Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

PLZEŇ, 2019/2020 **Adam Klečka**

Anotace

Bakalářská práce se zabývá strojovou detekcí lidí. Jsou zde podrobně probírány 2 způsoby detekčních systému. Systém s posuvným okénkem, který počítá *HOG* příznaky a za pomoci klasifikačního algoritmu detekuje objekty. Další systém funguje na principu predikování objektů s pomocí konvolučních neuronový sítí na základě *SSD512* architektury. Oba způsoby byly testovány na *Caltech* a mém vlastním datasetu.

Klíčová slova: strojové učení, strojové vidění, detekování objektů Histogram of Oriented Gradients, neuronová konvoluční síť, Single Shot MultiBox Detector

Annotation

This work deals with Automatic person detection. There are two approaches to person detection that are studied theoretically and practically tested in the work. The first approach is based on the small window that slides across an input image and computes *HOG* features. Based on these features, the SVM classifier is trained. The second approach is based on convolutional neural network *SSD512*. Both approaches were tested on the *Caltech* dataset and on my own dataset.

Keywords: Machine Learning, Computer Vision , Object Detection, Histogram of Oriented Gradients, Convolutional Neural Network, Single Shot MultiBox Detector

Poděkování

Především bych rád poděkoval vedoucímu mé bakalářské práce Ing. Ivanu Gruberovi, Ph.D. za užitečné rady a čas strávený při konzultacích. Dále děkuji panu Piotru Dollárovi za poskytnutí datasetu.

Obsah

1 Úvod

Rozpoznávání a detekce osob je pro zdravého člověka jednoduchá a fundamentální činnost. Kybernetici se snaží tuto lidskou schopnost implementovat do počítačů s využitím kamer, za účelem náhrady člověka strojem. Této oblasti se říká strojové vidění. Díky neustálému zlepšování a zrychlování výpočetních schopností počítačů se mohou vytvářet složitější a přesnější systémy pro detekci.

Obecně je detekování lidí poměrně náročná úloha, protože je každý člověk unikátní a má své specifické vlastnosti. Mezi tyto vlastnosti patří například výška, tvar, poloha ve které se zrovna nachází, nebo může mít nějakou atypickou pokrývku hlavy. Proto je snaha nalézt model člověka s pomocí velkého množství dat, podle kterého by mohl detekční systém rozpoznat a správně zaklasifikovat.

Je několik typů a několik metod pro detekování lidské postavy. Tato práce se zabývá dvěma typy detekčních systémů. Systému s hrubou početní silou, který prohledává pomocí posuvného okénka celý snímek (*HOG+SVM*). A systém s využitím konvoluční neuronové sítě, který je založen na principu umělých neuronových sítí (*SSD*). Obě metody jsou teoreticky probírány a prakticky tvořeny a testovány pomocí *Caltech Pedestrian* datasetu.

2 Digitální obraz

Okolní obraz je obvykle pořizován pomocí kamery, nebo fotoaparátu. Digitální obraz je následně uložen jako matice pixelů v paměti zařízení, kde každý pixel nese informaci o barvě. Většina digitálních obrazů využívá takzvaný 24 bitový RGB model. RGB model funguje na základě míchání aditivních barev (červená, zelená, modrá). Pixel v barevném snímku má tedy 3 hodnoty s velikostí 8 bitů a proto každá složka RGB modelu nabývá hodnot 0-255. Příklady RGB barev:

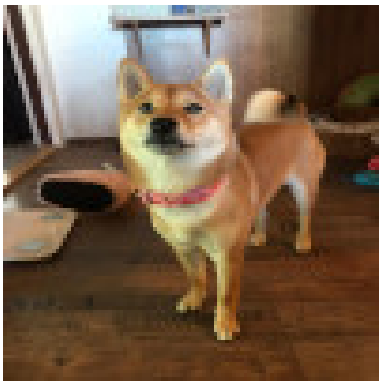
$$\begin{aligned} \textit{červená barva:} &\rightarrow rgb(255, 0, 0), \\ \textit{fialová barva:} &\rightarrow rgb(140, 0, 140), \\ \textit{žlutá barva:} &\rightarrow rgb(255, 255, 0), \\ \textit{bílá barva:} &\rightarrow rgb(255, 255, 255). \end{aligned}$$

V digitální světě se objevují dva základní typy RGB modelů. První z nich nese název *sRGB*. Tento typ vyvinula jako standard firma *Microsoft* a využívá ho většina kamer, monitorů i tiskáren. Druhý typ je známý jako *Adobe RGB*, který byl vytvořený s větším rozsahem barev. Existuje ještě také popis barevného obrázku pomocí *CMY(K)* modelu, který na rozdíl od RGB modelu využívá rozdílového míšení barev(tyrkysová, purpurová, žlutá). Často se do modelu přidává také černá, za účelem vytvoření tmavějších barev. Tento typ se uplatňuje se nejčastěji v tiskárnách.

RGB model dokáže vyprodukovat až 16 milionů barev, což je dostatečné pro kvalitní digitální obraz. Jak již bylo zmíněno, každý tento pixel nese barevnou informaci. Proto při kvalitě počítačového snímku hraje velkou roli také rozlišení. Tato kvalita se obvykle znační jako takzvaný *megapixel*. Tento údaj říká, kolik má obraz celkem pixelů. Jestli-že máme v počítači uloženou fotku, která má rozlišení (2000x1500) pixelů, kde první hodnota je počet sloupců pixelů a druhá počet řádků pixelů. Vynásobením těchto dvou hodnot zjistíme celkový počet pixelů, což je 3000000, neboli 3 megapixely. Digitální kamery a fotoaparáty jsou na základě této hodnoty kvalitativně hodnoceny. Příklad obrazu s barveným modelem sRGB a rozměren (1200x1600) pixelů je ukázaný na Obrázku 1.

Obrázek 1: sRGB digitální obraz

S digitálním obrazem pořízeným z fotoaparátu nebo kamery se nemusí dobře a pohodlně pracovat. Proto je často snaha zpracovat obraz do takové podoby, jakou vyžaduje daná úloha. Tato oblast se nazývá předzpracování obrazu [18]. Taková častá operace je škálování obrazu, při kterém dochází ke změně velikosti obrazu (viz. Obrázek 2). Další velmi častým předzpracováním bývá převedení snímku do odstínů šedi(viz. Obrázek 3). To má za následek ten, že snímek bude obsahovat pouze jeden kanál informací. Díky tomu bude mít každý pixel pouze 1 hodnotu a to odstín šedé. Hodnota pixelu nabývá (0-255), kde 0 je černá a 255 bílá.



Obrázek 2: Škálovaný obraz na (100x100) pixelů

Obrázek 3: Obrázek v odstínech šedi

3 Detekce lidí pomocí HOG

Histogram of Oriented Gradients (HOG) [7] [20] je metoda, která získává příznaky z obrazu. Ve spojením s klasifikačním algoritmem je *HOG* velmi populární způsob strojové detekce. Algoritmy na tomto principu dokáží z obrázků či videí nalézt požadovaný objekt, v našem případě lidskou postavu. Detekce lidí je obecně poměrně obtížná úloha, protože každý člověk je unikátní. Má určité tělesné proporce, může mít různé oblečení a může se také nacházet v různých pozicích. Proto je snaha nalézt obecný model člověka. To se dá získat pomocí informací obsažených v pixelech z mnoha snímku. Právě tyto informace získává HOG algoritmus. Hlavní myšlena HOG algoritmu je výpočet vektorů z pixelů, které určují největší směr růstu. Těmto vektorům se říká gradient. Jak je počítán bude ukázáno v kapitole **3.1**.

Princip HOG metody je založený na počítání gradientů z malých částí obrázku. Aby toto bylo možné, musí být nejprve obrázek rozdělen do malých regionů pixelů, které mají stejnou velikost a kompletně pokryjí obraz. Následně je z nich počítán histogram gradientů. Histogram je v tomto případě pole čísel, kde každý element odpovídá intenzitě gradientu pro určitý úhel. Velikost histogramu je závislá na zvoleném počtu úhlů, pro který budou intenzity gradientu přiřazovány. Sběr těchto informací je zařízené pomocí posuvného okénka, které se posouvá po malých regionech. Posuvné okénko získá veškeré HOG příznaky z malých regionů a zahrne je do jediného vektoru. Dále probíhá normalizace tohoto vektoru, která je nastíněný vzorcem (1) :

$$f = \frac{v}{\|v\| + \epsilon}, \tag{1}$$

kde v je nenormalizovaný vektor, obsahující všechny histogramy posuvného okénka, $\|v\|$ je norma vektoru a ϵ konstanta. Po získání všech HOG příznaků metoda končí. Na Obrázku 5 je popis pomocí HOG příznaků. Bílé čáry představují směr gradientu a intenzitu reprezentuje jejich tloušťka. Nalevo od něj je originální obrázek ze kterého byly počítány HOG příznaky (viz. Obrázek 4).

Obrázek 4: Lyžař

Obrázek 5: HOG vlastnosti obrázku lyžaře

Tento algoritmus sám o sobě nedokáže určit, zda se na zkoumaném snímku nachází požadovaný objekt. K tomu je třeba využít vhodnou klasifikační metodu. Jelikož chceme části obrazu klasifikovat pouze do dvou tříd, nabízí se pro tuto úlohu SVM klasifikátor. Ten je podrobněji popsán v kapitole **3.2**.

