VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE ZBRANÍ V 2D OBRAZU

DETECTION OF WEAPONS IN 2D IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JÁN DEMČÁK**
AUTHOR

VEDOUCÍ PRÁCE **Prof. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.**
SUPERVISOR

BRNO 2018

Abstrakt

Táto bakalárska práca sa zaoberá detekciou zbraní v 2D obraze. V teoretickej časti bol zadefinovaný pojem zbraň, uviedli sme možnosti detekcie zbrane v obraze s použitím klasických metód a s využitím hlbokých neurónových sietí. Popísali kľúčové kroky spracovania obrazu, klasifikácie objektov, detekcie. Uviedli prehľad nástrojov, knižníc. Pre realizáciu praktickej časti sme vybrali 3 modely. Jeden klasický model s využitím HOG transformácie. Druhý CNN model s prioritným cieľom presnosťi detekcie a dvoma rôznymi architektúrami neurónových sietí ako klasifikátormi. Tretí model s architektúrou siete YOLO mal ako prioritný cieľ detekciu v reálnom čase. Podstatnou časťou každého modelu bol výber, resp. vytvorenie vhodného datasetu. Nasledovalo zostrojenie a implementácia modelov s analýzou a vyhodnotením získaných dát.

Abstract

This bachelor thesis deals with detection of weapons in 2D image. In the theoretical part of the thesis the term weapon was defined and the possibilities of detection of weapons in image with using classic methods and deep neural networks were mentioned there. The key steps of image processing, objects classification and detection were described. The overview of frameworks, libraries was presented. To implement the pratical part of the thesis, 3 models were chosen. The first classic model with using HOG transformation. The second CNN model with priority target detection accuracy and with two different neural network architectures as classifiers. The third model with YOLO network architecture had as priority target real-time detection. The essential part of each model was choice, or more precisely creating suitable dataset. What followed was the construction and implementation of models and the evaluation of obtained data.

Kľúčové slová

detekcia, 2D obraz, zbraň, konvolučná neurónová sieť, spracovanie obrazu, klasifikátor,dataset

Keywords

detection, 2D image, convolutional neural network, image processing, classifier, dataset

Citácia

DEMČÁK, Ján. *Detekce zbraní v 2D obrazu.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.

Detekce zbraní v 2D obrazu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Prof. Ing., Dipl.-Ing. Martina Drahanského, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Ján Demčák
16. mája 2018

Poďakovanie

Týmto by som rád poďakoval Prof. Ing., Dipl.-Ing. Martinovi Drahanskému, Ph.D za ochotu, poskytnutú pomoc a odborné vedenie.

Obsah

Kapitola 1

Úvod

Bezpečnosť je dôležitou súčasťou našich každodenných životov. Medzi zvlášť závažné zločiny patrí kriminalita páchaná za pomoci strelnej zbrane. Je to globálny problém, obzvlášť v krajinách, kde je alebo v minulosti bolo držanie strelnej zbrane legálne. Prirodzenou funkciou bezpečnostných zložiek je redukcia miery kriminality na najnižšiu možnú úroveň s dôrazom na prevenciu, resp. minimalizáciu potenciálnych hrozieb. Jedným zo sprievodných javov prudkého rozvoja technológií a informatizácie spoločnosti v posledných rokoch je aj masový nárast výskytu zariadení používaných na snímanie obrazu, ako napríklad bezpečnostné kamery. Po celom svete sa profesionáli z oblasti bezpečnosti a ochrany venujú sledovaniu a analýze obrazu z videokamier s cieľom prevencie či eliminácie kriminálnej činnosti. Jednou z najčastejších úloh je včasné vizuálne odhalenie, že monitorovaná osoba má zbraň – obzvlášť zbraň strelnú. Ak by pomocou počítačového videnia bolo možné adekvátne nahradiť čo i len zlomok času a ľudskej práce investovanej do tejto činnosti, úspory by boli obrovské. Okrem toho existuje veľká pravdepodobnosť, že vhodne zvolené použitie softvérových nástrojov na základe počítačového videnia dokáže v snímanom obraze odhaliť mnohé udalosti, ktoré ľudské oko nezachytí, resp. človek nepostrehne.

V tejto bakalárskej práci sa budeme venovať konkrétnemu segmentu tejto problematiky, a to detekcii strelnej zbrane v obraze so zameraním predovšetkým na výber najvhodnejších parametrov pre detekciu za použitia rozličných algoritmov využívaných v tejto oblasti. Keďže pojem strelná zbraň v sématike prirodzeného jazyka zodpovedá množine objektov, ktoré môžu byť výrazne heterogénne, rozhodli sme sa, že pre teoretický návrh detekčných modelov a praktickú implementáciu budeme pod týmto pojmom chápať ručnú palnú krátku strelnú zbraň, teda dominantne pištole a revolvery, ktoré budeme detekovať. Tento výber nie je náhodný, ale podmienený štatisticky doloženým použitím tohto druhu zbraní pri páchaní závažnej trestnej činnosti a existenciou dostupných databáz (datasetov) obrazov zbraní nevyhnutných pre praktickú časť tejto práce.

V tejto práci sme si stanovili dva hlavné ciele. Prvým z nich bolo po podrobnej teoretickej príprave a analýze dostupných nástrojov (knižníc, datasetov a pod.) navrhnúť, zostrojiť, testovať a vyhodnotiť funkčný detekčný model pre detekciu zbrane v 2D obraze. Nakoniec sme sa rozhodli a aj skutočne implementovali 3 rôzne takéto modely.

Druhým hlavným cieľom bolo reálne aspoň trošku prispieť do riešenia zvolenej problematiky, či už funkčným modelom s pozitívnym výsledkom, zostrojením datasetu alebo preskúmaním slepej uličky implementáciou funkčného modelu s negatívnym výsledkom.

V teoretickej časti sme sa krátko venovali metódam detekcie zbraní založených na odlišných fyzikálnych princípoch snímania vstupných dát. Nami vybratú metódu vizuálneho snímania signálu pomocov kamier či podobných zariadení a následného počítačového spra-

covania sme podrobne rozpracovali v tretej kapitole. V nej je popísaný proces spracovania obrazu v jednotlivých fázach - digitalizácie obrazu, predspracovania, následný proces segmentácie dát, čo vedie k analýze obrazu a získaniu objektov. Nakoniec vzniká popis charakteristických vlastností objektu, na základe ktorého sa za použitia vhodného algoritmu klasifikuje daný objekt a vykonáva detekcia. Nezanedbateľný priestor sme venovali aj aktuálne dostupným architektúram konvolučných neurónových sietí, existujúcim datasetom s požadovaným obsahom a nástrojom typu knižnice a podobne.

Na základe poznatkov získaných v teoretickej časti sme si pre tvorbu a implementáciu modelov stanovili tieto parciálne ciele: zostrojiť ako klasický tak aj CNN model, použiť prvok, ktorý ešte podľa dostupných zdrojov pre detekciu zbraní nebol použitý (v našom prípade HOG deskriptor a Yolo sieť), modifikovať existujúci dataset a podľa návodu vytrénovať vlastnú CNN sieť. Stanovené ciele sa nám v plnej miere podarilo realizovať. Popis implementácií a experimentálne výsledky sú k dispozícii v záverečnej časti tejto práce.

Kapitola 2

Úvod do problematiky detekcie zbraní

Nie každú zbraň a nie na každom mieste je potrebné detekovať. Príkladom sú legálne držaná puška na poľovačke, pištoľ na strelnici, či kuchynský nôž v zásuvke. Vo veľkej väčšine prípadov je zmysluplné obmedziť sa na vybrané verejné alebo špeciálne priestory, kde je v zásade možné legálne inštalovať zariadenie umožňujúce detekciu a zároveň predpoklad zvýšeného bezpečnostného rizika. Z reálnej praxe môžeme uviesť skenery kovov na súdoch alebo letiskách, bezpečnostné kamery na čerpacích staniciach či frekventovaných verejných priestranstvách.

Rovnako tak nebudeme pracovať so všetkými možnými typmi zbraní, ale len s určitou cielene vybranou skupinou. Dôvody tohto výberu, stručné rozdelenie zbraní a definícia základných pojmov je obsahom tejto kapitoly. V jej druhej časti uvedieme prehľad vybraných existujúcich riešení v oblasti detekcie zbraní, stručne opíšeme funkčný princíp jednotlivých riešení tejto problematiky, porovnáme hlavné výhody a nevýhody a na záver urobíme krátke zhrnutie.

2.1 Definícia zbrane

Obvyklá definícia zbrane hovorí, že zbraň je nástroj, predmet alebo celé zariadenie, ktoré je prispôsobené k vyvolaniu ranivého účinku na živý organizmus alebo k ničeniu objektov [8].

2.2 Rozdelenie zbraní

Podľa toho, ako zbraň pôsobí na živú silu, rozdeľujeme zbrane na:

- **Úderné** - pôsobia na živý objekt tupým úderom svojej časti, ktorá býva spojená s vhodnou rukoväťou.
- **Chladné** - účinujú bodom alebo sekom naostrenej čepele.
- **Strelné** - rozrušujú vzdialený cieľ, živý či neživý prostredníctvom dopadovej energie strely vystrelenej zo zbrane.

Strelné zbrane sa ďalej delia podľa zdroja energie použitej k vypudeniu projektilu na:

- **Palné** - zdrojom pre výmet strely je okamžité uvoľnenie chemickej energie obsiahnutej v strelnom prachu, prípadne v samotnej nárazovej zloži. Horením sa energia expandujúcich plynov mení v hlavni na pohybovú energiu strely.
- **Mechanické** - zdrojom pre výmet strely je mechanická energia akumulovaná v pružnom médiu. Patrí sem napríklad prak alebo luk.
- **Plynové** - strelné zbrane, v ktorých je strela uvedená do pohybu energiou mechanicky stlačeného vzduchu alebo plynu.

Z hľadiska ovládateľnosti a prenosnosti sa delia strelné zbrane na:

- **Ručné** - môže ich prenášať a ovládať jedna osoba. Sú buď ovládateľné jednou rukou, tzv. krátke zbrane, alebo obomi rukami, tzv. dlhé zbrane. Medzi krátke zbrane patria predovšetkým pištole a revolvery. Dlhé zbrane sú vybavené pažbou, ktorá sa prikladá k ramenu alebo k lícu strelca.
- **Lafetované** - strelné zbrane kalibru väčšieho ako 20 mm vrátane, ktoré zväčša nemajú ovládacie prvky ručných strelných zbraní a vzhľadom na rozmery, hmotnosť alebo spôsob konštrukčného riešenia sú montované na stacionárne alebo mobilné lafety.

Podľa konštrukcie a režimu streľby sa palné poprípade aj plynové zbrane delia na:

- **Jednoranové**
- **Opakovacie**
- **Samonabíjacie**
- **Samočinné**

Každá ručná palná zbraň sa väčšinou skladá z niekoľko základných častí: hlavne, záveru, úderného, spúšťového a poistného ústrojenstva, vyťahovacieho a vyhadzovacieho ústrojenstva, zásobovacieho ústrojenstva, úchopovej časti a mieridla [8].

Chladné zbrane boli, a stále sú smrtiacim nástrojom. V našom kultúrnom priestore sú však v súčasnosti skôr zbraňou použitou v afekte pri neplánovaných skutkoch spáchaných často v súkromí ako prostriedok všeobecného ohrozenia alebo organizovanej trestnej činnosti. Naopak to platí o ručných palných strelných zbraniach. Tie boli použité pri väčšine vrážd, únosov, lúpežných prepadnutí i aktoch terorizmu. Je to jeden z podstatných dôvodov, prečo sa na ich včasnú a presnú detekciu v mnohých reálnych situáciách kladie stále väčší dôraz, čomu okrem iného zodpovedajú investície do vývoja detekčných zariadení, rozvoja detekčných metód i tvorbe databáz (datasetov) prakticky v celom civilizovanom svete. Je to i podstatný dôvod, prečo sa v tejto práci budeme cielene venovať detekcii palných strelných zbraní. Vzhľadom na ďalšie špecifiká, napr. dostupnosť vhodných datasetov, sme výber ešte zúžili na krátke zbrane. V ďalšom texte práce, ak nebude výslovne uvedené inak, budeme pod pojmom zbraň mať na mysli palnú strelnú ručnú krátku zbraň – štandardne pištoľ alebo revolver.

2.3 Detekcia zbraní

V tejto práci sa úplne vyhneme detekčným metódam založených na využití ľudských zmyslov. Perfektne vycvičený človek má stále svoje miesto pri identifkácii ohrozenia včítane vizuálnej detekcie zbrane, ale len vo veľmi špecifických prípadoch. V dnešnom bežnom svete je denne potrebné realizovať milióny detekcií, a to už nie je možné bez vhodných prístrojov a zobrazovacích metód. Kedže detekcia zbraní pomocou počítačového videnia je pomerne rozsiahla téma a má širokospektrálne využitie, rozdeľujeme práce venujúce sa tejto problematike do troch skupín, a to na základe špecifického využitia daného detekčného systému a použitých metód strojového učenia.

2.3.1 Detekčné metódy využívajúce mikrovlnné a infračervené pásmo. Detekcia skrytých zbraní

Prvá, v súčasnosti pravdepodobne najrozšírenejšia skupina detekcie zbraní sa zameriava na detekciu skrytých zbraní prístrojmi a detektormi využívajúcimi fyzikálne princípy. Najreprezentatívnejšou aplikáciou v bežnom živote v tejto súvislosti je kontrola osôb a batožiny na letiskách. Sú vysokoúčinné v situáciách, keď je možné zabezpečiť pohyb osôb (predmetov) cez vopred určený priestor snímaný príslušnými detektormi.

V situáciách, keď nieje možné zabezpečiť tok ľudí kontrolným procesom je žiadúce, aby bolo možné detekovať skrytú zbraň na diaľku. K tomu sú potrebné príslušné zobrazovacie systémy na skryté objekty. Podľa závislosti na osvetlení rozdeľujeme zobrazovacie systémy na aktívne, čiže tie, ktoré sú závislé na osvetlení a pasívne. Väčšina systémov závislých na osvetlení využíva nízkoenergetickú radiáciu (angl. *low-power radiation*), ktorá so sebou nesie určité právne problémy. Príkladom môže byť povinné umiestňovanie upozornení, ktoré by mohlo varovať prípadných nositeľov skrytých zbraní, a tak znížiť účinnosť a užitočnosť detekčných systémov, alebo žiadanie súhlasu od ľudí, k podrobeniu detekčnej kontrole. Preto sa väčšina štúdií zameriava na použitie pasívnych systémov. Sem môžeme zaradiť systémy využívajúce pasívne milimetrové vlny (angl. *passive millimeter wave imaging*) ,ďalej len MMV alebo IR (angl. *infrared*) systémy [4].

Zobrazovače infračerveného žiarenia sa používajú v aplikáciách pre nočné videnie ako napríklad zobrazovanie osôb. Využívajú informáciu o distribúcii teploty v obraze, resp. energiu, ktorá je vyžarovaná ľudským telom.Tá je absorbovaná odevom a následne vyžiarená do okolia. Pre identifikáciu ukrytého predmetu je pri tejto metóde ideálne tenké a priliehavé oblečenie. Naopak, hrubé a voľné oblečenie zväčšuje oblasť odevu, do ktorej bude energia rozložená. To znižuje efektivitu rozlíšenia ukrytého predmetu, a teda aj šancu detekovať prípadnú zbraň [4].

