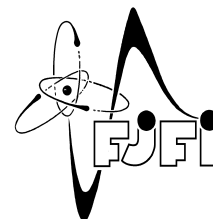




ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta jaderná a fyzikálně inženýrská



Odhad pozice rukou během řízení automobilu

Hand pose estimation during car driving

Diplomová práce

Autor: **Bc. Kajetán Poliak**
Vedoucí práce: **Ing. Adam Novozámský, Ph.D.**
Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Kajetán Poliak
Studijní program: Aplikace přírodních věd
Studijní obor: Aplikované matematicko-stochastické metody
Název práce (česky): Odhad pozice rukou během řízení automobilu
Název práce (anglicky): Hand pose estimation during car driving

Pokyny pro vypracování:

- 1) Proveďte rešerši stávajících metod strojového učení zaměřené na odhad pozice rukou.
- 2) Seznamte se a využijte dostupné datové sady pro řešení této problematiky.
- 3) Implementujte některá stávající řešení a ověřte jejich funkčnost na vybraných datech.
- 4) Adaptujte vybraný algoritmus na data pořízená při řízení automobilu.
- 5) Vyhodnoťte úspěšnost implementovaného řešení.

Doporučená literatura:

- 1) R. O. Duda, P. E. Hart, D.G. Stork, Pattern classification. John Wiley & Sons, 2012.
- 2) R. C. Gonzalez, R. E. Woods, Digital Image Processing (4th ed.), Prentice Hall, 2017.
- 3) I. Goodfellow, Y. Bengio, A. Courville, Deep learning. MIT press, 2016.
- 4) C. Zimmermann, T. Brox, Learning to Estimate 3D Hand Pose from Single RGB Images. In 'Proceedings of the IEEE international conference on computer vision', IEEE, 2017, 4903-4911.
- 5) P. Panteleris, I. Oikonomidis, A. Argyros, Using a Single RGB Frame for Real Time 3D Hand Pose Estimation in the Wild. WACV, 2018.
- 6) M. Martin, S. Stuehmer, M. Voit, R. Stiefelhagen, Real time driver body pose estimation for novel assistance systems. IEEE ITSC, 2017.

Jméno a pracoviště vedoucího diplomové práce:

Ing. Adam Novozámský, Ph.D.

ÚTIA AV ČR, , Pod Vodárenskou věží 4, , CZ-182 08, Praha 8, , Czech Republic

Jméno a pracoviště konzultanta:

Datum zadání diplomové práce: 31.10.2020

Datum odevzdání diplomové práce: 3.5.2021

Doba platnosti zadání je dva roky od data zadání.

Poděkování:

Chtěl bych zde poděkovat především svému školiteli Ing. Adamovi Novozámskému, Ph.D. za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé diplomové práce. Dále děkuji svým rodičům za podporu v průběhu mého studia.

Čestné prohlášení:

Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 3. května 2021

Kajetán Poliak

Název práce:

Odhad pozice rukou během řízení automobilu

Autor: Bc. Kajetán Poliak

Obor: Aplikované matematicko-stochastické metody

Druh práce: Diplomová práce

Vedoucí práce: Ing. Adam Novozámský, Ph.D., ÚTIA AV ČR

Abstrakt: Odhad pozice ruky hraje klíčovou roli v interakcích člověka s počítačem, navíc do jisté míry umožňuje analyzovat chování člověka. Řešení tohoto problému je netriviální kvůli komplikovaným variacím ruky způsobeným komplexní artikulací, překrýváním částí ruky i tvarové, velikostní a barevné nejednoznačnosti. V této práci se k této problematice stavíme komplexně návrhem několika různých metod jak pro detekci, tak pro odhad pozice ruky a ověřením jejich přesnosti. Porovnáváme metody odhadu z hloubkové mapy a z RGB obrazu a následně je aplikujeme na reálná data z trenažéru řízení. Navíc navrhujeme vlastní nástroj k anotaci RGB-D dat pomocí něhož jsme vytvořili testovací datové sady.

Klíčová slova: počítačové vidění, odhad pozice rukou, odhad 2D pozice, odhad 3D pozice, detekce rukou, neuronové sítě

Title:

Hand pose estimation during car driving

Author: Bc. Kajetán Poliak

Abstract: Hand pose estimation plays a fundamental role in human computer interactions, moreover it allows us to analyze human behavior. The problem is nontrivial due to complicated hand variations caused by complex articulations, self-occlusions or shape, size and color ambiguities. We provide complex proposal of different detection and pose estimation methods and their evaluation. The comparison of deep map and RGB based pose estimation is provided and applied to real-life data from the driving simulator. Furthermore we design an annotation tool for RGB-D data which we used to produce a test dataset.

Key words: computer vision, hand pose estimation, 2D pose estimation, 3D pose estimation, hand detection, neural networks

Obsah

Úvod	7
1 Související práce	9
1.1 Detekce a segmentace ruky	9
1.2 2D odhady pozice ruky	10
1.3 3D odhady pozice ruky	11
2 Teorie neuronových sítí	13
2.1 Historický vývoj	13
2.2 Architektura neuronových sítí	15
2.3 Trénování neuronových sítí	15
2.4 Přeučení a regularizace	20
2.5 Inicializace parametrů	22
2.6 Problém mizejícího gradientu	24
2.7 Konvoluční neuronové sítě	25
3 Detekce a segmentace ruky v obrazu	27
3.1 Segmentace	27
3.1.1 Metoda normalizovaných řezů grafu	27
3.1.2 Metoda aktivních kontur	29
3.2 Detekce	31
3.2.1 Detekce objektu podle barvy	32
3.2.2 Detekce objektu pomocí neuronových sítí	34
4 Odhad 2D pozice ruky	43
4.1 Odhad pozice ruky ve 2D	46
5 Odhad prostorové pozice ruky	48
5.1 Odhad pozice ruky z RGB-D obrazu	48
5.2 Odhad pozice ruky z RGB obrazu	50
6 Aplikace na data ze simulátoru řízení kamionu	54
Závěr	59

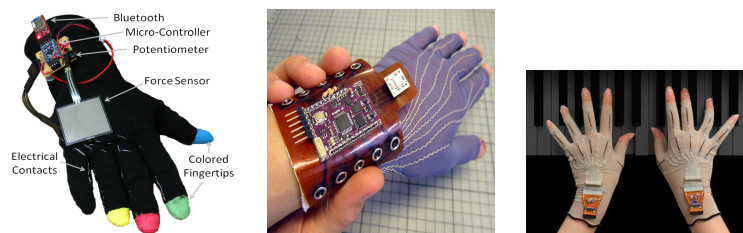
Úvod

Ruka je základním operačním nástrojem člověka a hlavním nástrojem umožňujícím interagovat s okolním prostředím. Funkce rukou není jen manipulační (úchop a upuštění předmětu, jemná motorika), ale také komunikační (gesta, podání ruky), smyslová a opěrná [1]. Pohyb ruky je naprosto přirozená činnost, při které nemusíme přemýšlet o mechanice pohybu. Rozhodneme-li se zvednout předmět, nemusíme přemýšlet nad výběrem prstů, které použijeme, a které klouby ohneme. Většinou v daný moment již přemýšlíme nad tím co s daným předmětem následovně uděláme. Naproti tomu většina osobní elektroniky vyžadující vstup uživatele, jako počítače a chytré telefony, nenabízí volnost ani využití této podvědomé aktivity ruky. Uživatel pomocí zařízení, jako je klávesnice nebo myš, ovládá dvou-dimenzionální reprezentaci virtuálního světa. Tato zařízení jsou pro 2D prostředí naprosto dostačující, protože poskytují přesnou a zodpovědnou projekci pohybu v rovině.

Poměrně nový trend 3D virtuální reality, dále VR, přináší mnoho výzev z čehož velkou část tvoří interakce s virtuálním světem. Z výše uvedeného jasně vyplývá, že výborným nástrojem by stejně jako ve skutečném světě, bylo nechat uživatele ovládat prostředí za využití vlastních rukou. Nejen proto se problém odhadu pozice rukou stává velmi aktivním odvětvím v počítačovém vidění. Určení pozice, orientace nebo artikulace rukou v prostoru je primární úlohou v mnoha potenciálních aplikacích např. manipulace s objektem v robotice, čtení znakového písma [2, 3], již zmíněná virtuální a rozšířená realita, nebo při používání rukou jako vstupního zdroje pro interakci mezi člověkem a počítačem.

V současné době jsou za nejvíce efektivní nástroj k sledování pohybu a gest rukou brána elektromechanická nebo magnetická zařízení (datové rukavice) k vidění na obrázku 1. Tato zařízení jsou schopna doručit nejkompaktnější, aplikačně-nezávislá měření v reálném čase. Na druhou stranu jde o velmi drahé nástroje, které navíc neumožňují absolutně přirozený pohyb ruky a vyžadují komplexní kalibraci a nastavení k dosažení přesných měření. Měli jsme možnost vidět i komerční řešení z dílen firem Oculus, Playstation nebo HTC. Jejich provedení si může čtenář prohlédnout na obrázku 2. Jejich kontrolery ovšem zdaleka nevyužívají možnosti lidské ruky, kvůli malému počtu stupňů volnosti, nedetekují jemnou motoriku ani pozice prstů.

Práce je rozdělena do šesti kapitol. První kapitola obsahuje rešerši stávajících metod strojového učení zaměřených na segmentaci, detekci a odhad pozice rukou. Následuje kapitola věnující se teorii neuronu-



Obrázek 1: Ukázka datových rukavic sloužících k zachycení pohybu a pozice ruky bez využití počítačového vidění.



Obrázek 2: Příklady kontrolérů od firem Oculus, Playstation a HTC, snímajících pohyby ruky pro VR.

vých sítí, které jsou hojně využívány skrz práci. Zaměřujeme se především na teoretický základ zpětného šíření sítí a problému mizejícího gradientu. Třetí kapitola představuje metody segmentace obrazu a detekce objektu v obrazu, uvádíme nejen metody hlubokého učení. Rozebíráme a porovnáváme existující algoritmy segmentace, navrhujeme nový algoritmus detekce objektu a upravujeme metody hlubokého učení pro aplikaci detekce ruky v obrazu. Odhad 2D pozice ruky je rozebírán ve čtvrté kapitole. Zde jsme navrhli dvě architektury pro odhad klíčových bodů v rovině a představili formalismus odhadu klíčových bodů ruky. Zároveň zde představujeme existující datasety k odhadu pozice rukou a popisujeme sestavení vlastních datových sad. Pátá kapitola navazuje odhadem prostorové pozice ruky z RGB-D resp. RGB obrazu. Poslední kapitola rozebírá aplikaci nastudovaných metod na data z тренаžeru řízení. Práce obsahuje dodatkovou část, kde je vedle několika teoretických poznámek, seznam zkratk a značení použitých v práci.

Kapitola 1

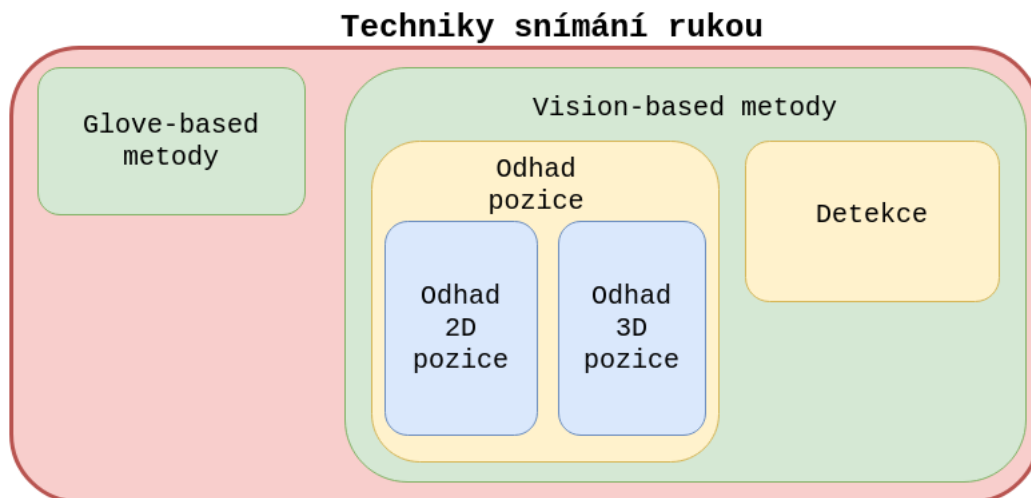
Související práce

Do dnešního dne vzniklo mnoho různých přístupů k detekci rukou, prstů a gest. Obecně jsou tyto přístupy děleny do dvou kategorií „glove-based“ a „vision-based“ přístup. Při využití glove-based analýzy dosahujeme detekce ruky pomocí senzorů, které jsou pevně definované na pozorované ruce. Výsledný 3D model je jednoduše přenositelný do virtuálního světa [5]. Tento přístup není ideální, jak bylo rozebráno výše. V této práci se tedy věnujeme „vision-based“ přístupu, kde se detekuje ruka nebo dokonce celý tvar ruky s jednotlivými prsty za využití obrazového záznamu z jednoho či více obrazových senzorů (fotoaparát, kamera, Kinect,...). Rozdělení oborů zabývajících se snímáním rukou ilustruje diagram na obrázku 1.1.

Problém odhadu pozice ruky je velmi podobný problému odhadu pozice těla. Oba přístupy jsou převoditelné na problém detekce „klíčových bodů“. Odhad pozice těla je velmi aktivní téma v počítačovém vidění a existuje mnoho vědeckých studií navrhuje možné přístupy k řešení tohoto problému, například pomocí náhodných lesů, [7, 6]. Navrhované náhodné lesy dosahovali v určitých podmínkách velmi dobrých výsledků v reálném čase, například pokud byla ruka zachycena z hřbetu a její části se vzájemně nepřekrývali. Velký přínos přišel se začátkem konvolučních neuronových sítí (CNN). Tyto sítě se ukázaly být skvělým nástrojem v problémech učení se znalostí z reálného světa pomocí označených či kategorizovaných obrázků. V následujícím textu shrneme dosavadní přístupy detekce a segmentace ruky v obrazu a odhadu pozice rukou jak z hlediska 2D, tak i 3D detekce „klíčových bodů“ a stručně shrneme datasety, které jsou užitečné při řešení této problematiky.

1.1 Detekce a segmentace ruky

Segmentace je aktivním tématem v oboru počítačového vidění s širokou škálou využití jako je zpracování lékařských obrazových dat [8], analýza obrazů získaných při dálkovém průzkumu Země nebo vyhledávání obrázků pomocí jejich obsahu [9]. Přístupy k řešení problému segmentace se dělí na „model-free“ metody a znalostní (knowledge-based) metody. První ze zmíněných metod bývají často založené na shlukování pixelů, které sdílí konzistentní vizuální vlastnosti vztažené k určitému kritériu podobnosti. Mezi skupinu model-free algoritmů se řadí například mean shift algoritmus [10], který řeší segmentaci pomocí přiřazení určitého módu sdružené hustoty pravděpodobnosti každému pixelu v jeho okolí. Jako další příklady můžeme zmínit variační formulace představené Mumfordem a Shahem [11] nebo metody založené na teorii grafů, konkrétně normalizované řezy grafu [12, 13, 16] a grafové řezy [17, 18, 19]. Kvůli nedostatku předpokladů na geometrický tvar objektu, který chceme segmentovat nejsou tyto metody vhodné k řešení problému, kterým se v této práci zabýváme. Znalostní metody využívají atlas předloh a modelů segmentovaných objektů. K často využívaným modelům patří „Active Appearance Models“, kde je z manuálně segmentovaných trénovacích dat, za využití PCA analýzy, sestaven model



Obrázek 1.1: Diagram rozdělení technik snímání rukou.

objektů. Přítomnost objektu v obraze ověříme pomocí úpravy parametrů modelu. Tento přístup vyžaduje rozsáhlou a reprezentativní množinu trénovacích dat a poměrně přesnou počáteční inicializaci [20, 21]. Dále zde zmíníme metody aktivních kontur, resp. povrchů [22, 23], které se řadí do podkategorie parametrických deformačních modelů. Tyto modely mají mnoho výhod jako je schopnost vypořádat se s otevřenými i uzavřenými parametrickými křivkami a povrchy, malou výpočetní komplexitu a možnost zahrnutí apriorní informace o segmentovaném objektu. Nevýhodou je, již výše zmíněná, potřeba inicializačního modelu.

V následujícím odstavci krátce představíme nejčastěji používané metody detekce rukou. Metody založené na detekci kůže, detekující ruce nezávisle na sobě, mají nevýhodu potřeby velkého množství trénovacích dat [25, 24]. Podobně na tom jsou detektory Viola-Jones jako jsou kaskádové detektory založené na Harrových vlastnostech. Mezi další často navrhované detektory patří metody detekující ruce z lidské obrazové struktury [27, 26]. Nevýhodou těchto metod je, že obrazy jak v trénovacích, tak ve vstupních datech musí obsahovat celou postavu a části ruky se nesmí překrývat. Velmi dobrých výsledků dosahují detektory kombinující více metod k utvoření hypotézy o poloze ruky s následným ověřením dvou-fázovým klasifikátorem [28]. Do problematiky detekce rukou stále více zasahují i neuronové sítě. Dobrých výsledků dosahují Region-based konvoluční neuronové sítě, konkrétně rodina R-CNN, které aplikuje hlubokou konvoluční síť „ConvNet“ ke klasifikaci daných návrhů regionů [29]. Tyto sítě jsou ovšem velmi pomalé, protože zpracovávají každý navržený objekt bez sdílení výpočtů. Tento problém řeší síť Fast R-CNN, která sdílí vlastnosti mezi navrženými objekty [30, 31]. Existuje mnoho návrhů sítí a jejich vylepšení, jako Faster R-CNN nebo YOLO, těmi se v této práci ovšem nezaobíráme.

1.2 2D odhady pozice ruky

V úvodním textu této kapitoly jsme si zmínili o podobnosti designu odhadu klíčových bodů pro polohu ruky a polohu těla. Obecně můžeme problém definovat jako hledání souřadnic pixelu (x_i, y_i) reprezentující klíčový bod i v obrázku. Metody používané k odhadu dělíme na generativní, diskriminativní a hybridní.

Generativní metody (model-based) jsou založené na principu testování hypotéz. První generativní metody k odhadu pozice těla z videa spatřily světlo světa již na začátku 80. let minulého století [32]. Navržený algoritmus využíval top-down přístup a tělo bylo reprezentováno modelem stromu. Generativní

metody většinou obsahují kódovaný parametrický model, který slouží jako nosič apriorní informace rozložení polohy těla a tento model je následně optimalizován technikami grafických inferencí [33].

Další kategorie diskriminativních metod řeší úlohu odhadu přímým mapováním vstupních dat na výstupní parametry. Oproti generativnímu přístupu zde není využíván apriorní model, ale generalizovaný model založený na trénovacích datech. Nejčastěji se zde můžeme setkat s učením s učitelem (supervised learning), které vyžaduje veliké množství dat. Nejčastěji se využívají metody přímé regrese, husté pixelové predikce (dense pixel-wise prediction).

Metody založené na regresi odhadují klíčové body v rovině přímo z obrázků nebo vlastností. Použitím náhodných lesů získáme lokace klíčových bodů z ohodnocení pixelu, které nalezneme v listech [34]. Dále jsou to metody využívající boosted regression [36], nebo rané konvoluční neuronové sítě predikující přímo klíčové body [35]. Problém nastává pokud máme více odhadovaných instancí v obrazu, tato skutečnost není problém v dále probírané husté pixelové predikci.

Další kategorie produkuje pixelové predikce a následně inferuje souřadnice klíčových bodů. Síť predikuje každému pixelu část těla nebo objektu, z kterého pixel pochází. Tyto predikce dostaneme například pomocí náhodných lesů a generováním hypotéz o objektu pomocí samplingu malé množiny pixelů s následným porovnáním s predikcemi z náhodného lesa [37]. V posledních letech je velmi atraktivním směrem heatmap regression. Tento typ regrese byl úspěšně použitý v odhadu polohy lidského těla [38, 39] či výrazu obličeje [40].

1.3 3D odhady pozice ruky

V této sekci představíme state-of-art 3D odhadu pozice ruky. Zaměříme se na pokrok v oblasti regrese odhadu pozice ruky a na přístupy založené na fitování modelu.

Diskriminativní metody, využívané pro řešení úlohy odhadu pozice ruky, jsou založené na učení se mapovací funkce ze vstupního obrázku a konfiguraci parametrů pozice. Odhad rekonstrukce polohy těla se řeší např. pomocí husté pixelové klasifikace, kde mapujeme každý pixel na část těla. Následuje metoda hledající střední hodnotu k nalezení středu této části a tím k rekonstrukci kostry. Také jsme zmínili využití regresního lesa k odhadu částí těla, které jsou na obrázku v zákrytu.

Stejně tak postupovali autoři při odhadu pozice ruky, kde místo přiřazování částí těla pixelu přiřazovali části ruky a následně odvodili pozici klíčového bodu [42, 41]. Oproti tomu metody využívající přímou regresi odhadují klíčové body rovnou [44, 43]. Jak už metoda pixelové klasifikace, tak i metoda přímé regrese utvářejí rozhodnutí na základě informace o lokálním okolí. Nemáme k dispozici kinematickou informaci a proto je s výstupními parametry zacházeno nezávisle na sobě.

Metody založené na modelu (generativní metody) se u většiny přístupu skládají ze tří částí: referenční 3D model, ohodnocující funkce (fitness function) a optimalizačního schématu sloužícího k minimalizaci ohodnocující funkce. Moderní literatura zahrnuje jak problémy odhadu pozice volné ruky, tak i ruky držící určitý objekt. V krátkosti představíme tři zmíněné části.

Trojrozměrný model ruky slouží k dodání informace o tom, jak dobře hypotéza odpovídá originálnímu obrázku. K odhadu pozice ruky nejčastěji používáme kinematický model ruky. Kinematický model je složen z pěti řetězců vcházejících ze zápěstí. Počet uzlů, ze kterých je model tvořen koresponduje s počtem stupňů volnosti, které modelujeme. Existují modely s 16 [47], 18 [48] i 26 [49] uzly.

Správně navržená ohodnocující funkce vrací svoje minimum pro hypotézu, která je nejbliž skutečnosti. Argumentem této funkce může být jak hypotetická pozice ruky, tak originální a vyrederovaná hloubková mapa nebo, navzorkovaný mrak bodů ve 3D reprezentující ruku.

Jako poslední si představíme problematiku optimalizačního schématu, které využíváme k minimalizaci ohodnocující funkce. Populární volbou je tzv. Optimalizace hejnem částic (PSO), kterou zpopularizovala práce [50], poté co v předcházejícím článku [46] autoři narazili na problém minimalizace ve

vysoko dimenzionálním prostoru. Výhodou metody PSO je možnost minimalizace nediferencovatelné funkce. PSO náhodně rozmístí určitý počet částic napříč prohledávaným prostorem. Každá částice má svoji polohu a rychlost. Částice se pohybují po prostoru svojí rychlostí, která se mění na základě různých kritérií. Může to být hodnota ohodnocující funkce v jejich poloze nebo poloze sousedů, atd. Každá částice si zároveň pamatuje minimální hodnotu, které během své cesty dosáhla.

Pokud je ohodnocující funkce diferencovatelná, nabízí se možnost využití gradientních metod. V literatuře se setkáváme například s přístupem, kdy ohodnocovací funkci sestavujeme z Gausiánů, která jsou diferencovatelné a používají metodu gradientního sestupu [51]. Nebo s řešením přes algoritmus Levenberga a Marquardta [52].

Kapitola 2

Teorie neuronových sítí

První kapitola slouží jako matematicko-teoretický základ pro práci s neuronovými sítěmi. Vycházíme především z přednášek doktora M. Straky [63] a on-line knihy M. Nielsena [64].

Nejdříve uvedeme zrod neuronových sítí a jejich stavbu, na které navážeme technikami učení, regularizací a inicializací parametrů sítě. Na konec představíme konvoluční neuronové sítě.

2.1 Historický vývoj

V šedesátých letech minulého století byl vyvinut nejjednodušší model dopředné neuronové sítě zvané *perceptron*. Perceptron má na vstupu vektor binárních hodnot (x_1, x_2, \dots) a vrací jeden binární výstup. Hodnota výstupu je závislá na prahu citlivosti neuronu a hodnotě vážené sumy $\sum_i w_i x_i$, kde w_i je váha udávající vliv vstupu x_i na výstup. Hodnotu výstupu potom můžeme vyjádřit jako

$$\text{výstup} = \begin{cases} 0 & \text{pro } \sum_i w_i x_i \leq \text{práh citlivosti} \\ 1 & \text{pro } \sum_i w_i x_i > \text{práh citlivosti} \end{cases} \quad (2.1)$$

Při volbě různých kombinací vah a prahů citlivost dostáváme modely různých rozhodovacích úloh. Samotný jeden perceptron není dostačujícím nástrojem, ale nabízí se utvořit komplexní síť složenou z perceptronů.

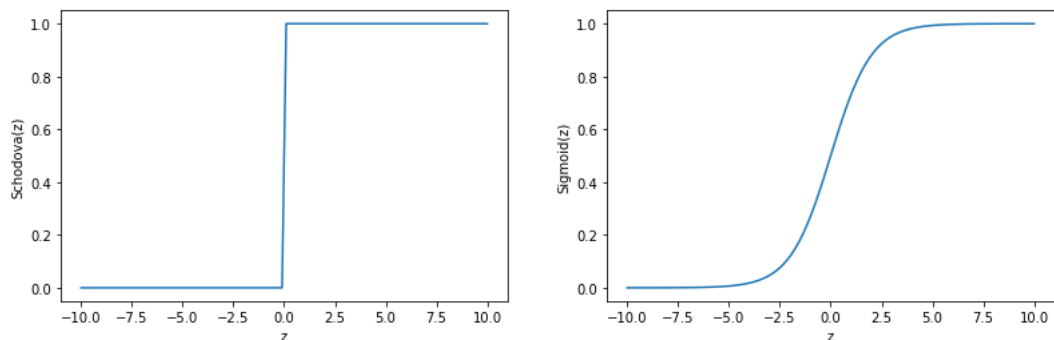
Při zavedení veličiny *bias*, $b = -\text{práh citlivosti}$, a přepsáním vážené sumy do skalárního součinu vektoru vstupů a vah $\mathbf{w} \cdot \mathbf{x} = \sum_i w_i x_i$, můžeme přepsat rovnici (2.1) do tvaru

$$\text{výstup} = \begin{cases} 0 & \text{pro } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1 & \text{pro } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}.$$

Zavedená veličina *bias* je jakousi měrou jak lehké je přimět perceptron, aby vrátil na výstupu 1. Zavedení této veličiny má pozitivní vliv především na zjednodušení notace pro složitější neuronové sítě.

