VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIO ELECTRONICS

SEGMENTACE OBRAZU POMOCÍ STROJOVÉHO UČENÍ
IMAGE SEGMENTATION USING MACHINE LEARNING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Libor Matějek

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Tomáš Bravenec

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Student: Libor Matějek ***ID:*** 209562

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Segmentace obrazu pomocí strojového učení

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte existující možnosti strojového učení, specificky pro sémantickou/instanční segmentaci obrazu. Při návrhu vlastního systému pro segmentaci obrazu se kromě běžného rozpoznání různých objektů, zaměřte i na rozlišení různých částí lidského těla. Dbejte na správné rozlišení částí těl, nezávisle na barvě a tvaru oblečení. Uvažujte a srovnejte možnosti existujících nástrojů a knihoven pro zpracování obrazu, počítačové vidění a strojové učení v programovacím jazyce Python nebo C++.

Proveďte detailní testování navrženého systému. Vyhodnoťte rychlost a přesnost detekce jak na statických snímcích, tak na video sekvencích s rozmanitým okolním prostředím a otestujte vliv osvětlení na správnou segmentaci. K navrženému kódu vytvořte dokumentaci a zveřejněte jej na GitHubu či GitLabu.

DOPORUČENÁ LITERATURA:

[1] OpenCV [online]. 2018 [cit. 2020-05-11]. Dostupné z: http://opencv.org/

[2] Image segmentation in 2020: Architectures, Losses, Datasets, and Frameworks [online]. 2020 [cit. 2020-0--11] Dostupné z: https://neptune.ai/blog/image-segmentation-in-2020

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 27.5.2021

Vedoucí práce: Ing. Tomáš Bravenec

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda rady studijního programu

Abstrakt

Tato práce se zabývá strojovým učením a jeho aplikací do oblasti segmentace obrazu a rozpoznávání objektů. V práci je popsána základní terminologie související se strojovým učením a dat, které s ním souvisí. Dále se zaměřuje na biologickou podstatu neuronu a jeho technologické aplikace. Jsou zde přiblíženy základní druhy neuronových sítí a pro zpracování obrazu stěžejní konvoluční neuronová síť. Práce také uvádí využívané architektury konvolučních neuronových sítí. Dále navazují metody předzpracování obrazu před konvoluční sítí R-CNN. Následně jsou rozebrány některé z datasetů vhodných pro rozpoznání obrazu. Implementace je pak realizována v jazyce Python s podporou frameworku PyTorch od Facebooku.

Klíčová slova

Strojové učení, neuronové sítě, segmentace obrazu, rozpoznání obrazu, R-CNN, PyTorch, Python

Abstract

This work deals with machine learning and its application in the field of image segmentation and object recognition. The thesis describes the basic terminology related to machine learning and data related to it. It also focuses on the biological nature of the neuron and its technological applications. The basic types of neural networks and the key convolutional neural network for image processing are described. The work also presents the used architectures of convolutional neural networks. Then follow the methods of image preprocessing before the convolutional network R-CNN. Subsequently, some of the datasets suitable for image recognition are analyzed. The implementation is then realized in Python with support for the PyTorch framework from Facebook.

Keywords

Machine learning, neural networks, image segmentation, image recognition, R-CNN, PyTorch, Python

Bibliografická citace

MATĚJEK, Libor. Segmentace obrazu pomocí strojového učení [online]. Brno, 2021 [cit. 2021-05-26]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/133592. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky. Vedoucí práce Tomáš Bravenec.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Libor Matějek*
VUT ID studenta:	*209562*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2020/21*
Téma závěrečné práce:	*Segmentace obrazu pomocí strojového učení*

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

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 26. května 2021 ----------

podpis autora

Poděkování

Děkuji vedoucímu semestrální práce Ing. Tomáši Bravencovi za odbornou pomoc v oblasti tématu mé práce i rady spojené s formátováním a formální podobou.

V Brně dne: 26. května 2021 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Segmentace obrazu pomocí strojového učení se opírá o metody zpracování obrazu jako takového a deep learningu jakožto podobor strojového učení. Ten se svoji podstatou snaží přiblížit funkci lidského mozku. Jedná se o v dnešní době velmi využívanou a žádanou specializaci. Produktem mohou být implementace do systémů pro zabezpečení, autonomního řízení nebo do systému ve zdravotnictví.

Nedílnou součástí každé umělé inteligence jsou neuronové sítě. V oblasti zpracování obrazu jsou stěžejní konvoluční neuronové sítě, které za pomocí CPU nebo GPU s CUDA architekturou, dokáží vytěžit informaci z obrazu a učit se z ní.

Technologickým krokem vpřed jsou metody vyjímání informativně „zajímavých“ oblastí na obrázcích a tedy aplikace konvoluční sítě cíleně. Průkopníky pokroku v tomto směru jsou organizace Facebook a Google, které svými frameworky PyTorch (Facebook) a TensorFlow (Google) zjednodušily práci s prvky jako jsou neuronové sítě jako takové a tensory, jež představují transformaci obrázku vhodnou pro vstup do neuronových sítí.

Záměrem mé práce je přiblížit strojové učení a základy neuronových sítí, dále jednoduše popsat strukturu již existujících variant konvolučních neuronových sítí a také možnosti segmentace obrazu.

Mým úkolem je získané informace dále implementovat do programu ve volném jazyce Python, který bude schopen rozpoznat obraz včetně detailu na úrovni lidských končetin. Program je tvořen pod záštitou frameworku PyTorch.

1. STROJOVÉ UČENÍ

Termín Strojové učení byl poprvé použit v roce 1959 američanem Arthurem Samuelem, odborníkem v oboru počítačových her a umělé inteligence.
Matematická definice přišla na svět v roce 1997. Jejím autorem byl Tom Mitchell. Definice říká, že „počítačový program se učí ze zkušenosti E s ohledem na úkol T a měřítko M. Jeho výkon P na T se s postupem zkušeností zlepšuje“.
Strojové učení se řadí mezi kategorie umělé inteligence. Tato oblast je svojí podstatou zaměřena na učení se, rozšiřování a zdokonalování počítačových programů. [1]

1.1 Rozdělení strojového učení

Realizace strojového učení se dělí na 3 kategorie podle povahy vstupních a výstupních informací.

1. Učení s učitelem (Supervised Learning) - Algoritmus se učí z ukázkových dat a souvisejících cílových odpovědí. Podobný algoritmus, jako lidské učení. Počítač předpovídá správnou odpověď z dříve získaných obecných pravidel.

2. Učení bez učitele (Unsupervised Learning) - Algoritmus vytváří sám datové vzory na základě určité podobnosti, není tedy nahlíženo do tréninkových dat.

3. Kombinace učení s učitelem a bez učitele (Semi Supervised Learning) - Je dán neúplný „tréninkový plán“. Nejsou dány všechny cílové výstupy, algoritmus tedy kombinuje tvoření nových datových modelů a již zadaných.

4. Zpětnovazebné učení (Reinforcement Learning) - Přirovnáme-li tento druh učení k člověku, nejlépe bychom jej popsali jako pokus-omyl. Algoritmus se učí z předchozích pokusů řešení problému. Nová iterace algoritmu zohledňuje úspěšnost té předchozí a učí se. [1]

1.2 Základní úlohy strojového učení

1. Klasifikace (Classification) - Vstupní data jsou třízena do tříd, příkladem může být třídění mailů do různých kategorií (gmail) – reklama, spam, primární důležité maily. Jedná se o supervised learning.

2. Regrese (Regression) - Také supervised learning, avšak výstup je spíše spojitý. Dle známých hodnot vytváří funkci, která pro neznámý vstup vrací pravděpodobný výstup. Například učení se z historických trendů trhu a předpověď jeho vývoje.

3. Shlukování (Clustering) - Jedná se o učení bez učitele. Jak název napovídá, vstupní data se „shlukují“ do skupin dle svojí podobnosti. Toto shlukování může být založeno na základě barvy, tvaru, velikosti. [1]

1.3 Terminologie související se strojovým učením

Model
Konkrétní struktura naučených dat určitým algoritmem.

Vlastnosti (Feature)
Měřitelné vlastnosti vstupních dat. Podle těchto dat se řídí algoritmus. Vlastnosti jsou psány ve tvaru vektorů, které přivádíme do modelu. Příklad vlastností může být tvar, barva, velikost. Správný výběr zohledněných vlastností je nezbytný pro samotné fungovaní rozlišovacího algoritmu.

Štítek (Label)
Hodnota výstupu získaná použitým algoritmem. U geometrických tvarů by mělo dojít k „rozpoznání“ čtverce, trojúhelníku, kruhu atd.

Trénink (Training)
Účelem tréninku je trénovat pravdivými štítky, vytvořit tak model, který dále dokáže mapovat nová data.

Předpověď (Prediction)
Dostatečně vybavený model predikuje štítky novým vstupním datům dle předchozího učení. [1] [2]

1.4 Data spojená se strojovým učením

Data jakéhokoliv typu (text, zvuk, obrázek) jsou v oblasti strojového učení nesmírně důležitá. Tato data jsou základem samotného trénování navržených modelů různými algoritmy. Jednoduše zde platí přímá úměra mezi množstvím dat a dokonalostí rozpoznávacího modelu.
Na trhu se dají data různých kategorií ve velkém množství považovat za produkt, který si mezi sebou velké společnosti, jako je Facebook, Google, Twitter přeprodávají a zdokonalují tak svoje modely. [2]

Typy dat
Data se na základě svých vlastností rozdělují na celkem 3 kategorie:

- Tréninková data (Training data) - data, která jsou využívána pro učení s učitelem, tedy taková, kde je znám vstup i výstup a model se pomocí algoritmu učí.

- Ověřovací data (Validation data) - data, která se používají k vyhodnocení, spadají do tréninkové sady.
- Testovaná data (Testing data) - pomocí těchto dat zjišťujeme, jakou úspěšností předpovědi náš model disponuje. Na základě predikovaných výstupů a srovnáním se známými správnými výstupy získáme úspěšnost. [2]

Vlastnosti dat

- Objem (Volume) – s technologickým pokrokem se každou sekundu generují obrovské objemy dat
- Druh dat (Variety) - obrázky, videa, zvukové stopy
- Rychlost (Velocity) – rychlost streamování a generování dat
- Hodnota (Value) – možnost odvození datových informací
- Pravdivost (Veracity) – správnost údajů, které zpracováváme [2]

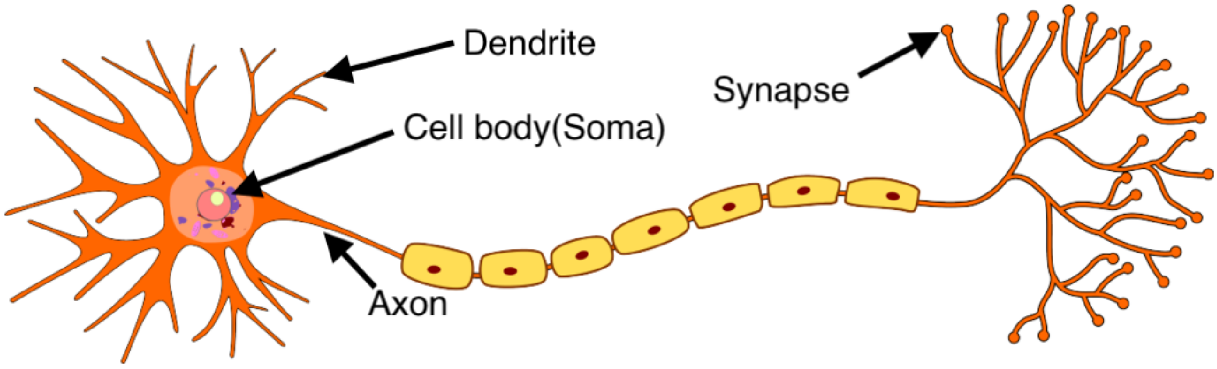
2. DEEP LEARNING

Deep learning je podmnožinou strojového učení, jedná se o evoluci strojového učení. Využívá a tvoří programovatelné neuronové sítě, které umožňují počítači přesně se rozhodnout bez lidské pomoci. [3]

Biologická podstata

Základní funkční jednotkou naší nervové soustavy je neuron, který je schopen generovat elektrické signály zvané akční potenciály, což jim umožňuje rychle přenášet informace. Struktura neuronu je vidět na obrázku 2.1. Téměř každý neuron má 3 funkce nezbytné pro normální fungování všech tělních buněk:

- Přijmout vnější signál.
- Zpracovat signál a rozhodnout, který by měl projít dál.
- Komunikovat s dalšími neurony, svaly nebo žlázami.



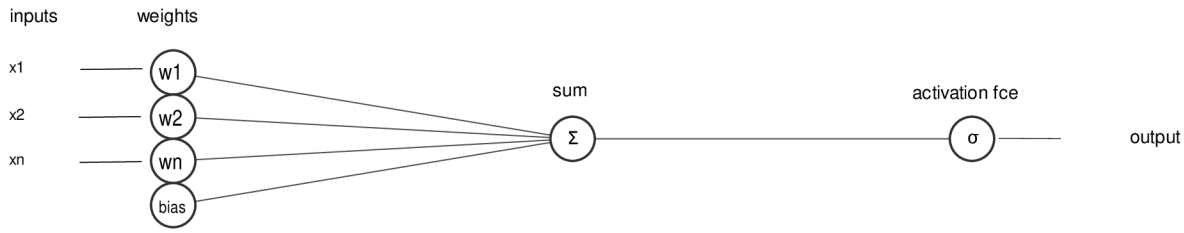
Obrázek 2.1 Biologický neuron. Převzato z https://miro.medium.com

Příjem signálu je realizován dendritem, dále zpracování a vyhodnocení provádí tělo buňky (Soma) a komunikaci s dalšími neurony obstará Axon, který je připojen na synapse, jimiž jsou propojeny neurony. [4] [3]

Technologická aplikace

Umělý neuron, také znám jako perceptron, je základní jednotkou umělé neuronové sítě. Realizován je jako logické hradlo s binárním výstupem. Perceptron vidíme na obrázku 2.2. Každý perceptron:

- Přijímá vstupy.
- Váží jejich důležitost a sčítá je.
- Propouští tento součet skrz nelineární aktivační funkci.



Obrázek 2.2 Perceptron

Pro každý perceptron platí

$$y = f\left(\sum_{i=1}^{n} x_i w_i\right) + bias, \tag{2.1}$$

kde f představuje aktivační funkci, x_i vstupní hodnotu a w_i váhu vstupu. Pokud má neuron dostatečně velký potenciál, je vyslán impuls. Intenzita impulsu závisí na typu aktivační funkce. Každý umělý neuron má také vstup x_0, s vahou w_0, jehož počáteční hodnota je rovna -h (hodnota prahu). Tento vstup se označuje jako $bias$. [5]

Příkladem aktivační funkce může být skoková funkce (hard limiter), kde

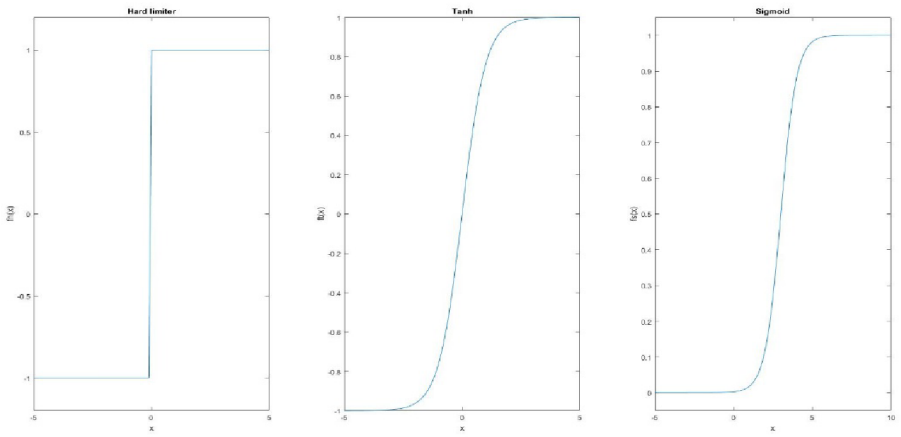
$$f_h(x) = \begin{cases} 1\, pro\ x \geq 0 \\ 0\, pro\ x < 0 \end{cases}. \tag{2.2}$$

Pro n náhodných reálných vstupů $x_0..x_n$ se svými vahami $w_0..w_n$ se neuron začne chovat jako třídička, která reaguje odezvou 0 nebo 1. Výsledkem jsou klasifikované 2 skupiny. [5]

2.1 Aktivační funkce

Mezi nejčastější aktivační funkce, obrázek 2.3, patří :

- skoková funkce (hard limiter)
- tangens hyperbolický (tanh)
- logistický sigmoid (sigmoid)
- ReLU (Rectified Linear Unit for non-linear operation), která je rozebrána níže



Obrázek 2.3 Aktivační funkce

Pro aktivační funkce platí [6]

$$f_h(x) = \begin{cases} 1 \; pro \; x \; \geq 0 \\ 0 \; pro \; x \; < 0 \end{cases}, \tag{2.3}$$

$$f_{th}(x) = \frac{e^{2x} - 1}{e^{2x} + 1}, \tag{2.4}$$

$$f_h(x) = \frac{1}{1 + e^{-x}}. \tag{2.5}$$

2.2 Umělé neuronové sítě (Artifical neural networks)

Pro širší využití umělých neuronů byly vyvinuty umělé neuronové sítě. Mezi základní modely neuronových sítí patří:

