VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE GRAFFITI TAGŮ V OBRAZE

GRAFFITI TAGS DETECTION AT PICTURE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAN PAVLICA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAKUB ŠPAŇHEL**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Pavlica Jan**
Obor: Informační technologie
Téma: **Detekce graffiti tagů v obraze**
Detection of Graffiti Tags in Image
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku detekce textu.
2. Vyberte vhodné metody a navrhněte řešení problému detekce graffiti tagů v obraze.
3. Experimentujte s vaší implementací a případně navrhněte vlastní modifikace metod.
4. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje. Zvažte další pokračování v rámci diplomové práce.
5. Vytvořte stručný plakát a video prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadaní.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Špaňhel Jakub, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zaměřuje na možnost využití současných metod v oblasti počítačového vidění za účelem automatické detekce graffiti tagů v obraze. Tagy jsou nejčastějším projevem graffiti, který slouží jako podpis autora. V rámci práce byly otestovány state-of-art detekční systémy, z nichž se nejvíce osvědčil Single Shot MultiBox Detector. Bylo u něj dosaženo 75,7% AP.

Abstract

The thesis is focused on the possible utilization of current methods in the area of computer vision with the purpose of automatic detection of graffiti tags in the image. Graffiti tags are the most common expression of graffiti, which serves as the author's signature. In the thesis, state-of-the-art detection systems were tested; the most effective one is the Single Shot MultiBox Detector. The result has reached 75.7% AP.

Klíčová slova

graffiti tagy, detekce objektů, konvoluční neuronové sítě, YOLOv2, Tiny YOLO, You Look Only Once, SSD, Single Shot MultiBox Detector, Faster R-CNN

Keywords

graffiti tags, object detection, convolutional neural networks, YOLOv2, Tiny YOLO, You Look Only Once, SSD, Single Shot MultiBox Detector, Faster R-CNN

Citace

PAVLICA, Jan. *Detekce graffiti tagů v obraze*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Detekce graffiti tagů v obraze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. J. Špaňhela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Jan Pavlica
16. května 2017

Poděkování

Rád bych poděkoval Ing. Jakubovi Špaňhelovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce. Dále bych chtěl poděkovat Ing. Jakubovi Sochorovi za možnost využití nezbytné výpočetní techniky.

Obsah

Kapitola 1

Úvod

S graffiti se již každý někdy v životě setkal. Ačkoliv je většinovou společností odsuzováno a tresty se zpřísňují, je tato forma vandalismu stále na vzestupu. Dle policejních statistik [2] se škody ročně pohybují v desítkách miliónů korun, avšak vymahatelnost je velmi nízká. Objasněnost případů sprejerství je totiž tristní a pohybuje se kolem 20%. Mnoho lidí z tohoto důvodu již nadále škody na majetku nehlásí, a tak reálné škody mohou být daleko vyšší. Problémem je, že v případě, kdy není pachatel přistižen přímo při činu, je velmi obtížné dodatečně trestný čin prokázat a kvůli nízké závažnosti nedochází ani k nasazení operativních prostředků.

V současné době se již vyskytují různé systémy, jež sdružují informace o hlášených graffitech. Příkladem mohou být zahraniční *Graffiti Tracker*[1] a *VandalTrak*[2], případně projekt bývalých studentů fakulty *TagBust*[3]. Avšak tyto systémy z velké části spoléhají na sběr informací od veřejnosti.

Cílem této práce je najít vhodný způsob automatické detekce graffiti tagů v obraze. Tento přístup by umožnil například zpětné zpracování policejních databází a jejich navedení do výše zmíněných systémů. Bylo by ušetřeno značné množství finančních prostředků a taktéž lidských zdrojů, které mohou být využity jinak.

V kapitole 2 této práce bude čtenáři přiblížena problematika sprejerství a detekce objektů. Sekce 2.1 se věnuje graffiti subkultuře. Bude v ní objasněna základní terminologie a detailněji rozebrán speciální případ graffitů, kterým jsou tagy. V sekci 2.2 bude čtenář seznámen s teorií týkající se detekce objektů s hlavním zaměřením na konvoluční neuronové sítě. V kapitole 3 budu popisovat stávající systémy zabývající se detekcí objektů. Kapitola 4 se bude věnovat přípravě a adaptaci jednotlivých modelů pro účely detekce graffiti tagů. V 5. kapitole budou vyhodnoceny jednotlivé experimenty a vybrán nejvhodnější model. V závěrečné, 6. kapitole, shrnu provedenou práci a navrhnu možná rozšíření.

[1] http://graffititracker.net/
[2] http://www.vandaltrak.com.au/
[3] http://www.tagbust.com/

Kapitola 2

Analýza problému

V této kapitole bych rád přiblížil danou problematiku. Kapitola samotná je rozdělena do dvou logických částí. Sekce 2.1 se věnuje graffiti s hlavním zaměřením na tagy a throw-upy. Cílem této práce není zkoumat graffiti jako takové. Navíc subkultura spojená s tímto fenoménem je přísně uzavřená, a proto se mohou jednotlivé pohledy na tuto oblast lišit.

V sekci 2.2 se pak věnuji samotné detekci objektů jakožto současnému problému v oboru počítačového vidění. Kapitola je cílena především na hloubkové učení a s ním spojené konvoluční neuronové sítě, které jsou v oblasti detekce trendem posledního desetiletí. Avšak jsou zde i zmíněny příklady klasických detekčních systémů, které těmto principům předcházely.

2.1 Graffiti

Graffiti, v Česku též známé pod ekvivalentem sprejerství, je druh výtvarného vyjádření na principu nanášení barev, nejčastěji fixem nebo sprejem, na veřejná místa. Ačkoliv se v rámci boji proti graffiti snaží města vyhradit legální plochy pro tvorbu, je většina tvorby nelegální. Informace uvedené v této kapitoly byly čerpány z bakalářské práce věnující se graffiti [22].

2.1.1 Autoři

Autoři jsou dle policejních statistik nejčastěji muži v rozmezí 16-24 let. V subkultuře je pak sprejer označován jako *writer*. Je možno setkat se i s pojmem *toy*, jež označuje člověka, který s graffiti teprve začíná a je bez zkušeností. Tito lidé vystupují anonymně a užívají nejrůznější přezdívky. Tyto přezdívky se zpravidla skládají pouze z písmen a čísel, která se potom dobře píší. Příkladem takových přezdívek mohou být například LAFOR, PAUSER nebo ASH721.

V zahraničí je činnost sprejerů v nemalé míře úzce spojená s gangy. Slouží k označování kontrolovaného území či k předávání pokynům ostatním členům. V Česku se writeři taktéž shlukují do skupin, takzvaných *crew*, ale účel těchto skupin je odlišný. Na rozdíl od gangů se nepodílí na organizovaném zločinu. Příkladem takových skupin může být *crew* z Brna FET - Fatal Error Team, jejíž tag je možné vidět na obrázku 2.1b nebo celorepubliková WATT s tagem na obrázku 2.1a.

Pro policii je velmi obtížné dopadnout jednotlivé členy. Sprejování totiž probíhá často nárazově a je předem důkladně plánováno. Samotná akce, pak nezahrnuje pouze jednotlivé *writery*, ale taktéž *checkery*, kteří mají za úkol hlídat okolí a případně varovat ostatní.

(a) WATT (b) F.E.T.

Obrázek 2.1: Tagy crew

Počínání těchto skupin se často označuje jako *bombing*. Cílem je posprejovat co největší plochu. Většinou jsou proto voleny jednodušší formy graffiti, jako jsou tagy nebo tzv. throw-upy. Graffiti mají více projevů než pouze tagy a throw-upy, nicméně jejich rozbor je nad rámec této práce.

2.1.2 Tagy a throw-upy

Obecně platí, že čím menší náročnost na tvorbu graffiti, tím častěji se bude vyskytovat. Vytvoření tagu či throw-upu zabere pár desítek vteřin. S přihlédnutím k faktu, že v Česku je možné činnost gangů vyloučit, připadá na tento typ graffitů až 95% ze všech případů. Proto je potřeba se na tento typ zaměřit nejvíce.

Throw-up je zpravidla graffiti větších rozměrů jedné barvy. Často se jedná pouze o konturu prováděnou jedním tahem. Není ale výjimkou, že graffiti obsahuje výplň. Vytvoření throw-upu nezabere většinou více než 2 minuty. Námětem bývají nejčastěji přezdívky *writers* případně *crew*. Cení se zde převážně jednoduchost, rychlost a kontinuita čar. Jednotlivá díla jednoho autora se již vyznačují vizuální podobností, a proto je má smysl taktéž detekovat.

Obrázek 2.2: Throw-up AL

Tag je nejzákladnější a nejjednodušším prvkem graffiti. V reálném životě by se dal nejvíce připodobnit k podpisu. Taggováním o sobě dává *writer* vědět okolí. Tagy se vyskytují povětšinou samostatně, případně mohou být společně s větším dílem (*piece*), kde slouží jako podpis autora.

Dle **U.S. Department of Justice Office of Community Oriented Policing Services**[1] lze tagy rozdělit do čtyř kategorií. Jednotlivé příklady je možno vidět na obrázcích 2.3.

- **Crew tagy** - Tyto nelze přisuzovat jednotlivci, jelikož je může psát kterýkoliv člen *crew*. Často se skládají z dvou až čtyř písmen a jsou dobře čitelná.
- **Tagy writera** - Jedná se o osobitý podpis autora. Každý *writer* má svůj jedinečný tag, který by si nikdo neměl dovolit kopírovat. Některé tagy může být velmi obtížné přečíst nebo dokonce určit jejich orientaci.
- **Konvenční** - Nejčastěji jde o ojedinělý, či spontánní akt jedince. Může se jednat například o vyznání lásky.
- **Ideologické** - Do poslední skupiny pak spadají politické či nenávistné graffiti. Obsahem bývá vyjádření ideologického postoje. Často se zde však vyskytují rasistické, náboženské či etnické urážky.

(a) Crew tag (b) Tag writera (c) Vyznání lásky (d) Zóna antifa

Obrázek 2.3: Jednotlivé druhy tagů

2.2 Možné přístupy při detekci objektů

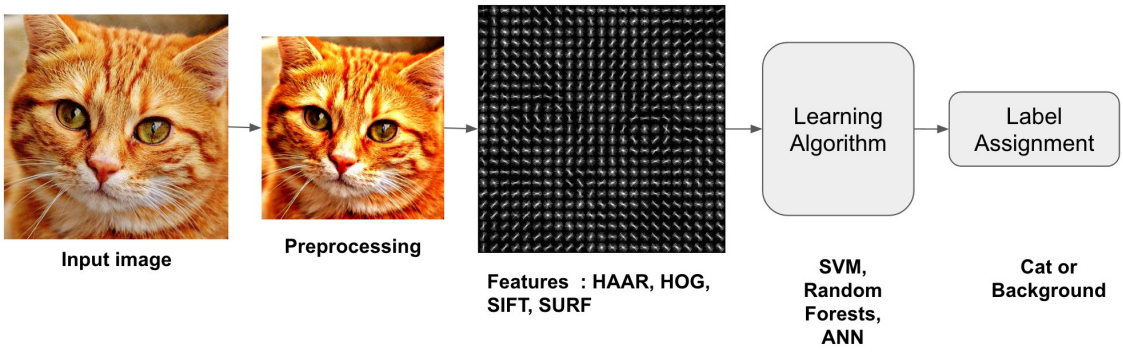
Detekce objektů patří mezi zásadní problémy v oblasti počítačového vidění a zpracování obrazu. Člověk bez větších problémů rozezná velké množství objektů, a to i přes fakt, že se mohou lišit ve velikostech, měřítku nebo mohou být různě natočené či deformované. Není problém ani rozpoznat objekt, který je zastíněn jiným. Ovšem tento úkol zůstává stále výzvou pro počítačové vidění. Oblast detekce objektů je poměrně novou a rychle se vyvíjející oblastí. V následující kapitole si rozebereme základní možné přístupy pro detekci objektů, jež se užívají posledních 15 let.