Problémem perceptronů je, že i malá změna váhy může mít velký vliv na výstup, což by v síti perceptronů mohlo způsobit naprostou změnu chování této sítě. Proto se zavedl nový typ neuronu zvaný *sigmoid*. Sigmoid má tu výhodu, že pro malé změny vah a biasu způsobí malé změny výstupu, čímž umožňuje síti se učit. Sigmoid neuron má stejně jako perceptron na vstupu vektor (x_1, x_2, \dots) , tyto hodnoty už ovšem nemusí být binární, ale z intervalu $(0, 1)$. Výstup tohoto neuronu je potom

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b) = \frac{1}{1 + \exp(-\sum_i w_i x_i - b)},$$



Obrázek 2.1: Graf schodové funkce (vlevo) a funkce sigmoid (vpravo).

kde $\sigma(\cdot)$ je funkce sigmoid a w_i váha vstupu x_i . Označme $y = \mathbf{w} \cdot \mathbf{x} + b$, potom platí

$$\begin{aligned} \lim_{y \rightarrow +\infty} \sigma(y) &= 1, \\ \lim_{y \rightarrow -\infty} \sigma(y) &= 0. \end{aligned} \quad (2.2)$$

V limitním případě, jak ukazuje rovnice (2.2), přechází sigmoid to perceptronu. Funkce sigmoid je vyhlazený perceptron resp. schodová funkce (modifikovaná pro argument 0, jelikož klasická schodová funkce vrací v tomto bodě hodnotu 1 a perceptron vrací hodnotu 0), důležitá část této funkce je tedy pro argumenty blízké nule, jak ilustruje obrázek 2.1. Klíčovou výhodou sigmoidu pro naši aplikaci je, že malá změna hodnoty váhy Δw_i a biasu Δb způsobí malou změnu Δ výstup výstupu, což ilustruje aproximace v rovnici (2.3). Z této rovnice vidíme, že změna na výstupu závisí lineárně na změně vah a biasu.

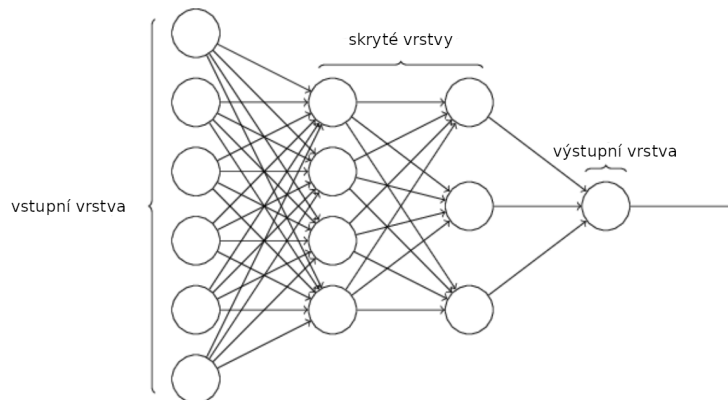
$$\Delta \text{výstup} \approx \sum_i \frac{\partial \text{výstup}}{\partial w_i} \Delta w_i + \frac{\partial \text{výstup}}{\partial b} \Delta b. \quad (2.3)$$

Funkce sigmoid řeší problém vyhlazeného učení, ale není symetrická podle nuly. To způsobuje, že v architekturách s více skrytými vrstvami konvergují vstupy s pravděpodobností větší než 1/2 k jedničce a málo pravděpodobné vstupy k nule. Proto se zavedla funkce $\tanh(x) = 2\sigma(2x) - 1$, která je symetrická a má derivaci v nule rovnou jedné (tato vlastnost je klíčová pro zpětnou propagaci při učení sítě, která je popsána v následující sekci). Funkce sigmoid, hyperbolický tangens nebo jiné funkce, které používáme na výstupu nebo ve skrytých vrstvách neuronové sítě nazýváme *aktivační funkce*. Přes to, že modely využívající hyperbolický tangens jako aktivační funkci dosahují lepších prediktivních výsledků a jednoduššího tréninku než sigmoid, tak narážejí na problém saturace. To znamená, že vysoké hodnoty konvergují k jedničce a nízké naopak k nule resp. mínus jedničce pro sigmoid resp. hyperbolický tangens. Tyto funkce jsou citlivé jen na hodnoty blízko jejich „středu“ (0, 5 pro sigmoid a 0 pro tanh).

Z výše uvedených důvodů se postupem času přešlo k ReLU funkci (rectified linear unit) definovanou předpisem

$$\text{ReLU}(x) = \max\{0, x\}.$$

Výhodou této aktivační funkce je nejen jednoduchost výpočtu oproti sigmoidu a tanh, pro které vyčíslujeme exponenciální funkce, ale i lineární chování funkce. Díky tomu, že funkce je téměř lineární je daleko jednodušší optimalizace sítě a klíčovou výhodou je, že se téměř zbavíme problému mizejícího gradientu, který rozebíráme na straně 24.



Obrázek 2.2: Schéma stavby jednoduché neuronové sítě.

2.2 Architektura neuronových sítí

V této sekci popíšeme stavbu neuronových sítí a zadefinujeme základní pojmy, na kterých vystavíme teoretický základ pro konvoluční neuronové sítě.

Základ neuronových sítí tvoří tři komponenty jak ilustruje obrázek 15. Vrstva, která je vyobrazena vlevo se nazývá *vstupní vrstva*. Tato vrstva se skládá z neuronů zvaných *vstupní neurony*. Tyto neurony komunikují s neurony v takzvané *skryté vrstvě*. Skrytých vrstev je ve většině neuronových sítí více. Vrstva, která je vpravo se nazývá *výstupní vrstva* a je složena z *výstupních neuronů*. Na schématu 15 je tato vrstva složena pouze z jednoho neuronu, může jich být ovšem více, záleží na aplikaci, pro kterou tvoříme síť.

Počet skrytých vrstev v síti ovlivňuje komplexitu modelu. Obecně platí, že neuronové sítě s 5-10 vrstvami mají lepší prediktivní vlastnosti, jsou schopny pojmout, větší hierarchickou komplexitu, než sítě s jednou skrytou vrstvou. Neuronové sítě s více skrytými vrstvami se nazývají *hluboké neuronové sítě*. Doposud jsme se zabývali sítěmi, kde výstup jedné vrstvy používáme jako vstup následující vrstvy. Tento typ sítě nazveme *dopředné*. Existují i sítě, které mají na vstupu vrstvy výstup z následné vrstvy. Takové neuronové sítě nazýváme *rekurentní*.

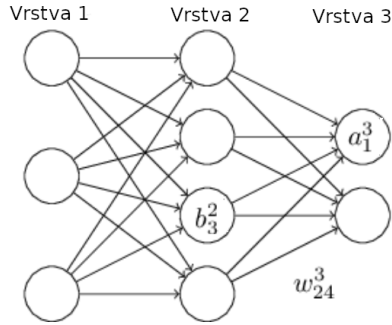
2.3 Trénování neuronových sítí

Fitování neuronové sítě zahrnuje použití tréninkového datasetu k aktualizaci vah modelu, abychom vytvořili dobré mapování vstupů na výstupy. Tento proces nazýváme *trénování* a je řešen pomocí optimalizačních algoritmů, které prohledávají prostor možných hodnot vah neuronové sítě k nalezení souboru vah, které vracejí dobré výsledky na trénovacím datasetu.

Nejprve si zavedeme notaci, která nám poslouží k jednoznačné referenci vah v síti. Váhu na spojení k -tého neuronu $(l - 1)$ vrstvy a j -tého neuronu l -té vrstvy označíme w_{jk}^l . Podobné označení použijeme na bias a aktivace sítě. Konkrétně notace b_j^l , označuje bias j -tého neuronu v l -té vrstvě a a_j^l označuje aktivaci (vyčíslenou hodnotu z aktivační funkce) j -tého neuronu v l -té vrstvě. K lepší orientaci v právě zavedených pojmech odkážeme na obrázek 2.3.

Aktivace j -tého neuronu l -té vrstvy a_j^l je závislá na vrstvě $(l - 1)$. Tuto závislost můžeme vyjádřit rovnicí

$$a_j^l = f \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right), \quad (2.4)$$



Obrázek 2.3: Schéma neuronové sítě se třemi vrstvami s veličinami a_1^3 aktivace prvního neuronu třetí vrstvy, b_3^2 bias třetího neuronu druhé vrstvy a w_{24}^3 váhu na spojení čtvrtého neuronu třetí vrstvy a druhého neuronu třetí vrstvy.

kde f je aktivační funkce l -té vrstvy a suma je přes všechny neurony v $l-1$ vrstvě. Pro větší přehlednost přejdeme k maticovému zápisu. Definujme matici vah l -té vrstvy jako w^l tzn. prvek v k -tém sloupci a j -té řadě je w_{jk}^l . Obdobně definujeme vektor biasu na l -té vrstvě jako b^l a vektor aktivací jako a^l . S nově zavedenou maticovou notací můžeme přepsat rovnici (2.4) do tvaru

$$a^l = f(w^l a^{l-1} + b^l). \quad (2.5)$$

Argument aktivační funkce v rovnici (2.5) označíme jako z^l a nazveme jej vážený vstup neuronu l , takže můžeme psát $a^l = f(z^l)$.

Nechť je x vektor vstupních hodnot a $y = y(x)$ kýžený výstup. Chceme najít algoritmus, který nachází hodnoty vah a biasu tak, že výstup sítě aproximuje $y(x)$ pro všechny tréninkové vstupy x co nejlépe. Pro kvantifikaci výchyly od chtěného výstupu definujeme *ztrátovou funkci*:

$$C(w, b) = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2, \quad (2.6)$$

kde w jsou váhy sítě, b biasy, n počet vstupních tréninkových dat, L označuje počet vrstev sítě a $a^L = a^L(x)$ vektor výstupů ze sítě pro vstup x . Tato ztrátová funkce se nazývá také *kvadratická ztrátová funkce* nebo *MSE* (z anglického *mean squared error*). Ztrátová funkce je nezáporná a nabývá malých hodnot pokud jsme blízko chtěného výsledku.

Naším úkolem je minimalizovat ztrátovou funkci, pro zavedení *gradientního sestupu* přejdeme k notaci $C(v)$, kde $v = (v_1, v_2)$. Budeme tedy měnit hodnoty v_1 a v_2 o malé hodnoty ($\Delta v_1, \Delta v_2$) a pozorovat změnu hodnoty ztrátové funkce:

$$\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 + \frac{\partial C}{\partial v_2} \Delta v_2. \quad (2.7)$$

Dále zavedeme gradient ztrátové funkce jako

$$\nabla C = \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2} \right)^T.$$

Potom můžeme přepsat rovnici (2.7) jako

$$\Delta C \approx \nabla C \cdot \Delta v. \quad (2.8)$$

Tento zápis nám jasně říká jak změnit Δv , abychom zmenšili ΔC . Řekněme, že vybereme

$$\Delta v = -\alpha \nabla C, \quad (2.9)$$

parametr α je kladný parametr nízké hodnoty, který nazýváme *rychlost učení*. Dosazením hodnoty (2.9) do (2.8) dostáváme $\Delta C \approx -\alpha \|\nabla C\|^2$. Z faktu $\|\nabla C\|^2 > 0$, vidíme, že ztrátová funkce vždy klesne. Opakováním aktualizace

$$v \rightarrow v' = v - \alpha \nabla C,$$

postupně konvergujeme k minimu funkce. Právě popsany algoritmus se nazývá *gradientní sestup*.

Při učení neuronových sítí, hledáme váhy w_k a biasy b_l minimalizující ztrátovou funkci definovanou rovnicí (2.6). Provádíme aktualizaci

$$\begin{aligned} w_k &\rightarrow w'_k = w_k - \alpha \frac{\partial C}{\partial w_k}, \\ b_l &\rightarrow b'_l = b_l - \alpha \frac{\partial C}{\partial b_l}. \end{aligned} \quad (2.10)$$

K rychlejšímu učení se používá algoritmus *stochastický gradientní sestup*. Ten funguje na základě náhodného výběru malého počtu m trénovacích vstupů. Označme tyto vstupy X_1, \dots, X_m a nazvěme je *mini-batch*. Označme

$$C_x = \frac{\|y(x) - a\|^2}{2}.$$

Potom pro dostatečně velký mini-batch můžeme předpokládat

$$\frac{\sum_{i=1}^m \nabla C_{X_i}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C,$$

kde v druhé sumě sčítáme přes všechny vstupy. Záměnou stran dostáváme rovnici

$$\nabla C \approx \frac{1}{m} \sum_{i=1}^m \nabla C_{X_i}.$$

Z této rovnice vidíme, že můžeme odhadovat celkový gradient z náhodně vybraného mini-batche.

Stochastický gradientní sestup vyžaduje výpočet gradientů pro každou proměnnou modelu. Proto se využívá algoritmu *zpětného šíření chyby* (back-propagation). Jedná se o algoritmus počítající gradienty ztrátové funkce vzhledem k proměnným modelu. V naší notaci počítá algoritmus zpětného šíření chyby parciální derivace $\partial C / \partial w$ a $\partial C / \partial b$. Tento algoritmus má dva hlavní předpoklady na tvar ztrátové funkce.

Prvním předpokladem je možnost vyjádřit ztrátovou funkci jako průměr

$$C = \frac{1}{n} \sum_x C_x$$

přes všechny ztrátové funkce C_x pro jednotlivé tréninkové vstupy x . Tento předpoklad je splněný pro kvadratickou ztrátovou funkci $C = \frac{1}{2} \|y - a^L\|^2$, kterou využíváme v této práci.

Druhý předpoklad je, že ztráta je vyjádřitelná jako funkce výstupů neuronové sítě. Pro kvadratickou ztrátovou funkci, můžeme pro jeden vstup x psát

$$C = \frac{1}{2} \|y - a^L\|^2 = \frac{1}{2} \sum_i (y_i - a_i^L)^2$$

z čehož vidíme, že funkce je vyjádřitelná jako závislá na výstupech neuronové sítě.

Algoritmus zpětného šíření chyby je založen na porozumění jak změna hodnot vah a biasů v síti mění hodnotu ztrátové funkce. Potřebujeme tedy spočítat parciální derivace ztrátové funkce podle vah a biasů. K jejich výpočtu je využito kvantit zvané *chyba*, která je definovaná jako:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l},$$

konkrétně jsme definovaly chybu j -tého neuronu l -té vrstvy a z_j^l je vážený vstup definovaný pod rovnicí (2.5). Stejně jako u jiných proměnných označíme δ^l vektor chyb asociovaný s l -tou vrstvou.

Algoritmus zpětného šíření chyby je založen na čtyřech základních rovnicích, pomocí kterých vypočítáme chybu δ^l a gradient ztrátové funkce.

Jako první představíme rovnici pro výpočet chyby výstupní vrstvy δ^L :

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} f'(z_j^L). \quad (2.11)$$

První člen $\partial C / \partial a_j^L$ udává jak rychle je mění ztráta jako funkce j -tého výstupu aktivace. V situaci, kdy ztrátová funkce nemá silnou závislost na určitém výstupu neuronu j , bude hodnota δ_j^L nízká. Druhý člen $f'(z_j^L)$ měří jak rychle aktivační se funkce f mění v z_j^L . Pro vyjádření rovnice (2.11) v maticovém zápisu si zdefinujeme *Hadamardův součin*.

Definice. Pro dvě matice \mathbb{A} a \mathbb{B} o stejné dimenzi definujeme Hadamardův součin $\mathbb{A} \odot \mathbb{B}$ jako

$$\mathbb{A} \odot \mathbb{B} = (\mathbb{A})_{ij} (\mathbb{B})_{ij}.$$

Pro matice různých dimenzí není Hadamardův součin definovaný.

Nyní můžeme rovnici (2.11) přepsat do tvaru

$$\delta^L = \nabla_a C \odot f'(z^L),$$

kde $\nabla_a C$ je gradient ztrátové funkce vzhledem k aktivacím a^L . Pro kvadratickou ztrátovou funkci můžeme tedy psát

$$\delta^L = (a^L - y) \odot f'(z^L).$$

Druhou rovnicí je rovnice pro výpočet chyby δ^l pomocí chyby v následující vrstvě δ^{l+1} :

$$\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot f'(z^l). \quad (2.12)$$

Kombinací rovnic (2.11) a (2.12) jsme schopni spočítat chybu δ^l pro jakoukoli vrstvu v síti. Prvně za pomoci (2.11) spočteme δ^L a aplikujeme rovnici (2.12) pomocí, které vypočítáme δ^{L-1} opakovaným aplikováním druhé rovnice dopočítáme všechny chyby.

Třetí rovnice už slouží přímo k výpočtu parciální derivace ztrátové funkce vůči biasu

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l.$$

Z této rovnice vidíme, že chyba δ_j^l je rovna parciální derivaci $\partial C / \partial b_j^l$. Přejdeme k zápisu

$$\frac{\partial C}{\partial b} = \delta,$$

kde chybu δ evalujeme ve stejném neuronu jako bias b .

Poslední, čtvrtou, rovnicí je rovnice pro výpočet parciální derivace ztrátové funkce vůči váze:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l.$$

Algoritmus zpětného šíření vlny můžeme nyní sepsat pomocí pseudokódu za využití právě zmíněných rovnic, konkrétně:

1. **Vstup x :** Nastav aktivace vstupní vrstvy a^1 .
2. **Dopředné šíření:** Pro každé $l = 2, 3, \dots, L$ vypočítej $z^l = w^l a^{l-1} + b^l$ a $a^l = f(z^l)$.
3. **Výstupní chyba δ^L :** Vypočti vektor $\delta^L = \nabla_a C \odot f'(z^L)$.
4. **Zpětné šíření chyby:** Pro každé $l = L - 1, L - 2, \dots, 2$ vypočítej $\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot f'(z^l)$.
5. **Výstup:** Gradient ztrátové funkce jako $\partial C / \partial w_{jk}^l = a_k^{l-1} \delta_j^l$ a $\partial C / \partial b_j^l = \delta_j^l$.

V praxi kombinujeme algoritmus zpětné propagace s učícím algoritmem jako je stochastický gradientní sestup, ve kterém počítáme gradienty pro velké množství trénovacích dat. Konkrétně by algoritmus, pro daný mini-batch m trénovacích dat vypadal následovně:

1. **Vstup:** Mini-batch trénovacích dat velikosti m .
2. Pro každý trénovací příklad x : Nastav aktivace vstupní vrstvy $a^{x,1}$ a vykonej tyto operace (stejně jako v algoritmu pro zpětné šíření výše):
 - (a) **Dopředné šíření:** Vraccjící $z^{x,l}$ a $a^{x,l}$ pro $l = 2, 3, \dots, L$.
 - (b) **Výstupní chyba $\delta^{x,L}$**
 - (c) **Zpětné šíření chyby:** Vraccjící $\delta^{x,l}$ pro $l = L - 1, L - 2, \dots, 2$.
3. **Gradientní sestup:** Pro každé $l = L - 1, L - 2, \dots, 2$ aktualizuj váhy a biasy podle pravidla

$$w^l \rightarrow w^l - \frac{\alpha}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T,$$


$$b^l \rightarrow b^l - \frac{\alpha}{m} \sum_x \delta^{x,l}.$$

Součástí tohoto algoritmu musí být zároveň vnější smyčka generující mini-batche, kterou zde pro jednoduchost neuvádíme.

Pro ilustraci rychlosti výpočtu gradientu ztrátové funkce pomocí zpětného šíření chyby uvažujme závislost ztrátové funkce jen na vahách tj. $C = C(w)$. Parciální derivace můžeme počítat např. pomocí aproximace

$$\frac{\partial C}{\partial w_j} \approx \frac{C(w + \epsilon e_j) - C(w)}{\epsilon},$$

kde ϵ je malé kladné číslo a e_j vektor s jedničkou na j -té pozici. Podobně bychom vyjádřili parciální derivace vzhledem k biasům. Pokud máme model s tisícem vah, musíme napočítat $C(w + \epsilon e_j)$ pro každou váhu a jednou vypočítat $C(w)$. Máme tedy tisíc a jeden výpočtů (projití sítí). Zpětné šíření chyby vypočítá všechny parciální derivace $\partial C / \partial w_j$ během jediného dopředného šíření sítí následovaným zpětným šířením. Místo tisíc a jednoho projití sítě máme tedy jen dvě projití sítě.

$$\text{IoU} = \frac{\text{Obsah průniku}}{\text{Obsah sjednocení}}$$


Obrázek 2.4: Ilustrace definice IoU. IoU je definován jako podíl obsahu průniku pravdivého ohraničení s predikovanými ohraničeními a obsahu sjednocení těchto ohraničení.

2.4 Přeučení a regularizace

Přeučení (overfitting) je stav modelu, kdy model vystihuje přesně určitý soubor dat a špatně vystihuje nová data nebo není schopen dobře predikovat další pozorování. Jinak řečeno, je to stav modelu, kdy model přesně vystihuje trénovací data, ale selhává na testovacích resp. validačních datech. Přeučení je statistický model, který má více parametrů než data obhajitelně potřebují. V modelech neuronových sítí, kde máme miliony parametrů je problému přeučení věnováno mnoho pozornosti. K efektivnímu tréninku potřebujeme nástroje pro detekci přeučení a k redukci jeho efektu.

Očividnou možností je kontrolovat *přesnost* (accuracy) definovanou jako

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

kde TP a TN je počet správně klasifikovaných (true positive resp. true negative), FP a FN je počet špatně klasifikovaných (false positive resp. false negative) tj. přesnost určuje poměr správně zařazených dat vůči celkovému počtu dat. Trénování potom zastavíme, když vidíme, že se přesnost nezvyšuje. Dobrým nástrojem k redukci přeučení je navýšení počtu dat v trénovacím datasetu nebo redukce velikosti sítě. Tyto možnosti nejsou vždy dostupné, zvětšování datasetu je v mnoha aplikacích náročné a drahé a velké sítě mají potenciál být daleko silnější.

Častěji se používají metriky *precision* a *recall*. Jejich formální definice je

$$\text{Precision} = \frac{TP}{TP + FP},$$

$$\text{Recall} = \frac{TP}{TP + FN}.$$

Definujme metriku IoU (z angl. „Intersection over union“), měřící překryv dvou ohraničení ve snímku. Při klasifikaci nebo detekci definujeme IoU jako hranici, pro rozhodnutí, zda-li je predikce TP nebo FP. Situaci ilustruje obrázek 2.4.

Metrika AP („average precision“) je definovaná jako oblast pod precision-recall křivkou, tj.

$$AP = \int_0^1 p(r) dr.$$

Precision a recall jsou omezeny hodnotami 0 a 1, proto $AP \in \langle 0, 1 \rangle$.

Konečně metrika, se kterou se v teorii neuronových sítí setkáváme nejčastěji je mAP („mean average precision“). Metrika mAP je průměr AP.

Existují techniky redukcující přeučení při stejné velikosti trénovacího datasetu a sítě. Tyto techniky nazýváme *regularizace*. Nejčastěji využívanou regularizací je *L2 regularizace*, která přidává ke ztrátové funkci *regularizační člen*. Regularizovaná kvadratická ztrátová funkce, označené C_0 , vypadá následovně

$$C = C_0 + \frac{\lambda}{2n} \sum_w w^2 = -\frac{1}{2n} \sum_x \|y - a^L\|^2 + \frac{\lambda}{2n} \sum_w w^2. \quad (2.13)$$

Z tvaru regularizačního členu vidíme, že model preferuje učení se malých vah a velké váhy se vyskytnout jen v případě, že mají velký vliv na přesnost model a tím snižuje hodnotu prvního členu. Parametr λ volíme podle preference. Při malých hodnotách preferujeme minimalizaci vlastní ztrátové funkce, při velkých hodnotách preferujeme malé váhy.

Parciální derivace rovnice (2.13) podle váhy a biasu je

$$\begin{aligned} \frac{\partial C}{\partial w} &= \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} w, \\ \frac{\partial C}{\partial b} &= \frac{\partial C_0}{\partial b}. \end{aligned}$$

Parciální derivace dostaneme z algoritmu zpětného šíření chyby a nakonec přičteme $\frac{\lambda}{n} w$ k parciální derivaci podle vah. Parciální derivace podle biasu zůstává stejný, aktualizace v gradientním sestupu je pro bias tedy stejná jako v (2.10). Váhy aktualizujeme podle pravidla

$$\begin{aligned} w &\rightarrow w - \alpha \frac{\partial C_0}{\partial w} - \frac{\alpha \lambda}{n} w = \\ &= \left(1 - \frac{\alpha \lambda}{n}\right) w - \alpha \frac{\partial C_0}{\partial w}. \end{aligned}$$

Zde je změna od (2.10) jen ve škálování váhy faktorem $\frac{\alpha \lambda}{n}$. U stochastického gradientního sestupu se aktualizace váhy provádí podle:

$$w \rightarrow \left(1 - \frac{\alpha \lambda}{n}\right) w - \frac{\alpha}{m} \sum_x \frac{\partial C_x}{\partial w}, \quad (2.14)$$

kde sčítáme přes všechny tréninkové příklady v mini-batchi a C_x je neregularizovaná ztrátová funkce každého příkladu.

Mějme síť s váhami malých hodnot. Nízké váhy zapříčiňují malou změnu chování sítě při změně určitého počtu náhodně vybraných vstupů. Kvůli tomu se regularizovaná síť špatně učí efekty lokálního šumu na datech a dobře se učí vystihnout příznaky společné pro dataset.

