovým modelem

Cílem této aplikační části je vytvoření animovaného počítačového modelu interpretující znaky ze slovní zásoby použitého WLASL datasetu. Jako počítačový model je využit volně dostupný model z online *Adobe Mixamo* knihovny [111]. Model obsahuje

barevné textury i základní kostru včetně dlaní a prstů. Pro práci s tímto modelem je využit *open-source* nástroj *Blender* [112]. Na obrázku 3.10 je vyobrazen použitý model a jeho kostra ve výchozí poloze. Z obrázku kostry modelu lze vidět absenci kontrolních kostí pro obličej modelu, v rámci práce jsou animace zjednodušeny pouze na pohyb rukou.

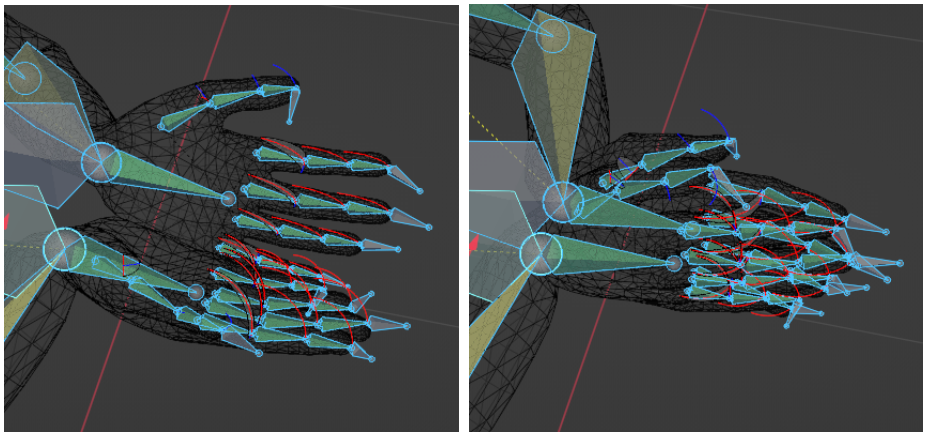


(a) Textury (b) Kostra

Obr. 3.10: Počítačový model avatara

Samotné animace jsou provedeny translací a rotací jednotlivých kontrolních kostí, čímž je docíleno postupné ukládání klíčových pozic, mezi kterými je pak pohyb interpolován. Pro translaci a rotaci jednotlivých kostí lze použít přímou i inverzní

kinematiku. Pro zajištění jisté míry kontinuity jednotlivých animací, všechny animace předpokládají začátek i konec svého pohybu ve výchozí poloze. Příkladem animace modelu jsou dvě klíčové pózy na obrázku 3.11, kde je animován znak *kniha* v českém znakovém jazyce a mezi klíčové pózy celkové animace patří vyobrazené dvě pózy, mezi kterými je pohyb interpolován. Jako interpolační metoda je využita Bézierova křivka.



Obr. 3.11: Dvě klíčové pózy animace modelu

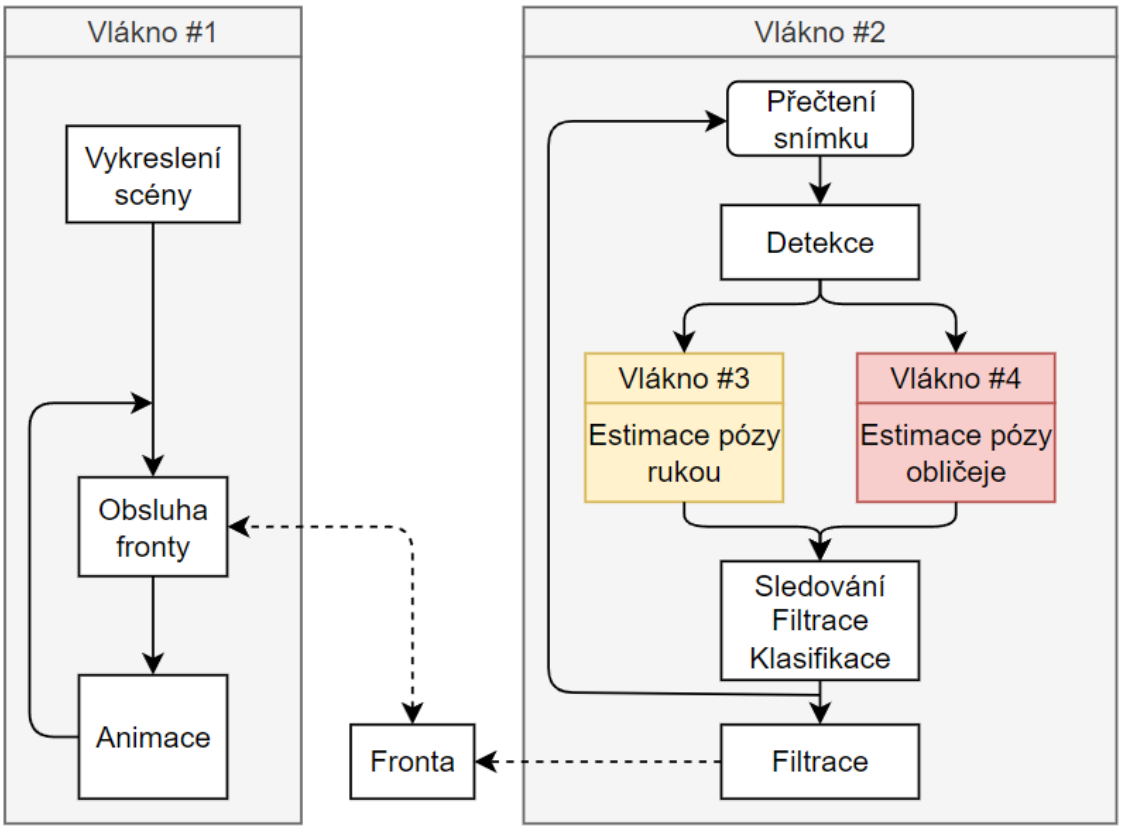
V rámci řešení práce jsou animovány znaky v českém znakovém jazyce, tedy je realizována úloha překladu do českého znakového jazyka. V rámci práce jsou animovány pouze tři znaky, sloužící jako ukázka funkcionality. Více znaků nebylo animováno, převážně z časových důvodů, jelikož úloha animace je pro nezkušeného animátora časově náročná. Animované znaky jsou *kniha*, *basketbal* a *rodina*.

Pro vykreslování modelu a animací v *Python* prostředí je využit *open-source* herní engine *Panda3D* [113]. V rámci vykreslování na základě výstupu z modelu klasifikátoru, je potřeba vzít v potaz opakovaných žádostí o vykreslení stejného znaku v sekvenci. Pro odfiltrování tohoto jevu, je udržována informace o vykreslených animacích v předchozích časových krocích. To znamená, že animace, které jsou vykreslené, jsou přidány do seznamu nedávno vykreslených animací. Tyto animace nemohou být v dalších časových krocích vykresleny, dokud nejsou ze seznamu odstraněny. K odstranění dojde ve chvíli, kdy daný znak není v top q predikcích klasifikátoru, kde za q je zvolena hodnota 5.

3.6 Integrace algoritmů

Algoritmy jsou integrovány do navržené pipeline, dle návrhu (obrázek 2.2). Na výstupu tohoto řetězce je pak přidána aplikace automatické interpretace. Z implementačního hlediska je potřeba spouštět aplikaci vykreslující 3D model interpreta v sa-

mostatném vlákně, z důvodů implementace použitého nástroje *Panda3D* je potřeba tuto aplikaci spustit v hlavním vlákně. V druhém vlákně je tedy spuštěna pipeline zpracovávající obrazovou informaci a hlavnímu vláknu pomocí fronty podává požadavky na vykreslení požadovaného znaku v požadovaném cílovém jazyce. Dále v rámci řetězce pro zpracování obrazové informace je možné paralelizovat výpočet estimace klíčových bodů rukou a obličeje. Vzhledem k faktu, že estimace klíčových bodů je provedena konvolučními neuronovými sítěmi, je možné využít hardwarovou akceleraci na GPU, případně TPU. V tomto případě paralelizace nemá příliš velký efekt, obě tato vlákna totiž využívají stejný výpočetní zdroj. Tato paralelizace má převážně výhodný efekt pouze v případě výpočtu na CPU. Ilustrace paralelizace výsledného systému je na obrázku 3.12.



Obr. 3.12: Ilustrace paralelizace výsledného systému

4 Výsledky práce

V rámci této kapitoly jsou navržené algoritmy evaluovány na testovacích sadách a jsou diskutovány jejich vlastnosti. V rámci evaluace je také analyzována výpočetní náročnost jednotlivých algoritmů. Základní specifikace výpočetní jednotky, na které jsou algoritmy testovány, je vysazena v tabulce 4.1.

CPU	AMD Ryzen 5 5600X
GPU	NVIDIA GTX 960 2GB
Operační paměť	16GB 3200 MHz

Tab. 4.1: Specifikace výpočetní jednotky

Krom přesnosti jednotlivých algoritmů jsou taktéž monitorovány jejich výpočetní náročnosti. V rámci evaluace výpočetní náročnosti jsou sledovány hodnoty počtu parametrů sítí, počtu operací nad čísly s pohyblivou desetinnou čárkou pro dopředný výpočet výstupu sítí (označováno jako *floating point operations - FLOPs*) a latence inference na výpočetní jednotce.

4.1 Detektory rukou a obličejů

V rámci evaluace objektových detektorů je použita metrika *Average Precision - AP* [114], která je definovaná jako

$$AP_{IOU} = \int P_{IOU}(R_{IOU}) dR_{IOU} \tag{4.1}$$

kde P je přesnost (z angličtiny *precision*) a R je poměr vybavení (z angličtiny *recall*), které jsou definované jako

$$\begin{aligned} P_{IOU} &= \frac{\#TP_{IOU}}{\#TP_{IOU} + \#FP_{IOU}} \\ R_{IOU} &= \frac{\#TP_{IOU}}{\#TP_{IOU} + \#FN_{IOU}} \end{aligned} \tag{4.2}$$

kde *TP*, *FP* a *FN* jsou definované původně pro klasifikační potřeby, kde jako *True Positive - TP* je označen vzorek, který je správně klasifikován do pozitivní třídy, za *False Positive - FP* je označen vzorek, který je nesprávně klasifikován do pozitivní třídy a *False negative - FN* je označení vzorku nesprávně klasifikovaného do negativní třídy. Ozačení $\#TP$, $\#FP$ a $\#FN$ pak značí počet vzorků označených jako *TP*, *FP* a *FN* respektive. V rovnicích 4.1 a 4.2 pak figuruje subskript *IOU*. V rámci úlohy detekce objektů jsou detekce označovány jako *TP*, *FP* a *FN* na základě *Intersecion Over Union* hodnoty (definováno rovnicí 3.4) detekce se známou

informací o detekovaném objektu. Subskript $IOU \in \langle 0, 1\rangle$ definuje hraniční hodnotu *Intersecion Over Union*, kdy je detekce s daným vzorem považovaná jako správná, tedy *TP*. V praxi je výsledná hodnota AP stanovena vzorkováním interpolované křivky $P_{IOU}(R_{IOU})$. Počet vzorků je uživatelsky stanoven, avšak v některých publikacích, které se zabývají tvorbou unifikovaného testovacího setu pro evaluaci různých objektových detektorů je počet vzorků přímo stanoven. V rámci práce bylo použito vzorkování 11-ti vzorky, podle publikace *The PASCAL Visual Object Classes (VOC) challenge* [114].

$$AP@IOU = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.2, \ldots, 1\}} P_{inter}@IOU(r) \tag{4.3}$$

kde $AP@IOU$ je alternativním označením AP_{IOU} a interpolovaná křivka funkce $P@IOU(R)$ je definovaná pro hodnotu argumentu r jako

$$P_{inter}@IOU(r) = \max_{\tilde{r}:\tilde{r} \geq r} P@IOU(\tilde{r}) \tag{4.4}$$

Samostatné detektory rukou a obličejů jsou naučeny na uvedených datasetech a jsou evaluovány pomocí AP metriky na jednotlivých testovací subsetech. Výsledky jsou uvedeny v tabulce 4.2 pro různé hodnoty IOU.

