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

DETECTION OF GRAFFITI TAGS IN IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN FISCHER**
AUTHOR

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

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Fischer Martin**
Program: Informační technologie
Název: **Detekce graffiti tagů v obraze**
Detection of Graffiti Tags in Image
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku obecné detekce objektů a textu.
2. Vyberte vhodné metody a navrhněte řešení problému detekce graffiti tagů v obraze.
3. Posbírejte vhodnou datovou sadu reálných fotografií graffiti tagů pro vyhodnocení vaší implementace.
4. Experimentujte s vaší implementací a případně navrhněte vlastní modifikace metod.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát a video prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Špaňhel Jakub, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Cílem této práce je porovnat různé přístupy počítačového vidění se záměrem automatické detekce graffiti tagů v obraze. Za tímto účelem byly v řešení použity modely založené na neuronových sítích. V práci byly otestovány jak osvědčené detekční modely, tak i modely experimentální. U nejpřesnějšího z nich (Faster R-CNN) bylo dosaženo přesnosti 83% mAP, což poukázalo na vhodnost těchto modelů při řešení otázky detekce tagů.

Abstract

The aim of this work is to compare different approaches of computer vision with the intention of automatic detection of graffiti tags in the image. The solution was based on models based on neural networks. Both the proven detection models and the experimental models were tested here. The most accurate one (Faster R-CNN) achieved an accuracy of 83% mAP, indicating the suitability of these models to the tag detection problem.

Klíčová slova

detekce objektů, graffiti tagy, konvoluční neuronové sítě, Faster R-CNN, R-FCN, Mask R-CNN, EAST detektor, CCNN, Counting CNN

Keywords

object detection, graffiti tags, convolutional neural network, Faster R-CNN, R-FCN, Mask R-CNN, EAST detector, CCNN, Counting CNN

Citace

FISCHER, Martin. *Detekce graffiti tagů v obraze*. Brno, 2019. 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.

.....
Martin Fischer
4. května 2019

Poděkování

Touto cestou bych rád poděkoval vedoucímu mé bakalářské práce, panu Ing. Jakubovi Špaňhelovi za trpělivost, cenné rady, vstřícnost při konzultacích a dohled při psaní této práce.

Obsah

1 Úvod **2**

2 Související výzkum **3**
2.1 Graffiti . . . 3
2.2 Detekce . . . 5

3 Modely detekce objektu **14**
3.1 R-CNN – použití CNN k detekci objektů . . . 14
3.2 R-FCN . . . 18
3.3 EAST . . . 19
3.4 Counting CNN . . . 20

4 Trénování detektorů **22**
4.1 Dataset . . . 22
4.2 Tensorflow Object Detection API . . . 25
4.3 CCNN . . . 27
4.4 EAST . . . 30

5 Dosažené výsledky **31**
5.1 *Region based* detektory . . . 31
5.2 Experimentální detektory . . . 32
5.3 Shrnutí výsledků . . . 36

6 Závěr **37**

Literatura **38**

Kapitola 1

Úvod

Úvahy, zda je fenomén graffiti pouhopouhý vandalismus, či pouliční umění, rozdělují společnost na dva tábory již dlouhou dobu a nejspíš i dlouho dobu rozdělovat stále budou. Kupříkladu pro policejní složky, napříč celým světem, je však takovéto jednání (kdy někdo poškodí cizí věc) považováno za trestněprávní, za které může být dotyčný potrestán i odnětím svobody. Ve valné hromadě případů je dopadení vandalů velice obtížnou záležitostí. Za rok 2018 se policie České republiky zabývala 4274 případy sprejerství. Objasnit se podařilo zhruba pětinu z nich [24]. Navíc se policisté musejí především spoléhat na výpomoc široké veřejnosti.

I právě z těchto důvodu má smysl porovnat různé přístupy dosažení automatické detekce graffiti tagů v přirozeném prostředí, což je i náplní této práce. Přístup vykazující co možná nejlepší výsledky, může tvořit solidní stavební kámen při tvorbě robustnějších systémů, zabývajících se potíráním nelegální činnosti spojené s pouličním uměním graffiti. To může například znamenat prohledávání policejních databází a jejich rychlou analýzu, která v nejlepším případě může vést i k identifikaci podezřelých osob. Jinými slovy by to mohlo vést například ke snížení finančních prostředků, či snížení času stráveného vyšetřováním těchto trestních činů. Jako příklad systému, který bojuje proti tomuto stále se rozšiřujícímu vandalismu, lze uvést *Tracking and Automated Graffiti Reporting System*[1], který využívá i policejní oddělení města Spojených státu amerických Los Angeles.

V další kapitole je okrajově nastíněná oblast týkající se pouličního uměni graffiti a následně detekce objektů. První část této kapitoly věnuje pozornost graffiti, s hlubším zaměřením na jednu z jeho nejpoužívanějších forem sebevyjádření, a tím je tagování. Druhá část je zaměřena na koncepci detekce objektů s důrazem na použití neuronových sítí, respektive konvolučních neurnových sítí. Principy fungování některých hlavních metod využívaných pří detekci jsou rozebrány v kapitole 3. V této kapitole jsou taktéž pospány experimentální metody EAST a Counting CNN. Obsahem následující kapitoly je objasnění veškerých postupů dodržených při trénování (za účelem detekce graffiti tagů v přírodních scénách) jednotlivých detekčních modelů. Předposlední 5. kapitola zahrnuje vyhodnocení jednotlivých pokusů i jejich následné porovnání. V poslední závěrečné 6. kapitole je nakonec zrekapitulována vykonaná práce.

[1] http://www.594graffiti.com/

Kapitola 2

Související výzkum

První část této kapitoly je zaměřena na graffiti. Celá problematika graffiti je zde zachycena pouze okrajově, jelikož není hlavním předmětem zkoumání této práce. Nicméně pro pochopení smyslu práce je však alespoň její nastínění podstatné. Druhá půle je zaměřena především na hlavní problematiku a tou je detekce objektů. Detekce představuje širokou oblast na poli umělé inteligence, ke které lze přistupovat různými způsoby. V této části je kladen důraz na neuronové sítě, respektive konvoluční neuronové sítě, které v současné době představují nejlepší algoritmy zaměřující se na tuto činnost.

2.1 Graffiti

Graffiti se stalo součástí naší každodenní vizuální zkušenosti. Během celého dne jsme schopni na něj narazit nesčetněkrát. Od malých značek, až po obrovské detailní protestní umění si graffiti vydobylo důležité místo v naší kultuře. V podstatě existuje celá řada typů graffiti. Tahle práce se však zaměřuje výhradně na graffiti tagy a na formu graffiti zvanou *throw-up*.

2.1.1 Graffiti tagy

Tagování (*tagging*), značkování, či jinými slovy akt psaní vlastního jména za pomocí spreje nebo zvýrazňovače, je jedna z nejpoužívanějších forem sebevyjádření, zároveň však i nejvíce stíhaná. Zatímco v očích umělců a autorů těchto podpisů může být tato činnost považována za pouliční umění (nástěnné malby), v očích většiny, a to už od samých počátků prvních výskytů v ulicích New Yorku, má tagování vážný problém. Z velké části je totiž v některých zemích úzce spojováno s aktivitou gangů a jejich násilnou činností, distribucí drog a jiných nezákonných aktivit [29], či jen s bezduchým vandalismem, jenž má malou nebo žádnou estetickou hodnotu a zároveň narušující naše původní myšlenky o vlastnictví nemovitostí. Možná tím největším nepříjemným aspektem je právě jejich umístění. Budovy, skla výloh, mosty, ulice, vlaky, tramvaje, podzemní dráhy, památky nebo jiné plochy, které se obvykle nacházejí ve veřejných prostorech.

Forma tagů

Jako tag je označována v podstatě jakákoli forma ručně psaného graffiti. Tagy mohou obsahovat rafinované a někdy i šifrované zprávy a mohou obsahovat iniciály skupiny, ke které umělec přísluší, případně i písmena jiná. Je považováno za neúctu napsat tag přes tag nebo práci jiného umělce. Nejčastější formou zápisu tagu je velice jednoduchá jednobarevná ná-

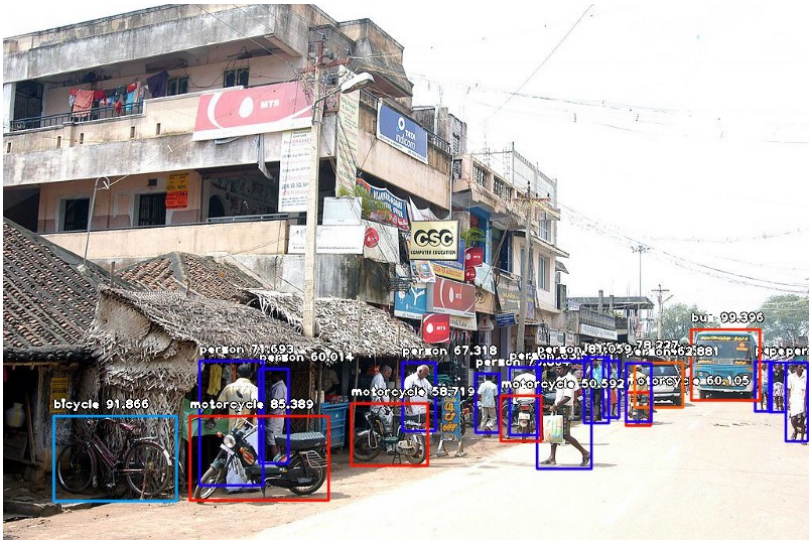
Obrázek 2.1: Vandaly poškozené objekty

Obrázek 2.2: Vlevo: Tag. Vpravo: Throw-up.

stěnná malba psaná zejména jedním nebo pouze několika tahy. Většinou je malovaná v sérii (více vedle sebe) a vždy je brán zřetel na rychlost, proto se vybírají taková písmena, na které lze bez větších problému navázat. Jedná se tak o nejjednodušší a nejrozšířenější typ graffiti. Oproti tomu *throw-up* (hovorově *throwie*) je daleko komplikovanější tag. Obvykle se skládá ze dvou či více barev (jednovrstvá barva pro vyplnění písmen) a specifického bublinového stylu psaní písmen. Nezřídka se stává, že umělci zahrnují do takovéhoto písmenka i tváře. *Throw-up* může být vyroben opakovaně a stejně tak rychle jako tag v rámci pouhých pár desítek vteřin [4, 35].

Autoři

Autoři graffiti tagů, v češtině hojně nazývanými jako sprejeři, mezi sebou zvaní *writeři*, pracují sami nebo ve skupinách lidí (*crews*). Ve větší míře se jedná o mladé lidi ve věku od 15 – 25 let. Jedním z mnoha důvodů této aktivity může být jisté umělecké vyžití a pocit seberealizace nebo je k tvorbě vede pocit napětí vyvolaný ze strany policie, kteří se je snaží přistihnout při činu (často bez výsledků). Jejich výtvory pak představují pravě oné značky a podpisy ve formě přezdívek, symbolů nebo celých obrazců [28].

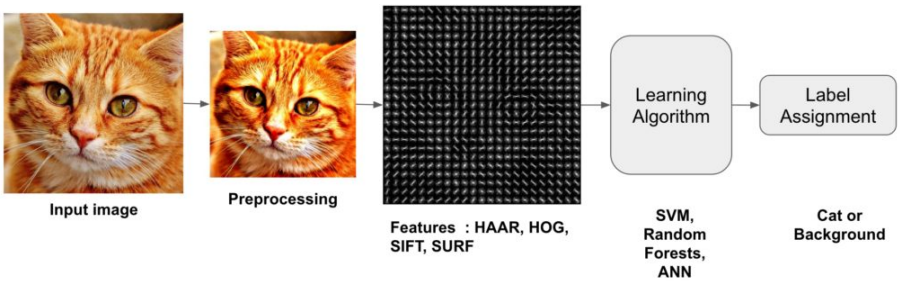


Obrázek 2.3: Detekce objektů ve scéně (Převzato z [22])

2.2 Detekce

Počítačové vidění hraje jednu z důležitých rolí na poli umělé inteligence. Je to věda počítačových a softwarových systému, které umí rozeznat a porozumět obrázkům a scénám. Počítačové vidění jako takové se také skládá z různých aspektů, jako je klasifikace obrázků, detekce objektů, generování obrázků a mnohého dalšího. V této práci se dále pojednává výhradně o oblasti detekce. Rozdíl mezi algoritmy detekce objektů a klasifikačními algoritmy spočívá v tom, že v detekčních algoritmech je snaha nakreslit kolem objektu, jenž je předmětem zájmu, ohraničující rámeček, abychom jej v rámci obrázku mohli nalézt. Nemusí se vždy jednat pouze o jeden ohraničující rám. V praxi je naprostou samozřejmostí, že se během detekce může objevit rámů více, aniž bychom předem věděli, kolik jich bude. Pod pojmem detekce objektu si tedy lze představit schopnost počítačových systémů lokalizovat objekty v obrázku, potažmo ve scéně a každý takto nalezený objekt identifikovat tak, jak je to možno vidět na obrázku 2.3.

Důležitost detekce objektů, jakožto aspektu počítačového vidění roste každým dnem s počtem praktických případů, kde jej lze použít. Detekce objektů je a byla široce používána pro detekci obličeje, detekci vozidel, počítání chodců a v dnešní době čím dál více pro bezpečnostní systémy a autonomní automobily. Tímto výčet samozřejmě nekončí, jelikož metody detekce objektů mohou být použity, tak jako každá počítačová technologie, ve všech různých koutech lidského poznání. Dřívější implementace zahrnovaly klasické algoritmy, jako ty podporované v populární knihovně počítačového vidění OpenCV. Tyto algoritmy ale nejsou schopné dosahovat dostatečné velkých výkonů pro práci za různých podmínek. Přelom v této oblasti nastal s příchodem hlubokého učení, což zapříčinilo vznik moderních a vysoce přesných algoritmů detekce objektů jako například Faster-RCNN, R-CNN, Mask-RCNN (těmito modely se dále zabývá i tato práce) a rychlé, ne však tak přesné, jako jsou SSD, YOLO a další [22, 25].



Obrázek 2.4: Schéma ilustrující kroky tradičního klasifikátoru obrazu (Převzato z [26])

2.2.1 Koncept detekce objektů

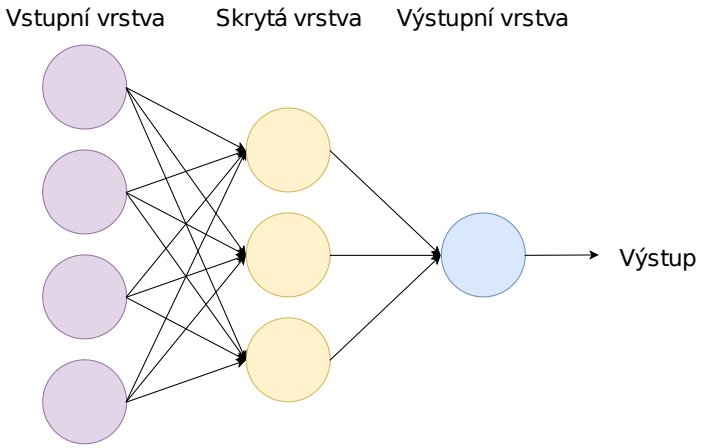
Celý koncept detekce spočívá v tom, že každá třída objektů má své vlastní speciální rysy, kterými se zvýrazňuje a které pomáhají při klasifikaci třídy. Při detekci je těchto rysů využíváno. Obecně se tyto rysy získávají v dílčích částech detekčních systémů. Prvním krokem bývá předzpracování obrázků za účelem normalizování kontrastu a jasu. Cílem bývá například odstranění zkreslení nebo zvýraznění rysů objektu. Tyto změny můžou nakonec pomoci k dosažení lepších výsledků. Nelze však s velkou přesností říci, jaké metody předzpracování obrázků jsou ty pravé v závislosti na různorodosti požadovaných systémů, a tak většina těchto aktivit spadá do kategorie pokusů a omylů. Součástí předzpracování dále bývá ořezání nebo změna velikosti vstupního obrazu, která je dosti klíčová pro další část, a tou je extrakce příznaků.

