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

MOBILNÍ APLIKACE PRO DETEKCI GRAFFITI TAGŮ

GRAFFITI TAGS DETECTION MOBILE APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PŘEMYSL CHOVANEČEK**
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: **Chovaneček Přemysl**
Program: Informační technologie
Název: **Mobilní aplikace pro detekci graffiti tagů**
Graffiti Tags Detection Mobile Application
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku obecné detekce objektů a detekce textu.
2. Vyberte vhodné metody a navrhněte řešení problému detekce graffiti tagů v obraze pro použití na mobilních zařízeních.
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

Práce se zaměřuje na rozpoznávání objektů v obraze za použití principů umělé inteligence. Řeší detekci podpisů autorů v oblasti umění zvané graffiti. Zabývá se základní problematikou této oblasti a dále poukazuje na použití počítačového vidění a jeho následnou, praktickou aplikaci na mobilních zařízeních, konkrétně tedy na platformě Android. Zvolenou neuronovou sítí byl model `ssdMobileNet_v2`. Naučený model dosahuje přesnosti mAP 73.5% při hodnotě IoU 0.6. Po provedení procesu kvantizace byla pak přesnost snížena na 68.5%. Samotná mobilní aplikace poskytuje real-time detekci a několik dalších potřebných funkcí pro lokalizaci a sběr dat.

Abstract

Thesis focuses on the object recognition of images, using the principles of artificial intelligence. It solves the signature detection of authors in the field of art called graffiti. It concerns about basic problematic of this field, it also points to the use of computer vision followed by practical application on mobile devices, specifically on the Android platform. The selected neural network models was the `ssdMobileNet_v2`. The trained model achieves mAP accuracy of 73.5% meanwhile the IoU was set to 0.6. After the quantization process, the accuracy was reduced to 68.5%. The mobile application provides real-time detection and several other necessary functions for localization and data collection.

Klíčová slova

Počítačové vidění, graffiti, mobilní aplikace, neuronové sítě, TensorFlow, Android

Keywords

Computer vision, graffiti, mobile application, neural networks, TensorFlow, Android

Citace

CHOVANEČEK, Přemysl. *Mobilní aplikace pro detekci graffiti tagů*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Špaňhel

Mobilní aplikace pro detekci graffiti tagů

Prohlášení

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

.....
Přemysl Chovaneček
1. května 2019

Poděkování

V prvé řadě bych rád poděkoval svému vedoucímu, Ing. Jakubovi Špaňhelovi, za věnovaný čas a pomoc při řešení problémů. Dále bych chtěl poděkovat své rodině a přátelům, kteří mě stále podporovali.

Obsah

Úvod

Během posledního půl století došlo k masivnímu rozvoji a vývinu v různých oblastech techniky. Dnešní průmysl, služby, ale také běžný život jsou postaveny na využívání inovativních a chytrých technologií. V době chytrých telefonů a rychle postupujícího vývoje technologií se obor počítačového vidění stává velmi populární. V praxi se s jeho využitím setkáváme denně – ať už jen při odemykaní telefonu, návštěvy lékaře či parkování našeho vozu před domem. Další případy využití počítačového vidění jsou například ve složkách bezpečnostních, pro rozpoznání automobilů, identifikaci a podobně.

Bakalářská práce Mobilní aplikace pro detekci graffiti tagů je založena a inspirována právě využitím principů počítačového vidění. Práce se zaobírá sběrem a lokalizací dat, poznáním moderních neuronových sítí, jejich aplikací na konkrétních datech a v poslední řadě implementací jednoduché mobilní aplikace, která s touto síti pracuje. Konkrétně se tedy mobilní aplikace zabývá detekováním graffiti tagů v reálném čase a dalšími základními prvky.

Práce se skládá ze šesti kapitol. První dvě kapitoly jsou obecného a teoretického charakteru. Kapitola první vysvětluje elementární pojmy pro pochopení kontextu. Dále je poskytnuto uvedení do problematiky, jako popis oblasti graffiti, oblasti zpracování obrazu, počítačového vidění a vysvětlení principu fungování neuronových sítí.

V kapitole třetí se dále konkretizuje použití neuronových sítí na základě požadavků. Jsou zde také popsány základní metriky od kterého se výsledky sítí odvíjí a také optimalizace v podobě kvantizace.

Od čtvrté kapitoly práce nabírá již praktický směr, zabývající se sběrem dat a především jejich využitím. Kapitola dále obsahuje popis použitých nástrojů a ostatních praktických aspektů vývoje.

Pátá kapitola se věnuje aplikaci – řeší stanovené cíle, návrh uživatelského rozhraní a popisuje způsob implementace. V druhé polovině této kapitoly jsou popsány výsledné experimenty a testování na různých zařízeních, testování co se týče uživatelské přívětivosti.

Kapitolou šestou je závěr shrnující bakalářskou práci a případná možná budoucí řešení.

Kapitola 1

Graffiti, zpracování obrazu a počítačové vidění

Kapitola detailněji popisuje základní problematiku práce. V úvodních podkapitolách postupně rozebírá obsažené pojmy, seznamuje čtenáře s oblastí graffiti. Druhá polovina kapitoly rozebírá metody a principy zpracování obrazu a počítačového vidění.

1.1 Elementární pojmy

Podkapitola popisuje základní pojmy užívané v následujících odstavcích a kapitolách. Jejich detailnější vysvětlení je obsaženo postupně v jednotlivých kapitolách.

Trénování, učení Trénováním je rozuměn proces, který bude aplikován nad nasbíraným souborem dat. Výstupem bude funkční neuronová síť, která (v našem případě) bude schopna detekovat graffiti tagy v obraze.

Síť, model Pojmy síť a model představují libovolný druh neuronové sítě, který je následně použit. Modelů existuje celá řada, ať už zaměřených na přesnost, či rychlost.

Dataset Datasetem lze označit kolekci nasbíraných dat. V případě této práce se jedná o kolekci fotografií tzv. graffiti tagů.

Intersection over Union (IoU) IoU úzce souvisí s následně vysvětleným pojmem pro měření přesnosti. Tímto pojmem se rozumí míra správnosti predikce ku skutečné pozici objektu.

Mean average precision (mAP) Další často vyskytující pojmem je metrika popisující výslednou přesnost našeho modelu. Přesnost je často uvedena v procentech. Později se práce zaobírá porovnáváním našeho modelu na základě proměnných parametrů. V souvislosti s IoU souvisí pojem mAP@0.7. Ten bude znamenat přesnost mAP s hodnotou IoU 0.7.

Bounding Box Bounding Box je nejzákladnější pomocný tvar pro detekci objektů v obraze. Zjednodušeně řečeno – jedná se o 2D čtverec či obdélník, jehož vrcholy reprezentují pozici objektu, který nás zajímá. Co se týče času a využití zdrojů, jedná se o nejlepší způsob pro anotace objektů. V práci se také často vyskytuje slovo ohraničení či rámec.

Obrázek 1.1: Ukázka graffiti tagů.

1.2 Graffiti

Pojem graffiti je dnes úzce spjat s uměním. V podstatě se jedná o malby, obrazy, zkrátka vizuální projevy, které lidé tvoří už od nepaměti. Populární rozepří posledních dekád bývá otázka, zda se jedná o umění, či vandalismus. Při ohlédnutí do historie se ukazuje, že tato forma projevu byla brána jako umění. Postupem času se však stala například jedním z charakteristických znaků hip–hopu či symbolem různých gangů po celém světě. Zároveň se na ní v mnoha ohledech často nahlíží jako na trestný čin.

Graffiti tagy

Tagy, jejichž detekcí skrze počítačové vidění se tato práce zabývá, jsou nedílným prvkem oblasti graffiti. Slovem **tag** se označuje podpis autora, jinak řečeno writera. Podpisy lze vidět na obrázku 1.1. Zvykem writerů bývá značkování různých území a teritorií jejich unikátním podpisem. Podpisem je obvykle nějaká přezdívka, seskupení písmen či číslic. Rozsáhlá a umělecky náročnější díla bývají nazývána **piece** [22].

Při prozkoumávání a sbírání podkladů pro bakalářskou práci se ukázalo, že existuje několik writerů, kteří si zakládají na udržení svých uměleckých jmen. Při mapování různých čtvrtí a úseků města je možno vidět snahu o zachování kontroly nad tímto územím. Lze zde pozorovat určité napodobování chování jiných kultur, avšak například ve Spojených státech je tato kontrola území často spojena s organizovaným zločinem.

V Praze, hlavním městě České republiky, existuje populární místo zvané Lennonova zeď. Ta je plná nápisů stylem odpovídajícímu graffiti tagům. Místo navštěvuje denně spousta turistů.

V roce 2014 došlo k přetření zdi na bílo street–artovou skupinou Pražská služba. Zeď obsahovala pouze nápis *„WALL IS OVER!“* – Konec zdi. Situaci lze pozorovat na obrázku 1.2. Paradoxně majitel zdi označil tento čin za vandalismus a podal trestní oznámení, které bylo později staženo. Během několika dní byla zeď opět plná tagů a nápisů.

Obrázek 1.2: Lennonova zeď v Praze. Obrázek převzat z článku [19].

1.3 Digitální zpracování obrazu

Další podkapitoly se věnují teoretickému popisu základních principů a disciplín zpracování obrazu. Ačkoli náplň práce spadá spíše do oblasti počítačového vidění, je vhodné si uvést jednoduchou, základní charakteristiku. Popisuje také odvození, využití a cíle v oblasti počítačového vidění. Dále jsou v kapitole vysvětleny základní rozdíly mezi zpracováním obrazu a počítačovým viděním. Beze sporu se oblasti často prolínají a doplňují, ovšem nejedná se zcela vždy o totéž.

Digitální zpracování obrazu je kategorie obecně spadající pod zpracování signálu. Laicky by se dalo říct, že se jedná o úpravu obrazu, odstranění šumů, nežádoucích jevů či jeho vylepšení. Disciplínu zpracování obrazu lze popsat v několika krocích.

Na vstup bývá přiveden požadovaný obraz ke zpracování. Následuje sada operací, či algoritmů pro modifikování či úpravy. Vzápětí je potom výstupem opět obraz nebo také konkrétní sada příznaku či informací týkající se právě tohoto vstupního obrazu, které nás zajímají. Mezi fundamentální řešené problémy při digitálním zpracování obrazu patří algoritmy například pro již zmíněné odstranění šumu, detekci hran, segmentace obrazu či různé transformace.

Odstranění šumu

Odstranění šumu je využíváno v obraze pro jeho zjemnění či redukci přítomných nečistot. Na vstup (obraz) jsou aplikovány filtry, pomocí kterých lze docílit požadovaného výsledku. Mezi nejběžnější filtry řadíme dolní propust, horní propust, pásmovou zádrž, medián a podobně.

„Více sofistikované metody modelují obraz na základě analýzy místních struktur, a na jeho základě odlišují signál od šumu. Použitím prvotní analýzy obrazových dat a vyhledání lokálních struktur, jako čar a hran, a následným ovládáním filtru na základě lokálních informací lze dosáhnout lepšího odstranění šumu s nižším poškozením signálu oproti plošnému použití filtrů.“ [6]

Segmentace obrazu

Jak již vypovídá název disciplíny, segmentace obrazu pracuje s cílem rozdělit obraz na segmenty. Často tomu bývá za účelem vyčlenění objektů, které nás v obraze zajímají. Zbylé

části zůstávají ignorovány. Pro segmentaci bývá často využita metoda prahování, které pomáhá identifikovat a roztřídit objekty [7]. Segmentace se svým způsobem podobá detekci, avšak v případě této práce je využit proces trénování a aplikování neuronových sítí.