	$AP_{0.3}$	$AP_{0.5}$	$AP_{0.7}$
Detekce rukou	0.66	0.57	0.30
Detekce obličejů	0.72	0.66	0.31

Tab. 4.2: Evaluace samostatných detektorů rukou a obličejů

Detektory naučené pro potřeby současné detekce rukou i obličejů v obraze jsou evaluovány podle AP metriky na testovacím subsetu. V případě, kdy detektor detekuje i klasifikuje výsledné detekce do $C > 1$ tříd, je metrika AP počítána pro každou třídu odděleně a výsledná hodnota aritmetického průměru jednotlivých AP je označována jako *mean AP - mAP*.

$$mAP_{IOU} = \frac{1}{C} \sum_{i=1}^{C} AP_{IOU,i} \tag{4.5}$$

Výsledky evaluace naučených detektorů pro různé hodnoty IOU jsou uvedeny v tabulce 4.3, kde jsou mimo hodnoty mAP uvedeny také dílčí hodnoty AP_R reprezentující AP detekce rukou a AP_O reprezentující AP detekce obličejů. Evaluace jednotlivých detektorů podle výpočetní náročnosti je uvedena v tabulce 4.4. Z tabulky 4.4 lze pozorovat, že architektury typu *EfficientDet* mají oproti detektorům

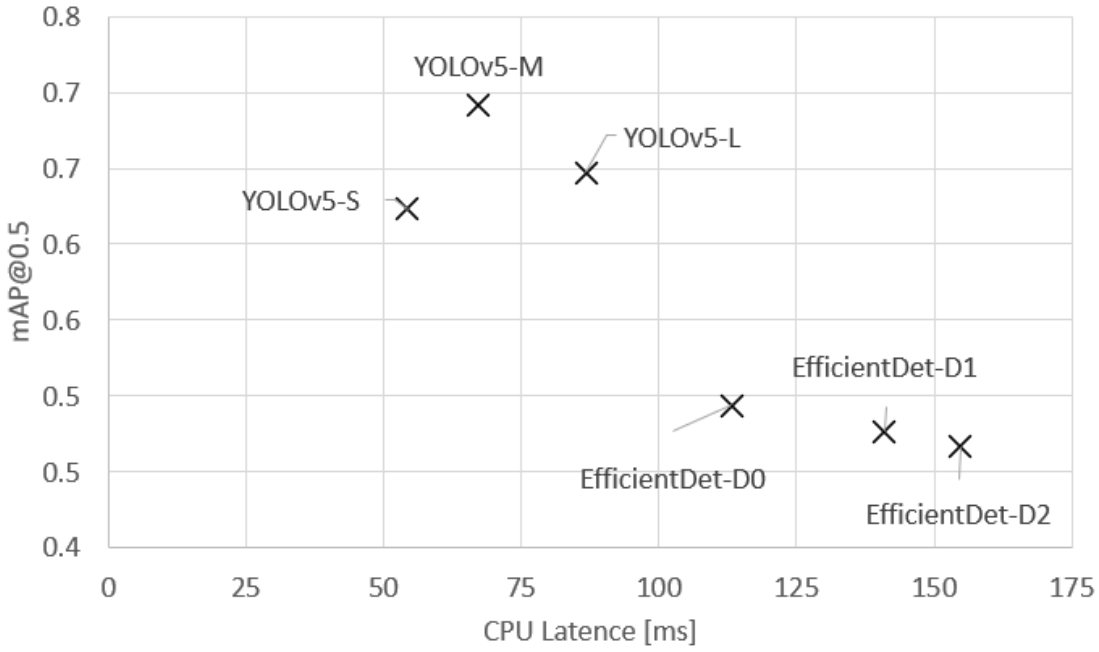
Konfigurace	$IOU = 0.3$			$IOU = 0.5$			$IOU = 0.7$		
	mAP	AP_R	AP_O	mAP	AP_R	AP_O	mAP	AP_R	AP_O
YOLOv5-S	0.70	0.68	0.72	0.62	0.58	0.66	0.34	0.31	0.38
YOLOv5-M	**0.76**	**0.76**	**0.76**	**0.68**	**0.67**	**0.69**	**0.42**	**0.41**	**0.43**
YOLOv5-L	0.70	0.73	0.67	0.64	0.62	0.65	0.38	0.37	0.39
EfficientDet-D0	0.58	0.59	0.57	0.49	0.54	0.44	0.25	0.24	0.26
EfficientDet-D1	0.56	0.57	0.54	0.46	0.47	0.46	0.24	0.25	0.24
EfficientDet-D2	0.55	0.55	0.55	0.46	0.46	0.45	0.22	0.22	0.22

Tab. 4.3: Evaluace kombinovaných detektorů rukou a obličejů

Konfigurace	#Parametrů	FLOPs	GPU Latence	CPU Latence
YOLOv5-S	$7.09 \cdot 10^6$	$10.46 \cdot 10^9$	18.98 ms	54.31 ms
YOLOv5-M	$21.10 \cdot 10^6$	$32.20 \cdot 10^9$	47.98 ms	67.26 ms
YOLOv5-L	$46.70 \cdot 10^6$	$73.01 \cdot 10^9$	132.32 ms	86.73 ms
EfficientDet-D0	$3.87 \cdot 10^6$	$4.14 \cdot 10^9$	43.28 ms	113.24 ms
EfficientDet-D1	$6.62 \cdot 10^6$	$6.40 \cdot 10^9$	114.70 ms	140.95 ms
EfficientDet-D2	$8.08 \cdot 10^6$	$7.82 \cdot 10^9$	133.52 ms	154.61 ms

Tab. 4.4: Výpočetní náročnost objektových detektorů

rodiny *YOLO-v5* nízký počet parametrů a nízké množství *FLOPs*, ale jejich inferenční latence je podstatně vyšší. Můžeme tedy pozorovat, že výsledná latence je závislá nejen na počtu parametrů a počtu *FLOPs*, ale také na vnitřní konfiguraci sítě. Jako hlavní parametry pro zhodnocení jednotlivých konfigurací volíme evaluaci podle *mAP* metriky a latenci na procesorové jednotce. Latenci na grafické jednotce není příliš zohledněna z důvodů nedostatečné paměti grafické jednotky na výpočetní jednotce, která způsobuje zavádějící výsledky výsledné inferenční latence. Dalším odůvodněním přihlížení k latenci na CPU je bližší simulace výsledné inferenční doby na mobilních zařízeních, které bývají obecně výpočetně méně výkonné. Výsledky evaluace detektorů jsou zaneseny do grafu závislosti *mAP* na CPU latenci jednotlivých konfigurací 4.1. Z výsledků je zřejmé, že nejlepších výsledků s ohledem na evaluace podle *mAP* a CPU latence dosahují konfigurace typu *YOLOv5-S* a *YOLOv5-M*, kde síť typu *YOLOv5-M* dosahuje vyšší přesnosti oproti síti typu *YOLOv5-S*, avšak za cenu vyšší výpočetní náročnosti. Finální volbu konfigurace, která bude použita v rámci výsledného algoritmu práce, není vhodné uskutečnit bez konkrétních znalosti vlastností cílové výpočetní jednotky uživatele, volbu však omezujeme na konfigurace typu *YOLOv5-S* a *YOLOv5-M*.



Obr. 4.1: Závislost *mAP* objektových detektorů na CPU latenci

4.2 Estimace klíčových bodů rukou

Modely estimující klíčové body rukou (obrázek 3.7) obsahují dva výstupy estimující klíčové body a třetí výstup klasifikující vstupní snímky podle prezence interagujících rukou. Výstupy estimující klíčové body jsou evaluovány pomocí metriky průměrné absolutní chyby (*Mean Absolute Error - MAE*), která je definovaná jako

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4.6}$$

kde N je počet anotovaných bodů, $\mathbf{y}$ je vektor predikovaných bodů a $\hat{\mathbf{y}}$ je vektor anotovaných bodů. Hodnota *MAE* se počítá vždy jen z validního výstupu sítě, tedy chyba estimace klíčových bodů jedné ruky je počítána pouze v případě, kdy je snímek anotován jako jedna ruka a obdobně pro ruce interagující. Metrika je počítána na snímcích izolovaných rukou o velikosti 256×256 pixelů, jednotka výsledných hodnot metriky jsou tedy pixely. Klasifikační výstup je hodnocen metrikou vyčíslující plochu pod křivkou *Reciever operating characteristic* (zkráceně *ROC*) [5], která je definovaná jako funkce

$$Recall = f(FPR) \tag{4.7}$$

kde *Recall* je definován rovnicí 4.2 a *FPR* (*false positive rate*) je definován jako

$$FPR = \frac{\#FP}{\#FP + \#TN} \tag{4.8}$$

ROC je pak závislost 4.7, kde hodnoty *Recall* a *FPR* jsou počítány pro různé prahy klasifikace $\mathbf{t}$, výstup klasifikátoru je pak označen jako pozitivní pokud splňuje podmínku $\hat{y} \geq t_i$.

Evaluace estimací klíčových bodů naučených estimátorů jsou počítány pro predikované body, které jsou získány z výstupních teplotních map sítí. Evaluace je provedena za použití obou metod dekódování teplotních map popsaných v sekci 3.2.4, jedná se o metodu maxima a metodu kompenzované střední hodnoty pro $\Delta = 3$. Výsledky evaluace na testovacím subsetu naučených modelů jsou zaneseny do tabulky 4.5, kde pro výpočet plochy pod křivkou *ROC* jsou zvoleny prahy v rozsahu $\langle 0, 1 \rangle$ s krokem 0.01. Hodnota metriky *MAE* je počítána samostatně pro samostatné ruce MAE_S a interagující ruce MAE_I. Evaluace je provedena také v rámci oddělených testovacích subsetů COCO (tabulka 4.6), MWB (tabulka 4.7) a RHD (tabulka 4.8) datasetů. Dále jsou jednotlivé modely evaluovány podle jejich výpočetní náročnosti, výsledky jsou zapsány v tabulce 4.9.

Konfigurace	**Maximum**		**Kompenzace**		**Plocha pod *ROC***
	MAE_S	MAE_I	MAE_S	MAE_I	
EfficientNet-B0	14.50	14.79	14.76	14.51	**0.98**
EfficientNet-B1	**13.17**	**13.82**	14.16	15.33	0.97
EfficientNet-B2	13.87	15.38	14.19	14.72	0.97

Tab. 4.5: Výsledky evaluace estimace pózy rukou na kompletním testovacím subsetu

Konfigurace	**Maximum**		**Kompenzace**		**Plocha pod *ROC***
	MAE_S	MAE_I	MAE_S	MAE_I	
EfficientNet-B0	18.51	16.01	18.55	15.89	**0.97**
EfficientNet-B1	**17.55**	**14.91**	18.19	16.81	0.96
EfficientNet-B2	18.39	16.27	18.47	15.69	**0.97**

Tab. 4.6: Výsledky evaluace estimace pózy rukou na COCO testovacím subsetu

Z výsledků evaluace pozorujeme vyšší hodnoty výsledných chyb. Částečný podíl na této chybě je výskyt šumu v anotacích, které jsou k datasetům přiloženy, kde některé anotace jsou umístěny až s řádovou odchylkou. Nezávisle na této chybě, je však z experimentů jasné, že model generalizuje i na pózy, které nejsou zastoupeny v subsetu pro učení. Dále sledujeme, že při použití kompenzující metody dekódování teplotních map dochází k obecnému snížení přesnosti estimace. Při prozkoumání generovaných teplotních map (příklad na obrázku 4.2) je zjevné, že krom případů kdy

Konfigurace	Maximum		Kompenzace		Plocha pod *ROC*
	MAE_S	MAE_I	MAE_S	MAE_I	
EfficientNet-B0	14.17	13.55	14.95	12.87	**0.98**
EfficientNet-B1	**13.30**	**12.47**	14.82	13.50	**0.98**
EfficientNet-B2	14.35	13.30	15.33	12.55	**0.98**

Tab. 4.7: Výsledky evaluace estimace pózy rukou na MWB testovacím subsetu

Konfigurace	Maximum		Kompenzace		Plocha pod *ROC*
	MAE_S	MAE_I	MAE_S	MAE_I	
EfficientNet-B0	12.76	13.21	13.04	12.82	**0.98**
EfficientNet-B1	**11.19**	**12.52**	12.23	13.61	0.97
EfficientNet-B2	11.77	14.74	12.07	13.98	0.97

Tab. 4.8: Výsledky evaluace estimace pózy rukou na RHD testovacím subsetu

Konfigurace	#Parametrů	FLOPs	GPU Latence	CPU Latence
EfficientNet-B0	$9.50 \cdot 10^6$	$2.46 \cdot 10^9$	35.55 ms	32.16 ms
EfficientNet-B1	$12.02 \cdot 10^6$	$2.95 \cdot 10^9$	47.80 ms	37.12 ms
EfficientNet-B2	$13.64 \cdot 10^6$	$3.23 \cdot 10^9$	50.67 ms	43.44 ms

Tab. 4.9: Výpočetní náročnost naučených modelů

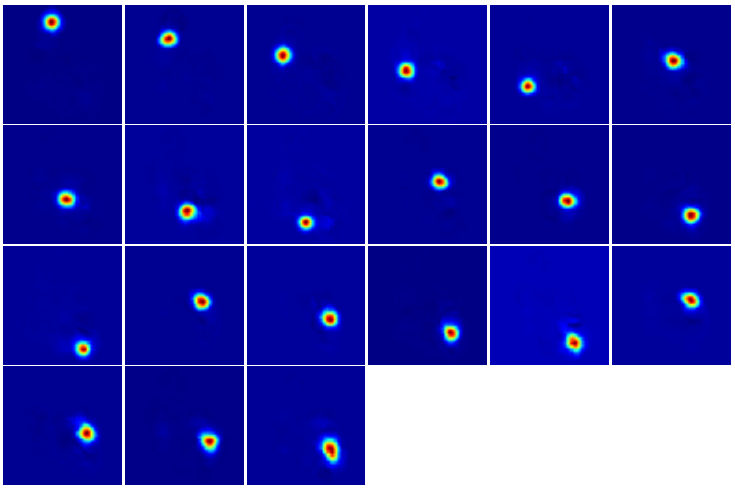
jsou generované mapy středově souměrného Gaussova rozložení s mírnou odchylkou, dochází také k jevům, kdy model si není jistý polohou estimovaného bodu a výsledné teplotní mapy nejsou středově souměrné. Volba dekódování podle maximální odezvy teplotní mapy se v tomto případě jeví jako přesnější estimace výsledného bodu než použití metody kompenzace. Z hlediska evaluace přesnosti a výpočetní náročnosti se jeví jako nejlepší konfigurace model s extraktorem příznaků *EfficientNet-B1*.