Ako metóda IR tak i MMW fungujú na princípe merania určitej formy žiarenia, ktoré objekty vyžarujú alebo odrážajú do okolia. Miera, akou objekt vyžaruje energiu je charakterizovaná emisivitou (ϵ). Detekcia zbraní je možná vďaka nízkej emisivite a vysokej reflektivite, v porovnaní s ľudským telom. Napríklad kovová pištoľ skrytá pod odevom nielenže zablokuje prirodzené emisie z tela, ale odráža aj ožiarenie z okolia. Dokonca aj nekovové predmety, ako sú plasty a keramika, budú mať podobný účinok. [16]. V [34] autori vyvinuli systém na detekciu skrytých zbraní, ktorý využíva práve MMW.

Aj keď IR a MMW systémy sú schopné detekovať skryté zbrane, lepšie detekčné výsledky dosahujú systémy založené na fúzii obrazových dát. Z tohto dôvodu vo väčšine publikácií [28, 7, 35, 4], ktoré sa venujú problematike detekcie skrytých zbraní, sú detekčné systémy

založené práve na fúzii obrazových dát. V [35] autori využívajú fúziu IR a klasického RGB (angl. *Red Green Blue*) obrazu.

Nutnou podmienkou funkčnosti vyššie spomínaných detekčných systémov je použitia IR, MMW alebo obdobných senzorov umiestnených v blízkosti miesta detekcie. Mobilita týchto senzorov je v reálnej praxi výrazne obmedzujúcim faktorom a ich využitie v rozličných prostrediach je preto výrazne limitované.

2.3.2 Detekcia zbrane v oblasti viditeľného spektra za použitia klasických metód

Vyššie uvedené metódy prvej skupiny síce dokážu vidieť aj skryté zbrane, ale len v dosahu svojich senzorov. V praxi je obrovské množstvo situácií, keď potrebné senzorové pokrytie nie je možné, čo značí, že ani zbraň, ktorá je viditeľná, nie je detekovaná. Toto je priestor pre iný typ riešení.

Druhou skupinou sú aplikácie, ktoré sú zamerané na detekciu rôznych typov zbraní v RGB obraze. Z pohľadu použitých nástrojov ich delíme na klasické metódy, tzn. bez použitia neurónových sietí (angl. *neural network*) a metódy založené na využití neurónových sietí. Tie sa ešte ďalej členia podľa použitej architektúry. Najprv uvedieme príklady klasických metód, v ďalšej časti riešenia s použitím neurónových sietí.

Detekcia s využitím SURF

Autori [33, 27] vytvorili dva veľmi podobné, až priam totožné systémy, ktoré sa líšia iba v použití rozdielnych príznakových detektorov a deskriptorov. [33] založil svoju prácu na príznakovom deskriptore a detektore SURF (angl. *speeded up robust features*). V [27] bol použitý FREAK (angl. *Fast Retina Keypoint*) deskriptor a *Harris interest point* detektor.

Systém si na začiatku načíta deskriptor zbraní, následne predspracuje obrázky, tj. odstráni šumy a upraví veľkosť na 400x300 pixelov. Na obrázkoch je následne aplikovaná segmentácia na základe farby za použitia *k-means* zhlukovania. Na výstup tak systém dostáva zhluky farieb, ktoré sú podobné zbraniam (v tomto prípade čierna). Kedže v dôsledku šumu mohli vzniknúť malé oblasti danej farby, pokračuje vyextrahovaním spojitých oblastí, ktorých obsah je väčší ako 1000 pixelov, ktoré týmto krokom odseparuje. Nastáva morfologické uzatvorenie vybraných oblastí, z ktorého detektor vyextrahuje príznaky tvaru objektu, ktoré porovná s už načítaným deskriptorom a ak sa zhoduje aspoň 50% príznakov, systém detekuje objekt ako zbraň.

Zhrnutie Systémy dosahujú veľmi podobné celkové úspešnosti, a to 84.26% a 88.67%. Výhody týchto riešení sú invariantnosť voči veľkosti a rotácii, a taktiež možnosť detekovať viacero objektov v 1 obraze. Kedže systémy najprv aplikujú segmentáciu na základe farby, sú preto výrazne závislé na farbe zbraní a niesú schopné detekovať zbrane atypickej farby.

2.3.3 Detekcia zbrane za použitia hlbokých neurónových sietí

V tejto skupine sa venujeme metódam, ktoré sa zaoberajú detekciou zbraní za použitia hlbokých neurónových sietí. V čase tvorby práce, sme oboznámení s existenciou iba dvoch publikácií, ktoré sa venujú danej problematike a to [26, 14]. Kedže správne trénovanie hlbokých konvolučných neurónových sietí, ktoré obsahujú milióny parametrov, vyžaduje enormne veľkú sadu dát, až milióny vzoriek, a taktiež výkonné výpočtové prostriedky, stalo

sa široko prijímanou alternatívou používanie dolaďovania (angl. *fine-tunning*) pre prekonanie týchto obmedzení [26].

Preto v oboch publikáciách na klasifikáciu používajú VGG-16 model konvolučnej neurónovej siete (CNN), ktorý je predtrénovaný na dátovom sete ImageNet (viac ako jeden milión obrázkov a cez 1000 tried objektov) na rozpoznávanie objektov, a následne je doladený (v oboch prípadoch) približne 3000 obrázkami zbraní.

Faster R-CNN a VGG16 model

V [26] vytvorili autori systém na automatickú detekciu prítomnosti zbrane v 2D. Výsledný systém dosahuje rýchlosť aj pre využitie v reálnom čase (angl. *real-time*). Pri tvorbe systému sa zamerali hlavne na nízky počet falošných poplachov. Pre detekciu využívajú dva rozdielne prístupy a to *sliding window* a *region proposals*. Z dôvodu rozdielov v týchto prístupoch bolo vytvorených viacero rozličných databáz s obrázkami zbraní.

Slidingwindow Pre tento prístup boli celkom vytvorené 4 databázy:

- **1. databáza** pozostáva z 9100 obrázkov rozdelených do 2 tried. Sú rozdelené na 3990 obrázkov triedy zbraň, ktorá obsahuje rozličné typy zbraní ako napríklad pištole, guľomety, granáty a raketomety. Zvyšných 5110 obrázkov pozostáva z rozličných objektov triedy "nie zbraň". Výsledný klasifikačný model preukazuje vysoký počet planých poplachov a nedetekovania zbrane. To možno vysvetliť skutočnosťou, že veľká rozmanitosť zbraní výrazne zťažuje proces učenia. Preto sa vo zvyšku práce rozhodli zamerať iba na detekciu pištolí.
- **2. databáza** pozostáva z 751 obrázkov triedy pištoľ a 1857 obrázkov triedy pozadie, ktorá obsahuje obrázky rúk držiacich iné objekty ako zbraň, napríklad telefón alebo pero. Výsledný klasifikačný model preukazuje vysoký počet planých poplachov,čo bolo vo väčšine prípadov zapríčinené považovaním bieleho pozadia za časť zbrane z dôvodu prítomnosti bieleho pozadia takmer vo všetkých trénovacích dátach.
- **3. a 4. databáza** obsahujú vyšší počet tried. Najlepší výsledok bol dosiahnutý pri databáze č. 4, ktorá obsahovala 200 obrázkov triedy pištoľ a 9061 obrázkov rozdelených do 102 tried.

Klasifikátor bol aplikovaný s krokom 60x60 pixelov a oknom o veľkosti 160x120 pixelov. Pri obrázkoch o veľkosti 640x360 pixelov celá detekcia trvala 1,5 sekundy, čo je čas nevyhovujúci pre real-time detekciu.

Region proposals Pri tomto prístupe bola použitá *Faster R-CNN*, ktorá kombinuje metódu selektívneho vyhľadávania s klasifikátorom na báze VGG-16. Databáza č. 5 obsahuje 3000 obrázkov pištolí. Používa model s 2 triedami, kde na základe súradníc sú ohraničené pištole. Zvyšok objektov v obrázkoch je považovaný za pozadie. Model dosahuje 100% pravdepodobnosť detekcie a počet planých poplachov bol znížený tým, že detekčný alarm sa aktivuje iba ak bola zbraň detekovaná v piatich po sebe nasledujúcich obrazoch (angl. *frame*) vo videu. Výsledná rýchlosť detekcie v obraze o veľkosti 1000x1000 pixelov je 0,19 sekundy.

Porovnanie Porovnaním dostupných publikovaných údajov sme dospeli k záveru, že najoptimálnejšie výsledky boli dosiahnuté modelom na báze Faster R-CNN, ktorý dosahuje rýchlosť využiteľnú aj v real-time aplikáciách a dosahuje postačujúce hodnoty aj pri nízkokvalitných videách. Tieto skutočnosti vytvárajú možnosť využitia aj v praktickej časti tejto práce.

OverFeat model

V [14] autori používajú na detekciu a klasifikáciu konvolučnú sieť OverFeat, ktorá využíva prístup posuvného okna. Systém bol implementovaný pomocou Tensorflow frameworku. Najlepší dosiahnutý model dosahuje 89% presnosť na testovacej sade. Rýchlosť celkovej detekcie na jednom obrázku je 1,3 sekundy, čo však nieje dostatočná rýchlosť pre realime detekciu.

2.4 Zhrnutie

V tejto kapitole sme stručne uviedli rozdelenie zbraní, zadefinovali pojem zbraň pre ďalšie použitie v tejto práci a zdôvodnili konkrétny výber obsahu tohto pojmu. Taktiež sme uviedli dôvody selektívneho výberu oblastí potreby detekcie zbraní. Poskytli sme krátky prehľad typov detekčných metód so záverom ďalej sa sústrediť na počítačovú detekciu zbrane vo vizuálnom obraze. Uviedli sme stručný prehľad existujúcich riešení problematiky a došli k záveru, že zatial najlepšie výsledky z hľadiska rýchlosti dosahujú v [26] pomocou Faster R-CNN.

Kapitola 3

Spracovanie obrazu, klasifikácia a detekcia objektov

Táto kapitola predstavuje teoretický manuál potrebný pre praktickú časť tejto práce. Odborná literatúra uvádza viacero alternatív v popise postupu spracovania obrazu, preto ho nie je možné presne definovať. Je dôležité zabezpečit, aby jednotlivé zvolené kroky boli realizované správne a tiež dodržať následnosť jednotlivých krokov podľa poradia, ktoré určuje konkrétna aplikácia. V celom procese budeme potrebovať pretransformovať získaný signál do vhodnej podoby, čo zahŕňa algoritmy a metódy, ktoré zo vstupného obrazu vyextrahujú potrebné informácie a naopak odseparujú nežiadúce. Taktiež budeme potrebovať opis a klasifikáciu objektov i samotnú detekciu, nevyhnutnú pri určení obrysov a polohy objektu v obraze. Túto časť kapitoly rozpracujeme podrobnejšie s uvedením viacerých typov architektúry neurónových sietí, klasifikátorov a potrebných nástrojov (knižníc) podľa vybranej metódy. Najčastejšie sa uvádza táto postupnosť:

Postupnosť základných krokov [30]:

- digitalizácia obrazu
- predspracovanie obrazu
- segmentácia obrazu
- popis objektov
- klasifikácia
- detekcia

3.1 Digitalizácia obrazu

Prvým krokom spracovania obrazu je jeho digitalizácia. Pod pojmom digitalizácia obrazu rozumieme prevod analógovej informácie na digitálnu. Vstupnou informáciou može byť jas, poprípade ak sa jedná o farebný snímok obrazu tak je to niekoľko spektrálnych zložiek. Tieto zložky rozlišujeme podľa použitého farebného modelu ako napríklad RGB, HSV (angl. *hue, saturation, value*), CMYK (angl. *cyan, magenta, yellow, key*).

Pre popis obrázku sa často používa jeho matematický zápis pomocou spojitej funkcie $f(x, y)$ kde x, y odpovedajú súradniciam bodu obrázka. Aby mohol byť obrázok spracovávaný počítačom, musí byť reprezentovaný pomocou primeranej diskrétnej štruktúry, napríklad matice. Pod digitalizáciou teda rozumieme navzorkovanie funkcie $f(x, y)$ do matice s M riadkami a N stĺpcami, kde každému vzorku spojitej funkcie je kvantovaním do K úrovní priradená celočíselná hodnota [11].

3.2 Predspracovanie obrazu

Druhým krokom po digitalizácii obrazu je jeho predspracovanie. Predspracovanie je spoločný názov pre operácie s obrazom na nízkej úrovni abstrakcie. Vstupom i výstupom metód predspracovania sú obrazové dáta na nízkej úrovni abstrakcie, čiže matice, ktoré predstavujú digitálnu obrazovú funkciu. Predspracovanie nezvyšuje počet informácií, ktoré obraz nesie, avšak je veľmi užitočné v situáciách, keď potrebujeme potlačiť šum, ktorý môže vzniknúť pri digitalizácii obrazu či jeho prenose. Rovnako môžeme potlačiť informácie, ktoré nepovažujeme za relevantné poprípade zvýrazniť tie, ktoré z hľadiska ďalšieho spracovania za dôležité považujeme. Príkladom môže byť hľadanie hrán v obraze, t.j. obrazových bodov s vysokými hodnotami gradientu obrazovej funkcie.

Základné rozdelenie predspracovania obrazu:

- **Transformácia jasu** - patrí sem napríklad prevod na negatív, použitím ktorého vzniká šedotónový obraz alebo roztiahnutie kontrastu.
- **Lokálne predspracovanie** - zaradzujeme sem metódy slúžiace na potlačenie alebo zvýraznenie vyšších frekvencií obrazovej funkcie ako napríklad vyhladzovanie obrazu a gradientné operácie [11].
- **Geometrické transformácie** - patria medzi najdôležitejšie operácie využívané pri spracovaní obrazu. Medzi základné geometrické 2D transformácie patrí skosenie, rotácia, zmena mierky a posun. Kombináciou základných geometrických transformácií vznikajú kombinované geometrické transformácie [19].

3.3 Segmentácia

Obecná definícia segmentácie hovorí, že sa jedná o proces delenia obrazu na časti, ktoré korešpondujú s konkrétnymi objektami v obraze. Inými slovami, každému pixelu obrázku je priradený jeden index segmentu, ktorý odpovedá určitému objektu v obraze. Následnú informáciu o rozdelení objektov do jednotlivých segmentov využívajú vyššie algoritmy spracovania obrazu.

3.3.1 Formálna definícia segmentácie

Segmentácia obrazu $f(h, x)$ je jeho delenie na podobrazy $R_1, R_2, R_3, ..., R_n$ tak, že podobrazy spĺňajú nasledujúce kritériá [19]:

$$\bigcup_{i=1}^{n} R_i = f(x, y) \tag{3.1}$$

$$R_i \cap R_j = \emptyset, i \neq j \tag{3.2}$$

Segmentačné metódy rozdeľujeme na:

- **Metódy vychádzajúce z detekcie hrán** - sa zameriavajú na detekciu významých hrán v obraze. Patrí sem detekcia hrán pomocou prvej derivácie a Houghove transformácie.
- **Region-based techniky** - nedetekujú hrany v obraze, ale priamo oblasti, ktoré by tieto hrany ohraničovali. Jedná sa napríklad o metódu delenia a spájania oblastí alebo metódu šírenia oblastí.
- **Štatistické metódy** - sú založené na zhlukovaní pixelov a priamo závisia na meraniach prevedených nad každým pixelom. Typickým predstaviteľom je K-means zhlukovanie.