Tento krok slouží k získání důležitých informací obsažených v obrázku a vynechání těch ostatních, tak jak se dá vidět na obrázku 2.4. Schéma na obrázku znázorňuje kroky používané v tradičním klasifikačním (potažmo detekčním) zařízení. Algoritmy založené na hlubokém učení kompletně překonávají krok extrakce, který je zde vidět. K extrakci informací (příznaků) muže posloužit celá řada algoritmů, příkladem může být histogram orientovaných gradientů, *Scale-Invariant Feature Transform* (SIFT) nebo *Speeded Up Robust Feature* (SURF). Jednotlivé přístupy návrhu takových to funkcí jsou rozhodující pro výkonnost algoritmu. Po použití extraktoru jsou získané informace poslány do další fáze. Tu představuje učící algoritmus, respektive aplikace různých klasifikátoru jako je *support vector machines* (SVM), náhodný les, KNN či MLP. Takový přístup se například používá u detekce vozidel, kde lze (jednoduše řečeno) nalézt kola, kapotu, dveře, poznávací značku a další vlastnosti jako je barva apod. (algoritmus se učí jednotlivé příznaky zpracovávat). Tyto algoritmy mohou být taktéž použity pro problémy regrese, a tudíž i k následné detekci objektu v obraze [26].

V současné době jsou nejlepšími algoritmy, které jsou zaměřeny na tuto činnost, založené na (konvolučních) neuronových sítích, viz níže 2.2.2. Jejich obrovské schopnosti ilustruje *Large Scale Visual Recognition Challenge* (ImageNet), což je *benchmark* v oblasti klasifikace a detekce objektů s milióny obrazů a stovkami klasifikačních tříd.

2.2.2 Neuronové sítě

Myšlenka umělé neuronové sítě je inspirována biologickou neuronovou sítí, která tvoří mozek živých bytostí. Lidský mozek se skládá z neuronů zpracovávající informace, které si sdělují



Obrázek 2.5: Příklad dopředné neuronové sítě s jednou skrytou vrstvou, která obsahuje 3 neuronony (Upraveno dle [31])

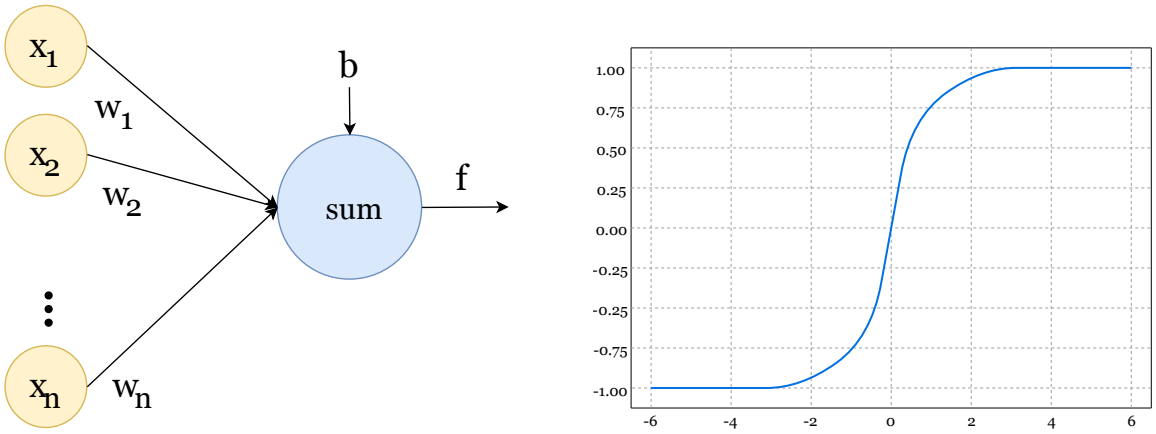
mezi sebou navzájem. Neuron se skládá z dendritů, na které přicházejí vstupy. Na základě těchto vstupů poté produkují výstup, který přes axon putuje k jinému neuronu.

Umělá neuronová síť se skládá z umělých neuronů nazývaných uzly, které taktéž přijímají vstupy, které zpracovávají a transformují pomocí určitých přenosových funkcí. Neuronová síť se obecně skládá ze spojených neuronů uspořádaných ve 3 vrstvách:

- vstupní vrstva, která je odpovědná za přijímání velkých objemů vstupů v různých formátech, jako jsou čísla, obrazové pixely, zvuk atd.,
- skrytá vrstva, jež je zodpovědná za zpracování dat, matematické výpočty, extrakci prvků atd. a
- výstupní vrstva, generující požadovaný výstup na základě informací, které obdrží od předchozích vrstev [1].

Obvykle se však síť skládá z vícero skrytých vrstev, aby ji bylo možno naučit zvládat složitější úkoly jako třeba právě detekci objektů. Pokud jsou neurony propojeny v několika vrstvách, je to známé jako hluboké učení (*deep learning*). Jeden ze základních elementů neuronové sítě je právě její schopnost se učit. Neuronový systém není jenom komplexní systém, ale i systém adaptivní, což znamená, že může měnit vnitřní strukturu na základě informací, které ji prostupují. Typicky je toto dosaženo nastavením vah. Každá čára v diagramu 2.6 reprezentuje spojení mezi dvěma neurony a indikuje cestu po které informace putují. Každé takové to spojení má váhu, číslo, které řídí signál mezi těmito dvěma neurony. Jestliže síť generuje správný výsledek, není třeba tyto váhy nějak upravovat. Pokud však ne, je třeba právě tyto váhy upravit s cílem zlepšit následné výsledky [8]. Proces, jakým způsobem neuron zpracovává svůj výstup, je možno rozdělit do třech kroku.

- Každý vstup se zvětší nebo zmenší – když přijde signál, vynásobí se hodnotou váhy, která je přiřazena tomuto konkrétnímu vstupu. To znamená, že pokud má neuron tři vstupy, má tři váhy, které lze během fáze učení individuálně upravit.
- Všechny signály jsou sečteny – v dalším kroku jsou upravené vstupní signály shrnuty do jediné hodnoty. V tomto kroku je k součtu přidána hodnota, která se nazývá *bias*. Ta je během učení také upravována.



Obrázek 2.6: Příklad neuronu (Upraveno dle [31]) a průběhu možné sigmoidální přenosové funkce

- Aktivace – nakonec se výsledek výpočtu neuronu změní na výstupní signál. Ten se tak stává argumentem aktivační funkce [5]. Rovnice ve tvaru

$$f\left(b + \sum_{n=1}^{n} x_i w_i\right) \tag{2.1}$$

  popisuje aktivační funkci aplikovanou na vážený součet všech vstupů [31].

Aktivační (přenosové) funkce jsou pro umělé neuronové sítě extrémně důležité. V podstatě rozhodují, zda by měl být neuron aktivován nebo ne. Zda jsou ty informace, které neuron přijímá, relevantní ,nebo by měly být ignorovány. Přenosových funkcí je užíváno více druhů [11]. Jednou z nich může být sigmoidální přenosová funkce, kterou je možno vidět na 2.6.

Učící proces neuronových sítí

Pro učení neuronových sítí je zde hned několik strategií [8]. V této práci je pozornost věnována především učení kontrolovanému.

Nekontrolované učení (*Unsupervised Learning*)

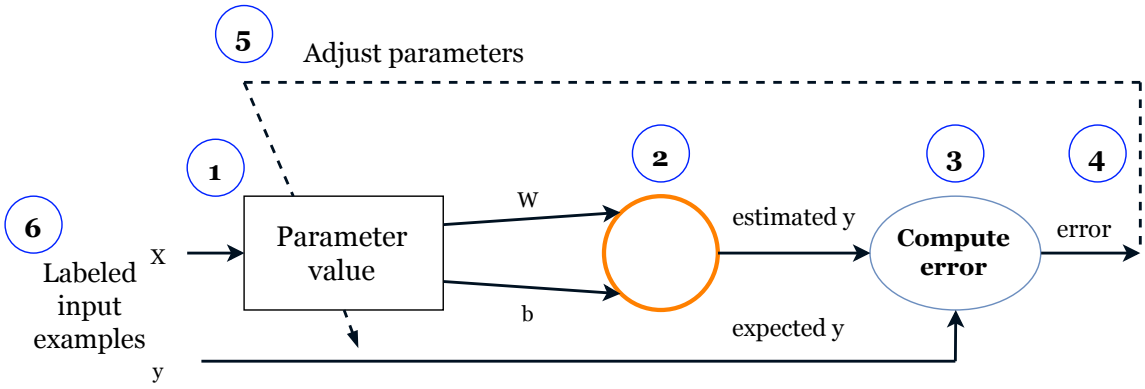
Tato strategie se výhradně použije tam, kde není sada příkladů s předem známými odpověďmi. Funguje na principu shlukování, tj. dělení souboru prvků do skupin podle nějakého neznámého vzoru.

Posilované učení (*Reinforcement Learning*)

Posilování učení je velice běžné v robotice. V čase t provádí robot úkol a sleduje výsledky. Tato strategie je postavena zejména na prvku pozorování.

Kontrolované učení (*Supervised Learning*)

V celku se jedná o strategii, která zahrnuje učitele chytřejšího, než je síť sama o sobě. Příkladem může být učitel, který síti ukazuje sadu obličejů a zároveň už ví, jaké jméno



Obrázek 2.7: Znázorněný proces učení neuronové sítě (Upraveno dle [19])

je, s jakým obličejem spojeno. Síť dělá své odhady a učitel posléze poskytuje sítí správně odpovědi. Síť potom může porovnat své odpovědi vzhledem k těm správným a provést úpravy podle svých chyb. Všechny neuronové sítě v této práci následují tento model učení.

Učení neuronové sítě lze tedy chápat jako proces učení hodnot parametrů sítě (váhy a *bias* hodnoty) a lze jej zároveň nazvat jako iterativní proces „chodu a návratu“ skrze vrstvy neuronů. „Chod“ značí dopřednou propagaci informací a „návrat“ pak propagaci zpětnou.

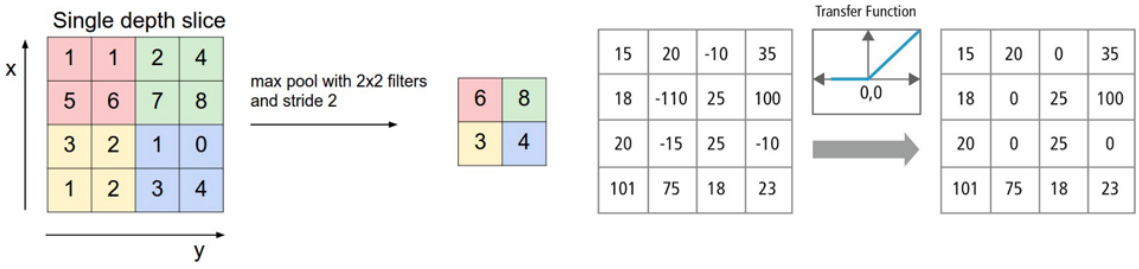
Při první fázi je síť vystavena trénovacím údajům (data) a ty procházejí neuronovou sítí za účelem výpočtu výsledných predikcí. Vstupní data jsou přenášena přes síť způsobem, kdy všechny neurony aplikují svou transformaci na informace, které dostávají od neuronů předchozí vrstvy a posílají je na neurony další vrstvy. Když data projdou všemi vrstvami a všechny jejich neurony provedly patřičné výpočty, poslední vrstva bude dosažena s hodnotou výsledné predikce pro vstupní příklady.

Dále je potřeba chybové funkce, která se používá pro odhad chyby a díky ní porovnat, jak moc dobrá, nebo špatná je výsledná predikce vzhledem ke správnému výsledku (učení s učitelem). Ideálně tak, aby rozdíl mezi odhadovanou a očekávanou hodnotou byl nulový. Při trénování sítě dochází k postupným úpravám trénovacích parametrů, právě tak, aby byla dosaženo co možná nejlepších výsledků.

Po výpočtu chyby se tato informace šíří zpět. Počínaje výstupní vrstvou se tato informace šíří na všechny neurony ve skryté vrstvě, která je přímo spojena s výstupem. Neurony obdrží pouze zlomek celkového signálu na základě relativního příspěvku, který každý neuron přispěl k původnímu výstupu. Tato činnost se stále opakuje, vrstva po vrstvě, dokud všechny neurony nedostanou chybový signál, který popisuje jejich relativní přínos k celkové chybě.

Když je tato informace rozšířena, je třeba upravit váhy spojů mezi jednotlivými neurony. Je to třeba provádět tak, aby se hodnota chyby blížila k nule. Pro tento účel je možno použít techniku gradientního sestupu. Tato technika mění váhy v malých přírůstcích pomocí výpočtu derivace (nebo gradientu) chybové funkce, což umožňuje vidět směr, kterým se dá sestoupit ke globálnímu minimu [19].

Pro analýzu a zpracování vizuálních vjemů, ať už se jedná o klasifikaci obrázků, rozpoznávání objektu, či detekci, se však nejčastěji používají konvoluční neuronové sítě (CNN).

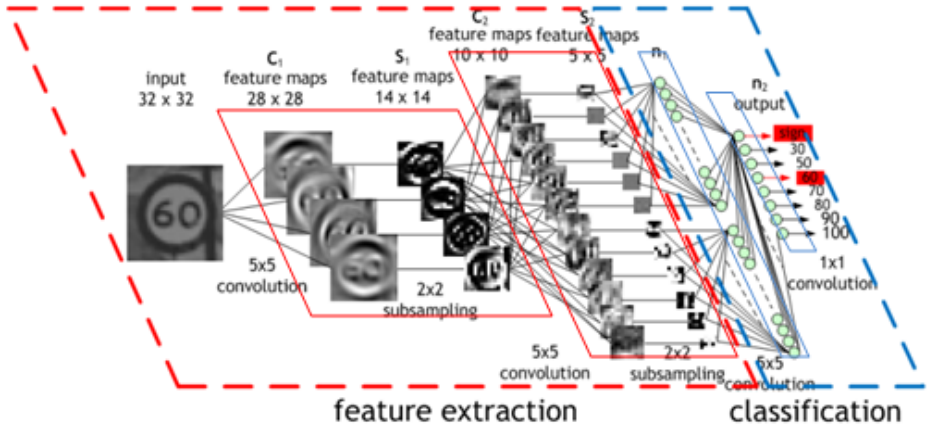


Obrázek 2.8: Vlevo: Princip metody *max pooling*. Vpravo: Princip *Rectified Linear Unit* (ReLU). (Převzato z [9, 13])

Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN), jako ty neuronové, jsou tvořeny neurony s váhami a *bias* parametry. Každý neuron přijímá několik vstupů, provede vážený součet, pošle jej dál do aktivační funkce a odpoví nějakým výstupem. Celá sít má chybovou funkci a vše, co již bylo zmíněno při neuronových sítích. Na rozdíl od regulérních neuronových sítí má CNN tři základní architektonické prvky. První je lokální konektivita. Neurony jedné vrstvy jsou spojeny s neurony následující vrstvy, které jsou si prostorově blízké. Tento návrh snižuje počet převážné většiny spojení mezi po sobě jdoucími vrstvami, ale udržuje ty, které nesou nejvíce podstatné informace. Předpokladem je, že vstupní data mají určitý prostorový význam. Příkladem může být vztah mezí dvěma vzdálenými pixely obrázku, jež bude pravděpodobně méně významný než u dvou sousedních. Dalším významným prvkem jsou sdílené váhy. Tenhle prvek tvoří CNN „konvolučními“. Tím, že neurony jedné vrstvy sdílejí váhy, posouvání dat skrze sít se stává stejné, jako provádění konvoluce obrázku a filtru se záměrem vytvoření obrázku nového. Trénování CNN se potom stává úlohou učení jednotlivých filtrů (rozhodování jaké rysy, by se měly hledat v datech). Napomáhá to ke snížení počtu parametrů v celém systému a činí tak výpočet účinnější, případně to pomáhá i jako kontrola nad přeučením sítě. A v neposlední řadě to jsou *pooling* vrstvy a ReLU funkce [14]. Princip těchto metod zachycují obrázek 2.8. Nejčastěji používaným typem metody *pooling* je *max pooling*, který v každém okně bere největší hodnotu. Velikost těchto oken je třeba předem zadat. Tato metoda zmenšuje velikost mapy příznaků (výstup pro daný filtr) a zároveň stále udržuje podstatné informace [9]. ReLu nahrazuje všechny negativní hodnoty pixelů příznakové mapy hodnotou nula. Účelem ReLU je zavést do sítě nelinearitu, protože většina dat v reálném světě, které bychom chtěli, aby se síť postupně naučila, by byla nelineární [13]. Architekturu konkrétní sítě je možné vidět na obrázku 2.9, kde je vstupní obrázek dopravní značky filtrován čtyřmi 5x5 konvolučními filtry, které vytvářejí 4 mapy příznaků. Ty jsou dále zmenšeny pomocí metody *max pooling*. Další vrstva aplikuje na tyto zmenšené obrázky deset 5x5 konvolučních filtrů a znova je zmenší. Konečná vrstva je plně propojená vrstva, ve které jsou všechny generované prvky kombinovány a používány v klasifikátoru.

