VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VIZUÁLNÍ DETEKCE MALÝCH PŘEDMĚTŮ POMOCÍ DOSTUPNÝCH NÁSTROJŮ V PROSTŘEDÍ MATLAB
VISUAL DETECTION OF SMALL OBJECTS USING AVAILABLE TOOLS IN MATLAB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jiří Sladký**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Martin Appel**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Jiří Sladký**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Martin Appel**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Vizuální detekce malých předmětů pomocí dostupných nástrojů v prostředí MATLAB

Stručná charakteristika problematiky úkolu:

Motivací práce je vyzkoušení nástrojů pro práci s metodou YOLO, jejíž druhá verze byla nedávno implementována do programu MATLABU. YOLO využívá unikátní architekturu neuronové sítě, díky které předčí konkurenční systémy svou rychlostí při zachování srovnatelné přesnosti detekce.
Práce se bude zabývat přizpůsobením této metody pro detekci malých a podobných předmětů. Jedná se například o počítání stromů z leteckého snímku nebo počtu zvířat či lidí na obrázku. Cílem práce je nalezení limitů metody YOLO, případně její kombinace s jinou metodou, a posouzení použitelnosti dosažených výsledků pro potenciální praktické aplikace. Problém vzniká, pokud předměty splývají, nemají pevně dané hranice. Dobrým příkladem jsou hranice stromů z leteckého pohledu.

Cíle bakalářské práce:

1) Seznamte se se způsoby detekce objektů v obraze s využitím strojového učení. Popište princip metody YOLO.
2) Proveďte rešerši v oblasti detekce malých předmětů v obrazu.
3) Vytvořte nástroj, který bude počítat malé předměty v obrazu
4) Zjistěte hranice programu YOLO v programu MATLAB

Seznam doporučené literatury:

CORKE, Peter I. Robotics, vision and control: fundamental algorithms in MATLAB. Berlin: Springer, 2011. Springer tracts in advanced robotics, v. 73. ISBN 9783642201431.

VALÁŠEK, Michael. Mechatronika. Praha: České vysoké učení technické, 1995. ISBN 80-01-01276-X.

GREPL, Robert. Kinematika a dynamika mechatronických systémů. Brno: Akademické nakladatelství CERM, 2007. ISBN 978-80-214-3530-8.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

Abstrakt

Tato práce zkoumá možnosti detekce malých objektů na obrázcích pomocí metody YOLO, algoritmu hlubokého učení dostupného v programu MATLAB. V práci byl navržen a natrénován detektor, jež dokáže detekovat hospodářská zvířata při pohledu shora. Vytvořen byl nástroj, který provede detekce s pomocí představeného modelu i na obrázcích o vysokém rozlišení a následně spočítá přítomné objekty. Naprogramován byl generátor syntetických obrázků, který výrazně pomohl s natrénováním tohoto modelu. Byly realizovány různé experimenty, jež vedly k nalezení hranic metody YOLO a ověřily přínos představených vylepšení.

Summary

This thesis investigates possibilities of small object detection in pictures using YOLO method, a deep learning algorithm available in MATLAB. In the thesis, a detector was designed and trained to detect cows from top-down view. A tool was created, that performs detection using the proposed model even on high-resolution images and counts the present objects. A generator of synthetic images was programmed, which helped with training the model. Various experiments were performed that found the limits of YOLO and validated contribution of the proposed improvements.

Klíčová slova

Strojové učení, hluboké učení, počítačové vidění, umělé neuronové sítě, detekce objektů, YOLO, syntetický generátor, augmentace dat, detekce dobytka

Keywords

machine learning, deep learning, computer vision, artificial neural networks, object detection, YOLO, synthetic generator, data augmentation, cattle detection

Bibliografická citace

SLADKÝ, Jiří. *Vizuální detekce malých předmětů pomocí dostupných nástrojů v prostředí MATLAB*. Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/125193. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Martin Appel.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením svého vedoucího, a že jsem veškeré použité zdroje v práci citoval a uvedl v seznamu literatury.

V Brně dne 25. 6. 2020 Jiří Sladký

Děkuji svému vedoucímu panu Ing. Martinu Appelovi za asistenci při výběru atraktivního tématu, trpělivost, připomínky a celkové vedení práce.

Jiří Sladký

Obsah

1 Úvod 9
1.1 Cíle řešení 9
2 Rešerše 10
2.1 Hluboké učení 10
2.1.1 Plně propojené neuronové sítě 10
2.1.2 Konvoluční neuronové sítě 11
2.1.3 Trénování sítě 12
2.2 Metody detekce objektů 13
2.2.1 R-CNN 14
2.2.2 Fast R-CNN 14
2.2.3 Faster R-CNN 14
2.2.4 R-FCN 15
2.2.5 SSD 15
2.3 YOLO 16
2.3.1 Darknet-19 16
2.3.2 Princip detekce 17
2.3.3 Ztrátová funkce 18
2.3.4 Srovnání jednotlivých verzí 19
2.3.5 Srovnání s ostatními metodami 20
2.4 Detekce malých objektů 21
2.4.1 Klasické metody 21
2.4.2 Hustotní mapy 22
2.4.3 Sémantická segmentace 22
2.4.4 Vylepšení rámečkových detektorů 23
3 Analýza problému 25
4 Použité nástroje 26
4.1 Generátor syntetických obrázků 26
4.1.1 Příprava dat a parametry generování 26
4.1.2 Princip generátoru 27
4.1.3 Přednosti a nedostatky 29
4.2 Zpracování obrázků o vysokém rozlišení 29
4.2.1 Zpracování datové sady 29

4.2.2 Detekce a počítání objektů na velkém obrázku 30
4.3 Existující nástroje 31
4.3.1 Image Labeler 31
4.3.2 Deep Network Designer 31
5 Příprava detektoru 32
5.1 Datové sady 32
5.1.1 Japonská datová sada 32
5.1.2 Skutečná datová sada 32
5.1.3 Syntetická datová sada 33
5.2 Návrh architektury sítě 33
5.2.1 Výběr předtrénované sítě 33
5.2.2 Rozšíření pro detekci 33
5.3 Vyhodnocení přesnosti detekce 35
5.3.1 Metrika Average Precision 35
5.3.2 K-násobná křížová validace 35
6 Experimenty 36
6.1 Velikost objektů 36
6.2 Pyramidová struktura 38
6.3 Užitečnost syntetických dat 40
6.4 Finální detektor 42
6.5 Vyhodnocení na japonské datové sadě 43
6.6 Vizuální posouzení detektoru 45
6.7 Diskuze 47
6.7.1 Schopnost počítat malé objekty 47
6.7.2 Využitelnost dosažených výsledků a náměty na pokračování 48
7 Závěr 49
Literatura 50
Seznam zkratek 54
Seznam obrázků 55
Seznam tabulek 55

1 Úvod

Mezi současné problémy vědeckého bádání patří oblast počítačového vidění. Schopnost strojů vnímat obraz podobně jako člověk má v praxi široká uplatnění – od robotiky a autonomních vozidel přes bezpečnostní systémy až po diagnostiku v lékařství. S lepší dostupností a zvyšující se kvalitou obrazových dat roste potřeba je efektivně zpracovávat. K těmto účelům skvěle slouží umělé neuronové sítě, jelikož jsou schopny podchytit široké významové informace a zpřesňují detekci s rostoucím množstvím dat.

Ke komplikovanějším problémům počítačového vidění se řadí detekce objektů. Velkou výzvu představují převážně malé a zahuštěné objekty, se kterými se výzkum v posledních letech intenzivně potýká.

Takovými objekty se zabývá právě tato práce. K detekci je využita metoda YOLO (You Only Look Once), která oproti jiným metodám zjednodušuje celý proces trénování sítě a zrychluje následnou detekci. V práci je ověřováno, nakolik obtížné objekty dokáže tato metoda rozeznat.

1.1 Cíle řešení

Hlavní smysl práce spočívá ve využití metody YOLO, algoritmu hlubokého učení, a to konkrétně pro detekci malých objektů v obrazových datech. Cílem rešeršní části je obeznámit čtenáře s podstatou hlubokého učení a jeho základním kamenem – umělými neuronovými sítěmi. Budou nastíněny běžně používané metody detekce objektů. Bude rozebrána metoda YOLO a srovnána s ostatními metodami. Proběhne průzkum, jak současná věda chápe malé objekty, jakými aplikacemi se zabývá a jak k problematice přistupuje.

Na základě poznatků z rešerše bude zvolen konkrétní problém v oblasti detekce malých objektů. Ten musí být pro metodu YOLO dostatečně obtížný, zároveň však řešitelný. Nabízí se např. počítání stromů z leteckého snímku, lidí na demonstraci či hospodářských zvířat ve stádě. Důraz bude při výběru kladen na realizovatelnost a následné snadné posouzení limitů metody YOLO.

Cílem praktické části je vytvoření samotného nástroje detekujícího vybraný typ objektů. Na základě rešeršní části bude zvolena vhodná architektura neuronové sítě, případně pomocná procedura potřebná k řešení daného problému. Výstupem bude model s natrénovanými parametry, eventuálně více verzí modelu pro pozdější srovnání.

Výsledný model bude podroben sérii experimentů, jež by měly ověřit hranice použitelnosti metody YOLO. Jako kritéria funkčnosti mohou sloužit velikost, hustota a podobnost objektů, charakter jejich okolí, použitá trénovací data aj.

2 Rešerše

2.1 Hluboké učení

V současnosti dosahuje výrazných pokroků umělá inteligence, řešící nejrůznější sofistikované problémy. Do oboru umělé inteligence spadá strojové učení, které se vyznačuje algoritmy, jež se učí na základě příznaků (angl. *features*) z předložených dat předpovědět výstup. Extrakce příznaků je však prováděna pomocí specifických algoritmů použitelných jen pro úzce vymezený problém.

Podoblastí strojového učení je tzv. hluboké učení, které využívá umělé neuronové sítě, v počátcích inspirované uspořádáním neuronů v mozku. Jejich vznik sahá do první poloviny 20. století, avšak z důvodu vysokých výpočetních nároků byl jejich rozvoj utlumen. Intenzivně se začaly zkoumat s nástupem výkonnějších počítačů ve 21. století a od té doby vykazují skvělé výsledky v mnoha oborech, od marketingu přes rozpoznání řeči až po řízení autonomních vozidel. Dokáží se naučit širokou škálu příznaků a nevyžadují jejich explicitní naprogramování, čímž výrazně snižují náročnost implementace a umožňují řešení komplikovanějších problémů.

Hluboké učení se od tradičního strojového učení odlišuje v několika důležitých aspektech. Patřičnou metodu je nutno vybrat podle požadavků na řešení daného problému. K využití hlubokého učení je vhodné přistoupit v případě, že:

- disponujeme dostatečným množstvím dat pro natrénování modelu;
- nevíme, jaké příznaky jsou pro daný problém podstatné, nebo je jejich manuální extrakce složitá či vyžaduje pokročilé oborové znalosti;
- máme k dispozici dostatečné výpočetní kapacity;
- nepotřebujeme spolehlivě znát, podle čeho se model pro predikci rozhodl.

Velikou výhodou hlubokého učení je také fakt, že přibývající množství dat, která jsou dnes v mnohých odvětvích snáze dostupná, přesnost predikce stále zlepšuje, na rozdíl od klasického strojového učení, u kterého po dosažení potřebného množství dat výrazně nepomáhá. [1] [2]

2.1.1 Plně propojené neuronové sítě

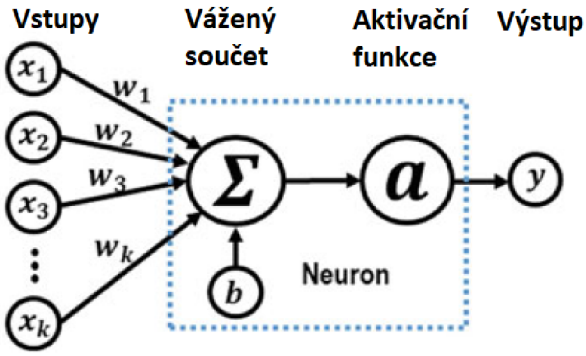
Základním kamenem hlubokého učení jsou neuronové sítě, které lze v nejjednodušším případě chápat jako shluky neuronů v několika navazujících vrstvách. První vrstva se nazývá vstupní a obsahuje tolik neuronů, kolik je vstupních dat. V případě vícerozměrných dat, např. obrázku, jsou jednotlivé pixely seřazeny do vektoru. Každý neuron reprezentuje jeden pixel. Poslední, výstupní vrstvu tvoří tolik neuronů, kolik je požadováno výstupů. Hloubka neuronové sítě se udává podle počtu hlubokých vrstev, tedy vrstev bez vstupní vrstvy. Velikost hlubokých vrstev jejich počet jsou voleny na základě zkušenosti a experimentů podle řešeného problému. Každý neuron hluboké vrstvy je na vstupu spojen se všemi neurony předchozí vrstvy, odtud přívlastek plně propojené. Spojení mají na neuron různý vliv, to ovlivňují váhy (*weights*). Výstup neuronu je upraven pevnou hodnotou zvanou práh (*bias*). Aktivační funkce vnáší do modelu nelinearitu. Spojení jednoho neuronu znázorňuje obr. 2.1. Hodnoty x_k jsou výstupy neuronů předchozí vrstvy o k neuronech, w_k váhy jednotlivých spojení, b prahová hodnota, a aktivační funkce, y výstup. Výstup neuronu se spočítá podle vztahu (2.1).

$$y = a\left(\sum_k (w_k \cdot x_k) + b\right) \tag{2.1}$$

Vztah lze rozšířit pro celou vrstvu do maticové podoby (2.2).

$$\mathbb{Y} = a(\mathbb{W} \cdot \mathbb{X} + \mathbb{B}) \tag{2.2}$$

Ve vztahu má $\mathbb{W}$ rozměr matice a $\mathbb{Y}, \mathbb{X}, \mathbb{B}$ jsou sloupcové vektory. [3]



Obr. 2.1: Model umělého neuronu. Přeloženo z [3].

2.1.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě našly své uplatnění zejména v oblasti počítačového vidění a jsou součástí téměř každého modelu zpracovávajícího obrazová data. Své jméno dostaly podle přítomnosti konvolučních vrstev. Ty se od plně propojených vrstev liší svou strukturou.

Princip konvoluční vrstvy znázorňuje obr. 2.2. Vstupní mapa příznaků X má velikost $x \times y \times z$, kde x, y značí velikost vrstvy a z udává počet kanálů. Na vrstvu se položí konvoluční jádro, tzv. kernel, značeno K. Jedná se o trojrozměrný tenzor o velikosti $n \times n \times z$, obsahuje tedy $n^2 z$ naučitelných vah. Výstupní mapa příznaků Y, neboli výsledek diskrétní konvoluce pro všechny polohy filtru, je velikosti $a \times b \times c$, kde a, b značí velikost vrstvy a c počet filtrů. Hodnota neuronu na pozici d, e, f vrstvy Y $Y_{d,e,f}$ se spočítá jako suma součinů překrývajících se prvků filtru a vstupní vrstvy. Matematicky psáno vztahem (2.3)

$$Y_{d,e,f} = b_Z + \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=0}^{z} K_{i,j,k} \cdot X_{d+i,e+j,z}, \tag{2.3}$$

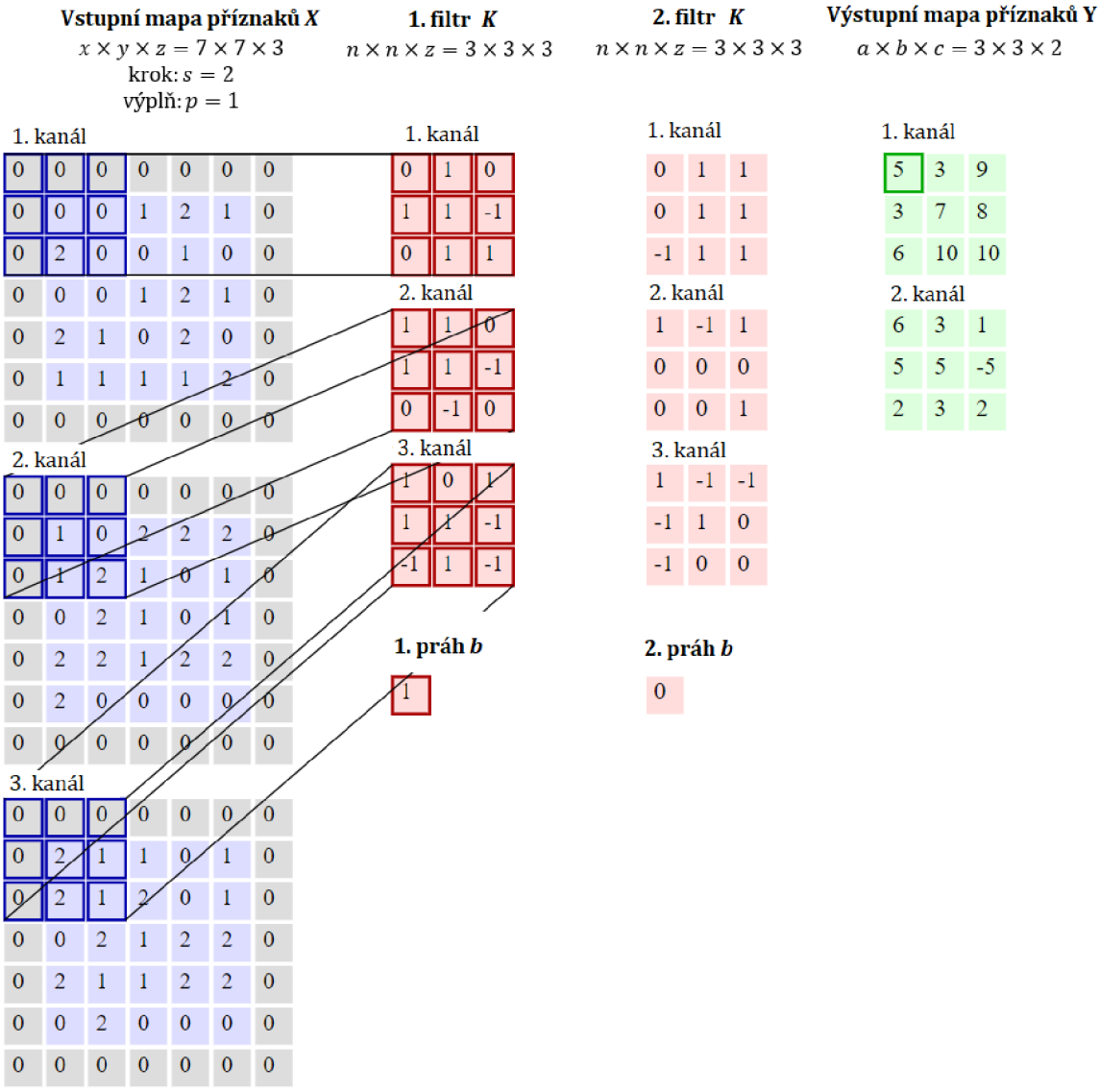
kde $K_{i,j,k}$ je váha filtru K na pozici i, j, k, $X_{d+i,e+j,z}$ je hodnota vstupní vrstvy X na pozici $d + i, e + j, z$ a b_Z je práh filtru na pozici z. Stejný filtr prochází celým obrazem, říká se, že má sdílené váhy. [1] [4]

Konvoluční vrstvy mají dva důležité parametry, jež je při návrhu sítě nutno zvolit. Výplň (*padding*) představuje rámeček samých nul o určité šířce, který je vložen okolo vstupní vrstvy. Konvoluce typicky zmenšuje první dva rozměry mapy příznaků o $n - 1$, výplň o hodnotě $(n - 1)/2$ tomu zabraňuje a označuje se jako nulová výplň. Krok s (*stride*) udává, po jakém kroku postupuje filtr přes vstupní vrstvu. Při nulové výplni se určí velikost výstupní mapy příznaků a, b jako $x/s, y/s$, dosahuje se tak zmenšení sítě a komprese příznaků. Alternativou je použití podvzorkovací (*pooling*) vrstvy.

Rozměr filtru n se nazývá *lokální receptivní pole*, jelikož charakterizuje, jak doširoka neuron „vidí“. Neuron není propojen se všemi neurony předchozí vrstvy, tak jako u plně propojených vrstev, ale jen s určitou oblastí. Z toho vyplývají aplikace konvolučních neuronových sítí. Hodí se na všechny typy dat, která mají charakter mřížky, např. zvukový signál, 2D nebo 3D obraz, neboli extrahují místní závislosti. Obrovská výhoda spočívá

v redukci parametrů. Oproti plně propojeným vrstvám jich obsahují mnohonásobně méně při zachování stejné efektivity. Další výhodou je proměnná velikost konvolučních vrstev, díky níž lze model trénovat na obrázcích různých rozměrů a dosáhnout tak rozměrové invariance.

Každý z c filtrů reprezentuje jistý příznak. Jak dokazuje článek [18], první konvoluční vrstvy rozlišují jednoduché příznaky jako hrany a naučí se je všechny sítě, přestože jsou trénovány na obrazových datech pro výrazně odlišné účely. Filtry hlubších vrstev se zaměřují na složitější tvary, specifické pro daný druh detekovaného objetu. [1] [4]



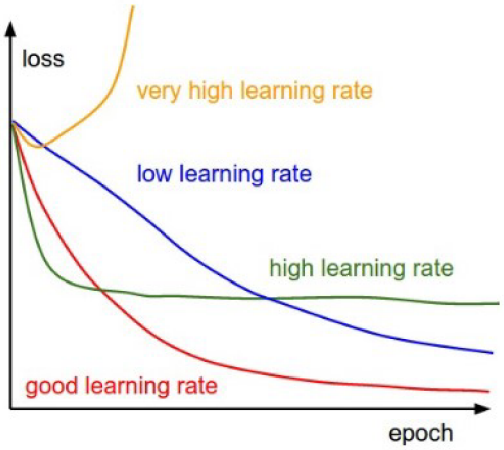
Obr. 2.2: Výpočet jedné konvoluce. Přeloženo z [35] a upraveno.

