VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ODHAD POZICE CHODIDEL V OBRAZE

ESTIMATION OF THE POSITION OF THE FEET IN A IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. LUKÁŠ HAVLÍČEK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ GOLDMANN**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Havlíček Lukáš, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Vývoj aplikací
Název: **Odhad pozice chodidel v obraze**
Estimation of the Position of the Feet in a Image
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s oblastí antropometrie lidského těla a zjistěte, na základě, jakých antropometrických vlastností je možné odhadnout pozici chodidel v obraze.
2. Nastudujte problematiku neuronových sítí pro řešení regresních úloh. Sumarizujte informace o dostupných neuronových sítí pro detekci významných bodů lidského těla.
3. Navrhněte neuronovou síť, která dokáže odhadnout pozice zakrytých chodidel osob v obraze. Při návrhu předpokládejte, že na snímcích mohou být zakryté i celé nohy.
4. Navrženou neuronovou síť implementujte v programovacím jazyce Python a natrénujte.
5. Proveďte experimenty zaměřené na určení přesnosti pozice chodidel v obraze. Získané výsledky diskutuje a navrhněte možná rozšíření.

Literatura:

- PREEDY, Victor R. (ed.). *Handbook of anthropometry: physical measures of human form in health and disease*. Springer Science & Business Media, 2012.
- HEATON, Jeff. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning. 2018.CONNOLLY, Janis H.; ARCH, M. NASA Standard 3000 Human Systems Integration Standards (HSIS) Update. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Sage CA: Los Angeles, CA: SAGE Publications, 2005. p. 2018-2022.
- HEATON, Jeff. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning. 2018.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a 3.

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

Vedoucí práce: **Goldmann Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Tato práce se zabývá návrhem a implementací řešení odhadu pozice chodidel v obraze se zaměřením na případy, kdy jsou nohy v obraze zakryté. Je zde popsána oblast antropometrie, neuronových sítí, návrhu a implementace řešení a následné experimenty. K řešení je využito vlastní konvoluční sítě, implementované pomocí rozhraní Keras s vytvořením vlastního datasetu se zaměřením na zakryté nohy. U experimentů je využito ostatních bodů na těle za účelem zvýšení přesnosti.

Abstract

This thesis deals with the design and implementation of solution for estimating the position of the feet in the image, focusing on cases where the feet are covered in the image. The area of anthropometry, neural networks, solution design and implementation and subsequent experiments are described here. The solution uses own convolutional network, implemented using the Keras interface with the creation of own dataset with a focus on covered legs. In the experiments, other points on the body are used to increase accuracy.

Klíčová slova

neuronové sítě, antropometrie, CNN, strojové učení, Python, Keras, COCO dataset

Keywords

neural networks, anthropometry, CNN, machine learning, Python, Keras, COCO dataset

Citace

HAVLÍČEK, Lukáš. *Odhad pozice chodidel v obraze*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann

Odhad pozice chodidel v obraze

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Tomáše Goldmanna. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Lukáš Havlíček
16. května 2022

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Tomášovi Goldmannovi za odborné vedení a konzultace, které mi poskytl při vypracování této práce.

Obsah

Kapitola 1

Úvod

Odhad pozice chodidel v obraze se zabývá problémem označení bodů na lidském těle za pomocí zpracování obrazu. Vyznačené body umožňují další analýzy, jako je detekce pózy, analýza pohybu u sportu a dalším možným. Vyznačení bodů většinou probíhá za pomocí neuronových sítí, které pomocí analýzy obrazu vyhledávají významné body na těle, které jsou následně vyznačeny. Cílem této práce je zaměřit se na neuronovou síť označující vybrané body, a to chodidla, se zaměřením na případy, kdy jsou chodidla, případně celé nohy v obraze zakryté, a není tedy možné z obrazu určit pozici chodidel. Tento odhad pozice chodidel bude spočívat na viditelné horní polovině těla, kde na základě poměrů lidského těla bude odhadnuta pozice chodidel, i když nejsou viditelné. Uplatnění je poté u zpracování obrazu z míst jako jsou kanceláře či obchodní prostory, kde se nachází objekty do úrovně pasu, jako stůl či přepážka a není přes tyto objekty z případného videozáznamu možné vidět pozici chodidel.

Tato práce je členěna následovně. První kapitola 2 je věnována antropometrii, kde je popsána antropometrie obecně, její využití v mnoha oblastech a vybrané metriky, mezi které patří i ty související s nohami a bylo by je možné použít k odhadu pozice chodidel. Následující kapitola 3 je věnována neuronovým sítím, kde je popsán princip neuronových sítí se zaměřením na zpracování obrazu a regresní úlohy a dále pak shrnutí detekce významných bodů lidského těla. V další kapitole 4 je poté popsán návrh řešení související s realizací neuronové sítě řešící odhad pozice chodidel a vytvoření vlastního datasetu, který bude obsahovat snímky osob se zakrytými nohami. Předposlední kapitola 5 obsahuje implementační detaily týkající se vytvoření datasetu a práce s tímto datasetem. Dále je zde popsáno trénování modelu a jeho postupné úpravy. V poslední kapitole 6 jsou poté popsány experimenty, kdy jsou modelu také předány informace o ostatních bodech lidského těla, které byly součástí datasetu, za účelem zvýšení přesnosti.

Kapitola 2

Antropometrie

Antropometrie je věda zabývající se fyzickým měřením rozměrů, poměrů a složení lidského těla a jeho jednotlivých částí [8]. Tyto hodnoty jsou následně využívány v mnoha oblastech jako je medicína, kde mohou sloužit pro úpravu léčby nebo pro návrh nových zařízení (auta, nábytek atd.), tak aby vyhovovaly lidskému tělu. Nebo také v oblasti IT, kde může být antropometrie použita u biometrie nebo u zpracování obrazu.

V této kapitole jsou v první části 2.1 popsány vybrané metriky, které patří jak k nejznámějším a nejpoužívanějším, tak také metriky, které souvisí s nohami vůči tělu. V druhé části 2.2 je poté obecně popsáno využití antropometrie a velice krátce o její historii.

2.1 Vybrané metriky

Následující sekce vychází z [22, 8, 6, 1]. V této sekci jsou popsané vybrané metriky, které patří mezi nejpoužívanější co se týče běžného použití, či metriky související s nohami a mohly by být využity k odhadu pozice chodidel.

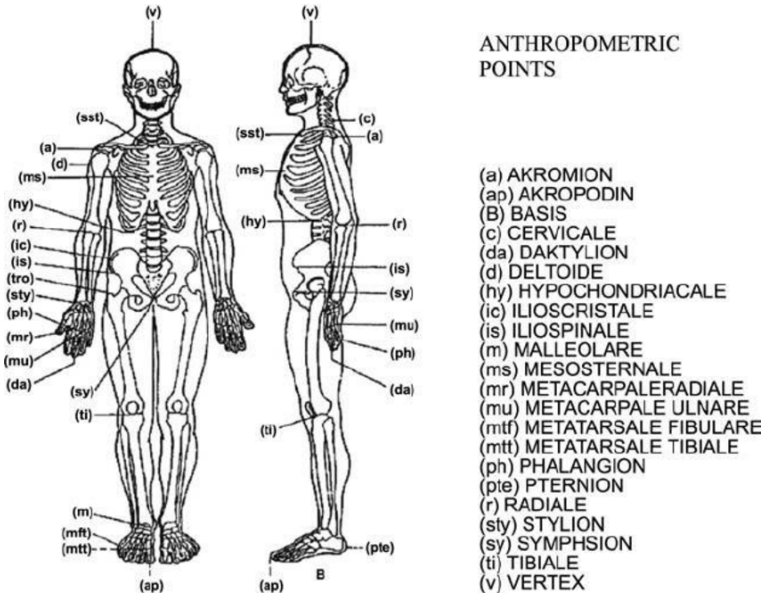
Metriky mohou být rozděleny do 2 kategorií:

- Fyzické hodnoty - váha, výška, délka končetin, tloušťka kůže apod.
- Poměrové hodnoty - BMI (váha vůči výšce), poměr pasu k výšce, BMI vůči věku atd.

Antropometrie se ale nezabývá pouze vnější částí lidského těla, ale zabývá se i vnitřní částí těla jako jsou orgány, například velikost mozku. Rovněž také antropometrie určuje body na lidském těle, vůči kterým se provádí měření a lze některé z nich vidět na obrázku 2.1.

2.1.1 Výška

Jako pravděpodobněji nejpoužívanější metrika bude výška osoby. Výška se měří ve stoje, kdy se záda dotýkají zdi či měřidla pomocí stadiometru, který se většinou skládá z pravítka a posuvné rysky. Vůči výšce se vztahují ostatní metriky, jako je rozpětí paží, které bývá téměř identické s výškou. Délka nohou, která lze velice nepřesně vyjádřit jako polovina výšky těla, z toho odvíjející se výška trupu a další jiné metriky lidského těla. Díky těmto závislostem je výška nejjednodušší parametr, při různém rozhodování, jako výběr správné velikosti, či metoda výběru, kde je výška například využívána jako rozhodující faktor při vstupu do armády.



Obrázek 2.1: Ukázka antropometrických bodů. Převzato z [27].

2.1.2 BMI

Následující sekce vychází z [29]. Související s výškou je také *Body Mass Index* (BMI) neboli index tělesné hmotnosti, který bývá nejčastěji využíván k porovnání hmotnosti mezi více lidmi s různou výškou s vzorcem 2.1 s jednotkami kilogram a metr.

$$BMI = \frac{hmotnost}{vyska^2} \tag{2.1}$$

Dle organizace WHO jsou pro dospělou osobu starší 20 let definovány hranice na základě tabulky 2.1.

Název	**BMI**
Podváha	<18,5
Normální váha	18,5 - 24,9
Nadváha	25,0 - 29,9
Obezita	>= 30 (dále děleno na obezitu 1. až 3. stupně)

Tabulka 2.1: Hranice BMI dle WHO [29].

V různých zdrojích se lze ale setkat s různými hranicemi pro BMI, většinou lišící se u spodní hranice normální váhy, lze se ale také setkat s rozdělením BMI pro mužské a ženské pohlaví. I když se jedná pravděpodobně o nejčastější způsob klasifikace váhy, tak není ideální, hlavně z důvodu nereflektujícího poměru svalové hmoty a tuku.

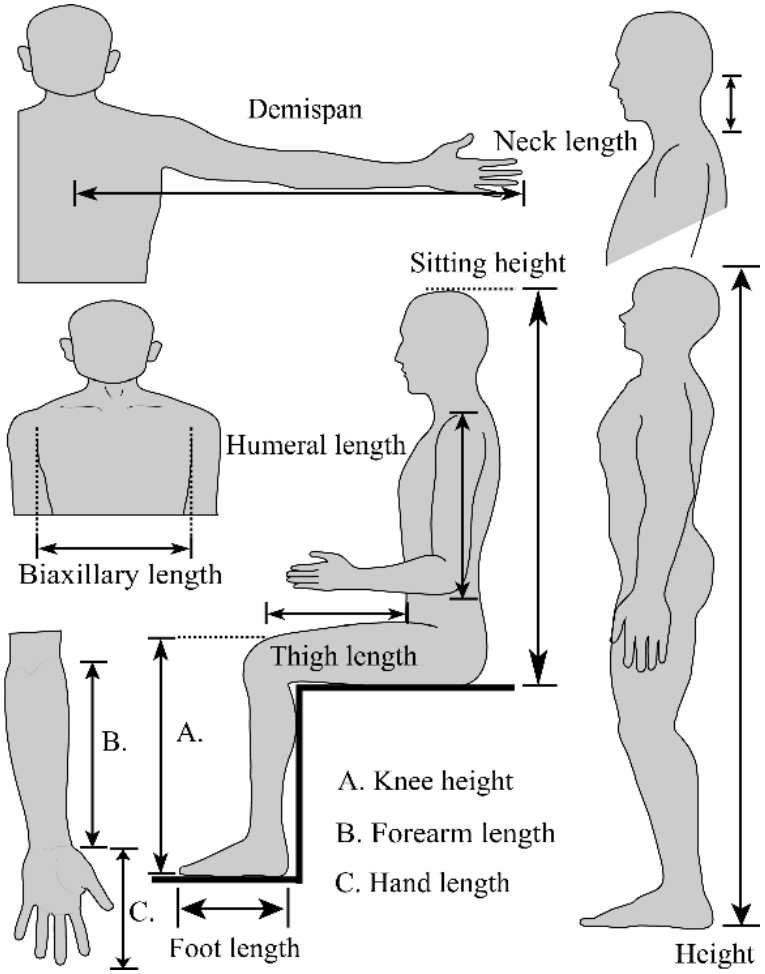
Waist-hip ratio a Waist-height ratio

Waist-hip ratio (WHR) a *Waist-height ratio* (WHtR) neboli poměr obvodu pasu vůči bokům a poměr obvodu pasu vůči výšce jsou další možné ukazatele zdravotního stavu, které by neměly být ovlivněny poměrem svalové hmoty a tuku a měly by tedy být lepšími ukazateli než BMI. Jejich výpočet lze vidět ve vzorci 2.2 s libovolnými jednotkami.

$$WHR = \frac{obvod\ pasu}{obvod\ boku} \qquad WHtR = \frac{obvod\ pasu}{vyska} \tag{2.2}$$

2.1.3 Délka nohou

Jednou z metrik je také délka nohou, která je potřebná pro odhad pozice chodidel člověka. Vzhledem k tomu, že nohy mohou být při tomto odhadu zakryté a zároveň logickým problémem je, že délka nohou se u jednotlivých jedinců liší a není tedy možné použít „univerzální“ hodnotu. Je tedy potřeba tuto délku odvodit z ostatních metrik lidského těla. Související s délkou nohou je výška jedince v sedě (označována jako SH - *Sitting height*), která se měří od židle po vrchol hlavy, jak lze vidět na obrázku 2.2. Pokud nás tedy zajímá délka nohou a znali bychom výšku člověka (označována jako H - *Height*), stačila by nám tedy tato metrika. Z obrazu při zakrytých nohou bude možné získat SH, ale celková výška pravděpodobně přítomna nebude, a proto bude potřeba ji odvodit. K tomuto bude potřeba využít další metriky.

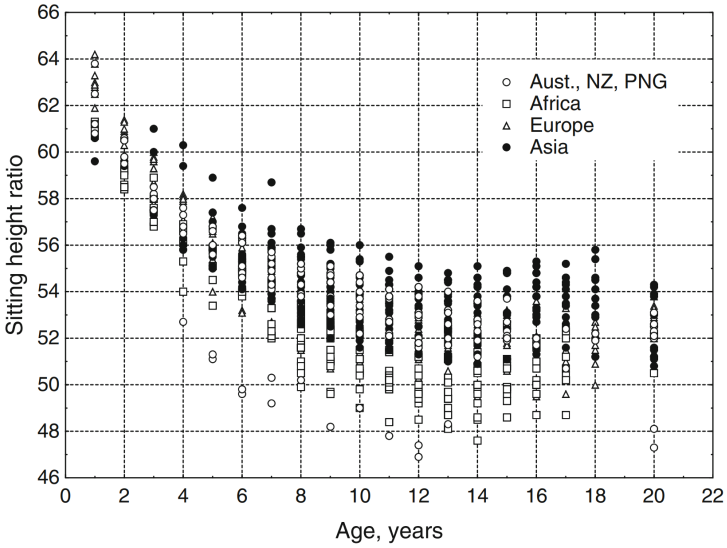


Obrázek 2.2: Ukázka antropometrie pro měření výšky a souvisejících metrik. Převzato z [5].

Sitting height ratio

Jednou z těchto metrik je SHR (*Sitting Height Ratio* 2.3), která už je více univerzální. Převážně tedy v dospělosti, kdy se již tělo dále nevyvíjí, jak lze vidět v grafu 2.3. Jedná se o poměr mezi výškou (ve vzorci H) a výškou v sedě (ve vzorci SH), kde tyto metriky na lidském těle lze vidět na obrázku 2.2.

$$SHR = \frac{SH}{H} * 100 \tag{2.3}$$



Obrázek 2.3: Graf SHR dle geografie. Převzato z [22].

Knee height ratio

Další související metrikou je KHR (*Knee Height Ratio*) neboli poměr mezi výškou a výškou kolene, která je označována jako KH - *Knee height* a lze vidět také na obrázku 2.2. Výpočet KHR 2.4 je víceméně stejný jako výpočet SHR. KHR a KH by teoreticky bylo možné využít v případě, kdy nebudou nohy v obrazu zakryty celé a bylo by možné lépe predikovat umístění chodidel.

$$KHR = \frac{KH}{H} * 100 \tag{2.4}$$

Existují další možné metriky, jako *Relative Subischial Leg Length* (RSLL) nebo *Thigh length* (TL), které souvisí s předchozími a dají se z vyjmenovaných odvodit. Všechny tyto hodnoty jsou ale do jisté míry ovlivněny jak „rasou“ člověka, tak prostředním, ve kterém daný jedinec vyrůstá. Z většiny výzkumů ale vyplývá, že právě prostředí má na délku nohou větší vliv jak geny, kde v případě kvalitnějšího prostředí bývá délka nohou větší, a naopak v horším prostředí poté kratší. Toto se následně vztahuje i na odvozené metriky, protože je tímto ovlivněna celková délka nohou.

2.2 Využití antropometrie

Následující sekce vychází z [28]. Vzhledem k tomu, že antropometrie se zabývá celkovým měřením lidského těla, je využívána v širokém spektru oblastí, ať už přímo jako je oblast lékařství, kde jsou hodnoty přímo měřeny. Nebo nepřímo, jako může být návrh nových zařízení, kde účelem není měřit a sledovat lidské tělo, ale navrhnout nové zařízení, které bude „kompatibilní“ s většinovou částí lidské populace. K tomu je potřeba znát typické hodnoty lidských rozměrů z antropometrie.

V úplném počátku bylo měření lidského těla využíváno v oblasti umění pro malby a později v sochařství, kde historie sahá až do 3. století před naším letopočtem ve starověkém Egyptě. Dalším rozšířením bylo v oblasti paleoantropologie, jež je součástí paleontologie, kde se zabývalo evolucí a porovnáváním lidských kostí, související také se starodávným Egyptem a zkoumání rasového původu mumií. Následně se antropometrie začala využívat v kriminalistice k identifikaci jedinců, kde také antropometrie získala svůj název viz 2.2.1.

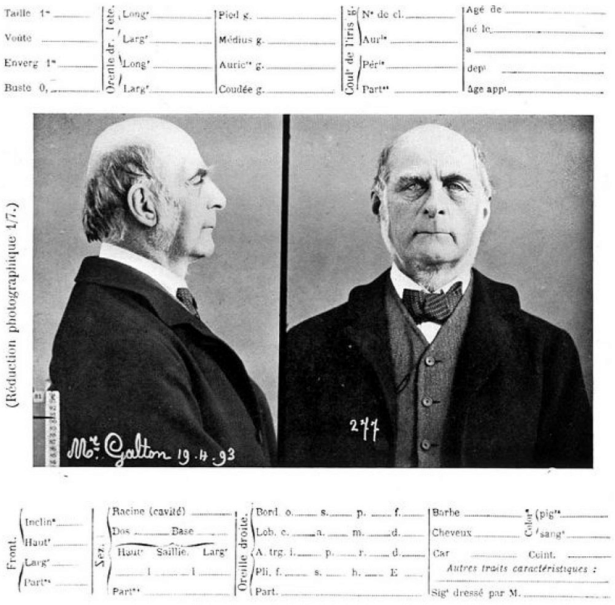
Nyní se antropometrie využívá v mnoha oblastech. Například při návrhu nových zařízení a oblečení, sledování růstu v průběhu dospívání, identifikace jedinců, jako jsou přístupové systémy nebo u kriminalistiky a v mnoha dalších případech. V následujících sekcích jsou poté stručně popsány vybrané kategorie využití antropometrie.

2.2.1 Kriminalistika

Následující sekce vychází z [8]. Jak již bylo zmíněno v úvodu této sekce, antropometrie má počátky v 19. století, přesněji počátek stanovení termínu antropometrie, samotné měření lidských veličin datuje mnohem dříve. Tento počátek začal v oblasti kriminalistiky v Paříži, kde Alphonse Bertillon na základě předpokladu, že se struktura lidského těla po věku 20 let nemění, zavedl sadu metrik k jednoznačnému určení osoby, skládající se z výšky, šířky, velkosti nohy, šířky hlavy, délky prostředníčku a několika dalších vlastností. Doposud bylo využíváno pouze fotky a jména, kde právě jméno bylo u kriminálníku často falešné, aby se vyhnuli přísnějším trestům za opakované přestupky, a proto bylo potřeba zajistit lepší způsob identifikace. Na obrázku 2.4 lze vidět, jak takový záznam vypadal. Je zde také možné vidět nový způsob pořizování fotografií, a to z předu a z profilu, který zároveň stanovoval světelné podmínky a vzdálenosti při focení, aby bylo možné fotografie lehčeji porovnávat. Tento způsob identifikace nebyl ale dokonalý, v tehdejší době byly měřící přístroje drahé a měření nebylo přesné a muselo být opakováno. Následně byla použita průměrná naměřená hodnota a při porovnání byly poté brány v potaz odchylky v rámci několika mm. Dalším hlavním problémem bylo, že dané měření bylo určeno převážně pro muže starší 20 let, a tak selhávalo u dětí a žen. Z těchto důvodů byl tento způsob identifikace později nahrazen otisky prstu, které zůstávají neměnné v průběhu života, z původní identifikace se však zachoval styl fotografií, se kterým je možné setkat se i v dnešní době.