3.4 Popis Objektov

Na popis objektov sa využívajú dva modely - kvalitatívny a kvantitatívny. Pri rozpoznávaní objektov tieto popisy slúžia ako vstupná informácia.

3.4.1 Kvantitatívny popis

Kvantitatívny popis využíva číselné charakteristiky objektu. Pod pojem číselné charakteristiky môžeme zahrnúť farebný rozptyl, veľkosť objektu a komparatívnosť.

3.4.2 Kvalitatívny popis

Kvalitatívny popis objektu zvažuje súvislosti medzi danými objektami a slúži pre popis ich tvaru [11].

3.5 Klasifikácia

Klasifikáciou sa rozumie zaradenie objektu do niektorej zo skupiny známych tried. Klasifikáciu rozdelujeme na dve základné oblasti - príznakovú klasifikáciu a štrukturálnu klasifikáciu.

3.5.1 Štrukturálna klasifikácia

Využíva kvalitatívny model popisu objektov. Objekt je charakterizovaný istými vlastnosťami, ktoré sú následne podrobené algoritmom slúžiacim na rozbor slova, kde sa kontroluje syntax, aby mohol byť definovaný jazyk, gramatika a abeceda.

3.5.2 Príznaková klasifikácia

Využíva kvantitatívny popis modelu objektov, čiže využíva čísla, ktoré popisujú daný objekt. Medzi najznámejšie metódy pre popis príznakov patria Haar-like príznaky, HOG (angl. *Histogram of OrientedGradients*), SIFT (angl. *Scale-Invariant Feature Transform*) a SURF (angl. *Speeded Up Robust Feature*). Taktiež pod príznakovú klasifikáciu môžeme zaradiť aj zhlukovú analýzu, ktorá rozdeľuje objekty na základe číselných vlastností do zhlukov [11].

Histogram of OrientedGradients

Základnou myšlienkou je, že objekt môže byť charakterizovaný pomocou intenzity gradientov, aj keď nevieme ich presnú polohu v obraze. Celý obraz je rozdelený do blokov rovnakej veľkosti, kde každý blok je reprezentovaný pomocou menších oblastí, ktoré sa nazývajú bunky. V rámci každej bunky sa určuje orientácia gradientu. Výsledné gradienty by mali vymedzovať hľadaný objekt v obraze. Rozmedzie smeru gradientov (360°) sa rozdelí do niekoľkých kanálov. Z týchto kanálov sa vytvorí lokálny histogram, ktorý obsahuje informácie o zastúpení jednotlivých smeroch gradientov. Výsledný vektor sa vytvorí zložením informácií z histogramov všetkých buniek v danom bloku. Vektor normalizovaných histogramov nazývame deskriptor. Z každého bloku tak získame jeden deskriptor [20].

3.5.3 Klasifikátory

Support Vector Machine (SVM)

Základom metódy je lineárna klasifikácia do dvoch tried. Cieľom je nájsť nadrovinu, ktorá priestor príznakov optimálne rozdelí tak, že trénovacie dáta, ktoré patria rozličným triedam budú ležať v opačných polopriestoroch. Inými slovami optimálna nadrovina je taká, kde minimálne vzdialenosti bodov od roviny sú čo najväčšie. Na popis nadroviny stačia iba body ležiace na jej okraji a tých je obvykle málo. Volajú sa podporné vektory (angl. Support Vectors).

Klasifikácia zahrňuje trénovanie a testovanie dát. Dáta na trénovanie sú určené dvojicou $(\mathbf{x_i}, y_i)$ pre $i = 1, 2, ..., n$ kde, $y_i \in \{-1, 1\}$.
X_i je vektor dimenzie, ktorý popisuje vlastnosti daného prvku. Y_i určuje, do ktorej množiny vektor patrí. Testovacie dáta teda obsahujú iba vektory $\mathbf{x}$, kde cieľom klasifikátoru je určiť jeho príznak y.

Obvykle nieje možné rozdeliť trénovacie dáta lineárne. Preto sa používa namapovanie dát (príznakov) do vyššej dimenzie, kde sa rozdelia nadrovinou. Takto sa z lineárne neseparovateľnej úlohy stáva lineárne separovateľná.

Na zjednodušenie a vyriešenie klasifikačného problému sa používajú jadrové funkcie. Výber funkie je však závislý na povahe trénovacích dát, a častokrát je časovo náročné nájsť najvhodnejšiu funkciu. Najčastejšie aplikovaný postup na výber je metóda pokus omyl. [6]

Neurónové siete

História neurónových sietí siaha až do roku 1943 kedy doktor Warren McCulloch a matematik WalterPitts skúmali možnosti fungovania mozgu a neurónov [17].

Neurónová sieť (ďalej NN) je masívne paralelný procesor, ktorý má sklon k uchovávaniu experimentálnych znalostí a ich ďalšieho využívania. Napodobňuje ľudský mozog v dvoch aspektoch:

- poznatky sú zbierané v NN počas učenia
- medzineurónové spojenia (synaptické váhy - SV) sú využívané na ukladanie znalostí

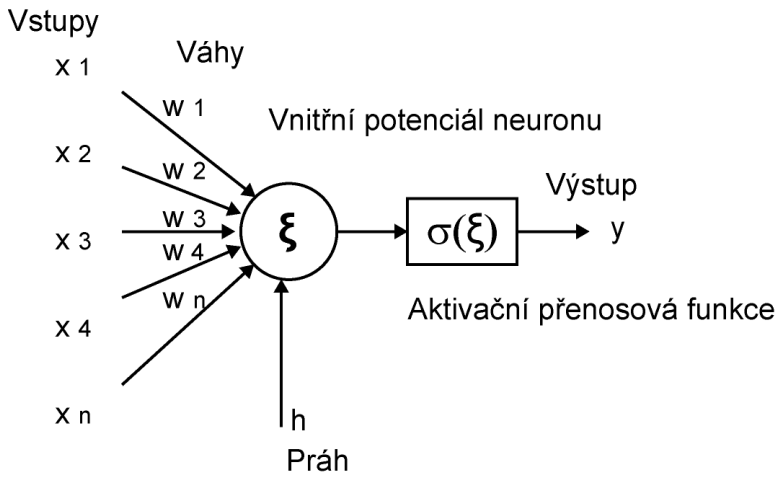
Toto je jedna z definícii NN, akceptovaná NN komunitou. Je zrejmé, že inšpirácia ku vzniku NN prišla z biologických systémov. Hrubo povedané ide o simuláciu mozgu. Na prvý dojem vysoko abstraktná disciplína nachádza množstvo aplikácii v praxi a stáva sa prostriedkom pre riešenie problémov v širokom spektre odborných oblastí. Jednou z veľmi významných vlastností NN je, že svojim spôsobom je tzv. univerzálnym aproximátorom

funkcií. Môže sa nám stať, že máme systém, ktorého popis je mimoriadne náročný alebo je systém natoľko zložitý, že jeho popis je skoro nemožný. Máme však dáta, ktoré do systému vstupujú, a k nim odpovedajúce výstupy. V takejto situácii, môžeme použiť vhodnú NN a pokúsiť sa ju naučiť chovať sa ako sledovaný systém pomocou trénovacích údajov (spomínaných vstupov a výstupov). Toto je veľmi dôležitý moment, ktorý determinuje aj aplikačné uplatnenie NN v praxi.

Vo všeobecnosti môžeme vymenovať nasledovné oblasti využitia NN pre [29] :

- problémy aproximácie funkcií
- klasifikácie do tried, klasifikácia situácií
- riešenie predikčných problémov
- problémy riadenia procesov
- transformácia signálov
- asociačné problémy, simulácia pamäte

Neurón Neurón je základnou stavebnou jednotkou NN. Bol inšpirovaný biologickou nervovou bunkou. Na obrázku 3.1 je načrtnutý matematický model neurónu.



Obr. 3.1: Matematický model neurónu. Prevzaté z [18].

Premenné $x_1, \ldots, x_n$ reprezentujú vstupné hodnoty, ktoré sa šíria do neurónu. Váhy pre vstupy označujeme $w_1, \ldots, w_n$ a sú značené šípkou od vstupu do tela. Premenná h nazývaná prah slúži ako hodnota pre ovplyvnenie aktivačnej funkcie neurónu. Tak ako aj pri biologickej nervovej bunke, neurón obsahuje istý vnútorný potenciál ζ.

$$\zeta = \sum_{i=1}^{n} w_i x_i - h \tag{3.3}$$

Vnútorný potenciál ζ je daný ako suma váh a vstupov, pričom sa dodatočne odráta prah. Na výstupe neurónu z, sa na prahovanie vnútorného potenciálu používa aktivačná funkcia.

$$y = g(\zeta) \tag{3.4}$$

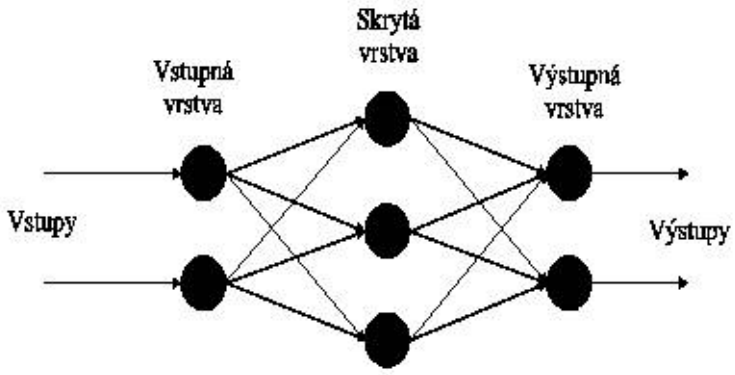
Názov	Funkcia	Obor hodnôt
Logistická sigmoida	$g(x) = \frac{1}{(1+e^{-x})}$	$(0, 1)$
Hyperbolický tangens	$g(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$(1, -1)$
ReLu	$g(x) = max(0, x)$	$< 0, \infty)$
Ostrá nelinearita	$g(x) = \begin{cases} 1 & x > 0,5 \\ 0 & x \leq 0,5 \end{cases}$	$\{0, 1\}$

Tabuľka 3.1: Základné aktivačné funkcie v NN.

Tabuľka 3.1 obsahuje základné aktivačné funkcie používané v NN.

Neurón ako binárny klasifikátor Majme neurónový model, ktorý tvorí jeden neurón. Nech aktivačná funkcia je typu $\sigma(x)$ alebo ostrá nelinearita. Potom takýto model sa nazýva Perceptron [17]. Perceptron dokáže reprezentovať jednoduchý binárny klasifikátor. Cieľom Perceptronu je naučiť váhy W tohoto neurónu tak, aby dokázal rozdeliť tieto dáta do dvoch kategórií.

Nevýhodou perceptronu je, že dokáže rozdeliť iba lineárne separovateľné dáta. V reálnych problémoch nám často ale binárny klasifikátor nestačí. Preto sa v praxi využíva prepojenie viacerých neurónov. Kolekcie neurónov, ktoré sú spojené v acyklickom grafe sa nazývajú neurónové siete. V modeloch neurónových sietí, sú neuróny usporiadavané do vrstiev. V klasických NN je najbežnejšou vrstvou plneprepojená vrstva (angl. *fully-connected*), v ktorej sú neuróny medzi dvoma susednými vrstvami plne párovo prepojené, ale v rámci jednej vrstvy medzi sebou nemajú žiadne prepojenia. Na obrázku 3.2 môžeme vidieť príklad NN. Takúto sieť označujeme ako doprednú, keďže neuróny šíria signál teda vstup jediným smerom na výstup. V skutočnosti sa signál nemusí šíriť iba dopredne, ale môžu existovať v sieti aj spätné prepojenia medzi neurónmi prípadne celými vrstvami. Vtedy hovoríme o rekurentných sieťach [17].



Obr. 3.2: Ukážka neurónovej siete. Prevzaté z [10].

Neurónová sieť oproti Perceptronu dokáže klasifikáciu do viacerých kategórií elegantne vyriešiť zavedením výstupnej vrstvy s neurónmi o počte C, kde C odpovedá počtu tried, do ktorých chceme dáta separovať. Majme neurónovú sieť kde váhy siete označíme ako W. Nech výstup z poslednej skrytej vrstvy označíme ako h. Výstupná vrstva potom robí softmax klasifikáciu, niekedy označovanú aj ako multinomiálnu logistickú regresiu. Pravdepodobnosť, že daný vstup x patrí do kategórie c je daná predpisom:

$$p(y = c|x, W) = \frac{e^{h_{W,c}(x)}}{\sum_{j=1}^{C} e^{h_{W,c}}(x)} \tag{3.5}$$

Softmax klasifikácia alebo softmax vrstva v podstate normalizuje výstupy jednotlivých neurónov tak aby odpovedali pravdepodobnostiam.

Konvolučné neurónové siete Konvolučné neurónové siete sa používajú pre spracovanie dát, ktoré majú topológiu podobnú mriežke. Svoje využitie nachádzajú predovšetkým pri spracovaní obrazu, kedže majú neuróny v troch rozmeroch a to šírka, dĺžka a hĺbka. Na vytvorenie architektúry CNN sa používajú prevažne tri typy vrstiev (angl. *layers*).

- **Convolutional layer** - pozostáva zo súboru filtrov, ktoré su schopné sa učiť. Každý filter je priestorovo malý (pozdĺž výšky a šírky), ale prechádza cez celú hĺbku vstupného objemu (konkrétne u RGB obrazu cez všetky 3 kanály farieb). Pri posúvaní filtra cez šírku a výšku vstupného objemu vytvoríme dvojrozmernú aktivačnú mapu, ktorá definuje hodnotu tohto filtra v každej priestorovej polohe. Sieť teda naučí filtre, aby sa aktivovali, keď vidia konkrétny typ vizuálne prvku ako napríklad hranu alebo škvrnu farby.

  Výstupný objem neurónov danej vrstvy je závislý na vstupe a 3 hyperparameroch: hĺbke (v tomto prípade hĺbka reprezentuje počet filtrov, ktoré chceme použiť, každý sa učí hľadať iné príznaky na vstupe), kroku (určuje, o koľko pixelov sa filter posúva v každom kroku prechodu) a nulovými okrajmi (angl. *zero-padding*). Ak veľkosť vstupného objemu označíme W, veľkosť vnímavého poľa neurónov F, krok S a nulové okraje P, veľkosť výstupného objemu určíme ako:

$$(W - F + 2P)/S + 1 \tag{3.6}$$

- **Pooling layer** - je zvykom vkladať pooling vrstvu medzi jednotlivé konvolučné vrstvy. Úlohou pooling vrstvy je znižovanie priestorovej veľkosti a tým aj počtu parametrov a výpočetnej náročnosti. Taktiež sa používa ako opatrenie proti pretrénovaniu.V prítomnosti sa na redukciu prevažne využíva metóda max poolingu.

- **Fully-connected layer** - vyskytuje sa aj v klasických NN. Vrstva, kde sú neuróny medzi dvoma susednými vrstvami plne párovo prepojené, ale v rámci jednej vrstvy medzi sebou nemajú žiadne prepojenia [15].