Obecně se detekční systémy skládají z dílčích částí. Prvním krokem bývá předzpracování obrazu. Předzpracování může zahrnovat mnoho procesů a závisí na problematice, které z nich je vhodné uplatnit. Cílem bývá například potlačení šumu, odstranění zkreslení nebo potlačení či zvýraznění rysů objektu. K dosažení se používá normalizace kontrastu a jasu. Další metody mohou zahrnovat gamma korekci nebo korekci barev. Při tvorbě detekčních systému je často obtížné rozhodnout jakou formu předzpracování užít, a tak je třeba se uchýlit k cestě pokusů a omylů. Součástí předzpracování je také případné ořezání nebo změna velikosti vstupního obrazu, která je velmi klíčová pro další část, kterou je extrakce příznaků.

K extrakci příznaků lze použít mnoho vhodných algoritmů. Příkladem takových algoritmů jsou SIFT [11], histogram orientovaných gradientů (HOG) [5] a v neposlední řadě

[1] http://www.popcenter.org/problems/pdfs/Graffiti.pdf

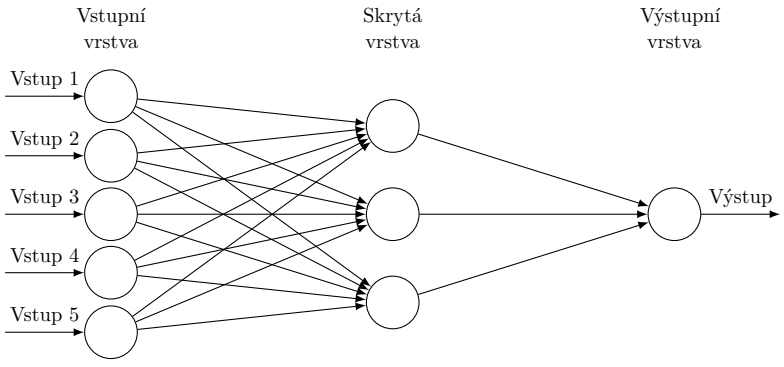
SURF [3]. Cílem je získat lokální příznaky, které potom dále poskytují zjištěné informace vhodnému algoritmu, který se naučí jednotlivé příznaky zpracovávat a klasifikovat či detekovat. Z učících algoritmů se nejčastěji užívá *support vector machines* (SVM) nebo *náhodný les*. Posledním krokem je pak klasifikace objektu a jeho pozice v obraze. Obecné schéma je možné vidět na obrázku 2.4.



Obrázek 2.4: Obecné schéma klasických detekčních systémů. Vstupní obraz je předzpracován pro následnou extrakci příznaků. Příznaky získané vhodným algoritmem jsou dále postoupeny učícímu algoritmu. V poslední fázi je vyhodnocen typ a pozice objektu v původním obraze. Převzato z [1].

2.2.1 Neuronové sítě a hloubkové učení

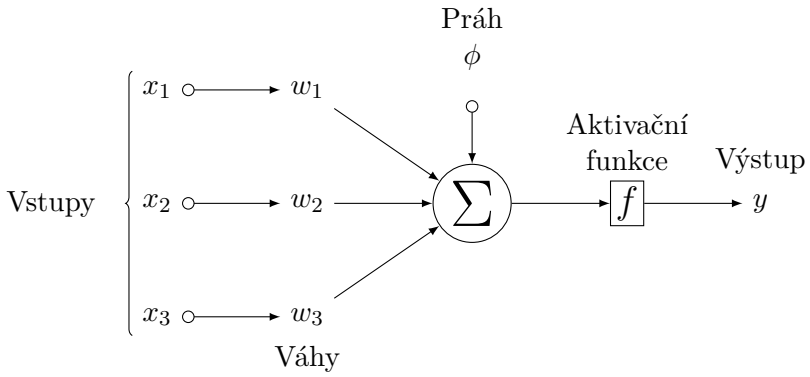
Neuronové sítě patří mezi modely inspirovány biologickými neuronovými sítěmi. Jednotlivé modely se skládají z několika vrstev, jak je možno vidět na obrázku 2.5, které se skládají z určitého počtu neuronů. Minimální počet vrstev jsou dvě, jež se označují jako vstupní a výstupní. Obvykle se však síť skládá z dalších skrytých vrstev, aby bylo možno síť naučit zvládat složitější úkoly jako například detekci objektů. Informace v této sekci byly čerpány z bakalářské práce [17].



Obrázek 2.5: Obecný model neuronové sítě

Všechny neurony jedné vrstvy jsou napojeny na neurony vrstvy následující. Vzniká tak rozsáhlá síť obsahující velké množství parametrů. Všechna jednotlivá spojení mají váhu,

která je postupně upravována během učící fáze. Neurony jsou poté aktivovány v případě, kdy součet vstupních signálů přesáhne určitou hranici a aktivační funkce spustí výstup. Ukázku neuronu je možné vidět na obrázku 2.6.



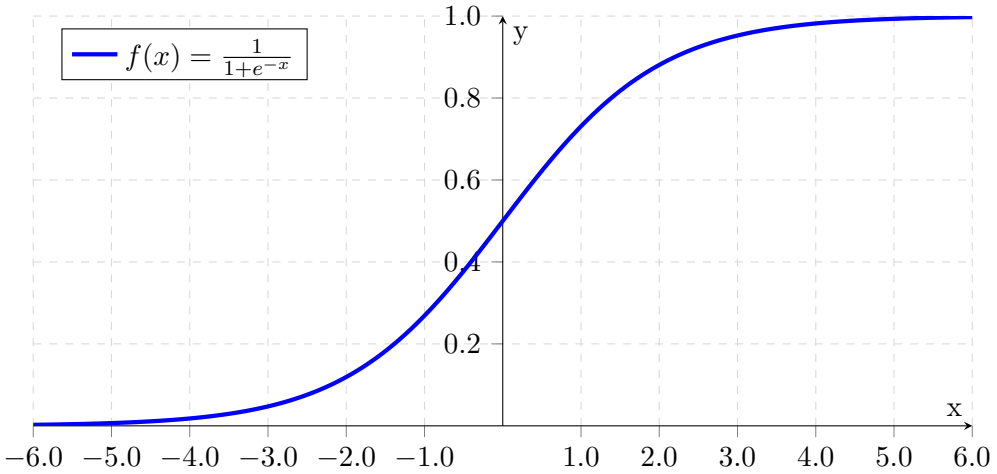
Obrázek 2.6: Model umělého neuronu

Výstup jednoho neuronu lze vypočítat jako:

$$y_k = \varphi\left(\sum_{i=1}^{N}(w_i x_i) + \phi\right) \tag{2.1}$$

kde y_k je výstup neuronu, φ aktivační funkce, x_i vstup a w_i jeho váha. Čím vyšší je hodnota váhy, tím důležitější je vstup. Součet prahu ϕ, který nabývá záporných hodnot, a součtu všech vstupů vynásobených jejich vahami se stává argumentem přenosové funkce.

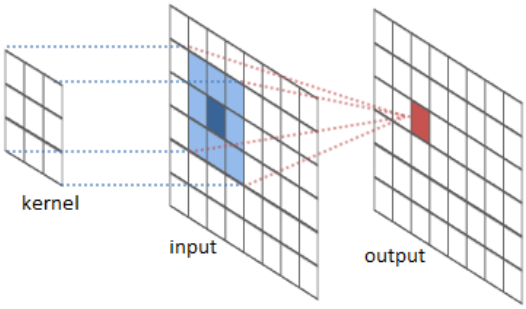
Přenosových funkcí je užíváno více druhů. Jedna z nich je sigmodiální přenosová funkce, kterou můžeme vidět na grafu 2.7.



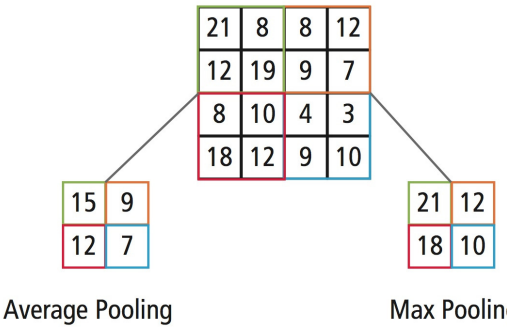
Obrázek 2.7: Průběh sigmodiální přenosové funkce.

Obecný princip neuronových sítí jsme nastínili. V oblasti grafiky se uplatňují především (hloubkové) konvoluční neuronové sítě (CNN) [19]. Hloubkové značí, že síť bude obsahovat alespoň jednu skrytou vrstvu. Tyto skryté vrstvy jsou pak konvoluční nebo pooling vrstvy. Ukázku principu konvolučních vrstev je možné vidět na obrázku 2.8a. Na obrázku 2.8b

lze vidět max a average pooling. Základní principy konvolučních neuronových sítí jsou inspirovány části lidského mozku, která má za úkol zpracovávat vizuální vjemy.



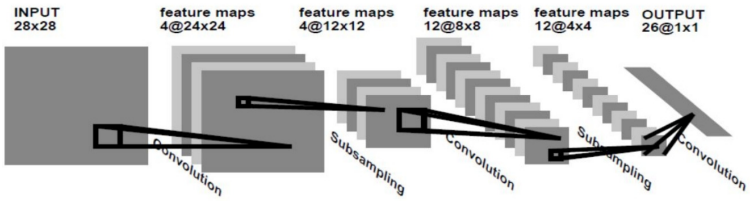
(a) Ukázka principu konvoluční vrstvy. Na vstupní data je aplikována konvoluční maska. Jednotlivé pixely jsou vynásobeny příslušným koeficientem v dané buňce konvoluční masky a posčítány, čímž vznikne hodnota výstupního pixelu.



(b) Ukázka poolingu. Při subsamplingu 2×2 je vstupní obraz o velikost 4×4 rozdělen na 4 matice velikosti 2×2. Při average poolingu je vypočítán průměr jednotlivých pixelů v 2×2 matici. U max-poolingu je pouze vybrána největší hodnota a zbylé se zahodí. Převzato z [8].

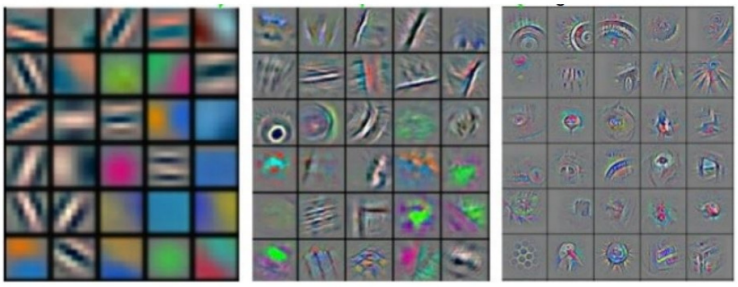
Obrázek 2.8: Ukázka konvoluce a poolingu.

Architekturu konkrétní sítě je možné vidět na obrázku 2.9. Vstupní obrázky s velikostí 28×28 pixelů jsou zpracovány konvolučním filtrem, který získává 3D mapy příznaků. Následují pak tzv. *pooling* vrstvy, kde dochází k zmenšení objemu dat. Stejná procedura se opakuje tak dlouho, dokud nedojde k vytvoření jednorozměrného vektoru, který obsahuje reprezentaci jednotlivých tříd, které v obraze zjistil.



Obrázek 2.9: Příklad sítě užívající skryté konvoluční a pooling vrstvy. Vstupní obrázek o velikosti 28×28 pixelů prochází přes tři konvoluční a dvě pooling vrstvy pro dosažení výsledného jednorozměrného vektoru.

Jednotlivé extrakce příznaků, ke kterým dochází během konvoluce, by se daly rozdělit do tří kategorií. Nízko-úrovňové příznaky, kde spadají barvy, čáry nebo kontrast. Do střední úrovně by se dalo zařadit rozeznání hran a rohů. Poslední úroveň příznaků už zahrnuje rozpoznání jednotlivých tříd či oblastí. Mezivýsledky jednotlivých kategorií je možno vidět na obrázku 2.10 níže.



Obrázek 2.10: Mezivýsledky skrytých vrstev od nejnižších příznaků po nejvyšší. Převzato z [21].