2.1.3 Trénování sítě

Základní myšlenka hlubokého učení je následující. Nejprve se navrhne vhodná architektura neuronové sítě o mnoha vrstvách. Modelu je předložen vstup, který je sítí zpracován. Na výstupu vrací síť požadovanou informaci, například predikci toho, co se nachází na obrázku. Parametry modelu, jako jsou váhy a prahy, je nutno před uvedením do provozu naladit tak, aby k žádaným predikcím došlo. Toho se docílí optimalizační metodou. Definuje se účelová funkce, častěji nazývaná ztrátová, jejíž hodnota se minimalizuje. Modelu jsou předkládána označená data z datové sady, výstup modelu je porovnán s požadovaným výstupem a model upravuje své

parametry tak, aby se daný výstup blížil požadovanému. Této metodě se říká učení s učitelem, model je během ní trénován.

Úprava parametrů je realizována prostřednictvím trénovacího algoritmu. Pro práci s obrazovými daty se používá nejčastěji minidávkový stochastický gradientní sestup (*minibatch SGD*[1]). Sestup znamená, že se hledá minimum ztrátové funkce. Gradientní značí, že je nutné získat parciální derivace ztrátové funkce podle jednotlivých parametrů neboli gradient. Stochastický proto, že se data rozdělí náhodně do dávek, výpočet gradientu a úprava parametrů probíhá po jednotlivých dávkách. Minima funkce se tak dosáhne rychleji než při výpočtu gradientu z celé datové sady. Aby gradient výrazně neměnil hodnotu mezi jednotlivými dávkami, zavádí se hybnost, typicky s hodnotou 0,9.



Obr. 2.3: Závislost ztrátové funkce na epoše pro různý krok učení. Převzato z [35].

Určení gradientu není triviální záležitost, model obsahuje mnohdy miliony parametrů, analyticky se gradient spočítat nedá. Uplatňuje se efektivní metoda zpětného šíření. Mapy příznaků (zvané též aktivace) se během dopředného průchodu ukládají. Parciální derivace se dopočítávají zpětně podle řetízkového pravidla na základě derivací předchozí vrstvy (z pohledu zpětného průchodu).

Klíčovým hyperparametrem (parametrem zvoleným tvůrcem) pro správné trénování je krok učení (*learning rate*). Volba příliš vysoké hodnoty způsobí oscilování kolem lokálního minima, nízká hodnota vede k příliš dlouhému trénování a konvergenci do nevhodného lokálního minima. Vliv kroku učení na konvergenci ilustruje obr. 2.3. Epocha zde udává, kolikrát proběhlo trénování na celé datové sadě. Obvykle se volí ze začátku vyšší hodnota kroku učení (10^{-2} až 10^{-4}, dle konkrétní úlohy) a během trénování se snižuje. [1] [35]

2.2 Metody detekce objektů

Podstatou detekce objektů je nalezení rámečků ohraničujících objekty na obrázku a určení jejich třídy. Na rozdíl od lokalizace mohou objekty vyskytující se v obrázku příslušet různým třídám. Jedná se o složitější úlohu než o pouhou klasifikaci obrázku jako celku, počet rámečků je pro každý obrázek proměnný, proto nelze jednoduše na konci konvoluční sítě použít plně propojenou vrstvu s předem daným počtem výstupů. Způsob, jakým je tento problém řešen, rozděluje metody detekce na dva základní typy – dvoustupňové a jednostupňové.

Dvoustupňové detektory sestávají ze dvou kroků. V prvním kroku se vytipují oblasti, které by mohly obsahovat objekt. Ve druhém kroku se každá oblast klasifikuje a určí se, zda oblast objekt obsahuje, případně jaký. Patří sem detektory z rodiny R-CNN[2] a jejich varianty.

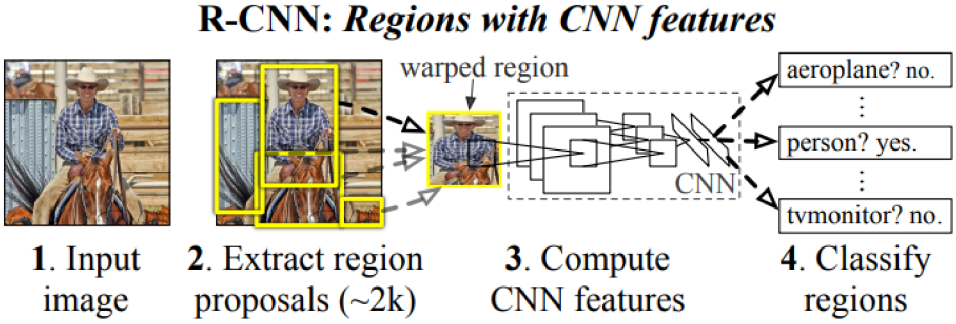
[1] Stochastic Gradient Descend
[2] Region Proposal Convolutional Neural Network

Jednostupňové detektory provádí detekci v rámci celé sítě, na výstupu predikují přímo polohu rámečků a třídy objektů. Díky tomu používají k predikci informace z větší části obrázku, ne pouze výřezu, a jsou schopny zaznamenat širší souvislosti. Řadí se mezi ně např. SSD a YOLO. [5]

Všechny metody představují svůj princip na konkrétních modelech natrénovaných pro obecnou detekci objektů. Typicky se používají datové sady ImageNet, Pascal VOC 2007 a 2012 a COCO [8] [22]. Pro konkrétní praktické aplikace se vytvářejí modely odlišné, princip původní metody však zůstává zachován.

2.2.1 R-CNN

R-CNN je prvním detektorem ze stejnojmenné rodiny. Princip ilustruje obr. 2.4. Síti je v prvním kroku předložen obrázek. Extrakce potenciálních oblastí v druhém kroku je prováděna algoritmem zvaným *selektivní výběr*, jež používá klasické segmentační metody. Navrhne typicky 2000 oblastí. Ve třetím kroku prochází každá z oblastí konvoluční sítí extrahující příznaky. Ty jsou ve čtvrtém kroku plně propojenými vrstvami analyzovány a dochází ke klasifikaci oblasti. Hlavní nevýhodu této metody představuje nízká rychlost. Kroky 3 a 4 se opakují pro každou oblast zvlášť, což je výpočetně náročné. [6]



Obr. 2.4: Princip R-CNN detektoru. Převzato z [6].

2.2.2 Fast R-CNN

Vylepšená metoda Fast R-CNN se oproti R-CNN liší ve využití příznaků extrahovaných konvoluční sítí. Obrázek prochází první částí sítě pouze jednou, což celkovou detekci urychluje zhruba 25krát. Speciální RoI[3] vrstva zpracuje příznaky příslušející navržené oblasti pomocí metody Max Pooling. Vznikne mapa příznaků o pevné, předem stanovené velikosti. Ta je předává plně propojeným vrstvám, které se na konci větví na část klasifikující oblast a část zpřesňující polohu rámečku. Navržené oblasti jsou získávány opět klasickým algoritmem z původního obrázku. [7]

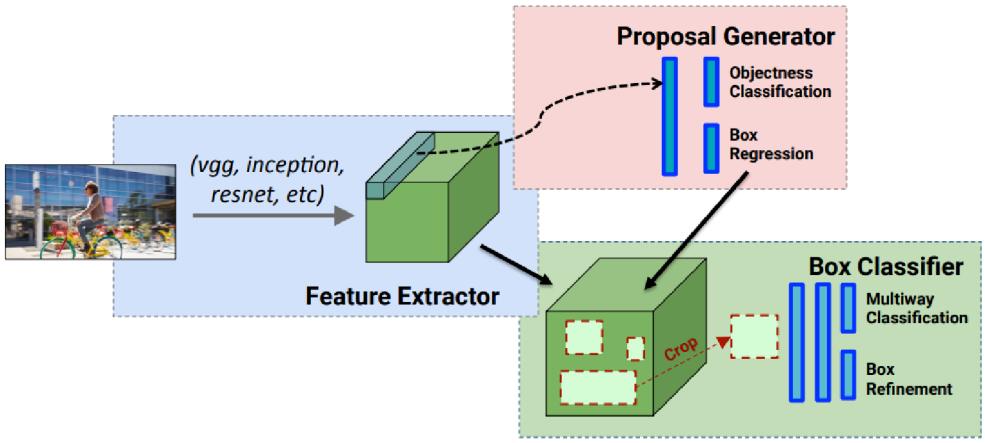
2.2.3 Faster R-CNN

Slabinou metody Fast R-CNN je právě relativně pomalý algoritmus selektivního výběru. V nové metodě je proto nahrazen neuronovou sítí, která je pro výběr potenciálních oblastí natrénována. Na obr. 2.5 je patrný princip této metody. Z původního obrázku jsou opět konvoluční sítí extrahovány příznaky. Ty vstupují do sítě navrhující oblasti (RPN[4]), příznaky jsou přidruženy návrhům stejnou vrstvou jako u Fast R-CNN a vznikají predikce v dalších

[3] Region of Interest
[4] Region Proposal Network

vrstvách. Odstraněním klasického algoritmu došlo k desetinásobnému zrychlení detekce. Detektory využívající tuto metodu patří v současnosti k těm nejpřesnějším, ale rychlostí a výpočetními nároky stále komplikují použití v reálném čase a v embedded systémech.

RPN využívá poprvé rámečky předdefinovaných rozměrů, tzv. kotevní rámečky (*anchor boxes*), které se hojně používají u jednostupňových detektorů. Rozměry oblasti jsou předpovídány vzhledem k předdefinovaným rámečkům. [7]



Obr. 2.5: Princip Faster R-CNN detektoru. Převzato z[36].

2.2.4 R-FCN

Jak prozrazuje název, R-FCN[5] je plně konvoluční neuronová síť využívající návrhy oblastí. Modifikovaná RoI vrstva citlivá na polohu analyzuje jednotlivé mapy příznaků. Na základě návrhu oblasti se část mapy příznaků (výstupu konvoluční sítě) rozdělí na podoblasti. Každá podoblast je analyzována filtrem, který udává, s jakou pravděpodobností daná podoblast patří příslušnému objektu. Tyto pravděpodobnosti se zprůměrují přes celou oblast, výstupem je pravděpodobnost, že navržená oblast obsahuje daný objekt. Intuitivně lze říci, že filtr levé horní podoblasti hlásí příslušnost ke třídě tvář, pokud se v podoblasti nachází pravé oko. [10] [37]

Na rozdíl od Faster R-CNN není každá oblast zpracovávána další sérií vrstev, ale pouze jednoduchou strukturou se zanedbatelnými nároky vůči zbytku sítě. To proces urychluje při drobném snížení přesnosti detekce. Díky tomu tvoří sítě na bázi R-FCN jistý kompromis mezi Faster R-CNN a jednostupňovými detektory. [11]

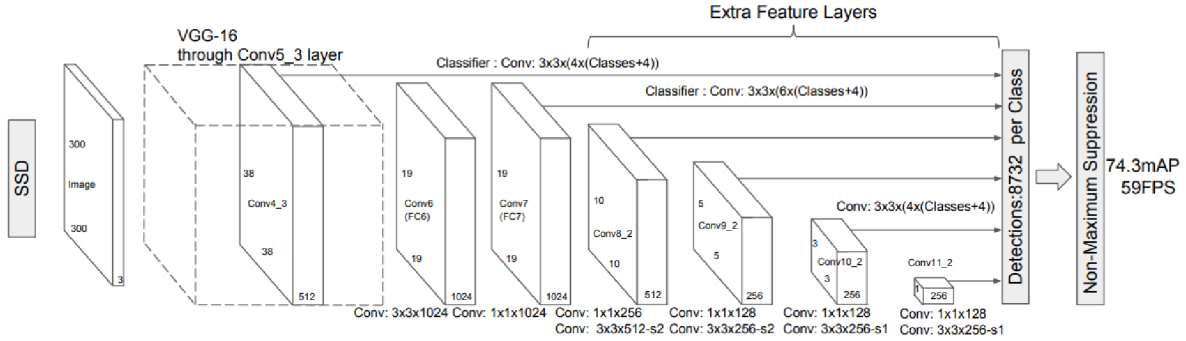
2.2.5 SSD

V roce 2015 byla představena první jednostupňová, velice rychlá metoda pro detekci objektů, YOLO, jež bude popsána v následující kapitole. Ještě tentýž rok byl publikován model metody SSD[6], který dosahuje rychlosti 59 FPS[7] při podobné přesnosti jako Faster-RCNN.

Základ tvoří konvoluční síť pro extrakci příznaků, tzv. páteřní síť (*backbone*). Model z článku [12] využívá část modelu VGG-16 [13] natrénovaného pro klasifikaci obrázků. Za ní následuje několik zmenšujících se konvolučních vrstev určených pro detekci (viz obr. 2.6). Jedna vrstva páteřní sítě a 5 detekčních vrstev vytvářejí predikce pro různá měřítka. Pozdější vrstvy slouží k detekci větších objektů, jelikož každá jejich buňka reprezentuje větší část obrázku.

[5] Region-based Fully Convolutional Network
[6] Single Shot MultiBox Detector
[7] Měřeno na grafické kartě NVIDIA Titan X

Pro každou buňku je získán vektor obsahující $n \cdot (4 + c + 1)$ kanálů, kde n udává počet kotevních rámečků a c počet tříd. Číslo 1 značí přidání třídy pozadí. Pro každou třídu je předpovězena pravděpodobnost jejího výskytu v daném rámečku. Číslo 4 představuje 4 parametry predikovaného rámečku – dva pro polohu středu a dva pro velikost. Parametry jsou vztaženy k poloze a velikosti kotevního rámečku a jsou zakódovány vhodnou funkcí, aby nenarušily stabilitu trénování. Každá výstupní vrstva předpovídá 4 nebo 6 kotevních rámečků. Jejich poměry stran jsou voleny manuálně, konkrétní velikost je dána vrstvou, v které jsou použity. [12] [38]



Obr. 2.6: Schéma SSD detektoru. Převzato z [12].

2.3 YOLO

Jak napovídá název, YOLO[8] detekuje objekty na základě jediného průchodu vstupního obrázku sítí. Zavádí jednoduchý a přehledný trénovací systém, tzv. *end-to-end training*, jelikož je celý proces součástí jediné konvoluční neuronové sítě s jednou ztrátovou funkcí. Na rozdíl od rodiny R-CNN chápe detekci jako regresní problém. Předpovídá spojité hodnoty reprezentující polohy rámečků a pravděpodobnosti tříd. [14]

Architekturu sítě na bázi YOLO lze pomyslně rozdělit na dvě části stejně jako u SSD. První část tvoří konvoluční síť pro extrakci příznaků. Druhá část zahrnuje několik vrstev určených pro detekci. [14]

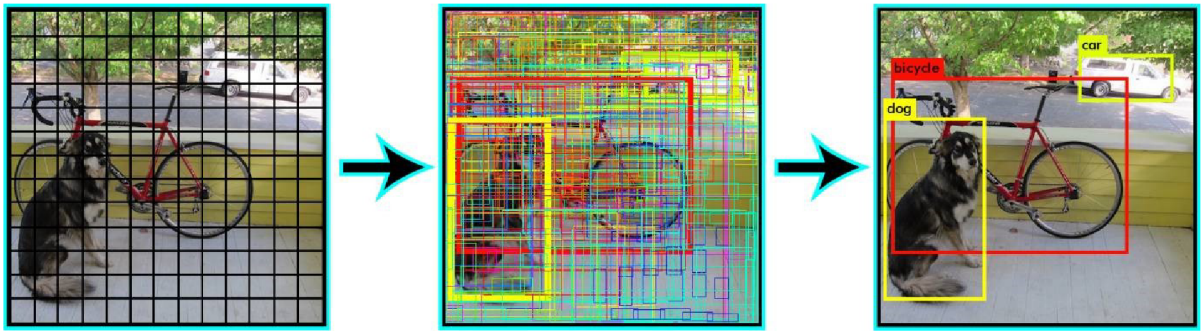
V současnosti existují tři verze metody YOLO. Články představující jednotlivé verze vyšly v červnu 2015 [14], v prosinci 2016 [15] a dubnu 2018 [16]. Došlo k výrazným vylepšením jak rychlosti detekce, tak především její přesnosti. Toho bylo docíleno převážně použitím nástrojů, které se začaly v posledních letech na poli detekce objektů hojně využívat [15]. Princip metody však zůstává zachován. V době vzniku této práce byla aktuální verzí MATLABu verze R2019b. Vzhledem k tomu, že je v ní implementováno YOLOv2, budou popsány detaily této verze a rozdíly vůči v1 a v3 následně shrnuty.

2.3.1 Darknet-19

Představený model verze YOLOv2 v článku [15] využívá vlastní síť pro extrakci příznaků, Darknet-19, tvořenou 19 konvolučními vrstvami. Ta byla nejprve natrénována na datasetu ImageNet na obrázcích o rozlišení 224 × 224 pixelů. Následně byla dotrénována na rozlišení 448 × 488 za účelem jemnějšího zpracování příznaků. Jednoduchou změnu rozměru umožnil fakt, že je na výstupu místo plně propojených vrstev použit speciální typ podvzorkovací vrstvy – *global average pooling* [17].Vznikla tak přesná a rychlá klasifikační síť tvořící základ detekce. Po odstranění podvzorkovací vrstvy a poslední konvoluční vrstvy byly přidány další 4

[8] You Only Look Once

konvoluční vrstvy pro detekci. Jako páteřní síť modelu metody YOLO lze ovšem použít jakoukoli jinou síť podle požadavků na přesnost, rychlost a paměťové nároky. [15]



Obr. 2.7: Princip metody YOLOv2. Převzato z [39].

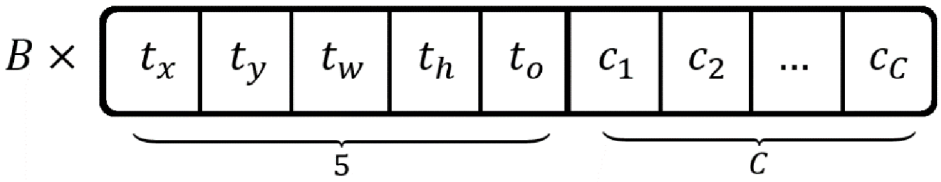
2.3.2 Princip detekce

Princip se odvíjí od toho, jak je interpretován výstup poslední, konvoluční vrstvy. Z podstaty konvolučních sítí tvoří výstup mřížku o velikosti $S \times S$ buněk, která obrázek pomyslně rozděluje (obr. 2.7 vlevo). Hodnota S se určí na základě velikosti vstupního obrázku N a tzv. podvzorkovacího faktoru p dle vztahu (2.4). N se volí jako celočíselný násobek p, aby S vyšlo celočíselně.

$$S = \frac{N}{p} \tag{2.4}$$

Podvzorkovací faktor je dán mírou podvzorkování v celé síti. Modely YOLO podvzorkovávají 32×, konkrétně základní model YOLOv2 redukuje obrázek o velikosti 416×416 pixelů na mřížku velikosti 13×13 buněk. Rozměr vstupního obrázku se proto volí jako násobek 32.

YOLOv2 predikuje pro každou buňku mřížky předem daný počet rámečků (obr. 2.7 uprostřed). Mapa příznaků poslední vrstvy obsahuje $B \cdot (5 + C)$ kanálů, kde B značí počet predikovaných rámečků a C počet tříd (viz obr. 2.8).



Obr. 2.8: Vektor hodnot pro jednu buňku výstupní mřížky u YOLOv2.

První dva parametry t_x, t_y reprezentují polohu středu predikovaného rámečku (na obr. 2.9 modře) vůči levému hornímu rohu buňky. Umístění tohoto rohu je dáno indexem buňky v mřížce, jeho polohu od levého horního rohu původního obrázku udávají hodnoty c_x, c_y. Další dva parametry t_w, t_h pak slouží k určení velikosti predikovaného rámečku vůči velikosti příslušného kotevního rámečku (čerchovaně). Rovnice na obr. 2.9 udávají, jak se získají výsledné souřadnice rámečku b_x, b_y a šířka a výška rámečku b_w, b_h. Pro stabilnější trénování jsou použity exponenciála a sigmoida σ, pro kterou platí vztah (2.5).

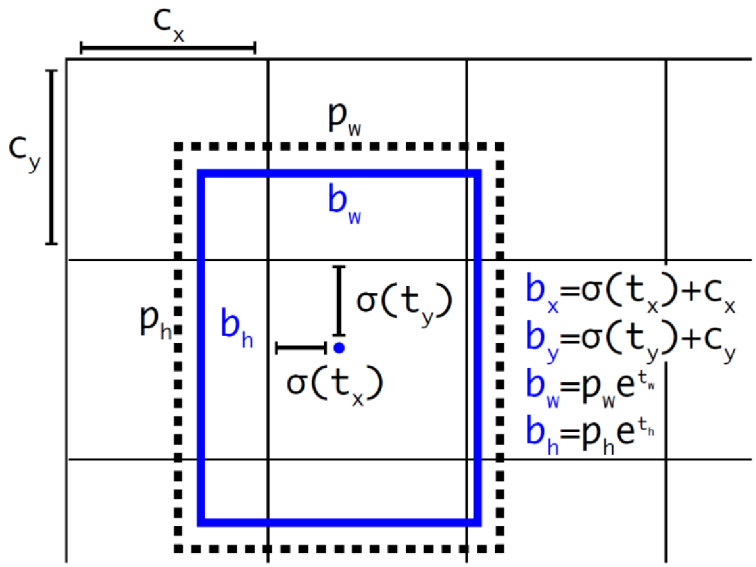
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Hodnoty c_i udávají podmíněnou pravděpodobnost – neboli pravděpodobnost, že rámeček patří i-té třídě za podmínky, že obsahuje objekt.

Parametr t_o slouží k výpočtu hodnoty, která reflektuje tzv. skóre jistoty (*confidence score*). To udává, jak moc si je model jistý, že rámeček obsahuje objekt ($Pr(\text{object})$), a jak přesné umístění model odhaduje ($IoU(b, \text{object})$). Formálně psáno vztahem (2.6). [14]

$$Pr(\text{object}) \cdot IoU(b, \text{object}) = \sigma(t_o) \tag{2.6}$$

IoU (*Intersection over Union*) značí obecně míru překryvu dvou obrazců, a to jako jejich průnik podělený sjednocením. [5] Model tedy předpovídá IoU mezi predikovaným rámečkem a skutečným rámečkem vymezujícím detekovaný objekt.



Obr. 2.9: Význam parametrů predikujících rámeček. Převzato z [15].