Perceptron

Nejjednodušší neuronovou sítí je perceptron model, také znám jako jednovrstvá (Single layer) neuronová síť. Základní perceptron je tvořen pouze:

- vstupní vrstvou
- výstupní vrstvou

Jeho funkcí je přijetí vstupů, zvážení jejích významností a klasifikace. Váha každého vstupu může být <-1,1>. Pomocí aktivační funkce (nejčastěji sigmoid) dojde ke klasifikaci do předem definovaných skupin. Váhy perceptronu se mohou učit pomocí učení s učitelem. Při nerovnosti výstupu perceptronu a očekávaném výstupu se aktivuje funkce založena na Hebbově metodě adaptaci neuronu, která upraví váhy podle chyby mezi predikovaným výstupem a očekáváným výstupem. [7]

Obnova vah perceptronu

Pro obnovu vah perceptronu platí

$$w_i \leftarrow w_i + \Delta w_i , \tag{2.6}$$

$$\Delta w_i = \eta(t-o)x_i , \tag{2.7}$$

kde η je učící koeficient, t správná hodnota, o predikovaná hodnota, x_i je vstupní hodnota

Z důvodu omezené využitelnosti jednoduchého perceptronu došlo k rozšíření do více vrstvého perceptronu (Multi layerd perceptron). [8]

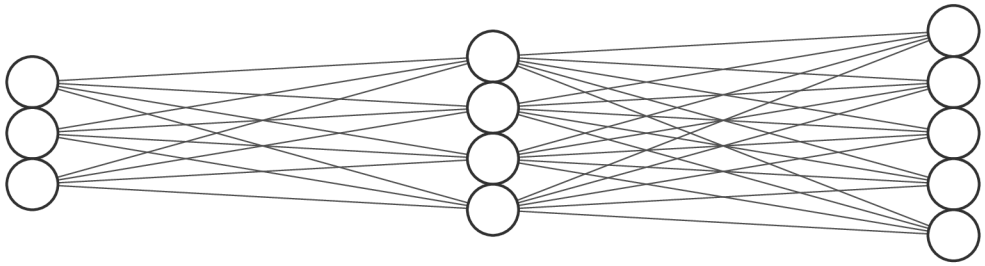
Feed Forward (FF):

Tato vícevrstvá neuronová síť je složena z jednoduchých neuronů, které zastávají funkci základní pracovní jednotky. Tyto jednoduché neurony jsou organizovány do vrstev. Každý neuron je propojen s každým neuronem předchozí vrstvy. Spojení jsou ovlivněny váhami (silou), nejsou tedy ekvivalentní. Váhy propojení v síti představují natrénované vědomosti. Tento typ neuronové sítě je tvořen vstupní vrstvou, jednou nebo více skrytými vrstvami a výstupní vrstvou. Pokud síť obsahuje víc, než jednu skrytou vrstvu, hovoříme o Deep Feed Forward (DFF). V této síti se informace šíří pouze jedním směrem od neuronů vstupní vrstvy přes skryté vrstvy do výstupní vrstvy, kdy v síti neexistuji žádné smyčky. [9] [10]

Backpropagation

Backpropagation je algoritmus pro učení s učitelem umělých neuronových sítí pomocí gradientního sestupu. Gradient je vypočítán vzhledem k chybové funkci a vahám neuronů a zpětně aplikován pro úpravu těchto vah.Tento algoritmus byl poprvé představen v 60. letech 19. století, ale popuralizován až o 30 let později vědci Rumelhartem, Hintonem a Williamsem. [11]

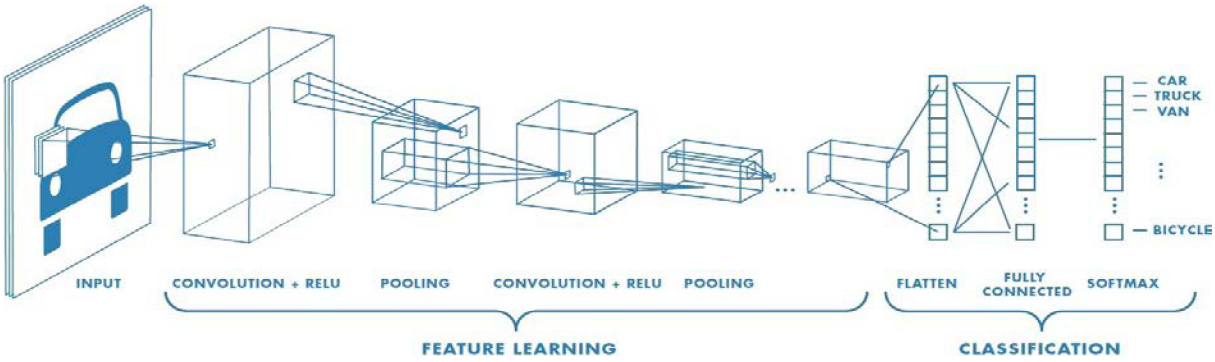
2.3 Konvoluční neuronové sítě (ConvNet/CNN)



Obrázek 2.4 Vícevrstvá neuronová síť

Konvoluční neuronové sítě viz. obrázek 2.4 a obrázek 2.5, je typ Deep learning algoritmu, který je schopen přijmout vstupní obrázek, zpracovat jeho znaky, rozpoznat objekty na obrázku a dokonce jim přiřadit význam. CNN oproti jiným druhům neuronových sítí nevyžaduje téměř žádný pre-processing.

První architekturu konvoluční sítě navrhl Yann Le Cun. Jeho architektura odráží procesy v lidském mozku, kdy jeden neuron stimuluje další neurony ve svém okolí (1980). [12] [13]



Obrázek 2.5 Konvoluční neuronová síť. Převzato z :"https://towardsdatascience.com/"

CNN tvoří:

- Konvoluční vrstva (Convolutation Layer).
- Škálovací vrstva (Pooling layer).
- Plně propojená vrstva (Fully connected layer).

Konvoluční vrstva

Jedná se o první vrstvu celého CNN systému, která extrahuje vlastnosti z vloženého obrázku, který je v prvotní fázi složen z tří RGB vrstev. Konvoluční vrstva dokáže odhalit vztahy mezi pixely pomocí matematických operací, které aplikují filtry nebo násobí vrstvy vstupního obrázku maticemi. Tato vrstva může současně redukovat velikost výstupních vrstev mapy vlastností. [12] [13]

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

Obrázek 2.6 Aplikace konvoluční vrstvy

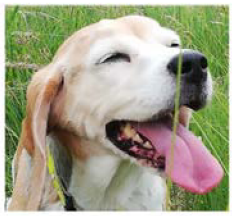
Na vstupní obrázek se opakovaně aplikuje 2D konvoluce, což je násobení segmentu obrázku konvoluční maticí. Konvoluce se postupně provede po celém obrázku. Proces vidíme na obrázku 2.6. Výstup konvoluce tvoří jednu dílčí vrstvu mapy vlastností. Lze volit způsob posunu konvoluční matice po konvolovaném obrázku, definovat okraje konvoluční matice.

V případě, že chceme aplikovat maticový filtr na větší obrázek, je možno zvolit větší krok posunutí. Zvolíme-li například krok posunutí 2 bude se 1. a 2. násobená oblast překrývat pouze 3x1 oblastí.

Mapa vlastností se však skládá z mnoha takových vrstev pro odhalení všech možných informací pro další zpracování. Níže, na obrázku 2.7-2.11, jsou uvedeny příklady základních konvolučních filtrů. [12] [13]

Příklady konvolučních filtrů [14], [12]:

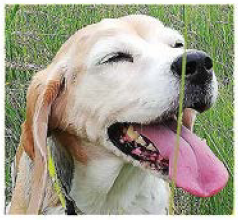
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow$$



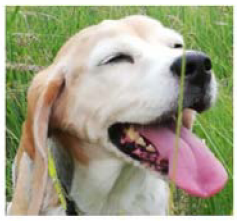
Obrázek 2.7 Konvoluční filtr pro identifikaci obrázku aplikován Matlabem

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}; \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \rightarrow$$

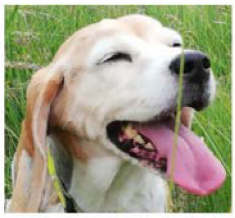
Obrázek 2.8 Konvoluční filtr pro detekci hran aplikován Matlabem

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \rightarrow$$ 

Obrázek 2.9 Konvoluční filtr pro ostření aplikován Matlabem

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow$$ 

Obrázek 2.10 Konvoluční filtr pro rozostření aplikován Matlabem

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \rightarrow$$ 

Obrázek 2.11 Konvoluční filtr pro Gaussovské rozostření aplikován Matlabem

Upravování velikosti (Padding)

Pokud nejsou rozměry obrázku a konvolučního filtru celočíselně dělitelné, dochází k úpravě obrázku dvěma možnými způsoby:

- V obrázku se chybějící prostor doplní nulami (zero padding).
- Část obrazové informace se ořízne (valid padding) [12] [13].

Odstranění záporných hodnot (ReLU)

Záporné hodnoty nejsou špatné pro učení neuronky, právě naopak u ReLU by bylo dobré zmínit i hlavní nevýhodu dying neuronu, kdy se neuron díky tomu, že ReLU zahazuje negativní hodnoty nic nenaučí a bude z něj vystupovat stále jen 0. Proto existuje Leaky ReLU. Ale i přes tu nevýhodu je ReLu stále nejpoužívanější

ReLU (Rectified Linear Unit for non-linear operation) je funkcí, která upravuje jednotlivé vrstvy mapy vlastností tak, že záporné hodnoty vrstvy nahrazuje nulou, což můžeme vidět na obrázku 2.12. Tato operace je prováděna proto, aby byla zachována aplikovatelnost učení v rámci CNN. V některých případech je podmínka nezáporných hodnot natolik omezující, že by se neuronová síť s klasickou ReLU nic nenaučila a vystupovaly by samé nuly, proto existují další varianty jako je Leaky ReLU, ale i přes

tuto nevýhodu je ReLU stále nejpoužívanější. Kromě ReLU je možné použít i funkci sigmoid nebo tanh, ale výkonově nejvýhodnější je právě ReLU. [12] [13]

$$f(x) = \max(0, x)$$

$$\begin{bmatrix} 50 & -15 & 5 & 8 \\ 12 & -9 & 2 & 1 \\ -3 & 45 & -100 & 26 \\ 4 & -40 & 32 & 8 \end{bmatrix} \rightarrow \begin{bmatrix} 50 & 0 & 5 & 8 \\ 12 & 0 & 2 & 1 \\ 0 & 45 & 0 & 26 \\ 4 & 0 & 32 & 8 \end{bmatrix}$$

Obrázek 2.12 Aplikace ReLU

Škálovací vrstva (Pooling layer)

V případě, že jsou jednotlivé vrstvy mapy vlastností moc velké pro vstup do plně propojené vrstvy, využíváme tzv. subsamplingu nebo downsamplingu. Tato metoda zmenší rozměr jednotlivých vrstev mapy vlastností a zároveň zachová klíčovou informaci.

Škálování může být realizováno dle různých pravidel viz obrázek 2.13-2.15 :

- maximální škálování
- průměrovací škálování
- součtové škálování

Maximální škálování propouští nejvyšší hodnotu redukované oblasti, průměrovací škálování průměr hodnot oblasti a součtové škálování jejich součet. [12] [13]

$$\begin{bmatrix} \begin{pmatrix} 14 & 15 \\ 21 & 6 \end{pmatrix} & \begin{pmatrix} 6 & 8 \\ 2 & 8 \end{pmatrix} \\ \begin{pmatrix} 3 & 4 \\ 28 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 5 \\ 12 & 6 \end{pmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} 21 & 8 \\ 28 & 12 \end{bmatrix}$$

Obrázek 2.13 Maximální škálování

$$\begin{bmatrix} \begin{pmatrix} 14 & 15 \\ 21 & 6 \end{pmatrix} & \begin{pmatrix} 6 & 8 \\ 2 & 8 \end{pmatrix} \\ \begin{pmatrix} 3 & 4 \\ 28 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 5 \\ 12 & 6 \end{pmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} 14 & 4 \\ 9 & 6 \end{bmatrix}$$

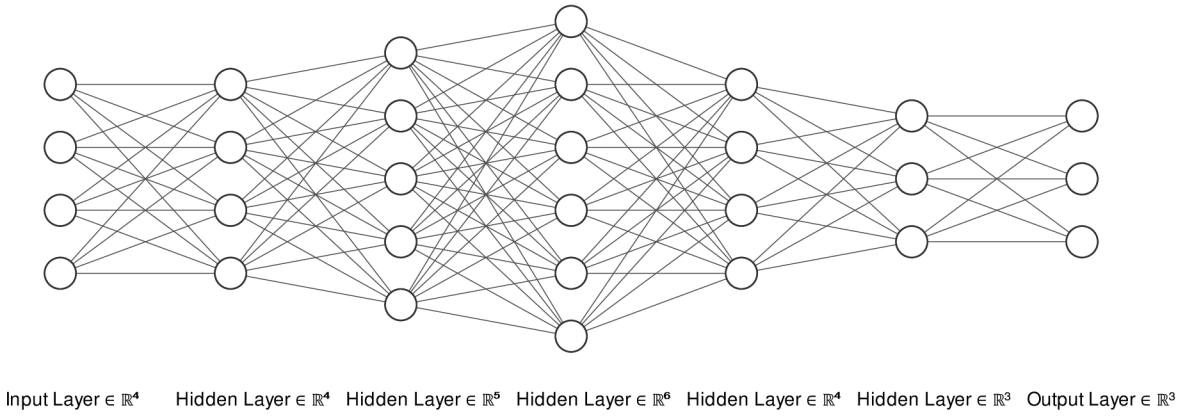
Obrázek 2.14 Průměrovací škálování

$$\begin{bmatrix} \begin{pmatrix} 14 & 15 \\ 21 & 6 \end{pmatrix} & \begin{pmatrix} 6 & 8 \\ 2 & 8 \end{pmatrix} \\ \begin{pmatrix} 3 & 4 \\ 28 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 5 \\ 12 & 6 \end{pmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} 56 & 16 \\ 36 & 24 \end{bmatrix}$$

Obrázek 2.15 Součtové škálování

Plně propustná vrstva (Fully connected layer)

Neuronová síť typu feed forward, do které vstupují vrstvy mapy vlastností ve formě zploštělého vektoru. Například 3x3 vrstvu převedeme do vektoru 9x1. Po naučení sítě trénovacími daty je schopna klasifikovat. Každý neuron výstupní vrstvy sítě odpovídá jedné klasifikační třídě. Jako aktivační funkce neuronů výstupní vrstvy se volí funkce softmax nebo sigmoid. Příklad plně propojené vrstvy můžeme vidět na obrázku 2.16. [12] [13]



Obrázek 2.16 Plně propustná vrstva

Učení konvolučních sítí

Funkcí učení (backpropagation) jsou aplikované filtry upravovány, což zajišťuje vyšší přesnost a informační hodnotu výstupů. Filtry jsou učením řazeny podle důležitosti informace, jež zajišťují.

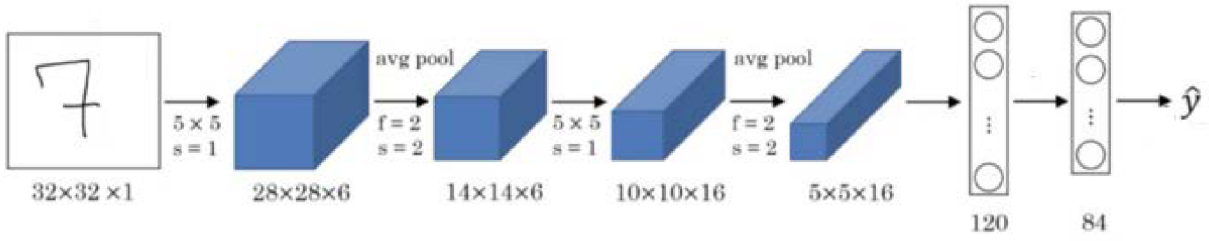
V oblasti učení konvolučních sítí se také v některých architekturách objevuje funkce Dropout, jež zrychluje proces učení a zároveň eliminuje "přeučení". Dropout náhodně vynechává některé neurony, čímž zabraňuje koadaptaci neuronů a tedy přeučení neuronové sítě. [12] [15]

2.3.1 Architektury CNN

Všechny níže zmíněné architektury jsou porovnané v tabulce 2.1.

