VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HLUBOKÉ NEURONOVÉ SÍTĚ PRO DETEKCI ANOMÁLIÍ PŘI KONTROLE KVALITY

DEEP NEURAL NETWORKS FOR DEFECT DETECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ JUŘICA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání diplomové práce

Student: **Juřica Tomáš, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Hluboké neuronové sítě pro detekci anomálií při kontrole kvality**
Deep Neural Networks for Defect Detection
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy teorie neuronových sítí, konvolučních neuronových sítí a zpětné propagace chyb.
2. Vytvořte si přehled o současných metodách pro extrakci příznaků a vizualizaci anomálií vhodných pro detekci anomálií při kontrole kvality.
3. Vyberte konkrétní metody a aplikujte je na úlohu detekce a vizualizace anomálií.
4. Implementujte navrženou metodu s využitím vhodných nástrojů a proveďte experimenty nad vhodnou datovou sadou.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(September):1-58,2009
- Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. ICML, 32:647-655, 2014.

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

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Cílem této práce je automatizovat detekci defektů při průmyslové výrobě plastových karet. Typickým defektem vzniklým při takovéto výrobě je kontaminace prachovými částicemi či vlasem. Hlavními výzvami, které v této práci řeším, je malý počet dostupných dat (214 karet), velmi malá plocha defektů v kontextu celé karty (průměrně 0,0068 % plochy karty) a zároveň velice rozmanité a komplexní pozadí, ve kterém defekty hledám. Realizaci úkolu jsem dosáhl za použití detekčního algoritmu Mask R-CNN a rozšíření datové sady pomocí namodelování vzhledu typických defektů a vytvoření syntetického datasetu o počtu 20 000 obrázků, na kterém jsem detektor natrénoval. Takovýmto způsobem jsem dosáhl 0,83 AP při IoU rovno 0,1 na testovací části původní datové sady.

Abstract

The goal of this work is to bring automatic defect detection to the manufacturing process of plastic cards. A card is considered defective when it is contaminated with a dust particle or a hair. The main challenges I am facing to accomplish this task are a very few training data samples (214 images), small area of target defects in context of an entire card (average defect area is 0.0068 % of the card) and also very complex background the detection task is performed on. In order to accomplish the task, I decided to use Mask R-CNN detection algorithm combined with augmentation techniques such as synthetic dataset generation. I trained the model on the synthetic dataset consisting of 20 000 images. This way I was able to create a model performing 0.83 AP at 0.1 IoU on the original data test set.

Klíčová slova

Hluboké neuronové sítě, detekce anomálií, detekce defektů, kontrola kvality, Mask R-CNN

Keywords

Deep neural networks, anomaly detection, defect detection, quality control, Mask R-CNN

Citace

JUŘICA, Tomáš. *Hluboké neuronové sítě pro detekci anomálií při kontrole kvality.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Hluboké neuronové sítě pro detekci anomálií při kontrole kvality

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše Ph.D. Další informace mi poskytli Ing. Pavel Svoboda Ph.D. a Ing. Marián Beszédeš Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Juřica
22. května 2019

Poděkování

V této sekci bych rád poděkoval mému vedoucímu práce Ing. Michalu Hradišovi Ph.D. za poskytnutí odborné pomoci při jejím vypracování. Dále bych rád poděkoval Ing. Pavlu Svobodovi Ph.D. a Ing. Mariánu Beszédešovi Ph.D.

Obsah

Kapitola 1

Úvod

Automatizovaná detekce defektů ve výrobním procesu za použití počítačového vidění a strojového učení se těší v průmyslových aplikacích velké pozornosti. Jedná se o zajímavý způsob jak zredukovat výrobní náklady, urychlit a zároveň zajistit kvalitu výroby [39].

Za poslední řadu let se neuronové sítě těší stále větší pozornosti, což celou oblast posouvá velice rychle dopředu a slibuje aplikovatelné řešení i pro průmyslovou oblast.

V této práci se zabývám návrhem algoritmu pro autonomní detekci defektů při průmyslové výrobě plastikových karet ve formě občanského průkazu nebo platební karty. Při výrobě takovéto karty občas dochází k její kontaminaci a pod její povrch je zalisována prachová částice nebo vlas. Na konci výrobního procesu je proto aktuálně prováděna manuální vizuální kontrola kvality lidským operátorem. Algoritmus automatické vizuální inspekce by mohl značně redukovat náklady na výrobu a redukovat potřebu lidského úsilí při této fázi procesu.

Při návrhu algoritmu však čelím řadě výzev. První z nich je velice omezené množství trénovacích dat, které mám pro tento úkol dostupné. Jedná se o 217 karet, ze kterých je část využita pouze na testovací účely. Další výzvou je velikost hledaných defektů, která s průměrnou plochou 179 pixelů tvoří v kontextu celé karty o velikosti 2040×1278 pixelů průměrně 0,0068 % plochy karty. Pozadí karty je navíc velice různorodé a komplexní.

Pro svou práci jsem se rozhodl použít detekční algoritmus Mask R-CNN [19] natrénovaný na synteticky generovaných datech, které co možná nejpřesněji napodobují distribuci reálných vad nacházejících na kartičkách. Tímto způsobem jsem dosáhl dostatečného množství trénovacích dat a dokázal tak natrénovat síť, která následně byla schopna generalizovat na datech reálných.

V poslední části práce jsou výsledky použitého postupu vyhodnoceny a diskutovány možnosti, jakým směrem by bylo vhodné práci posunout.

Kapitola 2

Detekce anomálií

V úvodu této práce nabídnu obecný náhled do problematiky detekce anomálií. Pokusím se definovat, co to anomálie kontextu různých datových domén znamená. V této kapitole bude zmíněna řada výzev, kterým návrháři algoritmů pro detekci anomálií v datech čelí. Rovněž uvedu řadu přístupů k jejich řešení s příklady praktických aplikací. Zde jsem čerpal především z informací shrnutých v publikaci napsané Varunem Chandolou z roku 2009 [7].

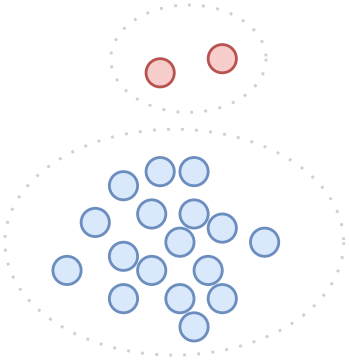
2.1 Definice anomálie

Pro návrh algoritmu pro detekci anomálních dat na dané doméně je žádoucí tuto vlastnost definovat. Definice anomálie však není triviální záležitostí a nějaká její univerzální forma prakticky neexistuje. Nachází se zde však množství úhlů pohledů, které často přímo závisí na dané doméně problému.

Úloha detekce anomálií můžeme být abstraktně definována jako problém hledání entit s vlastnostmi nebo chováním, které neodpovídají očekávanému vzoru neboli tomu, co v daném kontextu považujeme za normální. Tyto entity pak označujeme jako anomální. Takto alespoň problém definoval Varun Chandola [7]. Intuitivní chápání této vlastnosti je zobrazeno na obrázku 2.1.

Anomálie jakožto vlastnost je v praktických aplikacích typicky spojována s nějakým typem problému nebo závady. Například v kontextu bankovnictví a toku bankovních transakcí můžou anomálie odpovídat pokusům o krádež [67]. Při kontrole kvality ve výrobě jsou výrobky vykazující anomální vlastnosti typicky označeny za zmetky a jsou v této fázi procesu vyřazeny. V tomto kontextu se může jednat o strukturní vlastnosti výrobku.

V prostředí síťové komunikace je věnována pozornost vývoji systémů pro automatické detekování zvláštního chování uživatelů, které může prozrazovat síťový útok. D.E. Denning vytvořil model real-time expertního systému, jenž je schopen detekovat pokus o nežádoucí proniknutí útočníka do sítě [12].



Obrázek 2.1: Příklad chápání anomálie na 2D datech. Množina červených dat je v tomto případě považována za anomální vůči množině modrých dat. Množiny se výrazně liší svou lokací.

2.2 Povaha vstupních dat

Povaha vstupních dat je hlavním aspektem úlohy detekce anomálií. Pro různé domény se však může výrazně lišit. Pokud se hovoří o vstupních datech, typicky je tím myšlen nějaký objekt, událost, stav tohoto objektu, bod, vektor nebo pozorování objektu. Od povahy dat se pak odvíjí aplikovatelnost různých technik detekce. Tyto techniky jsou popsány dále v kapitole 2.4. Varun Chandola [7] navrhuje následující dělení podle povahy dat:

Bodové anomálie popisují nejčastější a nejjednodušší případ typu dat, kdy jejich jednotlivé instance můžeme vyjádřit bodem v prostoru. Pro tento bod následně určujeme, zda se jedná o anomalitu. Takovýto příklad je zobrazen na obrázku 2.1, kde třídu červených bodů označíme za anomální vůči třídě bodů modrých, protože leží mimo region považovaný za normální. Tento region je na obrázku ohraničený elipsou kolem modrých bodů.

V této třídě je vhodným příkladem detekce kriminální činnosti z analýzy pohybů na účtu uživatele. Data reprezentujeme hodnotou částky, která byla stržena z účtu. Pokud zaznamenáme pohyb na účtu daleko větší než je časté, můžeme jev označit za anomalitu a dále podniknou řešení potencionálního problému.

Kontextuální anomálie je případ takový, který může být označen za anomální pouze ze znalosti celkového kontextu dat. Bez informace o kontextu, tedy znalosti celkové struktury dat, by tento případ za anomální označen nebyl. Povaha kontextu dat je tedy v tomto případě nezbytnou součástí definice problému.

Následně naměřená hodnota může být typicky označena za anomální jen v některých kontextech. Ve zbylých může jít o zcela přirozené a očekávané chování.

Příkladem tohoto typu dat mohou být časové řady. Blíže se může jednat o záznamy teploty v různých zemích. Pokud například naměříme teplotu -10°C na Islandu, jedná se o nepřekvapivé zjištění. Pokud bychom tuto teplotu naměřili v letních měsících v Maroku, bude se jednat o velmi anomální zjištění. V tomto případě tedy záleží na kontextu oblasti a ročního období, kdy měření probíhá.

Kolektivní anomálie je takový jev, který označíme za nestandardní pouze pokud se vyskytne v kolekci více takovýchto podobných jevů.

Varun Chandola uvádí jako příklad graf EKG, kdy je výskyt hodnot o určité úrovni naprosto standardní. Pokud se však objeví delší sekvence těchto jevů za sebou, v tomto případě nazývaná kolekcí, označíme již jev za anomálii.

Varun Chandola detekci anomálií rozdělil také na základě znalosti příslušnosti dat do tříd do třech následujících kategorií. Každá kategorie má následně svůj specifický přístup k řešení úlohy:

Učení bez učitele (nesupervizované) zahrnuje data, o kterých nemáme žádné další informace. Neznáme tedy příslušnost dat do tříd, často ani počet těchto tříd. Dodatečné získání takovéto informace od lidského experta je často časově velice nákladné, pokud je vůbec možné. Tento případ však patří v kontextu detekce anomálií mezi nejčastější. Proto je zájem o schopnost detekovat anomálnosti i v této třídě dat velký.

Učení s učitelem (supervizované) pracuje s daty, jejichž instance mají známou příslušnost do tříd, což zjednodušuje řešení úlohy. Takovýto případ je v reálných aplikacích spíše výjimečný. Typickým přístupem je pak vytvoření prediktivního modelu pomocí těchto dat, který bude v ideálním případě generalizovat a správně klasifikovat data dosud neviděná.

Výzvou, která tenhle přístup musí řešit, je, že třídy anomálních a normálních dat jsou velice nevyvážené. Anomálních dat máme v praktických aplikacích typicky jen zlomek.

Částečně supervizované učení kombinuje oba předchozí případy, tedy nesupervizované a supervizované učení.

2.3 Výzvy při hledání anomálií

Na abstraktní úrovni je úkol detekce anomálií definován jako hledání entit, které neodpovídají normálním neboli očekávaným vlastnostem. Prvotním návrhem na řešení detekce anomálií by mohlo být vytyčení podprostoru, ve kterém se nacházejí položky odpovídající neanomální distribuci. Položky mimo tuto distribuci by byly následně označeny za anomální. Uvedený přístup je však komplikován řadou výzev, které úloha obsahuje:

- Určení přesné hranice mezi normálními a anomálními daty může být velice obtížné. Tato hranice často nemusí být vůbec přesná.
- V některých doménách se může definice normálních dat v průběhu času výrazně měnit. Data, která by v jednom časovém okamžiku byla určena jako anomální, se postupem času mohou stát naprosto majoritní a tedy normální. Systém se této možnosti musí umět korektně přizpůsobit.
- Přesná definice anomálnosti se mezi jednotlivými doménami dat může velice lišit. Vyvíjené postupy tedy většinou není jednoduché reaplikovat mezi různými doménami.
- Pro úkoly jsou typicky dostupná pouze neoznačená data. Vytvoření dostatečného množství označených dat je časově velice extenzivní práce. Často se tedy volí cesta použití dat neoznačených a tomu uzpůsobené algoritmy založené na učení bez učitele.
- Anomální data jsou často v datasetu velice minoritní. Obě třídy jsou tak velmi nevyvážené, což může způsobovat problémy při použití metod strojového učení.

- Množina anomálií není typicky konečná. V průběhu času očekáváme, že se můžou objevovat anomálie dosud neviděné. Přesto požadujeme jejich správnou identifikaci.

2.4 Techniky detekce anomálií

Povaha vstupních dat se může mezi jednotlivými doménami výrazně lišit. Z tohoto důvodu existuje široká variace metod k realizaci úlohy. Jednotlivé techniky vyvinuté jako řešení detekce anomálních jevů v datech mohou být rozděleny do kategorií popsaných v následující části.

Klasifikační metody se typicky skládají ze dvou fází. V první fázi je vytvořen model (klasifikátor), který je následně aplikován na dosud nepozorovaná datech s neznámou příslušností do třídy. Model, aby byl úspěšný, musí umět generalizovat na dosud neznámých datech.

V první fázi trénování jsou k vytvoření modelu nutná data se známou příslušností do tříd. Jejich získání je však často časově náročné a je nutná znalost doménového experta.

Druhým předpokladem je, že v daném prostoru použitých příznaků musí být možné rozlišit mezi normálními a anomálními daty, aby bylo vůbec možné klasifikátor natrénovat.

Typicky se v téhle kategorii používají konvoluční neuronové sítě [28] nebo SVM [9]. V úloze klasifikace anomálií se používá variace SVM zvané jednotřídní SVM (one-class SVM) [46]. Jednotřídní SVM se učí neostrou hranici trénovacího setu tak, že u nově příchozích dat dokáže určit, zda do trénovacího setu patří nebo ne.

Techniky založené na nejbližších sousedech. Tento přístup vychází z předpokladu, že příznakové vektory neanomálních dat jsou v prostoru seskupeny blízko sebe, zatímco data označená za anomální jsou od sebe rozmístěna dále. Prvním předpokladem k použití této techniky je schopnost vypočítat vzdálenosti mezi dvěma daty. Typickým příkladem metriky je pak euklidovská vzdálenost. Volba metriky je však kritická pro dobrou výkonnost algoritmu.

Na určení hodnoty anomálnosti testovacího data se pak můžeme dívat ze dvou různých úhlů. Prvním přístupem je určení hodnoty anomálnosti v závislosti na hustotě okolních sousedů. Tato technika je taktéž známá pod pojmem Local Outlier Factor (LOF) prezentována M. Breunigem [4].

Druhým přístupem je určení skóre anomálnosti datové instance jako vzdálenost k jeho N nejbližším sousedům v daném datasetu. Tato definice byla formulována S. Byersem v roce 1998 [6].

Techniky založené na shlukové analýze. Jedná se o techniku využívanou ke sdružování dat na základě jejich podobnosti [26].

Předpokladem této techniky je, že normální data jsou součástí nějakých clusterů, zatímco anomální data spadají mimo tyto clustery. Problémem tohoto předpokladu je, že data často obsahují šum a algoritmy shlukové analýzy nejsou předem navrhovány k detekci anomálií na datech. Pro tyto algoritmy také často není nutné, aby všechna data náležela nějakému clusteru. Prvotní předpoklad může být těmto faktům uzpůsoben následovně. Instance normálních dat leží blíže centroidům detekovaných clusterů, anomální pak naopak. Při použití této techniky je tedy v prvním kroku provedena shluková analýza. V dalším

kroku je pro každou novou instanci dat spočítána její anomálnost jako vzdálenost k centroidům clusterů.

Výsledkem evoluce těchto algoritmů je následující předpoklad: Normální data budou ležet ve velkých clusterech s hustou koncentrací dat. Anomální data budou náležet k malým a řídkým clusterům. Přístup založený na tomto předpokladu prezentoval v roce 2003 Z. He pod názvem FindCBLOF [21]. Jedná se o kombinaci LOF a přístupu založeného na shlukové analýze. CBLOF skóre je odvozeno od velikosti clusteru, ve kterém se instance nachází, a také od vzdálenosti od centroidu clusteru.

Může se zdát, že techniky založené na shlukové analýze a nejbližších sousedech jsou velmi podobné a volně se prolínají. Existuje však zásadní rozdíl rozlišující tyto techniky. Přístupy založené na shlukové analýze k ohodnocení nové instance dat používají pouze informace o clusteru, kterému instance náleží. V přístupu nejbližších sousedů jsou však využíváni všichni nejbližší sousedé a příslušnost ke clusterům není brána v zřetel.

Statistické metody počítají s vytvořením stochastického modelu. Následně pak předpokládáme, že normální data leží v regionech s velkou pravděpodobností výskytu zatímco anomálie naopak. Mezi tyto techniky patří Gaussovy modely nebo nebo modely založené na regresi. Tyto metody mají časté využití v analýze časových řad [3].

Faktické informace v této kapitole jsem čerpal z přehledu existujících technik pro detekci anomálií autora A. Patcha [44] z roku 2007 a přehledu autora V. Chandola [7] z roku 2009.

Kapitola 3

Techniky založené na neuronových sítích

Neuronové sítě se v posledních letech staly state-of-the-art přístupem pro řešení řady úloh. Od roku 2012, kdy se podařilo v soutěži ImageNet [11] autorům A. Krizhevsky, Ilya Sutskever a Geoffrey E. Hinton zvítězit za použití konvolučních neuronových sítí [28], získává tohle paradigma stále větší a větší pozornost.

Kapitola bude věnována mým poznatkům ze studia technik použití neuronových sítí. Tyto techniky jsou využívány v řadě aplikací zabývající se detekcí anomálií a tedy defektů při vizuální kontrole. Pozornost je věnována především autoenkodérům, Generetative Adversarial Networks a detekčním algoritmům.

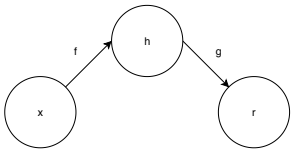
3.1 Autoenkodéry

Autoenkodér je speciální případ dopředné neuronové sítě trénované tak, aby prováděla funkci identity. Její výstup je tedy roven vstupu [17] a tedy rozměr vstupní vrstvy je roven velikosti vrstvy výstupní. Síť obsahuje vnitřní strukturu, která se učí zakódovat vstupní data tak, aby je na výstupu byla schopna co možná nejúspěšněji opět rekonstruovat. Autoenkodér může být popsat jako funkci skládající se ze dvou částí. Funkce enkodéru $h = f(x)$ se vstupem x a funkce dekodéru $r = g(h)$, která přijímá vstup h z enkodéru a na výstupu se ho pokouší zrekonstruovat na r. Schéma autoenkodéru je na obrázku 3.1.

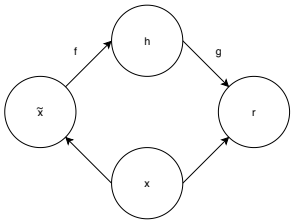
Proces učení lze vyjádřit jako minimalizaci chybové funkce vyjádřené následující rovnicí:

$$L(x, g(f(x))). \tag{3.1}$$

Myšlenka autoenkodéru byla odjakživa součástí výzkumu zabývajícího se neuronovými sítěmi. Funkce pouhého kopírování vstupu na výstup se může z počátku zdát jako nezajímavá a zbytečná. Cestou, jak z autoenkodéru získat zajímavá data je, že jeho latentní



Obrázek 3.1: Základní schéma autoenkodéru.



Obrázek 3.2: Schéma denoising autoenkodéru, kde $C(x) = \tilde{x}$, tedy nějakým způsobem poškozený vstup.

části h má menší rozměr než je vstup x. Takovýto autoenkodér je nucen naučit se nízkodimenzionální reprezentaci vstupu, ze které je opět schopen rekonstruovat výstup co nejvíce podobný vstupním datům. Je tedy nucen udržet si pouze nejdůležitější informace o datech. Takovýto autoenkodér je úspěšně využíván k redukci dimensionality. Na rozdíl od PCA [1] není autoenkodér omezen pouze na lineární mapování. Typicky se v praxi využívají různé variace, které vedou k lepším výsledkům.

Denoising autoenkodér [62] dostává jistým způsobem částečně poškozená data $\tilde{x}$. Síť je musí rekonstruovat jejich nepoškozenou předlohu x. Ilustrace takovéhoto modelu je na obrázku 3.2. Originální zdroj poškozeného obrázku by mělo být jednoduché odhadnout pouhým pohledem, což je inspirováno lidskou vlastností rozeznat i částečně zakryté objekty nebo vzory. Zmíněná úprava nutí síť učit se pouze robustní příznaky a díky tomu lépe generalizovat. Tato myšlenka následně sloužila jako inspirace pro dropout [59]. Tato variaci je vyjádřena chybovou funkcí

$$L(x, g(f(\tilde{x}))). \tag{3.2}$$

Řídké autoenkodéry (Sparse Autoencoders) [37] přidávají k základnímu konceptu také kritérium řídkosti posilující schopnost autoenkodéru učit se zajímavou strukturu vstupních dat. U řídkých autoenkodérů je tato schopnost zachována dokonce i v případě, kdy je skrytá vrstva větší než vrstva vstupní.

