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

VYUŽITÍ SÍTÍ TYPU GAN PRO ZPŘESŇOVÁNÍ DETEKCE A ROZPOZNÁVÁNÍ DOPRAVNÍCH ZNAČEK

IMPROVING ACCURACY OF DETECTION AND CLASSIFICATION OF TRAFFIC SIGNS WITH GANS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL GLOS**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Glos Michal**
Program: Informační technologie
Název: **Využití sítí typu GAN pro zpřesňování detekce a rozpoznávání dopravních značek**
Improving Accuracy of Detection and Recognition of Traffic Signs with GANs
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se se způsoby rozpoznávání objektů a s dostupnými implementacemi pro snadnou konfiguraci a trénování neuronových sítí.
2. Nastudujte problematiku použití sítí typu GAN pro rozšiřování datových sad a zvýšení robustnosti detekce a rozpoznávání.
3. Na základě získaných poznatků navrhněte a implementujte systém, který bude s využitím GAN zlepšovat detekci a rozpoznávání dopravních značek z reálných snímků.
4. Vyhodnoťte výsledky systému na reprezentativním vzorku dat.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 7. května 2021

Abstrakt

Cílem této práce bylo rozšíření datové sady pro detekci dopravních značek. K řešení byly použity generativní neuronové sítě typu PatchGAN a Wasserstein GAN, kombinující architektury DenseNet a U-Net. Modely byly navrženy k syntéze snímků reálně vypadajících dopravních značek z obrázků jejich norem. Model pro detekci objektů typu SSD, natrénován pouze na syntetických datech, dosáhl přesnosti 59.6 % mAP, což je o 9.4 % lepší výsledek oproti referenčnímu modelu, natrénovaném pouze na původních datech. V případě natrénování modelu na kombinaci syntetických a původních dat, dosáhl SSD model přesnosti až 80.1 % mAP.

Abstract

The goal of this thesis was to extend a dataset for traffic sign detection. The solution was based on generative neural networks PatchGAN and Wasserstein GAN of combined DenseNet and U-Net architecture. Those models were designed to synthesize real looking traffic signs from images of their norms. Model for object detection SSD, trained on synthetic data only, achieved mean average precision of 59.6 %, which is an improvement of 9.4 % over the model trained on the original data. SSD model trained on synthetic and original data combined achieved mean average precision of 80.1 %.

Klíčová slova

SSD, GAN, detekce dopravních značek, generativní model, Pix2Pix, U-Net, Wasserstein GAN, PatchGAN.

Keywords

SSD, GAN, traffic sign detection, generative model, Pix2Pix, U-Net, Wasserstein GAN, PatchGAN.

Citace

GLOS, Michal. *Využití sítí typu GAN pro zpřesňování detekce a rozpoznávání dopravních značek*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Využití sítí typu GAN pro zpřesňování detekce a rozpoznávání dopravních značek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Glos
11. května 2021

Poděkování

Rád bych poděkoval mému vedoucímu práce doc. RNDr. Pavlovi Smržovi Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Obsah

Kapitola 1

Úvod

Metody strojového vidění a umělé inteligence jsou nejen v posledních letech velice intenzivně studovány. Nově objevené algoritmy pro rozpoznávání objektů, společně s rostoucími výpočetními možnostmi moderních počítačů, nalezly uplatnění v nejrůznějších oblastech průmyslu i služeb. Jejich základem často bývají neuronové sítě, které však mnohdy potřebují velké množství dostatečně kvalitních dat na to, aby fungovaly spolehlivě. Sbírání dat a jejich anotace je ale finančně i časově velice nákladná činnost.

Cílem této práce je navrhnout a implementovat systém zahrnující generativní neuronové sítě, který by byl schopen automaticky rozšířit datovou sadu, určenou k trénování modelů pro detekci objektů. Použitím takového systému by bylo možné natrénovat přesný detektor i s použitím menší datové sady, čímž by mohly být částečně eliminovány výdaje na její tvorbu.

V této práci se soustředím především na implementaci a konfiguraci modelu pro syntézu anotovaných snímků dopravních značek z jejich norem definovaných legislativou. Postupně jsem ladil konfiguraci generátoru na základě znalostí nabytých experimentováním v kapitole 6. Nakonec jsem dva generativní modely, produkující vizuálně nejkvalitnější snímky, využil k rozšíření neveřejné datové sady pro detekci dopravních značek ze Slovenských silnic. Detektor, natrénovaný na kombinaci reálných a syntetický dat, dosáhl přesnosti 80.1 % mAP a po natrénování pouze na syntetických datech 59.6 % mAP. Oba výsledky vykazují zlepšení, oproti detektoru, natrénován pouze na originálních datech, dosahující přesnosti 50.2 % mAP.

V kapitole 2 nejprve popisuji moderní architektury neuronových sítí pro rozpoznávání objektů a možnosti kvantifikace jejich přesnosti. V další kapitole 3 se věnuji architekturám generativních modelů GAN a jejich použití k transformaci snímků. V kapitole 5 popisuji použité modely pro detekci i generování dat a postupy tvoření konkrétních trénovacích datových sad pro oba modely. Dále jsou v této kapitole shrnuty možnosti konfigurace použitých modelů. V poslední kapitole 6 je experimentováno s různými architekturami a konfiguracemi generativního modelu za účelem nalezení kombinace generující nejkvalitnější syntetická data.

Kapitola 2

Rozpoznávání objektů pomocí konvolučních neuronových sítí

Pojem *rozpoznávání objektů* může referovat buď na *klasifikaci* - úkol určit třídu objektů nacházejících se na snímku, nebo na *detekci* - úkol správně lokalizovat objekty ve snímku, včetně jejich klasifikace [31]. Většina metod rozpoznávání objektů funguje ve dvou krocích. Nejprve jsou, například pomocí konvolučních neuronových sítí, extrahovány charakteristické příznaky snímku, které jsou později klasifikovány do tříd, například za použití *Support Vector Machine (SVM)*. V posledních letech bylo experimentováno s mnoha různými přístupy k problému rozpoznávání objektů [25]. Pro extrakci příznaků ze snímků se používaly především metody založeny na *Haar [20], HOG (Histogram of Oriented Gradient) [5]* nebo *SIFT (Scale Invariant Feature Transform) [23]*. V roce 2012 však na soutěži *ImageNet Large Scale Visual Recognition Challenge* [31] v kategorii klasifikace objektů výrazně zvítězila konvoluční neuronová sít *AlexNet* [18] *viz. tabulka* 2.1. Tím vzrostla popularita konvolučních neuronových sítí pro extrakci příznaků natolik, že většina týmů v nadcházejících ročnících soutěže z této architektury vycházela [32].

Model	**Top-1**	**Top-5**
Sparse coding [31]	*47.1 %*	*28.2 %*
SIFT + FVs [33]	*45.7 %*	*25.7 %*
CNN	**37.5 %**	**17.0 %**

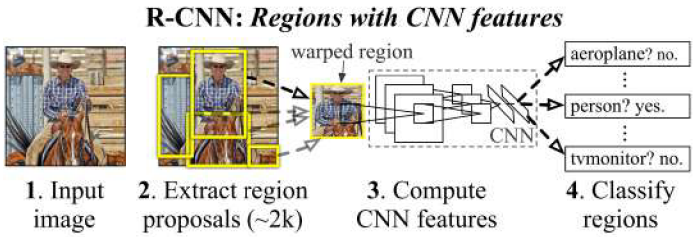
Tabulka 2.1: Porovnání nejlepších výsledků dosažených na testovací datové sadě ILSRVC-2010 k roku 2012 [18, 31]. Sloupec **Top-1** udává procento případů, kdy model nepředpověděl správnou třídu objektu. **Top-5** udává procento případů, kdy se správná třída klasifikovaného objektu nevyskytuje mezi 5 nejpravděpodobnějším odhady modelu [18].

2.1 Moderní architektury konvolučních neuronových sítí určeny k rozpoznávání objektů

Kvůli tak významnému pokroku v přesnosti klasifikace za použití konvolučních neuronových sítí, se výzkumníci snažili tento princip generalizovat i na úkol detekce. Vzniklo několik architektur, které svým výkonem daleko předčily ostatní, v té době *state of the art*, modely. Jednou z prvních úspěšných architektur byla R-CNN [10], ze které byly později odvozeny další úspěšné modely jako *Fast R-CNN* [9] a *Faster R-CNN* [29]. Dalšími velice úspěšnými metodami jsou SSD *(Single Shot Detector)* [22], nebo YOLO *(You Only Look Once)* [28].

R-CNN

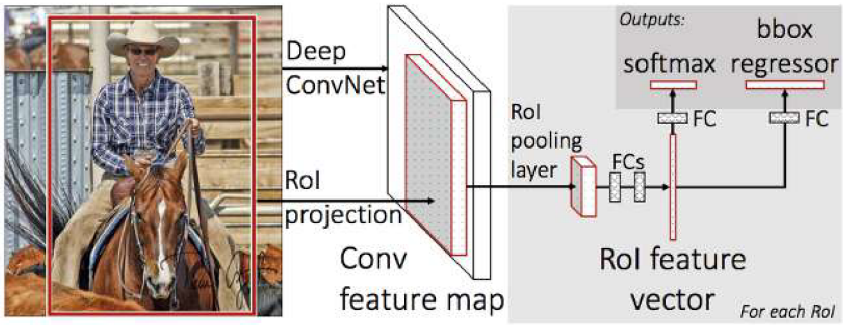
Region-Based Convolutional Neural Network neboli *Regionální konvoluční neuronová síť*, zkráceně *R-CNN*, je jedna z prvních metod úspěšně aplikujících konvoluční neuronové sítě na problém detekce objektů [10]. Model se skládá celkem ze tří komponentů, *viz obrázek 2.1*. Prvním z nich je algoritmus navrhující požadovaný počet regionů s pravděpodobným výskytem objektu. V původní práci byl použit algoritmus *selective search* [36]. Dalším je extraktor příznaků v podobě konvoluční neuronové sítě, který je postupně aplikován na všechny navrhované regiony. Posledním je klasifikátor, který extrahované příznaky všech navržených regionů klasifikuje do tříd. Nevýhodou tohoto modelu je, že každý z navržených regionů musí být samostatně klasifikován. Původní práce [10] navrhuje 2000 regionů během validace modelu, což prakticky znemožňuje použití R-CNN v reálném čase.



Obrázek 2.1: Grafické zobrazení všech kroků procesu detekce objektů modelem *R-CNN*. *Převzato z [10]*

Fast R-CNN

Po velkém úspěchu *R-CNN* byl v publikaci stejného autora v roce 2015, navržen model *Fast R-CNN* [9], který adresuje především problémy s rychlostí, avšak dosahuje i vyšší přesnosti. Oproti *R-CNN* je proces trénování 9 krát a proces vyhodnocování až 213 krát rychlejší [9]. Jeho hlavní výhodou je skutečnost, že je navržen jako jeden monolitní model, *viz obrázek 2.2*, na rozdíl od *R-CNN*, složené ze tří samostatných komponentů. Místo zpracování každého regionu konvolučním extraktorem příznaků zvlášť, Fast R-CNN vypočítá příznaky pro celý vstupní obrázek najednou, čímž utvoří mapu příznaků. Z této mapy se následně selektují navržené regiony, které jsou transformovány do vektorů konstantní délky, z nichž jsou následně aplikací plně propojených vrstev a aktivační vrstvy získány informace o třídě objektu a o pozici jeho *bounding boxu*.



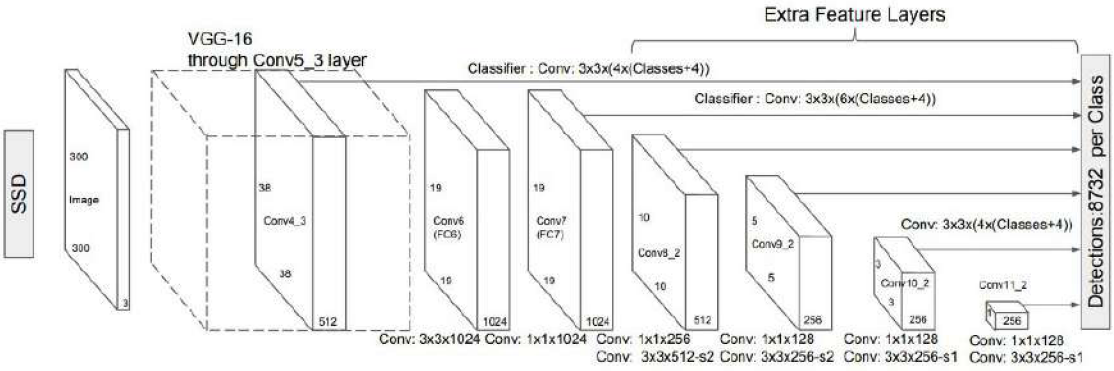
Obrázek 2.2: Schéma modelu *Fast R-CNN*. Na vstupu je obrázek, na který je aplikovaný extraktor příznaků v podobě hluboké konvoluční neuronové sítě. Příznaky uvnitř navržených regionů různých velikostí jsou pomocí vrstvy *RoI pooling* transformovány do vektorů stejných velikostí. Ty jsou pak zpracovány ve dvou větvích. V první větvi je na vektor aplikováno několik plně propojených vrstev a aktivační funkce *softmax*, čímž se určí třída objektu. V druhé větvi je na vektor také aplikováno několik plně propojených vrstev, ty však vedou do *bounding box regresoru*, čímž se upřesní poloha bounding boxu detekovaného objektu. *Převzato z [9]*

Faster R-CNN

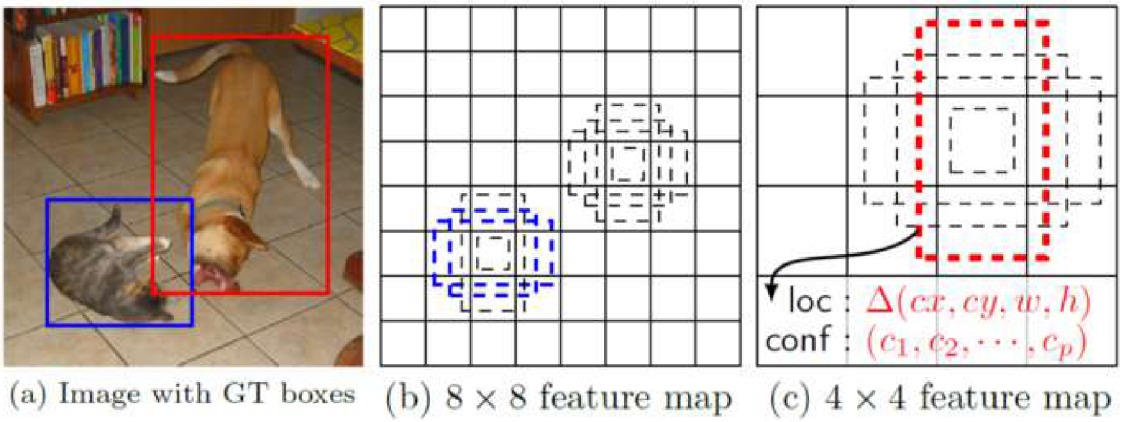
Metoda navrhování regionů pomocí algoritmu *Selective Search* je velice pomalá a jeho implementace na CPU generuje regiony pro každý obrázek 2 sekundy [29], což odhalilo úzké místo předchozích modelů. Proto metoda *Faster R-CNN* používá pro navrhování regionů samostatnou neuronovou síť *RPN (Region Proposal Network)*, dosahující řádově větší rychlosti oproti algoritmu *Selective Search*, a je považována za první metodu schopnou detekovat objekty v reálném čase [22]. Metoda *Fast R-CNN* je relativně vůči *R-CNN* řádově rychlejší, avšak stále nestačí na detekci objektů v reálném čase [9].