4.3 Estimace klíčových bodů obličeje

V rámci datasetu WFLW jsou k dispozici mimo jiné také anotace, označující snímky různých obtížných kontextů. Jedná se o označení typu

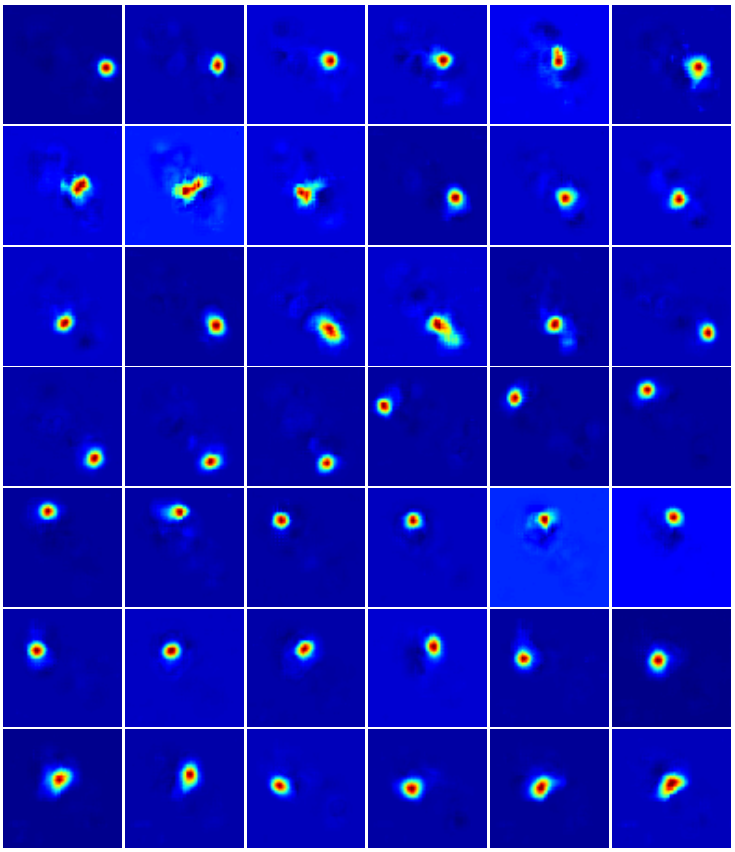
- Póza – snímek obsahuje obtížnou pózu člověka
- Výraz – výraz člověka na snímku je obtížný
- Osvětlení – snímek obsahuje obtížné osvětlení

(a) Vstupní snímek samostatné ruky



(b) Generované teplotní mapy

(c) Vstupní snímek interagujících rukou



(d) Generované teplotní mapy

Obr. 4.2: Generované teplotní mapy estimace pózy rukou

- Make-up – snímek obsahuje člověka s alterovanou barvou a motivy kůže
- Okluze – obličej člověka na snímku je do jisté míry okludován
- Rozostření – snímek je pořízen s nízkou kvalitou rozlišení nebo je postižen vysokou úrovní šumu

Snímky testovacího subsetu jsou evaluovány metrikou *MAE* 4.6, a to v rámci celého datasetu i v rámci jednotlivých kategorií snímků. Evaluace je opět provedena na izolovaných snímcích obličejů o velikost 256×256 pixelů. Výsledky této evaluace použitím metod dekódování maximem a kompenzací ($\Delta = 3$) jsou zaneseny do tabulky 4.10. Dále jsou naučené modely evaluovány podle výpočetní náročnosti, jejíž výsledky jsou zapsány v tabulce 4.11.

Konfigurace	**Maximum**						
	MAE	MAE_P	MAE_V	MAE_{Os}	MAE_M	MAE_{Ok}	MAE_R
Efficientnet-B0	4.03	5.57	4.25	4.07	4.16	4.81	4.46
Efficientnet-B1	**3.87**	**5.20**	**4.14**	**3.89**	**3.96**	**4.60**	**4.34**
Efficientnet-B2	**3.87**	5.33	4.15	3.93	4.00	4.70	4.43
Konfigurace	**Kompenzace**						
	MAE	MAE_P	MAE_V	MAE_{Os}	MAE_M	MAE_{Ok}	MAE_R
Efficientnet-B0	4.84	7.59	**5.21**	4.83	4.84	**5.55**	**5.36**
Efficientnet-B1	**4.49**	**6.63**	**5.21**	**4.82**	**4.81**	5.76	5.54
Efficientnet-B2	5.49	8.69	5.89	5.51	5.27	6.39	6.33

Pozn.: MAE_P – póza, MAE_V – výraz, MAE_{Os} – osvětlení, MAE_M – make-up, MAE_{Ok} – okluze, MAE_R – rozostření

Tab. 4.10: Výsledky evaluace estimace klíčových bodů obličeje

Konfigurace	**#Parametrů**	**FLOPs**	**GPU Latence**	**CPU Latence**
EfficientNet-B0	$5.50 \cdot 10^6$	$1.41 \cdot 10^9$	30.23 ms	25.48 ms
EfficientNet-B1	$8.02 \cdot 10^6$	$1.89 \cdot 10^9$	42.08 ms	35.40 ms
EfficientNet-B2	$9.35 \cdot 10^6$	$2.14 \cdot 10^9$	44.13 ms	39.93 ms

Tab. 4.11: Výpočetní náročnost naučených modelů

Z výsledků evaluace je zjevné, že nejtěžší úlohou estimace je úloha, kde subjekt vykazuje obtížnou pózu, převážně v případě, kdy není otočen směrem ke snímači. Obecně však pozorujeme výrazně menší hodnoty chyby než při estimaci pózy rukou, což je pravděpodobně způsobeno vysokou kvalitou datasetu nebo výrazně nižší obtížnosti úlohy estimace klíčových bodů obličeje oproti estimace klíčových bodů rukou. Obdobně jako při evaluaci estimace klíčových bodů rukou, zde sledujeme vyšší chybu při použití metody dekódování teplotních map kompenzací. Při pohledu na generované teplotní mapy (pro příklad obrázek 4.3), není na první pohled zjevné

z jakého důvodu toto pozorujeme, avšak při zpracování obtížných snímků jsou některé teplotní mapy silně nesymetrické. Toto pozorování je také podpořeno výslednými hodnotami chyby *MAE*, kde nejvyšší změna chyby je u snímků s obtížnou pózou, které obecně považujeme za nejobtížnější. Z hlediska výpočetní náročnosti je konfigurace s extraktorem *EfficientNet-B0* výrazně nejvýhodnější. Z hlediska přesnosti estimace, je nejlepší konfigurací model s extraktorem *EfficientNet-B1*. Jako výchozí model volíme konfiguraci s extraktorem *EfficientNet-B0*, zejména pro její nízkou výpočetní náročnost. V případě, kdy by byla vyžadována vyšší přesnost, může být zvolen model s extraktorem *EfficientNet-B1*.

4.4 Klasifikace znakového jazyka

V publikaci použitého datasetu WLASL [86] autoři klasifikátory evaluují metrikou *top k accuracy*. Jedná se o metriku, která zohledňuje prvních k nejvyšších predikcí. V případě, kdy je správná třída mezi prvními k predikovanými třídami, je predikce označena jako správná. Jelikož se jedná o klasifikaci mezi N tříd, jsou statistické výsledky počítány pro všechny třídy zvlášť, výsledná hodnota metriky je pak aritmetickým průměrem metrik jednotlivých tříd.

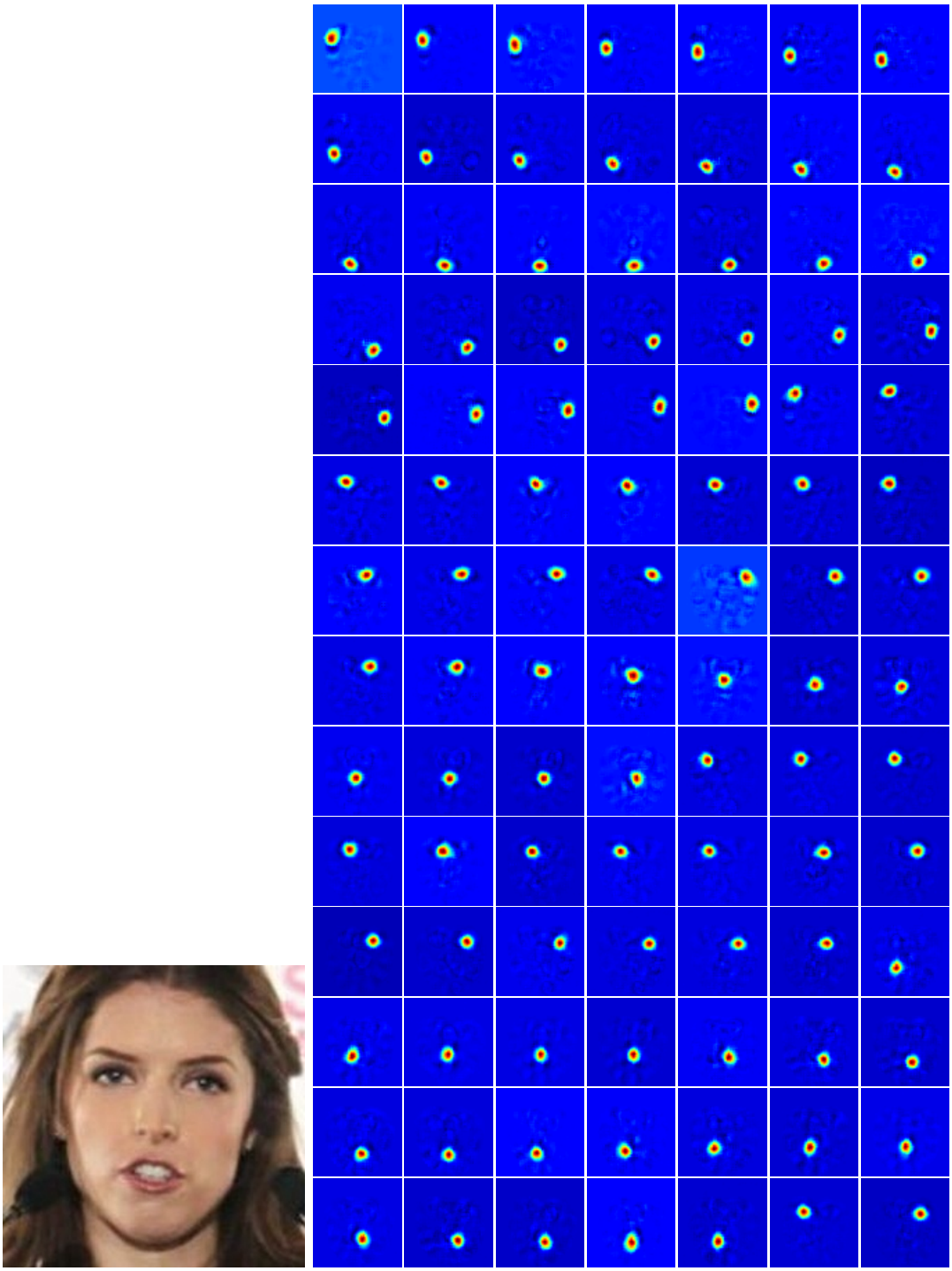
$$Accuracy_k = \frac{1}{N}\sum_{n=1}^{N}\frac{\#TP_k^{(n)} + \#TN_k^{(n)}}{\#TP_k^{(n)} + \#TN_k^{(n)} + \#FP_k^{(n)} + \#FN_k^{(n)}} \tag{4.9}$$

