Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

Bakalářská práce

Automatické sledování pohybu zvířat z kamerového záznamu

Plzeň 2022 Jakub Rada

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub RADA**
Osobní číslo:	**A19B0316P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Automatické řízení a robotika**
Téma práce:	**Automatické sledování pohybu zvířat z kamerového záznamu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

- Nastudování metod počítačového vidění v oblasti sledování pohybu objektů.
- Zpracování dodaných dat se záznamem pohybu zvířat.
- Vytvoření a implementace algoritmu pro sledování pohybu zvířat.
- Vyhodnocení kvality sledovacího algoritmu.

Rozsah bakalářské práce: **30 – 40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

SONKA, Milan; HLAVAC, Vaclav; BOYLE, Roger. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

Vedoucí bakalářské práce: **Ing. Miroslav Hlaváč, Ph.D.**
Výzkumný program 1

Datum zadání bakalářské práce: **15. října 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 15. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 22. května 2022

Jakub Rada

Poděkování

Chtěl bych poděkovat především vedoucímu své práce panu Ing. Miroslavovi Hlaváčovi PhD. za ochotu a pomoc při vypracování této bakalářské práce. Dále bych chtěl poděkovat panu Ing. Miroslavovi Jiříkovi za poskytnuté téma a poskytnutá data. Na konec bych také rád poděkoval MetaCentru za poskytnutou výpočetní kapacitu.

Abstrakt

V této práci se budu zabývat detekcí a sledováním objektů ve videu především za využití metod neuronových sítí a počítačového vidění. Konkrétně půjde o detekci a sledování prasete v kotci. V první části se budu věnovat obecnému popisu počítačového vidění a neuronových sítí. Představím různé příklady a postupy obou těchto metod. Ve druhé části, jež bude převážně praktická, použiji některé z představených metod a natrénuji je na mnou vytvořeném datasetu. V poslední části bude porovnání a vyhodnocení získaných výsledků.

Abstract

In this thesis I will focus on detection and tracking of an object in video mainly by using a methods of neural networks and computer vision. Specifically it will be about detection and tracking of a pig inside a paddock. In the first part I will give a general description of computer vision and neural networks. Mainly some examples and procedures of both of those metodhs. In second part, that will be mainly practical, I will use some of those procedures on my own specific dataset. In the last part there will be comparison and evaluation of obtained results.

Obsah

1 Úvod

V současné době se metody počítačového vidění objevují všude kolem nás, aniž bychom si toho byli vědomi. Ovšem většina z nás neví, co se za pojmem počítačové vidění vlastně skrývá. Proto bych rád v této bakalářské práci tyto metody objasnil a následně využil na příkladu, jenž má skutečně praktické využití. Základními metodami počítačového vidění jsou především detekce a sledování nějakého určitého a předem definovaného objektu. Tato úloha se může na první pohled zdát nám lidem velmi jednoduchá, neboť objekty kolem sebe rozeznáváme a detekujeme již od útlého věku. Ovšem i my jsme se nejdříve museli naučit o jaký objekt se jedná, či jak jej rozlišit. Čím více objektů poznáváme, tím menší problém s jejich identifikací máme. Postupem času dokonce dokážeme rozeznat objekty jedné instance i přesto, že jsme neviděli všechny existující objekty daného typu. Hezkým příkladem by nám mohlo být rozeznání kočky. Existuje spoustu plemen koček, jež se liší vzhledem. Ovšem i kočky jednoho plemene se mohou lišit vzhledem a my přesto dokážeme novou kočku správně identifikovat jakožto kočku, i když jsme se s ní setkali poprvé. Tento proces ovšem pro počítače neplatí. Dříve bylo výpočetně velmi náročné určit, co se v daném obraze vyskytuje, popřípadě kde se to vyskytuje. Tento problém však postupně ustupuje se zvýšením výpočetního výkonu a rozvojem metod strojového učení. S tímto aparátem již počítač dokáže efektivně detekovat a sledovat objekty.

V sedmdesátých letech minulého století, když už počítače byly schopné zpracovávat větší objemy dat nesoucí obrazové informace, vzniká nový obor, pro nějž se vžilo označení počítačové vidění. Mimo pojmenování tohoto oboru označuje počítačové vidění obecně všechny systémy, které dokáží pracovat automaticky na základě přijatých dat nebo informací právě ze zpracovaného obrazu. Počítačové vidění se může zabývat velmi širokým spektrem objektů

od dopravní situace až po rozpoznávání tváří v některých bezpečnostních systémech. Ovšem velmi významným oborem, kde se počítačové vidění využívá, je průmyslová automatizace. Pro využití počítačového vidění v průmyslové automatizaci se zavedl název strojové vidění. To je charakteristické především svou vazbou na výrobní proces a jeho řízení. Tím je myšlena hlavně vizuální inspekce předepsaných parametrů, počítání objektů či hledání defektů. V současné době oba tyto velmi úzce provázané obory využívají především metod neuronových sítí, což je jeden z výpočetních modelů využívaných v umělé inteligenci. Neuronové sítě se snaží co nejvíce napodobit chování odpovídající biologické struktuře mozku. V našem případě se tedy snaží co nejvíc napodobit to, jak by dané objekty rozeznával člověk.

Jako praktický příklad v této práci slouží detekce a sledování prasat z Biomedicínského centra v Plzni. Nachází se zde několik kotců s prasaty, jimž byla transplantována játra, a je u nich potřeba diagnostikovat zda jsou stále naživu. Výstupem této práce tedy bude program schopný detekovat a sledovat prase v jeho kotci. Tento program vzniká především za účelem snazší, rychlejší, efektivnější a hlavně automatické diagnostiky. Využiji zde jak klasických metod počítačového vidění, jako je například segmentace pozadí, tak i metod neuronových sítí.

Hlavními cíli této práce je nastudování metod počítačového vidění především v oblasti sledování zvířat, zpracování vstupních dat, vytvoření algoritmů na sledování prasat a nakonec jejich zhodnocení.

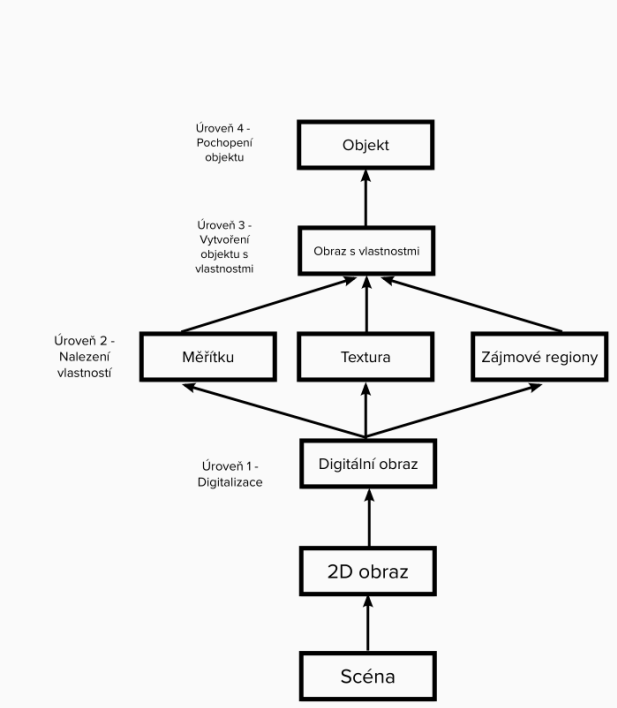
2 Počítačové vidění

To, jak počítač zpracovává obraz, může být chápáno jako hledání vztahu mezi vstupním obrazem a předem určeným modelem daného objektu. Přechod ze vstupního obrazu na daný model redukuje informace obsažené ve vstupním obraze pouze na námi potřebná data. Tento proces je běžně rozdělen na více kroků a úrovní reprezentace informací. Nižší úrovně většinou obsahují především nezpracovaná data vstupního obrazu a vyšší naopak dobře specifikovaná data. Úlohou počítačového vidění je tedy navrhnout reprezentace těchto dat a vytvořit algoritmy, jenž dokáží vyhledat či určit vztahy mezi různými úrovněmi těchto dat nebo mezi konkrétními daty samotnými. V následujících kapitolách budu vycházet především z knihy Image Processing, Analysis and Machine vision [14]

2.1 Reprezentace obrazu

. Reprezentaci obrazu v závislosti na datech můžeme rozdělit do čtyř úrovní, přičemž hranice mezi různými úrovněmi nejsou vždy jasně určené a jsou závislé především na vstupních datech. V zásadě jde o to, že v nižších úrovních získáváme téměř neabstrahované výstupy, a čím výše se posouváme, tím více abstrahované výstupy dostáváme pro lepší pochopení daného obrazu. Je ovšem důležité si uvědomit, že tento proces nemusí být nutně jednosměrný. Velmi často se totiž využívá zpětnovazebných smyček pro zpřesnění či upravení daného algoritmu za účelem upravení, či zpřesnění průběžných výstupů. Na diagramu níže 2.1 je vidět jedna z možností reprezentace obrazu. Jak je z diagramu patrné, tato reprezentace se dá klasifikovat jako hierarchická. Velmi často se tato reprezentace ještě zjednodušuje, a to na dvě části. První částí jsou metody nízkoúrovňového zpracování obrazu a druhou částí metody

vysokoúrovňového pochopení obrazu.