Z dalších regularizací zde zmíníme ještě další tři. První z nich je *L1 regularizace*. Při L1 regularizaci přechází ztrátová funkce do tvaru

$$C = C_0 + \frac{\lambda}{n} \sum_w |w|, \quad (2.15)$$

který je velmi podobný L2 regularizaci (2.13). Parciální derivace rovnice (2.15) má tvar

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} \text{sign}(w),$$

kde $\text{sgn}(w)$ je funkce signum, vracející +1 pro $w > 0$, -1 pro $w < 0$ a nulu pro $w = 0$. Aktualizace vah při stochastického gradientu probíhá podle předpisu

$$w \rightarrow w - \frac{\alpha \lambda}{n} \text{sign}(w) - \frac{\alpha}{m} \sum_x \frac{\partial C_x}{\partial w}. \quad (2.16)$$

Při srovnání aktualizací při L2 regularizaci (2.14) a L1 regularizaci (2.16) vidíme, že L1 regularizace posouvá váhy k nule o konstantní krok a L2 regularizace zmenšuje váhy o hodnotu proporčně velkou k w . Při vysoké hodnotě $|w|$ smrkává L1 váhy méně než L2, naopak pro malé $|w|$ smrkává L1 váhy rychleji než L2.

Dalším typem regulace je *dropout*, který neupravuje ztrátovou funkci, ale samotnou stavbu sítě. Mějme neuronovou síť s tréninkovým vstupem x a chtěným výstupem y . Za normálních okolností bychom provedli dopředné šíření x skrz síť a následné zpětné šíření k určení přírůstku gradientu. U dropoutu začínáme dočasným odebráním poloviny náhodně vybraných neuronů skrytých vrstev v síti, přičemž necháváme vstup a výstup stejný. Provedeme dopředné a zpětné šíření sítě s odebranými neurony, nazývanými *dropout neurony* nad mini-batchem dat a aktualizujeme hodnoty vah a biasů. Tento postup opakujeme s novým výběrem dropout neuronů.

Nad dropoutem můžeme přemýšlet jako nad trénováním různých neuronových sítí (stejná síť s různými dropouty) a následným průměrováním jejich výsledků. Trénujeme-li s dvěma různými dropouty, tak výsledné váhy vydělíme dvěma.

Tato technika redukuje komplexitu návaznosti neuronů, jelikož se neuron nemůže spoléhat na přítomnost ostatních neuronů. Tím je neuron nucen učit se robustnější příznaky.

Posledním typem regularizace, který v této práci představíme je *umělé rozšíření datasetu*. Jde o triviální myšlenku, kdy dosavadní dataset upravíme elementárními operacemi a takto upravená data přidáme k původním. Příkladem může být přidání nebo odebrání šumu ze záznamů hlasu při trénování neuronových sítí rozpoznávajících řeč, horizontální a vertikální převrácení pro síť rozpoznávající objekty na snímcích nebo rotace ručně psaných písmen a čísel na obrázcích, které používáme k jejich klasifikaci.

2.5 Inicializace parametrů

Při stavbě neuronové sítě musíme vybrat inicializační hodnoty pro váhy a biasy. Běžnou cestou je volit hodnoty těchto parametrů jako nezávislý výběr z Gaussova rozdělení $\mathcal{N}(0, 1)$.

Pro takovou inicializaci je velmi pravděpodobné, že model bude zatížen saturací. Představme si, že máme model s 500 váhami. Potom vážená suma $z = \sum_i w_i x_i + b$ bude mít také gaussovské rozdělení se střední hodnotou nula a variancí $\sqrt{501} \approx 22,4$. S velkou pravděpodobností budeme dostávat $z \gg 1$ nebo $z \ll -1$ a aktivační funkce např. $\sigma(z)$ bude vracet hodnoty blízké 1 nebo 0. Takové váhy se budou učit velmi pomalu při použití gradientního sestupu.

Dalším přístupem je volba parametrů z $\mathcal{N}\left(0, 1/\sqrt{n_i}\right)$, kde n_i je počet neuronů v i -té vrstvě, tedy gaussovské rozdělení, které je užší než původní (váhy mají varianci rovnou 1, 22...) nebo jak je doporučeno v [65]

$$w \sim U\left(-\sqrt{\frac{6}{n_i + n_{i+1}}}, \sqrt{\frac{6}{n_i + n_{i+1}}}\right),$$

kde $U(\cdot, \cdot)$ je rovnoměrné rozdělení pravděpodobnosti a n_i resp. n_{i+1} je počet neuronů v i -té resp. $(i+1)$. vrstvě.

Dalším problémem je nastavení hyperparametrů modelu jako je rychlost učení α , regularizační parametr λ , velikost mini-batche a další. Uvedeme zde nejčastěji používané metody jako prohledávání mřížky, náhodné prohledávání, ruční ladění a bayesovskou optimalizaci.

Prohledávání mřížky je nejjednodušším algoritmem k optimalizaci hyperparametrů. Je založený na definování množiny hodnot parametrů, trénování modelu pro všechny kombinace parametrů a výběru nejlepšího modelu. Tato metoda není příliš vhodná pro trénování neuronových sítí, kde trénink je časově velmi náročný.

Mějme například 5 parametrů, pro každý parametr definujeme množinu 10 různých hodnot. V tomto případě bychom potřebovali 10^5 evaluací. Pokud by trénink trval průměrně 10 min, dostali bychom výsledek za necelé dva roky.

Další metodou je *náhodné prohledávání*, která je podobná předchozí metodě, ale místo všech kombinací hodnot parametrů evaluujeme jen náhodně vybranou podmnožinu. Zde je důležitý výběr pravděpodobnostního rozdělení, podle kterého vybíráme. Při použití rovnoměrného rozdělení nemusí docházet k plnému pokrytí parametrického prostoru. Tento problém se řeší pomocí kvazináhodných posloupností. I když je tato metoda rychlejší, tak se stále nehodí pro optimalizaci na neuronových sítích.¹

Mezi nejčastěji využívané metody patří stále *ruční ladění* parametrů. Při stavbě neuronové sítě pro nějaký reálný problém se nabízí začínat od triviálního modelu. Vezměme si příklad klasifikace ručně psaných číslic. Tento problém můžeme trivializovat na klasifikaci pouze dvou čísel. Pro tento problém najdeme vhodné hyperparametry, jako je počet neuronů ve vrstvě, počet skrytých vrstev, rychlost učení aj. Následně přidáváme na komplexitě modelu a předpokládáme, že hodnoty parametrů se nebudou příliš měnit.

Jako poslední, nejsilnější a zároveň nejsložitější metodu co se týče implementace zmíníme metodu *bayesovské optimalizace*, konkrétně *Gaussovský proces* s akviziční funkcí. Bayesovská optimalizace je strategie globální optimalizace pro funkce, u kterých neznáme žádné funkční normy (blackbox). Z důvodu, že objektivní funkce není známá, považuje ji Bayesovská strategie za náhodnou funkci a utváří na ní apriorní rozdělení. Podle chování funkce následně buduje aposteriorní rozdělení na objektivní funkci a nakonec je aposteriorní funkce využita ke konstrukci akviziční funkce. Nejčastější metody sloužící k definování apriorní resp. aposteriorní funkce jsou Gaussovské procesy a Parzen-Tree Estimators.

Gaussovský proces používá množinu předem evaluovaných parametrů a přesnost modelu k vytvoření předpokladu o nepozorovaných parametrech. Akviziční funkce použije tuto informaci k návrhu další množiny parametrů.

Mějme vstup x a výstup $y = f(x)$, kde f je náhodná funkce. Tato funkce vzorkuje výstup z gaussovského rozdělení. Zároveň můžeme říct, že každý vstup x je asociován s gaussovským rozdělením a má tedy definovanou střední hodnotu μ a varianci σ^2 . Gaussovský proces je zobecnění vícerozměrného gaussova rozdělení. Oproti vícerozměrnému gaussovskému rozdělení, které je definováno pomocí vektoru středních hodnot a kovarianční maticí, je gaussovský proces definovaný pomocí funkcí střední hodnoty a kovarianční funkcí.

Mezi často využívané akviziční funkce patří *očekávané zlepšení* (Expected Improvement)

$$EI(x) = E[\max\{0, f(x) - y_{max}\}], \quad (2.17)$$

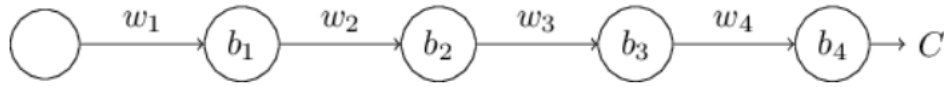
kde y_{max} je právě vrácené maximum.

Cílem je najít optimální množinu parametrů \mathbf{x}_{opt} v parametrickém prostoru Θ , tj.:

$$\mathbf{x}_{opt} = \arg \max_{\mathbf{x} \in \Theta} f(\mathbf{x}).$$

Z rovnice (2.17) vidíme, že očekávané zlepšení je vyšší pokud model očekává, že bod bude mít v průměru vyšší hodnotu.

¹Kvazináhodná posloupnost představují kompromis mezi zcela rovnoměrným a zcela náhodným pokrytím. Nelze je považovat za náhodné, jsou to deterministické posloupnosti. Konstruuji se tak, aby poskytovaly pro k náhodných vzorků co možná nejrovnoměrnější pokrytí vzorkovaného prostoru a zároveň, aby vypadaly dostatečně náhodně. Díky tomu nedochází ke shlukům.



Obrázek 2.5: Nákres jednoduché neuronové sítě se třemi skrytými vrstvami obsahujícími právě jeden neuron na vrstvu.

2.6 Problém mizejícího gradientu

V následující sekci bychom rádi krátce představili *problém mizejícího gradientu*. Při zapojení více skrytých vrstev s určitou aktivační funkcí nabývá gradient ztrátové funkce hodnoty blížíící se nule a tím znemožňují nebo alespoň zpomalují rychlost trénování. Teto problém nastává, protože určité aktivační funkce jako je sigmoid, smršťují veliký vstupní prostor mezi jedničku a nulu. Způsobující malé změny výstupu, při velkých změnách vstupu aktivační funkce a derivace nabývají malých hodnot.

Mějme jednoduchou neuronovou síť s právě jedním neuronem na každé vrstvě a třemi skrytými vrstvami, ilustrace na obrázku 2.5. Váhy této sítě označíme w_1, \dots, w_4 , biasy b_1, \dots, b_4 a C nějakou ztrátovou funkci. Výstup j -tého neuronu je $f(z_j)$, kde f je aktivační funkce a $z_j = w_j a_{j-1} + b_j$ je vážený vstup neuronu.

Při malé změně Δb_1 dostáváme změnu výstupu

$$\Delta a_1 \approx \frac{\partial f(w_1 a_0 + b_1)}{\partial b_1} \Delta b_1 = f'(z_1) \Delta b_1.$$

Změna biasu Δb_1 způsobí změnu výstupní aktivace Δa_1 , který způsobí změnu

$$\Delta z_2 \approx \frac{\partial z_2}{\partial a_1} \Delta a_1 = w_2 \Delta a_1 = w_2 f'(z_1) \Delta b_1.$$

Pokud bychom pokračovali a šířili se sítí dál dostali bychom výraz pro změnu ztrátové funkce ΔC v závislosti na změně biasu Δb_1 :

$$\Delta C \approx f'(z_1) w_2 f'(z_2) \dots f'(z_4) \frac{\partial C}{\partial a_4} \Delta b_1. \quad (2.18)$$

Vydělením rovnice (2.18) změnou biasu Δb_1 dostáváme konečně

$$\frac{\partial C}{\partial b_1} = f'(z_1) w_2 f'(z_2) \dots f'(z_4) \frac{\partial C}{\partial a_4}. \quad (2.19)$$

Uvažujme aktivační funkci sigmoid $f(x) = \sigma(x)$. Výraz na pravé straně rovnice (2.19) je až na poslední člen součinem výrazů $w_i \sigma'(z_i)$. Derivace funkce sigmoid dosahuje maxima v nule $\sigma'(0) = 1/4$. Použijeme-li standardní inicializaci pro váhy $w \sim \mathcal{N}(0, 1)$, bude pro většinu vah splněna nerovnice $|w_i| < 1$ a jednotlivé členy budou splňovat nerovnici $|w_i \sigma'(z_i)| < 1/4$. Což způsobí exponenciální pokles hodnoty součinu těchto členů.

Porovnejme nyní hodnoty výrazů $\partial C / \partial b_1$ a $\partial C / \partial b_3$. Obdobně jako jsme získali rovnici (2.19) dojdeme k

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3) w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4} = \frac{\frac{\partial C}{\partial b_1}}{\sigma'(z_1) w_2 \sigma'(z_2) w_3},$$

ze které je patrné, že parciální derivace ztrátové funkce podle b_1 bude nabývat mnohem menších hodnot než parciální derivace ztrátové funkce podle b_3 . Tím jsme popsali esenciální základ problému mizejícího gradientu.

Opačnou situaci dostáváme pokud váhy během tréninku zvyšují svoji hodnotu. Potom nemusí být splněna podmínka $|w_i \sigma'(z_i)| < 1/4$. Pokud bude hodnota tohoto členu větší než 1, začne gradient naopak exponenciálně růst během zpětného šíření a tím vznikne problém *explodujícího gradientu*.

Řešením je použití jiných aktivačních funkcí jako je funkce *ReLU* nebo použití takzvaných reziduálních sítí, které poskytují feedback předcházejícím vrstvám. Problém explodujícího gradientu řeší například metoda *gradient clipping*, která škáluje gradienty tak, že jejich norma je omezená určitou hodnotou.

2.7 Konvoluční neuronové sítě

Konvoluční neuronové síť (CNN z anglického *convolutional neural network*) je regularizovanou verzí vícevrstvého perceptronu, se kterým jsme pracovali doposud. Vícevrstvý perceptron je plně spojitá síť, tj. každý neuron vrstvy je spojen se všemi neurony sousedících vrstev. Plná spojitost zatěžuje síť problémem přeučení a je potřeba tyto sítě regularizovat. Jiný je přístup k regularizaci u CNN, které pracují s hierarchickými znaky dat a tvoří filtry, které si zachycují až elementární vzory.

Konvoluční neuronové sítě jsou inspirovány biologickými procesy, konkrétně spojením neuronů v CNN je velmi podobné organizaci ve zrakovém centru zvířat a lidí.

CNN využívá tři základních nástrojů, *lokální receptivní pole*, *sdílené váhy* a *pooling*, které představíme v následujícím textu.

Při klasifikaci obrazu o velikosti např. 28×28 , plně spojitými neuronovými sítěmi jsme vstup převedli do vektoru 784×1 , tvořících první, vstupní, vrstvu neuronů. Představa vhodnější pro CNN je čtverec tvořený 28×28 neurony s hodnotami korespondujícími s pixely obrazu. Neurony následující, skryté, vrstvy nejsou ovšem propojeny se všemi neurony vstupní vrstvy, ale jen s malým regionem neuronů této vrstvy. Tento region nazýváme *lokální receptivní pole* a operaci, která mapuje neurony z lokálního receptivního pole do neuronu skryté vrstvy *konvoluce*.

Konvoluce je funkce, která vezme matici čísel, nazývanou kernel, filtr, maska nebo v našem případě lokální receptivní pole a podle ní transformuje vstupní obraz, čímž vznikne tzv. *příznaková mapa*. Konvoluci obrázku \mathbb{I} a masky \mathbb{K} definujeme po prvcích jako

$$(\mathbb{I} * \mathbb{K})_{i,j} = \sum_{m,n} \mathbb{I}_{m,n} \mathbb{K}_{i-m,j-n}.$$

Konvoluce obrazu se používá k mnoha užitečným aplikacím v počítačovém vidění jako je detekce hran, rozostření nebo zaostření, záleží na velikosti a tvaru masky.

Pokud použijeme konvoluci na obraz velikosti 6×6 s maskou velikosti 3×3 dostaneme příznakovou mapu velikosti 4×4 . Existuje jen 16 unikátních možností jak umístit masku do obrazu. Při opakované aplikaci konvoluce by velikost obrázku klesala až na velikost 2×2 . Proto využíváme *padding*, česky *obtékání*, tj. přidáme pixely kolem obrazu při paddingu o 1px zvětšíme velikost obrazu před konvolucí na 8×8 a po konvoluci s maskou 3×3 budeme mít příznakovou mapu velikosti 6×6 . Většinou doplňujeme pixely o hodnotě 0. Konvoluci, kde přidáváme hranici nazýváme *same* pokud nepřidáváme *valid*.

Doposud jsme mluvili o lokálním receptivním poli, které se posouvá o jeden pixel. U CNN se často setkáme s hyperparametrem, který určuje posun masky po obraze s názvem *stride* (pro stride roven dvěma dostáváme poloviční vstoup).

Mějme vstupní obrázek \mathbb{I} s C kanály (pro RGB se $C = 3$). Konvoluce s maskou \mathbb{K} velikosti $w \times h \times C \times F$, šířky w výšky h s stridem s vrátí výstup s F kanály je počítána jako

$$(\mathbb{I} * \mathbb{K})_{i,j,k} = \sum_{m,n,o} \mathbb{I}_{i-s+m,j-s+n,o} \mathbb{K}_{m,n,o,k}.$$

Další specifikací CNN jsou *sdílené váhy* a *bias*. Z dosavadních znalostí bychom řekli, že v CNN budeme mít pro každý skrytý neuron tolik vah, jak velká je používaná maska. Pravdou ovšem je, že

používáme stejná váhy napříč celým obrázkem. Proto se všechny neurony v první vrstvě učí detekovat stejné příznaky na různých místech obrazu. Tímto radikálně zmenšíme počet parametrů modelu a tím urychlíme učení sítě. Pro každou příznakovou mapu, kterou dostaneme konvolucí s maskou velikosti např. 5×5 potřebujeme 25 vah a jeden bias, dohromady 26 parametrů. Pokud budeme chtít 20 příznakových map máme dohromady 520 parametrů definujících konvoluční vrstvu. V plně spojitě síti se skrytou vrstvou tvořenou 30 neurony, bychom pro obrázek velikosti 28×28 potřebovali učít 784×30 vah a 30 biasů, celkem 25550 parametrů.

Poslední důležitou specifikací, kterou si uvedeme je *pooling*. Kromě právě popsaných konvolučních vrstev obsahuje CNN ještě *pooling vrstvy*, které se většinou vážou hned za konvoluční vrstvy. Pooling vrstvy v určitém hledisku zjednodušují informaci vrácenou z konvolučních vrstev. Pooling layer má na vstupu příznakovou mapu vrácenou konvolucí, kterou „zhustí“. Vysvětleme si operaci na konkrétním příkladu *maximové* pooling layer s maskou velikosti 2×2 , kde pooling vrátí maximální hodnotu z regionu pod maskou.

Krom maximového pooling se používá například *L2 pooling*, který místo maxima vrací odmocninu součtu kvadrátů aktivací v regionu. Zase se jedná o kondenzaci informace z konvoluční vrstvy.

V práci využíváme architekturu Faster RCNN, která zahrnuje RoI pooling (z angl. „regions of interest“). Vrstva s RoI poolingem má dva vstupy:

1. Příznakovou mapu vrácenou z CNN s několika konvolučními a maximovými pooling vrstvami.
2. Matici velikosti $N \times 5$ reprezentující oblasti zájmu, kde N je počet oblastí zájmu a sloupce mají hodnoty indexu obrázku a souřadnice 4 rohů oblasti zájmu.

RoI pooling vezme ke každé oblasti zájmu korespondující část příznakové mapy a transformuje ji na předem definovanou velikost.

Kompletní konvoluční neuronová síť používá kombinace konvolučních vrstev a pooling vrstev k vytažení příznaků z dat a následně použije plně spojitě vrstvy ke klasifikaci stejně jak bylo popsáno výše.

Posledním nástrojem, který v této kapitole představíme je *nemaximální suprese*. Algoritmus detekce objektu v obraze navrhne ohraničení objektu, který na snímku detekujeme. Pro výběr nejlepšího možného ohraničení se využívá právě nemaximální suprese. Nemaximální suprese vybere ohraničení na základě tzv. *objectiveness* míry a IoU. Hodnotu míry objectiveness vrací model a vyjadřuje jakou část ohraničení zabírá objekt. Pokud máme ohraničení s IoU větším než námi zvolená míra, tak vypustíme ohraničení s menší hodnotou míry objectiveness.

Kapitola 3

Detekce a segmentace ruky v obrazu

V první kapitole představíme teorii a výsledky algoritmů a metod sloužících k segmentaci a detekci objektu v obraze. Probrané metody aplikujeme jak na akademické příklady, tak na reálné obrázky ruky. V první sekci probereme segmentaci obrazu. Konkrétně metodu normalizovaných řezů a metodu aktivních kontur. Druhá sekce se věnuje detekci objektu v obraze. Tento problém řešíme pomocí metody detekce objektu podle barvy a detekce pomocí konvolučních neuronových sítí.

3.1 Segmentace

Při řešení problému segmentace obrazu narážíme na otázku, jaké je správné rozdělení domény obrázku na podmnožiny. Víme, že takových dělení existuje mnoho, ale jak vybereme to „správné“? Musíme brát v úvahu dvě skutečnosti. Tou první je, že dělení je inherentně hierarchické. Přírozanější cestou k reprezentaci je tedy stromová struktura vystihující hierarchické dělení. Druhou skutečností je, že nemusí existovat jedno správně řešení. Nabízí se tedy využití Bayesovského formalizmu pro zapojení apriorní znalosti jako je barva, jas, textura nebo symetrie objektu.

3.1.1 Metoda normalizovaných řezů grafu

Jako první metodu segmentace obrazu si představíme metodu normalizovaných řezů grafu [13]. Tento přístup je založený na formulaci problému shlukování v teorii grafů. Mějme neorientovaný ohodnocený graf $G = (V, E)$ reprezentující množinu bodů v libovolném příznakovém prostoru. Každý uzel grafu reprezentuje právě jeden bod a každý uzel je spojen se všemi ostatními hranou. Hrana mezi uzly v_i, v_j je ohodnocena vahou $w(v_i, v_j)$, což je podobnostní funkce uzlů v_i, v_j . Cílem úlohy shlukování je rozdělit množinu uzlů na její disjunktní podmnožiny, pro které platí vysoká podobnost mezi uzly v jednotlivých podmnožinách a nízká podobnost mezi uzly spadající do různých podmnožin.

Graf $G = (V, E)$ rozdělíme na dvě disjunktní množiny A, B splňující: $A \cap B = \emptyset \wedge A \cup B = V$ odstraněním hran spojující tyto dvě části. Stupeň nepodobnosti dvou částí definujeme jako

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v).$$

Problém hledání optimálního dělení grafu na dvě části je aktivně studovaný problém. My jsme se rozhodli využít přístup představený v [13, 14], který je založený na minimalizaci normalizovaného řezu, definovaného následovně:

$$NCut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)},$$

kde $assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$ je totální propojení uzlů z A se všemi ostatními uzly.

Jak bylo v článku dokázáno, je možné tuto úlohu převést na úlohu řešení obecného problému vlastních čísel

$$(D - W)y = \lambda Dy,$$

kde prvky matice W jsou jednotlivé hodnoty vah a platí $W_{u,v} = w_{u,v}$, tudíž je tato matice symetrická. Diagonální matice D , pro kterou platí $D_{u,u} = \sum_{v \in V} w_{u,v}$.

Navržený shlukovací algoritmus se skládá ze čtyř kroků:

- Navrhnete graf $G = (V, E)$ a nastavte váhy na hranách.
- Vyřešte úlohu $(D - W)y = \lambda Dy$ pro vlastní vektory asociované s nejmenšími vlastními čísly.
- Použijte vlastní vektor s druhým nejmenším vlastním číslem k rozdělení grafu na dvě části.
- Učiňte rozhodnutí, zda-li má být daná část dělena dále, pokud ano postupujte s vybranou částí od prvního kroku.

Řešení úlohy hledání vlastních vektorů můžeme rozumět jako hledání vlastních čísel nenormalizovaného Laplaciánu [14], který je definovaný jako

$$L = D - W,$$

přehled jeho vlastností je zpracovaný např v [15]. V této práci se zmíníme jen o nejdůležitějších vlastnostech, které jsou dokázány v odkazované literatuře.

Věta. (Vlastnosti L)

Matice L má následující vlastnosti:

1. Pro každý vektor $v \in \mathbb{R}^n$ platí

$$v^T Lv = \frac{1}{2} \sum_{i,j=1}^n w_{ij} (v_i - v_j)^2.$$

2. Matice L je symetrická a pozitivně semidefinitní.
3. Nejmenší vlastní číslo matice L je 0 a vlastní vektor asociovaný s tímto vlastním číslem je jednotkový vektor.
4. Matice L má n nezáporných, reálných vlastních čísel.

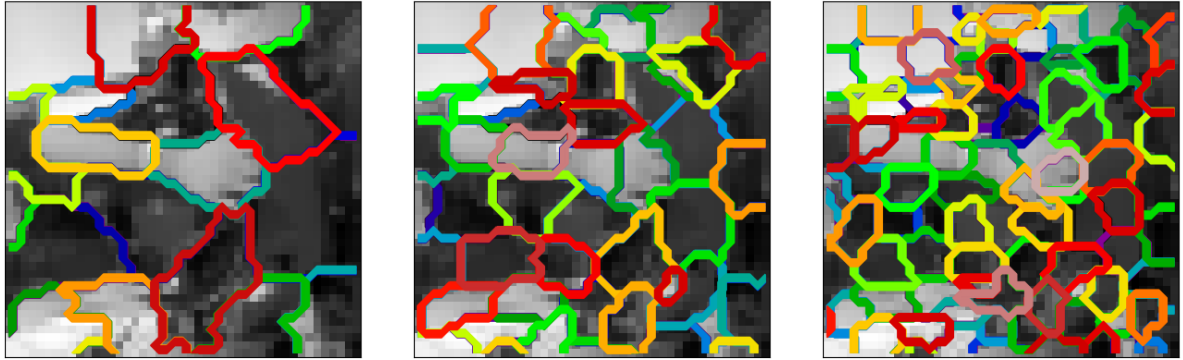
Věta. (Spektrum Laplaciánu a počet propojených komponent)

Nechť je G neorientovaný graf s nezápornými ohodnoceními hran. Potom platí, že násobnost k , nulových vlastních čísel nenormalizovaného Laplaciánu asociovaného s tímto grafem je rovna počtu propojených komponent tohoto grafu A_1, \dots, A_k .