3.1 Gradient vektorů

Gradient vektorů [12] je jedním ze základních principů, které se využívají ve spojitosti s počítačovým viděním. Ten počítá změnu hodnot pixelů ve směru osy x a y. V matematice se gradient využívá ke zkoumání funkcí více proměnných. S jeho pomocí lze odhalit směr největšího růstu funkce. Obdobně to funguje i při počítání gradientu v digitálních obrazech. Vztahy pro gradient:

$$\nabla F = \left[\frac{\delta F}{\delta x}, \quad \frac{\delta F}{\delta y}\right], \tag{2}$$

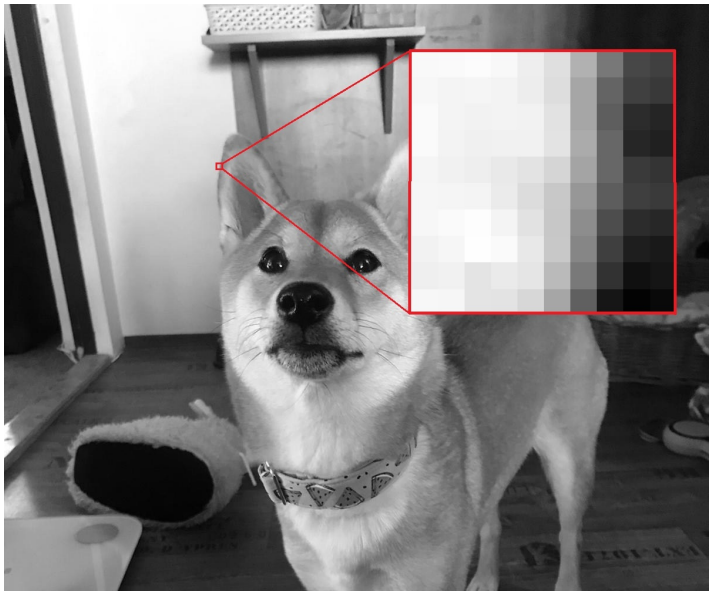
$$\theta = tan^{-1}\left[\frac{\frac{\delta F}{\delta y}}{\frac{\delta F}{\delta x}}\right], \tag{3}$$

$$\|\nabla F\| = \sqrt{(\frac{\delta F}{\delta x})^2 + (\frac{\delta F}{\delta y})^2}. \tag{4}$$

Ve vzorci (2) je ukázán gradient vektorů, který se značí jako ∇F. Protože digitální obraz má 2 dimenze, musí být vektor popsán dvěmi souřadnicemi. První souřadnicí je $\frac{\delta F}{\delta x}$, její hodnota udává změnu pixelů v ose x a $\frac{\delta F}{\delta y}$ udává změnu hodnot pixelů v ose y. Směr tohoto vektoru

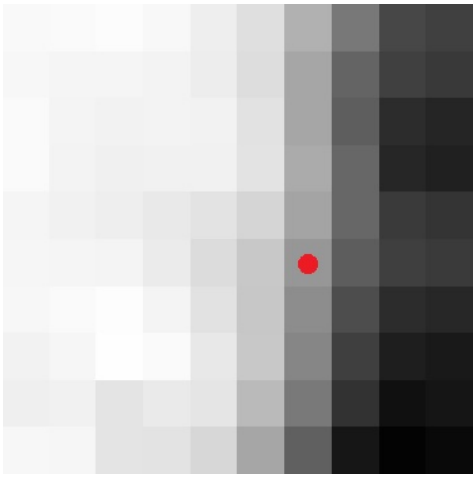
znázorněný ve vzorci (3). Ten se vypočítá z funkce tan^{-1} jako podíl změn pixelů v ose y a ose x. V posledním vzorci (4) je výpočet magnitudy gradientu. Magnituda určuje intenzitu celého vektoru a získá se z odmocniny součtů druhých mocnin $\frac{\delta F}{\delta x}$ a $\frac{\delta F}{\delta x}$.

Gradient se využívá k odhalení hran mezi jednotlivými objekty na snímcích, protože z něj lze zjistit intenzitu a směr změn v pixelech. Předpokládá se, že objekty mají jiné barvy a jiný jas, proto u hrany mezi objektem a pozadím bude intenzita gradientu velká. Jak gradient vektorů funguje, lze ukázat na jednoduchém příkladu. Pro názornost je toto demonstrováno na Obrázku 6, který byl předzpracován do odstínu šedi.



Obrázek 6: Přiblížení obrázku

Část testovaného obrázku byla vybrána a byla přiblížena až na úroveň jednotlivých pixelů. To je na Obrázku 6 znázorněné červeným rámečkem. Na první pohled je znatelná změna mezi pixely, to může signalizovat hranu. Gradient, jeho směr a magnitudy bude počítán z pixelu označeném červenou tečkou z Obrázku 7. Pod tímto obrázkem se nachází matice pixelů, které nesou informaci o jasu.



Obrázek 7: Přiblížená část

$$\begin{bmatrix}
212.00 & 212.00 & 213.00 & 211.00 & 209.00 & 204.00 & 191.00 & 174.00 & 161.00 & 156.00 \\
212.00 & 211.00 & 212.00 & 211.00 & 210.00 & 204.00 & 189.00 & 170.00 & 157.00 & 155.00 \\
212.00 & 211.00 & 210.00 & 209.00 & 208.00 & 204.00 & 188.00 & 165.00 & 152.00 & 150.00 \\
211.00 & 211.00 & 211.00 & 210.00 & 210.00 & 205.00 & 191.00 & 170.00 & 153.00 & 147.00 \\
211.00 & 211.00 & 208.00 & 207.00 & 205.00 & 200.00 & 187.00 & 169.00 & 157.00 & 154.00 \\
211.00 & 211.00 & 210.00 & 207.00 & 204.00 & 197.00 & 182.00 & 166.00 & 156.00 & 156.00 \\
211.00 & 211.00 & 214.00 & 211.00 & 207.00 & 197.00 & 180.00 & 161.00 & 152.00 & 151.00 \\
211.00 & 211.00 & 213.00 & 211.00 & 208.00 & 199.00 & 179.00 & 158.00 & 148.00 & 147.00 \\
211.00 & 211.00 & 206.00 & 207.00 & 205.00 & 196.00 & 175.00 & 154.00 & 145.00 & 146.00 \\
211.00 & 211.00 & 206.00 & 206.00 & 201.00 & 188.00 & 166.00 & 145.00 & 139.00 & 142.00
\end{bmatrix}$$

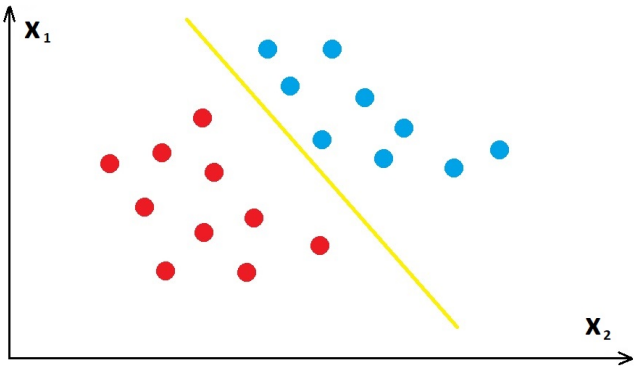
Pomocí vztahů (2),(3) a (4) byl pro označený pixel spočítán gradient, jeho směr a magnituda:

$$\textbf{Změna v ose x} \rightarrow \frac{\delta F}{\delta x} = |197 - 166| = 31,$$
$$\textbf{Změna v ose y} \rightarrow \frac{\delta F}{\delta x} = |187 - 180| = 7,$$
$$\textbf{Gradient} \rightarrow \nabla F = \begin{bmatrix} 31 \\ 7 \end{bmatrix},$$
$$\textbf{Magnituda} \rightarrow \|\nabla F\| = \sqrt{(31)^2 + 7^2} = 31.7805,$$
$$\textbf{Směr vektoru} \rightarrow \theta = \frac{7}{31} = 0.2221 rad.$$

3.2 SVM klasifikační metoda

Suport vector machine (SVM) [2] je metoda strojového učení, která plní klasifikační funkci a to právě pro 2 skupiny tříd. Je to algoritmus, který dokáže vstupní data rozpoznat a zařadit do správné skupiny. Toto rozhodnutí dělá na základě meze a pomocí počítání vzdáleností od ní.

Aby byl objekt vhodně zařazen, musí být také vhodně popsán. Proto je kladený důraz na sběr důležitých dat. Parametry SVM se můžou nastavovat ručně, hrozí ale zde riziko, že algoritmus bude špatně zařazovat vstupní data do tříd. Vhodnější řešení je natrénovaní algoritmu na základě trénovací množiny dat. Tyto data jsou korektně zaklasifikované do tříd. S jejich pomocí lze získat velmi přesný klasifikační algoritmus. Obecně platí, že čím je trénovací množina větší, tím je SVM přesnější. Stroj který následně autonomně provádí klasifikaci, je nazýván klasifikátor. SVM patří do skupiny lineárních klasifikátorů.



Obrázek 8: Lineární klasifikátor se separovatelnými případy

Na Obrázku 8 je jednoduchý příklad klasifikátoru ve dvojrozměrné dimenzi. Červeně jsou označena data z trénovací množiny, která patří do jedné třídy a modře data z druhé třídy. Pomocí těchto informací byl nastavený klasifikační algoritmus znázorněný žlutou barvou. Můžeme si pod tím představit hranici, která odděluje separabilní třídy. Přístup který který hledá SVM parametry pro kompletně separabilní třídy se jmenuje *Hard-Margin SVM*. Jelikož se jedná o lineární klasifikátor, bude mít tvar lineární přímky (viz. vzorec (5)):

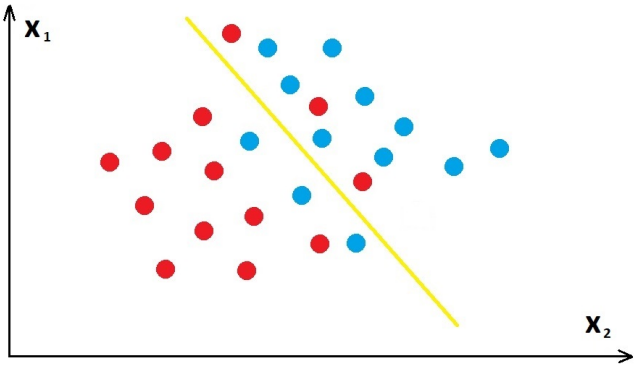
$$g(x) = w^T \cdot x + b, \tag{5}$$

kde x je vektor zvolených příznaků, w^T vektor koeficientů nastavení klasifikátoru a b je bias. Cíl SVM klasifikátoru je nalézt vhodně parametry w a b. To může být dosaženo pomocí

minimalizování objektové funkce(viz. vzorec (6)):

$$J(w) = \frac{1}{2} \cdot \|w\|^2, \tag{6}$$

Toto bude ale fungovat jen v případě, že jsou jednotlivé třídy separovatelné. Může nastat případ kdy se budou třídy překrývat(viz Obrázek 9).