SSD *(Single Shot Multibox Detector)*

Metoda *SSD* vznikla s velkým důrazem na rychlost celého procesu. Ten lze rozdělit do dvou částí. Jako první extrahuje pomocí sítě *VGG16* ze vstupního obrázku mapu příznaků, na kterou jsou následně aplikovány konvoluční vrstvy, čímž získáme mapy příznaků více velikostí [22], *viz obrázek 2.3*. Pro každou buňku obsaženou v těchto mapách je navrženo několik výchozích bounding boxů různých velikostí a poměrů stran. Jejich absolutní rozměry jsou ve všech vrstvách map příznaků stejné, tudíž v mapách s větším počtem buněk jsou detekovány objekty zabírající menší část snímku a při snižujícím se počtu buněk v mapách jsou detekovány objekty větší [22], *viz obrázek 2.4*. Velikost nejmenší i největší mapy příznaků lze v modelu konfigurovat [22]. Je výhodné tyto hodnoty nastavit podle použité datové sady, aby nedocházelo k plýtvání výpočetními zdroji skrze zbytečné pokusy o detekci objektů výrazně větších nebo menších. Na mapu příznaků v každém výchozím bounding boxu je aplikován konvoluční filtr rozměrů *3x3*, jehož výstupem je vektor obsahující informaci o třídě objektu a ofset jeho bounding boxu [22].



Obrázek 2.3: Na schématu architektury *SSD* lze vidět jak se postupnou aplikací konvolučních vrstev získávají mapy příznaků různých velikostí. Převzato z [22]



Obrázek 2.4: Prohledávání všech map příznaků pomocí bounding boxů stejných absolutních rozměrů umožňuje v každé z vrstev hledat objekty jiných velikostí. Převzato z [22]

2.2 Metriky vyhodnocování modelu

Soutěže a výzvy ve strojovém vidění posunují možnosti rozpoznávání objektů téměř každým rokem. Aby bylo možné porovnat přesnosti jednotlivých, nejen soutěžních, modelů, potřebujeme kvalitu jejich výstupu kvantifikovat. K tomu lze použít několik různých metod. Ty nejpopulárnější zmiňuji v této sekci. Přesnost modelů sloužících výhradně ke klasifikaci lze kvantifikovat relativně snadno, protože jejich výstupem jsou pouze dvě hodnoty - třída objektu a míra jistoty modelu v její správnost. Avšak výstup modelu určeného pro detekci objektů ve snímku je o něco komplexnější. Skládá se ze seznamu bounding boxů, tříd objektů a seznamu hodnot odpovídajících mírám jistoty modelu v danou predikci. Bounding box je "rámeček", ohraničující detekovaný objekt a jeho poloha se často udává jako pozice bodu v jeho levém horním a pravém dolním rohu ve formátu $(x_{min}, y_{min}, x_{max}, y_{max})$. Není to ale pravidlem, například algoritmus YOLO [28] definuje bounding box pozicí bodu v jeho středu a jeho rozměry [25]. Pro kvantifikaci přesnosti výstupu detektorů se v soutěžích a publikacích používají již pokročilejší metriky a jejich kombinace [26, 8, 31], *viz tabulka 2.2*.

Nejužívanější z nich jsou AP *(average precision)* a metody z ní odvozeny, jako je mAP *(mean average precision)* [25], kterým se podrobněji věnuji v sekci 2.2.

Metoda	Datová sada	Metriky
Fast R-CNN [9]	PASCAL VOC [8]	$AP, mAP(IoU = .50)$
Faster R-CNN [29]	PASCAL VOC [8]	$AP, mAP(IoU = .50)$
Faster R-CNN [29]	COCO [21]	$AP@[.5 : .05 : .95], AP@.50$
R-CNN [10]	PASCAL VOC [8]	$AP, mAP(IoU = .50)$
R-FCN [4]	PASCAL VOC [8]	$mAP(IoU = .50)$
SSD [22]	PASCAL VOC [8]	$mAP(IoU = .50)$
SSD [22]	ImageNet [32]	$mAP(IoU = .50)$
YOLO v1 [28]	PASCAL VOC [8]	$AP, mAP(IoU = .50)$

Tabulka 2.2: Populární architektury detektorů a metriky kvantifikující jejich přesnost na populárních datových sadách. *Data převzata z [26]*

Základní metriky vyhodnocování modelu

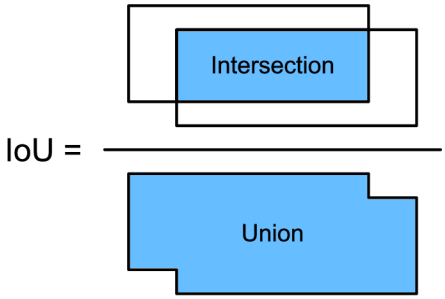
Nejzákladnějšími možnostmi kvantifikace výstupu detektoru je na:

- True positive (TP) - Správná detekce i lokalizace objektu
- False positive (FP) - Detekce neexistujícího objektu, nebo nepřesná lokalizace objektu existujícího
- False negative (FN) - Existující objekt nebyl správně detekován

Poslední z možností, True negative (TN), nelze na problém detekce objektu aplikovat, jelikož každá scéna obsahuje nekonečné množství bounding boxů, ve kterých se nenachází žádný objekt [26].

Aby bylo vůbec možné aplikovat takové rozdělení, je potřeba definovat si, v jakém případě model lokalizoval objekt ve scéně správně, a kdy nikoli. K tomu se nejčastěji používá hodnota *IoU (Intersection over Union)*, vyjadřující poměr obsahu průniku detekovaného bounding boxu B_p s bounding boxem reálným B_{gt}, k obsahu jejich sjednocení *viz rovnice 2.1 a obrázek 2.5*.

$$IoU = \frac{obsah(B_p \cap B_{gt})}{obsah(B_p \cup B_{gt})} \tag{2.1}$$



Obrázek 2.5: Grafické zobrazení výpočtu IoU. *Převzato z www.roelpeters.be*

Protože sjednocení dvou ploch bude mít vždy shodný nebo větší obsah než jejich průnik, hodnota IoU se pohybuje v intervalu $\langle 0, 1 \rangle$. Pokud je definován práh p, spadající do stejného intervalu, detekci lze považovat za úspěšnou, je-li $IoU \geq p$, a za neúspěšnou, je-li $IoU < p$ [25].

Dalšíma dvěma velice důležitými koncepty jsou *precision* a *recall*. *Precision* lze definovat jako schopnost modelu identifikovat pouze existující objekty. Jeho hodnota je poměr počtu správně detekovaných objektů k počtu všech detekcí modelu [25], *viz rovnice 2.2*. *Recall* lze definovat jako schopnost modelu detekovat všechny existující objekty [25]. Jeho hodnota se vypočítá jako poměr počtu všech správně detekovaných objektů k počtu všech anotací objektů existujících v datové sadě *viz rovnice 2.3* [26].

Součet správných detekcí a detekcí neexistujících objektů je roven počtu všech detekcí provedených daným modelem N. Součet správných a minutých detekcí odpovídá počtu všech anotací v dané datové sadě G. Pro výpočet *Precision* a *Recall* lze použít následující vztahy.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{N} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{G} \tag{2.3}$$

Pokročilé metriky vyhodnocování modelu založeny na *precision* a *recall*

Jak uvádím výše, *viz. 2.2*, výstup detektoru se skládá z bounding boxů, tříd a mír jistoty modelu. Jediná část tohoto výstupu, se kterou se ve výše uvedených metodách nepočítá, je míra jistoty modelu. Tu lze do výpočtů *precision* a *recall* zakomponovat, čímž získáme dodatečné informace o chování modelu. Toho lze docílit určením prahu p v intervalu $\langle 0, 1 \rangle$. Porovnáním prahu p s mírou jistoty modelu ve správnost detekce zjistíme, bude-li detekce do výpočtů zařazena, nebo jestli bude zahozena. *Precision* a *recall* lze vypočítat v závislosti na prahu p podle vztahů 2.5 a 2.4. Tyto rovnice jsou mírnou úpravou rovnic 2.2 a 2.3. Hodnoty $TP(p)$, $FP(p)$ a $N(p)$ lze vyjádřit jako počet detekcí v dané skupině s mírou jistoty modelu větší, než je práh p a jsou to nerostoucí funkce. Počet nedetekovaných objektů FN lze také vyjádřit v závislosti na prahu p, a to jako součet počtu úplně chybějících detekcí $FN(p)$ a počet správných detekcí *(TP)* s mírou jistoty nižší nebo rovnou prahu p. Tato funkce v závislosti na prahu p neklesá. Počet všech anotací v datové sadě G je hodnota na prahu p nezávislá, a tudíž má stejný význam jako v rovnicích 2.3 a 2.2.

$$Precision(p) = \frac{TP(p)}{TP(p) + FP(p)} = \frac{TP(p)}{N(p)} \tag{2.4}$$

$$Recall(p) = \frac{TP(p)}{TP(p) + FN(p)} = \frac{TP(p)}{G} \tag{2.5}$$

V praxi by dobrý detektor měl nalézt všechny anotované objekty v testovací datové sadě. To znamená, že hodnota FN by byla velmi nízká, tudíž by hodnota *recall* byla vysoká. Dále by měl identifikovat pouze relevantní objekty, tudíž by hodnota FP byla velice blízko nule, tudíž hodnota *precision* bude také vysoká. Proto se dá za dobrý detektor považovat ten, u kterého s klesající hodnotou prahu p zůstane *precision* na vysoké hodnotě a *Recall* bude stoupat [26]. Vykreslením závislosti *precision* na *recall* získáme tzv. *P-R* křivku, která je charakteristická svým 'zig-zag' tvarem [26], viz obrázek 2.6. I když vykreslením *P-R* křivky dostaneme mnoho informací o chování modelu, pro porovnání jejich výkonností je mnohem výhodnější kvantifikovat jejich přesnost jedinou hodnotou. Proto se často používá metoda *mAP (mean Average Precision)*, kdy se kvalita modelu vyjádří pomocí průměru *AP (Average precision)* pro každou třídu v datové sadě, *viz rovnice 2.6*.

$$mAP = \sum_{c \in C} \frac{AP(c)}{|C|} \tag{2.6}$$

Hodnota C referuje na množinu všech tříd v datové sadě a $AP(c)$ je *average precision* vypočítán pouze pro množinu detekcí objektů třídy c. AP lze vypočítat jako určitý integrál grafu pod *P-R* křivkou v intervalu $\langle 0, 1 \rangle$. Vzhledem ke zvláštnímu 'zig-zag' průběhu *P-R* křivky se výpočet často zjednodušuje interpolací této funkce v N bodech. Vzniklá funkce $P_{interp}(R)$ je monotónní a lze ji vyjádřit pomocí vztahu 2.7.

$$P_{interp}(R) = \max_{p:R(p) \geq R} \{P(p)\} \tag{2.7}$$

Hodnota $P_{interp}(R)$ je rovna maximu funkce $P(p)$, pro které platí $R(p) \geq R$.

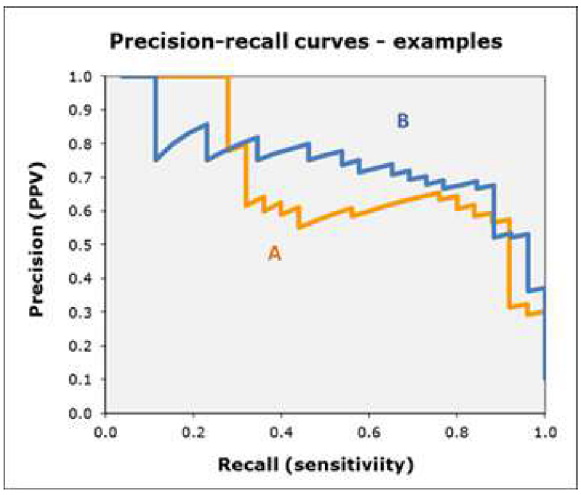
V praxi se aproximuje hodnota *average precision* výpočtem *Riemannova integrálu* v bodech z množiny $R_r(n)$ kardinality N, jejíž prvky jsou definovány vztahem

$$R_r(n) = \frac{N-n}{N-1}, \quad n = 1, 2, ..., N. \tag{2.8}$$

Prvky této množiny jsou rovnoměrně rozloženy na intervalu $\langle 0, 1 \rangle$ a obsahují hodnoty 0 a 1. Samotnou aproximaci *average precision* lze tedy zapsat výrazem 2.9.

$$AP = \frac{1}{N} \sum_{n=1}^{N} P_{interp}(R_r(n)) \tag{2.9}$$

Interpolace AP v 11 bodech se používá při vyhodnocování výsledků na datové sadě *Pascal* [8] a ve 101 bodech při vyhodnocování výsledků na datové sadě *COCO* [21].

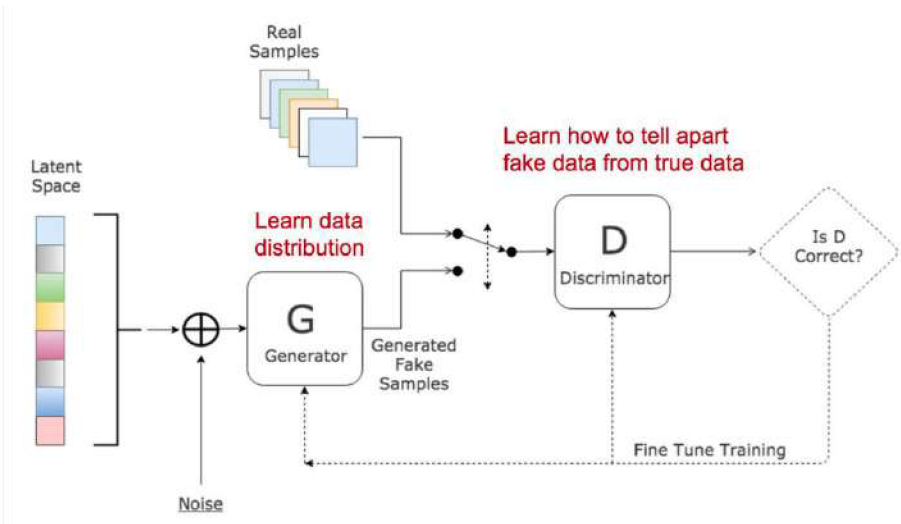


Obrázek 2.6: Graf ukazuje dvě P-R křivky s typickým "zig-zag"průběhem. Obrázek byl převzat ze zdroje acutecaretesting.org.