Detekce hran

Detekce hran pracuje s okolím, ve kterém existují rozdílné hodnoty jasu. Tento přístup opět využívá techniku prahování. Je nutno určit, při jaké hodnotě prahu rozhodnout, zda se daný pixel již považuje za hranou, či nikoli. Za nejznámější a nejpoužívanější detektor hran se považuje Canny edge detector. Zjištění těchto hran potom dále rozvíjí jiné oblasti zpracování obrazu.

Pravidelně dochází ke kombinaci různých metod, například odstranění šumu je důležité pro lepší detekci hran a podobně.

1.4 Počítačové vidění (Computer vision)

Lidé využívají zrak každý den a život bez něj si stěží dovedou představit. Tento smysl se bere jako samozřejmost – člověk dokáže vidět trojrozměrně, zaostřovat objekty, rozpoznávat, klasifikovat, či odůvodnit určité chování. V informatice, ale v mnoha dalších, na moderní technologii stavějících oborech, se poslední dobou rozmáhá užití počítačového vidění. Jedná se o snahu přiblížení veškerých možností počítačem zpracovaného obrazu směrem k úrovni, na které s těmito možnostmi dokáže pracovat člověk. Následující odstavce vysvětlují a popisují základní disciplíny a úlohy počítačového vidění.

Odvětví biologie a ostatních přírodních věd přinesly do informatiky mnoho schémat a návrhů algoritmů, jako je například mravenčí kolonie či formace hejna. Pozornost bude dále věnována oblasti vidění. Mezi tímto biologickým a počítačovým viděním zcela jistě existuje mnoha paralel, díky kterým je počítačové vidění inspirováno a rozvíjeno.

Člověk tyto aspekty vidění dokáže automaticky – počítač nikoli. Je třeba tohoto postupně docílit právě pomocí metod získávání, analýzy a zpracování obrazu. Tato oblast je úzce spjatá s umělou inteligencí.

Vývoj a užití počítačového vidění

Dalším tématem, kterým se práce zabývá jsou neuronové sítě a jejich vývoj. Neuronové sítě se dají vyjádřit jako podmnožina umělé inteligence. Počítačové vidění na tuto oblast spoléhá. Postupným vývojem neuronových sítí docházelo k jejich rozšíření a využití v mnoha jiných odvětvích, jako jsou zdravotnictví, fyzika, neurobiologie a podobně. Získávání informací z obrazu je tedy dnes velmi žádané. Mě například tato oblast zaujala poprvé v oblasti dopravy, kde jsem byl svědkem užití mnoha senzorů anebo řešení placeného parkování na základě detekce a identifikace poznávacích značek. Typické úkoly a oblasti se inspirují a čerpají z online článku [6].

- **Zdravotnictví** – použití v medicíně vede k diagnostice a řešením mnoha problémů. Od použití RTG u zubaře, přes ultrazvuk až po detekci nádorů.
- **Robotika a stroje** – v poslední době se zavádí a aplikuje počítačové vidění například ve formě senzorů do běžných automobilů, pro funkce jako je hlídání pozice automobilu

v pruzích, nebo vizualizace couvání. Dále také například k řízení bezpilotních, autonomních vozidel, kde systémy pomáhají hlídat různé funkce, nebo k navigací robotů a podobně.

- **Fyzika, průmysl** – ve fyzice dochází k využití například pro detekci elektromagnetického záření. V různých výrobních průmyslech se užívá počítačového vidění pro kontrolu kvality – například detekci deformace výrobků a podobně. Co se týče vojenského průmyslu, všichni jsme se setkali s detekci nepřátelských jednotek či vozidel, použití tepelného vidění a podobně.

Disciplíny počítačového vidění

Jedním z elementárních problémů, které počítačové vidění zkoumá a řeší bývá otázka, zda daný obraz, video nebo jiná data obsahují určitý objekt či zde probíhá nějaká činnost, která nás interesuje. Jak již bylo řečeno, snaha o napodobení principů lidského vidění je dnes moderní a také zajímavá. Dveře pro rozvoj a výzkum této problematiky zůstávají otevřeny dokořán a v budoucnu zajisté přinesou více a více možností. Z technického pohledu to určitě přináší spoustu pozitiv. Dle mého názoru to však také vrhá jakýsi technostres a obecně to narušuje běžné fungování světa. Následující odstavce jsou věnovány disciplínám souvisejícím s bakalářskou prací či problematice příbuzné.

Rozpoznávání obrazu

Dnešní možnosti nabízejí spoustu řešení – schopnost zaměřit se na rozpoznávání libovolných objektů, možnost detekovat vzdálenosti objektů, určit rychlost ze záznamu a podobně. Situace, kdy dnešní telefony využívají rozpoznání lidských tváří k následnému odemčení displeje nebo jiné manipulaci není rozhodně žádnou výjimkou. Lze také rozpoznávat objekty v různých podmínkách, jako je proměnlivé počasí, světlo, teplo a podobně. Níže jsou uvedeny typické **úlohy**, na které se rozpoznávání obrazu zaměřuje.

Rozpoznání (klasifikace) objektů: Typ úloh, který se snaží na základě učení předem získaných dat rozpoznat konkrétní objekty. Lze aplikovat na jednu třídu objektů, ovšem také na několik. Často spojeno s určením pozice objektu ve scéně nebo obrazu.

Identifikace objektů: Cílem je rozpoznání individuálního kusu objektu. V dnešní době například typicky v oblasti biometrie – identifikace osob, otisků prstů.

Detekce objektů: Obsah scény či obrazu bývá prohledáván na základě specifikovaných podmínek. Tímto druhem úkolu se zabývá praktická část této práce. V praxi bývá řešeným problémem například rozpoznávání a analýza různých problémů v medicíně nebo zmíněnou detekci poznávacích značek automobilů. Nemusí se vždy jednat pouze o detekování objektů. Díky dnešním možnostem bývá detekce relativně často užívána jako krok pro další možné práce s počítačovým viděním, jako její schopnost poskytnout úsek obrazu, který je objektem zájmu a následně s ním pracovat. Použité metody a způsoby jsou chronologicky popsány v následujících kapitolách.

Rozpoznávání písma

Mezi nejznámější a poměrně již dlouho dostupnou metodu patří takzvaný **OCR – Optical Character Recognition**. Jeho schopností je konvertování písmen, číslic anebo také jiných symbolů. Běžně se používá k převedení textu z obrázku nebo naskenovaných dokumentů – tedy pro konverzi do elektronické podoby. V současné době existuje mnoho software, které s metodou OCR pracují. Některé z nich zvládají převedení, jiné přímou editaci do obrazu. V praxi může být typicky využíván například pro extrahování dat z dokumentů jako jsou cestovní pasy, smlouvy, ale také mohou sloužit lidem se zrakovým postižením, kdy dochází k převedení do mluvené řeči. OCR spadá právě do oblastí počítačového vidění a staví na principech učení.

Intelligent character recognition (ICR) je pokročilá metoda OCR. Zatímco OCR bývá koncipován primárně pro text psaný na strojích, ICR se zaměřuje na takzvaný hand–writing, tedy text psaný ručně [5]. Základem je použití neuronové sítě a principu progresivního učení nových stylu písem a zvětšování databáze.

Intelligent Word Recognition (IWR) je metoda, která (co se týče rozpoznávání) kontrastuje s OCR a ICR. Zatímco obě předchozí metody využívají rozpoznávání znaků individuálně (například slovo auto bude zpracováno následovně „a“, „u“, „t“, „o“), metoda IWR pracuje na základě rozpoznání celých slov. Snaží se na základně vzorů a slovníku vyhledat slovo auto. IWR dokáže pracovat také s ručně psaným písmem [24].

Kromě zmíněných metod existuje také několik datasetů pro ručně psané znaky. Mezi nejznámější patří patří datasety **NIST**, **MNIST**, případně **EMNIST**. Databáze NIST obsahuje asi 810,000 obrázku, pořízených 3,600 autory [13]. Z této sady se dále rozvinul a modifikoval dataset MNIST, který čítá asi na 60,000 trénovacích dat, testovacích okolo 10,000 [20]. Všechny obrázky jsou centrovány a normalizovány na stejnou velikost obrázku (28x28 pixelů). Rozšířením tohoto datasetu je poměrně nová databáze EMNIST [8]. Dále také existují kolekce dat v různých jazycích, ku příkladu v arabštině a podobně.

Zmíněné principy často pracují se zachycením písma na jednom konkrétním, často neměnném pozadí, za stejného světla a povrchu. Dnešní algoritmy a koncept učení neuronových sítí přináší možnost, jak se s těmito proměnnými parametry vypořádat. Bohužel, použití obrázku při vysokém rozlišení je extrémně náročné na jejich zpracování a výpočet, tudíž často dochází ke zmenšování.

Kapitola 2

Hluboké učení a neuronové sítě

Kapitola má za úkol nastínit základní principy neuronových sítí, jejich dělení a stavbu z obecného hlediska.

2.1 Hluboké učení

Hluboké učení neboli deep–learning spadá do umělé inteligence. Ta se stává součásti našeho každodenního života. Bývá stále diskutovanější a nezbytná v různých směrech, ku příkladu v automatizovaném řešení problémů, rozpoznávání řeči a podobně. Hlavním prvkem, již z názvu vyplývajícím, je tedy proces učení, které posouvá celé toto odvětví neustále dopředu a přitom za neobyčejné rychlosti a náročnosti, která se nedá srovnat s učením realizovaným lidskou bytostí. Právě do tohoto segmentu spadají neuronové sítě.

2.2 Esenciální princip sítí

Koncept neuronových sítí používaný v této práci čerpá z oblasti biologie a chování živých organismů – tedy jednotlivých buněk vzájemně propojených v nervové soustavě. Bakalářská práce se zaměřuje na sféru umělé inteligence.

Jde o sadu algoritmů zaměřujících se na rozpoznávání různých vzorů, řešení obtížných úloh nebo zdokonalování různých vlastností. Mezi typické cíle patří klasifikace, predikce, regrese a podobně. Na základě procesu učení, který vyžaduje vstupní trénovací data, mohou být sítě schopné třídit a rozpoznávat data náhodná.

Neuronová síť je model často tvořený z několika **vrstev**. Tyto vrstvy bývají potom tvořeny z uzlů – neuronů, více podkapitola 2.4. Zmíněný neuron představuje základní prvek sítí. Ten se dá popsat několika elementárnějšími prvky, jako jsou vstupy, váhy nebo také takzvaná aktivační funkce. Vrstvy obsahující neurony bývají z pravidla navázány na další vrstvy, se kterými mohou vytvářet masivní model, který disponuje schopnosti řešit složitější a komplikovanější úlohy.

2.3 Historie neuronových sítí

Veškeré historické poznatky jsou čerpány z článku kanadské univerzity v Torontu [25].

Historie neuronových sítí se datuje do 40. let minulého století. První datovaná zmínka pochází z roku 1943. Neurofyzik Warren McCulloch a matematik Walter Pitts popsali jejich

základní vizi, jak by neuronové sítě mohly fungovat. Jejich první model neuronové sítě byl spojen s elektrickými obvody. Tento model perceptonu bývá používán dodnes.