Obrázek 9: Lineární klasifikátor s neseparovatelnými třídami

Z Obrázku 9 je patrné, že proložení lineární přímky tak, aby oddělovala červené a modré data je nemožné. Úloha je řešená kompromisem, tak aby bylo co nejvíce dat korektně zaklasifikováno a dovoluje modelu pár špatných zařazení. Tomuto přístupu se říká *Soft-Margin SVM*. Objektová funkce pro tento případ bude mít tvar(viz. vzorec (7,8)):

$$J(w, b, \xi) = \frac{1}{2} \cdot \|w\|^2 + C \sum_{i=1}^{N} \xi_i, \tag{7}$$

$$\xi_i \geq 0, i = 1, 2, ...N, \tag{8}$$

kde ξ_i dovoluje modelu špatné zaklasifikování a C je konstantní parametr který určuje pásmo, ve kterém je možné body špatně zařadit.

Existuje mnoho dalších klasifikačních metod, které jsou přesnější a dosahují tak daleko lepších výsledků. SVM se ve spojení s detekcí lidí používá především pro to, že klasifikujeme data pouze do 2 tříd. A to jestli je jsou data člověk a nebo okolí. Dále je vhodný pro svoje rychlé trénování a taktéž pro rychlé vyhodnocování.

3.3 Spojení HOG a SVM

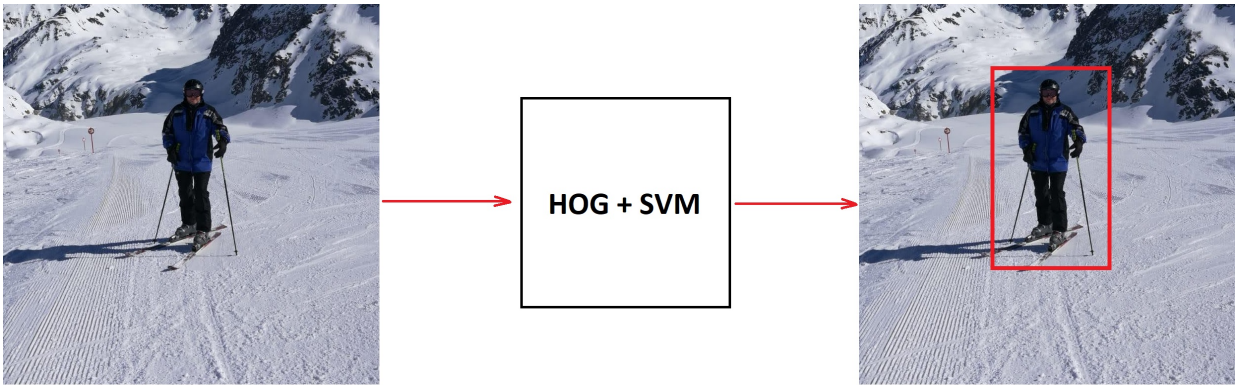
Pro tuto úlohu je zapotřebí 3 kroků:

1. Získání trénovací množiny.
2. Tvorba algoritmu a trénovaní.
3. Detekce.

Hlavním bodem přípravy je získání trénovací množiny dat obrázků, na kterých se nachází hledaný objekt. Požadované objekty, v případě této práce lidská postava, se ze snímku vyříznou a zmenší se na požadovanou velikost. Pak se zařadí do databáze kladných dat. Je potřeba ze snímku také extrahovat ostatní objekty a zařadit je do záporných dat. Toto se dělá z důvodu trénování klasifikátoru, proto potřebujeme i informace o tom, co člověk není. Trénovací množina se získává poměrně obtížně, protože se musí nejprve vytvořit snímky. Následně musí lidský expert na digitálních obraz označit lidi.

S připravenou databází kladných a záporných dat je možné natrénovat klasifikační metodu SVM. Pomocí HOG algoritmu se z těchto kladných a záporných množin dat vypočítají HOG vlastnosti. Ty se předkládají klasifikátoru. Na základě trénovací množiny upravuje SVM své koeficienty. Takto klasifikační algoritmus korektně nastavíme a poté je vše připraveno k poslednímu bodu a to detekci lidí z digitálního obrazu.

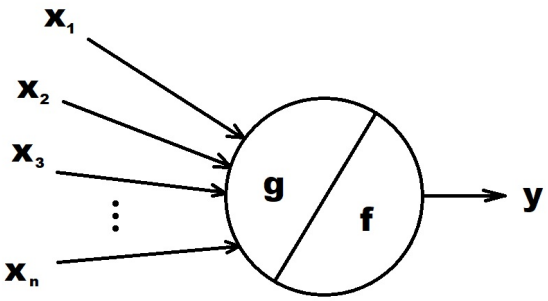
Poslední krok je nastavení způsobu detekce. To je obvykle řešeno pomocí posuvného okénka, které se posouvá po testovaném digitálním obrázku. Při každém posuvu počítá z vybrané oblasti HOG vlastnosti a pomocí klasifikátoru rozhoduje, zda-li je v okénku osoba. Pokud ano, vytvoří kolem ni rámeček. Celý tento algoritmus představuje systém, kde do systému vstupuje snímek a ze systému vystupuje ten samý obraz s označenými postavami(viz. Obrázek 10).



Obrázek 10: HOG a SVM jako systém

4 Detekce lidí s využitím neuronových sítí

Umělá neuronová síť [13] [16] je metoda strojového učení používaná v oblasti kybernetiky a umělé inteligence. Snaha umělých neuronových sítí je napodobit chování lidského mozku pomocí propojení neuronů tak, aby se mohla učit a dělat rozhodnutí jako člověk. Základní jednotkou jsou tedy neurony. Každý neuron má určitý počet propojení s ostatními neurony. Dohromady vytvářejí velkou síť. První pokusy pro sestavení umělé neuronové sítě byly před 70 lety, zabýval se jimi tehdy Warren McCulloch a Walter Pitts. Tito vědci navrhli v roce 1943 první model neuronu(viz Obrázek 11).



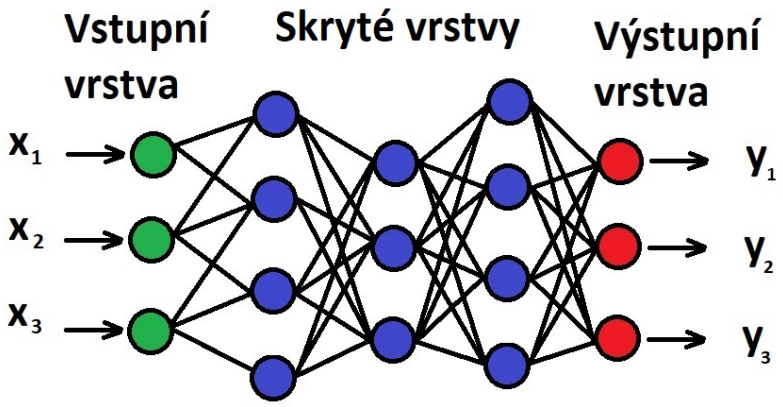
Obrázek 11: McCulloch-Pitts neuron

$$g(x) = \sum_{i=1}^{n} x_n \tag{9}$$

$$y = f(g(x)) \tag{10}$$

Na Obrázku 11 je první model neuronu. V levé části obrázku jsou vstupy, které směřují do modelu, ty nabývají hodnot $x_n \in \{0, 1\}$. Tyto vstupy se sečtou podle vzorce (9) a následně funkce y ze vzorce (10) vyhodnotí výstup z modelu, ta bude nabývat hodnot $y \in \{0, 1\}$. Později byly do modelu přidány váhy vstupů a výstupů.

Příklad neuronové sítě je znázorněný na Obrázku 11. Jedná se o *perceptron*, což je síť pouze s jedním neuronem. Příklad složitější sítě je ukázán na Obrázku 12. Zde je ukázaná neuronová síť která je složená z několika mezi sebou navzájem propojenými neurony. Síť obsahuje vstupní vrstvu, která předává hodnoty skryté vrstvě. Skryté vrstvy hodnoty zpracují a dají je výstupní vrstvě.



Obrázek 12: Neuronová síť

Neuronové sítě jsou poslední dobou stále častěji používané. Jsou považované za nejúčinnější metody v oblasti umělé inteligenci. Neuronovou síť používají například systémy pro rozpoznávání hlasu nebo automatické překladače jazyků. Neuronové sítě jsou také hojně aplikované v úlohách rozpoznávání objektů z digitálního obrazu. Stejně jako SVM klasifikátor, i neuronové sítě musí být trénovány. Toto učení upravuje váhy spojení mezi neurony, což ovlivňuje chod celé sítě.

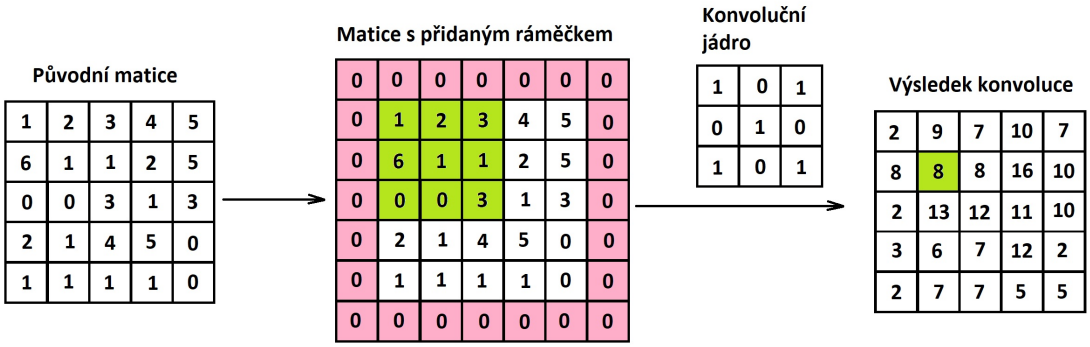
4.1 Konvoluční neuronové sítě

Pro počítačové vidění a rozpoznávání se hojně využívají konvoluční neuronové sítě [9]. Konvoluční neuronová síť (CNN) je dopředný typ neuronových sítí, která je schopná extrahovat důležité vlastnosti s využitím konvolučních struktur.

V porovnání s obecnou neuronovou sítí má konvoluční síť hned několik výhod. Jedna z výhod je, že má lokální připojení. To znamená, že neurony nejsou propojeny se všemi neurony z předešlé vrstvy, ale pouze s malým počtem neuronů. To má za následek zvýšení efektivity a zrychlení celé sítě. Další výhodou je, že mohou jednotlivá propojení mohou mezi sebou sdílet váhy. To má za následek zmenšení počtu parametrů. Tyhle vlastnosti dělají CNN jednu z nejvíce využívanou metodou v oblasti *Deep learningu*, což je odvětví které se zabývá vývojem umělých neuronových sítí.