Na rozdíl od neuronových sítí, kde je vstupem vektor, zde je jako vstup vícekanálový obrázek (3 kanály v případě obrázků). Možno vidět, že konvoluční vrstva je hlavním stavebním blokem těchto neuronových sítí. Vstupní obrázky jsou zpracovány konvolučním filtrem, který získává 3D mapy příznaků.



Obrázek 2.9: Architektura konvoluční sítě (Upraveno dle [21])

Obrázek 2.10 znázorňuje natrénovanou síť. Filtry v první vrstvě, vlivem zpětné propagace, mají nynější podobu ve formě, jakých si fleků s barevnými dílky a hranami. V hlubších konvolučních vrstvách se filtry konvolují se vstupy předchozích vrstev. Takže berou menší barevné dílky nebo hrany a vyrábějí z nich větší dílky.

Extrakce příznaků, ke kterým dochází během konvoluce, by se daly rozdělit do pomyslných kategorií. Nízko-úrovňové příznaky, do který je možno zařadit například čáry, křivky a barvy. Do střední úrovně spadá například rozeznání hran a rohů, které jsou daleko více komplexní. Poslední nejvyšší úroveň pak zahrnuje rozpoznání tříd, či oblastí [12].

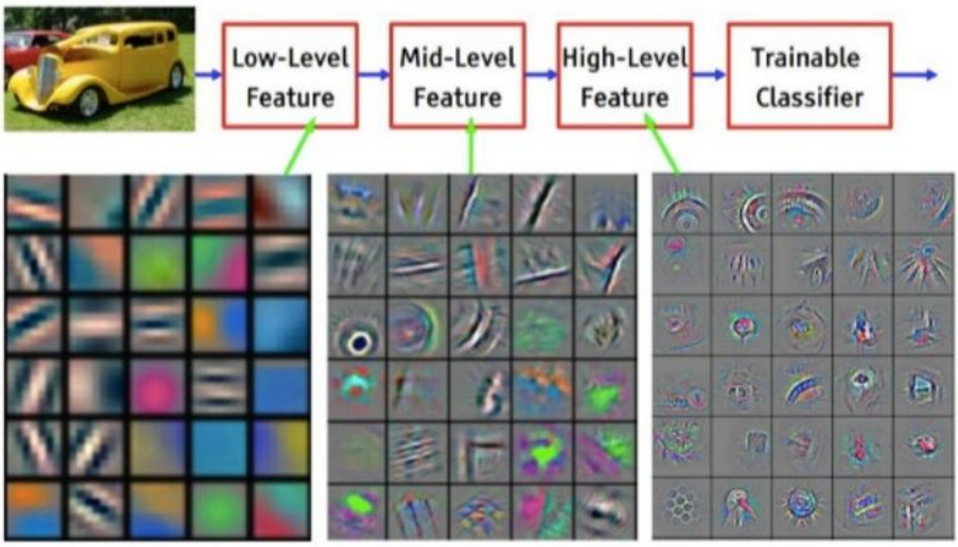
Frameworky pro detekci objektů

Většina frameworků (typicky) pro detekci objektů nabývá třech základních kroků.

- Model nebo algoritmus, který se použije k nalezání oblastí, jež jsou předmětem zájmu. Tyto oblasti představují velké množství ohraničujících rámečků, které pokrývají celý obrázek.
- Z každého takovéhoto rámečku jsou extrahovány vizuální příznaky. Posléze jsou vyhodnoceny a je zjištěno, zda a jaký objekt je v něm obsažen.
- V závěrečném kroku se překrývající rámečky skládají do jediného ohraničujícího rámečku (*non maximum suppression*).

K vytvoření takových to oblastí existuje hned několik různých přístupů. Původně se používal algoritmus *Selective Search*, který se pokouší seskupovat pixely a vytvářet návrhy (oblastí) založené na generovaných shlucích. Další přístupy ke generaci těchto oblastí využívají více komplexní příznaky a nebo „hrubou sílu“. Přístupy hrubé síly jsou podobné posuvnému okénku, které se na obrázek aplikuje v několika poměrech a velikostí. Tyto oblasti jsou generovány automaticky, aniž by se nějak zajímaly o významné rysy v obrázku.

Důležitými aspekty, na které je třeba brát zřetel, jsou počet oblastí, který vzniká při výrobě a výpočetní složitost. Čím více oblastí se vytvoří, tím pravděpodobněji bude objekt nalezen. Na druhé straně, pokud se vytvoří všechny možné návrhy, nebude možné detektor objektů spustit například v reálném čase.

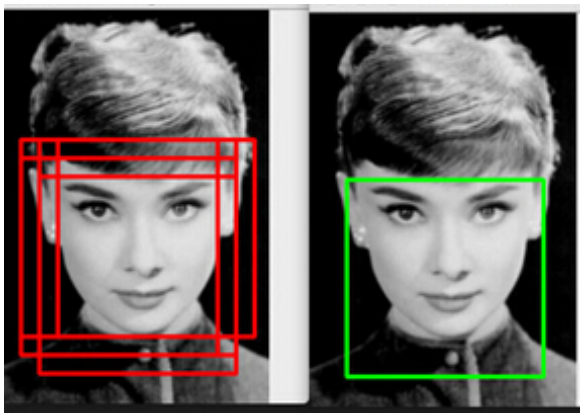


Obrázek 2.10: Vizualizace příznaků natrénované konvoluční sítě (Převzato z [12])

Cílem extrakce příznaků je zmenšit obrázek s proměnnou velikostí na sadu příznaků pevného počtu. Detekční modely jsou typicky konstruovány za užití silných extrakčních metod (ať už založené na algoritmech počítačového vidění nebo hlubokého učení). Ve frameworcích detekce objektů, se pro účely extrakce často využívá již před trénovaných klasifikačních modelů. Ty totiž mají tendenci generalizovat jednotlivé objekty poměrně dobře. Mezi moderní architektury používajících se jako extraktory pro detekční modely například patří Inception a ResNet a mezi starší, které daly za vznik těm moderním, například AlexNet nebo VGG 16. Téměř všechny architektury založené na konvoluční neuronových sítí sledují stejné obecné principy návrhu. To znamená na vstup aplikovat konvoluční vrstvy, postupně převzorkovávat prostorové rozměry a zároveň zvyšovat počet příznakových map (*feature/activation maps*). Zatímco klasické síťové architektury byly složeny z jednoduše skládaných konvolučních vrstev, moderní architektury zkoumají nové a inovativní způsoby konstruování konvoluční vrstev způsobem, který umožňuje více efektivní učení. Téměř všechny architektury jsou založeny na opakovatelné jednotce, která se používá v rámci celé sítě [16, 20].

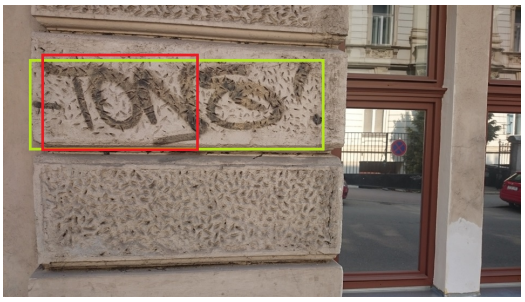
Hlavní myšlenkou *non-maximum suppression* (NMS) je zajistit, že je daný objekt identifikován, či detekován pouze jednou. Pokud nějaký objekt leží ve vícero predikovaných oblastech, NMS zaručuje identifikaci optimální oblasti mezi všemi kandidáty, pro které daný objekt leží vně. Nejprve odstraní všechny oblasti, pro které je pravděpodobnost, že obsahuje daný objekt menší než hodnota prahu. Poté ze všech kandidátů vybere tu oblast, ke které se vztahuje největší pravděpodobnost.

Nejběžnější metrikou pro vyhodnocení takových to modelů používaných v úlohách rozpoznávání objektů, či detekce je mAP (*mean average precision*), neboli střední průměrná přesnost. Je to hodnota, která se pohybuje mezi hodnotami od 0 až do 100 a větší hodnota typicky znamená lepší výsledek. Nejprve se AP vypočítá pro každou třídu a poté se zprů-



Obrázek 2.11: Výsledek detekce před (červeně) a po (zeleně) aplikování NMS (Převzato z [20])

Obrázek 2.12: Příklad dobrého skóre IoU



Obrázek 2.13: Příklad špatného skóre IoU

měruje přes ostatní třídy. Konečným výsledkem je mAP. Průměrná přesnost (AP) je oblast pod PR křivkou, kde P značí *Precision*. Ta říká, jak přesné jsou predikce (kolik procent ze všech pozitivních predikcí je správně). Kdežto R značí *Recall*. Tato hodnota měří, jak dobře detektor hledá vše, co by najít měl [18]. Zde jsou jejich matematické definice:

$$Precision = \frac{TP}{TP + FP}, \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN}, \tag{2.3}$$

kde TP je počet *true positive* detekcí, FP *false positive* detekcí a FP *false negative* detekcí.

Detekce (rámeček kolem objektu) se dá považovat za pozitivní, pokud hodnota IoU (*intersection over union*) se skutečným rámečkem (*ground-truth box*) objektu je větší než hodnota prahu. Ta většinou nabývá hodnoty 0.5. Pak namísto mAP je typicky psáno mAP@0.5 nebo mAP@0.25. IoU je tedy běžná metoda, používaná k vyhodnocení, jak přesně se výstup sítě shoduje se skutečným rámečkem kolem objektu. Výpočet hodnoty IoU je jednoduchý — vydělí se překrývající se oblast mezi dvěma rámečky oblastí jejich spojení [20]. Na obrázcích 2.12 a 2.13 je skutečný rámeček vykreslen zelenou barvou. Rámeček predikovaný sítí je vykreslen barvou červenou.

Kapitola 3

Modely detekce objektu

Už od doby, kdy v roce 2012 vyhrála síť Alexe Križevského, Geoffa Hintona a Ilya Sutskevera *Large Scale Visual Recognition Challenge*, se konvoluční neuronové sítě staly zlatým standardem pro klasifikaci obrazu, potažmo detekce. V této sekci práce jsou nastíněny myšlenky některých hlavních metod užívaných při detekci a segmentaci objektů a vývoje jednotlivých implementací. Zvláště se zaměřením na R-CNN (původní aplikace CNN pro tyto úlohy) spolu s jeho potomky Fast R-CNN a Faster R-CNN (či R-FCN). Nakonec bude nastíněna i struktura Mask R-CNN rozšiřující tyto metody detekce objektů o segmentaci na úrovni pixelů [10].

3.1 R-CNN – použití CNN k detekci objektů

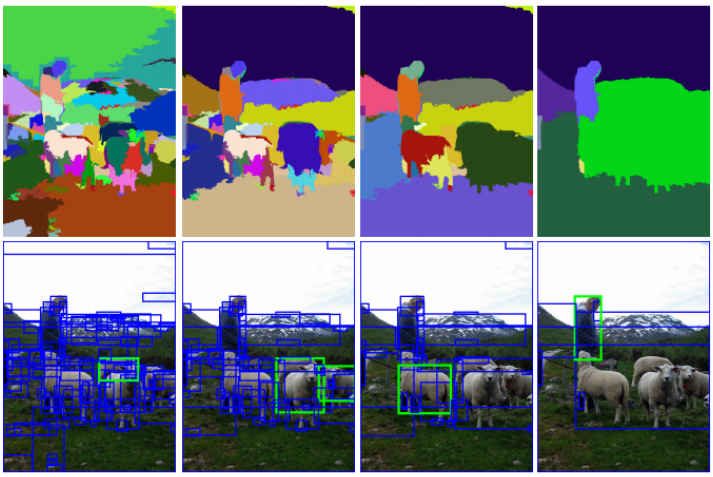
R-CNN pracuje tak, že v obrázku navrhne spoustu ohraničujících rámečků a zkoumá, zda některý z nich skutečně odpovídá nějakému objektu. R-CNN vytváří tyto ohraničující rámečky, nebo návrhy oblastí, za použití algoritmu *Selective Search*, který zkoumá obrázek za pomocí okének různých velikosti. Pro každou velikost se snaží seskupit dohromady sousední pixely podle textury, barvy nebo intenzity, tak aby byla schopna identifikace objektu. Takovéto seskupování pixelů ukazuje obrázek 3.1.

Když už jsou návrhy oblastí (rámečků, ve kterých by se mohly objekty vyskytovat) vytvořeny, R-CNN upraví jejich rozměry na čtverečky a pošle jej skrze upravenou verzi sítě AlexNet. Na poslední vrstvu konvoluční sítě, R-CNN přidá SVM, který jednoduše klasifikuje, zda se zde hledaný objekt nachází (případně jaký). Toto je zahrnuto ve čtvrtém kroku na obrázku 3.2. Posledním krokem R-CNN je spuštění jednoduché lineární regrese na konkrétní návrh rámečku, aby tak mohly být vygenerovány správné souřadnice rámečku [10].

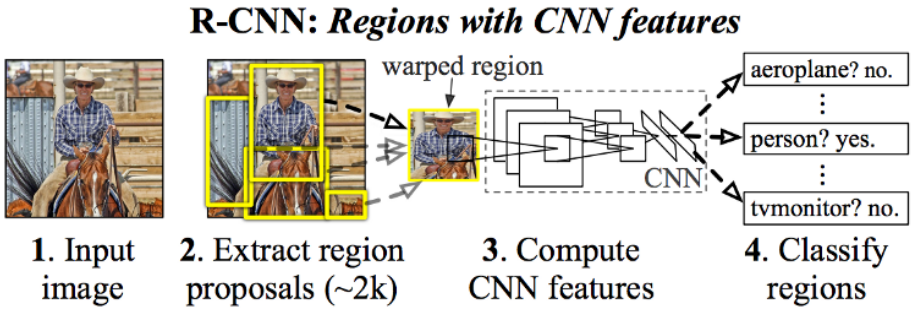
3.1.1 Fast R-CNN – zrychlení a zjednodušení R-CNN

Díky toho, že R-CNN vyžaduje pro každý obrázek, pro každý jeden návrh oblasti průchod konvoluční sítí (to je zhruba 2000 průchodu pro každý obrázek), je R-CNN docela pomalé. Je totiž třeba trénovat každou jednu část architektury, jak konvoluční sít pro generování příznaku, tak i klasifikátor, tak i závěrečný regresní model (výsledná úprava rámečku). Proto je model nesmírně náročný na trénování.