2.2.2 Lékařství

Následující sekce vychází z [28]. Antropometrie úzce souvisí s odvětvím medicíny, kde je potřeba měřit lidské tělo. Jako nejjednodušší příklad antropometrie v lékařství může být preventivní prohlídka u lékaře, kde bývá součástí měření výšky a váhy a následné zaznamenání těchto hodnot, což není nic jiné než antropometrie. Tyto hodnoty jsou následně využívány ke sledování růstu a může z nich lékař vyvést další možné závěry, jako může být léčba, další vyšetření či jen doporučení. Co se týče vnější části lidského těla, díky plastickým operacím je možné změnit antropometrii člověka za účelem dosažení „standardu



Obrázek 2.4: Ukázka Bertillonova záznamu (Francis Galton) z roku 1893. Převzato z [10].

krásy“, který již v minulosti popsal například Leonardo Da Vinci jako zlatý řez. Antropometrie se ale nezabývá jen vnější částí lidského těla, ale i vnitřní částí, jako jsou kosti a orgány. K tomuto jsou využívány pokročilé měřící přístroje jako rentgen, počítačová tomografie (CT) nebo magnetická rezonance (MRI). Díky těmto přístrojům je možné lépe určit problém, který nemusí být ani viditelný, a následně zvolit metodu léčby. Díky těmto možnostem měření a následně lepší léčby se celkově zlepšuje a prodlužuje kvalita života.

2.2.3 Návrh nových zařízení

Následující sekce vychází z [22, 19]. Jedním z dalších využití antropometrie nebo již existujících změřených antropometrických hodnot je návrh nových zařízení, nábytku, oblečení nebo třeba obecně pracovního místa. U návrhu se můžeme zabývat následujícími případy, kde jsou příklady uvedeny se zaměřením na výšku člověka:

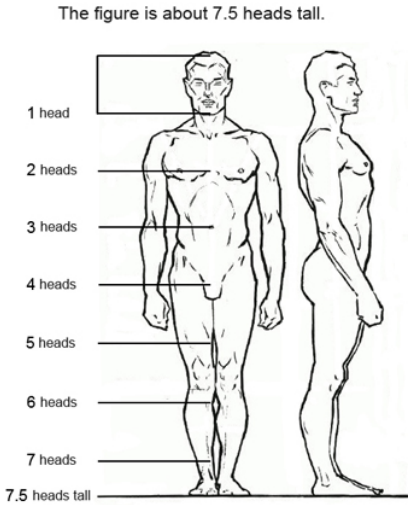
1. Design pro průměr - Návrhář se zaměřuje pro „největší“ skupinu s průměrnou výškou.
2. Design pro podprůměr - Návrhář se zaměřuje pro skupinu nižších lidí.
3. Design pro nadprůměr - Návrhář se zaměřuje pro skupinu vyšších lidí.
4. Design pro více typů - Je navrženo více typů (například velikosti S, M, L).
5. Design s nastavením - Navrhnutí s nastavitelnými parametry, například židle s nastavitelnou výškou.
6. Design pro všechny - Návrhář se snaží udělat výrobek, vyhovující značné většině.
7. Procrustus - Zaměření na uživatele.
8. Ego - Návrhář navrhuje podle své postavy.

Design dle lidské postavy je velice důležitý. Co se týče oblečení, je asi samozřejmé, že špatná velikost není vhodná či pohodlná k nošení, ale špatný design může mít horší následky. Pracovní místo se špatnou ergonomií může kromě nepohodlí vést k dlouhodobým problémům například s páteří. Na obrázku 2.5 lze vidět ukázku pracovního místa a jeho souvisejících metrik za účelem ideální ergonomie. Od těchto metrik se poté odvíjí jak doporučení například držení těla, ale také návrh nábytku, jako výška stojanu u monitoru, výška stolu, židle a dalších parametrů. Co se týče nevyhovujícího návrhu ochranných prostředků, tak nemusí uživatele ochránit a může to vést ke zranění. Stejně tak nevyhovující návrh místa v těžké technice, například v bagru, může vést ke slepým místům a zase vést ke zranění spolupracovníků. Proto je vždy potřeba u návrhu těchto zařízení dbát na rozměry lidského těla. K tomuto návrhu se využívá softwaru, kde je možné si vytvořit 3d postavu člověka s danými rozměry a vůči tomuto modelu následně navrhnout správné zařízení. Další možností je také simulace pracovního místa, kde může být sledován pohyb člověka a následně analyzováno zatížení na části těla, například zatížení na zápěstí.

Obrázek 2.5: Ukázka metrik pracovního místa u počítače. Převzato z [2].

2.2.4 Umění

Antropometrie, nebo spíš samotné měření lidského těla má počátky v umění. Malíři a sochaři se snaží, aby jejich výsledné dílo bylo realistické, a k tomu je potřeba znát rozměry lidského těla a poměry mezi jednotlivými částmi, jako jsou ruce, nohy nebo hlava. Při nedodržení těchto poměrů může poté být struktura těla abnormální, což někdy je žádoucí a je tento efekt zapříčiněn úmyslně. Většinou je ale snaha, aby lidské tělo vypadalo co nejvíce realisticky. Jedna z metod je odvození ostatních částí dle hlavy. Toto lze vidět na obrázku 2.6, kde výška postavy je brána jako 7,5x výška hlavy, výška vrchní poloviny postavy poté jako 4 hlavy a délka nohou jako zbývající 3,5 hlavy. Lze zde také odvodit umístění zápěstí, délka rukou a obecně ostatní body na lidském těle vůči hlavě.



Obrázek 2.6: Poměry u lidské postavy vůči hlavě. Převzato z [11].

2.2.5 Informatika

V poslední době se obor informatiky stává více a více populární a rozšířený vůči ostatním oblastem. Toto rozšíření sahá i do oblasti antropometrie, ať už se jedná o využití antropometrických dat, či jejich získání nebo vylepšení. Jedním z možným využitím je například návrh nových zařízení, za pomocí softwaru s importovanými antropometrickými daty, jak bylo popsáno v předcházející sekci Návrh nových zařízení 2.2.3. Dalším využitím můžou být biometrické systémy. Nebo co se týče zisku a vylepšení dat, tak může být využito strojového učení.

Biometrické systémy

Jak již bylo naznačeno v sekci Kriminalistika 2.2.1, antropometrie je také využívána k identifikaci jedince, k čemuž jsou využívány biometrické systémy. K identifikaci je možné využívat různé antropometrické hodnoty, jako může být tvar ruky, tvar žil na ruce, otisk prstu, sítnice a další možné, které jsou vůči jedinci jedinečné. Různé způsoby identifikace s sebou nesou různé výhody a nevýhody. Ze zmíněných například tvar ruky není nejbezpečnější k identifikaci, ale je jednoduchý na použití. Na druhou stranu sítnice bude k identifikaci bezpečnější, ale snímání sítnice je značně problematičtější.

Strojové učení

Následující sekce vychází z [7]. Antropometrická data jsou také využívána u strojového učení. Kde může být využito strojového učení k získání samotných dat, například extrakcí z obrazu. Nebo může být využito již získaných dat, jako je případ hledání korelací mezi daty, či sledování pohybu u fyzických aktivit a jejich následná analýza, kde se může sledovat pro příklad podání u tenisu či běh na trati. Tyto data lze extrahovat, jak již bylo zmíněno z obrazu nebo může být využito senzorů umístěných na lidském těle, jako jsou třeba sledovací náramky. Pro získání stejných dat z obrazu, jako z náramků, je nejdříve potřeba v daném obrazu nalézt tyto body, kde by se teoreticky náramky nacházely a následně tato data extrahovat, což oproti použití sledovacích náramků je pracnější. Na druhou stranu je

tento přístup více přívětivý pro uživatele, kde stačí pořídit video záznam, který je následně zpracován a není potřeba pořizovat další zařízení jako sledovací náramky. Tato data mohou být poté použita v případě analýzy tenisového podání k různým účelům, ať už jen jako získání dalších dat, tak k vylepšení podání daného hráče, či sledování zátěže na tělo.

Kapitola 3

Neuronové sítě

V této kapitole je v první části 3.1 popsán základní princip funkčnosti neuronových sítí, včetně základního modelu neuronu, sdružování neuronů do vrstev a jak probíhá učení neuronových sítí se zaměřením na regresi, což je víceméně hledání závislostí mezi proměnnými a jejich predikce. V druhé části 3.2 jsou poté zmíněna možná řešení detekce bodů na lidském těle.

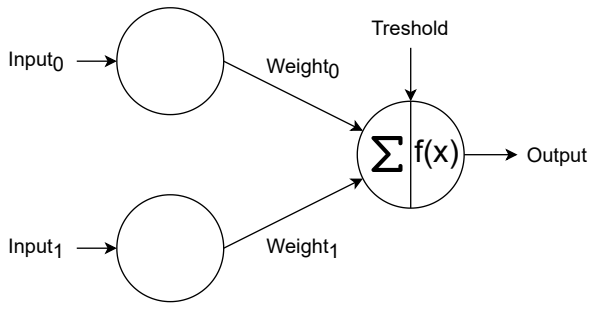
3.1 Princip

Následující sekce vychází z [12, 20]. Neuronové sítě jsou, jak z názvu vyplývá sítě složené z neuronů, a jsou tedy založeny na biologickém modelu neuronových sítí, jako je mozek. Na základě biologického modelu neuronu vznikl první model pro umělý neuron a to perceptron.

3.1.1 Perceptron

Perceptron vznikl v roce 1957 na základě biologického neuronu a obsahoval již učící pravidla. Princip spočívá v nastavení logického výstupu 0 či 1 na základě několika vstupů a jejich příslušných vah viz vzorec 3.1 a jedná se tedy o binární klasifikátor.

$$output = \begin{cases} 0 \; if \sum_{i=0}^{N} weight_i input_i \leq treshold \\ 1 \; if \sum_{i=0}^{N} weight_i input_i > treshold \end{cases} \tag{3.1}$$



Obrázek 3.1: Ukázka perceptronu s 2 vstupy.

Na obrázku 3.1 lze vidět jednoduchý perceptron s 2 vstupy. Pro jednodušší pochopení funkčnosti je zde následující příklad. Mějme perceptron, který se nám snaží rozhodnout

zda si vzít s sebou deštník a mějme 2 vstupy, dle kterých se rozhodujeme a to, zda plánuji jít pěšky (input_0) a zda má pršet dle předpovědi počasí (input_1). Následně se výsledné rozhodnutí odvíjí od zvolených vah a hraniční hodnoty. Pro příklad si můžeme zvolit obě váhy jako hodnotu 2 pro lehčí ilustraci. Následně pokud zvolíme hraniční hodnotu 1, tak bude rozhodnutí kladné, ať už plánuji jít pěšky nebo dle předpovědi bude pršet nebo obě možnosti zároveň. Pokud bychom ale zvolili hraniční hodnotu 3, tak už musí být splněny obě vstupní podmínky, abych si s sebou vzal deštník. Pokud bychom ale zvolili různé váhy, pro příklad weight_0 jako hodnotu 1 neboli váha vstupu, zda plánuji jít pěšky a weight_1 jako hodnotu 2 neboli váha vstupu předpovědi, zda bude pršet a hraniční hodnotu jako 1. V tomto případě nebude záležet na první vstupní hodnotě, při druhém vstupu máme možné hodnoty 0 a 2, a při první 0 a 1, kde se dostáváme k čtyřem možnostem a to 0, 1, 2 a 3. Zde hraniční hodnotu přesahuje pouze 2 a 3 a u obou těchto hodnot musí být druhá vstupní hodnota kladná a na první vstupní hodnotě nezáleží.

Zvolením těchto vah a hraničních hodnot můžeme získat různé modely rozhodování, samozřejmě zde máme pouze 2 vstupy, ale perceptron může mít více vstupů a mít následně složitější logiku rozhodování. Pro složitější logiku rozhodování nám ale nebude stačit 1 perceptron a proto se perceptrony spojují do vrstev. Na popsaném vzorci 3.1 lze sice jednoduše pochopit princip, není to ale vzorec, který se běžně uvádí. K tomu je potřeba tento vzorec mírně upravit. První úprava je nahrazení *weight* za w, *input* za x a využití skalárního součinu, kde se nám následně zápis $\sum_{i=0}^{N} weight_i input_i$ zjednoduší jako $w_i \cdot x_i$. Druhá úprava je poté využití *bias* (označený v rovnici jako b) místo *treshold*, jako $bias = -treshold$, kde se dostaneme na vzorec 3.2, se kterým se lze běžně setkat v literatuře. Tento vzorec se dá také nazvat aktivační funkcí, kde zde se jedná o *step* funkci.

$$output = \begin{cases} 0 \; if w_i \cdot x_i + b \leq 0 \\ 1 \; if w_i \cdot x_i + b > 0 \end{cases} \tag{3.2}$$

3.1.2 Aktivační funkce

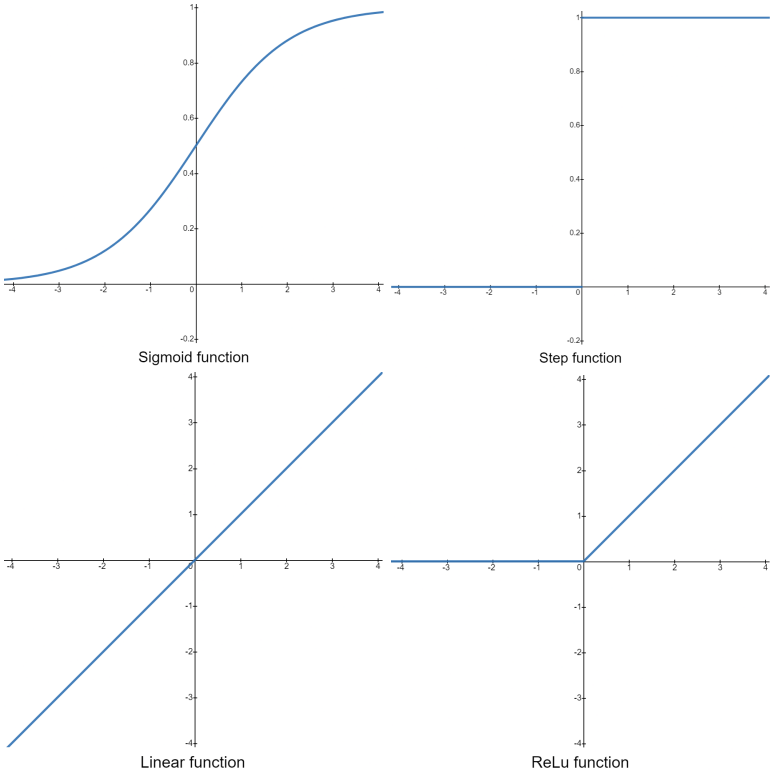
V případě že máme rozhodovací logiku pomocí sítě perceptronů, pravděpodobně chceme tuto logiku naučit se rozhodovat dle našich představ. Učení je víceméně upravení vah, případně bias, kde při malé změně, pravděpodobně očekáváme malou změnu na výstupu. Problém u perceptronů je, že i tato malá změna váhy, může způsobit změnu stavu z 0 na 1 a tato skoková změna může v celé síti změnit výstup více než jen trochu. Řešení je využití jiné aktivační funkce, kterou má například jiný model neuronu, a to sigmoid neuron. Sigmoid neuron se od perceptronu liší možnými vstupy a výstupy, kde u perceptronu je to pouze 0 nebo 1, zatímco u sigmoid neuronu to je jakákoliv hodnota mezi 0 a 1. Co se týče struktury, tak zde moc rozdílů není, obsahuje také vstupy, váhy a bias, stejně jako perceptron na obrázku 3.1 s rozdílem výstupní funkce, která je vyjádřena jako 3.3.

$$output = \sigma(w_i \cdot x_i + b) \tag{3.3}$$

Kde σ funkce je vyjádřena jako 3.4.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

Pro lehčí představu bude lepší graf aktivačních funkcí na obrázku 3.2. Kde horní levou sigmoid funkci využívá sigmoid neuron a pravou horní step funkci využívá perceptron.



Obrázek 3.2: (vlevo nahoře) sigmoid funkce (vpravo nahoře) step funkce (vlevo dole) linear funkce (vpravo dole) relu funkce

Další aktivační funkce

Další důležité aktivační funkce jsou například softmax využívající se u klasifikace nebo lineární a ReLu, běžně využívané jako finální aktivační funkce u regrese (mimo jiné ReLu se využívá také u konvolučních vrstev).

Softmax funkce se využívá u klasifikace a je vyjádřena jako 3.5 [30].

$$\sigma(\vec{Z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.5}$$

Kde $\vec{Z}$ značí vstupní vektor skládající se z Z_0 až Z_K, K značí počet výstupních tříd a Z_i jsou jednotlivé hodnoty vektoru, jako reálné číslo. Tato funkce na výstupu určuje pravděpodobnost dané třídy v rozmezí 0-1, kde součet všech pravděpodobností se rovná 1.

Lineární a ReLu jsou poté využívány u regrese nebo u konvoluce a jejich vyjádření lze vidět na 3.6.

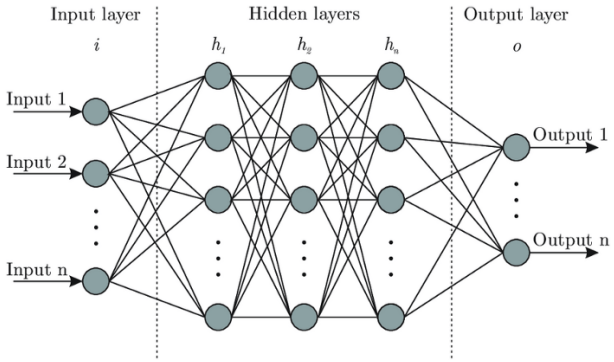
$$linear(x) = x \qquad relu(x) = max(0, x) \tag{3.6}$$

Lineární funkce je v rozsahu reálných čísel a ReLu v rozsahu kladných reálných čísel. Průběh těchto funkcí lze vidět na obrázku 3.2 ve spodní části.

3.1.3 Vrstvy

Následující sekce vychází z [25]. Jak již bylo řečeno dříve, jeden umělý neuron není dostačující pro rozhodovací logiku, a proto se spojují do vrstev, které se následně spojují mezi

sebou. Základní strukturu neuronové sítě lze vidět na obrázku 3.3, kde jsou vrstvy rozděleny do 3 kategorií: vstupní, skryté a výstupní. Vstupní vrstva obsahuje vstupní neurony, pomocí kterých se dostávají data do neuronové sítě. Například pro tabulku to může být počet sloupců, pro obraz to může být počet pixelů. Výstupní vrstva zase na druhou stranu obsahuje výstupní neurony, které značí celkový výstup neuronové sítě. Pro regresi zde většinou bývá jeden neuron vůči jedné predikované hodnotě, u ceny to například bude 1 neuron, v případě bodu 2, jeden pro x a druhý pro y. Vždy záleží na požadovaném výstupu. Skryté vrstvy jsou poté jednoduše ty vrstvy, které nejsou ani vstupní ani výstupní. Na rozdíl od vstupní a výstupní vrstvy, návrh této části bývá složitý, protože zde není univerzální postup.



Obrázek 3.3: Ukázka vrstev neuronové sítě. Převzato z [25].

Další možné dělení by bylo na *feedforward* a *recurrent* neuronové sítě. Kde *feedforward*(dopředná) síť je častěji využívána a jedná se o případy popsané doposud, kdy data postupují zleva doprava a vstup další vrstvy je pouze výstup předchozí vrstvy. U *recurrent* (obsahují smyčku) sítí mohou být vstupem i výstupní data dané vrstvy (z minulého kroku) a tvořit tak smyčku, mohou být využity například pro analýzu rukopisu nebo řeči. Obecně je tento druhý typ sítí složitější a méně využívaný.

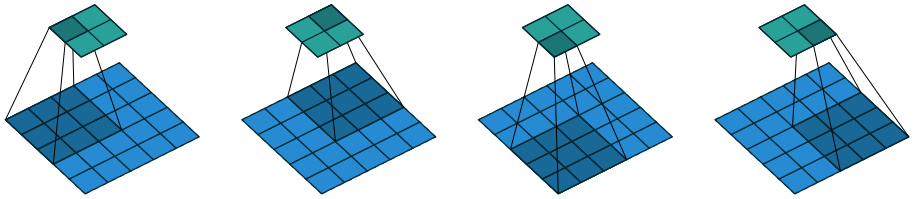
Konvoluční a pooling vrstva

Následující sekce vychází z [23]. Konvoluční a *pooling* vrstvy jsou využívány u zpracování obrazu a odlišují se od doposud popsaného přístupu, z důvodu, že rozměry obrazu bývají dostatečně vysoké. Bylo by příliš výpočetně náročné mít na vstupu všechny pixely, i když u velice malých obrázků by to možné bylo.