Aby bylo možné sestavit konvoluční neuronovou síť, je třeba několik kroků. První krok je aplikování *konvoluce* pro získání vlastností z obrazu. Konvoluce je u zpracování obrazu velmi častá operace. Obraz je pomocí *konvolučního jádra*, taktéž konvoluční filtr, postupně

násoben a výsledkem této operace jsou takzvané *feature maps*. Konvoluční jádro je obvykle matice o velikosti 3x3. Tento celý proces je znázorněný na Obrázku 13.

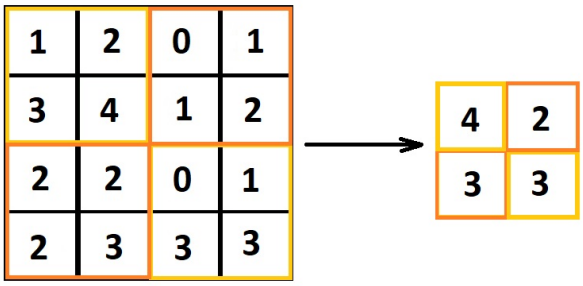


Obrázek 13: Konvoluce

Nalevo je původní matice (5x5), což můžou být například hodnoty jasu pixelů. Aby při konvoluci nedošlo ke ztrátě informací je k matici přidán rámeček, který je zvýrazněný růžovou barvou. Všechny hodnoty rámečku nabývají hodnoty nule. Následně je použita konvoluce s konvolučním jádrem o velikosti (3x3). Konvoluční jádro je násobeno s každým (3x3) úsekem obrázku, hodnoty se sčítají a zaznamenávají se do jedné buňky. Jeden takový proces je zvýrazněný zelenou barvou:

$$\begin{bmatrix} 1 & 2 & 3 \\ 6 & 1 & 1 \\ 0 & 0 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 1 + 3 + 1 + 3 = 8$$

Úplně napravo Obrázku 13 je výsledek konvoluce. Po konvoluci obsahují *feature maps* velké množství dat. Data jsou redukována pomocí sdružování malých regionů, jmenovitě maximální sdružování*(max-pooling)* a průměrné sdružování*(average-pooling)*. Příklad maximálního sdružování je ukázán na Obrázku 14.



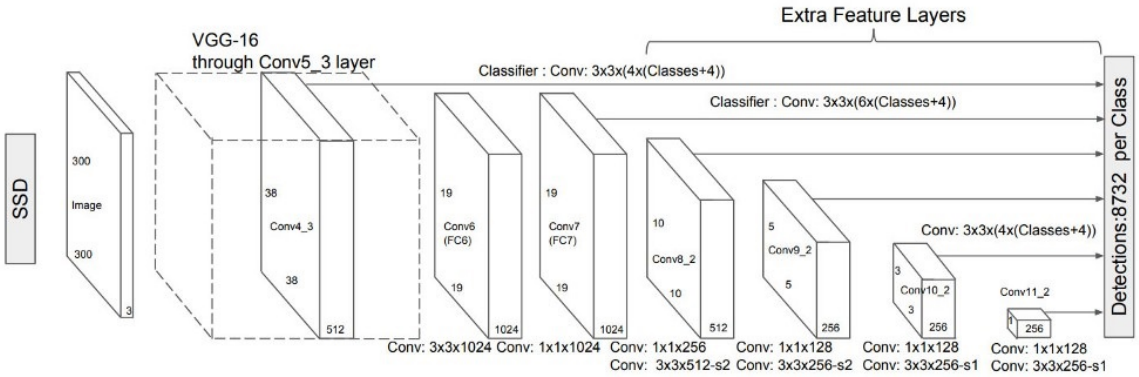
Obrázek 14: Maximální sdružování

Těchto konvolučních vrstev může být v CNN několik a nacházejí se ve skrytých vrstvách. Jednotlivé konvoluční vrstvy mohou mít odlišné konvoluční jádra, díky kterým je možné detekování různých objektů. Každá vrstva má obvykle více konvolučních jader.

4.2 SSD

Single Shot MultiBox Detector (SSD) [11] je velmi schopný detekční systém, který dokáže pracovat v reálném čase. Byl publikovaný v listopadu roku 2016 a dosáhl až 74% *Average precision* při 59 snímkách za sekundu. SSD je založen na dopředné konvoluční síti, která nachází objekty ze snímku a přiděluje jim skóre. Podle skóre následně objekt zaklasifikuje.

SSD v prvním kroku využívá konvoluční neuronovou síť *VGG-16* [17] pro získání *feature maps*. Důvod proč byla *VGG-16* vybrána jako základní síť je ten, že má kvalitní výkon a dobré klasifikační výsledky i s velmi kvalitními digitálními obrazy. Poté je do sítě přidáno několik konvolučních vrstev. Každá vrstva může předpovídat detekci s využitím malých konvolučních filtrů. Tyto předpovědi se skládají z ohraničení a skóre pro každou klasifikovanou třídu. Algoritmus jako objekt označí ten, která splňuje podmínku minimálního prahu a třídu, která má největší skóre. Pro detekci různých variant objektů využívá metoda konvoluci ke zmenšování dimenze *feature maps*. Díky tomu může nalézt objekty více velikostí. Přičemž každému potenciálnímu objektu přiřazuje set několika ohraničení a každému ohraničení také objektové skóre.



Obrázek 15: Schéma SSD, zdroj: https://arxiv.org/pdf/1512.02325.pdf

Na Obrázku 15 je schéma SSD konvoluční sítě. V prvním kroku je na obrázek (300x300) aplikovaná konvoluční síť *VGG-16* ve které je přidaná konvoluční vrstva *Conv4_3* pro predikování ohraničení. Dále jsou v modelu přidání další vrstvy, které nachází a predikují ohraničení objektů.

Nejvíce se SSD od ostatních typických detektorů odlišuje trénováním. Během trénování je důležité jaké predikované objekty odpovídají těm z trénovací množiny dat. K porovnání této přesnosti SSD používá ztrátovou funkci, která kombinuje dvě hodnoty:

- *Confidence Loss*,
- *Location Loss*.

Confidence Loss určuje hodnotu jak moc je vyznačená oblast objektem, *Location Loss* udává jak moc daleko jsou předpověděné oblasti od skutečných oblastí z trénovací množiny dat.

$$loss = \frac{1}{N}(confidenceLoss + \alpha \cdot locationLoss) \tag{11}$$

N parametr z vzorce (11) je číslo nalezených ohraničení a α určuje váhu *Location Loss*. Cílem trénování je najít takové parametry, aby byla váhová funkce co nejmenší, tím se bude předpovídaná detekce blížit hodnotám z trénovací množiny dat.

5 Dataset

Mít vhodný dataset je základní kámen trénování detekčního systému i jeho následné testování. Je žádoucí, aby byla data objemná a aby byla vhodně přizpůsobená pro danou úlohu. Data pro detekční úlohu musejí obsahovat digitální obraz v podobě obrázků, nebo videonahrávky. K těmto digitálním obrazům musejí být informace o korektně označených hledaných objektech, těmto informacím se říká *anotace*. Celkový dataset by měl být rozdělen do třech částí:

- trénovací data,
- validační data,
- testovací data.

Trénovací data slouží k trénování modelu. Při trénovacím procesu jsou pomocí dat z trénovací množiny upravovány váhy modelu. Konkrétně jsou upravovány hodnoty klasifikátoru v případě *HOG+SVM* metody, nebo váhy neuronů v případě *SSD*. Tato část datasetu by měla být největší, protože chceme aby byl model co nejpřesněji natrénován. To se dá dosáhnout velkou množinou dat, kde jsou osoby v různých polohách s různými vlastnostmi. Pomocí validačních dat se provede validace natrénovaného modelu. A testovací data se na závěr používají k testování kompletního detekčního systému. Tyto dvě množiny dat bývají v porovnání s trénovací množinou o poznání menší.

Pro celou práci byl použit dataset s názvem *Caltech Pedestrian Dataset* [5][6], který obsahoval tisíce obrázku s anotací lidských postav. Publikoval ho pan Piotr Dollár na stránce: *http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/*.
Z tohoto datasetu byla využita část s názvem *caltech-roadside-pedestrians*, která obsahuje zhruba 21000 obrázků ve formátu *.png* a soubor *.pkl*, ve kterém byly informace o označených osobách. Takto získané data bylo třeba rozdělit na trénovací, validační a testovací množiny v poměru:

- trénovací množina → 16000 obrázků,
- validační množina → 2500 obrázků,
- testovací množina → 2500 obrázků.

5.1 Příprava datasetu pro HOG+SVM

Na základě anotací z datasetu byly z obrázků vybrány a vyříznuty lidské postavy a následně transformovány v poměru 1:2 na velikost (64x128 pixelů). Takto zpracované části obrazu byly označeny jako pozitivní data. Dále se vybíraly náhodné části obrázků, které neobsahovali lidskou postavu a také se z obrazu vyřízly a transformovaly na velikost (64x128 pixelů). Tyto obrazy se zařadily do negativních dat(viz. Obrázek 16).

Obrázek 16: Příprava datasetu

Tímto způsobem byla vytvořena trénovací a validační množina dat. Na originálním datasetu byla část lidí, která nesplňovala minimální podmínku velikosti (64x128), v takovém případě byli tito lidé ignorováni. Kvůli tomu jsou pozitivní data o několik obrázků menší. Celkový počet nasbíraných pozitivních a negativních dat pro tyto množiny jsou znázorněny v Tabulce 1.

	Počet pozitivních dat	Počet negativních dat
Trénovací množina	11434	47684
Validační množina	1412	6947

Tabulka 1: Trénovací a validační množina pro HOG+SVM

Dále bylo třeba sestavit testovací množinu dat. To bylo provedeno zmenšením originálních digitálních obrazů na velikost (360x640) a anotace byly uloženy pro lepší manipulaci do datové struktury *.json*. S pomocí této množiny se provádí finální testování detekčního systému HOG+SVM.