Obrázek 2.1: Příklad reprezentace obrazu

2.2 Nízkoúrovňové metody

Jedná se o metody zpracování, jež potřebují velmi málo informací o obsahu daného obrazu. V případě, že už počítač má informace o obsahu daného obrazu, tak je to díky vysokoúrovňovým metodám pochopení nebo na základě vstupu člověka, který rozumí konkrétnímu problému. Nízkoúrovňové metody velmi často zahrnují kompresi obrazu, předzpracování obrazu kvůli filtraci šumu či zaostření obrazu. Tato zpracování využívají data, jež odpovídají vstupnímu obrazu. Příkladem může být vstupní obraz zachycený fotoaparátem, který už je 2D a je popsán například obrazovou funkcí $f(x, y)$, jejíž hodnoty většinou odpovídají například jasu, který je závislý na parametrech x, y , neboli souřadnicích dané lokace v obraze.

Pokud má být obraz zpracován počítačem, tak musí být nejdříve digita-

lizován. Poté může být reprezentován čtvercovou maticí s elementy odpovídajícími jasu v odpovídající lokaci takzvanými pixely. V současné době jsou však obrazy především barevné, proto je hodnota jasu nahrazená spíše danou hodnotou RGB barevného spektra. Jedná-li se o video, daný vstup se rozdělí na sekvenci více obrazů, které jsou dále zpracovávány a reprezentovány stejným postupem, který je popsán výše.

2.3 Vysokoúrovňové metody

Vysokoúrovňové metody jsou založeny na znalostech, cílech a plánech, jak těchto cílů dosáhnout. Velmi široké využití zde nacházejí metody umělé inteligence. Cílem těchto metod je napodobit lidské vnímání okolí a schopnost rozhodovat na základě informací získaných z daného obrazu. Od počítačového vidění se předpokládá, že dokáže řešit velmi komplexní úlohy a dostat výsledky podobné nebo lepší těm, jaké by určil člověk. Vysokoúrovňové metody většinou začínají s určitým předdefinovaným modelem reálného světa, který porovnávají s digitalizovaným obrazem. Vysokoúrovňové metody velmi často při zpracování obrazu využívají i ty nízkoúrovňové, například když se objeví rozdíl při porovnávání reálného modelu a daného digitalizovaného modelu. Jednou z možností je, že vysokoúrovňová metoda využije nízkoúrovňovou ve zpětnovazebné smyčce pro lepší pochopení daného obrazu. Vysokoúrovňové metody, na rozdíl od nízkoúrovňových, nevyužívají všechna data obsažená v obraze, ale pouze ta, která potřebují na dosažení daného cíle.

2.4 Koncepty reprezentace

Jako častý popis obrazu a signálu jsou využívány především matematické modely. Signálem je myšlena funkce závislá na proměnné, která reprezentuje nějakou fyzikální veličinu. Tato funkce může být jak jednodimenzionální, tak

vícedimenzionální. Například jednobarevný obraz může být popsán skalární funkcí, zatímco barevný obraz musí být popsán funkcí vektorovou. Funkce, které budu používat pro popis, budou především spojité. Příkladem použití spojité funkce pro popis obrazu je reprezentace obrazu vnímaného lidským okem, či kamerou. Tato reprezentace je modelována spojitou funkcí $f(x, y)$, kde x, y jsou souřadnice, nebo funkcí $f(x, y, t)$, kde je navíc t pro reprezentaci času. Jedná se o velmi častý a široce využitelný model. Je ovšem velmi důležité si uvědomit, že ne všechny obrazy mohou být popsány takovouto spojitou funkcí. Například obrazy a signály z magnetické rezonance či ultrazvuku jsou generovány velkými poli nebo vektory dat, která vyžadují hluboké pochopení a jsou často více než třídimenzionální.

2.4.1 Popis obrazu spojitou funkcí

V případě, že se jedná o černobílý vstupní obraz, odpovídají funkční hodnoty spojité funkce hodnotě jasu na daných souřadnicích. Funkční hodnota však nemusí nutně odpovídat pouze jasu. Může zde být uložena například teplota, tlak, či vzdálenost objektu od pozorovatele.

2.4.2 Digitalizace obrazu

Aby mohl být vstupní obraz zpracován počítačem, musí být reprezentován vhodnou diskrétní datovou strukturou. Obraz zachycený senzorem je reprezentován jako spojitá funkce $f(x, y)$. Digitalizace obrazu znamená, že je tato funkce $f(x, y)$ navzorkována do matice M o rozměrech $m \times n$. Kvantizací obrazu je poté každému vzorku přiřazena celočíselná hodnota. Platí zde, že čím menší vzorkování, tím větší matice M bude, čímž dojde i k lepší aproximaci vstupního obrazu.

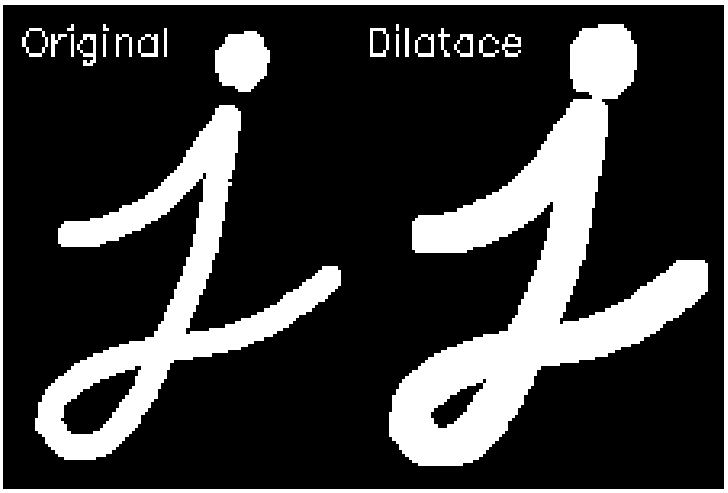
2.5 Morfologické operace

Morfologické operace se opírají o matematickou morfologii, která vychází z teorie množin a využívá vlastnosti bodových množin, přičemž jednu takovou bodovou množinu představuje samotný obraz a druhou určitý strukturní element. Morfologické operace se využívají především pro odstranění šumu, izolování a spojování prvků či nalezení nerovností.

2.5.1 Dilatace

Jedná se o operaci [1], která spočívá v konvoluci obrazu A s určitým jádrem B, jež může mít jakýkoliv tvar a velikost, přičemž nejčastěji se využívá čtverců a kruhů. Jádro B má definovaný kotvící bod, jímž je nejčastěji jeho střed. Jak je jádro B snímáno přes celý obraz, tak se počítá maximální hodnota pixelu, který je překrytý B a pixel obrazu v kotvícím bodu se nahrazuje touto maximální hodnotou. Z toho tedy vyplývá, že operace dilatace způsobuje zvětšení světlých míst. Operaci dilatace můžeme definovat následovně. V této rovnici vystupuje $dst()$ jako práh a $src()$ jako intenzita pixelu.

$$dst(x, y) = max_{(x',y'):element(x',y')\neq 0} src(x + x', y + y') \tag{2.1}$$



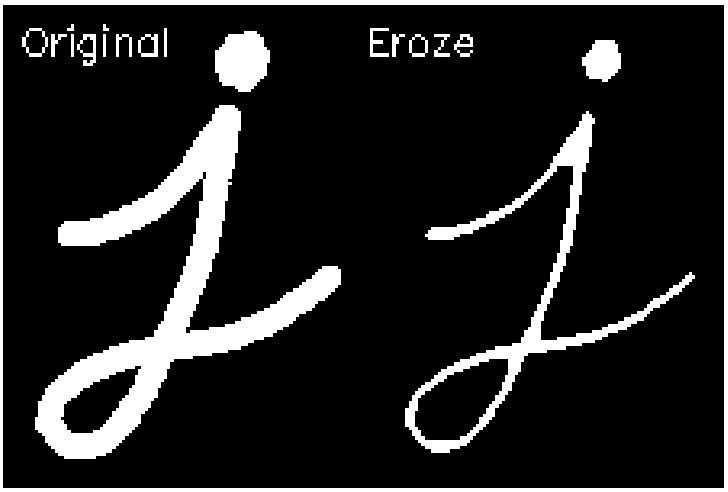
Obrázek 2.2: Příklad operace dilatace[1]

2.5.2 Eroze

Tato operace [1] je sesterskou operací k dilataci s tím rozdílem, že tato operace počítá lokální minimum oblasti dané jádrem B. Jak je jádro B snímáno přes celý obraz, tak se počítá minimální hodnota pixelu, který je překrytý B a pixel obrazu v kotvícím bodu se nahrazuje touto minimální hodnotou, přičemž operace eroze je definována následovně. V této rovnici opět vystupuje $dst()$ jako práh a $src()$ jako intenzita pixelu.

$$dst(x, y) = min_{(x',y'):element(x',y')\neq 0} src(x + x', y + y') \tag{2.2}$$

Analogicky k dilataci nyní víme, že operace eroze způsobuje zmenšení světlých míst v obraze.



Obrázek 2.3: Příklad operace eroze[1]

2.6 Neuronové sítě

Umělé neuronové sítě jsou výpočetním modelem, který je inspirovaný biologickými soustavami, například lidským mozkem. Základní jednotkou umělé neuronové sítě je umělý neuron (dále pouze neuron), jenž je inspirován biologickou strukturou, neuronem, neboli základní buňkou nervové soustavy.