- **Dropout layer** - jedná sa vstvu, ktorá sa používa na zníženie rizika pretrénovania. Využíva sa medzi plneprepojenými vrtsvami, ale taktiež aj po pooling vrstve medzi jednotlivými konvolučnými vrstvami.

Najznámejšie architektúry

- **LeNet** - Ide o prvú úspešnú aplikáciu neuronových sietí. Bola vyvinutá v deväťdesiatych rokoch minulého storočia a používaná na rozpoznávanie číslic.
- **AlexNet** - prvá architektúra, ktorá výrazne spopularizovala použitie konvolučných neurónových sietí. Architektúra je podobná LeNet sieti, akurát je väčšia, hlbšia a úspešne používa viacero konvolučných vrstiev za sebou, bez použitia pooling vrstvy.
- **ZFNet** - jedná sa o vylepšenie AlexNetu, predovšetkým rozšírením stredných konvolučných vrstiev a zmenšením rozmerov filtra a veľkosti kroku v prvej vrstve.
- **GoogLeNet** - Hlavný prínos je vo vynájdení počiatočného modulu, ktorý výrazne redukuje počet parametrov v sieti.
- **VGGNet** - ukazuje, že hĺbka siete je kritickým komponentom pre dosiahnutie dobrých výsledkov [15].

3.6 Detekcia objektov

Detekcia je komplexnejší problém ako klasifikácia, ktorá tiež rozpoznáva objekty, ale nehovorí presne o tom, kde je objekt umiestnený v obraze, teda nebude fungovať pre obrázky, ktoré obsahujú viac ako jeden objekt.

Detekcia objektu v obraze pozostáva z rozpoznania objektu a jeho následného nájdenia v obraze. Existujúce metódy riešia problém detekcie jeho preformulovaním na klasifikačný problém, kde sa ako prvý krok natrénuje klasifikátor, ktorý sa v priebehu detekcie spúšťa na viaceré oblasti vstupného obrazu pomocou zvoleného prístupu [26].

3.6.1 Sliding window

Ide o výpočtovo náročnú metódu, ktorá využíva veľký počet okien (rádovo až 10^4), kde by sa principiálne mohol nachádzať objekt, ktorý sa snažíme detekovať. Postupne skenuje vstupný obraz pomocou okna, ktoré sa posúva a mení svoju veľkosť. Nad každým oknom spúšťa klasifikátor, pomocou ktorého sa snažíme detekovať objekt. Príslušné práce venujúce sa tejto problematike zvyšujú výkonnosť pomocou vytvárania sofistikovanejších klasifikátorov. Tento model dosahuje pomerne vysokú presnosť, avšak detekčný proces je zvyčajne príliš pomalý na využitie tohto prístupu pre detekciu objektov v reálnom čase [26].

3.6.2 Region proposals

Namiesto detekovania objektu vo všetkých možných oknách tento prístup sa zameriava na výber kandidátnych okien pomocou *proposal detect* metód. Prvý model, ktorý bol použitý v neurónových sieťach na základe tohto prístupu bol Region-based CNN (R-CNN). Tento model vytvára okolo 2000 potenciálnych ohraničujúcich boxov, ktoré vyberie metódou selektívneho vyhľadávania. Vybrané oblasti zakreslí do obrazu, ktorý následne predá výkonnému klasifikátoru na báze konvolučnej neurónovej siete. Klasifikátor následne vyextrahuje príznaky a ohodnotí jednotlivé oblasti pomocou SVM, upraví ohraničujúce boxy pomocou lineárneho modelu a eliminuje duplicitné detekcie prostredníctvom *non-max suppression*. Dosahuje rýchlosť detekcie na dobre známej PASCAL-VOC približne 40 s/obrázok.

Tento model bol prekonaný vynájdením nového modelu Fast R-CNN. Hlavným účelom tejto siete je zrýchlenie detekcie. Kedže predchádzajúci model vytváral okolo 2000 oblastí, dochádzalo k prekrývaniu oblastí a s tým spojeným opakovaným výpočtom. Tento model to rieši pomocou techniky *Region of Interest Pooling*. Taktiež dochádza k zjednodušeniu modelu. R-CNN obsahuje osobitné moduly pre extrakciu príznakov, klasifikáciu a ohraničenie, Fast R-CNN to zvláda v rámci jednej siete. Fast R-CNN dosahuje rýchlosť 2 s/obrázok.

Najnovším modelom na báze R-CNN je momentálne Faster R-CNN. Model nahrádza selektívne vyhľadávanie. Je tvorený jednou hlbokou neurónovou sieťou. Mapa príznakov, ktorá je tvorená CNN je použitá na predikovanie oblastí. Dosiahlo sa to pridaním fully-connected vrstvy za konvolučnú sieť, čím vzniká Region Proposal Network. Funguje na báze posuvného okna, ktoré pracuje nad mapou príznakov. Faster R-CNN zlepšuje výpočet, prístup k dátam a využitie disku realizované R-CNN. Faster R-CNN dosahuje rýchlosť 140 ms/obrázok [26].

3.6.3 YOLO

You only look once (YOLO), je neurónová sieť zameraná na detekciu objektov v reálnom čase. YOLO predstavuje úplne iný detekčný prístup ako predchádzajúce. Na rozdiel on natrénovania klasifikátora, ktorý je potom v krokoch používaný na detekciu v rôznych častiach obrazu, YOLO vykonáva celý akt v jednom kroku. Taktiež spracuváva všetky objekty zároveň, a tak dosahuje vynikajúcu rýchlosť pri zachovaní relatívne dobrej presnosti.

Vstupný obraz je rozdelený na mriežku o veľkosti SxS. Jeden prvok tejto mriežky sa nazýva bunka. Každá bunka rozdeleného obrazu je zodpovedná za predpovedanie piatich ohraničujúcich boxov. YOLO tiež vygeneruje skóre dôvery, ktoré nám ukáže, aké si je isté, že predpovedané ohraničujúce boxy skutočne obklopujú nejaký objekt. Toto skóre nehovorí nič o tom, aký typ objektu je v boxe, vypovedá len o tom či je daný tvar boxu dobrý. Môže to vyzerať približne ako na obrázku 3.3.

Obr. 3.3: Ukážka tvorby ohraničujúcich boxov. Prevzaté z [9].

Pretože mriežka je o veľkosti SxS máme dokopy S^2 buniek, a každá bunka predpovedá päť ohraničujúcich boxov, skončíme spolu s $5S^2$ ohraničujúcimi boxami. Ukazuje sa, že väč-

šina týchto boxov bude mať veľmi nízke skóre dôvery, takže iba boxy, ktorých konečné skóre je vyššie ako nami stanovený prah (angl. *treshhold*), budú považované za detekované objekty [9]. Pre každý ohraničujúci box bunka taktiež určuje aj triedu objektu.

Dnes existuje viacero variánt YOLO sietí, ako napríklad Yolov2, Yolov3, Tiny Yolo.

3.6.4 SSD

Ide o doprednú NN. Prvé vrstvy modelu sú založené na architektúre VGG16. Za nimi nasleduje podporná sieť, ktorá zaisťuje detekciu. Na predchádzajúcu časť je napojená konvolučnými vrstvami, ktoré postupne znižujú veľkosť. Detekčný model tak môže predikovať nad viacerými vrstvami, čím sa odlišuje od YOLO siete, ktorá pracuje iba s jednou vrstvou príznakov.

3.7 Nástroje

3.7.1 NumPy a SciPy

SciPy je opensource softvér pre matematiku a vedu. Jedným zo základných balíčkov ktoré obsahuje je NumPy. *Numeric Python*, alebo NumPy je balík rozšírení pre jazyk Python. Umožnuje pracovať s viacrozmernými objektmi, ako napríklad multidimenzionálne polia [13].

3.7.2 Scikit-learn

Scikit-learn je voľne dostupná softvérová knižica pre strojové učenie v programovacom jazyku Python. Obsahuje rôzne klasifikačné, regresné a zhlukovacie algoritmy. Spolupracuje s knižnicami NumPy a SciPy. Je písaná primárne v jazyku Python, ale nájdu sa aj algoritmy, ktoré sú z dôvodu dosiahnutia lepšieho výkonu implementované v jazyku CPython [23].

3.7.3 OpenCV

OpenCV je pod BSD licenciou, čiže je zadarmo pre komerčné aj akademické účely. Je to rozhranie pre jazyky ako Java, Python C a C++ s podporou platforiem IOS, Android, Mac a Windows. Bolo navrhnuté pre výpočtovú efektívnosť pri klasifikácii so silným zameraním na aplikácie v reálnom čase využívané v komerčných produktoch [3].

3.7.4 TensorFlow

TensorFlow je open source softvérová knižnica, ktorá sa využíva pre numerické výpočty, ktoré sú realizované pomocou dátových vývojových diagramov, kde hranami sú v grafe reprezentované multidimenzionálne polia (angl. *tensors*) a uzlami rozličné matematické operácie. Flexibilná architektúra umožňuje jednoduché nasadenie a realizovanie výpočtov na rozličných platformách, ako aj využitie viacerých CPU (angl. *central processing unit*) a GPU (angl. *graphics processing unit*). Knižnica bola vytvorená pre podporu strojového učenia a hlbokého učenia. Numerické výpočtové jadro je dostatočne flexibilné a používa sa v mnohých iných vedeckých oblastiach [2].

3.7.5 Theano

Jedná sa o špecializovanú knižnicu v jazyku Python, ktorá umožnuje definovať, optimalizovať a efektívne vyhodnocovať matematické výrazy, zahŕňajúc aj multidimenzionálne polia. Spolupracuje s knižnicou NumPy. Theano poskytuje možnosti výpočtu na CPU, ale aj na GPU (pomocou NVIDIA CUDA) a to bez zmeny zdrojového kódu. V prípade požitia GPU je nutné ešte pridať knižnicu cuDNN (vysoko výkonná knižnica pre strojové učenie, ktorá obsahuje funkcionalitu využívanú hlbokými neurónovými sietami a poskytuje optimalizované verzie funkcií ako napríklad konvolúcia) [31].

3.7.6 Keras

Kvôli vysokej miere všeobecnosti knižníc ako TensorFlow alebo Theano, je tvorba neurónových sietí komplikovaným, zdĺhavým a náročným procesom. Z tohto dôvodu bolo vyvinuté v jazyku Python rozhranie, ktorého hlavnou úlohou je tento proces výrazne zjednodušiť a zrýchliť. Toto rozhranie sa nazýva Keras. Ako svoj *backend* využíva knižnicu TensorFlow, poprípade je možnosť použiť aj Theano alebo CNNK. Síce sa snaží o zjednodušenie procesu implementácie, ale zároveň si zachováva modularitu, vďaka čomu je možné modely neurónových sietí prispôsobovať podľa potreby [5].

3.7.7 Darknet

Jedná sa o opensource rozhranie pre tvorbu neurónových sietí. Je napísané v jazyku C. Medzi hlavné výhody patrí jeho rýchlosť, jednoduché používanie a jednoduchá inštalácia. Taktiež obsahuje podporu pre výpočty realizované nielen na CPU ale aj na GPU [25].

3.7.8 Caffe

Caffe je prostredie pre hĺbkové učenie, ktoré bolo pri vývoji zamerané na rýchlosť a modularitu. Toto prostredie je vyvíjané čisto v C++/CUDA a poskytuje rozhranie pre Python a MATLAB cez príkazový riadkok. Filozofiou tohto prostredia je päť základných princípov: expresívnosť, rýchlosť, modularita, otvorenosť a komunita. Stratégiou pre dosiahnutie obrovskej rýchlosti je redukcia problému na násobenie matíc. Momentálne sa jedná o jedno z najrýchlejších prostredí pre trénovanie neurónových sietí [12].

Kapitola 4

Návrh a Implementácia

V tejto kapitole sa budeme venovať návrhom a implementácii konkrétnych modelov detekčných systémov, ktoré sme sa rozhodli v tejto práci zostrojiť/naimplementovať. Na základe vyššie uvedených teoretických poznatkov sme sa rozhodli zostrojiť tri rozdielne modely – jeden postavený na báze vybranej klasickej metódy a dva modely využívajúce možnosti konvolučných neurónových sietí. Pri jednom z CNN modelov sme si ako prioritné kritérium zvolili presnosť detekcie zbrane v obraze, pri druhom rýchlosť detekcie v reálnom čase. Pri každom modeli uvedieme jeho základnú charakteristiku, datasety, ktoré boli v rámci modelu použité alebo zostrojené a vybrané špecifiká. Popíšeme kreovanie jednotlivých modelov, postupnosť krokov, ktoré tieto modely vykonávajú a jednotlivé architektúry, ktoré vyžívajú.

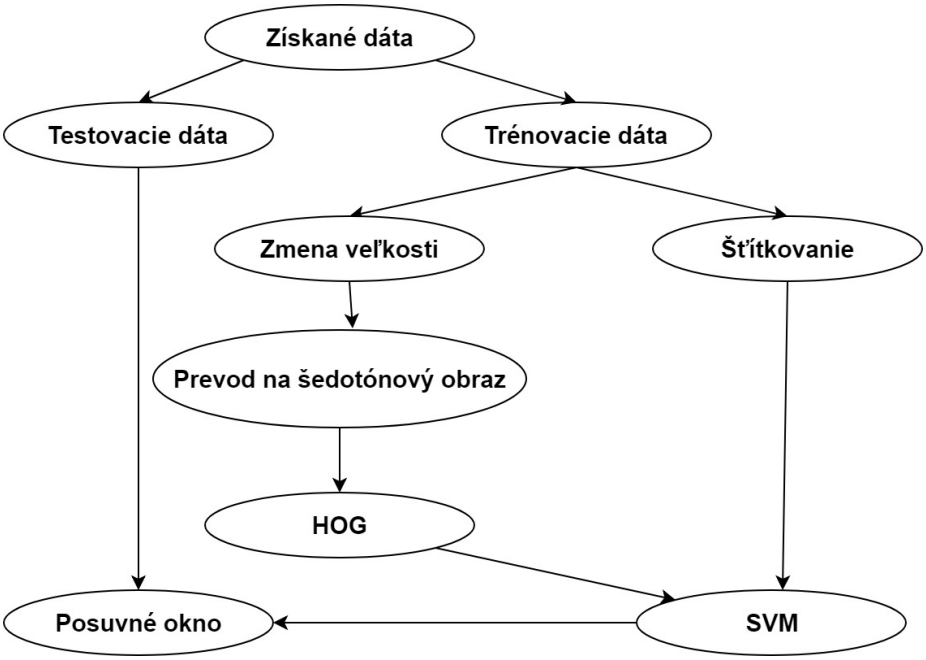
Prvým krokom všetkých modelov je získanie vstupných dát, inak povedané datasetu. Pod pojmom dataset rozumieme sadu obrázkov, ktorá je použitá na trénovanie, validovanie a testovanie klasifikátoru a detektoru. Kedže k našej vedomosti momentálne neexistujú dostupné datasety, ktoré by obsahovali dlhé zbrane, rozhodli sme sa zamerať pri všetkých modeloch na detekciu krátkych zbraní, predovšetkým pištolí. V reálnom živote je to, ako bolo spomenuté už vyššie, štatisticky jasne najčastejšie používaný druh strelných zbraní s potrebou detekcie, čomu zodpovedá množstvo a rozsah už vytvorených a prístupných datasetov.