5.2 Příprava datasetu pro SSD

Originální obrázky z *Caltech* datasetu mají rozměr (1280x720). Aby bylo možné z těchto obrázků natrénovat model, musely být transformovány na velikost (512x512). Tento rozměr je daný modelem SSD, kde jsou na vstup požadovány rozměry (512x512). Dále došlo k vydělení všech barevných kanálů obrázků hodnotou 255, aby nabývaly hodnot $< 0, 1 >$. Této operaci se říká normalizace a neuronová síť díky ní dosahuje lepších trénovacích výsledků. Celé předzpracování je znázorněné na Obrázku 17.

Obrázek 17: Předzpracování datasetu pro SSD

Veškeré informace o osobách byly uložené v souboru *.pkl*. Ty byly předělány do datové struktury *.json* podle vzoru *MS COCO* datasetu [10]. Příklad formátu *.json* je uvedený níže.


```
{
"images": [{
        "file_name": "2012-11-08_0.jpg",
        "height": 512,
        "width": 512,
        "id": 0
}],
"annotations": [{
        "segmentation": [],
        "area": "",
        "iscrowd": 0,
        "image_id": 0,
        "bbox": [13, 235, 15, 52],
        "category_id": 1,
        "id": 0
}],
```

```
"categories": [{
        "supercategory": "person",
        "id": 1,
        "name": "person"
}]
}
```


Jednotlivé prvky v souboru *.json* na sebe odkazují pomocí *"id", "image_id, category_id"*. Díky tomu může mít jeden obrázek více informací o osobách. Tyto informace jsou uložené v prvku *"bbox"*, který má 4 parametry → [x,y,šířka,výška].

V průběhu trénování sítě bylo zjištěno, že postavy které jsou na obrázcích velmi vzdálený tvoří problémový element. Po zmenšení digitálního obrazu se logicky zmenšili i samotní lidé. Na základě experimentů se ukázalo, že postavy které jsou nižší než 50 pixelů silně ztěžují trénink sítě. Síť se v jejich důsledku snažila i drobné a nevýznamné objekty zaklasifikovat jako osoby. Proto byly v rámci zvýšení stability vyřazeny veškeré osoby s velikostí menší než 50 pixelů. Díky tomu se rapidně zvýšila efektivita a přesnost trénování.

5.3 Příprava vlastního datasetu

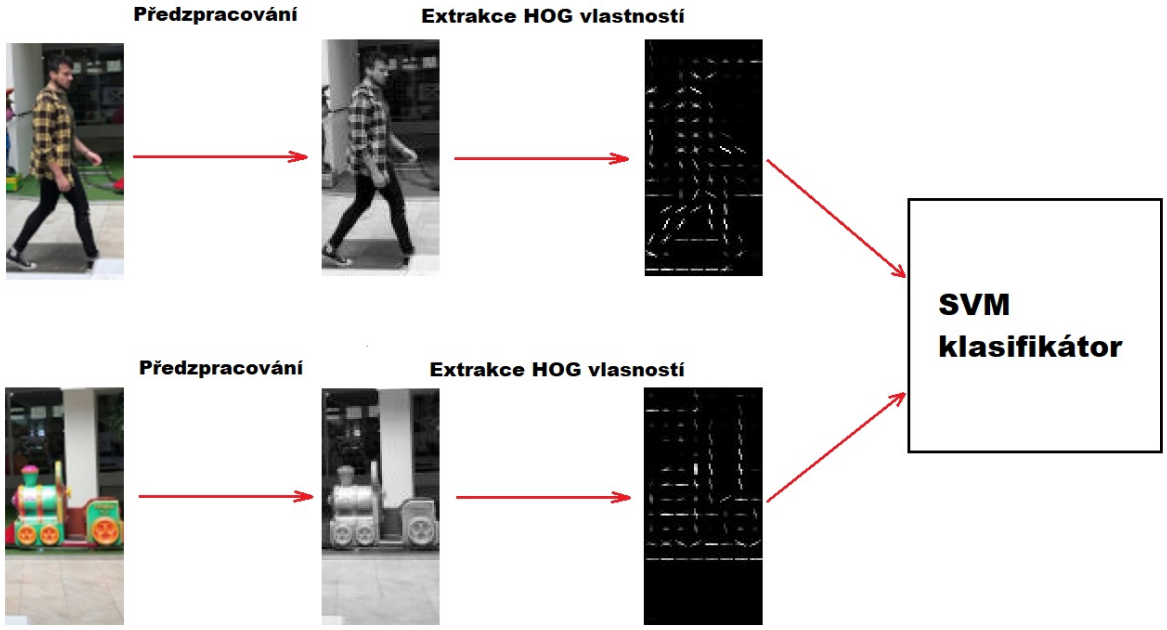
Pro otestování robustnosti byl vytvořen vlastní dataset. Obrázky byly pořízeny videozáznamem z kamery zabudované v chytrém telefonu. Celkem bylo nasbíráno zhruba 2000 obrázků. Obrázky bylo třeba protřídit a vybrat pouze ty, které obsahovaly lidské postavy. Následně byly anotační informace obdobně jako u datasetu pro SSD a HOG+SVM ukládány do datové struktury *.json*. Vlastní dataset obsahuje celkem 100 obrázků. Ukázka ručně zaanotovaného digitální obrazu je ukázaný na Obrázku 18, kde zelený rámeček reprezentuje anotaci.

Obrázek 18: Ukázka ruční anotace

6 Experimenty

6.1 Trénování SVM klasifikátoru

Prvním krokem trénování SVM klasifikátoru je předzpracování pozitivní i negativní množiny dat. Všechny obrázky byly převedeny do jednoho kanálu, odstínu šedi. Z takto předzpracovaných obrázků vypočítává algoritmus HOG příznaky. Příznaky jsou předkládány klasifikátoru a ten na jejich základě upravuje své koeficienty (viz. Obrázek 19).



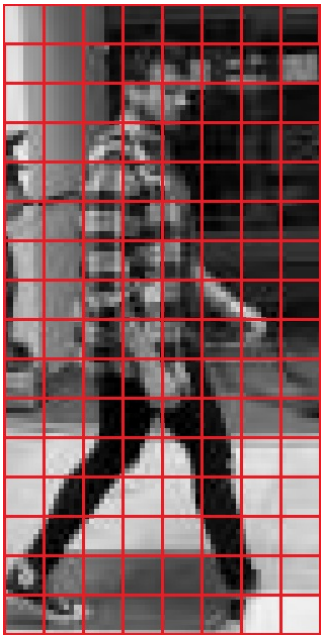
Obrázek 19: Předzpracování, extrahování HOG vlasností a trénování

Pro extrahování HOG příznaků je nutné určit parametry, podle kterých budou příznaky počítány a následně i extrahovány. První parametr je velikost malých pixelových regionů. Je třeba určit takovou velikost, aby pomocí malých regionů byl kompletně pokryt celý obrázek a nebyl vynechán žádný pixel. Dále je nutné zadefinovat pro kolik směrů gradientu bude z těchto malých regionů počítáno. Nakonec je třeba znát velikost posuvného okénka, které se posouvá po malých regionech a sbírá z nich hodnoty histogramů.

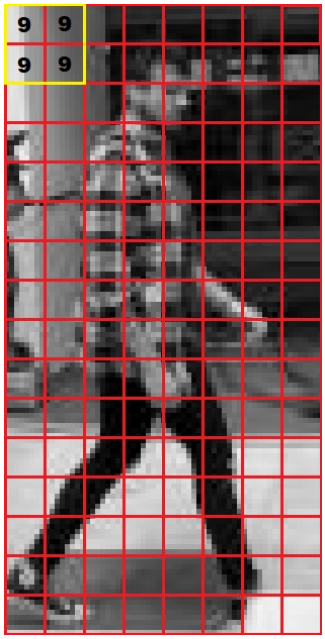
Pro extrahování HOG příznaků byla využita funkce *hog()* z knihovny *skimage* [19] určená pro programovací jazyk *Python*. Zvolené vstupní parametry *hog()* funkce:

- *orientations*=9,
- pixels_per_cell=(8,8),
- cells_per_block =(2,2).

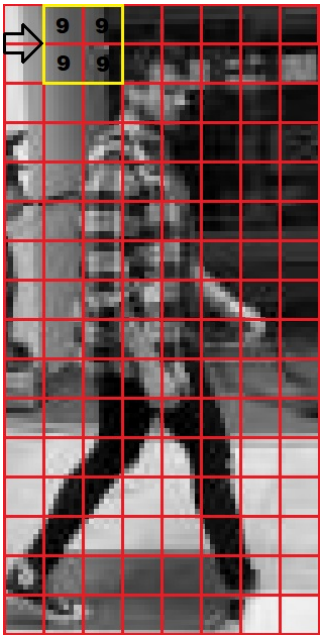
Jelikož jsou všechny obrázky z pozitivních i negativních dat normalizovány na velikost (64x128) pixelů a jeden region má obsahovat (8x8) pixelů, bude mít zkoumaný obrázek (8x16) těchto malých regionů (viz. Obrázek 20). Pro každý region jsou počítány histogramy pro 9 velikostí gradientu. Dále se pomocí posuvného okénka (2x2) vždy zkombinují 4 regiony (viz. Obrázek 21), které bude obsahovat celkem 36 HOG příznaků a zároveň bude tyto informace normalizovat. Okénko se bude pohybovat po všech buňkách přes celý obrázek (viz. Obrázek 22). Celkově bude nasbíráno 3780 HOG příznaků.



Obrázek 20: Obrázek z pozitivních dat, kde jsou pixely rozděleny do malých regionů



Obrázek 21: Obrázek z pozitivních dat, kde je ukázáno posuvné okénko



Obrázek 22: Obrázek z pozitivních dat, kde je ukázán pohyb posuvného okénka

Všech 3780 HOG příznaků jsou předkládány SVM klasifikátoru jak z negativních tak i z pozitivních dat. Tomuto procesu se říká učení, jejímž účelem je vytvořit model. Protože máme 3780 příznaků, budeme se pohybovat v 3780 prostorové dimenzi příznaků. To znamená že SVM klasifikační metoda bude obsahovat 3780 koeficientů + bias (viz. vzorec (12)):

$$g(x) = b + w_1x_1 + w_2x_2 + ... + w_{3780}x_{3780}, \tag{12}$$

kde b je bias, w_n koeficienty a x_n příznaky. Klasifikátoru jsou předkládány HOG příznaky z obrázků. Informaci o tom, jestli jsou předloženy příznaky obrázku z pozitivních nebo negativních dat určuje množina binárních čísel, která nese informaci o třídě objektu (1 →osoba, 0→ okolí). Díky tomu SVM mění své koeficienty tak, aby dokázal správně rozhodnout o osobě z obrázku. Model je tímto způsobem získáván pomocí funkce *LinearSVC()* a trénován funkcí *fit()* z knihovny *sklearn* [15] určenou pro *Python*.