Konvoluční vrstva, jak z názvu vyplývá, funguje na principu konvoluce. Kde je využito filtrů (*kernels*) pro zvýraznění důležitých vlastností v obraze, jako může být detekce hran. Zároveň dle typu filtru může být zredukován rozměr. Mezi důležité vlastnosti patří velikost filtru, *stride* (velikost kroku posunu) a *padding* (typ vycpání). Velikost filtru určuje velikost konvoluční matice a udává počet hodnot zpracovaných v jednom kroku, ku příkladu u matice 3×3 je vypočtena hodnota z 9 pixelů v jednom kroku.

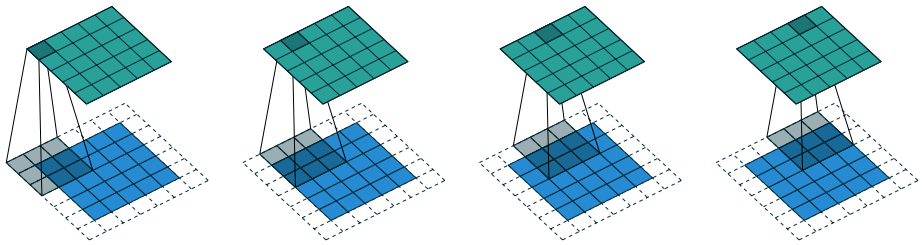
Stride neboli velikost kroku poté určuje o kolik hodnot se v každém kroku matice posune. Základním krokem bývá 1, kde pro příklad máme vstup 5×5 a konvoluční matici 3×3, pokud budeme sledovat pouze první řádek, tak jsou postupně zpracovány hodnoty 1-3, 2-4 a 3-5, což jsou 3 výstupní hodnoty a celkový výstup je poté 3×3. V případě *stride* 2, a stejného vstupu i matice jsou v prvním řádku zpracovány pixely 1-3 a poté 3-5, vycházející

na celkový výstup 2×2, *stride* se tedy dá také použít k redukci rozměru, tato ukázka lze vidět na obrázku 3.4.



Obrázek 3.4: Průběh konvoluce s krokem 2. Převzato z [9].

Padding poté mívá 2 možnosti a to *same* a *valid*. *Valid* znamená neupravení vstupu neboli bez žádného vycpání, jako tomu bylo například v příkladu u *stride*. Typ *same* poté znamená vycpání okrajů nulami neboli je vstup rozšířen z každé strany o nuly. Pokračující ve stejném příkladu, kdy je vstup 5×5, tak s *padding* typu *same*, jsou zpracovány data o velikosti 7×7. U konvoluční matice 3×3 jsou poté tedy zpracovány hodnoty 0-2, 1-3, 2-4, 3-5, 4-6, kde získáme 5 hodnot v prvním řádků a výsledkem je poté výstup 5×5, *padding* typu *same* se tedy většinou používá, pokud chceme zachovat stejnou velikost na výstupu jako byl vstup. Popsaný příklad[1] lze poté vidět na obrázku 3.5.



Obrázek 3.5: Průběh konvoluce s padding typu same. Převzato z [9].

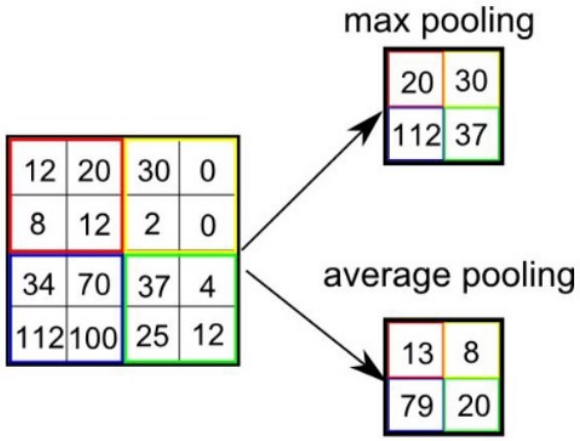
Pooling vrstva se poté stará pouze o redukci rozměru s možností výběru maximální nebo průměrné hodnoty v daném okolí. Na obrázku 3.6 lze poté vidět princip *pooling* vrstvy, stejně tak jako u konvolučních vrstev se zde definuje velikost a krok, kde právě na ukázce je velikost 2×2 a krok 2.

Na obrázku 3.7 lze poté vidět, jak může vypadat struktura konvoluční neuronové sítě pro klasifikaci, obsahující konvoluční vrstvy, *pooling* vrstvy a plně propojené vrstvy. Jako aktivační funkce v poslední vrstvě je zde softmax, kvůli klasifikaci, aby síť řešila regresní úlohu, je potřeba zde změnit aktivační funkci ze softmax na například lineární.

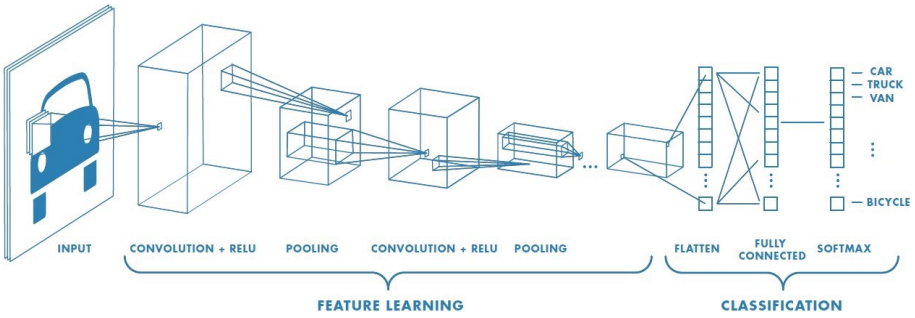
3.1.4 Učení

Aby neuronová síť mohla být využita pro konkrétní problém, je potřeba mít správně nastavené váhy a bias. Tyto parametry se nastavují pomocí učení neuronových sítí, za pomocí vstupních dat (datasety). Většinou se dataset dělí do 3 skupin. Trénovací dataset, na kterém je model trénován a jsou pomocí něj nastaven váhy a bias. Dále validační dataset pro

[1] Příklady průběhu konvoluce lze nalézt na https://github.com/vdumoulin/conv_arithmetic v animované verzi.



Obrázek 3.6: Ukázka funkce pooling. Převzato z [23].



Obrázek 3.7: Ukázka struktury konvoluční neuronové sítě pro klasifikaci. Převzato z [23].

nastavení *hyper parametrů*, jako *learning rate*, počet skrytých vrstev atd. A jako poslední je testovací dataset, vůči kterému je prováděno finální měření. Nastavení parametrů vah a bias, dle trénovacího datasetu provádí stroj, a to většinou za pomocí algoritmu *backpropagation*.

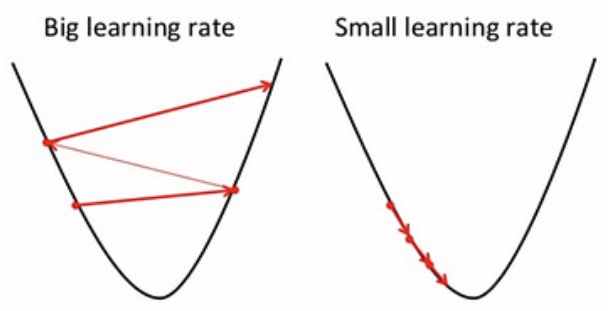
Backpropagation

Následující sekce vychází z [16]. Ještě před *backpropagation*, je potřeba popsat *forward propagation*. Jak již z názvu vyplývá, jedná se o postup dat směrem kupředu, data postupují od vstupu, přes skryté vrstvy až k výstupu. Pro příklad si můžeme představit, že máme v trénovacím datasetu dvojici hodnot (x,y), kde x bude použito jako vstup sítě a y je očekávaný výstup. Toto x postupně projde sítí, až se na konci objeví výstup, tento výstup poté porovnáme vůči y z datasetu. K tomuto porovnávání se používá tzv. *cost* funkce (nebo také nazývána *loss* funkce), na základě které se upravují parametry neuronové sítě a této úpravě parametrů se právě věnuje *backpropagation*, kde data postupují přesně opačně než u *forward propagation* a to postupně přes vrstvy „zprava doleva“. Jinak řečeno *backpropagation* se snaží upravovat váhy a bias sítě tak, aby minimalizoval *cost* funkci. Jedním z příkladů této *cost* funkce může být *Mean squared error* (MSE 3.7), která patří k nejvyužívanějším, co se týče regresních úloh. Kde se jedná víceméně o průměr součtu druhých mocnin rozdílu mezi očekávanou hodnotou a reálnou hodnotou.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{3.7}$$

Gradient Descent

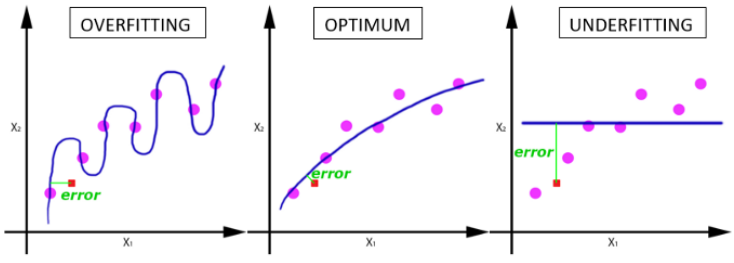
Když máme vypočítaný rozdíl mezi očekávanou a reálnou hodnotou, je potřeba upravit parametry sítě, k čemuž se využívá optimalizátor (neboli optimalizační algoritmus). Nejzákladnější optimalizátor je Gradient Descent, využívaný jak u regresních úloh, tak u klasifikace, který funguje na principu derivace. Na základě derivace *loss* funkce, je vypočten směr, kterým by se měly upravit parametry sítě, aby bylo dosaženo minima. Důležitá je také hodnota *learning rate*, která je jako globální hodnota dopředně nastavena a určuje rychlost konvergence k minimu. Na obrázku 3.8 lze vidět případy nastavení příliš vysokého *learning rate*, což může způsobit přeskočení minima nebo nastavení příliš nízkého *learning rate*, což způsobí příliš dlouhé hledání minima. Vždy je potřeba zvolit kompromis.



Obrázek 3.8: Ukázka *learning rate*. Převzato z [14].

Epocha, batch a iterace

Následující sekce vychází z [24]. Související s Gradient descent jsou pojmy jako epocha a *batch* (případně iterace), protože je Gradient descent iterativní algoritmus, je potřeba aktualizovat váhy a bias několikrát, než je docíleno ideálního výsledku. Jedno projití datasetu, skrz neuronovou síť, se nazývá epocha. Protože je Gradient descent iterativní, nestačí nám 1 epocha, ale je potřeba dataset projít vícekrát. Na obrázku 3.9 je možné vidět případy zleva doprava, kdy bylo epoch příliš a model byl přeučen pro daný dataset. Dále ideální případ. A následně nedostačující počet epoch, kdy se model ani nezvládl naučit. Bohužel neexistuje vhodná hodnota pro počet epoch a je potřeba tuto hodnotu zvolit dle toho, jak moc rozmanitý je dataset.



Obrázek 3.9: Ukázka možného dopadu počtu epoch na natrénování modelu. Převzato z [24].

Epocha se následně může dělit na *batche*, kde *batch* je část datasetu, protože zpracování celého datasetu zároveň by pro počítač mohlo být příliš náročné. Iterace následně určuje

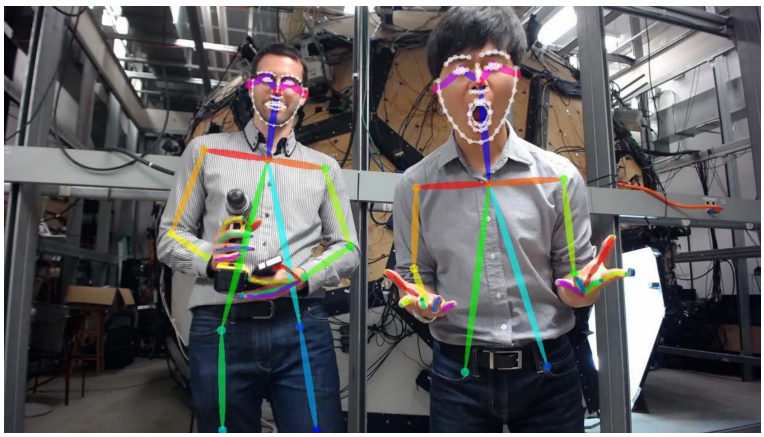
na kolik dílů je dataset rozdělen a kolik „kroků“ je potřeba provést, aby byla dokončena 1 epocha. Tento výpočet je možné vidět na 3.8, jedná se o velikost datasetu vydělenou velikostí *batch*.

$$Steps = \frac{DatasetSize}{BatchSize} \tag{3.8}$$

3.2 Neuronové sítě pro detekci bodů těla

Následující sekce vychází z [21]. Související s detekcí chodidel je detekce významných bodů na těle. Touto problematikou se převážně zabývají řešení odhadující pózu člověka, kterých je možné najít poměrně hodně. Jedním z nich je například OpenPose, kde lze na obrázku 3.10 vidět, jak vypadají detekované body v obraze. Většina z těchto řešení nemá problém, pokud lze v obraze vidět pouze část lidského těla. Stejně jako lze předpokládat u této práce, že budou nohy v obraze zakryté. Zmíněná řešení, ale řeší celkově zakrytou část těla vynecháním těchto bodů, kde cílem této práce je chybějící body (chodidla), predikovat. Obecně se detekce bodů dělí na 2 základní metody přístupu:

- Bottom-up - prvně odhadnutí jednotlivých bodů na těle a následné spojení v pózu
- Top-down - první proběhne detekce a ohraničení lidského těla, kde následně v tomto ohraničení jsou hledány body



Obrázek 3.10: Ukázka detekce bodů OpenPose. Převzato z [4].

Pro ukázku jsem zde vybral některá novější a známější řešení detekce bodů na lidském těle, mezi které patří OpenPose, Detectron2 a PoseNet. Těchto řešení ale existuje mnoho. Za zmínku by určitě ještě stálo DeepPose, AlphaPose, UniPose či DeepCut, které také patří k poměrně známým. Značná část řešení využívá pro detekci 17 bodů dle datasetu COCO[2], vůči kterému se také například provádí měření přesnosti.

3.2.1 OpenPose

Následující sekce vychází z [4]. OpenPose je populární open-source řešení, určené pro detekci bodů na lidském těle spadající pod Carnegie Mellon University z roku 2019 trénovaný na

[2]COCO dataset: https://cocodataset.org/#keypoints-eval

datasetu od CMU[3], který je dostupný pro vědecké účely. Jedná se o bottom-up detekci v reálném čase s vysokou přesností. Je zde několik možností, které body mají být na těle detekovány. A to s možností rozšířením počtu detekovaných bodů u obličeje, rukou nebo chodidel, kde „základní“ detekce obsahuje body dle COCO datasetu. I když se jedná o řešení v reálném čase, je toto řešení stále náročné na výpočetní výkon a je možné dosáhnout zpracování v reálném čase pouze na GPU. Zpracování jednoho snímku na GPU trvá asi 36 ms, zatímco u CPU varianty kolem 10 s. Je možné ale využít odlehčené verze, která běží v reálném čase i na běžném CPU.

3.2.2 Detectron2

Následující sekce vychází z [31]. Detectron2 je řešení z roku 2019, pochází od společnosti Facebook a je založen na původním Detectron s rozšířenou funkcionalitou a rychlejším učením. Detectron2 je zaměřen na více druhů detekce, jako je detekce objektů, jejich klasifikace nebo také detekce bodů na lidském těle, dle COCO datasetu. Dle jejich vlastního benchmarku, dosahuje Detectron2 až dvojnásobného zpracování snímků vůči porovnaným řešením. Tento benchmark byl ale proveden na vyhledání a rozdělení obrazu dle objektů, takže není jisté, zda je rychlejší i co se týče detekce bodů na těle.

3.2.3 PoseNet

Následující sekce vychází z [26]. PoseNet, stejně jako související MoveNet a BlazeNet (detekující další možné body navíc) jsou modely spadající pod TensorFlow. Kde právě PoseNet detekuje, jako zmíněné modely výše, 17 bodů dle COCO datasetu. PoseNet je zaměřený převážně na zařízení s nižším výpočetním výkonem, a tak dosahuje vysoké rychlosti zpracování s ale nižší přesností než doposud zmíněné.

[3]CMU dataset: http://domedb.perception.cs.cmu.edu/

Kapitola 4

Návrh řešení

Následující kapitola je věnována návrhu, kde první část 4.1 se zabývá výběrem vhodného nástroje pro realizaci. V následující části 4.2 se poté řeší zisk datasetu a jakým způsobem budou získány vstupní snímky. Dále jakým způsobem je možné tyto snímky upravit a zvolený způsob úpravy zakrytí nohou. A poslední část 4.3 je poté věnována použité neuronové síti. Jelikož se jedná o zpracování obrazu, tak se bude jednat o konvoluční neuronovou síť s aktivační funkcí na výstupu vhodné k regresi jako je lineární, jak bylo zmíněno v předchozí kapitole 3.1.3.

4.1 Programovací nástroje

Vzhledem k rostoucí popularitě umělé inteligence je zde velký výběr programovacích jazyků a následně použitého rozhraní. Mezi nejpopulárnější programovací jazyky, v oblasti strojového učení, patří Python, C/C++, Java, R a JavaScript. Z těchto jazyků je aktuálně nejpopulárnější Python, díky značnému množství knihoven, jednoduché syntaxi, velké komunitě a mnoho dostupných návodů. Z těchto důvodů bude u realizace využito jazyku Python. Kromě programovacího jazyku je také potřeba vybrat knihovnu pro implementaci neuronové sítě. Mezi nejpopulárnější patří Tensorflow, MXNet, CNTK a PyTorch, kde právě Tensorflow a PyTorch je věnována značná pozornost. Proto se rozhoduji mezi použitím některé z nich a následující část je věnována jejich popisu, vycházející z [3].

4.1.1 Tensorflow

Tensorflow je open-source knihovna spadající pod společnost Google s uvedením v roce 2015. Tensorflow využívá statické definice výpočetního grafu před samotným výpočtem (v pozdější verzi podporuje také dynamický výpočet), díky čemuž je následně složitější debugování, kde je potřeba využít jejich vlastní *utility tfdm* namísto běžných debugovacích prostředků pro Python. Pro vizualizaci modelu, sledování učení, profilování a další funkcionality je možné využít Tensorboard. Tensorflow také poskytuje možnosti pro vývoj modelů na vyšší úrovni nebo podporu pro mobilní zařízení (Tensorflow lite). Obecně je Tensorflow považován za více vyspělou knihovnu a je více využíván pro reálné nasazení.

Keras

Keras je knihovna, sloužící jako vysokoúrovňové rozhraní Tensorflow knihovny, která má sloužit jako více uživatelsky přívětivá varianta se zaměřením na snadnější použití v porovnání vůči Tensorflow.

4.1.2 PyTorch

PyTorch je open-source knihovna, která vyšla o rok později než Tensorflow (tj. 2016) a spadá pod společnost Facebook. PyTorch již od základu využívá dynamického výpočetního grafu, díky čemuž je možné provést modifikace za běhu. PyTorch je obecně více vhodný pro vytvoření vlastního modelu a díky možnosti větší interaktivity je více využíván k vědeckým účelům. Co se týče debugování, tak je zde možné využít běžných nástrojů určených pro Python. U vizualizace je poté možné použít například Visdom, který ale obsahuje značně méně možností než Tensorboard.

Na základě předchozích informací, jsou tyto knihovny víceméně vyrovnané v možnostech vůči sobě, a proto jsem se rozhodl využít Tensorflow, přesněji tedy jeho rozhraní Keras. Převážně proto, že Tensorflow byl více populární a lze tedy očekávat, že zde bude možné najít více informací.

4.2 Získání datasetu

Aby byla síť schopna odhadnout pozici chodidel v obraze, a to i v případě kdy nohy nebudou viditelné, je potřeba k tomu mít odpovídající dataset. Některé možné existující datasety již byly zmíněné v 3.2, přesněji CMU a COCO dataset, kde právě část COCO datasetu se zaměřením na *keypoints* je značně využívána při trénování neuronových sítí pro detekci bodů na lidském těle. Problémem je, že tyto datasety jsou zaměřeny na trénování sítí pro detekci bodů viditelných v obraze než na predikci bodů, které jsou zakryté.