Kedže hlavnou úlohou našej práce je zostrojiť systém schopný detekovať zbraň bez špecifikácie jazyka a platformy, rozhodli sme sa využiť jazyk Python a vytvoriť systém ktorý nebude závislý na platforme, teda bude použiteľný multiplatformovo.

Trénovanie konvolučných neurónových sietí je všeobecne výpočtovo veľmi náročné. Naše konkrétne trénovanie bolo realizované na grafickej karte NVIDIA GeForce 960M s 4GB pamäti.

4.1 Model na báze HOG a SVM

Ako prvý sme sa rozhodli zostrojiť model za pomoci využitia klasických metód, teda metód, ktoré nevyužívajú konvolučné neurónové siete. Medzi existujúcimi prácami zaoberajúcimi sa touto problematikou sme nenašli žiadnu, ktorej použitá metóda by na získavanie a popis príznakov používala HOG transformácie, a zároveň žiadnu štúdiu, ktorá by nevhodnosť jej aplikovania zdôvodňovala. Rozhodli sme sa preto využiť práve tento deskriptor. Popis jednotlivých krokov vykonávaných v rámci prvého modelu môžeme vidieť na grafe 4.1.



Obr. 4.1: Model č. 1: Sekvencia krokov

Pre tento model sme sa rozhodli využiť dataset dostupný z práce [26], vytvorený pre prístup s využitím posuvného okna. Obsahuje dokopy 9 857 obrázkov, ktoré sú rozdelené do 102 tried. Trieda *AAAPistol* obsahuje 795 obrázkov krátkych zbraní predovšetkým pištolí. Zvyšné triedy obsahujú obrázky iných objektov odlišných od zbraní.

Dáta načítané z datasetu rozdelíme do dvoch tried. Triedu čo zbrane obsahuje a tú, čo nie. Vstupné dáta následne predspracujeme tým, že ich pretransformujeme na rovnakú veľkosť a prevedieme do grayscale reprezentácie.

Nasleduje fáza extrakcie príznakov, na ktorú použijeme už vyššie spomínaný Histogram of Oriented Gradients. Výsledné pole príznakov, ktoré je výstupom HOG-u následne pošleme na vstup SVM klasifikátoru. Naučený klasifikátor následne využijeme pri postupnej detekcii zbrane v obraze posuvným oknom.

4.1.1 Implementácia modelu na báze HOG a SVM

Tento model aplikuje binárnu klasifikáciu a kedže už vyššie spomínaný dataset, ktorý využívame, obsahuje objekty rozdelené do 102 tried, musíme tieto dáta rozdeliť do 2 tried. Preto sme naimplementovali skript *create_one_class.py*, ktorý obrázky objektov, ktoré nie sú zbraň presunie do jedného priečinka *no_weapon*. Výsledný počet následne zredukujeme, aby odpovedal približne objemu triedy zbraň.

Zdrojový text vytvoreného klasifikátoru tohto modelu sa nachádza v *model1_train.py*. Pri implementácii využívame framework-y keras, skimage a sklearn. Pre načítanie vstupných dát a úpravu ich veľkosti využívame metódu *load_img* z knižnice *preprocessing*, ktorú poskytuje Keras. Veľkosť vstupných dát sme transformovali na veľkosť 128x128. Následne pomocou metódy *rgb2gray* knižnice *color* (skimage) prevedieme obrázky na šedotónové. V ďalšom kroku vytvoríme pole label-ov pre vstupné dáta a pole deskriptorov pre všetky vstupné obrázky. K tomu využijeme *hog* metódy (sklearn). Pre klasifikáciu využívame triedu *SVC* knižnice *svm* (sklearn), čím vznikne objekt klasifikátoru, ktorý natrénujeme pomocou

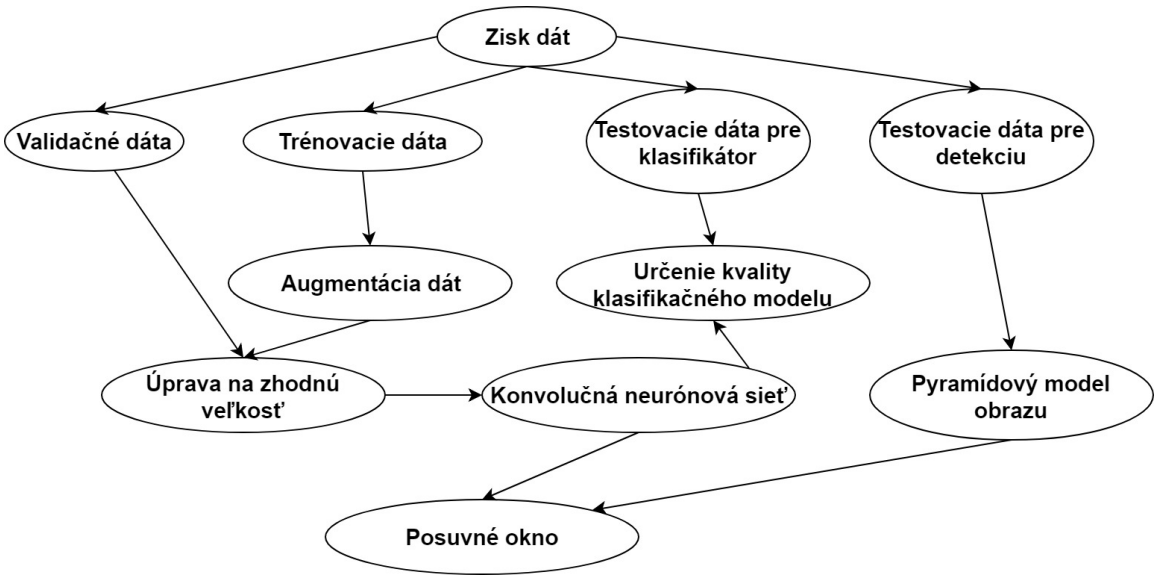
metódy *fit*, (vyžaduje pole príznakov a labelov-ov). Výsledný klasifikátor ukladáme pomocou triedy *joblib* (sklearn).

V *model1_predict_test.ty* testujeme pomocou metódy *predict* (sklearn) úspešnosť klasifikátoru, kde k vyhodnoteniu používame *average precision*, ktorej sa podrobnejšie venujeme v kapitole 5. Tento výpočet implementujeme pomocou volania metódy *average_precision-_score* (sklearn).

Výsledná detekcia je implementovaná v *model1_detect.py*.Pri implementácii prístupu posuvného okna sme sa rozhodli, že namiesto zmien veľkosti okna, ktorá je typická pre tento prístup budeme meniť veľkosť prehľadávaného obrazu a veľkosť okna tak ostane fixná. Pre lepšiu ilustráciu spätne dopočítavame zmenu veľkosti okna a celý priebeh detekcie je tak zobrazený v obrázku pôvodnej veľkosti kde sa mení práve veľkosť tohto okna. Pre uloženie a prácu s rôznymi veľkosťami obrázka využívame tzv. model obrazových pyramíd. Je to vlastne viacúrovňová reprezentácia obrazu. V spodnej vrstve sa nachádza originálna veľkosť obrazu a v každej ďalšej zmenšenina toho pôvodného, až kým nedosiahneme určenú minimálnu veľkosť. Tento pyramídový model je zostrojený volaním *pyramid_gaussian* rozhrania skimage.

4.2 Model na báze CNN a posuvného okna

Toto riešenie problematiky sa zameriava na využitie konvolučných neurónových sietí pri klasifikácii objektov. V tomto konkrétnom modeli sme si ako prioritný cieľ určili úspešnosť detekcie zbrane zo štatistického hľadiska. Zároveň sme pre lepšie dosiahnutie zvoleného cieľa, nadobudnutie praktických skúseností a možnosť porovnania výsledkov zvolili ako klasifikátor použitie dvoch rôznych architektúr – jednej, ktorej architektúru sme sami vhodne navrhli a následne ju natrénovali a druhej na báze architektúry VGG16 vynájdenej v roku 2014, kde sme kvôli robustnosti siete pri trénovaní použili váhy predtrénovaného modelu, ktoré sme doladili dotrénovaním tak, aby sieť vyhovovala nášmu zadaniu (klasifikovala zbrane). Celú postupnosť krokov, vykonávaných pri zostrojovaní modelu môžeme názorne vidieť na grafe 4.2.



Obr. 4.2: Model č. 2: Sekvencia krokov

Poradie	Typ vrstvy	Filtre	Veľkosť	Krok	Vstup	Výstup
1	konvolučná	32	3x3	1	64x64x3	64x64x32
2	maxpool		2x2	2	64x64x32	32x32x32
3	konvolučná	64	3x3	1	32x32x32	32x32x64
4	maxpool		2x2	2	32x32x64	16x16x64
5	konvolučná	128	3x3	1	16x16x64	16x16x128
6	maxpool		2x2	2	16x16x128	8x8x128

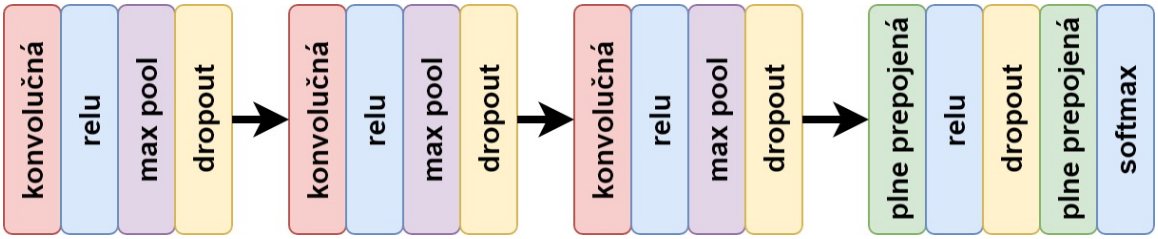
Tabuľka 4.1: Parametre vrstiev

Ako vstupné dáta sme použili rovnaký dataset, ako bol použitý v návrhu prvého modelu. Kedže tento dataset obsahuje veľké množstvo obrázkov s bielym pozadím, a my tentokrát nebudeme obrázky prevádzať na šedotónové, rozhodli sme sa dataset rozšíriť z dôvodu vyššej generalizácie modelu o obrázky z iným pozadím ako biele. K tomu sme vyžili dáta dostupné na stránke [1]. Vhodné dáta boli manuálne vyseparované v primeranom rozsahu a následne sme nimi rozšírili pôvodný dataset do požadovanej štruktúry obsahu a počtu.

Dáta boli následne rozdelené na trénovacie, validačné a testovacie a boli prevedené na zhodnú veľkosť očakávanú na vstupe CNN. Trénovacie a validačné dáta využijeme pri trénovaní konvolučnej neurónovej siete a testovacie pre následné určenie kvality klasifikačného modelu. Pre lepšie výsledky výsledného klasifikátora sme použili augmentáciu dát na trénovacie dáta.

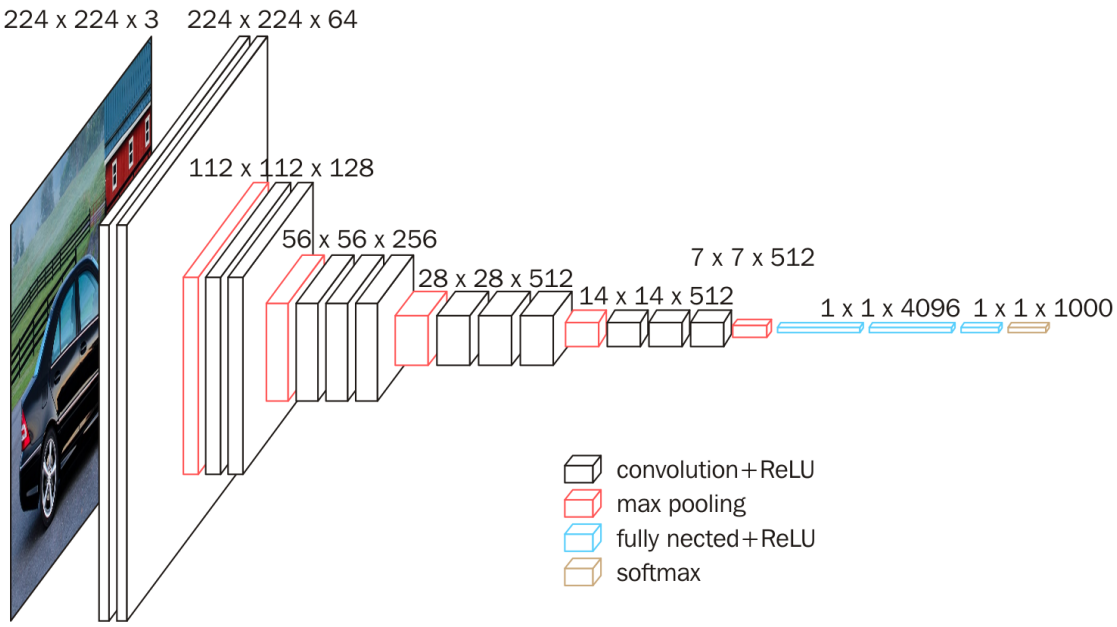
Ako klasifikátor sme vyskúšali z dôvodov uvedených v úvode odstavca použitie dvoch rozdielnych architektúr. Prvú sme sa rozhodli navrhnúť a natrénovať sami. Môžeme ju vidieť na 4.3. Pozostáva z troch konvolučných a troch max pool vrstiev. Ako aktivačná funkcia je použitá ReLu. Pre zvýšenie generalizácie modelu využívame aj dropout vrstvy. Kedže dropout vrstvy sa obvykle vkladajú po max pool vrstvách, ktoré sa nachádzajú medzi konvolučnými vrstvami, vložili sme ich tam aj my. Hodnotu týchto dropout vrstiev sme nastavili na 0,2. Taktiež sme vložili jednu dropout vrstvu medzi plne prepojené vrstvy, tentokrát s hodnotou 0,5. Hodnoty, ako veľkosť kroku, počet filtrov a veľkosť rámcov, sú pre použité max pool konvolučné vrstvy zaznamenané v 4.1. Pri tvorbe tohto návrhu sme postupovali podľa odporúčaní pre tvorbu CNN v [15].

Ako druhú architektúru sme sa rozhodli použiť VGG16. Je zachytená na obrázku 4.4.



Obr. 4.3: architektúra CNN

Posledným krokom tejto metódy je využitie natrénovaného klasifikátora a prístupu posuvného okna pre detekciu zbrane v obrázkoch. Kedže klasifikátor na báze CNN vyžaduje fixnú veľkosť vstupného obrazu a nie je známa veľkosť zbrane v pomere k veľkosti obrázku, bolo potrebné tento aspekt ošetriť. Rozhodli sme použiť obdobný prístup ako v modele na báze HOG a SVM.



Obr. 4.4: architektúra VGG16 [24]

4.2.1 Implementácia modelu na báze CNN a posuvného okna

Pri implementácii neurónových sietí využívame rozhranie Keras. Pri oboch klasifikátoroch využívame sekvenčný model Kerasu, ktorý umožňuje tvorbu siete ukladaním jednotlivých vrstiev za sebou. Pre načítanie a augmentáciu vstupných dát využívame triedu *ImageDataGenerator* knižnice Preprocessing. Dáta sú načítavané metódou tejto triedy *flow_from_directory*. Pre rozdelenie dát na trénovacie a validačné (poprípade testovacie) sme naimplementovali skript *split_training_data.py*. Pre nakonfigurovanie trénovania sme použili metódu *compile* triedy *model*. Výsledné trénovanie, uloženie a následné načítanie klasifikátoru prebieha pomocou metód *fit, load* a *save* tej istej triedy. Pri vytváraní modelu sme sa taktiež zamerali na vplyv počtu tried, ktoré klasifikátor rozoznáva pri výslednej detekcii. Rozhodli sme sa preto vytvoriť pre obe architektúry siete, ktoré klasifikujú do 2 tried, a tú, ktorá vykazovala lepšie výsledky následne preimplementovať na použitie pri klasifikácii do 102 tried.