Natrénovaný klasifikátor byl po trénování otestován na validačních datech. Testovaní probíhalo za účelem zjištění správného nastavení koeficientů. Nejprve byly na vstupu do SVM testovány pozitivní data a poté negativní data z validační množiny. Výsledek je znázorněný v Tabulce 2.

	SVM zaklasifikoval jako osobu [%]
Kladné data z validační množiny	93.14
Záporné data z validační množiny	1.15
Celková úspěšnost klasifikátoru	96.14

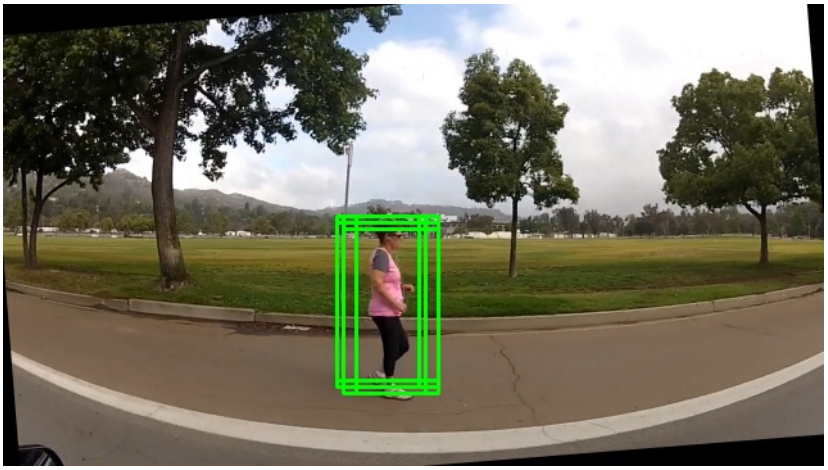
Tabulka 2: Zvalidování SVM klasifikátoru

6.2 Detekování HOG+SVM metody

Jak ale určit zda je na obrázku člověk? Nabízejí se 2 způsoby jak toto zjistit. Jeden ze způsobů je pomocí posuvného okénka, které bude mít velikost 64x128 pixelů. Pro okénko jsou počítané HOG vlastnosti. SVM klasifikátor následně na jejich základě určí, zda je v okénku osoba, či nikoliv. Toto je poměrně špatné řešení, jelikož osoby na digitálním snímku mohou mít různé velikosti a nemusí se vejít do okénka.

Daleko lepší způsob, který je také využíván v práci, je pomocí *obrázkové pyramidy*. Ta používá také posuvné okénko. Při každém posuvu ale zmenšuje zkoumaný obraz při kon-

stantním okénku a opět jsou z okénka počítány HOG vlastnosti. To zapříčiní, že můžeme na snímku nalézt i osoby, které jsou různých velikostí. Tímto způsobem ale nastává situace, kdy metoda označí osobu několika rámečky (viz. Obrázek 23).



Obrázek 23: Metoda po aplikování obrázkové pyramidy

Na Obrázku 23 je osoba, která byla označena několika zelenými rámečky. Je požadováno aby byl hledaný objekt označený pouze jedním rámečkem. K tomu slouží funkce *Non-maximum Suppression (NMS)* [3]. Ta vybere ze všech zelených rámečků ten, kterému detekční systém přiřadil největší pravděpodobnost výskytu objektu. Pak se na základě tohoto rámečku počítá *Intersection over Union* (IoU) s ostatními rámečky a pomocí prahu se odstraní sousední rámečky. *IoU* se počítá z toho důvodu, aby nedošlo k odstranění nějakého jiného rámečku, který by mohl označovat jiný objekt. Rámeček který byl pomocí *NMS* vybraný je na Obrázku 24 označený červenou barvou.

Obrázek 24: Metoda po aplikování NMS

6.3 Trénování a detekování SSD metody

Pro vytvoření SSD modelu byla využita již vytvořená implementaci SSD sítě. Originální repozitář:

https : *//github.com/pierluigiferrari/ssd_keras.*

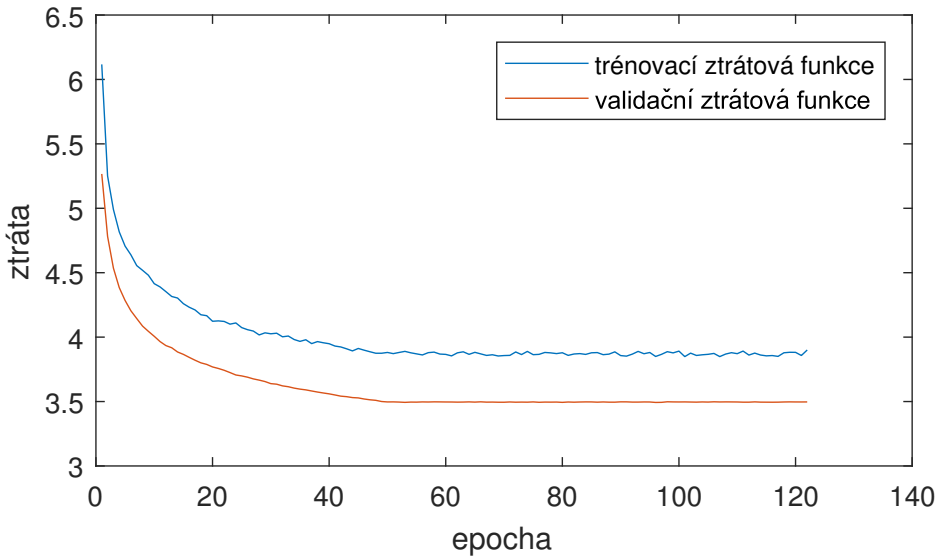
Ta má vytvořenou funkci *ssd512()*, která tvoří *Keras*[4] model na základě *SSD512* architektury [11]. *Keras* je knihovna pro práci s neuronovými sítěmi vytvořená pro programovací jazyk *Python* a funguje na základě platformy *TensorFlow* [1].

Pro vytvoření modelu pomocí *ssd512()* je zapotřebí zadefinovat několik parametrů:

- výšku obrazu,
- šířku obrazu,
- nastavení barevných kanálů obrazu,
- počet tříd,
- jaké tvary budou mít predikované ohraničení.

Jelikož byla zvolena architektura *SSD512*, která je určena pro zpracování obrazu (512x512), bylo třeba tak nastavit šířku i výšku vstupních obrazů. Předpoklad byl takový, že do detekčního systému budou vstupovat barevné obrázky, proto byly pro model nastaveny 3 barevné kanály (RGB). Váhy jednotlivých kanálu pak jako (123,117,104) a nakonec pořadí kanálů → (BRG). Tyto hodnoty vah kanálů a jejich pořadí jsou také využívány v originálním *SSD300* modelu. Počet tříd byl nastaven hodnotu 1, protože model je určený pro detekování pouze jednoho objektu a to lidí. Nastavení predikovaných ohraničení bylo zachováno takové, které opět využívá originální *SSD300* model.

SSD512 má v základu konvoluční síť VGG-16 [17]. Tu následuje 7 konvolučních vrstev, které predikují ohraničení. Celý model byl stavěn na předtrénovaných vahách neuronů. Tyto předtrénované váhy byly trénované s *MS COCO* datasetem [10] pro 80 tříd. Ze znalosti dokumentace tohoto modelu byly vybrány váhy, které jsou zodpovědné za klasifikaci lidí. S jejich pomocí byl následně vytvořen model s vahami klasifikačních vrstev incidentními s architekturou *SSD512*. S takto předpřipraveným modelem a datasetem ve správném formátu bylo vše připravené pro trénování.



Obrázek 25: Průběh trénování SSD

Trénování sítě proběhlo se startovní konstantou učení $lr = 0.0000001$. Z experimentálních trénovacích testů bylo zjištěno, že s agresivnější konstantou je trénovací průběh horší. Je to dáno tím, že trénování sítě je stavěno na předtrénovaných vahách.

Na Obrázku 25 je průběh trénování sítě. Na vodorovné ose je počet epoch a na svislé ose je hodnota ztrátové funkce. Epocha je okamžik, kdy při trénovacím procesu sítě projdou všechny trénovací data. Po každé epoše je počítána a zaznamenána ztrátová funkce. Trénovací ztrátová funkce reprezentuje v grafu modrou křivku a validační ztrátová funkce oranžovou křivku. Zhruba po 50 epochách se hodnoty ustálily:

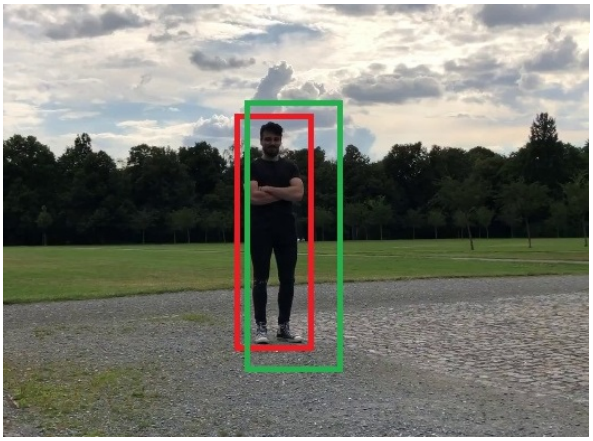
- trénovací ztrátová funkce ~ 3.850,
- validační ztrátová funkce ~ 3.495.

Nakonec pro celý model proběhla optimalizace pomocí *Adam optimizer* [8] algoritmu. Je to metoda využívaná v odvětví *deep learning* pro optimalizace modelů. Metoda funguje na principu stochastického klesání, která je založena na adaptivním odhadu momentů prvního a druhého řádu.