Kapitola 3

Neuronové sítě typu GAN

Jedním ze způsobů zlepšení výkonu neuronových sítí je rozšíření jejich trénovací datové sady pomocí generativního modelu. Jedna ze skupin generativních modelů jsou tzv. *GANy (Generative Adverserial Network)*, kterým se věnuji v této kapitole. Sítě typu GAN automaticky rozpoznají vzory ve vstupních datech. Za předpokladu konvergence procesu jejich trénování jsou schopny generovat syntetická data, téměř nerozpoznatelná oproti datům reálným, na základě pravděpodobnostního rozložení dat trénovacích [12]. Architektura GANu zahrnuje celkem 2 neuronové sítě a je znázorněná na *obrázku 3.1*. Neuronová síť, která na základě vstupu definovaného použitou architekturou, generuje syntetická data podobná původní datové sadě, se jmenuje *generátor*. Ten však ke svému natrénování potřebuje další neuronovou síť - *diskriminátor*. Jeho vstupem může být vzorek z původní datové sady nebo výstup *generátoru* a jeho úkolem je určit, jestli data na vstupu jsou reálná, čí nikoli. Na základě výstupu *diskriminátoru* jsou upravovány váhy *generátoru*.

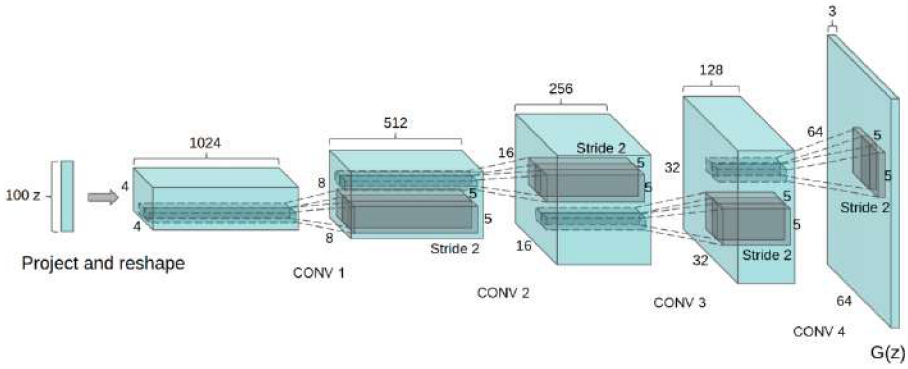


Obrázek 3.1: Obecné schéma GANu. *Generátor G* generuje vzorky z náhodného šumu. *Diskriminátor D* je trénován na reálných a generovaných datech a jeho váhy jsou upravovány nezávisle na celém modelu. Výstup diskriminátoru je pak společně se ztrátovou funkcí *generátoru* použit při výpočtech nových vah pro *generátor*.

GANy se dají zjednodušeně popsat analogií k padělatelům bankovek. Ti se je snaží padělat tak kvalitně, aby zůstali neodhaleni. Diskriminátor hraje roli policisty, snažícího se padělky detekovat. Soupeření policisty a padělatele vede oba zúčastněné k tomu, aby své metody zdokonalovali. Po dostatečně dlouhé době neustálého zlepšování techniky padělatele i policisty se padělatel naučí vytvářet bankovky naprosto totožné s těmi pravými [12].

Sítě typu GAN si od svého prvního představení prošly rapidním vývojem, *viz obrázek 3.2*. Architektura DCGAN, poprvé představena v roce 2016 [27], používá pro implementaci *generátoru* i *diskriminátoru*, *viz obrázek 3.3*, hluboké konvoluční neuronové sítě. Výsledky této metody jsou natolik uspokojivé, že většina moderních architektur sítí typu GAN je z DCGAN alespoň z části odvozena [11].

Obrázek 3.2: Demonstrace rychlého vývoje sítí typu GAN. Každý obrázek je výstupem *state of the art* generátoru v daném roce. *Převzato z [27]*



Obrázek 3.3: Architektura generativního modelu DCGAN pro modelování syntetických obrázků z datové sady LSUN [41]. 100 jednotek dlouhý vektor vzorkovaný z uniformního rozložení se promítne do prostorově malé konvoluční reprezentace s mnoha vrstvami příznaků. Následuje série čtyř konvolučních vrstev se zlomkovým krokem, které konvertují nízko rozměrný vstup až do obrázku rozměrů *64x64*. V modelu nejsou použity žádné plně propojené ani sdružující vrstvy. *Převzato z [27]*

3.1 Trénování generativního modelu

Sítě typu GAN jsou známy pro svou nestabilitu v průběhu trénovacího procesu a často generují naprosto nesmyslná data [27]. Původní publikace představující GANy přistupuje k jejich trénování jako ke hře dvou hráčů metodou minmax, kdy dochází k trénování *generátoru* i *diskriminátoru* zároveň. Tuto minmax hru lze zapsat rovnicí 3.3, kdy $D(x)$ představuje funkci *diskriminátoru*, která v případě, že si diskriminátor myslí, že má na vstupu reálná data, nabývá hodnoty 1, a pro data vygenerovaná nabírá hodnoty 0. Pro natrénování *diskriminátoru* je nutné maximalizovat pravděpodobnost, že *diskriminátor* správně rozpozná reálná data od těch falešných, *viz rovnice 3.1*. $G(z)$ představuje výstup *generátoru* závislého na šumu z, snažícího se tvořit reálně vypadající data [12]. Pro natrénování *generátoru* je proto potřeba minimalizovat pravděpodobnost, že *diskriminátor* správně rozezná generovaná data od těch reálných, *viz rovnice 3.2*. Pro výpočet pravděpodobností se používá křížová entropie [12].

$$\max_D V(D) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z))] \tag{3.1}$$

$$\min_G V(G) = \mathbb{E}_z[\log(1 - D(G(z))] \tag{3.2}$$

$$\min_G \max_D V(D, G) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z))] \tag{3.3}$$

V praxi se však ukázalo, že tyto funkce nemusí v brzké fázi učení poskytovat generátoru dostatečně silný gradient pro efektivní učení. V začátcích trénovacího procesu, kdy generátor neprodukuje data podobná těm reálným, může diskriminátor s velkou přesností data rozlišovat, čímž se trénink stane neefektivním [12]. Jedním z možných řešení je použití nesaturující ztrátové funkce, kdy místo minimalizování pravděpodobnosti, že diskriminátor odhalí falešná data, budeme maximalizovat pravděpodobnost, že diskriminátor klasifikuje falešná data jako reálná [12], *viz rovnice 3.4*

$$\max_G V(G) = \mathbb{E}_z[\log(D(G(z))] \tag{3.4}$$

Wasserstein GAN

Ačkoli jsou GANy schopny produkovat velice kvalitní syntetická data, jejich trénování je i přes dodržení doporučení uvedených dále v této sekci často nestabilní a může dojít k několika poruchovým režimům [2]. Jedním z takových režimů je tzv. *Mode Collapse*, kdy dochází ke generování dat pouze z některých částí pravděpodobnostního rozložení trénovací datové sady [35]. Problémy s nestabilitou sítí GAN stále nebyly úplně vyřešeny a jsou předmětem aktuálních výzkumů. Bylo však navrženo několik postupů, jak tyto problémy alespoň z části eliminovat [12, 35]. Jedním z nich je použití *Wasserstein GAN (WGAN)* [2], která je rozšířením k původnímu modelu [12]. *WGAN* dosahuje lepší stability během trénování a poskytuje ztrátovou funkci, která koreluje s kvalitou generovaných dat [2]. *WGAN* se během trénování snaží minimalizovat vzdálenost mezi daty poskytnutými na učení a generovanými daty, kterou definuje jako *Earth-Mover* nebo *wassersteinovu* vzdálenost. Tu lze formulovat jako nejnižší cenu přesunu hmoty při konverzi z pravděpodobnostního rozložení dat q do pravděpodobnostního rozložení dat p [2]. Hodnota této vzdálenosti je aproximována tzv. *kritikem*, který v modelu *WGAN* nahrazuje funkci diskriminátoru. Diskriminátor

se velice rychle naučí klasifikovat vstupní data na reálné a falešné, čímž přestane poskytovat spolehlivé informace o gradientu pro úpravu vah generátoru [12]. *Kritik* však poskytuje ztrátovou funkci spojitou a diferencovatelnou téměř ve všech bodech. Proto *kritik* v průběhu trénování poskytuje informace o gradientu se stále se zlepšující kvalitou [2], čímž roste stabilita generativního modelu a klesá jeho citlivost na konfiguraci hyperparametrů [2]. Během procesu trénování modelu se váhy *kritika* upravují častěji než váhy *generátoru*, v původní publikaci [2] autoři trénovali *kritika* pětkrát častěji než *generátor*.

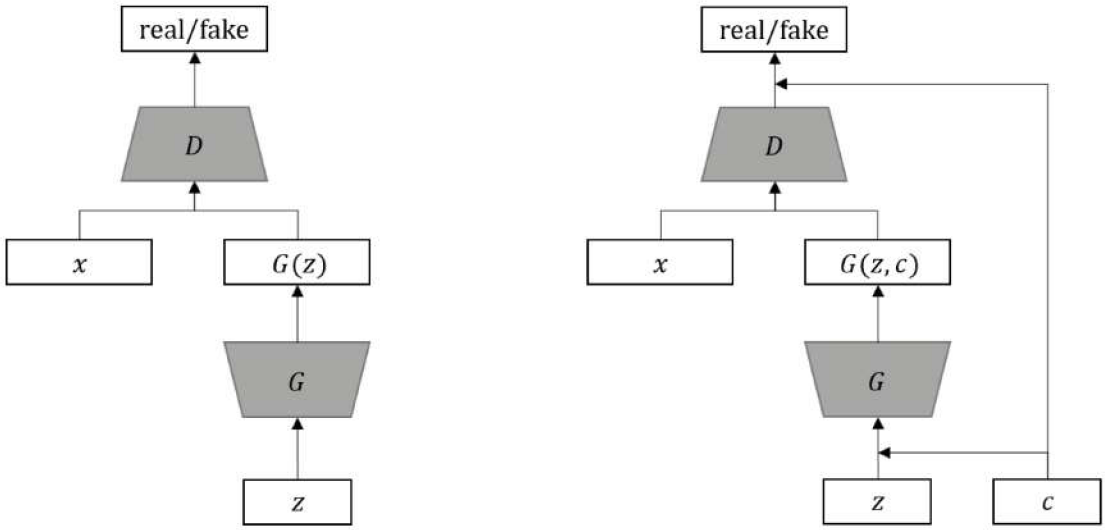
Pravidla pro stabilní trénování sítí GAN

V publikaci *DCGAN* [27] byl také zveřejněn seznam doporučení pro stabilní trénování, který se dá aplikovat na většinu architektur GANů.

- Všechny sdružující vrstvy vyměnit za konvoluční vrstvy s větším krokem pro diskriminátor i generátor a pro zvětšení rozlišení v generátoru použít konvoluční vrstvy se zlomkovým krokem
- Použít dávkovou normalizaci v generátoru i diskriminátoru
- Vyměnit plně propojené vrstvy za hlubší architektury
- Použít *ReLu* aktivační funkci pro všechny vrstvy generátoru, kromě té poslední, která použije aktivační funkci *Tanh*
- Použít *LeakyReLu* aktivační funkci pro všechny vrstvy *diskriminátoru*

3.2 Syntéza a transformace dat pomocí generativního modelu

Výše popsané modely jsou schopny generovat vizuálně velice kvalitní snímky [27]. Jejich funkce jsou však omezeny skutečností, že generátor přijímá na vstupu pouze náhodný šum. To je důvodem k tomu, že syntetická data jsou generována náhodně na základě rozložení dat trénovacích a během generování nelze dobře kontrolovat jejich vlastnosti [24]. Určitou kontrolu lze nad vlastnostmi generovaných snímků získat podmíněním vstupu diskriminátoru a generátoru dodatečnými informacemi o objektu, nejčastěji jejich třídou [24]. Vstup diskriminátoru je rozšířen o informaci, týkající se posuzovaného obrázku, a generátoru o informaci, kterou definujeme vlastnost generovaného snímku. Takový model se jmenuje *Conditional GAN* [24] a je znázorněn na *obrázku 3.4*.



Obrázek 3.4: Nalevo je schéma modelu nepodmíněné sítě typu GAN, složené z generátoru se vstupem v podobě šumu z a diskriminátoru se vstupem v podobě originálních nebo generovaných dat. Napravo lze vidět schéma modelu *Conditional GAN* a jeho rozšíření generátoru i diskriminátoru o dodatečnou informaci c. *Převzato z [19].*

Pix2Pix

Model *Pix2Pix* je rozšířením metody *Conditional GAN* a slouží k překladu obrázku na obrázek, kdy obrázek jedné formy transformuje na obrázek formy jiné, při zachování jeho sémantického významu [16]. Například pokud dostane generátor na vstupu obrázek vyfocený za deště, jeho výstupem může být odhad, jak by ta stejná scéna vypadala, kdyby nepršelo. Tradičně se každý z problémů překladu obrázku na obrázek řešil specializovanými metodami [16, 7], avšak model *Pix2Pix* je navržen s myšlenkou generalizace a lze ho aplikovat na většinu těchto problémů, například na transformaci satelitních fotografií na mapy, generování realistických fotek produktů z jejich náčrtů nebo zabarvování černobílých fotografií [16]. Ze své podstaty potřebuje *Pix2Pix* pro natrénování párovaná data, vzor pro transformaci a její cíl. Jako ostatní GANy je tvořen dvěma soutěžícími neuronovými sítěmi.

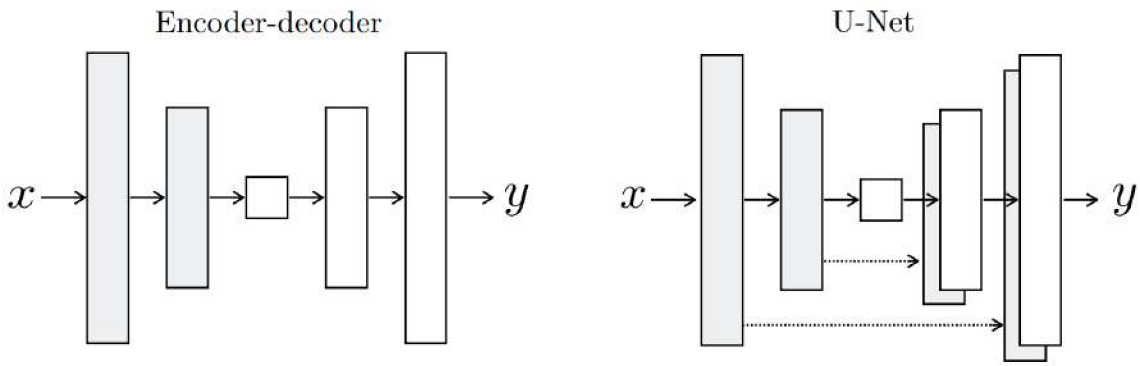
V původní publikaci [16] je použit model U-Net [30] jako generátor a Patch-GAN [16] jako diskriminátor.