Pri implementácii už vyššie spomínanej VGG16 konvolučnej neurónovej siete sme použili triedu *VGG16* knižnice *Applications*, ktorá načíta architektúru celej siete a váhy predtrénované na datasete Imagenet. Nastavením atribútu *trainable* pre jednotlivé vrstvy pozastavíme trénovanie všetkých vrstiev, ktoré slúžia na extrakciu nízkoúrovňových príznakov (ako farba alebo tvar), teda všetkých okrem posledných troch konvolučných vrstiev. Následne pridáme na koniec modelu dve plne prepojené vrstvy. Prvú s počtom filtrov 1024 a druhú s počtom filtrov, ktorý sa rovná počtu tried, medzi ktorými chceme vedieť klasifikovať (teda 2 alebo 102).

Trénovanie modelov na klasifikáciu do dvoch tried je naimplementované v súboroch *model2_2_classes_3_conv_classifier.py* a *model2_2_classes_VGG16_classifier.py*. Ich následné testovanie prebieha pomocou zdrojového súboru *model2_2_classes_prediction.py*. Average precision je počítaná obdobne ako v modeli na báze HOG a SVM. Model, ktorý implementuje trénovanie klasifikátora klasifikujúceho do 102 tried sa nachádza v *model2_102-*

__classes__classifier.py. Pre testovanie klasifikátora bol zostrojený *model2__102__classes__prediction.py* a výsledná detekcia je naimplementovaná v *model2__102__classes__detection.py*. Predikcia triedy prebieha pomocou metódy predict (Keras), ktorá však narozdiel od metódy rovnakého mena rozhrania sklearn vracia pole pravdepodobností pre jednotlivé triedy. Preto je možné pri detekcii nastavovať prah.

4.3 Model na báze Tiny Yolov3

Prioritným cieľom tohto modelu nebola na rozdiel od predchádzajúceho čo najvyššia úspešnosť detekcie. Bol zostrojený zo zámerom dosiahnuť detekciu zbrane v reálnom čase. Nezanedbateľným parametrom pri výbere vhodného nástroja bolo aj časové hľadisko a limity dostupného hardvéru. Po zvážení možností sme sa rozhodli použiť neurónovú sieť Yolov3, konkrétne jej Tiny verziu, ktorá nedosahuje až takú presnosť, ale je pomerne malá a vzhľadom k našim výpočtovým možnostiam relatívne natrénovateľná. Architektúra tejto siete je zobrazená v tabuľke 4.2. Ako vstupné dáta sme v tomto prípade použili dataset z práce [26] určený pre ich implementáciu R-CNN. Obsahuje 3000 obrázkov zbraní v rôznych scénach.

Poradie	**Typ vrstvy**	**Filtre**	**Veľkosť**	**Krok**	**Vstup**	**Výstup**
1	konvolučná	16	3x3	1	416x416x3	416x416x16
2	maxpool		2x2	2	416x416x16	208x208x16
3	konvolučná	32	3x3	1	208x208x16	208x208x32
4	maxpool		2x2	2	208x208x32	104x104x32
5	konvolučná	64	3x3	1	104x104x32	104x104x64
6	maxpool		2x2	2	104x104x64	52x52x64
7	konvolučná	128	3x3	1	52x52x64	52x52x128
8	maxpool		2x2	2	52x52x128	26x26x128
9	konvolučná	256	3x3	1	26x26x128	26x26x256
10	maxpool		2x2	2	26x26x256	13x13x256
11	konvolučná	512	3x3	1	13x13x256	13x13x512
12	maxpool		2x2	2	13x13x512	13x13x512
13	konvolučná	1024	3x3	1	13x13x512	13x13x1024
14	konvolučná	256	1x1	1	13x13x1024	13x13x256
15	konvolučná	512	3x3	1	13x13x256	13x13x512
16	konvolučná	18	1x1	1	13x13x512	13x13x18
17	yolo					
18	route					
19	konvolučná	128	1x1	1	13x13x256	13x13x128
20	upsample			2x	13x13x128	26x26x128
21	route					
22	konvolučná	256	3x3	1	26x26x384	26x26x256
23	konvolučná	18	1x1	1	26x26x256	26x26x18
24	yolo					

Tabuľka 4.2: Architektuúra Tiny Yolov3 modelu

4.3.1 Implementácia modelu na báze Tiny Yolov3

Pri tomto modele sme použili rozhranie Darknet. Zo vstupného datasetu sme vybrali 1000 obrázkov, ku ktorým sme vytvorili anotáciu pomocou *BBox Label Tool*-u. Forma zápisu tohto programu je zobrazená na 4.5. Prvý údaj zodpovedá triede objektu, ktorý daný box ohraničuje. Nasledujú dvojice celých čísel, ktoré udávajú pozíciu pravého horného a ľavého dolného rohu.

```
class_num x_min y_min x_max y_max
1         355   217   714   471
```

Obr. 4.5: BBox Label Tool anotácia

Darknet však pristupuje odlišne. Namiesto pevných súradníc používa pomery viz. 4.6. Význam prvého údaju ostáva nezmenený. Nasledujúca dvojica udáva ľavý horný roh obrázku. Hodnota odpovedá pomeru x súradnice k šírke obrázku a y súradnice k výške obrázku. Posledná dvojica udáva veľkosť rámčeka a počíta sa ako pomer šírky rámčeka k šírke obrázka a výšky rámčeka k výške obrázka. Výhoda tejto anotácie je, že v prípade zmeny veľkosti obrázka ostáva stále platná. Na prevod sme použili skript dostupný na [21], ktorý napísal Guanghan Ning.

```
class_num box_x_ratio box_y_ratio box_width_ratio box_height_ratio
1         0.65686     0.235678    0.984321        0.2356787
```

Obr. 4.6: Darknet notacia

Po anotácii dát sme použili skript dostupný na [32] ktorý vytvorí dva textové súbory, *train.txt* a *test.txt*, ktoré obsahujú názvy trénovacích a testovacích dát. Pomer rozdelenia dát sme nechali v pôvodnom nastavení teda 10% pre testovanie.

Následne boli vytvorené súbory *obj.data*, *obj.names* a *yolo-obj*. Na 4.7 môžme vidieť obsah súboru *obj.data*. Určuje koľko tried sa má sieť naučiť rozpoznávať, cestu k súborom

```
classes= 1

train  = train.txt

valid  = test.txt

names = obj.names

backup = backup/
```

Obr. 4.7: Obsah súboru obj.data

train.txt, *test.txt* a *obj.names*, a miesto kde sa majú ukladať výsledky trénovania. Súbor *obj.names* obsahuje názvy tried, v našom prípade iba triedu *weapon*. Súbor *yolo-obj* obsahuje architektúru siete ktorú budeme trénovať (Tiny Yolov3), a jej konfiguráciu. Bližšia špecifikácia v 5.

4.4 Súhrn

V tejto kapitole sme sa venovali konkrétnym modelom detekčných systémov, ktoré sme zostrojili/naimplementovali a ich návrhom a implementácii. Vytvorili sme tri rozdielne modely, pri ktorých sme popísali ich funkcionalitu a ako bola táto funkcionalita implementovaná. Ďalej sme uviedli datasety, ktoré boli v rámci modelu použité alebo zostrojené. V ďalšej kapitole sa budeme venovať výsledkom, ktoré tieto detekčné systémy dosiahli.

Kapitola 5

Experimenty a výsledky

5.1 Model na báze HOG a SVM

5.1.1 Metodika vyhodnocovania úspešnosti klasifikácie a detekcie

Pre vyhodnocovanie úspešnosti klasifikácie tohto modelu použijeme, tak ako aj pri ostatných modeloch, average precision. Túto hodnotu určíme spracovaním dát získaných použitím klasifikátora na testovací dataset. Pre vyhodnocovanie detekcie budeme využívať dve metriky, *precision* a *recall*. Vypočítame ich ako:

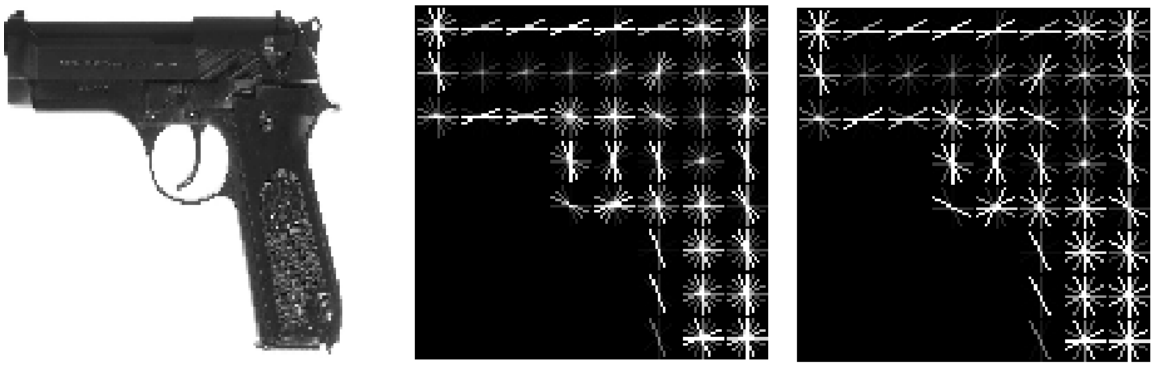
$$precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{5.1}$$

$$recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{5.2}$$

Zbraň budeme považovať za detekovanú v prípade, že jej obsah bude tvoriť viac ako 50% okna a okno bude obsahovať aspoň 80% obsahu zbrane.

5.1.2 Trénovanie

Potrebovali sme vybrať vhodné nastavenie parametrov HOG deskriptoru, a tiež najvyhovujúcejšiu jadrovú funkciu SVM klasifikátoru. Na obrázku 5.1 môžeme vidieť rozdiel pri rozdelení uhlu 360°, na 6 (vpravo) kanálov a 8 kanálov (v strede).



Obr. 5.1: HOG deskriptor zbrane

Po zohľadnení získaných dát sme zvolili 6 kánalov a bloky o veľkosti 16x16 pixelov, kde každý blok obsahuje práve 1 bunku. Jadrovú funkciu sme vyberali z týchto funkcií: *linear*, *poly*, *rbf*, *sigmoid* a *precomputed*. Najlepšie výsledky klasifikátoru sme dosiahli použitím jadrovej funkcie *rbf*.

5.1.3 Vyhodnotenie

Výsledná average precision na testovacích dátach klasifikátoru dosahovala 98,97%. Zdrojové obrázky datasetu použitého pri testovaní zobrazovali zbrane ako selektívne objekty, teda s uniformným pozadím a bez prítomnosti čohokoľvek iného v obraze. Pre detekciu sme ako vstupné dáta použili reálne (filmové) scény, kde sa nachádzali osoby držiace zbraň. Aj keď model bol schopný vo fáze testovania pozitívne klasifikovať s vysokou mierou pravdepodobnosti, nedosahoval žiadané výsledky pri detekcii. Vykazoval totiž vysokú mieru false negatívov, hlavne v pozadí scén, ako je napríklad obloha. Zbraň bol schopný detekovať najmä podľa oblasti spúšte, ktorá sa však pri zbrani, ktorú drží ruka nenachádza v takom tvare, aký detektor hľadal. Toto zistenie považujeme za kľúčový dôvod, prečo tento model nedetekoval zbraň držanú rukou.V použitom datasete s obrazmi z filmov bol podiel tých s pozadím a tých s so zbraňou v ruku zákonite dominantný, čo s ohľadom na vyššie uvedené viedlo k výsledku, že počet správnych detekcií bol veľmi nízky a hodnoty recall a precision boli hodnoty blízke nule. Ak sme ako vstupné dáta pre detekciu použili obrázky obsahujúce zbraň, ktorá nebola držaná, a bol vidieť celý jej tvar, počet true pozitívov sa zvýšil, počet false negatívov ale ostal stále vysoký. Presicion sa tak stále pohybovala na veľmi nízkych hodnotách. Recall hodnota sa zlepšila a dosahovala úroveň 30%, čo však stále podľa nášho názoru nebol uspokojivý výsledok.

Nepodarilo sa nám nájsť žiadne nastavenie parametrov HOG deskriptora, ktoré by vo finálnej fáze detekcie reálnych obrazov zvoleného datasetu viedlo, alebo sa aspoň dostatočne približovalo k výsledkom, ktoré by sme mohli považovať za v praxi použiteľné. Tento model tak nepovažujeme ako úspešný ani z hľadiska detekčných výsledkov, ani z hľadiska rýchlosti.

5.2 Model na báze Tiny Yolov3

5.2.1 Metodika vyhodnocovania získaných dát

Pre vyhodnotenie modelu používame average precision, ktorá sa používa pri VOC súťaži. Je vypočítavaná z *precision/recall* krivky. Hodnoty precision a recall sa určia obdobne ako pri metóde na báze HOG a SVM. Na určenie pravdivo pozitívneho výsledku používame IOU (angl. *Intersection over Union*) metriku. Ako pravdivo pozitívne berieme tie detekcie, kde IOU dosahuje aspoň hodnotu 0,5. Výpočet IOU môžme vidieť znázornený na obrázku 5.2.

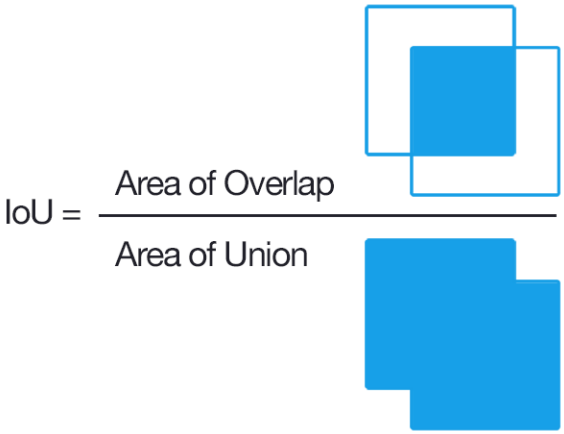
Zo zmienenej krivky je vybraných 11 rovnako vzdialených recall hodnôt $[0; 0,1; \ldots; 1]$:

$$AP = \frac{1}{11} \sum_{r \in 0;0,1;\ldots;1} P_{interp}(r) \tag{5.3}$$

kde je precision odpovedajúca dané recall hodnote interpolovaná tak, že ze sa vezme najväčšia nameraná hodnota precision, ktorá prekročí r:

$$P_{interp}(r) = \max_{\tilde{r}:\tilde{r}>\tilde{r}} p(\tilde{r}) \tag{5.4}$$

kde $p(\tilde{r})$ je nameraná *precision* odpovedajúca *recall* hodnote $\tilde{r}$.