Výsledky naučených klasifikátorů na jednotlivých WLASL datasetech v tabulce 4.12 jsou porovnány s výsledky, kterých dosáhli autoři datasetu použitím estimace pózy člověka v kombinaci s rekurentní neuronovou sítí. Z porovnání evaluací naučených

	WLASL-100			WLASL-300		
	$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$
Tato práce	0.4457	0.7597	0.8527	0.3323	**0.6482**	**0.7635**
Původní autoři	**0.4651**	**0.7674**	**0.8566**	**0.3368**	0.6437	0.7605
	WLASL-1000			WLASL-2000		
	$k=1$	$k=5$	$k=10$	$k=1$	$k=5$	$k=10$
Tato práce	**0.3214**	**0.6290**	**0.7255**	0.2179	0.4927	0.6032
Původní autoři	0.3001	0.5842	0.7015	**0.2254**	**0.4981**	**0.6138**

Tab. 4.12: Výsledky evaluace klasifikace znaků

klasifikátorů s výsledky, kterých dosáhli autoři datasetu, je zjevné, že výsledky jsou



(a) Vstupní snímek (b) Generované teplotní mapy

Obr. 4.3: Generované teplotní mapy estimace klíčových bodů obličeje

si velmi blízké, nejvyššího rozdílu bylo dosaženo na WLASL-1000 datasetu, kde na metrice *top-5 accuracy* bylo dosaženo v rámci této práce o 4.48% lepších výsledků.

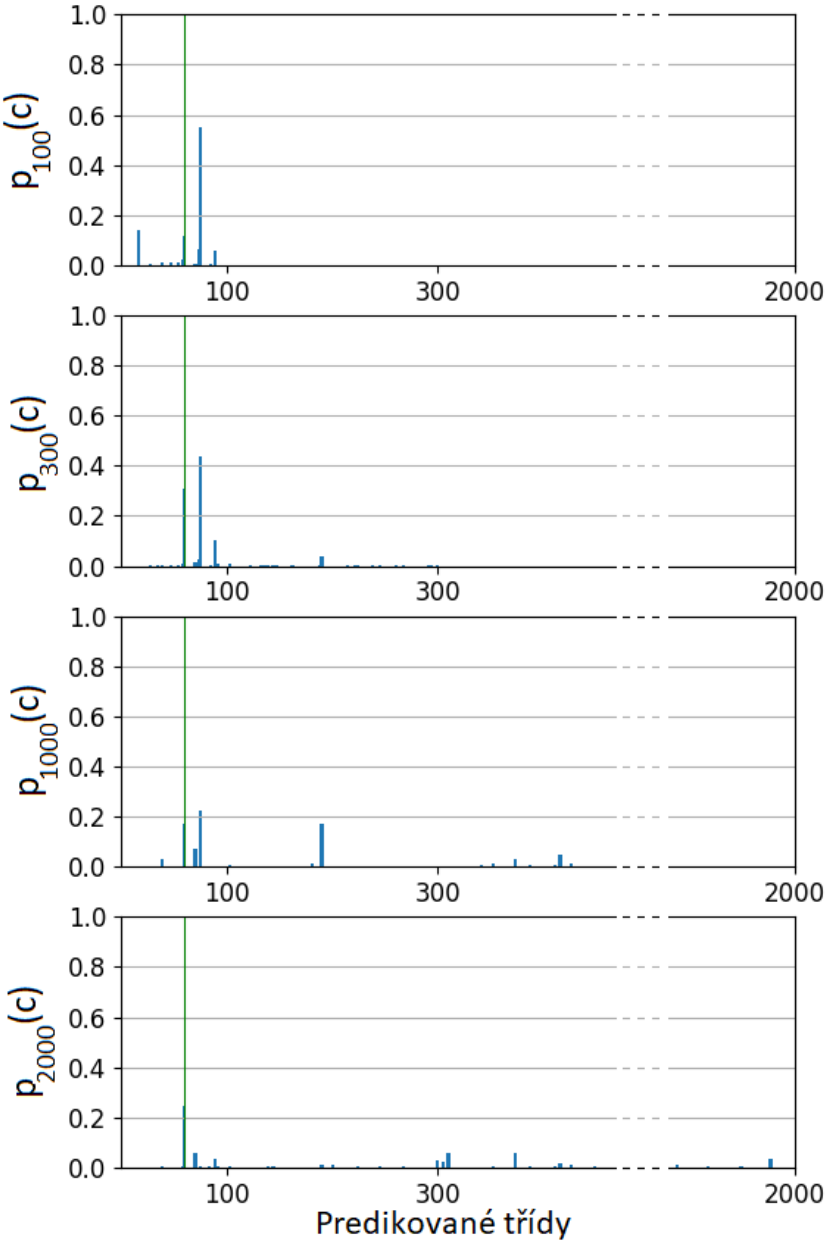
Mimo toto porovnání, je také možné pozorovat, že dosažená přesnost klasifi-

kace je na poměrně nízké úrovni - na nejmenším datasetu WLASL-100 je *top-1 accuracy* 44.57% a na největším datasetu WLASL-2000 je *top-1 accuracy* pouze 21.79%. Dále je pozoruhodné, že při $k = 5$ dochází oproti $k = 1$ zlepšení okolo 30% a při $k = 10$ je zlepšení přesnosti okolo 40%. Toto naznačuje, že modely predikují s větší jistotou několik tříd, mezi kterými nerozezná finální správnou odpověď. Toto tvrzení je dále prozkoumáno na obrázku 4.4, kde je vyobrazen detail distribuce výstupních pravděpodobností naučených modelů na testovacím vzorku třídy, která je zastoupená ve všech čtyřech datasetech, na kterých jsou modely naučeny. V grafu je správná třída označena zeleně a modře jsou vyznačeny výstupní distribuce modelů. Z grafu je vidět potvrzení předchozího tvrzení, modely svou predikci často omezí na výběr menšího množství tříd, mezi kterými nerozliší správnou třídu. Tento jev může být způsoben prezencí různých dialektů znaků v rámci jednotlivých tříd, nebo také podobností znaků různých tříd mezi sebou. Dalším vysvětlením nízké přesnosti je možnost, že klasifikátory jsou naučeny ignorovat některou z tříd. Pro inspekci této možnosti je možné sledovat výstupní váhy výstupních vrstev modelů. V případě, kdy je model naučen některou ze tříd nepredikovat, očekáváme nízké hodnoty vah daného neuronu. Jelikož váhy sítě jsou dvou-dimenzionální $\mathbf{w}_{C,out} \in \mathbb{R}^{\mathrm{I}\times\mathrm{C}}$, kde I je počet vstupních jednotek a C je počet tříd klasifikátoru, je pro potřeby vizualizace první dimenze zredukovaná a výsledný vektor je normalizovaný na rozsah $\langle 0, 1\rangle$.

$$\begin{aligned} \mathbf{w}_C(c) &= \sum_{i=1}^{I} \mathbf{w}_{C,out}(i, c) \\ \widetilde{\mathbf{w}}_C(c) &= \frac{\mathbf{w}_C(c)}{\max \mathbf{w}_C(c)} \end{aligned} \tag{4.10}$$

Vizualizace vektorů $\widetilde{\mathbf{w}}_C(c)$ jednotlivých klasifikátorů je na obrázku 4.5a. Na těchto grafech lze pozorovat jistou míru nízkých vah pro některé třídy, z celkového množství tříd 8% (ze 100 tříd), 4.3% (z 300 tříd), 4.7% (z 1000 tříd) a 1% (z 2000 tříd) jsou v tomto vektoru reprezentovány menší hodnotou než 0.2. Tyto hodnoty na první pohled nenaznačují možnost, že modely nepredikují větší množství tříd, čímž by drasticky ovlivnily výslednou evaluaci. Je tedy obtížné stanovit z jakého důvodu je klasifikační přesnost na poměrně nízké úrovni. Obecně univerzálním řešením takového problému může být rozšíření databáze pro naučení modelů. Aktuálně je v datasetu zastoupena každá třída průměrně 10-ti videi, které interpretují různí interpreti s různými dialekty. Pravděpodobně je pro zvýšení kvality klasifikace potřeba rozšíření celkové databáze jednotlivých znaků nebo redukce zastoupení dialektů.

Na vstupu klasifikačních modelů je umístěna plně propojená vrstva transformující vstupní příznakový vektor do vnitřního stavu modelu, vstupní vrstva tedy provádí formu extrakce příznaků definovanou maticí vah vrstvy $\mathbf{w}_{C,in} \in \mathbb{R}^{\mathrm{I}\times\mathrm{O}}$, kde O je počet neuronů vrstvy a I je velikost vstupního příznakového vektoru. Inspekcí



Obr. 4.4: Detail distribuce výstupních pravděpodobností klasifikátorů

těchto vah můžeme získat informaci, které příznaky jsou pro klasifikaci důležité. Matici opět pro potřeby vizualizace transformujeme do jedno-dimenzionálního prostoru a normujeme na rozsah $\langle 0, 1 \rangle$.

$$\begin{aligned} \mathbf{w}_C(i) &= \sum_{j=1}^{I} \mathbf{w}_{C,in}(i,j) \\ \widetilde{\mathbf{w}}_C(i) &= \frac{\mathbf{w}_C(i)}{\max \mathbf{w}_C(i)} \end{aligned} \tag{4.11}$$

Na základě vektoru $\tilde{\mathbf{w}}_C(i)$ jsou sestrojeny grafy na obrázku 4.5b, kde jsou vykresleny váhy jednotlivých klíčových bodů rukou (2×H1-H21) a obličeje (F1-F98). Z grafů lze pozorovat menší hodnoty vah klíčových bodů reprezentující zápěstí rukou, které samy o sobě pravděpodobně nepřináší klasifikátoru další informaci. Krom tohoto je velikost vah poměrně uniformní.

Dále jsou naučené modely evaluovány také podle své výpočetní náročnosti. Výsledky této evaluace jsou zaneseny v tabulce 4.12. Z výsledků lze pozorovat očekávanou nízkou výpočetní náročnost, danou převážně malou velikostí navržených modelů.

Konfigurace	**#Parametrů**	**FLOPs**	**GPU Latence**	**CPU Latence**
WLASL-100	$50.93{\cdot}10^3$	$3.61{\cdot}10^6$	3.38 ms	2.20 ms
WLASL-300	$63.42{\cdot}10^3$	$3.63{\cdot}10^6$	3.40 ms	2.12 ms
WLASL-1000	$107.53{\cdot}10^3$	$3.70{\cdot}10^6$	5.10 ms	2.14 ms
WLASL-2000	$142.26{\cdot}10^3$	$3.75{\cdot}10^6$	4.38 ms	3.02 ms

Tab. 4.13: Výpočetní náročnost klasifikátorů

4.5 Obecné shrnutí výsledků

V rámci kompletního systému jsou integrovány algoritmy detekce, estimace pózy, sledování objektů, klasifikace a automatické interpretace. Z hlediska výpočetní náročnosti jsou nejnáročnější algoritmy detekce a estimace pózy, které při použití nejrychlejší konfigurace (detektor *YOLOv5-S* a estimátory s extraktory *EfficientNet-B0*) na použité výpočetní jednotce způsobují latenci 113 ms. Následující algoritmy sledování objektů, klasifikace a interpretace pracují v řádech jednotek milisekund. Celkový systém byl testován pro použití v reálném čase na CPU a byla ověřena funkčnost systému. Z hlediska výpočetní náročnosti je pak možné docílit menších latencí modelů při použití hardwarové akcelerace grafickými akcelerátory nebo tensorovými akcelerátory, které dnes řada moderních mobilních zařízení má integrované na desce, právě pro potřeby inference modelů neuronových sítí.