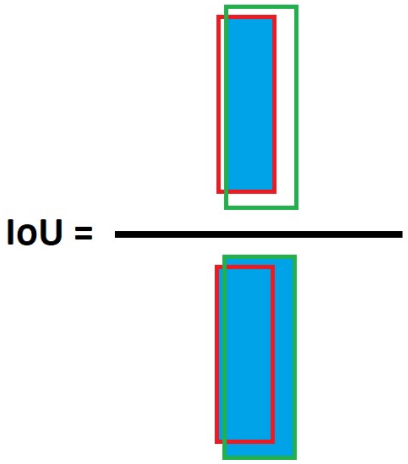
Detekce lidí z digitálního obrazu probíhá na základě predikování ohraničení z konvolučních vrstev. Každá vrstva může predikovat set ohraničení. O finální ohraničení se jako v případě HOG+SVM metody stará funkce *Non-maximum Suppression (NMS)* [3].

6.4 Testování a porovnání HOG+SVM a SSD metody

Testování modelů pro detekování objektů se často provádí na základě *Intersection over Union*(IoU) a hodnoty *Average Precision*(AP) [14]. *IoU* hodnotí přesnost detekce objektu na základě ohraničení.



Obrázek 26: Testovací obrázek s predikovanou oblastí modelem a danou oblastí testovacím datasetem



Obrázek 27: Výpočet IoU

Na Obrázku 26 je osoba, která je označena dvěma rámečky. Zeleným rámečkem je označená predikovaná oblast modelu pro výskyt osoby. Červený rámeček je *pravdivá* oblast určená ruční anotací lidského experta. Z těchto dvou oblastí je počítána hodnota *IoU* jako podíl průniku a sjednocení (viz. Obrázek 27).

Celé testování modelu je prováděno pro práh $IoU = 0.5$. To znamená, že správně detekované objekty budou hodnoceny podle pravidla:

$$\textit{pokud } IoU \geq 0.5 \rightarrow \textit{model správně detekoval objekt,}$$
$$\textit{pokud } IoU < 0.5 \rightarrow \textit{model špatně detekoval objekt.}$$

Nyní, když je zadefinováno kdy model správně rozhodl, je možné provést testování kvality sítě na základě hodnoty *Average Prcision*. Tato hodnota se získá pomocí *Precision*(P) a *Recall*(R).

$$P = \frac{T_P}{T_P + F_P} \tag{13}$$
$$R = \frac{T_P}{T_P + F_P} \tag{14}$$

Na vzorci (13) je výpočet *Precison*. V čitateli je počet správně správně zaklasifikovaných objektů (T_P) a ve jmenovateli je celkový počet zaklasifikovaných objektů ($T_P + F_P$). Hodnota *Precision* udává jak přesné jsou predikce. Čím vyšší je tato hodnota tím kvalitnější je detekční model. Výpočet *Recall* je na vzorci (14). V čitateli je také počet správně zaklasifikovaných objektů (T_P) a ve jmenovateli je součet detekovaných a nedetokovaných objektů ($T_P + F_P$). Opět je žádoucí, aby byl u testovaného modelu *Recall* co největší.

AP se poté počítá jako vzorec (15) [15]:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n, \tag{15}$$

kde R_n a P_n je *Recall* a *Precision* pro různé výsledky vyhodnocení modelu. AP kombinuje změnu *Recall* a *Precision* pro seřazené výsledky z modelu. Výpočet probíhá tak, že se zaznamenají všechny informace o detekci. Zaznamená se hodnota důvěry (*confidence*) systému o označené oblasti a informace o tom, jestli označená oblast je skutečně člověk na základě IoU. Informace se seřadí na základě hodnoty důvěry predikovaných oblastí a spočítají se pro ně *Recall* a *Precision*. Nakonec se AP spočítá podle vzorce (15). Pro výpočet AP lze použít implementovanou funkci *precision_score(y_true, y_scores)* z knihovny *sklearn* [15], které k výpočtu AP stačí neseřazené pole s hodnotami důvěry a pole s binární informací o správnosti klasifikace ($0 \rightarrow$ špatná detekce, $1 \rightarrow$ správná detekce).

AP se mění na základě prahu nastavený pro model, na jehož základě klasifikuje objekty. Proběhlo testování několik prahů pro HOG+SVM a SSD detekční systémy. Záznam testování je v Tabulce 3 a 4.

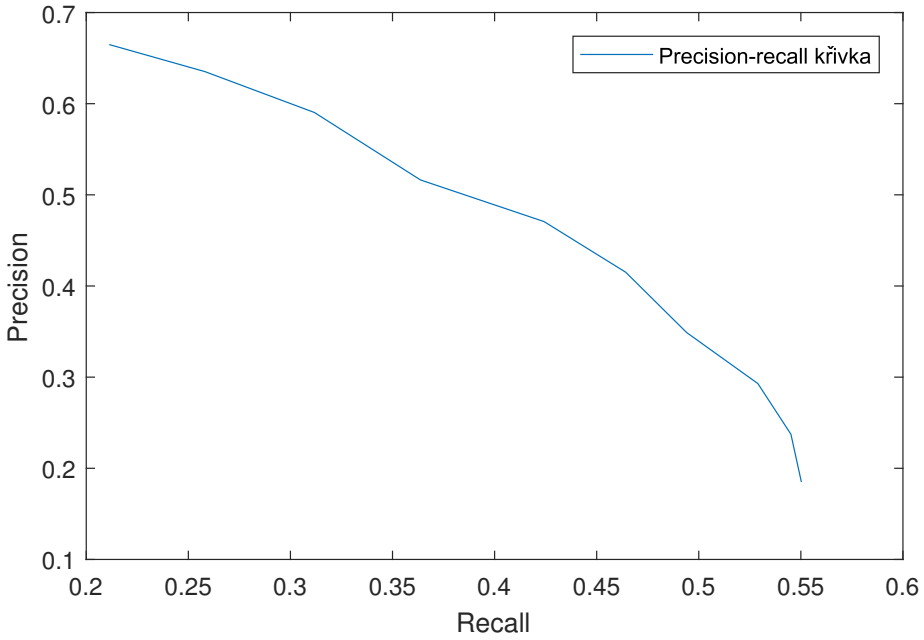
Práh	P	R	AP
2.0	0.185	0.550	0.230
2.4	0.237	0.545	0.216
2.8	0.293	0.529	0.239
3.2	0.349	0.494	0.282
3.6	0.415	0.464	0.345
4.0	0.471	0.424	0.400
4.4	0.516	0.364	0.450
4.8	0.590	0.312	0.536
5.2	0.635	0.258	0.596
5.6	0.665	0.211	0.637

Tabulka 3: Testování HOG+SVM modelu

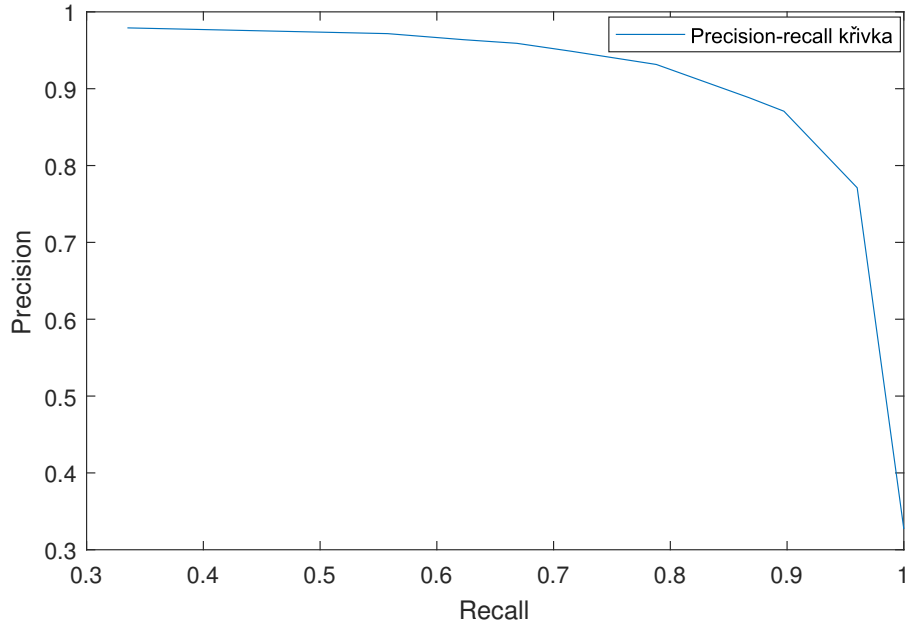
Práh	P	R	AP
0.10	0.327	1.000	0.897
0.15	0.771	0.960	0.957
0.185	0.871	0.897	0.966
0.20	0.888	0.867	0.969
0.25	0.931	0.788	0.975
0.30	0.949	0.716	0.979
0.35	0.960	0.668	0.981
0.40	0.964	0.629	0.982
0.50	0.972	0.558	0.984
0.80	0.979	0.335	0.988

Tabulka 4: Testování SSD modelu

Z Tabulek 3 a 4 je patrné, že se zvětšujícím se prahem se také zvětšuje *Precision*. Díky tomu je detekce lidí mnohem přesnější a méně se stává, že by metoda detekoval jako osobu něco jiného. Ale také se s rostoucím prahem snižuje *Recall*, protože model občas lidskou postavu vůbec nedetekuje. Dále je se zvětšujícím se prahem také zvětšovaná AP, což je opět žádoucí. Při volbě prahu jde o určitý kompromis, co je pro úlohu využití vhodnější. Závislost *Precision* a *Recallu* je znázorněný na grafech z Obrázků 28 a 29.