LeNet-5

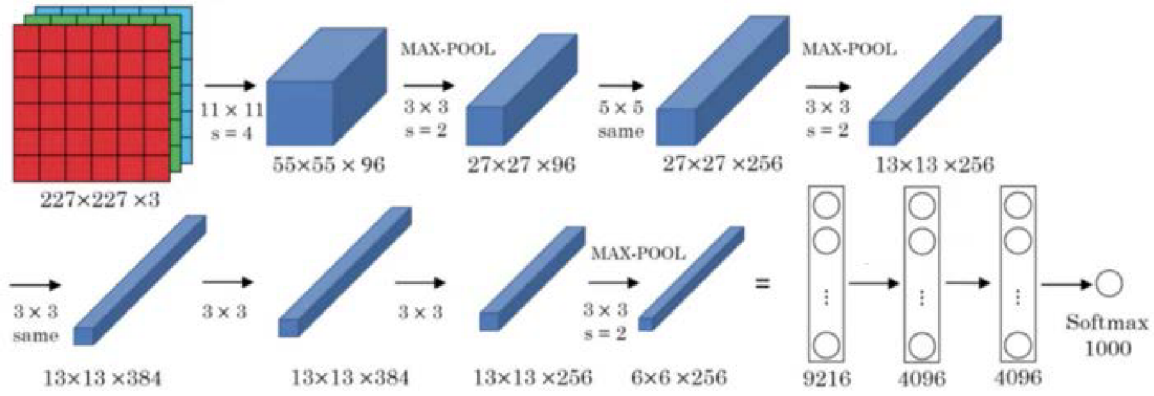
LeNet-5, obrázek 2.17, byla vytvořena vědcem z oboru deep learningu Yannem LeCunem v roce 1998. Byla využita při detekci ručně psaných šeků bank na základě datasetu MNIST. Tato architektura je považována za základnu všech ostatních. [16]



Obrázek 2.17 LeNet-5 Převzato z :"https://towardsdatascience.com/"

AlexNet

AlexNet, obrázek 2.18, byla publikována Alexem Krizhevskym v roce 2012. Tato architektura dosáhla chybovosti 15.3%. AlexNet byla první konvoluční sítí, která využívala GPU pro zvýšení výkonu. Skládá se celkem z 13 vrstev. Každá konvoluční vrstva je sestavou filtrů a nelineární ReLU funkce. Pooling funkce jsou typu Max-pooling. Vstupní velikost snímku je konstantně daná pro správnou funkci plně propojené vrstvy a to 227x227x3. AlexNet také disponuje Dropout funkcí. [17]

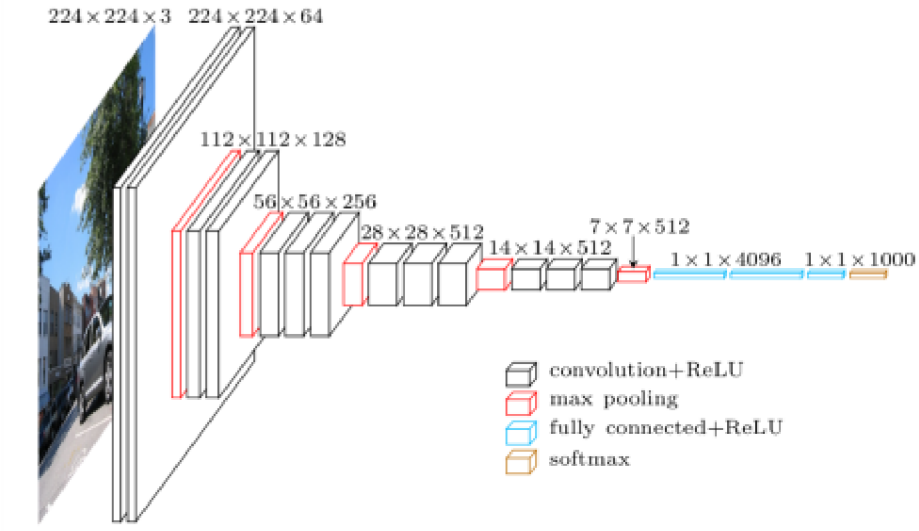


Obrázek 2.18 AlexNet. Převzato z: "https://towardsdatascience.com/"

VGG16 and VGG19

VGG architektura konvoluční sítě, obrázek 2.19, byla představena Karenem Simonyanem a Adrewem Zissermanem v roce 2014. Tento model je charakteristický svou jednoduchostí. Využívá pouze 3x3 konvoluční vrstvy na sobě s postupně zvyšující se hloubkou. Pooling funkce je zde typu max pooling. Na konci se vyskytují 2 plně propojené vrstvy s 4096 neurony a výstup zajišťuje softmax vrstva. Rozdíl mezi VGG16 a VGG19 je v počtu konvolučních vrstev 16 a 19. Nevýhodou VGG

architektury je pomalý trénink a velikost výsledného modelu (přes 533MB pro VGG16). [16]

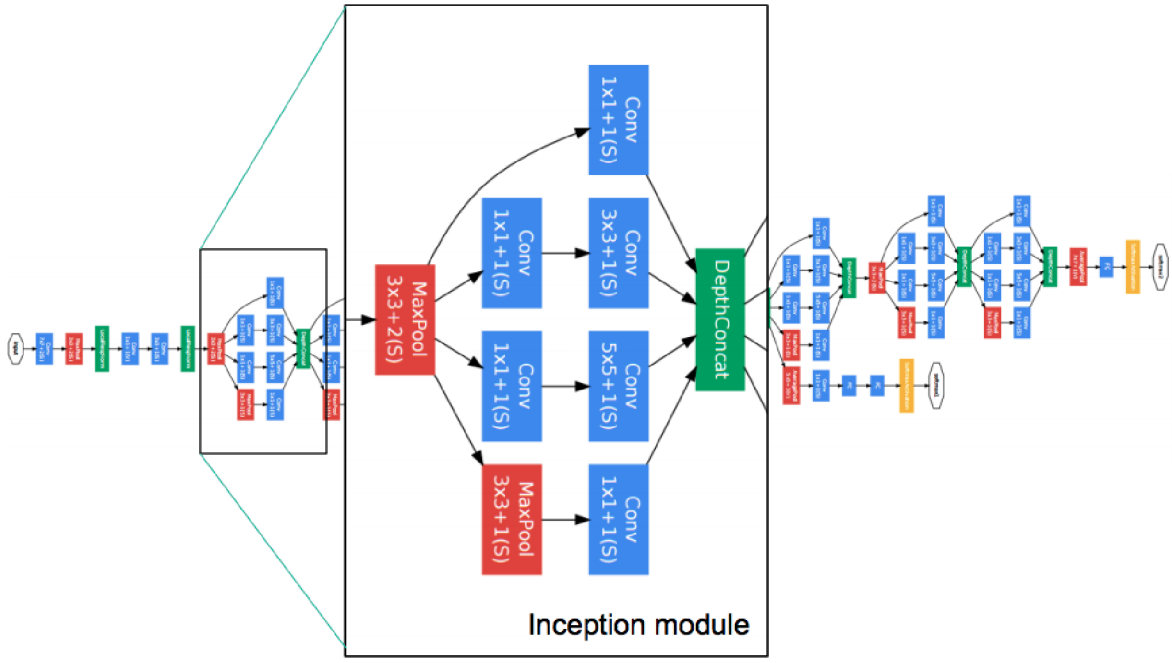


Obrázek 2.19 VGG19. Převzato z: "https://towardsdatascience.com/"

GoogLeNet

GoogLeNet, obrázek 2.20, je typ konvoluční neuronové sítě založena na Inception modulech. Tyto moduly umožňují síti vybírat mezi více velikostmi konvolučních filtrů v každém bloku. Konvoluční bloky na sebe navazují pooling funkcí typu max-pooling. GoogLeNet obsahuje 22 vrstev a dosahuje chybovosti pouhých 6,66 %, přičemž lidská chybovost je kolem 5,1%.

Myšlenkou **Inception** vrstvy je pokrýt větší oblast, ale současně zachovat rozlišení pro detail snímku. Vrstva tedy má za úkol spojit paralelně provedené konvoluce (1x1, 3x3, 5x5). Všechny konvoluční vrstvy jsou učitelné. GoogLeNet využívá 9 inception modulů. [18]

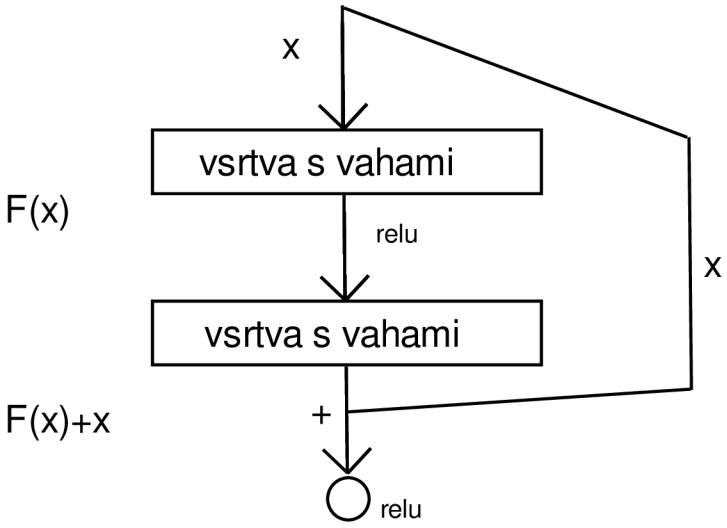


Obrázek 2.20 GoogLeNet. _Převzato z originálního datasheetu [18]

ResNet

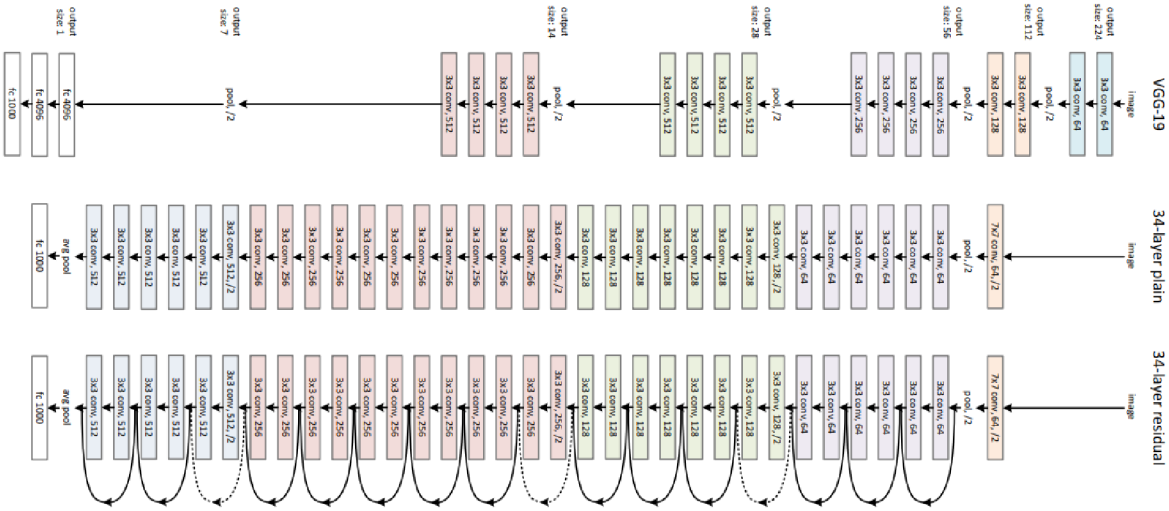
Od architektury CNN AlexNet, která zvítězila v soutěži ImageNet 2012, využívá každá další vítězná architektura více vrstev v hluboké neuronové síti ke snížení chybovosti. Po překonání určité hranice vrstev se však můžeme setkat s problémem zvaným Vanishing gradient. Tento problém má za následek zvýšení míry chyb. Vanishing gradient problém je založen na ztrátě učící informace při derivaci hodnoty aktivační funkce. Je-li vstupní hodnota do, například sigmoid, aktivační funkce příliš vysoká, sigmoid ji zredukuje na hodnotu mezi 0 a 1, což má za následek nízkou hodnotu derivace a malou změnu váhy. [19]

Pro eliminaci Vanishing gradient problému byla v roce 2015 představena Kaimingem He architektura ResNet (Residual Network), kterou lze vidět na obrázku 2.22. Oproti sekvenčních architekturám jako AlexNet nebo VGG využívá tzv. mikro-architekturové moduly, nebo-li sítě v síti. V síti využíváme techniku zvanou přeskakování spojů, viz. obrázek 2.21. Tato technika umožňuje při tréninku přeskočit některé vrstvy a připojit se rovnou na výstup. [16] [20]



Obrázek 2.21 Residual block

Výhodou přidání tohoto přeskakování spojů je ve schopnosti vynechat vrstvu, která by snížila výkon architektury. Výsledky nebyly ovlivněny Vanishing gradient problémem ani při testování s 100-1000 vrstvami na datasetu CIFAR-10. [16] [20]



Obrázek 2.22 ResNet. Převzato z :"https://towardsdatascience.com/"

Rok	Konvoluční síť	Autor	Chybovost[%]
1998	LaNet	Y.LeCun	15.3
2012	AlexNet	A.Krizhevsky	14.8
2014	GoogLeNet	Google	6.67
2014	VGGNet	Simonyan,Zisserman	7.3
2015	ResNet	Keiming He	3.6

Tabulka 2.1 Srovnání architektur

3. POČÍTAČOVÉ VIDĚNÍ

Počítačové vidění je obor, který se v poslední době velmi rychle vyvíjí. Jeho nedílnou součástí je detekce objektů v obraze. Tento podobor se využívá při detekci vozidel (autonomní řízení), u hlídacích algoritmů a podobně.

Rozdílem mezi detekčními algoritmy a jednoduššími klasifikačními algoritmy je konkretizace a ohraničení objektů ve vstupním obraze. Návrhy oblastí pro vstup do plně propojené vrstvy jsou již předdefinované a pro algoritmus „zajímavé“, obsahují objekt.

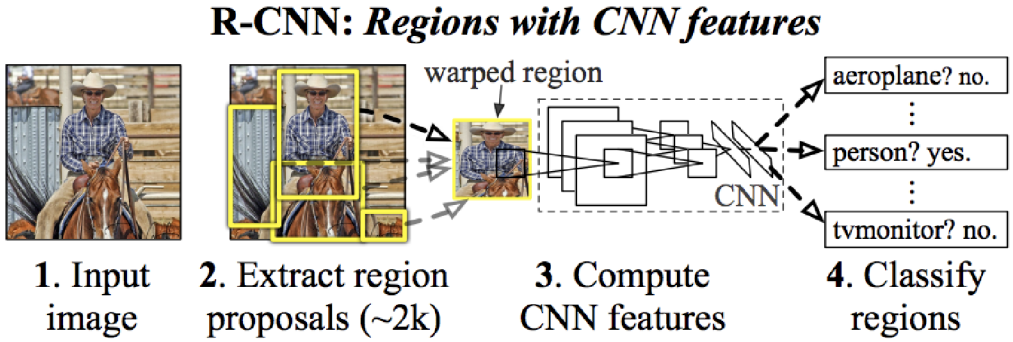
Nabízí se rozdělení obrazů na různé oblasti a použít klasickou konvoluční neuronovou síť (CNN) postupně na každou oblast. Objekty v obraze mohou mít však různá prostorová umístění a různé poměry stran, tzn. bylo by nezbytné vybrat obrovské množství oblastí (regionů), což je výpočetně náročné. Pro snazší nalezení výskytů objektů byly vyvinuty algoritmy, jimiž se budeme zabývat. [21]

3.1 R-CNN (Region based Convolutional Neural Networks)

Abychom se vyhnuli zpracovávání velkého množství oblastí (regionů), můžeme využít metodu Rosse Girsicketa, která využívá selektivního vyhledání a extrahování pouze 2000 regionů z obrázku, viz. obrázek 3.1. Tyto oblastí se nazývají region proposals (návrhy regionů). [21] [22]

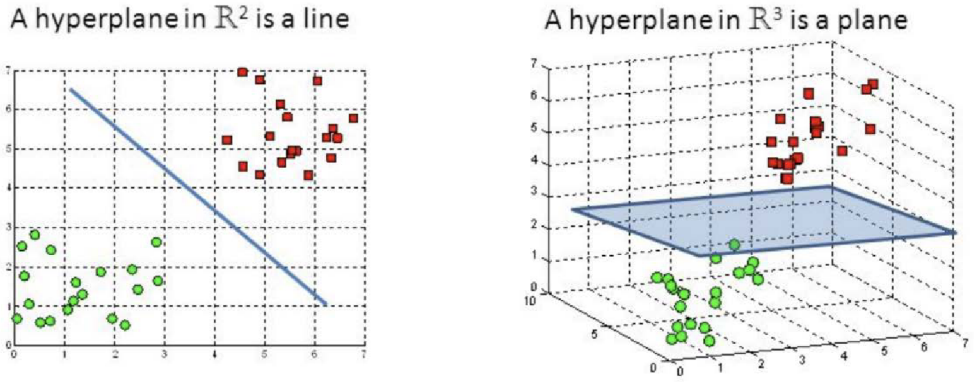
Selektivní vyhledávání

1. Generování počáteční subsegmentace, mnoho návrhů regionů.
2. Pomocí hladového algoritmu rekurzivní kombinace oblastí do větších.
3. Využití vygenerovaných oblastí k produkci finálních návrhů regionů.



Obrázek 3.1 Struktura R-CNN. Převzato z originálního datasheetu [21].

Navržené oblasti jsou dále znormovány do čtverce a přivedeny do konvoluční neuronové sítě. Výstup z této sítě je 4096-dimenzionální vektor vlastností. Můžeme tedy říct, že konvoluční vrstva funguje jako extraktor vlastností z obrázku a vrací jejich hustou vrstvu. Vlastnosti jsou dále přivedeny do SVM (Support Vector Machine), obrázek 3.2, pro rozpoznání nadroviny vlastností, které jasně definují datové body objektu. Tento proces zjednodušeně zjišťuje hranici mezi pozadím a potenciálním objektem. Kromě samotného zjištění, zda se objekt uvnitř navrhované oblasti nachází, algoritmus také vrací hodnoty posunu hranice této oblasti v případě, že by hranice oblasti rozdělovala obsažený objekt. [23]



Obrázek 3.2 Support vector machine. Převzato z "https://towardsdatascience.com/".