Patch GAN

Obyčejné GANy pracují s diskriminátorem kvantifikujícím reálnost celého snímku do jedné hodnoty [12, 27]. Tato metoda však může při použití ve větším rozlišení generovat rozmazané a jinak nekvalitní obrázky [16]. *Patch-GAN* tento problém do jisté míry řeší hodnocením reálnosti jednotlivých částí scény [16]. Reálnost vstupu je zkoumána po jeho částech a je kvantifikovaná maticí hodnot odpovídajících reálnosti těchto částí. Rozměry hodnocených výřezů jsou definovány architekturou diskriminátoru a nejlepších výsledků při transformaci obrázků rozměru *256x256* bylo dosaženo hodnocením reálnosti snímků po částech o velikosti *70x70* [16].

U-Net

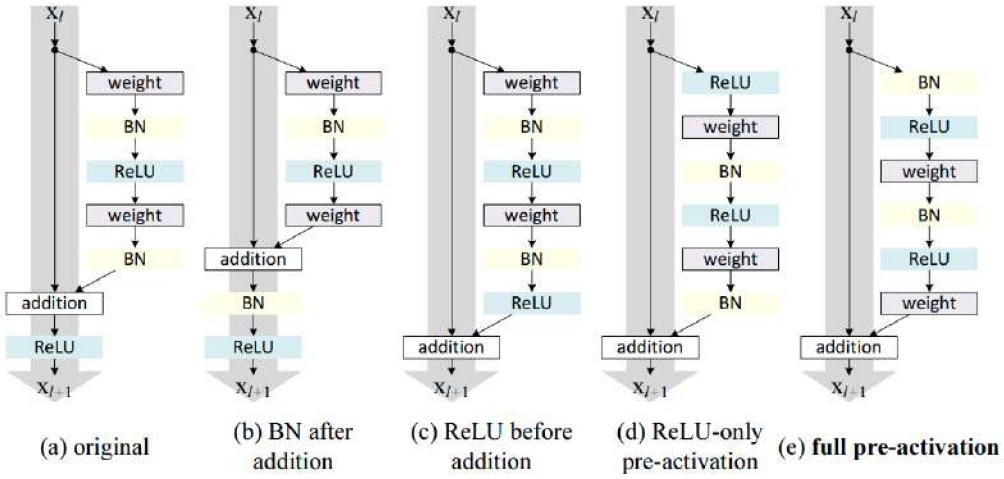
U-Net je moderní architektura neuronové sítě, transformující vstupní obrázky pomocí konvolučních filtrů [30]. Její struktura vychází z architektury *Encoder-Decoder* 3.5, avšak navíc využívá *skip connections*, propojení konvolučních vrstev stejných velikostí mezi *kodérem* a *dekodérem*. *U-net* oproti předchozím metodám dokáže vytvářet kvalitní výstupy i z menšího množství příkladů a závisí hlavně na velké augmentaci dat [30].



Obrázek 3.5: Porovnání architektur *Encoder-Decoder* a *U-Net*. Části sítě, zabarveny šedě, se nazývají kodér. Na jejich konci je nejužší místo modelu, tzv. *bottleneck* neboli úzké hrdlo. Následuje druhá část sítě s bíle zabarvenými bloky, dekodéry. Funkční bloky jednotlivých částí se skládají z konvolučních filtrů, dávkové normalizace a *ReLu* [16]. Konvoluční filtry v kodéru používají krok $s_k > 1$ a v dekodéru krok $s_d = 1/s_k$. *U-net* navíc používá *skip connections*, které propojují výstupy kodérů se vstupy dekodérů stejných velikostí. *Převzato z [16].*

RU-Net

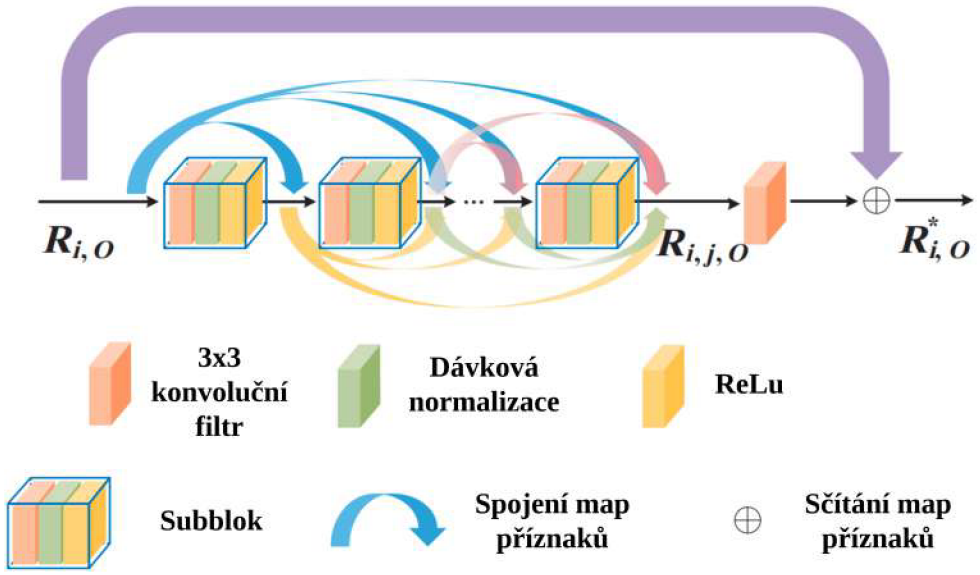
Architektura *ResNet* [13] oproti přímé architektuře konvolučních neuronových sítí pro extrakci příznaků dosahuje lepších výsledků a stabilnějšího trénování [13]. Sítě *ResNet* se skládají z reziduálních bloků *viz obrázek 3.6*. Reziduální blok se skládá z několika konvolučních a aktivačních vrstev a dávkové normalizace. Vstup reziduálního bloku je však přičten k výstupu jedné z jeho posledních vrstev. RU-Net [42] kombinuje principy architektury U-Net a ResNet, čímž lze docílit kvalitnějších výsledků při sémantické segmentaci scény [42]. Místo kodérů používá RU-Net reziduální blok následován konvoluční vrstvou s větším krokem a místo dekodérů reziduální blok následován konvoluční vrstvou se zlomkovým krokem.



Obrázek 3.6: Několik možných architektur reziduálního bloku skládajících se ze stejných komponent, avšak v jiném pořadí. Tyto architektury byly testovány na datové sadě CIFAR-10 [17] a nejlepšího výsledku dosáhla architektura na obrázku *e*, která přičítá vstup bloku až k jeho výstupu [14]. Druhá nejúspěšnější byla architektura použitá v původní publikaci ResNet [13, 14]. *Převzato z [14].*

Dense RU-Net

Dense RU-Net je podobným rozšířením jako RU-Net, avšak místo reziduálních bloků používá DenseNet [15] reziduální bloky. Ty se skládají z několika menších sub bloků, obsahujících konvoluční vrstvu, dávkovou normalizaci a aktivační vrstvu. Stejně jako u reziduálních bloků se vstup DenseNet bloku sečte s jeho výstupem [39]. Vstup každého ze sub bloků vznikne spojením vstupu celého bloku s výstupy všech předchozích sub bloků, *viz obrázek 3.7*



Obrázek 3.7: Schéma DenseNet bloku. Oproti klasickému se liší strukturou sestavenou z několika sub bloků. Každý vstup sub bloku je pak spojením výstupů všech předchozích sub bloků a vstupu celého bloku. Počet sub bloků může mít vliv na kvalitu generovaných snímků. *Převzato z [39].*

Kapitola 4

Možnosti využití generativního modelu k rozšiřování datových sad pro detekci dopravních značek

Detekce dopravních značek spadá pod množinu úkolů počítačového vidění. Obraz reálné situace je snímán kamerou a zpracován tak, že detekuje a klasifikuje dopravní značky ve snímané scéně. Dopravní značky hrají významnou roli při navigaci po silnicích. Definují maximální povolenou rychlost, určují nebo zakazují směr jízdy, upozorňují na možné nebezpečí nebo upravují přednost v jízdě vozidel.

4.1 Aplikace systémů detekce dopravních značek

Prvním automobilem, disponujícím funkcí detekce dopravních značek, byl Opel Insignia v roce 2008 [34]. Ten však dokázal číst pouze značky definující maximální povolenou rychlost a upozorňovat řidiče v případě, že ji překročí. S rostoucí přesností metod detekce objektů [31, 8] a výkonností čím dál tím menších čipů a senzorů, roste míra aplikovatelnosti takových systémů nejen v automobilovém průmyslu. Tyto systémy však již neslouží pouze pro poskytnutí informace řidiči, který koná na základě informací jeho samotného, ale systémy detekce slouží pouze jako sekundární zdroj informací. Dnešní moderní systémy jsou schopny velice přesného a konzistentního rozpoznávání dopravního značení [1] a poskytují informace systémům jim nadřazeným, které vozidlo autonomně ovládají, tudíž na přesnosti poskytnutých informací přímo závisí další činnost autonomního systému. Pokud takový systém selže alespoň v jednom z úkolů porozumění okolí (detekce cesty, detekce překážek a detekce a klasifikace dopravních značek) [6], informace o okolí nejsou zcela validní a může dojít k nehodě [38]. Navzdory této skutečnosti však k roku 2021 jezdí legálně po silnicích již několik automobilů s možností plně autonomního ovládání od firmy *Waymo*, *Honda* nebo *Tesla*.

4.2 Problémy při detekci dopravních značek

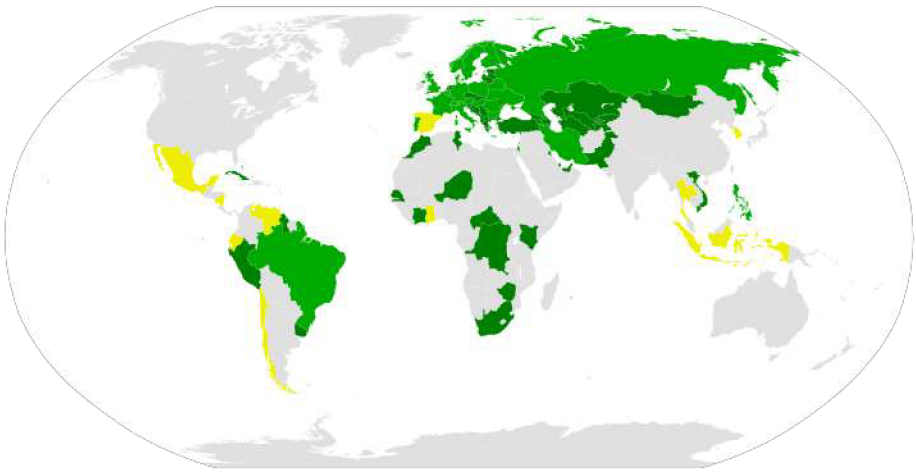
Ve vzhledu běžných objektů, jako jsou auta nebo osoby, se často objevují velké rozdíly i mezi instancemi objektů v rámci stejné třídy. Dopravní značky jsou však definovány legislativou dané země a v její rámci by se neměly příliš lišit. Většina značek je v relativně dobrém stavu a od své, zákonem definované, normy se příliš neliší. Takové značky jsou dobře detekovatelné, protože používají syté výrazné barvy. Na silnici se však nachází nemalé množství značek, u kterých je jejich rozpoznání náročnější z několika možných důvodů, *viz obrázek 4.1*. Schopnost detektoru správně se rozhodnout v náročných situacích lze podpořit častějším výběrem takových příkladů během tréninku, nebo rozšířením datové sady o náročné případy pomocí generativního modelu. Systém může na silnici narazit na nepřeberné množství jedinečných situací. Některé z nich jsou zachyceny v datových sadách, avšak je nemožné zachytit je všechny a v dostatečné kvantitě. Tím vzniká množina situací, které, pokud na ně model narazí v provozu, nemusí být vyhodnoceny správně, protože nastalá situace je pro model nezvyklá. Tím může být například zvláštní deformace dopravní značky, což může vézt k její špatné klasifikaci, nebo ji model nemusí detekovat vůbec. Pomocí generativního modelu pak lze tyto situace syntetizovat, čímž lze model dobře generalizovat i na takové případy. To může vést ke zlepšení konzistentnosti detekujícího modelu, a tudíž i k robustnějším a bezpečnějším systémům [3]. Společnosti zabývající se vývojem plně autonomních automobilů závisí z velké části na syntéze dat. Pokročilé generativní modely dokážou vyrobit násobně více syntetických dat oproti datům originálním, a to ve vysoké kvalitě [40]. Generování těchto dat je navíc kontrolováno tak, aby zachytilo situace pro systém relativně neznámé, čímž lze dosáhnout jeho výrazného zlepšení.

Obrázek 4.1: Na obrázcích jsou uvedeny případy náročné pro modely detekující dopravní značky. Pomocí generativního modelu lze tyto obtížné případy syntetizovat, čímž lze zpřesnit detekci deformovaných nebo jinak hůře detekovatelných, značek.

4.3 Rozdíly ve značkách napříč zeměmi

Velkou roli při detekci dopravních značek hraje geografická lokalizace. Historicky se dopravní značky lišily téměř ve všech zemích, ale již dlouhou dobu existují mezinárodní snahy tento systém sjednotit za účelem zvýšení bezpečnosti. Největší takovou snahou je *Vídeňská*

úmluva a dopravních značkách a signálech[1], kterou podepsalo k srpnu 2016 68 států, z nichž většina tuto konvenci ratifikovala *viz obrázek 4.2*.



Obrázek 4.2: Mapa světa s barevně zvýrazněnými státy užívajícími normy definované ve Vídeňské úmluvě. Světle zelené státy úmluvu ratifikovaly, tmavě zelené ji podepsaly a v praxi ji používají a žlutě vyznačené státy ji používají, aniž by ji podepsaly. *Převzato z www.wikipedia.org*

Skutečnost standardizace dopravních značek povede k větší přenositelnosti autonomních systémů řízení automobilu mezi různými státy. Do té doby však kvůli různorodosti dopravních značek je pro natrénování detektoru v různých zemích často potřeba datových sad z místních silnic, *viz obrázek 4.3*. Existence generativního modelu navrženého v kapitole 5 by tuto potřebu eliminovala vytvořením umělé datové sady vycházející pouze z norem dopravních značek definovaných danou zemí a výchozí datové sady.

Moderní architektury modelů pro detekci objektů jsou schopny velice spolehlivé detekce dopravních značek na veřejných datových sadách [1]. Dobře viditelné značky, jejichž instancí je v trénovací datové sadě více, modely detekují téměř perfektně. Modely však často detekci minou, je-li v trénovací datové sadě málo příkladů dané třídy značky, nebo pokud je daná dopravní značka vizuálně odlišná, například deformací nebo světelnými podmínkami. Generativní model navržen v kapitole 5 dokáže syntetizovat kvalitní snímky, obsahující objekty málo se vyskytujících tříd, a tím při učení vyrovná jejich deficit.