Naše implementace je založená na ohodnocující funkci reprezentující rozdíl v jasů dvou pixelů. Čím menší je rozdíl jasů dvou částí obrazu, tím větší je pravděpodobnost, že tyto dvě části tvoří společný celek. Volba příznaku může být ovšem libovolná, ať je to textura nebo barva. Ohodnocující funkce je definována následovně,

$$w_{ij} = \exp(-\|F(i) - F(j)\|_2^2) \times \mathcal{K}, \quad (3.1)$$

kde \mathcal{K} je rovno $\exp(-\|X(i) - X(j)\|_2^2)$, pokud $\|X(i) - X(j)\|_2 < r$, jinak rovno nule. Proměnné použité v (3.1) mají následující význam.



Obrázek 3.1: Aplikace metody normalizovaných řezů grafu. Zleva vidíme segmentaci na 20, 50 a 100 regionů.

- F je příznakový vektor, v našem případě reprezentující jas.
- $\|F(i) - F(j)\|_2^2$ je euklidovská vzdálenost mezi $F(i)$ a $F(j)$.
- X reprezentuje prostorové umístění uzlu.
- číslo r je libovolné kladné číslo, pokud je vzdálenost pixelů větší než r má hrana mezi těmito pixely váhu 0.

Následuje vyřešení spektrální úlohy, jejímž výstupem jsou vlastní vektory a rozdělení grafu na dvě části podle druhého nejmenšího vektoru. Rekurzivně postupujeme až do bodu, kdy je graf rozdělený do definovaného počtu částí.

Na obrázku 3.1 vidíme aplikaci metody normalizovaných řezů grafu. Vstupní obrázek o rozlišení 720×720 jsme čtyřikrát podvzorkovali. Sestavený graf měl velikost $(32400, 32400)$. Vygenerovali jsme segmentaci na 20, 50 a 100 regionů. Průměrný čas na vygenerování sta segmentů byl 4,92s.

3.1.2 Metoda aktivních kontur

Metoda aktivních kontur, v literatuře také označovaná jako *Snakes* [23], je metoda segmentace z kategorie deformovatelných parametrických modelů. Model aktivní kontury je reprezentovaný deformovatelnou parametrickou křivkou minimalizující energii závislou na omezujících silách a ovlivněnou silami obrazu, které ji přitahují k příznakům obrazu jako jsou hrany.

Mějme Jordanovu křivku C^1 resp. polygon určený posloupnostmi vrcholů $\{v_i | i = 0, \dots, n; v_0 = v_n\}$. Přejděme k parametrizaci této křivky:

$$s(p) = \begin{pmatrix} x(p) \\ y(p) \end{pmatrix}, \quad s(0) = s(1), \quad p \in \langle 0, 1 \rangle. \quad (3.2)$$

Funkcionál energie, který chceme minimalizovat je definovaný jako

$$E = \int E_{int}(s(p)) + E_{ext}(s(p)) dp,$$

kde E_{int} označuje vnitřní energii, která nutí křivku, resp. konturu, ke stahování se a hladkosti a eliminuje špatná řešení. Vnější energii jsme označili jako E_{ext} , jejíž úlohou je pomoci křivce najít hrany objektu v obrázku. Tato síla je tedy definována přímo na obraze.

¹Jordanova křivka je definovaná jako jednoduchá uzavřená rovinná křivka.

Rozeberme si prvně vnitřní energii, kterou [23] definovaly jako

$$E_{int}(s(p)) = \frac{1}{2} \left[\alpha(p) |s'(p)|^2 + \beta |s''(p)|^2 \right].$$

Parametry α a β ovlivňují relativní význam výrazu v součtu, většinou se definují jako konstanty. V prvním výrazu vystupuje čtverec velikosti první derivace křivky, která se zvětšuje s délkou křivky. Ve druhém výrazu je to čtverec velikosti druhé derivace, která nabývá vysokých hodnot pro ostré lomy křivky. Minimalizací vnitřní energie tedy dosahujeme krátké a hladké křivky.

Pomocí Eulerovi-Langrangeovi rovnice dosáhneme řešení úlohy minimalizace funkcionálu:

$$\alpha s''(p) - \beta s'''(p) - F_{ext}(s(p)) = 0, \quad (3.3)$$

kde $F_{ext}(s(p)) = \nabla E_{ext}(s(p))$ značí vnější sílu.

Rovnici (3.3) vyřešíme metodou gradientního sestupu. Převodeme křivku (3.2) do funkci času a stávajícího parametru p , navíc zaměníme nulu na levé straně (3.3) za parciální derivaci s podle času, tím dosáhneme faktu, že po uplynutí dlouhého času, kdy křivka zkonverguje do minima, bude derivace podle času rovna nule. Máme tedy rovnici

$$\frac{\partial s(p)}{\partial t} = \alpha s''(p, t) - \beta s'''(p, t) - F_{ext}(s(p, t)).$$

K numerické implementaci je nutno diskretizovat parametr p . Volíme diskretizaci $p = i/N$, kde $i = 0, \dots, N-1$. Složky vnější síly označíme jako f_x a f_y . Za využití diferenční aproximace dostaneme soubor $2N$ rovnic

$$\begin{aligned} \frac{\partial X_t}{\partial t} &= AX_t + f_x(X_t, Y_t), \\ \frac{\partial Y_t}{\partial t} &= AY_t + f_y(X_t, Y_t), \end{aligned}$$

kde matice $A \in \mathbb{R}^{N \times N}$ je pentadiagonální, tvaru

$$\begin{pmatrix} -2\alpha - 6\beta & \alpha + 4\beta & -\beta & 0 & 0 & \cdots & \alpha + 4\beta \\ \alpha + 4\beta & -2\alpha - 6\beta & \alpha + 4\beta & -\beta & 0 & \cdots & -\beta \\ -\beta & \alpha + 4\beta & -2\alpha - 6\beta & \alpha + 4\beta & -\beta & \cdots & 0 \\ 0 & -\beta & \alpha + 4\beta & -2\alpha - 6\beta & \alpha + 4\beta & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \vdots \\ -\beta & 0 & 0 & 0 & 0 & \ddots & \alpha + 4\beta \\ \alpha + 4\beta & -\beta & 0 & 0 & 0 & \cdots & -2\alpha - 6\beta \end{pmatrix}.$$

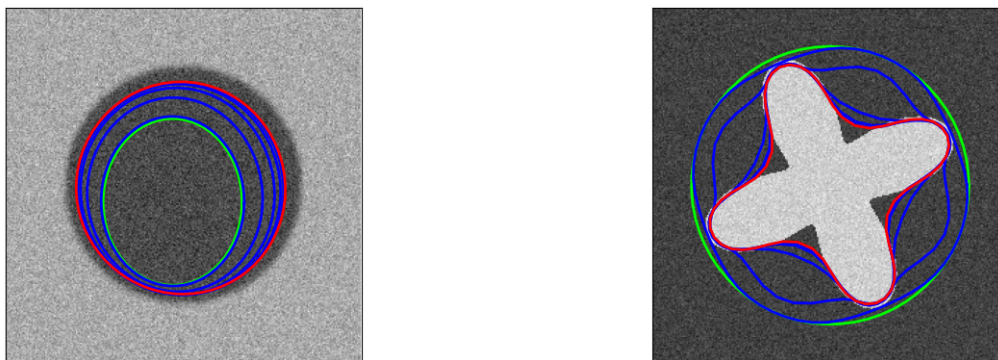
Posledním krokem je diskretizace v čase. Přidáním parametru τ vyjádříme

$$\frac{\partial X_t}{\partial t} = \frac{X_t - X_{t-1}}{\tau}.$$

Předpokládejme nyní pouze malé změny vnější síly v čase. Tento předpoklad nám umožní nahradit vnější sílu v souboru rovnic výrazem $t-1$. Pomocí Choleskyho dekompozice provedeme inverzi matice a konečně dostáváme

$$\begin{aligned} X_t &= (I - \tau A)^{-1} [X_{t-1} + \tau f_x(X_{t-1}, Y_{t-1})], \\ Y_t &= (I - \tau A)^{-1} [Y_{t-1} + \tau f_y(X_{t-1}, Y_{t-1})]. \end{aligned}$$

Vlastní použití výše popsané metody na jednoduchých příkladech lze vidět na obrázku 3.2. V levé části obrázku segmentujeme kruh, který je výsledkem vizualizace Gaussovy funkce v rovině. Nastavení parametrů se ukázalo ideální pro hodnoty $\alpha = 0,001$, $\beta = 0,4$. Vidíme, že volba počtu iterací rovna padesáti nebyla dostačující a kontura neobkresluje celý objekt zájmu. V pravé části obrázku vidíme mírně složitější objekt a výsledek metody aktivních kontur pro volbu parametrů $\alpha = 0,001$ a $\beta = 0,001$.



Obrázek 3.2: Ukázka aplikace metody aktivních kontur na segmentaci dvou různých objektů v 50 iteracích. Inicializační kontura je vyobrazena zelenou barvou. Modré kontury reprezentují stav v každé desáté iteraci. Výsledná kontura je označena červeně.

3.2 Detekce

Detekce objektu je technika počítačového vidění umožňující lokalizaci a identifikaci objektu v obraze nebo videu. Výsledkem detekce objektu je ohraničení (bounding box) kolem detekovaného objektu, který nám dává informaci o poloze objektu v obraze nebo pohybu objektu ve scéně. Jedné se o pokročilejší metodu ve srovnání s rozpoznáváním obrazu. Popíšme si rozdíl na jednoduchém příkladě.

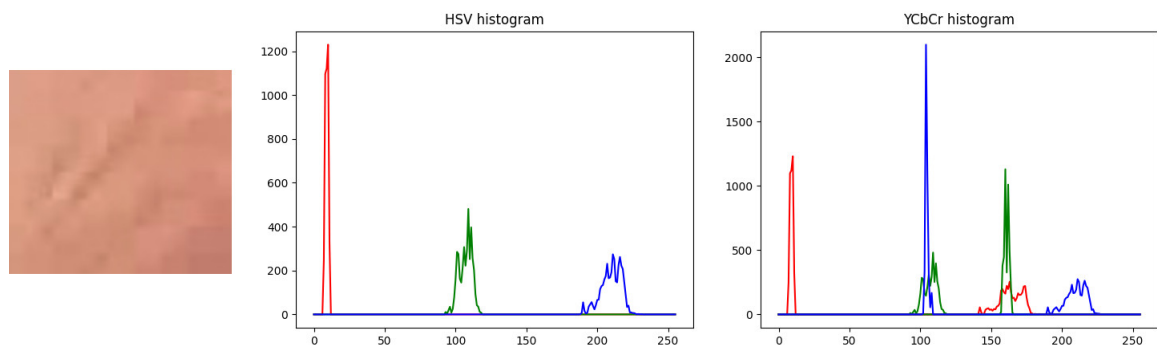
Na vstupu mějme obrázek se dvěma stromy. Rozpoznávání obrazu přiřadí obrazu označení (label) „strom“ oproti tomu z detekce objektu dostáváme ohraničení obou stromů v obraze, kde obě ohraničení mají label „strom“.

Obecně můžeme detekci objektu rozdělit na přístup založený na strojovém učení a přístup založený na hlubokém učení.

V tradičních přístupech založených na strojovém učení využíváme techniky počítačového vidění k analýze obrazu a vytažení určitých příznaků obrazu. Může se jednat analýzu frekvenční domény, barevný histogram obrazu nebo hrany. Na základě, kterých určujeme soubory souvisejících pixelů určujících detekovaný objekt. Vytažené příznaky jsou následně vloženy do regresního modelu, který poskytne predikci na polohu objektu a jeho označení.

Na druhé straně máme přístupy hlubokého učení využívající konvoluční neuronové sítě. Tyto sítě poskytují end-to-end detekci objektu bez učitele. V tomto přístupu není nutnost separátní definice a extrakce příznaků obrazu.

V následujících sekcích si představíme oba přístupy detekce objektu, resp. ruky v obraze. Jako první představíme metodu počítačového vidění využívající segmentace objektu v závislosti na barvě, na kterou navážeme detekci využívající neuronové sítě.



Obrázek 3.3: Na levé straně vidíme vyříznutou oblast z části ruky sloužící pro trénování a vpravo od ní její HSV, resp. YCbCr histogram. Červená, zelená a modrá barva zastupují odstín-sytost-jas, resp. jas-červenou-modrou chrominanci. Podle zastoupení jednotlivých komponent těchto barevných modelů, nastavujeme hranice pro segmentaci.

3.2.1 Detekce objektu podle barvy

Barva je jeden z nejvíce využívaných příznaků v oblasti detekce objektu a jeho sledování ve videu. Základní postup detekce objektu na základě barvy bychom mohli rozdělit do čtyř kroků. V prvním kroku přijde na vstup snímek, následuje vytažení informace o barvě z oblasti zájmu. Třetí krok zahrnuje propojení informace o barvě a následnou identifikaci objektu. Posledním krokem je vlastní odhad pozice objektu. Pro detekci objektů se často využívá barevných histogramů a to hlavně kvůli malým výpočetním nárokům a škálovací i rotační robustnosti. K lokalizaci objektu se využívá mean-shift algoritmu [24]. Tento algoritmus najde okno v obraze, které je nejpodobnější barevnému histogramu objektu zájmu. Podobnost dvou histogramů lze vyjádřit například pomocí Bhattacharyovi vzdálenosti.

Jedním z problémů detekce objektu podle barvy je definice barevného modelu. Při detekci objektu ve videu se osvětlení scény může velmi rychle měnit. Tato obtíž lze řešit zapojením Bayesovské statistiky a dynamické konstrukce objektu a pozadí pomocí smíšeného gaussovského modelu [25].

V ideálním případě má objekt, který chceme detekovat, barvu odlišnou od barvy pozadí. Pro takové případy lze použít běžných metod, které můžeme navíc vylepšit převodem RGB obrazu do obrazu jeho intenzity [24]. Mějme obraz, ve kterém detekujeme červený objekt. Každý pixel obrazu převedeme z RGB do intenzity pomocí vzorce

$$I = \frac{R \cdot R}{G \cdot B},$$

kde R , G a B jsou postupně hodnoty červené, zelené a modré barvy pixelu. U segmentace objektu barvy může být situace náročnější. Správné nastavení barevné hranice pro segmentaci může být široce studovaným tématem [53, 54]. Byly navrženy různé hodnoty hranice prahování pro různé rasy a různé barevné prostory, ať už se jedná o klasický RGB prostor nebo, v segmentaci může často využívané, prostory HSV a YCbCr². V našem případě jsme prováděli segmentaci jak v HSV tak YCbCr prostoru, kde jsme hodnotu parametru prahování určili z barevného histogramu trénovacího obrázku. Trénovací obrázek a jeho barevné histogramy můžeme vidět na obrázku 3.3. Následovala aplikace zpětné projekce histogramu do obrázku, ve kterém provádíme detekci. Zpětná projekce je metoda počítající poměr histogramu tréno-

²HSV je barevný model skládající se ze tří složek: odstín (hue), sytost barvy (saturation) a hodnota jasu (vue). YCbCr je barevný model často využívaný v digitální fotografii. Tento model se skládá ze tří komponent: jas, modrý a červený chrominanci komponent.



Obrázek 3.4: Detekce ruky podle barvy kůže. V levé části obrázku vidíme vstupní obrázek. Aplikací zpětné projekce kombinující pravděpodobnost z HSV a YCbCr prostoru a dilatace s křížovou maskou velikosti 5×5 získáme binární reprezentaci obrázku uprostřed. Na posledním obrázku vidíme výstup, s vykreslenou největší konturou (modře) a jejím konvexním obalem (zeleně) na vstupním obrázku.

cího obrázku T_i a histogramu samotného obrázku I_i :

$$R_i = \frac{T_i}{I_i}. \quad (3.4)$$

Pro každý pixel je vypočtena pravděpodobnost, že patří do hledaného obrázku jako počet binů R_i , do kterých je pixel indexován. Utvoříme-li obrázek, který místo trojkanálové reprezentace má jen jednonábovovou s hodnotami pravděpodobností vrácených zpětnou projekcí, dostaneme obrázek v odstínech šedi. Konturu s největší plochou nazveme detekovaným objektem. Metodu jsme aplikovali na obrázek z kamery snímající řidiče obsluhující trenážer řízení kamionu. Metoda i její výsledky jsou ilustrovány na obrázku 3.4. Postup můžeme shrnout v následujících bodech

- Na vstupu obdržíme trénovací obrázek a obrázek, na kterém segmentujeme objekt.
- Převědeme obrázky do HSV a YCbCr prostorů a spočítáme jejich histogramy.
- Podle vzorce 3.4 spočítáme poměry jejich histogramů a provedeme zpětnou propagaci, kde výslednou pravděpodobnost pixelu obdržíme jako kombinaci pravděpodobností z obou barevných modelů.
- Výsledný obrázek v odstínech šedi binarizujeme a provedeme dilataci.
- Nalezneme největší konturu binárního obrazu, kterou vyhladíme a najdeme její konvexní obal.
- Nalezenou konturu a její obal vykreslíme do originálního obrázku.

Výše zmíněná dilatace je základní morfologická operace provádějící zvětšení regionu. Mějme strukturální element (masku, kernel) S a binární obraz B , potom definujeme dilataci B o S jako

$$B \oplus S = \bigcup_{b \in B} S_b.$$

Posunujeme tedy strukturální element S po obraze B a pokaždé, když se počátek S překryje s hodnotou 1, zapíšeme strukturální element do výstupu inicializovaného s hodnotami 0. Nejčastěji se využívá k zaplnění děr v objektu. Opačná operace se nazývá eroze.

Navržená metoda dosahuje dobrých výsledků v prostředích, kde je barva pozadí co možná nejodlišnější od barvy detekované kůže. Pro náš případ se jedná o dobrou metodu, protože detektor pracuje na

obrazech se stále stejným pozadím, pokud nepočítáme řídicovo oblečení. Zároveň je tato metoda velmi rychlá střední čas zpracování padesáti vstupních obrázků v jazyce Python v3.7 je 56ms na laptopu o specifikacích CPU Intel i5, 7th gen., 2,5GHz, RAM 8GB.

Vhodným vylepšením by bylo aplikovat metodu aktivních kontur, probíranou výše, kde bychom za inicializační konturu braly největší konturu nalezenou metodou výše. Za využití znalosti konkávních míst kontury bychom byly schopni detekovat dlaň a prsty odkud bychom již jednoduše odhadli směr orientace celé ruky. Aktivní kontury jsme nakonec nepoužili. Získali bychom poměrně malé vylepšení tvaru segmentované části, na úkor vysoké výpočetní náročnosti.

3.2.2 Detekce objektu pomocí neuronových sítí

V této sekci se budeme věnovat detekci objektů, převážně rukou, v obraze a extrahovaných snímcích videa pomocí hlubokého učení. Pro řešení problému detekce za využití neuronových sítí využíváme již existující datasety, které rozšíříme o rotace, změny jasů a především o vlastně označené obrázky přímo z trenážeru řízení. Využijeme a porovnáme dvě existující architektury konvolučních neuronových sítí. První přístup využívá regionální konvoluční síť (RCNN), kterou jsme adaptovali na detekci rukou. Trénování této sítě bylo ovšem velmi pomalé a náročné na kapacitu úložiště, což je zapříčiněno hlavně potřebou pamatování si návrhů oblastí detekce. Uvedené problémy, nás přivedli k přechodu k vylepšené architektuře RCNN, která se nazývá Faster-RCNN. Tato síť odstraňuje problém pamatování si navrhovaných oblastí. Výstup této sítě je oblast, ve které se s určitou pravděpodobností vyskytuje lidská ruka, pro zlepšení výsledku redukcí false positive výsledků jsme kromě nemaximální suprese následovně aplikovali metodu, která porovná oblasti a vybere jen ty, které obsahují největší množství pixelů barvy lidské kůže.

Základy neuronových sítí, ale především konvolučních neuronových sítí a matematické metody, které využívají, jsou představeny v první kapitole, na straně 13.

3.2.2.1 Data

Pro trénování naší architektury, jejíž popis najdete níže, jsme využili již existující datasety, které jsme obohatili o svoje vlastně olabelované obrazy. Z již existujících datasetů jsme se rozhodli použít EgoHands [55] obsahující 4800 označených snímků z egocentrického pohledu, Oxford hand data [56] - 4170 snímků a našich vlastních 100 osobně označených snímků. Vzniklý dataset o 9070 snímcích jsme zvýšením a snížením jasu rozšířili na dataset o 27210 snímcích.

Dataset jsme rozdělili na tři části: trénovací, testovací a validační. Trénovací dataset používáme pro trénování sítě tj. hledání modelu, který nejlépe odhaduje parametry sítě. Validací dataset slouží k vyhodnocení parametrů modelu a následné vylepšení hyperparametrů modelu. Tento dataset, tedy používáme při vývoji modelu, ale neúčastní se vlastního učení. Poslední, testovací dataset používáme pro vyhodnocení modelu a jeho přesnosti. Tento model se nijak nepodílí na trénování ani vývoji modelu, proto je vhodným a nezávislým prostředkem k určení míry přesnosti modelu. Rozhodli jsme se pro rozdělení 60%-20%-20% pro postupně trénovací, testovací a validační data. Poznamenejme, že tento testovací dataset není ideální, jelikož obsahuje obrázky, které jsou od testovacích rozdílné jen hodnotou jasu.

Implementace jednotlivých architektur probíhala na aplikačním rámci, častěji nazývaném framework, TensorFlow [57], proto se data musela převést z XML a JSON souborů do TFRecord, který je preferovaným formátem souboru při práci v TensorFlow. Jedná se o jednoduchý formát, který se skládá ze sekvence binárních záznamů. Obecně se tedy jedná o protocol buffer, metodu serializace strukturovaných dat. Tato struktura je vhodná pro programy, které mezi sebou komunikují přes síť.

3.2.2.2 Detekce pomocí RCNN

Použití standardní konvoluční sítě napojené na plně propojenou vrstvu, která vykonává klasifikaci, naráží na problém proměnné délky výstupní vrstvy sítě. Tato délka není konstantní, protože nemáme pevné číslo určující počet výskytů objektu na snímcích. Naivním přístupem bychom mohli provádět klasifikaci pomocí CNN na jednotlivých oblastech zájmu. Problém, který by vyvstal z tohoto přístupu je, že oblasti zájmu mohou mít rozdílné prostorové rozměry a lokace. Potřebovali bychom tedy obrovské množství oblastí zájmu, což by vyústilo ve velkou výpočetní náročnost.

Řešení tohoto problému nabízí RCNN [31]. Navržená metoda obsahuje selektivní prohledávání, která vybere dva tisíce oblastí zájmu v obraze. Selektivní prohledávání můžeme shrnout do tří kroků:

1. Vygeneruj inicializační segmentaci vstupního obrazu, např. za využití metody normalizovaných řezů, podsektce 3.1.1.
2. Rekurzivně kombinujeme malé segmentované oblasti do větších pomocí hladového algoritmu popsaného níže.
3. Vygenerované oblasti použij k navržení finálních kandidátů na oblasti zájmu.

Hladový algoritmus, který využíváme v druhém kroku selektivního výběru, hledá v každém svém kroku lokální minimum resp. maximum, přičemž existuje šance nalezení extrému globálního. V našem případě vybíráme z množiny všech segmentů obrazu podmnožinu, splňující určitou vlastnost, a její ohodnocení je maximální. Obecně postupuje podle algoritmu:

1. Setřídíme posloupnosti původní množiny do posloupnosti podle klesající váhy.
2. Položíme $A_0 = \emptyset$.
3. Postupně procházíme posloupnost a vytváříme množiny A_i .
 - (a) Je-li pro množinu $A_{i-1} \cup \{i\}$ splněna daná podmínka, položíme $A_i = A_{i-1} \cup \{i\}$.
 - (b) Jinak platí $A_i = A_{i-1}$.
4. Tímto způsobem projedeme celou posloupnost, skončíme s množinou A_n , jejíž prvky splňují danou vlastnost a součet jejich ohodnocení je maximální.

V naší aplikaci máme množinu všech segmentů, z nichž vybereme dva sousedící sobě nejpodobnější. Vybrané segmenty sloučíme do jednoho velkého segmentu a tento postup opakujeme. Pro selektivní prohledávání používáme následující čtyři míry podobnosti [59].

Barevná podobnost definovaná jako

$$s_{color}(r_i, r_j) = \sum_{k=1}^n \min(c_i^k, c_j^k),$$

kde c_i^k je hodnota histogramu pro k-tý bin. K získání 75-dimenzionálního, barevného deskriptoru, sloučíme histogramy o 25 binech, které jsou vypočteny pro každý kanál obrazu.

Texturové příznaky jsou vypočteny vytáhnutím gaussovských derivací v osmi směrech pro každý kanál. Pro každou kombinaci směru a kanálu vypočteme histogram o deseti binech a dostáváme 240-dimenzionální deskriptor příznaků. Texturovou podobnost dvou oblastí vypočteme pomocí průniku histogramů, jako

$$s_{texture}(r_i, r_j) = \sum_{k=1}^n \min(t_i^k, t_j^k),$$

kde t_i^k je i -tá hodnota histogramu v k -tém binu texturového deskriptoru.

Velikostní podobnost zaručuje dřívější spojení malých regionů a je definována jako

$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(I)},$$

kde $\text{size}(I)$ je velikost celého vstupního obrázku v pixelech.

Poslední mírou podobnosti je kompatibilní podobnost, která udává jak dobře do sebe navzájem dva regiony (r_i a r_j) pasují. Pokud jeden pasuje do druhého, tak chceme, aby se sloučili a tím se zaplnily díry a mezery. Tato podobnost je definována jako

$$s_{fill}(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_j)}{\text{size}(I)},$$

kde $\text{size}(BB_{ij})$ je velikost ohraničení kolem r_i a r_j .

Celkovou podobnost potom definujeme jako lineární kombinaci výše definovaných podobností

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),$$

kde $a_i \in \{0, 1\}$ v závislosti na tom jestli danou podobnostní mírou používáme nebo ne.

Autoři článku [59] porovnávají tři různé strategie selektivního prohledávání: „Quality“, „Fast“ a „Single strategy“. Single strategy pracuje nad barevným prostorem HSV a používá všechny podobnosti. Strategie „Fast“ kombinuje hypotézy ze strategií pracujících nad HSV prostorem a Lab³ prostorem pro všechny podobnosti a výběrem podobností: texturová, velikostní a kompatibilní podobností. Poslední strategie „Quality“ kombinuje strategie z HSV, Lab, rgI^4 , H^5 , I^6 se čtyřmi různými kombinacemi podobností. K segmentaci používají algoritmus založený na grafech [58].