Obrázek 28: Precision-recall křivka pro HOG+SVM model



Obrázek 29: Precision-recall křivka pro SSD model

Pro rozhodnutí o tom, jaké nastavení detekčního systému je nejvýhodnější se využívá *F1* skóre. Je to vážený průměr hodnoty *Precision* a *Recall*. Největší možný *F1* skóre je pak nejefektivnější práh pro model. Výpočet *F1* je ukázáno na vzorci (16):

$$F1 = 2 \cdot \frac{(P \cdot R)}{(P + R)}. \tag{16}$$

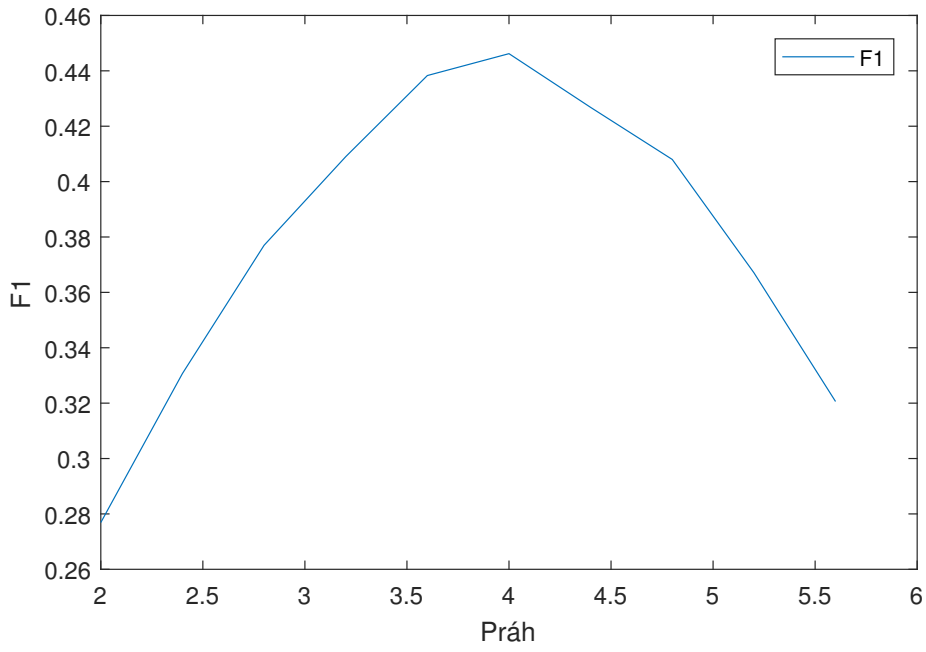
Výpočty *F1* skóre pro oba detekční systémy jsou znázorněný v Tabulkách 5,6 a graficky na Obrázcích 30,31.

Práh	F_1
2.0	0.277
2.4	0.331
2.8	0.377
3.2	0.409
3.6	0.438
4.0	0.446
4.4	0.426
4.8	0.408
5.2	0.367
5.6	0.321

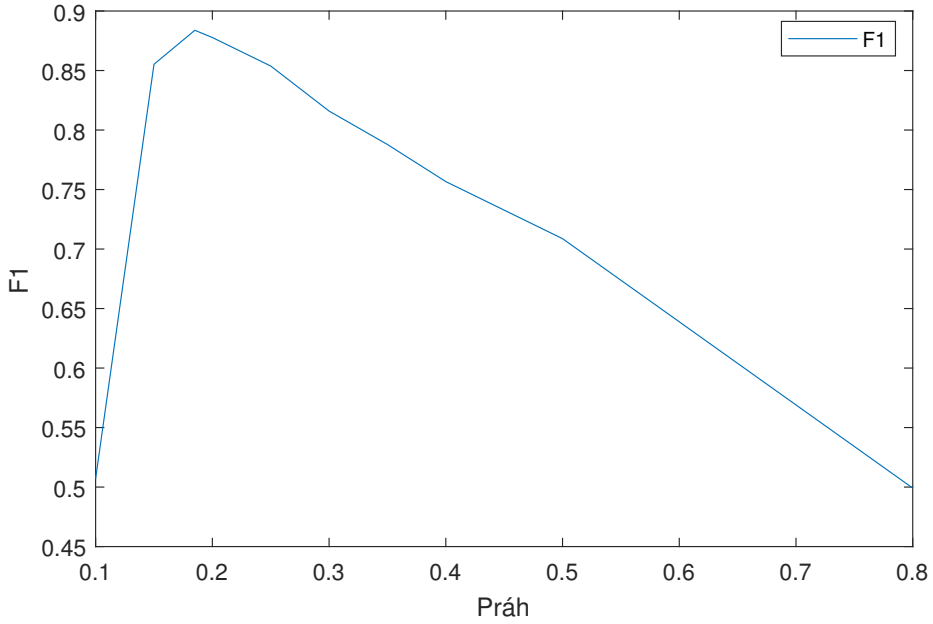
Tabulka 5: *F1* skóre pro HOG+SVM model

Práh	F_1
0.10	0.507
0.15	0.855
0.185	0.883
0.20	0.878
0.25	0.854
0.30	0.816
0.35	0.789
0.40	0.757
0.50	0.709
0.80	0.499

Tabulka 6: *F1* skóre pro SSD model



Obrázek 30: Graf *F1* skóre pro různé prahy HOG+SVM modelu



Obrázek 31: Graf *F1* skóre pro různé prahy SSD modelu

F1 křivky z obou grafů (Obrázky 30,31) zpočátku stoupají až k určité hodnotě prahu, poté začínají klesat. Maximum této křivky pro *HOG+SVM* dosahuje při prahu 4 a maximum pro *SSD* při prahu 0.185. Toto jsou prahy které jsou ve smyslu *F1* skóre nejefektivnější pro detekci. Na základě těchto prahů bylo proveden vizuální testování obou metod, kde predikované oblasti výskytu osob jsou označeny červeným rámečkem. Obrázky 32,34,36 jsou výstupy z *HOG+SVM* a Obrázky 33,35,37 z *SSD*.

Obrázek 32: Výstup z detekčního systému HOG+SVM (1)

Obrázek 33: Výstup z detekčního systému SSD (1)

Obrázek 34: Výstup z detekčního systému HOG+SVM (2)

Obrázek 35: Výstup z detekčního systému SSD (2)

Obrázek 36: Výstup z detekčního systému HOG+SVM (3)

Obrázek 37: Výstup z detekčního systému SSD (3)

Z vizualizovaných výstupních dat z Obrázků 32-37 je patrné, že SSD dosahuje při označování lidí daleko větší přesnosti. To je také patrné z hodnot *Average Precision*, kde SSD má hodnotu AP rovnou 0.966 a HOG+SVM 0.4. Také má SSD daleko rychlejší dobu výpočtu, to se ale nedá dobře srovnávat, protože SSD využívá k výpočtu grafickou kartu, přičemž HOG+SVM využívá procesor.

6.5 Testování na vlastních datech

Oba detekční systémy byly otestovány na mém vytvořeném testovacím datasetu. Toto je prováděno za účelem otestování robustnosti systémů. Výsledky testu jsou znázorněny v Tabulce 7.

	HOG+SVM	SSD
P	0.056	0.769
R	0.037	0.988
AP	0.069	0.974

Tabulka 7: Testování detekčních metod na vlastním datasetu

V testování obstál výborně SSD model, který má hodnotu *Average Precision* rovnou 0.974. Zatímco HOG+SVM model má *Average Precision* hodnotu pouze 0.069. HOG+SVM

má tak malou hodnotu AP pravděpodobně proto, že na testovacím datasetu je většina lidí menší než (64x128) pixelů, což je velikost posuvného okénka. Na Obrázcích 38,39 je vizualizován jeden výstup ze systémů testovaném na vlastním datasetu.

Obrázek 38: Výstup z detekčního systému HOG+SVM (4)

Obrázek 39: Výstup z detekčního systému SSD (4)

7 Závěr

Výsledkem bakalářské práce jsou dva funkční detekční systémy. První systém získává HOG příznaky pomocí posuvného okénka a pomocí klasifikátoru rozpoznává lidské postavy. Druhý detekční systém využívá konvoluční neuronovou síť z architektury *SSD512*, která predikuje výskyt osob z digitálních obrazů. Obě metody byly trénované, i testované pomocí stejných dat z *Caltech Pedestrian Dataset*, díky tomu je lze objektivně porovnat. Z experimentálního testování dosáhl *SSD* model o poznání lepších výsledků než *HOG+SVM*. Vyzkoušel jsem si také tvorbu vlastního datasetu u kterého bylo třeba ručně anotovat lidi. Na tomto datastu proběhlo testování robustnosti, kde detekční systém HOG+SVM moc dobře neobstál. Je to pravděpodobně způsobené tím, že většina postav na vlastním datasetu mají menší velikost než je velikost posuvného okénka. Proto většinu detekcí na základě *IoU* nebyla uznána jako správná.

V budoucnu by se dali otestovat další metody určené k detekci objektů, například *You Only Look Once* (*YOLO*) konvoluční neuronová síť. Práce by se dále v budoucnu dala rozšířit o detekování aktivit osob, tak aby *CNN* dokázala u detekovaných lidí určit například chůzi, běh, jízdu na kole, nebo třeba lyžování.

Veškeré zdrojové kódy byly zpracovány v jazyce Python a je možné je naleznout na odkazu viz. níže.
link: *https://bit.ly/2PzVp7U*

8 Seznam literatury

Odkazy

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[2] Mariette Awad a Rahul Khanna. "Support Vector Machines for Classification". In: led. 2015, s. 39–66. ISBN: 978-1-4302-5989-3. DOI: 10.1007/978-1-4302-5990-9_3.

[3] Navaneeth Bodla et al. *Soft-NMS – Improving Object Detection With One Line of Code*. 2017. arXiv: 1704.04503 [cs.CV].

[4] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[5] P. Dollár et al. "Pedestrian Detection: A Benchmark". In: *CVPR*. 2009.

[6] Piotr Dollár et al. "Pedestrian Detection: An Evaluation of the State of the Art". In: *PAMI* 34 (2012).

[7] M. Kachouane et al. "HOG based fast human detection". In: *2012 24th International Conference on Microelectronics (ICM)* (2012). DOI: 10.1109/icm.2012.6471380. URL: http://dx.doi.org/10.1109/ICM.2012.6471380.

[8] Diederik P. Kingma a Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[9] Zewen Li et al. *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*. 2020. arXiv: 2004.02806 [cs.CV].

[10] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: 1405.0312 [cs.CV].

[11] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[12] Chris McCormick. *Gradient Vectors*. URL: https://mccormickml.com/2013/05/07/gradient-vectors/. (accessed: 07 May 2013).

[13] Maad Mijwil, Adam Esen a Aysar Alsaadi. "Overview of Neural Networks". In: 1 (dub. 2019), s. 2.

[14] Kemal Oksuz et al. *Localization Recall Precision (LRP): A New Performance Metric for Object Detection*. 2018. arXiv: 1807.01696 [cs.CV].

[15] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), s. 2825–2830.

[16] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. URL: http://dx.doi.org/10.1016/j.neunet.2014.09.003.

[17] Karen Simonyan a Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[18] Jiri Stastny a Martin Minařík. "A Brief Introduction to Image Pre- Processing for Object Recognition". In: led. 2007.

[19] Stéfan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (červ. 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[20] Qiang Zhu et al. "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients". In: sv. 2. Ún. 2006, s. 1491 –1498. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.119.

9 Obrázky a tabulky

Seznam obrázků

Seznam tabulek