Z hlediska přesnosti jednotlivých algoritmů, jsou algoritmy detekce a estimace klíčových bodů vhodné pro použití, dosahují dostatečných přesností na testovacích sadách, a i při experimentech byla jejich efektivita potvrzena. Přesnost těchto algoritmů doplňuje algoritmus pro sledování objektů a filtraci chyb. Problémovou částí celého systému je část klasifikace. Pro správnou interpretaci predikovaného znaku v cílovém jazyce je potřeba stanovit přesně třídu odpovídající danému rozpoznávanému znaku. Vzhledem k tomu, že model rozpoznávající znaky z nejmenší slovní

zásoby (100 znaků) dosahuje *top-1 accuracy* pouze 44%, nelze kompletní systém prohlásit za plně fungující.

4.6 Návaznost práce

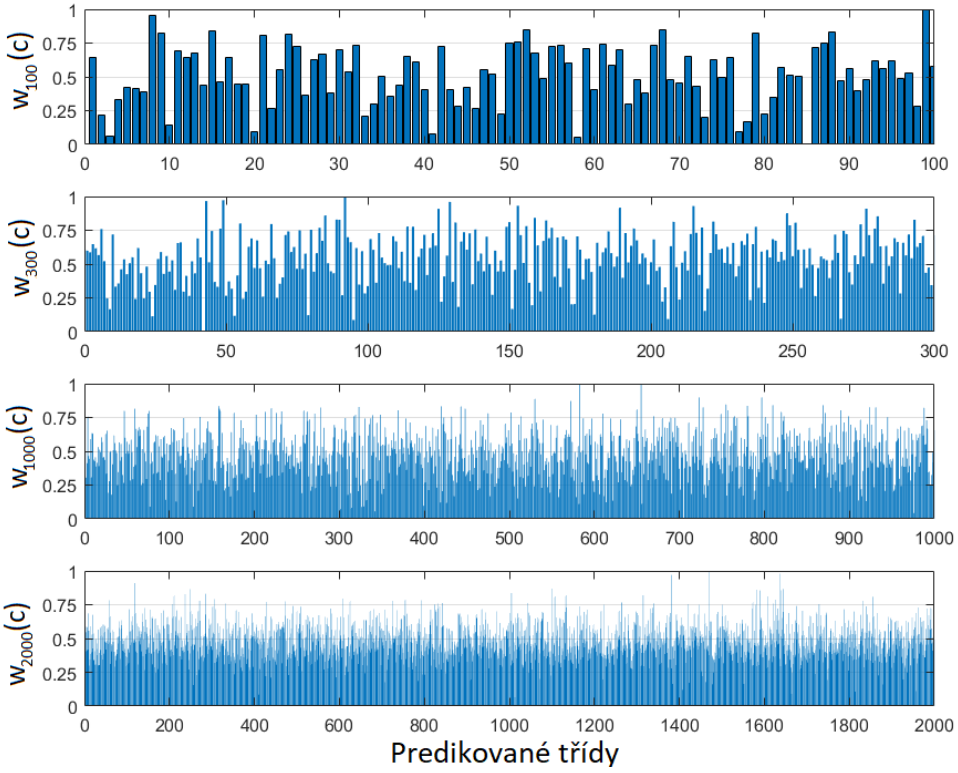
V rámci návaznosti se nabízí mnoho možností. Celý systém byl navržen takovým způsobem, aby bylo jednoduché jednotlivé části systému vyměnit za případné lepší varianty.

Z hlediska výpočetní náročnosti inference modelů se řada prací soustředí na využití technik prořezávání neuronových sítí [115] nebo využití řídkého formátu matic [116], čímž jsou modely optimalizovány pro použití na mobilních zařízení. V případě inference použitím grafického akcelerátoru je vhodné využít jeden z nástrojů pro optimalizaci modelů pro použití na dané hardwarové jednotce, např. *NVIDIA TensorRT* [117]. V případě, kdy by systém byl umístěn do produkce, by bylo vhodné se celkovou optimalizací zabývat.

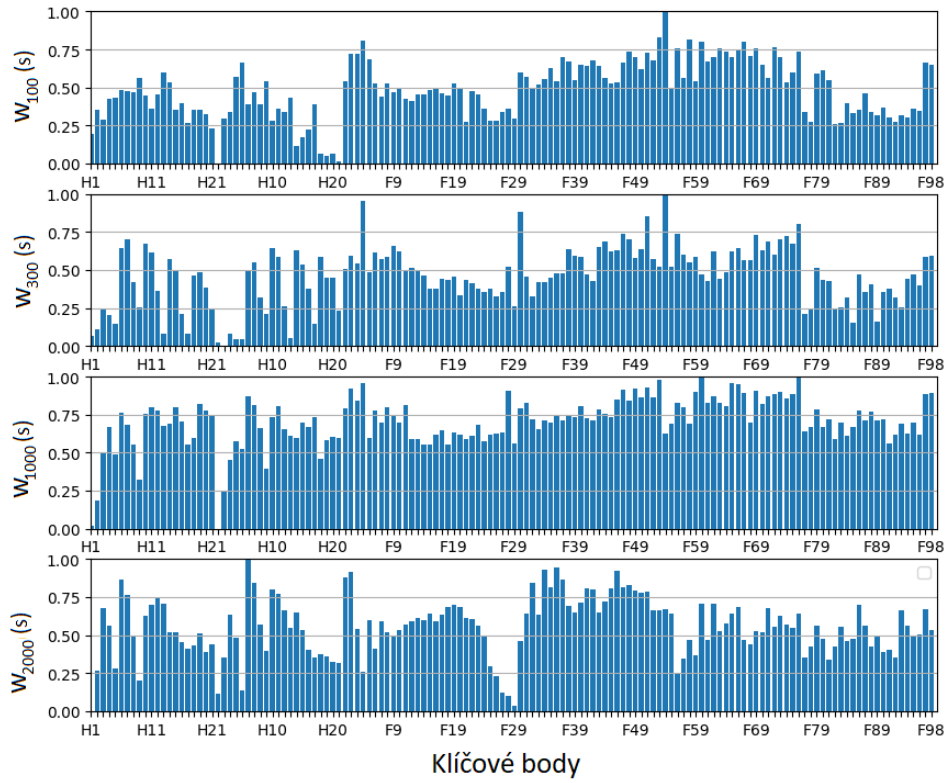
Z hlediska vylepšení jednotlivých algoritmů by bylo vhodné implementovat namísto chyby entropie pro učení detektorů *focal* chybovou funkci [118], která je obecně známá pro svou efektivitu při učení objektových detektorů.

V rámci estimace klíčových bodů obličeje dochází k nepřesnostem estimací v oblastech očí, rtů a nosu. Tyto oblasti je možné lokálně zpřesnit [119] tak, aby bylo z mimiky lidského obličeje při interpretaci získáno maximální množství informace.

Dále je široký prostor pro vylepšení klasifikace znakového jazyka. V tomto směru by bylo možné rekurentní klasifikátor nahradit grafovým klasifikátorem, pomocí kterého autoři WLASL datasetu [86] dosáhli lepších výsledků. Ovšem výsledky *top-1 accuracy* jsou obecně lepší pouze až o 9%, celkově dosahující hodnoty přesnosti 55% na nejlepším modelu, což stále není dostatečné. V tomto směru není v tuto chvíli jasné, jakým způsobem by bylo možné dosáhnout lepších výsledků.



(a) Výstupní váhy tříd



(b) Vstupní váhy příznaků

Obr. 4.5: Váhy klasifikátorů

Obr. 4.6: Příklady výsledků algoritmů zpracovávající statické snímky

Závěr

V rámci této práce jsou prozkoumány metody aplikace algoritmů umělých neuronových sítí pro potřeby rozpoznávání znakového jazyka. V rámci návrhu řešení byla provedena rešerše stávajících metod zabývající se touto problematikou. Na základě poznatků z průzkumu je navržen řetězec zpracování vizuální informace pro porozumění interpretujícímu člověku. Jako aplikace navrženého systému byla zvolena automatická interpretace sledovaného amerického znakového jazyka v českém znakovém jazyce pomocí počítačového modelu.

Z důvodu vysoké rozmanitosti používaných znakových jazyků ve světě je při návrhu systémů zabývající se rozpoznávání interpretovaných výrazů potřeba hledět na robustnost návrhu z hlediska dostupnosti dat jednotlivých jazyků. Z tohoto důvodu je adoptován přístup popisu interpretujícího člověka klíčovými body. Z hlediska druhu použitého snímače se práce zaměřuje na algoritmy zpracovávající informaci z jedné RGB kamery, primárně z důvodu široké dostupnosti těchto kamer a jejich integraci v mobilních a jiných komerčně dostupných zařízeních. Úloha popisu člověka je rozdělena na části segmentace rukou a obličeje člověka a následného popisu segmentovaných částí. Detekované ruce a obličej interpreta a jejich klíčové body jsou pak sledovány v čase a filtrovány lineárním Kalmanovým filtrem. Na základě tohoto popisu je pak interpretovaný jazyk klasifikován a interpretován počítačovým modelem v cílovém znakovém jazyce.

Úloha segmentace rukou a obličeje člověka byla pojata jako úloha detekce a je řešena pomocí architektur objektových detektorů konvolučních neuronových sítí. Pro potřeby jednorázové detekce rukou i obličeje ve snímcích je sestaven dataset skládající se ze dvou datasetů anotovaných pro potřeby samostatné detekce rukou a obličeje. Chybějící anotace byly doplněny detektory naučenými na samostatnou detekci rukou nebo obličejů.

V rámci popisu rukou je estimováno 21 klíčových bodů pro každou ruku ve 2D prostoru. Model dále klasifikuje segmentovaný snímek, zdali se jedná o případ, kdy ruce spolu interagují a estimuje klíčové body obou rukou i v případě kdy ruce spolu interagují. Lidský obličej je popsán 98 body, které jsou estimovány rovněž ve 2D prostoru.

Detekované lidské ruce a obličeje jsou sledovány v dynamické scéně pomocí *Deep SORT* algoritmu, kde je jako metrika použitá pro sestavení asociační matice nákladů použita vzdálenost jednotlivých klíčových bodů objektů. V rámci algoritmu je také využit Kalmanův filtr pro filtraci detekcí a estimací klíčových bodů.

Na základě detekovaných a popsaných objektů je klasifikována interpretovaná znaková řeč. Na základě dostupných datasetů pro rozpoznávání znakové řeči je jako sledovaný jazyk zvolen americký znakový jazyk. V rámci zvoleného datasetu je obsa-

žena slovní zásoba z 2000 znaků, z tohoto datasetu jsou také vybrány části datasetu pro naučení modelů s redukovanou slovní zásobou o 1000, 300 a 100 znaků. Jako klasifikační modely jsou použity čtyři rekurentní neuronové sítě, které jsou naučeny na jednotlivých datasetech.

Pro potřeby automatické interpretace počítačového avatara je použit volně dostupný model, pro který jsou vytvořeny tři příkladné animace znaků české znakové řeči. Model je pak vykreslován v reálném čase včetně animací.

Všechny naučené modely neuronových sítí jsou evaluovány vhodnou metrikou přesnosti i výpočetní náročnosti. Z výsledků evaluací jsou pak vybrány modely, které jsou vhodné pro použití v rámci výsledného řetězce zpracování. Problematickou částí výsledného systému je relativně nízká výsledná přesnost naučených klasifikátorů, z tohoto důvodu je funkčnost výsledného systému omezená.

Výstupem práce je primárně realizovaný řetězec zpracování vizuální informace extrahující informaci o póze lidského obličeje a lidských rukou, a to i v případě, kdy jsou ruce navzájem okludované. Tento řetězec pracuje v reálném čase na procesorové jednotce osobního počítače s možností hardwarové akcelerace. Realizovaná aplikace využívající tento řetězec je ukázkou jednoho z mnoha využití tohoto systému.

Literatura

[1] MACUROVÁ A.: *Poznáváme český znakový jazyk*. Specialní pedagogika. 2001. ISSN 1211-2720. Dostupné z URL: <http://dspace.specpeda.cz/handle/0/621>.

[2] REDLICH K.: *Slyšící dítě hluchých rodičů*. Praha: Univerzita Karlova v Praze, Fylozofická fakulta, Ústav českého jazyka a teorie komunikace, 2004. Bakalářská práce.

[3] FISHER S.: *Sign languages in their Historical Context*. The Routledge Handbook of Historical Linguistics 2015. ISBN 978-0-415-52789-7. Dostupné z URL: <http://dx.doi.org/10.13140/2.1.2085.5683>.

[4] MACUROVÁ A., NOVÁKOVÁ R.: *Poznáváme český znakový jazyk. Český znakový jazyk v kontaktu.*. Specialní pedagogika. 2008. ISSN 1211-2720. Dostupné z URL: <http://dspace.specpeda.cz/handle/0/434>.

[5] GÉRON A.: *Hands on machine learning Scikit-Learn, Keras & Tensorflow*. 2. vydání. O'Reilly Media, Inc. 2019. ISBN 978-1492032649.