S pozdějším vývojem a nástupem vlny výkonnějších počítačů došlo na první simulace. V roce 1959 Bernard Widrow a Marcian Hoff vyvinuli dva první modely. Jejich jména byla Adaline a Madaline. Madaline byl první model určen k řešení problémů reálného světa. Sloužil k eliminovaní ozvěn telefonních hovorů.

Až do 80. let výzkum v oblasti neuronových sítí stagnoval kvůli finanční náročnosti a podobně. Následně došlo k vlně rozvoje, která trvá dodnes. Uplatnění a trénování neuronových sítí je stále velmi náročné na vytížení hardware.

2.4 Jednovrstvé a vícevrstvé neuronové sítě

Jednovrstvá síť

Neuronová síť s jednou vrstvou funguje na bázi struktury o M neuronech, kde každý má N vstupů. N vstupů je potom namapováno na M výstupů. Pro jednovrstvé sítě lze použít algoritmy stejného typu, jako jsou využívány pouze pro jeden neuron. Tyto sítě jsou však schopny se naučit a řešit pouze lineárně separabilní problémy [28].

Vícevrstvá síť

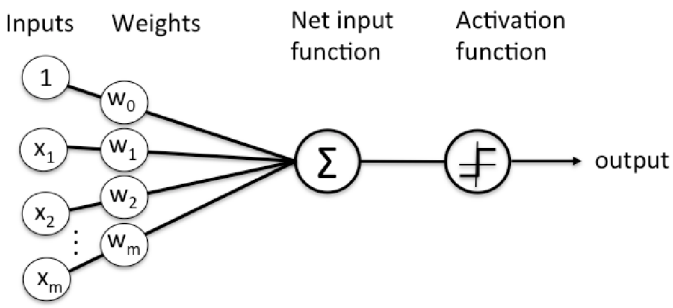
Dnes nejpoužívanější a nejrozšířenější typy sítí jsou tzv. vícevrstvé neuronové sítě. Bývají složeny z vrstvy vstupní, následují vrstvy skryté a ve finále vrstva výstupní. Stavba tohoto typu sítě je dána opakovaným použitím základního modelu, perceptonu. Perceptony obsaženy ve skrytých vrstvách nebývají pak napojeny na výstup, nýbrž na výstupní vrstvu. Základní princip fungování sítě zahajuje vstup pixelů do první vrstvy, průchodem vpřed jednotlivými vrstvami, přepočet jejích výstupů a opakování tohoto procesu až k výstupu celé sítě. Váhy zastoupení pixelů pro jednotlivé vrstvy mohou být nezávisle upravovány [29].

2.5 Konvoluční neuronová síť

Konvoluční neuronové sítě jsou podmnožinou vícevrstvých neuronových sítí. Jejich princip spočívá v tom, že alespoň jedna z vrstev bude vrstva konvoluční. Standardně se používají pro zpracování obrazu, kde vstupem jsou jeho rozměry a barevná škála. Rozdíl oproti normálním sítím je ten, že konvoluční sítě mají neurony uspořádány ve třech dimenzích, výšce, šířce a hloubce [9]. Na vstupu vezme síť obraz, na který lze konvolucí aplikovat nějaký filtr, např. pro znázornění hran. Následně dochází k získání sady příznaků. Takto lze používat filtry za sebou.

2.6 Přetrénování sítě

S pojmem trénování je vázán také pojem loss funkce, viz. 3.2. Česky chybová funkce v podstatě říká, jak moc se model liší od skutečnosti. Během trénování dochází k postupnému zmenšování chyby a zvolený model nabývá korektnějších hodnot. Cílem je dostat chybu na co nejnižší hodnotu vůči anotovaným datům. Ovšem nalezení této nejmenší hodnoty nemusí vždy znamenat úspěch, jelikož může dojít k případu takzvaného přeučení sítě. Natrénovaný model se následně začne učit nežádoucí detaily. Síť potom není schopna reagovat na reálné



Obrázek 2.1: Stavba neuronu. Obrázek převzat z článku [31].

použití, pouze na naše trénovací data. Je doporučeno tento průběh hlídat a čas od času výstup a hodnoty otestovat.

2.7 Podrobnější stavba

Neuron a jeho stavba

Na obrázku 2.1 lze vidět schéma neuronu. Jeho základní fungování začíná ohodnocením vstupů váhami (násobení vstupů koeficienty). Váhy udávají míru zastoupení vstupu, čím větší hodnotu mají, tím důležitější jsou. Následuje vážená suma jednotlivých vstupů (pixelů), která je předána a použita pro přepočet tzv. **aktivační funkce**.

Aktivační, nebo také **přenosová** funkce umožňuje vypočítat výstup daného neuronu. Funkce čeká na spuštění a následně se snaží namapovat výsledek v určitém rozsahu (podle volby této funkce) a přeposlat výstup jako další vstup. K její spuštění dochází při překročení určité prahové hodnoty.

Aktivační funkce se dělí na dva typy a to na **lineární** a **nelineární**. V dnešní době bývají používány převážně nelineární aktivační funkce. Mezi nejpoužívanější z nich patří například sigmoidální funkce, která spadá mezi funkce logistické. Používá se pro modely, kde požadovaným výstupem bývá pravděpodobnost. Funkce vrací hodnoty v intervalu (0,1), což je interval, popisující hodnoty rozložení pravděpodobnosti. Jinou používanou funkcí může být například funkce RELU, skoková či hyperbolická [30].

Dalším prvkem trénovacího procesu je **gradient**. O gradientu by se dalo zjednodušeně říct, že udává míru změny všech vah vstupů v závislosti na změně chyby. Gradient si lze představit jako strmý kopec, který chceme sjet na kole. Čím méně je tento kopec strmý, tím pomaleji pojedeme – obdobná situace nastává při učení modelu. Dosáhne-li gradient hodnoty nulové, proces učení dosáhne stádia stagnace. Jde tedy o jakýsi optimalizační algoritmus, založený na tvaru konvexní funkce, snažící se minimalizovat funkci na své lokální minimum. Matematicky lze gradient popsat parciálními derivacemi na základě vstupů [10].

2.8 Backpropagation a jeho princip

Backpropagation spočívá v porovnávání výstupu skutečně natrénovaného modelu vůči výstupu, které ho je třeba dosáhnout. Na základě snahy najít co největší podobnost, nebo docílit jakési ekvivalence mezi těmito dvěma výstupy, dochází k postupnému průchodu napříč vrstvami opačně – pracuje tedy zpětně od výstupní vrstvy po vstupní. Back propagation bývá využíván pro algoritmus výpočtu gradientů. Poté dochází k úpravě vah pixelů a optimalizování výstupů vrstev [12].

Kapitola 3

Používané modely a metriky

Neuronové sítě se staly primární a vedoucí metodou v oblasti detekce objektů. Následující odstavce uvádí přehled nejpoužívanějších modelů. Zevrubně potom rozvádí model aplikovaný v bakalářské práci a jeho principy.

3.1 Přehled modelů

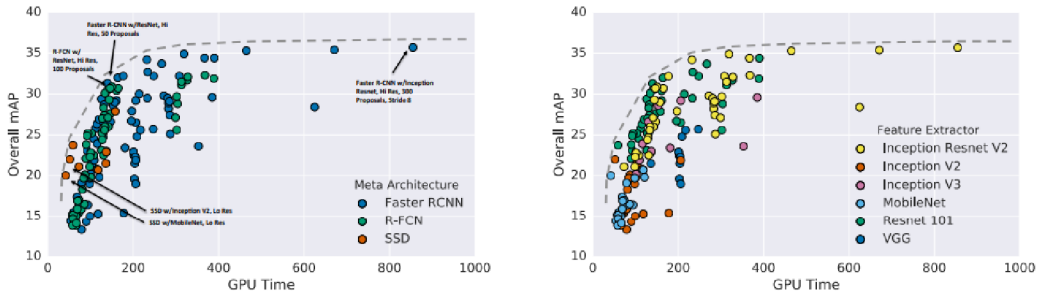
Oblast detekce objektů dnes nabízí celou řadu konvolučních objektových detektorů. Pro vývoj aplikace bylo třeba vybrat některý z nich. Jelikož je aplikace zaměřena na mobilní zařízení, mělo by se jednat o model, který zajistí optimální poměr mezi rychlosti a přesnosti, v poslední řadě také optimální využití paměti.

Vývoj nových modelů je opravdu moderní disciplína. Dalo by se říct, že aktualizované verze modelu vychází poměrně frekventovaně. Práce se zaměří na výběr sítě z novějších a nejpoužívanějších z nich. Existuje celá řada sítí, které však nejsou pro tuto práci relevantní, jelikož jejich stavba a časová náročnost je pro mobilní aplikaci nepoužitelná. Mobilní zařízení nejsou zatím schopna s takovýmito modely pracovat. Odstavce shrnující detektory *YOLO, SSD* a extraktor *Inception v2* jsou čerpány a zkoumány v online publikaci o evaluaci těchto modelů [11].

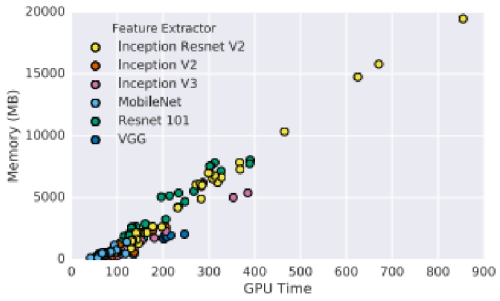
YOLO – you only look once je detektor, který vyšel v roce 2015. Jeho přesnost na základě VOC 2017 datasetu dosahuje 58.7%. Opravdu stručně řečeno, YOLO vezme vstupní obrázek, rozdělí ho na regiony a vypočítá pravděpodobnost predikovaných ohraničení. Na základě vah těchto rámců a nastavení úrovně prahu akceptovatelných ohraničení potom dochází k eliminaci těch, které nedosahují nastavené hodnoty. Existuje také zmenšená verze ***Tiny YOLO***.

SSD – Single Shot Detector se objevil v roce 2016. Jeho přesnost dosahuje až 74%. Tomuto detektoru se bude práce věnovat níže, tudíž následné experimenty ověří, zda tomu tak je. V kombinaci s extraktorem ***MobileNet*** dosáhl přesnosti 71.1% na testovaném ImageNet.

Inception v2 je extraktor vytvořený Googlem v roce 2014. Tehdy se ukázal jako nejspolehlivější, při dosažené přesnosti 73.9% na datasetu ImageNet.



Obrázek 3.1: Graf architektury a extraktoru. Obrázek převzat z práce [17].

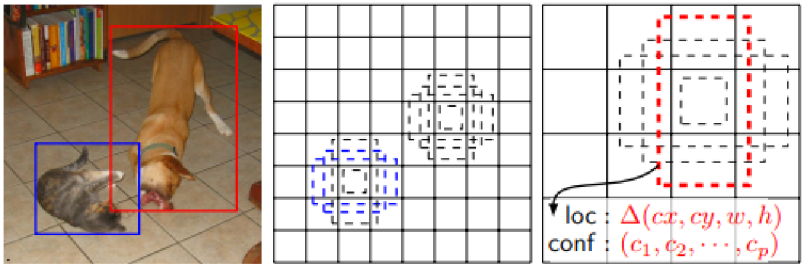


Obrázek 3.2: Graf využití paměti a času extraktoru. Obrázek převzat z publikace [17].