Nejlepších výsledků dosahovala strategie „Quality“, která navrhla 22491 regionů za 17s se senzitivitou 0,904. Strategie „Fast“ navrhla 3572 regionů za 3,8s se senzitivitou 0,829. Tyto dvě strategie jsme implementovali a otestovali na 900 obrázcích z EgoHands a 100 vlastních. K segmentaci jsme využili metodu normalizovaných řezů grafu a metodu založenou na grafech [58]. Selektivní prohledávání s metodou normalizovaných grafů se projevilo být mnohem pomalejší než při použití metody založené na grafech. Průměrný čas běhu algoritmu s metodou normalizovaných řezů pro strategii „Fast“ byla 18,7s pro 1000 regionů pro „Quality“ 5,2s na 1000 regionů. Oproti tomu selektivní prohledávání s grafovou metodou, použitou v článku, a strategii „Fast“ dosahuje průměrného času 0,7s na 1000 regionů a 1,2s pro strategii „Quality“. Ukázka výstupu algoritmu je na obrázku 3.5. V RCNN použijeme strategii „Quality“ a segmentaci založenou na grafech.

Selektivní prohledávání vrátí návrhy na objekty zájmu, které jsou transformovány na čtverce a poslány do konvoluční sítě, která vrátí příznakový vektor dimenze 4096. CNN má funkci extraktoru příznaků, které jsou poslány do navazující husté vrstvy (dense layer). Tyto příznaky jsou následně zpracovány pomocí metody podpůrných vektorů (SVM z anglického *support vector machine*), která provede klasifikaci přítomnosti objektu v navrženém regionu.⁷ Krom predikce přítomnosti objektu v oblasti zájmu vrací algoritmus čtyři predikované hodnoty. Tyto hodnoty jsou parametry odchylek v různých směrech, které pomáhají k vykreslení ohraničení objektu v obrázku. Schéma celkové architektury ilustruje obrázek 3.6.

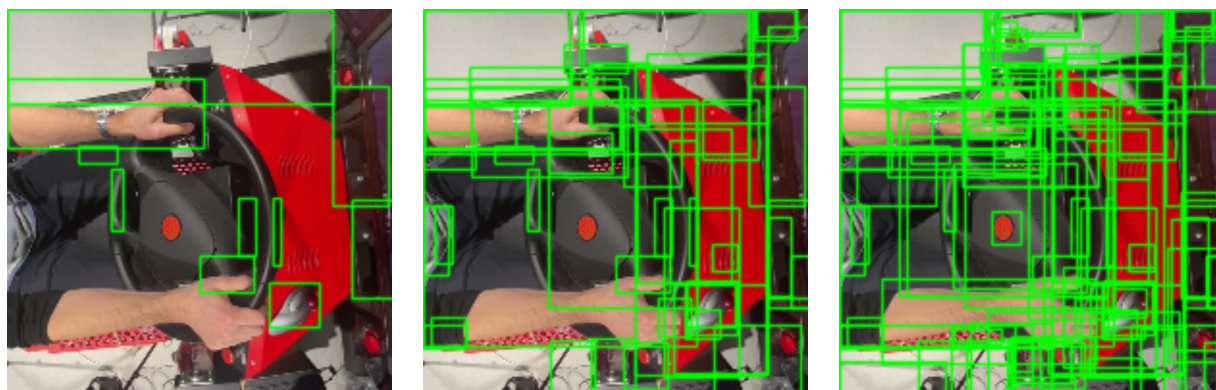
³Barevný prostor Lab má barvu definovanou hodnotami ze tří os: světllost (L), zelená-červená (a), modrá-žlutá (b).

⁴Barevný prostor rgI má červený a zelený normalizovaný kanál RGB a intenzitu I.

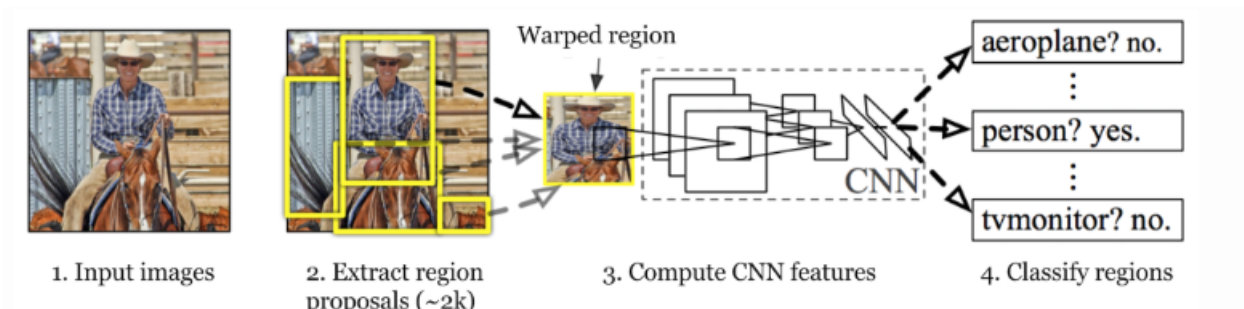
⁵Kanál H (odstín) prostoru HSV.

⁶Intenzita resp. obrázek v odstínech šedi.

⁷Teorie SVM je k nalezení v apendixu na straně 63.



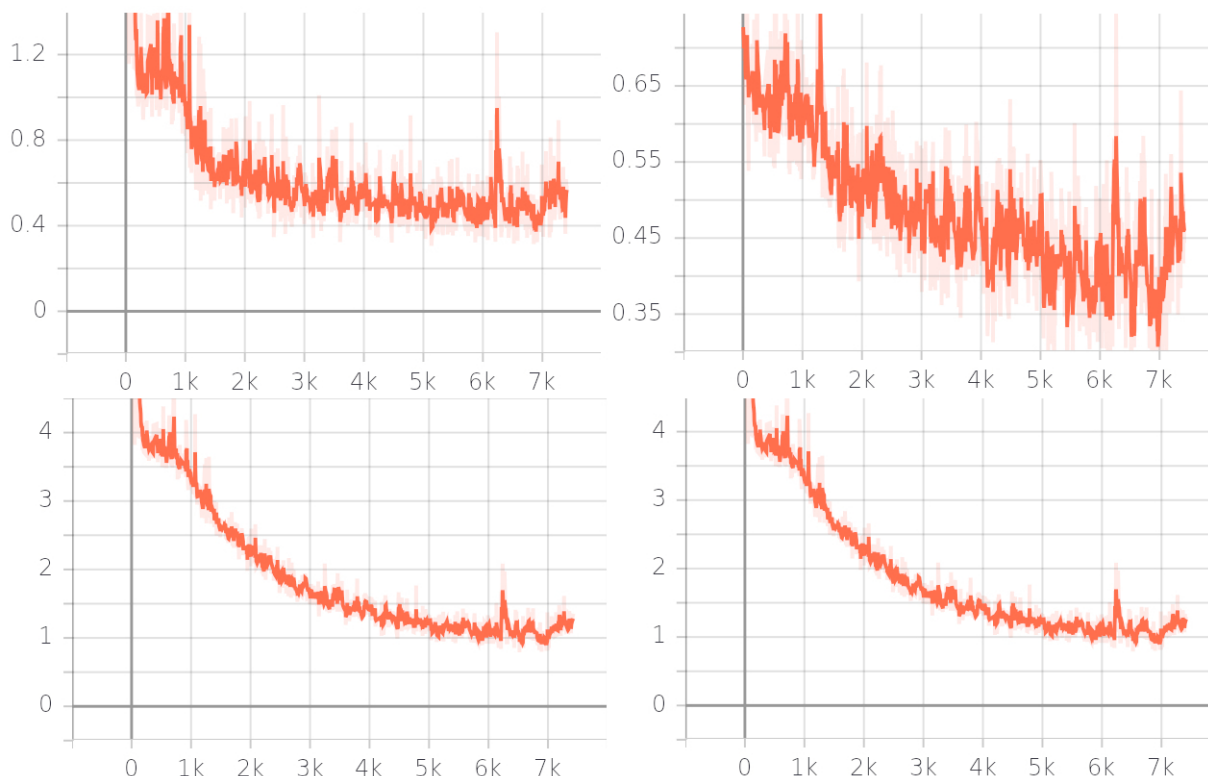
Obrázek 3.5: Regiony navržené selektivním prohledáváním využívající k segmentaci metodu založenou na grafech z [58]. Zleva máme vykreslené postupně 10, 50 a 100 regionů.



Obrázek 3.6: Schéma architektury RCNN. Selektivní výběr navrhně na vstupním obraze (1) kolem dvou tisíc oblastí zájmu (2), které jsou následně předány konvoluční síti ke generaci příznaků (3) a klasifikovány pomocí SVM klasifikátoru (4). [31]

ztráta klasifikace	ztráta lokalizace	celková ztráta	celková normalizovaná ztráta
0,49	0,36	1,08	0,92

Tabulka 3.1: Tabulka s hodnotami ztrátových funkcí po osmi tisíc krocích na architektuře RCNN s CNN ResNet-50.



Obrázek 3.7: Grafy ztrátových funkcí klasifikace (vlevo nahoře), lokalizace (vpravo nahoře), celkové ztráty (vlevo dole) a celkové normalizované ztráty (vpravo dole). Osa x reprezentuje počet iterací učení, osa y reprezentuje hodnotu ztrátové funkce.

V našem případě jsme kromě nemaximální suprese zapojily část z algoritmu navrženém v sekci 3.2.1. Tento algoritmus porovnal navržené oblasti a vybral maximálně dvě oblasti (v naší praktické aplikaci počítáme s maximálně dvěma rukama v obraze), jejichž kontury kolem segmentovaných oblastí obsahující pixely kůže mají největší plochu.

K vytažení příznaků z oblastí zájmu byla použita předtrénovaná síť ResNet-50, která je shrnutá v dodatku na straně 61. Architekturu jsme nechali na našem datasetu dotrénovat na necelých osmi tisících krocích po dobu 15ti hodin, kdy dosahovala hodnota ztrátové funkce hodnoty 1,08 pokud bychom zastavili trénování o přibližně tisíc kroků dříve dosáhli bychom ještě menší hodnoty ztrátové funkce cca. 0,81. Ztráta na klasifikaci nabyla hodnoty 0,49 a ztráta na lokalizaci hodnoty 0,36. Právě zmíněné hodnoty jsou uvedeny v tabulce 3.1 a jednotlivé ztrátové funkce na obrázku 3.7. Využití metody normalizovaných řezů prodlužovalo trénink i samotné vybavování až o 20 sekund. Proto jsme nakonec implementovali běžně užívanou grafovou metodu [58]. Minimum ztrátové funkce bylo hledáno pomocí stochastického gradientního sestupu s momentem, ve kterém v každé iteraci upravíme hodnoty parametrů

podle následujícího vztahu

$$\begin{aligned} v_t &= \beta v_{t-1} + \alpha \nabla_{\theta} C(\theta, X, y), \\ \theta_{t+1} &= \theta_t - v_t, \end{aligned} \quad (3.5)$$

kde θ je vektor parametrů modelu⁸, C ztrátová funkce, V_t je rychlost v iteraci t ⁹, parametr α je rychlost učení a β je hyperparametr z intervalu $(0, 1)$ udávající rychlost vymizení příspěvků předcházejících gradientů. Poslední parametr jsme nastavili na obvyklou hodnotu 0,9 [60]. Rychlost učení jsme v průběhu tréninku nastavovali podle algoritmu kosinového sestupu [61]. Kosinový sestup vrací hodnoty rychlosti učení podle iteračního předpisu

$$\alpha_{t+1} = \alpha_{min}^i + \frac{1}{2} (\alpha_{max}^i - \alpha_{min}^i) \left(1 + \cos \left(\frac{T_{curr}}{T_i} \pi \right) \right),$$

kde T_i je počet iterací, po kterých se znovu spustí stochastický gradientní sestup, nastaveno na 500. T_{curr} udává počet iterací, které proběhly od posledního restartu. Hyperparametry α_{min} a α_{max} určují interval učení, α_{min} jsme nastavili na 0,8 a rozdíl těchto dvou hyperparametrů byl inicializován na hodnotu 1,3.

Pro toto nastavení architektury jsme na testovacích datech obdrželi hodnotu mAP 64%, bez použití následného algoritmu detekce barvy kůže se jednalo jen o 37%. Pro vyhodnocení této metriky byla zvolena mez IoU 0,9.

Samotné vybavování sítě (inference) trvá v průměru 58 sekund, což je pro naše praktické účely, on-line detekce rukou, nedostačující. Pro každý snímek se klasifikuje kolem dvou tisíc návrhů oblastí a navíc musíme ukládat příznakový prostor každého z navrhovaných oblastí, což je velmi náročné jak na výpočetní sílu, tak na paměť grafické karty.

3.2.2.3 Detekce pomocí Faster RCNN

V roce 2015 byla představena vylepšená architektura Faster RCNN [62]. Faster RCNN se skládá ze dvou sítí, síť navrhuje oblasti (region proposal network, RPN) a síť detekující objekty na navržených oblastech. Autoři sítí RCNN a Faster RCNN vytvořili i síť Fast RCNN, kterou můžeme časově a technologicky zařadit mezi dvě zmíněné sítě. Fast RCNN bylo přirozené vylepšení RCNN a snaha o menší výpočetní složitost. Fast RCNN na rozdíl od RCNN neposílá všechny navržené oblasti ze selektivního prohledávání do CNN k nalezení příznaků, ale vytvoří příznakový prostor celého obrázku, vytáhne příznaky příslušející každé oblasti zájmu a přeškáluje na danou velikost pomocí RoI slučovací vrstvy (region of interest pooling layer). Nakonec je provedena predikce pomocí softmax vrstvy.

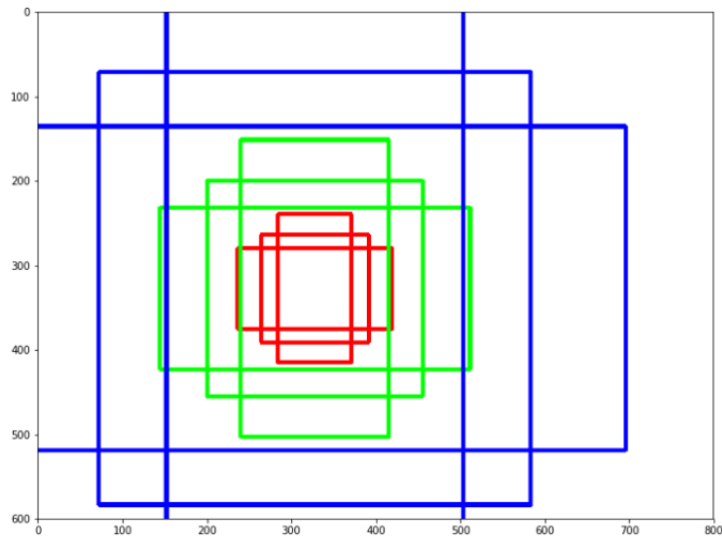
Slučovací vrstva RoI obdrží dva vstupy:

1. Příznakový prostor o pevné velikosti, který je vrácen CNN složené z několika konvolučních vrstev a slučovacích vrstev přes maxima (max pooling).
2. Matici velikosti $N \times 5$ reprezentující oblasti zájmu. Číslo N udává počet oblastí zájmu. První sloupec reprezentuje index obrázku a zbývající čtyři jsou souřadnice rohů oblastí zájmu.

Každé oblasti zájmu je přidělena část vstupního příznakového prostoru, se kterou je asociována a přeškálujeme ji na předem definovanou velikost (pro Faster RCNN přeškálujeme na velikost 7×7). Přeškálování provádíme následovně

⁸Konvoluční síť ResNet50 má 25,6 milionů parametrů.

⁹Používá se označení rychlost nebo moment. Moment je výsledek násobení rychlosti s hmotností a hmotnost je pro daný algoritmus nastavená na hodnotu jedna.



Obrázek 3.8: Rozmístění kotev na pozici (320, 320). Červenou, zelenou a modrou barvou jsme označili postupně velikosti 128×128 , 256×256 a 512×512 . Každá z velikostí je navíc vykreslena pro tři různé poměry stran (1 : 1, 1 : 2, 2 : 1).

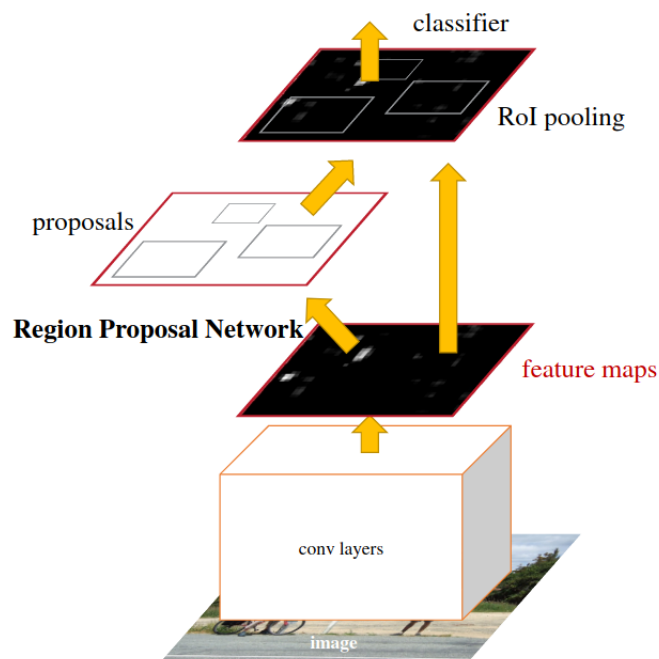
1. Rozdělíme oblast zájmu na stejně velké části (počet částí je stejný jako dimenze výstupu).
2. V každé části najdeme největší hodnotu.
3. Tato velikost reprezentuje výstupní část.

Tuto slučovací vrstvu využívá i architektura Faster RCNN, která navíc od svých dvou předchůdkyň nevyužívá selektivní prohledávání k nalezení oblastí zájmu, ale místo toho zapojuje algoritmus detekce objektů a nechává síť naučit se oblasti zájmu. RPN ohodnotí ohraničení objektů, které nazýváme kotvy (anchors), a vrátí takové ohraničení, které obsahuje objekt s největší pravděpodobností. V našem nastavení architektury jsme využívali 9 kotev na pozici v obrázku. Ilustrace rozmístění kotev pro pozici (320, 320) na obrázku velikosti (600, 800) je na obrázku 3.8. Výstupem RPN je soubor regionů, návrhů, které budou následně zpracovány klasifikátorem a regresorem. Přesněji řečeno RPN predikuje pravděpodobnost, že kotva obsahuje pozadí nebo popředí a zároveň vrací predikované hodnoty posunutých souřadnic ohraničení. Na obrázku 3.9 vidíme celkové schéma architektury Faster RCNN.

Pro naši aplikaci jsme využili architekturu Faster RCNN se sítí Inception v2, podsektce 6, pro vytažení příznaků a RPN (4 konvoluční vrstvy softmax pro návrh regionů, 4 konvoluční a regresní vrstva pro predikci polohy ohraničení). Síť je předtrénovaná na COCO datasetu. Hledání minima ztrátové funkce obstarával stejně jako u RCNN stochastický gradientní sestup s momenty (3.5). Hyperparametr β pro momentový gradientní sestup jsme nastavili na hodnotu 0,9. Trénovali jsme na deseti tisíci krocích s parametrem rychlosti učení inicializovaným na hodnotu 0,01 a krokovým sestupem, který snižuje hodnotu parametru podle předpisu

$$\alpha_{i+1} = \alpha_i \frac{1}{1 + d \cdot N},$$

kde α_{i+1} je rychlost učení pro další iteraci tréninku, d je sestup, který jsme nastavili jako podíl inicializační rychlosti učení a počtu epoch. Počet epoch je podíl počtu trénovacích obrázků a velikosti batche (velikost batche je 8), tj. jedna epocha je jedeno dopředné a zpětné šíření všech trénovacích dat. Parametr



Obrázek 3.9: Schéma architektury Faster RCNN. Konvoluční síť vytáhne příznaky obrazu, ze kterých navrhne RPN oblasti zájmu. Nakonec aplikujeme RoI slučovací vrstvu a klasifikátor softmax k predikci objektu v oblasti.

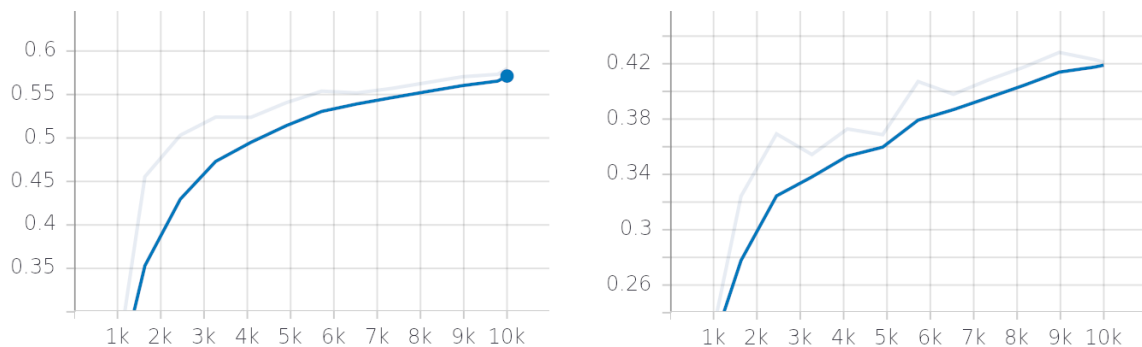
N reprezentuje počet iterací. Velikost posunutí (stride) pro konvoluční síť Inception v2 byla nastavena na šestnáct pixelů. Parametr IoU pro nemaximální supresi nakonec nastaven na hodnotu 0,6, prvotní hodnota 0,4 byla nedostatečná a dostávali jsme veliké množství false positive výsledků.

S tímto nastavením sítě jsme dosáhli hodnoty mAP 58% bez použití algoritmu detekce rukou podle barvy kůže. S tímto algoritmem jsme dosáhli výsledku 69%. Na obrázku 3.10 vidíme vývoj hodnoty mAP v průběhu dotrénování sítě. Na stejném obrázku vidíme vývoj průměrné citlivosti (average recall) přes 100 detekcí, průměrná citlivost na konci tréninku dosahuje hodnoty 0,42. Hodnota celkové ztrátové funkce na konci tréninku byla 1,52. Pro naši architekturu uvádíme navíc hodnoty ztrátových funkcí RPN, konkrétně ztráty na lokalizaci, nabývající hodnoty 0,31, a ztráty na detekci 0,23. Ztráty na klasifikátoru a RPN jsou vyobrazeny na obrázku 3.11. Celková ztráta dosahuje na konci tréninku hodnoty 1,02.

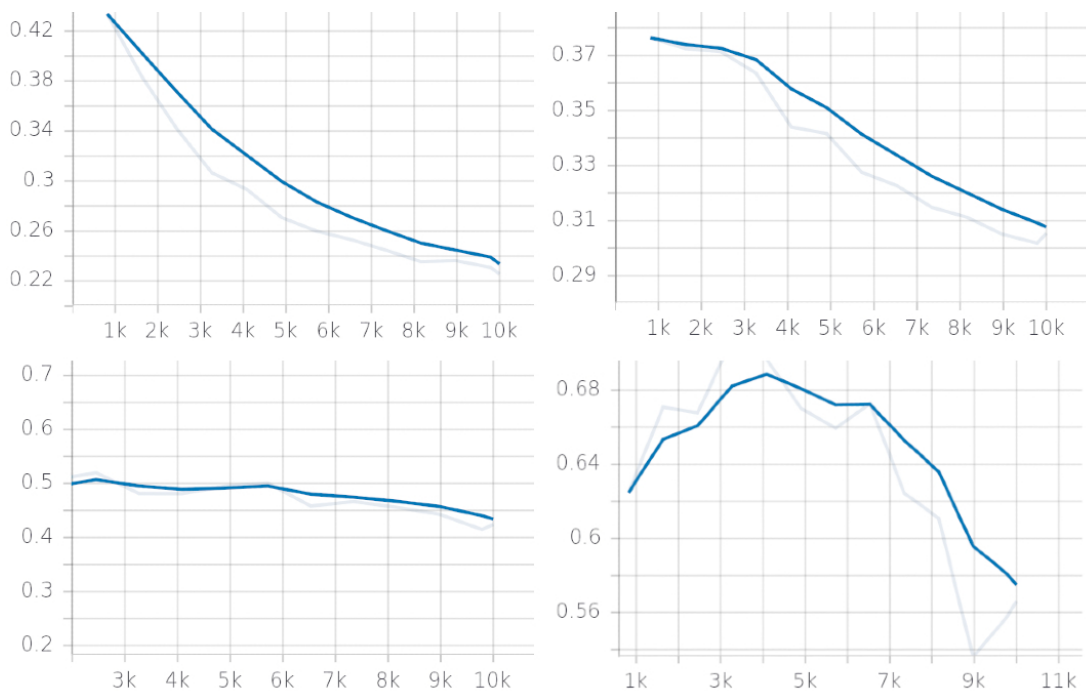
Čas tréninku byl kratší než u architektury RCNN. Průměrný čas na sto iterací byl 72s. Čas vybavování sítě klesl na 2,1s, což stále není ideální čas k použití v on-line režimu, ale při použití lepšího výpočetního zařízení bychom měli dostat nižší čas¹⁰.

Celkově můžeme prohlásit druhou architekturu, s nastavením hyperparametrů zmíněným výše, za postačující k úloze detekce rukou na trenažeru řízení, kde je kamera umístěna vždy na stejném místě, známe pozadí obrazu, které je také konstantní. Navíc víme, že se na obraze budou vyskytovat maximálně dvě ruce. Pro použití v on-line režimu se staticky umístěnou kamerou můžeme předpokládat malý posun ruky v záběru a tím optimalizovat detektor i návrh regionů.

¹⁰Důležitá je především paměť a počet jader na grafické kartě.



Obrázek 3.10: Vlevo máme graf průběhu mAP v závislosti na iteracích tréninku, kde jsme zvolili konec tréninku po deseti tisíc iteracích na hodnotě mAP rovné 0,58. Napravo od něj vidíme graf vývoje průměrné citlivosti v závislosti na iteracích. Světlejší křivka je vlastní průběh funkcí, tmavší její vyhlazení.