Jednou z hlavních myšlenek tohoto modelu je se vypořádat se zbytečným spouštěním stejných CNN výpočtu znova a znova pro jeden obrázek, které vznikaly díky velkému počtu překrývajících se návrhu oblastí. Tedy onou myšlenkou je spustit CNN pouze jednou pro každý obrázek a najít způsob, jak sdílet výpočty skrze návrhy. Fast R-CNN toho dosahuje



Obrázek 3.1: Příklad, jakým způsobem pracuje algoritmus *Selective Search* (Převzato z [10])



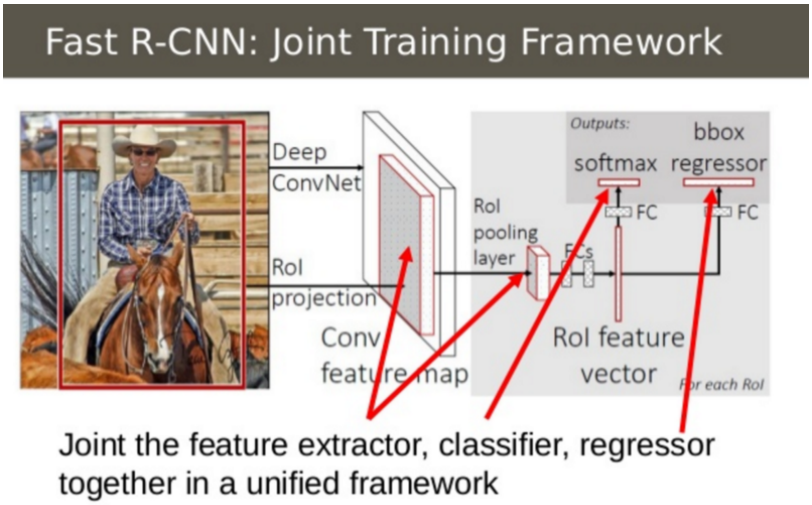
Obrázek 3.2: Architektura R-CNN (Převzato z [10])

pomocí techniky zvané RoIPool (*Region of Interest Pooling*). Příznaky konvoluční sítě jsou pro každou oblast získány výběrem příslušné oblastí z CNN příznakové mapy. Následně jsou příznaky každé oblasti shromažďovány (obvykle pomocí metody max pooling). Takže na rozdíl od předchozích 2000 průchodů, teď stačí pouze jeden.

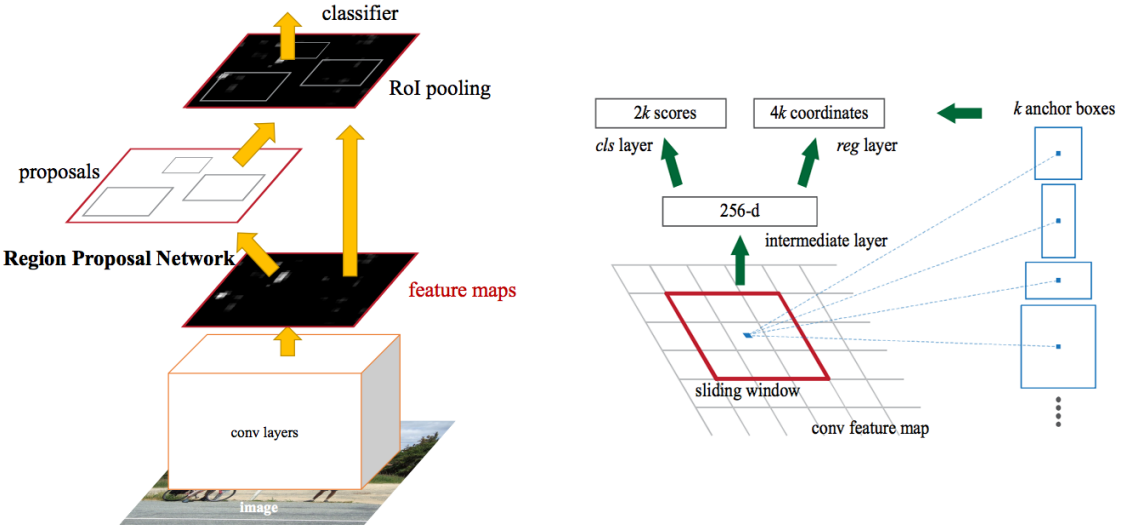
Jak již bylo zmíněno, je potřeba trénovat každou jednu část architektury. Proto druhou hlavní myšlenkou je společné trénování CNN, klasifikátoru a regrese pro ohraničující rámečky v jediném modelu. Fast-RCNN, na místo předchozího přístupu, pro to používá pouze jedinou síť. Na obrázku 3.3 lze vidět, že ve vrchní části CNN Fast-RCNN k získání klasifikace objektů nahradil SVM vrstvou *softmax*. Taktéž přidal vrstvu lineární regrese, aby generovala souřadnice ohraničujícího rámečku. Tímto všechny důležité výstupy pocházejí pouze z jedné jediné sítě [10].

3.1.2 Faster R-CNN – Urychlení návrhů oblastí

I přes veškerá vylepšení, která byla v předchozích modelech docílena, se ve Fast R-CNN stále objevuje problém v podobě návrhu daných oblastí. Prvním krokem pro detekování pozice bylo dříve generování velkého množství potencionálních ohraničujících rámečků, které bylo třeba otestovat. V modelu Fast R-CNN byly tyto návrhy vytvářeny taktéž pomocí



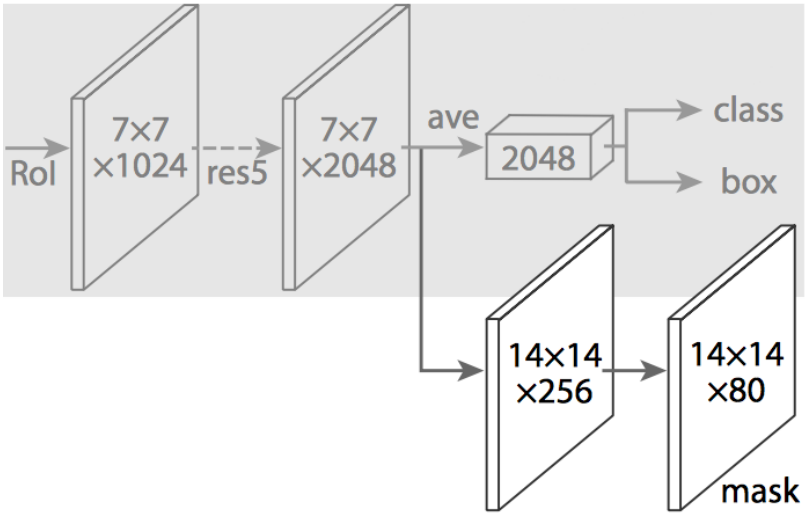
Obrázek 3.3: Architektura Fast R-CNN (Převzato z [10])



Obrázek 3.4: Vlevo: Architektura Faster R-CNN. Vpravo: Princip fungování *Region Proposal Network*. (Převzato z [10])

algoritmu *Selective Search*. A jakožto poměrně pomalý proces, se tento proces ukázal být také překážejícím aspektem celé činnosti.

U předchozího modelu byly návrhy oblastí závislé na příznacích obrázku, které již byly vypočítány při průchodu CNN. Faster R-CNN se při návrhu oblastí snaží využít znova těch stejných výsledků sítě, namísto spuštění algoritmu *Selective Search*. Obrázek 3.4 naznačuje, jak je jediná CNN schopna vypořádat se s návrhem oblastí a zároveň i s klasifikací. Tímto způsobem je nutno trénovat pouze jednu CNN a návrhy jsou tak dosaženy prakticky zadarmo. Faster R-CNN generuje tyto oblasti tak, že je navíc přidána plně konvoluční síť, známá jako RPN (*Region Proposal Network*). RPN pracuje na principu posuvného okénka nad mapou příznaků a u každého okénka vrací k možných ohraničujících rámečků a hodnocení (jak dobré se očekává, že budou) ke každému z nich 3.4.



Obrázek 3.5: Přidaná větev k původnímu Faster R-CNN (Převzato z [10])

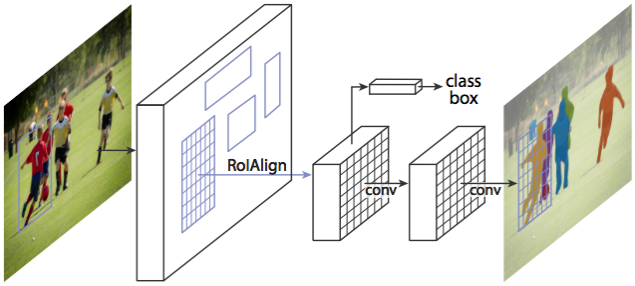
Objekty v obrázku by měly intuitivně vyhovovat určitým běžným poměrům a velikostem. Například mohou být požadovány obdélníkové rámečky, které se podobají tvarům lidí. Tak samo je logicky požadováno, aby v tomto případě tyto rámečky nebyly například příliš tenké. Tímto způsobem se vytvoří k běžných poměrů stran, které se nazývají *anchor boxes*. Výstupem pro každý takovýto „speciální“ rámeček je ohodnocení rámečku a samostatný ohraničující rámeček [10].

3.1.3 Mask R-CNN – Rozšíření o segmentaci na úrovni pixelů

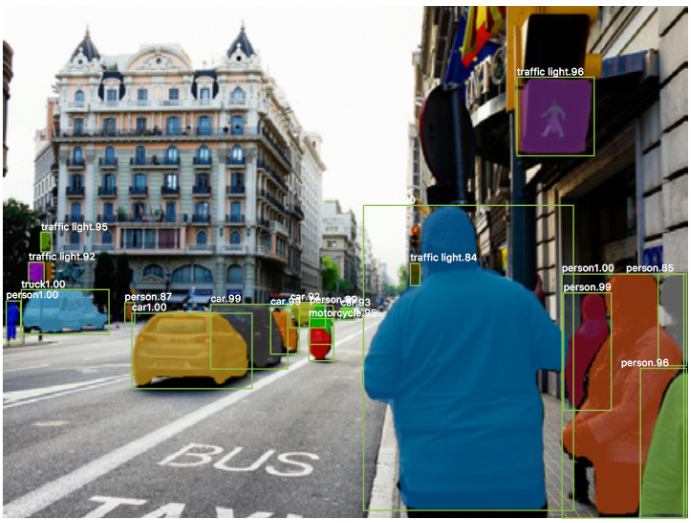
Architektrura Mask R-CNN umožňuje řešit problém, který se obecně nazývá segmentace obrazu. Tedy že namísto lokalizace objektu pomocí pouhého ohraničujícího rámečku, lokalizuje přesné pixely každého objektu.

Mask R-CNN to dělá přidáním větve (k původnímu Faster R-CNN), jejíž výstupem je binární maska, která určuje, zda je, či není daný pixel součástí hledaného objektu. Obrázek 3.5 zobrazuje přidanou větev, a tak jako před tím se stejně jedná o plně konvoluční síť nad mapou příznaku generovanou CNN. Výstupem je tedy matice s jedničkami na všech pozicích, kde pixely přísluší danému objektu a s nulami všude jinde (známé jako binární maska). Vzhledem k tomu, že segmentace obrazu oproti ohraničujícím rámečkům vyžaduje specificitu na úrovni pixelů (což přirozeně vede nepřesnostem) je přesnost zvýšena použitím metody známé jako *RoIAlign* (lepší nastavení metody *RoIPool*).

Na rozdíl od metody *RoIPool*, zde nedochází k zaokrouhlování, což se umožňuje vyhnout nesouososti způsobené metodou *RoIPool*. Příkladem může být obrázek o velikosti 128x128 a mapa příznaků o velikosti 25x25. Každý pixel z původního obrázku logicky odpovídá 25/128 pixelům v mapě příznaků. Při výběru 15 pixelů z původního obrázku, se jednoduše vybere $15 \times 25/128 = 2.93$ pixelů. V *RoIPool* by se hodnota zaokrouhlila na 2 pixely, což by způsobilo mírnou nesouosost. *RoIAlign* se takovémuto zaokrouhlování zdárně vyhýbá. Pro přesnou představu o tom, co by mělo být na pixelu 2.93 se místo toho použije bilineární interpolace. Princip, kdy namísto metody *RoIPool* se obrázek předává přes *RoIAlign* tak, aby oblasti příznakové mapy vybrané pomocí *RoIPool* přesněji odpovídaly oblastem původního obrázku, lze vidět na 3.6.



Obrázek 3.6: Použití *RoIAlign* (Převzato z [10])



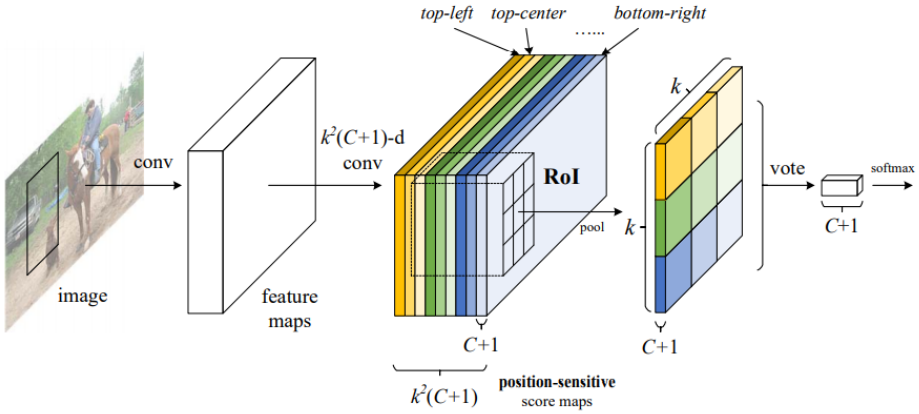
Obrázek 3.7: Příklad výstupu modelu Mask R-CNN (Převzato z [10])

Jakmile jsou tyto masky vygenerovány, Mast R-CNN je kombinuje s klasifikací a původními ohraničujícími rámečky z Faster R-CNN, čímž je docíleno přesné segmentace [10].

3.2 R-FCN

Jedná se o plně konvoluční detektor založený na principu získávání oblastí (*region-based fully convolutional network*), který sdílí téměř veškeré výpočty přes celý obrázek. Pro tradiční modely využívající návrhy oblastí výskytu daného objektu jako například Fast R-CNN jsou nejprve tyto návrhy vygenerovány pomocí sítě RPN. Poté je proveden *ROI pooling* a pro následnou klasifikaci a detekci se dále pokračuje přes plně propojené vrstvy. Celý proces po *ROI pooling* není mezi sebou nikterak sdílen, a tak zabírá spoustu času, což tyto modely značně zpomaluje. Navíc plně propojené vrstvy zvyšují počet spojení (parametrů sítě), což také zvyšuje jejich složitost.

V R-FCN se k získání návrhů oblastí stále používá síť RPN, ale na rozdíl od série R-CNN jsou odstraněny plně propojené vrstvy. Namísto toho je celá hlavní složitost přesunuta před *ROI pooling*, tak aby mohly být vygenerovány mapy skóre (*score maps*). Všechny návrhy, po *ROI pooling*, využijí stejné sady map k průměrnému hlasování (*average voting*), což je jednoduchý výpočet. Žádná vrstva, která by se dále musela učit, značně snižuje složitost.