Nejlépe se potom časově osvědčil model ***SSD***, s použitím extraktoru ***MobileNet***. To se ovšem dalo očekávat vzhledem k jeho stavbě. Jedná se o model přizpůsobený na mobilní a vestavěná zařízení, více v podkapitole 3.3. V současné době existují již i novější verze, avšak MobileNet je stále ve vedení.

Uvedené tři obrázky popisují výsledky průzkumu jiných autorů. Konkrétně znázorňují srovnání použitých architektur, extraktorů s ohledem na přesnost, využití paměti a náročnost.

Obrázek 3.1 popisuje ukazuje dva grafy. Graf vlevo popisuje poměr mAP ku výpočetnímu času GPU, co se týče meta-architektur. Druhý graf popisuje poměr tutéž informaci jako první, avšak tentokrát z pohledu feature extraktorů, které jsou tedy modely aplikovány na začátek zmíněných architektur. Je mnoho kritérií, podle kterých extraktor vybrat. To může být rozhodující z hlediska souvisejících parametrů. Graf na obrázku 3.2 srovnává využití paměti oproti využití výpočetnímu času GPU. Dle kritérii lze vypozorovat jako cílovou kombinaci architekturu SSD s použitím extraktoru MobileNet.



Obrázek 3.3: Princip detekce metodou SSD. Obrázek převzat z publikace [21].

3.2 SSD – Single Shot MultiBox Detector

Jedná se o architekturu používanou převážně pro real-time detection. Tato architektura nedosahuje tak dobrých výsledků jako například Faster–RCNN či R–FCN. Na rozdíl od obou z nich se soustředí na rychlost. Dle publikací je tedy SSD v mnoha kategoriích rychlejší a také přesnější než jiné konkurenční detektory s podobným zaměřením. Podobně jako u většiny dnešních sítí, SSD stačí na vstupu obrázky a jejich anotace. Použité informace v celé sekci 3.2 jsou čerpány z online publikace [21].

Architektura staví na principu feed-forward (dopředné) neuronové síti, kde průchod je dán pouze jedním směrem (ze vstupu na výstup) a nevrací se na začátek. Tato síť produkuje pro detekování objektů několik ohraničení s proměnnou velikosti a řekněme pravděpodobnostním ohodnocením výskytu objektu v těchto ohraničeních. Dále následuje eliminace rámců s nemaximálním ohodnocením. Tento přístup (díky různým měřítkům) zajišťuje vysokou přesnost dokonce i na obrázcích s malým rozlišením. Vrstvy na počátku jsou zastoupeny v podobě dalších sítí. Běžně využívá model SSD v základu síť VGG–16. Místo VGG–16 lze aplikovat extraktor, variantu sítě MobileNet.

První část obrázku 3.3 zachycuje anotovaný vstup. Na druhé a třetí je již mapa příznaků (pokaždé v jiném měřítku). Jedná se tedy o predikované rámce s ohodnocením míry výskytu objektu. Barevná ohraničení potom znázorňují nalezený objekt.

Výhodou oproti detektoru YOLO nebo také Overfeat lze považovat fakt, že SSD využívá mapy příznaků v různém měřítku. Extraktor obsahuje konvoluční vrstvy, které postupně snižují svou velikost. Výsledkem bude konvoluční vrstva jiná pro každou mapu příznaků. Na rozdíl od typických detektorů, informace o rámcích potřebují být přiřazeny k jednotlivých výstupům. Následuje proces kalkulace chybové funkce a zavedení back–trackingu. Je třeba také zvolit počáteční rámce, které se liší velikostí, měřítkem a pozicí. Tyto rámce jsou pak porovnávány s původními (anotovanými) pomocí IoU a prahové hodnoty větší než 0.5.

Chybová funkce

Localization loss udává rozdíl mezi anotovaným ohraničením a predikovaným ohraničením. Snažíme se tuto ztrátu zmenšovat a přibližovat pozitivní nálezy ke skutečnému ohraničení.

Confidence loss je spojená s predikcí třídy. Pro pozitivní predikce dochází k úpravě této funkce podle ohodnocení „důvěry“ odpovídající třídy. Pro negativní predikce je postup stejný, jako by žádný objekt nebyl nenalezen.
Celkovou hodnotu loss funkce lze potom vyjádřit váženou sumou složek localization loss a confidence loss.

Nechť $x_{ij}^{p} = 1,0$ je představitelem úspěšné shody i-té predikovaného ohraničení ku j-té ground-truth (anotované oblasti) třídy p. Celkový počet těchto oblastí může být ≥ 1. Celková chybová funkce lze vyjádřit jako

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)). \tag{3.1}$$

N odpovídá počtů shodně vyhodnocených oblastí, přičemž pokud je tento počet 0, je hodnota **celkové loss** nulová. Localization loss je dána smooth loss mezi predikovaným ohraničením *(l)* a anotovaným *(g)*. α je váha pro localization loss. Chyba lokalizace má potom tvar

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in cx,cy,w,h} x_{ij}^{k} \text{smooth}_{L1}(l_i^m - \hat{g}_j^m), \text{kde} \tag{3.2}$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_j^{cx})/d_i^w, \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_j^{cy})/d_i^w,$$

$$\hat{g}_w^j = log(\tfrac{g_w^j}{d_w^i}), \qquad \hat{g}_h^j = log(\tfrac{g_h^j}{d_h^i}).$$

Confidence loss je potom vypočítána jako softmax loss přes mnohonásobné confidence tříd.

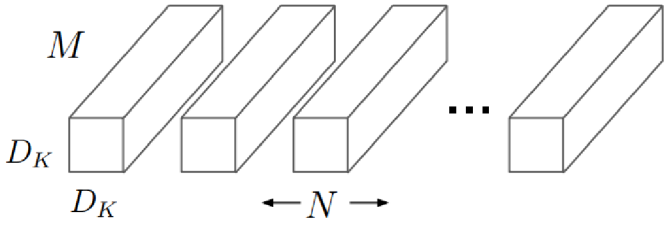
$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{P} log(\hat{c}_i^p - \sum_{i \in Neg} log(\hat{c}_i^0) \qquad \text{kde} \qquad \hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{3.3}$$

3.3 Model MobileNet, MobileNet Verze 2

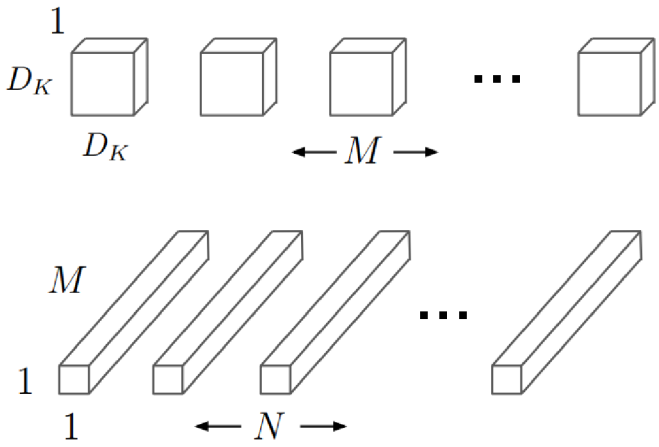
MobileNet

Jak již název napovídá, neuronové sítě MobileNet se zaměřují primárně na mobilní a vestavěná zařízení. Snahou těchto modelů je prioritně nízká latence, avšak neméně důležitou bývá i velikost modelu. Myšlenku sítí MobileNet lze pozorovat v použití takzvaných ***depthwise separable convolutions*** – hloubkových separovatelných konvolucí na místo klasických. Následující informace o popisu modelu a architektury MobileNet jsou čerpány z online publikace [16].

Tyto oddělitelné konvoluce do hloubky používají jednotlivé filtry na každý ze vstupních pixelů. Například ze 3 vstupních pixelů vzniknou opět 3 výstupní pixely. Vstupních pixelů do další fáze lze označit jako množství *M*. Na tyto jednotlivě filtrované výstupy se následně používá bodová konvoluce. Ta za pomocí již běžné konvoluce, ovšem o rozměru filtru *1x1xM* vytváří nový prostor z pixelů. Běžná konvoluce bloky nerozděluje na jednotlivé pixely. Používá filtr a rovnou tvoří novou sadu pixelů, například pro 3 vstupní pixely vytvoří 1 výstupní, podobně se tomu děje v již zmíněné bodové vrstvě. Takových sad je potom množství *N*, které je následně znovu aplikováno.



Obrázek 3.4: Ukázka standardních konvolučních filtrů. Obrázek převzat z publikace [16].

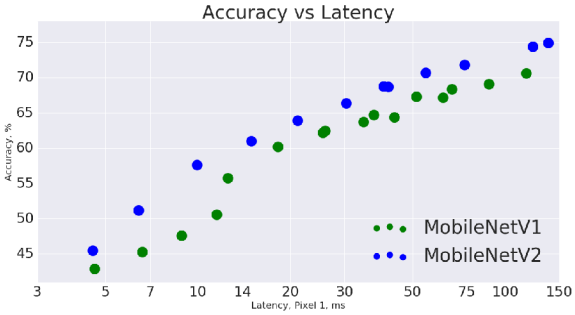


Obrázek 3.5: Ukázka hloubkově konvolučních a bodově konvolučních filtrů. Obrázek převzat z publikace [16].

Architektura MobileNet používá k optimalizování účinnosti dva takzvané hyperparametry. První z nich je width multiplier neboli multiplikátor šířky, druhý potom resolution multiplier, tedy multiplikátor rozlišení.

Multiplikátor šířky se stará o redukci velikosti modelů a snížení výpočetní náročnosti. K výše uvedeným vrstvám je zaveden koeficient α, který umožňuje udělat každou vrstvu ještě více subtilní. Koeficient se používá v rovnici pro zmíněné proměnné M tedy počet vstupních pixelů a N, neboli počet výstupních pixelů.

Druhým hyperparametrem je **multiplikátor rozlišení** ρ, který se stará opětovně o snížení výpočetní náročnosti. Tento koeficient ρ se používá na vstupní obrázek a také na každou aplikovanou vrstvu. Ty jsou pak všechny postupně redukovány stejnou hodnotou multiplikátoru.



Obrázek 3.6: Graf srovnání verzi modelu MobileNet. Obrázek převzat z článku [27].

MobileNet Verze 2

Druhá verze MobileNetu přinesla relativně velké zlepšení. Jedná se o verzi mnohem rychlejší, avšak zároveň dosahuje stejných, ba dokonce lepších výsledků přesnosti. MobileNet_v2 potřebuje 2x méně operací, o 30% méně parametrů a zároveň rychlost stoupá o 30–40%, přičemž přesnost oproti první verzi je větší [27]. Jednoduché grafické porovnání lze vidět na obrázku 3.6.

Nový model využívá o jednu vrstvu navíc. Jedná se o tak zvanou rozšiřovací vrstvu, jež zvyšuje počet vstupních pixelů. Následuje standardní hloubková vrstva, která aplikuje operace na předchozí vrstvou rozšířený vstup. V poslední řadě je projekční vrstva snižující počet pixelů opět na malou hodnotu. Použitím principu snižování počtu pixelů dochází právě také ke snižovaní počtu operací. Residuální spojení lze považovat za další nový prvek oproti první verzi. Napomáhá toků gradientu skrze síť tím, že přeskakuje některé vrstvy za účelem urychlení učení. Používá se pouze tehdy, kdy počet vstupních kanálu bloku je ekvivalentní počtu výstupním [15].