Neuronová síť se poté skládá z velkého počtu vzájemně spojených a provázaných neuronů. Ve výpočetní technice se neuronové sítě uplatňují především v oblastech klasifikace, predikce, rozpoznávání objektu či robotiky a mnohých dalších.

2.6.1 Historie neuronových sítí

Za počátky vzniku neuronových sítí [6] se pokládají 40.léta 20.století, kdy Warren McCulloch a Walter Pittse napsali společnou práci, kde vytvořili velmi jednoduchý model neuronu. Číselné hodnoty parametru tohoto modelu byly převážně z množiny $\{-1, 0, 1\}$. Ukázali tím, že nejjednodušší typy neuronových sítí zvládnou počítat libovolnou aritmetickou či logickou funkci. Ačkoliv autoři článek publikovali, aniž by počítali s bezprostředním praktickým využitím, tak tato práce inspirovala ostatní vědce. Jedním z příkladů je Norbert Wiener, který se touto prací inspiroval při studiu podobnosti činnosti nervové soustavy a systémů výpočetní techniky. Přestože se touto prací z počátku zabývalo spoustu vědců, nedošlo k žádnému převratnému objevu.

V roce 1949 napsal Donald Hebb knihu „The Organization of Behaviour“, ve které přišel s učícím pravidlem pro synapse neuronů. Hebb se inspiroval myšlenkou, že podmíněné reflexy, jež jsou pozorovatelné u živočichů, mohou být vlastnostmi jednotlivých neuronů. Jednalo se o další práci, která inspirovala ostatní, aby se zabývali podobnými otázkami. Typickým příkladem výzkumu z tohoto období je konstrukce prvního neuropočítače Snark v roce 1951, za jehož zrodem stál Marvin Minsky. Ačkoli byl Minsky úspěšný z technického hlediska, nebyl jeho neuropočítač ve skutečnosti nikdy využit v praxi.

Další pokrok přinesl až rok 1957, kdy Frank Rosenblatt vynalezl takzvaný perceptron, který je zobecněním McCullochova a Pittsova modelu neuronu pro reálný číselný obor parametrů. Společně s tímto modelem navrhl učící algoritmus, pro který dokázal, že pro danou sadu tréninkových dat nalezne po

konečném počtu kroků odpovídající váhový vektor parametrů nezávisle na počátečním nastavení. Na základě tohoto výzkumu poté Rosenblatt společně s Charlesem Wightmanem a dalšími sestrojil první úspěšný neuropočítač. Tento počítač měl název „Mark I Perceptron“, který dokázal rozpoznávat znaky, jež byly promítány na světelnou tabuli, ze které byly snímány polem 20x20 fotovodičů. Vstupem do neuronové sítě perceptronů zde byla intenzita obrazových bodů a úkolem této sítě bylo klasifikovat, o který znak se jedná.

Na přelomu 50. a 60.let dochází k úspěšnému rozvoji neurovýpočtů v oblasti návrhů nových modelů neuronových sítí a jejich implementací. Ovšem i přes nesporné úspěchy se obor neuronových sítí stále potýkal se dvěma velkými problémy. Prvním bylo, že většina badatelů přistupovala k neuronovým sítím z experimentálního hlediska a zanedbávala analytický výzkum. A druhým problémem bylo nadšení některých výzkumných pracovníků, které vedlo k velké publicitě neopodstatněných prohlášení, která diskreditovala neuronové sítě v očích odborníků z jiných oblastí a odradila je více se zajímat o neurovýpočty.

Dále probíhal výzkum neuronových sítí spíše ojediněle a izolovaně, převážně mimo území USA. Většina prací byla publikována pod názvy jako: rozpoznávání obrazů, biologické modelování nebo adaptivní zpracování signálů. Jelikož se v počátcích tohoto období, kdy byly neuronové sítě spíše v pozadí, věnovali tématu talentovaní vědci jako např. James Anderson, Kunihiko Fukushima, Harry Klopf, či David Willshaw dochází zde k takzvané renesanci neuronových sítí.

Počátkem 80.let se tedy badatelé v této oblasti osmělili a začali podávat vlastní grantové projekty zaměřené na vývoj neuropočítačů a jejich aplikaci. Další zásluhu na této takzvané renesanci měl i světově uznávaný fyzik John Hopfield. V letech 1982 a 1984 publikoval své výsledky, na kterých ukázal souvislost některých modelů neuronových sítí s modely magnetických materiálů. V roce 1986 poté publikovali své výsledky i badatelé z PDP sku-

piny (Parallel Distributed Procesing), ve kterých popsali učící algoritmus zpětného šíření chyby pro vícevrstvou neuronovou síť. Tento algoritmus je doposud nejpoužívanější učící metodou neuronových sítí. Touto publikací dosáhl zájem o neuronové sítě svého vrcholu.

V roce 1987 se konala první větší konference specializovaná na neuronové sítě. Tato konference se konala v San Diegu v USA a byla zde založena mezinárodní společnost pro výzkum neuronových sítí INNS (International Neural Network Society). Od tohoto roku také začalo mnoho renovovaných univerzit zakládat nové výzkumné ústavy a výukové programy zaměřené na neurovýpočty, přičemž tento trend neustal ani v současné době.

2.7 Konvoluční neuronové sítě

Jednou z podmnožin neuronových sítí jsou konvoluční neuronové sítě [7], jejichž název vychází právě z využití konvolučních a pooling vrstev. Jedná se o speciální vrstvy uvnitř neuronových sítí, díky kterým jsou schopné zpracovávat strukturované vstupy velkých rozměrů, jimiž jsou například obrázky. Využívají mnohem méně parametrů než obecné konvoluční sítě, díky čemuž je snazší takovou síť natrénovat.

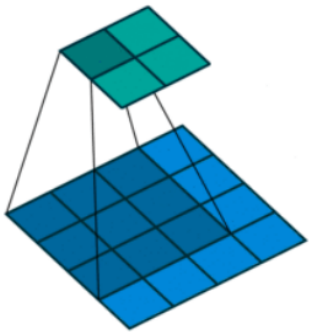
2.7.1 Konvoluční vrstva

Tato vrstva vykonává operaci konvoluce. Operace konvoluce má definováno konvoluční jádro, neboli matici 3x3. Pro operaci konvoluce musíme znát také velikost kroku a šířku rámu vstupní matice vyplněného nulami. V následujících vztazích (2.3) a (2.4) je definována konvoluce pro vstupní matici $m \times n$ s jádrem $M \times N$, jednotkový krok a nulový padding. V těchto vztazích je Y jako výstupní matice, X jako vstupní matice, F jako jádro a $x_{i,j}, y_{i,j}, f_{i,j}$ jako prvky příslušných matic.

$$Y = X \cdot F \tag{2.3}$$

$$y_{i,j} = \sum_{k=1}^{3} \sum_{l=1}^{3} = f_{k,l} x_{i+k,j+l} \tag{2.4}$$

Nespornou výhodou konvolučních vrstev je jejich schopnost rozeznat specifické příznaky při zachované informaci o jejich poloze. Konvoluční vrstvy se také často využívají ke zvýšení počtu kanálů výstupu pro lepší přenos informace. Inverzní operací ke konvoluci je takzvaná dekonvoluce, jejímž úkolem je odhadnout vstupní matici konvoluce ze znalosti výstupní matice.



Obrázek 2.4: Grafické znázornění konvoluce [7].

2.7.2 Batch normalizace

Jedná se o nejvyužívanější formu normalizace [13] pro zvýšení výkonu a zrychlení umělých neuronových sítí. Průběžně se přemapovává výstup tak, aby měl nulový průměr a jednotkovou varianci.

$$N(\mu = 0, \sigma = 1) \tag{2.5}$$

Batch normalizace pomáhá především proti problémům při trénování, jako jsou rozdílné distribuce mezi odezvami.

2.7.3 Učení konvolučních sítí

Stejně jako koncepce neuronových sítí, tak i jejich učení, je inspirováno člověkem. Cílem učení neuronových sítí je optimalizace jejich vnitřních parametrů s cílem minimalizovat chybu predikce. Fáze učení se skládá z mnoha cyklů a iterací, obsahujících dopřednou propagaci, vyhodnocení chyby predikce, zpětnou propagaci a optimalizaci parametrů.

Jelikož má většinou prostor parametrů neuronové sítě vysoký počet parametrů, je úloha hledání globálního minima nelineární funkce na tomto prostoru velmi obtížná. Optimalizační parametry často naleznou lokální minimum, ze kterého se už nedokážou dostat. Učící algoritmy tedy využívají různé metody, které zabrání tomu se do těchto minim dostat.

Další problém nastává, když dostáváme neúplnou informaci o průběhu učení. Tuto informaci poskytuje pouze trénovací a testovací chyba. Ideálně by obě tyto veličiny měly konvergovat k nule, zatímco v reálné situaci dochází k tomu, že v jeden moment ve fázi učení dojde k zastavení klesajícího trendu testovací chyby, která začne opět stoupat, zatímco trénovací chyba bude dále klesat. Tento jev nazýváme jako přetrénování, tedy moment, kdy se síť naučí rozpoznávat příznaky, které jsou obsažené pouze v trénovacích datech.