Jednou z možností je úprava zmíněných datasetů. Cílem této práce je detekce chodidel, kde existující datasety se zaměřují na detekci všech významných bodů. V případě COCO datasetu jich je například 17, zatímco účelem této práce je detekovat pouze dva z těchto bodů, a to chodidla. První modifikace datasetu by tedy byla zachování pouze těchto dvou zmíněných bodů a odstranění zbývajících. Následná modifikace se bude zabývat zakrytím nohou, případně vybráním snímků, kde jsou nohy alespoň částečně zakryté a stále jsou anotované. Díky čemuž by se neuronová síť měla být schopna naučit odhadnout pozici nohou na základě ostatních bodů těla, a ne na základě rozpoznání chodidel v obraze.

Další možností je poté místo modifikace existujícího datasetu vytvoření kompletně nového datasetu. K tomuto účelu je možné využít již existující řešení pro detekci bodů na lidském těle, ku příkladu OpenPose, který dosahuje vysoké přesnosti. Z detekovaných bodů by poté byly využity detekovaná chodidla a následně by bylo potřeba vstupní obraz upravit, aby byly chodidla nebo nohy zakryté tak, aby se síť zvládla naučit predikovat i zakrytá chodidla.

4.2.1 Úprava obrazu

Ať už při úpravě existujícího datasetu, tak při vytváření kompletně nového datasetu, v obou těchto případech nastává problém, kdy je potřeba obraz upravit, aby osoby na snímku měly zakryté nohy. Nejjednodušší varianta je poté překrytí nohou jednoduchým obrazcem, jako může být obdélník určité barvy. Což ve výsledku bude vypadat nerealisticky a také zde

vznikne ostrá hrana, na kterou by mohla „špatně“ reagovat neuronová síť, která by sice šla zredukovat, ale stále bude výsledná úprava nerealistická a příliš umělá. Složitější varianta je poté zakrytí nohou vybraným objektem. Kde nevznikne ani ostrá hrana a výsledná úprava nebude takto umělá a bude se více podobat reálným případům. Například v lese mohou nohy zakrývat rostliny nebo v kanceláři nábytek. Kde právě variantu zakrytí nohou vybraným objektem jsem se rozhodl použít. Tato úprava lze provést různými způsoby, které by šlo rozdělit na 3 možnosti úpravy, kterými jsou: manuální, poloautomatizovaná a plně automatizovaná.

Manuální úprava

Manuální úprava je asi nejvíce přímá možnost. Vstupní obraz upraví manuálně osoba s využitím, k tomu vhodnému programu, jako může být Photoshop. Tímto způsobem vznikne nejvíce realisticky vypadající výstup, kde při dobré úpravě může být výstup nerozpoznatelný od neupraveného obrazu. Tato metoda má poté nejjednodušší prvotní nastavení, kde je víceméně pouze potřeba spustit existující vhodný program, pomocí kterého je obraz upraven. Samotná doba úpravy poté záleží na kvalitě provedení a do jaké míry je potřeba, aby vstupní obraz vypadal co nejvíce reálně. Každopádně lze počítat minimálně s několika minutami na jednu úpravu.

Poloautomatizovaná úprava

Poloautomatizovaná úprava je poté lehce složitější možnost, která spočívá v kombinaci využití programu a manuálního zadávání. Samozřejmě i u manuální metody je využito vhodného programu, tento přístup se liší typem programu a poté složitostí úpravy. Program pro tento typ není pro běžné použití, jako je ku příkladu zmíněný Photoshop, ale jedná se o program, který slouží pravděpodobně pouze k vytvoření datasetu s velice omezenou funkcionalitou. Cílem je co nejsnadnější vytvoření datasetu, kde každá úprava má zabrat co nejmenší možný čas. V případě zakrytí nohou se může jednat o 3 kliknutí, kde se jedná o výběr jednoho z již předem připravených objektů určených k zakrytí nohou, další kliknutí umístění objektu na správné místo a jako poslední přepnutí na další snímek. Vzhledem k tomu, že vytváření datasetu není až tak neobvyklá záležitost, je možné najít již existující aplikace k tomuto určené. Každopádně nejčastější úlohou bývá klasifikace a je možné, že by zde nebylo možné přidat další objekt určený k zakrytí nohou. Každopádně s takto omezenou funkcionalitou by nebylo náročné vytvořit si vlastní aplikaci.

Samotná úprava poté v porovnání s manuální metodou zabírá zlomek času, kde každá úprava může zabírat několik sekund. Toto zrychlení je poté na úkor času investovaného do začátku, kde je potřeba získat a nastavit aplikaci, která k tomu bude vyžita. A dále také na úkor výsledné kvality. Aby byla úprava co nejjednodušší a nejrychlejší, lze očekávat, že bude na výběr jen omezený počet objektů k zakrytí, stejně tak zde nebude možné upravit jas, stínování a další detaily, které způsobí, že výsledek nebude vypadat úplně realisticky, každopádně stále dostatečně realisticky.

I když je zde značné zrychlení oproti čistě manuální metodě, stále je zde zahrnuta manuální činnost, značně zpomalující celý proces. A pokud by výsledný dataset měl být příliš rozsáhlý, nebylo by možné, nebo by bylo příliš časově náročně vytvořit dataset touto formou v jednom člověku. Je proto tedy časté využití větší skupiny lidí k vytvoření datasetu, čímž je značně redukován celkový čas.

Plně automatizovaná úprava

Plně automatizovaná úprava je poté nejsložitější, co se týče prvotního nastavení. Jak vyplývá z názvu, jedná se o plně automatizovaný proces neboli bez lidského zásahu pomocí programu či skriptu. Jelikož se jedná o velice specifický program určený přímo k této jedné dané činnosti, nelze očekávat, že bude možné najít vhodný program již hotový. Muselo by se jednat o specifický dataset, který již někdo vytvářel tímto způsobem a ve chvíli kdy již někdo takovýto dataset vytvářel, pravděpodobně bude také dostupný a odpadá nutnost vytvoření vlastního datasetu. Tedy za předpokladu, že dostupný dataset by dosahoval dostatečné velikosti. Samotná doba úpravy poté záleží pouze na složitosti daného programu a výkonu stroje, který úpravu vykonává. Přestože na základě těchto parametrů se rychlost může značně lišit, v případě zakrytí nohou se stále jedná o práci s obrazem a lze tak očekávat se zpracováním několika desítek až stovek snímků za sekundu.

Díky odpadnutí nutnosti manuální činnosti je tedy proces značně urychlen. Na druhou stranu lze očekávat značné snížení kvality, protože nelze porovnávat lidské rozhodování oproti předdefinované logice skriptu. Přestože se jedná pouze o jednoduchou činnost zakrytí nohou vybraným objektem, stále je zde spousta proměnných. Při porovnání vůči předchozí metodě, a to poloautomatizovaná úprava, kde může mít uživatel na výběr z několika různých objektů pro zakrytí nohou. Pravděpodobně vybere objekt, který nejlépe zapadne do okolí a bude výsledná úprava vypadat alespoň částečně realisticky. V případě plně automatizované úpravy by rozhodnutí, který objekt nejlépe zapadne do okolí, byla příliš složitá operace. Reálný výběr bude tedy pravděpodobně náhodný, což povede k víceméně nerealistickým úpravám.

Problémem mohou být také různé chyby či anomálie, jako může být snímek s daty, kde se údajně nachází osoba. Každopádně se na daném snímku nebude žádná osoba nacházet. V případě manuální úpravy je možné si této chyby všimnout a snímek vyřadit. Při plně automatickém zpracování to ale skript nemá jak detekovat a provede klasickou úpravu na chybném snímku. Dalším problémem je poté umístění samotného objektu. I když skript bude znát umístění chodidel, stále bude problém objekt umístit dostatečně univerzálně s ohledem na možné pózy. Na druhou stranu špatně zakryté nohy nemusí být nežádoucí, i když se jedná dataset pro odhad pozice zakrytých nohou. Stále lze totiž očekávat, že v reálné situaci nemusí být nohy zakryté po celou dobu a neuronová síť tak bude schopna řešit i tento případ.

Tento plně automatizovaný způsob úpravy jsem se také rozhodl využít při vytváření datasetu. Při rozhodování hraje asi největší roli požadovaná velikost datasetu. Manuální úprava je pro vytváření datasetu nevhodná, tímto způsobem by bylo náročné upravit již několik set snímků, což pro potřeby datasetu je málo. Poloautomatizovaná metoda by byla vhodná pro několik tisíc snímků, v případě desetitisíců snímků už by pravděpodobně bylo vhodné vytvářet dataset ve více lidech. Plně automatizovaná metoda víceméně nemá omezení na počet snímků. Díky tomuto je poté možné dataset vždy rozšířit o další snímky s minimální námahou, kde se víceméně jedná pouze o získání více vstupních dat a spuštění skriptu. Kde snadné rozšíření, a tedy příprava do budoucna byl hlavní důvod, proč jsem zvolil tento způsob.

4.2.2 Zisk dat

Ať už se jedná o jakýkoliv druh úpravy obrazu, vždy je potřeba dvojice snímků, a to snímek osoby a snímek s objektem, kterým budou zakryty nohy. Tyto snímky je potřeba někde získat a je potřeba aby tyto snímky byly dostatečně rozdílné. Pokud by byl dataset složen

ku příkladu z tisíce snímků jedné a té stejné osoby a zároveň by k zakrytí nohou byl využit jeden stejný objekt, nebyla by dosažena dostatečná variabilita a síť by se pravděpodobně nenaučila obecné řešení, ale řešení vůči této jedné osobě. A pokud by byla na snímku jiná osoba, tak by selhala. Stejně tak by síť pravděpodobně selhala ve chvíli, kdy by byly nohy zakryty jiným objektem, protože síť není naučena na obecné řešení. Proto je potřeba aby snímky byly dostatečně variabilní a nedošlo tak k přeučení vůči specifickému vstupu. Z čehož vyplývá, že je potřeba získat spoustu různých snímků, k čemuž jsem se rozhodl využít datasetu COCO.

Jak již bylo zmíněno dříve, tak se COCO dataset využívá k detekci bodů. Každopádně je to jeden z mnoha účelů, ke kterým lze využít. Mezi které dále například patří detekce, segmentace a slovní popis obrazu. Díky tomu, že je určen k více úlohám, obsahuje mimo snímky osob, také snímky pouze složené z objektů, které bude možné využít k zakrytí nohou. Zároveň bude možné využít zmíněné body na lidském těle, kde součástí výsledného datasetu budou využity body chodidel, které se bude snažit neuronová síť predikovat a zbylé body na lidském těle mohou být využity při úpravě obrazu k umístění a upravení velikosti vloženého objektu. Bude se tedy jednat o úpravu existujícího datasetu než vytvoření kompletně nového datasetu, každopádně bude každý snímek upraven zakrytím nohou a ve výsledku se bude jednat o jiný „nový“ dataset.

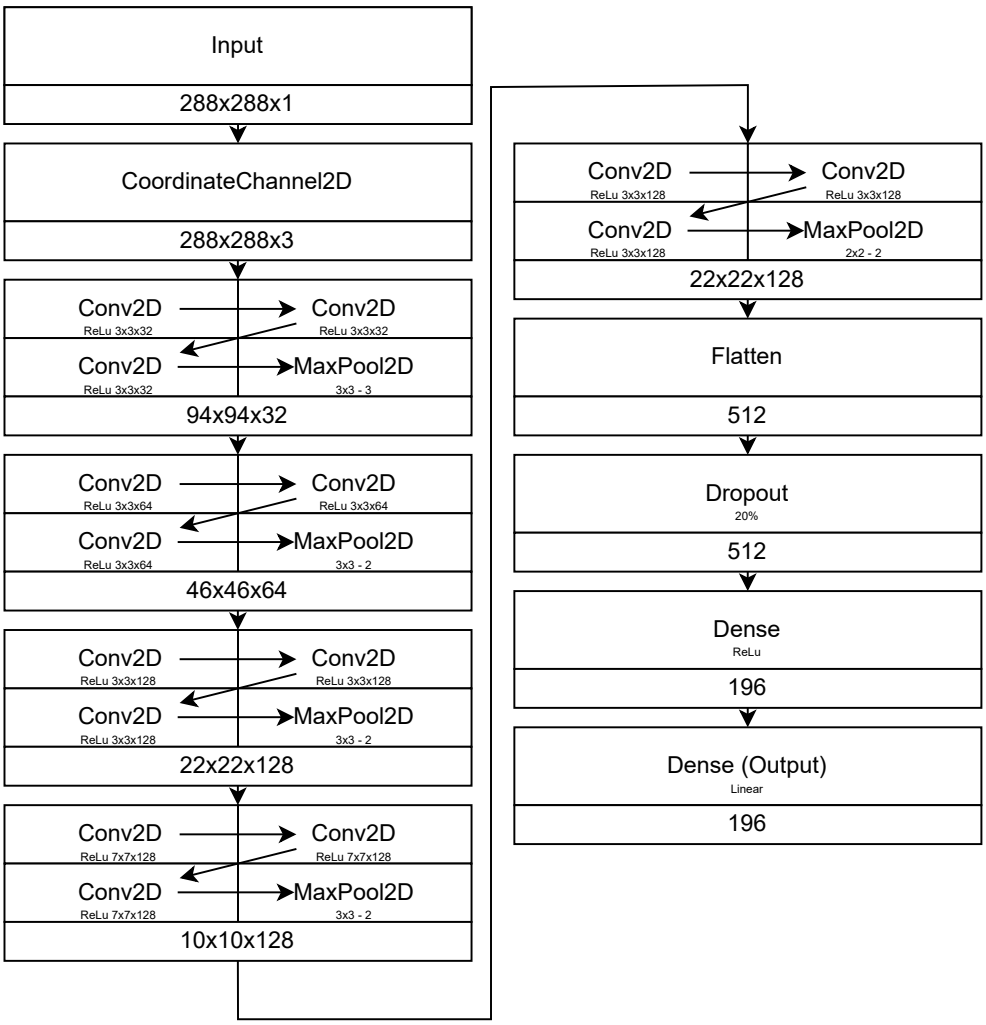
4.3 Neuronová síť

Jak již bylo zmíněno dříve, jelikož se jedná o zpracování obrazu, tak se bude jednat o konvoluční neuronovou síť. Základní princip jednoduché konvoluční neuronové sítě je postupná redukce velikosti vstupu pomocí konvolučních a pooling vrstev, a následné napojení na plně propojené vrstvy. Kde samotný návrh je složitý proces, a proto se většinou využívají již existující modely nebo se z nich vychází. V této práci budu vycházet z detekčního modelu od vedoucího práce, který lze vidět na obrázku 4.1.

Tato síť se skládá z pěti konvolučních bloků, kde jsou vždy 3 konvoluční vrstvy a následně *max pooling* vrstva, zakončené dvěma plně propojenými vrstvami. U většiny vrstev se jedná o základní vrsty popsané v 3.1.3, výjimku zde tvoří *Flatten*, *Dropout* a *CoordinateChannel2D*. *Flatten* se stará pouze o změnu dimenze, kde z nD dat vytvoří 1D data. *Dropout* poté náhodně nastaví zvolený počet vstupů na 0, v případě 0,2 se bude jednat o 20 % hodnot. Tato vrstva se využívá jako prevence proti přeučení. Všechny tyto vrstvy s výjimkou *CoordinateChannel2D* jsou dostupné v knihovně Keras. *CoordinateChannel2D* je poté speciální vrstva z externí knihovny[1] vycházející z publikace [18], která by měla přidávat konvoluci informaci o umístění daných pixelů.

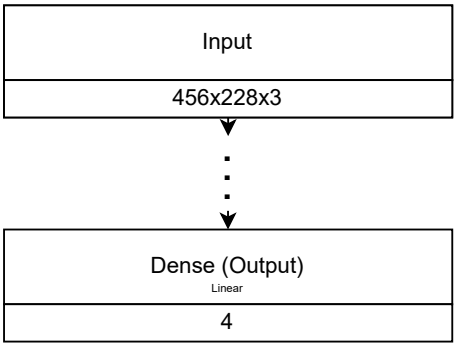
Každopádně tento model nelze přímo použít na detekci chodidel, kde minimálním důvodem je výstup sítě. Zde se nachází 196 hodnot a požadovaným výstupem jsou 4 hodnoty (x a y pro každé chodidlo). Bude se tedy jednat o predikci pouze jedné osoby, která se bude očekávat na vstupním snímku sítě. Tím vzniká požadavek na předzpracování obrazu, kde je potřeba, aby na vstupu sítě byly snímky obsahující pouze jednu osobu a budou dle této osoby ořezány. S čímž přichází druhá změna tohoto modelu, a to rozměry vstupu. Aktuálně model má na vstupu rozměr $288 \times 288 \times 1$. Jedná se tedy o čtvercový obraz o velikosti 288×288 s jedním kanálem, tedy v úrovni šedi. Nejčastější očekávanou pózou ve které se bude osoba nacházet bude pravděpodobně, kdy osoba stojí, a tedy čtvercový vstup by nebyl ideální, protože stojící osoba nemá výšku a šířku v poměru 1:1. Poměr výška:šířka rozměru

[1] Dostupné z https://github.com/titu1994/keras-coordconv



Obrázek 4.1: Architektura výchozího modelu. Konvoluční bloky jsou ve formátu velikosti a počtu filtrů. Pooling vrstvy ve formátu velikosti a velikosti kroku. Na konci každého bloku, či skupiny bloků je napsána aktuální velikost dat.

vstupu bude tedy upraven alespoň na 2:1, což se více přibližuje realitě. Bude zde také jako základní počet kanálů 3. Bude se tedy jednat o barevný vstup namísto úrovní šedi. S tímto poměrem vstupu mohou ale nastat problémy, kdy ku příkladu bude osoba ležet, kde lze očekávat větší šířku než výšku obrazu. Každopádně tyto případy budou ojedinělé a případy kdy osoba stojí budou velice běžné. Bude tedy potřeba minimálně upravit vstup a výstup výchozího modelu viz obrázek 4.2. Jelikož je postupně redukována velikost vstupu, je potřeba zvolit vstupní rozměr, alespoň na 456×228. Pokud by byl zvolen menší rozměr, ku příkladu poloviční z originálního 288×144, tak by tento model nešel sestrojit, protože se jedna velikost rozměru snížila na tolik, že by ji nebylo možné použít a takový model by nešel ani přeložit.



Obrázek 4.2: Minimální úprava výchozího modelu z 4.1.

Kapitola 5

Implementace

Tato kapitola se věnuje praktické realizaci. Kde v první části 5.1 je nejdříve popsán algoritmus vytvoření datasetu, kde je popsán formát, algoritmus výběru vhodných snímků a poté samotný algoritmus zakrytí nohou. V následující části 5.2 je poté popsán *Dataloader*, který se stará o práci s datasetem vůči neuronové síti. Neboli připravuje vhodná data na základě parametrů, které jsou nataveny dle aktuálních požadavků, které se mohou vztahovat vůči experimentům. Poslední část 5.3 je poté věnována samotné implementaci a trénování modelu. Kde implementace je rozdělena na 3 etapy, ve kterých se model postupně upravuje dle výsledků trénování, které je zde možné vidět jako grafy.

5.1 Dataset

Než bude možné věnovat se trénování je potřeba vytvořit požadovaný dataset. Jak bylo zmíněno v návrhu jako zdroj vstupních dat bude využito datasetu COCO, ze kterého budou vybrány vhodné snímky. Tyto snímky budou vybrány a poté plně automatizovaně upraveny pomocí Python skriptu. Původní anotace bude využita k automatickému zpracování a část bude využita i ve výsledném datasetu ať už s původními nebo upravenými hodnotami. Výsledný dataset bude ve stejném formátu jako COCO dataset neboli bude se jednat o JSON soubor se stejnou strukturou.

5.1.1 COCO formát

Dataset COCO využívá formátu JSON pro anotaci na rozdíl od formátu XML se kterým se lze často setkat. Jedná se o 1 JSON soubor na celou sadu obrázků vůči jednomu typu anotace, ku příkladu anotace *person_keypoints* trénovací data jsou 1 JSON soubor a validační data poté další 1 JSON soubor. Tento JSON soubor obsahuje zvlášť informace o obrázcích a k nim přiřazené anotace. Struktura tohoto souboru lze vidět na ukázce A.1, která vychází z [17, 15].

Tato struktura není pevně daná a mění se v závislosti na dané úloze, uvedená struktura je pro případ, kdy se jedná o detekci nebo o predikci bodů na těle. Při ostatních úlohách se zde můžou nacházet další možné hodnoty, které ale nejsou podstatné vůči této práci. Zmíněný formát tedy obsahuje 5 základních hodnot:

- info - Obsahuje obecné informace o datasetu jako jeho popis, verze atd.
- licences - Obsahuje informace o použitých licencích. Všechny obrázky nejsou pod stejnou licencí, ale pod různými licencemi, a právě zde jsou vypsány.