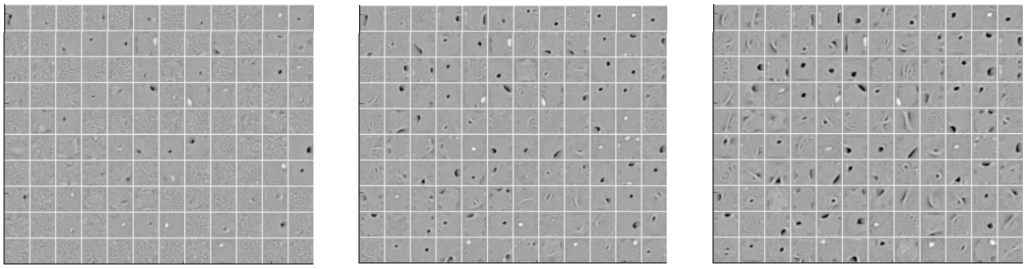
Aktivovaným neuronem nazýváme neuron s výstupní hodnotu blízké 1. Naopak pokud se hodnota blíží 0, říkáme, že neuron je neaktivní. V případě řídkých autoenkodérů chceme udržet počet aktivovaných neuronů v latentní vrstvě autoenkodéru co možná nejmenší. Dosáhneme toho tím, že k určitému typu vstupu se aktivuje pouze část neuronů ve skryté vrstvě, nikoliv většina. Aplikace této myšlenky vede k lepším vnitřním reprezentacím dat autoenkodérem.

Kontraktivní autoenkodéry publikované S. Rifaiem v roce 2011 [52] přidává speciální regularizační složku k chybové funkci. Cílem této složky je, aby derivace funkce $h = f(x)$ byly co možná nejmenší. To nutí model učit se funkci takovou, že se její výstup příliš nezmění, pokud je změna vstupu taktéž minimální.

Chybová funkce vypadá následovně:

$$L(x, g(f(x))) + \Omega(h), \tag{3.3}$$

$$\Omega(h) = \lambda \left|\left| \frac{\partial f(x)}{\partial x} \right|\right|_F^2, \tag{3.4}$$



(a) 0% zašumění vstupu. (b) 25% zašumění vstupu. (c) 50% zašumění vstupu.

Obrázek 3.3: Filtry získané tréninkem autoenkodérů s různým poměrem zašumění vstupu na MNIST datasetu [29]. Obrázek pochází z [62].

kde $\Omega(h)$ je druhá odmocnina Frobeniovy normy Jacobiho matice parciálních derivací funkce enkodéru.

3.2 Generative Adversarial Networks

Koncept Generative Adversarial Networks (dále GAN) byl v roce 2014 popularizován I. Goodfellowem a dalšími [18]. Tuto publikaci následovala velká řada prací. Byla tak odstartována velká vlna výzkumu zaměřeného tímto směrem, která pomohla oblast strojového učení významně posunout blíže ke generování realistických dat.

Podstatnou výhodou GANů je fakt, že nevyžadují anotovaná data, což jenom umocňuje jejich popularitu a jednoduchost použití.

Základní architektura GANu se skládá ze dvou neuronových sítí. První z nich je generátor G, jehož úloha je z N-dimenzionálního vektoru z produkovat výstup co možná nejvíce podobný reálné distribuci. Druhou součástí je síť diskriminátoru. Diskriminátor D dostává výstupy z generátoru a reálné obrázky, mezi kterými se učí rozlišovat. Jeho výstupem je skalární hodnota $D(x)$ reprezentující pravděpodobnost, že x pochází z reálné distribuce a není výstupem generátoru. Pro diskriminátor D se tedy snažíme zároveň maximalizovat $D(x)$ a minimalizovat $D(G(z))$. Chybová funkce je tedy složena ze dvou částí

$$L(D) = L(D_r) + L(D_f), \tag{3.5}$$

$$L(D_f) = log(1 - D(G(z^{(i)}))), \tag{3.6}$$

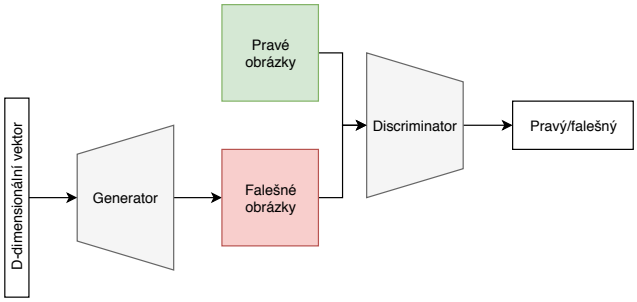
$$(D_r) = log(D(x^{(i)})), \tag{3.7}$$

$$(D) = \frac{1}{m}\sum_{i=1}^{m}\left[log(D(x^{(i)})) + log(1 - D(G(z^{(i)})))\right]. \tag{3.8}$$

Z perspektivy generátoru se snažíme maximalizovat $D(G(z))$, tedy diskriminátor obelstít. Chybová funkce generátoru je definována jako

$$L(G) = \frac{1}{m}\sum_{i=1}^{m} log(1 - D(G(z^{(i)}))). \tag{3.9}$$

Tyto dvě sítě jsou vedeny k vzájemnému soupeření a zároveň se každá uvedeným procesem v ideálním případě zlepšuje ve své vlastní úloze. Generátor se tak naučí generovat



Obrázek 3.4: Základní schéma GANu. Architektura se skládá ze dvou neuronových sítí: z generátoru a diskriminátoru. Generátor je trénován z náhodného N-dimenzionálního vektoru vygenerovat data co nejvíce odpovídající reálné distribuci. Diskriminátor střídavě dostává reálná data a data vygenerovaná generátorem. Jeho úloha je naučit se oba typy dat rozlišit.

data z reálné distribuce na základě vstupního N-dimenzionálního vektoru. Toto vzájemné učení autoři definují jako minimax hru o dvou hráčích

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[log(D(x))\right] + \mathbb{E}_{z\sim p_z(z)}\left[log(1-D(G(z))\right]. \tag{3.10}$$

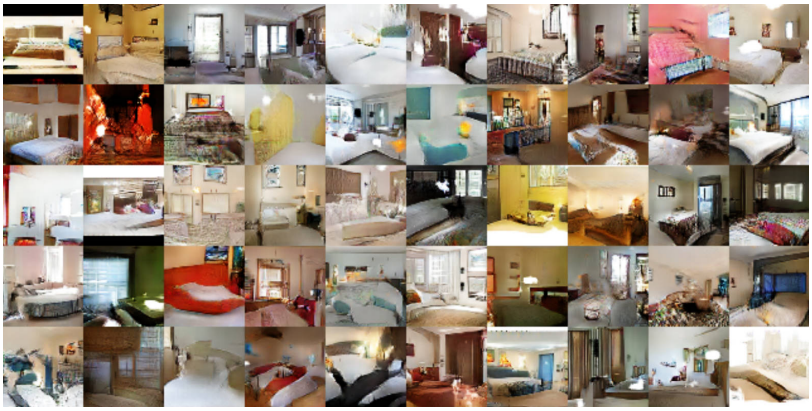
Zmíněné definice byly převzaty z publikace od I. Goodfellow [18]. Základní architektura GANů je ilustrována na obrázku 3.4. V této sekci se nejprve zaměřím na vývoj architektur GANů. Následně se zaměřím na jejich aplikace ovlivňující mou práci a rozeberu je detailněji.

Ačkoliv se architektura GAN těší velkému zájmu, její trénování s sebou nese řadu problémů a výzev. Tyto problémy byly adresovány a další výzkum se je snažil vyřešit. Problémy vyskytující se při jejich tréninku jsou následující.

Mode collapse je stav, kdy generátor produkuje pouze omezený počet vzorků reálné množiny. V extrémním případě se jedná o situaci, kdy generátor opakovaně generuje pouze jeden vzorek. K tomuto stavu dochází, když generátor není nucen učit se celou reprezentaci, ale opakovaně dokáže úspěšně diskriminátor obelstít pouze pomocí jediného vzorku.

Mizející gradienty jsou typickým problémem při trénování hlubokých neuronových sítí, který se tedy bohužel týká taktéž GANů. Zde je situace posílena tím, že gradienty z diskriminátoru nejsou propagovány jen do diskriminátoru samotného, ale také do generátoru. Je proto nutné držet diskriminátor a generátor v rovnováze. Pokud je chyba diskriminátoru příliš velká, generátor nedostává dostatečnou odezvu. Situace není dobrá ani v případě, kdy je dominantní diskriminátor. Trénink GANů je proto nestabilní.

DCGAN [45] (Deep Convolutional GANs, 2015) je intuitivním pokračováním architektury GAN. Jde o myšlenku kombinace GAN a konvoluční neuronové sítě [28]. Konvoluční neuronové sítě dosahovaly skvělých výsledků v úlohách, kde se učily z anotovaných dat. Touto publikací úspěšně pronikly i do sféry učení bez učitele a vylepšily tak možnosti GAN sítě. Autoři článku trénovali DCGAN na více datových doménách a následně ukázali, že hluboké konvoluční GANy lze efektivně použít pro učení se vnitřních reprezentací od malých objektů až pro celé scény. Tohle autoři dokázali vizualizací skrytých konvolučních vrstev a jejich filtrů se závěrem, že specifické filtry se naučily generovat určité objekty. Svou



Obrázek 3.5: Výsledky získané s architekturou DCGAN [45] trénovanou na datasetu ložnic LSUN. Obrázek převzat z [45].

publikací taktéž rozšířili poznatky o GAN sítích a především se podíleli na zvýšení stability jejich tréninku.

Autoři nepoužili vrstvy s pooling operací, které se typicky v konvolučních sítích používají. Místo nich využili konvoluční vrstvu s větším stridem. Síť se tak operaci převzorkování učí sama a nedochází zde ke ztrátě informace.

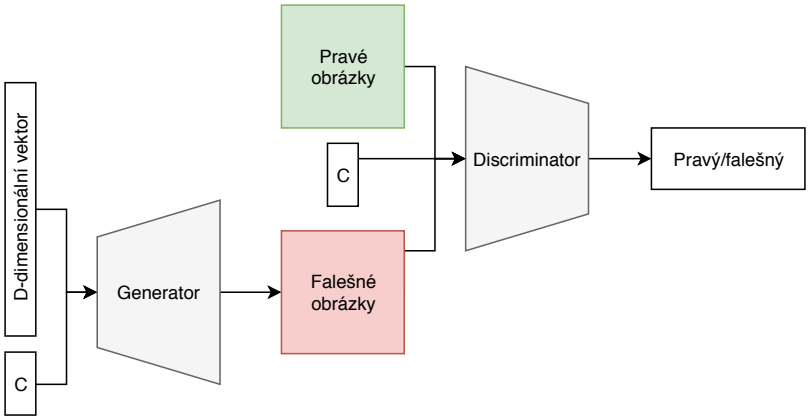
Dále doporučují použití batch normalizace [24] v generátoru i diskriminátoru. Vstup každé vrstvy, kromě výstupní vrstvy generátoru i diskriminátoru, se tedy normalizoval tak, aby měl střed v nule a požadovanou varianci. Batch normalizace pomohla s lepší distribucí gradientů hlubokou sítí a také redukovala problémy způsobené nevhodnou inicializací vah na začátku tréninku. Před touto změnou se objevovaly problémy, kdy se například generátor nezačal učit. Dále nastávaly situace, kdy se generátor neučil vhodnou distribuci dat, ale soustředil všechny tréninkové příklady do jednoho bodu.

Autoři taktéž experimentovali s použitím různých typů aktivačních funkcí. Nejlepších výsledků dosáhli při použití ReLU [42] aktivační funkce ve všech vrstvách generátoru, kromě vrstvy poslední, kde použili Tanh funkci. S tímto nastavením se generátor začal učit dříve a lépe se dokázal využít všechny barevné složky vstupní tréninkové distribuce. V případě diskriminátoru bylo zjištěno, že aktivační funkce Leaky ReLU [36] funguje lépe v případě generování výstupů o větším rozlišení.

Jedním z datasetů, na kterém byl publikován experiment, je dataset ložnic LSUN [66]. Na tomto datasetu ukázali pozoruhodné generační schopnosti GANu, který se naučil generovat obrázky ložnic. Výsledky jsou na obrázku 3.5. Zároveň interpolací mezi dvou body z distribuce z ukázali schopnost GANů plynule interpolovat mezi dvěma obrázky. Taktéž bylo ukázáno, že pokud mezi dvěma latentními proměnnými z_1, z_2 aplikují operaci vektorové aritmetiky, výsledek se také patřičně projeví ve výstupních obrázcích. Autoři tak ukazují, že síť se smysluplně naučila sémantické vizuální koncepty.

T. Salimans a spol. [55] (2016) přinesli další řadu technik pro lepší výsledky a stabilnější trénink DCGAN. Těmi jsou například *Feature matching*, kdy se mění přístup v učení generátoru G. V tomto případě se generátor neučí pouze obelstít diskriminátor, ale také vygenerovat obrázek takový, aby výstupy vrstev diskriminátoru byly statisticky co nejvíce podobné jako při reálném obrázku.

Další technikou pro eliminaci mode collapse je tzv. *Minibatch discrimination*. V případě mode collapse je podobnost výstupních obrázků velice malá. Autoři tak doporučují skládat



Obrázek 3.6: Architektura CGAN. Výstupy modelu jsou podmíněny proměnnou C. Může se jednat například o třídu objektu.

batch pouze z reálných neb vygenerovaných obrázků. Následně je vypočítána podobnost všech obrázků v mini batchi. Výsledná hodnota je předána diskriminátoru, který ji vezme v potaz při klasifikaci reálnosti obrázků.

Jako diskriminační techniku zlepšující kvalitu tréninku představili autoři tzv. *One-sided label smoothing*, kdy reálné a vygenerované obrázky nejsou označeny celými hodnotami 0,0 a 1,0, ale hodnotami s mírnou odchylkou. Tedy například 0,1 a 0,9. Tímto způsobem je tak zabráněno tomu, aby se model soustředil jen na malou podmnožinu použitých příznaků.

Conditional GAN [41] (2014) jsou dalším důležitým milníkem a rozšířením původní GAN architektury. Autoři tak učinili přidáním dalšího vstupu do generátoru i diskriminátoru. Tímto vstupem je proměnná podmiňující například příslušnost do třídy zpracovávané instance. Výsledkem je v ideálním případě model generující své výstupy na základě této podmínky. Výsledná architektura je na obrázku 3.6.

GANy donedávna trpěly nedostatečnou stabilitou při tréninku s větším rozlišením obrázku. T. Karras a spol. ze společnosti NVIDIA publikovali v článku *Progressive Growing of GANs for Improved Quality, Stability, and Variation* (2018) [27] metodologii, jakým způsobem je možné dosáhnout stability i při tréninku na datech s rozlišením 1024^2. Stěžejní myšlenkou jejich přístupu je při tréninku postupně zvětšovat velikost generátoru a diskriminátoru přidáváním dalších větších vrstev, které se učí jemnější detaily. Začínali tak trénovat na obrázcích s malým rozlišením 4^2 a postupně toto rozlišení přidáváním vrstev zvyšovali až na 1024^2. Dosáhli tak rychlejšího tréninku a vyšší stability a tím doposud nevídané kvality na datasetu CelebA [34].

Aktuálním state-of-the-art přístupem je architektura BigGAN [5] (2019), která dokáže generovat velmi realistické obrázky v rozlišení 512^2. Příklad výstupů je na obrázků 3.7.

Vliv myšlenky GAN měl od svého představení velký dopad na techniky zpracování obrazu a strojového učení. Za tu dobu se objevila dlouhá řada zajímavých publikací. V nedávné době značnou pozornost získala publikace StackGAN [69] od autorů H. Zhang a spol. Ve svém přístupu ukázali, jakým způsobem je možné syntetizovat obrázek pouze na základě jeho textového popisu. Takováto architektura pracuje ve dvou fázích. Prví fáze vygeneruje prvotní obrázek v podobně jednoduchých tvarů a barev o rozlišení 64^2. V druhé fázi je tento obrázek vylepšen a je vygenerován obrázek s vyšším rozlišením 256^2 a detaily. Jejich výsledky jsou na obrázku 3.8.

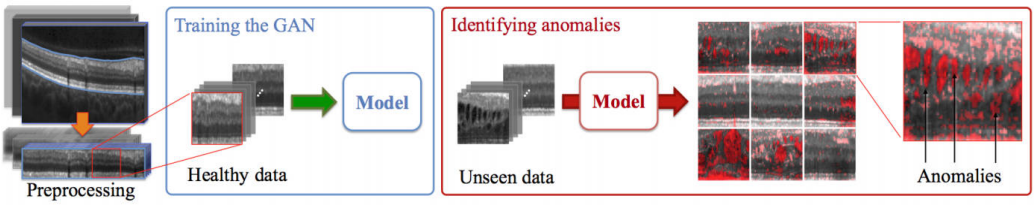
Obrázek 3.7: Ukázka obrázků v rozlišení 512^2 vygenerovaných architekturou BigGAN [5]. Převzato z [5].

Pozoruhodné výsledky dosáhl P. Isola a spol. [25] za použití architektury CGAN. Ukázali, jak lze naučit síť mapování mezi dvěma obrázky. Tento přístup se uchytil pod názvem Pix2Pix. Autoři předvedli široké schopnosti této sítě na několik příkladech. Ukázali mapování mezi například náčrtem a realisticky vypadajícím obrázkem, mapování mezi denním a nočním obrázkem, transformaci mezi typem mapového podkladu nebo dokonce vytvoření scény ze sémantické mapy. Na tuto publikace navázalo Pix2PixHD [64] s ještě věrohodnější kvalitou generovaných obrázků.

Jednou z aplikací příbuzné mé práci je detekce anomálií na snímcích z optické koherentní tomografie sítnice oka představené autorem T. Schlegl [56]. V tomto případě je generátor DCGANu použit pro modelování normální distribuce. GAN se tedy v ideálním případě naučí distribuci obrázků sítnice oka, ve kterých se žádná vada nenachází. V čase inference je nutné posoudit, zda se ve vstupním obrázku nachází nějaká vada nebo ne. K tomuto autoři využili distribuci naučenou generátorem. Myšlenka je taková, že generátor dokáže generovat pouze části obrázku bez vady a tedy defekty v tomto obrázku nebudou nikdy přítomny. Je tedy nutné najít hodnotu z takovou, že rozdíl $G(z)$ a vstupního obrázku je minimální. Tento úkon autoři realizovali optimalizací algoritmem SGD. Vizualizace tohoto přístupu je na obrázku 3.9.

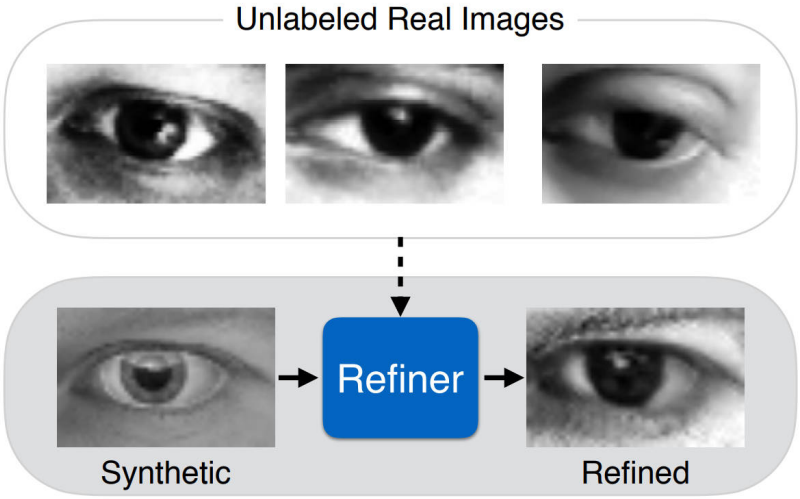
GANy si svoje uplatnění našly i v publikacích zabývající se hledáním anomálií a defektů v průmyslové výrobě. Tyto publikace blíže proberu v kapitole 4.

Obrázek 3.8: Ukázka obrázků vygenerovaných architekturou StackGAN [69]. Výstupní obrázky jsou vygenerovány na základě textového popisu. Převzato z [69].



Obrázek 3.9: V případě publikace T. Schlegla [56] byl GAN využit k naučení se distribuce obrázku bez defektu. Následně z této distribuce vygenerovali obrázek nejpodobnější vstupnímu a z jejich rozdílu detekovali případné anomálie. Obrázek převzat z [56].

Další velice zajímavou aplikací příbuznou mé práci je použití GANů pro augmentaci a rozšíření datasetů. A. Shrivastava publikoval v roce 2017 techniku, jak využít GAN ke zvýšení realističnosti syntetických dat a rozšít tak trénovací sadu [57]. Naučil generátor modifikovat vstupní obrázek tak, aby ho nebylo možné rozlišit od reálné trénovací sady. Vizualizace přístupu je na obrázku 3.10. Tato publikace získala cenu Best Paper Award na CVPR 2017.



Obrázek 3.10: V případě publikace T. Schlegla [57] byl GAN naučení se mapování syntetických obrázků do distribuce reálných dat. Dokázal tak rozšířit trénovací sadu. Obrázek převzat z [57].

3.3 Detekční algoritmy

Detekční algoritmy se v posledních letech těšily společně s neuronovými sítěmi velké pozornosti, což vedlo k jejich strmému rozvoji. Pro realizaci mé práce jsem se věnoval také studiu detekčních algoritmů, kterým bude tato sekce věnována.