Z predikovaných rámečků zůstávají pouze ty, které přesáhnou svou hodnotou skóre jistoty předem definovaný práh. U velkých objektů a objektů ležících na hranici buněk mohou i tak vznikat redundantní detekce [14]. K jejich vyfiltrování slouží algoritmus *Non-max suppression*. Ten vybere ze shluku rámečků ten s nejvyšší hodnotou jistoty a odstraní všechny ostatní, které se s ním překrývají více, než je dovoleno další prahovou hodnotou. Výsledek pak vypadá jako na obr. 2.7 vpravo. [14] [5]

2.3.3 Ztrátová funkce

Klíčovým prvkem každé metody je ztrátová funkce, která je během trénování minimalizována. Přesná podoba ztrátové funkce pro YOLOv1 je uvedena v původním článku [14]. Pro YOLOv2 se liší pouze nepatrně, avšak není oficiálně dostupná. K dohledání jsou pouze odlišné interpretace, dokonce i ztrátová funkce uvedená v dokumentaci MATLABu [40]. je očividně nepřesná. Představená funkce sestávající z členů (2.7) – (2.11) vychází z [40] a snaží se vystihnout odlišnosti od první verze, nelze ji však brát doslovně. Význam jednotlivých členů je ale zachován a rozebrán níže.

$$Ztráta = K_1 \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{2.7}$$

$$+ K_1 \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2.8)$$

$$+ K_2 \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} \left(C_{ij} - \hat{C}_{ij} \right)^2 \quad (2.9)$$

$$+ K_3 \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{noobj} \left(C_{ij} - \hat{C}_{ij} \right)^2 \quad (2.10)$$

$$+ K_4 \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \quad (2.11)$$

V uvedených členech vrací $\mathbb{1}_{ij}^{obj}$ hodnotu 1, je-li *j*-tý kotevní rámeček v *i*-té buňce zodpovědný za detekování objektu, v opačném případě hodnotu 0. Zodpovědnost je přidělena kotevnímu rámečku, který má s *ground-truth* (označeným objektem v trénovacím obrázku) nejvyšší IoU. S^2 značí počet buněk, B počet kotevních rámečků, znak se stříškou náleží hodnotě z ground-truth, znak bez stříšky odkazuje na predikovanou hodnotu. [14]

Ztrátová funkce metody YOLO se skládá z pěti členů. Vliv členů na celkovou ztrátu ovlivňují váhy K_1, K_2, K_3, K_4. YOLOv1 volí $K_1 = 5$, $K_2 = 1$, $K_3 = 0{,}5$, $K_4 = 1$ z důvodu stability trénování. Výrazy charakterizují tři typy chyb:

- **Chyba lokalizace** – Člen (2.7) vystihuje odchylku od správné polohy rámečku, člen (2.8) odchylku od správné velikosti. Odmocniny zde zajišťují, že absolutní odchylka u malých rámečků má větší vliv na chybu než u velkých rámečků.
- **Chyba jistoty** – Pokud je rámeček zodpovědný za predikci objektu, užije se člen (2.9) a penalizuje se odchylka od IoU mezi předpovězeným a skutečným rámečkem, což vychází z rovnice (2.6). Pro ostatní případy je aplikován člen (2.10), žádaná hodnota je zde rovna 0.
- **Chyba klasifikace** – Člen (2.11) udává odchylku od požadované třídy pro ty rámečky, které obsahují objekt. Žádaná hodnota je zde 1 pro správnou třídu a 0 pro ostatní třídy.

2.3.4 Srovnání jednotlivých verzí

Druhá verze metody YOLO přinesla mnoho změn, které přispěly ke zkrácení výpočetního času a k výraznému zpřesnění detekce (mAP[9] stouplo o 15,2 %). Uvedený seznam shrnuje podstatné změny. [15]

- **Plně konvoluční síť** – Byly odstraněny plně propojené vrstvy na konci sítě a nahrazeny konvolučními vrstvami. To dovoluje předkládat síti obrázky o různém rozlišení. Výsledné detekce v jednotlivých buňkách už však nejsou ovlivněny příznaky z celého obrázku, ale pouze z okolí charakterizovaného receptivním polem poslední konvoluční vrstvy. Navzdory tomu je stále zohledněn větší kontext než u dvoustupňových metod.
- **Klasifikační síť s vyšším rozlišením** – Síť pro extrakci příznaků se běžně nejprve trénuje na klasifikačním problému při malém rozlišení. YOLOv2 ke konci trénování toto rozlišení zvyšuje, aby se síť naučila detekovat na základě jemnějších příznaků.
- **Nová klasifikační síť** – Použití sítě Darknet-19 celou detekci zrychlilo.

[9] mean average precision – vysvětleno v kapitole 5.3.1.

- **Jemnější mřížka** – YOLOv1 podvzorkovávalo 64×, což vedlo ke ztrátě detailů. Při fixní vstupní velikosti 448 × 448 tvořilo mřížku 7 × 7. Nyní tvoří při základním rozlišení 416 × 416 mřížku 13 × 13, rozšiřitelnou při použití vyššího rozlišení.
- **Větší variabilita predikce** – YOLOv1 mohlo detekovat pro každou buňku pouze jednu třídu. Navíc používalo jen 2 rámečky na buňku, YOLOv2 jich využívá 5.
- **Kotevní rámečky** – YOLOv1 předpovídalo velikost rámečků jako násobek původní velikosti obrázku. Oba rámečky vycházely z poměru stran 1:1 a v průběhu trénování se specializovaly. Stejný poměr stran však způsoboval nestabilitu. Proto využívá YOLOv2 kotevní rámečky o předem zvolených velikostech. Po jejich manuálním zadání dokázal model podchytit větší množství objektů při drobném snížení přesnosti. Tu výrazně zvýšila definice kotevních rámečků pomocí algoritmu *k-means clustering*. Takto získané rámečky jsou v článku nazývány *priors*.
- **Propojení map příznaků** – YOLOv2 spojuje příznaky z poslední extrakční vrstvy s jemnějšími příznaky z předchozí vrstvy. Spojení se projeví v navýšení počtu kanálů.
- **Vícerozměrné trénování** – Během trénování je měněna velikost vstupních obrázků v intervalu 320 až 608 pixelů, vždy jako násobek 32. Síť je tak použitelná při více rozlišeních a snáze rozpoznává objekty při různých velikostech.
- **Vyšší rozlišení** – zvětšení vstupní velikosti na 608 × 608 zvýšilo přesnost detekce. To samozřejmě vedlo ke zpomalení, avšak model stále pracuje rychleji, než je hraniční hodnota pro reálný čas – 30 FPS.

Třetí verze YOLO se zaměřuje hlavně na zvýšení přesnosti. Toho bylo dosaženo použitím komplexnějšího modelu a chytrých řešení jiných metod. Došlo k následujícím změnám. [16]

- **Rozsáhlejší síť** – autoři natrénovali novou klasifikační síť pro extrakci příznaků. Je tvořena 53 konvolučními vrstvami, odtud název Darknet-53. Inspiruje se propojeními z metody ResNet [19], ale je rychlejší při stejné přesnosti.
- **Vícestupňové predikce** – YOLOv3 generuje predikce pomocí 3 vrstev na základě různého rozlišení. První z nich vychází z nejhrubší mřížky (podvzorkování 32×) a detekuje velké objekty. Následně je inverzní operací k podvzorkování mřížka 2× zvětšena. Vzniklá mapa příznaků je poté spojena s jemnější mapou z nižších vrstev, stejně jako u YOLOv2. Proces se ještě jednou opakuje, což vede k velmi jemné mřížce s podvzorkováním pouze 8×. Pro každou úroveň predikuje YOLOv3 tři kotevní rámečky. Tato inovace inspirovaná FPN[10] metodou [20] výrazně pomohla s detekcí malých objektů, lehce však zhoršila detekci těch středních a větších.
- **Změny ve ztrátové funkci** – YOLOv3 počítá chybu jistoty přes logistickou regresi, ne přes střední kvadratickou chybu jako předchozí dvě verze. Navíc zavádí prahovou hodnotu, díky které nepenalizuje rámečky s dostatečným IoU s ground-truth.
- **Predikce více tříd** – nová verze je schopná pracovat s objekty definovanými více třídami, např. třídy *osoba* a *žena* se nevylučují. Ve ztrátové funkci je to zohledněno použitím logistické regrese v členu (2.11) místo standardní křížové entropie.

2.3.5 Srovnání s ostatními metodami

Rodina YOLO detektorů vyniká svou použitelností v reálném čase nebo na slabších zařízeních. Oproti SSD metodě je o něco přesnější a rychlejší, záleží však na konkrétní aplikaci. Rychlost je vykoupena snížením přesnosti oproti kombinaci Faster R-CNN s FPN nebo RetinaNet [21].

[10] Feature Pyramid Networks

Pro aplikace vyžadující velmi nízké výpočetní nároky představili autoři pro každou verzi odlehčené modely TinyYOLO.

Při porovnání různých metrik přesnosti v článku [16] je patrné, že se YOLO dopouští větších odchylek v umístění rámečků než zbylé metody. Projevuje se to v podmínce, kdy je detekce považována za správou při IoU alespoň 0,75 [16]. To je podmínka přísná, ne vždy vyžadovaná. Postačují-li pro aplikaci hrubější detekce, YOLO je vhodným řešením.

2.4 Detekce malých objektů

Malé objekty lze definovat na základě jejich rozlišení. Jejich detekce a přesná lokalizace představuje mnohem větší výzvu. Detektory popsané v předchozí kapitole jsou trénovány a testovány na datasetech, které obsahují převážně relativně velké objekty. V sadě COCO jsou malé objekty zastoupeny ve větší míře než například u Pascal VOC [22]. COCO je rozděluje do tří kategorií podle plochy objektu. Plochou je myšlen počet pixelů, kterými je objekt tvořen. Malé objekty mají plochu menší než 32^2 pixelů, velké plochu větší než 96^2, střední jsou mezi tím. [41] Současné detektory vykazují v kategorii malých objektů 2× až 3× horší výsledky než u středních a velkých [23] [41]. Plocha rámečku opsaného objektu je však vždy větší než plocha objektu, jak demonstruje obr. 2.10 (vlevo 3,4×, vpravo 2,1×), proto není metrika z COCO tak přísná, jak by mohlo vyplynout ze samotných čísel. V mnohých aplikacích je nutné detekovat výrazně menší objekty, v extrémních případech vozidla o velikosti 12 × 6 pixelů [24].

Obr. 2.10: Srovnání plochy rámečku a plochy objektu

Za malé objekty lze považovat i objekty, jež zabírají malou plochu původního obrázku. Satelitní snímky nebo fotografie z dronů zachycují rozsáhlý prostor a mohou být pořízeny s dobrým rozlišením. Klasické detektory by tyto snímky zmenšily na vstupní velikost sítě, čímž by došlo k výrazné ztrátě informací. Vhodným zpracováním takovýchto obrázků lze zachovat potřebné detaily, jak bude rozebráno později.

2.4.1 Klasické metody

Mezi problematické relativně malé objekty lze zařadit i překrývající se a splývající předměty. Ukázkovým příkladem jsou koruny stromů z leteckých snímků, jejichž přesné oddělení je náročné i pro člověka. Na nich je vhodné demonstrovat algoritmy nevyužívající hluboké učení, jejichž cílem je nalezení hranic jednotlivých stromů, tzv. delineace. Té je dosaženo kombinací následujících metod využívajících poznatku, že špičky stromů bývají světlejší než stinné okraje. Úspěšnost detekce se pohybuje cca od 80 % po 96 %, v závislosti na konkrétním typu lesa a podmínkách. [25]

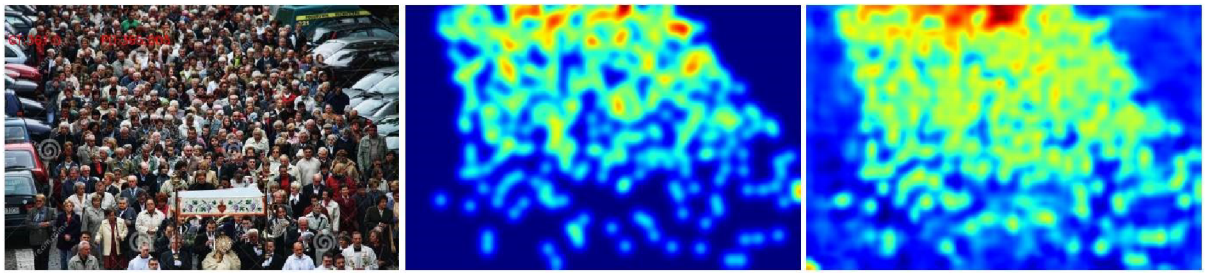
- **Nalezení lokálního maxima** – Pro odstranění změn jasu v rámci jednotlivých stromů je aplikován Gaussův filtr pro rozostření. Velikost filtru hraje klíčovou roli a je volena experimentálně. Ke spolehlivému odlišení stromů od pozadí je nutno využít multispektrální data. Metoda je vhodná pro jehličnaté stromy.

- **Zaplavovací metoda** – Tmavé pixely (lokální minima) jsou postupně „zaplavovány“ po určitou prahovou hodnotu a ohraničují jednotlivé koruny.
- **Oblastní metoda** – Nejprve se na základě lokálních maxim vytipují potenciální oblasti. Ty jsou následně prověřovány, zda obsahují rozložení jasu odpovídající stromu. Rámcově se podobá R-CNN metodě z kapitoly 2.2.1.
- **Shoda šablony** – Vytvoří se modely stromů postihující nejrůznější případy, ty slouží jako šablona. Oblasti kolem každého pixelu obrázku se srovnávají se šablonou, dostatečná shoda reprezentuje strom. Metoda vyžaduje obrovské množství šablon.

2.4.2 Hustotní mapy

Některé aplikace vyžadují práci s výrazně zhuštěnými, překrývajícími se objekty o ploše blížící se jednotkám pixelů. Jednou z nich je odhad počtu lidí v davu, obtížnost ilustruje obr. 2.11. Cílem metody je předpovědět hustotní mapu, která po integraci udává počet lidí na obrázku. Dříve se k tomu používaly klasické metody, v současnosti je výrazně překonávají konvoluční neuronové sítě. Vynalézají se různé architektury sítě, které se snaží podchytit kontext a měřítko – klíčové aspekty pro stanovení přesného počtu objektů, jež jsou samostatně neidentifikovatelné. [26]

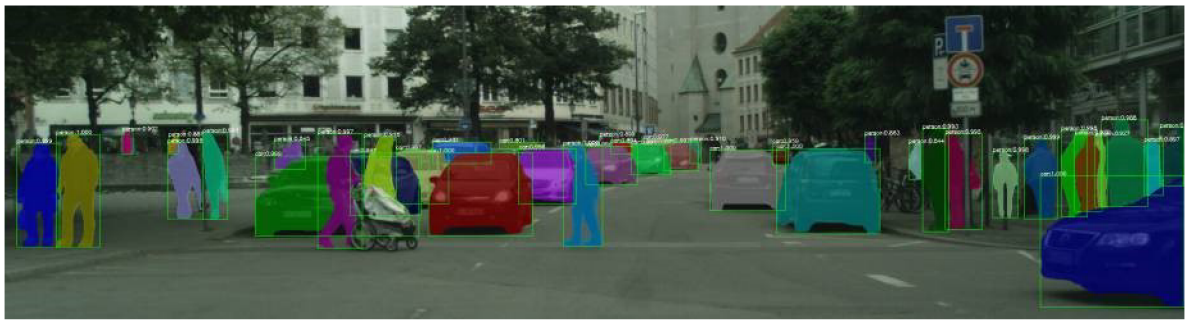
Mnohé kamery zabírají obraz od popředí k pozadí. Vlivem perspektivy vznikají obrovské rozdíly mezi velikostí lidí v obraze – od výrazných osob v popředí po neznatelný dav v pozadí. Tento problém řeší práce [27]. Nejprve se vygeneruje hloubková mapa, která segmentací rozdělí obraz na blízkou a vzdálenou oblast. V blízké oblasti jsou detekováni lidé pomocí rámečků metodou YOLOv1. Ze vzdálené oblasti se vytvoří hustotní mapa. Celkový počet lidí na obrázku je dán součtem obou metod. Práce dokazuje, že v popředí selhávají hustotní mapy, v pozadí detekování objektů pomocí rámečků. Novější verze metody YOLO by pravděpodobně rozpoznala větší část lidí, stále má však své limity a pro detekci extrémně zahuštěných oblastí není vhodná.



Obr. 2.11: Odhad počtu lidí na obrázku. Vlevo: původní obrázek. Uprostřed: skutečná hustotní mapa (361 lidí). Vpravo: predikovaná hustotní mapa (365 lidí). Převzato z [28].

2.4.3 Sémantická segmentace

Cílem sémantické segmentace je přiřadit každý pixel obrázku určité třídě. To však nerozlišuje jednotlivé objekty stejné třídy. Separované objekty lze získat pomocí instanční segmentace. Jedná se o klasický problém počítačového vidění, v kterém v současnosti vyniká hluboké učení. Nejznámější metodou je Mask R-CNN [29], vylepšenou metodou např. PANet [30]. Z předpovězených masek lze snadno stanovit rámečky ohraničující detekované objekty (viz obr. 2.12). Díky jemnějšímu rozlišení objektů na úrovni pixelů došlo k zpřesnění detekce, zlepšení se však projevuje pro různé velikosti objektů úměrně. [30]



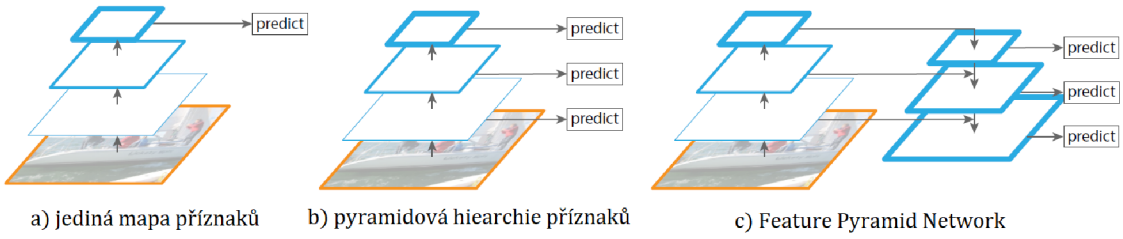
Obr. 2.12: Výsledky detekce pomocí metody PANet. Převzato z [30].

2.4.4 Vylepšení rámečkových detektorů

Metody pro detekci nebo počítání malých objektů popsané v předchozích kapitolách jsou postaveny na nerámečkovém principu. Praktická část práce je však založena na rámečkové metodě YOLO, proto nejsou přímo využitelné. Obecné detektory jako YOLO představené v kapitolách 2.2 a 2.3 slouží k detekci rozmanitých objektů různých velikostí. Nástroje prezentované v jedné metodě se často uplatňují v upravené podobě v druhé metodě, např. kotevní rámečky, vícestupňová detekce, rozdělení obrázku na mřížku. Jejich syntéza vede ke zdokonalení současných detektorů. Některé nástroje pomáhají převážně s detekcí malých objektů a při návrhu vhodné architektury sítě v praktické části je rozumné zvážit jejich užití.

Pyramidová struktura

Predikce z jediné mapy příznaků (např. YOLOv1) pro malé objekty nestačí, zachovány jsou pouze hrubé rysy charakterizující velké objekty (obr. 2.13a). Víceúrovňové predikce (např. SSD) sice zachovávají v nižších vrstvách detaily, neobsahují ale širší významové informace (obr. 2.13b). Kombinace jemných rysů s kontextem obrázků zajišťuje pyramidová struktura FPN [20] na *obr. 2.13*c. Mapy příznaků jsou v sestupném kroku interpolovány na dvojnásobný rozměr, v nejjednodušším případě algoritmem k-nejbližších sousedů. Ke vzniklým mapám jsou po prvcích přičteny hodnoty z map páteřní sítě ze vzestupného kroku. Před přičtením je aplikována konvoluce filtrem 1×1 z důvodu redukce počtu kanálů. Interpolaci lze provádět i komplexnějšími způsoby, jako je max-unpooling či sofistikovaná transponovaná konvoluce [31] (obvykle matematicky nesprávně nazývaná dekonvolucí).

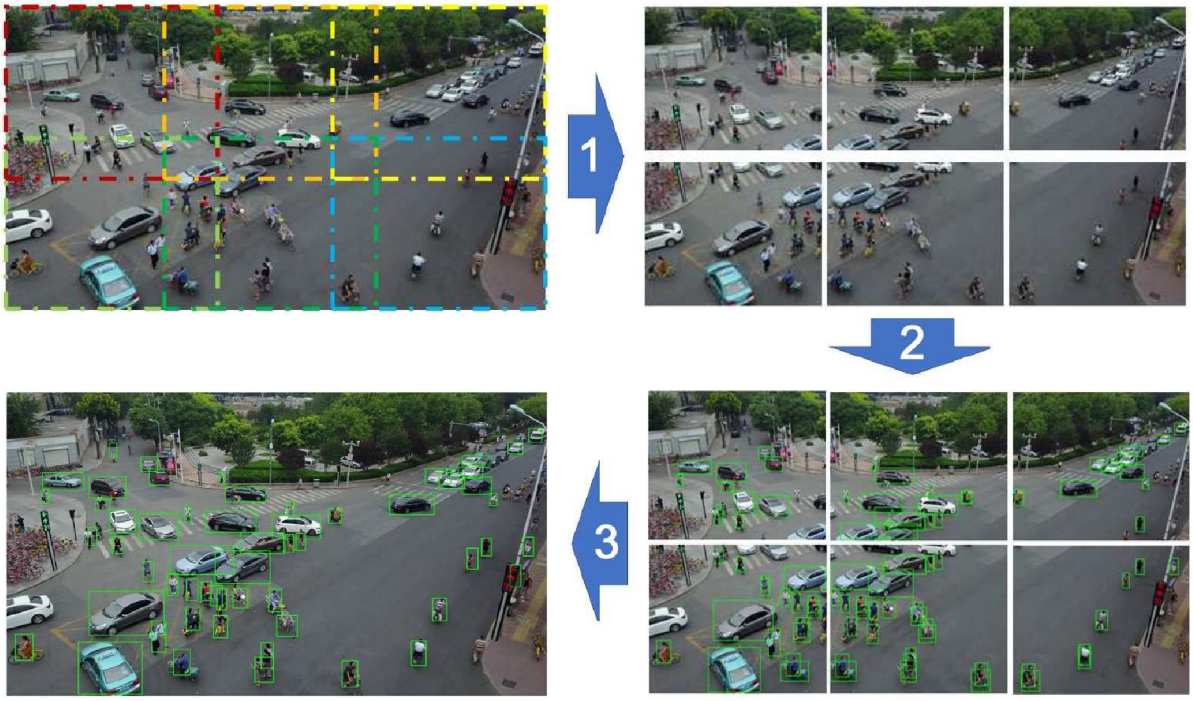


Obr. 2.13: Způsoby predikce z map příznaků. Převzato z [20].