Obrázek 3.11: Vývoj ztrátových funkcí RPN a klasifikátoru v průběhu tréninku. Na vrchních dvou obrázcích vidíme ztrátové funkce RPN. Vlevo je objectness loss udávající jestli ohraničení obsahuje objekt nebo pozadí, vpravo vidíme lokalizační ztrátu regresoru predikujícího ohraničení. Dolní část obrázku obsahuje grafy ztrát klasifikátoru. Klasifikační ztrátu vidíme vlevo (ztráta klasifikace detekovaných objektů do třídy) a lokalizační ztrátu vpravo. Světlejší křivka je vlastní průběh funkcí, tmavší její vyhlazení.

Kapitola 4

Odhad 2D pozice ruky

Hluboké učení je rychle se vyvíjející odvětví, které v posledních letech našlo uplatnění i v úloze odhadu polohy ruky v prostoru. Stejně jako u detekce ruky v obrazu budeme trénovat hluboké neuronové sítě učit se jejich parametry z předem označených dat. Úloha je založena na odhadu bodů reprezentujících klouby ruky, které jsou dostatečnou reprezentací struktury a polohy ruky v prostoru.

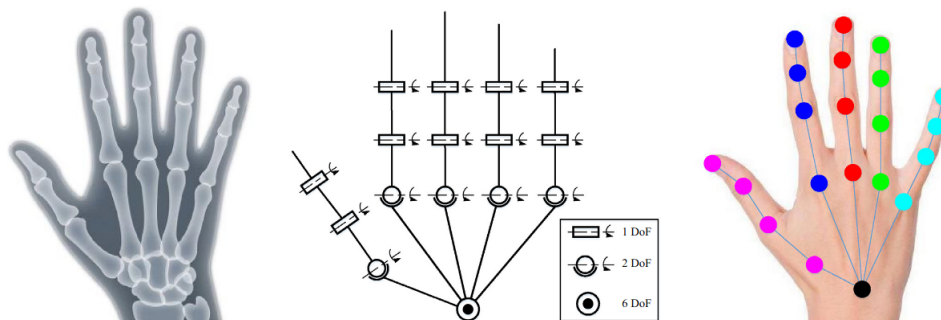
V úvodu této kapitoly představíme používanou reprezentaci ruky, která je vhodná pro formulaci modelu. Následně popíšeme dostupné a použité datasey a možnosti vytvoření syntetických příkladů, které mohou zvýšit robustnost dat a tím pomoci k lepším predikcím modelů. V hlavní části představíme a otestujeme modely využívající hlubokého učení k predikci polohy ruky ve 2D z jediného RGB snímku. Navážeme kapitolou o odhadu 3D pozice ruky z hloubkové mapy a z RGB obrazu. Odhad prostorové pozice ruky z jediného RGB snímku je velmi aktivním a složitým problémem, hlavně kvůli hloubkové nejednoznačnosti a nedostupnosti anotovaných dat. Většina uživatelů samozřejmě nedisponuje kamerou, které je schopna zachytit vedle RGB jeho obrazu hloubkovou mapu. Jedná se tedy o atraktivní téma a možnou širokou škálou využití.

Lidská ruka je komplexní částí těla, což činí složitějším modelování její dynamických a kinematických vlastností, zvláště potom v on-line režimu. Základní reprezentací ruky je kinematický model její kostry. Jednoduchost modelu spočívá v ignorování deformací měkké tkáně ruky.

Lidská ruka se skládá z 27 kostí, které tvoří zápěstí, dlaň a prsty. Klouby mezi prvními třemi články prstů mají jen jeden stupeň volnosti (DF z anglického „degree of freedom“). Klouby napojující prsty na dlaň mají 2DF a zápěstí celkem 6DF. Celkem můžeme ruku popsat modelem s 26DF. Problém odhadu pozice ruky potom převádíme na problém pozice bodů reprezentujících jednotlivé klouby. Model s 21 body, který nazveme 21-kloubový model, patří mezi nejčastěji využívané [66]. Kostra lidské ruky a její kinematický model je znázorněn na obrázku 4.1.

Nyní víme jaký model budeme používat v našich experimentech, zásadním můstkem je představení datasetů, které se používají pro trénování modelů s učitelem. Datasety obsahují příklady, které jsou buď syntetizovány pomocí počítačové grafiky jako v [67] nebo získány z kamery zachycující hloubkovou mapu a následnou anotací. K vytváření anotovaných dat se používá datových rukavic, magnetických senzorů, které jsou náročné na kalibraci a navíc nákladné. Dále se používá metoda, kdy jsou různé části ruky nebo rukavice obarvené různými barvami. Datasety se neliší jen ve specifickém modelu odhadu, ale i ve specifickém kinematickém modelu ruky tj. různé datasety využívají jiný počet kloubů. Pro naše modely jsme vybrali datasety obsahující RGB-D nebo RGB snímky s 21 anotovanými body na ruce 4.1. Celkem jsme měli k dispozici necelých 21 tisíc anotovaných 2D snímků a cca. 2,3M anotovaných 3D snímků.

Navíc jsme anotovali vlastní data z trenažéru. Pro odhad 2D pozice klíčových bodů jsme anotovali



Obrázek 4.1: Ukázka kinematického modelu ruky. Zleva: rentgenový snímek ruky, kinematický model s 26 stupni volnosti a reprezentace ruky pomocí 21-kloubového modelu.

Dataset	RGB/D	Objekty	Anotace	Subjektů	#Snímků
Stereo Hand Pose Tracking	RGB-D	Ano	2D+3D	24	18000
BigHand2.2M	RGB-D	Ne	3D	10	2,2M
MSRA15	D	Ne	3D	9	76375
Dexter	RGB-D	Ano	2D+3D	2	3129
LSM-HPD	RGB	Ne	2D	9	20 500

Tabulka 4.1: Tabulka datasetů využitých v práci. Vedle jména datasetu a obrazového typu máme informaci o prezenci objektu v ruce a počtu různých subjektů, jejichž ruce jsou anotovány.

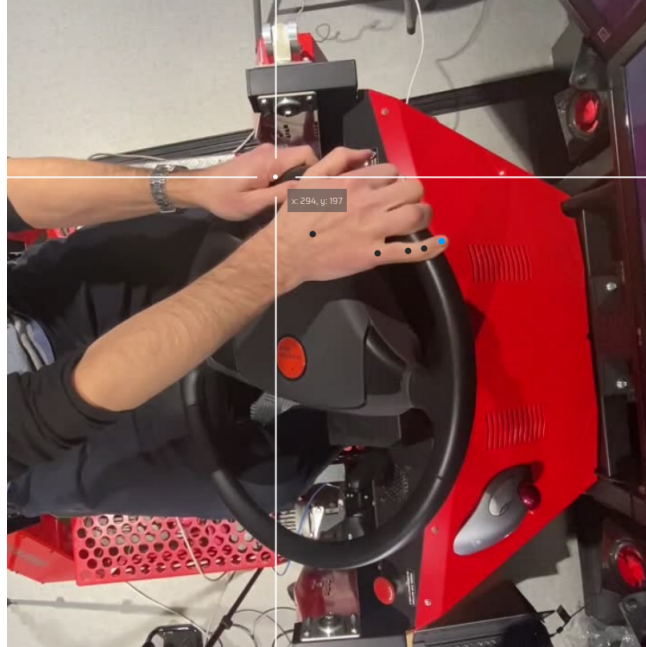
100 snímků. K anotaci jsme použili online nástroj *makesense*¹. Tento software umožní uživateli nahrát obrázky, označit na obrázku klíčové body a přiřadit jim označení. Výstupem je *csv* soubor, který obsahuje souřadnice jednotlivých označení klíčových bodů v obrazu. Ukázka použití softwaru *makesense* je k vidění na obrázku 4.2.

Pro anotaci 3D klíčových bodů jsme implementovali vlastní anotační nástroj v jazyku Python v3.7 a frameworku *open3d v0.12.0*. Nástroj umožní uživateli zobrazit hloubkovou mapu jako oblak bodů v interaktivním prostředí. V prvním kroku uživatel vybere 3D ohraničení ruky a potvrdí uložení souřadnic osmi rohů ohraničení do textového souboru. Následuje druhý krok, kde uživatel označí 21 klíčových bodů na ruce. Interaktivní prostředí umožní uživateli kontrolu umístění bodů z různých úhlů pohledu. Práci v nástroji ilustruje obrázek 4.3.

Úlohu odhadu pozice ruky můžeme rozdělit podle dvou metod, které jsou při jejím řešení používány. Jedná se o metody založené na detekci a metody založené na regresi. Při řešení pomocí první z nich vrací model pravděpodobnostní rozdělení pro každý odhadovaný bod/kloub. Pro 21-kloubový model dostáváme 21 různých pravděpodobnostních rozdělení, resp. teplotních map², pro každý obrázek. Vlastní pozice kloubu je následně odvozená jako argument maxima na příslušné teplotní mapě. Oproti tomu regresní metoda provádí přímý odhad pozice každého kloubu. Pro 21-kloubový model máme 21×3 neuronů ve výstupní vrstvě architektury, které predikují prostorové souřadnice bodu. Regresní metody jsou náročnější na trénink modelu a velikost datasetu. V následujících sekcích představíme modely využívající právě popsané metody k odhadu pozice ruky jak z RGB-D, tak RGB obrazu.

¹<https://www.makesense.ai/>

²Teplotní mapa, angl. heatmap, je typ vizualizace dat, kde je každá hodnota reprezentována určitou barvou ze spojitého barevného spektra.



Obrázek 4.2: Průběh anotace v softwaru *makesense*. Klíčové body jsou označené černým, resp. modrým, bodem.



Obrázek 4.3: Práce v námi implementovaném softwaru pro anotaci 3D klíčových bodů. Barevné koule označují klíčové body. Na obrázku je jedna ruka ze tří různých úhlů pohledu.



Obrázek 4.4: Schéma architektury odhadující 2D klíčové body. Na vstupu dostaneme obrázek, na kterém je pomocí Faster RCNN detekovaná ruka. Ruku ořízneme a pomocí CPM dostaneme teplotní mapy polohy klíčových bodů.

4.1 Odhad pozice ruky ve 2D

Po představení kinematického modelu ruky a dostupných datových sad přistupujeme přímo k problematice odhadu souřadnic jednotlivých klíčových bodů ruky. Cílem je získat 21 souřadnic (x_i, y_i) , určujících pozice kloubů ruky. Při řešení této úlohy využijeme natrénovanou architekturu RCNN k detekci ruky a nad ní vystavíme model, který na detekované ruce najde pozice klíčových bodů.

Algoritmus můžeme rozdělit do dvou částí. V první části detekujeme ruku pomocí Faster RCNN a vyřízneme ji z obrazu. Ve druhé části zapojíme model detekující klíčové body. Lokalizace ruky pomáhá odstraněním šumu v pozadí a tím činí učení se detektoru klíčových bodů jednodušším.

První část je podrobně popsána v sekci 3.2.2.3. Věnujme se tedy pouze části architektury, která bere jako vstup oříznutý obrázek vrácený ze sítě Faster RCNN. Na vstupu dostane RCNN obrázek velikosti 720×720 a vrátí souřadnice rohů ohraničení vztažené k původní velikosti. Pro odhad pozice klíčových bodů využijeme postupu navrženém v [69]. Predikujeme 2D teplotní mapy udávající pravděpodobnost umístění klíčového bodu na snímku, čímž redukuje. Označme $k_i = (x_i, y_i)$ souřadnice klíčového bodu i , $i \in \hat{21}$. Dále označíme $\mathbf{k} = (k_1, \dots, k_{21})$. Vstup normalizujeme na velikost 256×256 . Upravená architektura CPM (Convolutional Pose Machine, [69]), pracuje v pěti etapách. V každé etapě používáme jinou velikost receptivního pole. Postupně měníme velikost receptivního pole z původní velikosti 8×8 px až k velikosti 240×240 .

První etapa vrací pravděpodobnost, že určitý pixel obsahuje klíčový bod. Následující etapy upravují odhad z první etapy pomocí informace z okolí závislého na velikosti receptivního pole. Výstupem jsem teplotní mapy pravděpodobnosti polohy každého kloubu. Celkovou stavbu architektury mapuje obrázek 4.4.

V aplikaci jsme použili již natrénovanou síť Faster RCNN a síť CPM jsme dotrénovali na 10000 obrázcích z datasetu LSM-HPD, tabulka 4.1. Trénovali jsme ve 30 tisících iteracích s rychlostí učení $\alpha = 0,0004$. Křivka totální ztráty se stabilizovala na hodnotě 1,32 a hodnota mAP, na námi anotovaných obrázcích nepodílejících se na učení, byla 0,92. Na testovacích datech z datasetu LSM-HPD bylo $mAP = 0,89$. Jako ztrátovou funkci jsme zvolili vzdálenost predikovaného bodu od anotovaného bodu, tj.:

$$C = \sum_{i \in \hat{21}} \|\hat{k} - k\|_2^2, \quad (4.1)$$

kde \hat{k} jsou predikované souřadnice kloubu a k jsou souřadnice anotované. Ztrátovou funkci minimalizujeme pomocí stochastického gradientního sestupu s velikostí minibatche nastavenou na 8.

Čas vybavování sítě je průměrně 2,56s. Většinu z tohoto času zabere vlastní detekce ruky pomocí Faster RCNN. Pokud vynecháme detekující část, čímž vypustíme použití Faster RCNN, dostáváme se na čas 0,29s. Čas detekce klíčových bodů z oříznutého obrázku je téměř dostatečný pro použití v živém přenosu.



Obrázek 4.5: Schéma finální architektury sloužící k odhadu 2D klíčových bodů z RGB obrazu. Z původního obrázku je segmentována ruka, kterou vyřízneme a pošleme jako vstup do CPM, která vrátí teplotní mapy pozice kloubů.

Možným vylepšením do budoucna je nahrazení sítě Faster RCNN jinou, rychlejší metodou. Inspirovali jsme se článkem [67] a rozhodli jsme se k využití HandSegNet k segmentaci ruky z obrazu. Segmentace ruky nahradí detekci ruky v obraze, pomocí které určujeme vyříznutí obrazu, který slouží jako vstup CPM. Je vhodné poznamenat, že HandSegNet je derivátem CPM. Architektura HandSegNet má na vstupu RGB obraz dimenze $256 \times 256 \times 3$, ten je zpracovaný pomocí čtyř bloků sestávajících se z konvolučních vrstev s aktivační funkcí ReLu a maximovým poolingem. Výstupem je segmentovaná maska ruky dimenze $256 \times 256 \times 1$. Velikost batche byla nastavená na hodnotu osm snímků a extrémní ztrátové funkce byly hledány pomocí Adam řešiče, který je představený v sekci 6. Inicializace vah byla ponechána na hodnotách z [69]. Parametr rychlosti učení α byl nastaven pro prvních dvacet tisíc iterací nastaven na hodnotu $1 \cdot 10^{-5}$, následně byl snížen na $1 \cdot 10^{-6}$ pro následujících deset tisíc iterací a posledních deset tisíc iterací probíhalo s rychlostí učení $1 \cdot 10^{-7}$. Ztrátová funkce byla ponechána na klasické ztrátové funkci křížové entropie, které je věnována sekce 6.

Implementovanou síť provádějící segmentaci jsme trénovali na 3500 snímcích z datasetu EgoHands [55], 400 snímků jsme použili pro validaci a 800 pro následné testování. Trénink probíhal v 10 tisíci iterací. Průměr IoU na testovacích datech dosahovalo hodnoty 81,2%, kde IoU je bráno jako poměr průniku segmentované oblasti s anotovanou oblastí a jejich sjednocení. Oblasti jsou definované po pixelech. Průměrný čas vybavování sítě HandSegNet je 0,21s.

Výřez obrázku, který je posílán do CPM k detekci klíčových bodů, získáme výřezem nejmenší plochy obsahující celou segmentovanou část. Konečné schéma architektury pro 2D detekci klíčových bodů ruky vidíme na obrázku 4.5.

Zpracování obrázku celou architekturou trvá v průměru 0,67s. Při ztrátové funkci (4.1) a použití námi anotovaných obrázků jsme dosáhli hodnoty $mAP = 0,87$. Dostáváme tedy nižší přesnost odhadu než při použití sítě Faster RCNN za cenu snížení času vybavování sítě. Pro praktickou aplikaci prohlašujeme druhou architekturu za vhodnější. Pět procentní ztráta na přesnosti odhadu bodů je vyvážena téměř 75% zrychlením. Zároveň je nutné si uvědomit, že detekce ruky i samotných klíčových bodů může být optimalizována pro použití na video. Vzdálenost ruky a jejích klíčových bodů se mezi snímky liší jen velmi málo, tím získáváme určité omezení, které činí detekci jednodušší.

Kapitola 5

Odhad prostorové pozice ruky

Poslední kapitola pojednává o odhadu pozice ruky v prostoru. Jedná se o nadstavbu minulé kapitoly, kde jsme odhadovali pouze 2D souřadnice pozice kloubů v obraze. Kapitolu jsme rozdělili na dvě části, kde první část věnujeme odhadu pozice ruky z hloubkové mapy, druhou část odhadu pozice ruky přímo z RGB snímku.

5.1 Odhad pozice ruky z RGB-D obrazu

Odhad pozice ruky z RGB-D obrazu tj. RGB obrazu s hloubkovou mapou. Hloubková mapa, také označovaná jako hloubkový obraz, je kanál obrazu, ve kterém každý pixel nese hodnotu udávající vzdálenost daného pixelu obrazu od senzoru. Důležité je si uvědomit, že obraz, který vrací senzor zachycující RGB-D (např. Kinect) není 3D, ale 2.5D. Máme 2D obraz a hloubkovou informaci o každém pixelu. Nemáme tedy přímou informaci o částech, které jsou v zákrytu ve směru paprsku senzoru. Data, která máme k dispozici nejsou vhodná ke zpracování pomocí 2D ani 3D konvoluce.

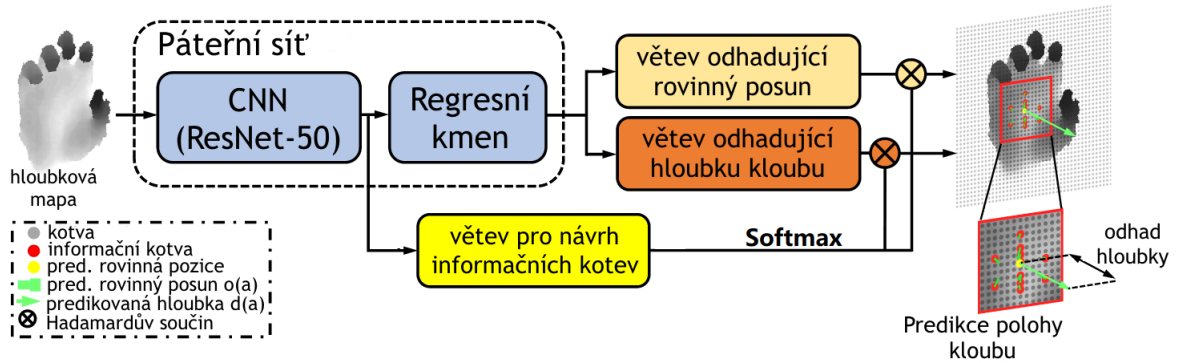
V této sekci představíme přístup, který je založený na regresi. Regresní model je založený na návrhu kotev ke kloubu a následné regresi [68]. Tento model byl vytvořený v roce 2019 a díky své skladbě se vyhýbá nutnosti použití, výpočetně náročné, 3D konvoluce. Autoři tohoto modelu ve své práci nezmiňují jakým způsobem upravují vstupní obrazy, pro vlastní odhad pozice kloubů a počítají s normalizovaným vstupem. Normalizovaným vstupem je myšleno, že obraz má ruku ve svém středu a ruka je nejbližším objektem k senzoru. Architektura nese jméno A2J (z anglického „anchor to joint“) a vedle páteřní neuronové sítě ji doplňují tři větve.

Na vstupním obrazu jsou nejprve rozmístěny kotvy se stridem čtyři, tj. vzdálenost mezi kotvami je 4 pixely, které slouží jako regresory k predikci pozice kloubu v prostoru. Pozici kotvy označíme $s(a)$.

Samotná páteřní síť využívá ResNet-50, kde výstup z prvních tří vrstev slouží jako vstup pro síť predikující váhy a navazuje na regresní kmen jehož výstup slouží jako vstup pro větve predikující rovinný posun a hloubku. Mezi úpravy architektury ResNet-50 patří změna stridu ve čtvrté vrstvě na jeden pixel, který zapříčiňuje 16krát podvzorkování na výstupu. Původní architektura ResNet-50 má 32krát podvzorkování, dosahujeme tedy větší citlivosti.

Následuje zpracování prvními dvěma větvemi, které predikují rovinný posun $\mathbf{o}_j(a)$ kloubu j od kotvy a a hloubku $d_j(a)$ kloubu j od kotvy a respektive. Obě větve navazují na výstup regresního kmene páteřní sítě. Hloubkový obraz nejdříve šestnáctkrát podvzorkujeme a rozmístíme kotvy se stridem 4, potom každý bod příznakové mapy reprezentuje 16 kotev. Obě větve obsahují 4 konvoluční vrstvy s 256 kanály a výstupní konvoluční vrstvu velikosti $16 \times K \times 2$ resp. $16 \times K \times 1$, kde K je počet kloubů.

Z páteřní sítě vystupuje, po zpracování CNN ResNet-50, větev, která objevuje *informační kotvy*, pro určitý kloub pomocí přiřazení váhy kotvě. Informační kotvou nazveme kotvu, která splňuje $\tilde{p}_j(a) > 0.02$,



Obrázek 5.1: Schéma architektury A2J [68]. Vedle pátevní sítě skládající se z CNN a regresního kmene obsahuje dvě větve pro odhady rovinného posunu a hloubku a jednu pro návrh informačních kotev.

kde

$$\tilde{p}_j(a) = \frac{\exp(\mathbf{p}_j(a))}{\sum_{a \in A} \exp(\mathbf{p}_j(a))},$$

kde A je množina všech kotev obrazu a $\mathbf{p}_j(a)$ je predikovaná váha kotvy a ke kloubu j . Tato větev má na vstupu příznakovou mapu z pátevní sítě. Stejně jako u dvou předchozích sítí obsahuje tato síť 4 konvoluční vrstvy s 256 kanály a výstupní vrstva má velikost $16 \times K \times 1$. Celkové schéma architektury je ilustrováno na obrázku (5.1).

Na následujících řádcích stručně popíšeme trénink právě popsané architektury. Pro vytvoření vhodného vstupu sítě, mapujeme každý pixel hloubkové mapy do 3D prostoru a následně vyřízneme krychli, která obsahuje analyzovanou ruku. Pro kloub j máme dvě hodnoty: t_j^p skutečné pozice kloubu j v rovině a t_j^d je transformovaná hodnota skutečné hloubky z_j kloubu j podle předpisu

$$\mathbf{t}_j^d = z_j - \theta,$$

kde θ je transformační parametr, který má hodnotu hloubky středového bodu.

Samotný trénink probíhá pod dvěma ztrátovými funkcemi. První z nich měří ztrátu vzhledem k odhadu pozice kloubu jako

$$L_1 = \frac{1}{2} \sum_{j \in J} C_{\tau_1} \left(\sum_{a \in A} \tilde{p}_j(a) (s(a) + \mathbf{o}_j(a)) - \mathbf{t}_j^p \right) + \sum_{j \in J} C_{\tau_2} \left(\sum_{a \in A} \tilde{p}_j(a) d_j(a) - \mathbf{t}_j^d \right),$$

kde J je množina kloubů (v našem případě $|J| = 21$), $s(a)$ je rovinná pozice kotvy a , $C_\tau(\cdot)$ je vyhlazení L1-ztráty definované jako

$$L_\tau(x) = \begin{cases} \frac{1}{2\tau} x^2, & \text{pro } |x| < \tau \\ |x| - \frac{\tau}{2}, & \text{jinde.} \end{cases}$$

Vyhlazovací parametr τ byl v původním článku nastavený jako $\tau_1 = 1$ a $\tau_2 = 3$, pro naši implementaci jsme zvolili $\tau_2 = 5$, kvůli velkému šumu na hloubkové mapě. Druhou ztrátovou funkcí je funkce měřící ztrátu na okolí informativních kotev, definovanou jako

$$L_2 = \sum_{j \in J} C_{\tau_1} \left(\sum_{a \in A} \tilde{p}_j(a) s(a) - \mathbf{t}_j^p \right).$$

Tato funkce pomáhá vybírat informativní kotvy kolem kloubu.

Celkovou ztrátu definujeme jako

$$C = \omega L_1 + L_2,$$

kde hyperparametr ω udává důležitost jednotlivých složek ztráty. V naší implementaci jsme použili $\omega = 3$ stejně jako autoři původního článku.

Trénovali jsme v 50 tisících krocích na náhodně vybraných 295 510 obrázcích. Vstupní obraz je transformován na velikost 176×176 . Trénovali jsme v patnácti epochách. Rychlost učení byla inicializována na hodnotu 4×10^{-4} a každou epochu ji snížíme o 10^{-4} . Na konci tréninku dosahovala ztrátová funkce hodnoty 11, 2.

Implementaci architektury jsme otestovali na námi anotovaných datech, které jsme získali pomocí kamery *Kinect for Windows v1*. Vybraných 50 snímků jsme anotovali pomocí nástroje, který jsme implementovali a představujeme jej na začátku kapitoly 4. Přesnost jsme měřili pomocí průměrné prostorové vzdálenosti predikce od anotace (v milimetrech) a poměru úspěšnosti. Poměr úspěšnosti je definovaný jako poměr správně odhadnutých snímků a všech snímků. Správně odhadnutý snímek je takový snímek, který má maximální vzdálenost predikce od anotace přes všechny klíčové body menší než 10mm. Průměr prostorové vzdálenosti predikce od anotace je roven 9,35mm. Poměr úspěšnosti je 0,74 tj. 74%. Vybavování sítě trvá v průměru 1,37s. Tento čas je řádově vyšší než čas, který uvádějí autoři metody. To je způsobeno během programu na slabším systému a jiné implementaci. Příklady aplikace na reálných datech jsou k nalezení v poslední kapitole na straně 54.