Nejprve je třeba říci, co to detekční úloha je a co je výstupem detekčního algoritmu. Narozdíl od klasifikačního úkolu, při detekci je třeba objekt nejen klasifikovat, ale i lokalizovat. V detekční úloze také nastává možnost, že se v obrázku nachází objektů více. Výstupem detekčního algoritmu je tedy lokace objektu typicky reprezentovaná bounding boxem a třída objektu (nebo pravděpodobnosti tříd) pro N detekovaných objektů.

Před několika lety byl dominantním řešením kaskádový detektor Viola–Jones [63] založený na Haar příznacích nebo další metody založené na SIFT [35] nebo HOG [10] příznacích.

S rozvojem technik hlubokého učení se začaly objevovat metody, které již nepracovaly s pečlivě a ručně vybíranými příznaky, ale byly schopny se učit od začátku až do konce tedy i včetně těchto příznaků. Uvedené přístupy jsou typicky založeny na konvolučních neuronových sítích (CNN).

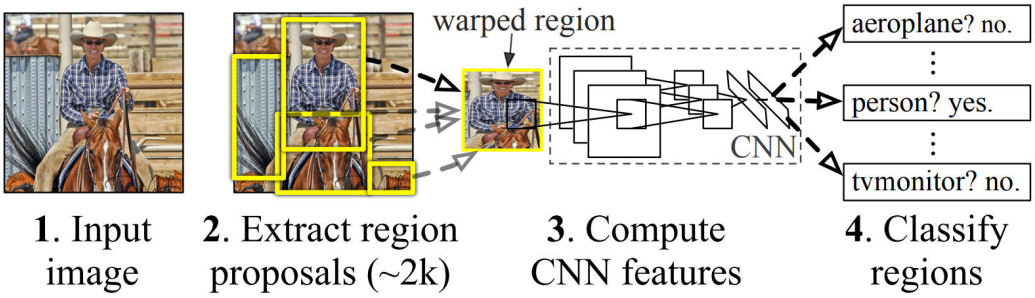
Neurosíťové detekční algoritmy jsou děleny na dvě skupiny. První z nich se snaží výsledky získat již po jediném dopředném průchodu natrénovanou sítí. Tyto algoritmy jsou tedy velice rychlé, avšak za cenu přesnosti. Typickými zástupci jsou algoritmy YOLO (You Look Only Once) [47, 48, 49], SSD (Single Shot Detector) [33] nebo RetinaNet [31].

Druhou skupinu algoritmů reprezentuje množina R-CNN [16], Fast R-CNN [15], Faster R-CNN [51] a Mask R-CNN [19]. Tyto algoritmy zpracovávají obrázek ve více fázích. První fáze spočívá v identifikaci zajímavých kandidátních regionů. Druhá fáze pracuje už jen s těmito regiony a dochází k jejich dalšímu zpracování, filtrování a upřesňování výsledků. Jedná se o algoritmy přesnější, avšak za cenu rychlosti.

R-CNN (Regions with CNN features) [16], představen v roce 2014 R. Girshickem, je prvním z algoritmů založený na dvou fázích. V první fázi je vybrán konečný počet zajímavých regionů, ty jsou následně zpracovány. První fáze je v případě R-CNN založena na algoritmu zvaném Selective Search [61]. Vstupní obrázek je shlukovacím segmentačním algoritmem rozdělen na konečný počet regionů na základě jejich podobnosti. Takto je vybráno 2000 potencionálně zajímavých oblastí, tedy potencionálních detekcí. V další fázi jsou tyto regiony transformovány do unifikované velikosti. Z každého takovéhoto regionu je pomocí předtrénované neuronové sítě vyextrahován příznakový vektor o velikosti 4096 dimenzí. Příznakové vektory jsou následně klasifikovány pomocí SVM [9] pro přítomnost objektu. Kromě klasifikace zde probíhá regrese pozice bounding boxu, tedy upřesnění jeho prvotní pozice dané Selective Search algoritmem. Architektura R-CNN je shrnuta obrázkem 3.11.

Přístup R-CNN trpí řadou neduhů, které byly adresovány v navazujících publikacích. Selective Search algoritmus je těžce paralelizovatelný na GPU a navíc je fixní, není možné ho tedy učit. R-CNN algoritmus je navíc velice pomalý. Běh detekce pro jeden obrázek trvá podle autorů 47 sekund. Hlavním důvodem je počet 2000 regionů, ze kterých v každém běhu musí být extrahován příznakový vektor. Navíc pokud se regiony překrývají, je tato operace provedena duplicitně. Problém s rychlostí se týká taktéž tréninku sítě.

Fast R-CNN [15] od stejného autora na problémy s rychlostí R-CNN reaguje a navazuje podobnou architekturou. Tentokrát je konvoluční neuronovou sítí vyextrahována příznaková mapa z celého vstupního obrázku najednou při jednom běhu. Z příznakové mapy se následně pomocí ROI pooling operace vyřezává příznakový vektor pro každý potencionálně zajímavý



Obrázek 3.11: Architektura algoritmu R-CNN. V prvním kroku je pomocí algoritmu Selective Search vybrán konečný počet zajímavých regionů. Z těchto regionů je následně pomocí neuronové sítě vyextrahován příznakový vektor, který je klasifikován pro přítomnost objektu určité třídy. Obrázek převzat z [16].

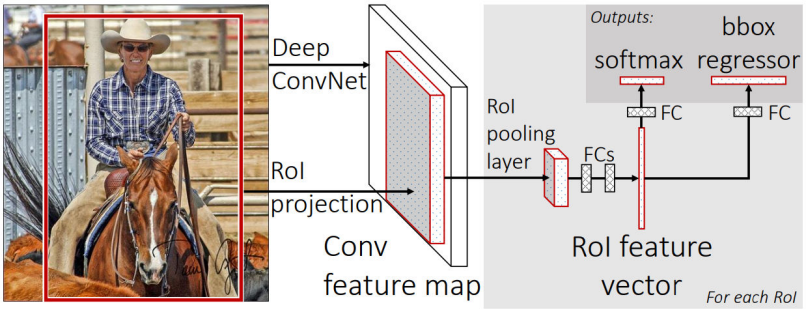
region. Tato změna vede k razantní úspoře času díky redukci duplicitních operací při extrakci příznakového vektoru z každého jednoho regionu zvlášť. Podle autora se jedná o čas 223ms na jeden obrázek.

Příznakový vektor každého regionu je po průchodu plně propojenou vrstvou klasifikován operací softmax. Taktéž jsou regresovány hodnoty upřesňující polohu bounding boxu. Celá architektura Fast R-CNN je na obrázku 3.12.

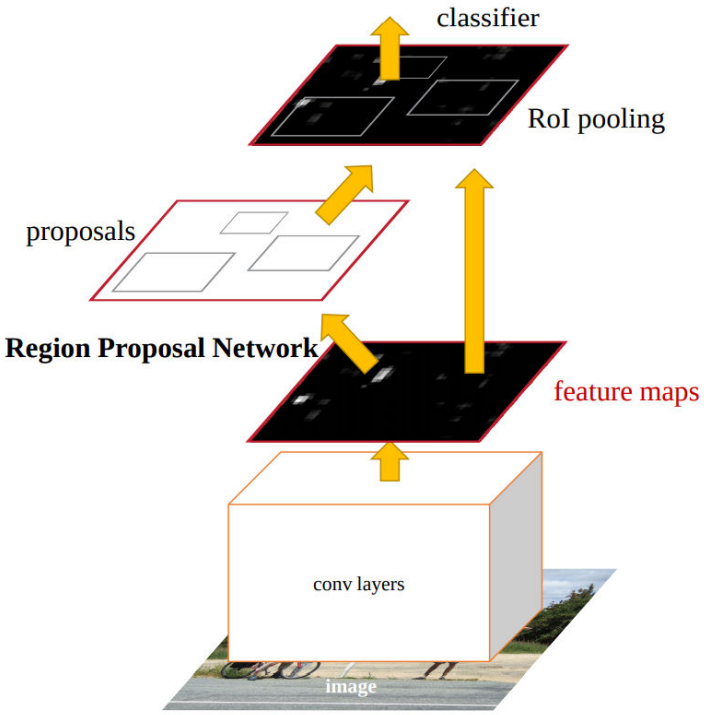
Zajímavé regiony jsou stále vybírány pomocí Selective Search.

Faster R-CNN [51] reaguje na fakt, že oba předchozí algoritmy R-CNN a Fast R-CNN využívají k predikci zajímavých regionů netrénovatelný Selective Search algoritmus. Faster R-CNN přichází s úpravou architektury a zajímavost regionu hodnotí pomocí natrénované neuronové sítě (RPN, Region Proposal Network), a to přímo nad vyextrahovanou příznakovou mapou. RPN je učena pro každou lokaci předpovídat s jakou pravděpodobností se zde nachází objekt (tzv. objectness score). Kromě hledání objektu taktéž dochází k regresi polohy bounding boxu. Jedná se 4 hodnoty upřesňující polohu objektu.

Dochází tak k jakési unifikaci architektury a zavedení následujících pojmů. Přímo nad obrázkem pracuje konvoluční extraktor příznaků nazývaný backbone. Ten je zaměnitelný za různé architektury. Backbone se v aplikacích typicky využívá již předtrénovaný pro lepší generalizační vlastnosti a zkrácení doby tréninku. Nad příznakovou mapou pracuje RPN,



Obrázek 3.12: Architektura algoritmu Fast R-CNN. Na rozdíl od R-CNN je extrakce příznaků provedena na začátku v jediném běhu. Pro každý region se z této mapy následně pooling operací extrahuje příslušná část vstupu. Obrázek převzat z [15].



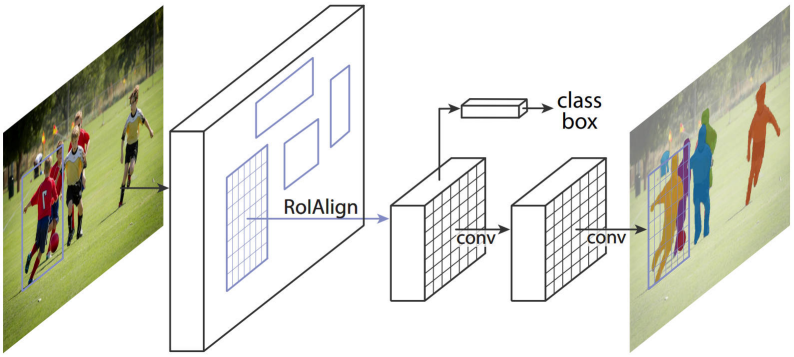
Obrázek 3.13: Architektura algoritmu Faster R-CNN přináší nový přístup v podobě hledaní zajímaných regionů pomocí neuronové sítě (zvané RPN) přímo nad mapou příznaků. Obrázek převzat z [51].

které predikuje přítomnost objektu a prvotní odhad jeho bounding boxu. Výstupy RPN se transformují do jednotného tvaru pomocí ROI poolingu a jsou předány do neuronové sítě pro predikci třídy objektu a sítě pro regresi bounding boxu. Těmto dvěma větvím se také v literatuře říká heads. Architektura Faster R-CNN je naznačena na obrázku 3.13.

Mask R-CNN [19] přímo navazuje na Faster R-CNN a přidává několik vylepšení. K větvím pro predikci třídy a bounding boxu přidává další větev, a to paralelně pracující větev pro predikci binární segmentace objektu. Kromě detektoru tak Mask R-CNN získala schopnost segmentovat instance. Architektura je na obrázku 3.14.

Pro výpočet segmentace však již není dostatečné získávat data z mapy příznaků pouze ROI pooling operace. Dochází zde ke ztrátě lokalizační informace z důvodu kvantizační chyby, což je v případě tvorby segmentační mapy velký problém. Výsledná binární maska nebyla ideálně zarovnaná na vstupní obrázek. Autoři proto přicházejí s řešením v podobě ROIAllign přístupu pro extrakci regionu zájmu z příznakové mapy. ROIAlign počítá každou hodnotu bodu pomocí bilineární interpolace nejbližších bodů na mřížce mapy příznaků. Autoři s tímto přístupem reportují dosažení lepších výsledků a to i v případě původního Faster R-CNN v kombinaci s ROIAllign.

Autoři také zkoušeli experimentovat s novými backbone architekturami. S FPN [30] architekturou dostali lepších výsledků než s C4 příznaky, tedy lepších výsledků dosáhli, když použili příznaky z více úrovní. Dále lepších výsledků dosáhli s backbone využívající ResNeXt [65] síť o hloubce 101 vrstev v porovnání vůči ResNet [20] architektuře.



Obrázek 3.14: Architektura algoritmu Mask R-CNN k původnímu Faster R-CNN přidává větev pro predikci binární masky. Obrázek převzat z [19].

Autoři Mask R-CNN vyzdvihují celkovou flexibilitu tohoto řešení. Jako příklad ukázali použití Mask R-CNN i jako detektor důležitých bodů na lidském těle, kde dosáhli taktéž dobrých výsledků za cenu minimálních zásahů do implementace sítě.

Jednostupňové detektory (bez extrakce zajímavých regionů) se od předchozích přístupů značně liší. Tato skupina detekčních algoritmů nepracuje s navrhovanými regiony, ale zpracovává obrázek konvolučně jednom běhu do výstupního tensoru. Jednostupňové algoritmy excelují svou rychlostí, avšak za cenu přesnosti. S postupným vývojem se však rozdíl v přesnosti zmenšuje a algoritmy tohoto typu se stávají velice zajímavé především pro aplikace pracující v reálném čase.

YOLO [47] (You Only Look Once) od autorů J. Redmon a spol. reagovali na nedostatečnou rychlost vícefázových přístupů a v roce 2015 představili detektor schopný pracovat rychlostí 45 snímků za sekundu na GPU. Autoři pojali úkol detekce objektů jako regresní problém a predikují konečnou množinu bounding boxů společně s jejich pravděpodobností pro všechny třídy. To vše se děje v jednom běhu sítě.

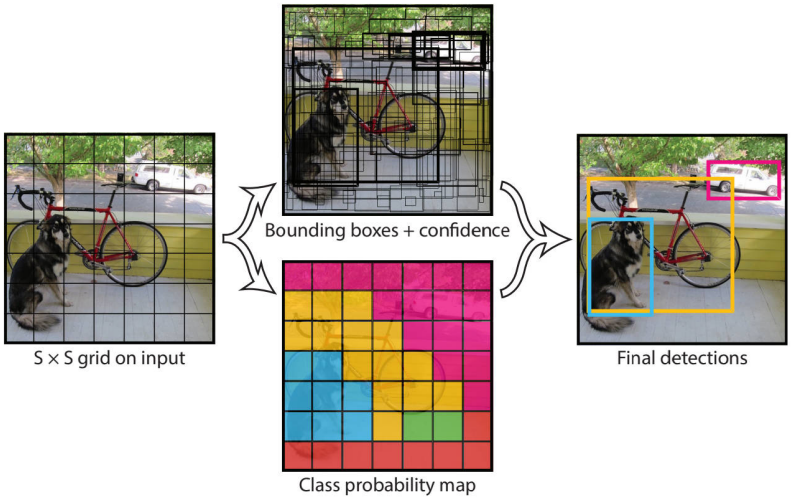
YOLO algoritmus nejprve vstupní obrázek rozdělí do $S \times S$ regionů. Pro každý tento region je predikováno B bounding boxů, jejich důvěryhodnost a C pravděpodobnostních hodnot pro každou z tříd. Výstupem sítě je tedy tensor o velikosti $S \times S \times (B * 5 + C)$. Architektura algoritmu je přiblížena na obrázku 3.15.

Algoritmus však nevyniká v situacích, kdy je na vstupu mnoho drobných objektů v jedné lokaci, například hejno ptáků.

YOLOv2 [48] přináší řadu drobných změn a zmenšuje přesnostní rozdíl mezi YOLO a R-CNN přístupy. Bylo identifikováno, že menší recall u YOLO je způsoben větším množstvím lokalizačních chyb a jejich pozornost se ubírala směrem k jejich minimalizaci.

Autoři přidáním batch normalizace [24] na všech konvolučních vrstvách získali více než 2% zlepšení v mAP. Následně pak mohli odebrat dropout [59] aniž by začalo docházet k přeučení sítě.

Dále trénovali nejprve klasifikační část sítě na ImageNet datasetu [11] po dobu 10 epoch. Následně trénovali až zbývající detekční část. Tímto postupem získali zlepšení téměř 4% mAP.



Obrázek 3.15: Architektura algoritmu YOLO (You Look Only Once). Vstup je nejprve rozdělen do $S \times S$ regionů. Pro každý tento region je predikováno B bounding boxů, jejich důvěryhodnost a C pravděpodobnostních hodnot pro každou z tříd. Obrázek převzat z [47].

V druhé verzi byla použita jemnější příznaková mapa. Zatímco původní mapa byla dostatečná pro velké objekty, při detekci malých objektů měla původní síť problémy. Proto byly do výsledné mapy konkatenovány příznaky z více úrovní. Síť by tak měla mít jemnější informaci a dokázat lépe detekovat drobné objekty.

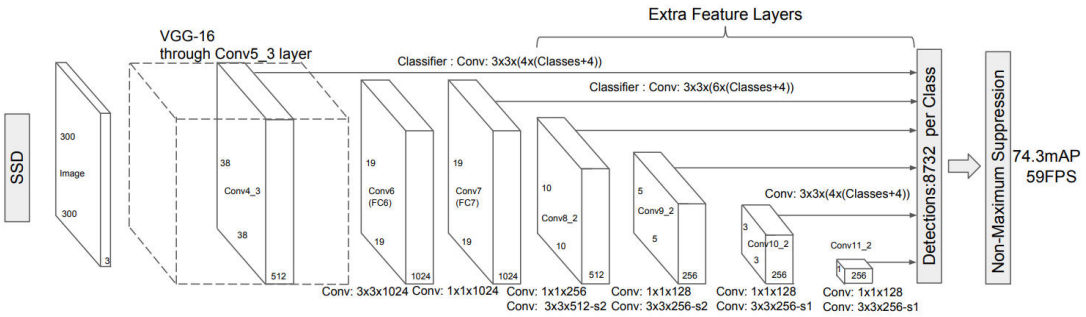
YOLOv3 [49] dál vylepšilo svou přesnost a snížilo tak rozdíl mezi algoritmy na bázi nabízených zajímavých regionů. Změnou ve třetí verzi byl například hlubšího extraktoru příznaků zvaný Darknet-53 s 53 konvolučními vrstvami. Jsou zde také propojení přes vícero vrstev podobně jako v architektuře ResNet.

SSD [33] (Single Shot MultiBox Detector) algoritmus nejprve vstupní obrázek transformuje neuronovou sítí architektury VGG-16 [58]. Následně je aplikována řada konvolučních vrstev redukujících velikost obrázku. Hlubší konvoluční vrstvy tak pokrývají větší receptivní pole a učí se více abstraktní příznaky. Příznaky z 6 úrovní této sítě jsou následně použity na detekci objektů o proměnné velikosti. Architektura SSD je na obrázku 3.16.

RetinaNet. [31] Autor RetinaNet Tsung-Yi Lin identifikoval problém jednofázových detektorů v nevyváženosti tříd dat, na kterých se detektor učí. U dvoufázových detektorů typu R-CNN se nejprve vyextrahuje příznaková mapa, ze které jsou vybrány regiony potencionálně patřící objektu. V této fázi jsou tedy odfiltrovány oblasti jasně patřící pozadí. V další fázi jsou aplikovány heuristiky pro výběr regionů pro trénování dalších částí sítě. Tyto regiony jsou tedy vybrány vhodným způsobem umožňujícím stabilní a efektivní trénink. Důležitým parametrem výběru je například vyvážení pozitivních a negativních příkladů pro trénink sítě, který je v případě dvoufázových detektorů zabezpečen.

U jednofázových detektorů typu YOLO a SSD vstupuje do tréninku mnohem větší množství kandidátních lokací, přičemž většina z nich je jednoduše klasifikovatelná jako pozadí a pro trénink nepřináší dostatečnou přidanou hodnotu.

Autoři RetinaNet tento fakt řeší úpravou chybové funkce zvanou Focal Loss. Jedná se o modifikaci cross entropy funkce váhováním tak, aby se byla síť více zaměřena na těžké

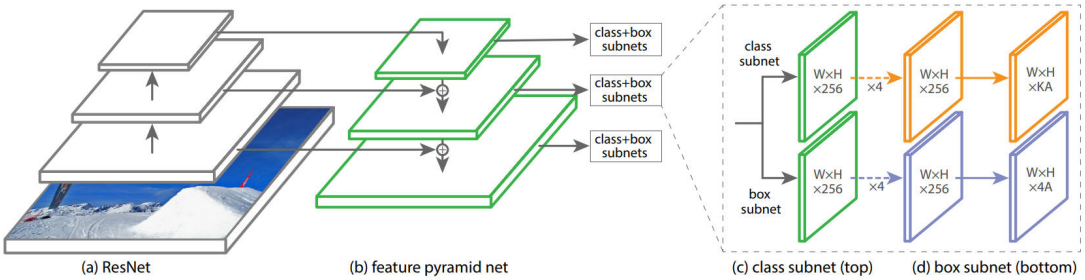


Obrázek 3.16: Architektura algoritmu SSD (Single Shot MultiBox Detector). Algoritmus nejprve na svůj vstup aplikuje síť typu VGG-16 [58]. Následně je obrázek transformován konvolučními vrstvami a jsou tak vytvořeny víceúrovňové příznaky. Hlubší konvoluční vrstvy tak pokrývají větší receptivní pole a učí se více abstraktní příznaky. Každá tato úroveň se následně přímo podílí na výpočtu výsledků detekce. Obrázek převzat z [33].

příklady a efektivněji se tak učila. Lehkým příkladům je tady dána malá váha a těžkým naopak větší. Proces učení se tak stává stabilnějším a efektivnějším.