2.8 Intersection over union

Intersection over union [12] (dále pouze IoU) je vyhodnocovací metrika používaná pro měření přesnosti detektoru objektů na určitém datasetu. IoU se typicky používá pro vyhodnocení výkonu a přesnosti konvolučních neuronových sítí, které detekují objekty. Je ale důležité si uvědomit, že u IoU nejde o algoritmus, který generuje predikce, ale pouze o výsledky.

Abychom mohli tuto metriku vyhodnocení aplikovat, potřebujeme znát skutečnou pozici detekovaného objektu a predikovanou pozici detekovaného objektu. Obě tyto pozice jsou určeny bounding boxem. Výpočet IoU může

být definován následovně.

$$IoU = \frac{průnik}{sjednocení} \tag{2.6}$$

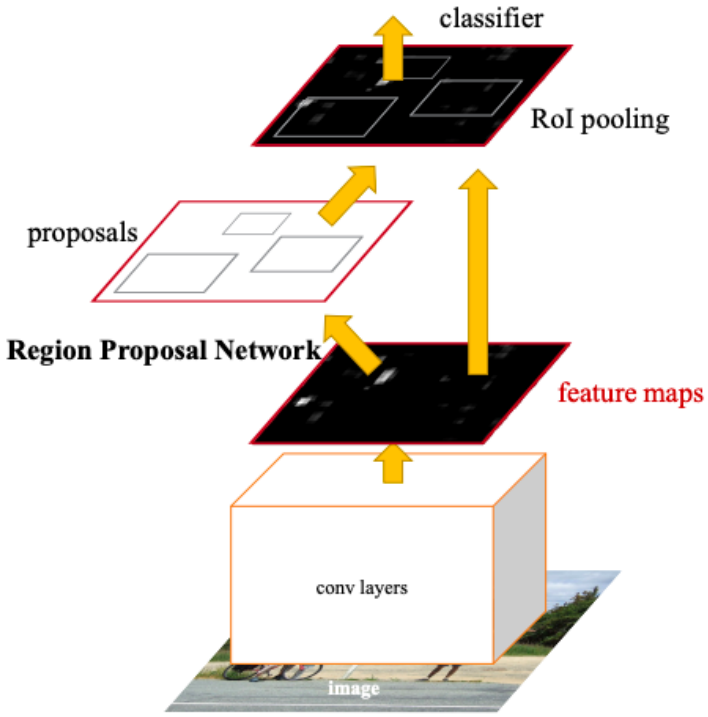
Kde *průnik* je oblast průniku predikovaného a reálného bounding boxu a *sjednocení* je oblast sjednocení obou bounding boxů.

2.9 Faster R-CNN

Faster-RCNN [11] je model neuronové sítě pro detekci objektů. Architektura tohoto modelu se skládá z dalších dvou modulů. První modulem je hluboká plně konvoluční síť, která navrhuje regiony výskytu objektu. Druhým modulem je detekční síť Fast R-CNN, která detekuje objekty na navržených regionech. Celý tento systém tvoří jednotnou neuronovou detekční síť.

2.10 RPN

RPN [11], neboli Region Proposal Network, je síť, jež pracuje s obrazem jakékoli velikosti, který bere jako vstup a na výstupu vrací set regionů s možnými výskyty daného detekovaného objektu. Tento proces je modelován plně konvoluční sítí.



Obrázek 2.5: Proces Faster-RCNN [11]

2.11 YoloF

Yolo je zkratka pro You only look once, což napovídá, že YoloF [9] je jednoduchá síť. Jedná se o rychlou a přímočarou síť s jednoúrovňovou funkcí. YoloF se skládá ze 3 částí: základ, enkodér a dekodér.

Základ je postaven na sítích ResNet[5] a ResNeXt[17]. Všechny modely jsou předtrénované na ImageNet[3]. Výstupem je C5 mapa funkcí jež má 2048 kanálů s downsample ratem 32.

Pro enkodér nejdříve vycházíme z FPN přidáním dvou projekčních vrstev po základu, což má za výsledek mapu funkcí s 512 kanály. Aby výstupní funkce enkodéru zabrala všechny objekty na různých úrovních, je nutné k ní přidat residuální bloky, které jsou složené ze tří po sobě jdoucích konvolucí.

Pro dekodér se přebírá hlavní design RetinaNet[8], jenž se skládá ze dvou úlohově-specifických hlav. První je klasifikační a druhá regresní.

2.11.1 RetinaNet

RetinaNet[8] je jeden z nejlepších jednoetapových modelů, u kterého se potvrdilo, že pracuje dobře s objekty jež jsou malé a hustě zobrazené. Z tohoto důvodu je tento model hojně využíván pro detekci objektů ze satelitních snímků. RetinaNet byla vytvořena za pomoci dvou vylepšení již existujících jednoetapových modelů. Prvním bylo FPN a druhým Focal Loss.

2.11.2 Res2Net

Res2Net[4] je model, který využívá variance na residuálních blocích. Motivací je reprezentace objektu na různých úrovních a měřítkách. Toho je dosaženo novým stavebním blokem pro CNN (Convolutional neural network), který vytváří hiearchické spoje podobné residuálním, v rámci jednoho residuálního bloku. Tento celek pak vytváří víceúrovňové funkce na granulární úrovni a zvyšuje rozsah receptivních polí pro každou neuronovou vrstvu.

3 Příprava dat

V následující části budu popisovat, jaká data jsem použil a jak jsem tato data zpracoval. Jedná se o jednu z nejdůležitějších činností, neboť bez kvalitně připravených dat bych nemohl obdržet dobré výsledky.

3.1 Vstupní data

Vstupními daty k této bakalářské práci jsou videozáznamy z Biomedicínského centra Lékařské fakulty UK v Plzni. Toto centrum je zaměřeno především na výzkum a vývoj v oblasti nahrazování a regenerace orgánů. Konkrétně se jedná o videozáznamy prasat z několika kotců. Celkem jsem měl k dispozici 36 videozáznamů, z nichž jsem využil 21. Tato videa jsem vybíral především na základě aktivity prasat, a to proto abych tím dokázal zaznamenat co nejvíc různých stavů prasat. Dále jsem zařadil i noční videozáznamy a záznamy, kde se v jednom kotci vyskytovalo více prasat. Všechny tyto videozáznamy byly ve formátu mp4. V tabulce 3.2 je vidět celkový počet videí, celkový počet snímků a velikost datasetu. Videa mají dohromady 34 minut.

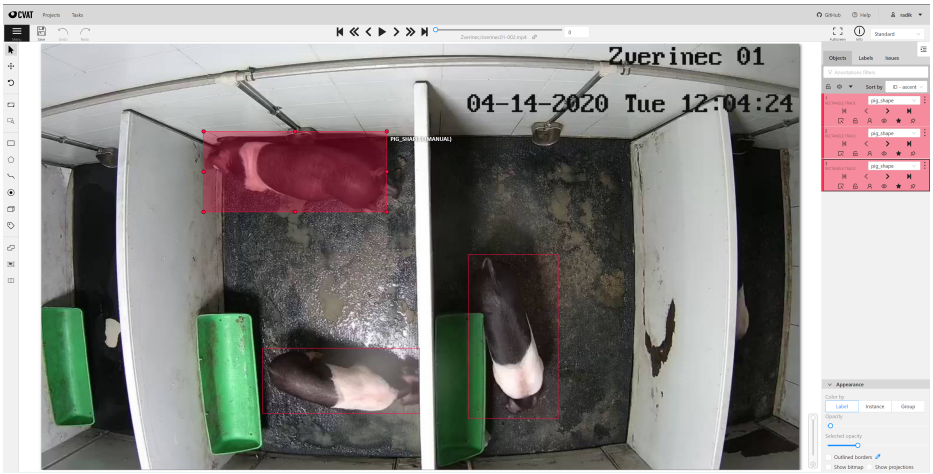
3.2 Zpracování dat

Všechna data jsem zpracovával pomocí webového nástroje CVAT [10]. CVAT, neboli Computer vision annotation tool, je bezplatný a volně přístupný webový nástroj pro anotaci obrázků a videí, jež se využívá k označování dat pro algoritmy počítačového vidění. Tento nástroj v současné době podporuje i strojové učení s učitelem, klasifikaci a segmentaci obrazů a 3D anotaci. Já jsem však využil pouze možnosti anotace videa a vytvoření datasetu.

Celkový počet videí	18
Celkový počet snímků	44 821
Celková velikost datasetů	48.97 GB

Tabulka 3.1: Statistiky anotace

Prvním krokem ve zpracování je nahrání dat do projektu v nástroji CVAT. Tento nástroj poté každé video vezme a rozdělí ho na příslušný počet snímků tak, abych na každém mohl označit jedno či více prasat. Samotná anotace pak už probíhala pouze označením polohy prasete ohraničujícím boxem, díky čemuž poté nástroj zaznamenal polohu prasete pro daný snímek.Celkové statisky týkající se anotace jsou uvedeny v tabulce 3.2.



Obrázek 3.1: Pohled v nástroji CVAT během anotace

Po dokončení všech anotací jsem využil další z funkcí nástroje CVAT, kterou je možnost vyexportování datasetu ve zvoleném formátu. Pro budoucí kompatibilitu jsem zvolil formát COCO [16]. Jedná se o formát datasetu pro detekci a segmentaci velkých objektů. COCO formát datasetu je určen svou vlastní adresářovou strukturou.