Obr. 5.2: IOU výpočet [32]

5.2.2 Trénovanie Tiny Yolov3

Pri trénovaní tohto modelu sme zvolili učiaci krok 0,001 z dôvodu správneho nastavenia váh na začiatku. Vyhli sme sa tak prípadnej divergencii modelu. Následne sme po 100 iteráciách zmenili tento krok na 0,01. Po 2000 iteráciách sme opäť zmenili učiaci krok, tentokrát na 0,001, aby sa model neuchyľoval k lokálnym minimám namiesto požadovaných globálnych. Batch size bola nastavená na hodnotu 64. Celkovo sme vykonali 20000 iterácií. Zároveň sa pri trénovaní použila kombinovaná chybová funkcia, ktorá sa skladá z chyby podmienenej pravdepodobnosti:

$$Loss_p = \sum_{i}^{S^2} \sum_{c \in classes} B1_i^{resposible\_obj} \{p_i^{pred}(c) - p_i^{truth}(c)\}^2 \tag{5.5}$$

kde $1_i^{resposible\_obj}$ označuje, že sa objekt vyskytuje v bunke i. S udáva rozmer mriežky a B predikovaných ohraničení pre jednotlivé bunky. Ďalej sa skladá z chyby pre *anchor box*:

$$\begin{aligned} Loss_{box} = \lambda_{obj}^{cord} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{resposible\_obj} \big[(x_{ij}^{pred} - x_{ij}^{obj})^2 + (y_{ij}^{pred} - y_{ij}^{obj})^2 \\ + (w_{ij}^{pred} - w_{ij}^{obj})^2 + (h_{ij}^{pred} - h_{ij}^{obj})^2 \big] \\ + \lambda_{noobj}^{cord} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{no\_resposible\_obj} \big[(x_{ij}^{pred} - x_{ij}^{anchor\_center})^2 + (y_{ij}^{pred} - y_{ij}^{anchor\_center})^2 \\ + (w_{ij}^{pred} - w_{ij}^{anchor\_default})^2 + (h_{ij}^{pred} - h_{ij}^{anchor\_default})^2 \big] \end{aligned} \tag{5.6}$$

kde $1_{ij}^{resposible\_obj}$ značí j-tý prediktor ohraničenia pre konkrétnu bunku i. $x_{ij}^{anchor\_center}$ a $y_{ij}^{anchor\_center}$ označujú pôvodnú pozíciu stredu bunky a majú hodnotu 0,5. Premenné $w_{ij}^{anchor\_default}$ a $h_{ij}^{anchor\_default}$ nadobúdajú hodnotu 1. λ_{obj}^{cord} a λ_{obj}^{cord} prisudzujú väčší význam bunkám, ktoré obsahujú požadovaný objekt. Pre bunky ktoré ho obsahujú je hodnota

1, pre ostatné 0,1. Poslednou súčasťou výpočtu chyby je *confidence score*[22]:

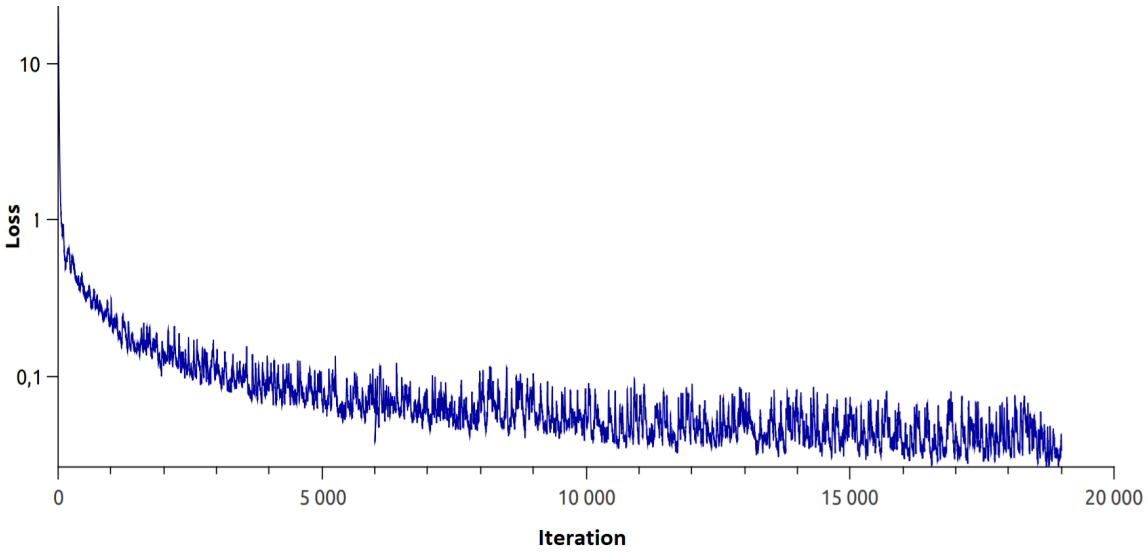
$$Loss_{conf} = \lambda_{obj}^{conf} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{resposible\_obj} \{conf_{ij}^{pred} - iou(box_{ij}^{pred}, box_{ij}^{truth})^2 + \lambda_{noobj}^{conf} \sum_{i}^{S^2} \sum_{j}^{B} 1_{ij}^{no\_resposible\_obj} (conf_{ij}^{pred})\}^2 \quad (5.7)$$

takže výsledná chybová funkcia sa počíta ako:

$$Loss = Loss_{conf} + Loss_{box} + Loss_{p} \quad (5.8)$$

5.2.3 Analýza získaných výsledkov

Priebeh chybovej funkcie počas trénovania je možné vidieť na grafe 5.3.



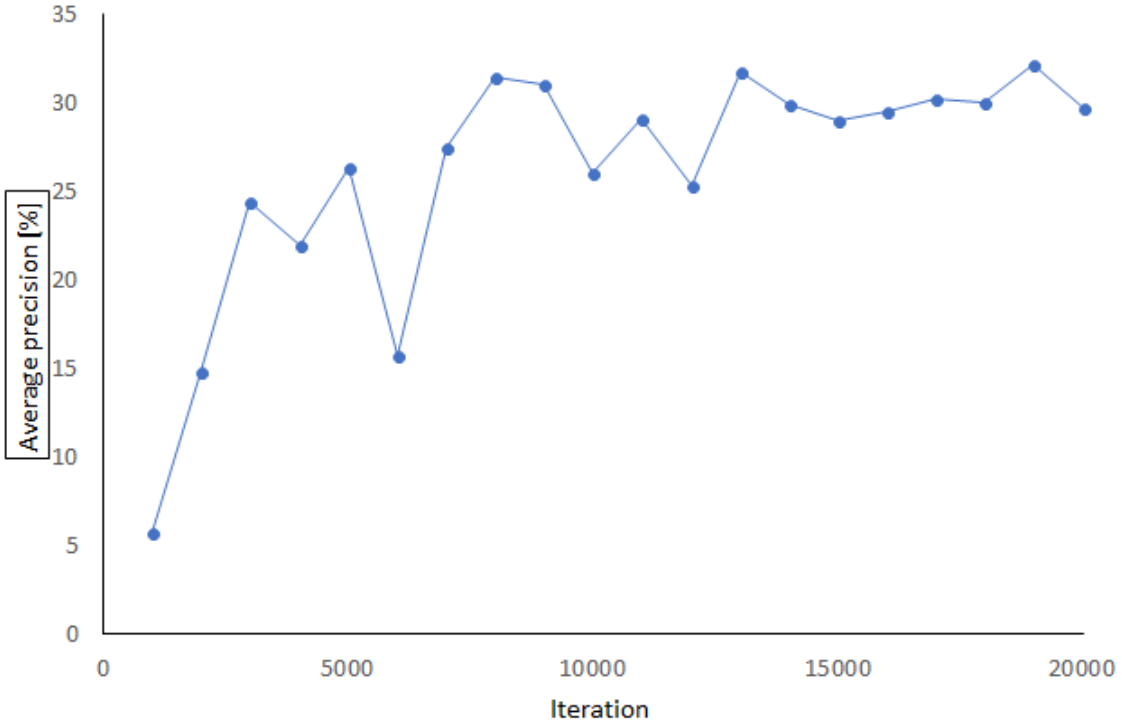
Obr. 5.3: Priebeh loss funkcie pri trénovaní Tiny Yolov3

Počas trénovania sme ukladali váhy každých 1000 iterácií. Pre každú váhu sme následne vyhodnotili average precision na testovacích dátach. Jednoltivé hodnoty je možno vidieť v grafe 5.4. Najvyššiu hodnotou sme dostali pri 19000 iteráciách. Hodnota average precision bola 32,16%, čo je veľmi blízko referenčnej hodnote, ktorú Tiny Yolov3 dosahuje a to 33,1%. Taktiež je tu predpoklad, že by sa nami dosiahnutá hodnota pri vyššom počte iterácií mohla ešte zvýšiť kedže maximálna hodnota bola nameraná pri 20000.

5.3 Model na báze CNN a posuvného okna

5.3.1 Výber metódy realizácie experimentu

Pri tomto modeli sme viacero etáp experimentálnej časti realizovali paralelne pre obe verzie zvolených architektúr neurónových sietí. V obidvoch verziách sme porovnateľne menili vstupné parametre v tréningovom procese oboch klasifikátorov a vyhodnocovali ich vplyv



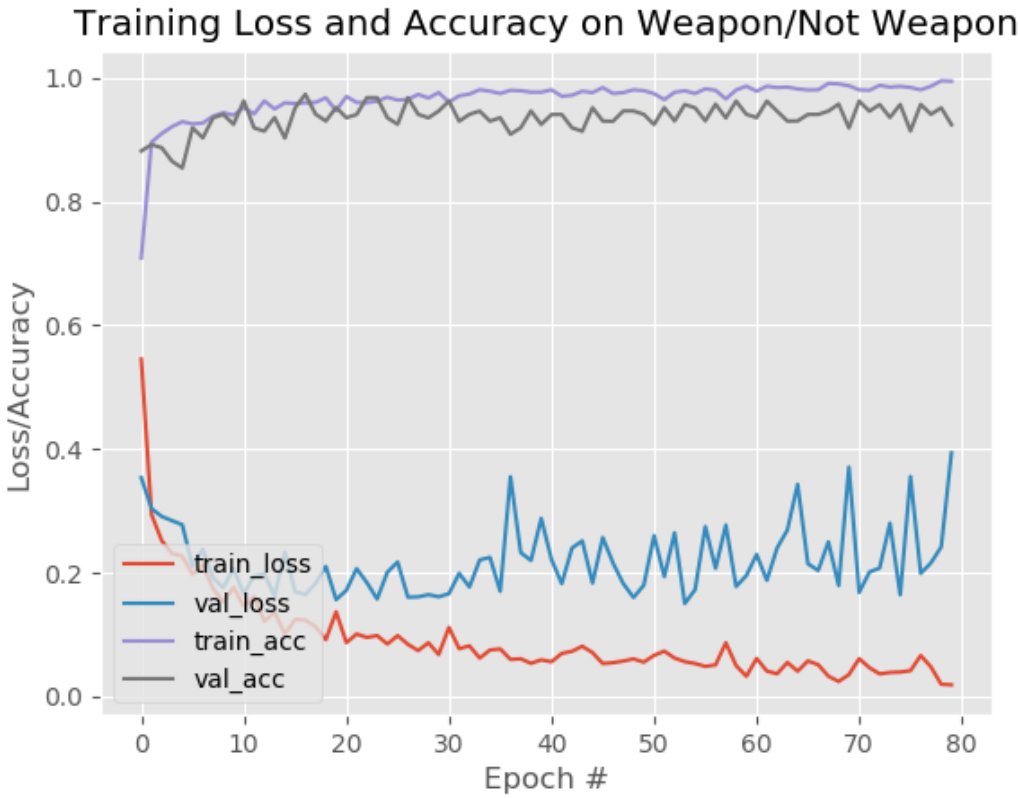
Obr. 5.4: Average precision pre jednotlivé váhy Tiny Yolov3

na výsledné hodnoty úspešnosti klasifikácie a v ďalšej fáze detekcie. Od určitého bodu, v ktorom sa úspešnosť jedného klasifikátora prejavila ako výrazne vyššia , sme sa aj s ohľadom na vyššie uvádzané výpočtové a časové limity rozhodli finálnu optimalizáciu parametrov urobiť pre úspešnejšiu verziu. Pre porovnanie výsledkov sme používali hodnoty počtu true pozitívov, true negatívov a false pozitívov. Podrobnejšie je celý proces uvedený v 5.3.2.

5.3.2 Trénovací a testovací proces

Po zostrojení oboch verzií zvoleného modelu, teda klasifikátora na báze troch konvolučných vrstiev a klasifikátora na báze VGG16 sme prikročili k trénovaniu oboch klasifikátorov. V súlade s teoretickou prípravou sme testovali vplyv počtu epoch, veľkosti dát použitých na vstupe do siete, optimalizačných funkcií a chybových funkcií. Pri binárnej klasifikácii siete na báze 3 konvolučných vrstiev sa nám najviac osvedčil počet epoch 80, kde pri VGG16 architektúre ich stačilo 40. Pri výslednom 102 triednom klasifikátore sme najlepšieho výsledku dosiahli pri znížení learning rate-u na 1/10 hodnoty používanej pri sieti na báze 3 konvolučných vrstiev. Keďže rozmer dát na vstupe do siete by malo byť číslo deliteľné dvojkou, ideálne jeho násobok, a kedže tento rozmer vzhľadom na našu implementáciu určuje minimálnu veľkosť posuvného okna, rozhodli sme sa použiť vstupné rozmery 64x64 a 128x128. Trošku lepšie klasifikačné výsledky dosahoval vstup o veľkosti 128x128, avšak v niektorých prípadoch to bola príliš veľká počiatočná veľkosť posuvného okna, a nedošlo tak k detekcii malých zbraní (malé obsahovo v pomere voči obsahu celého obrázku). Vo všetkých prípadoch sme ako výstupnú funkciu použili softmax. V prípade optimalizátorov sa nám pri architektúre troch konvolučných vrstiev najviac osvedčil *adam*. Vo VGG16 architektúre to bol zas *rmsprop*. Ako loss funkciu sme vždy využívali *binary_crossentropy*. Na

nasledujúcich grafoch môžme vidieť hodnoty presností a chyby na trénovacích a validačných dátach počas procesu trénovania. Na grafe 5.5 môžeme vidieť priebeh trénovania binárneho klasifikátora na báze troch konvolučných vrstiev. Pri testovaní tento klasifikátor dosiahol výslednú average precision 85,71%.