Obrázek 3.8: V této ilistraci jsou vidět $k \times k = 3 \times 3$ tzv. *position-senstive score* mapy vygenerované plně konvoluční sítí. Pro každé políčko v $k \times k$ RoI oblasti se *pooling* provede pouze na jedné z $k \times k$ map (každá označená jinou barvou). (Převzato z [17])

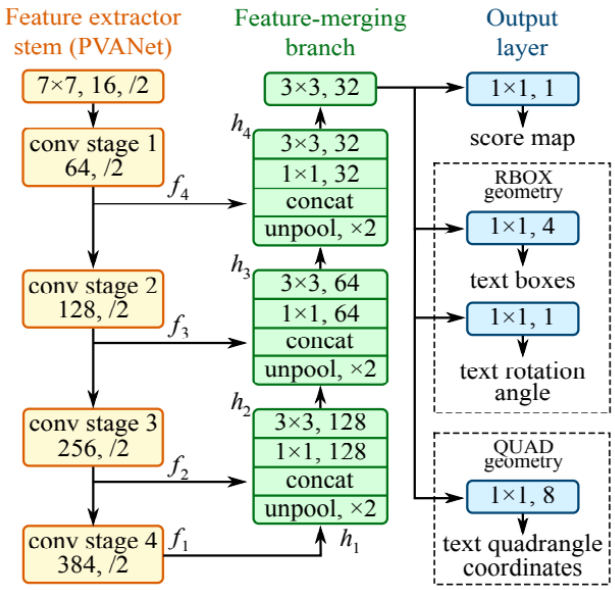
Výsledkem je R-FCN dokonce rychlejší než Faster R-CNN i s celkem obstojnou přesností. Hlavní myšlenku R-FCN ukazuje ilustrace 3.8 [17, 27].

3.3 EAST

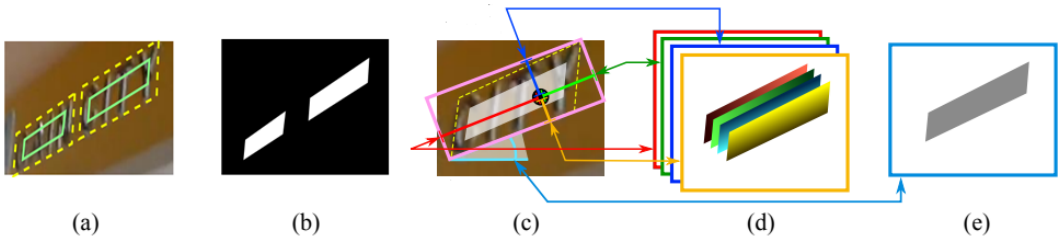
An Efficient and Accurate Scene Text Detector. Volně přeloženo, jako účinný a přesný detektor textu nacházející se v přírodní scéně. Jde o *deep learning* model založený na nové architektuře. Je schopen predikovat slova a řádky textu libovolné orientace a podle autorů může běžet až na 13FPS.

Vzhledem k tomu, že je tento model *end-to-end*, je možné se vyhnout výpočetně drahým algoritmům, které se používají u jiných textových detektorů (například algoritmy pro dělení slov). Jednoduše řečeno, model eliminuje zbytečné mezikroky za pomocí jedné neuronové sítě. Takto „zlehčený“ přístup umožňuje autorům se soustředit na tvorbu architektury sítě a návrhu chybových funkcí. Byť někdo může navrhovat, že se text, jako takový, podstatně liší od graffiti tagů a nemají spolu takřka nic společného (co do tvarů, jednoty velikosti apod.), většina tagů je ale čitelná a může se tak jednat pouze o jistou formu nového druhu písma. Fakt, že tento model je schopen rozeznávat text v reálném prostředí, tedy možnost rozeznat jiný barevný povrch určitého tvaru od jiného, je možno aplikovat i na problematiku graffiti tagů.

Klíčovou složkou navrhovaného algoritmu je model neuronové sítě (lze vidět na obrázku 3.9), který je trénován pro přímou predikci existence textu rovnou z obrázků. Model je plně konvoluční neuronová síť přizpůsobená pro detekci textu či řádků textu. Tím, jak už bylo zmíněno, se eliminují přechodné kroky. Další kroky po zpracování obrázku už pak pouze zahrnují práhování (*tresholding*) a NMS na predikovaných geometrických tvarech. Na obrázku můžeme vidět strukturu plně konvolučín sítě (*fully convolutional networks*). Algoritmus kopíruje obecný vzorec, ve kterém je obrázek vložen do plně konvoluční sítě (FCN) a následně je vygenerována tzv. *score* mapa textu a geometrie a to vše na úrovni pixelů viz 3.10. Mapa bodů (*score map* 3.10 (b)) obsahuje hodnoty pixelů v rozsahu od 0 do 1. Skóre značí, jak moc si je síť jistá s geometrickým tvarem predikovaným na stejném místě. Autoři experimentují se dvěma geometrickými tvary pro označení oblasti textu. Tzv.



Obrázek 3.9: Struktura plně konvoluční sítě pro detekci textu (Převzato z [33])

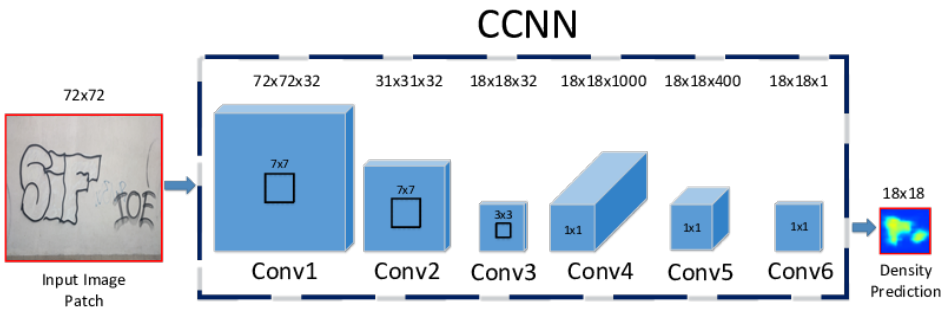


Obrázek 3.10: Proces generování rámečků: (a) čtyřúhelník kolem textu (žlutá čárkovaná) a zmenšený čtyřúhelník (zelená); (b) mapa skóre; (c) generování RBOX mapy; (d) 4 kanály vzdáleností každého pixelu k hranicím obdélníku; (e) úhel natočení (Převzato z [33])

RBOX a QUAD a pro každou z nich navrhují jinou chybovou funkci. Následně je na každou predikovanou oblast aplikováno prahování. Tvary, jejichž skóre je větší než přednastavená hodnota prahu, jsou považovány za validní a jsou uložené pro pozdější NMS. Výsledky NMS jsou brány jako finální výstup tohoto modelu [33].

3.4 Counting CNN

Prvotní záměr této neuronové sítě je adresován problému počítání určitého druhu objektu v obrázcích. Podle autorů je tento model schopen s velkou přesností odhadnou počet vozidel v dopravní zácpě nebo počet lidí v davu. CCNN je formulována jako regresní model, kde se síť učí mapovat výskyty jednotlivých objektů v obrázků na jejich odpovídající mapy hustoty. Autoři této práce přišli s návrhem nové architektury hluboké neuronové sítě zvané Counting CNN (CCNN), což je plně konvoluční neuronová síť schopná provádět přesnou



Obrázek 3.11: Proces zpracování vstupního obrázku na výslednou mapu hostoty (Upraveno dle [7])

regresi map hustoty přímo z částí obrázků (*image patches*). Oproti většiny *state-of-the-art* metod dále ukazuje, že hustota objektů může být odhadována bez potřeby použití perspektivních map nebo dalších geometrických informací. Oproti modelům, ze kterých tato síť vychází, má tento jednu velkou výhodu. Ke své činnosti nepotřebuje užití dvou, či více chybových funkcí, nebo použití nějakého dalšího regresoru. Mapa hustoty je přímým výstupem sítě, která je zároveň trénována pouze s jednou chybovou funkcí.

Architektura sítě je jednoduchá a opravdu minimalistická. Proto neuškodí, když bude tady rozepsána celá. Architekturu lze vidět na obrázku 3.11. Skládá se pouze z 6 konvolučních vrstev. První a druhá má filtry velikostí 7x7 s hloubkou 32 a jsou následovány *max-pooling* vrstvou s velikostí jádra 2x2. Třetí vrstva má pak filtry menší 5x5 s hloubkou 64 a je taktéž následována *max-pooling* vrstvou stejné velikosti, jako předchozí. Čtvrtá a pátá jsou tvořeny pouze filtry velikostí 1x1 s hloubkou 1 000, respektive 400. Je potřeba zmínit, že zde není použita ani jedna plně propojena vrstva. Díky těmto vrstvám vzniká plně konvoluční neuronová síť. Všechny předchozí vrstvy jsou ještě následovány vrstvami ReLU. Poslední šestá konvoluční vrstva se taktéž skládá z jednoho 1x1 filtru ale o hloubce 1. Tato vrstva je zodpovědná za vrácení odhadu mapy hustoty $D_{pred}^{(P)}$ pro vstupní obrázek P. Aby se tato síť byla schopna naučit lineární mapování výskytu objektu na mapu hustoty, musí být tato síť speciálně trénována k vyřešení takového regresního problému. Proto je poslední vrstva spojena s chybovou funkcí zvanou jako*Euclidean regression loss*.

$$l\left(\Omega\right) = \frac{1}{2N}\sum_{n=1}^{N}\left\|R\left(P_N|\Omega\right) - D_{gt}^{P_n}\right\|_2^2 \tag{3.1}$$

kde N odpovídá počtu vstupních obrázků v trénovací dávce a D reprezentuje *ground-truth* hustotu pro trénovací obrázek. Ω kóduje parametry sítě. Pro úpravu parametrů sítě se zde využívá populární SGD algoritmus.

Co se týče fáze predikce, prvně se extrahují části obrázků na základě hustoty objektů. Jak lze vidět na obrázku 3.11, CCNN se plní kousky obrázků, které jsou upraveny na pevnou velikost 72x72. Tyto obrázky jsou postupně zpracovány modelem CCNN, který predikuje odhad mapy hustoty pro každý z nich. Je potřeba si všimnout, že díky dvěma *max-pooling* vrstvám je výsledná mapa pouze 1/4 velikosti původního kousku obrázku (18x18). Proto se velikost všech predikovaných map musí změnit do původní velikosti. Poslední fáze spočívá ve spojení všech dílčích predikovaných map hustot dohromady [7].

Kapitola 4

Trénování detektorů

Hlavní metodou pro učení většiny zde použitých modelů detekce (kromě CCNN a EAST) je tzv. *transfer learning*. Je to metoda, kde model vyvíjen pro nějaký úkol je znovu využit jako výchozí bod pro úkoly další. Před trénované modely se tak stávají velice populárním přístupem v oblasti hlubokého učení. A to zejména proto, protože urychlují celý proces trénování a zvyšují výkonnost jednotlivých modelů, respektive pomáhá ke zlepšení zobecnění daných modelů. Modely v této práci obsahují velké množství parametrů a jejich trénování od základů by zabralo týdny, potažmo i měsíce [15]. Proto i zde trénování probíhá dle této šablony. Před trénovaný model je samozřejmě nutné přizpůsobit nebo zpřesnit na vlastních datech. V tomto případě to jsou konkrétně obrázky graffiti tagů.

Všechno trénování probíhalo v prostředí Google Colaboratory [1]. Je to prostředí, které nevyžaduje žádné nastavení a běží kompletně na cloudu přímo z webového prohlížeče. S Colaboratory je možno přistupovat k výkonným výpočetním zdrojům (nutno podotknout, že zcela zdarma). Je zde možno pracovat s GPU anebo také s TPU. Sítě byly trénovány za využití volby GPU, a to na grafické kartě Tesla K80 GPU (přístup asi k 11 GB RAM) a byla s ní možnost pracovat až 12 hodin bez přerušení.

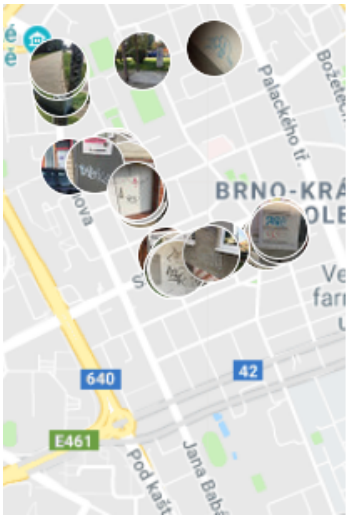
4.1 Dataset

Tato datová sada obsahuje obrázky, na kterých jsou zachyceny graffiti tagy v přírodních scénách, respektive v ulicích města Brna 4.1. Na každém z těchto obrázků se nachází minimálně jeden tag, maximálně však 18. Většina fotek byla pořízena ze vzdáleností 1 – 3 metrů ve snaze zachytit i přírodní prostřední, ve kterém se tagy nachází. Ve výjimečných případech byly zachyceny ze vzdáleností větších. Dataset byl zpracováván ve spoluprací se studentem Přemyslem Chovanečkem. 747 fotografií bylo přidáno z datasetu použitém při bakalářské práci Jana Pavlici [23], proto se rozlišení jednotlivých obrázků liší. K potřebám trénování byla všechna fota datasetu zmenšena na 20%, respektive 30% původní velikosti. Rozlišení obrázků posléze činí 612x816, 605x806, či 900x540.

Výsledný dataset obsahuje 2 258 fotografií. Fotografie jsou ve formátu JPEG. Data jsou dále rozdělena na data trénovací (1 808) a na data testovací (450). Celkový počet zachycených graffiti tagů činí 5 596.

Jednotlivé anotace byly tvořeny za použití programu LabelImg[2]. Anotace získané tímto programem jsou ve formátu xml. Takto jsou užívané v datasetech Pascal VOC. Pro tréno-

[1] https://colab.research.google.com
[2] https://github.com/tzutalin/labelImg



Obrázek 4.1: Ulice města Brna, kde byly zaznamenány tagy

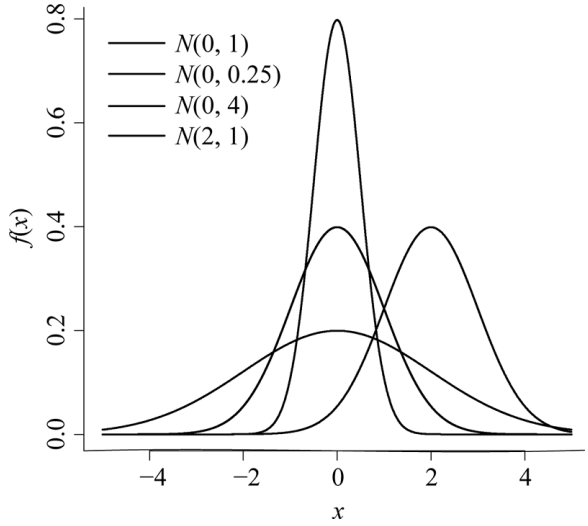
Obrázek 4.2: Rozmanitost datové sady

vání dalších modelů bylo však potřeba vytvořit i další patřičné vhodné anotace, které již používají jinou formu zápisu.

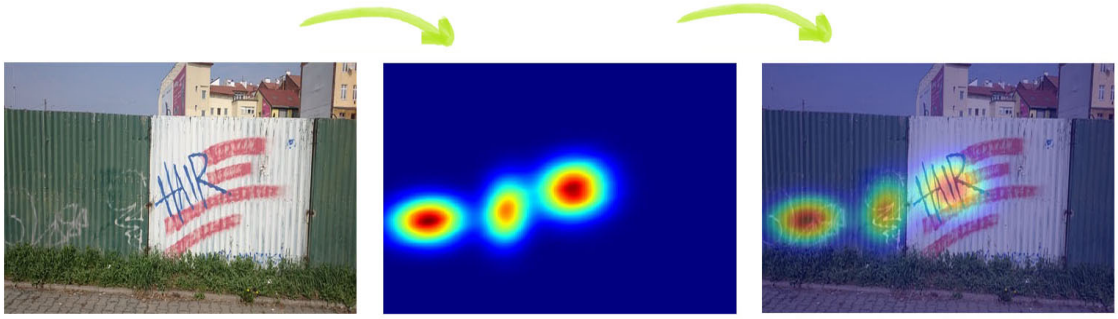
K natrénování modelu Mask-RCNN je spolu s ohraničujícími rámečky potřeba i sada masek graffiti tagů, viz v kapitole 4.2.3. Z níže uvedených důvodů je velikost datasetu použitelného pro tento model zmenšena a rozdělena na 289 testovacích dat a 373 dat trénovacích.