Detekce po výřezech

Většina detektorů pracuje se čtvercovými vstupními obrázky o délce strany (224–608) px. Zmenšením původního obrázku se ztrácí schopnost detekce velice malých objektů. Moderní kamery o rozlišení 2K až 8K jsou schopny zaznamenat scénu s jemnými detaily vystihujícími malé objekty. Takto velké snímky nelze přímo analyzovat neuronovou sítí, paměťové nároky by přesahovaly stovky GB. Nabízí se snímky rozdělit na malé výřezy a ty analyzovat zvlášť.

Celý postup znázorňuje obr. 2.14. Nejprve se určí velikost výřezů podle rozměrů obrázku a překryvu. Překryv slouží ke spolehlivému zachycení objektů, které by jinak mohly být rozpůleny a opomenuty. Obrázek je rozdělen (krok 1). Poté jsou jednotlivé výřezy postupně předkládány síti a předpovídány rámečky (krok 2). Nakonec se všechny rámečky vloží do původního obrázku podle polohy výřezu a odstraní se duplicitní rámečky algoritmem *non-max suppression* (krok 3). Obsahuje-li obrázek i velké objekty, které by se do výřezů nemusely vejít, lze provést detekce i na původním zmenšeném obrázku. [32].



Obr. 2.14: Postup detekce po výřezech. Převzato z [32] a upraveno.

Analýza mnoha výřezů je výpočetně náročná, vymýšlí se proto způsoby, jak jejich počet omezit. V práci [33] zabývající se detekcí lidí metodou YOLO je obrázek z 4K videa nejprve prozkoumán na dvou zmenšených čtvercových výřezech (z důvodu širokoúhlého snímku). Vznikají hrubé detekce lidí, které je nutno upřesnit na výřezech s vyšším rozlišením. Stejnou sítí jsou analyzovány už jen ty části obrázku, do kterých zasahují předchozí detekce. Tento způsob je efektivní, pokud není zaplněna převážná část obrázku. Nevýhodou je, že první krok nemusí zachytit oblasti s osamocenými malými objekty. [33]

Extrémní situace nastává u satelitních snímků, které řeší metoda $\mathcal{R}^2$-CNN. Rozdělení více než 300megapixelových snímků na části je zde jediným východiskem. Malé objekty (např. letadla, lodě) se mohou vyskytovat kdekoli, zároveň je ale analýza celé plochy robustním detektorem v původním rozlišení nepřípustná. Metoda funguje následovně. Obrázek je rozdělen na výřezy jako obvykle. Jednoduchá, leč efektivní síť extrahuje příznaky z výřezů. Následuje tzv. *blok globální pozornosti*, který zvětší receptivní pole a získá širší kontextuální informace. Výsledek vstupuje do klasifikátoru, jenž posoudí, zda se ve výřezu může nacházet hledaný objekt. Pokud ano, předchozí výsledek je zpracován robustním detektorem na bázi Faster R-CNN, který přesně vyhledá objekty. Drtivá většina výřezů žádné objekty neobsahuje, proto prochází pouze nenáročnou částí procesu. Došlo tak k výraznému snížení výpočetní náročnosti. [34]

3 Analýza problému

Shrnutí rešeršní části

Na základě provedené rešerše vyplývá, že je metoda YOLO pro počítání malých objektů použitelná. Její slabina vzhledem k ostatním metodám – méně přesná lokalizace – nemá na určení počtu objektů zásadní vliv. První dvě verze YOLO ale zaostávají v přesnosti detekce malých objektů, tuto komplikaci řeší až třetí verze. Jelikož je v této práci využívána teprve druhá verze, bude nutné ji vhodně přizpůsobit. Jako vodítko poslouží jednak změny ve třetí verzi, jednak podstatná vylepšení v ostatních metodách. Mezi ně patří zmíněná pyramidová struktura pro extrakci jemnějších příznaků a detekce po výřezech pro zachování vysokého rozlišení a schopnost detekce na obrázcích zachycujících velký prostor. Je také potřeba zvolit typ objektů, na němž bude metoda testována.

Výběr typu objektů

Lidi – Počítat lidi v kompaktním davu nemá význam, protože to zvládají lépe hustotní mapy. Reálnější je detekovat lidi v menších skupinách na snímcích například z bezpečnostních kamer. Takový problém je však řešený širokým spektrem výzkumníků a nenabízí tolik prostoru pro nové objevy.

Stromy – Atraktivní úkol představují koruny stromů v leteckých snímcích. Dosavadní výzkum zabývající se detekcí jednotlivých stromů pomocí konvolučních sítí vykazuje slibné výsledky. Před započetím praktické části práce proběhly experimenty s datovou sadou kokosovníků. Ačkoliv nebyla data kvalitně anotována, přesnost detekce jevila jistý potenciál. Šlo však o jednodušší úlohu ve srovnání s detekcí stromů z leteckých snímků českých lesů. Anotace takových stromů vyžaduje obrovské množství času – na první pohled rozeznatelné stromy tvoří neurčité shluky a tvorba kvalitní datové sady by vyžadovala mnoho korekcí v terénu s nejistým výsledkem. Data by navíc byla stejného charakteru po stránce velikosti a uspořádání, hranice metody YOLO by se špatně posuzovaly. Proto bylo od tohoto typu objektů upuštěno.

Hospodářská zvířata – Mezi málo prozkoumané oblasti na poli detekce malých objektů patří detekce hospodářských zvířat. Na širokoúhlých snímcích z dronů zaujímají zvířata příhodně malou velikost. Vyskytují se jednak osamoceně, jednak v menších stádech. Zároveň mohou být částečně překryta vegetací. Všechny tyto aspekty tvoří rozumnou obtížnost pro testování metody YOLO. Jako řešený objekt byly zvoleny krávy, protože nabízí větší variabilitu z hlediska barev a tvarů než např. ovce.

Další postup

Díky charakteru detekovaných objektů se nabízí uměle rozšířit trénovací datovou sadu. K těmto účelům bude vytvořen syntetický generátor obrázků, který z několika snímků pozadí a výřezů krav vygeneruje obrázky stád spolu s tabulkou použitelnou přímo pro následné trénování sítě. Pro práci s obrázky o vysokém rozlišení je nutno napsat program, který je dokáže pro účely trénování a detekce patřičně zpracovat. Proběhnou experimenty s různými předtrénovanými sítěmi a úpravou architektury. Nejvhodnější kombinace bude vybrána pro finální detektor. Cílem detektoru není precizní lokalizace objektů, ale určení co nejpřesnějšího výsledného počtu objektů. Důraz bude kladen také na nižší výpočetní nároky, aby byl použitelný i na mobilních zařízeních.

4 Použité nástroje

4.1 Generátor syntetických obrázků

Nedílnou součástí tvorby každého modelu hlubokého učení je datová sada. Kvalita a rozsah datové sady výrazně ovlivňuje přesnost výsledného modelu. U způsobu učení s učitelem je nutno na datech označit správný výstup. Model se učí tento výstup replikovat. Pro detekci objektů nestačí označit obrázek jako celek, je potřeba nakreslit kolem každého objektu rámeček s příslušnou třídou (typem objektu). Sestavení kvalitní datové sady vyžaduje mnoho času i finančních prostředků. Obzvlášť to platí pro snímky malých objektů, v této práci snímky krav. Existující datové sady krav obsahují pouze sporadicky rozmístěné krávy podobného vzhledu a velikosti. [42]

Inspirací, jak tento problém vyřešit, je článek [43] zabývající se počítáním lidí v davu metodou z kapitoly 2.4.2. Ve videohře GTA V je vygenerováno obrovské množství snímků obrazovky různorodých scenérií, na kterých je následně síť trénována, poté dotrénována na menším počtu reálných obrázků. Autoři tak dosáhli výrazně lepších výsledků než předchozí práce. Pro účely této bakalářské práce byl vytvořen méně sofistikovaný, avšak stále efektivní, syntetický generátor.

Generátor je napsán v MATLABu R2019b pomocí funkcionálního programování. Základní myšlenka spočívá v tom, že jsou do obrázku pozadí automaticky vkládány výřezy objektů a ukládány polohy rámečků. Na výběr jsou dvě možnosti rozmístění objektů: náhodné a uspořádané – pro každé je vytvořen zvláštní skript. Jedná se o inženýrský nástroj, uživatel si může navolit různé parametry generování a vytvořit si tak datovou sadu na míru.

4.1.1 Příprava dat a parametry generování

Nejprve je nutno shromáždit dostatečné množství zdrojových obrázků formátu JPEG nebo PNG a roztřídit je do adresářů dle struktury na obr. 4.1. Prvním adresářem je *background,* jež obsahuje obrázky pozadí, volitelně masky značící bílou barvou překážky. Následuje adresář *classes* s adresáři pojmenovanými dle názvů tříd. V každém z nich se nachází výřezy a masky výřezů objektů příslušné třídy.

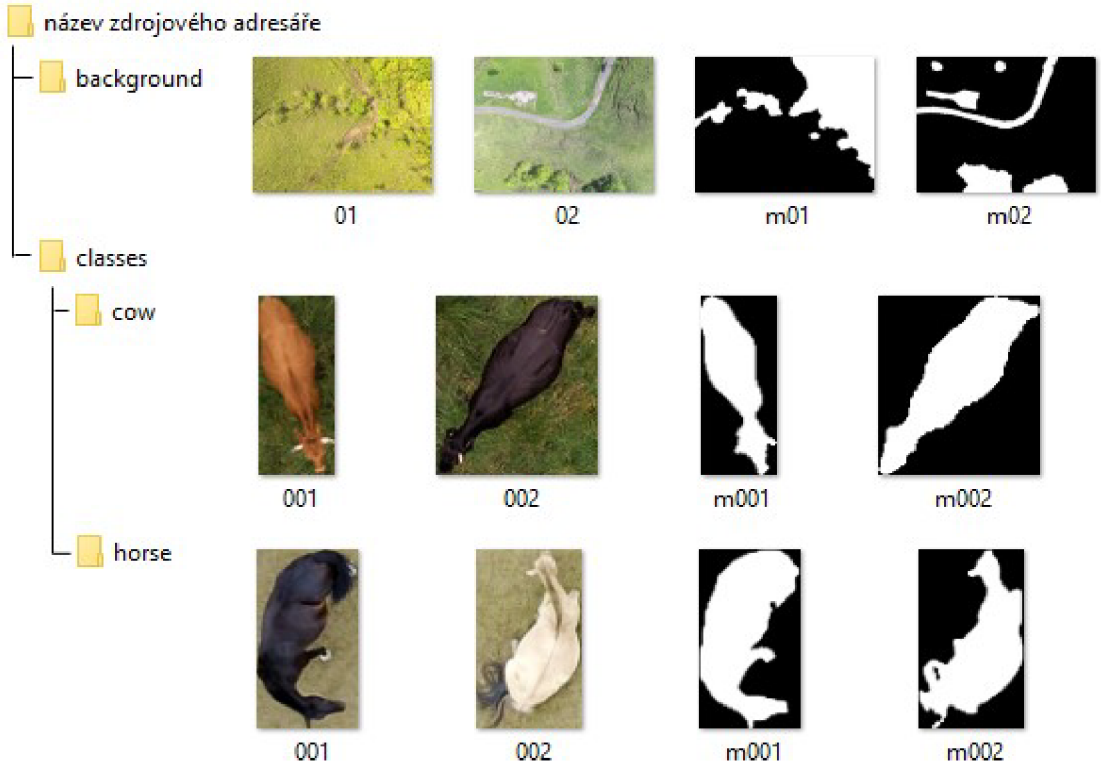
Jako pozadí se použijí 24bitové RGB obrázky ideálně takového pozadí, v kterém má detektor pracovat. V práci jsou využity pastviny focené dronem z datové sady v [43]. Výřezy objektů (krav) pochází z fotografií z [49]. K objektu je přidružena maska v 8 bitech stupňů šedi, kde bílá barva reprezentuje pixely příslušející objektu, černá ostatní. Šedá barva značí hrany objektu a pomáhá s přirozenějším prolnutím, není však nezbytná. Tvorba výřezů proběhla v programu Adobe Photoshop CS5, jelikož umožňuje snadný výběr objektů a zautomatizované uložení výřezu i masky. Lze použít i jiný program, např. MATLAB.

Na začátku obou skriptů generátoru je zvolena cesta k libovolně pojmenovanému zdrojovému adresáři a adresáři výstupnímu. Do výstupního adresáře jsou ukládány vygenerované obrázky a na závěr soubor typu *mat*. Ten obsahuje proměnnou pro trénování typu *table* s názvem `datasetLabels` obsahující informace o rámečcích.

Následuje výběr parametrů generování. Podrobnější význam jednotlivých parametrů je popsán v příslušném skriptu. Patří sem:

- počáteční index obrázku – lze rozšířit dříve vygenerovanou datovou sadu;
- počet a velikost vygenerovaných obrázků;
- interval počtu objektů z každé třídy na jeden obrázek;

- možnost výběru náhodného počtu objektů v logaritmickém prostoru – upřednostňuje menší počty ve zvoleném intervalu;
- možnost normalizace velikosti objektů – užitečné pro generaci objektů o stejné velikosti, i když se ve zdrojových obrázcích liší;
- možnost realistického měřítka objektů – škáluje velikost pozadí úměrně normalizované velikosti objektů;
- maximální procentuální překryv mezi objekty a pozadím v násobku plochy vkládaného objektu;
- změna rozměrů objektů či pozadí – slouží k variabilitě velikostí a deformací;
- rotace objektů;
- změna jasu, kontrastu, saturace a barvy;
- maximální počet neúspěšných pokusů o vložení objektu – zabraňuje opakovanému vkládání objektů do již zaplněného pozadí.



Obr. 4.1: Ukázka požadované struktury adresáře a pojmenování souborů.

4.1.2 Princip generátoru

Po spuštění skriptu se nejdříve sestaví seznam obrázků pozadí a výřezů objektů a jejich masek. Pro každý generovaný obrázek se vybere náhodně pozadí a provede augmentace – drobná změna původního snímku. Poté jsou náhodně voleny objekty ze seznamu dle žádané třídy a počtu, upraveny jejich rozměry a augmentovány. Proběhne prolnutí objektu s pozadím podle vybrané metody. Nakonec je zhotovena a uložena trénovací tabulka (formát viz. [40]).

Náhodné rozmístění

Ke generaci touto metodou je určený skript `mainRandomPlacement`. Metoda rozmísťuje objekty po pozadí náhodně, jak ukazuje obr. 4.2. Objekt je do pozadí vložen nejprve zkušebně a je ověřen jeho překryv s překážkami pozadí a dalšími objekty současně. Překryv je definován

jako průnik masky objektu s maskou existujících prvků. V případě splnění podmínky je objekt vložen natrvalo a celý proces se opakuje.

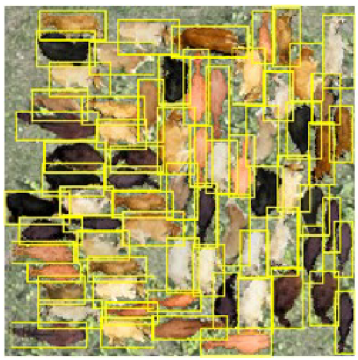
Uspořádané rozmístění

Metoda je součástí skriptu `mainHerdPlacement`. Je vyvinuta speciálně pro rozmístění krav a má simulovat situaci, kdy jsou ve stádu namačkané na sobě. V takovém případě bývají přibližně stejně orientovány, jejich osy (podélné osy těla) jsou rovnoběžné. Osa krávy je určena jako osa minimálního momentu setrvačnosti masky krávy.

Proces vkládání probíhá následovně. Prvních několik krav je rozmístěno stejným způsobem jako náhodným rozmístěním. Jejich počet spolu s některými parametry generování se nastavuje ve funkci `mergeImagesHerd`. Pro každou další vkládanou krávu je napřed vybrána referenční kráva a strana, na kterou bude umístěna. Naleznou se nejvzdálenější body obou krav od jejich os na správných stranách. Umístění vkládané krávy, při kterém se oba body překrývají, je považováno za výchozí. Následuje hledání nejvhodnější polohy. Vkládaná kráva je zkušebně několikrát posouvána kolem výchozí polohy v podélném a příčném směru, každé poloze je uděleno skóre na základě níže vysvětleného kritéria. Kráva je nakonec vložena do polohy s nejlepším skóre. Pokud žádná z poloh není validní (kvůli překryvu s ostatními prvky či okraji obrázku), hledá se ještě několik poloh s tím, že první validní je zvolena. V případě neúspěchu je referenční krávě a zvolené straně inkrementována hodnota neúspěchu. Eliminuje to příliš mnoho opakovaných pokusů o vkládání vedle referenční krávy, která kolem sebe již nemá dost místa. Příklad vygenerovaného obrázku zachycuje obr. 4.3.

Obr. 4.2: Náhodné rozmístění.

Obr. 4.3: Uspořádané rozmístění

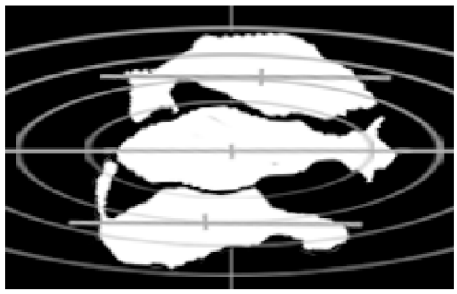


Obr. 4.4: Zobrazení rámečků (ground truth).

Opodstatnění zvoleného kritéria ilustruje obr. 4.5. Uprostřed se nachází referenční kráva. Každá elipsa značí množinu těžišť vkládané krávy se stejnou vhodností. Po různém umístění vkládaných krav lze pozorovat, že se vizuálně nejvhodnější polohy shodují se zvoleným kritériem.

Hodnota kritéria se určí jako velikost hlavní poloosy elipsy vyhovující následujícím podmínkám:

- její střed leží v těžišti referenční krávy;
- má hlavní poloosu rovnoběžnou s osou referenční krávy;
- poměr hlavní a vedlejší poloosy je shodný s poměrem délky a šířky ref. krávy;
- prochází těžištěm vkládané krávy.



Obr. 4.5: Vizualizace kritéria vhodnosti.

4.1.3 Přednosti a nedostatky

Běžně se trénovací data rozšiřují na základě prosté augmentace. Jedná se o globální úpravy obrázku, nemění se prostorové uspořádání objektů. Generátor naopak vytváří nepřeberné množství kombinací rozmístění a díky tomu efektivněji rozšiřuje datovou sadu. Navíc lze umístit objekty na různé podklady a ve větším množství, než by bylo možné získat v realitě. Lze vygenerovat více snímků v takovém uspořádání, s jakým má detektor potíže. Odpadá potřeba označovat mnoho komplikovaných snímků. Příklad takového snímku ilustruje obr. 4.4. Generátor je použitelný na jakékoliv objekty snímané z pohledu shora, ne jenom krávy. Aplikovat se tak dá např. na detekci vozidel, lidí, jiných zvířat, staveb atd.

Slabou stránkou generátoru je rychlost generace, převážně u uspořádaného rozmístění. Zpomalení se projevuje se zvyšující velikostí výsledných obrázků, jednak kvůli výpočtům s většími maticemi, jednak kvůli rostoucímu počtu objektů na jednom snímku. Výpočty by bylo vhodné do budoucna zefektivnit, případně uplatnit grafickou kartu. Další komplikaci přináší objekty s velikou variabilitou. Ke spolehlivé interpretaci by bylo nutno použít velký počet zdrojových obrázků, díky generátoru se síť naučí především hledat obecné rysy v nejrůznějších polohách. Obecně poskytuje spíše kvalitní základ pro trénování před dotrénováním na menším množství reálných snímků.

4.2 Zpracování obrázků o vysokém rozlišení

Jak bylo rozebráno v kapitole 2.4.4, obrázky o vysokém rozlišení je vhodné rozdělit na překrývající se části a detekci provádět na každé zvlášť. Aby byl detektor co nejpřesnější, je rozumné ho trénovat na obrázcích o stejném rozlišení. Získaná datová sada pro praktickou část obsahuje obrázky o vysokém rozlišení různých velikostí, je ji tedy nutno nejprve zpracovat. Proto byly napsány skripty, jež toto zpracování provedou.

4.2.1 Zpracování datové sady

Úkol provádí skript `mainPreprocessDataset`. Napřed uživatel definuje cestu k trénovací tabulce a úložnému adresáři, velikost a překryv výřezů, faktor zmenšení původního obrázku. Vše podstatné je předáno funkci `preprocessDataset`, která zpracovává po jednom všechny obrázky. Počet výřezů N_x v horizontálním směru se určí ze vztahu (4.1):

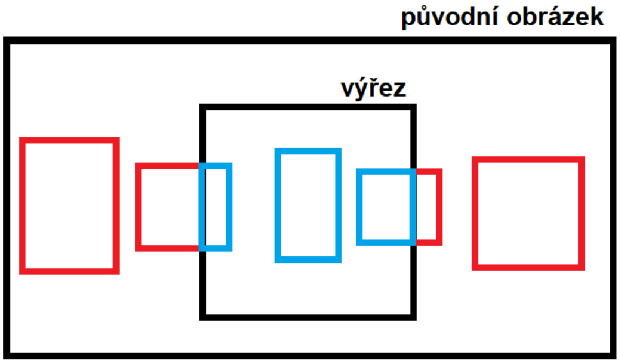
$$N_x = round\left(\frac{x - p_x}{w - p_x}\right), \tag{4.1}$$

kde x značí šířku původního obrázku, p_x horizontální překryv a w šířku výřezu. Kvůli zaokrouhlení na celočíselný počet výřezů je nutno zpětně dopočíst upravený rozměr obrázku x' ze vztahu (4.2), potom obrázku i rámečkům změnit velikost.

$$x' = N_x \cdot w - (N_x - 1) \cdot p_x \tag{4.2}$$

Analogické vztahy platí i pro vertikální směr. Dále se určí souřadnice rohů jednotlivých výřezů. Je třeba přepočíst globální souřadnice rámečků do lokálních souřadnic každého výřezu. Rámeček přitom může zaujmout jednu z 25 poloh. Možné polohy v jednom směru zobrazuje obr. 4.6, ve výřezu jsou zahrnuty pouze modré části rámečku. Rámeček může ležet buď vně výřezu, částečně uvnitř, nebo zcela uvnitř. Leží-li na okraji výřezu, pak je zaznamenán pouze v případě, že a) plocha vnitřní části rámečku přesahuje určitou část jeho celkové plochy, b) menší z rozměrů vnitřní části přesahuje určitý počet pixelů. Uvedené podmínky zaručují, že nebude anotována příliš malá část objektu, která je špatně rozpoznatelná. Obě prahové hodnoty lze změnit na začátku funkce. Nakonec funkce uloží výřezy s alespoň jedním objektem (MATLAB netrénuje na prázdných snímcích) do adresáře a vrátí novou trénovací tabulku.