Podobně jako velké množství detekčních algoritmů i konvoluční nerunové sítě potřebují natrénovat jednotlivé váhy všech neuronů. Cílem algoritmu učení je nastavení vah w, aby se výstup sítě, co nejvíce přibližoval požadovanému výstupu z trénovací množiny. Tato skutečnost je charakterizována chybovou funkcí ERR. Ta pracuje s výstupy sítě, které porovnává s výstupy požadovanými. Cílem učení je tuto chybovou funkci minimalizovat.

Je k tomu užíván například algoritmus zpětného šíření chyby. Jedná se o jeden ze základních algoritmů pro učení vícevrstvých sítí. Tato gradientní metoda iterativně adaptuje jednotlivé váhy na základě parciální derivace chybové funkce viz rovnice 2.2.

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \frac{\partial ERR}{\partial w_{ij}(t)} \tag{2.2}$$

Parametr α zde představuje učící krok a t příslušnou iteraci.

Kapitola 3

Současné detekční sítě

V mé práci jsem se rozhodl využít nejaktuálnějších modelů pro detekci objektů. Při jejich výběru byl kladen důraz nejen na přesnost, nýbrž také na rychlost. Ve všech případech se jedná o neuronové sítě. Vzhledem k výpočetní náročnosti při tréninku vlastních modelů jsem se uchýlil pouze k doladění stávajících modelů (finetuning). Jednotlivé modely v následující kapitole přiblížím a popíši jejich klíčové vlastnosti.

3.1 Faster R-CNN

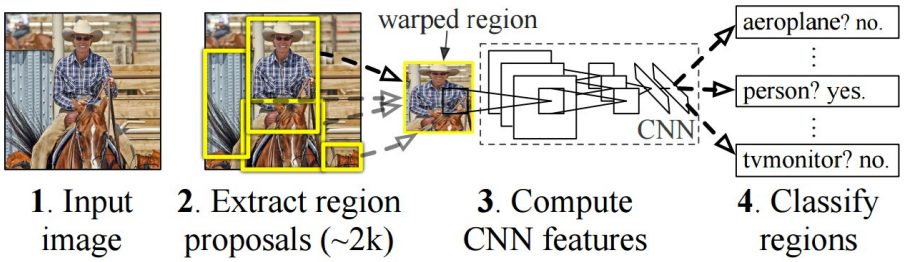
R-CNN model byl pravděpodobně první, který ukázal, že konvoluční sítě mohou vést k daleko lepším výsledkům než předchozí detekční systémy využívající HOG příznaky. Model byl představen v roce 2014 na CVPR konferenci [7] a poprvé představil využití neuronových konvolučních sítí v detekci.

V první fázi R-CNN navrhne oblasti, kde by se mohl objekt vyskytovat. K nalezení těchto oblastí využívá selektivní vyhledávání [16]. Při selektivním vyhledávání je postupně procházen obraz okny různé velikosti. Pro každé okno se pak snaží spojit pixely se stejnou texturou, barvou nebo intenzitou, která by mohla značit jeden objekt. Ukázku je možné vidět na obrázku 3.1.

(a) Různé objekty (b) Stejné objekty různých velikostí

Obrázek 3.1: Dva příklady názorně ukazují důležitost různých velikostí oblastí. Nalevo je možné vidět mnoho objektů různých velikostí. Napravo pak osobu ořezanou televizí. Převzato z [16].

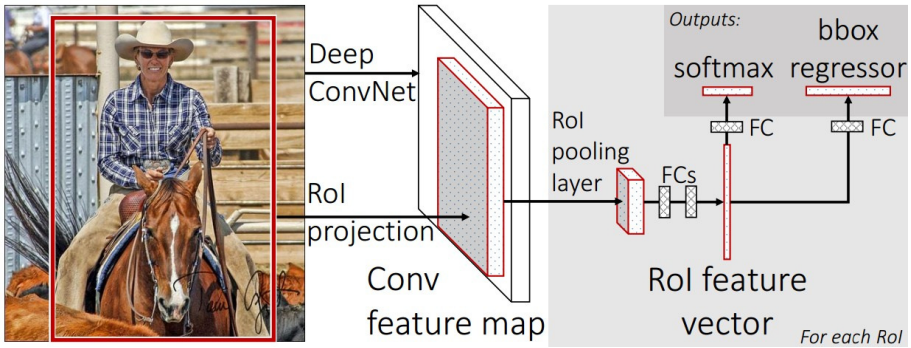
Tyto oblasti jsou pak upraveny na čtverec a stanou se vstupem do modifikované sítě AlexNet. Po konvolučních výpočtech jsou data poslána do SVM, kde je vyhodnoceno, zda se jedná skutečně o objekt a případně jaký. Celý proces je možné vidět na obrázku 3.2 níže.



Obrázek 3.2: Schéma R-CNN modelu. Pro vstupní obrázek (1) je navrhnuto přibližně 2000 odhadovaných oblastí (2). Konvoluční síť (3) potom získá příznaky, které nakonec vyhodnotí SVM (4). Převzato z [7].

V roce 2015 byl tento model vylepšen a příchodem **Fast R-CNN** [6]. Hlavním cílem bylo zrychlení detekce. Jelikož se v předchozí verzi vytvářelo přibližně 2000 oblastí, které byly dále zpracovány konvoluční sítí, docházelo k častému překrývání oblastí, a tudíž opakovaným výpočtům nad stejnou oblastí. Z toho důvodu byla do sítě zařazena technika *Region of Interest Pooling*. Tato technika sdílí výstupy konvoluční sítě a zpracovává pouze ty, které odpovídají navrženým oblastem.

Další novinkou je zjednodušení celého modelu. Namísto jednotlivých modulů, které se staraly o extrakci příznaků (AlexNet), klasifikaci (SVM) a upřesnění ohraničení (regresor), využívá jednu síť starající se o všechny tři operace. SVM klasifikátor byl nahrazen *softmax vrstvou*. Souběžně s ní byla přidána vrstva využívající lineární regresi k vytvoření ohraničení. Schéma modelu můžeme vidět na obrázku 3.3.

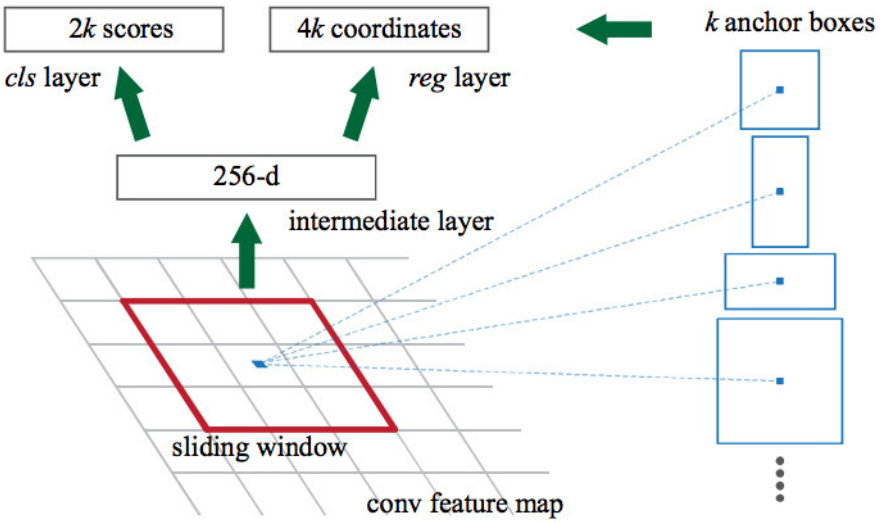


Obrázek 3.3: Schéma Fast R-CNN modelu. Vstupní obraz společně s oblastmi zájmu jsou vstupem do plně konvoluční sítě. Každá oblast zájmu je zpracována do mapy příznaků, která je dále namapována na vektor příznaků plně propojenými vrstvami. Síť má dva výstupní vektory pro každou oblast, a to pravděpodobnost výskytu objektu a posun odpovídajícího ohraničení oblasti. Převzato z [6].

Nejaktuálnějším modelem z rodiny R-CNN je však **Faster R-CNN** [15], který byl představen v roce 2016. Model nahrazuje selektivní vyhledávání, které bylo úzkým hrdlem celého procesu. Namísto jednotlivých modulů je tvořen pouze jednou hlubokou sítí. Mapa příznaků tvořená konvoluční sítí může být totiž použita k predikování oblastí.

Bylo toho dosaženo přidání plně propojené vrstvy za konvoluční síť, čímž byla vytvořena RPN (Region Proposal Network). Tato síť funguje za pomoci posuvného okna, které pracuje nad mapou příznaků. Nad každým oknem pak tvoří k potencionálních ohraničení obsahující hodnotu, která reprezentuje odhadovanou kvalitu ohraničení.

Je pochopitelné, že různé objekty mají různé velikosti a poměry. Například pro detekci chodců můžeme očekávat obdélníkové ohraničení, které odpovídá poměrům lidské postavy. Stejně tak můžeme očekávat jisté poměry a velikosti u ostatních objektů. Tyto poměry jsou předem zadány jako takzvané *anchor boxes*. Pro každý anchor box jsou pak generovány souřadnice ohraničení a hodnotu značící pozici v obraze. Princip RPN lze vidět na obrázku 3.4 níže.



Obrázek 3.4: Princip Region Proposal Network. RPN za použití posuvného okna velikosti 3×3 prochází mapu příznaků. Okno pro každou oblast je převzorkováno do 256 rozměrného vektoru, který je dále postoupen do dvou plně propojených vrstev. První určující pravděpodobnost výskytu objektu v dané oblasti a druhá posun ohraničení oproti anchor boxes. Pro každou pozici je tak generováno $4k$ souřadnic a $2k$ pravděpodobností výskytu. Pouze oblasti, které vyhovují požadavků jsou postoupeny dále. Převzato z [15].

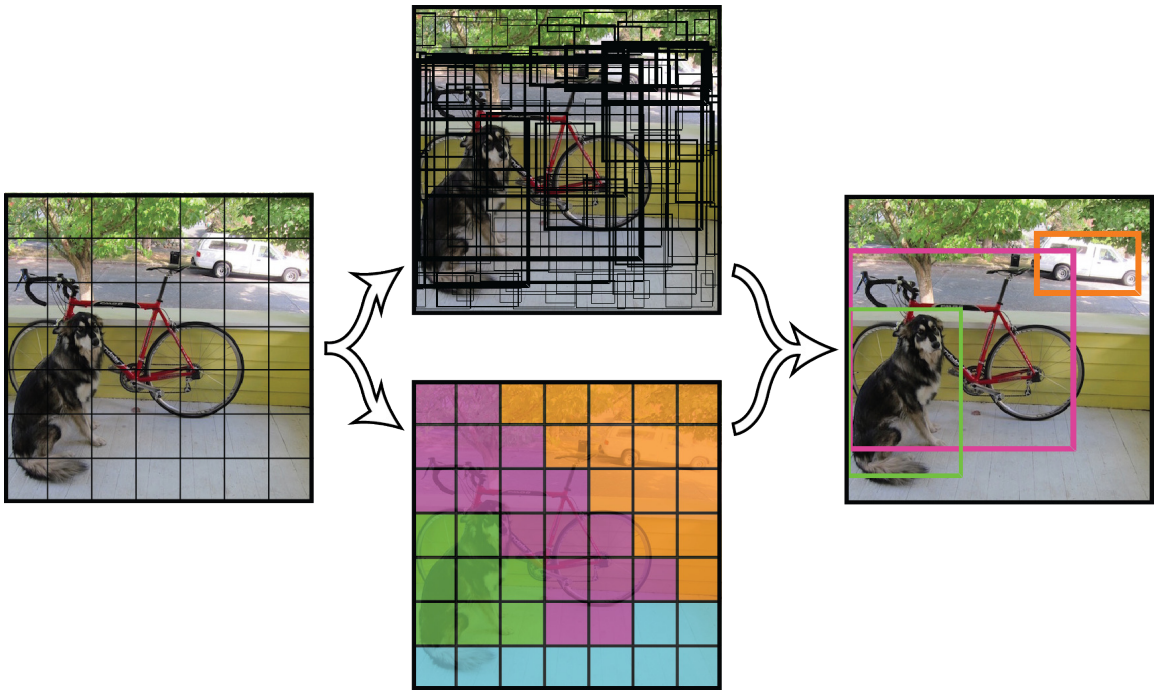
3.2 Darknet

Darknet [12] je open-source framework pro práci s neuronovými sítěmi. Je napsán v jazyce C. Vyniká převážně svou rychlostí a jednoduchou instalací. Podporuje, jak práci s CPU, tak i GPU s využitím CUDA jader.