Dataset, který je uložen v této adresářové struktuře, poté obsahuje jak snímky, tak i samotný soubor typu JSON, který obsahuje všechny anotace,

metadata, kategorie a všechny ostatní informace, které jsou potřeba. CVAT nám umožňuje exportovat jak jednotlivé datasety ke konkrétnímu videozáznamu, tak i jeden velký dataset pro všechny videozáznamy. Této funkce jsem využil při vytváření adresářové struktury pro trénování. Videozáznamy pro testovaní a validaci jsem vyexportoval zvlášť do dvou různých datasetů a zbylé videozáznamy jsem vyexportoval do jednoho velkého datasetu. Videa byla vyexportována v poměru 15-1-2, tedy 15 videí pro trénink, 1 na validaci a 2 na testování. Adresářová struktura je zobrazena na obrázku (3.2). Trénovací množina slouží k samotnému učení sítě, testovací množina by měla reprezentovat reálná data, na která se síť nenaučila. V jednotlivých iteracích poté počítám chybu na validačních datech. Chyba testovacích dat je ukazatelem schopnosti sítě generalizovat.

```
<dataset_dir>/
 coco_annotations/
        train/
            images/
                <filename_1>.png
                ...
                <filename_n>.png
            annotations/
                train.json
        valid/
            images/
                <filename_1>.png
                ...
                <filename_m>.png
            annotations/
                valid.json
        test/
            images/
                <filename_1>.png
                ...
                <filename_k>.png
            annotations/
                test.json
```

Obrázek 3.2: Adresářová struktura

Poslední částí v přípravě dat je vytvoření konfiguračního souboru, který později využijeme při trénování připravených dat. Ve výpočetní technice se konfiguračními soubory nazývají takové soubory, jež se používají pro konfiguraci parametrů a počátečního nastavení pro některé výpočetní programy. V mém konkrétním případě se jedná o python skript specifikující počet detekovaných objektů a jejich názvů, formát datasetu a absolutní cesty k datasetům na trénink, test, validaci a předtrénované neuronové síti pro zvýšení výkonnosti.

3.2.1 OpenMMLab

OpenMMLab [2] je open-source python knihovna uzpůsobená pro metody počítačového vidění. K prvnímu vydání této knihovny došlo v roce 2018 a od toho roku se na vývoji této platformy podílelo více než 800 přispěvatelů, jejichž hlavní vizí je vytvořit jednu z nejvyužívanějších open-source platforem pro počítačové vidění. OpenMMLab se skládá z více podprojektů, které se zaměřují na různá specifická témata. Já v této práci využiji podprojekt MMDetection [2].

3.2.2 MMDetection

MMDetection [2] je toolbox pro detekci objektů, jež obsahuje velké množství detekčních a segmentačních metod pracujících s PyTorch. Jedná se o postupně rozvíjející se platformu, která zahrnuje populární detekční metody a modely. Součástí nejsou pouze skripty pro trénink a inferenci, ale i váhy pro více než 200 modelů sítí. Mezi největší výhody tohoto toolboxu patří například modulární design a vysoká efektivita.

3.2.3 MetaCentrum

Jedná se o aktivitu sdružení CESNET, která se věnuje především provozu a rozvoji gridové infrastruktury v České republice. MetaCentrum [15] provozuje a spravuje distribuovanou výpočetní infrastrukturu, jež se skládá z vlastních i svěřených výpočetních a úložných kapacit akademických center v celé České republice. Všichni registrovaní uživatelé mají možnost bezplatného využití aplikačních programů a výpočetního času.

3.2.4 OpenCV

OpenCV [1] je open-source knihovna, která obsahuje několik set různých algoritmů pro počítačové vidění v reálném čase. Tuto knihovnu je možné využít v prostředí jazyků C/C++, Python a Java.

4 Experimenty

V této kapitole budu popisovat provedené experimenty. Začnu popisem experimentu, u něhož jsem nevyužil metod neuronových sítí, ale klasických metod počítačového vidění. Ve všech dalších experimentech už použiji pouze metod neuronových sítí.

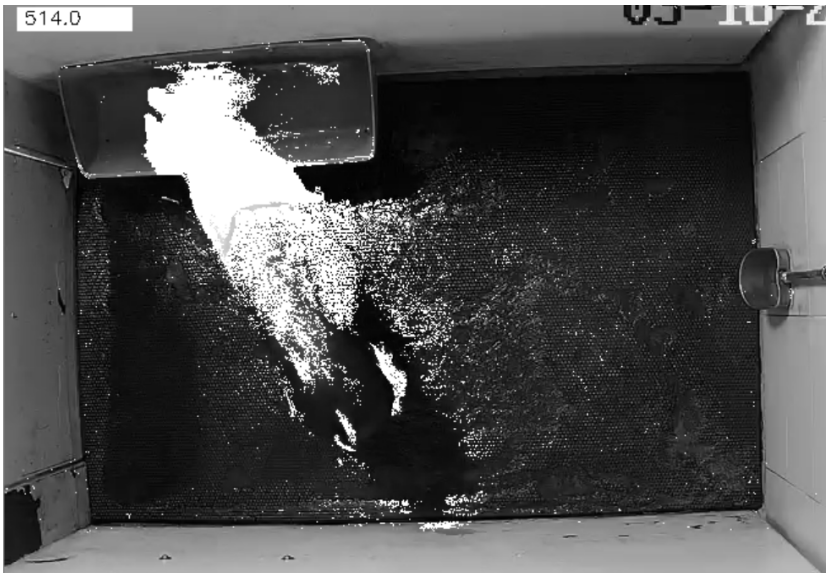
4.1 Oddělování pozadí

V prvním experimentu jsem využil metod oddělování pozadí v kombinaci s morfologickými operacemi. Pro tento experiment nebylo nutné předem zpracovávat vstupní data. Vstupními daty v tomto experimentu jsou pouze videozáznamy. Využil jsem zde knihovny OpenCV.

Obrázek 4.1: Ukázka vstupního videa

4.1.1 Metoda oddělování pozadí

Začal jsem napsáním krátkého programu, který nejdříve rozdělil vstupní video na určitý počet snímků v závislosti na jeho délce. Pro oddělení pozadí jsem využil metody oddělování pozadí. Jedná se o velmi běžnou a často využívanou metodu. Každý snímek je využit pro výpočet masky popředí a pro aktualizaci pozadí. Tímto procesem mi vznikla výstupní matice obsahující pouze prvky $\{0, 255\}$, přičemž 0 definuje černý pixel a 255 bílý. Celý tento proces jsem sledoval průběžným vykreslováním těchto matic ve formě černobílého videozáznamu, kde je pohybující se objekt reprezentován bílými pixely.



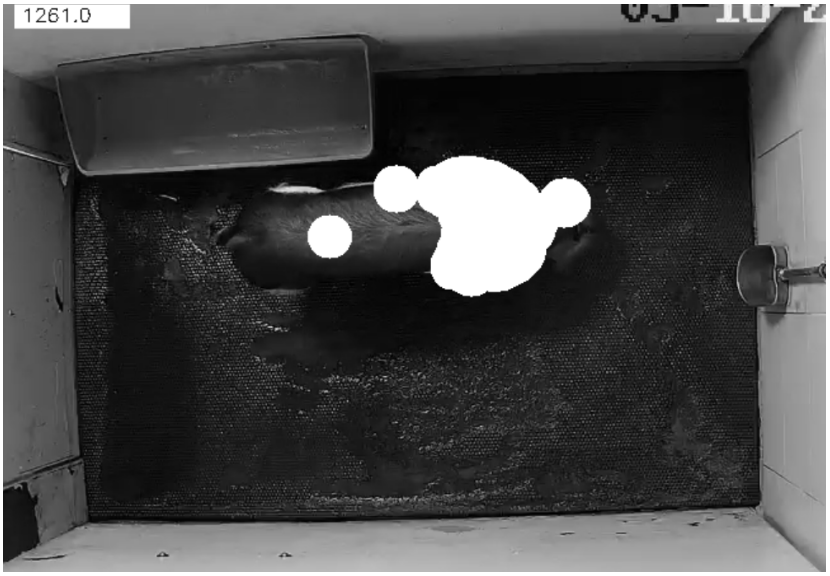
Obrázek 4.2: Ukázka výstupního videa

4.1.2 Výsledky metody oddělování pozadí

Jelikož byl výstup poměrně zašuměný a pohybující se objekt svým obrysem příliš neodpovídal realitě, využil jsem v dalším kroku morfologických operací, především pak erozi a dilataci. Tyto dvě operace jsem poté aplikoval na každý snímek. Z výstupu pak bylo jasně vidět, že operátor eroze velkou

mírou snížil šum a operátor dilatace dokázal zvýraznit obrys pohybujícího se objektu tak, že více odpovídal realitě.

Nicméně tento postup se ukázal jako ne příliš vhodný pro tuto úlohu především ze dvou důvodů. Prvním důvodem je obecnost algoritmu. Ačkoli algoritmus detekuje pohybující se prasata dobře, tak detekuje i všechny ostatní pohybující se objekty, čímž by mohlo v dalších částech práce dojít ke komplikacím. Druhým a hlavním problémem je, že tento algoritmus detekuje pouze pohybující se objekty a nedokáže tedy detekovat prasata, jež se nepohybují, a docházelo by tak ke ztrátě informací. Z těchto dvou důvodů jsem se rozhodl v tomto přístupu nepokračovat a pokračovat dále především s využitím neuronových sítí.