Velikost výřezů má být obecně volena větší než největší objekt, jinak by nemusel být rámeček zaznamenán. Překryv je rozumné nastavit podobně veliký jako největší objekt, hlavně během detekce [34]. S příliš malým překryvem by nemusel být detekován celý objekt, příliš malý překryv zase vede k duplicitním detekcím.



Obr. 4.6: Ukázka možných poloh rámečku vůči výřezu v jednom směru.

4.2.2 Detekce a počítání objektů na velkém obrázku

Nativní funkce[11] `detect` mění před detekcí velikost vstupního obrázku na nejbližší velikost použitou při trénování. Razantní zmenšení velkých obrázků je nepřijatelné, proto byla vytvořena funkce nová, `detectLargeImage`, jež původní funkci rozšiřuje. Navenek se chová stejně, vstupem je použitý detektor, detekovaný obrázek, práh detekce, navíc parametry výřezů jako u zpracování datové sady. Vrací parametry rámečků, skóre jistoty a kategorie tříd.

Struktura funkce se skládá ze tří částí. V první části nastane rozdělení obrázku na výřezy funkcí `preprocessImage`, obdobně jako u zpracování datové sady, ale s tím rozdílem, že se nepřepočítávají žádné rámečky. Funkce vrací výřezy ve čtyřdimenzionální matici. Druhou část tvoří for cyklus, v němž jsou prováděny detekce na jednotlivých výřezech původní funkcí `detect`. V poslední části proběhne přepočet parametrů rámečků do souřadného systému původního obrázku funkcí `postprocessImage`.

Funkce tvoří hlavní část skriptu `mainDetectLargeImage`, kde s použitím detektoru provede detekce. Ty jsou ve skriptu následně vyfiltrovány algoritmem non-max suppression. Na základě počtu zbylých rámečků je určen počet objektů na obrázku.

[11] Ve skutečnosti se nejedná o funkci, ale o metodu objektu *yolov2ObjectDetector*

Nejdůležitější částí celého procesu je právě zmíněný detektor, tedy model s konkrétními parametry. Aby došlo ke spolehlivým detekcím, je nutno navrhnout vhodnou architekturu modelu a natrénovat ho na reprezentativních datech. To je náplní kapitoly 5.1.

Jedním z cílů práce bylo vytvořit nástroj na bázi metody YOLO, jež spočítá objekty zvoleného typu nacházející se na obrázku. K tomu slouží skript `mainDetectLargeImage`. Hlavní část skriptu tvoří funkce `detectLargeImage`, blíže popsaná v kapitole 4.2.2. Funkce rozdělí v případě potřeby obrázek na menší výřezy, na nichž provede detekce a zpětně je sloučí do původního obrázku. Počet objektů přítomných na obrázku se určí na základě počtu rámečků značících detekce.

4.3 Existující nástroje

4.3.1 Image Labeler

K manuálním anotacím skutečných obrázků byla využita nativní aplikace Image Labeler, jež je součástí Computer Vision Toolbox 9.1. Umožňuje interaktivně označit objekty různých tříd, po dokončení exportuje do workspace trénovací tabulku. V některých případech ji exportuje v chybném tvaru, funkcí `i_correctImageLabelerLabels` ji lze opravit.

4.3.2 Deep Network Designer

Aplikace Deep Network Designer z Deep Learning Toolbox 13.0 slouží ke grafickému návrhu a úpravám neuronových sítí. Její přínos je patrný převážně při práci s komplexní architekturou, jejíž tvorba pouze programovou cestou by se stávala nepřehlednou. Hotovou síť lze do workspace exportovat jako proměnnou typu *Layer*, případně formou živého skriptu. Ve verzi R2020a došlo k výrazným rozšířením aplikace, probíhá zde např. celé trénování sítě. V rámci této práce však tato nejnovější verze využita nebyla.

5 Příprava detektoru

5.1 Datové sady

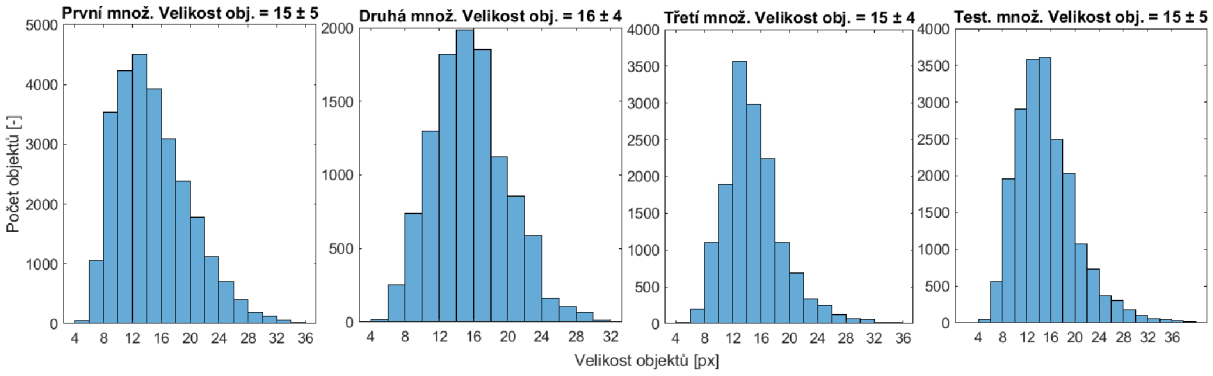
5.1.1 Japonská datová sada

Datová sada z článku [42] je využita v této práci k vyhodnocení přesnosti detekce metody YOLO na různě malých objektech. Několik obrázků také slouží jako snímky pozadí v syntetickém generátoru. Obrázky v sadě byly získány z dronu a zachycují sporadicky rozmístěné krávy na japonských pastvinách. Sadu tvoří přes 600 obrázků pro trénování a testování z jedné pastviny, součástí je i dodatečných 14 testovacích obrázků z jiné pastviny fotografovaných v ostřejším slunci. Všechny obrázky datové sady mají vysoké rozlišení 12 Mpx, krávy jsou na nich tvořeny velkým počtem pixelů. Autoři se věnovali v článku mírně odlišnému problému, než kterým se zabývá tato práce, výsledky s ní proto nelze přímo porovnávat. Na dodatečných testovacích obrázcích ovšem dosáhli vyšší chybovosti než na zbylých snímcích, což se projevuje i v této práci.

Za účelem testování, jak malé objekty vzhledem k počtu pixelů je YOLO schopno rozeznat, bylo vytvořeno několik datových sad o měřítkách 0,50 a méně. Měřítko udává faktor, kterým byl výchozí obrázek zmenšen. Bylo tak získáno několik datových sad o různých velikostech objektů. K tvorbě datových sad byl využit nástroj pro zpracování datové sady z kapitoly 4.2.1, který rozděluje výchozí obrázky na výřezy. Pro všechny sady byla zvolena velikost výřezů 224 px a přesah 35 px.

5.1.2 Skutečná datová sada

Skutečná datová sada slouží v této práci převážně k dotrénování finálního detektoru. Obsahuje 10 fotografií o různém rozlišení čítajících celkem 1043 krav v odlišných úhlech pohledu, uspořádání a prostření. V původním rozlišení mají krávy průměrnou velikost (36 ± 10) px. Velikost označuje geometrický průměr šířky a výšky rámečku. Obrázky byly rozděleny do 4 množin: tří pro křížovou validaci a jedné pro kontrolní vyhodnocení přesnosti detekce (testovací). Jelikož se obrázky liší obtížností (některé obsahují překrývající se krávy, některé osamocené), byly rozděleny pokud možno rovnoměrně. Z původních obrázků bylo pro každou množinu vytvořeno nástrojem pro zpracování datové sady cca 500 výřezů o délce strany 224 px a různém překryvu a měřítku. Bylo tak dosaženo větší variability velikostí a poloh objektů ve výřezech. Obecně byly obrázky zmenšeny, histogramy velikostí rámečků v jednotlivých množinách znázorňuje obr. 5.1.



Obr. 5.1: Histogramy velikostí krav ve výřezech ze skutečných obrázků.

5.1.3 Syntetická datová sada

Syntetická datová sada ověřuje užitečnost představeného generátoru syntetických obrázků a tvoří základ pro trénování finálního detektoru. Celkem byly vygenerovány 4 datové sady (viz tab. 5.1). První typ sady obsahuje velmi zaplněné obrázky s jednotným měřítkem pozadí a různou velikostí krav. Druhý typ tvoří různě zaplněné obrázky, kde měřítko pozadí odpovídá velikosti krav, což představuje více realistické snímky. Každý druh je rozdělen na dvě sady dle způsobu rozmístění krav: náhodně a uspořádaně.

Tab. 5.1: Přehled syntetických datových sad

typ sady	rozmístění objektů	počet dat		velikost objektů [px]
		trénovací	testovací	
bez měřítka	náhodně	3000	300	24 ± 12
	uspořádaně	3000	300	23 ± 12
s měřítkem	náhodně	3000	300	15 ± 7
	uspořádaně	3000	300	15 ± 7

5.2 Návrh architektury sítě

5.2.1 Výběr předtrénované sítě

Kostru výsledné navržené sítě tvoří páteřní síť – síť natrénovaná na klasifikační úloze, jež slouží k extrakci příznaků. Při výběru vhodné páteřní sítě bylo dbáno na to, aby neobsahovala příliš mnoho vrstev a parametrů a nezpomalovala tak chod výsledného detektoru. Do užšího výběru se dostaly následující sítě: ResNet18 [19], ResNet50 [19], TinyYOLOv2 [15], TinyYOLOv3 [16], MobileNetV2 [40].

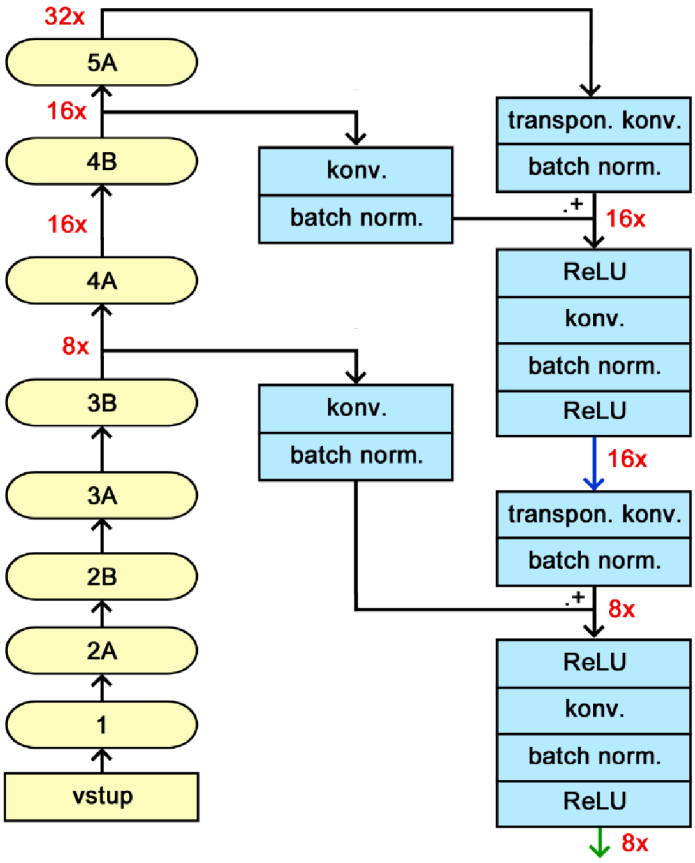
Po několika experimentech byl zvolen model ResNet18. Ostatní modely vykazovaly srovnatelné či pouze nepatrně lepší výsledky, přestože svou komplexností zvolený model převyšovaly. Struktura ResNet18 je členěna do reziduálních bloků označených jako 1, 2A, 2B, 3A, …, 5B. Míra podvzorkování výstupní vrstvy každého bloku se určí jako $2^{\text{číslo bloku}}$. Model tedy podvzorkovává 32×. Bylo zjištěno, že je možno odstranit blok 5B, aniž by došlo k poklesu přesnosti detekce. Rychlost modelu přitom výrazně vzrostla. Lze to odůvodnit tím, že poslední blok obsahuje velice specifické významové informace, které nejsou pro řešený problém (krávy v pohledu shora) příliš přínosné. Jako páteřní síť byl tedy vybrán model ResNet18 po blok 5A.

5.2.2 Rozšíření pro detekci

Vysoká míra podvzorkování páteřní sítě vede k příliš hrubé výstupní mřížce, jejíž buňky jsou větší než detekované objekty. K překonání této komplikace byla využita myšlenka pyramidové struktury rozebrané v kapitole 2.4.4. Schéma navržené architektury sítě ilustruje obr. 5.2. Základ tvoří síť ResNet18 po blok 5A – ResNet18_32x, značen také ResNet18_5a (zvýrazněno žlutě). Za ním následuje vrstva transponované konvoluce, která 2× zvětší mapu příznaků díky nulové výplni a kroku 2, a normalizační vrstva (*batch normalization*). Výstup bloku 4B je zpracován konvoluční a normalizační vrstvou, poté jsou obě mapy příznaků po prvcích sečteny a vstupují do nelineární funkce (ReLU). Následuje typický blok konvolučních sítí – konvoluční a normalizační vrstva zakončené ReLU. Ve všech konvolučních vrstvách je 128 filtrů

o velikosti 3 × 3. Dosavadní část (po modrou šipku) tvoří první verzi modelu pro testování – ResNet18_16x – označenou podle míry podvzorkování (červeně). Celý proces přidání vrstev je ještě opakován, vzniká tak druhá verze modelu – ResNet18_8x (po zelenou šipku).

Za modelem, ať už se jedná o kteroukoliv verzi, následují 2 typické konvoluční bloky určené pro naučení příznaků podstatných pro detekci. Konvoluční sítě v nich obsažené jsou tvořeny 128 filtry o velikosti 3 × 3 a 1 × 1 s krokem 1 a nulovou výplní. Za bloky je umístěna samostatná konvoluční vrstva s filtry o velikosti 1 × 1, jež produkuje mapu příznaků reprezentující typické parametry metody YOLOv2. Počet filtrů této vrstvy byl diskutován v kapitole 2.3.2, záleží na počtu kotevních rámečků. Model je zakončen pomocnou *yolov2TransformLayer* a výstupní *yolov2OutputLayer* vrstvou.



Obr. 5.2: Schéma navržené architektury.

Parametrem výstupní vrstvy jsou velikosti kotevních rámečků. Ty lze zadat ručně na základě odhadu, v lepším případě určitým algoritmem. V práci je využíván algoritmus *k-means clustering* dostupný v MATLABu pod funkcí `estimateAnchorBoxes`. Ve všech experimentech jsou zvoleny vždy 4 kotevní rámečky.

Váhy přidaných konvolučních vrstev byly inicializovány hodnotami z normálního rozdělení s rozptylem o hodnotě 10^{-4} – v MATLABu označeno jako možnost *narrow normal*. S aktivačními funkcemi ReLU se doporučuje inicializace vah metodou He [2]. Ta však vedla k obtížnějšímu trénování a zlepšení nepřinesla.

5.3 Vyhodnocení přesnosti detekce

5.3.1 Metrika Average Precision

Přesnost detekce objektů se vyhodnocuje obvykle metrikou *average precision* (*AP*) [-]. *AP* udává plochu pod *precision-recall* křivkou. Precision (přesnost) [-] vyjadřuje, jaká část výsledných rámečků označuje správný objekt, matematicky psáno vztahem (5.1).

$$precision = \frac{TP}{TP + FP} \tag{5.1}$$

Recall (výtěžnost) [-] vyjadřuje, jaká část všech přítomných objektů byla úspěšně detekována, matematicky psáno vztahem (5.2).

$$recall = \frac{TP}{TP + FN} \tag{5.2}$$

Ve vztazích (5.1) a (5.2) značí TP správný rámeček (skutečně pozitivní), FP nesprávný rámeček (falešně pozitivní), FN nesprávně vynechaný rámeček (falešně negativní). [4] Za správný rámeček se považuje ten, který dosahuje s ground-truth větší hodnoty IoU, než je prahová hodnota. Právě zvolená prahová hodnota rozděluje AP na několik kategorií. V rámci této práce je použita *AP* pro IoU větší než 0,5.

Precision/recall křivka vzniká následovně. Podle skóre jistoty je vytvořen sestupný seznam všech detekcí na vyhodnocované datové sadě. Jsou určeny precision a recall na základě první detekce v seznamu. K první detekci je přidána druhá detekce a obě veličiny jsou vyhodnoceny znovu. Opakovaně jsou zohledňovány další detekce, dokud není obsažen celý seznam. Dvojice precision, recall ze všech případů jsou vyneseny do výsledné křivky. Ta má obecně zubatý tvar, způsobený skokovou změnou precision při přidání chybné detekce ze seznamu. S přibývajícími správnými detekcemi rostou spojitě hodnoty obou parametrů, proto zaoblený nárůst. Ideální detektor dosahuje hodnoty *AP* rovné 1. Příklad výsledné křivky zachycuje obr. 6.10. Při vyhodnocení detekce více tříd jsou obvykle zprůměrovány hodnoty AP jednotlivých tříd, metrika se pak nazývá *mean Average Precision* (*mAP*). [44]

5.3.2 K-násobná křížová validace

Hotový model s natrénovanými parametry je nutno před praktickým použitím evaluovat. Nejčastěji se datová sada rozděluje do tří množin: trénovací, validační a testovací [4] [35]. Přesnost na validační množině se během trénování vyhodnocuje průběžně a slouží jako ukazatel, kdy má být trénování ukončeno. Následně je finální detektor evaluován na testovací množině. Je-li datová sada příliš malá, není buď testovací množina dostatečně reprezentativní, nebo vyžaduje velkou část dostupných dat, které pak nemohou být použita pro trénování [45].

Lepší proceduru představuje k-násobná křížová validace. Datová sada je rozdělena rovnoměrně do *k* množin. Jedna z množin slouží k validaci, zbylé množiny ke trénování. Po natrénování je detektor evaluován na zvolené validační množině, následně smazán. Poté se použije k validaci jiná množina, proběhne opět trénování a vyhodnocení. Proces se opakuje tak dlouho, dokud se nevystřídají pro validaci všechny množiny. Zprůměrováním hodnot přesnosti je získán odhad a rozptyl výsledné přesnosti detekce. Parametr *k* je volen obvykle v intervalu $\langle 5; 10 \rangle$. Vyšší hodnoty dávají výsledky s větším rozptylem a menší systematickou chybou (biasem) než nižší hodnoty. [45]

6 Experimenty

Následující experimenty slouží k nalezení limitů metody YOLO. Rovněž ověřují užitečnost představených nástrojů a modifikací předtrénované sítě. Každý experiment je zaměřen na odlišný aspekt, vzájemně se však prolínají.

6.1 Velikost objektů

První experiment měl za cíl nalézt hraniční velikost objektů, při které dosahuje YOLO ještě přijatelné přesnosti detekce. Srovnával také vliv podvzorkování sítě na schopnost detekce malých objektů. Jednotlivé detektory byly trénovány na snímcích z japonské datové sady. Pokud začala validační AP klesat, byl uvažován výsledek modelu ještě před dosažením maximálního počtu iterací. Parametry trénování, jež vedly k nejrychlejší konvergenci ztrátové funkce, jsou v tab. 6.1.

Tab. 6.1: Parametry trénování v prvním experimentu

model	minibatch	maximální počet iterací	počáteční krok učení	koeficient snížení kroku učení	Počet iterací pro snížení kroku učení
ResNet18_32x	40	1200	$5 \cdot 10^{-3}$	0,65	200
ResNet18_8x	40	1800	$1 \cdot 10^{-3}$	0,90	200

Trénování proběhlo na snímcích z první pastviny odděleně pro 6 různých měřítek výřezů. Průměrná velikost rámečků ohraničujících krávy a počet výřezů pro každé měřítko jsou uvedeny v tab. 6.2. Sady pro jednotlivá měřítka jsou nevyvážené počtem výřezů, snímky s menším měřítkem totiž zachycují větší prostor. Na druhou stranu obsahují více krav na jeden snímek, počtem objektů jsou proto srovnatelné a trénovaná síť má stále dost informací pro učení. Tato hypotéza byla ověřena při trénování s měřítkem 0,50 se stejným počtem snímků, jako u měřítka 0,20. *AP* klesla pouze o cca 0,10; menší počet výřezů proto neměl zásadní vliv na hodnoty *AP* porovnávané v tomto experimentu.