Nevýhody R-CNN

- Oproti modernějším metodám stále pomalé, což mimo jiné znemožňuje real-time využití.
- Vyhledávací algoritmus se neučí, chybovost návrhu oblastí je tedy konstantní.

3.2 Fast R-CNN

Rosse Girsicket dále pracoval na svém algoritmu a došel k několika inovacím. Jak název napovídá, jedná se o optimalizaci a zrychlení použitých metod. Rozdílem je prohození kroků zpracování, tedy nejprve se vstupní obraz zpracuje v konvoluční neuronové síti CNN, čímž dostaneme mapu vlastností (feature map). Z této mapy dále algoritmus identifikuje návrhy oblastí a znormuje je na čtverec. Aplikací RoI (Region of interest) vrstvy, která svojí funkcí sdružuje hodnoty z CNN do pevně dané velikosti, dostaneme validní vstup pro plně propojenou neuronovou vrstvu (fully connected layer). Z upraveného vektoru z RoI vrstvy se nakonec pomocí softmax vrstvy předpoví třídy a hodnoty posunutí navrhovaných oblastí. [21] [24]

Softmax vrstva (softmax layer)
Je poslední neuronovou vrstvou klasifikační sítě. Aktivační funkce této vrstvy znormuje všechny vstupní váhy do pravděpodobností, kterých součet je 1. [25]

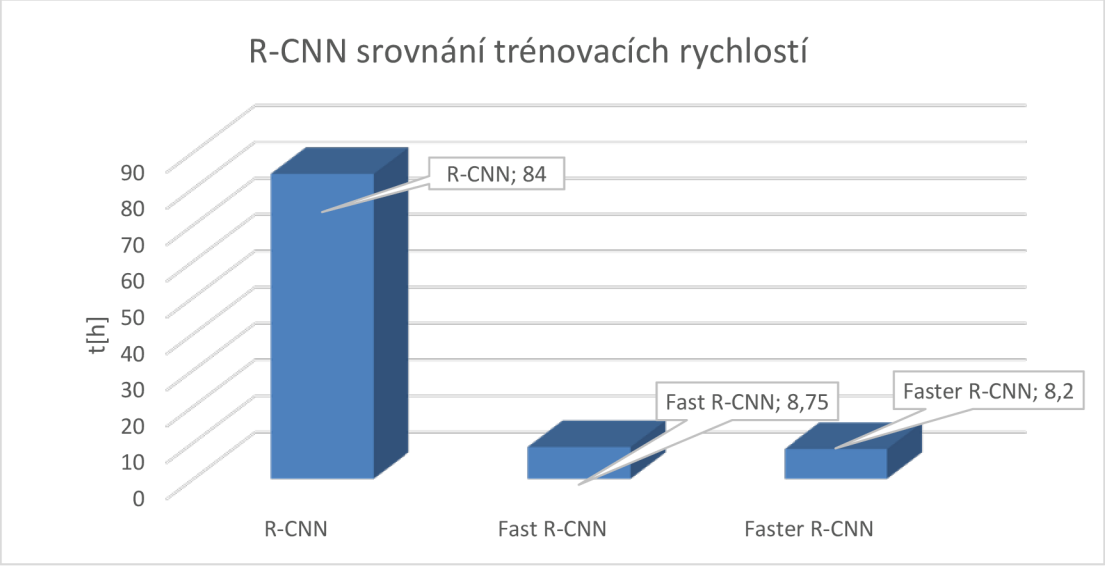
$$f_s = \frac{e^{z_i}}{\sum_{j=1}^{K_w} e^{z_j}}, \tag{3.1}$$

kde z_i je hodnota prvku vektoru vah a K_w počet prvků vektoru vah.

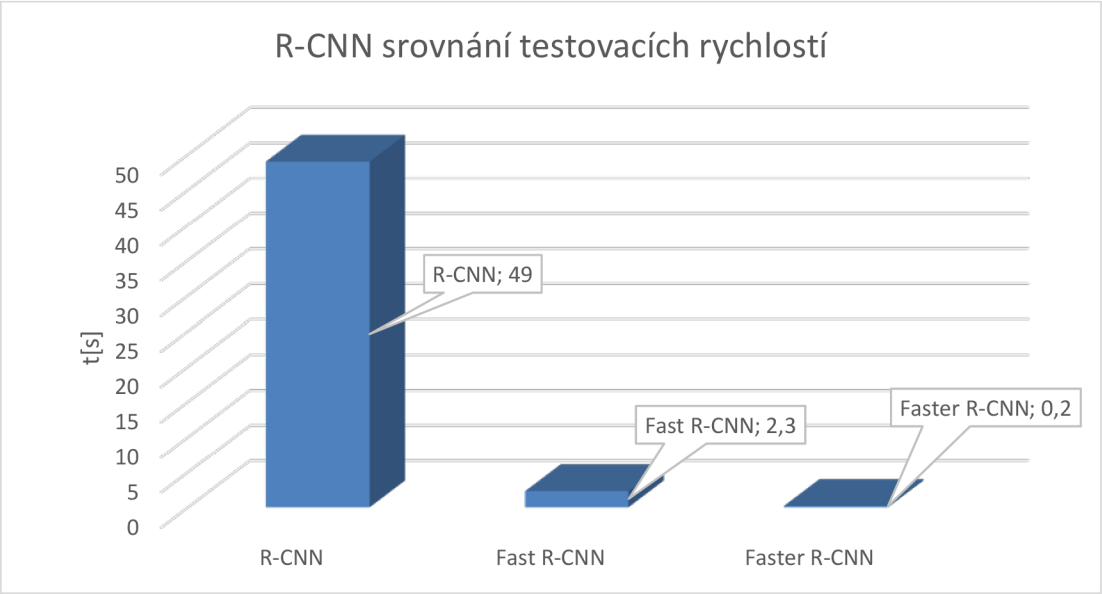
Pro příklad platí, že

$$\begin{bmatrix} 1 \\ 2.8 \\ 0.6 \\ 1.5 \end{bmatrix} \rightarrow \frac{e^{z_i}}{\sum_{j=1}^{K_w} e^{z_j}} \rightarrow \begin{bmatrix} 0.16 \\ 0.48 \\ 0.11 \\ 0.25 \end{bmatrix}. \tag{3.2}$$

Obrovskou výhodou oproti starší R-CNN je to, že již nemusíme aplikovat CNN na každou z 2000 oblastí. Proces je tedy o mnoho rychlejší jak v trénovací, tak testovací relaci. Na obrázku 3.3 a 3.4 je vidět, že nejvýraznější zpomalovací faktor bylo u R-CNN dle předpokladů navrhování oblastí. [21] [24]



Obrázek 3.3 Srovnání trénovacích rychlostí R-CNN



Obrázek 3.4 Srovnání testovacích rychlostí R-CNN

3.3 Faster R-CNN

Oba předchozí algoritmy využívají pomalé vyhledávací algoritmy pro určení navržených oblastí. Tento proces je ve Faster R-CNN, viz. obrázek 3.5, nahrazen metodou programátora jménem Shaoqing Ren. Tato metoda je inovativní ve využívání rozpoznávacích neuronových sítí, které se jsou schopny učit. Tyto sítě se nazývají Region Proposal Network (Síť pro návrh oblastí). Faster R-CNN stejně jako Fast R-CNN nejprve na vstupní obraz aplikuje CNN, z které vystupuje mapa vlastností. Tato mapa je dále zpracována RPN, čímž jsou navrženy oblasti. Tyto oblasti jsou dále znormovány RoI vrstvou do formátu vhodného pro plně propojenou neuronovou síť (fully connected layer). Faster R-CNN je prvním algoritmem, který je schopen pracovat v reálném čase. [21] [26]

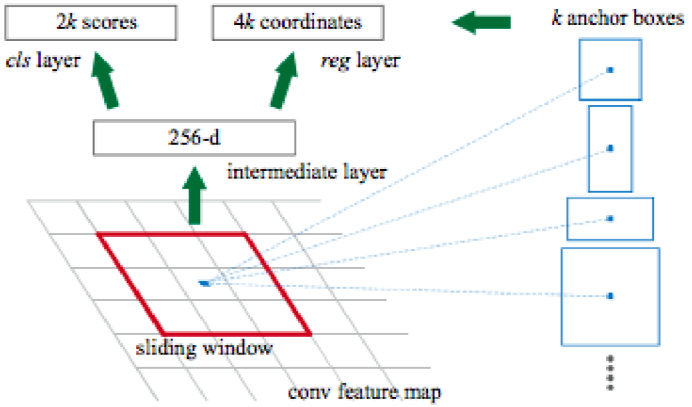
RPN (Region Proposal Network)

Tato návrhová oblastní neuronová síť má specializovanou a jedinečnou architekturu. RPN se skládá z klasifikátoru (classifier) a regresoru (regressor). Architektura jako taková je založena na konceptu kotev (anchors). Tato kotva je centrálním bodem posuvného okna, které je například u ZF modelu, což je rozšíření AlexNetu, 256dimenzionální a 512dimenzionální u VGG. Klasifikátor svojí funkcí určuje pravděpodobnost, zda návrh obsahuje cílový objekt, regresor upravuje souřadnice návrhu. Pro každý obrázek je stěžejním měřítko a poměr stran. V RPN je možné použít celkem 3 měřítka a 3 různé poměry stran, z čehož vychází 9 vzájemných kombinací pro každý pixel. Pro výpočet celkového počtu kotev v obrázku použijeme vztah :

$$number\ of\ anchors = W * H * K_p, \tag{3.3}$$

kde W je šířka obrázku, H je výška obrázku ,K_p je počet kombinací poměr stran/měřítko

RPN využívá namísto pyramidy filtrů, jako tomu bylo v předchozích metodách, pyramidy kotev, což je efektivnější a méně náročné. Výhodou metody je současně její translační invariantnost, objekty jsou tedy rozpoznány bez ohledu na to, zda jsou posunuty nebo otočeny. [27]
Popis funkce:
Prvním krokem je generace kotev na mapě vlastností a na každé z kotev 9 druhů oken (měřítka, poměry stran). Vektory vlastností kotev jsou dále rozděleny do dvou paralelních vrstev. Jedna pro klasifikátor, druhá pro regresor, který upravuje ohraničení.
Klasifikátor – vrstva vrací pravděpodobnost toho, že je aktuální vstup v popředí, tedy obsahuje objekt a pravděpodobnost pozadí. Velikost výstupu tohoto procesu je 2·k parametrů pro počet k oblastí kotev, viz. zobrazena na obrázku 3.5.
Regresor - Vrací 4 předpovězené hodnoty offsetu, tedy hodnoty potřebného posunutí hranic. Výstup má 4·k parametrů pro počet k oblastí kotev, viz. zobrazena na obrázku 3.5. [27]



Obrázek 3.5 Oblasti kotev. Převzato z : "https://medium.com/".

Trénování RPN
Každý bod mapy vlastností obsahuje 9 oblastí kotev, což pro celý obrázek kumuluje obrovské množství informace. Ne každá informace je však relevantní. Zajímají nás pouze ty oblasti, které obsahují objekt, nebo pozadí.
Abychom zjistili, které konkrétní oblasti jsou pro další algoritmus informačně zajímavé, musíme přiřadit každé z nich jeden z 3 druhů štítku založených na IoU (Intersection over Union) [27]

IoU (Intersection over Union)

Obrázek 3.6 IoU

Pro IoU platí

$$IoU = \frac{Area\_of\_Overlap}{Area\_of\_Union}. \tag{3.4}$$

Štítek = 1 (popředí) : Kotvě je přiřazen štítek 1 pokud

- Má nejvyšší IoU s pravdivým ohraničením (ground truth).
- Má IoU s pravdivým ohraničením > 0,7.

Štítek = -1 (pozadí) : Kotvě je přiřazen štítek -1 pokud.

- Má IoU s pravdivým ohraničením < 0.3.

Štítek = 0 (nerelevantní informace) : Kotvě je přiřazen štítek 0 pokud

- Nesplňuje žádnou z přechozích podmínek, takovéto hodnoty tréninkový algoritmus ignoruje.

Správnou formou dat pro učící se RPN je malá várka 256 náhodně zvolených oblastí kotev ze stejného obrázku. Poměr oblastí s pozitivním štítkem by měl být 1:1 vůči oblastem s negativním štítkem. Pokud však není oblastí s pozitivním štítkem dostatek, jsou doplněny těmi s negativním. [28] Podstata IoU je zachycena na obrázku 3.6.

Ztrátová funkce RPN (Loss function)

Každá RPN má svoji ztrátovou funkci, které hodnotu se snažíme minimalizovat.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{3.5}$$

kde i je index oblasti kotvy ve vstupní várce. Každá RPN ztrátová funkce je rozdělena na součet dvou částí. Jedna z částí se týká klasifikátoru, druhá regresoru.

1. **část (klasifikace)**

Proměnná p_i vyjadřuje pravděpodobnost, že vstupní oblast kotvy je v popředí (obsahuje objekt). p_i^* je hodnota štítku podle IoU vysvětlena výše. L_{cls} je křížovou entropickou funkcí, která vrací logaritmickou hodnotu ztráty. $\frac{1}{N_{cls}}$ je normovací činitel, kde N_{cls} je počet prvku ve várce.

2. **část (regrese)**

Zde t_i je 4 rozměrný parametr predikované hodnoty posunutí porovnávaný s t_i^*, pravdivými parametry ohraničení danými trénovacím datasetem. Ztráta $L_{reg}(t_i, t_i^*)$ je rovna $R(t_i - t_i^*)$, což je tzv. Smooth L1 ztráta která je definovaná jako

$$\mathrm{R}; smooth\left(t_i - t_i^*\right) = \begin{cases} \left|t_i - t_i^*\right| & \text{if } |x| > \alpha; \\ \frac{1}{\left|t_i - t_i^*\right|} & \text{if } |x| \leq \alpha; \end{cases}, \tag{3.6}$$

kde α je hyperparametr, obvykle $\alpha = 1$.

Parametry ohraničení jsou definovány jako

$$t_x = (x - x_a)/w_a, \tag{3.7}$$

$$t_y = (y - y_a)/h_a, \tag{3.8}$$

$$t_w = \log(w/w_a), \tag{3.9}$$

$$t_h = \log(h/h_a), \tag{3.10}$$

$$t_x^* = (x^* - x_a)/w_a, \tag{3.11}$$

$$t_y^* = (y^* - y_a)/h_a, \tag{3.12}$$

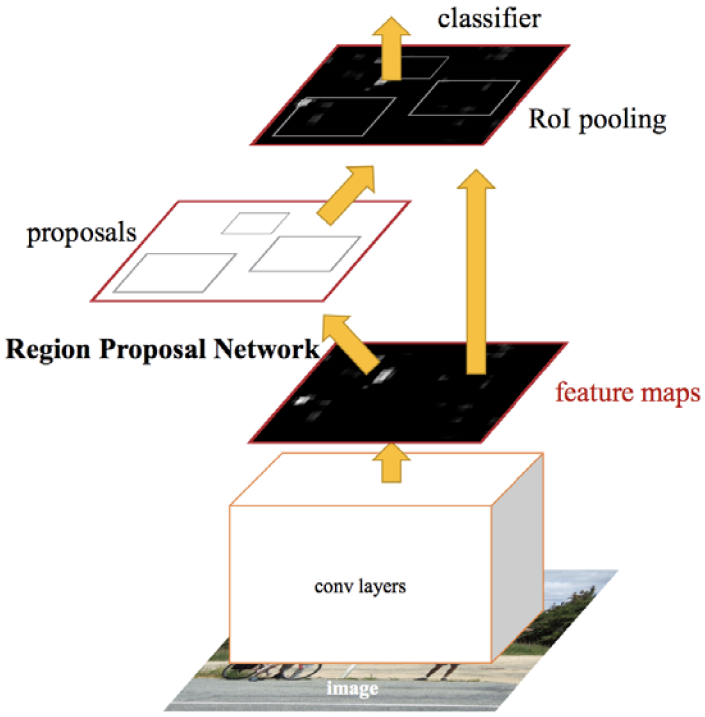
$$t_w^* = \log(w^* / w_a), . \tag{3.13}$$

$$t_h^* = \log(h^* / h_a), \tag{3.14}$$

Kde (x,y) jsou souřadnice středu ohraničení, (w,h) jsou velikosti stran.
(x,y,w,h) definuje parametry predikovaného ohraničení, (x_a , x_b, x_w , x_h) jsou parametry oblasti kotvy, a (x^*,y^*,w^*,h^*) jsou parametry ohraničení dané datasetem.