3.4 Intersection over Union a mean Average Precison

Intersection over Union

Intersection over Union, česky průnik nad sjednocením, je tedy metrika používána pro vypočtení přesnosti nad datasetem. Metriku lze aplikovat na jakékoli algoritmy využívající predikci ohraničení. Metrika užívá dvou fundamentálních elementů.

- **Predikce (Predicated bounding boxes)**
- **Anotace (The ground–truth bounding boxes)**

Mluvíme-li o **predikci**, jedná se o ohraničení, která již vychází z našeho natrénovaného modelu. V opačném případě **anotace** jsou ty, jejichž fotografie prošly procesem manuální lokalizace tagů a zařazují se do testovacích dat viz. kapitola 4. IoU lze tedy vyjádřit jako

$$IoU = \frac{AreaofOverlap}{AreaofUnion}. \tag{3.4}$$

Za pomocí zmíněných proměnných vzniká takzvaná oblast překrytí, ***(Area of Overlap)***, která je dána průnikem predikovaných a pravdivých ohraničení, druhou oblast lze definovat sjednocením těchto proměnných, ***Area of Union*** [26]. Příklady lze pozorovat na obrázku 3.7.

Obrázek 3.7: Možné případy oblastí průniku a sjednocení.

Obrázek 3.8: Příklady hodnot IoU. Obrázek převzat z článku [26].

Výslednou hodnotou je poměr (rovnice 3.4) nad kterým se provádí mAP. Je velice nepravděpodobné, že se stane, abychom dostali přesnou shodu, tedy hodnotu 1. Obecně platí, že hodnota IoU by měla být větší než 0.5. Takovou predikci lze považovat za dobrou. Více vysvětluje obrázek 3.8.

Mean average precison

Následující poznatky vycházejí z online článku [18]. *Precision* a *recall* jsou pojmy, které s následujícími metodami souvisí.

Precision udává procentuální přesnost vůči pozitivních predikcím.

Recall popisuje míru nalezení pozitiv vůči počtu celkových predikcí.

Výpočet hodnot **precision** a **recall** lze získat pomocí vzorců. Potřebnými proměnnými jsou správná a nesprávná pozitiva, avšak i negativa při daném IoU. Ty lze považovat za vyhovující při překročení prahové hodnoty IoU.

TP – správné pozitivní detekce, které lze považovat za vyhovující.

FP – nesprávné pozitivní detekce, které lze považovat za nevyhovující.

TN – správné negativní detekce znamená dosažení detekce v oblasti, kde není potřeba.

FN – nesprávná negativní detekce, kdy je detekce chybějící.

Mean Average Precision je metrika, která se používá pro zjištění přesnosti napříč mnohočetnými třídami objektů. Mějme ku příkladu 90 tříd objektů, z niž každá třída obsahuje jiný počet dat k dispozici.

K výpočtu je potřeba určit AP (Average Precision). AP uvádí individuální přesnost pro jednotlivé třídy. Provedením průměru přes všechny třídy získáme mAP [4].

$$mAP = \frac{\sum_{i=1}^{I} = AP(i)}{I} \tag{3.5}$$

Používaná neuronová síť zahrnuje pouze jednu třídu (třídu pro detekci tagů), takže hodnota mAP bude rovna hodnotě AP. Obecný výpočet lze pozorovat v rovnici 3.5.

Kapitola 4

Sběr dat a trénování modelu

4.1 Sběr dat

Jak již bylo výše zmíněno, pro natrénování a následné použití neuronové sítě je zapotřebí vhodný dataset. Ten by měl být dostatečně robustní a zároveň různorodý (aby přesnost dosahovala co nejlepších hodnot).

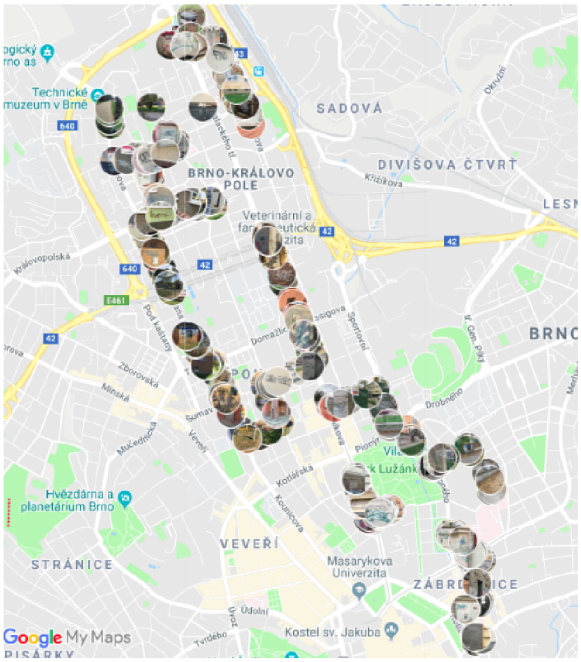
Při sběru byla navázána spolupráce se studentem Martinem Fischerem, který se zabývá podobnou problematikou nad stejnou množinou dat. Dále byl do trénování zahrnut dataset Jana Pavlicy (747 fotografií obsahující 1696 tagů), který byl využit v jeho bakalářské práci v roce 2017 [23].

Sběr dat probíhal napříč Brnem, kde ovšem samozřejmě dochází k nalezení duplicitních podpisů. Tyto duplicity jsme se po domluvě snažili částečně eliminovat, aby došlo k zachování různorodosti. Dalším problémem je překrývání mnoha tagů přes sebe. Při sběru a analýze dat tak někdy docházelo k zahazování lukrativních, avšak někdy extrémně nepoužitelných fotografií právě z výše zmíněných důvodů. Na obrázku 4.1 lze vidět příklad přehlcených oblastí, jež trénování zahrnovalo. Snažili jsme se také zachytit tagy z různých úhlů a vzdálenosti. Struktury povrchu byly různé, od jednobarevných ploch, přes sklo, kameny a jiné. Variabilita stylů je poměrné znát. Dataset byl tvořen za dne. Pořízených fotografií s bleskem není ani desetina, proto nelze předpokládat případnou úspěšnost modelu za jiných podmínek, například za nedostatečného osvětlení.

Co se týče části datasetu pořízeného mnou, k zachycení fotografií došlo pomocí mobilního telefonu Apple iPhone 6S. Původní rozlišení všech fotografií bylo 3024 x 4032 pixelů, což ovšem vedlo k nucenému zmenšení fotografií (dataset dosahoval opravdu velkých hodnot z hlediska velikosti, čímž znemožňoval většinu práce s pamětí) na 20% tzn. 605x806 pixelů. Nástroj Nautilus Image Converter usnadnil právě změnu velikosti. Zmenšení velikosti se však týká pouze procesu trénování. Všechny fotografie jsou ve formátu JPEG. Použité fotografie lze vidět na obrázku 4.2.

Obrázek 4.1: Přehlcené oblasti.

Obrázek 4.2: Ukázka pořízených dat.



Obrázek 4.3: Mapa pokrytých oblastí.

Obrázek 4.4: Anotované fotografie.

Všechny fotografie byly zachyceny s aktivními polohovými službami GPS. Díky získaným souřadnicím mohlo dojít k následnému sdílení a vykreslení lokalit na mapě viz. obrázek 4.3.

Veškerá nově pořízená data prošla procesem anotování (zobrazeno na obrázku 4.4), data získána z výše zmíněné bakalářské práce anotace již obsahovala [23]. K anotovaní jsme zvolili nástroj LabelImg, jehož výstupem jsou soubory ve formátu XML, nesoucí informace primárně o tzv. *„bounding boxech"* potřebných k trénování. Tyto xml soubory se poté zpracovávají a konvertují pro následné trénování. Trénování bylo provedeno pouze s jednou třídou objektů, ***tag***.

Natrénování neuronové sítě s ohledem na data neznamená pouze sběr dat a anotaci dat. Samozřejmosti je vytvořit trénovací a testovací sadu. V ideálním případě pro tyto sady tvořit alternativy, a zkoušet jejich různé variace. V rámci zmíněné spolupráce byla stanovena velikost testovací složky o 469 fotografiích. Trénovací složka datasetu potom obsahovala zbytek.

Za pomocí použití webové aplikace My Maps (společnost Google) došlo k vytvoření sdíleného datasetu, který ve finální fázi zahrnuje 2258 fotografií. Celá databáze celkově obsahuje 5596 anotovaných graffiti tagů.

4.2 Trénování modelu

Trénování probíhalo pomoci open-souce knihovny TensorFlow popsané v podkapitole 4.4. Před spuštěním samotného trénování došlo k převedení fotografií a anotací na tabulkový formát csv, který byl následně převeden na takzvané TensorFlow záznamy, s nimiž už dokáže knihovna pracovat.

Konkrétně se tedy jednalo o již dříve zmíněný model `ssdMobileNet_v2`. Po experimentech s nastavením (s ohledem na velikost dostupné paměti), musela být konfigurace učení sítě upravena na konkrétní parametry.

Jedním z nich je zredukování velikosti vstupu na 300x300 pixelů. Inicializovaná hodnota učícího kroku byla vymezena na 0.004. Prahová hodnota iOU byla ekvivalentní 0.6 a počet vzorků v jedné dávce (batch size) byl optimalizován pro hodnotu 24. Během trénování došlo k testování mnoho variant, jako například experimentování s počtem iterací či zvolení počátku kvantizace sítě. Počet iterací se po testování a porovnávání mAP ustálil na 20,000 kroků, přičemž při 19,000 byl stanoven počátek kvantizace s 8 bity pro váhy a aktivaci.

Ačkoli kvantizací ztratíme na přesnosti (viz. kapitola výsledky a experimenty), jedná se o nezbytný krok pro převedení natrénované sítě do formátu TensorFlow Lite, jenž je moderním a rozvíjejícím se řešením pro nasazení modelů na mobilní a vestavěná zařízení. Velikost a rychlost modelu byla mnohonásobně regulována, což lze považovat v dnešní době za nezbytné pro minimalizaci odezvy.

TensorFlow poskytuje také vizualizační nástroj jménem TensorBoard, který usnadňuje orientaci a popisuje průběh trénování. Tento nástroj umožnil monitorování průběhu učení, přičemž sledovanými subjekty byl vývoj hodnoty loss funkce, vliv změny parametrů a také čas učení.

Testování a modifikace modelu probíhala opakovaně. Následně došlo k jeho konverzi, tak aby mohl ve finální fázi posloužit jako back–end pro mobilní aplikaci a současně pro testování časové náročnosti.

4.3 Kvantizace

Kvantizací rozumíme v obecném směru proces konverze rozsahu spojitých hodnot na konečný rozsah diskrétních hodnot. V případě neuronových sítí se jedná o kvantizaci bitů. Počet bitů, na kterých kvantizace probíhá, potom rozhoduje o přesnosti a kvalitě výsledných hodnot. V následujících odstavcích je vycházeno z online dostupného výzkumu [14].