Tab. 6.2: Výsledky křížové validace v prvním experimentu

první pastvina			ResNet18_32x	ResNet18_8x
měřítko	počet výřezů	velikost rámečků [px]	AP [−]	AP [−]
0,50	1975	39 ± 8	0,90 ± 0,01	0,94 ± 0,01
0,33	1502	27 ± 5	0,80 ± 0,02	0,94 ± 0,03
0,25	1213	21 ± 4	0,63 ± 0,04	0,93 ± 0,04
0,20	982	17 ± 3	0,43 ± 0,06	0,91 ± 0,04
0,14	714	13 ± 2	0,15 ± 0,04	0,86 ± 0,06
0,09	442	8 ± 1	0,01 ± 0,01	0,59 ± 0,04

Tab. 6.3: Výsledky na testovacích snímcích

druhá pastvina			ResNet18_32x	ResNet18_8x
měřítko	počet výřezů	velikost rámečků [px]	AP [−]	AP [−]
0,50	60	35 ± 7	0,61 ± 0,01	0,84 ± 0,03
0,33	49	24 ± 4	0,41 ± 0,11	0,78 ± 0,05
0,25	48	19 ± 3	0,21 ± 0,10	0,62 ± 0,02
0,20	34	15 ± 3	0,13 ± 0,02	0,47 ± 0,08
0,14	25	12 ± 2	0,02 ± 0,02	0,44 ± 0,03
0,09	18	7 ± 1	0,00 ± 0,01	0,15 ± 0,01

K vyhodnocení přesnosti pomocí metriky *AP* byla vzhledem k časové náročnosti experimentu aplikována pouze 3násobná křížová validace. tab. 6.2 znázorňuje průměrnou hodnotu *AP* z křížové validace pro dva testované modely o odlišné míře podvzorkování. Po natrénování na každé validační množině byla změřena *AP* na testovacích snímcích z druhé pastviny. Její průměrné hodnoty spolu s parametry testovacích snímků jsou zaznamenány v tab. 6.3.

Naměřené hodnoty *AP* jsou vyneseny do grafu na obr. 6.1, pro přehlednost proloženy. Z grafu jsou patrné následující poznatky:

- **Jemnější mřížka zlepšuje detekci malých objektů**

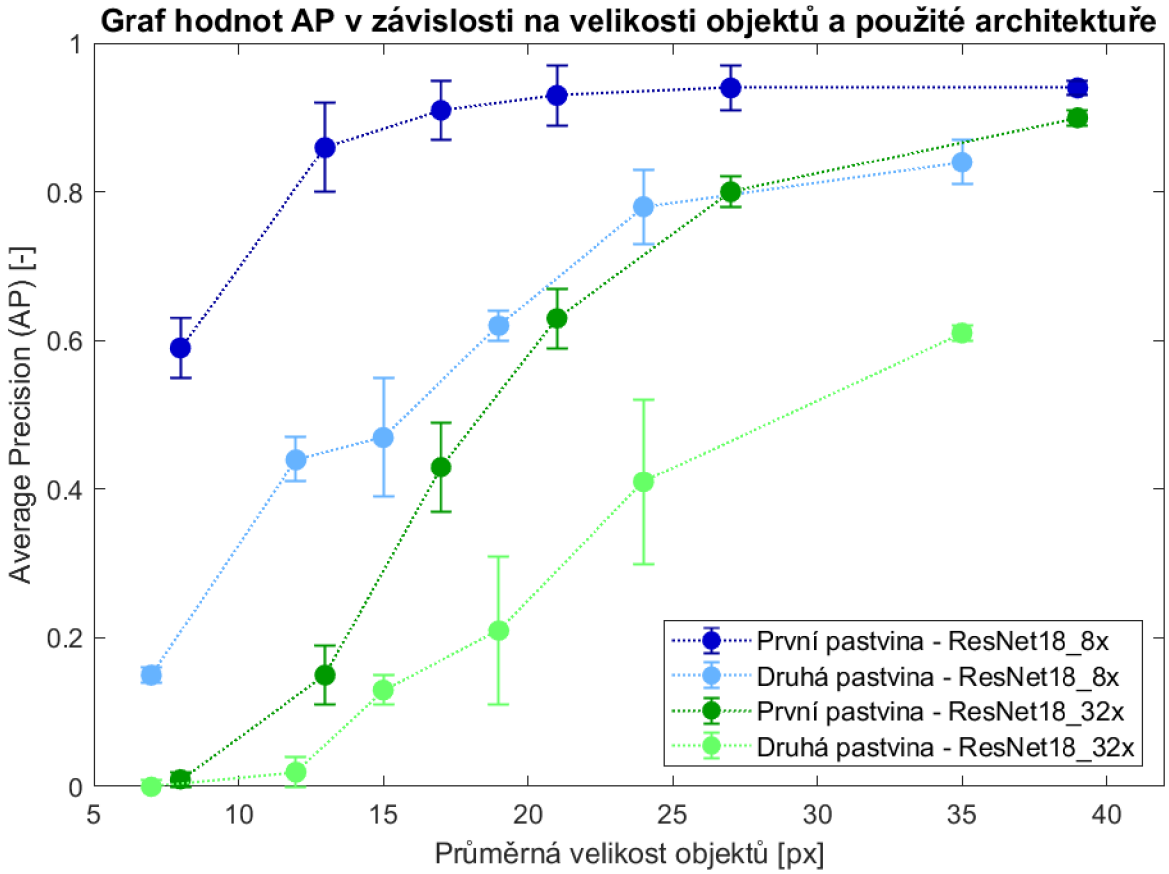
  Model o původní míře podvzorkování (tmavě zelená) dosahuje přijatelné *AP* kolem 0,9 pro objekty větší než cca 32 px. S klesající velikostí objektů klesá rychle i *AP* – pro velikost 16 px dosahuje nepoužitelných výsledků a objekty menší než asi 10 px nedokáže detekovat vůbec.

  Upravený model se 4× menší mírou podvzorkování (tmavě modrá), neboli 16× větším počtem buněk ve výsledné mřížce, je schopen rozeznat výrazně menší objekty. Hodnota *AP* začíná klesat až u objektů menších než cca 16 px, dokonce i 8pixelové objekty dokáže model detekovat s překvapivou přesností.

- **Pro dostatečnou robustnost detektoru je potřebné větší rozlišení**

  Na testovacích snímcích z druhé pastviny, jež jsou svým charakterem mírně odlišné od snímků z první pastviny, dosahuje *AP* obecně nižších hodnot. *AP* na nich začíná s klesající velikostí objektů padat dříve než na snímcích z první pastviny. Při malém rozlišení se detektor sice objekty stejného charakteru rozpoznávat naučí, větší odchylky ovšem nedokáže podchytit.

  Projevuje se to zejména u modelu ResNet18_32x natrénovaném na objektech o velikosti kolem 39 px. I přes vysokou přesnost na snímcích z první pastviny vykazuje detektor výrazně horší přesnost na snímcích z druhé pastviny. Naopak model ResNet18_32x si při té stejné velikosti objektů udržuje přesnost na snímcích z druhé pastviny velice dobrou. S klesající velikostí se už přesnosti také rozbíhají výrazněji.



Obr. 6.1: Přesnost detekce dle velikosti objektů a zvolené architektury.

6.2 Pyramidová struktura

Druhý experiment zkoumal vliv výsledného podvzorkování na přesnost detekce a ověřoval účinnost pyramidové struktury. Probíhal na reálné datové sadě rozdělné do 3 množin pro křížovou validaci (I, II, III) a jedné testovací. Natrénováno a porovnáno bylo 5 verzí modelu ResNet18 s pyramidou strukturou nebo bez ní. Jejich názvy a parametry trénování každého modelu znázorňuje tab. 6.4. Architektura každého modelu je patrná z obr. 5.2.

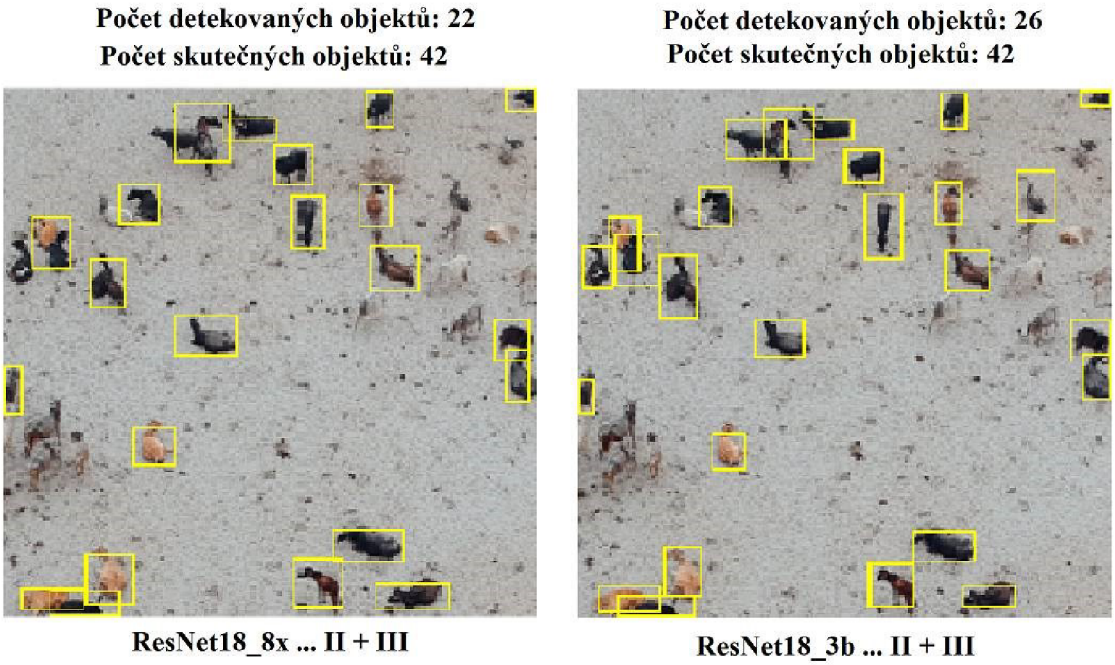
Výsledné hodnoty *AP* po každé části křížové validace jsou zaznamenány v tab. 6.5. V záhlaví tabulky je uvedeno, na jakých množinách byl detektor trénován. Testován byl následně na zbylé validační množině a pro ověření odhadu *AP* také na testovací množině.

Tab. 6.4: Parametry trénování v druhém experimentu

model	podvzorkování	maximální počet iterací	počáteční krok učení	koeficient snížení kroku učení	počet iterací pro snížení kroku učení
ResNet18_8x	8×	600	$5 \cdot 10^{-3}$	0,5	200
ResNet18_16x	16×	1200	$5 \cdot 10^{-3}$	0,7	300
ResNet18_5a	32×	600	$5 \cdot 10^{-3}$	0,5	200
ResNet18_4b	16×	1200	$5 \cdot 10^{-3}$	0,7	300
ResNet18_3b	8×	600	$5 \cdot 10^{-3}$	0,5	200

Tab. 6.5: Výsledky křížové validace v druhém experimentu

model	*AP* na validační množině [-]				*AP* na testovací množině [-]			
	I + II	I + III	II + III	průměr	I + II	I + III	II + III	průměr
ResNet18_8x	0,35	0,56	0,31	0,41 ± 0,13	0,33	0,34	0,37	0,35 ± 0,02
ResNet18_16x	0,10	0,14	0,12	0,12 ± 0,02	0,12	0,12	0,15	0,13 ± 0,02
ResNet18_5a	0,00	0,00	0,00	0,00 ± 0,00	0,00	0,00	0,00	0,00 ± 0,00
ResNet18_4b	0,13	0,19	0,13	0,15 ± 0,03	0,14	0,14	0,18	0,15 ± 0,02
ResNet18_3b	0,41	0,50	0,33	0,41 ± 0,09	0,34	0,33	0,38	0,35 ± 0,03



Obr. 6.2: Detekce na obrázku z testovací množiny. Zdrojový obrázek z [50].

Z hodnot *AP* v *tab. 6.5* vyplývají následující zjištění:

- **Jemnější mřížka výrazně zlepšuje detekci malých objektů**

  Potvrzuje se závěr z předchozího experimentu, že míra výsledného podvzorkování má zásadní vliv na schopnost detekce malých objektů. Rozdíly v přesnosti detekce v závislosti na podvzorkování sítě jsou na různorodějších a komplikovanějších datech, oproti japonské datové sadě, výraznější.

- **Pyramidová struktura nemusí přinést zlepšení**

  Model s pyramidovou strukturou na některých množinách detekci oproti modelu bez ní zpřesnil, na jiných zhoršil. Je možné, že u zvoleného typu objektů vznikají detekce spíše na základě příznaků z nižších vrstev nežli ucelenějších významových vzorů. Také se můžou příznaky špatně přenášet přes vrstvy transponované konvoluce. Bylo by vhodné vyzkoušet jiné způsoby zvětšení mapy příznaků, ty jsou však dostupné až od vyšší verze MATLABu.

  Podobnost úspěšnosti detekce ilustruje obr. 6.2 zachycující detekce na snímku z testovací množiny. Zobrazuje výsledky detektorů o 8násobném podvzorkování s pyramidovou strukturou (ResNet18_8x, vlevo) a bez ní (ResNet18_3b, vpravo) natrénované na množinách II a III.

Komplexnější architektura navíc zpomaluje detekci. Měřením doby detekce[12] bylo zjištěno, že model ResNet18_8x detekoval rychlostí $(22{,}4 \pm 0{,}4)$ FPS, zatímco model ResNet18_3b rychlostí $(34{,}5 \pm 0{,}6)$ FPS při velikosti obrázku (224×224) px. Přítomnost pyramidové struktury zpomalila detekci o 35 %.

6.3 Užitečnost syntetických dat

Třetí experiment zkoumal vliv různého charakteru syntetických trénovacích dat na detekci. Trénování probíhalo na všech typech dat se stejnými parametry uvedenými v tab. 6.6. Výchozím předtrénovaným modelem byl ResNet18_8x. Díky dostatečnému množství obrázků nebylo nutné aplikovat křížovou validaci – postačilo obvyklé rozdělení na trénovací, validační a testovací množinu.

Tab. 6.6: Parametry trénování ve třetím experimentu

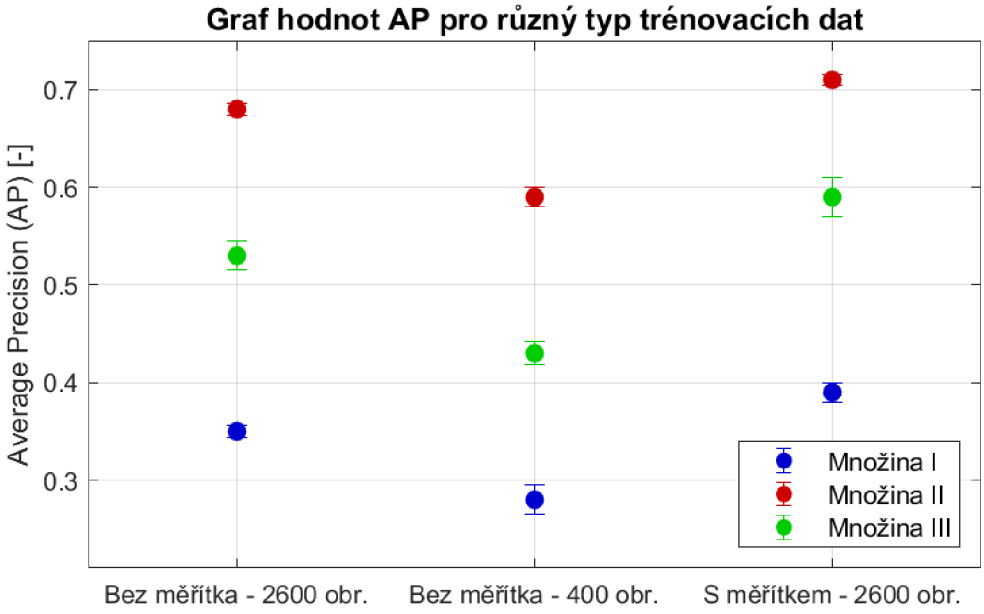
typ dat	minibatch	maximální počet iterací	počáteční krok učení	koeficient snížení kroku učení	počet iterací pro snížení kroku učení
bez měřítka 2600	40	1800	$1 \cdot 10^{-2}$	0,9	200
bez měřítka 400	40	1800	$1 \cdot 10^{-2}$	0,9	200
s měřítkem 2600	40	1800	$1 \cdot 10^{-2}$	0,9	200

Detektory byly trénovány tak dlouho, dokud se validační *AP* neustálila, následně byla vyhodnocena testovací *AP*. Vyhodnocena byla také *AP* na třech validačních množinách skutečné datové sady, aby bylo patrné, zda mají syntetická data pozitivní přínos pro detekci na reálných datech. Výsledky pro všechny kombinace trénovacích dat zachycuje tab. 6.7. Hodnoty *AP* dosažené pro různá rozmístění byly zprůměrovány. Takto byla pro každý typ trénovacích dat a každou skutečnou množinu získána jediná hodnota *AP*, jak je vyneseno do grafu na obr. 6.3.

Tab. 6.7: Výsledky na validační a testovací množině ve třetím experimentu

typ trénovacích dat	rozmístění	*AP* na syntetické množině [-]		*AP* na skutečné množině [-]		
		validační	testovací	I	II	III
bez měřítka 2600 obr.	náhodně	0,95	0,90	0,35	0,67	0,53
	uspořádaně	0,93	0,90	0,35	0,68	0,51
	kombinace	0,92	0,91	0,36	0,68	0,54
bez měřítka 400 obr.	náhodně	0,92	0,83	0,29	0,59	0,44
	uspořádaně	0,92	0,89	0,28	0,60	0,44
	kombinace	0,90	0,87	0,26	0,58	0,42
s měřítkem 2600 obr.	náhodně	0,80	0,81	0,40	0,71	0,59
	uspořádaně	0,82	0,84	0,39	0,72	0,61
	kombinace	0,80	0,80	0,38	0,71	0,57

[12] Měřeno na Intel Core i3 5010U 2,1 GHz + NVIDIA GeForce 920M 2 GB



Obr. 6.3: AP na testovacích množinách pro různé typy trénovacích dat

Z experimentu lze vyvodit několik závěrů:

- **Syntetická data umožnila přesnější natrénování modelu**

Srovnáním validační *AP* modelu ResNet18_8x z tab. 6.5, trénovaného na skutečných datech, a modelu ResNet18_8x s měřítkem z tab. 6.7, trénovaného pouze na syntetických datech, je patrný nárůst *AP* o 0,12. Prokazuje se tak užitečnost syntetického generátoru.

- **S rostoucím množstvím vygenerovaných obrázků roste přesnost detekce**

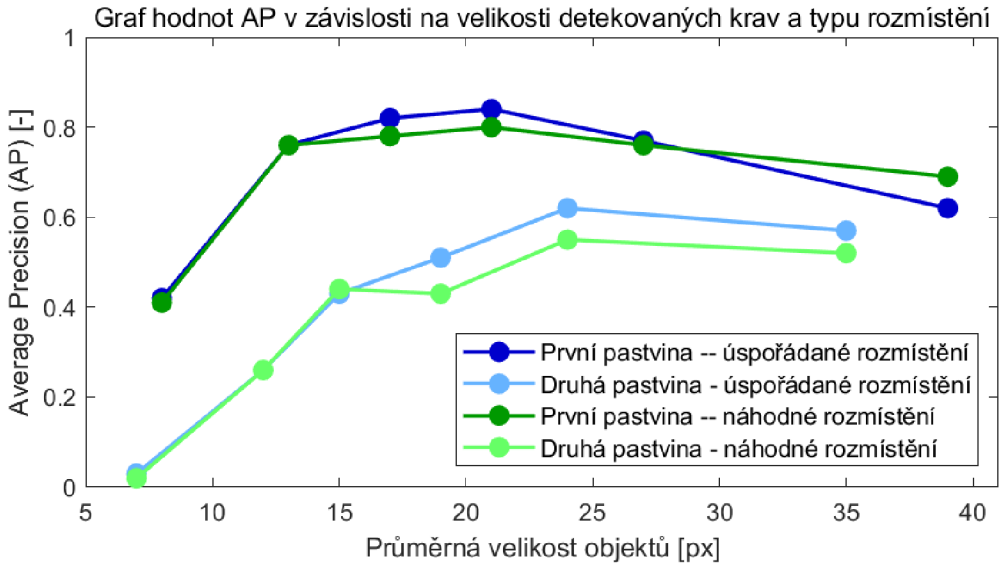
Po srovnání hodnot *AP* prvních dvou typů dat na obr. 6.3 je patrné, že větší množství vygenerovaných obrázků přesnost detekce podstatně zlepšilo, přestože byly do umělých obrázků vkládány stále stejné výřezy krav. Rozšíření syntetické datové sady o nové obrázky pozadí a výřezy krav by mělo vést k dalšímu zvýšení robustnosti detektoru, a to časově relativně nenáročnou cestou.

- **Reprezentativnější trénovací data vedou k lepší detekci**

Srovnáním typu dat s přizpůsobeným měřítkem pozadí a bez něj (první a třetí typ dat na obr. 6.3) lze dospět k závěru, že reprezentativnější charakter dat pomáhá s lepší detekcí, *AP* stoupla v průměru o 0,05. Způsobeno je to převážně velikostí objektů, jež se na obrázcích s měřítkem více blíží velikosti objektů v testovacích množinách. Různorodě zaplněné trénovací obrázky s měřítkem navíc obsahují více odkrytého pozadí, zároveň i části zahuštěné kravami.

- **Uspořádané rozmístění může přispět k přesnější detekci**

Porovnáním hodnot *AP* z tab. 6.7 pro uspořádané a náhodné rozmístění vyplývá, že neměl na skutečné datové sadě typ rozmístění na přesnost detekce vliv. Srovnání proto proběhlo také na snímcích japonské datové sady o různých velikostech krav. Hodnoty *AP* dvou modelů, jež byly natrénovány na datech s měřítkem s uspořádaným rozmístěním a náhodným rozmístěním, zobrazuje graf na obr. 6.4. Uspořádané rozmístění vedlo paradoxně k lepší detekci osamoceně rozmístěných krav, a to v oblasti velikostí kolem 20 px. Užitečnost uspořádaného rozmístění je proto potřeba posuzovat individuálně konkrétním typu dat.



Obr. 6.4: AP pro různé typy rozmístění objektů

6.4 Finální detektor

Na základě proběhlých experimentů byla zvolena architektura finálního modelu a způsob jeho natrénování. Použit byl model s pyramidovou strukturou ResNet_8x pro případ, že by se s větším množstvím dat projevil vliv pyramidové struktury. Trénování finálního detektoru proběhlo ve 3 fázích, parametry trénování každého kroku znázorňuje tab. 6.8.