RetinaNet podobně jako Mask R-CNN zakládá svou architekturu na Feature Pyramid Network (FPN) a ResNet. Pracuje tak na příznakové mapě z různých úrovní.

Díky Focal Loss se RetinaNet přiblížila svou přesností mnohem blíže dvoufázovým variantám, přičemž si ponechala svou výhodu v podobě rychlosti. Autoři publikovali výsledky na COCO *test-dev* [32] datasetu, kde RetinaNet překonala všechny jedno i dvoufázové řešení (v té době reprezentováno Faster R-CNN). Dle oficiálních výsledků COCO soutěže bylo řešení RetinaNet překonáno Mask R-CNN.



Obrázek 3.17: Architektura algoritmu RetinaNet využívající FPN [30] jako extraktor příznaků. Pracuje tedy s příznaky z více úrovní. Hlavním přínosem publikace je však chybová funkce. Obrázek převzat z [31].

Kapitola 4

Existující přístupy detekce defektů při výrobním procesu

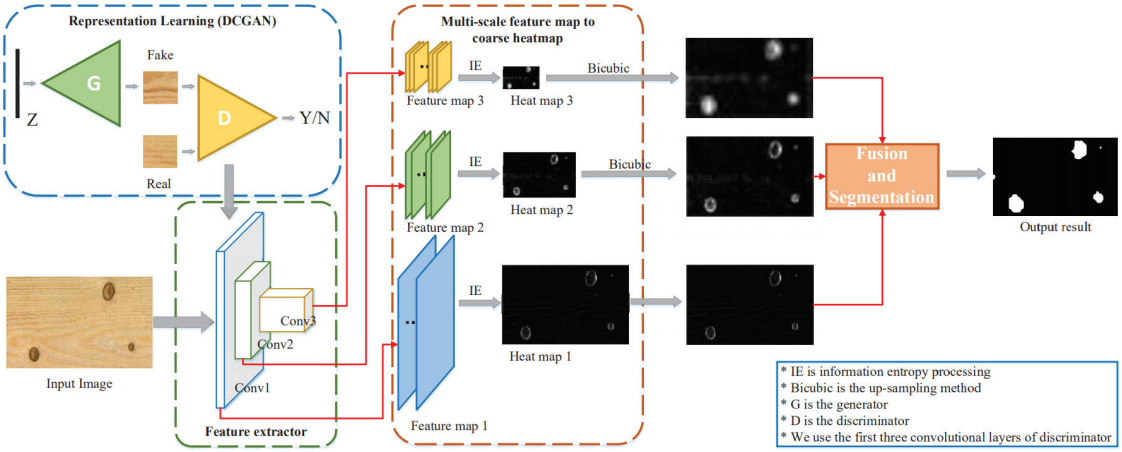
Automatizovaná detekce defektů ve výrobním procesu za použití počítačového vidění a strojového učení se těší v průmyslových aplikacích velké pozornosti. Jedná se o zajímavý způsob jak zefektivnit výrobu a získat konkurenční výhodu. Obor autonomní vizuální inspekce se však potýká s velkou řadou výzev. Struktura povrchu výrobku je v reálných podmínkách velice proměnlivá a nekonzistentní. S tím souvisí těžká definovatelnost hledaného defektu, jehož podoba se obtížně predikuje.

Zároveň je v tomto oboru náročné získat potřebná data k vytvoření takovéhoto algoritmu. Kontrolované získání dostatečného počtu entit obsahující vadu je často nemožné. Algoritmy se tak často vydávají cestou použití pouze dat bez vad a následné vytvoření modelu zachycujícího jejich strukturu. Následně jsou příchozí data kontrolována vůči tomuto modelu a jsou sledovány odchylky potencionálně znamenající defekt, podobně jako pracují metody pro hledání anomálií.

V této kapitole se budu věnovat publikovaným technikám a vytvořím tak přehled aktuálních řešení. V řadě následujících prací se objevují řešení využívají různé variace aplikace autoenkodérů, GANů a i detekčních algoritmů. Těmto technikám byla věnována předchozí kapitola 3.

W. Zhai a spol. [68] představili metodu založenou DCGAN modelu. Tímto modelem vytvořili vnitřní reprezentaci dat neobsahující žádné vady. Získali tedy model, který podobně jako v [56], dokázal generovat data bez vad. V tomto případě však W. Zhai a spol. dále využili diskriminátor DCGAN modelu. Diskriminátor pojali jako jednotřídní klasifikátor. K odhalení abnormálních dat použili příznaky získané z prvních třech vrstev diskriminátoru. Jejich myšlenka byla postavena na tom, že odezva diskriminátoru bude pravděpodobně velmi citlivá a silná v případě abnormálních dat. Na základě výstupů těchto vrstev diskriminátoru následně získali segmentaci abnormálních regionů. Přístup je ilustrován na obrázku 4.1.

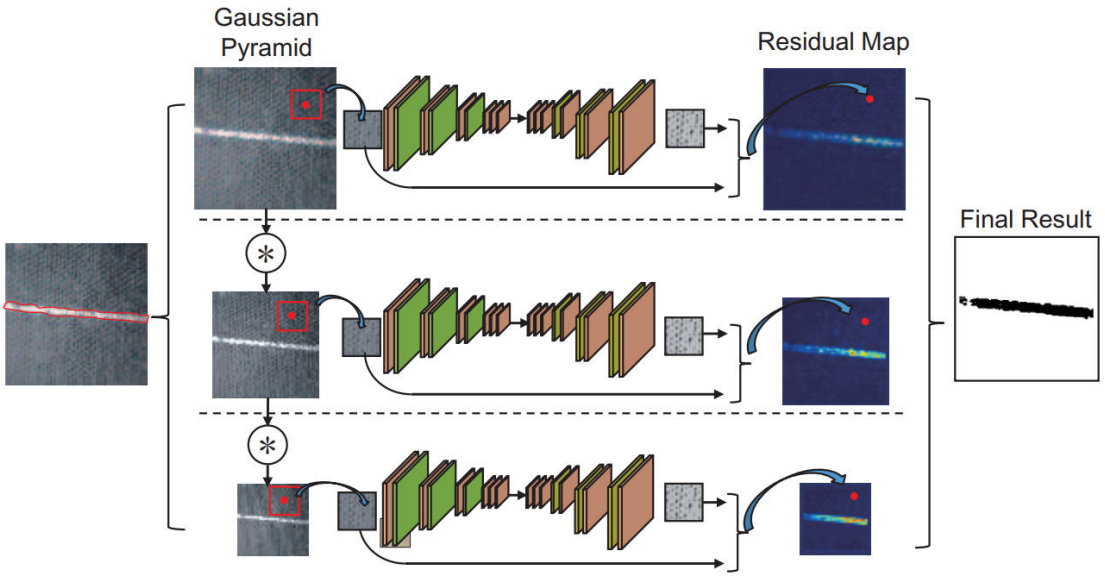
S. Mei a spol. [40] se taktéž z důvodu nedostatku anotovaných dat a nejasného vzhledu hledaného defektu vydali cestou modelování distribuce podmnožiny dat, která neobsahovala žádné vady. K tomuto účelu využili konvoluční denoising autoenkodér. Vstupní data nejprve transformovali do tzv. Gaussovy pyramidy o rozlišení různých úrovní. Autoři vycházeli z předpokladu, že denoising konvoluční autoenkodér dokáže namodelovat distribuci dat bez vady. Autoenkodér se tedy dokáže naučit jejich reprezentaci a z té je dokáže dostatečně kvalitně rekonstruovat. V případě, kdy na vstupu budou data doposud neviděná, tedy anomální



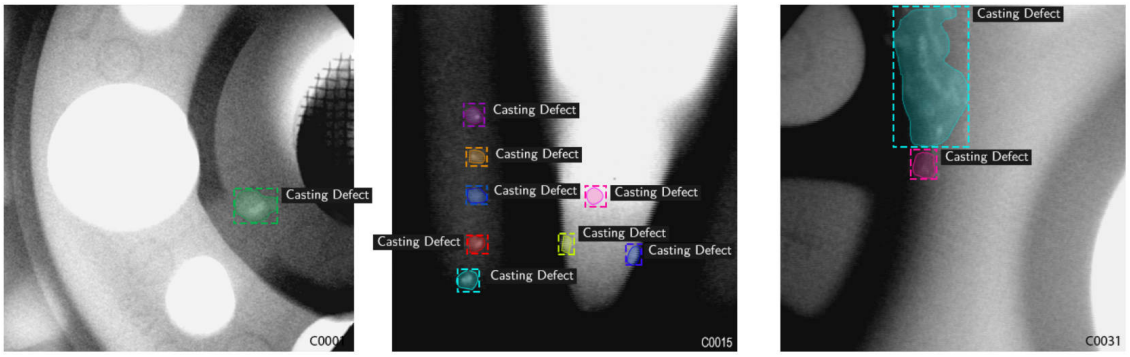
Obrázek 4.1: W. Zhai a spol. [68] využili DCGAN model pro reprezentování distribuce dat bez vady. V čase inference pro detekci anomálií použili aktivační mapy z prvních tří vrstev diskriminátoru. Obrázek převzat z [68].

a obsahující defekt, autoenkodér bude mít problém takováto data správně rekonstruovat. Defekty jsou tedy predikovány na základě rekonstrukční chyby autoenkodéru. Přístup je přiblížen na obrázku 4.2.

Y. Huang [23] se ve své práci vydal směrem supervizovaného učení a model trénoval přímo na datech obsahující defekty. Nedostatečné množství dat se snažil kompenzovat jejich silnou augmentací a rozšíření datasetu tímto způsobem. Jeho cílem bylo predikovat segmentační mapu defektů připomínající praskliny. Autoři se vydali cestou tréninku modelu predikující informaci na úrovni pixelů. K tomu využili model založený na architektuře U-Net [53].



Obrázek 4.2: S. Mei a spol. [40] reprezentovali distribuci dat bez vady pomocí autoenkodéru. Případné anomálie následně detekovali na základě rekonstrukční chyby autoenkodéru. Obrázek převzat z [40].

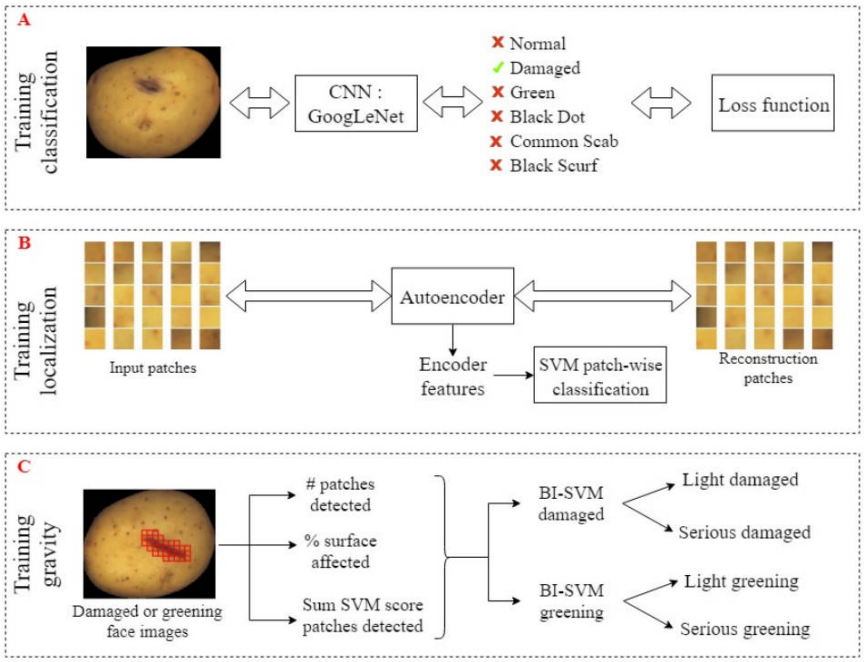


Obrázek 4.3: M. Ferguson a spol. [14] řešili detekci defektů na rentgenových snímcích řady kovových objektů. Jako cestu k řešení problému zvolili detekční algoritmus Mask R-CNN [19]. Obrázek převzat z [14].

M. Ferguson a spol. [14] ve své práci řeší detekci defektů na rentgenových snímcích kovových výrobků (příklad zpracovávaných dat je na obrázku 4.3). K uvedenému úkolu se postavili jako k detekčnímu problému. Použili jeden z detekčních algoritmů, konkrétně Mask R-CNN. Nedostatek dat v jeho případě kompenzoval technikou transfer learning, kdy váhy extraktoru příznaků (backbonu) nejprve inicializovali tréninkem na datasetu ImageNet [11], následně Mask R-CNN model trénovali na detekčním datasetu COCO [32]. Teprve takto předtrénovaný model aplikovali a trénovali na jejich doméně dat. Autoři dosáhli pozoruhodného výsledku 0,950 při IoU 0,5. V publikaci je taktéž často poukazováno na fakt, že velikost datasetu má naprosto marginální vliv na výkonnost tohoto řešení.

Autoři R. Ren a spol. [50] se vydali cestou vytvoření jednoduché konvoluční sítě, kterou trénovali na anotovaných výřezech datasetu. Pro každý výřez regressovali pravděpodobnostní hodnotu, zda se jedná o výřez obsahující defekt. V čase inference takovouto natrénovanou síť konvolučně aplikovali na celý obrázek a získali tak sémantickou mapu predikující výskyt defektu.

Ačkoliv přímo nejedná o detekci defektů při výrobním procesu, S. Marino [38] řeší automatickou vizuální kontrolu kvality brambor. V první fázi byla pro tuto aplikaci použita konvoluční neuronová síť a tou brambor klasifikován do několika tříd reprezentující stav bramboru (normální, zelenající, obsahující černé skvrny atd.). V druhé fázi bylo třeba určit rozsah poškození reprezentovaný hrubou sémantickou mapou. Obraz byl rozdělen do výřezů, na kterých byl natrénován autoenkodér. Model autoenkodéru byl následně použit pro extrakci příznaků a ty byly následně klasifikovány SVM. Takto byla získána hrubá sémantická informace pro každý z výřezů a na základě toho odvozen rozsah poškození bramboru. Přístup je zobrazen na obrázku 4.4.



Obrázek 4.4: Na obrázku je znázorněna architektura systému pro kontrolu kvality brambor. V první fázi je každý jeden brambor klasifikován neuronovou sítí do tříd odpovídají vadám. V druhé fázi je obrázek bramboru rozdělen do menších výřezů a každý tento výřez je klasifikován. V druhé fázi je tak získána přesnější informace o rozsahu škody. Obrázek převzat z [38].

Kapitola 5

Detekce defektů při výrobním procesu plastových karet

V této kapitole se zaměřím na praktickou aplikaci získaných poznatků, a to při detekci defektů při výrobě plastových karet. Jedná se o karty typu občanského nebo řidičského průkazu, platební karty nebo věrnostní karty obchodních řetězců, tedy plastikovou kartu libovolného potisku o velikosti $85,60 \times 53,98$mm.

Ve výrobním procesu je nejprve na společný plát vytištěna základní textura karty (pozadí). Každá jedna kartička je z tohoto výchozího plátu následně vyřezána. Do kartičky jsou pak dále přidávány různé varianty bezpečnostních prvků nebo embosování. Další možností kustomizace je například vypálení jména budoucího majitele nebo podobné specifické informace. Příklad mnoha variant takovýchto kartiček je na obrázku 5.2.

Při uvedeném procesu se mnohokrát stane, že výstupní produkt nesplňuje požadavky na výstupní kvalitu. Závada se může projevit v jakékoliv fázi výroby. Příkladem může být chybně vytištěné pozadí, chybějící nebo špatně lokalizované bezpečnostní prvky nebo kontaminace kartičky nečistotou. Příklady defektů je na obrázku 5.1. Nevyhnutelnou součástí výrobního procesu je tedy finální kontrola kvality kartiček. Ta je však aktuálně prováděna manuálně lidským operátorem, což je značně časově náročné a taky drahé.

Cílem mé práce je minimalizace lidského úsilí při této kontrole kvality a vytvoření algoritmu, který tuto činnost zvládne automaticky. V této práci se zaměřím na detekci defektů způsobených kontaminací prachovou částicí nebo vlasem.

Požadavky na práci algoritmu jsou tedy následující. Vstupem bude digitální kamerový snímek kartičky s garantovanou kvalitou snímání. Zabezpečení snímacího aparátu není předmětem této práce.

Každý kus bude zpracován okamžitě přímo po opuštění výroby. Doba na práce algoritmu by neměla překročit dobu práce lidského operátora. Ideálním cílem je však zpracovat 3000 kartiček za minutu, tedy přibližně 800 milisekund na jednu kartičku, což pro dnešní algoritmy není nedosažitelný cíl.

Na hardware neexistují žádné specifické omezení. Jelikož bude algoritmus založen na použití neuronových sítí, předpokládá se využití GPU. Algoritmus bude spuštěn lokálně na pracovišti.

Výrobce produkuje velké množství typů, přičemž každý z těchto typů má rozdílné požadavky na výstupní kontrolu kvality. Například věrnostní kartička pro supermarket má menší důraz na úroveň výstupní kvality v porovnání s občanským nebo řidičským průka-


Obrázek 5.1: Na obrázku je ukázána řada plastikových kartiček, kde každá obsahuje vadu vzniklou při její výrobě. Jedná se v nejčastějším případě o kontaminaci prachovými částicemi nebo vlisovaná vlákna. Defekty jsou často velice malé a nezřetelné.

zem, kde je tolerance vad naprosto minimální. Algoritmus by tedy měl disponovat jakousi měkkou rozhodovací hranicí v závislosti na závažnosti vady, kterou si uživatel určí podle svých potřeb. Nejdůležitějším výstupem pro danou kartičku je informace, zda se na ní nachází nějaká vada. Lokace vady a její případná segmentace není v tomto případě stěžejní, nicméně by se jednalo o příjemný benefit řešení.

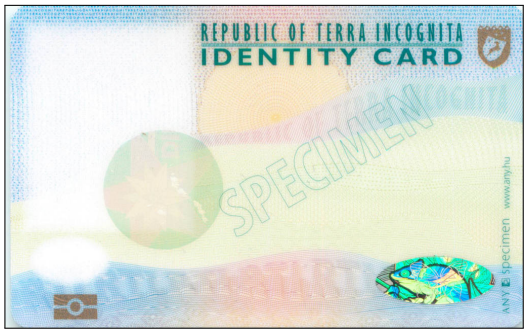
5.1 Příprava dat

Data jsem měl k dispozici v podobě mnoha fyzických plastových kartiček několika typů, jak je ukázáno na obrázcích 5.2 a 5.1. Od každého typu kartičky bylo k dispozici přibližně 200 kusů.

Pro svou práci jsem si zvolil kartičku simulující vzorek občanského průkazu, konkrétně její přední stranu (obrázek 5.1). Tento typ jsem zvolil proto, že jde o nejkomplexnější kartičku, kterou jsem dostal k dispozici. Pozadí této kartičky je ze všech typů nejméně uniformní, obsahuje velké množství proměnlivých tvarů, opakujících se vzorů, mikro textů a bezpečnostních prvků. Předpokládám, že pokud dokážu úkol řešit na takto složité kartičce, ostatní jednodušší kartičky by měly být navrhnutým přístupem bez obtíží zvládnutelné. Z této třídy mám k dispozici celkem 217 kusů.

Skenování. V prvním kroku bylo nutné převézt fyzické karty do digitální podoby jako na obrázku 5.2. K tomu posloužil běžný kancelářský skener a karty byly převedeny do tříkanálového barevného obrazu o rozlišení 2040×1278 pixelů.

Obrázek 5.2: Příklad různých typů vyráběných karet a také formátu vstupních dat. Z důvodu dostatečného počtu kusů a největší komplexnosti pozadí jsem se rozhodl svůj přístup otestovat kartičce vlevo nahoře. Jedná se o přední stranu ukázkové kartičky občanského průkazu.



Obrázek 5.3: Součástí předzpracování dat bylo zarovnání všech kartiček vůči společnému vzoru a následně spočítána střední hodnota pro každý pixel. Od každé kartičky byla následně střední hodnota odečtena. Tímto jsem zvýraznil případné anomálie. Na levém obrázku je původní kartička, na pravé straně je potom kartička s odečtenou střední hodnotou.

Zarovnání. Dalším krokem v předzpracování dat bylo převedení dat do společné normalizované polohy. Cílem bylo získat kartičky zarovnané na sebe, což usnadní další operace nad daty jako například spočítání průměrného obrazu kartičky. Zarovnaní bylo dosaženo extrakcí zajímavých bodů ORB [54], nalezením homografie vůči zvolenému cílovému obrazu a aplikací získané transformace.

Pořízení středního obrazu. Pro další usnadnění výpočtu a práce s daty jsem vypočítal střední hodnotu obrázku pro celý dataset. Tento střední obrázek jsem použil pro normalizaci celého datasetu. Výsledná střední hodnota je na obrázku 5.4.

Další příjemnou vlastností vypočítané průměrné kartičky je, že obsah defektů v ní byl minimalizován. Odečtením střední kartičky tedy získáme více viditelné defekty a anomálie, jako je ukázáno na obrázku 5.3

Tento fakt mi subjektivně pomohl při anotaci dat, kdy jsem nejdříve dataset anotoval v jeho původním formátu. Následně jsem měl k dispozici obrázky s odečtenou hodnotou, kde byly defekty výrazně viditelnější a podařilo se mi výslednou anotaci ještě více zkvalitnit.