[1] Donstupné na adrese: https://unece.org/DAM/trans/conventn/Conv_road_signs_2006v_EN.pdf

Obrázek 4.3: Srovnání vzorku dopravních značek tří různých zemí definovaných jejimi zákony. Česká republika i Slovensko jsou součástí *Vídeňské úmluvy o dopravních značkách a signálech*. Přesto si lze mezi jednotlivými značkami všimnout menších rozdílů. Naprostá většina značek si jsou však velice podobné. Irsko, které její součástí není, používá normy dopravního značení značně odlišné od norem používaných v jiných zemích. *Obrázky značek převzaty z* www.wikipedia.org

Kapitola 5

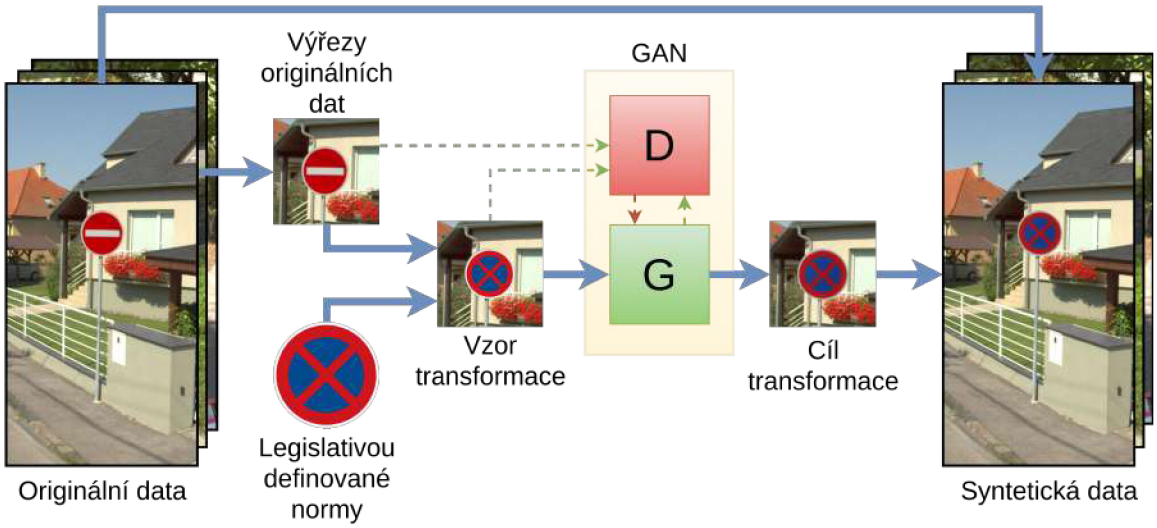
Architektura použitých modelů a tvorba datových sad

Celý systém je implementován v jazyce **Python3**[1]. Generativní model je implementován pomocí knihovny *TensorFlow*[2] ve verzi *2.4.1*, využívající technologií *CUDA*[3] pro akceleraci výpočtů na grafické kartě a *cuDNN*[4], poskytující efektivní implementace primitiv neuronových sítí, jako konvoluční filtry nebo aktivační vrstvy. Pro manipulaci s grafickými daty jsem použil knihovnu *OpenCV*[5] a pro augmentaci dat knihovnu *imgaug*. Pro trénování a evaluaci detektoru jsem použil framework *TensorFlow Object Detection API*[6], který poskytuje implementaci modelů pro detekci objektů a skripty potřebné pro jejich natrénování a vyhodnocení. Skript `model_main_tf2.py` však musel být mírně upraven, protože ve své původní verzi chybně alokoval paměť na grafické kartě. Systém se skládá z celkem dvou modulů. Jeden implementuje operace s neuronovými sítěmi a druhý operace s výřezy a celou datovou sadou. Funkcionalita těchto modulů je pak využita ve skriptu **train.py**, který umožňuje nakonfigurovat a natrénovat generativní model a **createTFR.py**, který je schopen vytvářet trénovací datové sady pro detektor ve formátu TFRecord. Tento skript je schopen generovat záznamy TFRecord ze syntetických i originálních dat a jejich kombinací.

5.1 Generativní model

Při rozšiřování datové sady pro detekci je důležité znát informace o třídě a poloze objektů v syntetických snímcích. Kontrolu nad třídou generovaného objektu lze zajistit použitím podmíněné sítě, *viz sekce 3.2*. Zachování polohy objektu ve snímku lze docílit transformací pouze oblasti, v níž se objekt nachází. Transformovanou oblast pak stačí vložit zpět do snímku na stejné místo. Tím si objekt zachová svou pozici, ale jeho vzhled se změní. Během tohoto procesu je však nutností, aby generativní model neměnil pozadí, protože by po vložení objektu zpět do původního snímku data nevypadala přirozeně. Tyto anomálie by se nacházely pouze v oblasti objektu a detektor by se mohl naučit objekty lokalizovat na základě jejich přítomnosti, čímž by byl na reálných příkladech nepoužitelný.

[1] Dostupné na adrese: python.org/
[2] Dostupné na adrese: tensorflow.org/
[3] Dostupné na adrese: developer.nvidia.com/cuda-downloads
[4] Dostupné na adrese: developer.nvidia.com/cudnn
[5] Dostupné na adrese: opencv.org/
[6] Dostupné na adrese: github.com/tensorflow/models/tree/master/research/object_detection



Obrázek 5.1: Na obrázku je znázorněno schéma generativního modelu. Nepřerušované šipky popisují proces tvorby syntetických dat. Nejprve je z původního snímku vyříznuta dopravní značka. Ta je poté překryta normou jiné značky stejného tvaru. Překrytý obrázek je poté zpracován generativním modelem, který transformuje vložený obrázek na reálně vypadající dopravní značku. Výstup generátoru je vložen zpět na původní místo ve scéně, čímž získáme nový trénovací příklad pro detektor. Každá dopravní značka v původní datové sadě tedy může být transformována na značku jiné třídy, avšak stejného tvaru. Přerušované šipky popisují proces učení generativního modelu. Diskriminátor posuzuje reálnost originálních a transformovaných výřezů, architektura PatchGAN, *viz sekce 5.1.2* však při vyhodnocení reálnosti vstupu, narozdíl od WGAN, poskytuje diskriminátoru také vzor transformace. Na základě výstupu diskriminátoru a ztrátové funkce generátoru se upravují váhy generátoru. Diskriminátor je trénován samostatně.

5.1.1 Tvorba trénovací datové sady pro generativní model

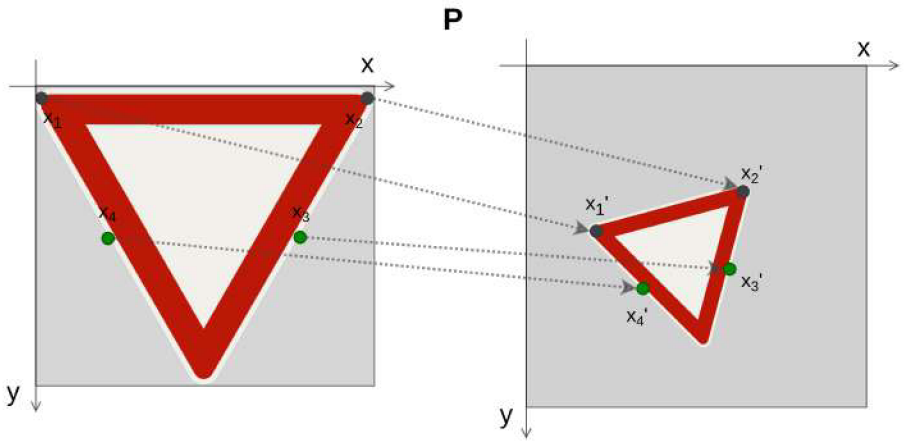
Každá dopravní značka je vyrobena podle legislativou definovaného vzoru. Na Slovensku je definuje *Vyhláška ministerstva vnútra č. 9/2009 Z. z*, jejíž obsah je volně dostupný zde[7]. Toho lze využít při tvorbě trénovací datové sady pro generativní model. Vyříznutím značek z původních scén získáme cíle transformace. Tyto cíle odpovídají vzhledem obrázkům, které by generátor měl syntetizovat. Pro trénování neuronové sítě pro transformaci dat je potřeba každému cílovému příkladu přiřadit jeho vzor. Generativní model se v nejlepším případe naučí generalizovaný způsob transformace dat ze vzorové domény na data odpovídající doméně cílové. Datovou sadu vzorů jsem vytvořil překrytím cílových dat obrázkem normy značky stejné třídy, definovaným legislativou a transformovaným tak, aby kopíroval obrys původní značky, *viz obrázek 5.4*. Obrázky norem jsou před vložením do výřezu mírně upraveny pomocí náhodného šumu, změny kontrastu a náhodným vynásobením celé matice obrázku číslem z intervalu $\langle 0.9, 1.1 \rangle$. Vzniklá párová datová sada je pak poskytnuta generativnímu modelu během trénování.
Po úspěšném natrénování modelu lze množinu vzorových trénovacích snímků rozšířit překrytím výřezu značky obrázkem normy značky jiné třídy. Vzorové výřezy jsou pak zpracovány

[7]Obsah dostupný na adrese: https://www.ssc.sk/sk/Technicke-predpisy-rezortu/zoznam-vl.ssc

generativním modelem a jeho výstup je vložen zpět do původní scény, ve které se nachází původní výřez, *viz obrázek 5.1*. Pro účely trénování a validace systému byla použita neveřejná datová sada ze Slovenských silnic.

Perspektivní transformace

Perspektivní transformace je lineární projekce trojrozměrného tělesa na plochu. Vzhledem k trojrozměrné povaze prostoru, v němž byly dopravní značky zachyceny, se jedná o vhodnou metodu transformace normy dopravní značky pro překrytí značky v původním výřezu, *viz obrázek 5.4*. Matici perspektivní transformace lze vypočítat, známe-li pozice čtyř bodů před i po transformaci. Výhodou datové sady ze Slovenských silnic je anotace objektů pomocí *keypoints* , *viz obrázek 5.3*. *Keypoints* definují objekty jednoduchých tvarů nejkratším seznamem bodů, podle kterého lze tento tvar přesně určit, včetně jeho pozice ve snímku. Například n-úhelník je anotován n body v jeho vrcholech. Většina značek je čtyřúhelníkového nebo kruhového tvaru, které jsou anotovány čtyřmi body. Ty lze použít při výpočtu transformační matice. Trojúhelníkové značky jsou však anotovány pouze třemi body, proto jsem pro účel transformace zvolil pouze body vrcholů u základny trojúhelníku a další dva cílové body jsou vypočítány jako středy jeho ramen, *viz obrázek 5.2*.



Obrázek 5.2: Příklad použití perspektivní transformace během vytváření vzorů pro generativní model. Obrázku normy značky je nejprve změněna velikost, aby byla shodná s výřezy značek. Poté jsou v závislosti na tvaru značky určeny čtyři vzorové body pro transformaci x_1, x_2, x_3, x_4, které se v obrázku normy nachází na stejném místě značky, jako cílové body transformace x'_1, x'_2, x'_3, x'_4 v původním výřezu značky. Z těchto bodů vypočítáme transformační matici P a její aplikací získáme obrázek značky, který přesně překryje značku ve výřezu.

Obrázek 5.3: Příklad anotace dopravní značky z použité datové sady pomocí dvou metod. Vlevo je objekt anotován pomocí *keypoints*, vpravo pomocí *bounding boxu*. Lze si povšimnout, že anotace pomocí *keypoints* poskytuje na rozdíl od bounding boxu informace o přesné poloze značky v prostoru.

Obrázek 5.4: Ve většině případů je použití afinní transformace sloužící k projekci dvourozměrného objektu na rovinu postačující. Avšak v některých případech, kdy je zachycená značka blízko kameře a na snímku je znatelný její třetí rozměr, neposkytuje dostatečně kvalitní překrytí původní značky.

Charakteristika trénovací datové sady pro generativní model

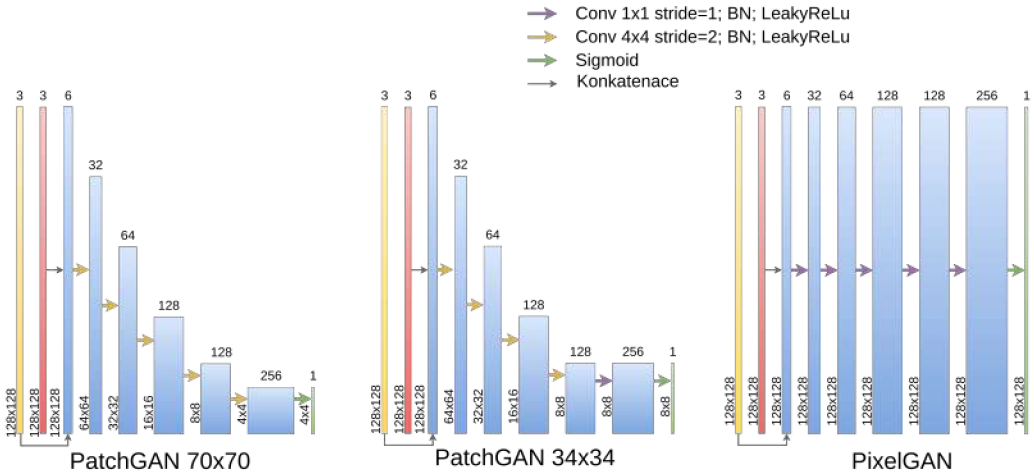
Některé výřezy, u kterých značka přesahovala přes okraj, byly vyfiltrovány, protože nejsou vhodnými příklady pro generátor, který se učí upravovat pouze objekty uprostřed výřezů a jejich okolí měnit co nejméně. Dále byly vyfiltrovány výřezy se značkami zachycenými zezadu a také výřezy s třídami značek, které v rámci své třídy mění svůj vzhled, jako třeba názvy měst nebo maximální nosnost mostu. Tyto značky je složité spárovat s příslušným obrázkem normy. Celkem datová sada ze Slovenských silnic pro trénování generativního modelu obsahuje 1554 párů vzorových a cílových dat v rozlišení 128x128, které poskytuje dostatek informací o výřezu, při jeho vložení zpět do scény, aby výsledný snímek nevypadal rozmazaně.

5.1.2 Architektura diskriminátoru

Volba diskriminátoru může mít velký vliv na kvalitu generovaných dat [2]. V této práci jsem použil diskriminátory *PatchGAN*, hodnotící reálnost snímků po jejich částech, *viz sekce 3.2* a kritiky aproximující *Earth-Mover* vzdálenost mezi syntetickými a originálními daty, *viz sekce 5.1.2*.