[6] MCCULLOCH W., PITTS W.: *A logical calculus of the ideas immanent in nervous activity*. Bulletin of Mathematical Biophysics 5, 115–133, 1943. Dostupné z URL: <https://doi.org/10.1007/BF02478259>.

[7] ROSENBLATT F.: *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review 65(6), 386–408. Dostupné z URL: <https://doi.org/10.1037/h0042519>.

[8] MINSKY M., PAPERT S.: *Perceptrons*. 1969. ISBN 978-0262130431.

[9] HOPFIELD J. J.: *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*. Proceedings of the National Academy of Sciences of the United States of America 79, 2554-2558, 1982. Dostupné z URL: <https://doi.org/10.1073/pnas.79.8.2554>.

[10] RUMELHART D., HINTON G., WILLIAMS R.: *Learning Internal Representations by Error Propagation*. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 318-362. 1986. Dostupné z URL: <https://ieeexplore.ieee.org/document/6302929>.

[11] LECUN Y., BOSER B., DENKER J. S., HENDERSON D., HOWARD R. E., HUBBARD W., JACKEL L. D. : *Backpropagation Applied to Handwritten Zip Code Recognition*. Neural Computation, 4, 541-551. 1989. Dostupné z URL: <https://doi.org/10.1162/neco.1989.1.4.541>.

[12] HEARST M., DUMAIS S, OSUNA E., PLATT J., SCHOLKOPF B.: *Support vector machines*. IEEE Intelligent Systems and their Applications, 13, 18-28. 1998. Dostupné z URL: <https://doi.org/10.1109/5254.708428>.

[13] KRIZHEVSKY A., SUTSKEVER I., HINTON G.: *ImageNet classification with deep convolutional neural networks*. Advances in neural information processing systems, 25, 1097-1105. 2012. Dostupné z URL: <https://dl.acm.org/doi/10.5555/2999134.2999257>.

[14] RUSSAKOVSKY O., DENG J., SU H., KRAUSE J., SATHESSH S., MA S., HUANG. Z, KARPATHY A., KHOSLA A., BERNSTEIN M., BERG A., FEI-FEI L.: *ImageNet Large Scale Visual Recognition Challenge*. International Journal of Computer Vision, 3, 211-252. 2015. Dostupné z URL: <https://doi.org/10.1007/s11263-015-0816-y>.

[15] BLAUS B. *Multipolar Neuron* [online]. Dostupné z URL: <https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png>.

[16] HERBERT R. *A Stochastic Approximation Method*. Annals of Mathematical Statistics, 22, 400-407. 1951. Dostupné z URL: <https://doi.org/10.1214/aoms/1177729586>.

[17] KEIFER J., WOLFOWITZ J. *Stochastic estimation of the maximum of a regression function*. Annals of Mathematical Statistics, 23, 462-466. 1952. Dostupné z URL: <http://www.jstor.org/stable/2236690>.

[18] POLYAK B. *Some methods of speeding up the convergence of iteration methods*. USSR computational mathematics and mathematical physics, 4.5, 1-17. 1964. ISSN 0041-5553. Dostupné z URL: <https://doi.org/10.1016/0041-5553(64)90137-5>.

[19] DUCHI J., HAZAN E., SINGER Y. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. Journal of Machine Learning Research, 12, 2121-2159. 2011. Dostupné z URL: <https://dl.acm.org/doi/10.5555/1953048.2021068>.

[20] ZEILER M. *ADADELTA: An adaptive learning rate method*. 2012. Dostupné z URL:
<https://arxiv.org/abs/1212.5701>.

[21] KINGMA D., BA J. *Adam: A Method for Stochastic Optimization*. International Conference for Learning Representations. 2015. Dostupné z URL:
<https://arxiv.org/abs/1412.6980>.

[22] LEDERER J. *Activation Functions in Artificial Neural Networks: A Systematic Overview*. 2021. Dostupné z URL:
<https://arxiv.org/abs/2101.09957>.

[23] HUBEL D. *Single unit activity in striate cortex of unrestrained cats*. The Journal of Physiology, 147, 226-238. 1958. Dostupné z URL:
<https://doi.org/10.1113/jphysiol.1959.sp006238>.

[24] HUBEL D., WIESEL T. *Receptive fields of single neurones in the cat's striate cortex*. The Journal of Physiology, 148, 574-591. 1959. Dostupné z URL:
<https://doi.org/10.1113/jphysiol.1959.sp006308>.

[25] HUBEL D., WIESEL T. *Receptive fields and functional architecture of monkey striate cortex*. The Journal of Physiology, 195, 215-243. 1968. Dostupné z URL:
<https://doi.org/10.1113/jphysiol.1968.sp008455>.

[26] ZEILER D., KRISHNAN D., TAYLOR W., FERGUS R. *Deconvolutional networks*. IEEE Computer Society Conference on computer vision and pattern recognition, 2528-2535. 2010. Dostupné z URL:
<https://doi.org/10.1109/CVPR.2010.5539957>.

[27] LONG J., SHELHAMER E., DARRELL T. *Fully convolutional networks for semantic segmentation*. IEEE conference on computer vision and pattern recognition, 3431-3440. 2015. Dostupné z URL:
<https://arxiv.org/abs/1411.4038>.

[28] IOFFE S., SZEGEDY C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. International conference on machine learning, PMLR, 448-456. 2015. Dostupné z URL:
<https://arxiv.org/abs/1502.03167>.

[29] HOCHREITER, S., SCHMIDHUBER, J. *Long short-term memory*. Neural computation, 9(8), 1735-1780. 1997. Dostupné z URL:
<https://doi.org/10.1162/neco.1997.9.8.1735>.

[30] CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., BENGIO, Y. *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. Dostupné z URL:
<https://arxiv.org/abs/1406.1078>.

[31] RASTGOO R., KIANI K., ESCALERA S. *Sign Language Recognition: A Deep Survey*. Expert Systems with Applications, 164, 113794. 2021. ISSN 0957-4174. Dostupné z URL:
<https://doi.org/10.1016/j.eswa.2020.113794>.

[32] GIRSHICK R., DONAHUE J., DARELL T., MALIK J. *Region-Based Convolutional Networks for Accurate Object Detection and Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38, 142-158. 2015. Dostupné z URL:
<https://doi.org/10.1109/TPAMI.2015.2437384>.

[33] GIRSHICK R. *Fast R-CNN*. 2015. Dostupné z URL:
<https://arxiv.org/abs/1504.08083>.

[34] REN S., HE K., GIRSHICK R., SUN J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39, 1137-1149. 2017. Dostupné z URL:
<https://doi.org/10.1109/TPAMI.2016.2577031>.

[35] LIU W., ANGUELOV D., ERHAN D., SZEGEDY C., REED S., FU C., BERG A. *SSD: Single Shot MultiBox Detector*. Lecture Notes in Computer Science, 21-37. 2016. Dostupné z URL:
<https://doi.org/10.1007/978-3-319-46448-0_2>.

[36] REDMON J., DIVVALA S., GIRSHICK R., FARHADI A. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. Dostupné z URL:
<https://arxiv.org/abs/1506.02640>.

[37] REDMON J., FARHADI A. *YOLO9000: Better, Faster, Stronger*. 2016. Dostupné z URL:
<https://arxiv.org/abs/1612.08242>.

[38] REDMON J., FARHADI A. *YOLOv3: An Incremental Improvement*. 2018. Dostupné z URL:
<https://arxiv.org/abs/1804.02767>.

[39] BROCHKOVSKIY A., WANG C., LIAO H. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. Dostupné z URL: <https://arxiv.org/abs/2004.10934>.

[40] JOCHER G., STOKEN A., BOROVEC J. a ostatní. *ultralytics/yolov5: v3.1*. 2020. Dostupné z URL: <https://doi.org/10.5281/zenodo.4154370>.

[41] TAN M., PANG R., LE Q. *EfficientDet: Scalable and Efficient Object Detection*. 2020. Dostupné z URL: <https://arxiv.org/abs/1911.09070>.

[42] MUELLER F., BERNARD F., SOTNYCHENKO O., MEHTA D., SRIDHAR S., CASAS D., THEOBALT C. *GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB*. International Conference on Computer Vision & Pattern Recognition. 2018. Dostupné z URL: <https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/>.

[43] ZIMMERMANN C., BROX T. *Learning to Estimate 3D Hand Pose from Single RGB Images*. 2017. Dostupné z URL: <https://lmb.informatik.uni-freiburg.de/projects/hand3d/>.

[44] MUELLER F., SOTNYCHENKO O., MEHTA D., SRIDHAR S., CASAS D., THEOBALT C. *Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor*. International Conference on Computer Vision. 2017. Dostupné z URL: <http://handtracker.mpi-inf.mpg.de/projects/OccludedHands/>.

[45] HASSON Y., VAROL G., TZIONAS D., KALEVATYKH I., BLACK M., LAPTEV I., SCHMID C. *Learning joint reconstruction of hands and manipulated objects*. International Conference on Computer Vision. 2019. Dostupné z URL: <https://www.di.ens.fr/willow/research/obman/data/>.

[46] LIN, F., WILHELM C. MARTINEZ T. *Two-Hand Global 3D Pose Estimation Using Monocular RGB*. Winter Conference on Applications of Computer Vision, 2373-2381. 2021. Dostupné z URL: <https://arxiv.org/abs/2006.01320>.

[47] MOON G., YU S., WEHN H., SHIRATORI T., LEE K. *InterHand2.6M: A Dataset and Baseline for 3D Interacting Hand Pose Estimation from a Single RGB Image*. European Conference on Computer Vision. 2020. Dostupné z URL: <https://arxiv.org/abs/2008.09309>.

[48] ZIMMERMANN C., CEYLAN D., YAND J., RUSSELL B., ARGUS M., BROX T. *FreiHAND: A Dataset for Markerless Capture of Hand Pose and Shape from Single RGB Images*. International Conference on Computer Vision. 2019. Dostupné z URL:
<https://lmb.informatik.uni-freiburg.de/resources/datasets/FreihandDataset.en.html>.

[49] ZIMMERMANN C., ARGUS M., BROX T. *Contrastive Representation Learning for Hand Shape Estimation*. Pattern Recognition - 43st German Conference. 2021. Dostupné z URL:
<https://arxiv.org/abs/2106.04324>.

[50] QIAN C., SUN X., WEI Y., TANG X., SUN J. *Realtime and Robust Hand Tracking from Depth*. International Conference on Computer Vision & Pattern Recognition, 1106-1113. 2014. Dostupné z URL:
<https://doi.org/10.1109/CVPR.2014.145>.

[51] GE L., LIANG H., YUAN J., THALMANN D. *Robust 3D Hand Pose Estimation in Single Depth Images: from Single-View CNN to Multi-View CNNs*. International Conference on Computer Vision & Pattern Recognition. 2016. Dostupné z URL:
<https://arxiv.org/abs/1606.07253>.

[52] YUAN S., YE Q., GARCIA-HERNANDO G., KIM T. *The 2017 Hands in the Million Challenge on 3D Hand Pose Estimation*. International Conference on Computer Vision & Pattern Recognition. 2017. Dostupné z URL:
<https://arxiv.org/abs/1707.02237>.

[53] GARCIA-HERNANDO G., YUAN S., BAEK S., KIM T. *First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations*. International Conference on Computer Vision & Pattern Recognition. 2018. Dostupné z URL:
<https://arxiv.org/abs/1704.02463>.

[54] TZIONAS, D., BALLAN, L., SRIKANTHA, A., APONTE, P., POLLEFEYS, M., GALL, J. *Capturing Hands in Action using Discriminative Salient Points and Physics Simulation*. International Journal of Computer Vision. 2016. Dostupné z URL:
<http://files.is.tue.mpg.de/dtzionas/Hand-Object-Capture>.

[55] BRAHMBHATT S., TANG C., TWIGG C., KEMP C., HAYS J. *ContactPose: A Dataset of Grasps with Object Contact and Hand Pose*. The European Conference on Computer Vision. 2020. Dostupné z URL: <https://arxiv.org/abs/2007.09545>.

[56] SRIDHAR S., OULASVIRTA A., THEOBALT C. *Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor*. International Conference on Computer Vision. 2013. Dostupné z URL: <http://handtracker.mpi-inf.mpg.de/projects/handtracker_iccv2013/>.

[57] MUELLER F., MEHTA D., SOTNYCHENKO O., SRIDHAR S., CASAS D., THEOBALT C. *Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor*. International Conference on Computer Vision. 2017. Dostupné z URL: <http://handtracker.mpi-inf.mpg.de/projects/OccludedHands/>.