Anotace dat je neodlučitelnou součástí přípravy dat pro úlohu tohoto typu. Anotaci jsem vyhotovil v podobě binárních masek, a to pro každou instanci zvlášť. Každý defekt má tedy vlastní anotaci.

Anotaci jsem provedl pomocí nástroje COCO Annotator[1]. Jedná se o webově založená aplikace pro anotaci datasetů pro lokalizaci a detekci objektů. Anotační nástroj jako výchozí možnost podporuje export do COCO formátu anotací[2].

Tímto způsobem jsem anotoval všech 217 obrázků. Celkově bylo anotováno 3964 instancí vad, tedy $18,26$ vad na jeden obrázek. Příklad formátu anotace a anotovaných defektů je na obrázku 5.5.

Průměrná plocha jedné instance anotace byla 179 pixelů. To je k kontextu celého obrazu o rozlišení 2040×1278 pouze $0,0068$ % plochy obrazu, což není mnoho.

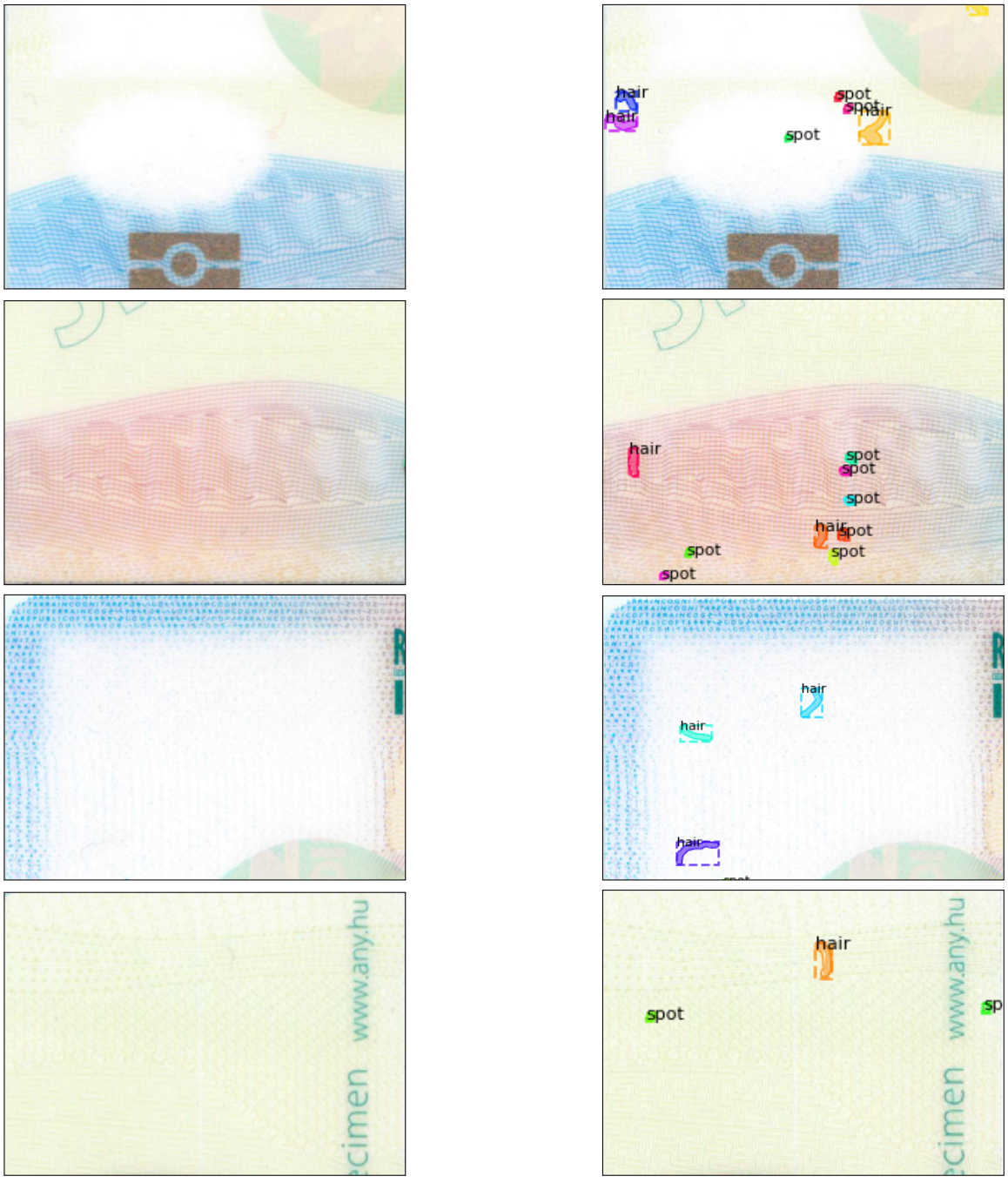
Anotace obsahuje velké množství krajních případů, kdy jsem si sám nebyl jistý, zda se jedná o vadu a nebo ne. Přistupoval jsem k anotaci co možná nejpřísněji. Jako vady jsou tedy označené i minimálně viditelné instance.

[1] COCO Annotator. 2019. URL: https://github.com/jsbroks/coco-annotator
[2] COCO Dataset, Data Format. URL: http://cocodataset.org/#format-data

Obrázek 5.4: Z celého datasetu jsem pro každý pixel spočítal střední hodnotu, kterou zobrazuje tento obrázek.

Binární masky jsem z důvodu uspoření paměti ukládal jako řídké matice, a to pouze souřadnice pixelů, kde jejich hodnota nebyla nulová. Další pomocné struktury jsem uložil jako tabulku do souboru h5.

Rozdělení na trénovací a testovací množinu. Anotovaná data jsem následně rozdělil do trénovací a testovací podmnožiny o počtu 165, respektive 46. Dalších 6 souborů jsem z datasetu vyřadil kvůli nedostatečné kvalitě nebo dalšímu poškození (například poškození způsobené označením fixem).



Obrázek 5.5: Finální anotace vad kartičky v podobě binární masky. V levém sloupci se nachází výřez zdrojového obrázku s defekty. V pravém sloupci se pak nachází korespondující výřez s binární anotací. Za povšimnutí také stojí komplexnost pozadí a špatná viditelnost hledaných vad. Průměrný počet anotovaných instancí vad na jednu kartičku byl $18,26$.

5.2 Návrh řešení

V případě hledání vad na plastikové kartičce se jedna o složitý problém. Na ploše kartičky o rozlišení 2040×1278, hledáme objekty s průměrnou plochou 179 pixelů. Pozadí kartičky je značně variabilní a obsahuje velké množství barev, periodických struktur a mikrotextů. Hledané vady taktéž nejsou na první pohled zřetelně viditelné. Příklady vad jsou pro lepší představu na obrázku 5.5.

Hledáme tedy algoritmus, který je schopen detekovat miniaturní špatně viditelné vady relativně k velké ploše kartičky. Zároveň by pro výsledné řešení bylo benefitem, kdyby výstupem algoritmu byla i segmentace a poloha vady. Další výzvou v této úloze je relativně malá velikost dostupné datové sady. K dispozici pro trénink mám pouze 165 obrázků.

Prvním přístupem k řešení problému jsem použil znalosti detekce anomálií, neboli outlierů, a pokusil jsem se aplikovat klasické přístupy. Využij jsem také faktu, že mám všechna data na sebe zarovnaná. Z každé jedné kartičky jsem na stejném místě vyřezal čtvercovou část obrazu. Z této části jsem následně vyextrahoval příznakové vektory. Zkoušel jsem například histogramy gradientů HOG [10] nebo předtrénovanou neuronovou síť BVLC GoogleNet[3] [60]. Myšlenkou tohoto přístupu je, že pokud je pozadí mezi výřezy minimálně variabilní, měla by se přítomnost vady dát mezi příznakovými vektory odhalit. Touto metodou se mi však nepodařilo docílit uspokojivých výsledků, proto se jí v této práci nebudu dále věnovat.

Mask R-CNN. Po zkušenosti s prvotním přístupem, jsem se rozhodl změnit přístup k hledání řešení. V posledních letech neurosíťové detekční algoritmy zaznamenaly veliký posun vpřed. Konkrétně jsem se rozhodl pro použití rodiny detekčních algoritmů R-CNN [16], Fast R-CNN [15], Faster R-CNN [51] a nedávno představenou iteraci tohoto přístupu Mask R-CNN [19].

Další variantou při výběru detekčního neurosíťového algoritmu byla podmnožina jednofázových algoritmů typu YOLO [47, 48, 49], SSD [33], RetinaNet [31] a další. Tato skupina se vyznačuje především rychlostí výpočtu a je tak často využívaná jako detektor objektů ve videu. Pro mou aplikaci by však rychlost Mask R-CNN měla být naprosto dostatečná. Dalším benefitem Mask R-CNN potencionálně vyšší přesnost a navíc výstup kromě samotných detekcí obsahuje i segmentační pixelovou mapu. Navíc přístupy typu YOLO mají problémy s velmi malými objekty. V mém případě defekty zaujímají naprosté minimum plochy celé kartičky.

Pro vyřešení detekce defektů na plastikové kartičce jsem se rozhodl použít algoritmus Mask R-CNN, konkrétně jeho publikovanou implementaci od firmy Matterport [2].

Potencionální rizika. Zvolení Mask R-CNN jakožto neurosíťového detekčního algoritmu pro řešení mého problému s sebou nese potencionální rizika. Největší obavu mám z nedostatečně velké trénovací sady, kdy mám k dispozici jen 165 obrázků pro trénink.

Riziko nedostatečně velké trénovací sady by mohlo být redukováno použití již předtrénovanou neuronovou sítí na jiném problému a použít takzvaný transfer learning. Použitý model má k dispozici natrénované váhy na COCO datasetu [32], ze kterých budu vycházet a použiji je pro inicializaci modelu.

[3] Caffe Models, BVLC GoogleNet. URL: https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet

Další obavu mám nedostatečné velikosti hledaných objektů relativně k velikosti obrázku. Nejsem schopen odhadnout, zda na takovémto problému bude detekční algoritmus Mask R-CNN schopen fungovat. Zároveň plánuji použít model, který byl předtrénován na COCO datasetu, což je dosti odlišný problém od mého. V mých experimentech proto vyzkouším jak trénování pouze části heads s konstantními vahami backbonu (extraktoru příznaků), tak trénink celé sítě včetně backbonu.

Použité technologie. Pro realizaci algoritmu jsem použil programovací jazyk Python společně s knihovnami pro neuronové sítě Keras a Tensorflow také s nástrojem pro vizualizaci logovaných záznamů o tréninku Tensorboard. Portabilitu řešení jsem zajistil vytvořením běhového prostředí v Dockeru. Pro práci s daty jsem používal převážně knihovnu Pandas, NumPy a OpenCV v případě obrazových dat. Data jsem následně ukládal do formátu H5. Pro vyhodnocení jsem použil knihovnu Scikit-Learn.

5.3 Konfigurace Mask R-CNN

V této sekci popíšu, jakým způsobem jsem zvolil konfiguraci modelu Mask R-CNN pro dosažení co možná nejlepších výsledků.

Rozlišení vstupních dat. Jelikož jsou všechny mé vstupní data ve stejném rozlišení, není třeba u nich provádět žádné škálování. Z důvodu zachování co nejvyšší možné přesnosti jsem zvolil zachovat maximální dostupné rozlišení. Jednou z podmínek použité implementace bylo uvést data do čtvercového tvaru. Data jsem tedy na jejich delší straně doplnil černými pixely na finální velikost 2048×2048.

Nastavení RPN. Pro trénink RPN jsem použil anchory o velikostech $[16; 32; 64; 96; 128]$ a poměrech stran $[0,5; 1; 2]$. Velikost anchorů vůči celému obrázku je na obrázku 5.6. Anchory o této velikosti jsem tvořil se stridem 1. Pro trénink RPN je takto vybráno 256 anchorů na obrázek. V dalším kroku je jich 200 použito pro trénink dalších stupňů sítě. Těmi jsou regressor bounding boxu, klasifikátor třídy detekce a segmentace masky. Data pro tyto stupně jsou zvolena tak, aby počet pozitivních anchorů byl 33%. Po RPN je aplikováno non-maximum suppression (NMS) pro intersection over union (IoU) větší než 0,7.

Dále jsem použil learning rate 0,001. Kvůli velikosti vstupních obrázků jsem byl schopen trénovat s velikostí batche jednoho obrázku na GPU. Trénink probíhal na grafických kartách Nvidia GTX TITAN X s pamětí 12GB.

Vstupní data. Pro načítání dat do sítě jsem vytvořil třídu k tomu určenou. Tato třída pro maximální rychlost udržuje anotace datasetu a pomocné struktury v operační paměti pro celou dobu běhu. Pro každou iteraci je načten obrázek z datasetu. K tomuto obrázku je pro každou instanci anotace vytvořena binární maska s příslušnou třídou. Výstupem třídy je tedy obrázek, list masek o délce korespondující s počtem instancí vad a také list tříd shodné délky.

Obrázek 5.6: Příklad velikosti anchorů vůči celému obrázku.

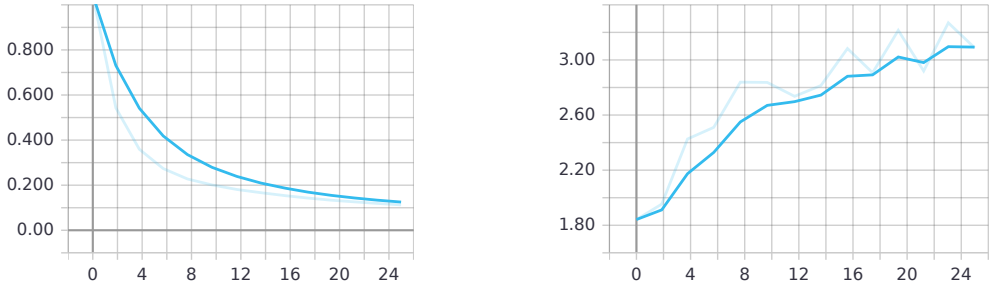
5.4 Trénink na reálných datech

Následujícím krokem bylo spuštění samotného tréninku na trénovací množině reálných dat. Chybová funkce modelu vykazovala na trénovacích datech klesající tendenci. Na validačních datech (neobsažených v trénovací sadě) však již od začátku stoupala, což se s pokračujícím tréninkem jen zhoršovalo. Model se tedy dokázal na trénovacích datech učit a minimalizovat chybu, ale nedokázal následně generalizovat na datech doposud neviděných. Vykazoval tedy známky přeučení (overfittingu). Výsledek tréninkových chybových funkcí je na grafech na obrázku 5.7.

Potencionální riziko nedostatečného množství dat se tedy ukázalo jako reálným problémem a model nebyl schopen se dostatečně dobře natrénovat. Řešením by mohlo být omezení kapacity modelu a jeho regularizace. Nicméně jsem použil již existující model a chci se vyvarovat jeho editaci.

Další cestou by mohlo být přímé řešení problému nedostatečné velikosti datové sady a tedy její rozšíření. K tomuto účelu se používají techniky augmentace, kdy jsou vstupní data náhodně transformována. Mezi takovéto transformace patří například ořez na náhodné poloze, převrácení obrazu v horizontální nebo vertikální ose, přidáváním šumu, rotace a mnoho dalších.

Já jsem zvolil podobnou cestu, a to augmentaci dat pomocí jejich generování a následný trénink modelu na syntetickém datasetu.



(a) Tréninková chybová funkce

(b) Validační chybová funkce

Obrázek 5.7: Obrázek ukazuje tréninkovou chybovou funkci (a) a validační chybovou funkci (b) tréninku na reálných datech. Trénink vykazuje známky přeučení, kdy modelu klesá chybovost na trénovacích datech, ale ztrácí schopnost generalizace na datech mimo tréninkovou sadu. Na svislé ose je hodnota chybové funkce, na vodorovné je pak čas v hodinách.

5.5 Augmentace dat

V předchozí kapitole jsem ukázal, že zvolený model má problém natrénovat se na dané tréninkové datové sadě a zároveň generalizovat. Z toho důvodu jsem zvolil dalším postup práce v podobě augmentace datové sady, konkrétně její rozšíření o syntetické defekty.

Vycházím z předpokladu, že v doméně mého úkolu vzniká uzavřená množina defektů. Mám tím na mysli, že prachové částice a vláskové částice opakující se na kartičkách se s určitou variací stále opakují. Pokud dokážu tuto distribuci namodelovat, dokážu natrénovat i kvalitní detektor, který snad bude generalizovat i na doposud neviděné tvary defektů.

Návrh. Ke generování defektů přistoupím tak, že vytvořím konečný počet binárních masek co nejvěrněji napodobující distribuci reálných defektů. Pomocí těchto masek budu následně do původních obrázků syntetizovat defekty. Následně na těchto syntetických datech natrénuji model a budu studovat jeho funkčnost opět na reálných datech.

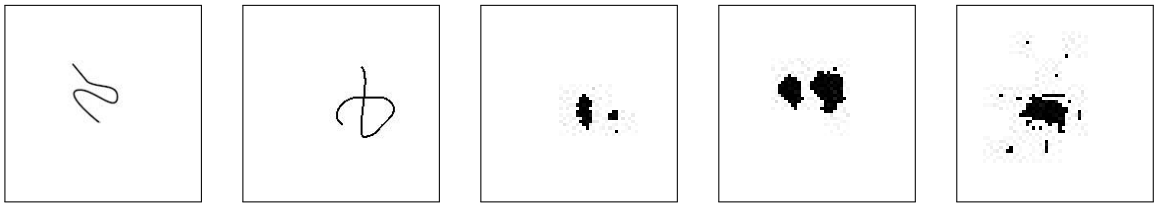
Výhodou takto vygenerovaného datasetu může být fakt, že za doby jeho generování volím parametry vytvoření každé jedné vady. Tyto informace si budu ukládat a dále je mohu využít ke studiu chování a výkonnosti modelu. Například budu ukládat informaci o velikosti defektu a intenzitě blendování do původního obrázku. Nakonec mohu slabě viditelné vady vyfiltrovat a ověřit funkčnost modelu pouze na zřetelných vadách. Tyto informace mi mohou být cenné při rozhodování o dalším postupu vývoje.

V prvním kroku jsem tedy vytvořil 90 masek imitujících reálné defekty. K vytvoření této sady jsem využil nástroj GIMP[4]. Masky jsou vytvořeny pouze s jedním kanálem. Podmnožina těchto masek je ukázána na obrázku 5.8.

Následně jsem tyto masky s různými parametry vkládal do původních obrázků. Tyto parametry a jejich hodnoty jsou popsány v tabulce 5.1. Každá jedna augmentace defektu byla ještě navíc podmíněna pravděpodobností a provedena jen v 90 % případů. Tyto defekty jsem vkládal uniformě po celé ploše obrázku. Celkem jsem do jednoho obrázku vložil 30 až 50 jednotlivých defektů.

Porovnání syntetických a reálných defektů je na obrázku 5.9. Subjektivně se touto metodou podařilo věrně napodobit vzhled defektů.

[4]GNU Image Manipulation Program, GIMP 2.10.10, URL: https://www.gimp.org/, 1997-2019



Obrázek 5.8: Trénovací sadu jsem rozšířil přidáváním umělých defektů. Obrázek zobrazuje jednotlivé masky, ze kterých byly defekty v obrázku syntetizovány. První dva obrázky modelují vlasové vady, zatímco další tři masky bodové prachové částice. Celkově jsem vytvořil 90 takovýchto masek.

Typ augmentace defektu	Parametry augmentace (hranice uniformního rozdělení parametrů)
Rotace	(-180, 180)
Gaussovo rozostření	kernel_size = (3, 7), sigma = (1, 5)
Gaussův šum	mean = 0,0, std = 0,013
Měřítko	(0,4; 1,0)
Síla blendingu	(0,1; 0,4)
Třída defektu	50:50 pro dvě třídy (bodová částice, vlasový defekt)

Tabulka 5.1: V tabulce je uveden seznam jednotlivých transformací použitých při tvorbě falešných defektů. Každá tato transformace měla volitelné parametry. Tyto parametry byly při tvorbě datasetu náhodně vybírány s uniformním rozdělení pravděpodobnosti.

Potencionální rizika této metody mohou způsobena neúplným pokrytím distribuce reálných vad. Jak lze vidět na obrázku 5.9, touto metodou jsem nebyl schopen imitovat defekty využívající celé RGB spektrum a disponující jakýmsi duhovým efektem. Na tyto reálné vady se zaměřím při vyhodnocení modelu a jsem zvědavý na jeho funkčnost v těchto případech.