- categories - Obsahuje informace o kategoriích využívaných u detekce. S volitelnou hodnotou superkategorie neboli nadřazené kategorie. Ku příkladu může být kategorie *dog* a její nadřazená kategorie *animal*.
- images - Obsahuje informace o jednotlivých obrázcích. Kde mezi nejdůležitější informace patří ID, licence, výška, šířka a název souboru.
- annotations - Obsahuje informace o jednotlivých anotacích. Na každém snímku může být více anotací, proto je ID anotace a ID snímku jiné. Obsahuje tedy vlastní ID, poté ID snímku, vůči kterému se anotace váže a ID kategorie. Ohraničující body ve formátu *x,y* levého horního rohu, *šířka* a *výška*. Informaci zda se jedná o dav. Segmentaci ve formátu RLE, pokud se jedná o dav (*iscrowd* je 1) jinak jako polygon. Velikost plochy, kterou objekt zabírá (plocha je počítána ze segmentace). Jednotlivé body lidského těla ve formátu *x, y, v*, kde *v* je viditelnost: 0 = bez anotace (x a y je poté také 0), 1 = anotováno, ale bod není viditelný a 2 = anotováno a bod je viditelný. A jako poslední počet anotovaných bodů na těle, kde v základu jich je 0-17 a to dle počtu reálně anotovaných bodů.

Práce s COCO datasetem

Jelikož se jedná o JSON, bylo by možné využít *json* knihovny, kde by celý soubor byl převeden na slovník a následně by s datasetem bylo pracováno manuálně. Každopádně je zde značný počet prvků propojených pomocí ID a manuální procházení by mohlo být zbytečně složité. Druhou možností je totiž využít existujícího rozhraní[1], které poskytuje možnost načtení datasetu a poté řadu pomocných funkcí pro práci s datasetem. Mezi tyto funkce patří například filtrování dle kategorií, získání všech anotací vůči snímku, práce s maskou a zobrazení anotací na snímku. Zmíněné rozhraní jsem použil při práci s původním COCO datasetem.

5.1.2 Výběr vhodných snímků

Jak již bylo naznačeno v návrhu, COCO dataset obsahuje data pro mnoho různých účelů. Proto je potřeba vybrat pouze vhodné snímky, které budou použity. Jedná se o dva typy snímků (osoby a objekty), kde každý typ snímku má jiný způsob výběru. Zároveň se na tyto snímky vztahuje výběr dle licence, protože některé licence nedovolují úpravy.

Výběr snímků osob

Po vyfiltrování snímků dle licencí je potřeba vybrat snímky, které obsahují osoby a jsou vhodné k řešení tohoto problému. Výběr snímků, které obsahují osobu je poměrně jednoduchá záležitost. Dataset totiž obsahuje informace o kategoriích, jedná se tedy pouze o vyfiltrování snímků dle kategorie *person*.

Lehce složitější je poté výběr snímků, přesněji tedy přímo anotací na daném snímku, které jsou vhodné. Základní podmínkou je, že jsou anotovány oboje chodidla, jelikož budou očekávány na výstupu. Fakt, jestli nejsou viditelné a anotované nebo jsou viditelné a anotované, zde nehraje roli, jelikož cílem je nohy zakrýt. Dalším požadavkem je, aby na lidském těle bylo anotováno dostatek bodů. A to z důvodu, že dataset obsahuje také pouze části těla, jako třeba pouze nohy, což v případě predikce bodů viditelných končetin může být

[1] COCO rozhraní je dostupné zde: https://github.com/cocodataset/cocoapi

výhodou. Každopádně pokud by zde na snímku byly pouze chodidla, které by poté byly zakryty, tak by upravený snímek byl bez jakékoliv části osoby. A na takovém snímku není možné odhadovat pozici chodidel. Proto je potřeba, aby na daném snímku byla viditelná značná část lidského těla. Z čehož také vychází formát vstupu modelu, kde je očekávána jedna osoba, oříznutá dle ohraničujícího boxu. Jednou z možností by bylo využít počet anotovaných bodů, ale jelikož obličej obsahuje 5 bodů, což je skoro třetina z celkového počtu, bude lepší mít podmínku přímo na jednotlivé body dle umístění. Výsledná podmínka tedy bude, aby byly anotovány oboje chodidla, 1 bod kyčlí, alespoň 1 bod na rukou a 1 bod na hlavě, což by mělo zajistit, že na snímku je většina lidského těla.

Poslední podmínkou při filtrování snímků je poté velikost dané osoby na snímku. Jelikož vstupní velikost neuronové sítě bude 456 pixelů na výšku. Tak budou vybrány pouze snímky, kde ohraničující box okolo osoby je vyšší než 65 pixelů, tedy alespoň sedmina. Pokud by byly ohraničené snímky příliš malé, tak po změně velikosti na vstup sítě, by mohly být příliš rozmazané a nekvalitní.

Výběr snímků k zakrytí nohou

Stejně jako u výběru snímků osob, jsou nejdříve vyfiltrovány snímky dle licence. Následně jsou vybrány snímky, na kterých se nevyskytuje žádná osoba, což lze provést jako odečtení snímků z předchozího výběru, na kterých se osoba vyskytuje, od množiny všech snímků. Vstupní COCO dataset je složen přibližně polovinou snímků, co obsahují osobu a druhou polovinou jež osobu neobsahují. Toto filtrování snímků je zde z důvodu, aby následně při výběru objektu k zakrytí nohou, nebyla vybrána jiná osoba.

Další podmínkou při výběru jsou poté rozměry ohraničujících boxů a plocha, kterou v daném boxu pokrývají. Jedna z těchto podmínek je, že rozměry ohraničujícího boxu jsou v maximálním poměru od 1:4 do 4:1. Snímky mimo tento rozměr jsou příliš vysoké nebo široké a spadají zde většinou nevhodné snímky, které špatně zakryjí nohy. Příkladem vysokého objektu může být stojící knížka a širokého objektu vlak. Druhou podmínkou je poté pokrytá plocha z boxu, která je nastavena na minimálně 50 %. Například tyč, otočená o 45° může mít vhodné rozměry z pohledu ohraničujícího boxu, ale k zakrytí je to nevhodný objekt.

5.1.3 Algoritmus pro překryv nohou

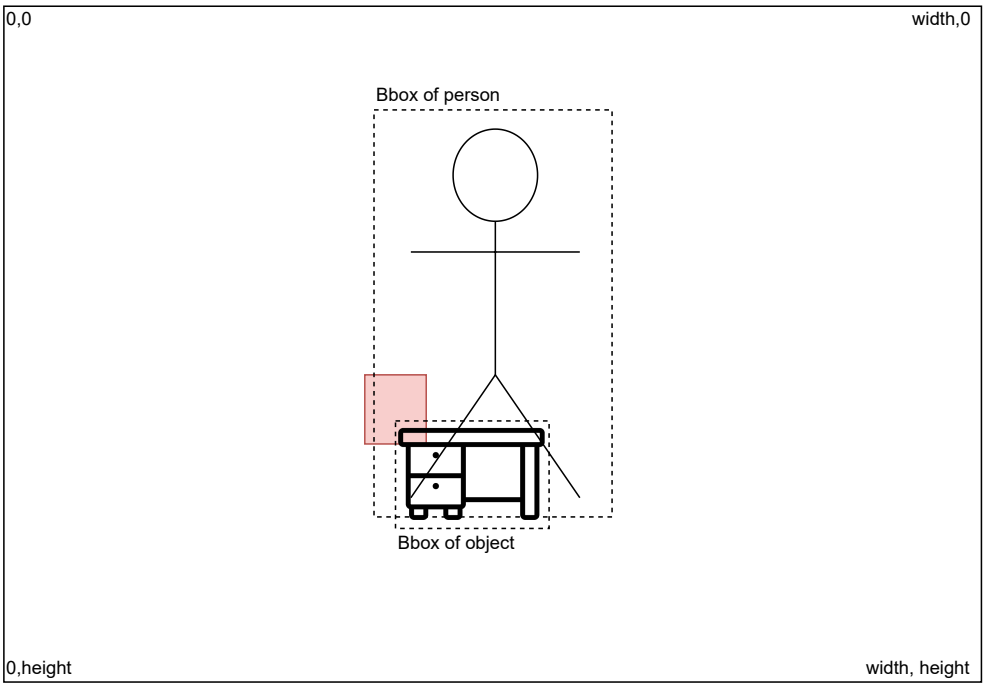
Po vybrání vhodných snímků přichází samotná úprava snímků, kdy jsou zakryty nohy. Pro práci se snímky jsem se rozhodl využít knihovny PIL. Jelikož při výběru byla podmínka na minimální anotované body, je možné je využít při úpravě, ať už k určení pozice nebo k úpravě velikosti zvoleného objektu. Daná úprava zakrytí není ideální, obzvlášť při specifických pózách, každopádně není problém mít nohy nedokonale zakryté nebo občas i nezakryté. Protože cílem je predikce zakrytých chodidel, případně i zakrytých celých nohou. A je pouze potřeba zakrýt nohy dostatečně, aby se síť nenaučila predikovat chodidla na základě vzhledu, ale na základě horní poloviny těla.

Samotný algoritmus poté prochází v rámci iterací všechny vybrané snímky s osobami a k nim vyfiltrované anotace. Na základě těchto anotací je zde poté umístěn objekt, který je vybrán náhodně z druhé sady snímků neobsahující osobu. A jelikož dataset obsahuje dostatečný počet snímků a anotací objektů mimo osoby, tak bude pro zakrytí nohou využito různých objektů a nehrozí, že by se vůči nim síť naučila predikovat. Tento náhodně vybraný objekt je poté pomocí segmentace oříznut, díky čemuž má každý objekt do jisté míry

unikátní tvar. A síť se tak nemůže naučit vůči ostré hraně, kdyby byl objekt oříznut pomocí ohraničujícího boxu.

Následně je vypočtena požadovaná výška obrázku, na kterou bude obrázek změněn. Tato výška je vypočtena jako rozdíl hodnot y mezi chodidly a kyčlemi, případně jako rozdíl hodnot x mezi chodidly, dle toho, která hodnota je větší. Kde hodnota y pro chodidla je vypočtena jako průměr levého a pravého chodila. Hodnota y pro kyčle je vypočtena také jako průměr, pokud jsou anotovány obě nebo je zvolena hodnota z anotovaného bodu. Kde filtrování zaručuje minimální anotaci, aby to bylo možné. Tato výška je poté použita jako výsledná výška zvoleného objektu k zakrytí nohou, šířka je poté změněna, tak aby byl zachován původní poměr objektu.

Ve chvíli, kdy tedy máme náhodný objekt připravený k zakrytí, tak je potřeba zvolit místo, kde bude objekt vložen. Toto místo určuje bod, kde bude umístěn levý horní roh objektu (zde začínají nulové souřadnice v PIL obrázku). Výpočet tohoto bodu poté záleží na póze člověka. Výpočet se poté liší v případě, kdy je osoba obličejem vpřed nebo vzad, což většinou určuje, jestli je levá nebo pravá noha blíže k levému okraji snímku (nižší hodnota x). Stejně tak se výpočet liší v závislosti, jestli je osoba vzhůru nohami, i když tento případ je výjimečný, stále může nastat. Pro výpočet tohoto bodu je určena plocha, kde by se bod mohl nacházet, ze které je poté jeden bod náhodně vybrán. Tímto je znovu zaručeno, že se vložený objekt bude nacházet vždy na jiném místě a neuronová síť se nemůže učit na základě umístění tohoto objektu. Hodnota y je vybrána víceméně od poloviny nohou až po pas (v případě, kdy je osoba vzhůru nohami je y posunuto až nad nohy). Díky tomuto by mělo být zajištěno, že někdy jsou zakryté celé nohy a někdy pouze spodní část nohou. Hodnota x se poté nachází nalevo od nohou, kde je posunuta na základě rozdílu hodnot x levého a pravého chodidla a zároveň tedy nesmí být posunuta za hranici obrazu. Tento princip je možné vidět na obrázku 5.1, kde právě červeně vyznačená oblast značí, kde se může umístit levý horní roh vybraného objektu.



Obrázek 5.1: Ukázka způsobu zakrytí nohou.

Další úpravou je poté rozšíření ohraničujícího boxu, a to nejdříve na minimálně hodnoty anotovaných bodů chodidel, pokud se nacházejí mimo tento box. Dále poté o 5 % na každou stranu, aby body nebyly tak blízko vůči okraji snímku, který bude na vstupu sítě. Vložený objekt je dále potřeba oříznout v případě, kdy přesahuje ohraničující box osoby, aby se nestalo, že by z části zakryl i další osobu, která může být na stejném vstupním snímku.

Upravené snímky se poté dle náhodnosti liší, jak dobře jsou upravené. Na obrázku 5.2 lze poté vidět případ, kdy úprava vypadá dostatečně reálně a také zakrývá značnou část nohou a poté také úpravu, která je méně kvalitní. Bylo by možné zde umístit i podmínku na velikost snímku, aby nebyl vložený objekt takto rozmazaný, každopádně zde to není tak podstatné.

Obrázek 5.2: Ukázka upravených snímků.

5.1.4 Výsledný formát

Výsledný formát je stejně jako vstupní, formát COCO a jedná se tedy o JSON. Protože se jedná o dataset s jedním účelem, jedná se celkově o 1 JSON soubor pro trénovací a 1 JSON soubor pro validační data. Některé informace už ale nejsou potřeba, a tak byly odstraněny. Stejně tak některé informace jsou zde pozměněné. Z původní anotace byly odstraněny informace o kategoriích, jelikož se jedná pouze o dataset obsahující osoby. Části *info*, *licenses* a *images* zůstaly v původním formátu a většinou i s původními hodnotami, změněná je poté část *annotations*. Kde *keypoints* jsou požadované body chodidel a *other_keypoints* jsou poté zbylé anotované body, bez informace viditelnosti. Pokud nebyly anotované, jsou stejně jako na vstupu nastaveny na hodnotu 0. Ukázku, jak vypadá výstupní formát lze vidět na A.2.

5.2 Dataloader

Jakmile je připravený dataset, tak je potřeba zajistit jeho předávání neuronové síti. K tomuto Keras poskytuje různé funkce, například při řešení klasifikace stačí mít dataset ve formě připravených snímků roztříděných do adresářů pojmenovaných dle dané třídy a o vět-

šinu se postará Keras. Pokud se ale jedná o vlastní dataset, je postup složitější. Dataset je potřeba předávat postupně, kde záleží na velikosti *batche*, stejně tak je potřeba dané snímky v datasetu připravit na vstup sítě. Aktuálně jsou v datasetu stále celé upravené snímky bez ořezání, kde se může na snímku vyskytovat více osob.

K tomuto je možné vytvořit si vlastní třídu, která dědí z `tf.keras.utils.Sequence`. Tato třída dané operace zapouzdří a Keras bude vybrané funkce poté automaticky volat při učení. Nejdůležitější je implementace následujících tří funkcí, kde dvě z nich jsou povinné.

- `__getitem__` - Tato funkce musí být implementována a vrací data o velikosti *batch* na základě předaného indexu ve formátu (*x, y*), kde x značí vstup sítě a y poté výstup. Výstup a vstup je poté ve formátu [batch_size, ...].
- `__len__` - Tato funkce musí být implementována a vrací velikost datasetu vypočtenou jako celková velikost vydělená velikostí *batch*.
- `on_epoch_end` - Tato funkce je volitelná a je volána automaticky na konci epochy, kde může upravit daný dataset, například zamícháním hodnot.

5.2.1 Načtení datasetu

Při inicializaci *dataloaderu* je potřeba načíst dané anotace datasetu. Jednou z možností by bylo využít COCO rozhraní, které bylo využito při vytváření datasetu. Jelikož se ale jedná o značně upravený a zmenšený dataset, tak je možné toto načtení provést i ručně pomocí *json* knihovny. Jsou zde vytvořeny dva slovníky obsahující informace o snímcích a anotacích podle jejich ID. A jeden obsahující dostupné anotace ke snímku dle image_ID, které je víceméně identické jako název souboru snímku, a lze díky tomu lehčeji testovat specifické případy.

Slovník obsahující anotace je převeden na list všech dostupných ID, které jsou využívány při trénování, kde toto pole je na konci každé epochy náhodně zamícháno, aby se síť nemohla učit na základě toho, jaké snímky postupně přichází na vstup. Tedy zamícháno je na základě vstupního parametru, který je v základu nastaven na True. Toto míchání lze tedy vypnout pro testovací potřeby.

5.2.2 Vstup sítě

Základním vstupem sítě je obraz v požadovaném rozměru, kde jako základní rozměr byl zvolen $456 \times 228 \times 3$. Součástí přípravy vstupu je potřeba z celého snímku získat pouze požadovanou oblast obsahující osobu. Z celkového snímku je takto vyříznuta pouze osoba pomocí ohraničujícího boxu. Tomuto výřezu je následně změněna velikost na požadovanou vstupní velikost sítě (na základě vstupního parametru). Dle parametrů je poté převeden do úrovní šedi nebo ponechán jako RGB a případně jsou jednotlivé hodnoty pixelů převedeny do rozsahu 0-1 z 0-255 vydělením hodnotou 255.

Na základě vstupních parametrů jsou zde další možné možnosti typu vstupních dat, které jsou dále součástí experimentů. Je možné mít na vstupu pouze body, a to 30 hodnot (15 bodů x a y) nebo 26 hodnot (předchozí hodnoty bez kolen). Další možností je kombinace vstupu, kdy na vstupu jsou zároveň body i obraz.

5.2.3 Výstup sítě

Základním výstupem sítě jsou body chodidel, jedná se tedy o 4 hodnoty (x a y pro každé chodidlo). Tyto hodnoty z datasetu je ale potřeba upravit. V datasetu jsou tyto hodnoty

v rozmezí pro x 0-img_width a y 0-img_height neboli jsou určeny v rámci celého snímku. Nejdříve je tedy potřeba posunout tyto body, aby jejich počátek byl v levém horním rohu ohraničujícího boxu a ne celého snímku. K této úpravě stačí od bodů odečíst levý horní bod ohraničujícího boxu. Jelikož každý ohraničující box má jiné rozměry. Tak je dále potřeba převést body z rozsahu x 0-bbox_width a y 0-bbox_height do rozsahu dle požadovaného rozměru na vstupu sítě. Dle vstupních parametrů může být následně rozsah převeden do rozsahu 0-1.

Dle vstupních parametrů může být jiný formát výstupu. Jedná se o kombinaci bodů chodidel a zbylých bodů na těle, jako součást experimentů. Ve stejném formátu jako byla kombinace obrazu a bodů na vstupu.

5.3 Implementace modelů

Jakmile je připravený dataset i vhodný *dataloader*, je možné začít samotné trénování modelu. Nejdříve je ale potřeba daný model přeložit pomocí metody *compile*, kde je modelu předána funkce, která je využita u výpočtu *loss*. Zde se jedná o *Mean squared error* (MSE 3.7). Dále je zde předán optimalizátor, který byl zvolen *Adam*. Samotné trénování pomocí rozhraní Keras probíhá voláním metody *fit* na model. Této metodě je potřeba předat minimálně vstupní a výstupní data neboli data z trénovacího datasetu a požadovaný počet epoch. Je možné dále předat validační data na kterých model není trénován, ale na konci epochy je vůči těmto datům spočtena validační *loss* (*vall_loss*).

Je také možné předat seznam *callback* funkcí, které jsou volány v průběhu trénování. Kde byly využity následující funkce. `EarlyStopping`, kde je možné předčasně ukončit trénování dle nastavených parametrů, například pokud 10 epoch po sobě neklesne *val_loss*. Tato funkce byla využita víceméně jen ze začátku, kdy je neznámo, jaký počet epoch je ideální (následně se ukázal jako ideální počet 30). `ModelCheckpoint`, který ukládá postupně model, například na konci každé epochy, díky čemuž je zachován všechen postup. Na druhou stranu každé uložení modelu může zabírat spoustu místa na disku. Při jednom z experimentů jsem použil model VGG16, kde uložení celého modelu zabírá na disku přibližně 2,6 GB, což při trénování 30 epoch je ve výsledku přibližně 80 GB. `CSVLogger`, který ukládá na konci dané epochy výsledky ve formátu CSV, díky čemuž se lze lehce podívat na výsledky zpětně.

Trénování a experimenty jsou zde částečně provázány, jelikož pomocí experimentů byl model postupně upravován až do jeho výsledné formy. Samotný proces jsem měl víceméně ve 3 etapách, ve kterých byly provedeny značné změny v modelu. S výjimkou první etapy, trénování bylo provedeno na datasetu čítající 26 996 anotací a validace byla provedena na 1 106 anotacích.