5.2 Odhad pozice ruky z RGB obrazu

Úloha odhadu pozice ruky z RGB obrazu je složitější úlohou než odhad z RGB-D obrazu, kvůli přítomnosti mnoha nejednoznačností plynoucích z absence hloubkové informace. Dalším problémem je nedostatek trénovacích dat. K tréninku neuronové sítě je potřeba velké množství s anotovanými 3D klíčovými body (klouby), proto se při řešení této úlohy přistupuje k syntéze datasetů a jejich augmentaci pro zvýšení mohutnosti množiny trénovacích dat.

V této práci představíme první návrh řešení úlohy odhadu pozice ruky z RGB obrazu z roku 2017 [67]. Předchozí přístupy využívali hloubkovou informaci alespoň v nějaké části vývoje. Zmíněná práce přistupuje k nejednoznačnosti v měřítku přechodem k tréninku a odhadu škálově invariantních souřadnic, definovaných pro kloub j jako

$$\mathbf{w}_j = \frac{z_j}{\|z_{k+1} - z_k\|_2},$$

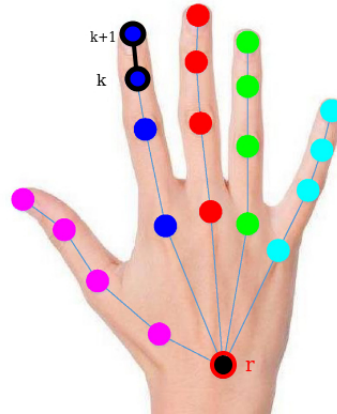
kde z_j je skutečná pozice kloubu a ve jmenovateli je normalizační konstanta závislá na snímku. Volíme k tak, aby konstanta byla rovna jedné pro první kost ukazováčku, obrázek (5.2). Dále použijeme souřadnice kořenového bodu z_r , obrázek (5.2), k odvození translačně invariantních souřadnic

$$\mathbf{w}_j^{transl} = \mathbf{w}_j - \mathbf{w}_r.$$

Tímto jsme převedli úlohu odhadu bodů z_j , $j \in \hat{21}$ na odhad $\mathbf{w}_j^{transl} = (x_j, y_j, z_j)$, $j \in \hat{21}$ z jediného vstupního RGB obrazu.

Celá architektura se skládá z neuronové sítě HandSegNet, která provádí segmentaci ruky ze snímku, ze sítě PoseNet odhadující pravděpodobnost umístění kloubu na obraze a sítě PosePrior predikující normalizované a invariantní prostorové souřadnice na pravděpodobnostních rozděleních. Schéma sítě ilustruje obrázek (5.3).

Segmentační síť HandSegNet, je upravenou, zredukovanou variantou části sítě Convolutional pose machines [69], která byla trénovaná na syntetickém datasetu vytvořeném autory. Tuto síť využíváme i



Obrázek 5.2: Klouby k a $k+1$, zvýrazněné černým obrysem, používáme jako normalizační konstantu pro výpočet škálově invariantních souřadnic odhadovaných bodů. Kloub r s červeným obrysem používáme jako referenční bod pro translačně invariantní souřadnice.

v architektuře odhadující 2D pozici ruky. Díky segmentaci ruky, která je na obrázcích v různých velikostech, můžeme segmentovanou ruku oříznout z obrazu a tento oříznutý obraz použít jako vstup pro navazující síť odhadující pravděpodobnost výskytu kloubu.

Lokalizace 2D klíčových bodů je zde formulovaná jako odhad 2D teplotních map $\mathbf{m} = \{\mathbf{m}_1(x, y), \dots, \mathbf{m}_{21}(x, y)\}$, reprezentujících pravděpodobností rozdělení umístění kloubu v obraze. V této části je zapojena upravená síť Pose Network [69], kde v první části extrahujeme příznakové mapy augmentovaných verzí analyzovaného snímku, které jsou následně spojeny do jediné, ze které je odhadována teplotní mapa.

Poslední krok je odbaven pomocí PosePrior sítě, které predikuje invariantní 3D souřadnice \mathbf{w}_j^{transl} na potenciálně nekompletní a zašuměné teplotní mapě $\mathbf{m}_j(x, y)$. Tato síť se učí varietu¹ možných artikulací ruky z teplotních map a vrací nejpravděpodobnější 3D souřadnice z 2D informace. K této problematice je přistoupeno přechodem do kanonické báze, která zaručuje invarianci v globální orientaci ruky. Přechod do této báze je dán rovnicí

$$\mathbf{w}^c = \mathcal{R}(\mathbf{w}^{transl}) \cdot \mathbf{w}^{transl},$$

kde $\mathcal{R}(\cdot) \in \mathbb{R}^{3 \times 3}$ je 3D rotační matice, kterou dostaneme jako součin rotace kolem osy x a z , \mathcal{R}_{xz} a rotace kolem osy y , \mathcal{R}_y . Rotaci \mathcal{R}_{xz} hledáme tak, že určitý klíčový bod \mathbf{w}_a^c leží na ose y kanonické báze, tj:

$$\mathcal{R}_{xz} \cdot \mathbf{w}_a^c = \lambda \cdot (0, 1, 0)^T, \quad \lambda \geq 0.$$

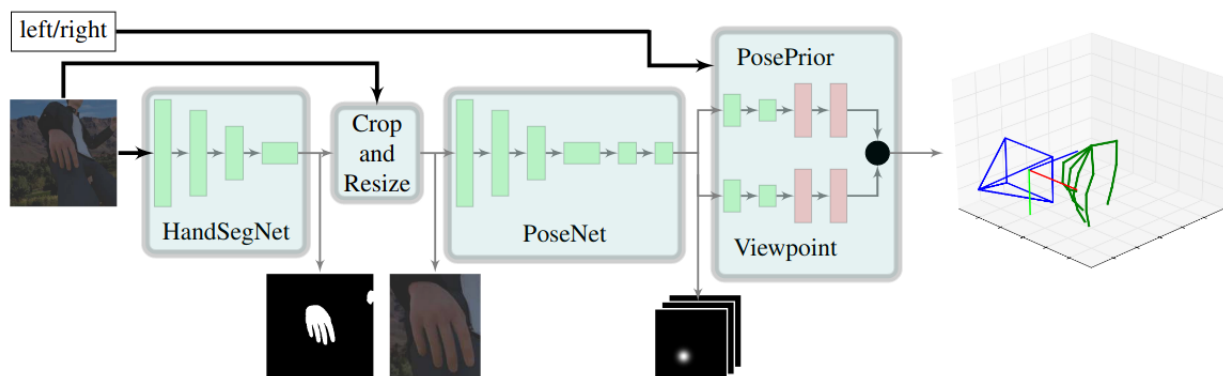
A následně nalezneme rotaci \mathcal{R}_y , aby platilo

$$\mathcal{R}_y \cdot \mathcal{R}_{xz} \cdot \mathbf{w}_a^c = (\alpha, \beta, 0), \quad \alpha \geq 0.$$

K dosažení symetrie mezi pravou a levou rukou, překlopíme pravé ruce podle osy z v kanonických souřadnicích. Konečně můžeme vyjádřit souřadnice kloubů rukou jako

$$\mathbf{w}_j^c = \begin{cases} \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix}^T, & \text{pro levou ruku,} \\ \begin{pmatrix} x_j^c \\ y_j^c \\ -z_j^c \end{pmatrix}^T, & \text{pro pravou ruku.} \end{cases}$$

¹Varieta je topologický prostor, lokálně podobný obecně n -rozměrnému Euklidovskému prostoru.



Obrázek 5.3: Schéma architektury převzaté z [67]. V první části probíhá lokalizace/segmentace ruky pomocí sítě HandSegNet. Výsledná maska ruky je oříznuta a poslána jako vstup do sítě PoseNet. CNN PoseNet lokalizuje 21 kloubů ruky a teplotní mapy korespondující každému kloubu. V poslední části jsou teplotní mapy zpracovány dvěma větvemi, které odhadují pozici snímáče (viewpoint) resp. 3D pozici ruky.

Trénování sítě probíhá v těchto kanonických souřadnicích, dostáváme tedy 3D souřadnice v kanonických souřadnicích a odděleně odhadujeme rotační matici, která je parametrizována pomocí tří parametrů. Nad odhadem rotační matice můžeme přemýšlet jako nad odhadem pozice pozorovatele resp. senzoru, kamery vzhledem k ruce.

Pro naši aplikaci používáme předtrénovaný model, který byl vydán autory [67]. Architektura HandSegNet má na vstupu RGB obraz dimenze $256 \times 256 \times 3$, ten je zpracovaný pomocí čtyř bloků sestávajících se z konvolučních vrstev s aktivační funkcí ReLu a maximovým poolingem. Výstupem je segmentovaná maska ruky dimenze $256 \times 256 \times 1$. Velikost batche byla nastavená na hodnotu osm snímků a extrémní ztrátové funkce byly hledány pomocí Adam řešiče, který je představený v podsekcí 6. Inicializace vah byla ponechána na hodnotách z [69]. Parametr rychlosti učení α byl nastaven pro prvních dvacet tisíc iterací nastaven na hodnotu $1 \cdot 10^{-5}$, následně byl snížen na $1 \cdot 10^{-6}$ pro následujících deset tisíc iterací a posledních deset tisíc iterací probíhalo s rychlostí učení $1 \cdot 10^{-7}$. Ztrátová funkce byla ponechána na klasické ztrátě křížové entropie, kterou popisujeme v podsekcí 6.

Následující architektura PoseNet má váhy inicializované podle [69]. Velikost batche a volba řešiče byla ponechána stejná jako u sítě HandSegNet. Trénink probíhal ve 30 tisíc iteracích, kde se parametr rychlosti učení snižoval po každých deseti tisíc iteracích po hodnotách $1 \cdot 10^{-4}$, $1 \cdot 10^{-5}$, $1 \cdot 10^{-6}$. Jako ztrátová funkce byla zvolena L2 ztráta. Tato architektura vrací výstup o dimenzi $32 \times 32 \times 21$, tj. máme 21 teplotních map o velikosti 32×32 .

Poslední část architektury, která je nazvána PosePrior má dva paralelní kanály, které mají téměř stejnou architekturu². Nejdříve zpracují 21 teplotních map, normalizovaných na hodnoty z intervalu $\langle 0, 1 \rangle$, pomocí šesti konvolučních vrstev s ReLu aktivačními funkcemi. Informace o pravolevé orientaci ruky je spojena s příznakovou mapou a zpracována dvěma FCN s dropoutem 0,2, tzn. v trénovací epoše vynecháme daný neuron s pravděpodobností 0,2. Oba kanály končí jednou FCN s lineární aktivační funkcí, která vrací odhad rotační matice \mathcal{R} a kanonické souřadnice w^c . Spojením těchto dvou odhadů dostáváme odhad w^{transl} . Tato architektura používá ztrátovou funkci, která je součtem kvadrátu

²Detail architektury je k nalezení v doplňujících materiálech článku [67]. Zároveň jsou tam k nalezení schémata dvou probraných sítí.

L2 ztráty na kanonických souřadnicích

$$C_s = \left\| \mathbf{z}^c - \mathbf{w}_{pred}^c \right\|_2^2,$$

kde \mathbf{z}^c je kanonická souřadnice skutečná polohy kloubu a \mathbf{w}_{pred}^c je predikovaná kanonická souřadnice kloubu, a kvadrátu L2 ztráty na matici kanonické transformace

$$C_r = \left\| \mathcal{R}_{pred} - \mathcal{R} \right\|_2^2,$$

kde \mathcal{R} je skutečná matice rotace a \mathcal{R}_{pred} je predikovaná matice. Celkově dostáváme ztrátu

$$C = C_s + C_r,$$

jejíž minimalizace byla řešena pomocí řešiče Adam. Poslední FCN vrací v jednom kanálu 3 parametry rotace \mathcal{R} a ve druhém 63 hodnot reprezentujících 3 souřadnice pro každý kloub ruky.

Pro odhad pozice ruky z RGB obrazu jsou zapotřebí data, která mají jak notace ve 2D, tak ve 3D. Několik existujících datasetů, které jsou vhodné pro trénování sítě řešící tuto úlohu, tabulka (4.1), představují autoři metodu k vytvoření syntetického datasetu. Dostupné datasety neobsahují dostatečně rozmanitá data, nemají kompletní anotaci aj., proto autoři sestavili vlastní syntetický dataset ve volně dostupném softwaru Blender³.

Tento dataset obsahuje celkem 20 postav, které vykonávají 39 různých akcí. Na každá scéně je kamera umístěna náhodně na sféře kolem ruky postavy. Pravá a levá ruka je zastoupená na snímcích se stejnou frekvencí. Pro určité nastavení scény a kamery bylo náhodně vybráno pozadí z množiny 1231 různých obrázků pozadí. Finální dataset obsahuje 41258 snímků v tréninkovém datasetu a 2728 snímků ve validačním datasetu. Snímky mají rozlišení 320×320 pixelů. Každá ruka na snímku obsahuje 21 anotovaných kloubů, navíc nabízí 33 anotovaných segmentačních masek a pozadí. O každém kloubu máme navíc informaci o tom zda-li je viditelný nebo v zákrytu.

Jelikož se jedná o první práci, zabývající se metodou odhadu prostorové pozice ruky z RGB obrazu nebylo možné přímo porovnat navrženou architekturu oproti jiným. Autoři se proto rozhodli pro srovnání jednotlivých částí architektury. Model PosePrior dosahuje lepších výsledků než dva existující modely [70, 71], které byly zahrnuty v porovnání. Použitá metrika byla založena na správném určení polohy kloubu, kde za správné určení uvažujeme takovou polohu, která má euklidovskou vzdálenost od skutečné polohy kloubu menší než určitá mez. PosePrior překonává ostatní modely na všech testovaných mezích (20mm, 30mm, 40mm, 50mm). Výkon architektury je nejvíce limitován nedostatkem trénovacích dat s reálnými scénériemi a větší rozmanitostí snímků.

Testování proběhlo na námi anotovaných datech, které jsme získali nahrání kamerou *Kinect for Windows v1*. Vedle anotace ve 3D, kterou jsme použili i pro předchozí metodu, jsme anotovali korespondující snímky ve 2D. Jako míru přesnosti jsme zvolili průměrnou chybu prostorové vzdálenosti a poměr úspěšnosti. Hodnoty metrik jsou 6,28mm, resp. 0,94 pro mez nastavenou na 10mm. Vybavování sítě na snímek je průměrně 2,07s. Autoři metody ve svém článku [67] neuvádějí rychlost jejich implementace.

Metoda navržená k odhadu 3D pozice ruky z hloubkové mapy dosahuje menší přesnosti, na našich datech, než právě představená metoda, ale vyšší rychlosti zpracování obrazu. Metoda odhadu z RGB obrazu je robustnější na pozici ruky vůči senzoru. Pro naši aplikaci na snímky z trenažéru řízení automobilu jsme vyhodnotili odhad přímo z RGB obrazu jako lepší, nejen kvůli vysoké přesnosti, ale také kvůli hardwarové nenáročnosti na senzor.

³<https://www.blender.org>

Kapitola 6

Aplikace na data ze simulátoru řízení kamionu

Jedním z cílů práce bylo aplikovat prostudované metody k aplikaci na data ze simulátoru řízení od firmy SCS Software¹, obrázek 6.1. Konkrétně odhadovat pozici rukou řidiče obsluhujícího trenažér. Detekce rukou je užitečná ke sledování pohybu celé ruky a její aktivity. Známe-li lokaci ruky můžeme určovat, jestli řidič používá k řízení obě ruce, nebo čas který řidič stráví vykonáváním určité akce (řazení, přepínání světel, ...). Tuto kontrolu zkvalitňuje odhad pozice ruky v prostoru. Se znalostí pozice jednotlivých kloubů ruky jednoduše odhadneme kvalitu provádění jednotlivých akcí. Můžeme sledovat jemnou motoriku řidiče, prodlevy mezi situacím, která se odehrává v simulátoru a reakci řidiče.

Propojením znalosti pozice ruky s dalšími znalostmi, jako je výraz obličeje, pohyb nohou nebo celého těla, získáme téměř kompletní informaci o chování řidiče. To poskytuje odezvu jak řidičům, kteří používají trenažér k tréninku svých řídičských vlastností, tak inženýrům navrhujících hardware i software trenažéru.

¹<https://scssoft.com>



Obrázek 6.1: Obrázek trenažérů řízení kamionu od firmy SCS Software.

Detekce rukou řidiče

Teorii problému detekce objektu v obraze probíráme v sekci 3.2. Konkrétně probíráme metodu detekce podle barvy objektu a detekci pomocí neuronových sítí. Cílem bylo zpracovat video z trenážeru řízení zachycující ruce v ptačím pohledu pomocí vybrané metody a zjistit polohu ruky na jednotlivých snímcích.

Video jsme zpracovali pomocí softwaru *FFmpeg*², který slouží k práci s audiem a videem. Video o délce 32s jsme vzorkovali s frekvencí 10 snímků za sekundu. Na výsledných 320 snímků jsme aplikovali implementované metody a následně je znovu spojili do videa.

Ukázka aplikace metody detekce podle barvy byla ukázána na obrázku 3.4. Tato metoda je velmi rychlá a pro základní informaci o pozici ruky dostačující. Konkrétně jsme na padesáti testovaných obrázcích dostali střední čas běhu algoritmu 56ms. Na zpracovávaném videu se vyskytuje jen jedna osoba a trenážer má pevné umístění a tedy konstantní pozadí. Menší kvalitu detekce by mohlo způsobit proměnlivé pozadí nebo výskyt jiných objektů barevně podobných ruce řidiče.

K detekci rukou pomocí neuronových sítí jsme natrénovali architekturu Faster RCNN s algoritmem hledajícím největší konturu barvy ruky v jednom z navržených regionů. Vybavování sítě trvá průměrně 2,1s, což není dostačující pro on-line režim kontroly. Je sice možné video zpracovat pomocí této architektury a dělat analýzu řízení později, ale videa trvající několik hodin by se zpracovávali velmi dlouho. Při vzorkování 10 snímků za sekundu, trvá zpracování 10s videa přibližně 3,5 minuty.

Výsledky na našem praktickém příkladu byli velmi dobré až na několik situací, kdy se ruce překrývali. Ukázka správné i špatné detekce ruky na jednotlivých snímcích je na obrázku 6.2. Při překrytí rukou detekovala architektura jen jednu ruku. Její ohraničení obsahovalo buď jen jednu ruku nebo obě ruce. Tato informace je také užitečná, jelikož můžeme předpokládat, že ve chvíli, kdy máme jen jedno ohraničení, jsou ruce u sebe.

Odhad pozice ruky řidiče

V této sekci představíme aplikaci odhadu klíčových bodů. Nejprve ukážeme aplikaci 2D odhadu klíčových bodů z RGB obrazu představenou v kapitole 4, na kterou navážeme 3D odhadem klíčových bodů jak z hloubkové mapy, tak z jediného RGB snímku.

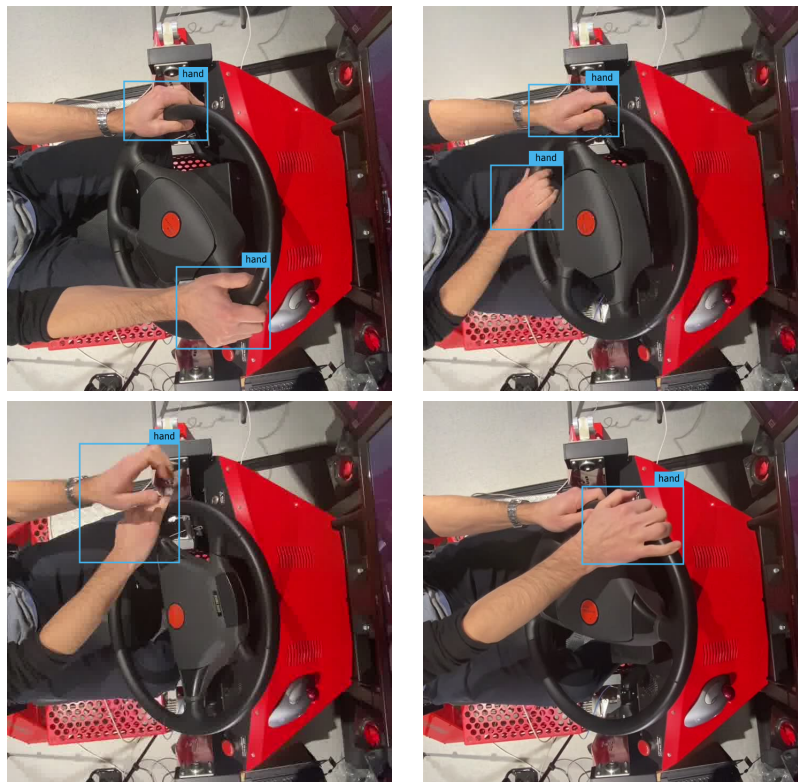
K odhadu 2D klíčových bodů jsme natrénovali architekturu skládající se ze sítí Faster RCNN a CPM, které detekují ruku, resp. klíčové body v obraze. Na námi anotovaných datech jsme dosáhli mAP rovné 92%. Vybavování celé sítě zabere na našem systému přes dvě sekundy na snímek. Nahrazením detekující části architektury sítí HandSegNet jsme dosáhli lepšího času vybavování sítě za cenu snížení přesnosti o 5%. Ukázka aplikace vylepšené architektury na snímky z trenážeru řízení je na obrázku 6.3.

Kvalita detekce pomocí naší architektury je dostatečná pro aplikaci na odhad 2D klíčových bodů. Možným zlepšením, při zpracování videa, je využití informace o poloze klíčových bodů v předchozím snímku. Pomocí této optimalizace bychom byli teoreticky schopni snížit čas vybavování sítě.

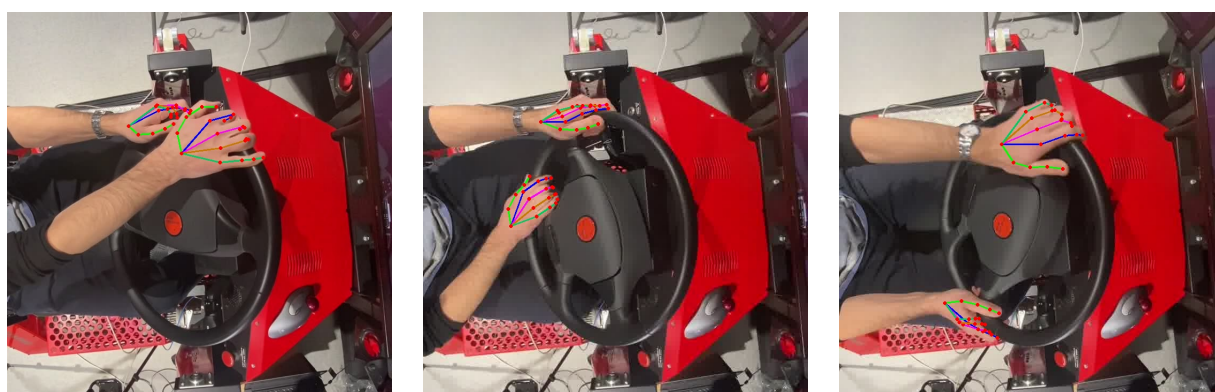
Metody odhadu z RGB-D obrazu, které jsme představili v sekci 5.1 nebyly aplikovány přímo na data z trenážeru řízení. Jelikož video s hloubkovou informací nebylo k dispozici nahráli jsme vlastní video pomocí zařízení *Kinect for Windows v1*, obrázek 6.4.

Metoda odhadu z hloubkové mapy měla průměrnou chybu ve vzdálenosti rovnou 9,34mm. Jedná se o rychlou metodu, která by mohla být upravena k aplikaci v živém přenosu. Na našem osobním systému trvá vybavování sítě 1,37s. Pokud bychom zapojili informaci o pozici klíčových bodů v předcházejícím snímku, byli bychom teoreticky schopni tento čas snížit. Nevýhodou této metody je potřeba předcházející detekce ruky v hloubkovém obraze a její vyřízení. Této problematice jsme se v této práci nevěnovali.

²<https://www.ffmpeg.org>



Obrázek 6.2: Ukázky zpracovávaných snímků z videa řízení trenážeru. Vrchní dva obrázky zachycují situaci se správnou detekcí rukou. Spodní dvojice obrázků zachycuje chybnou detekci, zapříčiněnou vzájemným překrytím rukou.



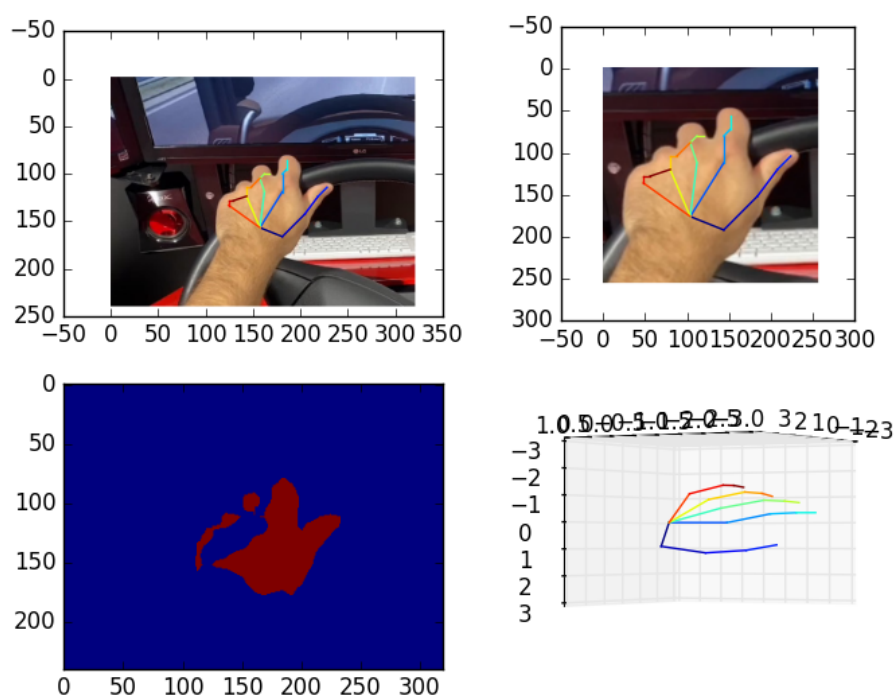
Obrázek 6.3: Aplikace architektury k 2D odhadu klíčových bodů na data z trenážeru řízení.



Obrázek 6.4: Snímky získané ze senzoru Kinect v1. V levém sloupci je RGB složka. Korespondující hloubkové mapy vidíme v pravém sloupci.