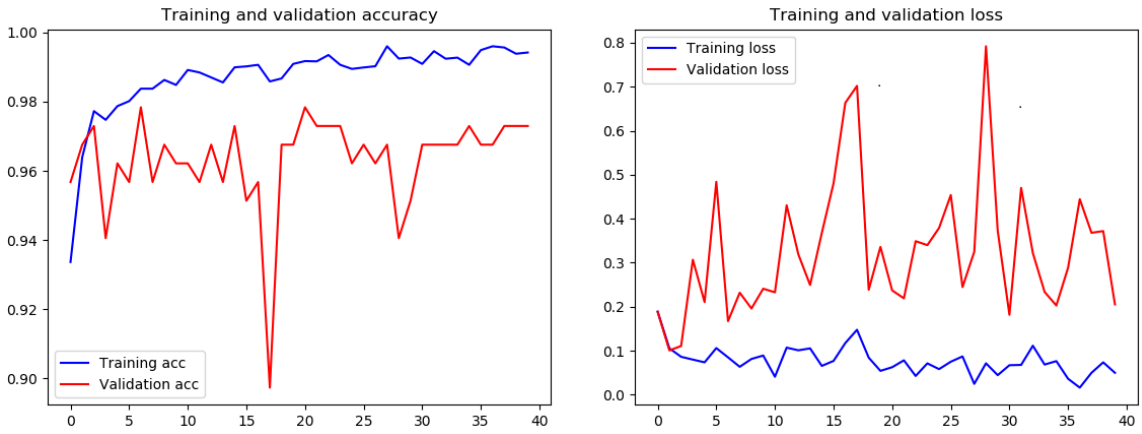
Obr. 5.5: Priebeh trénovania binárneho klasifikátora na báze troch konvolučných vrstiev

Na grafe 5.6 je zobrazený priebeh trénovania binárneho klasifikátoru na báze VGG16. Skoky pri validácii sú v tomto prípade zapríčinené malou dávkou validačných dát na konci epochy, čo viedlo k vzniku štatistickej odchýlky. Táto verzia modelu dosiahla výslednú average precision pri klasifikácii na testovacích dátach 95,89%.

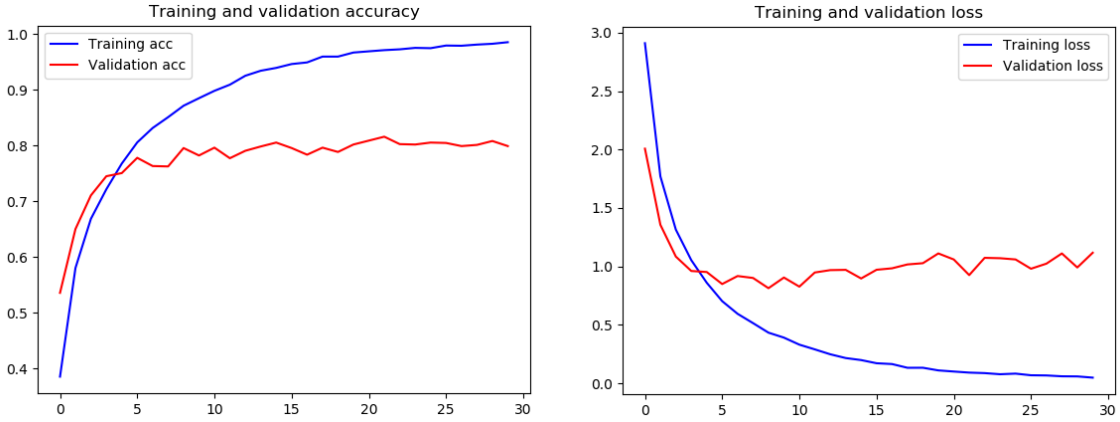
5.3.3 Finálna etapa a zhodnotenie výsledkov

Obe testované verzie dosiahli výslednú average precision pri klasifikácii na testovacom datasete v prijateľnom intervale, avšak pri výslednej detekcii nepreukazovali vhodné výsledky. Množstvo vykazovaných false pozitívov bolo neprijateľne vysoké. Preto sme do modelu implementovali 102 triednu klasifikáciu, ktorá výslednú pravdepodobnosť rozdeľuje do viacerých tried. Dôsledkom je, že pri rovnakej hodnote prahu sa výrazne znižuje množstvo false pozitívov. Kedže binárny klasifikačný model na báze VGG16 dosahoval vyššiu average precision, rozhodli sme sa využiť práve tento model pri tvorbe 102 triedneho klasifikátoru. Priebeh trénovania môžeme vidieť na grafe 5.7.

Na testovacích dátach dosiahol average precision 97,79%. Pri výslednej detekcii táto verzia modelu preukazovala oveľa lepšie výsledky ako predchádzajúce. Na vzorku 12 scén,



Obr. 5.6: Priebeh trénovania binárneho klasifikátora na báze VGG16



Obr. 5.7: Priebeh trénovania klasifikácie do 102 tried

ktoré dokopy obsahovali 13 zbraní sme experimentálne určovali hodnotu prahu. Pri hodnote prahu 0,999 bol počet true pozitívov 12, true negatívov tiež 12, a v jednom prípade zbraň nedetekoval. Pri hodnote prahu 0,9999 ostali hodnoty zachované akurát počet true negatívov, čiže nesprávnych detekcií sa znížil na 10. Pri nastavení prahu na 0,99999 bol počet true pozitívov 7 true negatívov 2 a zbraň nedetekoval v 6 prípadoch.

Kapitola 6

Záver

Nebezpečenstvo je stále prítomnou súčasťou bežného ľudského života. Pozorného čitateľa napadne, že touto myšlienkou sme začali túto bakalársku prácu. Pravdou je, že tá úvodná myšlienka je predsa len trochu odlišná. Hovoríme v nej o bezpečnosti a nie o nebezpečenstve. Ani najlepší ľudský mozog sa kapacitou a presnosťou svojej pamäte nevyrovná možnostiam súčasnej výpočtovej techniky. A cesta k väčšej bezpečnosti vedie cez zníženie nebezpečenstva. Počítače a bezpečnosť – tieto dva pojmy boli kostrou zadania tejto bakalárskej práce – počítačovej detekcie zbrane v 2D obraze. Táto práca nám priniesla množstvo podnetov a prinútila nás ísť do hĺbky v mnohých rovinách. Týkalo sa to podrobnej analýzy rôznych princípov detekcie v odlišných situáciách reálneho života, postupov spracovania obrazu, metód rozpoznávania objektov, architektúr neurónových sietí, tvorby a trénovania klasifikátorov, softvérových knižníc a ďalších oblastí. Mnohé parciálne poznatky do seba zapadli a ukázali súvislosti a možnosti, ktoré nám predtým neboli viditeľné.

Implementáciou, trénovaním, testovaním a vyhodnocovaním získaných dát sme získali skúsenosti, ktoré sa nedajú získať inak ako konkrétnou experimentálnou prácou a umožnili nám splniť ciele, ktoré sme si stanovili.

Pri implementácii klasického modelu s použitím HOG-u ako deskriptora sme zistili, že je značne obtiažne nájsť takú konfiguráciu parametrov, ktoré by viedli k prijateľným výsledkom detekcie obrazov reálnych scén týmto modelom.

Model na báze Tiny Yolov3 naopak potvrdil náš predpoklad, že je vhodný aj pre detekciu zbraní tým, že sa nám podarilo dosiahnuť pri testovaní average precision takmer zhodnú s referenčnou hodnotou modelu.

Posledný model na báze neurónových sietí s cieľom dostatočne percentuálne úspešnej detekcie plne splnil naše predpoklady. Reálne nameranými výsledkami sme potvrdili, že pri splnení určitých podmienok je to model schopný detekcie zbraní v 2D obraze s dostatočnou mierou pozitívnej i prijateľnou hodnotou falošnej detekcie. Zároveň sme overili predpoklad, že predtrénované konvolučné neurónové siete predstavujú výraznú výhodu v situáciách, keď nedisponujeme dostatkom trénovacích dát. S potenciálnou aplikáciou niektorých vylepšení uvedených nižššie je reálne dosiahnuť ďalšie významné zlepšenie funkcionality modelu.

V úplnom závere tejto práce chceme ešte priniesť niekoľko dôležitých postrehov. Veľmi podstatným faktorom, ktorý ovplyvňoval dosiahnuté výsledky bol počet dát v datasete. Keďže neurónové siete pre správne fungovanie potrebujú vysoký počet vstupných dát, prvým krokom pre získanie lepších výsledkov by malo byť vytvorenie výrazne väčšieho datasetu, alebo programu, ktorý by tento dataset dokázal generovať. Ďalším vylepšením by s dosť značnou pravdepodobnosťou mohlo byť využitie najnovších architektúr CNN. Ako možný príklad uvádzame architektúru ResNet. Analogicky platí, že k lepším výsledkom by zrejme

viedlo aj použitie väčších verzií Yolo siete ako je jeho Tiny verzia. Zároveň je možné, že pre tvorbu anotácií k datasetu do sietí typu Yolo, by mohlo byť užitočné používať detektor na báze posuvného okna, ktorý je schopný dosiahnuť vysokú presnosť a generovať tak tieto anotácie. Taktiež, rozdeliť triedu krátkych zbraní na triedu krátkych zbraní držaných rukou, a krátkych zbraní umiestnených v scéne samostatne by mohlo priniesť pozitívne výsledky.

I najväčšie diela ľudského umu boli vytvorené nota po note, slovo za slovom. K najväčším objavom sme došli krok za krokom. A tak pevne veríme, že výsledky tejto práce aspoň malým kúskom prispejú k bezpečnosti nás všetkých.

Literatúra

[1] Internet Movie Firearms Database. [Online; navštívené 11.05.2018].
URL http://www.imfdb.org/wiki/Main_Page

[2] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, [Online; accessed 11.05.2018].
URL https://www.tensorflow.org/

[3] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Chen, H. M.; Lee, S.; Rao, R.; aj.: Imaging for concealed weapon detection: a tutorial overview of development in imaging sensors and processing. [Online; 11.05.2018].
URL https://ieeexplore.ieee.org/document/1406480/

[5] Chollet, F.; aj.: Keras. https://keras.io, 2015, [Online; navštívené 11.05.2018].

[6] CRISTIANINI, N. a. J. S.-T.: *An introduction to support vector machines: and other kernel-based learning methods.* New York: Cambridge University Press, 2000, ISBN 0521780195.

[7] Ekta M. Upadhyay, N. K. R.: Exposure fusion for concealed weapon detection. [Online; 11.05.2018].
URL https://ieeexplore.ieee.org/document/6926141/

[8] Faktor, Z.: *Střelné zbraně Konstrukce a funkce*. Praha: Magnet-Press, 1995, ISBN 8085847469.

[9] HOLLEMANS, M.: Real-time object detection with YOLO. [Online; navštívené 11.05.2018].
URL http://machinethink.net/blog/object-detection-with-yolo/

[10] Hrickova, M.: Neurónové siete. [Online; navštívené 11.05.2018].
URL http://neuron-ai.tuke.sk/corba/cigOld/source/publications/thesis/master_thesis/2000/hric/html/

[11] JAIN, R. K. a. B. G. S., Ramesh: *Machine vision*. New York: McGraw-Hill, 1995, ISBN 0070320187.

[12] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] Jones, E.; Oliphant, T.; Peterson, P.; aj.: SciPy: Open source scientific tools for Python. 2001–, [Online; navštívené 11.05.2018].
URL http://www.scipy.org/

[14] Justin Lai, S. M.: Developing a Real-Time Gun Detection Classifier. [Online; navštívené 11.05.2018].
URL http://cs231n.stanford.edu/reports/2017/pdfs/716.pdf

[15] Karpathy, A.: Convolutional Neural Networks. [Online; navštívené 11.05.2018].
URL http://cs231n.github.io/convolutional-networks/

[16] L. Yujiri, P. M., M. Shoucri: Passive millimeter wave imaging. [Online; navštívené 11.05.2018].
URL https://ieeexplore.ieee.org/document/1237476/

[17] Lukáč, M.: *Hlboké neurónové siete pre spracovanie multimédií*. Diplomová práca, Masarykova univerzita, Fakulta informatiky, 2016.
URL https://is.muni.cz/th/r1pur/thesis.pdf

[18] M., B.: Umělá inteligence. [Online; navštívené 11.05.2018].
URL
http://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--jednotlivy-neuron--matematicky-model-a-aktivni-dynamika-neuronu

[19] Michal Španěl, V. B.: Obrazové segmentační techniky. [Online; navštívené 11.05.2018].
URL http://www.fit.vutbr.cz/~spanel/segmentace/

[20] Mrázek, Z.: *Aplikace pro demonstraci metody Histogram of Oriented Gradients pro detekci objektů*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014.
URL https://dspace.vutbr.cz/bitstream/handle/11012/34220/Hlavn%C3%AD_dokument.pdf?sequence=2

[21] Ning, G.: convert.py. [Online; navštívené 11.05.2018].
URL https://github.com/Guanghan/darknet/blob/master/scripts/convert.py

[22] Pavlica, J.: *Detekce graffity tagu v obraze*. Bakalářská práce, Vysoké učení technické v Brně, Fakulta informačných technologií, 2017.
URL http://www.fit.vutbr.cz/study/DP/BP.php.cs?id=19524&file=t

[23] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[24] Punyawiwat, P.: CNN Models. [Online; navštívené 11.05.2018].
URL https://blog.datawow.io/cnn-models-ef356bc11032

[25] Redmon, J.: Darknet: Open Source Neural Networks in C.
http://pjreddie.com/darknet/, 2013–2016, [Online; navštívené 11.05.2018].

[26] Roberto Olmos, F. H., SihamTabik: Automatic handgun detection alarm in videos using deep learning. [Online; navštívené 11.05.2018].
URL https://www.sciencedirect.com/science/article/pii/S0925231217308196

[27] Rohit Kumar Tiwari, G. K.: A Computer Vision based Framework for Visual Gun Detection Using Harris Interest Point Detector. [Online; navštívené 11.05.2018].
URL https://www.sciencedirect.com/science/article/pii/S1877050915014076

[28] Samir K. Bandyopadhyay, S. R., Biswajita Datta: Identifications of concealed weapon in a Human Body. [Online; navštívené 11.05.2018].
URL https://arxiv.org/ftp/arxiv/papers/1210/1210.5653.pdf

[29] Sinčák P., A. G.: Neurónové siete: Inžiniersky prístup. [Online; navštívené 11.05.2018].
URL http://neuron-ai.tuke.sk/cig/source/publications/books/NS1/html/node8.html

[30] Sonka, M.; Hlavac, V.; Boyle, R.: *Image Processing: Analysis and Machine Vision*. CL-Engineering, druhé vydání, 1998, ISBN 053495393X.

[31] Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, ročník abs/1605.02688, Květen 2016, [Online; navštívené 11.05.2018].
URL http://arxiv.org/abs/1605.02688

[32] Tijtgat, N.: Understanding YOLOv2 training output. [Online; navštívené 11.05.2018].
URL https://timebutt.github.io/static/understanding-yolov2-training-output/

[33] Verma, G. K.: A computer vision based framework for visual gun detection using SURF. [Online;navštívené 11.05.2018].
URL https://www.researchgate.net/publication/281776103

[34] Xiao, Z.; Lu, X.; Yan, J.; aj.: Automatic detection of concealed pistols using passive millimeter wave imaging. *2015 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2015: s. 1–4.

[35] Zhiyun Xue, Y. L. L., Rick S. Blum: Fusion of visual and IR images for concealed weapon detection. *2015 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2015: s. 1–4.

Príloha A

Obsah priloženého DVD

DVD priložené k tejto práci obsahuje :

- Programs - adresár ktorý obsahuje zdrojové súbory implementovaných modelov, a v prípade modelu na báze Tiny Yolov3 configuračný súbor a súbory potrebné pre spustenie v rozhraní darknet.
- Datasets - adresár obsahujúci datasety použité pri implementácii jednotlivých modelov.
- bakalarskapraca.pdf - elektronická verzia práce.
- Sablona2018 - adresár obsahujúci súbory potrebné pre tvorbu pdf pomocou programu LaTeX.
- Readme - manuál.
- Models - adresár s natrénovanými modelmi a váhami v prípade Yolov3