- **Předtrénování na syntetické datové sadě** – Model ResNet_8x byl předtrénován na syntetických obrázcích s měřítkem napůl s uspořádaným a napůl náhodným rozmístěním – celkem na 5600 obrázcích. Zbylých 400 obrázků sloužilo v průběhu trénování k validaci pro kontrolu, jestli nedochází k nadměrnému naučení sítě na trénovacích datech. Výsledná validační AP = 0,87, testovací AP = 0,87.
- **Evaluace křížovou validací na skutečné datové sadě** – Předtrénovaný model byl následně trénován na skutečné datové sadě. Trénování probíhalo křížovou validací na 3 množinách obdobným způsobem jako v experimentu v kapitole 6.2.
- **Dotrénování na skutečné datové sadě** – Předtrénovaný model z prvního kroku byl závěrem znovu natrénován – tentokrát na všech 3 množinách. Smysl této procedury je blíže popsán v [46]. K dotrénování se volí stejný počet iterací jako u evaluace, pouze zvýšený úměrně počtu trénovacích dat. Trénování už není kontrolováno žádnou validační množinou.

Tab. 6.8: Parametry trénování finálního detektoru

fáze trénování	minibatch	počet iterací	počáteční krok učení	koeficient snížení kroku učení	počet iterací pro snížení kroku učení
předtrénování	40	6720	$1 \cdot 10^{-2}$	0,9	420
evaluace	40	1200	$5 \cdot 10^{-5}$	0,9	200
dotrénování	40	1840	$5 \cdot 10^{-5}$	0,9	230

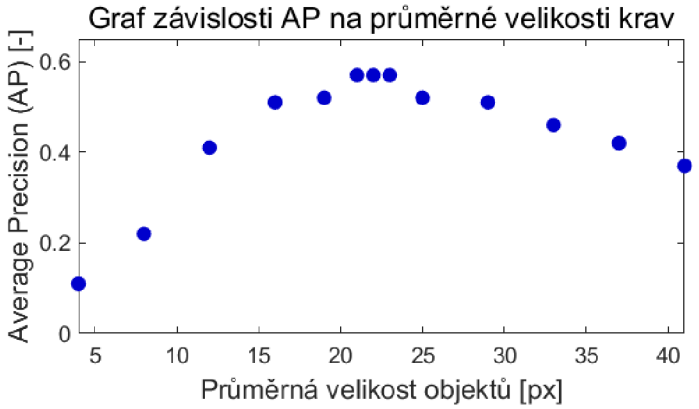
Ukázalo se, že pro dotrénování modelu na skutečných datech je potřeba zvolit velice malý krok učení. Skutečné obrázky, kterých nebylo k dispozici mnoho, byly příliš specifické a snadno přetrénovaly detektor, který byl již dobře natrénován na syntetických datech. Malý krok učení tak nevedl ke zvýšení validační *AP*, jak je vidět v tab. 6.9. Nepatrně se však zvýšila testovací *AP*, která byla pro kontrolu na závěr vyhodnocena. Dotrénování detektoru na všech třech množinách podle očekávání opět zvýšilo testovací *AP*. Provedením experimentu byl získán finální model *detektor_malych_krav*.

Tab. 6.9: Vyhodnocení finálního detektoru

fáze trénování	*AP* na validační množině [-]				*AP* na testovací množině [-]
	I	II	III	průměr	
předtrénování	0,42	0,73	0,66	0,60 ± 0,16	0,52
evaluace	0,43	0,75	0,61	0,60 ± 0,16	0,53 ± 0,02
dotrénování	–	–	–	–	0,56

Na celém testovacím obrázku o vysokém původním rozlišení, z něhož pochází výřezy testovací množiny, byla vyhodnocena *AP* pro různá rozlišení tohoto obrázku. Z grafu na obr. 6.5 je patrné, že dochází k nejpřesnějším detekcím při velikosti objektů (22 ± 7) px, *AP* zde činila 0,57. Detektor byl však trénován na syntetických a skutečných obrázcích, které měly všechny průměrnou velikost objektů 15 px.

Toto chování lze vysvětlit kombinací dvou jevů. Model detekuje obecně nejpřesněji objekty o velikosti rovné velikosti trénovacích objektů. Zároveň s klesající velikostí objektů při takovýchto hraničních velikostech výrazně ubývají detaily objektů. Průnik těchto dvou aspektů způsobuje nejlepší výsledky detektoru právě při mírně větším rozlišení objektů, než na jakých byl detektor natrénován. Potvrzuje se tak myšlenka, která vyplynula z prvního experimentu, že s nízkým rozlišením klesá schopnost detektoru generalizovat na mírně odlišných datech.



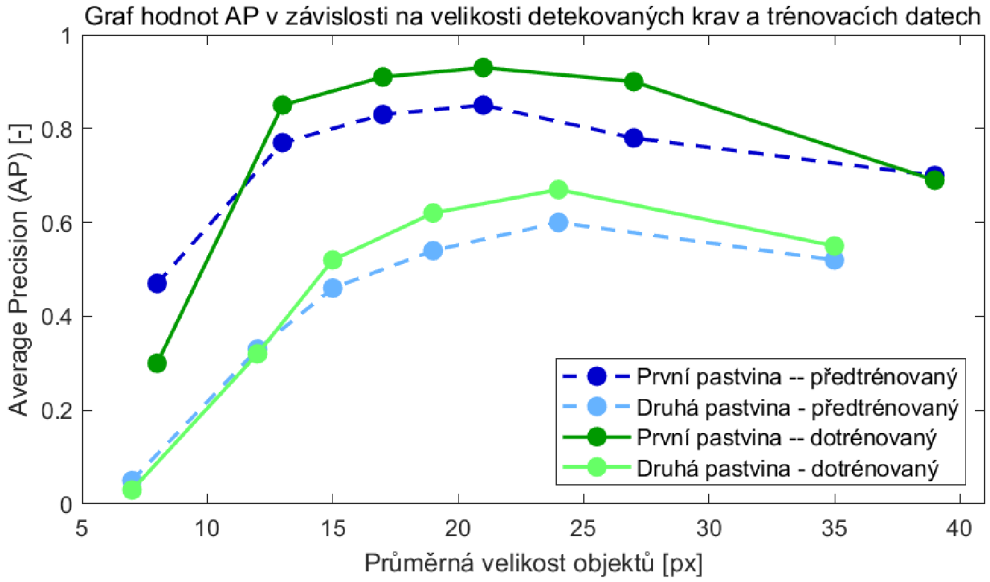
Obr. 6.5: AP pro různá rozlišení testovacího obrázku

6.5 Vyhodnocení na japonské datové sadě

Finální detektor vykazuje na testovací množině skutečné datové sady relativně malé hodnoty *AP*. Byl proto otestován také na japonské datové sadě, aby se ukázalo, zda je tato nízká hodnota způsobena obtížností testovacích dat.

V grafu na obr. 6.6 jsou vyneseny *AP* dosažené na obrázcích stejných měřítek jako v experimentu v kapitole 6.1. Finální detektor byl bez trénování otestován na všech snímcích z první pastviny (tmavě zelená). Při průměrné velikosti objektů (21 ± 4) px dosahuje nejlepší

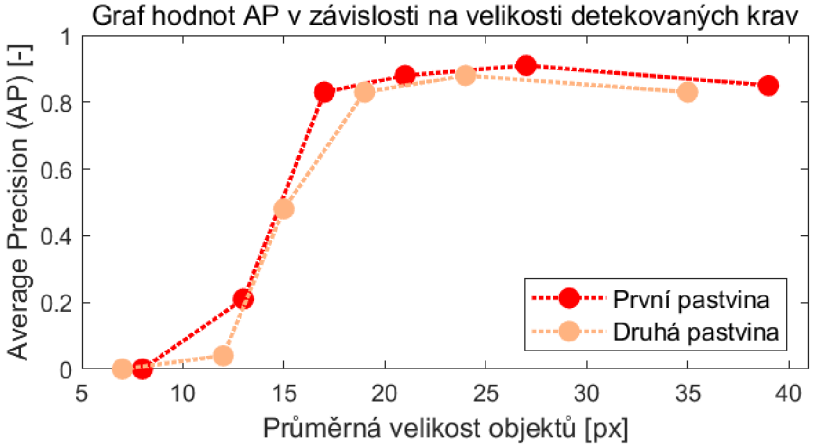
AP rovné 0,93, malé odchylky od této velikosti přesnost výrazně nesnižují. Detektor natrénovaný na syntetických datech a dotrénovaný na hrstce skutečných dat tedy dosáhl na japonské sadě stejných výsledků, jako kdyby na ní byl trénován. Odpovídají tomu i hodnoty *AP* na druhé pastvině dané velikosti objektů (světle zelená).



Obr. 6.6: AP po různých fázích trénování finálního detektoru.

Graf zobrazuje také *AP* změřené bezprostředně po předtrénování finálního detektoru na syntetických datech (modře). Na rozdíl od skutečné takové sady je patrný rozdíl v přesnosti detekce před dotrénováním na skutečných datech a po něm. Dotrénování mělo za následek podstatné zvýšení *AP* v oblasti velikostí blízkých trénovacím datům. Naopak se zhoršila schopnost detektoru rozpoznávat objekty o výrazně odlišných velikostech. Dodatečně bylo zjištěno, že lze dotrénováním na japonské datové sadě *AP* ještě zvýšit.

Bylo prozkoumáno, zda je možné dorovnat rozdíl *AP* na první a druhé pastvině. Ukázalo se, že se tak stane při větším rozlišení obrázků. Natrénován byl detektor na nové sadě čítající 2000 syntetických obrázků s měřítkem a náhodným rozmístěním. Krávy měly na snímcích průměrnou velikost (32 ± 10) px, což lze stále dle COCO považovat za malé objekty. Účinek je patrný z grafu na obr. 6.7. Díky zachovalejším detailům přestal model nesprávně označovat kontrastní stíny, jež zdánlivě připomínaly krávy.



Obr. 6.7: AP detektoru natrénovaného na větších objektech

6.6 Vizuální posouzení detektoru

První ukázku detekce finálního detektoru zobrazuje obr. 6.8. Jedná o snímek z japonské datové sady s měřítkem 0,20; průměrná velikost objektů na něm činí (17 ± 2) px. Číslo nad každým rámečkem udává skóry jistoty. Na obrázku se nachází 16 objektů, detekováno bylo 21. *AP* zde dosahuje hodnoty 0,86.

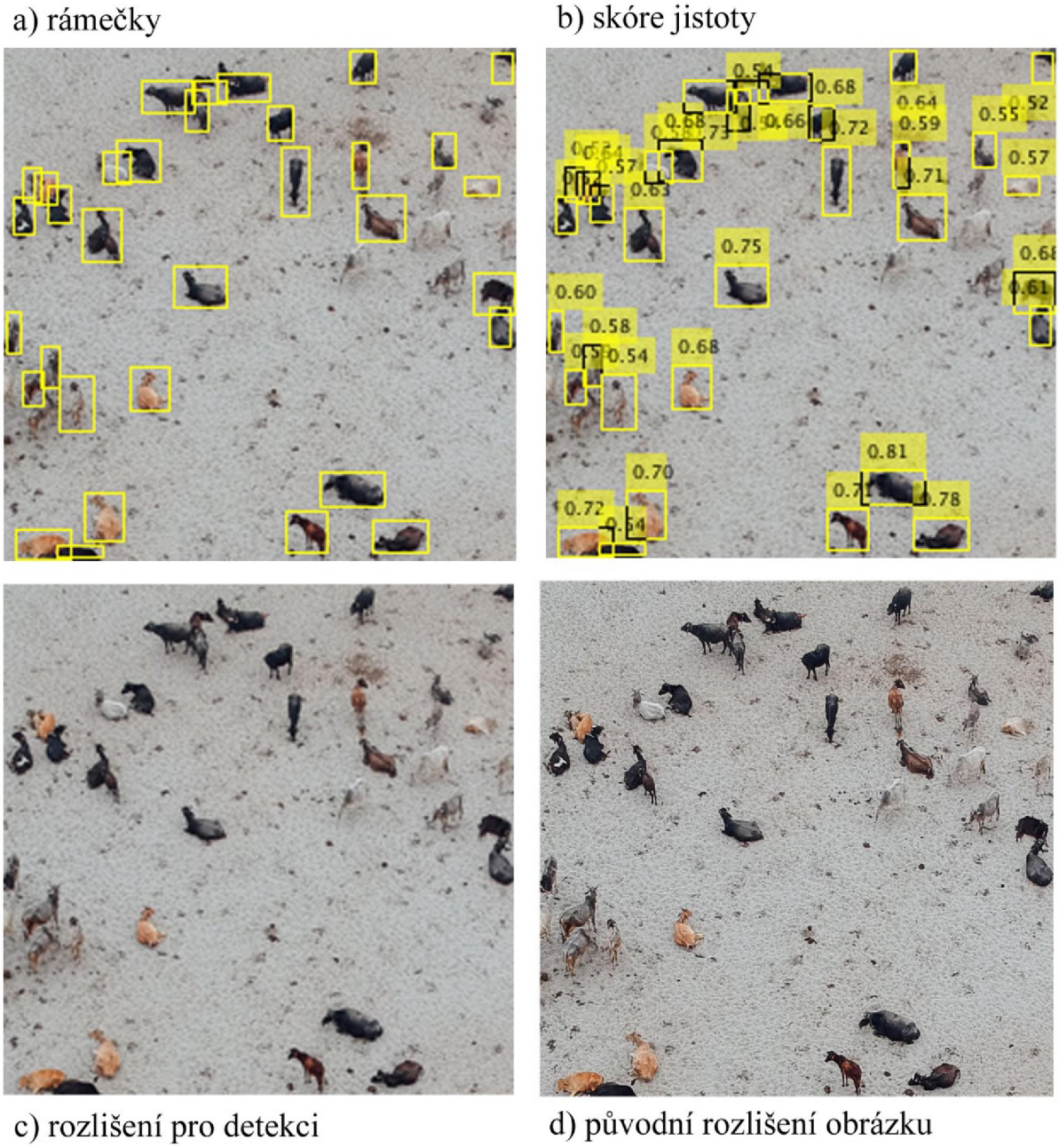
Mezi nadbytečné detekce patří převážně kontrastní tvary v terénu. Malý bílý kámen detektor zaměnil za krávu, jelikož byl trénován na syntetických snímcích, kde objekty s nejmenším rozlišením vypadaly podobně. Pro aplikaci s úzce vymezenými velikostmi objektů by nebylo potřeba trénovat na příliš malých objektech bez detailů a těmto detekcím by bylo zamezeno. Rovněž by bylo možno vyfiltrovat všechny rámečky o výrazně odlišných velikostech, než jsou v aplikaci předpokládány.

Za povšimnutí stojí detekce ve stádě krav, které je v obrázku zvětšeno. Model vykazuje jisté lokalizační odchylky pro YOLO typické, objekty však určil spolehlivě. Zmaten byl ale skálou v pravé části výřezu, kterou označil s vysokou jistotou za krávu. Po provedení detekce na vyšším rozlišení, tedy na měřítku 0,50, díky větším detailům skálu odhalil a snížil skóre jistoty na pomezí prahové hodnoty, která byla nastavena pro detektor právě na 0,55.

Obr. 6.8: Detekce na japonské datové sadě. Zdrojový obrázek z [42].

Druhý ukázkový snímek je na obr. 6.9. Jedná se o výřez, který zahrnuje 8,3 % plochy celého testovacího snímku. Jde o stejný výřez, jež byl rozebrán na obr. 6.2,. Porovnáním snímků na obou obrázcích je patrné zlepšení v detekci, způsobené trénováním na syntetických datech. Hodnota AP spolu s precesion-recall křivkou je na obr. 6.10. Detekce proběhly na snímku na obr. 6.9 c) obsahujícím objekty o velikosti (17 ± 3) px. Detektor rozpoznal 33 ze 42 přítomných

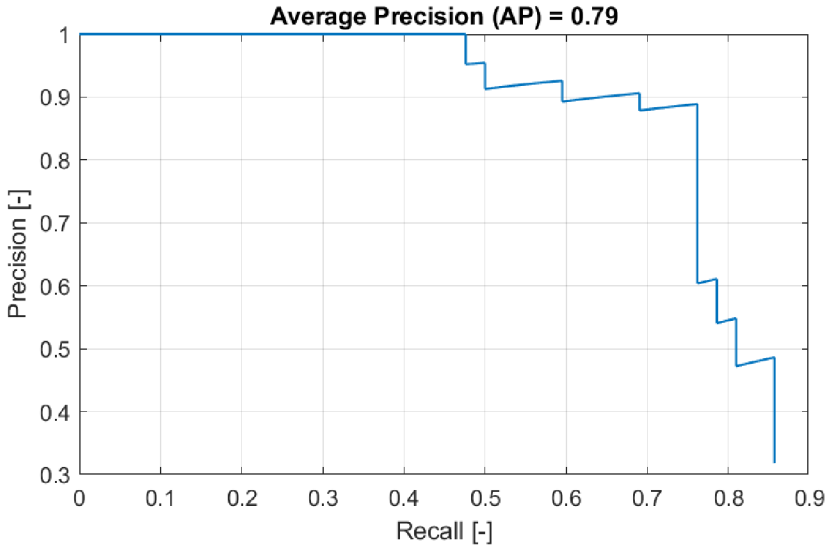
krav. Samostatné rámečky detekcí a příslušná skóre jistoty jsou na obr. 6.9 a) a b). Jelikož se vlivem nízkého rozlišení jedná o obtížný snímek na vizuální posouzení, zobrazuje ho obr. 6.9 d) v maximálním rozlišení.



Obr. 6.9: Detekce na výřezu testovacího obrázku. Zdrojový obrázek z [50].

Na snímku detektor nerozpoznává většinu splývajících objektů. Díky natrénování na syntetických datech jich nicméně detekuje výrazně více, než když byl trénován pouze na skutečných snímcích. Je pravděpodobné, že by se situace ještě zlepšila, pokud by byly úmyslně vygenerovány syntetické snímky podobného charakteru.

Problematická situace nastává, pokud se objekty překrývají nebo na sebe vizuálně navazují, jak se děje převážně v levém horním rohu snímku. Dvě hnědé krávy u levého okraje, které by člověk pravděpodobně považoval za jedinou krávu, detektor rozlišit dokázal. V jiných, jednodušších situacích se ale zmýlil, určení přesného počtu krav v těsných uskupeních bylo proto nespolehlivé.



Obr. 6.10: Precision-recall křivka

6.7 Diskuze

6.7.1 Schopnost počítat malé objekty

Metoda YOLO prokázala veliký potenciál v detekci malých osamocených objektů. V práci vyplynuly tři zásadní problémy, jejichž řešení byla alespoň částečně nalezena:

- **Neschopnost detekce malých objektů** – Původní architektura metody YOLO podvzorkovávala 32× a byla schopná s přijatelnou přesností detekovat krávy do velikosti přibližně 30 pixelů. Upravená verze, jež podvzorkovávala pouze 8×, posunula tuto hranici na cca 13 pixelů. Velikost výsledné mřížky tedy vyřešila tento problém.
- **Redundantní rámečky** – Jemnější mřížka vedla, především u větších objektů, ke vzniku většího počtu duplicitních rámečků. Za předpokladu, že se objekty vyskytují osamoceně, mohly být tyto rámečky vyfiltrovány přísným nastavením algoritmu non-max suppression.
- **Falešná pozitiva v pozadí** – S klesajícím rozlišením obrázků bylo složitější odlišit detekované objekty od podobných tvarů v pozadí. Vznikaly tak nadbytečné rámečky zkreslující počet přítomných objektů. Z předpokladu velikosti objektů, s nimiž bude detektor pracovat, lze provést filtrace rámečků o odlišné velikosti. Dalším řešením je nasnímání velkého množství různých pozadí pro trénování. Objekty do nich mohou být jednoduše vloženy syntetickým generátorem. Pomohl by také systém, který by poznal měřítko celého snímku a vzal ho v potaz při návrhu detekcí.

Komplikace s detekcí nastaly ve shlucích objektů a nepříznivě ovlivnily výsledný počet objektů. Následující problémy nebyly v práci dostatečně vyřešeny a vyžadují další výzkum:

- **Splynutí objektů** – Některé sousedící objekty vyhodnotil detektor jako jeden objekt. Bylo by vhodné nalézt architekturu, která by je dokázala odlišit.
- **Filtrace rámečků** – Rámečky označující výrazně se překrývající objekty ve skupinách nelze vyfiltrovat univerzálním nastavením non-max suppresion. Odlišení redundantních rámečků od rámečků patřících okolním objektům je obtížné a věda se jím zabývá [23].

Z uvedených poznatků lze posoudit, že metoda YOLO přinese spolehlivé výsledky při počítání malých a osamocených objektů, které se výrazně neshlukují. K počítání koncentrovaných objektů lze v současné době raději doporučit jiné metody.

Potenciálním řešením, jak počítat např. krávy ve velkých stádech, by byl systém, jež by standardně detekoval krávy metodou YOLO. Simultánně by však vyhledával zahuštěné oblasti, na nichž by aplikoval adekvátnější metodu jako třeba hustotními mapy.

6.7.2 Využitelnost dosažených výsledků a náměty na pokračování

Výsledky této práce by mohly najít uplatnění v aktuálně vyvíjeném komerčním produktu Cownter [47]. Projekt se zaměřuje na počítání krav na australských pastvinách o různě komplikovaném pozadí. Při rozlišení, na které je detektor optimalizován, mají krávy na snímcích velikost přibližně (15-20) px. [47] Jedná se právě o velikosti uvažované v této práci. Použitím syntetického generátoru by také bylo možné vyvážit datovou sadu tak, aby byly rovnoměrně zastoupeny snímky všech typů pozadí.

Dosažené poznatky lze využít i na jiný typ objektů podobného charakteru. Jedním z nich je detekce a počítání vozidel z leteckých snímků. Jak bylo zjištěno v této práci, malá velikost objektů není díky vhodným přizpůsobením pro metodu YOLO překážkou. Vozidla se navíc na snímcích navzájem nepřekrývají, odpadají tak nedostatky ve filtrování detekcí. Na rozdíl od hustotních map je možné metodou YOLO detekovat více tříd a vozidla tak rozdělit do kategorií. Rozšířením syntetického generátoru by bylo možné rozmístit vozidla realisticky. Za zvážení stojí úprava metody YOLO tak, aby pracovala s natočenými rámečky, a tak přesně označila obdélníkové tvary aut. V nové verzi MATLABu by takové přizpůsobení mohlo být proveditelná. Nabízí se podrobná analýza např. využití parkovacích míst a hustoty provozu. Taková analýza může pomoci s plánováním rozvoje lidských sídel a infrastruktury.