Metoda odhadu z jediného RGB snímku dosahovala vyšší přesnosti než metoda shrnutá výše. Průměrná chyba ve vzdálenosti odhadu byla $6,28mm$. Časová náročnost je vyšší než u druhé metody. Otázkou zůstává jakým způsobem implementovat přenos informace o předchozí poloze ruky při aplikaci metody na video. Ukázka aplikace metody je na obrázku 6.5.

Řešení úlohy odhadu pozice ruky prohlašujeme za úspěšné. Implementované metody dosahují dostatečné přesnosti. Možným zlepšením by bylo optimalizovat metody 3D odhadu pozice ruky k použití v živém přenosu a rozšířit trénovací dataset o větší množství vlastních anotovaných snímků, čímž bychom měli být schopni zvýšit přesnost metod a jejich robustnost.



Obrázek 6.5: Aplikace metody odhadu 3D pozice ruky z RGB obrazu. Nahoře vidíme oříznutý obraz s vykreslenými spojnicemi odhadnutých klíčových bodů. Vlevo dole je segmentovaná ruka, vpravo vedle vidíme vykreslený tvar ruky ve 3D souřadnicích.

Závěr

Práce se věnuje problematice odhadu pozice rukou při řízení automobilu z vizuálních dat. Úkolem bylo provést rešerši stávajících metod strojového učení zaměřených na odhad pozice rukou. Seznámit se s dostupnými datovými sadami pro řešení této problematiky. Implementovat některá stávající řešení a ověřit jejich funkčnost na vybraných datech. Následně jsme měli adaptovat vybraný algoritmus na data pořízená při řízení automobilu a vyhodnotit úspěšnost implementovaného řešení.

Práci jsme rozdělili do tří částí. První část se zabývá segmentací obrazu a detekcí objektu v obrazu. Představili a implementovali jsme dvě metody segmentace. Konkrétně se jednalo o segmentaci obrazu metodou normalizovaných řezů a segmentace pomocí aktivních kontur. Tyto segmentační metody dále sloužili jako nástroj v algoritmech pro detekci obrazu. Nejlepších vlastností dosahovala metoda založená na grafech, hlavně kvůli své časové nenáročnosti.

Pro detekci objektu v obraze jsme implementovali algoritmus založený na barvě objektu. Tento algoritmus se ukázal být velmi rychlým a dostačujícím pro detekci objektu, který se barevně odlišuje od ostatních instancí v obrazu. Jeho nevýhodou je potřeba trénovacího obrázku, pro detekci určitého objektu. Dále jsme implementovali dvě metody založené na hlubokém učení. Konkrétně se jednalo o architektury RCNN a Faster RCNN, které jsme podpořili našim algoritmem detekce v závislosti na barvě. První z architektur dosahuje dobrých výsledků přesnosti detekce, ale trpí velkou časovou náročností. Tento problém částečně vyřešila architektura Faster RCNN, která je oprostěná o návrh regionů zájmu pomocí selektivního prohledávání.

Druhá část práce se věnuje odhadu 2D pozice klíčových bodů ruky. Úlohu jsme formalizovali jako odhad 21 klíčových bodů kinematického modelu ruky. Pro jejich odhad jsme sestavili konvoluční neuronovou síť, která v první části využívá Faster RCNN pro detekci ruky v obrazu a následně odhaduje vlastní pozici kloubů ruky pomocí upravené sítě CPM. Tato metoda byla zpomalována první částí, tj. detekcí. Proto jsme navrhli segmentační síť nahrazující Faster RCNN. Metoda se ukázala být dostatečně přesná a rychlá i pro živý přenos.

V poslední části se věnujeme odhadu prostorové pozice ruky. Otestovali jsme metody pracující jak s hloubkovou mapou, tak s RGB snímkem. Metoda odhadu z hloubkové mapy je velmi rychlá, ale méně přesná než metoda odhadující pozici ruky jen z RGB snímku. Jako vhodnější jsme nakonec zvolili metodu odhadu z jediného RGB snímku nejen kvůli své přesnosti, ale také kvůli potřebě použití kamery snímající pouze RGB složku.

Nakonec předkládáme ukázkou aplikace implementovaných metod na data z trenažéru řízení. Výsledky, které nám metody vrací jsou dobře použitelné pro další analýzu chování řidiče. Další vylepšení jednotlivých metod je mimo jiné závislé na potřebách analytiků pracujících na modelech chování a kontroly řidiče.

Kromě seznámení se s existujícími datovými sadami jsme anotovali vlastní data jak pro detekci, tak pro odhad pozice klíčových bodů ve 2D i 3D. Pro anotaci hloubkových dat jsme sestavili vlastní nástroj, který v jednoduchém grafickém prostředí dovoluje uživateli volně prohlížet hloubkové dat z

různých pohledů a tím přesné umístění klíčových bodů. Výstupem je textový dokument se souřadnicemi klíčových bodů přiřazených ke korespondujícím obrázkům.

Pokračování práce může být zaměřeno na úpravu metod k použití na video. Tím bychom měli být schopni nejen snížit potřebný čas k detekci, resp. odhadu, ale také dosáhnou větší přesnosti výsledků. Nabízí se také rozšířit vlastní datové sady o anotace obrázků přímo z aplikace. Lepší kvality anotovaných dat bychom byli schopni dosáhnout nahráním RGB-D obrazu přímo z trenažéru řízení. Dále bychom mohli porovnat detekci pomocí Faster RCNN a HandSegNet. Segmentační síť jsme využívali až při řešení problému odhadu klíčových bodů, kvůli její malé časové náročnosti, proto nemáme přímé srovnání s ostatními metodami detekce objektu.

Apendix

V dodatkové části nabízíme představení dvou konvolučních neuronových sítí, které využíváme v architekturách detekce objektu, ztrátu křížové entropie, optimalizační algoritmus Adam a metodu podpůrných vektorů. Na konci této kapitoly přikládáme tabulku s referencemi na označení, které v práci používáme.

ResNet-50

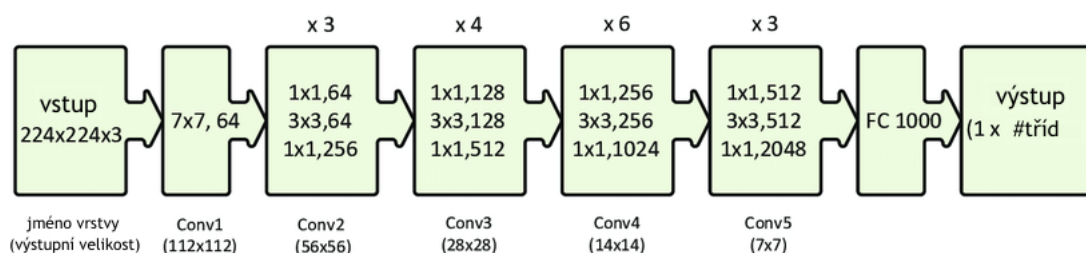
ResNet je zkratkou pro „Residual Network“, tj. reziduální síť. Jak název napovídá, novou metodou, kterou síť aplikuje je reziduální učení. Tento přístup představili v roce 2016 autoři článku [72].

V hlubokých konvolučních neuronových sítích jsou jednotlivé vrstvy poskládány za sebe a natrénovány k řešení problému. Síť se naučí různé úrovně příznaků, které slouží např. ke klasifikaci. Reziduální síť se, namísto příznaků, učí rezidua. Rezidua jsou výsledkem po odečtení naučeného příznaku od vstupu určité vrstvy. ResNet používá „zkratky“ spojující n. vrstvu s n+1. vrstvou. Bylo dokázáno, že trénink tohoto typu sítí je jednodušší, než trénink hlubokých neuronových sítí. Navíc dochází k samoregularizaci sítě.

ResNet-50 je reziduální neuronová síť s 50 vrstvami, jejíž schéma vidíme na obrázku 6.6. Existují i jiné typy, jako jsou ResNet-32, ResNet-101 a další.

Inception v2

Příchod sítě Inception [73] byl důležitým milníkem ve vývoji klasifikátorů založených na CNN. Předchozí konvoluční neuronové sítě se snažili dosáhnout lepších výsledků jen pomocí řazení většího



Obrázek 6.6: Blokové schéma architektury ResNet-50, představené v [72].

počtu skrytých vrstev za sebe. Tvůrci sítě Inception se přemýšleli nad vylepšením sítě z jiného hlediska. Postupně přicházeli s verzemi sítě Inception v1, v2, v3, v4 až k poslední Inception-ResNet.

Prvním vylepšením stávajícího přístupu je zapojení různých konvolučních masek v konvolučních vrstvách. Tím jsme schopni správně detekovat objekty, které mají veliký rozptyl ve své velikosti. Sít' tím nabývá více do „šířky“ než do „hloubky“.

Dalším zlepšením je faktorizace konvoluce. Autoři uvádějí, že konvoluce s maskou velikosti 5×5 je 2,78krát pomalejší než konvoluce s maskou 3×3 . Rozdělením konvoluce na dvě konvoluce s maskou 3×3 , vede k rychlejšímu výpočtu. Následně konvoluci typu $n \times n$ faktorizují na kombinaci konvolucí $n \times 1$ a $1 \times n$. Tímto ušetří až 33% výpočetní paměti.

Další vylepšení přicházeli se sítěmi Inception v3 výše. V naší práci jsme použili sít' Inception v2, proto nepopisujeme vlastnosti jejich nástupkyň.

Optimalizační algoritmus Adam

Název optimalizační algoritmu „Adam“ je zkratkou „Adaptivní odhad momentu“. Jedná se o rozšíření stochastických metod gradientního sestupu. Aktualizace parametrů probíhá podle předpisu

$$w \rightarrow w - \hat{m} \frac{\alpha}{\sqrt{\hat{v} + \epsilon}},$$

kde \hat{m} je zanikající průměr s konstantou β_1 a \hat{v} je necentrováný rozptyl s konstantou β_2 daný předpisem

$$\hat{m} = \frac{m}{1 - \beta_1},$$

$$\hat{v} = \frac{v}{1 - \beta_2}.$$

Pro hodnoty m, v platí aktualizace

$$m \rightarrow \beta_1 m + (1 - \beta_1) \frac{\partial C}{\partial w},$$

$$v \rightarrow \beta_2 v + (1 - \beta_2) \left(\frac{\partial C}{\partial w} \right)^2$$

a jsou inicializované na hodnotu 0. Doporučená inicializace pro další parametry je: $\epsilon = 10^{-8}$, $\beta_1 = 0,9$ a $\beta_2 = 0,999$.

Ztráta křížové entropie

Křížová entropie je míra odlišnosti mezi dvěma rozděleními dané náhodné proměnné. Informaci $I(x)$ události x s danou pravděpodobností $P(x)$ definujeme jako

$$I(x) = -\log_2(P(x)).$$

Dále definujeme entropii pro diskrétní stavy $x \in X$ jako

$$H(X) = -\sum_{x \in X} P(x) \log_2(P(x)).$$

Křížovou entropii rozdělení Q vůči rozdělení P se stejným nosičem X definujeme jako

$$H(P, Q) = -\sum_{x \in X} P(x) \log_2(Q(x)).$$

Nakonec ukážeme příklad ztráty křížové entropie pro úlohu klasifikace do tříd $i \in T$

$$C = - \sum_{i \in C} y_i \log_2(\hat{y}_i),$$

kde y_i je pravdivá, anotovaná hodnota a \hat{y}_i je predikovaná hodnota.

Metoda podpůrných vektorů

Metoda podpůrných vektorů, také známá jako SVM (z angl. „support vector machine“), je lineární model pro klasifikaci regresi. Myšlenka algoritmu je jednoduchá. Vytvořit křivku nebo nadplochu separující data do tříd.

Mějme

$$\{\mathbf{x}_i, y_i\}_{i=1}^n,$$

kde $\mathbf{x}_i \in \mathbb{R}^p$ jsou data a $y_i \in \{1, -1\}$ třídy. Naším úkolem je klasifikovat nové \mathbf{x} do třídy -1 nebo 1 . Hledáme nadplochu, které maximalizuje šířku hranice. Hranice je dána předpisem

$$\mathbf{w}^T \mathbf{x} + b = 0,$$

kde $b \in \mathbb{R}$ je bias a \mathbf{w} je vektor vah. Šířka hranice je dána předpisem $d = 2 / \|\mathbf{w}\|_2$.

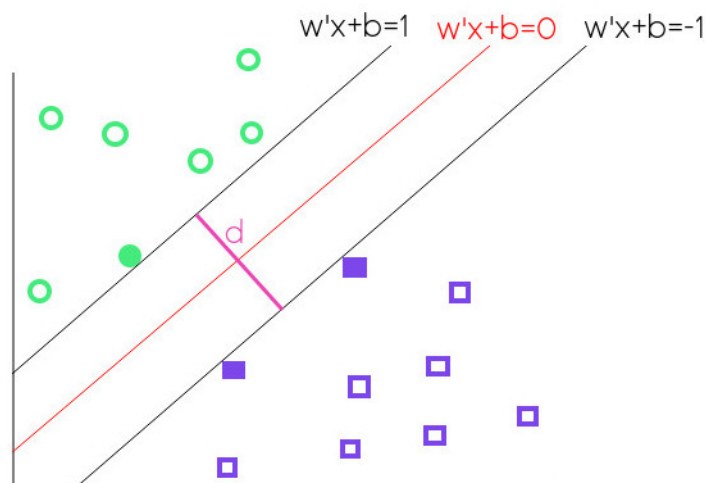
Optimalizační úloha potom vypadá následovně. Hledáme

$$\arg \max_{\mathbf{w}, b} \left\{ \frac{2}{\|\mathbf{w}\|_2} \right\}, \text{za podmínek}$$

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1, \text{pro } y_i = 1,$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1, \text{pro } y_i = -1.$$

Obrázek 6.7 ilustruje praktický příklad na zavedené notaci.



Obrázek 6.7: Aplikace SVM ke klasifikaci do dvou tříd. Červeně je znázorněna separující nadplocha. Fialově šířka, kterou se snažíme maximalizovat.

Seznam zkratek a značek

Následující tabulka slouží jako přehled zkratek a notací, které používáme v práci.

Označení	Význam
CNN	konvoluční neuronová síť
FCN	plně propojená síť
RPN	síť navrhuující oblasti zájmu
CPM	convolutional pose machine [69]
VR	virtuální realita
mAP	mean average precision
IoU	intersection over union
RGB	barevný model červená-zelená-modrá
RGB-D	barevný model RGB s hloubkovou mapou
HSV	barevný model odstín-sytost-jas
SVM	metoda podpůrných vektorů
XD	X -dimenzionální prostor
px	pixel
x	skalár
\mathbf{x}	vektor
\mathbb{X}	matice
\mathbf{x}^T	transponovaný vektor \mathbf{x}
$x \in \hat{n}$	$x = 1, \dots, n$
\wedge resp. \vee	logické operátory „a“ resp. „nebo“
\cap resp. \cup	průnik resp. sjednocení
$\ \cdot\ _2$	Euklidovská norma (L2 norma)
$C(\cdot)$	ztrátová funkce
\mathbb{R}	prostor reálných čísel
$\mathbb{R}^{m \times n}$	vektorový prostor reálných matic velikosti $m \times n$
$\partial f / \partial x$	parciální derivace funkce f podle proměnné x
∇f	gradient funkce f
$A \circ B$	Hadamardův součin matic A a B
*	konvoluce
$\mathcal{N}(\mu, \sigma^2)$	Gaussovo rozdělení se střední hodnotou μ a rozptylem σ^2
$\mathcal{U}(a, b)$	Rovnoměrné rozdělení v intervalu $\langle a, b \rangle$
$x \ll a$ resp. $x \gg a$	x je mnohem menší resp. větší než a
$E[\cdot]$	střední hodnota

Literatura

- [1] M. Mayer, P. Hlušík, Ruka hemiparetického pacienta: Neurofyziologie, parofyziologie, rehabilitace. Rehabilitacia, 2004.
- [2] H. Chang, G. Garcia-Hernando, D. Tang, T. K. Kim, Spatio-temporal hough forest for efficient detection-localisation-recognition of fingerwriting in egocentric camera. Computer Vision and Image Understanding, 2016.
- [3] F. Yin, X. Chai, X. Chen, Iterative reference driven metric learning for signer independent isolated sign language recognition. In European Conference on Computer Vision, 2016.
- [4] J. Weissmann, R. Salomon, Gesture recognition for virtual reality applications using data gloves and neural networks, In international Joint Conference on Neural Networks, 1999, 2043-2046.
- [5] D. J. Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, J. Shotton, Fits like a glove: Rapid and reliable hand shape personalization. In Computer Vision and Pattern Recognition, 2016.
- [6] D. Tang, T. H. Yu, T. K. Kim, Real-time articulated hand pose estimation using semi-supervised transductive regression forest. In International Conference on Computer Vision, 2013.
- [7] D. Tang, H. J. Chang, A. Tejani, T. K. Kim, Latent regression forest: Structured estimation of 3D hand posture. In International Conference on Computer Vision, 2014.
- [8] T. Rohlfing, R. Brandt, R. Menzel, D. B. Russakoff, C. R. Maurer Jr., Quo vadis, atlas-based segmentation? In Handbook of Biomedical Image Analysis, Springer, 2005, 435-486.
- [9] T. Homola, Techniques for Content-Based Sub-Image Retrieval. Masarykova univerzita, Brno, 2014.
- [10] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis. In IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, 2002, 603-619.
- [11] D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems. In Communications on Pure and Applied Mathematics, vol. 42, no. 5, Harvard University, 1989, 577-685.
- [12] S. E. Chew, N. D. Cahill, Semi-Supervised Normalized Cuts for Image Segmentation. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, 1716-1723.
- [13] J. Shi, J. Malik, Normalized Cuts and Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, 2000, 888-905.

- [14] U. von Luxburg, A Tutorial on Spectral Clustering. Technical Report No. TR-149, Max Planck Institute for Biological Cybernetics, 2007.
- [15] B. Mohar, Some applications of Laplace eigenvalues of graphs. In *Graph Symmetry: Algebraic Methods and Applications*, 1997, pp. 225 – 275.
- [16] F. Sun, J. He, A Normalized Cuts Based Image Segmentation Method. In *Second International Conference on Information and Computing Science*, Manchester, UK, 2009, 333-336.
- [17] M. B. Salah, A. Mitiche, I. B. Ayed, Multiregion Image Segmentation by Parametric Kernel Graph Cuts. In *IEEE Transactions on Image Processing*, vol. 20, no. 2, 2011, 545-557.
- [18] Y. Boykov, G. Funka-Lea, Graph cuts and efficient ND image segmentation. *International Journal of Computer Vision*, vol. 70, no. 2, 2006, 109-131.
- [19] B. Peng, L. Zhang, J. Yang, Iterated Graph Cuts for Image Segmentation. *Lecture Notes in Computer Science*, vol 5995. Springer, Berlin, Heidelberg, 2010.
- [20] M. Španěl, V. Beran, *Obrazové segmentační techniky: přehled existujících metod*. VUT, fakulta informačních technologií, 2006.
- [21] T. F. Cootes, G. J. Edwards, Ch. J. Taylor, Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, 2001, 681-685.
- [22] L. H. Staib, J. S. Duncan, Model-based deformable surface finding for medical images. *IEEE Transactions on Medical Imaging*, vol. 15, no. 5, 1996, 720-731.
- [23] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active Contour Models. In *International Journal of Computer Vision*, Kluwer Academic Publishers, Boston, 1998, 321-331.
- [24] R. Verma, An Efficient Color-based Object Detection and Tracking in Videos. In *International Journal of Computer Engineering and Applications*, Vol. 11, Issue 11, 2017, 2321-3469.
- [25] X. Zhu, J. Yang, A. Waibel, Segmenting hands of arbitrary color. *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, 2000, 446-453.
- [26] P. Buehler, M. Everingham, D. Huttenlocher, A. Zisserman. Long Term Arm and Hand Tracking for Continuous Sign Language TV Broadcasts. In *Proceedings of the British Machine Conference*, 2008, 110.1-110.10.
- [27] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.
- [28] A. Mittal, A. Zisserman, P. HS Torr, Hand detection using multiple proposals. *BMVC*, Vol. 2. No. 3. 2011.
- [29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [30] R. Girshick. Fast r-cnn, In *Proceedings of the IEEE International Conference on Computer Vision*, 2015, 1440–1448.

- [31] R. Girshick, J. Donahue, J. M. T. Darrell, Region-based convolutional networks for accurate object detection and semantic segmentation. *IEEE Transactions on PAMI*, 2015, 142-158.
- [32] J. ORourke, N. Bradler, Model-based image analysis of human motion using constraints propagation. *IEEE TPAMI*, vol. 2, 1980.
- [33] A. Ali, 3D Human Pose Estimation. Georgia Institute of Technology, Dissertation, 2019.
- [34] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, Efficient regression of general-activity human poses from depth images. In *International Conference on Computer Vision*, 2011.
- [35] A. Toshev, C. Szegedy, Deeppose: Human pose estimation via deep neural networks. In *Computer Vision and Pattern Recognition*, 2013.
- [36] M. Valstar, B. Martinez, X. Binefa, M. Pantic. Facial point detection using boosted regression and graph models. In *Computer Vision and Pattern Recognition*, 2010.
- [37] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European Conference on Computer Vision*, 2014.
- [38] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, K. P. Murphy. Towards accurate multi-person pose estimation in the wild. In *Computer Vision and Pattern Recognition*, 2017.
- [39] S.-E. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh. Convolutional pose machines. In *Computer Vision and Pattern Recognition*, 2016.
- [40] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *Computer Vision and Pattern Recognition*, 2013.
- [41] C. Xu, L. Cheng, Efficient hand pose estimation from a single depth image. In *International Conference on Computer Vision*, 2013.
- [42] N. Neverova, C. Wolf, G. Taylor, F. Nebout, Hand segmentation with structured convolutional learning. In *Asian Conference on Computer Vision*, 2014.
- [43] C. Wan, A. Yao, L. V. Gool, Hand pose estimation from local surface normals. In *European Conference on Computer Vision*, 2016.
- [44] M. Oberweger, P. Wohlhart, V. Lepetit, Hands deep in deep learning for handpose estimation. In *Computer Vision Winter Workshop*, 2015.
- [45] J. Taylor, et al, Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics*, 2016.
- [46] I. Oikonomidis, N. Kyriazis, A. A. Argyros, Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *International Conference on Computer Vision*, 2011.
- [47] A. Caffaz, G. Cannata, The design and development of the DIST-Hand dextrous gripper. In *Robotics and Automation Proceedings. IEEE International Conference on*. Vol. 3. 1998, 2075 –2080.
- [48] H. Kawasaki, T. Komatsu, K. Uchiyama, Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II. In *Mechatronics, IEEE/ASME Transactions on*, 2002, 296 –303.

- [49] J. Han, L. Shao, D. Xu, J. Shotton, Enhanced computer vision with mi-crosoft kinect sensor: A review. *IEEE transactions on cybernetics*, vol. 43, no. 5, 2013, 1318–1334.
- [50] I. Oikonomidis, N. Kyriazis, A. A. Argyros, Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference*, 2011.
- [51] S. Sridhar, A. Oulasvirta, C. Theobalt, Interactive markerless articulated hand motion tracking using rgb and depth data. In *Computer Vision IEEE International Conference*, 2013, 2456–2463.
- [52] D. Goudie, *Pose Estimation from Depth Images Using Convolutional Neural Networks*, Thesis, University of Manchester, 2018.
- [53] K. B. Shaik, P. Ganesan, V. Kalis, B. S. Sathish, J. M. M. Jenitha, Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space. In *3rd International Conference on Recent Trends in Computing*, 2015, 41-48.
- [54] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, J. Jatakia, Human Skin Detection Using RGB, HSV and YCbCr Color Models. In *CCASP/ICMMD- Advances in Intelligent Systems Research*, Vol. 137, 2017, 324-332.
- [55] S. Bambach, S. Lee, D. J. Crandall, Ch. Yu, Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In *IEEE International Conference on Computer Vision (ICCV)*, 2015, 1949-1957.
- [56] A. Mittal, A. Zisserman, P. Torr, Hand detection using multiple proposals. In *Proceedings of the British Machine Vision Conference*, 2011, 75.1-75.11.
- [57] M. Abadi, et al., *Tensorflow: A system for large-scale machine learning*. In *12th USENIX symposium on operating systems design and implementation*, 2016.
- [58] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient Graph-Based Image Segmentation. In *International Journal of Computer Vision* 59, 2004, 167-181.
- [59] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, Selective Search for Object Recognition. In *International Journal of Computer Vision*, 104(2), 2013, 154–171.
- [60] L. N. Leslie, A disciplined approach to neural network hyper-parameters: Part1 - Learning rate, batch size, momentum and weight decay. *US Naval Research Laboratory Technical Report 5510-026*, 2018.
- [61] I. Loshchilov, F. Hutter, SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [62] S. Ren, K. He, R. Girshick, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv preprint arXiv:1506.01497*, 2015.
- [63] M. Straka, *Deep Learning. Lecture by MFF CUNI*, <https://ufal.mff.cuni.cz/courses/npfl114/1819-summer>, 2019.
- [64] M. Nielsen, *Neural Networks and Deep Learning. Online book* <http://neuralnetworksanddeeplearning.com>, 2021.

- [65] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010, 249-256.
- [66] B. Doosti, Hand pose estimation: A survey. In arXiv preprint arXiv:1903.01013, 2019.
- [67] C. Zimmermann, T. Brox, Learning to estimate 3d hand pose from single rgb images. In Proceedings of the IEEE international conference on computer vision, 2017, 4903-4911.
- [68] F. Xiong, et al. A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, 793-802.
- [69] S. Wei, V. Ramakrishna, T. Kanade, Y. Sheikh, Convolutional pose machines. In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016, 4724-4732.
- [70] J. Zhang, et al., 3d Hand Pose Tracking and Estimation Using Stereo Matching. arXiv preprint arXiv:1610.07214, 2016.
- [71] R. Zhao, Y. Wang, A. Martinez, A Simple, Fast and Highly-Accurate Algorithm to Recover 3d Shape from 2d Landmarks on a Single Image. arXiv preprint arXiv:1609.09058, 2016.
- [72] He, Kaiming, et al., Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, 770-778.
- [73] Ch. Szegedy, et al., Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, 2812-2826.