5.3.1 Trénovací prostředí

Pro trénování jsem se rozhodl využít vlastních zdrojů a trénovat modely lokálně, což ulehčuje řadu věcí, jako je například nahrání datasetu nebo poté stáhnutí výsledného modelu. Na druhou stranu trénování vyžaduje dostatek výkonu a v průběhu trénování není možné využít počítač k účelům, které vyžadují určitý výkon grafické karty. Z tohoto omezení jsem se pokusil využit služby Google Colab, která poskytuje zdarma přístup k rozhraní Colab Notebook, kde je možné spustit Python skript. Hlavní výhodou je možnost výběru běhového prostředí CPU, GPU a TPU, kde lze tedy snadno trénovat na grafické kartě, zdarma bez vytížení vlastního zařízení. Každopádně při testování této služby jsem narazil na několik problémů, kde většina vychází z faktu, že Google Colab je určen pro interaktivní práci,

a ne pro dlouhodobé trénování. Google u této služby, dokud je zdarma, nijak nezaručuje výpočetní výkon a výkon je poté přiřazen dle využití, stejně tak není možné mít jeden Notebook aktivní déle než 12 hodin dle [13]. Dále je potřeba být „aktivní“, kde je potřeba jednou za čas projevit aktivitu jako je například kliknutí myší. Osobně jsem se s tímto také setkal, kdy po mně po delší době bylo vyžadováno vyřešit Captcha úlohu. Dalším problémem bylo, že rychlost trénování byla asi 3x pomalejší než lokálně s využitím jedné grafické karty RTX 3060 Ti. Posledním problémem bylo, že mi bylo trénování neočekávaně vypnuto po překročení asi 4 hodin, kde jsem tedy nestihl model natrénovat. Z tohoto důvodu jsem zvolil trénování lokální, namísto využití této nebo podobné služby.

Samozřejmě využití této služby nepřináší jen nevýhody. Odpadá nutnost lokálního nastavení, jako je instalace všech potřebných knihoven nebo dalších nástrojů jako Cuda. Jak již bylo zmíněno, není vytíženo vlastní zařízení, kde minimálně kromě omezení využívání v době trénování je také potřeba počítat s cenou elektrické energie. Tato služba by pravděpodobně byla vhodná pro trénování menších modelů, které mají na vstupu několik hodnot a nejedná se tedy o konvoluční sítě, kde je potřeba zpracovávat obraz.

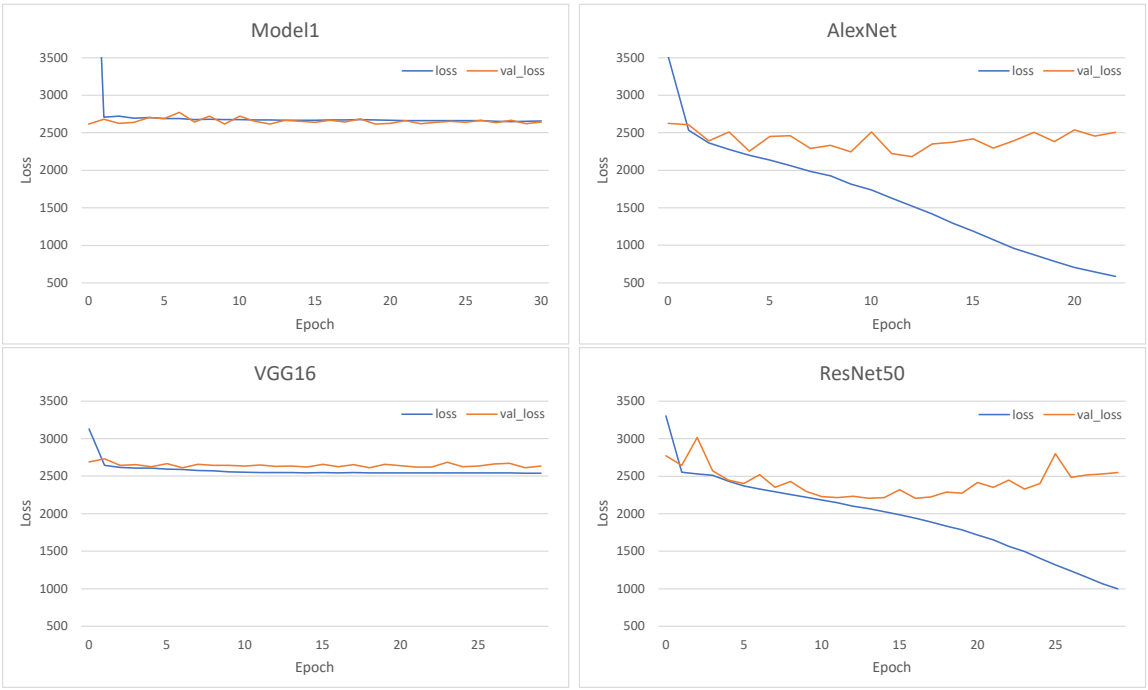
5.3.2 Základní model

V první etapě probíhalo prvotní trénování modelu, vůči kterému byly následně provedeny lehké změny, které ale ve výsledku víceméně nepřinesly zlepšení. Zároveň v této části ještě docházelo k lehké úpravě datasetu, kde aktuálně ještě nebylo ohraničení rozšířeno o 5 % na každou stranu, nebyly ořezávány vložené objekty dle ohraničujícího boxu a také byly lehce mírnější podmínky na výběr vhodných snímků osob. Tyto úpravy v průběhu vedly k lehce kvalitnějšímu datasetu za cenu snížení jeho velikosti asi o 10 % z přibližně 30 000 anotací na 27 000 anotací. Minimálně z důvodu rozšíření ohraničujícího boxu je v dalších etapách nižší *loss*, jelikož se chodidla nachází v menším prostoru.

V této části byl také problém s výchozím modelem, který se víceméně neučil neboli se nesnižovala funkce *loss*. Díky tomuto byl problém s předčasným zastavením, kdy se učení zastavovalo velice brzy. Proto jsem se rozhodl natrénovat tento model na 100 epochách, abych si byl jist, že se doopravdy neučí. Výsledky trénování je možné vidět na obrázku 5.3, kde lze vidět, že *loss* klesá extrémně pomalu a vypadá to, že se tedy neučí. Co se týče samotných hodnot, tak jedná se o MSE funkci, kde očekávané výstupní hodnoty jsou v rozsahu 0-228 pro x a 0-456 pro y a při druhé mocnině vzniknou takto vysoké hodnoty.

Obrázek 5.3: Výsledky trénování prvního modelu.

Byly zde prováděny lehké změny, převážně tedy pokus o zvýšení počtu plně propojených vrstev na konci sítě. Každopádně tyto pokusy nevedly ke zlepšení, stejně tak se model stále neučil. Proto jsem se rozhodl zkusit natrénovat již existující modely, kde jsem zvolil AlexNet, ResNet50 a VGG16. Abych se částečně ujistil, že je doopravdy potřeba upravit model a také jaký je očekávaný počet epoch. Jelikož AlexNet má poměrně nízký počet vrstev, tak jsem jej nadefinoval manuálně. To se ale nedá říct o dalších dvou modelech, které jsou více komplexní. Tyto modely jsem tedy importoval z `tf.keras.applications`, kde je možné modelům předat jak požadovaný vstupní rozměr tak požadovaný počet výstupů a aktivační funkce na výstupu.



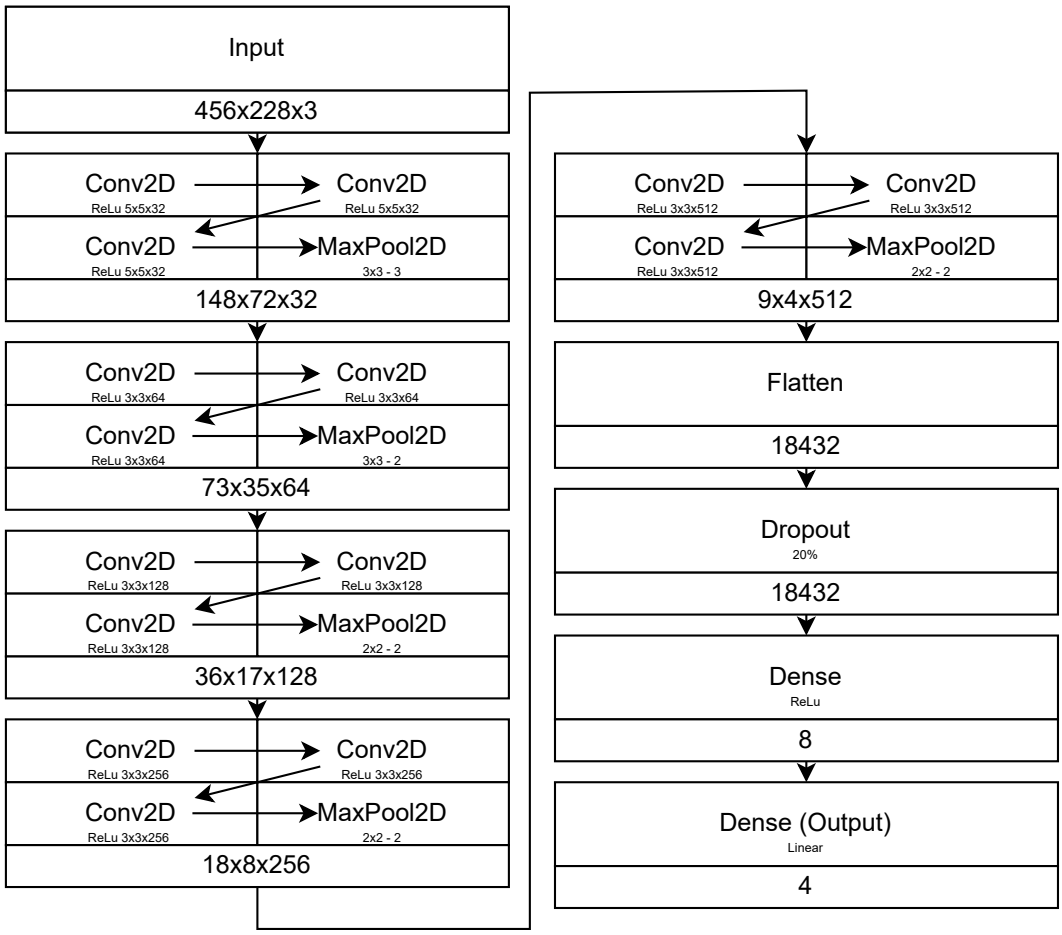
Obrázek 5.4: Srovnání prvního modelu s AlexNet, VGG16 a ResNet50.

Výsledky srovnání trénování těchto modelů lze vidět na obrázku 5.4. Kde *Model1* byl trénován na 100 epochách, a zbylé modely jsem plánoval 30 epoch pro každý model. U modelu AlexNet šlo vidět přeučení dostatečně brzy, takže jsem zde po 23 epochách trénování ukončil, jelikož by asi nikam dále nevedlo. Z grafů si lze všimnout, že model VGG16 má víceméně stejný průběh. Jeho hodnoty *loss* funkce jsou víceméně stejné a také se neučí. Na druhou stranu modely AlexNet a ResNet50 se učí, jak lze vidět dle postupně klesající *loss* funkce a zároveň jejich hodnoty validační *loss* funkce jsou nižší než u předchozích 2 případů, takže dosahují mírně lepších výsledků. Jelikož se tyto modely učí, je tedy potřeba upravit *Model1*.

Pokud například porovnáme počet trénovatelných parametrů jednotlivých modelů, tak *Model1* má přibližně 3,5 milionu, ResNet50 23,5 milionu, AlexNet 102 milionu a VGG16 237 milionu. Je tedy možné, že aktuální model má příliš málo trénovatelných parametrů. Jak již bylo zmíněno dříve, vyzkoušel jsem zvýšit počet plně propojených vrstev, což zvýší počet trénovatelných parametrů, a to z jedné vrstvy o 196 prvcích na 2 vrstvy s 1024 a 512. Ale tento model měl značně horší výsledky. Zajímavé jsou také výsledky modelu VGG16, který se také neučí, je možné, že zde je těchto parametrů až příliš mnoho.

5.3.3 Vylepšený model

V druhé etapě byl model více upraven a jeho základní podobu lze vidět na obrázku 5.5. První změna je odstranění vrstvy *CoordinateChannel2D*, která dle testování nepřinášela výhodu. Další změnou je poté celková úprava parametrů konvolučních vrstev se zachováním původního počtu a následné snížení počtu prvků v plně propojené vrstvě. Samotná úprava konvolučních vrstev se odráží od architektury AlexNet, kde jsou nejdříve konvoluční jádra s větší velikostí a následně až s nižší.



Obrázek 5.5: Úprava výchozího modelu na *Model2*.

Zároveň zde byla porovnána vnitřní struktura v místě napojení na plně propojené vrstvy za pomocí metody *summary* volané na daný model. U původního modelu byl příliš nízký rozměr a je možné, že právě toto způsobovalo problém. Srovnání napojení na plně propojené vrstvy původního modelu, modelu AlexNet a aktuálního modelu lze vidět na ukázce 5.1. Každopádně VGG16 mělo velice podobné napojení jako AlexNet, ale i tak se model neučil. Zároveň díky celkové úpravě vzrostl počet trénovatelných parametrů na přibližně 8 milionu.

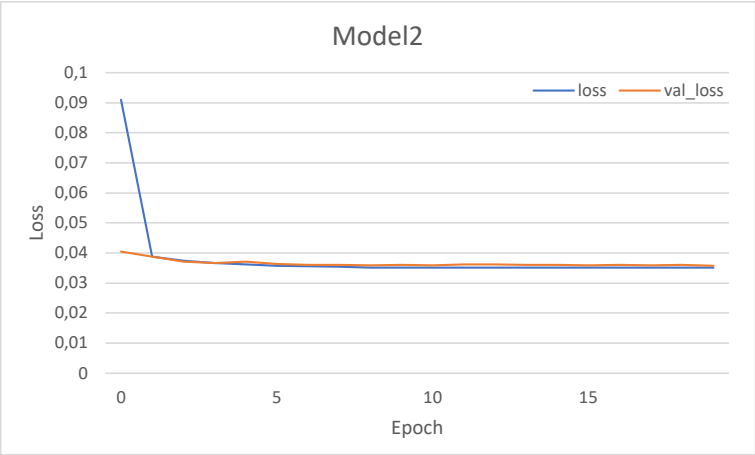
```
----------------------Model1----------------------
MaxPooling2D (None, 5, 1, 128)
Flatten (None, 640)
```

```
----------------------AlexNet---------------------
MaxPooling2D (None, 13, 6, 256)
Flatten (None, 19968)

----------------------Model2----------------------
MaxPooling2D (None, 9, 4, 512)
Flatten (None, 18432)
```

Výpis 5.1: Srovnání modelů v místě napojení na plně propojené vrstvy

Další významnou změnou byla změna rozsahu na výstupu do 0-1, kde by teoreticky síť mohla mít lepší výsledky. Tento model byl poté natrénován a jak lze vidět z obrázku 5.6, tak tento model se taktéž neučí. Jelikož už v průběhu trénování bylo vidět, že se model neučí, tak bylo trénování ukončeno po 20 epochách.



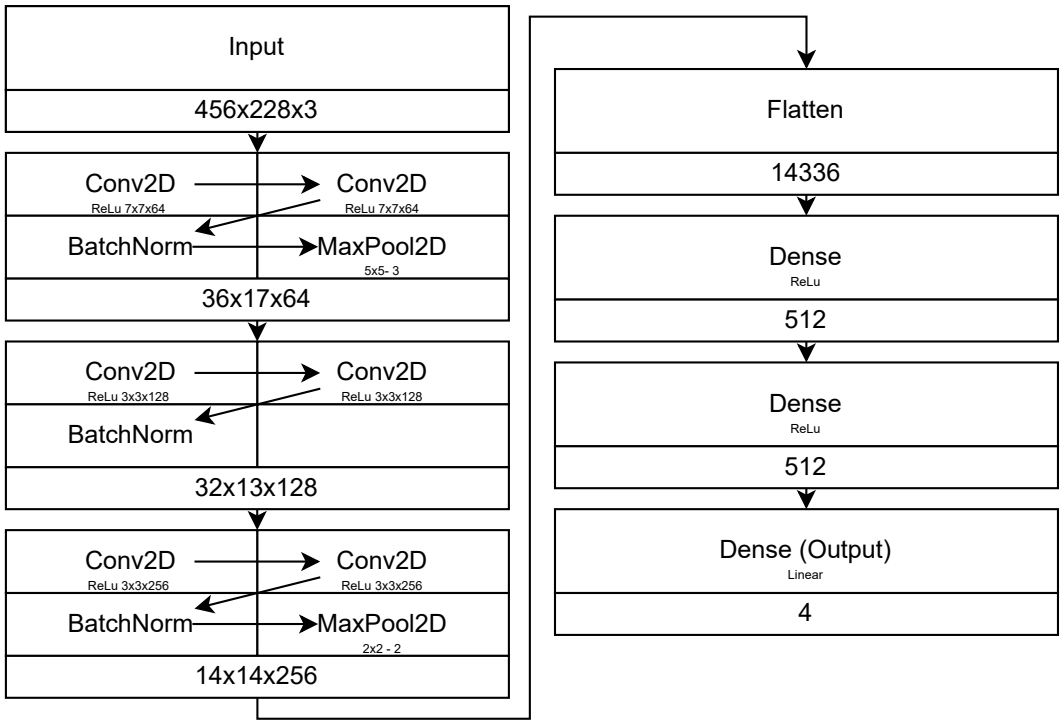
Obrázek 5.6: Výsledky trénování druhého modelu.

Dále byl upraven dataset o zmíněné rozšíření ohraničujícího boxu a také o ořezávání vložených objektů dle tohoto ohraničení. Díky čemuž se následně snížila loss funkce, jelikož se snížil prostor, ve kterých se nachází chodidla. Tento model byl dále lehce upravován, opět se jednalo o pokus se zvýšením velikosti plně propojených vrstev. Každopádně tyto pokusy nepřinesly zlepšení.

5.3.4 Finální model

Jelikož se v obou předchozích případech model neučil, tak byl model znovu zásadně upraven, a to směrem k modelu AlexNet. Kde byl značně snížen počet vrstev se zachování trénovatelných parametrů přibližně 9 milionu. Dále zde byla odstraněna *Dropout* vrstva, jelikož jejím účelem je pomoct proti přeučení, což zde není potřeba. Tento upravený model je možné vidět na obrázku 5.7. Tato úprava zajistila, že se model již učí, jak lze vidět na obrázku 5.8, kde je znázorněno 30 epoch trénování.

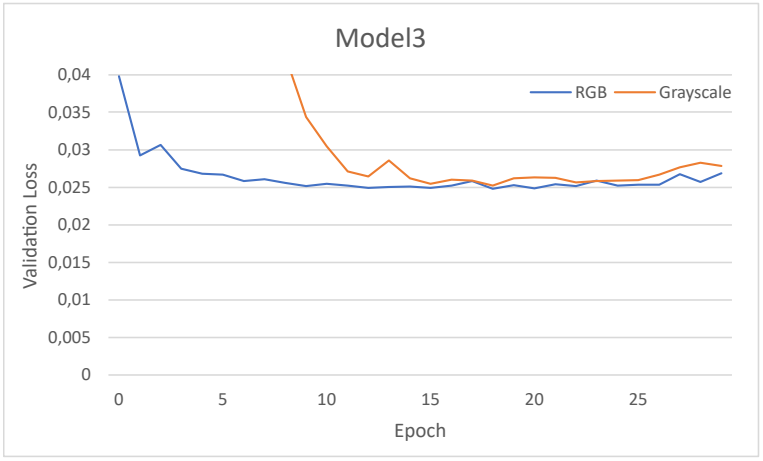
Jedním z pokusů bylo dále vyzkoušení modelu pouze na snímcích v úrovní šedi namísto RGB. Kde by teoreticky síť mohla dosáhnout lepších výsledků, kde by barevný vstup mohl být složitější pro síť kvůli vysoké variaci. Jelikož se mohou od sebe příliš lišit osoby pouze změnou barvy trička. Výsledky je možné vidět na obrázku 5.9. Kde lze vidět, že v obou případech dosahuje podobných hodnot, každopádně v úrovni šedi trvá déle, než se výsledky



Obrázek 5.7: Úprava modelu na *Model3*.

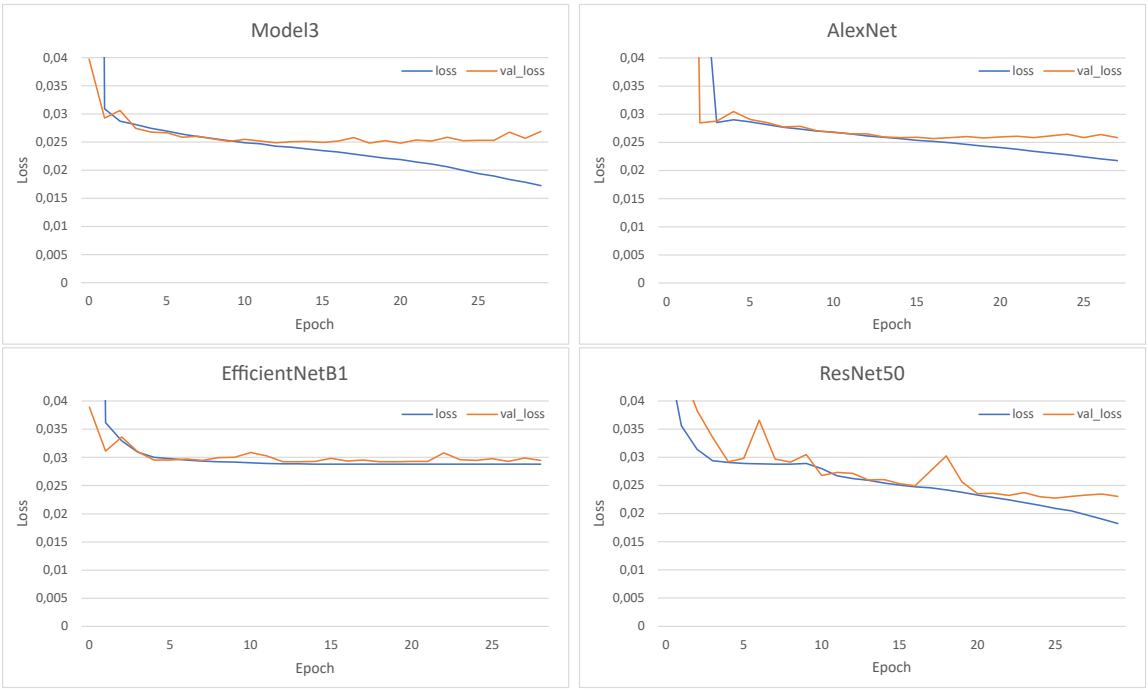
dostanou na podobnou úroveň. Na druhou stranu by úroveň šedi mohla vylepšit nedokonalost datasetu, který z důvodu plně automatického zpracování zakrývá nohy náhodným objektem. Tento objekt by mohl mít barvu, která nezapadá do okolí a síť by na toto mohla špatně reagovat.