Ztrátový výpočet je realizován pouze pro pozitivní oblasti kotev. $\frac{1}{N_{reg}}$ je podobně jako u klasifikace normovací činitel, kde N_{reg} je počet všech oblastí kotev. Hodnota λ je standardně 10, jedná se o vyrovnávací činitel. [29] [26]

Potlačení nemaximálních hodnot (Non-Max Suppression)
RPN navrhuje velké množství oblastí, z nichž nejsou všechny dostatečně přesné. Pro potlačení nedostatečně přesných oblastí využíváme metodu **Non-Max Suppression.** Metoda spočívá v sestupném seřazení všech návrhů podle jejich objektového skóre (hodnota IoU s trénovacím ohraničením z datasetu). Dále vybereme ten návrh s nejvyšším IoU skóre a vymažeme všechny ty návrhy, které mají s nejlépe navrženým IoU > 0.5. Tyto návrhy už další krok neovlivňují. Takto iterujeme po všech zbylých návrzích oblastí od nejlépe hodnoceného až po nejhůře hodnoceného. [30]

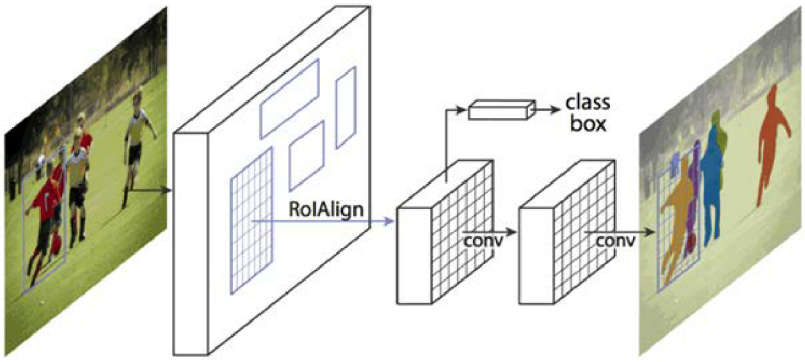


Obrázek 3.7 Struktura Faster R-CNN. Převzato z originálního datasheetu [26]

3.4 Mask R-CNN

Nadstavbou Faster R-CNN jménem Mask R-CNN, ilustrován na obrázku 3.8, získáváme jednotlivé masky kopírující přesně tělo objektu, tedy nejen pouhé hranaté oblasti.

Mask R-CNN nástavba skenuje celou FPN (Feature Pyramid Network), tedy všechny předešlé informace získané z předchozích operací pro zhuštění mapy vlastností, aplikuje další konvoluční vrstvu a generuje body pro masku objektu. [31] [32]



Obrázek 3.8 Struktura Mask R-CNN. Převzato z originálního datasheetu [31]

3.5 Keypoint R-CNN

Detekce klíčových bodů zahrnuje oproti předchozím metodám detekci bodů, jež definují lidské tělo, popřípadě I konstrukci jiných objektů. Body jsou neměnné vůči rotaci, změně velikosti nebo zkreslení obrazu.

Metoda čerpá z definovaných klíčových bodů ze speciálních anotací, kterými je učena neuronová síť. Jedná se o další nástavbu konvoluční a klasifikační vrstvy Faster-RCNN (Mask-RCNN). Ve své práci budu vycházet z datasetu COCO, jež disponuje potřebnými anotacemi. [33]

4. FRAMEWORK

Pro práci s neuronovými sítěmi a deep learnig existuje celá řada frameworků, kde nejznámější a volně dostupné jsou PyTorch a TensorFlow. Framework je nezbytným nástrojem pro trénování i vyhodnocování neuronových sítí. V dnešní době je v souvislosti s jazykem Python tento typ knihoven velice intenzivně využíván i dále rozvíjen.

4.1 PyTorch

PyTorch byl představen v roce 2016 organizací Facebook's AI Research (FAIR). Jedná se o balíček výpočtů založený na programovacím jazyce Python, který umí využít GPU skrz interpretaci pro CUDA jádra na grafických čipech od NVIDIA. Současně je PyTorch jedna z platforem pro Deep learning zaměřených na rychlost a flexibilitu. Hlavní pracovní jednotkou PyTorche jsou tzv. tensory. Jedná se o multidimenzionální pole hodnot. Tensory v tomto frameworku jsou velice podobné polím z knihovny NumPy, avšak s výhodou výše uvedené schopnosti využití GPU. Při budování neuronových sítí PyTorch využívá techniku zvanou automatická diferenciace, která vyhodnocuje derivace funkce. Na začátku tréninku sítě jsou váhy náhodně inicializovány na hodnoty blízké nule a metoda je při průchodu sítí zpět upravuje. Inverzním postupem je upravování vah při průchodu vpřed. [34]

4.2 TensorFlow

TensorFlow je velmi využívaná knihovna pro implementaci strojového učení a širokého spektra matematických operací s tensory. Jedná se o počin společnosti Google, který je implementován do téměř každého softwaru této firmy. Samotný vyhledávač od Googlu se učí pomocí operací, které zprostředkovává tento framework. Oproti knihovně PyTorch je však TensorFlow uživatelsky méně přívětivý a jeho implementace je složitější. Z tohoto důvodu jsem při práci zvolil právě sadu funkcí od Facebooku PyTorch. [35]

5. DATASET

Důležitou součástí každého učení neuronové sítě je správně zvolený dataset. Každý dataset obsahuje vstupní data, k nim žádoucí výstupní data a validační data. Při tvoření rozpoznávací neuronové sítě je nezbytné správná volba datasetu. Je třeba zohledňovat kategorie, které dataset umí natrénovat a například obsáhlost informací, z nichž plyne dokonalost natrénování.

COCO (Common Object in Context)
Microsoft COCO dataset je ideální volbou pro určování výkonu nejmodernějších modelů počítačového vidění. Dataset, jak již název napovídá, obsahuje obrovské množství objektů v různých prostředích. Snímky ze sady zachycují objekty v běžných každodenních situacích. [36]

Nejnovější verze datasetu COCO 2020 :

- obsahuje 121,408 snímků
- obsahuje 883,331 anotací k objektům
- obsahuje 80 tříd objektů
- obsahuje 91 tříd věcí
- medián rozlišení snímků je 640 x 480

Snímky mají svoje informace psány ve formě

```
1.     "images": [
2.         {
3.             "license": 3,
4.             "file_name": "000000025560.jpg",
5.             "COCO_url": "http://images.COCOdataset.org/val2017/000000025560.jpg"
6.             "height": 480,
7.             "width": 640,
8.             "date_captured":  "2013-11-17 21:48:19",
9.             "flickr_url": "http://farm1.staticflickr.com/198/488201322_ef2ebfeccb_
z.jpg",
10.            "id": 25560
11.        },
12.        ]
```

Každý snímek v datasetu má licenční informace, datum pořízení, rozlišení a jednu nebo více url adres.

COCO dataset je vyvíjen s 5 různými typy anotací: [36]

- detekce objektů(segmentace)
- detekce klíčových bodů
- segmentace věcí
- panoptická segmentace
- titulky k obrázkům

Anotace ke snímkům, tedy souřadnice ohraničení kategorie atd., jsou uvedeny v souboru typu JSON (JavaScriptObjectNotation)
Detekce objektů(segmentace):

```
1.       "categories": [
2.         {"supercategory": "person","id": 1,"name": "person"},
3.         {"supercategory": "vehicle","id": 2,"name": "bicycle"},
4.         {"supercategory": "vehicle","id": 3,"name": "car"},
5.         {"supercategory": "vehicle","id": 4,"name": "motorcycle"},
6.         ...
7.         {"supercategory": "indoor","id": 89,"name": "hair drier"},
8.         {"supercategory": "indoor","id": 90,"name": "toothbrush"}
9.     ]
```

Kategorie objektů mají vždy svoje id, kategorii a superkategorii.

```
1.     "annotations": [
2.         {
3.            "segmentation": [[451.03,386.85,...,453.91,386.85],
4.             "area": 14853.304350000002,
5.             "iscrowd": 0,
6.             "image_id": 327701,
7.             "bbox": [329.16,266.89,154.5,152.59],
8.             "category_id": 1,
9.             "id": 444736
10.        },..
```

V anotaci pro objektovou segmentaci `"segmentation"` určuje souřadnice masky objektu, `"area"` je počet pixelů uvnitř hranatého ohraničení, `"iscrowd"` říká, zda se jedná o objekt nebo o skupinu objektů (0 pro samostatný objekt), `"bbox"` jsou souřadnice hranatého ohraničení, `"category_id"` je id kategorie objektu a `"id"` je id samotného snímku. [36]

Detekce klíčových bodů

Klíčové body přidávají další informaci k segmentovaným objektům. Specifikuje informačně významné body a spojení mezi nimi. [36]

```
1.     "categories": [
2.     {
3.     "supercategory": "person",
4.     "id": 1,
5.     "name": "person",
6.     "keypoints":["nose","left_eye","right_eye","left_ear","right_ear","left_should
er","right_shoulder","left_elbow","right_elbow",
7.     "left_wrist","right_wrist","left_hip","right_hip",
8.     "left_knee","right_knee","left_ankle","right_ankle"
9.     ],
10.    "skeleton": [
11.    [16,14],[14,12],[17,15],[15,13],[12,13],[6,12],[7,13],[6,7],
12.    [6,8],[7,9],[8,10],[9,11],[2,3],[1,2],[1,3],[2,4],[3,5],[4,6],[5,7]]}]
```

Superkategorie je zde pouze jedna, avšak je dále dělena na klíčové body mající význam různých částí lidského těla. Parametr `"skeleton"` určuje souřadnice spojů bodů.

```
"annotations": [
    {
        "segmentation": [[204.01,306.23,...206.53,307.95]],
        "num_keypoints": 15,
        "area": 5463.6864,
        "iscrowd": 0,
        "keypoints": [229,256,2,...,223,369,2],
        "image_id": 289343,
        "bbox": [204.01,235.08,60.84,177.36],
        "category_id": 1,
        "id": 201376
    }
]
```

Anotace jsou strukturovány podobně, jako u předchozí objektové segmentace. Navíc se zde objevuje parametr `"keypoints",` jenž určuje souřadnice klíčových bodů. Tento parametr je psán ve formě souřadnice x, y + třetí parametr určující viditelnost bodu. [36]

Segmentace věcí

Velice podobná jako objektová segmentace s rozdílem ignorování parametru `"iscrowd".` Nerozlišujeme zde tedy, zda se jedná o jeden objekt, či více instancí objektu. [36]

Panoptická segmentace

Jedná se o úplnou sémantickou segmentaci obrazu, tedy úplné vidění. Obraz je rozdělen do tříd bez slepých míst. [36]

Titulky k obrázkům

Oproti předešlým druhům COCO anotací je tato velice odlišná. Neobjevují se zde žádné x, y souřadnice, není zde žádná segmentace. Tento druh anotací pouze určuje děj na snímku. [36]

```
"annotations": [
    {
        "image_id": 289343,
        "id": 433580,
        "caption": "A person riding a very tall bike in the street.}]
```

5.1 Jiné datasety

MNIST

Jeden z nejpopulárnějších datasetů pro deep learning. Jedná se o sadu ručně psaných čísel. Obsahuje 60,000 tréninkových snímků a 10,000 testovacích snímků. MNIST,

obrázek 5.1, je ideálním datasetem pro zkoušení učících se technik s potřebou minima času a preprocessingu. [37] [38] [39]

Obrázek 5.1 MNIST Převzato z: "/https://medium.com"

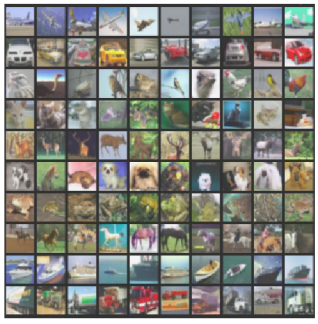
ImageNet

Jedná se o dataset obsahující snímky, které jsou řazeny podle WordNetu (anglická lexikální databáze). WordNet obsahuje kolem 100,000 frází, které jsou ilustrovány každý okolo 1000 snímky z ImageNetu, ilustrován obrázkem 5.2. Anotace obsahují hranatá ohraničení a štítky tříd. [37] [38] [39]

Obrázek 5.2 ImageNet. Převzato z: "/https://medium.com"

CIFAR-10

Dataset pro klasifikaci běžných objektů. Obsahuje 60,000 snímků a 10 kategorii, viz. obrázek 5.3. Snímky jsou strukturovány do 6 částí, z toho 5 trénovacích a 1 testovací. [37] [38] [39]



Obrázek 5.3 CIFAR-10 Převzato z: "/https://medium.com"

6. APLIKACE

Aplikace byla vyvíjena na počítačové sestavě s procesorem Ryzen 5 1600 a grafickou kartou s podporou CUDA od Nvidie GeForce GTX 1080. Karta disponuje 2560 CUDA jádry, což ji i řadí mezi nejvýkonnější dostupné GPU své generace. Sestava je doplněna o 8 Gb DDR4 operační paměti o frekvenci 2400 MHz. Operační systém jsem využil linux Ubuntu 20.04 a Python verze 3.8.

6.1 Zobrazovací kód

Hlavní main funkce načte parametry z příkazové řádky, viz tabulka 6.1.

Parametr	Popis	Default
input	Adresář pro vstupní soubory	. (aktuální adresář)
output	Adresář pro výstupní soubory	. (aktuální adresář)
show	Zobrazovat výstupní obrázky interaktivně	True

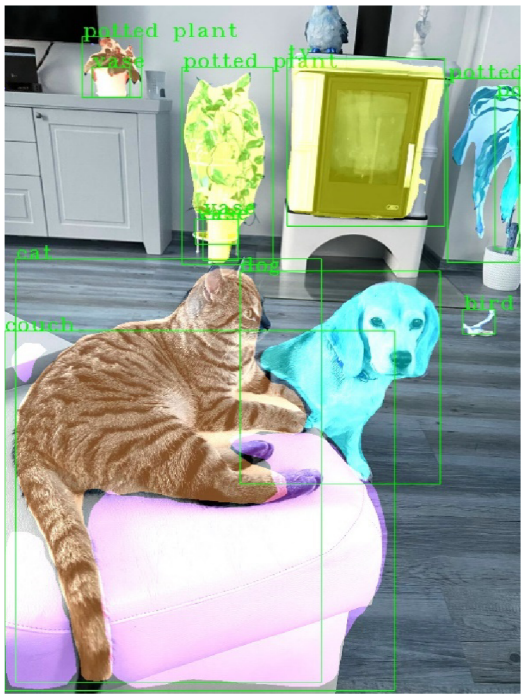
Tabulka 6.1 Vstupní parametry zobrazovacího programu

Vytvoří se instance dvou základních tásků TaskMask a TaskKp. Následně se prochází adresář vstupních souborů, kde se vybírají pouze obrázky jpg, png, jpeg a dále se odfiltrují již segmentované výstupní soubory obsahující postfix _mask nebo _kp. Každý tásk obrázek načte, zpracuje a uloží do výstupního adresáře případně ihned po zpracování zobrazí uživateli. Pokud TaskMask nalezne alespoň jednu osobu, spouští se také vyhledání keypoints a na jejich základě se dále vyhodnocují i bodyparts.

```
1.      def main():
2.          #parses comandline to get input parameters
3.          args = parseCmdline()
4.          #creates instance of task TaskMask
5.          taskMask = TaskMask()
6.          #creates instance of task TaskKp
7.          taskKp = TaskKp()
8.
9.          #files from directory args.input are filtered and segmented
10.         for fn in filter(fileFilter, os.listdir(args.input)) :
11.             #joins directory and filename for input file
12.             inputFile = os.path.join(args.input,fn)
13.             #joins directory and filename for outputfile file`
14.             outputFile = os.path.join(args.output, fn.replace(".", "_mask."))
15.             #does mask segmentation od input image and retuns clases
16.             classes = taskMask.segmentation(inputFile, outputFile, args.show)
17.             #if retuns atleast one person does keypoint prediction
18.             if("person" in classes):
19.                 outputFile = os.path.join(args.output, fn.replace(".", "_kp."))
20.                 taskKp.segmentation(inputFile, outputFile, args.show)
```

TaskMask

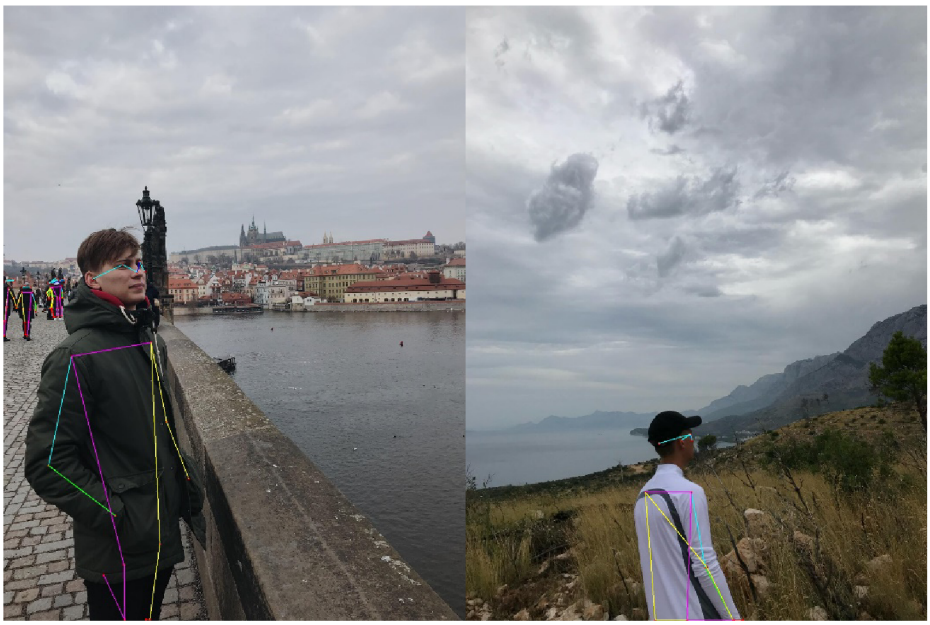
Vstupní obrázek se převede na numpy array poté na tenzor, provede se klasifikace prostřednictvím modelu maskrcnn_resnet50_fpn a výpočet masek, zpracuje se pravděpodobnostní rozdělení klasifikovaných objektů, vyberou se objekty s pravděpodobností min 0.5. Dále se prochází jednotlivé masky a zajistí se jejich grafické zvýraznění modifikací barvy bodu. Výstupní obrázek obsahující masky nalezených objektů a nejmenší obepínající obdelníky se podle nastavení zobrazí uživateli a vždy se uloží na disk. Výstupní obrázek je vidět na obrázku 6.1.