Pro účely trénování modelu EAST jsou anotace ve formátu txt. Každý nový řádek obsahuje souřadnice ohraničujícího rámečku. Jednotlivé čísla reprezentují pozice levého horního, pravého horního, pravého dolního a levého dolního rohu rámečku (vždy dvojice souřadnic x a y) následované údajem nesoucí název třídy, ke které přísluší.

Pro účely trénování *Counting CNN* je nutno si nachystat speciální dataset. Ten se v tomto případě skládá z obrázků graffiti tagů a jejich odpovídající *ground-truth* mapy hustoty (*heatmap*). Základem každé takovéto mapy je normální rozdělení neboli Gaussovo rozdělení. Důležitou vlastností jsou tedy hodnoty, které se symetricky shlukují kolem střední hodnoty a vytvářejí tak charakteristický tvar v podobě zvonu viz 4.3, čehož je využito právě při generování map hustoty. Normální rozdělení je popsáno parametry, které jsou označovány jako μ a σ^2 [2].



Obrázek 4.3: Ukázky hustot náhodných veličin s normálním rozdělením pro různé hodnoty parametrů (Převzato z [2])



Obrázek 4.4: Vygenerovaná mapa hustoty pro tři graffiti tagy

První z nich představuje střední hodnotu normálního rozdělení a druhý představuje rozptyl normálního rozdělení. Pro generování map hustoty, střední hodnotou je střed rámečku ohraničující daný graffiti tag. Hodnotu rozptylu bylo třeba odzkoušet tak, aby hodnoty rozdělení nepřeskakovaly velikost vymezující ohraničujícího rámečku (případně tak, aby se rozdělení nepřekrývala, což ale bylo s četnou hustotou tagů v obrazcích prakticky nemožné). Nejlépe se osvědčila hodnota šířky rámečku podělená hodnotou 4. Při zvolení jiných čísel se spousta hodnot rozdělení objevovala mimo danou výseč. Pro každý rámeček je vygenerováno $(width \times weight)/2$ bodů normálního rozdělení s výše určeným středem a odchylkou. Takovýto počet bodů se taktéž ukázal jako nejvhodnější, co se do pokrytí rámečku týče. Ukázku takto vygenerované mapy lze vidět na obrázku 4.4.

4.2 Tensorflow Object Detection API

Trénování tzv. *state-of-the-art* detektorů objektů úplně od základu může trvat několik dnů, i za použití několika GPU. Aby se trénování urychlilo, je dobré použít detektor, který je předen natrénován na jiném datasetu (například COCO [30]), a tak znovu použít některé z jeho parametrů k inicializaci modelu nového. *Tensorflow Object Dection API* je open source framework postavený na TensorFlow, který usnadňuje stavbu, trénování a nasazení jednotlivých modelů. API je součástí velkého oficiálního úložiště obsahující spoustu různých modelů. Například níže použité Faster R-CNN (sekce 4.2.2), Mask R-CNN (sekce 4.2.3) nebo R-FCN (sekce 4.2.4).

Tensorflow Object Detection API vyžaduje, aby všechna data datasetu, která jsou anotována, byla ve formátu TFRecord. TFRecord kombinuje veškeré obrázky datasetu a jejich odpovídající ohraničující rámečky (*labels*) do jediného velkého binárního souboru.

Ke trénování, potažmo testování je potřeba mít konfigurační soubor. Konfigurační soubor výhradně záleží na modelu, který je vybrán. V *Tensorflow Object Detection API* totiž, jak parametry modelu, tak parametry trénování, tak i parametry vyhodnocení, jsou všechny definovány jedním konfiguračním souborem. V tomto souboru jsou tedy zachyceny veškeré důležité parametry pro snadné definování modelu a jeho trénování. Příkladem můžou být parametry velikosti *batch*, *learning rate*, či definování počtu tříd. Konfigurační soubor tak tvoří poslední krok před spuštěním trénování. Během trénování se pravidelně (zhruba co 5 minut) ukládají kontrolní body (*checkpoints*). Kontrolní bod v nejvyšším kroku trénování se následně použije pro vygenerování *frozen inference* grafu. Tento graf pak obsahuje nově natrénovaný detektor.

4.2.1 Tensorboad

Důležitým prvkem trénování je sledování, jak dobře se daný model trénuje. K tomuto účelu lze třeba použít Tensorboard, který přichází společně s Tensorflow a je zodpovědný za spuštění webového rozhraní, ve kterém lze sledovat průběh trénování a testování. Stránka TensorBoard poskytuje různé informace a grafy. Důležitým grafem je tzv. *Loss graph*, který ukazuje celkovou chybu detektoru v čase.

4.2.2 Faster R-CNN

Faster R-CNN je momentálně považován za odzkoušenou špičku v oblasti detekce objektů (založené na *deep-learning*). Tento model nemusí být nejjednodušší nebo nejrychlejší modelem pro detekci objektů, ale je stále jedním z nejlepších, což dokazuje i tím, že jeho kostra tvoří základ mnoha dalších detektorů (například R-FCN). Nicméně jen na základě standardních metrik je složité vybrat, která architektura je pro danou problematiku ta nejvhodnější. Z toho důvodu byla zvolena architektura jak s ResNet extraktorem, tak i extraktorem Inception.

ResNet101 extraktor

Pro trénování byl vybrán před trénovaný model, jež byl trénován a optimalizován na datasetu MS-COCO. Proto byly veškeré parametry sítě nastavené na původních hodnotách. Snahou byla nechat model trénovat do té doby, dokud chyba (*loss*) nepřestane konzistentně klesat pod určitou hodnotu.

Obrázek 4.5: Maska grafitti tagu

Pro vyhodnocení finální detekce bylo použita metrika používaná na soutěžích COCO. Ta přidává i další statistiky jako je mAP na IOU prazích od 0.5 do 0.95, a statistiky *precision / recall* pro malé, střední a velké objekty. Výsledná hodnota pro tento model byla získána po zhruba 74 000 krocích, přičemž při 30 500. kroku byl snížen *learning rate* a pak znova při 40 660. kroku. Beze změny parametru byla síť trénována přes hranici 100 000 kroků, kde ale už zhruba po 30 000. kroku docházelo ke snížení přesnosti.

Inception Resnet V2 extraktor

Ke trénování toho modelu byl opět vybrán model, který byl již před trénován a optimalizován na MS-COCO datasetu. Avšak s výše nabytými poznatky bylo přistupováno k trénování tohoto modelu v podobném směru, jako tomu bylo v případě ResNet. Hodnoty *learning rate* tedy byly pozměněny ve stejném duchu. Konenčná hodnota byla získána po zhruba 45 000 krocích. Po tomto počtu kroků opět docházelo k postupnému snížení přesnosti detektoru.

4.2.3 Mask R-CNN

Trénování Mask R-CNN se oproti předchozím R-CNN modelům liší především tím, že do trénování je třeba zahrnout i masku objektu. Obvyklý model detekce objektu vyžaduje anotování obrázku za pomocí ohraničujícího rámečku. Ale pro tento model je vstupem i soubor PNG obsahující masku objektu 4.5.

S tímhle obrázkem binární masky je model schopen extrahovat obojí, jak souřadnice ohraničujícího rámečku, tak i lokaci pixelů daného objektu.

Nástroj, který je za tímto účelem použit je Pixel Annotation Tool[3]. Výstupem tohoto nástroje je soubor PNG ve formátu, který vyžaduje *Tensorflow Object Detection API*. V programu je třeba daný objekt „obarvit“. Je taktéž velice důležité obarvit i okolní prostředí a označit jej jako „*outside the region of interest*“. Tedy v podstatě to, co pro detekci není zajímavé. Pokud je zapotřebí mít masku co možná nejpřesnější, je třeba klást důraz na barvení hran objektů. V případě graffiti tagů je tato práce časově náročná a prakticky nemožná, vlivem například navzájem se překrývajících tagů. Bráno v potaz i to, že většina tagů je doplněna o další různé symboly (hvězdičky, uvozovky apod.), nebo to, že písmena některých tagů na sebe nenavazují, je k této činnosti přistupováno principem pokusu a omylu. Výsledné masky jsou tvořené tedy tak, aby pokrývaly celou oblast, kterou tag zabírá (nehledě například na mezery mezi jednotlivými písmeny). Je to asi jediný možný a vhodný způsob. S touto myšlenkou je tak trénovací dataset pro tuto síť značně zredukován.

[3] https://github.com/abreheret/PixelAnnotationTool

Obrázek 4.6: Ukázka modelem vygenerovaných masek

Tento model byl takto trénován zhruba po 45 000 kroků a bylo při tom dosaženo přesnoti 78.4% mAP. Tento výsledek zahrnuje pouze přesnost, která se týká ohraničujících rámečků. Pokud je ale tento model vyhodnocen pomocí metriky, která počítá IoU založené na masce namísto rámečků objektu, výsledná přesnost činí pouze 2% mAP. Příklad výsledně vygenerované masky lze vidět na obrázku 4.6.

4.2.4 R-FCN

Trénování modelu R-FCN se neslo v naprosto stejném duchu, jako ve výše zmíněných modelech *Tensorflow Object detection API*. To znamená, že byl použit již předem trénovaný model, u kterého byla pozměněna pouze hodnota *learning rate*. Trénování bylo opět zastaveno zhruba kolem 40 000. kroku.

4.3 CCNN

CCNN je implementováno za pomocí Kerasu. Keras je *open-source* uživatelsky přívětivá knihovna psaná v jazyce Python pro práci s neuronovými sítěmi. Konkrétně je tato síť implementována tzv. funkčním API Kerasu, které poskytuje flexibilnější způsob definování modelů. Modely jsou tvořeny vrstvami, které jsou spojovány za sebou v párech. Následně se definuje model jako takový a určí se, které vrstvy budou použity jako vstupní, respektive výstupní.

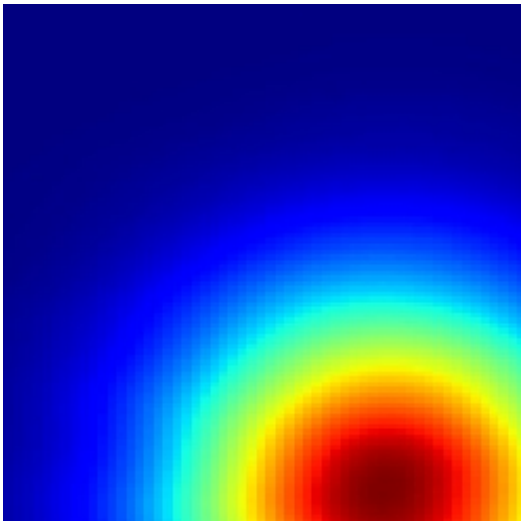
4.3.1 Přístupy trénování

Ke trénování této sítě se dá přistoupit několika způsoby, které se liší především zpracováním datasetu.

Prvním přístupem je sledování strategie, které využívají autoři této sítě. Podle Zhanga a spol. [6] jsou z každého trénovacího obrázku (158x238) extrahovány (bez normalizace) kousky o velikosti 72x72. Ke trénování se jich náhodně vygeneruje 800 (pro každý obrázek). Co se týče testovacího setu jsou kousky extrahovány za pomocí posuvného okénka s 50% překrytím. Odhad hustoty každého pixelu je získán průměrováním všech predikcí (těch kterých se překrývají), které se na hodnotě pixelu podílí. V případě této práce, v důsledku omezené velikosti paměti RAM, se pro trénovaní generuje pouze 15 náhodných kousků o velikosti zhruba 250x250 (snaha držet se poměru mezi velikostí obrázku a velikostí posuvného okénka uvedeného v práci autorů).

Další testovací sada je složena z kousků, které byly z každého obrázku extrahovány dle mřížky o velikosti 3x3. Jak už bylo výše zmíněno, pro testování CCNN se kousky extrahují

Obrázek 4.7: Náhodně generované kousky obrázků



Obrázek 4.8: Kus mapy hustoty pro červený rámeček

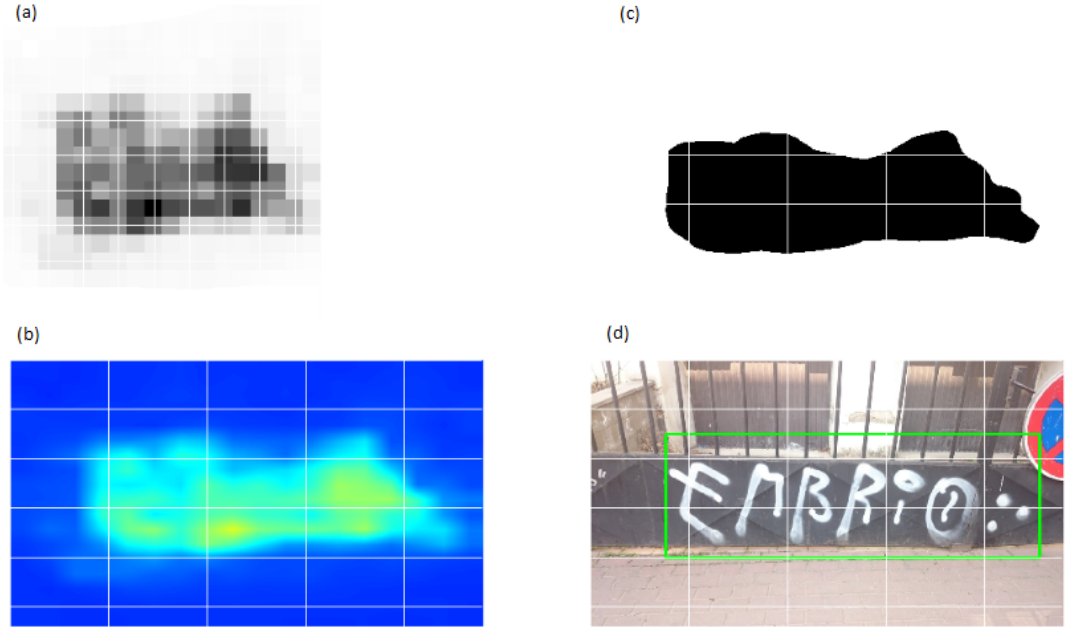
Obrázek 4.9: Vlevo: Obrázek s aplikovovanou maskou. Vpravo: Obrázek bez použití masky.

dle hustoty objektů v obraze. Z toho se nabízí další možnost, a to na trénovacích obrázcích vymazat („vymaskovat“) vše, co pro konkrétní detekci není zajímavé. Příklady takových to testovacích obrázků můžeme vidět na obrázku 4.9. Pro maskování byla vybrána stejná barva, jako podklad již zmíněných map hustot.

A v poslední řadě poskládat výsledný trénovací dataset pouze s nijak nezměněnými původními obrázky graffiti tagů. Všechny zmíněné úpravy testovací sady samozřejmě platí i pro jejich odpovídající *ground-truth* mapy hustoty.

4.3.2 Generování ohraničujících rámečků

Proces generace ohraničujících rámečku ilustruje obrázek 4.10. Výstupem CCNN sítě je mapa hustoty o velikosti 18x18 (lze na obrázku vidět v bodě (a)). Pro získání ohraničujících rámečků správné velikosti je zapotřebí zvětšit predikovanou mapu hustoty na velikost původního obrázku vloženého do neuronové sítě (ilustruje bod (b)). Pakliže byl původní obrázek rozdělen na vícero menších kousků, je třeba je následně seskládat do původních pozic, což ve výsledku poskládá predikovanou úplnou mapu hustoty rozložení jednotlivých tagů v obraze. Ohraničující rámečky lze následně získat převedením mapy hustoty do její černobíle podoby. Dále na základě zvoleného prahu převést již černobílou mapu na mapu binární (zobrazena v bodě (c)). Ta poté obsahuje pouze hodnoty 0 a 1. Díky tomuto pře-



Obrázek 4.10: Proces generování rámečku: (a) výstup modelu (18x18); (b) zvětšení na původní velikost; (c) binární mapa; (d) vykreslení finálního rámečku

vodu lze vykreslit ohraničující rámečky kolem vzniklých objektů. Výsledný rámeček je vidět na výše zmíněném obrázku v bodě (d).