3.2.1 YOLO

You Only Look Once [13], zkráceně YOLO model byl představen v roce 2015. Namísto řešení detekce po jednotlivých částech, je celý detekční systém sjednocen do jedné neuronové sítě. Pro predikci jednotlivých ohraničení detekovaných objektů využívá příznaky z celého obrázku. Všechny objekty navíc zpracovává zároveň, a tak exceluje vysokou rychlostí při zachování relativně vysoké přesnosti.

Prvním krokem je zmenšení vstupního obrázku na rozlišení 448×448. Následně je obraz rozdělen do $S \times S$ mřížky. V případě, že střed objektu spadá do jedné z těchto buněk mřížky, tak právě tato buňka zodpovídá za detekci objektu. Každá buňka poté predikuje B ohraničení detekovaného objektu a pravděpodobnostní skóre pro tato ohraničení. Toto skóre odráží pravděpodobnost, s jakou model správně určil přítomnost objektu a zároveň jak přesné toto ohraničení je. V případě, že se v dané buňce žádný objekt nenachází, hodnota by měla odpovídat nule. Ukázka principu je na obrázku 3.5.



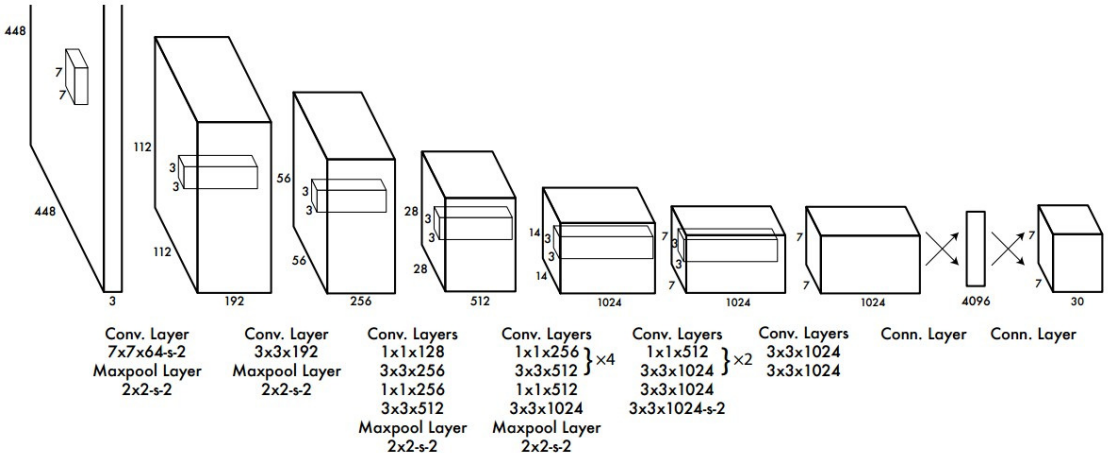
Obrázek 3.5: Model YOLO

Každé ohraničení objektu se skládá z pěti predikcí x, y, w, h a $ConfidenceScore$. (x, y) označují souřadnice středu detekovaného objektu relativně k okraji buňky. (w, h) reprezentují šířku a výšku detekovaného objektu, ta je však relativní k velikosti celého obrázku. Poslední položkou je pravděpodobnostní skóre, které určuje IOU mezi predikovaným ohraničením a jakýmkoliv anotovaným ohraničením. Jednotlivé buňky ještě navíc určují pravděpodobnost označující jednotlivé třídy. Tímto faktem se však nemusíme zabývat, jelikož máme pouze jednu třídu - tagy.

$$ConfidenceScore = P(Object) \cdot IOU_{pred}^{truth}$$

Výsledný tenzor je pak vypočítán jako $S \times S \times (B \cdot 5 + C)$. Pro detekci tagů využívám hodnoty $B = 2$, $S = 7$ a $C = 1$. Výsledné predikce jsou tudíž zakódovány do $7 \times 7 \times 11$ tenzoru.

Samotná síť pak vychází z modelu GoogLeNet pro pouhou klasifikaci objektů. Obsahuje celkem 24 konvolučních vrstev, které se starají o extrakci příznaků následované dvěma plně propojenými vrstvami, které zajišťují následnou predikci. Model využívá 1×1 redukční vrstvy následované 3×3 konvolučními. Kompletní model lze vidět na obrázku 3.6.



Obrázek 3.6: Architektura YOLO modelu

3.2.2 Tiny YOLO

Dalším modelem, který jsem se rozhodl vyzkoušet je menší síť **Tiny YOLO**. Ačkoliv nedosahuje takové přesnosti jako YOLO, je několikanásobně rychlejší. Obsahuje pouze devět konvolučních vrstev a šest maxpool vrstev, jak je možno vidět v tabulce 3.1. Do konečného výběru modelů jsem se ji rozhodl zařadit pro případné nasazení do systémů s nižší výpočetní kapacitou s možným využitím u vestavěných systémů nebo detektorů pracujících v reálném čase, například u bezpečnostních kamer.

Pořadí	Typ	Filtry	Velikost	Krok	Vstup	Výstup
1	konvoluční	16	3×3	1	416×416×3	416×416×16
2	maxpool		2×2	2	416×416×16	208×208×16
3	konvoluční	32	3×3	1	208×208×16	208×208×32
4	maxpool		2×2	2	208×208×32	104×104×32
5	konvoluční	64	3×3	1	104×104×32	104×104×64
6	maxpool		2×2	2	104×104×64	52×52×64
7	konvoluční	128	3×3	1	52×52×64	52×52×128
8	maxpool		2×2	2	52×52×128	26×26×128
9	konvoluční	256	3×3	1	26×26×128	26×26×256
10	maxpool		2×2	2	26×26×256	13×13×256
11	konvoluční	512	3×3	1	13×13×256	13×13×512
12	maxpool		2×2	1	13×13×512	13×13×512
13	konvoluční	1024	3×3	1	13×13×512	13×13×1024
14	konvoluční	1024	3×3	1	13×13×1024	13×13×1024
15	konvoluční	30	1×1	1	13×13×1024	13×13×30

Tabulka 3.1: Architektura Tiny YOLO modelu

3.2.3 YOLOv2

V roce 2016 byl představen vylepšený model YOLOv2 [14]. Oproti předchozímu modelu představeném v sekci 3.2.1 přichází s řadou vylepšení, které vylepšují rychlost, přesnost a flexibilitu. Oproti klasickému přístupu, kdy dochází za účelem zvýšení přesnosti k rozšiřování sítí, je zde síť zjednodušena, čímž je docíleno jednoduššího učení a zpracování dané datové reprezentace.

Prvním vylepšením je normalizace jednotlivých dávek při trénování jejich průměrem a odchylkou. Díky této normalizaci dochází k výraznému zlepšení konvergence při učení sítě. Umožňuje navíc odstranění *dropout* metody, která cíleně nuluje neurony s určitou pravděpodobností. Normalizace dávek totiž stejně jako *dropout* předchází *overfitting-u*. Bližší informace k algoritmu lze najít v článku [9].

Další změnou oproti YOLO modelu přichází se zvětšením velikosti vstupních dat. Zatímco většina modelů využívající klasifikátorů založených na AlexNet síti, pracují se vstupy menšími než 256×256 pixelů, operuje YOLOv2 s rozlišením 448×448. Díky tomu model pracuje daleko lépe s obrazy ve větším rozlišením.

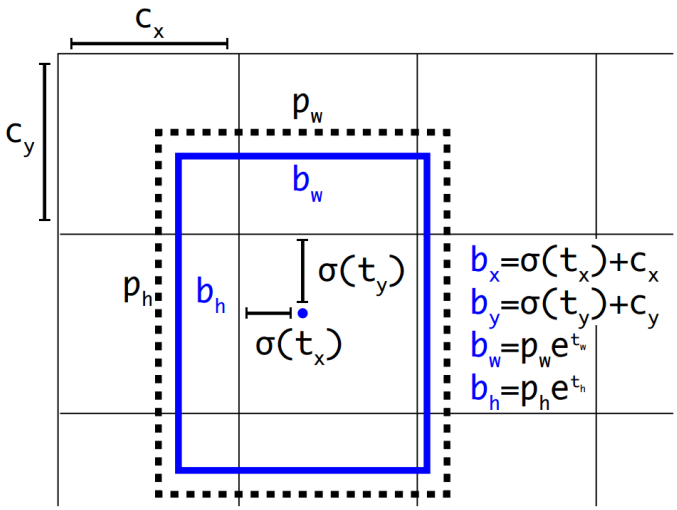
Došlo také k odstranění plně propojených vrstev. K predikci možných ohraničení dochází podobně jako u Faster R-CNN modelu za pomoci *anchor boxes*. Na výstupu je mapa příznaků o velikosti 13×13, kdy při využití *anchor boxes* dostaneme přes 1000 možných oblastí, kde se může tag vyskytovat. Byl zvolen lichý počet výstupů, protože je vysoká pravděpodobnost, že hledaný tag bude uprostřed obrazu, a tak je efektivnější vytvořit jeden střed než čtyři pro daný objekt. Jednotlivé rozměry a poměry jsou průběžně upravovány při trénování automaticky, aby co nejvíce odpovídaly reálným výskytům. Dochází tak k zvýšení přesnosti a docílení větší těsnosti výsledných ohraničení.

Síť predikuje 5 různých ohraničení pro každou buňku. Každé predikované ohraničení pak sestává z 5 souřadnic, t_x, t_y, t_w, t_h a t_o. Posun konkrétní buňky oproti levému hornímu rohu je dán hodnotami c_x a c_y. Hodnoty p_w a p_h odpovídají očekávané šířce a výšce ohraničení. Konečné predikce se vypočítají jako:

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(Object) \times IOU(b, object) &= \sigma(t_o)
\end{aligned}
$$

Díky tomu, že jsou predikce vztahovány ke konkrétní buňce a tím je jejich poloha omezena, je síť stabilnější a jednodušeji se učí. Vizuální reprezentace je na obrázku 3.7.

Poslední změnou je průběžné upravování sítě pro náhodné vstupní velikosti. Model během trénování náhodně vybírá nový rozměr po určitém počtu iterací. Jelikož je model postupně upravován násobky 32, byly pro vstupní rozměry vybrány násobky této hodnoty - 320,352, ..., 608. Nejmenší vstupní rozměr je tudíž 320×320 pixelů a největší 608×608 pixelů. Síť je přizpůsobena novému rozměru a trénování pokračuje. S vyšším vstupním rozměrem pochopitelně narůstá výpočetní náročnost, nicméně dochází k znatelnému nárostu přesnosti.

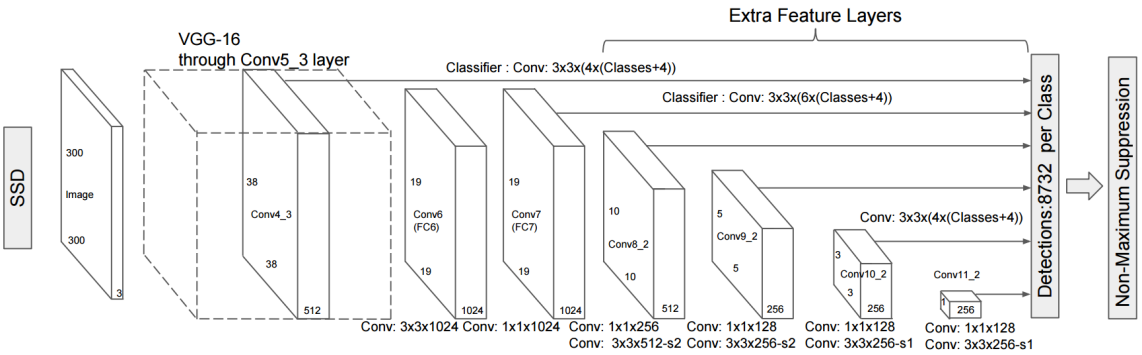


Obrázek 3.7: Predikce finálního ohraničení. Střed objektu je vypočítán relativně k levému hornímu rohu konkrétní buňky za pomocí speciálního případu logistické funkce - sigmoidy. Výška a šířka jsou dopočítány z apriorních predikcí. Obrázek převzat z článku [14].