PatchGAN

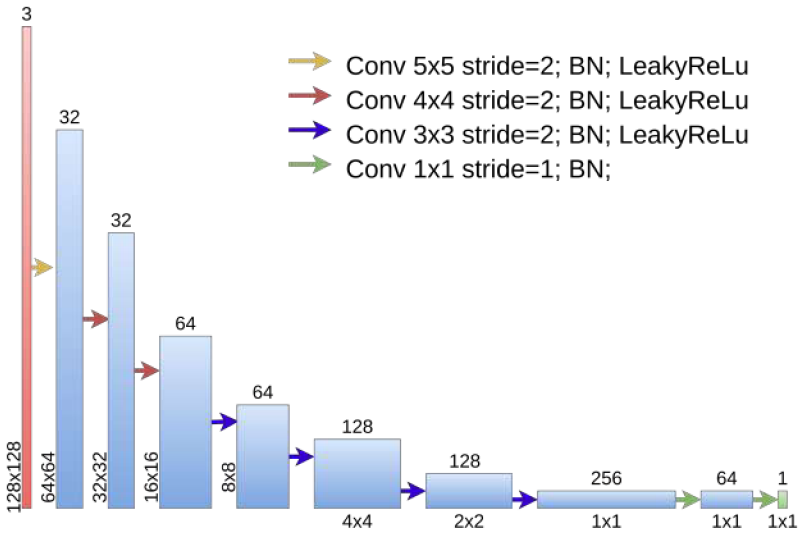
Diskriminátor sítě PatchGAN, na rozdíl od obyčejného diskriminátoru, hodnotí reálnost vstupního obrázku po jeho částech. Na vstupu přijímá vzorový a originální nebo syntetický snímek. V původní publikaci Pix2Pix [16] došli její autoři k závěru, že na datech s rozlišením 256x256 dosahuje nejuspokojivějších výsledků architektura hodnotící reálnost snímku po částech velikosti 70x70. V této práci jsem však zvolil data rozlišení 128x128, a proto je možné, že nejkvalitnějších výsledků bude model dosahovat v jiné konfiguraci. Transformace dat generativním modelem je velice mírná a týká se pouze určitých oblastí výřezu značky, proto jsem provedl sérii experimentů s architekturami uvedenými *na obrázku 5.5*, *viz experiment 6.4*, abych zjistil, která konfigurace poskytne nejlepší řešení. Pro předejití situace, kdy se diskriminátor naučí až příliš dobře hodnotit reálnost vstupu, čímž by zastavil trénování generátoru, je jeho ztrátová funkce při výpočtu nových vah vynásobena hodnotou 0.5, čímž se zpomalí jeho trénování, avšak poskytne dostatečný gradient pro trénování generátoru. Jako ztrátová funkce generátoru se používá průměrná absolutní chyba [16]. Při výpočtu nových vah generátoru se používá kombinace ztrátové funkce generátoru a výstupu diskriminátoru v poměru, který také může mít vliv na kvalitu generovaných snímků. V původní publikaci Pix2Pix [16] autoři dosáhli nejlepších výsledků použitím poměru 100:1 ve prospěch ztrátové funkce generátoru.



Obrázek 5.5: Schéma architektury sítí PatchGAN a PixelGAN (speciální případ PatchGAN, hodnotící reálnost snímku po jednotlivých pixelech). Žlutá vrstva odpovídá vstupu v podobě vzorového a červená v podobě hodnoceného snímku. Ty jsou spojeny a dále je na ně aplikováno několik konvolučních vrstev. Velikost filtru a kroku konvoluční vrstvy určuje velikost hodnocených částí obrázku. Po poslední konvoluční vrstvě následuje aktivační funkce sigmoid, jejíž výstupem je matice hodnot aproximujících reálnost jednotlivých částí snímků.

WGAN

Další architektura GAN, použita v této práci, je WGAN, *viz 3.1*. Jejím vstupem je obrázek v rozlišení 128x128 a jejím výstupem je jediná hodnota aproximující Earth-Mover vzdálenost [2]. Tato síť obsahuje pouze konvoluční vrstvy, dávkové normalizace a aktivační funkci *LeakyReLu* s parametrem $\alpha = 0.2$ [2]. Vliv na kvalitu snímků může mít změna architektury kritika nebo změna poměru četnosti úpravy vah kritika a generátoru. Aarchitektura použita v této práci je znázorněna na obrázku 5.6. Při implementaci výpočtu ztrátové funkce a úpravy vah generativního modelu jsem se inspiroval existujícím řešením[8].



Obrázek 5.6: Schéma architektury kritika použitého při experimentování v kapitole 6. Rozměry vstupního obrázku se pomocí konvolučních vrstev s krokem 2 postupně redukují až na mapu příznaků velikosti 1x1x256. Na tu jsou poté aplikovány další dvě konvoluční vrstvy s filtrem 1x1 a krokem 1, čímž dojde k jeho redukci na skalární hodnotu aproximující Earth-Mover vzdálenost.

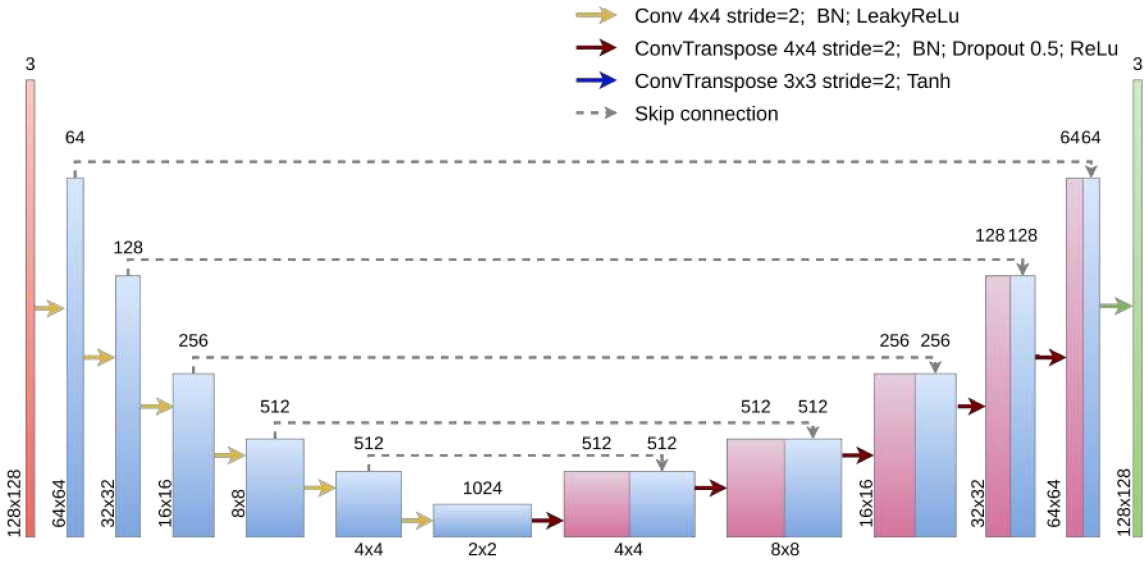
[8] Dostupné na adrese: kaggle.com/amanooo/wgan-gp-keras

5.1.3 Architektura generátoru

V této práci jsem použil několik architektur generátorů, založených na modelu U-Net [16], *viz sekce 3.2*. Konkrétní použité modely se od sebe mohou lišit svou hloubkou (počtem kodérů a dekodérů), použitím vrstev *dropout* a architekturou funkčních bloků.

U-Net

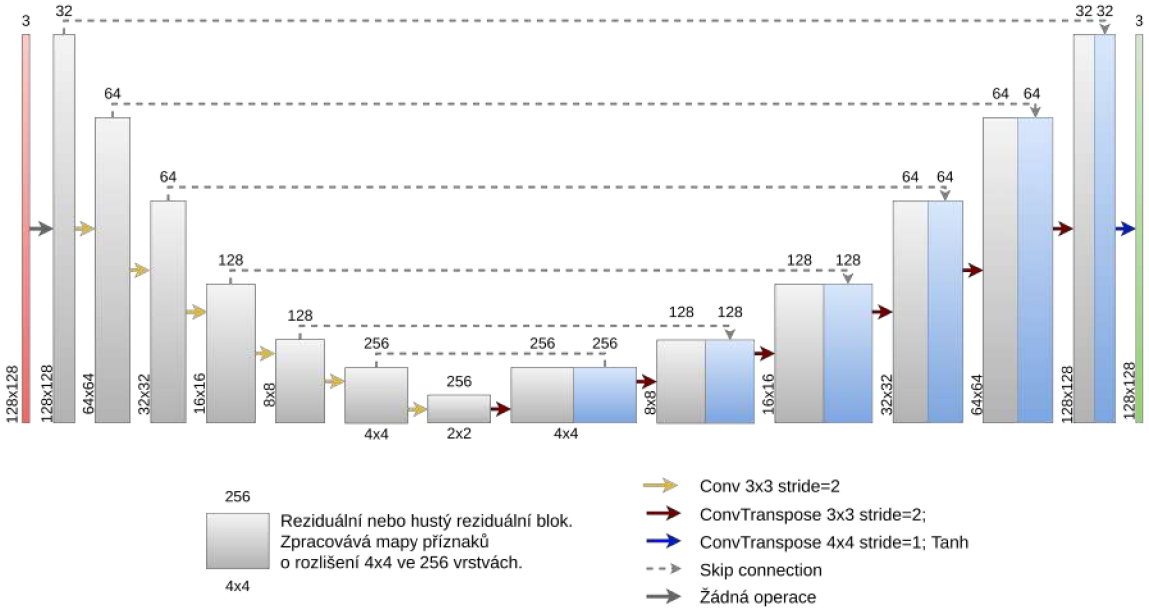
U-Net je nejjednodušší z použitých architektur. V první části, pomocí konvolučních vrstev s větším krokem, postupně zmenšuje rozlišení vstupního obrázku a v druhé části, pomocí konvolučních vrstev se zlomkovým krokem, rozlišení navyšuje do původního stavu, *viz sekce 3.2*. Všechny obyčejné sítě U-Net, použity v experimentech v kapitole 6, obsahují 6 kodérů a 6 dekodérů, *viz obrázek 5.7*. Není-li uvedeno jinak, ve všech dekodérech je použita vrstva *dropout* s parametrem $p = 0.5$.



Obrázek 5.7: Funkční schéma použité architektury U-Net.

RU-Net a Dense RU-Net

RU-Net a Dense RU-Net, *viz sekce 3.2*, jsou rozšířením architektury U-Net o reziduální bloky, *viz obrázek 5.8*. V modelech byly během experimentů použity celkem 3 druhy funkčních bloků. Prvním z nich je původní reziduální blok [13] a druhým je tzv. *pre-activation* reziduální blok, viz sekce 3.2 a obrázek 3.6, který by měl dosahovat lepších výsledků [14]. Dále byly použity DenseNet bloky v několika variantách s rozdílným počtem sub bloků, *viz obrázek 3.7*. Ve všech blocích po aktivačních vrstvách následuje vrstva *dropout* s parametrem 0.8.



Obrázek 5.8: Generalizované schéma sítě RU-Net a Dense RU-Net. Síť, stejně jako U-Net, používá *skip connections* a také vstupní obrázek postupně pomocí konvolučních filtrů s krokem 2 zmenšuje na velice malou mapu příznaků až do tzv. úzkého hrdla a následně jej, opět pomocí konvolučních filtrů se zlomkovým krokem, transformuje v obrázek stejných rozměrů, jako vstup. Oproti architektuře U-Net však před každou konvoluční vrstvou předchází reziduální nebo DenseNet blok. Použité modely se liší převážně v architektuře funkčních bloků a v jejich počtu.

5.2 Model detektoru

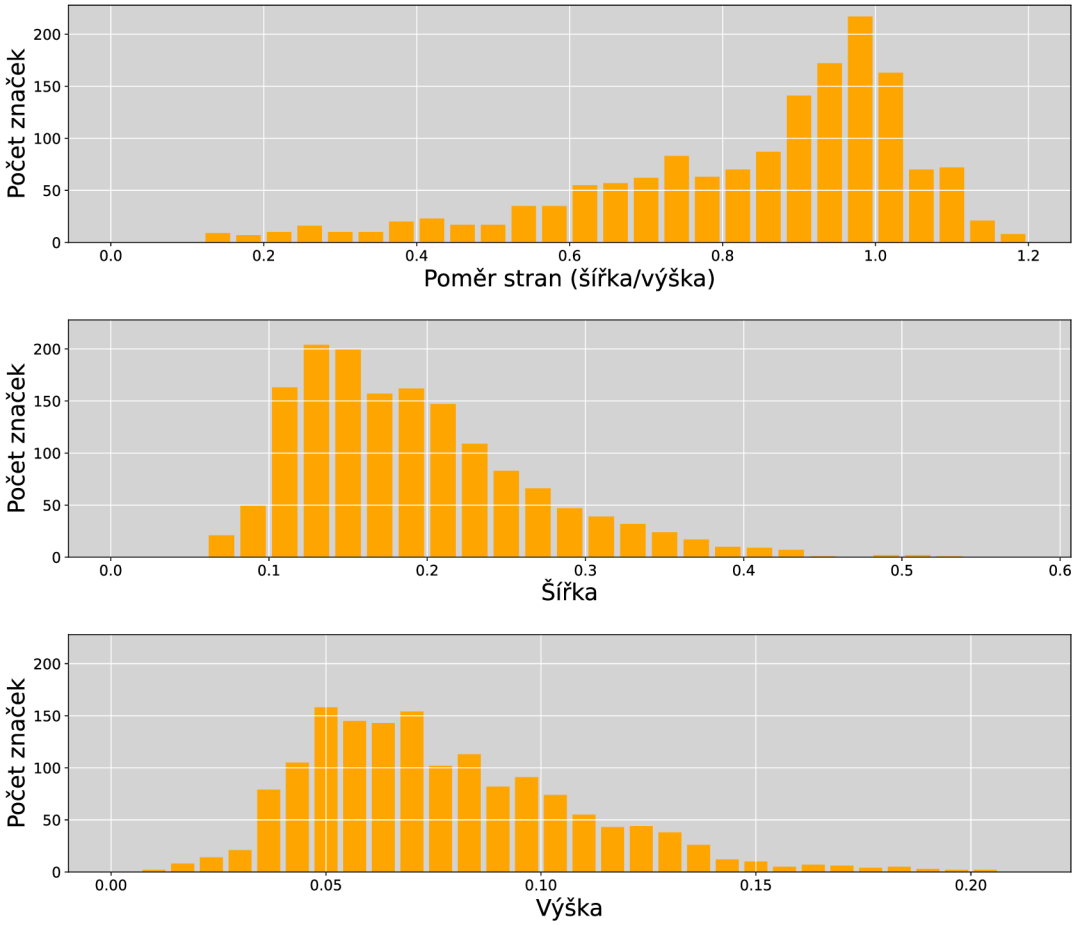
Jako model pro detekci dopravních značek jsem zvolil neuronovou síť architektury SSD, viz sekce 2.1, konkrétně jeho *open source* implementaci *SSD MobileNet V2 FPNLite 640x640*, která je veřejně dostupná z[9]. Použití tohoto modelu na problém detekce dopravního značení je vhodné, především díky jeho vysoké rychlosti a malé náročnosti na paměť. Architektura SSD zvládá vyhodnocovat snímky několikrát za sekundu a je schopna detekce objektů v reálném čase [22], což je klíčová vlastnost detektoru určeného pro rozpoznávání dopravních značek.