[58] SRIDHAR S., MUELLER F., ZOLLHOEFER M., CASAS D., OULASVIRTA A., THEOBALT C. *Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input*. European Conference on Computer Vision. 2016. Dostupné z URL: <http://handtracker.mpi-inf.mpg.de/projects/RealtimeHO/>.

[59] ZHANG J., JIAO J., CHEN M., QU L., XU X., YANG Q. *3D Hand Pose Tracking and Estimation Using Stereo Matching*. 2016. Dostupné z URL: <https://arxiv.org/abs/1610.07214>.

[60] LUO Z., WANG Z., HUANG Y., WANG L., TAN T., ZHOU E. *Rethinking the Heatmap Regression for Bottom-up Human Pose Estimation*. Conference on Computer Vision and Pattern Recognition, 13264-13273. 2021. Dostupné z URL: <https://arxiv.org/abs/2012.15175>.

[61] HAN M., CHEN J., LI L., CHANG Y. *Visual hand gesture recognition with convolution neural network*. International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 287-291. 2016. Dostupné z URL: <https://doi.org/10.1109/SNPD.2016.7515915>.

[62] DADASHZADEH A., TARGHI T., TAHMASBI M., MIRMEHDI M. *HGR-Net: a fusion network for hand gesture segmentation and recognition*. IET Computer

Vision, 700-707. 2019. Dostupné z URL:
<https://arxiv.org/abs/1806.05653>.

[63] ELBOUSHAKI A., HANNANE R., AFDEL K., KOUTTI L. *MultiD-CNN: A multi-dimensional feature learning approach based on deep convolutional networks for gesture recognition in RGB-D image sequences*. Expert Systems with Applications, 139. 2020. Dostupné z URL:
<https://doi.org/10.1016/j.eswa.2019.112829>.

[64] CHEN Y., ZHAO L., PENG X., YUAN J., METAXAS N. *Construct dynamic graphs for hand gesture recognition via spatial-temporal attention*. 2019. Dostupné z URL:
<https://arxiv.org/abs/1907.08871>.

[65] KANG B., TRIPATHI S., NGUYEN Q. *Real-time sign language fingerspelling recognition using convolutional neural networks from depth map*. Asian Conference on Pattern Recognition, 136-140. 2015. Dostupné z URL:
<https://arxiv.org/abs/1509.03001>.

[66] ISAACS J., FOO S. *Hand pose estimation for American sign language recognition*. Thirty-Sixth Southeastern Symposium on System Theory, 132-136. 2004. Dostupné z URL:
<https://doi.org/10.1109/SSST.2004.1295634>.

[67] MORYOSSEF A., TSOCHANTARIDIS I., AHARONI R., EBLING S., NARAYANAN S. *Real-time sign language detection using human pose estimation*. European Conference on Computer Vision, 237-248. 2020. Dostupné z URL:
<https://arxiv.org/abs/2008.04637>.

[68] SAGONAS C., TZIMIROPOULOS G., ZAFEIRIOU S., PANTIC M. *300 faces in-the-wild challenge: The first facial landmark localization challenge*. International Conference on Computer Vision Workshops, 397-403. 2013. Dostupné z URL:
<https://doi.org/10.1109/ICCVW.2013.59>.

[69] WU W., QIAN C., YANG S., WANG Q., CAI Y., ZHOU Q. *Look at Boundary: A Boundary-Aware Face Alignment Algorithm*. International Conference on Computer Vision & Pattern Recognition, 2129-2138. 2018. Dostupné z URL:
<https://openaccess.thecvf.com/content_cvpr_2018/html/Wu_Look_at_Boundary_CVPR_2018_paper.html>.

[70] KOESTINGER M., WOHLHART P., ROTH M., BISCHOF H. *Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization*. IEEE international conference on computer vision workshops, 2144-2151. 2011. Dostupné z URL: <https://doi.org/10.1109/ICCVW.2011.6130513>.

[71] BURGOS-ARTIZZU P., PERONA P., DOLLÁR P. *Robust face landmark estimation under occlusion*. IEEE international conference on computer vision, 1513-1520. 2013. Dostupné z URL: <https://openaccess.thecvf.com/content_iccv_2013/html/Burgos-Artizzu_Robust_Face_Landmark_2013_ICCV_paper.html>.

[72] RAO A., SYAMALA K., KISHORE V., SASTRY S. *Deep convolutional neural networks for sign language recognition*. Conference on Signal Processing And Communication Engineering Systems, 194-197. 2018. Dostupné z URL: <https://doi.org/10.1109/SPACES.2018.8316344>.

[73] KOLLER O., NEY H., BOWDEN R. *Deep learning of mouth shapes for sign language*. IEEE International Conference on Computer Vision Workshops, 85-91. 2015. Dostupné z URL: <https://www.cv-foundation.org/openaccess/content_iccv_2015_workshops/w12/html/Koller_Deep_Learning_of_ICCV_2015_paper.html>.

[74] PFISTER T., CHARLES J., ZISSERMAN A. *Large-scale Learning of Sign Language by Watching TV (Using Co-occurrences)*. BMVC. 2013. Dostupné z URL: <https://doi.org/10.5244/C.27.20>.

[75] ZHENG C., WU W., YANG T., ZHU S., CHEN C., LIU R., SHEN J., KEHTARNAVAZ N., SHAH M. *Deep Learning-Based Human Pose Estimation: A Survey*. 2021. Dostupné z URL: <https://arxiv.org/abs/2012.13392>.

[76] IONESCU C., PAPAVA D., OLARU V., SMINCHISESCU C. *Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments*. IEEE transactions on pattern analysis and machine intelligence, 36, 1325-1339. 2013. Dostupné z URL: <https://doi.org/10.1109/TPAMI.2013.248>.

[77] MEHTA D., SOTNYCHENKO O., MUELLER F., XU W., SRIDHAR S., PONS-MOLL G., THEOBALT C. *Single-shot multi-person 3d body pose estimation from monocular rgb input*. International Conference on 3D Vision

(3DV), 120-130. 2018. Dostupné z URL:
<https://doi.org/10.1109/3DV.2018.00024>.

[78] VON MARCARD T., HENSCHEL R., BLACK J., ROSENHAHN B., PONS-MOLL G. *Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera*. European Conference on Computer Vision, 601-617. 2018. Dostupné z URL:
<https://doi.org/10.1007/978-3-030-01249-6_37>.

[79] ANDRILUKA M., PISHCHULIN L., GEHLER P., SCHIELE B. *2D Human Pose Estimation: New Benchmark and State of the Art Analysis*. IEEE Conference on computer Vision and Pattern Recognition, 3686-3693. 2014. Dostupné z URL:
<https://doi.org/10.1109/CVPR.2014.471>.

[80] DUAN H., LIN Y., JIN S., LIU W., QIAN C., OUYANG W. *TRB: A Novel Triplet Representation for Understanding 2D Human Body*. IEEE/CVF International Conference on Computer Vision, 9479-9488. 2019. Dostupné z URL:
<https://arxiv.org/abs/1910.11535>.

[81] LI J., WANG C., ZHU H., MAO Y., FANG S., LU C. *CrowdPose: Efficient Crowded Scenes Pose Estimation and A New Benchmark*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10863-10872. 2019. Dostupné z URL:
<https://arxiv.org/abs/1812.00324>.

[82] ZHANG H., LI R., DONG X., ROSIN P., CAI Z., HAN X., YANG D., HUANG H., HU S. *Pose2Seg: Detection Free Human Instance Segmentation*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 889-898. 2019. Dostupné z URL:
<https://arxiv.org/abs/1803.10683>.

[83] ANDRILUKA M., IQBAL U., INSAFUTDINOV E., PISHCHULIN L., MILAN A., GALL J., SCHIELE B. *Posetrack: A benchmark for human pose estimation and tracking*. IEEE conference on computer vision and pattern recognition, 5167-5176. 2018. Dostupné z URL:
<https://arxiv.org/abs/1710.10000>.

[84] DE COSTER M., VAN HERREWEGHE M., DAMBRE J. *Isolated Sign Recognition from RGB Video using Pose Flow and Self-Attention*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3441-3450. 2021.

Dostupné z URL:
<https://doi.org/10.1109/CVPRW53098.2021.00383>.

[85] KONSTANTINIDIS D., DIMITROPOULOS K., DARAS P. *A Deep Learning Approach for Analyzing Video and Skeletal Features in Sign Language Recognition*. IEEE International Conference on Imaging Systems and Techniques, 1-6. 2018. Dostupné z URL:
<https://doi.org/10.1109/IST.2018.8577085>.

[86] LI D., RODRIGUEZ C., YU X., LI H. *Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison*. IEEE/CVF winter conference on applications of computer vision, 1459-1469. 2020. Dostupné z URL:
<https://arxiv.org/abs/1910.11006>.

[87] GATTUPALLI S., GHADERI A., ATHITSOS V. *Evaluation of deep learning based pose estimation for sign language recognition*. ACM international conference on pervasive technologies related to assistive environments, 1-7. 2016. Dostupné z URL:
<https://doi.org/10.1145/2910674.2910716>.

[88] PARELLI M., PAPADIMITRIOU K., POTAMIANOS G., PAVLAKOS G., MARAGOS P. *Exploiting 3d hand pose estimation in deep learning-based sign language recognition from RGB videos*. European Conference on Computer Vision, 249-263. 2020. ISBN 978-3-030-66096-3. Dostupné z URL:
<https://doi.org/10.1007/978-3-030-66096-3_18>.

[89] *SignAll Technologies*. [online]. [cit. 2021-12-18]. Dostupné z URL:
<https://www.signall.us/>.

[90] *GnoSys app translates sign language into speech in real time using the power of AI*. [online]. [cit. 2021-12-18]. Dostupné z URL:
<https://newzhook.com/story/20387/>.

[91] TSOTSIS A. *MotionSavvy Is A Tablet App That Understands Sign Language*. [online]. [cit. 2021-12-18]. Dostupné z URL:
<https://techcrunch.com/2014/06/06/motionsavvy-is-a-tablet-app-that-understands-sign-language/>.

[92] *Hand Talk*. [online]. [cit. 2021-12-18]. Dostupné z URL:
<https://handtalk.me/en>.

[93] WOJKE N., BEWLEY A., PAULUS D. *Simple online and realtime tracking with a deep association metric*. 2017 IEEE international conference on image processing (ICIP), 3645-3649. 2017. Dostupné z URL: <https://arxiv.org/abs/1703.07402>.

[94] SUTTON V. *SignWriting*. [online]. [cit. 2021-12-28]. Dostupné z URL: <https://www.signwriting.org/>.

[95] *Unicode*. [online]. [cit. 2021-12-28]. Dostupné z URL: <https://home.unicode.org/>.

[96] VAN ROSSUM G., DRAKE F. *Python 3 Reference Manual*. 2009. ISBN 1441412697. Dostupné z URL: <https://docs.python.org/3/reference/>.

[97] ABADI M., AGARWAL A., BARHAM P., BREVDO E. a ostatní *TensorFlow: Large-scale machine learning on heterogeneous systems*. 2015. Dostupné z URL: <https://www.tensorflow.org/>.

[98] SHAN D., GENG J., SHU M., FOUHEY F. *Understanding human hands in contact at internet scale*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9869-9878. 2020. Dostupné z URL: <https://arxiv.org/abs/2006.06669>.

[99] YANG S., LUO P., LOY C., TANG X. *WIDER FACE: A Face Detection Benchmark*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 5525-5533. 2016. Dostupné z URL: <https://arxiv.org/abs/1511.06523>.

[100] LIN T. Y., MAIRE M., BELONGIE S., HAYS J., PERONA P., RAMANAN D., BOURDEV L., GIRSHICK R., RAMANAN D., DOLLÁR P., ZITNICK C. L. *Microsoft coco: Common objects in context*. European conference on computer vision, 740-755. 2014. Dostupné z URL: <https://arxiv.org/abs/1405.0312>.

[101] JIN S., XU L., XU J., WANG C., LIU W., QIAN C., OUYANG W., LUO P. *Whole-Body Human Pose Estimation in the Wild*. European Conference on Computer Vision (ECCV), 196-214. 2020. Dostupné z URL: <https://arxiv.org/abs/2007.11858>.

[102] ZHU H., FANG H. *Halpe Full-Body Human Keypoints and HOI-Det dataset*. [online]. Dostupné z URL:

<https://github.com/Fang-Haoshu/Halpe-FullBody/blob/master/docs/hand.jpg>.

[103] FANG S., XIE S., TAI W., LU C. *RMPE: Regional Multi-person Pose Estimation*. IEEE international conference on computer vision, 2334-2343. 2017. Dostupné z URL:
<https://arxiv.org/abs/1612.00137>.