Obrázek 6.1 Mask R-CNN

TaskKp

Vstupní obrázek se převede na numpy array poté na tenzor, takto vytvořený vstupní vektor vstupuje do modelu keypointrcnn_resnet50_fpn, který vrací nalezené objekty typu (score, keypoints). Score má význam pravděpodobnosti, že se jedná o osobu a vybírají se pouze objekty s pravděpodobností > 0.9. Pro takto vybrané objekty se do původního obrázku zakreslí keypointy. Propojení keypointů je staticky definované, prediktor vrací pouze souřadnice bodů v definovaném pořadí. Výsledný obrázek, se volitelně zobrazí při zpracování a současně se vždy uloží do nastaveného výstupního adresáře. Vstup lze vidět na obrázku 6.2.



Obrázek 6.2 Keypoint R-CNN

BodyParts

Modifikací výstupu neuronové sítě v podobě keypointů bylo dosaženo schopnosti rozpoznání jednotlivých částí končetin lidského těla. Tato modifikace je založena na výpočtu konstanty za pomoci vzdálenosti určitých keypointů a rozšíření spojnice mezi nimi, popřípadě vytvoření určitého geometrického tvaru na správné pozice. Moji modifikaci rozpoznávání lidských končetin lze vidět na obrázku 6.3.

Předtrénovaná neuronová síť ResNet-50 na základě vstupního obrázku vrací 3 sady souřadnic keypointů, které by měly být umístěny v klíčových bodech definujících lidské tělo. Každá sada je zároveň ohodnocena tzv. keypoint score, což nám říká, jak přesně jsou keypointy umístěny vzhledem k "vědomostem", které neuronový model při tréninku získal. Algoritmus dále využívá tu nejlepší sadu, která musí zároveň splňovat keypoint score větší, než 0.9.

Máme-li tedy vhodnou sadu souřadnic 17 keypointů, můžeme pokročit k metodám, které vhodně rozšíří spojnice těchto bodů na vhodném místě a označí tak jednotlivé lidské končetiny.

Šířka spojnice keypointů rukou je definována vektorovou délkou začátek - konec předloktí dále vynásobenou konstantou viz. optimizer, která je vyhodnocena testováním na více obrázcích

Šířka spojnice nohou je řešena stejným algoritmem, pouze měříme vektorovou vzdálenost bodů začátek - konec stehna a násobíme ho konstantou viz. optimizer.

Označení oblasti hlavy vychází ze spojnice uší. Nalezneme střed spojnice uší, ve středu vztyčíme kolmici a na kolmici vyznačíme dva body symetricky kolem středu, jejichž vektorovou vzdálenost určíme pomocí vzdálenosti uší a konstanty viz. optimizer. Mezi těmito body vykreslíme spojnici o tloušťce vektorové vzdálenosti uší.

Trup je rozeznáván pomocí 4 keypointů, 2 v ramenou a 2 na začátku nohou. Naleznou se středy těchto dvojic a vektorová vzdálenost se symetricky zmenší na

hodnotu viz optimizer. Poté se vykreslí spojnice mezi těmito středy s tloušťkou, která se rovná vektorové vzdálenosti bodů v ramenou.

Obrázek 6.3 BodyParts

Optimizer

Algoritmus zvýraznění body parts je založen na rozšíření spojnic definovaných keypointů, které jsou výstupem z natrénované neuronové sítě resnet_50_keypoints. Keypointy dávají lokaci a rozšíření poskytuje výplň definovaných oblastí jednotlivých částí těla.

Rozšíření je navrženo takovým způsobem, že šířka výplně je odvozena z délky příslušných spojnic bodů. Jedná se o lineární funkční závislost, kdy vzniká potřeba vhodným způsobem určit multiplikativní konstanty v definici výplně.

Pro určení se využívá optimalizační strategie, kdy definujeme hodnotící funkci jako rozdíl počtu bodů které vyhovují ground true a počty bodů, které jsou mimo oblast ground true. Algoritmus je vyhodnocován přes definovaný interval multiplikativnich konstant. Výsledkem jsou jednotlivé závislosti hodnotícího kritéria na hodnotě konstanty v daném intervalu pro každou body part.

Jednotlivé funkce vykazují hladký průběh s výrazným maximum, kde konstantu určíme právě z tohoto extrému. Výsledné křivky pro určení konstant jsou uvedeny v příloze B.

Kód je zveřejněn na githubu viz. [40] Při programování zobrazovací aplikace jsem vycházel z [41] [42] [43] [33] [44] [45]

6.2 Trénovací kód

V předchozích případech jsem vycházel z předtrénovaného modelu resnet50 pro Mask R-CNN a pro Keypoint R-CNN na COCO datasetu.Trénovací procedura si načte dataset, kde dataset se předpokládá COCO nebo COCO_KP nebo jakýkoliv jiný, který splňuje dané rozhraní. Načtou se datasety zvlášť pro trénování a pro testování. Dále se pro datasety vytvoří datové loadery a zkonstruuje se vlastní model pro daný počet klasifikačních tříd. Model může být předtrénovaný. V přípravné fázi se ještě vytváří plánovač pro learning rate, který se uplatňuje v procesu optimalizace učení. Následně se spustí standardizovaný blok učení, kdy se pro jednotlivé učící epochy volá cyklus dopředného výpočtu klasifikace a zpětné propagace a úpravy vah neuronové sítě. V jedné epoše se jedenkrát projde trénovací dataset a provede se ověření na testovací sadě. Výstupy jsou pravidelně vypisovány na standardní výstup. Po průchodu všemi epochami se výsledný naučený model uloží do souboru na disk. Tento kód je standardizovaný a byl kompletně převzat z frameworku PyTorch. [46]

Zobrazovací, optimalizační, porovnávací i trénovací kód je přiložen v : Příloha A-Aplikace.

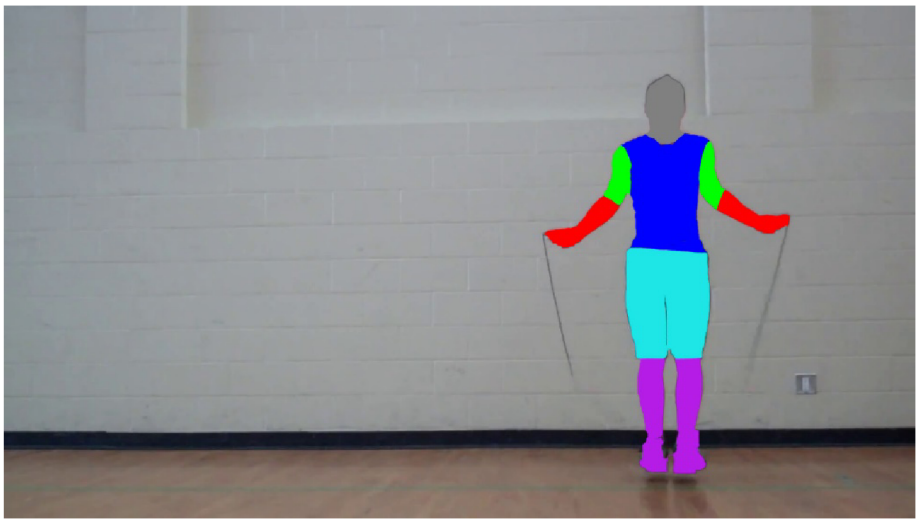
7. VYHODNOCOVÁNÍ

K vyhodnocování v oblasti úspěšnosti rozeznávání lidských končetin bylo přistoupeno nejběžnější metodou, tedy vyhodnocením IoU (Intersection over Union). Tato metoda reflektuje přesnost pokrytí algoritmu části v procentech.
Rychlost zpracování byla vyhodnocena v závislosti k velikosti obrázku, velikosti objektu na obrázku a relativní velikosti objektu na obrázku.
Logicky lze očekávat zpomalování algoritmu se zvětšujícím se obrázkem, případně objektem na obrázku. Velikost obrázku má však vliv pouze na fázi předzpracovávání a testem zjistíme významnost na celkovou rychlost vyhodnocení.

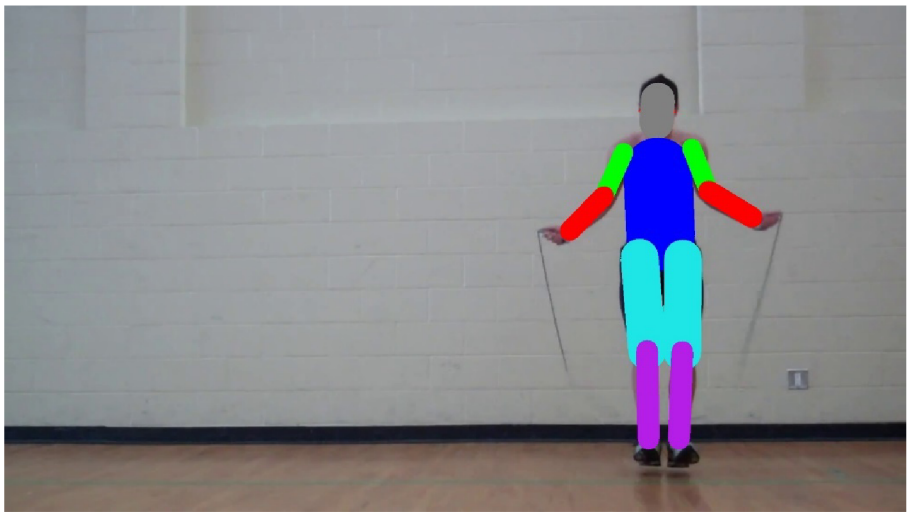
7.1 Intersection Over Union (IoU)

Vyhodnocení úspěšnosti mé nadstavby na keypoint R-CNN pro rozlišování lidských končetin, bylo řešeno metodou porovnávání pixelů na groud truth obrázku, tedy dokonalé segmentovaného lidského těla, s pixely na obrázku vyhodnoceném právě řešeným algoritmem.

Testovací sada byla vybrána z volně dostupného datasetu MPII. Jedná se o dataset zaměřený na lidské postavy v různých kontextech s různými světelnými podmínkami. Groud truth obrázek 7.1 byl získán za pomocí programu Adobe Photoshop. Stejný obrázek vyhodnocený algoritmem můžeme vidět na obrázku 7.2.



Obrázek 7.1 Ground truth

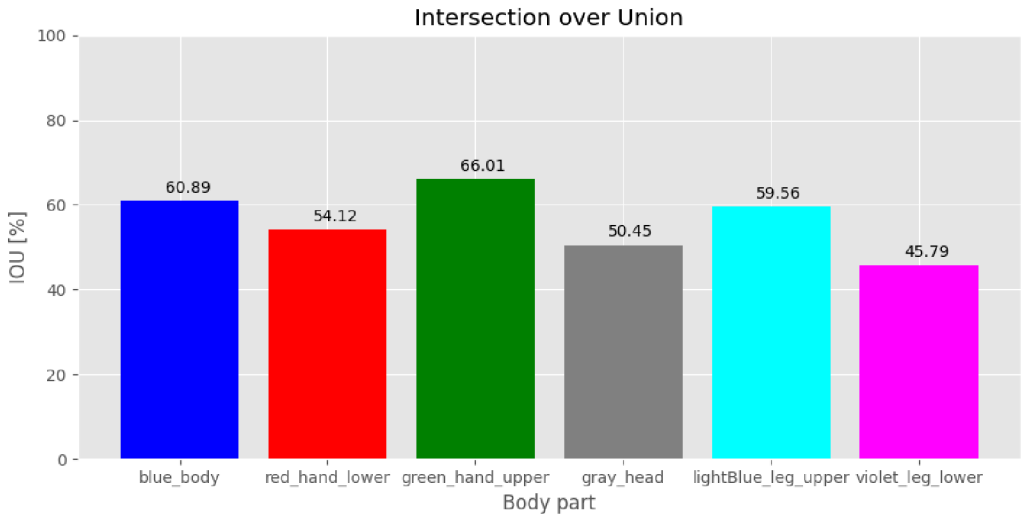


Obrázek 7.2 Výstup algoritmu

Vyhodnocovací script byl napsán v jazyce python. Pomocí knihovny numpy zpracovává vstupní obrázky ve formě rgb tezorů a počítá počet například červené (255,0,0) pixely, kterým přiřadí význam dolní část ruky. Tento postup je aplikován na obrázek s groud truth i na obrázek z rozpoznávacího algoritmu.

„Score“ na obrázku vyhodnoceném algoritmem je však podmíněno tak, že je přičten bod pouze v momentě, kdy je stejný pixel vyhodnocen jako např. dolní část ruky, na obou obrázcích. Nebo-li je červený (255,0,0) i na referenčním obrázku.

Poměrem těchto získaných počtů barevných pixelů získáváme hodnotu IoU (Intersection over Union). Hodnoty IoU jsou vyhodnocovány zvlášť pro každou končetinu a dále průměrovány s postupem zpracování celého testovacího subsetu. Vyhodnocení lze vidět na obrázku 7.3 a tabulce 7.1.



Obrázek 7.3 IoU

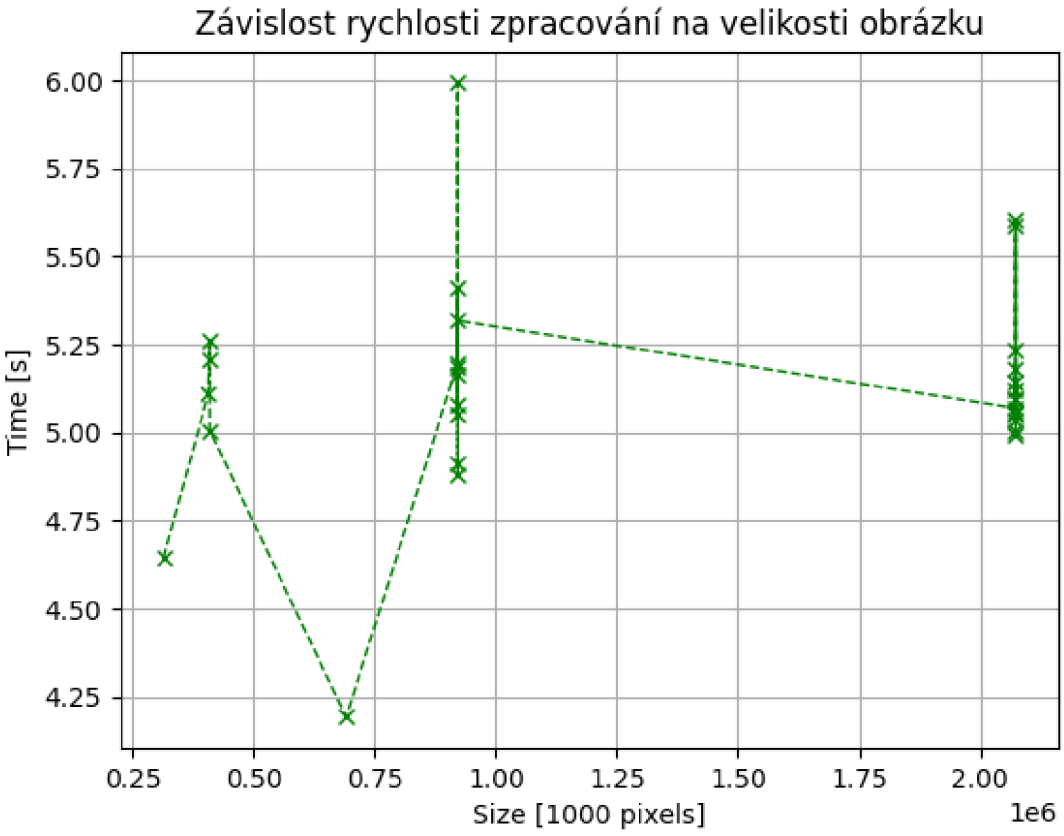
Body part	IoU[%]
blue_body	60,89
red_hand_lower	54,12
green_hand_upper	66,01
gray_head	50,45
lightBlue_leg_upper	59,56
violet_leg_lower	45,79

Tabulka 7.1 IoU

Rozšíření metody keypoints R-CNN na oblasti lidských končetin je parametrizované konstantami, které definují šířku oblastí v závislosti na vzdálenosti relevantních keypointů. Hodnoty těchto konstant byly zjištěny pomocí optimalizačního algoritmu implementovaného ve skriptu optimizer.py. Po maximalizaci efektivity metody bylo docíleno následujících výsledků, které jsou uvedeny v tabulce 7.1. Kromě dolní oblasti nohou se všechny hodnoty IoU pohybují nad 50%. Ve srovnání s generováním masek lidských těl pomocí Mask R-CNN, kde se pohybujeme v IoU nad 90%, je úspěšnost tohoto algoritmu nízká, avšak výhodou je právě rozpoznávání jednotlivých končetin. Konkrétní nedokonalost algoritmu je znatelná v oblasti hlavy. Tento problém je popsán níže.