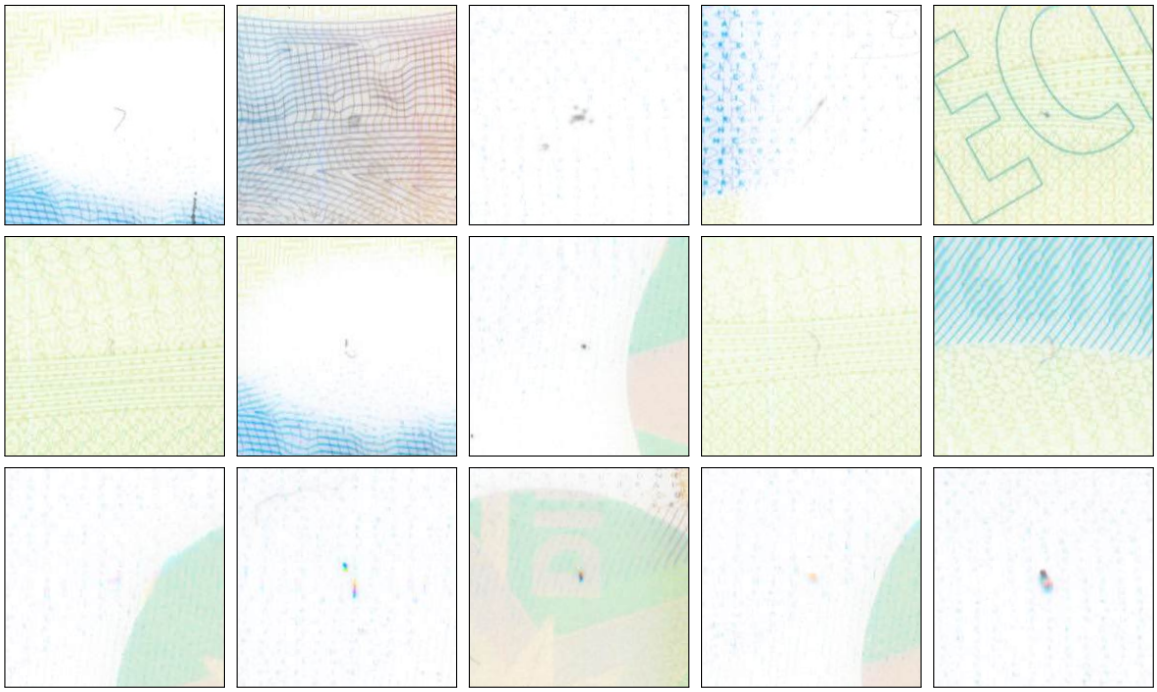
Tvorba datasetů. Při původní přípravě dat jsem vytvořil dvě množiny. Reálnou trénovací a reálnou testovací množinu.

Při tvorbě syntetického datasetu jsem na tyhle dvě množiny obrázků navázal. Výstupem byly celkově tři datasety: reálný testovací, syntetický testovací a syntetický trénovací dataset. Tyto datasety byly vytvořeny následovně.

Reálný testovací dataset je shodný s datasetem popsaným dříve. Skládá se tedy ze 46 obrázků obsahující pouze reálné vady.

Syntetický testovací dataset byl vytvořen na základě reálného testovacího datasetu. Tyto obrázky byly rozšiřovány o syntetické vady. Celkový počet obrázků v tomto datasetu je 1000. Dataset navíc obsahuje meta informace o parametrech vad pro případné další testování algoritmu.

Syntetický trénovací dataset je založen na 165 reálných obrázcích. Tyto obrázky byly opět rozšířeny o syntetické vady, až do celkového počtu 20 000 obrázků.



Obrázek 5.9: Trénovací sadu jsem rozšířil přidáváním umělých defektů. V rámci rozšíření datové sady jsem se snažil co možná nejvěrněji napodobit reálné vady. V první řadě jsou ukázány obrázky obsahující uměle vytvořené defekty. Druhá řada ukazuje původní reálné defekty. V třetí řadě jsou opět reálné defekty. Jedná se nicméně o mou představenou technikou těžce napodobitelné defekty z důvodu jejich proměnlivé barevnosti. Tento nedostatek bude adresován při nastavení vstupních dat modelu.

5.6 Trénink na syntetických datech

Z důvodu vyhnutí se přeučení jsem se vydal cestou augmentace dat. Pokusil jsem se namodelovat distribuci vad nacházejících se na kartičkách a tyto defekty synteticky vytvářet a vkládat do původních obrázků kartiček. Takovýmto postupem jsem dosáhl vytvoření datasetu o velikosti 20 tisíc obrázků. Na datasetu v tomto kroku provedu trénink modelu.

Zároveň na základě nabyté znalosti o vzhledu vad navrhnu více variant modelu. Vlastnosti navržených modelů a jejich trénink popíšu v této sekci.

Konfigurace jednotlivých tréninků. Na základě znalosti vzhledu vad jsem navrhl více konfigurací modelu Mask R-CNN.

RGB model je shodný s dříve popsaným modelem použitým pro trénink pouze na reálných datech. Vstupem je taktéž obrázek kartičky s odečteným průměrným obrázkem.

RGB backbone model vychází z RGB modelu. Jediným rozdílem je odemčení vah backbonu při tréninku sítě. Původní váhy byly trénovány na COCO datasetu. Od tohoto rozhodnutí si slibuji, že se podaří natrénovat extraktor příznaků specifičtější pro mou doménu dat, která se od COCO datasetu liší.

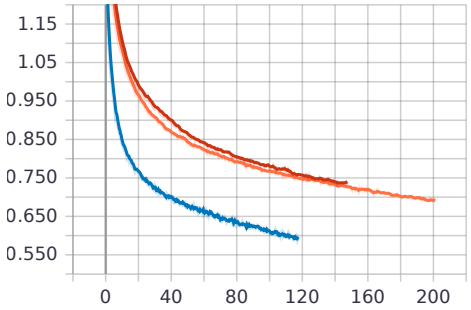
S tréninkem větší části sítě se zároveň zvýšily paměťové nároky modelu. Tentokrát již při 12GB paměti nemohu trénovat na celém obrázku. Možným řešením je snížit rozlišení nebo trénovat na výřezech obrázku. Z důvodu zachování kompatibility s

ostatními modely jsem se rozhodl pro zachování rozlišení a tréninku na výřezech o poloviční velikosti hrany obrázku, tedy čtvrtinové plochy.

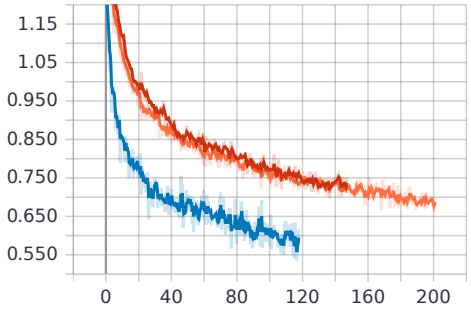
GRAY model opět vychází z RGB modelu. Tentokrát je vstupem do sítě pouze černobílý obrázek. Tímto přístupem si slibuji redukování velikosti distribuce defektů na kartičce. Konkrétně od tohoto modelu očekávám, že bude lépe generalizovat na defekty složené z více barevných složek, jako je ukázáno v poslední řadě na obrázku 5.9. Zároveň se ale obávám snížené přesnosti s redukováním vstupní informace. Také používám předtrénovaný backbone trénovaný na RGB obrázcích, což také může přinést sníženou přesnost.

Následně jsem spustil tréninky modelů. Každý z tréninků probíhal na jednom GPU Nvidia GTX TITAN X a trval přibližně 6 dnů a po dobu 600 tisíc iterací. Poté byl trénink ukončen. Výsledky tréninků v podobě chybových funkcí jsou na obrázku 5.10. Z průběhu těchto funkcí se zdá, že by se pokračováním tréninků byly ještě dále minimalizovány. Avšak na ještě delší dobu už jsem neměl alokován výpočetní stroj.

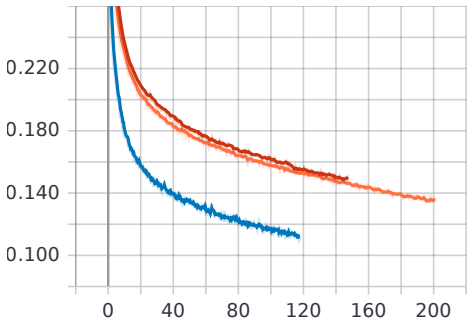
Z podoby chybových funkcí lze předběžně usoudit, že model RGB backbone vykazuje nejlepší vlastnosti. Nejvíce viditelný rozdíl je v chybové funkci patřící RPN, což se dá očekávat, jelikož RPN pracuje přímo nad daty z backbone, kterou jsem v tomto případě trénoval také.



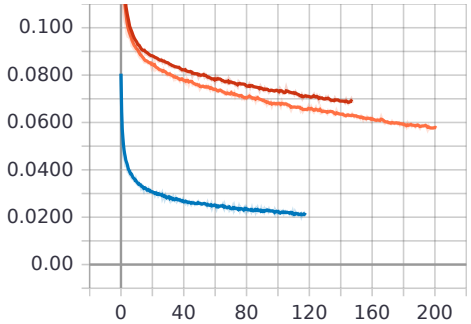
(a) Tréninková chybová funkce



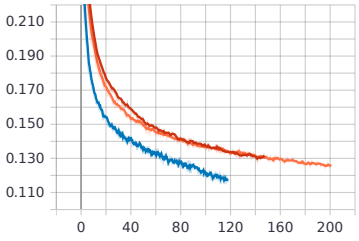
(b) Validační chybová funkce



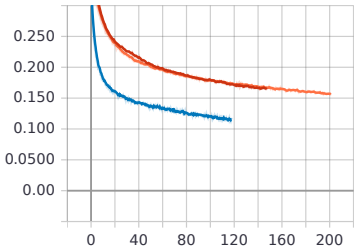
(c) RPN, regressor bounding boxu



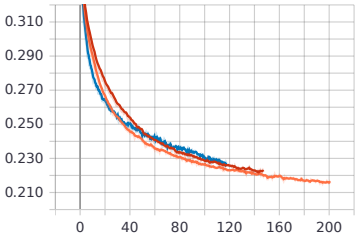
(d) RPN, klasifikátor třídy



(e) Heads, regressor bounding boxu



(f) Heads, klasifikátor třídy



(g) Heads, sémantická segmentace

Obrázek 5.10: Obrázek ukazuje tréninkovou (a) a validační (b) chybovou funkci tréninků modelů RGB (oranžová), GRAY (červená) a RGB-backbone (modrá). Vodorovná osa ukazuje dobu tréninku v hodinách, svislá pak hodnotu chybové funkce. RGB-backbone model v průměru vykazuje nejlepší vlastnosti a to především při tréninku RPN. Na obrázcích (c) a (d) se nachází tréninkové chybové funkce RPN. Tréninkové chybové funkce části sítě heads se nachází na obrázcích (e), (f) a (g).

5.7 Vyhodnocení

Výstupem tréninku jsou 3 modely. V této sekci vyhodnotím jejich inferenci nad reálnou a syntetickou testovací sadou. Přesnost jednotlivých modelů porovnám. Zároveň v úvodu této sekce popíšu použité vyhodnocovací metriky. Na konci pak budou uvedené příklady detekcí a diskutována přesnost a použitelnost modelů.

Metrika vyhodnocení. V této práci budu pro maximální porovnatelnost výsledků primárně následovat metriku použitou v soutěžích v kategorii detekce objektů. Těmi jsou COCO (Common Objects in Context) [32] a starší Pascal VOC (Visual Object Classes) [13].

Pro začátek je třeba definovat řadu nezbytných pojmů. Výstupem detekčního algoritmu je lokace objektu v podobě jeho minimálního ohraničení, což je v našem případě bounding box rovnoběžný s oběma osami obrázku. Každá detekce pak nese informaci příslušnosti ke třídě a skóre, které chápeme jako úroveň spolehlivosti dané detekce.

Výstupy detekčního algoritmu jsou porovnány s ground truth (GT) množinou. Tato množina je taktéž definována poziční informací v podobě bounding boxů a příslušností ke třídě. V případě GT je skóre informace explicitně vždy 1,0.

V prvním kroku je nezbytné pro detekce každého z obrázků najít korespondující GT instance. To provedeme na základě překrytí dvou bounding boxů, tedy na základě hodnoty IoU (Intersection Over Union) definované jako

$$IoU = \frac{plocha(b_{pred} \bigcap b_{gt})}{plocha(b_{pred} \bigcup b_{gt})}, \tag{5.1}$$

kde b_{pred} je predikovaný bounding box a b_{gt} je bounding box patřící GT. IoU tedy pro dva bounding boxy říká jakou plochu zaujímá jejich společný průnik z jejich celkové sjednocené plochy.

Hodnotou IoU je nastavena minimální akceptovatelná přesnost detekce. Detekce je správná jen tehdy, pokud se GT má IoU větší než požadovaná hodnota. V případě vyhodnocování může dojít ke čtyřem možným scénářům, které jsou nazývány následovně:

Správná pozitivní detekce (TP, True Positive): Jedná se o správnou detekci, která má hodnotu IoU s instancí GT větší než zvolená hranice.

Falešná pozitivní detekce (FP, False Positive): Jedná se o detekci, která je špatná. Tedy detekci bez dostatečné IoU s nějakou GT.

Falešná negativní detekce (FN, False Negative): Jedná se o případ, kdy GT instance zůstala bez přiřazené detekce.

Správná negativní detekce (TN, True Negative): TN se typicky nepoužívá. V kontextu vyhodnocování detekcí totiž existuje velké množství těchto případů, kdy není přítomna ani detekce a ani GT.

Dalšími důležitými pojmy jsou precision a recall, které jsou definovány následovně

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{Všechny detekce}}, \tag{5.2}$$

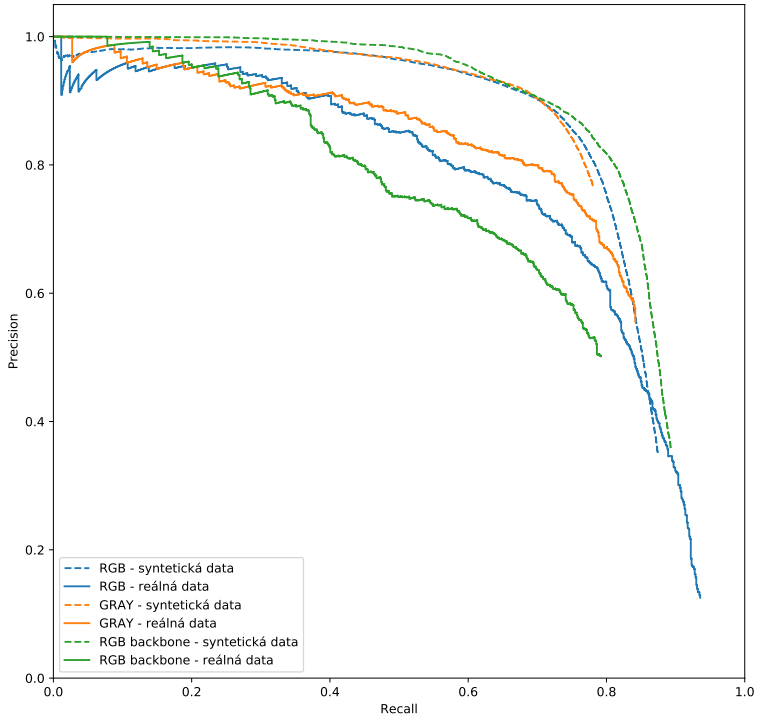
$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{Všechny GT}}. \tag{5.3}$$

Recall tedy udává schopnost modelu detekovat všechny GT entity. Precision zase říká, kolik těchto detekovaných entit bylo správných.

Vodnou metrikou pro popsání přesnosti a celkového chování detekčního algoritmu je precision × recall křivka. Jedná se o dvourozměrný graf, kde je pro proměnlivou hranici skóre vykreslena závislost precision na recall. V ideálním případě chceme, aby se se zvyšujícím recallem zachovávala i vysoká precision.

Další metrikou v návaznosti na precision × recall křivku je definováno tzv. average precision (AP). AP je rovno ploše pod precision × recall křivkou, tedy

$$AP = \int_0^1 P(R)dR, \tag{5.4}$$



Obrázek 5.11: Precision × recall křivky pro všechny modely při IoU 0,1. Dle křivek dosahuje na reálných datech nejlepších výsledků GRAY model.

kde P je precision a R je recall. AP bylo zavedeno pro zjednodušení porovnání jednotlivých detektorů. Namísto obtížného porovnávání více křivek, které se navíc často křižují, se modely porovnávají jedinou numerickou hodnotou AP. V praktických případech se počítá s aproximovanou hodnotou AP numericky spočítanou přes N vzorků jako

$$AP = \sum_{N} (R_N - R_{N-1}) P_N. \tag{5.5}$$

Pro účely vyhodnocení jsem provedl inferenci všech modelů na reálném i syntetickém testovacím datasetu. Pro standardní hodnoty IoU jsem spočítal hodnoty AP a uvedl je v tabulce 5.2. Na základě AP metriky vychází na reálných datech nejlépe GRAY model.

Dále jsem pro každý model na obou datasetech vyhotovil precision × recall křivku. Tyto křivky jsou vizualizovány na obrázku 5.11. Pro každý model jsem dále spočítal bod, kdy je precision roven recallu. Výsledky jsem uvedl v tabulce 5.3.

Z výsledků dle obrázku 5.11 je patrné, že na syntetické sadě dosahuje nejlepších výsledků model, u kterého byly trénovány i váhy backbonu. Tento výsledek koresponduje s grafy chybové funkce při tréninku z obrázku 5.10. Model však nejhůře generalizuje na reálných vadách.

Naopak nejlepších generalizačních schopností dosáhl model trénovaný pouze na datech černobílých, který si se zvyšujícím se recallem nejdéle udržuje hodnotu precision. Shodný výsledek je taktéž patrný z tabulky 5.2 uvádějící AP pro různé IoU hodnoty.

Pro černobílý model jsem následně vytvořil řadu statistik a vizualizací a podrobněji jsem sledoval jeho chování. Vizualizoval jsem množinu příkladů, které model správně detekoval. Výsledkem je matice obrázků správných detekcí 5.13. Dále jsem vizualizoval případy, kde model chybuje. Těmi jsou falešně pozitivní detekce (FP) 5.15 a nedetekované objekty 5.14.

TP a FP jsou seřazeny od nejvyššího skóre modelu. Model je při tvoření vizualizací nastaven tak, aby měl recall roven precision. Skóre, od kterého akceptuji detekci, je tedy podle tabulky 5.3. IoU je nastaveno na 0,1, protože v případě této úlohy není přesnost detekce kritická. 0,1 se tedy jeví jako vhodný bod, kdy neodmítneme řadu ještě pro praktickou aplikaci akceptovatelných detekcí a zároveň. Navíc jsou takto malé objekty citlivé na IoU a nepřesnost o jeden pixel znamená velkou změnu v hodnotě IoU. Detekce v kontextu celého obrázku jsou zobrazeny na obrázku 5.17.

Chybně nedetekované (FN) na obrázku 5.14 jsou typicky velmi slabě viditelné až hraniční případy, kdy si sám nejsem jistý, zda se jedná ještě o vady. Snažil jsem se však anotovat co možná nejpřísněji a zachytit i minimální defekty.

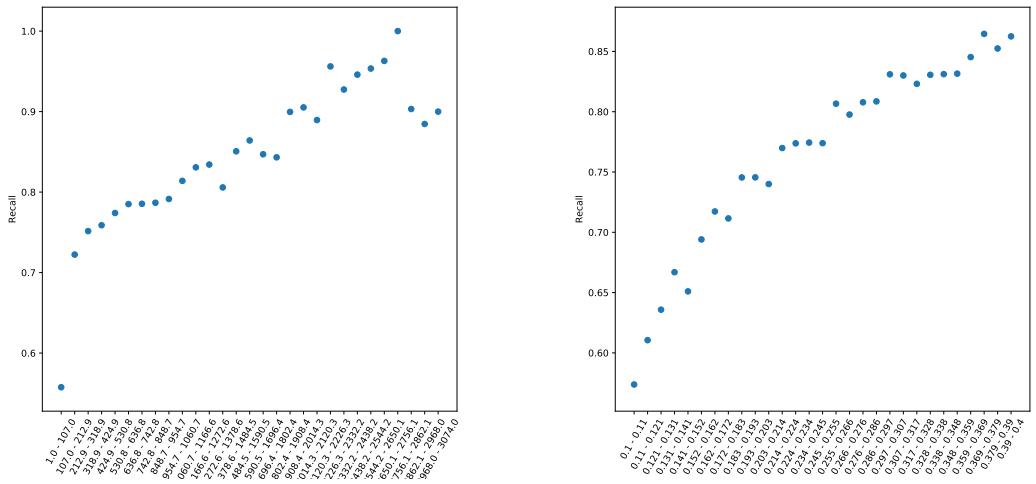
Chybně detekované (FP) na obrázku 5.15 jsou tvořeny opět často hraničními případy. Dále jsou zde obrázky, které mají binární anotaci na příslušném místě, avšak byly označeny jako chybně detekované. Jedná se o případy, kdy příslušná detekce a anotace neměla dostatečné IoU a detekce tedy byla označena za nesprávnou. U dlouhých vlasových defektů se zde nastává situace, kdy byl po své délce detekován vícekrát. Jedna detekce byla tedy označena za chybnou.

Správně detekované (TP) instance jsou na obrázku 5.13. Zde se jedná o velké zřetelné a jasně viditelné defekty. Pozitivní je, že model prokázal schopnost generalizace a detekoval i barevné a různě tvarované vady.

Diskuze nad výsledky modelů. Podle precision recall křivek na obrázku 5.11 mají všechny modely problém dosáhnout recallu hodnoty 1. V případě reálných dat si tento fakt vysvětluji jejich obtížností, kdy i pro mě bylo při anotaci velice těžké rozhodnout, zda už se jedná o vadu nebo přirozenou strukturu kartičky. Hranice mezi vadnou a bezvadnou oblastí je velice široká a nezřetelná. Tento fakt potvrzují i vizualizace falešných detekcí 5.15 a nedetekovaných instancí 5.14.

V případě testování na syntetických datech pozoruji tento jev taktéž. I v tomto testu mají modely problém dosáhnout recallu 1.0. Vysvětlení opět nacházím v povaze hledaných vad a tedy způsobu jejich generování. Generování vad je kontrolováno pouze řadou parametrů a může docházet k situacím, kdy je vada o malé ploše generována ještě navíc s velkou průsvitností a je tedy vygenerována velice nezřetelně. V těchto případech by jejich detekce byla obtížná i pro lidského operátora. Tuto hypotézu potvrzuje i obrázek 5.12, na kterém je zřetelná korelace mezi intenzitou/průhledností vady a detekčním skóre modelu. Podobný vztah lze sledovat s měnící se plochou vady.