3.3 SSD

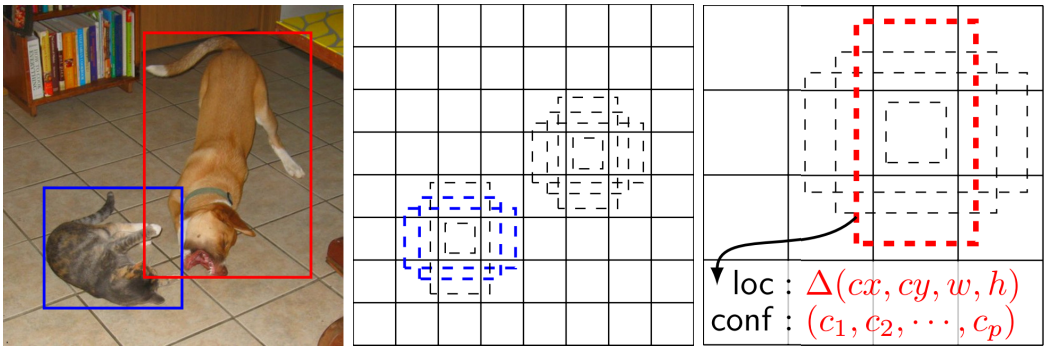
Dalším modelem, který jsem se rozhodl využít je **Single Shot MultiBox Detector** [10]. Jedná se o dopřednou neuronovou síť, která vytváří kolekci předem známých rozměrů ohraničení a jejich ohodnocení pro případnou přítomnost objektu v nich. Následuje finální krok, kdy dojde k potlačení nemaxim pro dosažení výsledných detekcí. První vrstvy modelu jsou založeny na standardní architektuře používané v současných detekčních systémech, kterou je VGG16 [18].

Následuje podpůrná síť, která se stará o finální detekci. Tato síť navazuje na předchozí část konvolučními vrstvami. Tyto vrstvy postupně snižují velikost a umožňují detekci pro různá měřítka. Modul starající se o detekci pak pracuje s různými vrstvami příznaků, čímž zajišťuje větší počet predikcí. Tímto se odlišuje od YOLO modelu, který pracuje pouze s jednou vrstvou příznaků.



Obrázek 3.8: SSD model. Modelu předchází standardní VGG16 architektura. Na konci modelu jsou přidány různé vrstvy příznaků, které umožňují predikce u různých měřítek a poměrů. Obrázek převzat z článku [10].

Každá vrstva příznaků je rozdělena do mřížky, kde je jednotlivým buňkám přiřazen soubor standardních ohraničení odpovídající danému měřítku. Tento přístup je možné vidět na obrázku 3.9. Při detekci je potom predikován posun vzhledem k původnímu ohraničení a zároveň pravděpodobnost, že se v dané oblasti hledaný objekt nachází.



(a) Anotovaný vstupní obrázek (b) Mapa příznaků 8×8 (c) Mapa příznaků 4×4

Obrázek 3.9: Princip SSD modelu. Anotovaný vstupní obrázek 3.9a je porovnán se standardními predikcemi jednotlivých map příznaků. Na každou buňku připadají 4 různé poměry a měřítka. Pro každou predikci je také odhadnut posun ohraničení a pravděpodobnost výskytu objektu v dané oblasti. Na obrázku 3.9b je zaznamenána shoda pro dvě standardní ohraničení odpovídající jednomu objektu a na obrázku 3.9c další shoda při změně měřítka. Obrázek převzat z článku [10].

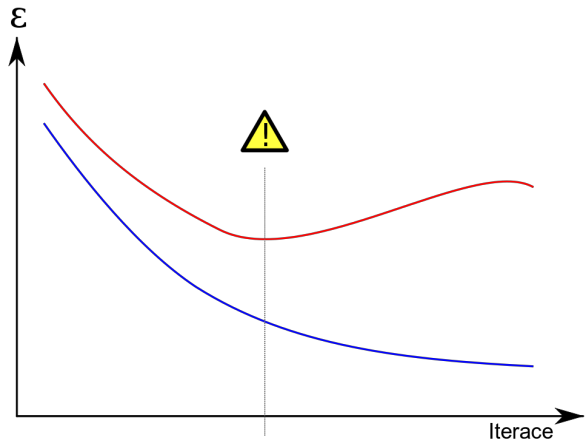
Kapitola 4

Trénování jednotlivých modelů

Trénování konvolučních neuronových sítí je výpočetně velice náročné. První experimenty byly prováděny na grafické kartě NVIDIA GeForce 840M s 4GB paměti. Cílem bylo zjistit vhodné parametry vedoucí ke konvergenci při trénování. Mezi tyto parametry patří například učící krok, jeho změny během pozdějších iterací nebo velikost učících dávek.

Následovalo finální trénování, které probíhalo na grafické kartě NVIDIA GeForce GTX 1080 disponující 8GB paměti. Vzhledem k tomu, že nebylo třeba trénovat celý model od začátku, nýbrž byly využity již předtrénované modely, které byly doučeny na cílových datech, došlo k značnému úbytku potřebných iterací.

Jistým problémem, který bylo třeba ohlídat, bylo přeučení. Z důvodu malé velikosti trénovacích dat totiž v pozdějších fázích učení dochází ke ztrátě generalizace. Výsledný model je pak specializován na trénovací data a ztrácí schopnost detekovat jiné objekty. Princip přeučení je ukázán na obrázku 4.1.



Obrázek 4.1: Modrá křivka znázorňuje trénovací chybu a červená křivka odpovídá validační chybě. Obě chyby napříč iteracemi klesají, dokud nenarazí do bodu, kdy začne validační chyba narůstat z důvodu přeučení. Je důležité proto vybrat vhodné váhy, které odpovídají minimu u validační chyby.

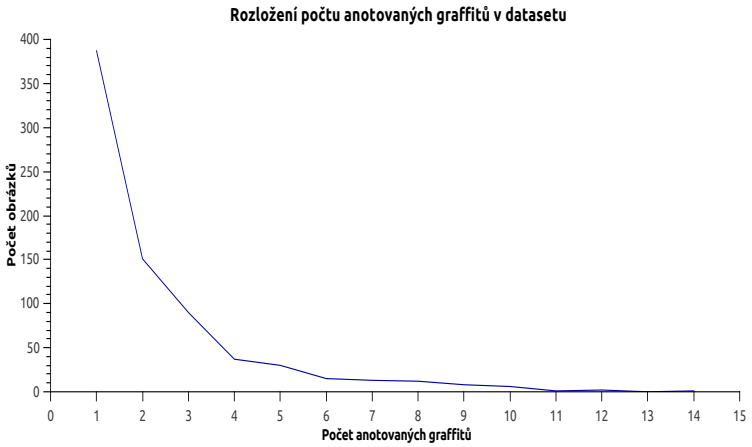
4.1 Dataset

Tvorba datasetu probíhala na území Brna viz obrázek 4.6. Při tvoření datasetu byl kladen důraz na rozmanitost jednotlivých tagů. Nezřídka totiž bývá jedna oblast pokryta převážně tagy jednoho writera. Graffiti byly foceny mobilním telefonem Sony Xperia Z1 Compact (D5503). Fotografie v datasetu jsou ve formátu JPEG. Původní fotografie byly nafoceny v rozlišení 3840×2160 pixelů, které byly pro účely trénování a testování zmenšeny na 25%, tudíž na rozlišení 960×540 pixelů.

Dataset obsahuje celkem 747 fotografií zachycujících 1696 tagů, které jsou dále rozděleny na trénovací a testovací data. Trénovací data obsahují 497 obrázků a testovací zbývajících 250. Z důvodu častého výskytu velkého množství tagů blízko u sebe, je obtížné zachytit pouze jeden tag z adekvátní vzdálenosti. Ukázka datasetu je na obrázku 4.2.

Obrázek 4.2: Ukázka datasetu.

Na jedné fotografii se tak občas vyskytuje až 14 tagů, což působí problémy při následné detekci. Zastoupení počtu graffitů v jednotlivých fotografiích lze vidět na grafu 4.3.



Obrázek 4.3: Rozložení počtu anotovaných tagů v datasetu.

Pro tvoření anotací bylo využito programu LabelImg[1]. Výstupem jsou anotace ve formátu xml užívaném v Pascal VOC datasetech. Pro trénování modelů však bylo třeba vytvořit i anotace vhodné pro Darknet a SSD, které užívají jinou formu zápisu.

V případě SSD je použit formát txt. Pozice jednotlivých ohraničujících rámečků je zapsána vždy na jednom řádku. První údaj obsahuje informaci o třídě, kterou daný rámeček označuje. Poté následují dvě dvojice celých čísel označující pozice levého horního a pravého dolního rohu rámečku. Ukázku můžeme vidět na obrázku 4.4.

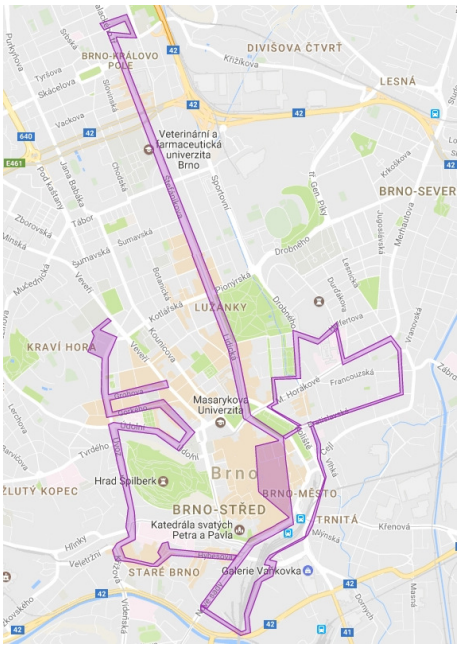
```
class_num x_min y_min x_max y_max
1         324   219   761   483
```

Obrázek 4.4: Ukázka zápisu anotací pro SSD.

Darknet přistupuje k problematice odlišně. Namísto pevně daných souřadnic využívá poměrů. Stejně jako u předchozího zápisu se první informace vztahuje k třídě, do které daný objekt spadá. Další informace jsou pak zapsány jako čísla s plovoucí čárkou. První dvojice označuje pozici levého horního rohu. Hodnota je zjištěna jako poměr souřadnice x k šířce obrázku a souřadnice y k výšce obrazu. Následující dvojice označuje velikost rámečku a vypočítá se jako šířka rámečku k šířce obrázku a výška rámečku k výšce obrázku. Tento přístup má nesmírnou výhodu, protože v případě změny rozlišení obrázku zůstává anotace stále platnou. Ukázku zápisu je možno vidět na obrázku 4.5.

```
class_num box_x_ratio box_y_ratio box_width_ratio box_height_ratio
0         0.744791667 0.441666667 0.0666666666667 0.387037037037
```

Obrázek 4.5: Ukázka zápisu anotací pro Darknet.



Obrázek 4.6: Mapa oblastí tagů v datasetu.

[1] https://github.com/tzutalin/labelImg

4.2 Trénování YOLOv2

Pro trénování modelu YOLOv2 byl zvolen učící krok 0,001, který byl po 100 počátečních iteracích navýšen na 0,01. Důvodem bylo v prvotních iteracích správně nastavit nové váhy. Malým učícím krokem jsem se vyhnul případné divergenci modelu. Po 5000 iteracích došlo k opětovnému snížení na 0,001, aby se model neuchýlil k lokálnímu minimu namísto globálního. Velikost trénovací dávky byla nastavena na 64.

Jak již bylo zmíněno, YOLOv2 využívá při trénování vstupní data různé velikosti. Při trénování byl pravidelně každých 640 iterací náhodně vybrán nový rozměr vstupních dat. Celkově bylo provedeno 19000 iterací.