5.2.1 Konfigurace detektoru

Model byl nakonfigurován, aby během procesu učení aplikoval augmentaci v podobě náhodného upravení jasu, odstínu a zabarvení a s 50% pravděpodobností vertikální otočení snímku. Náhodné horizontální otočení snímku u některých tříd značek způsobí změnu jejího sémantického významu, na rozdíl od vertikálního, proto jsem horizontální otočení do možností augmentace snímků nezařadil. Dále byly upraveny poměry stran výchozích bounding boxů SSD na $0.5, 0.75, 1$, *viz sekce 2.1*, podle informací zobrazených v grafu 5.9. Při je-

[9]Dostupné na adrese: https://github.com/tensorflow/models/tree/master/research/object_detection

jich nastavování je nutné vzít v potaz transformaci dat z rozměru *1000x500* na *640x640* z důvodu fixní konfigurace vstupu detektoru.



Obrázek 5.9: Graf znázorňující informace o použité datové sadě ze Slovenských silnic. Podle těchto informací lze odhadnout a vyladit konfiguraci generování výchozích bounding boxů u modelu SSD [22]. Na prvním grafu lze vidět rozložení poměru šířky objektů k jejich výškám. Další 2 grafy ukazují rozložení jednotlivých rozměrů (výšky a šířky) v relativní velikosti vůči celkové velikosti snímků v daném rozměru.

5.2.2 Tvorba trénovací datové sady pro detektor

Datová sada, použita v této práci, obsahuje 6225 snímků a 246 různých tříd značek. Následně byla redukována pouze na dopravní značky, jejichž obsah se napříč jejími instancemi nemění, protože pouze s takovými značkami pracuje generativní model, jehož je detektor metrikou. Takto upravená datová sada čítá 1554 snímků s celkem 145 třídami dopravních značek a byla použita pro natrénování referenčního modelu detekujícího dopravní značky. Model SSD dosáhl na nerozšířené datové sadě referenční přesnosti **50.2 %** mAP pro $IoU >= 0.5$, *viz grafy 6.8*. Tato datová sada byla následně rozšiřována stejným počtem syntetických příkladů pro každou třídu značky pomocí generativních modelů. Se syntetickými a originálními datovými sadami a jejich kombinacemi bylo experimentováno v sekci 6.2.

Kapitola 6

Experimentování

V této kapitole se zabývám konfigurací a vyhodnocením experimentů s generativním modelem i detektorem. Ověřuji předpoklady z předchozí kapitoly 5 a snažím se najít nejvhodnější konfiguraci obou modelů.

6.1 Experimenty s generativním modelem

Za účelem vytvoření generativního modelu, schopného syntézy vizuálně kvalitních dat, bylo provedeno několik experimentů s různými architekturami diskriminátoru i generátoru a jejich různými konfiguracemi. Generativní model byl trénován po dávkách 64, s optimalizátorem Adam s parametry $\beta_1 = 0.5$ a $\beta_2 = 0.9$. U diskriminátoru je míra učení nastavena na 0.002 a u kritika na 0.001. Váhy všech konvolučních vrstev v generátoru i diskriminátoru jsou inicializovány náhodně z normálního rozložení se středem v bodě 0 a odchylkou 0.02. Během učení jsou párová data augmentována změnou kontrastu, afinní transformací a náhodným násobením a přičítáním k hodnotám jednotlivých pixelů. V rámci páru dat (vzor a cíl) jsou vždy augmentace zcela totožné, aby páry dat zůstaly konzistentní.

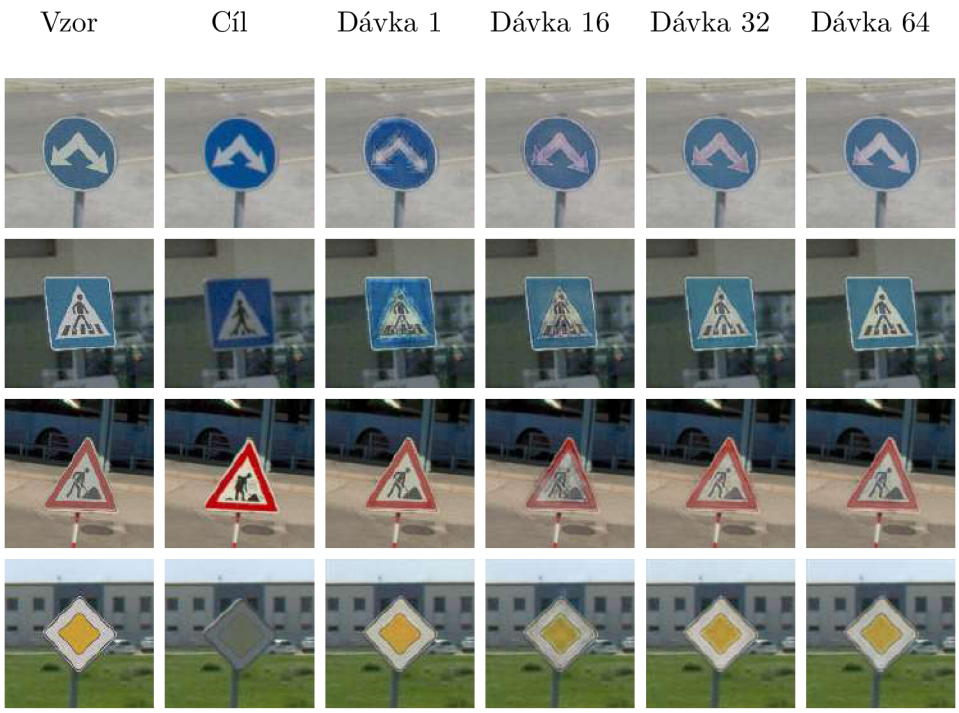
Úprava pozadí

Absolutního zamezení úpravy pozadí kombinací původního snímku a generovaných dat pouze z anotované oblasti značky není příliš efektivní [37]. Z důvodu nepřesných anotací je úplné překrytí značky obrázkem normy nemožné, *viz obrázek 5.4*, kde lze i při použití perspektivní transformace vidět části původní značky. Proto je nutné, aby generátor upravoval i bezprostřední okolí značky, zatímco vzdálenější okolí nechal netknuté. Výsledky experimentů ukázaly, že je generativní model schopen takové transformace, jsou-li mu poskytnuty během trénování vhodné páry dat, obsahující konzistentní příklady dané transformace, *viz obrázek 6.1*.

Velikost dávky dat během trénování

Autoři architektury *Pix2Pix* doporučují během trénování používat dávky snímků o velikosti 1 [16]. Použití takové dávky však vede ke zbytečné režii programu a proces trénování trvá násobně déle, oproti použití větší dávky. Proto jsem provedl sérii experimentů, *viz obrázek 6.1*, abych ověřil, jestli je možné získat kvalitní výstup generátoru při použití větší dávky. Všechny generativní modely v této kapitole byly trénovány po dobu 500 epoch, počet kroků se tedy odvíjí od velikosti použité dávky, *viz tabulka 6.2*. V těchto experimentech

byl použit diskriminátor PatchGAN 70x70, *viz 5.1.2* a generátor U-Net obsahující vrstvy *dropout* s parametrem 0.5, *viz sekce 5.1.3*. Příklady syntetických dat z experimentů jsou v tabulce 6.1.



Obrázek 6.1: Ukázka transformace vzorového snímku testovanými generátory. Výsledky experimentu ukazují, že použití dávky velikosti jedna generuje některé značky značně deformované. Použití větší dávky mírně zlepšuje vizuální kvalitu generovaných dat, avšak výstupy jsou podobnější vzoru, než-li cíli. Experiment ukázal, že je možné trénovat generativní model po větších dávkách, bez znatelných ztrát na kvalitě jeho výstupu.

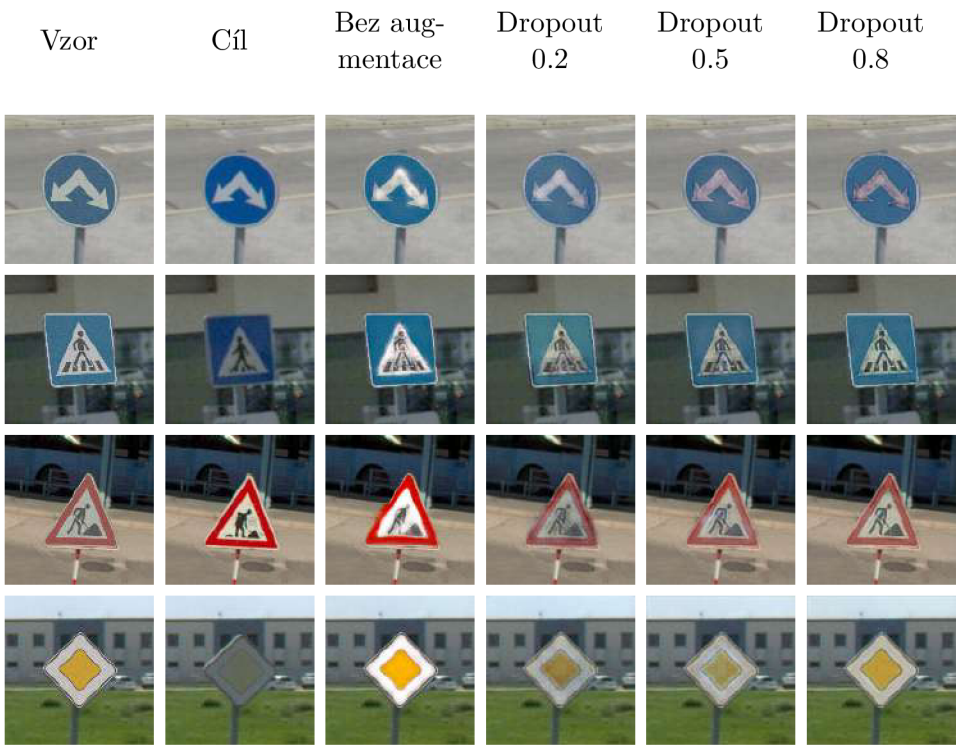
Velikost dávky	**Čas tréninku**	**Počet kroků**
1	13:49:38	770000
16	1:58:58	48500
32	1:37:12	24000
64	1:29:51	12000

Obrázek 6.2: Porovnání trénovacích časů a počtu potřebných kroků k dokončení 500 epoch při použití různých velikostí dávek.

Na základě výsledků tohoto experimentu jsem ve všech následujících experimentech použil během trénování dávky velikosti 64.

Variace generovaných dat

Během trénování GANů může dojít k několika poruchovým stavům, *viz sekce 3.1*, kdy generativní model produkuje například data pouze z omezeného pravděpodobnostního rozložení dat trénovacích. Avšak pro účely syntézy kvalitní datové sady musí být generativní model schopen generovat i rozdílně vypadající instance objektů stejných tříd. Určitou variaci ve výstupních datech lze zajistit vrstvou *dropout*, která má stejně vstupních i výstupních spojení a její parametr určuje pravděpodobnost, že spojení přes vrstvu *dropout* projde do následující vrstvy. Dalším zdrojem variability v generovaných datech je náhodné upravení normy značky před jejím vložením do výřezu, *viz sekce 5.1.1*, čímž vzniká variace nejen na výstupu generátoru ale také na jeho vstupu. Experimenty zaměřeny na efekt vrstvy *dropout*, stejně jako všechny následující experimenty, tyto úpravy norem zahrnují. Liší se pouze v hodnotě parametru této vrstvy. Experiment s daty vytvořenými bez úprav obrázků norem používá vrstvy dropout s parametrem 0.5. Příklady syntetických dat z experimentů jsou v tabulce 6.3.



Obrázek 6.3: Ukázka transformace vzorového snímku testovanými generátory. Experiment ukázal, že použití vrstev *dropout* v dekodéru U-Net s vhodným parametrem zlepšuje kvalitu výsledků. Vizuálně nejkvalitnější výsledky byly dosaženy použitím vrstvy *dropout* s parametrem 0.8. Generátor natrénován na datové sadě vytvořené bez upravení obrázků norem značek generuje značky rozmazané a jinak deformované, což implikuje zlepšení kvality výstupu použitím těchto úprav.

Na základě výsledků tohoto experimentu jsem ve všech následujících experimentech použil vrstvy *dropout* s parametrem 0.8.

Experimenty s různými architekturami diskriminátoru

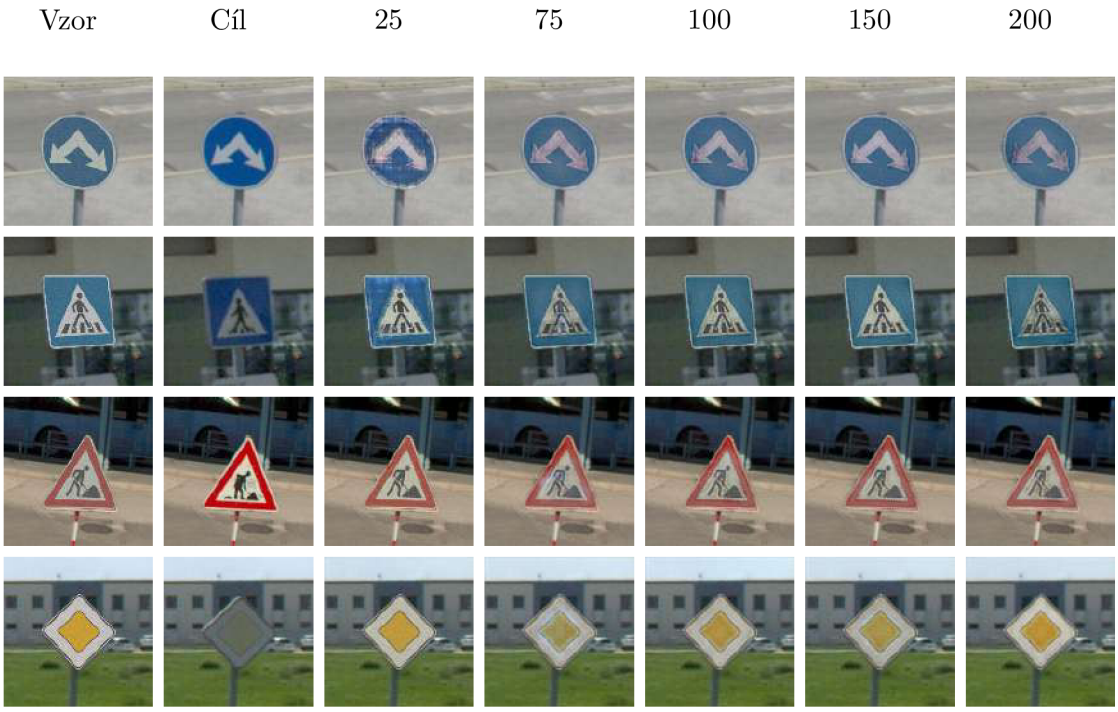
Kvalita generovaných dat lze zlepšit volbou jiného diskriminátoru [16, 2]. Diskriminátor typu *PatchGAN* hodnotí reálnost snímku maticí hodnot reprezentujících reálnost výřezů rozměrů *NxN* [16], *viz sekce 3.2*, a jeho speciální varianta *PixelGAN* hodnotí výřezy velikosti *1x1*. Velikost rozměru N může mít vliv na kvalitu generovaných dat, proto jsem provedl sérii experimentů s diskriminátory hodnotícími výřezy velikosti *70x70, 34x34, 1x1 (pixelGAN)*, *viz sekce 5.1.2*. Další dva experimenty byly provedeny s použitím WGAN, *viz sekce 3.1* a byly zaměřeny na vliv poměru úpravy vah kritika a generátoru na kvalitu generovaných dat. Příklady syntetických dat z experimentů jsou v tabulce 6.4.