[104] LI Y., XU L., LIU X., HUANG X., XU Y., WANG S., FANG H., MA Z., CHEN M., LU C. *PaStaNet: Toward Human Activity Knowledge Engine*. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 382-391. 2020. Dostupné z URL:
<https://arxiv.org/abs/2004.00945>.

[105] SIMON T., JOO H., MATTHEWS I., SHEIKH Y. *Hand Keypoint Detection in Single Images using Multiview Bootstrapping*. IEEE conference on Computer Vision and Pattern Recognition, 1145-1153. 2017. Dostupné z URL:
<https://arxiv.org/abs/1704.07809>.

[106] TAN M., LE Q. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. International Conference on Machine Learning, 6105-6114. 2019. Dostupné z URL:
<https://arxiv.org/abs/1905.11946>.

[107] FEIYU Y., ZHAN S., ZHENZHONG X., YAOYANG M., YU C., ZHE P., ZHANG Y., BEIBEI Q., WU J. *Error Compensation Heatmap Decoding for Human Pose Estimation*. IEEE Access, 9, 114514-114522. 2021. Dostupné z URL:
<https://doi.org/10.1109/ACCESS.2021.3105553>.

[108] BEWLEY A., GE Z., OTT L., RAMOS F., UPCROFT B. *Simple online and realtime tracking*. IEEE international conference on image processing (ICIP), 3464-3468. 2016. Dostupné z URL:
<https://doi.org/10.1109/ICIP.2016.7533003.>.

[109] KALMAN R. E. *A new approach to linear filtering and prediction problems*. J. Basic Eng, 35-45. 1960. Dostupné z URL:
<https://doi.org/10.1115/1.3662552>.

[110] GLOROT X., BENGIO Y. *Understanding the difficulty of training deep feedforward neural networks*. International conference on artificial intelligence and statistics, 249-256. 2010. Dostupné z URL:
<https://proceedings.mlr.press/v9/glorot10a.html>.

[111] *Adobe Mixamo - Animate 3D characters for games, film, and more*. [online]. [cit. 2022-5-1]. Dostupné z URL:
<https://www.mixamo.com/>.

[112] *Blender - a 3D modelling and rendering package*. [online]. [cit. 2022-5-1]. Dostupné z URL:
<https://www.blender.org/>.

[113] *Panda3D - The open source framework for 3D rendering and games*. [online]. [cit. 2022-5-1]. Dostupné z URL:
<https://www.panda3d.org/>.

[114] EVERINGHAM M., VAN GOOL L., WILLIAMS K., WINN J., ZISSERMAN A. *The PASCAL Visual Object Classes (VOC) challenge*. International journal of computer vision, 88, 303-338. 2010. Dostupné z URL:
<https://doi.org/10.1007/s11263-009-0275-4>.

[115] LI H., KADAV A., DURDANOVIC I., SAMET H., GRAF H. P. *Pruning filters for efficient convnets*. 2016. Dostupné z URL:
<https://arxiv.org/abs/1608.08710>.

[116] YAN Y., MAO Y., LI B. *SECOND: Sparsely Embedded Convolutional Detection*. Sensors, 18(10), 3337. 2018. Dostupné z URL:
<https://doi.org/10.3390/s18103337>.

[117] *NVIDIA TensorRT*. [online]. [cit. 2022-5-4]. Dostupné z URL:
<https://developer.nvidia.com/tensorrt>.

[118] LIN T. Y., GOYAL P., GIRSHICK R., HE K., DOLLÁR P. *Focal loss for dense object detection*. Proceedings of the IEEE international conference on computer vision, 2980-2988. 2016. Dostupné z URL:
<https://arxiv.org/abs/1708.02002v2>.

[119] GRISHCHENKO I., ABLAVATSKI A., KARTYNNIK Y., RAVEENDRAN K., GRUNDMANN M. *Attention Mesh: High-fidelity Face Mesh Prediction in Real-time*. 2020. Dostupné z URL:
<https://arxiv.org/abs/2006.10962>.

Seznam symbolů a zkratek

LSF	francouzský znakový jazyk – Langue de Signe Française
LUT	lineární prahovaná jednotka – Linear Threshold Unit
MLP	vícevrstvý perceptron – Multi-Layer Perceptron
SGD	Stochastic Gradient Descent
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
2D	Dvoudimenzionální
3D	Trojdimenzionální
RGB	Reg Green Blue
RCNN	Region-based Convolutional Neural Network
SSD	Single Shot Detector
YOLO	You Only Look Once
SOTA	State of the Art
FPN	Feature Pyramid Network
NMS	Non-aximum Suppression
IOU	Intersection Over Union
SVD	dekompozice singulárních hodnot – Singular Value Decomposition
CPU	procesorová jednotka – Central Processing Unit
GPU	grafická jednotka – Graphical Processing Unit
TPU	tensorová jednotka – Tenor Processing Unit
FLOPS	operace nad čísly s pohyblivou desetinnou čárkou – Floating point Operations
AP	průměrná přesnost – Average Precision
P	přesnost – Precision

R	poměr vybavení – Recall
TP	správně označen – True Positive
FP	nesprávně označen – False Positive
TN	správně neoznačen – True Negative
FN	nesprávně neoznačen – False Negative
mAP	střední průměrná přesnost – mean Average Precision
MAE	průměrná absolutní chyba – Mean Absolute Error
ROC	Reciever operating characteristic
FPR	False Positive Rate

A Obsah elektronické přílohy

V rámci elektronické přílohy je přiloženo programové vybavení vytvořené v rámci řešení práce. Příloha je strukturovaná do několika subprojektům, které řeší v izolaci danou úlohu. Tyto subprojekty jsou:

- Detekce
- Estimace pózy
- Sledování objektů
- Překlad znakové řeči
- Počítačový model

Součástí všech subprojektů jsou README soubory, ve kterých je stručně popsána funkcionalita jednotlivých skriptů a doporučený postup pre replikaci výsledků dosažených touto prací. Uvedený README soubor je také k nalezení v hlavním adresáři přílohy, kde je popsána struktura celého projektu a postup pro instalaci použitých knihoven. Základní struktura subprojektů je v adresářovém formátu:

```
subprojekt ..... kořenový adresář subprojektu
├── data ..... data použitá v rámci subprojektu
├── modules ..... knihovny subprojektu
│   └── <knihovny>.py
├── weights ..... váhy modelů subprojektu
├── README.md ..... README soubor subprojektu
└── <skripty>.py ..... spustitelné skripty
```

Pozn.: Z důvodu nedostatečné kapacity elektronické přílohy, jsou některé objemné části přílohy přiloženy pouze na přiloženém DVD disku. Jedná se především o váhy naučených modelů, které mají až stovky megabytů.

A.1 Adresářová struktura přílohy

```
/ ..... kořenový adresář přiloženého archivu
├── avatar ..... subprojekt počítačového modelu
├── detection ..... subprojekt detekce
├── pose_estimation ..... subprojekt estimace pózy
├── tracking ..... subprojekt sledování objektů
├── translation ..... subprojekt překladu znakové řeči
├── data
│   └── readme
│       └── example.gif
├── README.md ..... README soubor projektu
├── requirements.txt ..... seznam použitých Python knihoven
├── images.py
├── videos.py
└── webcam.py
```

```
detection ..... kořenový adresář subprojektu detekce
├── data
│   ├── anchors
│   │   ├── combined
│   │   │   ├── anchors.txt
│   │   │   └── enfficientdet_anchors.txt
│   │   ├── faces
│   │   │   └── anchors.txt
│   │   └── hands
│   │       └── anchors.txt
│   └── readme
│       ├── example.gif
│       └── example.jpg
├── modules ..... knihovny subprojektu
│   ├── __init__.py
│   ├── config.py
│   ├── dataset_utils.py
│   ├── detector_utils.py
│   ├── efficientdet.py
│   ├── inference_utils.py
│   ├── mean_average_precision.py
│   └── yolov5.py
├── weights ..... váhy naučených modelů
│   ├── combined_efficientdet_d0.tf.data-00000-of-00001
│   ├── combined_efficientdet_d0.tf.index
│   ├── combined_efficientdet_d1.tf.data-00000-of-00001
│   ├── combined_efficientdet_d1.tf.index
│   ├── combined_efficientdet_d2.tf.data-00000-of-00001
│   ├── combined_efficientdet_d2.tf.index
│   ├── combined_yolov5l.tf.data-00000-of-00001
│   ├── combined_yolov5l.tf.index
│   ├── combined_yolov5m.tf.data-00000-of-00001
│   ├── combined_yolov5m.tf.index
│   ├── combined_yolov5s.tf.data-00000-of-00001
│   ├── combined_yolov5s.tf.index
│   ├── faces_yolov5s.tf.data-00000-of-00001
│   ├── faces_yolov5s.tf.index
│   ├── hands_yolov5s.tf.data-00000-of-00001
│   └── hands_yolov5s.tf.index
├── README.md ..... README soubor subprojektu
├── conbine_datasets.py
├── construct_tfrecords.py
├── convert_weights.py
├── performance.py
├── train.py
└── videos.py
```

```
├── visualize_dataset.py
└── webcam.py

pose_estimation ..... kořenový adresář subprojektu estimace pózy
├── data
│   └── readme
│       └── example.jpg
├── modules ..... knihovny subprojektu
│   ├── __init__.py
│   ├── coco_utils.py
│   ├── config.py
│   ├── estimation_utils.py
│   ├── face_model.py
│   ├── halpe_utils.py
│   ├── hands_model.py
│   ├── hands_utils.py
│   ├── inference_utils.py
│   ├── interhand.py
│   ├── interhand_utils.py
│   ├── metrics.py
│   ├── mwb_utils.py
│   ├── rhd_utils.py
│   └── wflw_utils.py
├── weights ..... váhy naučených modelů
│   ├── effnetb0_hands.tf.data-00000-of-00001
│   ├── effnetb0_hands.tf.index
│   ├── effnetb0_wflw.tf.data-00000-of-00001
│   ├── effnetb0_wflw.tf.index
│   ├── effnetb1_hands.tf.data-00000-of-00001
│   ├── effnetb1_hands.tf.index
│   ├── effnetb1_wflw.tf.data-00000-of-00001
│   ├── effnetb1_wflw.tf.index
│   ├── effnetb2_hands.tf.data-00000-of-00001
│   ├── effnetb2_hands.tf.index
│   ├── effnetb2_wflw.tf.data-00000-of-00001
│   └── effnetb2_wflw.tf.index
├── README.md ..... README soubor subprojektu
├── contruct_tfrecords.py
├── convert_weights.py
├── evaluate.py
├── interhand_residuals_viz.m
├── performance.py
├── train.py
├── visualize_dataset.py
├── visualize_outputs.py
└── webcam.py
```

```
tracking ..... kořenový adresář subprojektu sledování objektů
├── modules ..... knihovny subprojektu
│   ├── __init__.py
│   ├── kalman_filter.py
│   ├── tracker.py
│   ├── tracker_utils.py
│   └── utils.py
└── README.md ..... README soubor subprojektu

translation ..... kořenový adresář subprojektu překladu znakové řeči
├── data
├── modules ..... knihovny subprojektu
│   ├── __init__.py
│   ├── config.py
│   ├── dataset_utils.py
│   ├── inference_utils.py
│   ├── metrics.py
│   ├── model.py
│   └── wlasl_utils.py
├── weights ..... váhy naučených modelů
│   ├── wlasl100.tf.data-00000-of-00001
│   ├── wlasl100.tf.index
│   ├── wlasl300.tf.data-00000-of-00001
│   ├── wlasl300.tf.index
│   ├── wlasl1000.tf.data-00000-of-00001
│   ├── wlasl1000.tf.index
│   ├── wlasl2000.tf.data-00000-of-00001
│   └── wlasl2000.tf.index
├── README.md ..... README soubor subprojektu
├── annotate_data.py
├── construct_tfrecords.py
├── evaluate.py
├── performance.py
├── train.py
├── visualize_dataset.py
├── visualize_model.py
└── vocabulary.py
```

```
avatar ..... kořenový adresář subprojektu počítačového modelu
├── data
│   ├── readme
│   │   ├── animation_names.png
│   │   ├── avatar.gif
│   │   ├── export_1.png
│   │   ├── export_2.png
│   │   ├── model.png
│   │   └── nla.png
│   ├── avatar.blend
│   └── avatar.gltf
├── modules ..... knihovny subprojektu
│   ├── __init__.py
│   ├── config.py
│   ├── renderer.py
│   └── sign_tracker.py
├── README.md ..... README soubor subprojektu
└── standalone.py
```