Obrázek 5.8: Výsledky trénování třetího modelu.



Obrázek 5.9: Srovnání validační *loss* třetího modelu na RGB a Grayscale.

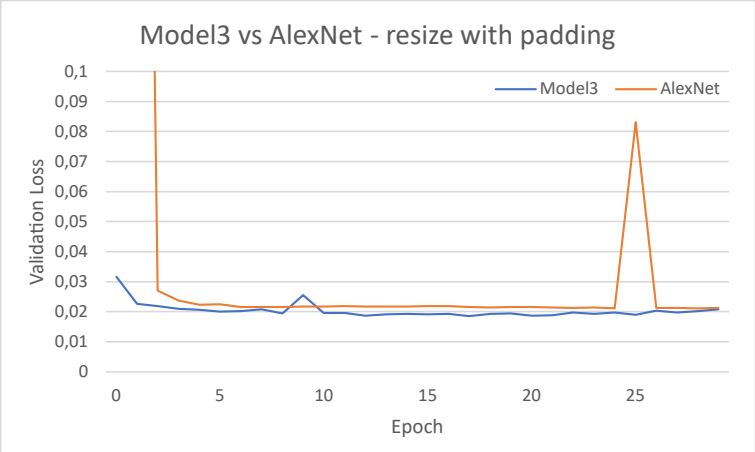
Tento model byl dále porovnán stejně jako v první etapě s dalšími modely, kde tentokrát byl model VGG16, který se neučil, nahrazen za model EfficientNetB1. Jako další modely poté byly ResNet50 a AlexNet, stejně jako tomu bylo v předchozím porovnání. Porovnání lze poté vidět na obrázku 5.10, kde lze vidět, že aktuální model se chová podobně jako AlexNet. Zatímco EfficientNetB1 se stejně jako VGG16 neučí. A ResNet50 dosahuje lehce lepších výsledků. U modelu ResNet50 bylo dále trénováno dalších 20 epoch, protože zde byla šance, že by se ještě snížila validační *loss*. Každopádně nižší hodnoty už nebylo dosaženo.



Obrázek 5.10: Srovnání prvního modelu s AlexNet, EfficientNetB1 a ResNet50.

Dále byl proveden pokus, kdy vstupní obraz nebyl rozšířen na vstupní rozměry „roztáhnutím“, ale vyplněním. Díky čemuž je následně zachován původní rozměr výřezu a nedochází tedy k deformacím původních poměrů na lidském těle. Na druhou stranu je tato část obrazu

vyplněna nulami, tedy černou barvou, a je výsledná plocha vstupního obrazu menší. Což způsobí, že se chodidla ve výsledku nachází v menším prostoru a lze tedy očekávat nižší hodnotu *loss*. Proto jsem se rozhodl provést tento pokus na dvou modelech. Na obrázku 5.11 lze poté vidět srovnání validační *loss* třetího modelu a modelu AlexNet, kde lze vidět nižší *loss* než bez této úpravy, oba modely se poté chovají podobně jako tom bylo při předchozím porovnání na obrázku 5.10, až na 2 hodnoty, které zde vytvořili špičku. Z čehož nelze přímo vyčíst, jestli se jedná o zlepšení nebo pouze o snížené hodnoty kvůli menšímu prostoru. Každopádně se nejedná o extrémní změnu a je poté složitější práce s body, kdy je potřeba při změně měřítka počítat s vyplněným prostorem.



Obrázek 5.11: Srovnání při změně velikosti s vyplněním třetího modelu a AlexNet.

Kapitola 6

Experimenty

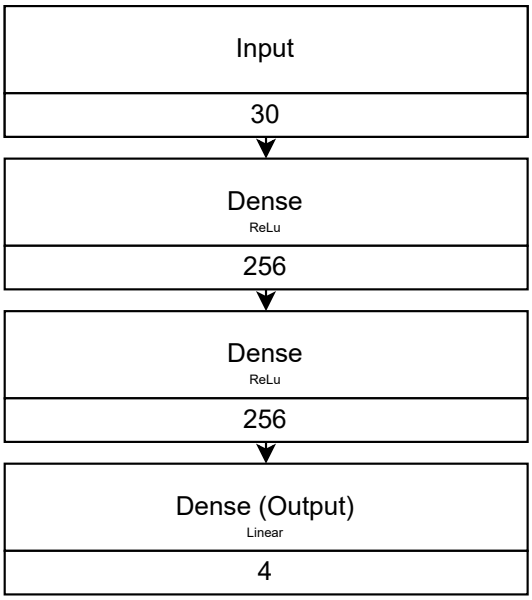
V této kapitole jsou provedeny experimenty vůči zvýšení přesnosti odhadu pozice chodidel. V těchto experimentech bylo využito zbylých bodů na lidském těle, které byly součástí anotace. V první části 6.1 je zde otestováno řešení, kde jsou využity pouze tyto body. V další části 6.2 kombinace těchto bodů na vstupu s obrazem. A v následující sekci 6.3 kombinace těchto bodů na výstupu, kde by mohly pomoct při trénování, se zachováním pouze obrazu na vstupu sítě. V poslední části 6.4 jsou poté výsledky jednotlivých modelů, včetně snímků s vykreslenými body odhadnutých chodidel, kde lze vidět přesnost jednotlivých přístupů z experimentů.

6.1 Vstup pouze body

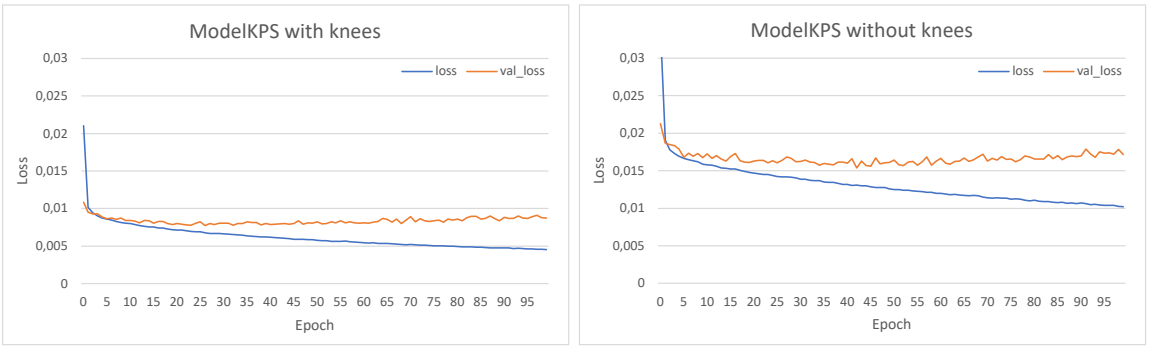
V této části jsou provedeny experimenty, kde se na vstupu budou nacházet pouze pozice bodů na lidském těle. Nejedná se tedy o konvoluční neuronovou síť, ale síť složenou pouze z plně propojených vrstev. Byly zde testovány postupně modely, které obsahovaly 64, 256, 512 a 1024 prvků v plně propojených vrstvách, většina z nich měla podobné výsledky. Jako reprezentativní model jsem tedy vybral model s 256 prvky, který lze vidět na obrázku 6.1 a dále je označován jako *ModelKPS*. Vstupem je v základu 30 hodnot, což je 15 bodů x a y, kde v případě kdy bod nebyl anotován je zde hodnota 0, jak bylo zmíněno v procesu zisku datasetu. Dále zde byly provedeny experimenty, kde na vstupu bylo pouze 26 hodnot, kde byly vynechány body na kolenou, jelikož tyto body mohou být taky zakryté a nelze s nimi tedy vždy počítat.

Na obrázku 6.2 lze poté vidět výsledky z trénování 100 epoch tohoto modelu jak s body včetně kolen, tak bez nich. Lze zde vidět, že v případě kdy jsou body včetně kolen, model dosahuje značně vyšší přesnosti v porovnání s hodnotami třetího modelu 5.8. Na druhou stranu, když jsou vynechány body kolen, model dosahuje značně nižší přesnosti než když jsou součástí vstupu. Každopádně stále dosahuje vyšší přesnosti než pomocí konvoluční sítě.

Samozřejmě stále se jedná o řešení problému zpracování obrazu, kde tyto body normálně nejsou dostupné. V reálném použití by tedy bylo potřeba pravděpodobně předřadit konvoluční síť, která bude odhadovat právě tyto viditelné body.



Obrázek 6.1: *ModelKPS* kde jsou vstupem body.

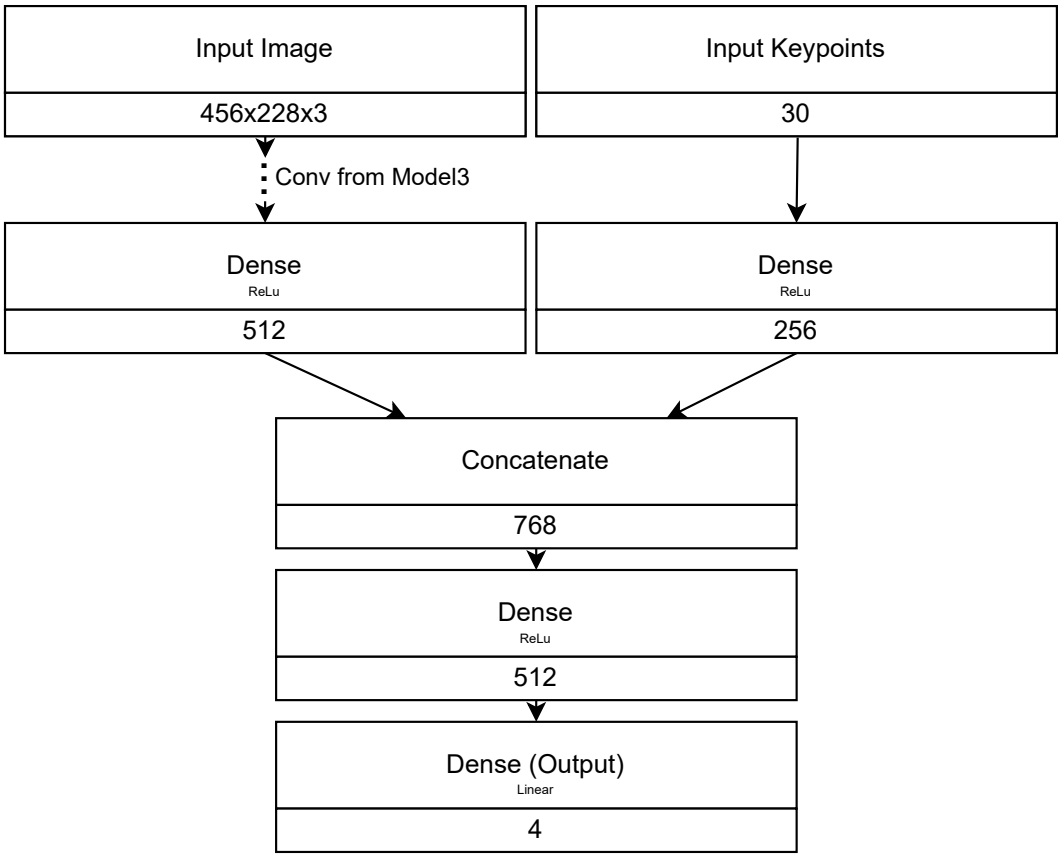


Obrázek 6.2: Srovnání výsledků s body včetně kolen a bez nich.

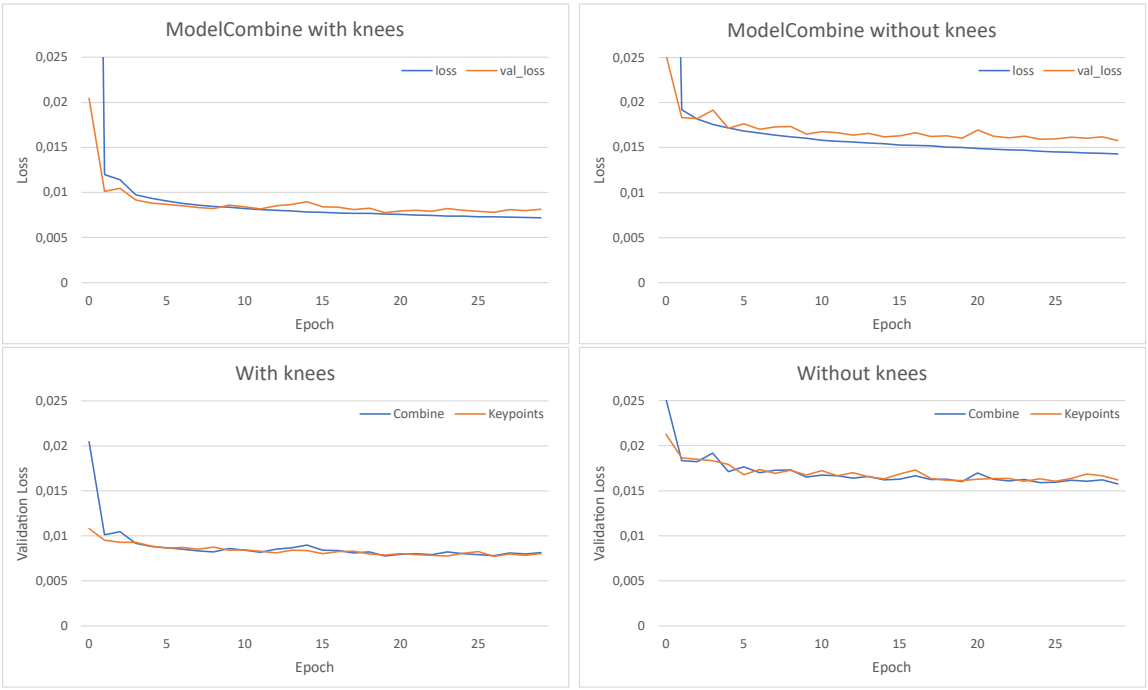
6.2 Kombinace vstupu obrazu a bodů

Dalším experimentem poté byla kombinace vstupu, kde síť obsahuje 2 vstupy. Jedním vstupem je obraz a druhým vstupem poté body na těle. Základem byl *Model3* z 5.7, kde byl přidán druhý vstup, který byl následně napojen na plně propojené vrstvy. Díky čemuž by teoreticky mohla být dosažena vyšší přesnost. Každopádně první 2 pokusy o spojení vrstev dopadly s horšími výsledky, než když byly použity body samotné a až třetí pokus dosahoval podobných výsledků. Tento model lze vidět na obrázku 6.3.

Na obrázku 6.4 lze vidět výsledky tohoto kombinačního modelu, kde jsou nejdříve výsledky samotného trénování a poté srovnání s výsledky, kdy byly použity pouze body. Z obrázku lze vidět, že model dosahuje víceméně stejných výsledků jako při použití pouze bodů a nebylo tedy dosaženo vyšší přesnosti.



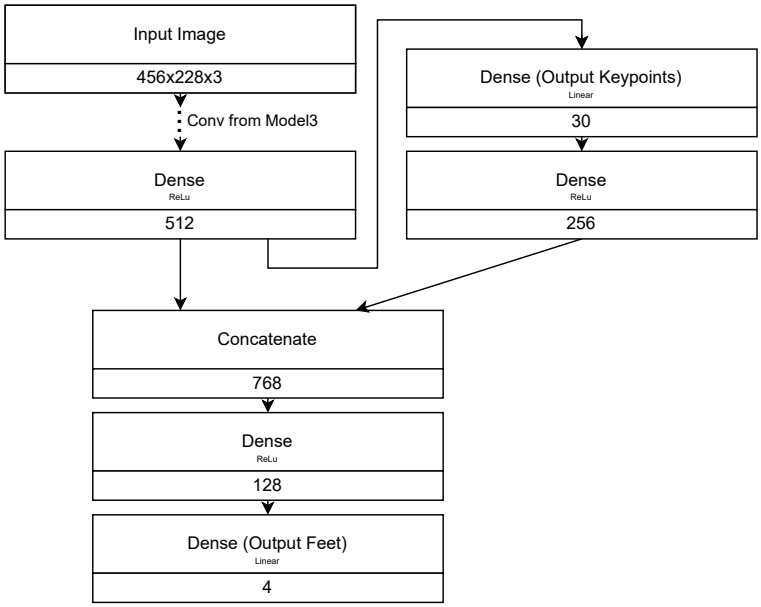
Obrázek 6.3: *ModelCombine* kde je vstupem jak obraz tak body.



Obrázek 6.4: Výsledky kombinačního modelu a porovnání s výsledky jen dle bodů.

6.3 Kombinace bodů na výstupu

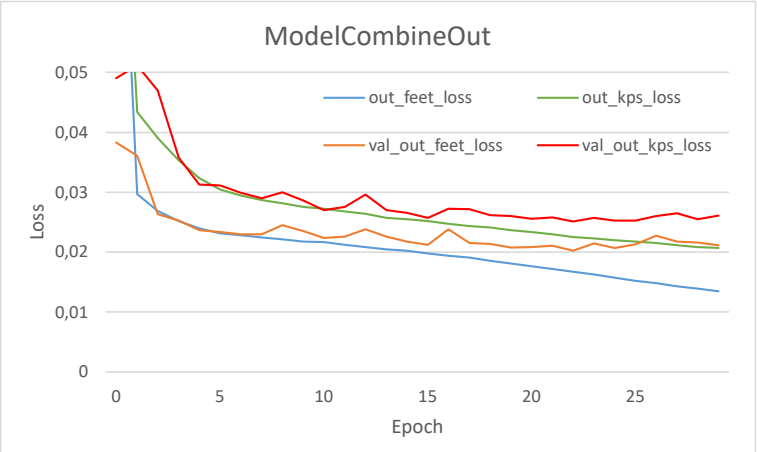
Jelikož samotné body dosahují značně lepších výsledků než samotný obraz, tak jsem zkusil provést experiment, kdy jsou body sice využity při trénování, ale na vstupu zůstane pouze samotný obraz. Toto bylo docíleno zahrnutím bodů na výstupu, tím pádem tato síť má dvojici výstupů, kde první část je očekávaný výstup odhadující pozici chodidel a druhý výstup jsou zbylé body na lidském těle. Tento druhý výstup je zároveň napojen na plně propojené vrstvy prvního výstupu. Toto napojení bylo opět testováno s různými hodnotami, kde se nejvíce osvědčila architektura na obrázku 6.5. Zároveň byly testovány body včetně kolen a také bez kolen, kde výsledky byly víceméně podobné s nepatrnými lepšími výsledky bodů bez kolen.



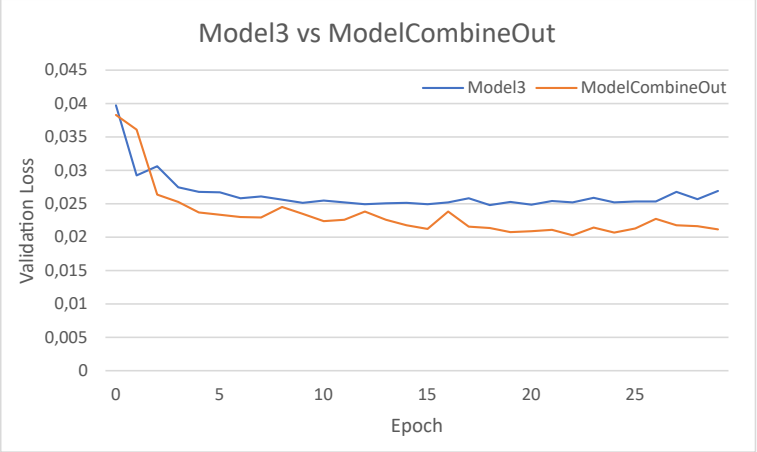
Obrázek 6.5: *ModelCombineOut* kde je výstupem dvojice bodů.