Obrázek 4.3: Výstupní video po aplikaci eroze a dilatace

4.2 Faster-RCNN

Ve druhém experimentu jsem pracoval především s Python toolboxem MMDetection, který jsem využil pro trénink neuronové sítě vycházející ze sítě Faster-RCNN. Pro trénink neuronových sítí jsem využil služeb MetaCentra.

Epocha	tst-0.50:0.95	tst-0.50	tst-0.75	val-0.50:0.95	val-0.50	val-0.75
1	0.70196	0.98765	0.91692	0.7020	0.9880	0.9200
2	0.66997	0.98367	0.83775	0.6700	0.9840	0.8380
3	0.66142	0.98530	0.85568	0.6610	0.9850	0.8560
4	0.63958	0.98330	0.82485	0.6400	0.9830	0.8250
5	0.69795	0.98783	0.91319	0.6980	0.9880	0.9130
6	0.65804	0.98248	0.83494	0.6580	0.9820	0.8350
7	0.66071	0.97892	0.85782	0.6610	0.9790	0.8580
8	0.69493	0.98716	0.91021	0.6950	0.9870	0.9100
9	0.69709	0.98645	0.92992	0.6970	0.9860	0.9300
10	0.68531	0.98653	0.90461	0.6900	0.9870	0.9050
11	0.68863	0.98556	0.90327	0.6890	0.9860	0.9030
12	0.68877	0.98614	0.91501	0.6890	0.9860	0.9150

Tabulka 4.1: Statistiky IoU trénování Fatsrer-RCNN

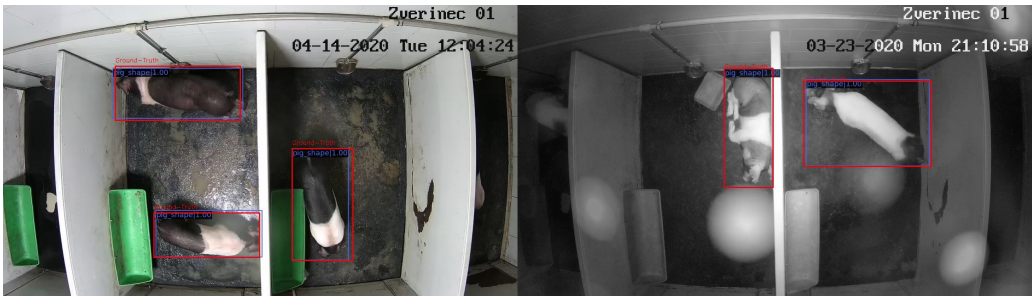
4.2.1 Trénování Faster-RCNN

Při trénování na vstupních datech jsem využil model Faster-RCNN. Konfigurací byl nastaven optimalizátor na typ SGD s učící konstantou na 0,02. Třída určená k detekci byla pouze jedna a to třída *pig_shape*. Celkem proběhlo 12 epoch, každá po 21 162 iteracích. Trénování trvalo celkem 10 hodin 32 minut a 5 sekund. V tabulce 4.1 je možné vidět statistiky celého trénování. Využívám zde metriky Intersection over Union.

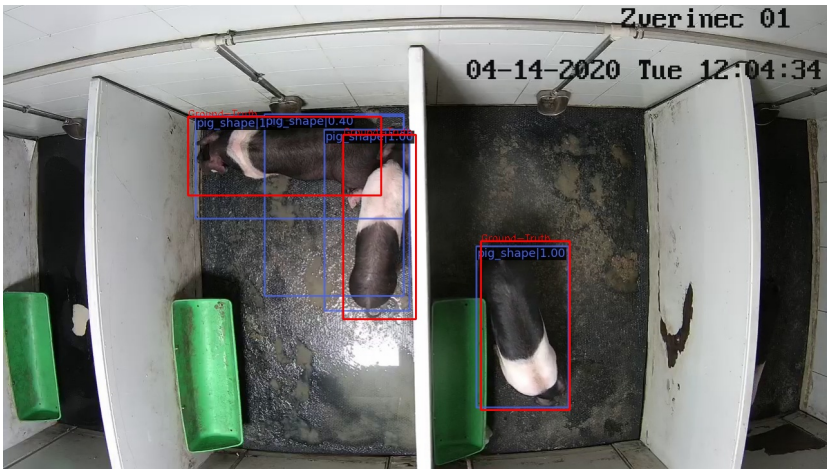
4.2.2 Výsledky Faster-RCNN

Jak je z tabulky 4.1 zřetelné, že vzhledem ke snižující se přesnosti dochází k přetrénování. I přesto jsou výsledky velmi uspokojivé, neboť metrika IoU = 0.50 dosáhla v poslední epoše 98% a IoU=0.75 dosáhla 91%. Po dotrénování

jsem provedl inferenci na testovacích datech. Jednalo se o video se třemi prasaty za dne a video se dvěma prasaty za tmy. Na denním videu funguje detekce výborně za předpokladu, že se prasata nedotýkají, pokud se ovšem dotýkají detekuje síť místy o jedno prase víc. Ovšem toto prase je detekováno pouze s pravděpodobností v rozmezí od 30% do 40%, což by se v následné nadstavbě dalo vyfiltrovat. Tento šum přisuzuji především jasně danému zbarvení prasat, kdy se střídá černý pruh s bílým a jsou-li prasat vhodně postavená může dojít k šumu. Tento šum je možné vidět na obrázku 4.6. Na videu za tmy funguje síť skvěle, vzhledem k tomu, že v trénovacích datech byla pouze dvě noční videa. Bohužel nebyla poskytnuta noční videa, kde bylo v jednom kotci více než jedno prase a nedochází tedy k jejich překrývání. Ovšem největším problémem na nočních videích, především pro předchozí metodu, byl velký šum způsobovaný částečkami prachu poletujícími před kamerou. Tento šum detektor dokonale zanedbal a detekoval pouze a jen pohyb prasat, který detekovat měl.



Obrázek 4.4: Správná detekce sítě Faster-RCNN



Obrázek 4.5: Chybná detekce sítě Faster-RCNN - naddetekce

4.3 YoloF

V tomto experimentu jsem využil stejného frameworku jako v předchozím experimentu. Síť jsem trénoval opět na MetaCentru, ovšem tentokrát na síti YoloF.

4.3.1 Trénování YoloF

Pro trénování jsem využil model YoloF. Optimalizátor byl opět typu SGD s učící konstantou 0,12. Třída pro detekci byla pouze jedna a to *pig_shape*. Trénování bylo nastaveno na 12 epoch, každá po 5 291 iteracích. Trénování trvalo celkem 5 hodin 18 minut a 23 sekund. Statistiky trénování jsou k nahlédnutí v tabulce **??**

4.3.2 Výsledky YoloF

Vzhledem k jednoduchosti sítě jsem v tomto experimentu nepředpokládal výborné výsledky. Už po shlédnutí statistik 4.2 z trénování je jasné, že opět dochází k přetrénování, a také že síť nebude detekovat vše, co by měla. Inferenci jsem provedl na stejných testovacích videích jako v předchozím

experimentu. Na denním videu je jasně vidět, že síť detekuje prasata, jež stojí diagonálně vůči pohledu kamery, poměrně efektivně, ale prasata, jež jsou vůči tomuto pohledu horizontálně postavená, detekuje buď špatně nebo vůbec. Přiblíží-li se k sobě více prasat, dochází opět k detekci více prasat, než se na snímku nachází. V tomto případě detekuje místy až o dvě prasata více. I přes tento šum ale detekuje i prase, jež vůbec nebylo označeno při anotaci. Na nočním záznamu tato síť dosahuje o dost lepších výsledků. Pokud jsou prasata natočena diagonálně vůči pohledu kamery, jsou detekována téměř bezchybně a velmi přesně. Pokud se ovšem natočí horizontálně, dochází k občasné detekci jednoho prasete navíc, avšak toto prase je opět detekováno s velmi malou pravděpodobností. Lepší výsledky na nočním záznamu přisuzuji především nižšímu počtu detekovaných prasat, jež jsou v oddělených kotcích.

Epocha	tst-0.50:0.95	tst-0.50	tst-0.75	val-0.50:0.95	val-0.50	val-0.75
1	0.25027	0.74606	0.06681	0.2500	0.7460	0.0670
2	0.22576	0.69834	0.03504	0.2260	0.6980	0.0350
3	0.15586	0.48365	0.01875	0.1560	0.4840	0.0190
4	0.28905	0.78244	0.06995	0.2890	0.7820	0.0700
5	0.25049	0.80797	0.05112	0.2500	0.8080	0.0510
6	0.20222	0.61074	0.07437	0.2020	0.6110	0.0740
7	0.12773	0.44989	0.02547	0.1280	0.4500	0.0250
8	0.20558	0.57306	0.06680	0.2060	0.5730	0.0670
9	0.32228	0.86118	0.10452	0.3220	0.8610	0.1050
10	0.30278	0.84508	0.07871	0.3030	0.8450	0.0790
11	0.32107	0.87831	0.08845	0.3210	0.8780	0.0880
12	0.30713	0.85479	0.08599	0.3070	0.8550	0.0860

Tabulka 4.2: Statistiky IoU trénování YoloF

Obrázek 4.6: Správná detekce sítě YoloF

Obrázek 4.7: Chybná detekce sítě YoloF - nepřesné označení některých prasat

4.4 RetinaNet

Ve čtvrtém experimentu pokračuji v používání frameworku MMDetection. Při trénování budu využívat modelu sítě RetinaNet.