Při trénování se používá kombinovaná loss funkce, která se skládá z chyby podmíněné pravděpodobnosti:

$$Loss_p = \sum_{i}^{S^2} \sum_{c \in classes} \mathbb{1}_i^{responsible\_obj} \{p_i^{pred}(c) - p_i^{truth}(c)\}^2 \tag{4.1}$$

Kde $\mathbb{1}_i^{responsible\_obj}$ značí, zda se objekt vyskytuje v buňce i. S udává rozměr mřížky a B počet predikovaných ohraničení pro jednotlivé buňky. Dále z chyby pro *anchor box*:

$$\begin{aligned} Loss_{box} = \lambda_{obj}^{cord} \sum_{i}^{S^2} \sum_{j}^{B} \mathbb{1}_{ij}^{responsible\_obj} \Big[& (x_{ij}^{pred} - x_{ij}^{obj})^2 + (y_{ij}^{pred} - y_{ij}^{obj})^2 \\ & + (w_{ij}^{pred} - w_{ij}^{obj})^2 + (h_{ij}^{pred} - h_{ij}^{obj})^2 \Big] \\ + \lambda_{noobj}^{cord} \sum_{i}^{S^2} \sum_{j}^{B} \mathbb{1}_{ij}^{no\_responsible\_obj} \Big[& (x_{ij}^{pred} - x_{ij}^{anchor\_center})^2 + (y_{ij}^{pred} - y_{ij}^{anchor\_center})^2 \\ & + (w_{ij}^{pred} - w_{ij}^{anchor\_default})^2 + (h_{ij}^{pred} - h_{ij}^{anchor\_default})^2 \Big] \end{aligned} \tag{4.2}$$

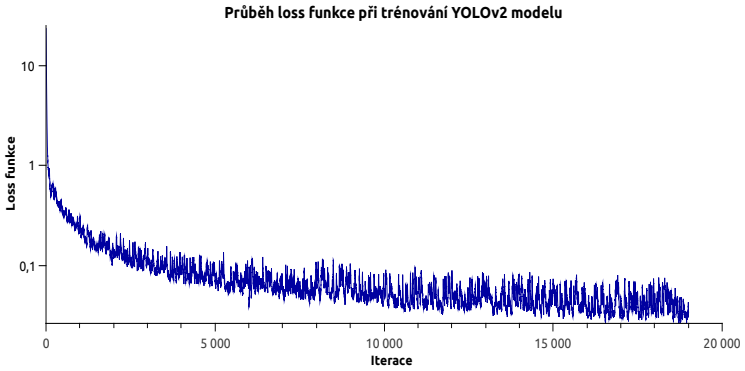
Kde $\mathbb{1}_{ij}^{responsible\_obj}$ značí j-tý prediktor ohraničení pro konkrétní buňku i. $x_{ij}^{anchor\_center}$ a $y_{ij}^{anchor\_center}$ označující původní pozici středu buňky a odpovídají hodnotě 0,5. Proměnné $h_{ij}^{anchor\_default}$ společně s $w_{ij}^{anchor\_default}$ nabývají hodnoty 1. λ_{obj}^{cord} a λ_{noobj}^{cord} přisuzují větší význam buňkám, které obsahují požadovaný objekt. Pro buňky zodpovídající za ohraničení, kde se objekt nachází je hodnota 1, pro ostatní pak 0,1. Poslední součástí je výpočet chyby pro *confidence score*:

$$\begin{aligned} Loss_{conf} = \lambda_{obj}^{conf} \sum_{i}^{S^2} \sum_{j}^{B} \mathbb{1}_{ij}^{responsible\_obj} & \{conf_{ij}^{pred} - iou(box_{ij}^{pred}, box_{ij}^{truth})\}^2 \\ & + \lambda_{noobj}^{conf} \sum_{i}^{S^2} \sum_{j}^{B} \mathbb{1}_{ij}^{no\_responsible\_obj} \{conf_{ij}^{pred})\}^2 \end{aligned} \tag{4.3}$$

Kompletní loss funkce odpovídá součtu těchto tří složek:

$$Loss = Loss_{conf} + Loss_{box} + Loss_p \tag{4.4}$$

Průběh kompletní loss funkce při trénování je možné vidět na grafu 4.7.



Obrázek 4.7: Průběh loss funkce při trénování YOLOv2 modelu.

4.3 Trénování SSD

Při trénování SSD modelu byl vybrán model SSD300. Učící krok byl zvolen stejně jako u YOLOv2 0,001. Nebyl však upravován s postupnými iteracemi, protože při vyšších hodnotách začal ihned divergovat, a naopak při nižších hodnotách docházelo k minimálnímu snižovaní loss funkce. Velikost jedné trénovací dávky byla nastavena na hodnotu čtyři. Těchto dávek je však zpracováno celkově osm předtím, než dojde k úpravě gradientu. Toto nastavení vedlo jako jediné ke konvergování modelu při limitované velikosti paměti grafické karty. Bylo provedeno celkem 20000 iterací a loss funkce se ustálila kolem hodnoty 1.

Loss funkce se obdobně jako u YOLOv2 modelu skládá z více složek:

$$Loss(x, c, l, g) = \frac{1}{N}(Loss_{conf}(x, c) + \alpha Loss_{loc}(x, l, g)) \tag{4.5}$$

Kde $x_{ij}^c = \{0, 1\}$ a značí úspěšnou detekci i-tého, který nabývá hodnotu IOU větší než 0,5 pro anotovanou oblast j třídy c. Pro jednu buňku může být označeno více predikovaných ohraničení jako správných. N pak odpovídá počtu oblastí u kterých byla vyhodnocena shoda. V případě, kdy není detekována žádná shoda je $Loss$ nastavena na 0. Chyba detekované oblasti je vypočítána jako Smooth L1 loss, definována v článku [6], mezi predikovaným ohraničením (l) a anotovaným (g). Predikované ohraničení je vypočítáno oproti standardním ohraničením:

$$Loss_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in cx,cy,w,h} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{4.6}$$

$$\hat{g}_j^{cx} = \frac{(g_j^{cx} - d_i^{cx})}{d_i^w} \tag{4.7}$$

$$\hat{g}_j^{cy} = \frac{(g_j^{cy} - d_i^{cy})}{d_i^h} \tag{4.8}$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \tag{4.9}$$

$$\hat{g}_j^h = \log(\frac{g_j^h}{d_i^h}) \tag{4.10}$$

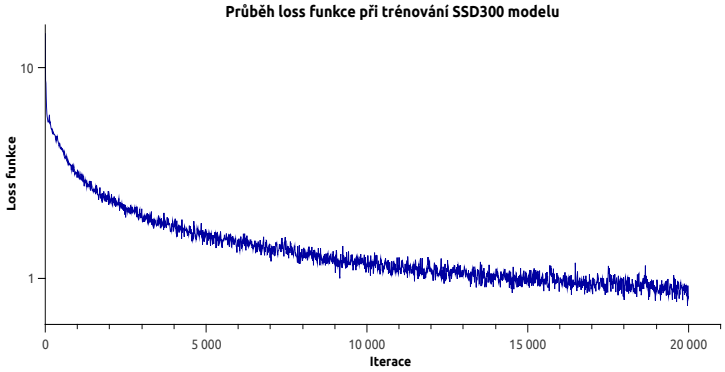
Confidence loss je vypočítána jako softmax loss:

$$Loss_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} \log(\hat{c}_i^p) - \sum_{i \in Neg}^{N} x_{ij}^{p} \log(\hat{c}_i^0) \tag{4.11}$$

kde:

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{4.12}$$

Průběh celé loss funkce lze vidět na grafu 4.8.



Obrázek 4.8: Průběh loss funkce při trénování SSD300 modelu.

4.4 Trénování Faster R-CNN

Poslední z trénovaných modelů je Faster R-CNN. Byl u něj zvolen podobný přístup jako u YOLOv2, kdy byl počáteční učící krok nastaven na 0,001, ale byl po 1000 iteracích zvýšen na 0,01. U 5000 iterace byl opět snížen na původní učící krok, který zůstal až do 20000 iterace. Pro trénování byla zvolena end-to-end verze.

Loss funkce je v tomto případě:

$$Loss(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{4.13}$$

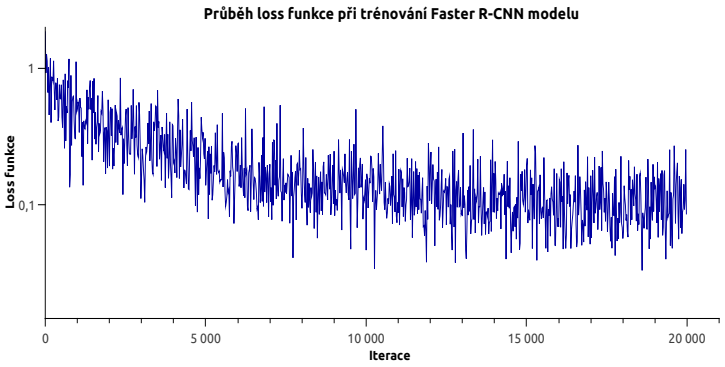
V tomto případě je i index příslušného standardního ohraničení a p_i odhadovaná pravděpodobnost výskytu hledaného objektu v této oblasti. p_i^* nabývá hodnoty 1 v případě pozitivní detekce a 0 v případě negativní. Všechny čtyři souřadnice predikovaného ohraničení jsou reprezentovány jako vektor t_i. Vektor t_i^* obsahuje informaci o souřadnicích anotovaného ohraničení, které patří k příslušné oblasti.

Pro určení chyby detekované oblasti užívá stejně jako SSD Smooth L1 loss, která byla ukázána v rovnici 4.6. Pro klasifikační část je pro výpočet použita logaritmická loss funkce:

$$logloss = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(p_{ij}) \tag{4.14}$$

Kde N je počet detekcí, M počet tříd a y_{ij} nabývá hodnot 1 v případě, kdy je detekce i pro třídu j a 0 v opačném případě. p_{ij} značí odhadovanou pravděpodobnost, že detekce i odpovídá třídě j.

Průběh testování lze vidět na grafu 4.9 níže.

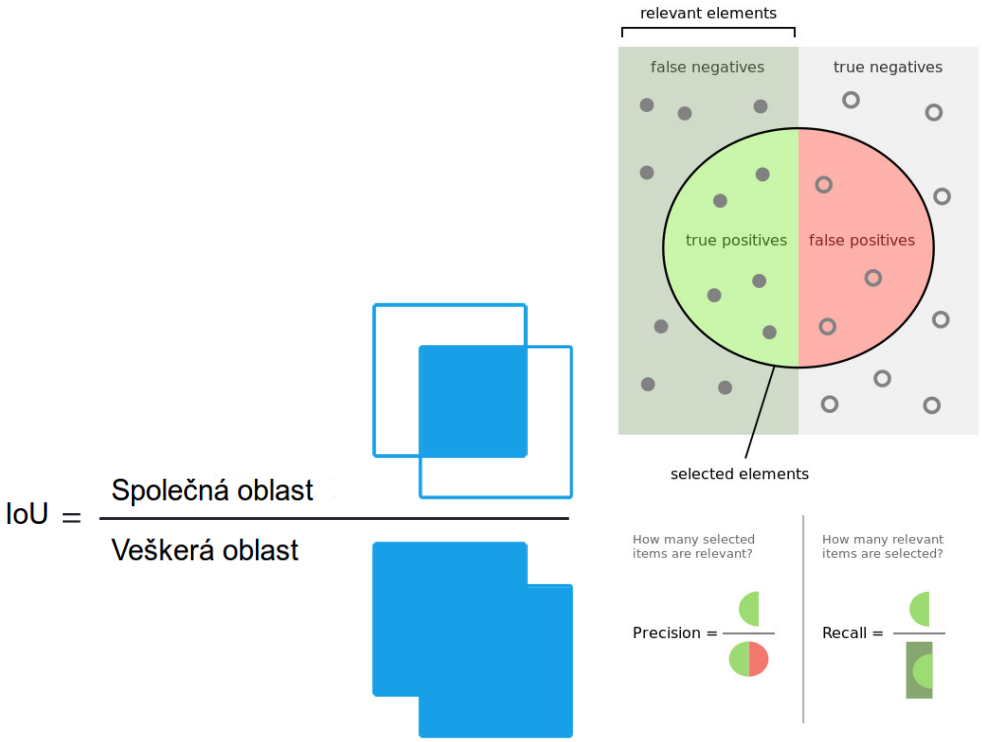


Obrázek 4.9: Průběh loss funkce při trénování Faster R-CNN modelu.