Kvantizace neuronových sítí

V oblasti hlubokého učení se pro reprezentaci vah, aktivační funkce a gradientů používal (stále ještě používá) numerický formát 32 bitové plovoucí čárky. Snaha o snížení převážně

Komponenty	Výhody získané kvantizací	Problémy
Váhy	Menší velikost modelu Rychlejší trénování, lepší časová náročnost Méně zdrojů	Obtížnější konvergence s kvantizovanými váhami Požadovány přibližné gradienty Ztráta přesnosti
Aktivace	Menší paměťová náročnost při trénování Umožňuje použití bitových operací Méně zdrojů	Nesoulad gradientů
Gradient	Komunikace a úspora paměti při paralelním trénování sítí	Požadována konvergence

Tabulka 4.1: Výhody a nevýhody kvantizace. Tabulka převzána z publikace [14].

výpočetní náročnosti a velikosti vedla ke kvantizaci na celočíselný typ, ba dokonce na binární reprezentaci. Bylo dokázáno, že při používání 8 bitového formátu nedochází ke snížení přesnosti. Výzkum se v současnosti zabývá možnosti snížení na 4,2 až 1 bit. Tabulka výše 4.1 zachycuje přehled výhod a nevýhod z výzkumu o kvantizaci.

V případě **vah** dochází tedy ke snížení velikosti modelu, využití menšího množství energie a rychlejšímu trénování a výpočtu modelu. Nevýhodou je, že dosažení konvergence se stává náročnější v případě kvantizace vah, vyžaduje přibližné gradienty a dochází k poklesu přesnosti. Kvantizace vah probíhá ve dvou krocích. Během prvního se hodnoty vah komprimují do rozsahu $[-1, 1]$. V následujícím kroku jsou tyto komprimované váhy použité pro inicializaci sítě. Při kvantizaci vah třeba zmenšit také učící krok sítě, aby docházelo k dobrým výsledkům.

Co se **aktivační funkce** týče, za výhodu lze považovat menší využití paměti při trénování, umožňuje výpočet skalárního součinu pomocí bitových operací a opět také menší potřebná náročnost energie. Nevýhoda je nesoulad gradientů. Kvantizace aktivačních funkcí spočívá ve využití binárních operací, které dokážou urychlit trénování sítě. Vynecháním plné přesnosti aktivačních funkcí dochází k šetření paměti. Zmíněný nesoulad gradientů znamená, že může vést k odlišnosti v případě gradientu kvantizované aktivační funkce a zpětně počítaného gradientu.

Ohledně **gradientů** lze označit výhodou komunikace a šetření paměti paralelně s trénováním neuronové sítě. Nevýhodou bývá požadovaná konvergence. Kvantizace gradientů je v této oblasti výzkumu zánovní. Dochází ke snižování nákladů při šíření gradientu u trénování velkých sítí.

Tento proces byl uskutečněn při trénování sítě, aby mohlo dojít k dosažení výše zmíněných výhod, protože aplikace cílí přece jen na mobilní zařízení.

4.4 Užité jazyky, knihovny a nástroje

Programovací jazyky

Během praktické části byly použity jazyky Python a Java. Knihovna TensorFlow, Google Collaboratory pracují s Pythonem. Všechna práce až po vývoj samotné aplikace tedy probíhala v Pythonu. Bylo také třeba použít skripty pro převedení souborů formátu xml do potřebných formátů. Dále byl použit na různé menší skripty, které mi usnadnili práci s robustním datasetem jako s přepočítáním souřadnic pro anotace, manipulaci nad daty a podobně. Samotná aplikace potom byla psána v jazyce Java. Pro anotaci jsme využili labelImg, což je grafický nástroj, který slouží pro anotování obrázků prostřednictvím ohraničení a přiřazením jejich labelů.

TensorFlow

TensorFlow je open-source knihovna, která se specializuje na vysoce náročné numerické výpočty. Tato knihovna poskytuje širokou flexibilitu například ve volbě použitého hardware (CPU,GPU, TPU), umožňuje také nasazení na zařízeních jako jsou klasické počítače, přes servery, až po vestavěné či mobilní zařízení [1]. Co se týče tříd objektů, knihovna oplývá flexibilními prostředky – tudíž i možností natrénovat v podstatě cokoli. Pro natrénování našeho modelu došlo k využití právě této knihovny.

Google Collaboratory + My Drive

Pro trénování modelu byla využita webová aplikace Google Colaboratory, která poskytuje virtuální stroj, kde lze instalovat závislosti a provádět náročné výpočty prostřednictvím bloků kódu. Díky možnosti počítaní na GPU (Tesla K80 o kapacitě paměti 12GB), se uskutečnilo veškeré trénování a experimenty zde.

AndroidStudio

Jedná se o integrované vývojové prostředí pro vývoj aplikací založeného na IntelliJ IDEA. Prostředí umožňuje práci s GitHubem, možnost vývoje a ladění aplikace ať prostřednictvím virtuálního zařízení, tak zařízení skutečného. Aplikace byla sestavována a vyvíjena právě v něm.

Android

Tato sekce se snaží popsat motivaci, základní strukturu a fungování operačního systému Android spolu s představením vývoje jeho aplikací.

Android koupila společnost Google v roce 2005. Vzhledem k vzestupnému počtu použití a popularitě chytrých telefonů se brzy stal jedničkou na trhu. Jednotlivé verze nesou své jméno po dezertech. Jádro systému stojí na bázi linuxu, avšak je upraveno k potřebám googlu [3]. Jádro má na starost komunikaci mezi hardwarem a softwarem. K dispozici jsou nejrůznější ovladače, jako třeba pro kameru, displej, zvuk, wifi, klávesnici a podobně. V nynější době je aktuální verzi Android 9 Pie. S rostoucí verzí pochopitelně klesá procentuální schopnost aplikace pracovat na různých, často již SDK nepodporovaných zařízeních. Mezi nejpoužívanější verze současnosti patří Android 7.0, 7.1 Nougat, hned za ní potom Android 6.0 Marshmallow.

Stavba aplikace

Detailní popis životního cyklu lze nastudovat z oficiálních stránek, odkud tyto poznatky plynou [2]. Jde o základní posloupnost, na které běžné aplikace staví.

Samotné vyvíjené aplikace mají určitou strukturu. Mezi základní komponenty patří aktivity rozšířené o fragmenty, layouty – pro vizuální kontext a podobně. Aplikace také používá tzv. gradle skripty, které definují závislosti a pluginy, verze SDK, javy a v poslední řadě napojují ostatní knihovny. Pravidelně dochází k synchronizaci skriptů při sestavování projektu a to včetně relevantních kontrol.

Soubor `AndroidManifest.xml` zprostředkovává a zaobírá se formalitami jako například názvem balíčku a aktivit, povolením kamery, úložiště či polohy. Dále obsahuje řešení orientace obrazovky aplikace, která je v tomto konkrétním případě povolena pouze ve formátu portrétu. Manifest musí být zahrnut v každém projektu.

Základním kamenem životního cyklu aplikace je tedy aktivita. Jedná se o třídu, nesoucí sebou callback metody, popisující aktuální stav a dění. Konkrétně jde o metody (`onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()` a `onDestroy()`), které systém invokuje v různých stavech. Fragmenty potom představují jakési moduly, jež jsou volány v aktivitách při callbacích, například při zrušení, pozastavení aktivity. Tyto požadavky se samozřejmě vztahují i na fragmenty dané aktivity. Layouty v podobě dokumentů xml definují prvky, které se do aktivit, případně do fragmentů vykreslují. Metoda `OnCreate()` je spouštěna pouze při startu aktivity – probíhají zde požadavky, jako přiřazení do proměnných, instanciace lokálních proměnných a podobně. Po přechodu do dalšího stavu (startovacího) dochází volání metody `OnStart()`. Tato metoda dělá aktivitu viditelnou pro uživatele – vykresluje uživatelské rozhraní. Tento callback není rezidentní, po splnění úkolů okamžitě přichází metoda `OnResume()`. Zde probíhá interakce aplikace s uživatelem, Spouští se potřebné funkce, jako například pro vytvoření náhledu kamery a jiné. Při přerušení aktivity dochází k volání `onPause()` a zpětně potom naopak. Při pauze by měly být komponenty, ku příkladu kamera, uvolňovány pro případné využití jiné aplikace. Případně lze také alternativně použít `onStop()` volání, které se využívá při zastavení aktivity či delším přerušení. Poslední callback `onDestroy()` je prováděn při ukončení aktivity či například změny orientace zobrazení do landscape módu.

Není třeba využívat všechny tyto callbacky. Závisí to na požadavcích či potřebách aplikace. Za účelem nejlepšího využití a optimalizace by však vývojář měl problematice porozumět a implementovat řešení dle preferencí.

Kapitola 5

Aplikace a dosažené výsledky

5.1 Stanovené cíle

Ačkoli primární přínos práce spočívá ve vyzkoušení, použití a testování neuronových sítí na mobilních zařízeních, byly stanoveny cíle také pro samotnou aplikaci.

Mělo by se tedy jednat o funkční, stabilní aplikaci, která bude mít pár základních funkcí. Konkrétně tedy detekci, pořízení fotky, vytvoření databáze. Databáze umožňuje manipulaci s galerií a práci s geografickými souřadnicemi – lokalizace fotografie na mapě. Pro splnění účelu – tedy sběru dat je také cíleno na uložení souřadnice ve formátu anotací xml. Tím dojde k usnadnění sběru dat.

Porovnání přesností natrénovaných modelů, vyzkoušení časové náročnosti na různých zařízeních a základnímu srovnání vůči ostatním výsledkům se řadí k dalším řešeným problémům.

Aplikace je vydána pod licencí APACHE 2.0, přiloženou ke zdrojovým souborům aplikace v podobě komentáře.

Při trénování a vývoji aplikace bylo využito několik knihoven, open source softwaru, modulů a v neposlední řadě vlastních skriptů.

5.2 Návrh aplikace

Zprvu nebylo zřejmé, jak aplikace bude vypadat, fungovat a co bude umět. Prvotní návrh zahrnoval pořízení fotky, následné uložení a potom provedení samotné detekce z fotografie. Za atraktivní lze považovat aplikování real-time detekce, tedy přímé detekce v kameře s použitím popsaných modelů. Právě u té je časová náročnost prioritní.

Co se týče detailnějšího návrhu, při otevření aplikace by mělo dojít ke spuštění okna s náhledem kamery, ve kterém bude probíhat samotná real-time detekce. Dále by zde mělo být obsaženo tlačítko pro zachycení fotografie. Pravý horní roh obsahuje menu s možnostmi, kde uživatel smí měnit aktivity z náhledu a použití kamery do galerie pořízených snímku. Galerie by měla obsahovat fotografie, které by měly mít možnost smazání a také speciální tlačítko, při jehož sepnutí se aplikace odkáže přenesením souřadnic na aplikaci google map a vykreslit pozici pořízené fotografie na mapě. Samozřejmě toto menu bude umožňovat také zpětný přechod na původní obrazovku (detekci s kamerou). Pro výpis základních informací o aplikací bylo zvoleno tlačítko v menu.

5.3 Implementace

Zvolenou platformou pro aplikaci je tedy Android, primárně z důvodu snazšího testování, samotnou dokumentaci k vývoji a použití lze prohlásit za obsáhlejší. Aplikace cílí na verzi Android 6.0 (Marshmallow), která využívá API verze 23 a samozřejmě na verze vyšší. Jako referenční telefon byl zvolen Huawei P8 lite. Ten bohužel nepatří do kategorie výkonných telefonů dnešní doby, avšak k řešení vývoje aplikace postačí. Postupem času probíhalo testování na jiných, aktuálnějších modelech, jejichž výsledky jsou níže srovnány.