7.2 Rychlost

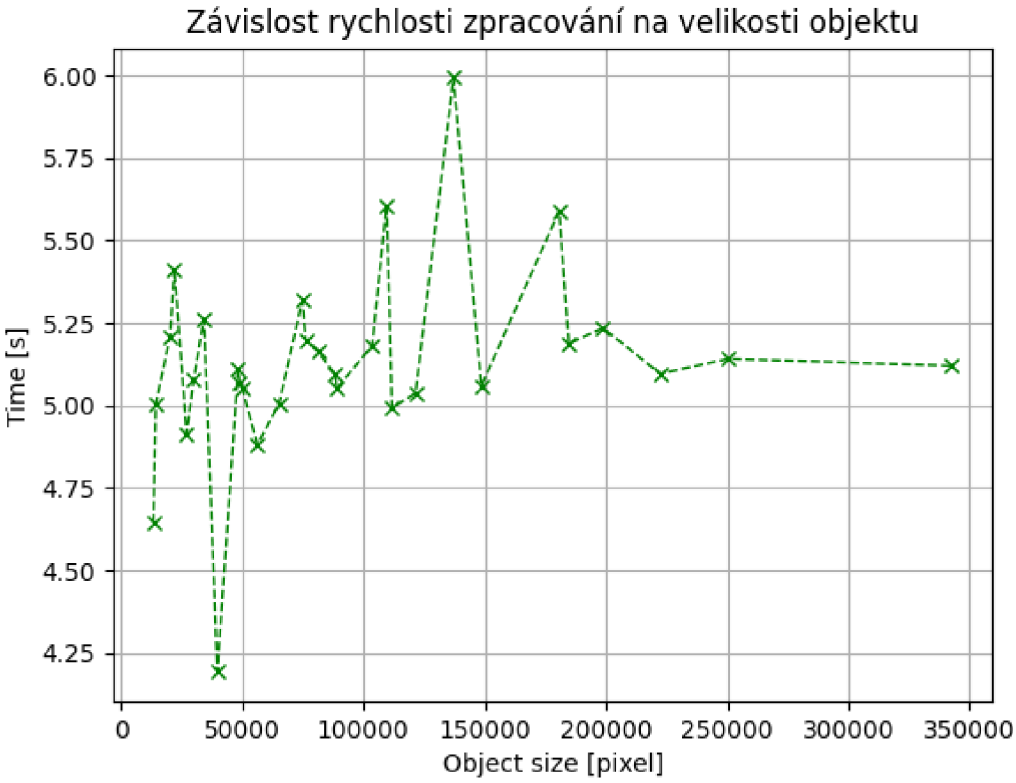
Rychlost vyhodnocování algoritmu jsem testoval za pomocí přídavných funkcí v samotném kódu algoritmu na mé testovací sadě. Aspektem, který jsem testoval, byla závislost rychlosti zpracování obrázku na velikosti. Na obrázku 7.4 je jasné, že tato metodika vyhodnocování nenastiňuje žádný trend, jelikož dochází k různým výsledkům na stejně velkých obrázcích.



Obrázek 7.4 Závislost rychlosti zpracování na velikosti obrázku

Druhým přístupem k vyhodnocení rychlosti je vztahovat ji k velikosti jednotlivých postav na obrázku, ne velikosti celého obrázku. Na obrázku 7.5 můžeme vidět závislost rychlosti na absolutní velikosti vyhodnocované postavy, na obrázku 7.6 pak závislost rychlosti na relativní velikosti vyhodnocované postavy vůči celému obrázku.

Z grafů je jasně pozorovatelné, že velikost samotného objektu není stěžejním faktorem pro rychlost zpracování. Objekty jsou totiž před vstupem do plně propojené vrstvy normovány. Mírný vzestupný trend času zpracování je způsoben náročností předzpracování do právě této normované podoby. Ve všech případech se však blížíme k průměrné hodnotě s malou odchylkou, čímž je prokázána irelevantnost rozměrů vstupního obrázku.



Obrázek 7.5 Závislost rychlosti zpracování na velikosti objektu

Obrázek 7.6 Závislost rychlosti zpracování na relativní velikosti objektu

7.3 Problémy

Algoritmus je svým základem velmi houževnatý, COCO dataset pro keypointy je velice rozmanitý a proto se při vyhodnocování nevyskytují problémy v souvislosti s osvětlením, ani vlivem barev lidských postav. Nepřesnosti nastávají v některých komplikovanějších pózách lidských těl, kdy nejsou zcela správně rozmístěné keypointy a rozpoznaná informace o končetinách je tak zcela znehodnocená. Vstupní obrázek a nepřesné vyhodnocení lze vidět na obrázcích 7.7 a 7.8.

Obrázek současně zachycuje druhý problém, který u algoritmu nastává a to sice nepřesné zachycení celé oblasti hlavy. Tato nepřesnost je způsobena odečtem vzdálenosti keypoint levé ucho – pravé ucho. V případě, že je postava z profilu, je tato vzdálenost téměř nulová, a proto je zvýrazněná oblast hlavy výrazně menší, než na ground truth.

Obrázek 7.7 Vstupní obrázek

Obrázek 7.8 Chybné vyhodnocení

8. ZÁVĚR

V první části bakalářské práce jsem popsal pojmy týkající se strojového učení a zpracování obrazu. Rozepsal jsem základní terminologii ve strojovém učení a souvisejících datech. Tento popis byl jednoduchý a obecný pro pouhé nastínění problematiky. Klíčovou oblastí strojového učení je až Deep learning. Společně s ním jsem uvedl základní principy neuronových sítí a jejich nejčastější typy. Touto oblastí jsem procházel poprvé, učil jsem se o sítích až postupem vypracovávání práce, proto jsem se snažil vše psát srozumitelně a jednoduše.

Dále jsem přiblížil, pro zpracování obrazu, stěžejní typ neuronové sítě – konvoluční neuronovou síť. V tomto přiblížení rozebírám samostatně kroky, kterými síť snímek rozebírá a získává z něj informace – mapu vlastností. Pro demonstraci konvolučních filtrů, což je zásadní nástroj konvolučních neuronových sítí, jsem je aplikoval pomocí prostředí Matlab na mnou pořízený obrázek. Snažil jsem se také ilustrovat proces odstranění záporných hodnot a využívané typy škálování. Následuje uvedení nejčastějších architektur konvolučních neuronových sítí, kde je dále pro moji aplikaci nejvýznamnější architektura ResNet.

V další části se věnuji segmentačním nástavbám konvolučních neuronových sítí, typům R-CNN. Postupuji zde od základního R-CNN, přes Fast R-CNN, až po Faster R-CNN. V práci se nachází i rozšíření Faster R-CNN pro generování masek Mask R-CNN a rozšíření pro generování klíčových bodů objektů. Každý z typů této nástavby je popsán a vysvětlen.

Pro každou neuronovou sít pro zpracování obrazu je důležitý trénovací dataset, jehož typy rozebírám v další kapitole. Nejvíce se zde zaměřuji na COCO dataset, který využívám při implementaci do vlastní aplikace. COCO dataset je popsán na úroveň struktury anotací.

Tvorba aplikace a popis kódu je popsán v poslední kapitole. Aplikace je schopna využívat metody Faster R-CNN s rozšířeními Mask-RCNN i Keypoint-RCNN.

V druhé části této práce jsem se zabýval rozšířením aplikace ve smyslu segmentace lidských těl na jednotlivé končetiny. Realizace této funkce vychází z informace získáme metodou Keypoint-RCNN. Keypointy získané touto metodou jsou vždy zasazeny na stejná logická místa lidského těla. Spojnice mezi nimi moje nástavba rozšiřuje na chtěnou velikost tak, aby segmentovali lidské tělo právě na jednotlivé končetiny. Vhodná rozšiřovací konstanta byla zjištěna pomocí optimalizačního scriptu, který na testovací sadě ověřoval hodnotu překrytí a zároveň přesahu pro různé hodnoty těchto konstant.

Ověření efektivity algoritmu bylo realizováno zjištěním hodnoty IoU (Intersection over Union) tedy procentuální překrytí ground truth. Kromě spodní části nohy algoritmus dosahuje hodnot IoU přes 50%.

Dále byla vyhodnocována rychlost algoritmu v závislosti na rozlišení obrázku, velikosti vyhodnocovaného obrázku, či relativní velikosti objektu, vůči obrázku. Závěrem této statistiky bylo ověření, že velikost obrázku nebo zpracovávaného objektu

není stěžejním faktorem pro rychlost zpracovávání. Objekty jsou totiž normovány před vstupem do plně propojené vrstvy konvoluční sítě.

Metoda, jíž se práce zabývá, má určité problémy. Jedná se o znehodnocení výstupu v případě neobvyklých poloh lidských těl. Získá-li algoritmus nesprávnou vzdálenost dvou keypointů, z kterých dále vychází, jsou potom rozlišovací plochy pro končetiny nesprávně široké. Druhý problém nastává u vyhodnocování části obrázku s hlavou postavy. Je-li postava vůči fotografovi z profilu, vzdálenost levé ucho - pravé ucho, z které další výpočet vychází, je nepatrná a oblast pro hlavu malá.

Možným směrem, kterým se vydat pro zefektivnění algoritmu by byla jeho implementace do embedded systému, což mi výrazně zrychlilo proces, až k potenciálním real-time rychlostem.

9. LITERATURA

[1] „Learning, Getting started with Machine,“ [Online]. Available: https://www.geeksforgeeks.org/getting-started-machine-learning/?ref=lbp. [Přístup získán 1 11 2020].

[2] „ML | Introduction to Data in Machine Learning,“ [Online]. Available: https://www.geeksforgeeks.org/ml-introduction-data-machine-learning/?ref=lbp. [Přístup získán 1 11 2020].

[3] B. Grossfeld, „Deep learning vs machine learning: a simple way to understand the difference,“ [Online]. Available: https://www.zendesk.com/blog/machine-learning-and-deep-learning/. [Přístup získán 4 11 2020].

[4] S. Mhatre, „What Is The Relation Between Artificial And Biological Neuron?,“ [Online]. Available: https://towardsdatascience.com/what-is-the-relation-between-artificial-and-biological-neuron-18b05831036. [Přístup získán 6 11 2020].

[5] R. Keim, „How to Train a Basic Perceptron Neural Network,“ [Online]. Available: https://www.allaboutcircuits.com/technical-articles/how-to-train-a-basic-perceptron-neural-network/. [Přístup získán 10 11 2020].

[6] M. Afsharizadeh, „https://www.researchgate.net/,“ [Online]. Available: https://www.researchgate.net/figure/An-artificial-neural-network-neuron-Each-neuron-has-an-activation-function-the-input-for_fig1_328403993. [Přístup získán 10 11 2020].

[7] Team Towards AI, „Main Types of Neural Networks and its Applications — Tutorial,“ [Online]. Available: https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e. [Přístup získán 12 11 2020].

[8] P. Mishra, „What is Hebbian Learning?,“ [Online]. Available: https://medium.com/datadriveninvestor/what-is-hebbian-learning-3a027e8e4bbb. [Přístup získán 15 11 2020].

[9] „Feedforward neural networks 1. What is a feedforward neural network?,“ [Online]. Available: https://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1__What_is_a_feedforward_ne.html. [Přístup získán 17 11 2020].

[10] E. Beqari, „A Very Basic Introduction to Feed-Forward Neural Networks,“ [Online]. Available: https://dzone.com/articles/the-very-basic-introduction-to-feed-forward-neural. [Přístup získán 15 11 2020].

[11] Wikipedia, „Backpropagation,“ [Online]. Available: https://en.wikipedia.org/wiki/Backpropagation. [Přístup získán 15 11 2020].

[12] S. Saha, „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,“ [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Přístup získán 18 11 2020].

[13] S. Bansari, „Introduction to how CNNs Work,“ [Online]. Available:

https://medium.com/datadriveninvestor/introduction-to-how-cnns-work-77e0e4cde99b. [Přístup získán 20 11 2020].

[14] Wikipedia, „Convolutional neural network,“ [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network. [Přístup získán 20 11 2020].

[15] J. Brownlee, „A Gentle Introduction to Dropout for Regularizing Deep Neural Networks,“ [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Přístup získán 10 11 2020].

[16] R. Karim, „Illustrated: 10 CNN Architectures,“ [Online]. Available: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d. [Přístup získán 21 11 2020].

[17] Great Learning Team, „AlexNet: The First CNN to win Image Net,“ [Online]. Available: https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/. [Přístup získán 22 11 2020].

[18] Google Inc, „Going Deeper with Convolutions,“ [Online]. Available: https://static.googleusercontent.com/media/research.google.com/cs//pubs/archive/43022.pdf. [Přístup získán 23 11 2020].

[19] C.-F. Wang, „The Vanishing Gradient Problem,“ [Online]. Available: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484.

[20] „Residual Networks (ResNet) – Deep Learning,“ [Online]. Available: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/. [Přístup získán 24 11 2020].

[21] R. Gandhi, „R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms,“ [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Přístup získán 25 11 2020].

[22] R. Girshick, J. Donahue, T. Darrell a J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation,“ [Online]. Available: https://arxiv.org/pdf/1311.2524.pdf. [Přístup získán 26 11 2020].

[23] R. Gandhi, „Support Vector Machine — Introduction to Machine Learning Algorithms,“ [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Přístup získán 25 11 2020].

[24] R. Girshick, „Fast R-CNN,“ [Online]. Available: https://arxiv.org/pdf/1504.08083.pdf. [Přístup získán 27 11 2020].

[25] D. Radečić, „Softmax Activation Function Explained,“ [Online]. Available: https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60. [Přístup získán 25 11 2020].

[26] R. Shaoqing, H. Kaiming, Sun a J. Sun, „Faster R-CNN: Towards Real-Time Object,“ [Online]. Available: https://arxiv.org/pdf/1506.01497.pdf. [Přístup získán 20 11 2020].

[27] T. Karmarkar, „Region Proposal Network (RPN) — Backbone of Faster R-CNN,“ [Online]. Available: https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9. [Přístup získán 20

11 2020].

[28] A. Rosebrock, „Intersection over Union (IoU) for object detection,“ [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. [Přístup získán 20 11 2020].

[29] S.-H. Tsang, „Review: Faster R-CNN (Object Detection),“ [Online]. Available: https://towardsdatascience.com/review-faster-r-cnn-object-detection-f5685cb30202. [Přístup získán 1 12 2020].

[30] M. Kersner, „Non Maximum Suppression,“ [Online]. Available: https://paperswithcode.com/method/non-maximum-suppression. [Přístup získán 10 11 2020].

[31] K. He, G. Gkioxari, P. Dollár a R. Girshick, „Mask R-CNN,“ [Online]. Available: https://arxiv.org/pdf/1703.06870.pdf. [Přístup získán 25 11 2020].

[32] X. Zhang, „Simple Understanding of Mask RCNN,“ [Online]. Available: https://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95. [Přístup získán 25 11 2020].

[33] S. R. Rath, „Human Pose Detection using PyTorch Keypoint RCNN,“ [Online]. Available: https://debuggercafe.com/human-pose-detection-using-pytorch-keypoint-rcnn/. [Přístup získán 1 12 2020].

[34] Wikipedia, „PyTorch,“ [Online]. Available: https://en.wikipedia.org/wiki/PyTorch. [Přístup získán 28 11 2020].

[35] „What is TensorFlow? Introduction, Architecture & Example,“ [Online]. Available: https://www.guru99.com/what-is-tensorflow.html. [Přístup získán 26 11 2020].

[36] „Create COCO Annotations From Scratch,“ [Online]. Available: https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch. [Přístup získán 25 11 2020].

[37] P. Dar, „25 Open Datasets for Deep Learning Every Data Scientist Must Work With,“ [Online]. Available: https://medium.com/analytics-vidhya/25-open-datasets-for-deep-learning-every-data-scientist-must-work-with-1232371569a1. [Přístup získán 25 11 2020].

[38] Prabhu, „CNN Architectures — LeNet, AlexNet, VGG, GoogLeNet and ResNet,“ [Online]. Available: https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848. [Přístup získán 20 11 2020].

[39] A. H. Kumar, „Object Detection Datasets,“ [Online]. Available: https://medium.com/towards-artificial-intelligence/9-object-detection-datasets-773a11ebaa2c. [Přístup získán 1 12 2020].

[40] L. Matějek, „Github-Libor Matějek,“ [Online]. Available: https://github.com/libormatejek/Human-pose-recognition.

[41] C. Fotache, „How to Train an Image Classifier in PyTorch and use it to Perform Basic Inference on Single Images,“ [Online]. Available: https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5. [Přístup získán 1 12 2020].

[42] Pytorch Team, „RESNET,“ [Online]. Available:

https://pytorch.org/hub/pytorch_vision_resnet/. [Přístup získán 1 12 2020].

[43] S. Mallick, „Mask R-CNN Instance Segmentation with PyTorch,“ [Online]. Available: https://www.learnopencv.com/mask-r-cnn-instance-segmentation-with-pytorch/. [Přístup získán 1 12 2020].

[44] „AN OVERVIEW OF HUMAN POSE ESTIMATION WITH DEEP LEARNING,“ [Online]. Available: https://beyondminds.ai/an-overview-of-human-pose-estimation-with-deep-learning/. [Přístup získán 28 11 2020].

[45] P. SHARMA, „Computer Vision Tutorial: Implementing Mask R-CNN for Image Segmentation (with Python Code),“ [Online]. Available: https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/. [Přístup získán 28 11 2020].

[46] Pytorch, „pytorch/vision,“ [Online]. Available: https://github.com/pytorch/vision. [Přístup získán 20 11 2020].