Kapitola 5

Výsledky experimentů

Pro výsledné porovnání přesnosti modelů byla zvolena *average precision* užívána při VOC soutěži [4]. *Average precision* je vypočítána z *precision/recall* křivky. *Precision* je vypočítána jako podíl pravdivě pozitivních a všech pozitivních detekcí. *Recall* je pak vypočítán jako podíl pravdivě pozitivních a pravdivě pozitivních společně s nepravdivě pozitivními. Pro názornost je výpočet znázorněn na obrázku 5.1b. Pravdivě pozitivní detekce jsou ty, které vykazují IoU, ukázané na obrázku 5.1a, větší než 0,5.



(a) IoU je vypočítána vydělením průniku anotované a detekované oblasti jejich sjednocením.

(b) Ukázka precision a recall. Převzato z [20].

Obrázek 5.1: Princip výpočtu IoU a precision s recall.

Ze zmíněné křivky je vybráno 11 od sebe stejně vzdálených hodnot odpovídajícím příslušným *recall* hodnotám [0;0,1;...;1]:

$$AP = \frac{1}{11} \sum_{r \in 0;0,1;...;1} P_{interp}(r)$$

kde je *precision* odpovídající dané *recall* hodnotě interpolována tak, že je vzata nejvyšší *precision* naměřená pro hodnotu, která překročí r:

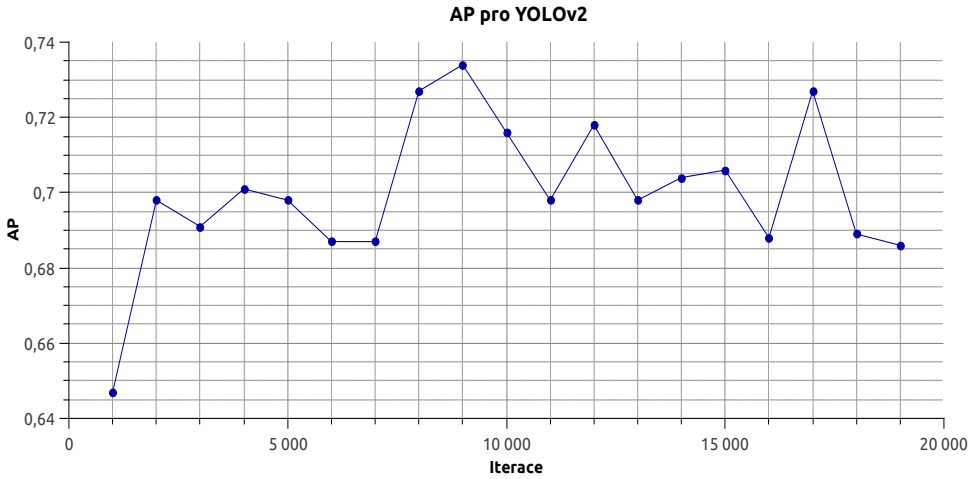
$$P_{interp}(r) = \max_{\tilde{r}:\tilde{r}>r} p(\tilde{r})$$

kde $p(\tilde{r})$ je naměřená *precision* odpovídající *recall* hodnotě $\tilde{r}$.

5.1 Vyhodnocení jednotlivých modelů

Při trénování byly průběžně ukládány váhy každých 1000 iterací. Pro každý model byla posléze vyhodnocena *average precision* pro každou z těchto vah, aby byla vybrána ta, která nejlépe detekuje na testovacích datech. Data byla vyhodnocována na grafické kartě NVIDIA GeForce GTX 1080 disponující 8GB paměti. Na spuštění jsou však dostačující 4GB paměti v případě užití cuDNN. Pro ohodnocení jednotlivý vah byly použity skripty dostupné v balíčku VOCdevkit[1].

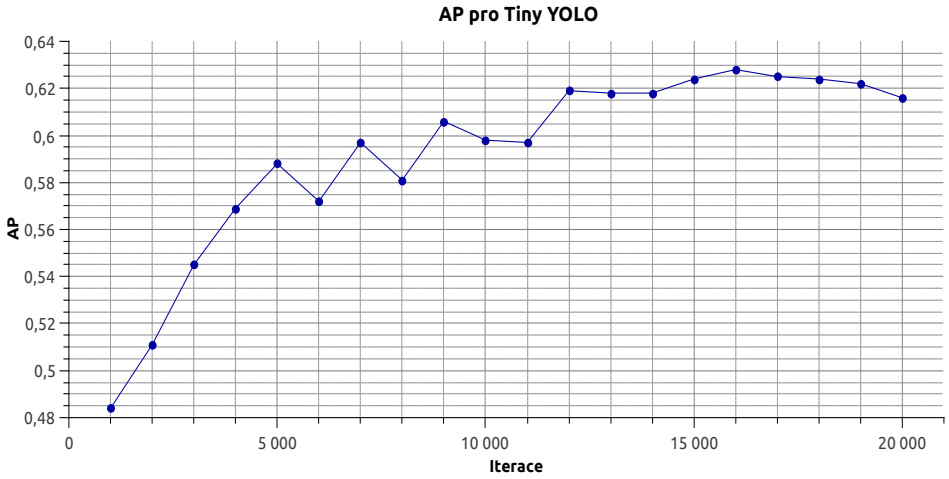
U YOLOv2 modelu byla nejlépe naměřená AP hodnota 73,4%, což je lehce pod hodnotou mAP zmiňovanou v článku [14], která je 76,8%. Při vyhodnocování byla zjištěna rychlost detekce přibližne 70 snímků za vteřinu. Na grafu 5.2 lze vidět vyhodnocování AP pro jednotlivé váhy získané postupně při trénování. Nejvyšší hodnota připadá na 9000 iteraci.



Obrázek 5.2: Hodnoty AP YOLOv2 modelu u jednotlivých vah získaných během trénování.

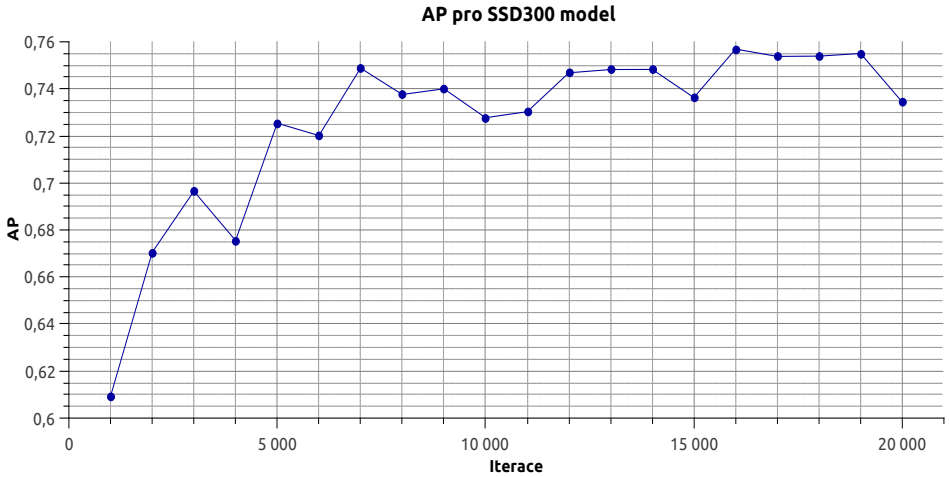
[1] https://github.com/weiliu89/VOCdevkit

Z rodiny YOLO modelů byl také testován model Tiny YOLO. Byl vyzkoušen hlavně z důvodu jeho rychlosti, která přesahovala 150 snímků za vteřinu. Nejvyšší hodnota AP byla zjištěna u 16000 iterace a odpovídá 62,8%. Model, co se detekce týče, překonal původní očekávání a přesností je na tom přibližně jako původní YOLO. Vyhodnocení všech vah je možné vidět na grafu 5.3.



Obrázek 5.3: Hodnoty AP Tiny YOLO modelu u jednotlivých vah získaných během trénování.

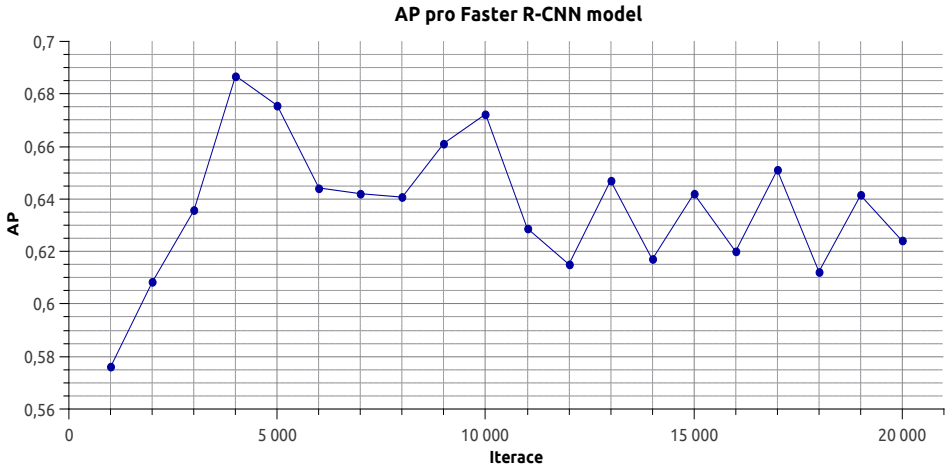
SSD300 model dosáhl menší rychlosti než YOLOv2 a to přibližně 50 snímků za vteřinu. V přesnosti jej však překonal o 2,3% s nejlepší hodnotou 75,7%. Tato hodnota odpovídá 16000 iteraci, jak je možno vidět na grafu 5.4. Tento výsledek není až tak překvapivý, protože se jedná o jeden z předních modelů ve VOC soutěži.



Obrázek 5.4: Hodnoty AP SSD300 modelu u jednotlivých vah získaných během trénování.

Poslední testovaný model byl Faster R-CNN, který je nejstarší z testovaných modelů. Jeho největší slabinou je bezpochyby rychlost. Přibližná rychlost je totiž 7 snímků za vteřinu při detekci. Je tak 9× pomalejší než modely SSD a YOLOv2. Nedosahuje ani takové

přesnosti, jak je možno vidět na grafu 5.5. Nejlepší AP byla zjištěna již při 4000 iteraci a odpovídá 68,7%.



Obrázek 5.5: Hodnoty AP Faster R-CNN modelu u jendnotlivých vah získaných během trénování.

5.2 Porovnání detekcí na konkrétních ukázkách

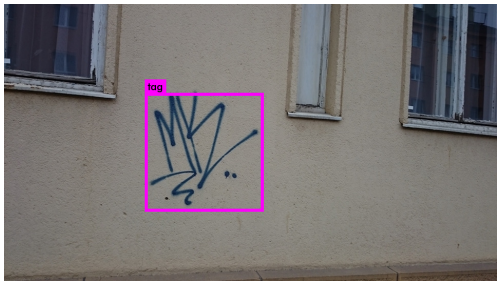
Jednotlivé modely byly testovány z různých hledisek při praktické detekci. Pro každou kategorii byly provedeny detekce na fotografiích zachycující požadovanou situaci. Tyto detekce slouží k porovnání úspěšnosti jednotlivých modelů v daných situacích. Pro praktickou názornost byly vybrány reprezentativní vzorky z testovací sady, které nejlépe demonstrují rozdíly jednotlivých modelů.

5.2.1 Detekce jednoho graffitu na kontrastním pozadí

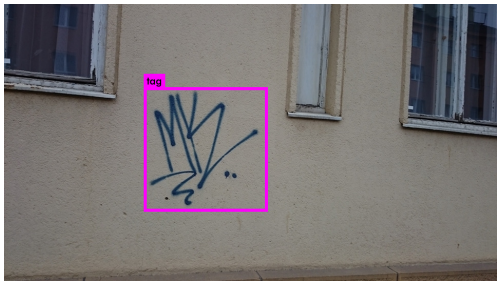
Nejjednodušší detekce spočívá v detekci jediného graffitu na kontrastním pozadí. Tuto základní úlohu zvládají všechny modely bez problémů. Výsledky je možné vidět na obrázcích 5.6.