4.4.1 Trénování RetinaNet

Pro tuto síť jsem využil model RetinaNet. Nejdříve jsem využil optimalizátoru SGD s učící konstantou 0,01. Bohužel se ukázalo, že učící konstanta byla příliš velká a trénování divergovalo, což způsobilo nenatrénování, a už po první epoše síť nebyla schopná cokoli detekovat. Proto jsem učící konstantu

nastavil na 0,0001, což způsobilo konvergenci a dotrénování sítě. Proběhlo celkem 12 epoch každá po 21 162 iteracích. Trénování trvalo 24 hodin 57 minut a 5 sekund.

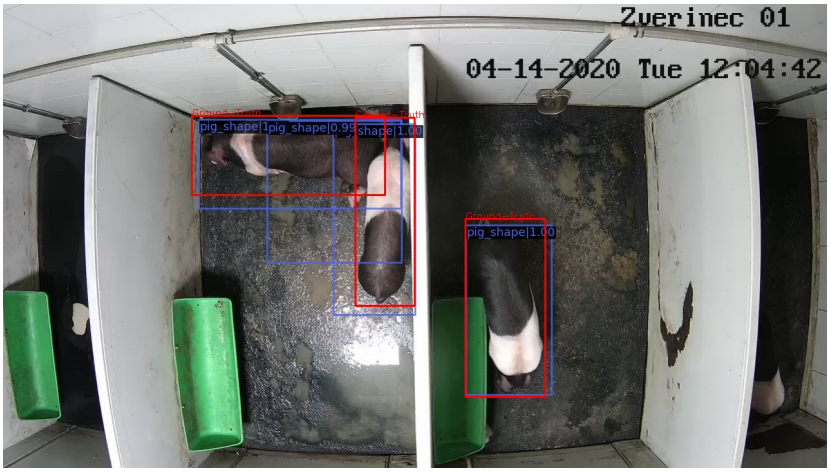
4.4.2 Výsledky RetinaNet

Při pohledu na tabulku 4.3 je vidět, že dochází k přetrénování. Ovšem i přes svou vlastnost lepšího detekování především menších objektů jsou výsledky velmi uspokojivé. V denním videu je jasně vidět, že za předpokladu rozptýlení prasat dochází k téměř bezchybné detekci. Problém opět nastává v moment, kdy se k sobě prasata přiblíží, neboť dochází k detekci jednoho prasete navíc. Z nočního videa dostávám skvělé výsledky, neboť se síti povedlo detekovat i neoznačené prase v pozadí.

Epocha	tst-0.50:0.95	tst-0.50	tst-0.75	val-0.50:0.95	val-0.50	val-0.75
1	0.51412	0.98844	0.38029	0.5140	0.9880	0.3800
2	0.54122	0.98805	0.46071	0.5410	0.9880	0.4610
3	0.56000	0.98809	0.55420	0.5600	0.9880	0.5540
4	0.54781	0.98791	0.51579	0.5480	0.9880	0.5160
5	0.53472	0.98788	0.49006	0.5350	0.9880	0.4900
6	0.57221	0.98786	0.63632	0.5720	0.9880	0.6360
7	0.55523	0.98803	0.54267	0.5550	0.9880	0.5430
8	0.54731	0.98788	0.51948	0.5470	0.9880	0.5190
9	0.55114	0.98786	0.55725	0.5510	0.9880	0.5570
10	0.54943	0.98784	0.54516	0.5490	0.9880	0.5450
11	0.55195	0.98786	0.56305	0.5520	0.9880	0.5630
12	0.55082	0.9878	0.56250	0.5510	0.9880	0.5620

Tabulka 4.3: Statistiky IoU trénování RetinaNet

Obrázek 4.8: Správná detekce sítě RetinaNet



Obrázek 4.9: Chybná detekce sítě RetinaNet - naddetekce

4.5 Res2Net

V posledním experimentu opět využiji MMDetection frameworku. Při trénování využiji model sítě Res2Net [4].

4.5.1 Trénování Res2Net

Pro trénování mé poslední sítě jsem využil modelu Res2Net. Vstupní třídou pro detekcti byla pouze třída *pig_shape*. Využil jsem opět optimalizátor typu SGD. Při prvním trénování této sítě byla učící konstanta nastavena na

0,01, což se již při první epoše ukazuje jako příliš vysoká hodnota a trénování diverguje. Při druhém trénování jsem nastavil hodnotu učící konstanty na 0,0001. Při této hodnotě trénování již dokonvergovalo. Proběhlo celkem 24 epoch každá po 21 162 iteracích. Trénování trvalo celkem 15 hodin 26 minut a 39 sekund.

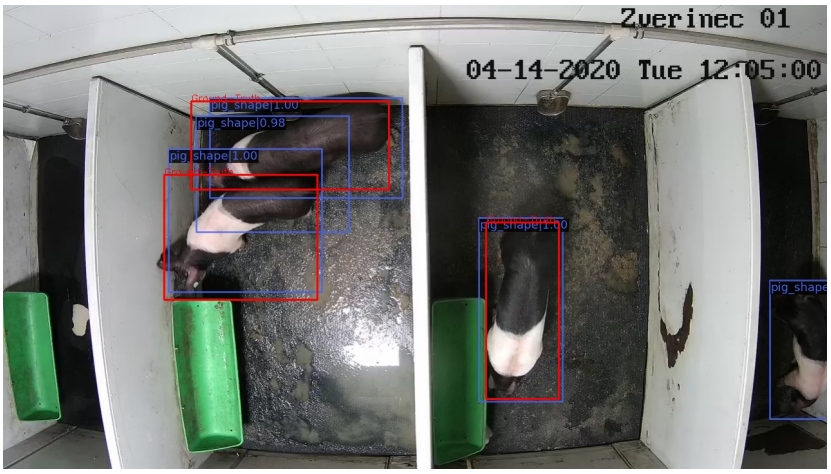
4.5.2 Výsledky Res2Net

Jelikož je model Res2Net složitější než Retinanet a YoloF, očekával jsem lepší výsledky. Toto se potvrdilo už při prvotním pohledu na statistiky trénování (tabulka 4.5), ačkoli i zde je vidět, že dochází k přetrénování. Inferenci jsem nejdříve provedl na denním videu. Opět je vidět, že při rozptylu prasat je síť v podstatě bezchybná, detekuje vše a přesně. Při shlukování prasat však znovu dochází k detekci více prasat. V tomto případě vždy jen o jedno prase. Z obrázku je jasné, že toto je zapříčiněné především zbarvením a postavením prasat. V nočním videu je detekce bezchybná a velmi přesná. Síť skvěle ignorovala rušení prachu a detekovala pouze prasata, co měla. Celkově se tato síť ukázala jako velmi efektivní.

Obrázek 4.10: Správná detekce sítě Res2Net

Epocha	tst-0.50:0.95	tst-0.50	tst-0.75	val-0.50:0.95	val-0.50	val-0.75
1	0.59157	0.98827	0.66065	0.5920	0.9880	0.6610
3	0.59545	0.98652	0.69145	0.5950	0.9870	0.6910
5	0.59622	0.98680	0.72884	0.5960	0.9870	0.7290
7	0.59479	0.97685	0.70184	0.5950	0.9770	0.7020
9	0.59679	0.97692	0.72380	0.5970	0.9770	0.7240
11	0.61331	0.97609	0.75646	0.6130	0.9760	0.7560
13	0.58319	0.97417	0.68451	0.5830	0.9740	0.6850
15	0.60074	0.96686	0.74696	0.6010	0.9670	0.7470
17	0.60350	0.97509	0.75297	0.6030	0.9750	0.7530
19	0.59615	0.97381	0.73648	0.5960	0.9740	0.7360
22	0.59740	0.97471	0.72676	0.5970	0.9750	0.7270
24	0.59495	0.97403	0.72202	0.5950	0.9740	0.7220