Ve všech zmíněných přístupech se k trénování sítě přistupovalo stejně, měnily se pouze hodnoty parametru *batch size* a to na základě velikosti trénovacího datasetu.

Nejlepší výsledek při přístupu, kdy se použily celé obrázky, byl získán po dosažení 3 600. epochy trénování s nastavenou hodnotou *batch size* na 32. Při delším trénování nebyly výsledky nikterak lepší, ba naopak, po zhruba 9 000. epoše byly výsledné mapy hustoty nepoužitelné.

Pro přístup, kdy byl dataset rozdělen na základě mřížky, byl parametr *batch size* nastaven na hodnotu 128. Nejlepšího výsledku bylo dosaženo po zhruba 1 400. epoše trénování a nastavení velikosti posuvného okénka na 1/3 velikosti původního obrázku.

Pokud se dataset skládal pouze z obrázku, ve kterém byly „vymaskovány“ nepotřebné části, byla velikost *batch size* opět nastavena na hodnotu 32. Nejlepšího výsledku bylo dosaženo také po zhruba 3 000. epoše, kdy obrázky byly při následném testování vkládány do sítě posuvným okénkem velikosti 1/3 původního obrázku s 50% překrytím.

Klíčovým prvkem při trénování modelu posledním přístupem, kde z každého obrázku datasetu bylo vybráno 15 náhodných kousků, bylo nastavení právě počtu a velikosti daných kousků. Omezení počtu náhodných kousků spočívá především v nedostatku výpočetních zdrojů. Velikost okénka byla vybrána se snahou zachytit podstatnou informaci na obrázku. Pakliže by byly obrázky příliš malé, nebo naopak příliš velké, v obou případech by docházelo ke ztrátě informace, jež obrázky nesou. Trénování probíhalo s hodnotou *batch size* nastenou na 256 a bylo ukončeno zhruba po 600. epoše.

4.4 EAST

Pro účely trénování bylo využito modelu implementovaného za použití *tensorflow*. Implementace vychází z dokumentace autora modelu [33]. Narozdíl od originálního textu se zde používá pro extrakci vlastností obrázků ResNet-50 oproti původní PVANET. Dále je u této implementace změněna chybová funkce. Namísto *balanced cross entropy* je použita *dice loss*. A v poslední řadě byla pro označení textu naimplementována pouze část RBOX [34].

Pro spuštění trénování byly následovány původní hodnoty nastavení parametrů sítě. Velikost *batch* byla teda ponechána na hodnotě 14 a *learning rate* hodnota na 0.0001. Trénování tohoto modelu se od předchozích liší tím, že zde nebyl použit žádný, již předem před učený model. Opět zde byla snaha trénovat model tak dlouho, dokud hodnota chyby konzistentně klesala. Trénování bylo zastaveno po zhruba třech dnech trénování.

Kapitola 5

Dosažené výsledky

V této kapitole jsou vyhodnoceny výsledky získané z výše uvedených pokusů nalézt, co možná nejvhodnější způsoby automatické detekce graffiti tagů v přirozeném prostředí, respektive porovnat nynější „osvědčené“ přístupy, které se nám v dnešní době nabízí – tedy natrénovat vhodný detektor založený na neuronových sítích.

Kapitola je rozdělena do dvou částí, kdy v první části jsou vyhodnoceny tzv. *region based* detektory (vedoucí algoritmy detekce), jejichž hlavní náplní je právě detekce různých objektů. V druhé části pak přístupy, kde detekce různých objektů není podstatou, nicméně se jedná o zajímavé způsoby, jak je možno k této činnosti přistupovat.

5.1 *Region based* detektory

Jak už bylo řečeno, *region based* detektory, jako Faster R-CNN a R-FCN jsou jedny z nejlepších a nejpoužívanějších modelů detekce objektů. Další populární modely mají tendenci být poměrně těmto dvou podobné.

Co se týče přesnosti jednotlivých modelů, na základě výchozích hodnot [32], které byly získané při testování výkonosti modelů na podmnožině COCO datasetu, by měl model Faster-RCNN s architekturou Inception-Resnet-v2 dosahovat nejlepších výsledků. I v případě detekce graffiti tagů je tomu taky tak s naměřenou hodnotou mAP zhruba 83%. Následována stejným modelem s architekturou Resnet101, pro kterou byla naměřena přesnost 80.28% mAP. Model Mask-RCNN dosáhl přesnosti 78.4% mAP a plně konvoluční sít R-FCN přibližně 76.3% mAP.

Pokud jde o rychlost modelu, ani zde nedochází k žádnému rapidnímu odklonění od avizovaných hodnot. Nejrychlejším modelem je podle všech předpokladů R-FCN s naměřenou rychlostí 86 ms (zhruba 11 snímků za sekundu). Druhým nejrychlejším modelem je Mask-RCNN s rychlostí 111 ms (9 FPS), což je dáno jednodušší architekturou extraktoru příznaků, než které byly použity pro modely Faster-RCNN. Model s architekturou Resnet101 dosahuje rychlosti 170 ms (5.9 FPS), s architekturou Inception to činí 742 ms (1.3 FPS). Nutno podotknout, že se jedná o průměrné rychlosti. Každý obrázek byl zpracován sítí desetkrát a výsledný čas je tak průměrnou hodnotou. Přehledně je vše vidět v tabulce 5.1.

Pro zajímavost byla testovací sada rozdělena do pěti skupin podle počtu graffiti tagů nacházejících se na jednotlivých obrázcích. Početnost graffiti tagů v testovací sadě na jednotlivých snímcích ukazuje obrázek 5.3.

modely	rychlost (ms)	mAP (%)	doba trénování (h)
Faster-RCNN-Inception-resnet-v2	742	83.0	16
Faster-RCNN-Resnet101	170	80.3	8.3
R-FCN	86	76.3	7.5
Mask-RCNN	111	78.4	6.1

Tabulka 5.1: Přesnost/rychlost detektorů

modely	1 tag	2 tagy	3 tagy	4 tagy	5 + tagů
Faster-RCNN-Inception-resnet-v2	98.2	85.3	78.6	80.6	78.8
Faster-RCNN-Resnet101	95.2	81.3	79.3	71.0	75.7
R-FCN	95.7	77.1	75.9	69.9	69.6
Mask-RCNN	87.6	83.0	71.2	81.9	70.2
EAST	51.5	43.7	52.1	37.9	44.4
CCNN	13.5	5.5	9.8	8.4	4.7

Tabulka 5.2: Přesnost modelu pro konkrétní počet tagů

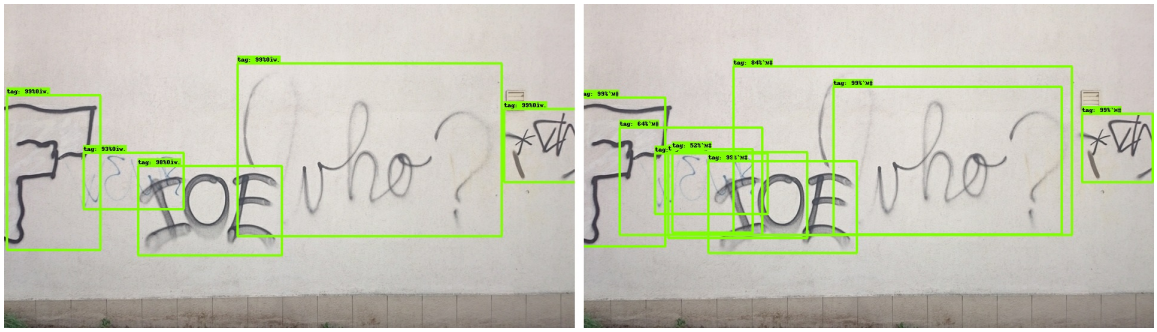
Dle tabulky 5.2 lze vidět, že přesnost jednotlivých modelů klesá s rostoucím počtem graffiti tagů v obraze. To je výhradně dáno menší velikostí přibývajících tagů s tím, že pokud je tagů na obraze více, je zde i větší pravděpodobnost, že se velká většina z nich bude navzájem různě překrývat. Obecně však platí, že model s architekturou Inception zvládá rostoucí náročnost s rostoucím počtem tagů ze všech modelů nejlépe. Model R-FCN se zdá být o něco „citlivější“ na změnu barvy pozadí a hledaného objektu 5.1, špatně však generalizuje jednotlivé graffiti tagy. Zejména pokud jsou tagy blízko u sebe nebo se navzájem překrývají. To má za následek ono snížení přesnosti tohoto modelu. Tento fakt ilustruje obrázek 5.2. Na levé straně se nachází výsledky detekce modelu Faster-RCNN, na pravé R-FCN.

Z rodiny modelů R-CNN dosáhl nejhorších výsledků model Mask-RCNN, kde kromě ohraničujících rámečků je výstupem sítě i maska tagů. Ačkoli na výsledných obrázcích vypadají masky jednotlivých tagů relativně obstojně (pokud se tagy nepřekrývají), výsledná přesnost (co do použití masek) byla pouhá 2% mAP. To je pravděpodobně dáno především nepravidelným tvarem graffiti tagů, jejich vzájemným překrýváním a malou učící množinou obrázků, na které byl model trénován 4.2.3. U většiny případů segmentace je maska hledaného objektu (respektive pixely objektu) ohraničena pixely, které spadají pod třídu jinou (například dvojice „auto-silnice“). V tomto případě zde žádná dvojice typu „tag-stěna“ není, neboť je jejich tvorba prakticky nemožná. Masky byly tvořeny tak, aby pokryly celou oblast graffiti tagu, což zahrnuje graffiti tag a část plochy za ním. Zvláště z tohoto důvodu jsou výsledné masky tak „nepřesné“.

5.2 Experimentální detektory

Jedním z mnoha dalších způsobů, jak pohlížet na graffiti tagy, bylo vnímat je jako formu písma. Proto zde byl otestován i model, jehož doménou je detekce textu v přirozeném prostředí. Nejvyšší dosažená přesnost modelu EAST na testovacích datech činila zhruba 45.58% mAP. Zajímavostí je, že průměrný čas predikce se pohybuje kolem 32 ms. Poslední řádek tabulky 5.2 značí, že si model vede podobným způsobem pro jakýkoliv počet tagů v obraze a drží si tak skrze různý počet tagů průměrnou hodnotu přesnosti 44% mAP. Pro-

Obrázek 5.1: Citlivost modelu R-FCN (vpravo)



Obrázek 5.2: Špatná generalizace graffiti tagů modelem R-FCN (vpravo)

blémem sítě v aktuálním bodě trénování je občasné generování tzv. *false positives*, případně nezaregistrování tagů, které do větší míry splývají s prostředím nebo které byly pořízeny pod větším úhlem 5.4.

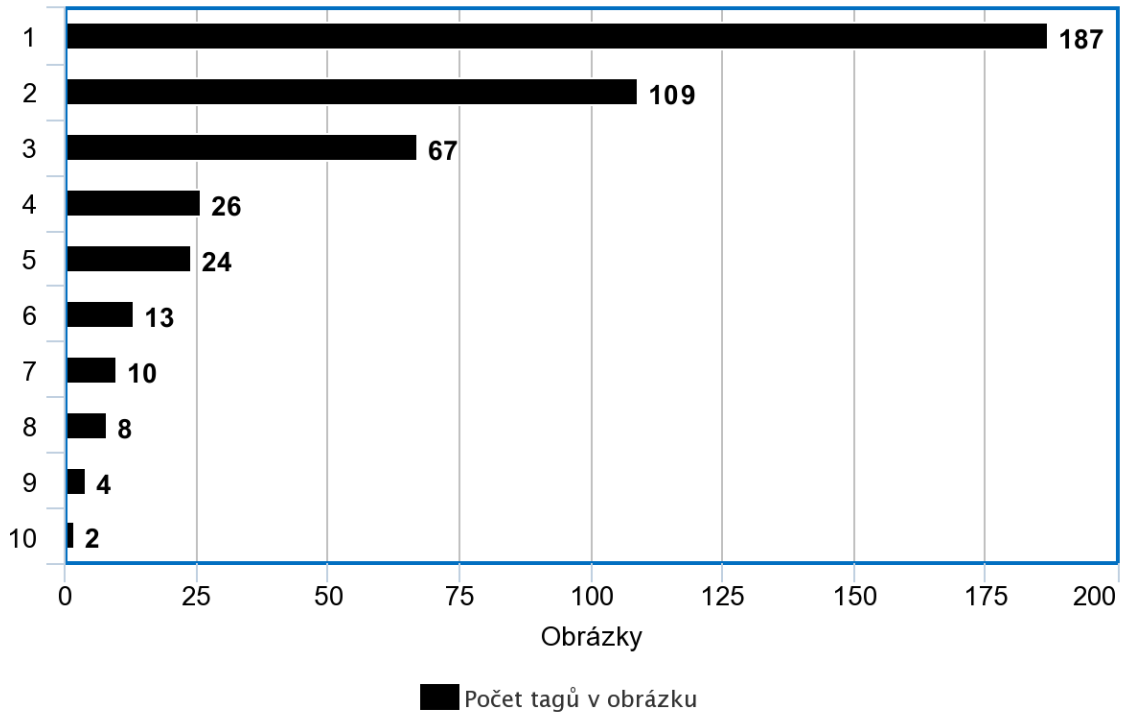
Jako poslední byl otestován model Counting CNN. Už podle názvu, je jeho hlavní činností určování počtu objektů v obraze, ne však jejich následná detekce. Na druhou stranu lze princip tohoto modelu aplikovat i na činnost detekce. Proces trénování byl rozdělen na několik částí, a to na základě datasatu, na kterých byl daný model trénován.

V první řadě se jednalo o dataset skládajících se z celých nepozměněných obrázků. Výsledná přesnost u tohoto přístupu činila zhruba 6.26% mAP. Pokud byl předán sítí obrázek, na kterém byl pouze jeden graffiti tag, model byl schopen reagovat s přesností 30% mAP, tedy nejlépe, ze všech ostatních v této sekci.

Dále byl dataset rozdělen na základě mřížky. V takovém případě byla nejvyšší naměřená hodnota 6.02% mAP.

Jednou z dalších nabízených možností bylo před trénováním „vymaskovat“ veškeré nepotřebné informace z obrázků pryč. Tímto způsobem ale nejlepší výsledek sáhá pouze k hranici 1% mAP a vesměs se dá říci, že většina výsledných map hustot (v tomto případě) byla naprosto nepoužitelná. Takto síť nějakým způsobem reagovala prakticky na každou hranu v obraze.