Aplikace je sestavena ze 4 hlavních aktivit, se kterými pracuje. Některé aktivity jsou potom extendovány o jiné, což umožňuje použití třídních funkcí a proměnných z rozšiřující aktivity. Aplikace také při spuštění vyvolá inicializaci či načtení databáze. Layouty aktivit mají obsah definován v jednotlivých xml souborech, jež se pojí s příslušnými aktivitami. Části implementace zaobírající se detekcí, sledováním pozice a napojením kamery čerpají z open-source projektů TensorFlow, spadajících pod licenci APACHE 2.0 a funkcí knihovny TensorFlow.

Jednotlivé aktivity

Třídu `BaseActivity` lze považovat za aktivitu pojící se s ostatními a následně tvořící funkční celek. Pracuje s uživatelským rozhraním menu, ve kterém dochází k přepínání aktivit z kamery do galerie, infa či opačně. To se děje na základě fuknce `startActivity()`, která spustí daný intent (tedy invokuje jinou aktivitu) podle zvoleného tlačítka. Dalším řešeným problémem podléhající této třídě je inicializace a samotné určení polohy pomocí knihovny `smartLocation`. Určení polohy běží neustále dokola a přepočítává se, přičemž při pořízení fotografie v hlavní aktivitě dochází k uložení právě poslední možné.

`GalleryActivity` vykresluje pořízené fotografie z databáze do instance fragmentu. Třída obsahuje dvě tlačítka `delete` a `showMap`, která při stisku vykonají požadované akce s aktuální fotografií. V případě `delete` se zavolá funkce `deletePhoto`, která vymaže fotografii a na základě `notifyDataSetChanged()` obnoví data, ze kterých je fragment sestavován. Druhé tlačítko pomocí databázových proměnných GPS souřadnic (jsou-li k dispozici) umožňuje otevřít takzvaný nový intent (aplikaci google maps), která lokalizuje pořízenou fotografii. V případě nesprávnosti či nedostupnosti dojde k vypsání informativního pole s chybovou hláškou. Stejně tak je tomu v případě snahy o lokalizaci a mazání v případě prázdné galerie.

Třída také pracuje se souřadnicemi uloženými v databází společně s fotografií. Konkrétně tedy v cyklu prochází detekovaná ohraničení a jednotlivě dochází k jejich vykreslování v rámci instance třídy `MyRectangle`. Tato třída implementuje pouze kreslení pomocí metod třídy `Canvas`. Metoda `void drawRect(float left, float top, float right, float bottom, Paint paint)` vykreslí čtverec na základně čtveřice bodů a jiných kreslících parametrů. Jelikož pořízené fotografie disponují rozlišením o velikosti 640x480px, bylo třeba přepočítat pozice bodů individuálně pro jednotlivá zařízení a potom využít transformaci zvětšení – multiplikátor byl určen na základě rozlišení fotografie, šířky displeje zařízení a relativní výšky fragmentu galerie na konkrétním zařízení. Dalším problémem lze označit nastavení jejich pozic. Protože se rámce vykreslovaly naopak, musely být transformací překlopeny skrze osu x.

Třída `MainActivity` pracuje s náhledem kamery, samotnou detekcí a pořízením fotografií. Na začátku při spuštění dochází instanciaci fragmentu obsluhující kameru. Následně se volí parametry – velikost vůči displeji, rotace, načtení detektoru a sledování polohy.
Ve funkci `onImageAvailable(ImageReader reader)` běží neustále obraz. Jednoduchá řídící stavová logika zajišťuje synchronizaci snímání s tlačítkem. V momentě, kdy dojde k sepnutí tlačítka se aktuální obraz pošle do funkce`saveImage()`. Funkce se stará o vytvoření souboru, uložení obrazu (třída `imageSaver`) a sním také uložení veškerých dat do databáze fotografií společně s cestou, gps souřadnicemi a podobně. Každá fotografie obsahuje databázi souřadnic detekovaných ohraničení. Metoda `processImage()` potom ve vlákně na pozadí pouští detekci, sledování pozice a vykreslování na základě níže uvedených TensorFlow tříd. V dolní části obrazovky se nachází oblast s výpisem inference time – časové náročnosti jednotlivých snímků detekce v milisekundách. Systémová metoda `SystemClock.uptimeMillis()` poskytuje aktuální hodnotu. Výpočet je dán rozdílem času startu detekce a času jejího konce. Současně v aktivitě také po uložení fotografie vzniká nový soubor formátu XML. Obsahem jsou anotace vyfocených tagů. Fotografie i anotace jsou pojmenovány na základě aktuálního času jejich pořízení – pro případnou snadnou budoucí manipulaci. Zapisování anotací do souboru probíhá obdobně jako ukládání do databázi, tedy vyvoláním cyklu detekcí dané fotografie a následným skládáním stromu.

Databáze

Třída `App` se stará o inicializaci databáze v aplikaci. Jako databázový systém byla zvolena Realm Database. Ta je dnes populárním a často používaným řešením pro mobilní platformy.

`PhotoObject` obsahuje data a metody pro práci s konkrétní fotografií. Mezi metody, které databáze používá patří metody pro získání a přiřazení hodnot atributům fotografií, cest a přepočet následujícího ID.

Další databázová třída `DetectionPosition` prezentuje objekty pro jednotlivé detekce dané fotografie – obsahuje tedy primární klíč a čtyři souřadnice rámce.

Kamera a detekce

Neuronová síť ve kvantizovaném formátu tflite a textový soubor nesoucí label se řadí mezi aktiva (z původního assets), která jsou načtena za pomocí funkce knihovny TensorFlow.

Třída `CameraFragment` je fragment, starající se o náhled kamery v hlavní aktivitě a další práci s ní. Dochází zde například k načtení kamery, jejího otevření, uvolnění, určení velikostí náhledu a podobně. Samotná detekce je poměrně náročná na hardware telefonu, zvlášť u starších zařízení, tudíž velikost výsledné fotografie byla stanovena na 640x480 bodů. Instanciace tohoto fragmentu probíhá tedy ve třídě `MainActivity`. Rozlišení lze považovat dostačující pro další trénování, avšak lze jej ovlivnit změnou konstant právě v tomto fragmentu, přičemž se také přizpůsobí anotace fotografií. Jelikož spuštěná aplikace začíná aktivitou kamery a zmíněnou instanciací tohoto fragmentu, bylo nutné při volání funkce `openCamera()` ošetřit oprávnění pro přístup ke kameře, úložišti a polohovým službám.

Ostatní metody a třídy (pro tracking objektů apod.) jsou modifikací příkladu TensorFlow. Licence a významné změny jsou přiloženy ve zdrojových souborech.

Testované zařízení	Čas při 1 vláknu a 1 iteraci	Průměrný čas při 1 vláknu a 50 iteracích	Čas při 4 vláknech a 1 iteraci	Průměrný čas při 4 vláknech a 50 iteracích
LG L Fino	867 ms	841 ms	484 ms	367 ms
Huawei Y6	915 ms	839 ms	412 ms	368 ms
Huawei P8 lite	567 ms	558 ms	274 ms	248 ms
HTC one X	677 ms	816 ms	603 ms	347 ms
HTC One M8	249 ms	201 ms	453 ms	99 ms
HTC One M9	466 ms	459 ms	218 ms	132 ms
CAT S60	455 ms	437 ms	251 ms	194 ms
NEXUS 5	237 ms	199 ms	504 ms	78 ms
NEXUS 5X	217 ms	210 ms	161 ms	130 ms
Galaxy S7 Edge	584 ms	246 ms	121 ms	122 ms
LENOVO G50-70	804 ms	785 ms	445 ms	447 ms

Tabulka 5.1: Časová náročnost natrénovaného modelu.

5.4 Dosažená přesnost a časová náročnost

Nejprve je třeba zhodnotit, jakých výsledků náš model dosahoval a jak se lišily v případě různých parametrů a zařízení. Během trénování docházelo často k experimentování v podobě úpravy vstupních parametrů a hledání optimálního řešení. V podkapitole 4.2 již bylo vysvětleno finální nastavení modelu. Při větším počtů kroků přesnost mAP klesala. Docházelo tedy k přetrénování sítě. Při zahájení kvantizace v dřívějších krocích přesnost také klesala, avšak čas nutný pro vyhodnocení detekce sítě (její inference time) se prakticky nijak neměnil.

Dosažená přesnost mAP

V případě použití nekvantizované sítě `MobileNet_v2` s IoU prahem 0.6 bylo dosaženo přesnosti mAP **73.5%**. V případě **kvantizace** při stejném IoU přesnost dosáhla **68.5%**. Dále došlo na testování při změně prahu IoU na hodnotu 0.51, 0.75 a 0.9. Při první hodnotě byl výsledek **69%**. Při hodnotě 0.75 se přesnost rovnala **66.3%**, závěrem při hodnotě 0.9 pak **51.6%**.

Spotřebovaný čas

Dalším zajímavým bodem experimentů je čas zpracování, při kterém dokáže model vyhodnotit případná pozitiva či negativa. K vyhodnocení těchto experimentů posloužil TensorFlow benchmark, který je dělaný právě na posouzení modelů formátu tflite. Bylo třeba nahrát model a benchmark do testovaných zařízení a zahájit experimenty. Na tabulce ukázkových experimentů TensorFlow [32], jsou vidět časy trvání detekce `MobileNet_v1` na zařízeních Pixel 2 a Pixel XL. Tabulka 5.1 zobrazuje již vlastní experimenty s natrénovaným modelem.

Jelikož dostupnost různých a nejnovějších mobilních zařízeních nebyla úplně možná, nebylo možné zhodnotit síť vůči zmíněným modelům Pixel. Pro prvotní vývoj došlo k aplikování `MobileNet_v1`, posléze pak k nasazení druhé verze.

Zařízení	Čas při 1 iteraci	Průměrný čas 50 iteracích
Huawei Y6	905 ms	864 ms
Huawei P8 lite	486 ms	434 ms
HTC One M8	857 ms	770 ms
HTC One M9	349 ms	317 ms
CAT S60	455 ms	437 ms
NEXUS 5	485 ms	552 ms
NEXUS 5X	236 ms	216 ms
Samsung Galaxy S7 Edge	54 ms	98 ms

Tabulka 5.2: Časová náročnost natrénovaného modelu v rámci aplikace.

Benchmark lze spustit při více běžících vláknech. Větší počet vláken klade větší požadavky na výkon a energetickou náročnost, avšak odezva, tedy výsledek, který nás primárně zajímá, je mnohem přijatelnější. Z výše testovaných zařízení vyplývá, že nejlépe se umístil mobilní telefon **Samsung Galaxy S7 Edge**.

Co se týče časové náročnosti detekce měřené v aplikaci, měření proběhlo v rámci jedné a také padesáti iteracích, přičemž aplikace pracuje s jedním vláknem v případě komunikace s uživatelem. Handler potom spouští vlákno v pozadí, kde probíhá detekce a podobně. Všechny testy probíhaly při rozlišení 640x480 pixelů. Výsledky jsou zachyceny v tabulce 5.2.