Dále jsem se zabýval hypotézou, zda má lokace hledané vady vliv na úspěšnost jejího nalezení. Cílem bylo zjistit, zda má model obtíže hledat vady v oblastech se složitější strukturou pozadí. Vytvořil jsem proto 2d histogramy lokací nedetekovaných objektů a falešných detekcí na obrázcích 5.16b respektive 5.16a. Histogramy tuto hypotézu nepotvrdily. Chybovost tedy nejeví jakoukoliv korelaci s polohou na obrázku a tedy s pozadím.



(a) Závislost mezi plochou synteticky generované vady a recallem.

(b) Závislost mezi průhledností synteticky generované vady a recallem.

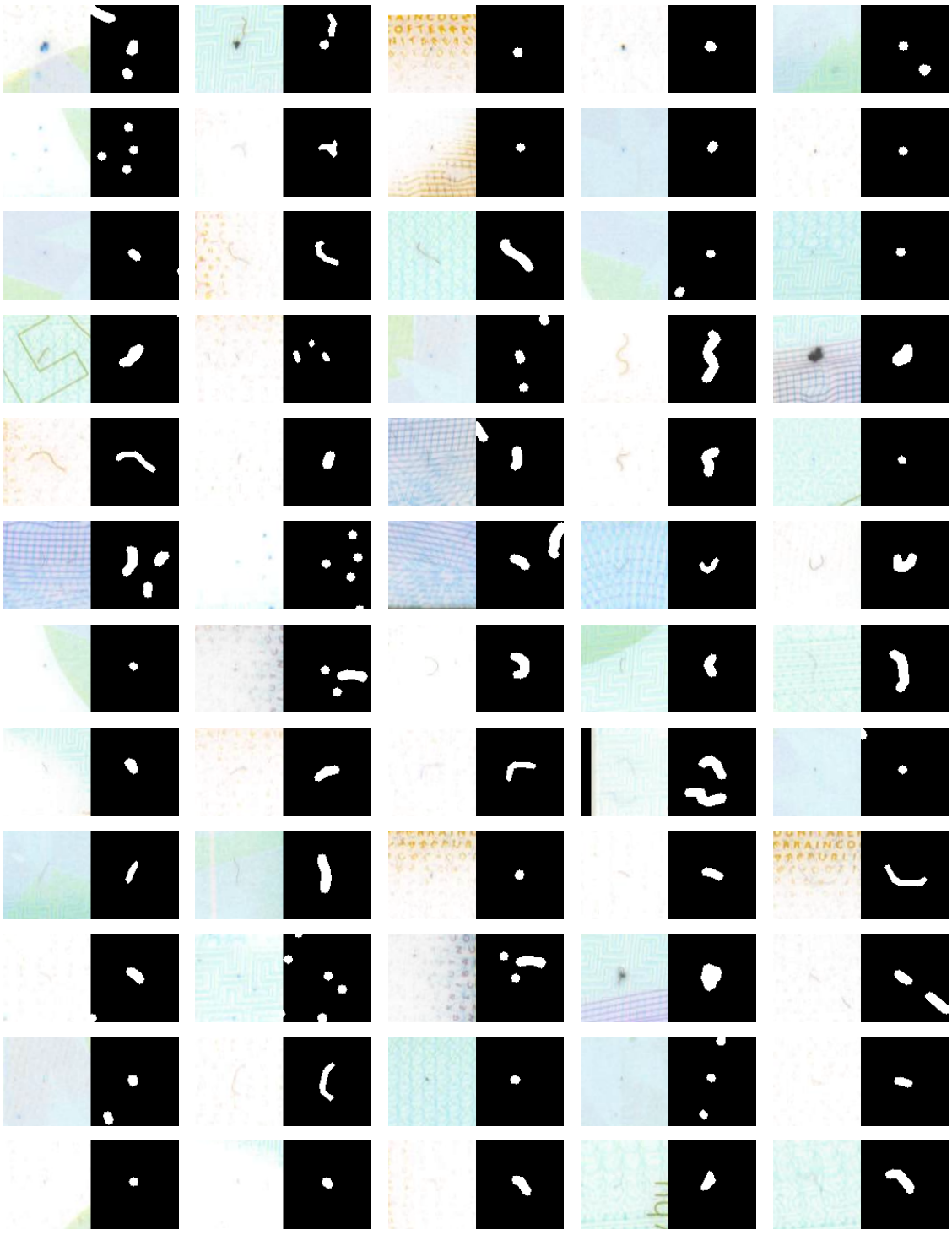
Obrázek 5.12: Na obrázku jsou zobrazeny korelace mezi parametry generování a výstupem detektoru. Parametry, které nejvíce ovlivňují recall, jsou plocha defektu a jeho průhlednost. V rámci tohoto experimentu jsem rozdělil syntetický dataset do 30 podmnožin podle plochy a následně podle průhlednosti. Pro každou tuto podmnožinu jsem vypočítal recall. Byl použit GRAY model na syntetickém testovacím datasetu. Úroveň skóre je nastavena tak, aby se precision rovnalo recallu podle hodnoty v tabulce 5.3.

Model	AP_1	AP_3	AP_5	AP_{75}	AP_9
RGB - syntetická data	0,85	0,72	0,54	0,30	0,28
RGB - reálná data	0,75	0,47	0,21	0,11	0,11
GRAY - syntetická data	**0,92**	**0,84**	**0,71**	**0,51**	**0,49**
GRAY - reálná data	**0,83**	**0,62**	**0,44**	**0,39**	**0,39**
RGB backbone - syntetická data	0,87	0,78	0,62	0,33	0,28
RGB backbone - reálná data	0,70	0,48	0,23	0,11	0,11

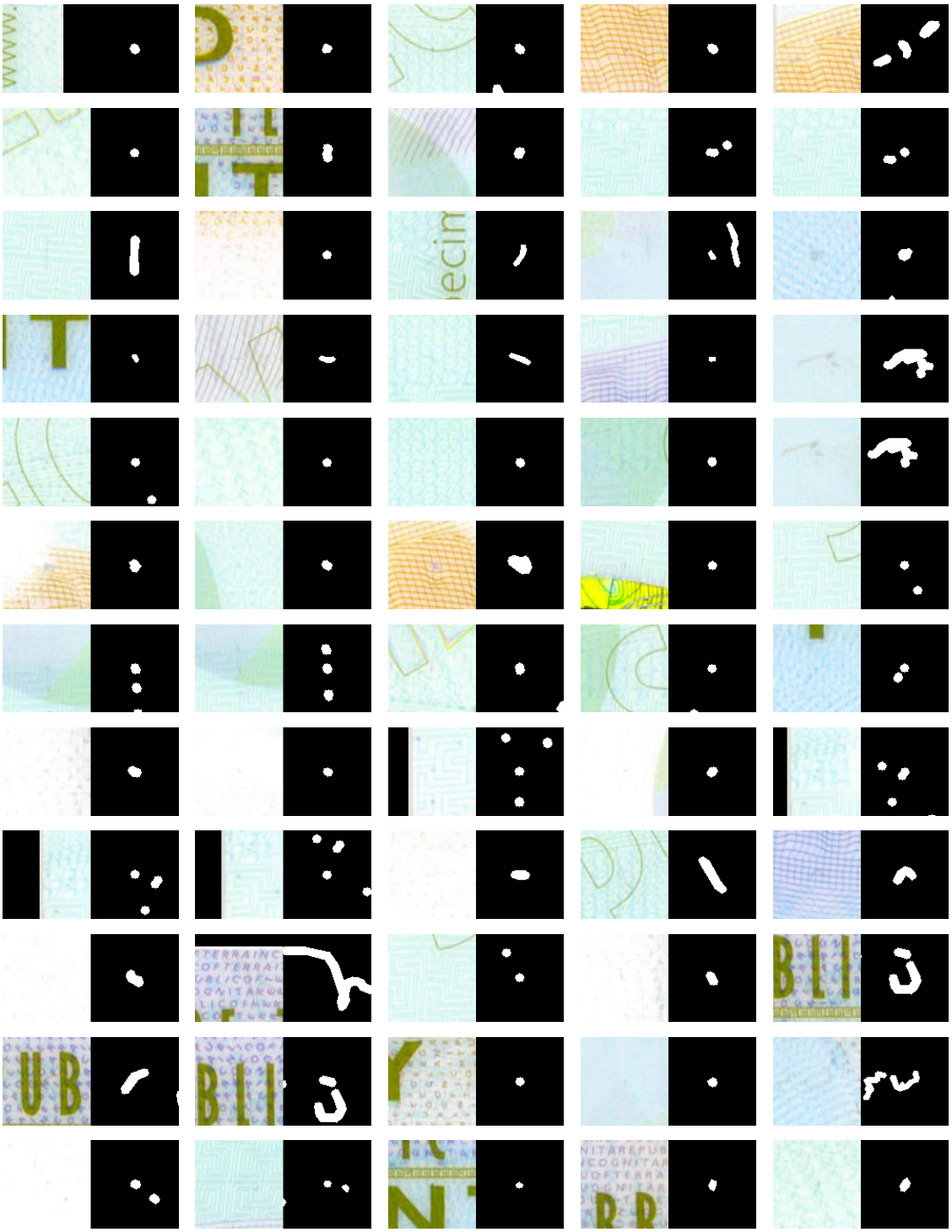
Tabulka 5.2: Tabulka s hodnotami AP pro různé úrovně IoU.

Model	Precision	Recall	Skóre
RGB, syntetická data	0,788	0,788	0,5918
RGB, reálná data	0,718	0,718	0,8040
GRAY, syntetická data	**0,777**	**0,777**	0,4965
GRAY, reálná data	**0,752**	**0,752**	0,7589
RGB backbone, syntetická data	0,807	0,807	0,8465
RGB backbone, reálná data	0,669	0,669	0,9299

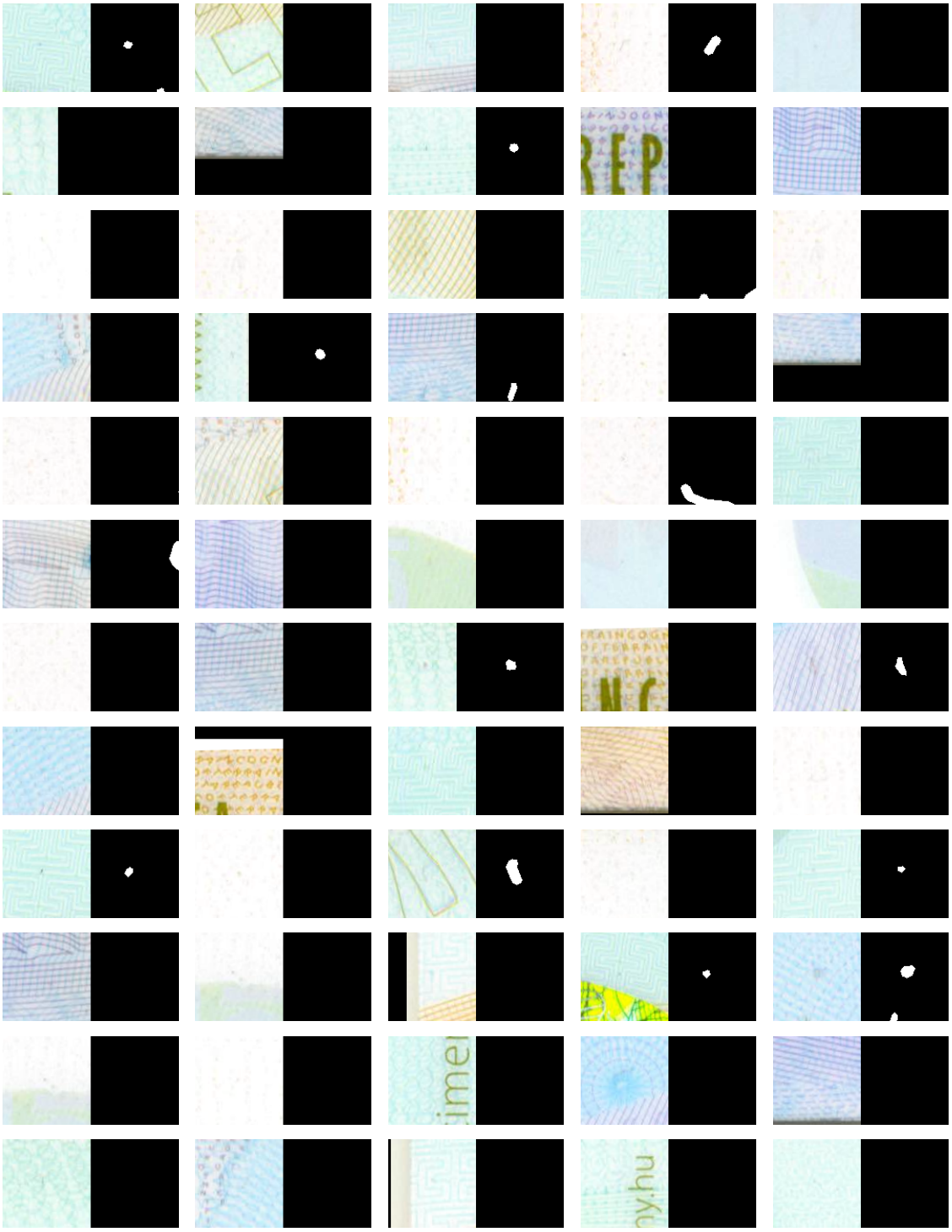
Tabulka 5.3: Významné body na precision recall křivce, kde precision je roven recallu.



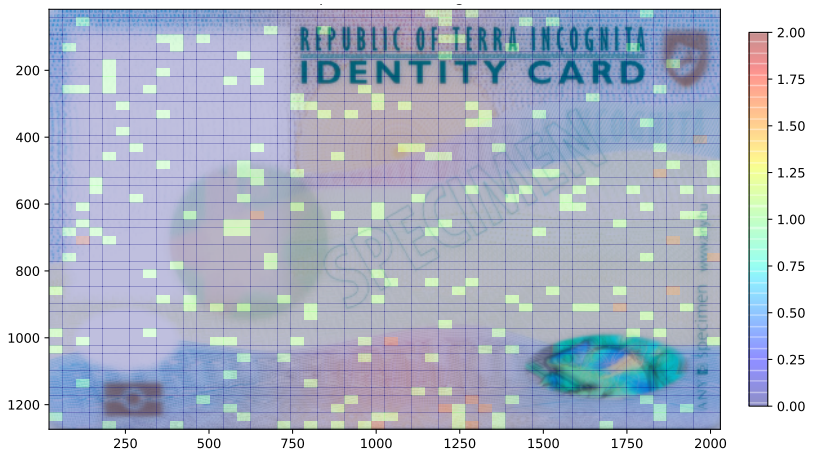
Obrázek 5.13: Vizualizace TP příkladů, tedy správných detekcí. Detekce jsou seřazeny a zobrazeny od nejvyššího skóre. Jedná se o GRAY model na reálných datech. Detekční hranice skóre je nastavena na 0,7589, aby byl recall roven precision. Hodnota IoU při pořízení vizualizace je 0,1. Na obrázku je 60 výřezů centrovaných na vadu zájmu společně s korespondující binární maskou.



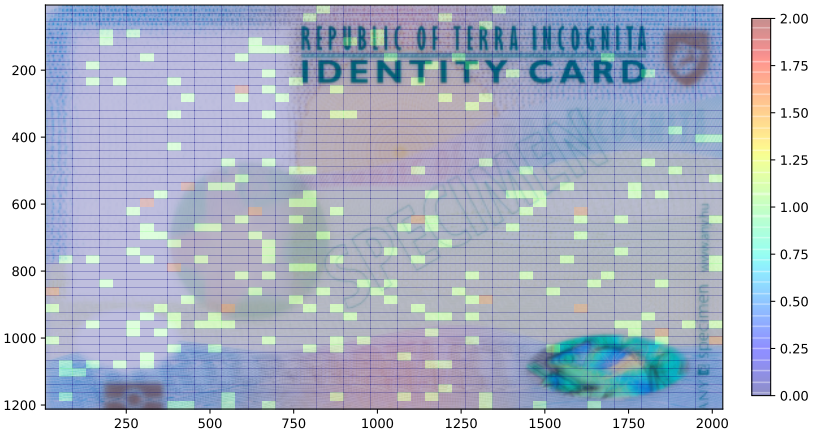
Obrázek 5.14: Vizualizace FN příkladů, tedy nedetekovaných objektů. Jedná se o GRAY model na reálných datech. Detekční hranice skóre je nastavena na 0,7589, tedy aby byl recall roven precision. Hodnota IoU při pořízení vizualizace je 0,1. Na obrázku je 60 výřezů centrovaných na objekt zájmu společně s korespondující binární maskou vpravo. Ve většině případů se jedná o velmi nezřetelné. Velké část těchto anotovaných by mohla být označena za hraniční příklady. Není úplně zřetelné, zda se jedná o vadu či ne.



Obrázek 5.15: Vizualizace nesprávných detekcí seřazených od nejvyššího skóre. Jedná se o GRAY model na reálných datech. Detekční hranice skóre je nastavena na 0,7589, tedy aby byl recall roven precision. Hodnota IoU při pořízení vizualizace je 0,1. Na obrázku je 60 výřezů centrovaných na vadu zájmu společně s korespondující binární maskou vpravo. V tomto případě se často jedná o vady, které již detekovány byly, a všechny další detekce jsou označeny za chybné. Dále jsou zde případy, kdy je detekce přítomna, ale není k ní s dostatečným IoU přiřazena příslušná GT instance. Detekce je tedy označena za přebývající a tedy nesprávnou. Potom se zde nachází spousta hraničních případů, které by mohly být zváženy na anotaci.



(a) Histogram lokací FP.



(b) Histogram lokací FN.

Obrázek 5.16: Na obrázku jsou 2d histogramy míst, kde algoritmus chyboval. V rámci této vizualizace jsem se snažil odhalit případnou korelaci pozadí kartičky a chybovosti detektoru. Hypotéza se touhle vizualizaci nepotvrdila, tedy korelace zde není přítomna. Jedná se o GRAY model na reálných datech. Detekční hranice skóre je nastavena na 0,7589, aby byl recall roven precision. Hodnota IoU při pořízení vizualizace je 0,1.

Obrázek 5.17: Výsledné detekce v kontextu obrázku celé kartičky. GT instance jsou zobrazeny zeleně, detekce červeně. Nad každou detekcí se nachází dvě čísla značící skóre detektoru a IoU s případnou korespondující GT instancí.

5.8 Další postup

V předchozí sekci jsem ukázal, jak pomocí vymodelování hledaných objektů, jejich náhodné augmentace a následného vkládání do původních obrázků řešit detekční úlohu v prostředí vizuální kontroly defektů. Tvorbou takovéhoto syntetického datasetu jsem vyřešil nedostatečné množství dat. Navrhnutý způsob tvorby datasetu však disponuje řadou hendikepů, které budu adresovat v této kapitole a navrhnu možné směrování práce k jejich vyřešení.

Při modelování defektů a vytváření jejich syntetických reprezentací jsem vycházel z předpokladu, že tímto úkonem plně pokryji jejich reálnou distribuci. Při pohledu na precision recall křivky na obrázku 5.11 je však patrné, že modely testované na syntetickém datasetu dosahují menší chyby než modely testované na reálných datech. Z toho vyvozuji, že syntetická data plně nereprezentovala distribuci reálných vad.

Dalším z hendikepů tohoto způsobu tvorby syntetického datasetu je omezená možnost jeho aplikace na jiných doménách dat. V tomto případě jsem pracoval na úloze, kde bylo možné jednotlivé defekty vytvořit pomocí relativně základních operací v oboru zpracování obrazu. Tyto vědomosti a zkušenosti bych však rád aplikoval i na jiné úlohy z jiných domén dat. Lze však s jistotou říci, že jedna z dalších úloh bude obsahovat mnohem rozmanitější a složitější defekty (například praskliny), které mnou použitou metodou nebude jednoduché imitovat.

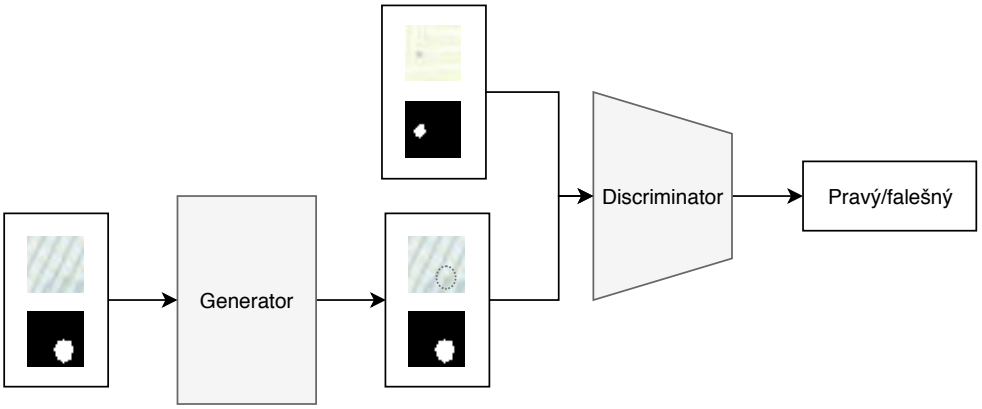
V nedávných letech získaly velkou pozornost architektury neuronových sítí typu GAN. Tyto modely se také zasloužily pokroku v oblasti augmentace a rozšiřování datasetů, kdy se například dokázaly naučit transformaci syntetických obrázků tak, aby více odpovídaly reálné distribuci [57]. Já taktéž GANy vnímám jako slibnou cestu k řešení těchto problémů. Za pomoci GANu by mohlo být možné lépe reprezentovat reálnou distribuci vad a takovéto řešení by mohlo být jednodušeji aplikovatelné napříč různými doménami dat.