5.2.2 Detekce více graffitů v jednom obrázku

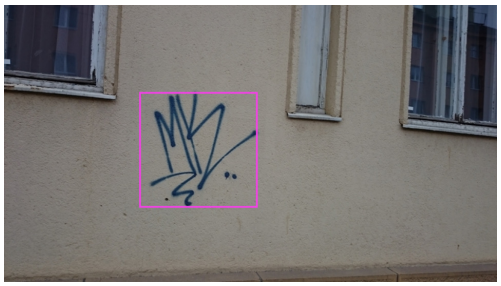
Daleko častěji však nastává situace, kdy je na jednom obrázku soustředěno více graffitů. Zde je možné pozorovat odlišnosti jednotlivých modelů. Jako nejúspěšnější se ukázal model SSD300, který detekuje všechny možné graffiti v obraze a ohraničení je velmi přesné. Modelům YOLOv2 a Faster R-CNN se nepodařilo detekovat jeden graffiti tag na obrázku. Kromě toho také dochází k zhoršení přesnosti detekovaných oblastí. Model Tiny YOLO je již pro detekci více tagů nedostačující. Na ukázce byl schopen detekovat pouze polovinu graffiti tagů, které navíc mají velmi špatně detekovanou oblast výskytu. Jednotlivé ukázky je možné vidět na obrázcích 5.7.



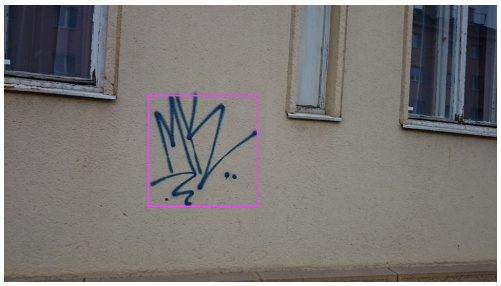
(a) YOLOv2



(b) Tiny YOLO



(c) Faster R-CNN



(d) SSD300

Obrázek 5.6: Srovnání modelů při detekci jednoho graffitu na kontrastním pozadí.

(a) YOLOv2

(b) Tiny YOLO

(c) Faster R-CNN

(d) SSD300

Obrázek 5.7: Srovnání modelů při detekci více graffitů na kontrastním pozadí.

5.2.3 Detekce na špatném pozadí

Většina graffiti tagů se vyskytuje na dobře viditelných místech a jsou čitelná. Najdou se však tagy, které jsou vytvořeny na poškozené omítce nebo členitém podkladu. V takových případech modely YOLO na detekci nestačí. Modely SSD a Faster R-CNN jsou na tom lépe, jak je možné vidět na obrázcích 5.8, kdy se podařilo detekovat graffit.

(a) Faster R-CNN (b) SSD300

Obrázek 5.8: Srovnání modelů při detekci graffitu na špatném pozadí.

V testovací sadě se však objevil i obrázek 5.9, u kterého se nepovedla detekce žádnému modelu. Příčinou může být nedostatečné zastoupení podobných obrázků v trénovací sadě. V případě začlenění většího počtu podobných vzorků by však pravděpodobně došlo k nárůstu nesprávných detekcí v rozmanitých scénách.

Obrázek 5.9: Graffit z testovací sady nedetekovaný žádným modelem.

5.2.4 Detekce v rozmanité scéně

Detekční systém byl natrénován tak, že očekává vstupní obrázky, kde jsou centrem zájmu tagy. Trénovací sada obsahuje převážně graffiti tagy focené ze vzdálenosti 2-5 metrů s minimalizací rušivých elementů. V případě, kdy je k detekci předána fotografie, kde je okolní scéna rozmanitá, dochází k velkému počtu nesprávných detekcí a nedochází k detekci skutečných oblastí. Výsledky detekce je možné vidět na obrázcích 5.10.

5.2.5 Shrnutí

Ze všech otestovaných modelů se jako nejlepší ukázal SSD představený v sekci 3.3. Vykazuje největší přesnost a schopnost detekce za ztížených podmínek.

Modely Faster R-CNN a YOLOv2 dobře zvládají základní detekci tagů na kontrastním pozadí. Nedosahují však takové přesnosti jako model SSD. YOLOv2 model SSD rychlostně

(a) YOLOv2 (b) Tiny YOLO

(c) Faster R-CNN (d) SSD300

Obrázek 5.10: Srovnání modelů při detekci v rozmanité scéně.

překoná s rozdílem 20 FPS, Faster R-CNN je ovšem v tomto ohledu několikanásobně pomalejší.

Tiny YOLO byl vyzkoušen jako případná varianta pro detekci nasazenou v systémech s menším výpočetním výkonem. Ukázalo se, že základní detekci zvládá obstojně, ale v případě většího počtu graffiti tagů v obraze nedetekuje nezanedbatelné množství. Jeho výhodou je rychlost, která až trojnásobná oproti SSD modelu.

Shrnutí jednotlivých výsledků je možné vidět v tabulce 5.1 níže.

	Tiny YOLO	**YOLOv2**	**SSD300**	**Faster R-CNN**
Average precision	62,8%	73,4%	75,7%	68,7%
FPS	~150	~70	~50	~7
Doba trénování	33 hodin	61 hodin	17 hodin	4 hodiny
Detekce jednoho tagu	✓	✓	✓	✓
Detekce více tagů		✓	✓	✓
Detekce za zhoršených podmínek			✓	✓

Tabulka 5.1: Shrnutí jednotlivých aspektů testovaných modelů.

Kapitola 6

Závěr

Hlavním výsledkem této práce je program schopný detekce graffiti tagů z obrazu. Samotný program je založen na detekčním systému Single Shot MultiBox Detector s konkrétním modelem SSD300. Mimo tento program vznikly také další systémy schopné detekce graffiti tagů, které však nedosahují takové přesnosti. Dále byl s touto prací vytvořen dataset čítající 1696 graffiti tagů zachycených na 747 fotografiích.

Výsledný detekční systém byl vybrán na základě srovnání s dalšími nejmodernějšími detekčními systémy, které byly upraveny pro potřeby detekce graffiti tagů. Program v požadovaném obrázku vytvoří ohraničení oblastí detekovaných graffiti. Původním výstupem měly být výřezy jednotlivých graffitů, ale z tohoto řešení sešlo, jelikož některé systémy zabývající se nelegálním graffiti, požadují pouze anotovaný obraz.

Dalším cílem bude zajištění větší datové sady pro případné zvýšení přesnosti detekčního systému. Dále pak uvedení systému do praxe zpracováním některé, z již existujících databází shromažďující graffiti tagy nebo integrací do systémů zabývajících se potíráním nelegálního graffiti.

Volba systému využívající konvoluční neuronové sítě vytváří prostor pro budoucí vytvoření autonomního systému, který graffiti tagy nejenom detekuje, ale zároveň hledá již podobné v databázi. Docíleno by tím mohlo být vytvořením siamské neuronové sítě nebo porovnáváním výstupů finálních konvolučních vrstev stávajícího detekčního systému. Tento systém by mohl být vytvořen později v rámci diplomové práce.

Je nutné podotknout, že oblast detekce objektů s využitím konvolučních neuronových sítí je velmi progresivní. Testované modely jsou z let 2015 a 2016, takže není vyloučeno, že se v blízké budoucnosti objeví efektivnější způsoby detekce. Příkladem by mohl být model Mask R-CNN.

Literatura

[1] Image Recognition and Object Detection : Part 1. 2016.
URL http://www.learnopencv.com/image-recognition-and-object-detection-part1/

[2] Statistické přehledy kriminality za rok 2015. 2016.
URL http://www.policie.cz/soubor/statistiky-od-01-01-2015-do-31-12-2015-zip.aspx

[3] Bay, H.; Tuytelaars, T.; Van Gool, L.: Surf: Speeded up robust features. *Computer vision–ECCV 2006*, 2006: s. 404–417.

[4] Everingham, M.; Gool, L. J. V.; Williams, C. K. I.; aj.: The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, ročník 88, č. 2, 2010: s. 303–338, doi:10.1007/s11263-009-0275-4.
URL http://dx.doi.org/10.1007/s11263-009-0275-4

[5] Freeman, W. T.; Roth, M.: Orientation Histograms for Hand Gesture Recognition. Technická Zpráva TR94-03, MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, Prosinec 1994.
URL http://www.merl.com/publications/TR94-03/

[6] Girshick, R. B.: Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, IEEE Computer Society, 2015, ISBN 978-1-4673-8391-2, s. 1440–1448, doi:10.1109/ICCV.2015.169.
URL http://dx.doi.org/10.1109/ICCV.2015.169

[7] Girshick, R. B.; Donahue, J.; Darrell, T.; aj.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, IEEE Computer Society, 2014, ISBN 978-1-4799-5118-5, s. 580–587, doi:10.1109/CVPR.2014.81.
URL http://dx.doi.org/10.1109/CVPR.2014.81

[8] Hijazi, S.; Kumar, R.; Rowen, C.: Using Convolutional Neural Networks for Image Recognition. 2015.

[9] Ioffe, S.; Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, *JMLR Workshop and Conference Proceedings*, ročník 37, editace F. R. Bach; D. M. Blei, JMLR.org, 2015, s. 448–456.
URL http://jmlr.org/proceedings/papers/v37/ioffe15.html

[10] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, *Lecture Notes in Computer Science*, ročník 9905, editace B. Leibe; J. Matas; N. Sebe; M. Welling, Springer, 2016, ISBN 978-3-319-46447-3, s. 21–37, doi:10.1007/978-3-319-46448-0_2.
URL http://dx.doi.org/10.1007/978-3-319-46448-0_2

[11] Lowe, D. G.: Object Recognition from Local Scale-Invariant Features. In *ICCV*, 1999, s. 1150–1157, doi:10.1109/ICCV.1999.790410.
URL http://dx.doi.org/10.1109/ICCV.1999.790410

[12] Redmon, J.: Darknet: Open Source Neural Networks in C.
http://pjreddie.com/darknet/, 2013–2016.

[13] Redmon, J.; Divvala, S. K.; Girshick, R. B.; aj.: You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, ISBN 978-1-4673-8851-1, s. 779–788, doi:10.1109/CVPR.2016.91.
URL http://dx.doi.org/10.1109/CVPR.2016.91

[14] Redmon, J.; Farhadi, A.: YOLO9000: Better, Faster, Stronger. *CoRR*, ročník abs/1612.08242, 2016.
URL http://arxiv.org/abs/1612.08242

[15] Ren, S.; He, K.; Girshick, R. B.; aj.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 39, č. 6, 2017: s. 1137–1149, doi:10.1109/TPAMI.2016.2577031.
URL https://doi.org/10.1109/TPAMI.2016.2577031

[16] Van de Sande, K. E.; Uijlings, J. R.; Gevers, T.; aj.: Segmentation as selective search for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, s. 1879–1886.

[17] Schmid, M.: *Konvoluční neuronové sítě a jejich implementace*. Diplomová práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra teoretické informatiky a matematické logiky, 2011.

[18] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, ročník abs/1409.1556, 2014.
URL http://arxiv.org/abs/1409.1556

[19] Srinivas, S.; Sarvadevabhatla, R. K.; Mopuri, K. R.; aj.: A Taxonomy of Deep Convolutional Neural Nets for Computer Vision. *Front. Robotics and AI*, ročník 2016, 2016, doi:10.3389/frobt.2015.00036.
URL http://dx.doi.org/10.3389/frobt.2015.00036

[20] Walber: Precision and recall. 2014, [Online; accessed 25-March-2017].
URL
https://upload.wikimedia.org/wikipedia/commons/2/26/Precisionrecall.svg

[21] Zeiler, M. D.; Fergus, R.: Visualizing and Understanding Convolutional Networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, *Lecture Notes in Computer Science*, ročník 8689, editace D. J. Fleet; T. Pajdla; B. Schiele; T. Tuytelaars, Springer, 2014, ISBN 978-3-319-10589-5, s. 818–833, doi:10.1007/978-3-319-10590-1_53.
URL http://dx.doi.org/10.1007/978-3-319-10590-1_53

[22] Šťastný, M.: *Graffiti a sprejerství*. Diplomová práce, Bankovní institut vysoká škola Praha, Katedra ekonomických a sociálních věd, 2010.