Důležitým prvkem při testování posledního přístupu, kde z každého obrázku datasetu bylo vybráno 15 náhodných kousků, bylo nastavení velikosti posuvného okénka. V tomto případě vybrat jednu danou velikost, která by byla schopna rozumně zachytit graffiti tagy všech velikostí, je prakticky nemožné. Čím je velikost okénka menší, tím více dochází ke kopírování hran jednotlivých písmen, namísto zahrnutí tagu, jakožto celku. Toto je vidět na obrázku 5.5. I když lze vidět, že použití posuvného okénka o velikosti 72x72 obstojně

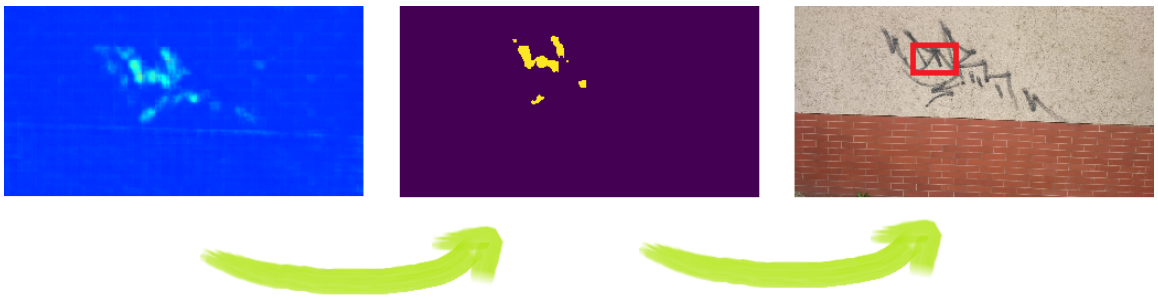


Obrázek 5.3: Rozdělení graffiti tagů v testovacím datasetu

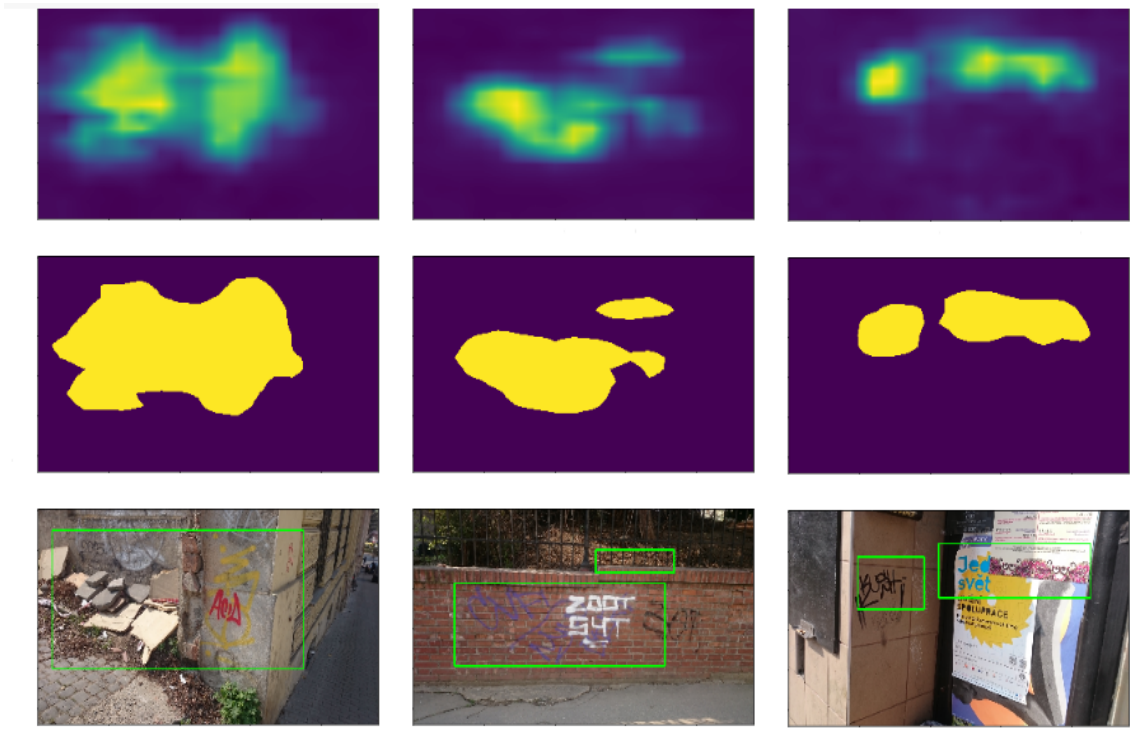
kopíruje tvar daného tagu, následné generování ohraničujícího rámečku je obtížné. Nicméně nejlepších výsledků bylo dosaženo při nastavení rozměrů okénka na 200x200 s výslednou přesností 6.85% mAP. Oproti prvního přístupu je tento schopný lépe reagovat na větší množství tagů v obraze. Průměrná hodnota predikce hustoty mapy (za předpokladu, že se neskládá z více částí) je zhruba 1.76 ms.

Hlavním problémem tohoto modelu v situaci, kdy je aplikován pro účely detekce graffiti tagů v obrazu je ten, že obrázek vkládaný do sítě je zmenšen na rozměry 72x72. Při tomto zmenšení, na rozdíl od obrázků aut, či lidí v davu (což jsou hlavní předměty zkoumání sítě CCNN), dochází ke ztrátě informací, kdy jsou například menší tagy prakticky vymazány. Velkým dílčím problémem je v první řadě nesourodost a odlišnost tvarů tagů. Graffiti tag, na rozdíl od objektu auta nemá pevnou strukturu nebo tvar. Prakticky se jedná o tmavou zeď odlišenou od druhé zdi jiné barvy. Z toho důvodu síť jen obtížně reaguje na graffiti tagy, které jsou například tenké, roztáhlé, či které zkrátka nejsou nijakým způsobem zhuštěné. Druhým neméně podstatným problémem je větší počet graffiti tagů v otázce tvorby map hustot. Už při generování *ground-truth* map hustot je velice obtížné je vygenerovat tak, aby se navzájem dílčí části nedotýkaly (většina povrchů obsahuje více tagů, které na sebe různě navazují, či se překrývají). Způsobem, jakým poté dochází ke generování ohraničujících rámečků je v podstatě nereálné zachytit vícero tagů, které jsou si buď to blízko a nebo se navzájem překrývají (ty ale tvoří podstatnou část datasetu) 5.6.

Obrázek 5.4: Příklady detekce modelu EAST



Obrázek 5.5: Proces generace rámečku při malém posuvném okénku



Obrázek 5.6: Problém přístupu CCNN při snaze zpracovat obrázek s větším počtem tagů

5.3 Shrnutí výsledků

Z výše uvedeného vyplývá, že jasně nejpřesnějším modelem je Faster-RCNN s architekturou Inception-Restnet-V2, která dosahuje až 83% přesnosti. Na druhé straně je možno v rámci testovaných modelů tento označit za nejpomalejší. Optimální rovnováhu mezi rychlostí a přesností nabízí model Faster-RCNN s architekturou Resnet101, jehož přesnost je s předchozím modelem srovnatelná, přičemž je ale 4x rychlejší. Nejrychleji však predikuje výstup síť CCNN, ale s její nízkou přesností není ani zdaleka konkurence schopná. Do konkurence schopné nespadá ani druhý nejrychlejší model EAST, jehož přesnost nepřekračuje ani hranici 50% mAP. Z toho důvodu se dá za nejrychlejší detekčně konkurenčně schopný považovat plně konvoluční model R-FCN. Mask R-CNN se z pohledu masek nejeví jako vhodný model pro detekci graffiti tagů (zejména právě kvůli typickým vlastnostem tagů), byť překvapil svou relativně vyšší přesností ohraničujících rámečků.

Kapitola 6

Závěr

Cílem této práce bylo analyzovat a porovnat různé přístupy počítačového vidění se záměrem automatické detekce graffiti tagů v obraze. Výsledkem práce je nalezení a vyhodnocení některých možných přístupů. Jinými slovy ilustrovat, jak jednotlivé přístupy zvládají náročnou obtížnost detekce graffiti tagů v přirozených scénách, jež nabývají specifických vlastností (tvar, struktura, množství), případně nastínit čtenáři přístupy, které se sice neřadí do kolonky „*state-of-the-art*“ algoritmů, ale nabízejí zajímavé způsoby detekce. Spolu s tím vznikl při práci i dataset čítající na 2 258 fotografií, které zachycují grafitti tagy v ulicích města Brna. Celkový počet zachycených graffiti tagů v tomto datasatu pak činí 5 596.

Testované detekční systémy byly vybrány na základě porovnání s dalšími detekčními systémy, se snahou v první řadě vybrat již ty „osvědčené“ na poli detekce objektů a vyzkoušet, zda jsou tyto přístupy vhodné i pro problematiku detekce graffiti tagů v obraze. V druhé řadě byl kladen důraz na snadnou implementaci jednotlivých modelů, což také vedlo k použití *Tensorflow Object Detection API*. Mezi další přístupy k otestování byly vybrány ty, které v této sféře odzkoušené zatím nebyly, nicméně by mohly, vzhledem ke svým principům, vést k dobrým výsledkům.

Na základě výše uvedeného pozorování byly osvědčené modely, konkrétně tedy modely z rodiny algoritmů založených na generování oblastí (*region based*), schopné detekovat tagy s přesností nad 76% mAP. V nejlepším případě byla dosažena přenost 83% mAP (Faster R-CNN). Experimentální metody nedosáhly srovnatelných výsledků, alespoň však poukázaly na zajímavé způsoby detekce objektů a s provedením určitých změn, by mohly být i zajímavým předmětem dalšího zkoumání.

Je třeba zmínit, že vybrané testované modely pochází z let 2016 a 2017. Oblast detekce objektů, s použitím konvolučních neuronových sítí, se každým dnem vyvíjí, proto je zde předpoklad dosažení daleko přesvědčivějších výsledků za použití nejnovějších způsobů detekce založených na novějších modelech a jejich architekturách, či principech. Takovým případem může být například RetinaNet nebo jeden z nejnovějších experimentů roku 2019 TridentNet, který drží v globálním žebříčku detekce objektu datasetu COCO první příčku [3].

Literatura

[1] Anirudh Sharma: *Anirudh Sharma's answer to 'What is an intuitive explanation for neural networks?'*. [Online; navštíveno 17.04.2019].
URL https://www.quora.com/What-is-an-intuitive-explanation-for-neural-networks/answer/Anirudh-Sharma-555

[2] Anon: *Normální rozdělení pravděpodobnosti*. [Online; navštíveno 17.04.2019].
URL http://portal.matematickabiologie.cz/index.php?pg=aplikovana-analyza-klinickych-a-biologickych-dat--biostatistika-pro-matematickou-biologii--nahodna-velicina-rozdeleni-pravdepodobnosti-a-realna-data--normalni-rozdeleni-pravdepodobnosti

[3] Anon: *Object Detection on COCO*. [Online; navštíveno 26.04.2019].
URL https://paperswithcode.com/sota/object-detection-on-coco

[4] Anon: *Throw-up*. [Online; navštíveno 16.04.2019].
URL http://graffiti.wikia.com/wiki/Throw-up

[5] Christoph Berger: *Perceptrons - the most basic form of a neural network*. [Online; navštíveno 27.04.2019].
URL https://appliedgo.net/perceptron/

[6] Cong Zhang, Hongsheng Li, Xiaogang Wang a Xiaokang Yang: *Cross-scene Crowd Counting via Deep Convolutional Neural Networks*. [Online; navštíveno 17.04.2019].
URL http://www.ee.cuhk.edu.hk/~xgwang/papers/zhangLWYcvpr15.pdf

[7] Daniel Oñoro and Roberto J López-Sastre: *Towards Perspective-Free Object Counting with Deep Learning*. [Online; navštíveno 17.04.2019].
URL https://www.researchgate.net/publication/306118084_Towards_Perspective-Free_Object_Counting_with_Deep_Learning

[8] Daniel Shiffman: *Chapter 10. Neural Networks*. [Online; navštíveno 17.04.2019].
URL https://natureofcode.com/book/chapter-10-neural-networks/

[9] Daphne Cornelisse: *An intuitive guide to Convolutional Neural Networks*. [Online; navštíveno 17.04.2019].
URL https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

[10] Dhruv Parthasarathy: *A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN*. [Online; navštíveno 17.04.2019].
URL https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

[11] Dishashree Gupta: *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* [Online; navštíveno 17.04.2019].
URL https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/

[12] Harsh Pokharna: *The best explanation of Convolutional Neural Networks on the Internet!* [Online; navštíveno 17.04.2019].
URL https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

[13] Harsh Singhal: *Convolutional Neural Network with TensorFlow implementation.* [Online; navštíveno 17.04.2019].
URL https://medium.com/data-science-group-iitr/building-a-convolutional-neural-network-in-python-with-tensorflow-d251c3ca8117

[14] Jackson Waschura: *What is the difference between a Convolutional Neural Network and a regular Neural Network?* [Online; navštíveno 17.04.2019].
URL https://ai.stackexchange.com/a/5566

[15] Jason Brownlee: *A Gentle Introduction to Transfer Learning for Deep Learning.* [Online; navštíveno 17.04.2019].
URL
https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[16] Jeremy Jordan: *Common architectures in convolutional neural networks.* [Online; navštíveno 17.04.2019].
URL https://www.jeremyjordan.me/convnet-architectures/#inception

[17] Jifeng Dai, Yi Li, Kaiming He and Jian Sun: *R-FCN: Object Detection via Region-based Fully Convolutional Networks.* [Online; navštíveno 17.04.2019].
URL https://arxiv.org/abs/1605.06409

[18] Jonathan Hui: *mAP (mean Average Precision) for Object Detection.* [Online; navštíveno 17.04.2019].
URL https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[19] Jordi Torres: *Learning process of a neural network.* [Online; navštíveno 17.04.2019].
URL https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7

[20] Lars Hulstaert: *A Beginner's Guide to Object Detection.* [Online; navštíveno 17.04.2019].
URL https://www.datacamp.com/community/tutorials/object-detection-guide

[21] Maurice Peemen: *Convolutional Neural Network (CNN).* [Online; navštíveno 17.04.2019].
URL https://developer.nvidia.com/discover/convolutional-neural-network

[22] Moses Olafenwa: *Object Detection with 10 lines of code.* [Online; navštíveno 17.04.2019].
URL https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606

[23] Pavlica, B. J.: *Detekce graffiti tagů v obraze*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2016.

[24] Petr Waleczko: *Sprejeři loni způsobili škody za desítky milionů. Úspěšní při hledání vandalů byli ve Zlínském kraji*. [Online; navštíveno 25.04.2019].
URL http://www.svoboda.info/zpravy/cerna-kronika/na-dopadeni-sprejeru-je-vypsana-odmena-20-000-korun/

[25] Rohith Gandhi: *R-CNN, Fast R-CNN, Faster R-CNN, YOLO – Object Detection Algorithms*. [Online; navštíveno 17.04.2019].
URL https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[26] Satya Mallick: *Image Recognition and Object Detection : Part 1*. [Online; navštíveno 17.04.2019].
URL https://www.learnopencv.com/image-recognition-and-object-detection-part1/

[27] Sik-Ho Tsang: *Review: R-FCN – Positive-Sensitive Score Maps (Object Detection)*. [Online; navštíveno 17.04.2019].
URL https://towardsdatascience.com/review-r-fcn-positive-sensitive-score-maps-object-detection-91cd2389345c

[28] Svobodová, D.; Kavalová, E.: *O jazyce autorů graffiti, I.* [Online; navštíveno 16.04.2019].
URL http://nase-rec.ujc.cas.cz/archiv.php?art=7541

[29] Tim Stone: *Graffiti: Art of the tag*. [Online; navštíveno 26.04.2019].
URL https://www.abc.net.au/news/2016-02-04/the-art-of-graffiti-tagging/6959396

[30] Tsung-Yi Lin: *Common Objects in Context*. [Online; navštíveno 17.04.2019].
URL http://cocodataset.org/#home

[31] Vikas Gupta: *Understanding Feedforward Neural Networks*. [Online; navštíveno 17.04.2019].
URL https://www.learnopencv.com/understanding-feedforward-neural-networks/

[32] Vivek Rathod and Neal Wu: *Tensorflow detection model zoo*. [Online; navštíveno 23.04.2019].
URL https://ct24.ceskatelevize.cz/domaci/2733389-sprejeri-loni-zpusobili-skody-za-desitky-milionu-uspesni-pri-hledani-vandalu-byli-ve

[33] Xinyu Zhou: *EAST: An Efficient and Accurate Scene Text Detector*. [Online; navštíveno 26.04.2019].
URL https://arxiv.org/pdf/1704.03155.pdf

[34] Xinyu Zhou: *EAST: An Efficient and Accurate Scene Text Detector*. [Online; navštíveno 17.04.2019].
URL https://github.com/argman/EAST

[35] Zacpalová, P.: *Graffiti a street art v Brně*. Diplomová práce, Masarykova univerzita, Filozofická fakulta, 2008.