Tabulka 4.4: Statistiky IoU trénování Res2Net



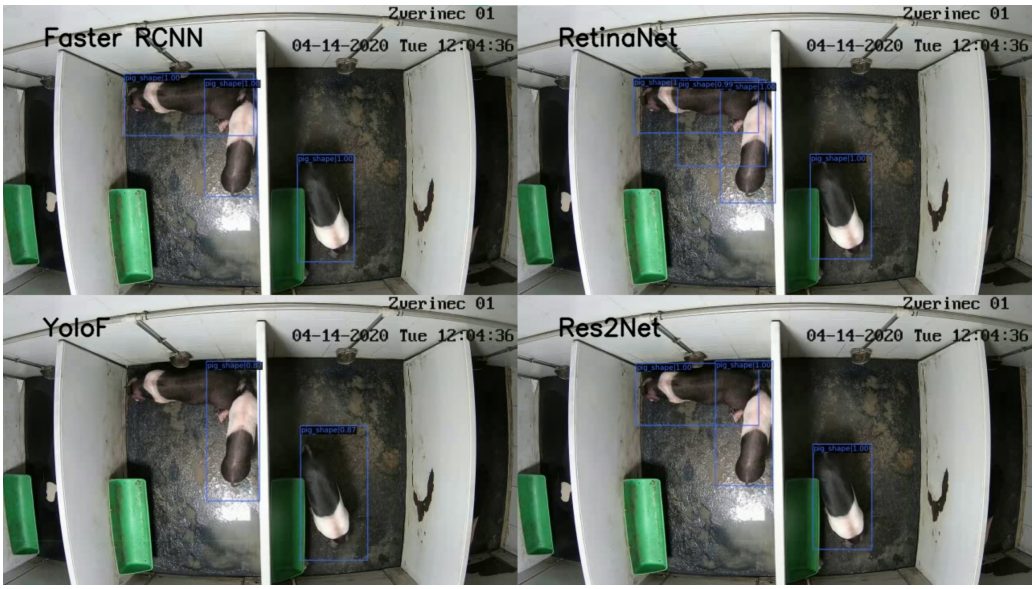
Obrázek 4.11: Chybná detekce sítě Res2Net

4.6 Srovnání experimentů

V závěru této kapitoly budu srovnávat jednotlivé experimenty. V prvním experimentu jsem využil klasických metod počítačového vidění, konkrétně metody oddělování pozadí. Ačkoli tato klasická metoda detekovala prasata poměrně dobře, docházelo zde k bohužel k velkému okolnímu šumu a noční videa zde dopadla jednoznačně nejhůře. V dalších experimentech jsem už využíval jen metod neuronových sítí. Zde se ukázalo, že při využití těchto metod dochází k velkému zvýšení kvality detekce jak při denních videích, tak hlavně při těch nočních, přestože ve všech případech došlo k přetrénování. Tři ze čtyř mnou natrénovaných sítí se ukázaly jako bezchybné za tmy, na jednom konkrétním videu. Na denních záběrech, kde docházelo i ke shlukování prasat, se jako nejefektivnější ukázala síť natrénovaná na modelu sítě Faster-RCNN. Naopak nejhorší výsledky přinesla síť trénovaná na modelu sítě YoloF. Ačkoli se tato síť ukázala jako nejhorší, co se detekce týče, byla natrénována za nejkratší čas. Sítě natrénované na modelech Res2Net a RetinaNet se obě ukázaly jako poměrně efektivní, avšak obě měly stejný problém pomyslné naddetekce s vysokými pravděpodobnostmi výskytu, což by mohl být v budoucnu problém při dalším zpracování. Celkově si troufám říct, že všechny experimenty přinesly uspokojivé výsledky, se kterými se bude dát v budoucnu dále pracovat.

Síť	tst-0.50:0.95	tst-0.50	tst-0.75	val-0.50:0.95	val-0.50	val-0.75
Faster-RCNN	**0.68877**	**0.98614**	**0.91501**	**0.6890**	**0.9860**	**0.9150**
YoloF	0.30713	0.85479	0.08599	0.3070	0.8550	0.0860
RetinaNet	0.55082	0.9878	0.56250	0.5510	0.9880	0.5620
ResNet	0.59495	0.97403	0.72202	0.5950	0.9740	0.7220

Tabulka 4.5: Srovnání IoU přesnosti jednotlivých sítí



Obrázek 4.12: Výsledné srovnání detekčních sítí

5 Závěr

Hlavními cíli této bakalářské práce bylo seznámení s metodami počítačového vidění, natrénování několika neuronových sítí pro detekci prasat a jejich následné otestování. V první části jsem představil počítačové vidění, jakožto vědní obor a poté i několik metod určených pro práci se vstupním obrazem a detekci objektů v obraze. Velkou částí této kapitoly byla podkapitola o neuronových sítích, kde jsem se soustředil především na konvoluční neuronové sítě. V této části jsem také představil modely konkrétních neuronových sítí, které jsem následně použil při trénování.

Ve druhé části jsem představil vstupní dataset poskytnutý Biomedicínským centrem. Jednalo se o 21 videí, která jsem následně zpracoval pomocí nástroje CVAT. Dále jsem popsal samotný proces předzpracování vstupních dat tak, aby byla použitelná v následujících experimentech. Nakonec jsem představil všechny nástroje, jež jsem využil jak při zpracovávání dat, tak při následujících experimentech.

V poslední, třetí části, jsem představil všechny provedené experimenty a porovnal jsem dosažené výsledky. V prvním experimentu jsem využil metody oddělování pozadí. Pro tento experiment nebylo nutné žádné předzpracování vstupních dat, neboť jsem využil funkcí frameworku OpenCV. Výstupem tohoto experimentu bylo video, kde se pohybující se prase zobrazilo jako světlá silueta. Tento experiment se ukázal jako neúspěšný, jelikož tato metoda detekovala nejen prasata, ale všechny pohybující se objekty, a také zde docházelo k velkému šumu v nočních videích. Z tohoto důvodu jsem dále využil pouze metod neuronových sítí. Celkem jsem využil čtyři různé modely neuronových sítí pro následující trénování. Při trénování jsem využil frameworku MMDetection, což je odnož OpenMMLab, a MetaCentra, které poskytlo výpočetní kapacitu tak, aby trénování proběhlo v unesitelném čase. Pro trénování jsem

využil modely sítí Faster-RCNN, YoloF, RetinaNet a ResNet. Jako testovací videa jsem zvolil jedno denní video, kde docházelo ke shlukování prasat, a jedno noční video, kde docházelo k velkému šumu, jenž byl způsoben poletujícími částečkami prachu těsně před čočkou videokamery. Největším sdíleným problémem všech natrénovaných sítí byla nadbytečná detekce při shlukování prasat. Tento problém přisuzuji především specifickému zbarvení prasat a malému množství vstupních videí v nichž se prasata shlukovala. Za úspěch bych označil dokonalé vyrušení šumu v nočních videích. Na tomto videu bylo také velmi dobře vidět, že sítě natrénované na modelech Faster-RCNN, RetinaNet a Res2Net detekují v podstatě bezchybně, když je v každém kotci pouze jedno prase a nedochází ke shlukování. Celkově se jako nejúspěšnější a nejefektivnější síť ukázala ta natrénovaná na modelu Faster-RCNN. To, že se jedná o nejúspěšnější síť, je zřetelné, především pokud se zaměříme na první problém nadbytečné detekce. Pokud tato síť detekuje více než opravdový počet prasat, tak je vždy detekuje s nízkou pravděpodobností, a proto by v následujícím zpracování mělo být možné tuto nadbytečnou detekci odfiltrovat. Jako nejhorší vyšla síť natrénovaná na modelu sítě YoloF. Tato síť téměř vždy při shlukování prasat detekovala více, než by měla, a zároveň vždy s vysokou pravděpodobností. Ani na nočním videu se neosvědčila jako velmi kvalitní nástroj pro detekci, neboť generované bounding boxy nad detekovanými prasaty byly místy až 1,5krát větší než originálně označené bounding boxy.

Celkové výsledky byly velmi uspokojivé, neboť se mi povedlo natrénovat tři použitelné sítě pro detekci. V budoucnu bych rád v této práci pokračoval, tak aby tyto výsledky byly použitelné v praxi a byly užitečné pracovníkům biomedicínského centra, jakožto nástroj pro efektivní hlídání prasat v kotcích. Rád bych v počátku vytvořil uživatelské rozhraní, v němž by docházelo ke sledování pohybu prasat, které bych následně rozšířil o další funkce, jako je například sledování nachozených metrů či rozpoznání zda prase dýchá.

Literatura

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

[2] Chen, K. et al. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*. 2019.

[3] Deng, J. et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, s. 248–255. Ieee, 2009.

[4] Gao, S.-H. et al. Res2Net: A New Multi-scale Backbone Architecture. *IEEE TPAMI*. 2020. doi: 10.1109/TPAMI.2019.2938758.

[5] He, K. et al. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*. 2015.

[6] Horkỳ, L. – Břinda, K. Neuronové sítě, 2009.

[7] Jiří, Z. Konvoluční neuronové sítě pro klasifikaci objekt z LiDARovỳch dat. B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2019.

[8] Lin, T.-Y. et al. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[9] Nguyen, P. et al. Page Object Detection with YOLOF. In *2021 8th NAFOSTED Conference on Information and Computer Science (NICS)*, s. 205–210. IEEE, 2021.

[10] Pangal, D. J. et al. A guide to annotation of neurosurgical intraoperative video for machine learning analysis and computer vision. *World Neurosurgery*. 2021, 150, s. 26–30.

[11] Ren, S. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*. 2015, abs/1506.01497. Dostupné z: `http://arxiv.org/abs/1506.01497`.

[12] Rezatofighi, H. et al. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, s. 658–666, 2019.

[13] Santurkar, S. et al. How does batch normalization help optimization? *Advances in neural information processing systems*. 2018, 31.

[14] Sonka, M. – Hlavac, V. – Boyle, R. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[15] Šustr, Z. et al. Metacentrum, the czech virtualized ngi. In *EGEE Technical Forum*, 2009.

[16] Veit, A. et al. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*. 2016.

[17] Xie, S. et al. Aggregated Residual Transformations for Deep Neural Networks. *arXiv preprint arXiv:1611.05431*. 2016.