5.5 Testování a chování aplikace

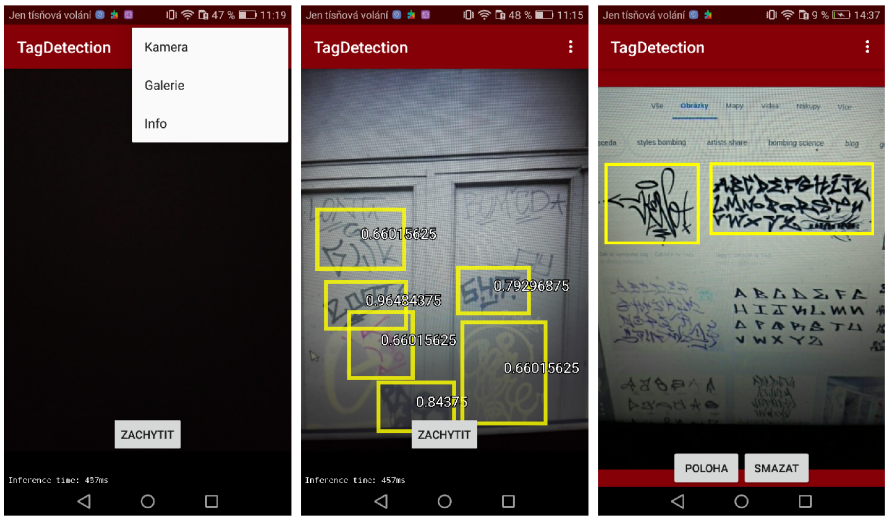
Sekce popisuje neočekávané chování aplikace, její podobu, nasazení a využití aplikace v praxi, finálně pak testování.

Chování aplikace

Problémem je raritní mylná detekce, zapříčiněna dostupnými nasbíranými daty. Při namíření kamery na zkroucenou kabeláž došlo k mylnému výsledku, ačkoli se nejedná o graffiti tag. Celkový dojem z přesnosti je však pozitivně překvapivý. Dalším nepatrným problémem může být povolování oprávnění. V případě některých zařízení aplikace selhávala. Tento problém by měl být v současnosti eliminován, avšak nebyl testován na problémových zařízeních. Při případné nestabilitě je tudíž třeba v nastavení aplikace povolit přístup ke kameře, úložišti a sdílení polohy.

Publikování aplikace

Platforma Android využívá pro distribuci aplikací online obchod – Google Play, který využívají milióny uživatelů. Tento způsob usnadňuje přístup jak na straně klienta, tak na straně vývojáře. Bohužel pro publikování aplikace je třeba zaplatit poplatek 25$, tudíž lze aplikaci získat pouze prostřednictvím přímého nahrání do telefonu.



Obrázek 5.1: Finální podoba aplikace.

Finální podoba aplikace

Aplikace disponuje jednoduchým uživatelským rozhraním. Vizuální podoba je zachycena na obrázku 5.1. Ikona byla navrhnuta v 5 rozlišeních kvůli variabilitě dpi na různých zařízeních. Uživatelské rozhraní aplikace je přizpůsobeno dvěma jazykům. Volba toho jazyku proběhne na základě nastavení jazyku telefonu. Konkrétně tedy aplikace poskytuje české a anglické uživatelské rozhraní.

Uživatelské testování aplikace

Nejčastěji byl k testování aplikace využit telefon Huawei P8 lite. Dále byla aplikace rozšířena mezi několik studentů s novějšími zařízeními. Jednalo se především o běžné uživatelé mobilních zařízení. Většině z nich se líbila detekce v reálném čase, lokalizace fotografie. Ovládání všem přišlo velice intuitivní. Dalším pozitivním prvkem testování je rozšíření povědomí o fungování neuronových sítí mezi běžnou populaci. Naopak někteří uživatelé občasně nedokázali porozumět smyslu a výhodě získávání anotací a sběru dat kvůli neznalosti principů procesu učení.

Kapitola 6

Závěr

V bakalářské práci byl poskytnut základní pohled na problematiku strojového učení. Práce popsala jak teoretické disciplíny a principy, tak proces trénování a následného praktického aplikování neuronových sítí na mobilní zařízení platformy Android.

Z pohledu výsledků experimentů, naměřené hodnoty modelu (rychlost a přesnost) jsou v souladu s průzkumem, neuronová síť `MobileNet_v2` dosáhla přesnosti **73.5%** po kvantizaci pak **68.5%**. Nejlepšího naměřeného času pak dosáhl telefon Samsung Galaxy S7 Edge spolu s Nexusem 5X.

Závěrem práce je funkční mobilní aplikace sloužící ke sběru dat na základě předchozí detekce. Aplikace by tedy měla ušetřit čas strávený nad anotováním robustních datasetů. Díky způsobu implementace a načtení neuronové sítě je možno aplikaci přizpůsobit k detekci mnoha jiných tříd a objektů. Tohoto lze dosáhnout pouhou záměnou modelu a seznamu tříd.

Na práci by bylo možno navázat rozšíření aplikace o možnosti sdílení dat se serverem a také o nasazení širšího spektra detektorů. Graffiti tagy by se daly klasifikovat na konkrétní autory (v závislosti na nasbíraných datech). Další možností by mohlo být rozpoznávání konkrétního textu s rozšířením na více tříd.

Co se týče usnadnění práce uživatelů, aplikaci lze rozšířit o možnost výběru rozlišení fotografie přímo vně aplikace, případné implementaci mapy lokalit, kde již uživatel byl. Dalším elementem by mohlo být rozpoznávání textu z videí či fotografií na základě interní galerie zařízení.

Dle mého názoru je na aplikaci zajímavá real-time detekce, která ukazuje jak přesná síť dokáže být a to za minimální odezvy (v případě moderních zařízení). Nejvíce si však cením osvojení hned několika témat, kterými mě práce zaujala. Dostalo se mi zkušeností komplexního charakteru, jež věřím, že mi budou dobrým základem pro budoucí studium či praxi.

Literatura

[1] Abadi, M.; Agarwal, A.; Barham, P.; aj.: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015, [Online; navštíveno 14.01.2019].
URL https://www.tensorflow.org/

[2] Android: *Understand the Activity Lifecycle*. [Online; navštíveno 18.02.2019].
URL https://developer.android.com/guide/components/activities/activity-lifecycle

[3] Anon: *Android Operating System*. [Online; navštíveno 18.02.2019].
URL https://www.techopedia.com/definition/25106/android-operating-system

[4] Anon: *Evaluation measures (information retrieval)*. [Online; navštíveno 07.01.2019].
URL https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision

[5] Anon: *ICR - Intelligent Character Recognition*. [Online; navštíveno 08.02.2019].
URL https://abbyy.technology/en:features:ocr:icr

[6] Anon: *Počítačové vidění*. [Online; navštíveno 12.01.2019].
URL https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%A9_vid%C4%9Bn%C3%AD

[7] Chitradevi, B.; P.Srimathi: *An Overview on Image Processing Techniques*. [Online; navštíveno 07.01.2019].
URL http://www.rroij.com/open-access/an-overview-on-image-processing-techniques.pdf

[8] Cohen, G.; Afshar, S.; Tapson, J.; aj.: *EMNIST: an extension of MNIST to handwritten letters*. [Online; navštíveno 08.02.2019].
URL https://arxiv.org/pdf/1702.05373.pdf

[9] Cornelisse, D.: *An intuitive guide to Convolutional Neural Networks*. [Online; navštíveno 16.01.2019].
URL https://medium.freecodecamp.org/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050

[10] Donges, N.: *Gradient Descent in a Nutshell*. [Online; navštíveno 16.01.2019].
URL https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0

[11] Foley, D.; O'Reilly, R.: *An Evaluation of Convolutional Neural Network Models for Object Detection in Images on Low-End Devices*. [Online; navštíveno 06.02.2019].
URL http://ceur-ws.org/Vol-2259/aics_32.pdf

[12] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, [Online; navštíveno 16.01.2019].
URL http://www.deeplearningbook.org

[13] Grother, P. J.: *NIST Handprinted Forms and Characters Database*. [Online; navštíveno 08.02.2019].
URL https://www.nist.gov/srd/nist-special-database-19

[14] Guo, Y.: *A Survey on Methods and Theories of Quantized Neural Networks*. [Online; navštíveno 02.02.2019].
URL https://arxiv.org/pdf/1808.04752.pdf

[15] Hollemans, M.: *MobileNet version 2*. [Online; navštíveno 14.01.2019].
URL http://machinethink.net/blog/mobilenet-v2/

[16] Howard, A. G.; Zhu, M.; Chen, B.; aj.: *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. [Online; navštíveno 28.01.2019].
URL https://arxiv.org/pdf/1704.04861.pdf

[17] Huang, J.; Fathi, A.; Fischer, I.; aj.: *Speed/accuracy trade-offs for modern convolutional object detectors*. [Online; navštíveno 23.01.2019].
URL http://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_SpeedAccuracy_Trade-Offs_for_CVPR_2017_paper.pdf

[18] Hui, J.: *mAP (mean Average Precision) for Object Detection*. [Online; navštíveno 09.01.2019].
URL https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[19] Šárka Kabátová: *Proč přemalovali Lennonovu zeď? Přišlo nám dobré ji vyčistit, říká student*. [Online; navštíveno 05.01.2019].
URL https://www.lidovky.cz/domov/proc-premalovali-lennonovu-zed-prislo-nam-dobre-ji-vycistit-rika-student.A141118_162102_ln_domov_ele

[20] LeCun, Y.; Cortes, C.; Burges, C. J.: *THE MNIST DATABASE of handwritten digits*. [Online; navštíveno 08.02.2019].
URL http://yann.lecun.com/exdb/mnist/

[21] Liu, W.; Anguelov, D.; Erhan, D.; aj.: *SSD: Single Shot MultiBox Detector*. [Online; navštíveno 26.01.2019].
URL https://arxiv.org/pdf/1512.02325.pdf

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

[23] Pavlica, J.: *Detekce graffiti tagů v obraze*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2017.

[24] Pay, B.: *What is IWR? (Intelligent Word Recognition) How is it Related to Document Management?* [Online; navštíveno 08.02.2019].
URL https://www.efilecabinet.com/what-is-iwr-intelligent-word-recognition-how-is-it-related-to-document-management/

[25] Reingold, E.: *Artificial Neural Networks Technology.* [Online; navštíveno 14.01.2019].
URL
http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural4.html

[26] Rosebrock, A.: *Intersection over Union (IoU) for object detection.* [Online; navštíveno 07.01.2019].
URL https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[27] Sandler, M.; Howard, A.: *MobileNetV2: The Next Generation of On-Device Computer Vision Networks.* [Online; navštíveno 14.01.2019].
URL
https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html

[28] Schmidt, A.: *A Single Layer Network.* [Online; navštíveno 15.01.2019].
URL https://www.teco.edu/~albrecht/neuro/html/node17.html

[29] Schmidt, A.: *Multilayer Neural Network.* [Online; navštíveno 14.01.2019].
URL https://www.teco.edu/~albrecht/neuro/html/node18.html

[30] Sharma, S.: *Activation Functions in Neural Networks.* [Online; navštíveno 14.01.2019].
URL https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[31] Skymind: *A Beginner's Guide to Neural Networks and Deep Learning.* [Online; navštíveno 27.01.2019].
URL https://skymind.ai/wiki/neural-network#define

[32] TensorFlow: *Performance benchmarks.* [Online; navštíveno 18.01.2019].
URL https://www.tensorflow.org/lite/performance/benchmarks