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

CPU	Central processing unit / Centrální procesová jednotka
GPU	Graphics processing unit / Grafická procesová jednotka
CUDA	Compute Unified Device Architecture / Jednotná výpočetní architektura
GPU	Graphics processing unit / Grafická procesová jednotka
FF	Feed Forward / Dopředná neuronová síť
DFF	Deep Feed Forward / Hloubková dopředná neuronová síť
CNN	Convolutional Neural Network / Konvoluční neuronová síť
RGB	Red-Green-Blue / Červená-Zelená-Modrá
ReLU	Rectified Linear Unit for non-linear operation / Usměrňovací jednotka pro nelineární operace
R-CNN	Region based Convolutional Neural Networks / Konvoluční neuronová síť založena na regionech
SVM	Support Vector Machine / Podpůrná vektorová jednotka
RoI	Region of Interest / Informativně zajimavá oblast
RPN	Region Proposal Network/ Síťový návrh oblastí
IoU	Intersection over Union / Sjednocení oblastí
COCO	Common Object in Conctext / Běžné objekty v kontextu
FAIR	Facebook's AI Research / Facebook výzkum umělé inteligence

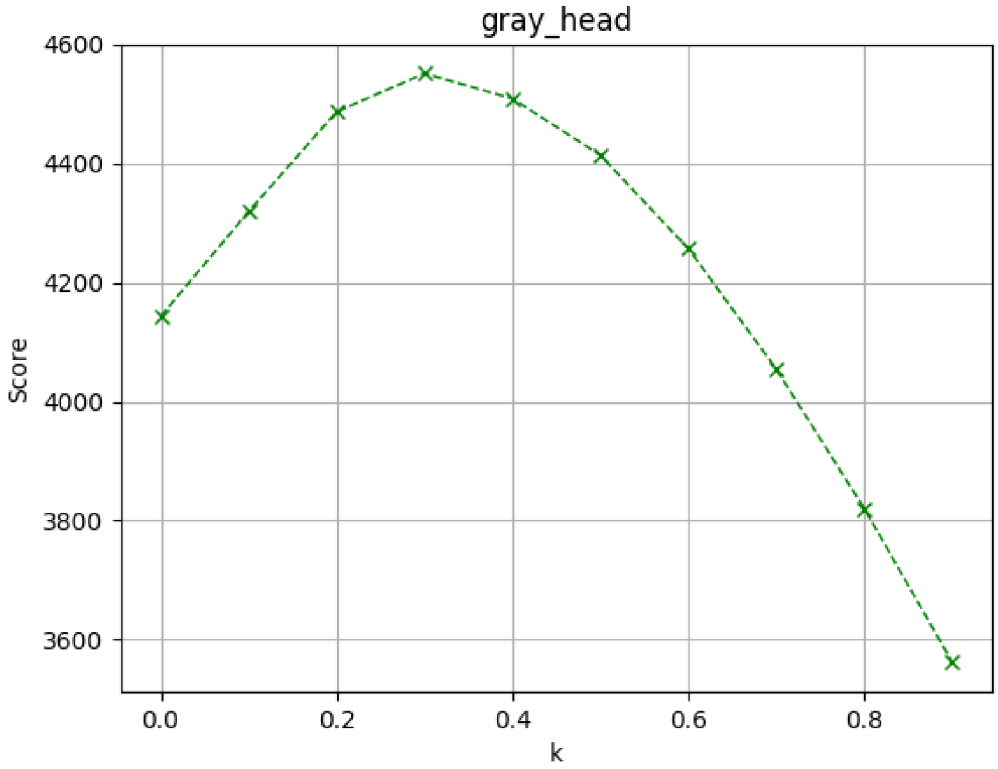
Symboly:

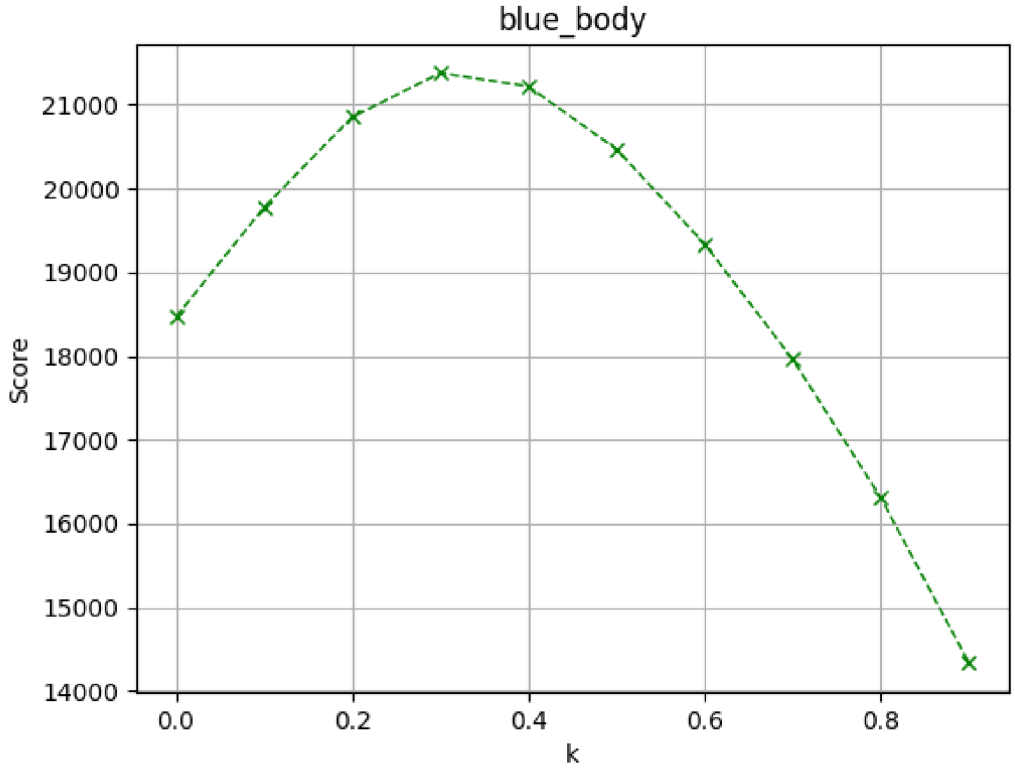
w_i	neuronová váha	[-]
bias	bias	[-]
η	učící koeficient	[-]
t	správná hodnota	[-]
o	predikovaná hodnota	[-]
z_i	hodnota prvku vektoru vah	[-]
K_w	počet prvků vektoru vah	[-]
W	šířka obrázku	[px]
H	výška obrázku	[px]
K_p	počet kombinací poměru stran	[-]
p_i	pravděpodobnost oblasti v popředí	[-]
p_i^*	hodnota štítku podle IoU	[-]
N_{cls}	počet prvků ve várce	[-]
t_i	4-rozměrný parametr predikovaného posunutí	[-]

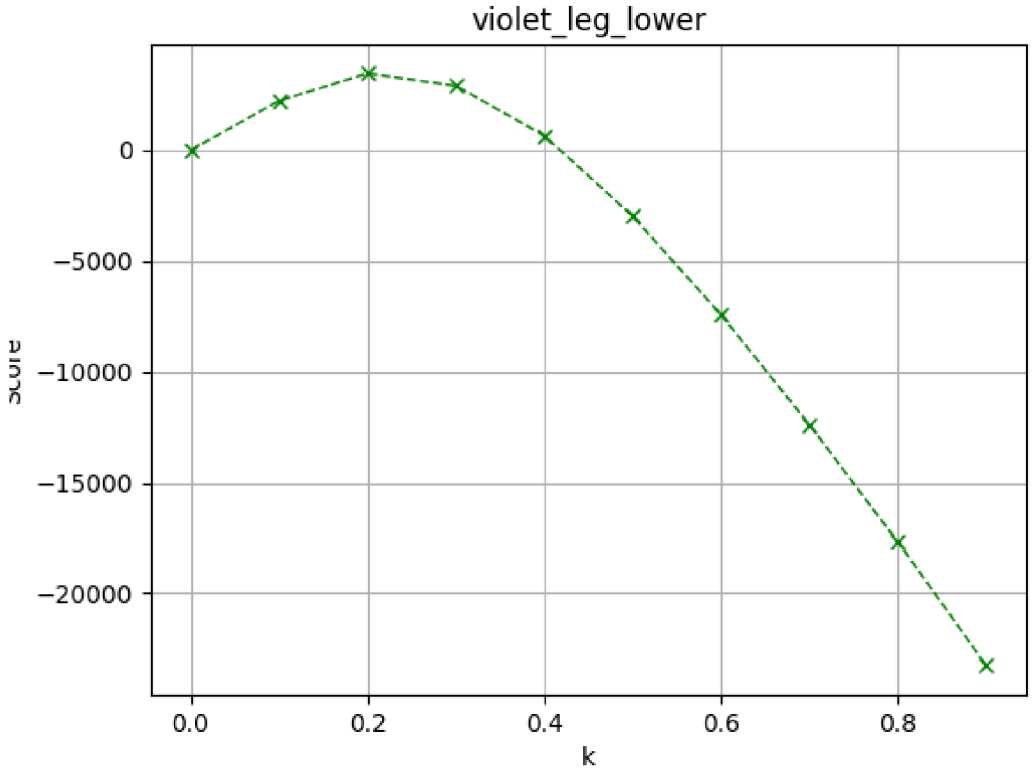
t_i^*	4-rozměrný pravdivý parametr ohraničení	[-]
N_{reg}	počet všech oblastí kotev	[-]
λ	vyrovnávací činitel	[-]

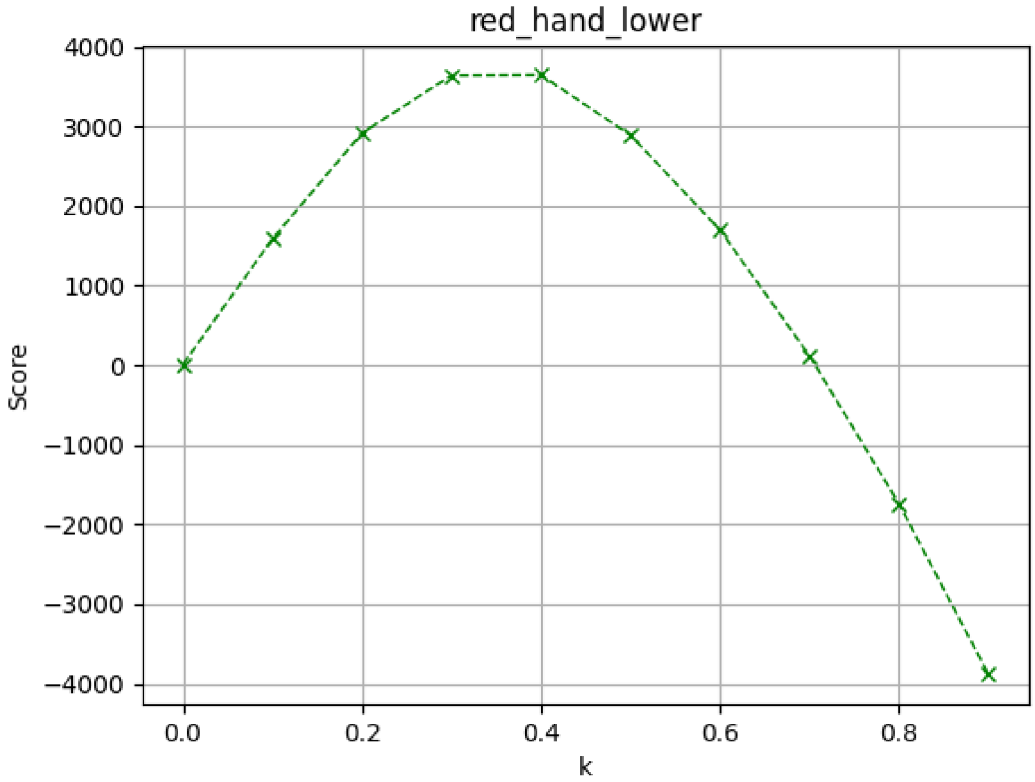
SEZNAM PŘÍLOH

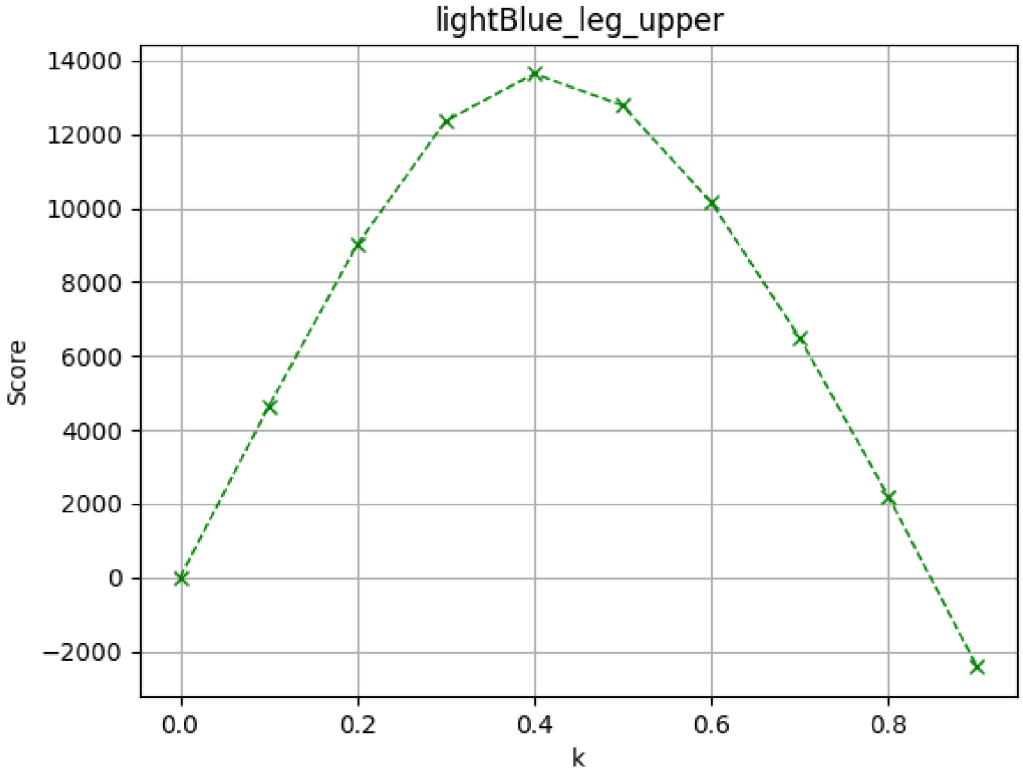
Příloha B – Výstupní hodnoty – optimizer

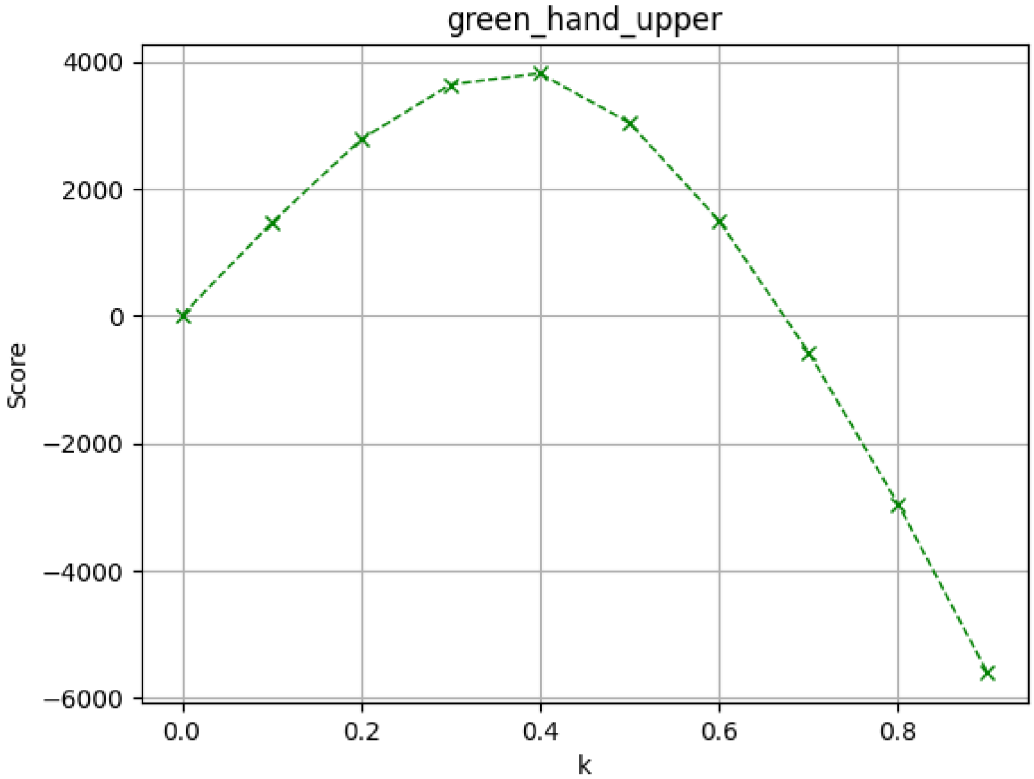





violet_leg_lower
0
−5000
−10000
−15000
−20000
0.0
0.2
0.4
0.6
0.8
k


red_hand_lower
4000
3000
2000
1000
0
−1000
−2000
−3000
−4000
Score
0.0
0.2
0.4
0.6
0.8
k


lightBlue_leg_upper
14000
12000
10000
8000
6000
4000
2000
0
−2000
Score
0.0
0.2
0.4
0.6
0.8
k


green_hand_upper
4000
2000
0
−2000
−4000
−6000
Score
0.0
0.2
0.4
0.6
0.8
k