Výsledky trénování lze poté vidět na obrázku 6.6 ve verzi bodů bez kolen, jelikož tato verze dosahovala lehce lepších výsledků. Lze si všimnout, že samotná predikce chodidel je lehce lepší než u verze *Model3*. Každopádně stále byly dosaženy nejlepší výsledky, kdy jsou body přímo na vstupu. Na tomto grafu lze také vidět *loss* funkci týkající se ostatních bodů na těle. Lze si všimnout, že také nedosahují vysoké přesnosti jako je tomu u chodidel, přestože tyto body jsou většinou viditelné. Je možné, že je to z důvodu, že nebývají všechny body anotované a tyto vynechané body poté způsobují nepřesnost. Dalším pokusem tedy bylo upravení *loss* funkce tak, aby nezapočítávala body, které nebyly anotované. Každopádně tento pokus měl lehce horší výsledky.

Srovnání tohoto přístupu a původního přístupu, kde výstupem byly pouze chodidla (*Model3*) lze vidět na obrázku 6.7. Je zde vidět, že tento přístup má tedy lepší výsledky. Což naznačuje, že by využití bodů při trénování mohlo mít pozitivní následky.



Obrázek 6.6: Výsledky modelu při kombinaci bodů na výstupu.



Obrázek 6.7: Srovnání přístupu kombinace bodů na výstupu a výstupu pouze chodidel.

6.4 Výsledky modelů

V následující části jsou znázorněny výsledky jednotlivých modelů (a tedy jednotlivých experimentů) vůči validačnímu datasetu. Nejdříve je zde tabulka 6.1 s nejnižší hodnotou MSE (vzorec 3.7) jednotlivých modelů, která značí rozdíl predikovaných dat od reálných dat. Lze si zde všimnout, že *Model3* dosahuje nejnižší přesnosti v porovnání vůči ostatním experimentům. Dále je zde jeho vylepšená varianta *ModelCombineOut*, která má vyšší přesnost se zachováním stejného vstupu.

Dále jsou zde modely, které měly na vstupu zbylé anotované body dané osoby. Byly zde otestovány 2 přístupy, a to kombinace obrazu a bodů na vstupu a pouze body na vstupu. Kde daná kombinace by mohla dosáhnout nejlepší přesnosti. Každopádně dle výsledků, nejvyšší přesnosti bylo dosaženo pomocí pouze bodů. Lze zde vidět značné rozdíly v přesnosti. Kde závisí na tom, jestli byly na vstupu body včetně kolen nebo bez kolen. Díky těmto bodům byla značně zvýšena přesnost, tyto body ale nelze běžně očekávat na vstupu. Tyto experimenty vedly k vzniku předem zmíněného modelu *ModelCombineOut*, který dosahuje

lepších výsledků než původní model, přestože nedosahuje takové přesnosti, jako modely, které mají na vstupu body.

V tabulce 6.2 je možné vidět odchylky (vzorec 6.1) jednotlivých predikovaných hodnot a reálných hodnot. Odchylka značí jak moc se odlišují hodnoty od průměru a byla počítána jako odmocnina rozptylu, který byl získán pomocí funkce `tf.nn.moments`. Lze zde vidět, že odchylka pro hodnoty x je vyšší jak pro hodnoty y, pravděpodobně z důvodu, že závisí, jestli je osoba na snímku otočená směrem vpřed nebo vzad. Toto ovlivní, na které straně snímku bude levé a pravé chodidlo. Výsledky jsou podobné, jako tomu bylo u předchozí tabulky, kde modely s vyšší přesností mají více podobný rozptyl vůči reálným datům. Zajímavé můžou být rozdíly mezi hodnotami x a y, kde například lze vidět vyšší „zlepšení“ hodnoty y než hodnoty x u modelu *Model3* a *ModelCombineOut*.

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x_i})^2} \tag{6.1}$$

Zdroj dat	**MSE**
Model3	0,024817701
ModelCombineOut	0,020261737
ModelKPS	0,007735877
ModelKPS (no knees)	0,015383153
ModelCombine	0,007772393
ModelCombine (no knees)	0,015758291

Tabulka 6.1: Nejnižší MSE na validačním datesetu.

	Left foot		Right foot	
Zdroj dat	**x**	**y**	**x**	**y**
Validation dataset	0,220789855	0,082473	0,231563	0,090078
Model3	0,125629495	0,00963	0,139725	0,009001
ModelCombineOut	0,132128044	0,043056	0,147186	0,042013
ModelKPS	0,193133607	0,063165	0,199595	0,074209
ModelKPS (no knees)	0,163837175	0,055074	0,17271	0,062538
ModelCombine	0,191205335	0,054404	0,199143	0,078781
ModelCombine (no knees)	0,146408948	0,046017	0,165995	0,054115

Tabulka 6.2: Odchylky z validačních dat a predikce modelů.

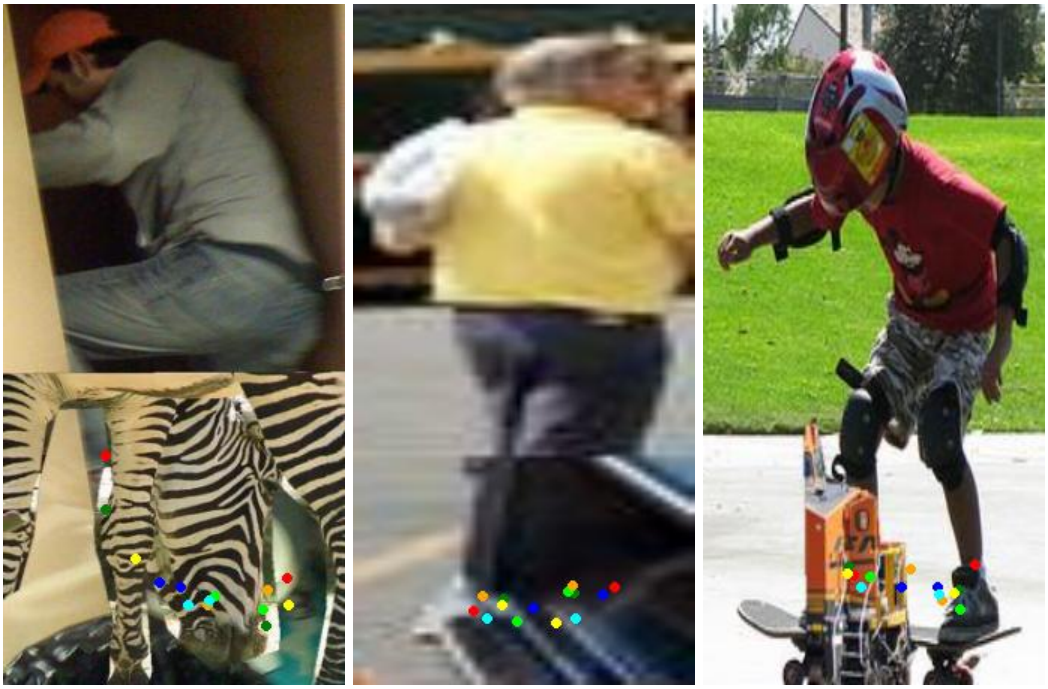
6.4.1 Výsledky na snímcích

Samotná predikce poté značně závisí na daném snímku a v jaké póze je osoba. Na ukázku jsem zde vykreslil všech 6 druhů přístupu a reálné hodnoty různými barvami.

- Reálné hodnoty - reálné umístění chodidel
- Model3 - vstupem je pouze obraz a výstupem pouze chodidla
- ModelCombineOut - vstupem je pouze obraz a výstupem je kombinace chodidel a zbylých bodů

- ModelCombine - vstupem je jak obraz tak zbylé body (bez kolen)
- ModelCombine - vstupem je jak obraz tak zbylé body (včetně kolen)
- ModelKPS - vstupem jsou pouze body (bez kolen)
- ModelKPS - vstupem jsou pouze body (včetně kolen)

Pro ukázku jsem zde vybral 3 vstupní snímky, které je možné vidět na 6.8. Kde si lze všimnout, že hned na prvním obrázku jsou predikce značně nepřesné, pravděpodobně z důvodu neobvyklé pózy. Na dalších 2 obrázcích se jedná o dostatečně běžné pózy a predikce jsou zde poté lepší. Nejdůležitější body jsou poté červená dvojice, značící reálnou pozici. Modrá značící *Model3* a světle modrá značící vylepšenou predikci, stále za použití pouze obrazu. Dále si lze všimnout značného rozdílu mezi oranžovou a žlutou a stejně tak zelenou a světle zelenou. Kde jsou tyto rozdíly z důvodu vynechání či zahrnutí bodů kolen na vstupu. Kde právě pokud byly zahrnuty i tyto body, jsou predikce většinou dost přesné, naopak s vynecháním už takto přesné nejsou.



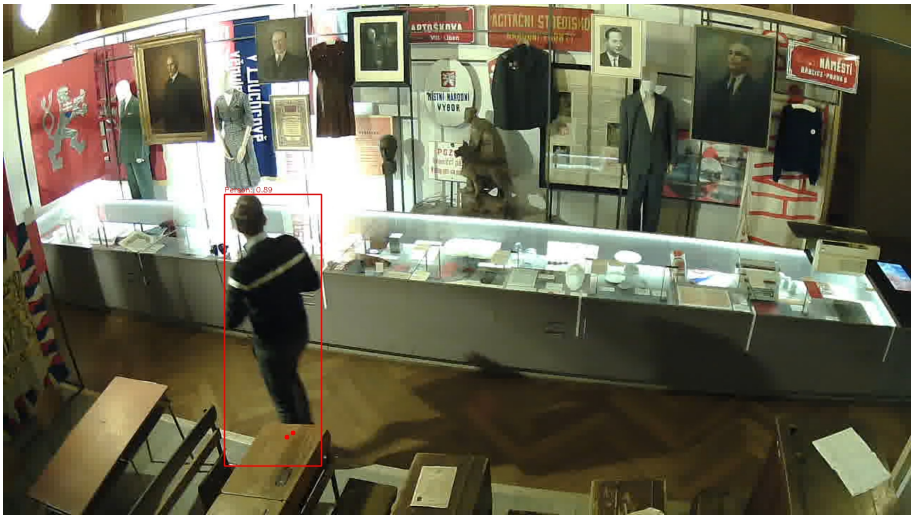
Obrázek 6.8: Ukázka predikce bodů na snímku.

Ukázka reálného případu

V případě reálného využití je potřeba nejdříve provést na snímku detekci osob, jelikož na vstupu modelu se očekává snímek jedné osoby. K tomuto bylo využito modelu YOLOv4[1]. S detekcí ale nastal problém, protože vyznačený ohraničující box, nezahrnuje zakryté nohy a končí tedy nad objektem, který nohy zakrývá. Což je problém, jelikož model očekává na snímku celou osobu, včetně zakrytých nohou. Na obrázku 6.9 lze vidět detekci osoby z kamery muzea, kde nohy zakrývá pravděpodobně exponát. Tento ohraničující box byl

[1]Model YOLOv4 je dostupný zde: https://github.com/AlexeyAB/darknet

následně prodloužen tak, aby pravděpodobně zahrnoval i celé nohy osoby na snímku. Na snímku jsou vyznačeny body z odhadu pozice chodidel červenými body, kde bylo využito modelu *ModelCombineOut*, který na předchozích ukázkách byl značen světle modrou barvou. Rozšířením této práce by mohla být změna vstupu modelu. Místo očekávané celé osoby na vstupu by mohl být celý snímek, na kterém by následně byly predikovány pozice chodidel. Což by vyřešilo problém detekce, která vytvoří ohraničující box bez nohou, když jsou zakryté.



Obrázek 6.9: Ukázka predikce bodů na reálném snímku.

Kapitola 7

Závěr

Cílem této práce bylo vytvoření neuronové sítě, která odhaduje pozici chodidel v obraze i v případě, kdy jsou chodidla zakrytá. Nejdříve bylo potřeba seznámit se s oblastí antropometrie, která souvisí s proporcemi lidského těla a souvisí tedy s odhadem pozice chodidel. Následně se bylo potřeba seznámit se s neuronovými sítěmi, kde byl popsán základní princip funkčnosti neuronových sítí s detaily pro regresní úlohy a zpracování obrazu a se shrnutím existujících řešení pro detekci bodů na lidském těle.

Výsledná neuronová síť byla implementována a natrénována v jazyce Python pomocí rozhraní Keras. Pro trénování sítě byl vytvořen nový dataset. Na kterém byly upraveny snímky, tak aby byly zakryté nohy, za pomocí plně automatického zpracování COCO datasetu. Díky čemuž by mělo být snadné daný dataset rozšířit či modifikovat. U trénování nastal problém, kde se výchozí model neučil. Tento model byl postupně upravován na základě trénování, až do jeho finální podoby, kde model dosáhl hodnoty validační *loss* 0,0248. Dále zde byly provedeny experimenty, za účelem srovnání a zvýšení přesnosti odhadu pozice chodidel. V těchto experimentech bylo využito převážně ostatních významných bodů na těle, které byly součástí datasetu. Z výsledků lze vidět, že značně závisí na tom, zda síť zná nebo nezná pozici bodů kolen. Na základě těchto experimentů byl model dále upraven tak, aby využíval ostatní body na lidském těle při trénování, kde dosáhl hodnoty validační *loss* 0,0203.

Jako rozšíření by mohlo být značné zvýšení počtu anotací v datasetu ve stejném formátu získáním dalších snímků, kde větší velikost datasetu by mohla pomoct zvýšení přesnosti. Dalším rozšířením by mohlo být zaměření se na ostatní body na lidském těle, kde jejich znalost značně zvyšuje přesnost. Každopádně při zpracování obrazu nelze tyto body očekávat na vstupu. Proto by bylo možné mít tyto body také na výstupu sítě a bylo by možné značně navýšit velikost datasetu o snímky neobsahující celou osobu, ale pouze její část. Toto přesnější určení ostatních bodů by poté mohlo vést ke zvýšení přesnosti odhadu pozice samotných chodidel.

Literatura

[1] *International Space Station Flight Crew Integration Standard (NASA-STD-3000/T): International Space Station*. NASA, 1995 [cit. 2021-11-11].

[2] BBC. *Considering usability when designing* [online]. [cit. 2022-01-06]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/z42j2nb/revision/3.

[3] Berga, M. a Coelho, P. *PYTORCH VS TENSORFLOW: COMPARING DEEP LEARNING FRAMEWORKS* [online]. 2021 [cit. 2022-01-17]. Dostupné z: https://www.imaginarycloud.com/blog/pytorch-vs-tensorflow/.

[4] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S. a Sheikh, Y. A. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2019, [cit. 2022-01-09].

[5] Chittawatanarat, K., Pruenglampoo, S., Trakulhoon, V., Ungpinitpong, W. a Patumanond, J. Height prediction from anthropometric length parameters in Thai people. *Asia Pacific journal of clinical nutrition*. 2012, 21 3, [cit. 2021-12-08].

[6] Council, M. R. *Introduction* [online]. 2016 [cit. 2021-11-11]. Dostupné z: https://dapa-toolkit.mrc.ac.uk/anthropometry/introduction/anthropometry.

[7] Cronin, N. Using deep neural networks for kinematic analysis: Challenges and opportunities. *Journal of Biomechanics*. Květen 2021, sv. 123, s. 110460, [cit. 2022-01-07]. DOI: 10.1016/j.jbiomech.2021.110460.

[8] Dictionary, B. *Anthropometry* [online]. 2017 [cit. 2021-11-11]. Dostupné z: https://biologydictionary.net/anthropometry/.

[9] Dumoulin, V. a Visin, F. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*. mar 2016, [cit. 2022-04-10].

[10] Ferrie, J. The irresistible rise of the Cohort Profile. *International journal of epidemiology*. Srpen 2012, sv. 41, s. 899–904, [cit. 2022-01-04]. DOI: 10.1093/ije/dys119.

[11] Fridman, M. *7 Figure Drawing Proportions to Know* [online]. [cit. 2022-01-06]. Dostupné z: https://www.thedrawingsource.com/figure-drawing-proportions.html.

[12] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016.

[13] Google. *Colaboratory* [online]. [cit. 2022-04-24]. Dostupné z: https://research.google.com/colaboratory/faq.html.

[14] GOYAL, C. *Complete Guide to Gradient-Based Optimizers in Deep Learning* [online]. 2021 [cit. 2022-01-08]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-gradient-based-optimizers/.

[15] HOFESMANN, E. *How to work with object detection datasets in COCO format* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://towardsdatascience.com/how-to-work-with-object-detection-datasets-in-coco-format-9bf4fb5848a4.

[16] KOSTADINOV, S. *Understanding Backpropagation Algorithm* [online]. 2019 [cit. 2022-01-08]. Dostupné z: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd.

[17] LIN, T., MAIRE, M., BELONGIE, S. J., BOURDEV, L. D., GIRSHICK, R. B. et al. Microsoft COCO: Common Objects in Context. *CoRR*. 2014, abs/1405.0312. Dostupné z: http://arxiv.org/abs/1405.0312.

[18] LIU, R., LEHMAN, J., MOLINO, P., SUCH, F. P., FRANK, E. et al. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. *CoRR*. 2018, abs/1807.03247, [cit. 2022-04-17]. Dostupné z: http://arxiv.org/abs/1807.03247.

[19] MOLENBROEK, J. a BRUIN, R. de. Enhancing the use of anthropometric data. *Human Factors in Design, Safety, and Management*. Leden 2000, [cit. 2022-01-06].

[20] NIELSEN, M. *Using neural nets to recognize handwritten digits* [online]. 2019 [cit. 2022-01-07]. Dostupné z: http://neuralnetworksanddeeplearning.com/chap1.html.

[21] ODEMAKINDE, E. *Human Pose Estimation with Deep Learning* [online]. 2021 [cit. 2022-01-09]. Dostupné z: https://viso.ai/deep-learning/pose-estimation-ultimate-overview/.

[22] PREEDY, V. R. *Handbook of anthropometry: Physical measures of human form in health and disease*. Springer, 2012 [cit. 2021-11-11].

[23] SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks* [online]. 2018 [cit. 2022-01-10]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[24] SHARMA, S. *Epoch vs Batch Size vs Iterations* [online]. 2017 [cit. 2022-01-08]. Dostupné z: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9.

[25] SHUKLA, L. *Designing Your Neural Networks* [online]. 2019 [cit. 2022-01-08]. Dostupné z: https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed.

[26] TENSORFLOW. *Pose Detection* [https://github.com/tensorflow/tfjs-models/tree/master/pose-detection]. [cit. 2022-01-09].

[27] UJEVIC, D., ROGALE, D., DRENOVAC, M., PEZELJ, D., HRASTINSKI, M. et al. Croatian anthropometric system meeting the European Union. *International Journal of Clothing Science and Technology*. Květen 2006, sv. 18, s. 200–218, [cit. 2021-11-11]. DOI: 10.1108/09556220610657961.

[28] Utkualp, N. a Ercan, I. Anthropometric Measurements Usage in Medical Sciences. *BioMed Research International*. Srpen 2015, sv. 2015, s. 1–7, [cit. 2022-01-03]. DOI: 10.1155/2015/404261.

[29] WHO. *Body mass index - BMI* [online]. [cit. 2021-12-15]. Dostupné z: https://www.euro.who.int/en/health-topics/disease-prevention/nutrition/a-healthy-lifestyle/body-mass-index-bmi.

[30] Wood, T. *What is the Softmax Function?* [online]. 2020 [cit. 2022-05-12]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[31] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y. a Girshick, R. *Detectron2* [https://github.com/facebookresearch/detectron2]. 2019 [cit. 2022-01-09].

Příloha A

Formát anotace datasetu

Vstupní COCO formát

```
{
    "info": {
        "year": int,
        "version": str,
        "description": str,
        "contributor": str,
        "url": str,
        "date_created": datetime
    },
    "licenses": [
        {
          "id": int,
          "url": str,
          "name": str
        },
        ...
    ],
    "categories": [
        {
            "id": int,
            "name": str,
            "supercategory": str
        },
        ...
    ],
    "images": [
        {
            "id": int,
            "license": int,
            "file_name": str,
            "height": int,
            "width": int,
```

```
            "date_captured": datetime
        },
        ...
    ],
    "annotations": [
        {
            "id": int,
            "image_id": int,
            "category_id": int,
            "bbox": [x, y, width, height],
            "segmentation": RLE or [polygon],
            "area": float,
            "iscrowd": 0 or 1,
            "keypoints": [x1, y1, v1, ...],
            "num_keypoints": int
        },
        ...
    ]
}
```

Výpis A.1: Struktura JSON formátu COCO

Výstupní formát

```
{
    "info": same,
    "licenses": same,
    "images": same,
    "annotations": [
        {
            "id": int,
            "image_id": int,
            "bbox": [x, y, width, height],
            "segmentation": [polygon],
            "keypoints": [x1, y1, x2, y2],
            "other_keypoints": [x1, y1, ...]
        },
        ...
    ]
}
```

Výpis A.2: Struktura výstupního JSON formátu