Experiment s GANy. V rámci návrhu pokračování tohoto projektu jsem provedl experiment a pokusil jsem se distribuci vad namodelovat pomocí architektury GAN. V experimentu jsem se inspiroval přístupem autorů A. Shrivastava a spol. [57], kteří aplikovali architekturu GAN tak, že synteticky generovaný obrázek transformovali do podoby reálných vzorků. Já jsem použil veřejně dostupnou neoficiální implementaci[5], kterou jsem uzpůsobil pro svá data.

V rámci experimentu jsem postupoval následovně. Na vstup generátoru jsem vložil obrázek o velikosti 32^2, který neobsahoval žádný defekt. Jako čtvrtý kanál tohoto vstupu jsem přidal binární masku s anotací vady, kde bych ve výstupu generátoru vadu očekával. Na vstup diskriminátoru jsem jako reálný příklad vkládal obrázek obsahují vadu a korespondující binární masku ve čtvrtém kanálu obrázku. Výsledná architektura je na obrázku 5.18. Podobně jako v publikovaném přístupu, kde autoři porovnávali rozdíly mezi vstupem a výstupem generátoru a přílišné změny trestali, i já jsem výstup generátoru regularizoval. V mém případě jsem trestal jakékoliv změny čtvrtého kanálu (masky) a oblastí s nulovou hodnotou této masky. Generátoru jsem tedy povolil manipulovat pouze v oblasti zájmu, která byla vyznačena binární maskou.

S popsanou experimentální architekturou jsem však nedosáhl uspokojujících výsledků a další pokusy směrem k realizaci této myšlenky budou uskutečněny mimo rozsah této práce. Výsledné výstupy generátoru jsou na obrázku 5.19.

[5] SimGAN. 2019. URL: https://github.com/mjdietzx/SimGAN



Obrázek 5.18: Experimentální architektura GANu pro učení distribuce reálných defektů a následnou automatickou augmentaci datasetu. Myšlenka tohoto experimentu byla naučit generátor vkládat defekty do místa určeného binární maskou.

Další experimenty pro realizaci myšlenky generování vad bych navrhoval směřovat následovně. Mé studium se ubíralo směrem k publikaci Pix2Pix od P. Isola [25], který ukázal způsob, jakým zle využít architektura typu CGAN (Conditional Generative Adversarial Network). Algoritmus například dokázal transformovat mapové poklady do různých domén (letecké snímky, vektorové mapy) nebo renderovat obrázky z jednoduchých kreseb. Navazující přístupy přinesly ještě kvalitnější a fotorealističtější výsledky [8].

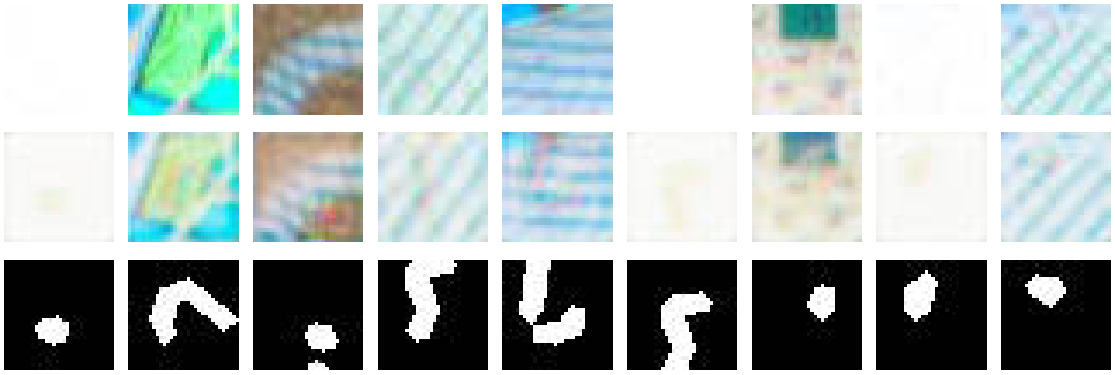
Vedle Pix2Pix se objevila řada podobných aplikací na principu sémantické manipulace obrazu podmíněné binární maskou. Zaujala mě práce autorů T. Park a spol. [43], ve které dokázali použitím GANů syntetizovat ze sémantické segmentační masky fotorealistický obraz. Atmosféra výsledného obrázku je v tomto případě navíc parametrizovatelná dalším vstupním obrázkem.

Další velice zajímavou aplikací je sémantická manipulace obrazu pouze v regionu zájmu. Autoři z Google Brain S. Hong a spol. [22] publikovali způsob, jakým do vstupního obrázku na základě sémantické masky zakomponovat objekt požadované třídy. Základní struktura algoritmu je přiblížena na obrázku 5.20. Tato myšlenka by se dala efektivně využít pro generování vad do obrázku a snad tak zkvalitnit výsledky detektoru.

Zapojení doménového experta jakožto učitele. Z vizualizací výsledků na obrázcích 5.14 a 5.15 je patrné, že rozhodovací hranice mezi tím, zda se jedná o defekt nebo ne, je velice široká a často nejasná. I pro mě, jakožto autora anotací datasetu, bylo mnohdy těžké určit, zda se již jedná o vadu. Vyvíjený algoritmus bude navíc sloužit uživateli, který by taktéž jistě ocenil větší kontrolu nad jeho výstupy. Jako expert nad danou doménou dat by tak na rozdíl od vývojáře mohl rozhodovací hranici určit mnohem přesněji.

Přínosným vylepšením by tedy bylo umožnit uživateli zapojení se a poskytnout mu možnost algoritmus kontrolovat. Učení by tak mohlo probíhat iterativně a postupně se zlepšovat směrem k potřebám zákazníka.

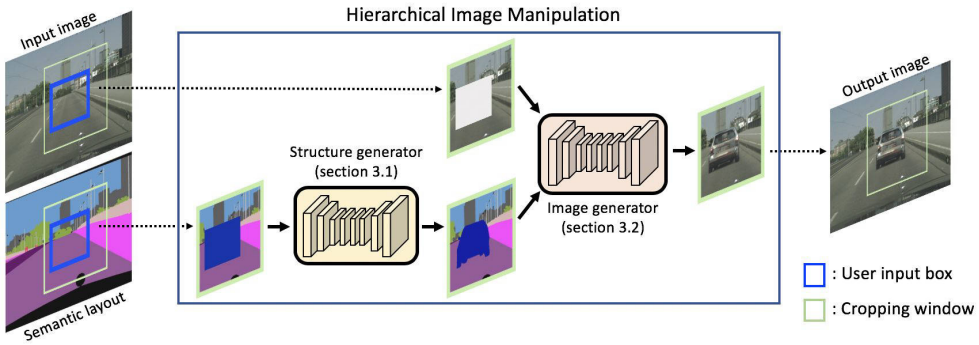
Problém široké a nejasné rozhodovací hranice jsem se pokusil se zákazníkem řešit. Dohodli jsme se na dodání testovací sady anotované doménovým expertem, což by mělo zajistit dostatečnou kvalitu. Anotace však nedosahovaly dostatečné kvality a konsistence. Algoritmus nakonec na dodané datové sadě dokázal najít přibližně dvojnásobné množství defektů,



Obrázek 5.19: Výsledky prvního experimentu použití GANu pro generování defektů nebyly přesvědčivé. Na obrázku jsou v první řadě vstupy, v druhé řadě jsou výstupy generátoru. V třetím řádku jsou použitě binární masky pro každý obrázek. Ve výstupech je možné vidět určité náznaky směrem k tvoření defektu v určeném místě. Avšak také celý obrázek je určitým způsobem modifikován, což je pro mou aplikaci nežádoucí.

než bylo anotováno. Tyto defekty byly vizuálně prohlédnuty a jednalo se o jasné a zřetelné defekty.

Efektivnější trénink. Ačkoliv finální trénink probíhal řádově po dobu 6 dnů. Z obrázků průběhu chybové funkce 5.10 je patrné, že měl stále mírně klesající tendenci a delší běh by mohl dále zlepšit výsledek. Vybraná implementace firmy Matterport navíc při použití více GPU neškálovala a nebyla dostatečně efektivní. Delší dobu jsem však neměl výpočetní stroj k dispozici. Řešením by mohlo být použití optimálnější implementace Mask R-CNN.



Obrázek 5.20: Obrázek znázorňuje architekturu sítě, která na základě binární segmentace dokáže sémanticky zakomponovat objekt požadované třídy do obrázku. Tímto směrem bych navrhoval směrovat pokračování této práce a získal tak pokročilejší augmentaci datasetu, která by navíc mohla být aplikovatelnější napříč různými doménami dat. Převzato z [22].

Kapitola 6

Závěr

Ve své práci jsem navrhl a implementoval systém pro detekci defektů vznikajících při výrobním procesu plastikových karet. Realizace takovéhoto úkolu byla výzvou především kvůli nedostatku trénovacích dat, relativní velikosti a viditelnosti hledaných defektů a rozmanitosti pozadí kartičky.

Pro realizaci úkolu jsem nastudoval techniky a řešení používané v oblasti vizuální kontroly defektů při průmyslových aplikacích. Úlohu jsem se následně rozhodl pojmout jako supervizovaný detekční problém realizovaný aktuálním state-of-the-art algoritmem Mask R-CNN. První iterace tréninku poukázala na nedostatečné množství trénovacích dat, kdy natrénovaný model nedokázal generalizovat na testovací sadě. Tento problém jsem vyřešil namodelováním hledaných defektů a vytvořením syntetického trénovacího datasetu. Model natrénovaný na takovýchto datech následně generalizoval i na původním datasetu s reálnými defekty. Model dosáhl 0,752 precision při 0,752 recallu. Ačkoliv se tato hodnota nezdá být vysoká, je podle mého názoru obhajitelná povahou a obtížností dat. Při pohledu na vizualizace chybných a chybějících detekcí je patrné, jak nejasná a široká rozhodovací hranice v tomto úkolu je. Na těchto příkladech je i pro člověka obtížné rozhodnout, zda se ještě jedná o defekt či ne.

Tento problém vede k dalšímu možnému směřování práce a tím by mohlo být zapojení doménového experta a kontrolovaně směřovat chování algoritmu pomocí aktivního učení.

Výsledky práce byly prezentovány zákazníkovi, který byl pozitivně překvapen dosaženými výsledky a možnostmi dnešních metod zpracování obrazu a strojového učení. Se zákazníkem byla navázána další spolupráce s cílem algoritmus optimalizovat a uvést systém do reálné produkce.

Literatura

[1] Abdi, H.; Williams, L. J.: Principal Component Analysis. *WIREs Comput. Stat.*, ročník 2, č. 4, Červenec 2010: s. 433–459, ISSN 1939-5108, doi:10.1002/wics.101.

[2] Abdulla, W.: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN, 2017.

[3] Abraham, B.; Chuang, A.: Outlier Detection and Time Series Modeling. *Technometrics*, ročník 31, č. 2, Květen 1989: s. 241–248, ISSN 0040-1706, doi:10.2307/1268821.

[4] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; aj.: LOF: Identifying Density-based Local Outliers. *SIGMOD Rec.*, ročník 29, č. 2, Květen 2000: s. 93–104, ISSN 0163-5808, doi:10.1145/335191.335388.

[5] Brock, A.; Donahue, J.; Simonyan, K.: Large Scale GAN Training for High Fidelity Natural Image Synthesis. *CoRR*, ročník abs/1809.11096, 2018, 1809.11096.

[6] Byers, S.; Raftery, A. E.: Nearest-Neighbor Clutter Removal for Estimating Features in Spatial Point Processes. *Journal of the American Statistical Association*, ročník 93, č. 442, 1998: s. 577–584, ISSN 01621459.

[7] Chandola, V.; Banerjee, A.; Kumar, V.: Anomaly Detection: A Survey. *ACM Comput. Surv.*, ročník 41, č. 3, Červenec 2009: s. 15:1–15:58, ISSN 0360-0300, doi:10.1145/1541880.1541882.

[8] Chen, Q.; Koltun, V.: Photographic Image Synthesis with Cascaded Refinement Networks. *CoRR*, ročník abs/1707.09405, 2017, 1707.09405.

[9] Cortes, C.; Vapnik, V.: Support-vector networks. *Machine Learning*, ročník 20, č. 3, Sep 1995: s. 273–297, ISSN 1573-0565, doi:10.1007/BF00994018.

[10] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, Washington, DC, USA: IEEE Computer Society, 2005, ISBN 0-7695-2372-2, s. 886–893, doi:10.1109/CVPR.2005.177.

[11] Deng, J.; Dong, W.; Socher, R.; aj.: ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[12] Denning, D. E.: An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, ročník SE-13, č. 2, Feb 1987: s. 222–232, ISSN 0098-5589, doi:10.1109/TSE.1987.232894.

[13] Everingham, M.; Eslami, S. M. A.; Van Gool, L.; aj.: The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, ročník 111, č. 1, Leden 2015: s. 98–136.

[14] Ferguson, M.; Ak, R.; Lee, Y. T.; aj.: Detection and Segmentation of Manufacturing Defects with Convolutional Neural Networks and Transfer Learning. *CoRR*, ročník abs/1808.02518, 2018, 1808.02518.

[15] Girshick, R. B.: Fast R-CNN. *CoRR*, ročník abs/1504.08083, 2015, 1504.08083.

[16] Girshick, R. B.; Donahue, J.; Darrell, T.; aj.: Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, ročník abs/1311.2524, 2013, 1311.2524.

[17] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[18] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; aj.: Generative adversarial nets. In *Advances in neural information processing systems*, 2014, s. 2672–2680.

[19] He, K.; Gkioxari, G.; Dollár, P.; aj.: Mask R-CNN. *CoRR*, ročník abs/1703.06870, 2017, 1703.06870.

[20] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. *CoRR*, ročník abs/1512.03385, 2015, 1512.03385.

[21] He, Z.; Xu, X.; Deng, S.: Discovering cluster-based local outliers. *Pattern Recognition Letters*, ročník 24, č. 9, 2003: s. 1641–1650.

[22] Hong, S.; Yan, X.; Huang, T.; aj.: Learning Hierarchical Semantic Image Manipulation through Structured Representations. *CoRR*, ročník abs/1808.07535, 2018, 1808.07535.

[23] Huang, Y.; Qiu, C.; Yuan, K.: Surface defect saliency of magnetic tile. *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018: s. 612–617.

[24] Ioffe, S.; Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, JMLR.org, 2015, s. 448–456.

[25] Isola, P.; Zhu, J.; Zhou, T.; aj.: Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*, ročník abs/1611.07004, 2016, 1611.07004.

[26] Jain, A. K.; Dubes, R. C.: *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988, ISBN 0-13-022278-X.

[27] Karras, T.; Aila, T.; Laine, S.; aj.: Progressive Growing of GANs for Improved Quality, Stability, and Variation. *CoRR*, ročník abs/1710.10196, 2017, 1710.10196.

[28] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. J. C. Burges; L. Bottou; K. Q. Weinberger, Curran Associates, Inc., 2012, s. 1097–1105.

[29] LeCun, Y.; Cortes, C.: MNIST handwritten digit database. 2010.
URL http://yann.lecun.com/exdb/mnist/

[30] Lin, T.; Dollár, P.; Girshick, R. B.; aj.: Feature Pyramid Networks for Object Detection. *CoRR*, ročník abs/1612.03144, 2016, 1612.03144.

[31] Lin, T.; Goyal, P.; Girshick, R. B.; aj.: Focal Loss for Dense Object Detection. *CoRR*, ročník abs/1708.02002, 2017, 1708.02002.

[32] Lin, T.; Maire, M.; Belongie, S. J.; aj.: Microsoft COCO: Common Objects in Context. *CoRR*, ročník abs/1405.0312, 2014, 1405.0312.

[33] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. *CoRR*, ročník abs/1512.02325, 2015, 1512.02325.

[34] Liu, Z.; Luo, P.; Wang, X.; aj.: Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[35] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, ročník 60, 2004: s. 91–110.

[36] Maas, A. L.; Hannun, A. Y.; Ng, A. Y.: Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, ročník 30, 2013, str. 3.

[37] Makhzani, A.; Frey, B.: K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

[38] Marino, S.; Beauseroy, P.; Smolarz, A.: Deep Learning-based Method for Classifying and Localizing Potato Blemishes. 01 2019, s. 107–117, doi:10.5220/0007350101070117.

[39] Mařík, V.: *Národní iniciativa Průmysl 4.0*. 2015.

[40] Mei, S.; Yang, H.; Yin, Z.: An Unsupervised-Learning-Based Approach for Automated Defect Inspection on Textured Surfaces. *IEEE Transactions on Instrumentation and Measurement*, ročník 67, č. 6, June 2018: s. 1266–1277, ISSN 0018-9456, doi:10.1109/TIM.2018.2795178.

[41] Mirza, M.; Osindero, S.: Conditional Generative Adversarial Nets. *CoRR*, ročník abs/1411.1784, 2014, 1411.1784.

[42] Nair, V.; Hinton, G. E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, USA: Omnipress, 2010, ISBN 978-1-60558-907-7, s. 807–814.

[43] Park, T.; Liu, M.; Wang, T.; aj.: Semantic Image Synthesis with Spatially-Adaptive Normalization. *CoRR*, ročník abs/1903.07291, 2019, 1903.07291.

[44] Patcha, A.; Park, J.-M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, ročník 51, č. 12, 2007: s. 3448 – 3470, ISSN 1389-1286, doi:https://doi.org/10.1016/j.comnet.2007.02.001.

[45] Radford, A.; Metz, L.; Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[46] Ratsch, G.; Mika, S.; Scholkopf, B.; aj.: Constructing boosting algorithms from SVMs: an application to one-class classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 24, č. 9, Sep 2002: s. 1184–1199, ISSN 0162-8828, doi:10.1109/TPAMI.2002.1033211.

[47] Redmon, J.; Divvala, S. K.; Girshick, R. B.; aj.: You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, ročník abs/1506.02640, 2015, 1506.02640.

[48] Redmon, J.; Farhadi, A.: YOLO9000: Better, Faster, Stronger. *CoRR*, ročník abs/1612.08242, 2016, 1612.08242.

[49] Redmon, J.; Farhadi, A.: YOLOv3: An Incremental Improvement. *CoRR*, ročník abs/1804.02767, 2018, 1804.02767.

[50] Ren, R.; Hung, T.; Tan, K. C.: A Generic Deep-Learning-Based Approach for Automated Surface Inspection. *IEEE Transactions on Cybernetics*, ročník 48, č. 3, March 2018: s. 929–940, ISSN 2168-2267, doi:10.1109/TCYB.2017.2668395.

[51] Ren, S.; He, K.; Girshick, R. B.; aj.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, ročník abs/1506.01497, 2015, 1506.01497.

[52] Rifai, S.; Vincent, P.; Muller, X.; aj.: Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Omnipress, 2011, s. 833–840.

[53] Ronneberger, O.; P.Fischer; Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, *LNCS*, ročník 9351, Springer, 2015, s. 234–241, (available on arXiv:1505.04597 [cs.CV]).

[54] Rublee, E.; Rabaud, V.; Konolige, K.; aj.: ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, Washington, DC, USA: IEEE Computer Society, 2011, ISBN 978-1-4577-1101-5, s. 2564–2571, doi:10.1109/ICCV.2011.6126544.

[55] Salimans, T.; Goodfellow, I. J.; Zaremba, W.; aj.: Improved Techniques for Training GANs. *CoRR*, ročník abs/1606.03498, 2016, 1606.03498.

[56] Schlegl, T.; Seeböck, P.; Waldstein, S. M.; aj.: Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. *CoRR*, ročník abs/1703.05921, 2017, 1703.05921.

[57] Shrivastava, A.; Pfister, T.; Tuzel, O.; aj.: Learning from Simulated and Unsupervised Images through Adversarial Training. *CoRR*, ročník abs/1612.07828, 2016, 1612.07828.

[58] Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[59] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, ročník 15, 2014: s. 1929–1958.

[60] Szegedy, C.; Liu, W.; Jia, Y.; aj.: Going Deeper with Convolutions. *CoRR*, ročník abs/1409.4842, 2014, 1409.4842.

[61] Uijlings, J.; van de Sande, K.; Gevers, T.; aj.: Selective Search for Object Recognition. *International Journal of Computer Vision*, 2013, doi:10.1007/s11263-013-0620-5.

[62] Vincent, P.; Larochelle, H.; Bengio, Y.; aj.: Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, New York, NY, USA: ACM, 2008, ISBN 978-1-60558-205-4, s. 1096–1103, doi:10.1145/1390156.1390294.

[63] Viola, P.; Jones, M.: Rapid object detection using a boosted cascade of simple features. 2001.

[64] Wang, T.; Liu, M.; Zhu, J.; aj.: High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *CoRR*, ročník abs/1711.11585, 2017, 1711.11585.

[65] Xie, S.; Girshick, R. B.; Dollár, P.; aj.: Aggregated Residual Transformations for Deep Neural Networks. *CoRR*, ročník abs/1611.05431, 2016, 1611.05431.

[66] Yu, F.; Zhang, Y.; Song, S.; aj.: LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *arXiv preprint arXiv:1506.03365*, 2015.

[67] Zanin, M.; Romance, M.; Moral, S.; aj.: Credit card fraud detection through parenclitic network analysis. *CoRR*, ročník abs/1706.01953, 2017, 1706.01953.

[68] Zhai, W.; Zhu, J.; Cao, Y.; aj.: A Generative Adversarial Network Based Framework for Unsupervised Visual Surface Inspection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, ISSN 2379-190X, s. 1283–1287, doi:10.1109/ICASSP.2018.8462364.

[69] Zhang, H.; Xu, T.; Li, H.; aj.: StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *CoRR*, ročník abs/1612.03242, 2016, 1612.03242.