Námětem na další práci může být použití sémantické segmentace diskutované v kapitole 2.4.3. Bylo by dobré ji aplikovat na podobný problém jako v této práci a porovnat dosažené výsledky. Tuto metodu využívá k počítání široké škály objektů, např. tuleňů a kormoránů, platforma Picterra [48].

7 Závěr

Práce se zabývala detekcí malých objektů v obrazových datech, ke které byla využita metoda YOLO fungující na bázi hlubokého učení. V rešeršní části byly nejdříve nastíněny základy hlubokého učení, následně představeny běžně používané metody detekce objektů. Podrobně byl popsán princip metody YOLO spolu s přednostmi a nedostatky jejích jednotlivých verzí. Bylo zjištěno, jak lze interpretovat malé objekty, jakými problémovými situacemi se v oblasti malých objektů současná věda zabývá a jak k problematice přistupuje. Získané poznatky posloužily k rozumné volbě řešeného problému a zároveň jako vodítko pro návrh potřebných nástrojů.

Za typ objektu, k jehož detekci měla být metoda YOLO přizpůsobena, byla zvolena hospodářská zvířata nasnímaná z pohledu shora. Vzhledem k charakteru snímků zvoleného objektu a jejich dostupnosti bylo přistoupeno k tvorbě syntetického generátoru obrázků. Ten dokázal ze skromného množství obrázků pastvin a výřezů krav vytvořit rozsáhlou datovou sadu bez nutnosti zdlouhavé anotace dat. Nabízí mnohá nastavení, jež mohou přispět ke generaci i jiného typu syntetických obrázků.

V zájmu zachování všech detailů obrázků o vysokém rozlišení byl v práci vytvořen nástroj, jež tyto obrázky dokáže rozdělit na části, na nich provést detekci natrénovaným detektorem a výsledky sloučit. Následně spočítá objekty na obrázku podle počtu výsledných rámečků.

Pro natrénování výsledného detektoru schopného detekovat malé objekty bylo potřebné nejprve zajistit odpovídající datové sady. Rozsáhlá sada snímků krav z japonských pastvin byla využita k hledání limitů metody YOLO. Zkompletována byla skutečná datová sada z několika obrázků o vysokém rozlišení, jež zachycovaly obtížně rozeznatelné krávy ve stádech. Rovněž byla vygenerována objemná sada syntetických obrázků.

V práci došlo také k návrhu architektury sítě pro finální model. Za předtrénovanou síť sloužící k extrakci příznaků byl zvolen model ResNet18 pro svou jednoduchost a adekvátní přesnosti. Za něj byla přidána struktura metody FPN, která zjemnila výstupní mřížku finálního modelu.

Představená řešení byla otestována v několika experimentech, které mimo jiné ověřovaly použitelnost metody YOLO pro detekci malých objektů. V prvním experimentu provedeném na japonské datové sadě byla nalezena limitní velikost krav pro původní strukturu YOLO a pro upravenou verzi. Druhý experiment proběhl na skutečné datové sadě a ukázal, že hlavní vliv na zpřesnění detekce neměla samotná pyramidová struktura, ale převážně zjemnění mřížky. Třetí experiment vyzdvihl užitečnost syntetické datové sady, díky níž stoupla hodnota metriky AP na testovací množině o 0,19 vůči trénování pouze na malém počtu skutečných snímků. Komplikovaněji probíhalo následné dotrénování právě na skutečných snímcích, jež mělo tendenci model výrazně přeučit a snížit jeho přesnost. Toto dotrénování však vedlo ke zvýšení přesnosti detektoru na japonské datové sadě.

V práci byly splněny všechny vytyčené cíle. Bylo posouzeno, jaký druh objektů lze metodou YOLO spolehlivě počítat. Výsledky práce by našly využití např. ve vyvíjeném produktu Cownter, který má sloužit k monitoringu australských krav. Dosažené závěry lze aplikovat i na příbuzné problémy, jako je analýza rozmístění vozidel na leteckých snímcích.

Literatura

[1] WANI, M. Arif, Farooq Ahmad BHAT, Saduf AFZAL a Asif Iqbal KHAN. *Advances in Deep Learning*. Cham: Springer, 2020. ISBN 978-981-13-6493-9.

[2] SKANSI, Sandro. *Introduction to deep learning: from logical calculus to artificial intelligence*. Cham: Springer, 2018. Undergraduate texts in computer science. ISBN 978-3-319-73003-5.

[3] BAKAMBEKOVA, Adilya a Alex Pappachen JAMES. Deep Learning Theory Simplified. JAMES, Alex Pappachen, ed. *Deep Learning Classifiers with Memristive Networks: Theory and Applications*. Cham: Springer, 2020, s. 41-55. ISBN 978-3-030-14522-4.

[4] AGHDAM, Hamed Habibi a Elnaz Jahani HERAVI. *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. Cham: Springer, 2017. ISBN 978-3-319-57549-0.

[5] MICHELUCCI, Umberto. *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*. Berkeley: Apress, 2019. ISBN 978-1-4842-4975-8.

[6] GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. 2014, s. 580-587 [cit. 2020-05-19]. DOI: 10.1109/ CVPR.2014.81. ISBN 978-147995117-8. ISSN 1063-6919. Dostupné z: https://ieeexplore. ieee.org/document/6909475

[7] GIRSHICK, Ross. Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision* [online]. 2015, s. 1440-1448 [cit. 2020-05-19]. DOI: 10.1109/ICCV. 2015.169. ISBN 978-1-4673-8391-2. ISSN 15505499. Dostupné z: http://ieeexplore.ieee. org/document/7410526/

[8] REN, Shaoqing, Kaiming HE, Ross GIRSHICK a Jian SUN. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2017, **39**(6), 1137-1149 [cit. 2020-06-12]. DOI: 10.1109/TPAMI.2016.2577031. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/document/7485869/

[9] KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey E. HINTON. ImageNet Large Scale Visual Recognition Challenge. In: *Advances in Neural Information Processing Systems 25 (NIPS 2012)* [online]. 2012, s. 1097-1105 [cit. 2020-05-28]. ISBN 978-162748003-1. ISSN 10495258. Dostupné z: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[10] DAI, Jifeng, Yi LI, Kaiming HE a Jian SUN. R-FCN: object detection via region-based fully convolutional networks. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems* [online]. 2016, s. 379–387 [cit. 2020-06-13]. DOI: 10.5555/3157096.3157139. ISBN 978-1-5108-3881-9. ISSN 1049-5258. Dostupné z: https://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf

[11] HUANG, Jonathan, Vivek RATHOD, Chen SUN, et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In: *Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition* [online]. 2017, s. 3296-3305 [cit. 2020-05-28]. DOI: 10.1109/CVPR.2017.351. ISBN 978-153860457-1. Dostupné z: https://ieeexplore. ieee.org/document/8099834

[12] LIU, Wei, Dragomir ANGUELOV, Dumitru ERHAN, et al. SSD: Single Shot MultiBox Detector. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science* [online]. 2016, s. 21-37 [cit. 2020-05-28]. DOI: 10.1007/978-3-319-46448-0_2. ISBN 978-

331946447-3. ISSN 0302-9743. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2

[13] SIMONYAN, Karen a Andrew ZISSERMAN. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* [online]. 2015 [cit. 2020-05-28]. Dostupné z: http://arxiv.org/abs/1409.1556

[14] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. You Only Look Once: Unified, Real-Time Object Detection. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2016, s. 779-788. DOI: 10.1109/CVPR.2016.91. ISBN 978-146738850-4. ISSN 1063-6919. Dostupné také z: https://ieeexplore.ieee.org/document/7780460

[15] REDMON, Joseph a Ali FARHADI. YOLO9000: Better, Faster, Stronger. In: *Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* [online]. 2017, s. 6517-6525 [cit. 2020-05-28]. DOI: 10.1109/CVPR.2017. 690. ISBN 978-153860457-1. Dostupné z: https://ieeexplore.ieee.org/document/8100173

[16] REDMON, Joseph a Ali FARHADI. YOLOv3: An Incremental Improvement. *CoRR* [online]. 2018 [cit. 2020-06-14]. Dostupné z: https://arxiv.org/abs/1804.02767

[17] LIN, Min, Qiang CHEN a Shuicheng YAN. Network In Network. In: *2nd International Conference on Learning Representations, {ICLR} 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* [online]. 2014 [cit. 2020-06-13]. Dostupné z: http://arxiv.org/abs/1312.4400

[18] YOSINSKI, Jason, Jeff CLUNE, Yoshua BENGIO a Hod LIPSON. How transferable are features in deep neural networks? In: *Advances in Neural Information Processing Systems 27 (NIPS 2014)* [online]. 2014, s. 3320-3328 [cit. 2020-06-12]. ISSN 1049-5258. Dostupné z: https://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

[19] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep residual learning for image recognition. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. 2016, s. 770-778 [cit. 2020-06-13]. DOI: 10.1109/CVPR.2016.90. ISBN 978-146738850-4. ISSN 1063-6919. Dostupné z: https://ieeexplore.ieee.org/document/7780459

[20] LIN, Tsung-Yi, Piotr DOLLÁR, Ross GIRSHICK, et al. Feature pyramid networks for object detection. In: *Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* [online]. s. 936-944 [cit. 2020-06-13]. DOI: 10.1109/CVPR. 2017.106. ISBN 978-153860457-1. Dostupné z: https://ieeexplore.ieee.org/document/8099589

[21] LIN, Tsung-Yi, Priya GOYAL, Ross GIRSHICK a Piotr DOLLÁR. Focal Loss for Dense Object Detection. In: *Proceedings of the IEEE International Conference on Computer Vision* [online]. 2017, s. 2999-3007 [cit. 2020-06-13]. DOI: 10.1109/ICCV.2017.324. ISBN 978-153861032-9. ISSN 1550-5499. Dostupné z: https://ieeexplore.ieee.org/document/8237586

[22] LIN, Tsung-Yi, Michael MAIRE, Belongie SERGE, James HAYS, Pietro PERONA, et al. Microsoft COCO: Common objects in context. In: Computer Vision – ECCV 2014. Lecture Notes in Computer Science [online]. 2014, s. 740-755 [cit. 2020-06-14]. ISSN 0302-9743. Dostupné také z: https://arxiv.org/abs/1405.0312

[23] JIAO, Licheng, Fan ZHANG, Fang LIU, Shuyuan YANG, Lingling LI, Zhixi FENG a Rong QU. A survey of deep learning-based object detection. IEEE Access [online]. 2019, 7, 128837-128868 [cit. 2020-06-14]. DOI: 10.1109/ACCESS.2019.2939201. ISSN 2169-3536. Dostupné z: https://ieeexplore.ieee.org/document/8825470

[24] ETTEN, Adam Van. You Only Look Twice: Rapid Multi-Scale Object Detection In Satellite Imagery. *ArXiv* [online]. 2018 [cit. 2020-06-14]. Dostupné z: https://arxiv.org/abs/1805.09512

[25] SHAO, Guofan a Keith M. REYNOLDS, ed. *Computer Applications in Sustainable Forest Management: including Perspectives on Collaboration and Integration*. Dordrecht: Springer – Verlag, 2006. Managing Forest Ecosystems. ISBN 1-4020-4305-8.

[26] SINDAGI, Vishwanath A. a Vishal M. PATEL. A survey of recent advances in CNN-based single image crowd counting and density estimation. *Pattern Recognition Letters* [online]. 2018, **107**, 3-16 [cit. 2020-06-23]. DOI: 10.1016/j.patrec.2017.07.007. Dostupné z: https://arxiv.org/pdf/1707.01202.pdf

[27] XU, Mingliang, Zhaoyang GE, Xiaoheng JIANG, Gaoge CUI, Pei LV a Changsheng XU. Depth Information Guided Crowd Counting for complex crowd scenes. *Pattern Recognition Letters* [online]. 2019, **125**, 563-569 [cit. 2020-06-23]. DOI: 10.1016/j.patrec.2019.02.026. Dostupné z: https://arxiv.org/pdf/1803.02256.pdf

[28] LIU, Xialei, Joost VAN DE WEIJER a Andrew D. BAGDANOV. Leveraging Unlabeled Data for Crowd Counting by Learning to Rank. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. 2018, s. 7661-7669 [cit. 2020-06-23]. DOI: 10.1109/CVPR.2018.00799. ISBN 978-153866420-9. ISSN 10636919. Dostupné z: https://ieeexplore.ieee.org/document/8578897

[29] HE, Kaiming, Georgia GKIOXARI, Piotr DOLLÁR a Ross GIRSHICK. Mask R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision* [online]. 2017, s. 2980-2988 [cit. 2020-06-23]. DOI: 10.1109/ICCV.2017.322. ISBN 978-153861032-9. ISSN 15505499. Dostupné z: https://ieeexplore.ieee.org/document/8237584

[30] LIU, Shu, Lu QI, Haifang QIN, Jianping SHI a Jiaya JIA. Path Aggregation Network for Instance Segmentation. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* [online]. Salt Lake City, 2018, s. 8759-8768 [cit. 2020-06-23]. DOI: 10.1109/CVPR.2018.00913. ISBN 978-1-5386-6420-9. ISSN 2575-7075. Dostupné z: https://ieeexplore.ieee.org/document/8579011

[31] FU, Cheng-Yang, Wei LIU, Ananth RANGA a Akexander C. BERG. Dssd: Deconvolutional single shot detector. *ArXiv* [online]. 2017 [cit. 2020-06-23]. Dostupné z: https://arxiv.org/pdf/1701.06659.pdf

[32] ÜNEL, F. Özge, Burak O. ÖZKALAYCI a Cevahir CIGLA. The power of tiling for small object detection. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* [online]. 2019, s. 582-591 [cit. 2020-06-23]. DOI: 10.1109/CVPRW.2019.00084. ISBN 978-172812506-0. ISSN 21607508. Dostupné z: https://ieeexplore.ieee.org/document/9025422

[33] RŮŽIČKA, Vít a Franz FRANCHETTI. Fast and accurate object detection in high resolution 4K and 8K video using GPUs. In: *2018 IEEE High Performance extreme Computing Conference (HPEC)* [online]. 2018 [cit. 2020-06-23]. DOI: 10.1109/HPEC.2018.8547574. ISBN 978-1-5386-5989-2. Dostupné z: https://ieeexplore.ieee.org/document/8547574

[34] PANG, Jiangmiao, Cong LI, Jianping SHI, Zhihai XU a Huajun FENG. R2-CNN: Fast Tiny Object Detection in Large-Scale Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing* [online]. 2019, **57**(8), 5512-5524 [cit. 2020-06-23]. DOI: 10.1109/TGRS.2019.2899955. Dostupné z: https://ieeexplore.ieee.org/document/8672899

[35] *CS231n Convolutional Neural Networks for Visual Recognition* [online]. [cit. 2020-06-23]. Dostupné z: https://cs231n.github.io/

[36] LAU, Felix. Speed/accuracy trade-offs for modern convolutional object detectors. In: *Medium – Get smarter about what matters to you.* [online]. [cit. 2020-06-23]. Dostupné

z: https://medium.com/@phelixlau/speed-accuracy-trade-offs-for-modern-convolutional-object-detectors-bbad4e4e0718
[37] HUI, Jonathan. Understanding Region-based Fully Convolutional Networks (R-FCN) for object detection. In: *Medium – Get smarter about what matters to you.* [online]. [cit. 2020-06-23]. Dostupné z: https://medium.com/@jonathan_hui/understanding-region-based-fully-convolutional-networks-r-fcn-for-object-detection-828316f07c99
[38] HUI, Jonathan. SSD object detection: Single Shot MultiBox Detector for real-time processing. In: *Medium – Get smarter about what matters to you.* [online]. [cit. 2020-06-23]. Dostupné z: SSD object detection: Single Shot MultiBox Detector for real-time processing
[39] REDMON, Joseph. *YOLO: Real-Time Object Detection* [online]. [cit. 2020-06-23]. Dostupné z: https://pjreddie.com/darknet/yolov2/
[40] MATLAB Documentation. *MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink* [online]. The MathWorks, Inc. The MathWorks, c1994-2020 [cit. 2020-06-23]. Dostupné z: https://www.mathworks.com/help/matlab/
[41] *COCO - Common Objects in Context* [online]. [cit. 2020-06-23]. Dostupné z: https://cocodataset.org/
[42] SHAO, Wen, Rei KAWAKAMI, Ryota YOSHIHASHI, Shaodi YOU, Hidemichi KAWASE a Takeshi NAEMURA. Cattle detection and counting in UAV images based on convolutional neural networks. *International Journal of Remote Sensing* [online]. 2020, **41**(1), 31-52 [cit. 2020-06-23]. DOI: 10.1080/01431161.2019.1624858. ISSN 01431161. Dostupné z: https://www.tandfonline.com/doi/full/10.1080/01431161.2019.1624858
[43] Qi WANG, a Yuan YUAN. Learning from synthetic data for crowd counting in the wild. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. 2019, s. 8190-8199 [cit. 2020-06-23]. DOI: 10.1109/CVPR.2019.00839. ISBN 978-172813293-8. ISSN 10636919. Dostupné z: https://ieeexplore.ieee.org/document/8953868
[44] HUI, Jonathan. MAP (mean Average Precision) for Object Detection. In: *Medium – Get smarter about what matters to you.* [online]. [cit. 2020-06-24]. Dostupné z: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173
[45] BROWNLEE, Jason. A Gentle Introduction to k-fold Cross-Validation. In: *Machine Learning Mastery* [online]. [cit. 2020-06-24]. Dostupné z: https://machinelearningmastery.com/k-fold-cross-validation/
[46] BROWNLEE, Jason. How to Train a Final Machine Learning Model. In: *Machine Learning Mastery* [online]. [cit. 2020-06-24]. Dostupné z: https://machinelearningmastery.com/train-final-machine-learning-model/
[47] Cownter - Farm Doctors. *Precision Farming Solutions – Farm Doctors* [online]. Copyright ©2020 Farm Doctors Pty Ltd [cit. 24.06.2020]. Dostupné z: https://www.farmdoctors.com.au/cownter/
[48] *Picterra - Geospatial imagery analysis made easy* [online]. Chavannes: Picterra, c2020 [cit. 2020-06-24]. Dostupné z: https://picterra.ch/
[49] *Find your inspiration. | Flickr* [online]. [cit. 2020-06-24]. Dostupné z: https://www.flickr.Com
[50] COHN, Adam. Cattle on Riverbank, Mathura India. In: *Find your inspiration. | Flickr* [online]. [cit. 2020-06-24]. Dostupné z: https://www.flickr.com/photos/adamcohn/43789281184/

Seznam zkratek

YOLO	You Only Look Once
SGD	Stochastic Gradient Descend
R-CNN	Region Proposal Convolutional Neural Network
SSD	Single Shot Detector
COCO	Common Objects in Context
RoI	Region of Interest
RPN	Region Proposal Network
R-FCN	Region-based Fully Convolutional Network
FPS	Frames per Second
IoU	Intersection over Union
mAP	mean Average Precision
FPN	Feature Pyramid Network
PANet	Path Aggregation Network
GTA V	Grand Theft Auto V
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
RGB	Red Green Blue

Seznam obrázků

Obr. 2.1: Model umělého neuronu 11
Obr. 2.2: Výpočet jedné konvoluce 12
Obr. 2.3: Závislost ztrátové funkce na epoše pro různý krok učení 13
Obr. 2.4: Princip R-CNN detektoru 14
Obr. 2.5: Princip Faster R-CNN detektoru 15
Obr. 2.6: Schéma SSD detektoru 16
Obr. 2.7: Princip metody YOLOv2 17
Obr. 2.8: Vektor hodnot pro jednu buňku výstupní mřížky u YOLOv2 17
Obr. 2.9: Význam parametrů predikujících rámeček 18
Obr. 2.10: Srovnání plochy rámečku a plochy objektu 21
Obr. 2.11: Odhad počtu lidí na obrázku 22
Obr. 2.12: Výsledky detekce pomocí metody PANet 23
Obr. 2.13: Způsoby predikce z map příznaků 23
Obr. 2.14: Postup detekce po výřezech 24
Obr. 4.1: Ukázka požadované struktury adresáře a pojmenování souborů 27
Obr. 4.2: Náhodné rozmístění 28
Obr. 4.3: Uspořádané rozmístění 28
Obr. 4.4: Zobrazení rámečků (ground truth) 28
Obr. 4.5: Vizualizace kritéria vhodnosti 29
Obr. 4.6: Ukázka možných poloh rámečku vůči výřezu v jednom směru 30
Obr. 5.1: Histogramy velikostí krav ve výřezech ze skutečných obrázků 32
Obr. 5.2: Schéma navržené architektury 34
Obr. 6.1: Přesnost detekce dle velikosti objektů a zvolené architektury 38
Obr. 6.2: Detekce na obrázku z testovací množiny 39
Obr. 6.3: AP na testovacích množinách pro různé typy trénovacích dat 41
Obr. 6.4: AP pro různé typy rozmístění objektů 42
Obr. 6.5: AP pro různá rozlišení testovacího obrázku 43
Obr. 6.6: AP po různých fázích trénování finálního detektoru 44
Obr. 6.7: AP detektoru natrénovaného na větších objektech 44
Obr. 6.8: Detekce na japonské datové sadě 45
Obr. 6.9: Detekce na výřezu testovacího obrázku 46
Obr. 6.10: Precision-recall křivka 47

Seznam tabulek

Tab. 5.1: Přehled syntetických datových sad 33
Tab. 6.1: Parametry trénování v prvním experimentu 36
Tab. 6.2: Výsledky křížové validace v prvním experimentu 36
Tab. 6.3: Výsledky na testovacích snímcích 37
Tab. 6.4: Parametry trénování v druhém experimentu 38
Tab. 6.5: Výsledky křížové validace v druhém experimentu 39
Tab. 6.6: Parametry trénování ve třetím experimentu 40
Tab. 6.7: Výsledky na validační a testovací množině ve třetím experimentu 40
Tab. 6.8: Parametry trénování finálního detektoru 42
Tab. 6.9: Vyhodnocení finálního detektoru 43