Obrázek 6.4: Ukázka transformace vzorového snímku testovanými generátory. U sítě *PatchGAN* je kvalita generovaných dat poměrně konzistentní u všech použitých konfigurací, avšak konfigurace *34x34* mírně převyšuje kvalitou výstupu konfigurace ostatní. Další experimenty ukázaly, že poměr úprav vah kritika a generátoru má na kvalitu generovaných dat vliv. V konfiguraci s poměrem 3, ve prospěch kritika, je na výstupu generátoru zřetelný periodický šum, avšak v konfiguraci s poměrem 5 je šum zanedbatelný. Použitím WGAN lze dosáhnout reálněji vypadajících výstupů, avšak metoda PatchGAN má menší vliv na změnu pozadí snímku.

Na základě tohoto experimentu jsem ve všech následujících experimentech zvolil konfiguraci *34x34* pro PatchGAN a konfiguraci s úpravou vah kritika a generátoru v poměru 5:1 pro WGAN.

Poměr vah ztrátových funkcí během trénování generativního modelu

Dalším parametrem u sítě PatchGAN, který může mít vliv na kvalitu generovaných dat, je poměr ztrátových funkcí generátoru a diskriminátoru při upravování vah generátoru, *viz sekce* 3.2. Původní architektura Pix2Pix váží ztrátovou funkci generátoru 100 a diskriminátoru 1 [16] při řešení úkolů od sémantické segmentace scény po zabarvování černobílých obrázků. Transformace dopravních značek je však mnohem mírnějšího charakteru, a proto je pravděpodobné, že model bude dosahovat nejkvalitnějších výstupů s jinou konfigurací. Příklady syntetických dat z experimentů jsou v tabulce 6.5.



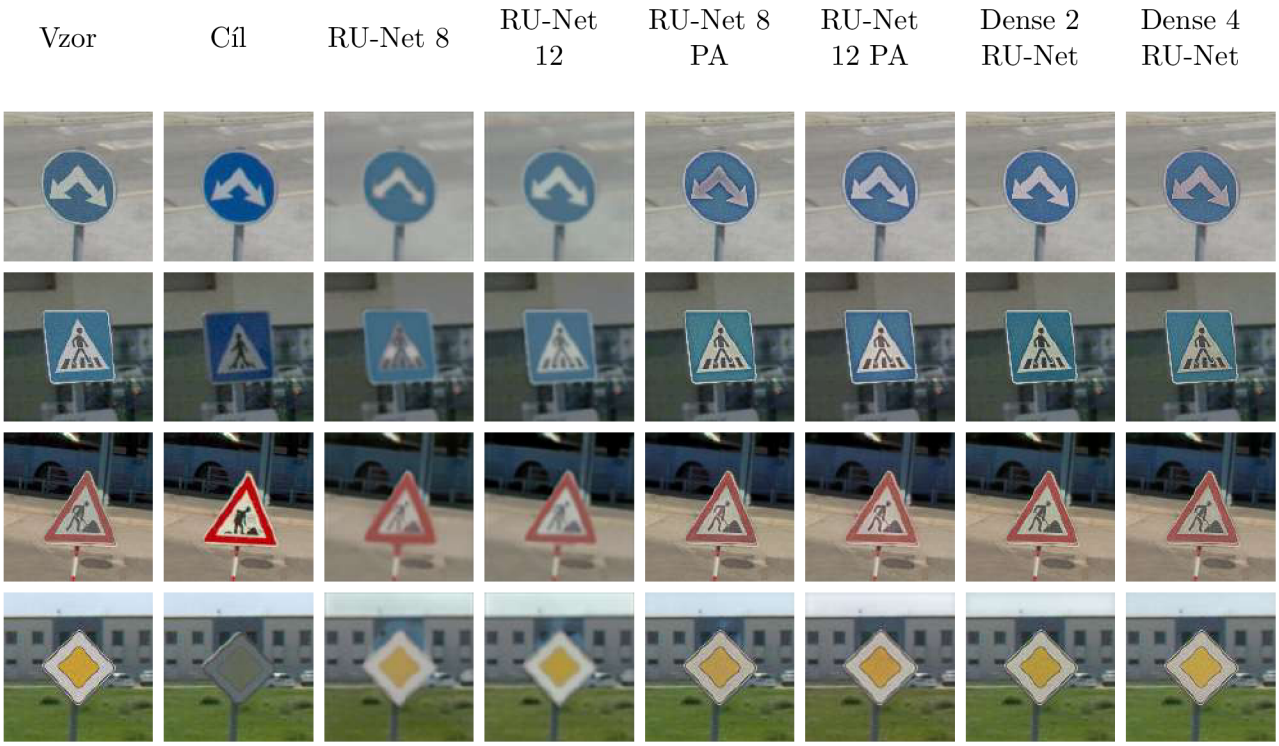
Obrázek 6.5: Ukázka transformace vzorového snímku testovanými generátory. Výsledky experimentů ukázaly, že tento parametr má relativně velký vliv na kvalitu výstupu generátoru. Vizuálně nejkvalitnějších výstupů generátor dosahuje v konfiguraci poměru ztrátových funkcí 150:1 a výstupů kvalitativně velice podobných dosahuje i v konfiguraci 100:1. Se snižující se váhou ztrátové funkce generátoru se model začne více zaměřovat na oklamání diskriminátoru na úkor kvality generovaných dat.

Na základě tohoto experimentu jsem ve všech následujících experimentech zvolil konfiguraci diskriminátoru PatchGAN, která upravuje váhy generátoru ze ztrátových funkcí generátoru a diskriminátoru v poměru 150:1.

Experimentování s reziduálními modely a PatchGAN

Dalším způsobem, jak zlepšit kvalitu syntetických dat, je použití kombinace reziduálních nebo DenseNet bloků se sítí U-Net, *viz sekce 5.1.3*. První čtyři experimenty byly provedeny se sítí RU-Net, z toho první dvě sítě obsahovaly původní reziduální bloky a další dvě *pre-activation* reziduální bloky (v tabulce označeno PA). Experimenty se stejnou architekturou reziduálních bloků se od sebe liší hloubkou použité sítě, kdy první sada experimentů používá RU-Net s 8 reziduálními bloky a druhá s 12 reziduálními bloky. Hlubší sítě by měly dosahovat lepších výsledků [13] a architektura reziduálního bloku *pre-activation* by měla generovat kvalitnější data oproti architektuře originální [14]. V dalších dvou experimentech jsem použil dvě sítě Dense RU-Net s 12 DenseNet bloky, které se liší v počtu sub bloků. Cílem těchto experimentů je zjistit vliv počtu sub bloků na kvalitu výstupu sítě.

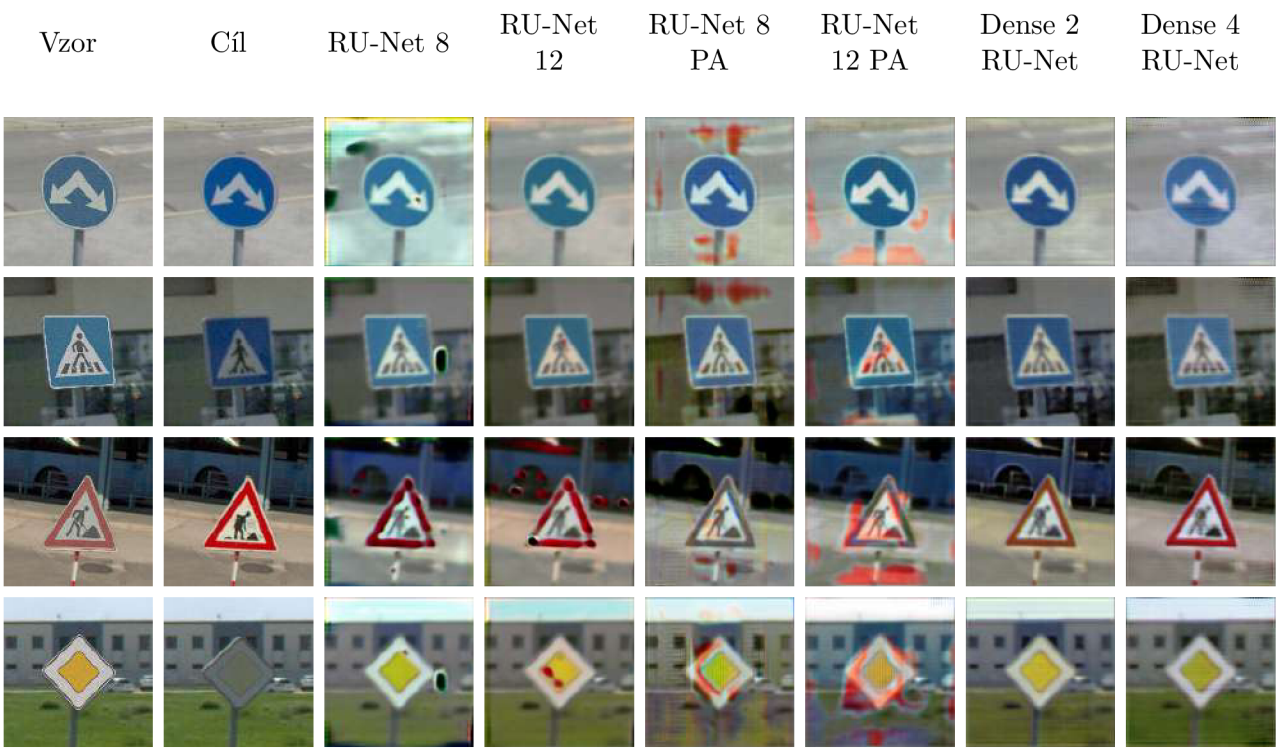
Všechny výše zmíněné generativní modely byly natrénovány pomocí sítí PatchGAN, *obrázek 6.6* a WGAN, *obrázek 6.7*, konfigurovaných podle výsledků předchozích experimentů.



Obrázek 6.6: Ukázka transformace vzorového snímku testovanými generátory. Výsledky experimentů ukazují, že hlubší sítě RU-Net dosahují lepších výsledků. Ačkoli klasická RU-Net generuje velice rozmazaná a nekvalitní data, použití reziduálního bloku *pre-activation* velice zvyšuje kvalitu syntetických dat. Nejkvalitnějšího výstupu však dosahují sítě Dense RU-Net. Porovnáním výstupů obou Dense RU-Net sítí zjistíme, že počet sub bloků nemá v tomto případě viditelný vliv na výslednou kvalitu snímků.

Experimentování s reziduálními modely a WGAN

Poslední sadu experimentů jsem provedl s totožnou konfigurací generátoru jako v experimentu předchozím. Jediným rozdílem je jejich natrénováni pomocí metody WGAN. Váhy kritika byly upravovány 5x častěji, než-li váhy generátoru.

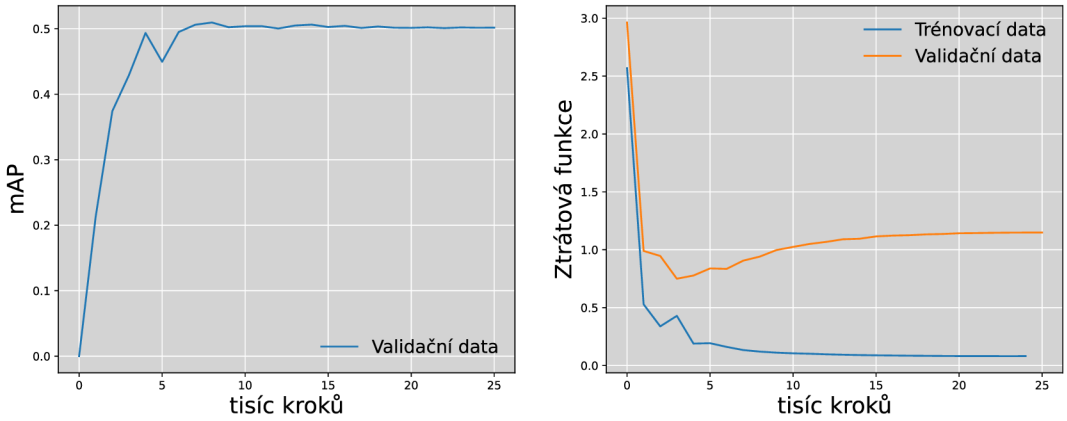


Obrázek 6.7: Ukázka transformace vzorového snímku testovanými generátory. Stejně, jako při použití PatchGAN, je z výsledků experimentů evidentní, že hloubka reziduální sítě má mírný vliv na kvalitu generovaných dat. V tomto případě však použití bloku *pre-activation* vede k mírnému zhoršení výsledků. Ačkoli generovaná data vypadají deformované a méně kvalitně, než při použití PatchGAN, výsledné snímky jsou mnohem podobnější těm cílovým. Dense RU-Net však dokáže generovat relativně realistické snímky oproti ostatním architekturám generátoru v tomto experimentu. Použitím 4 sub bloků místo 2 nedochází k viditelné změně kvality dat.

Ačkoli žádný z výše uvedených modelů neposkytuje dokonalou transformaci z normy na reálně vypadající dopravní značku, pro účely natrénování detektoru je možné data syntetizovat pomocí více generativních modelů a následně je kombinovat. Data syntetizovaná generátorem natrénovaným pomocí WGAN sice vypadají reálněji, avšak obsahují poměrně velké množství nechtěných transformací pozadí. Na druhou stranu, výstup generátorů natrénovaných pomocí PatchGAN, pozadí zachovává relativně netknuté, avšak výsledná značka je podobnější jejímu vzoru, než cíli.
Právě kombinací těchto dvou modelů a jejich charakteristik můžeme dosáhnout vzájemného doplnění těchto klíčových vlastností do syntetické datové sady a tím natrénovat přesnější model pro detekci objektů.

6.2 Experimenty s detektorem značek

Při všech experimentech s detektorem byl trénován na 25000 kroků, po dávkách 8 snímků. Detektor rozpoznává celkem 145 tříd z obrázků rozměru *640x640*. Prvních 1000 kroků je trénováno s nízkou mírou učení 0.025, ta je poté zvednuta na 0.08 a postupně klesá až k nule v kroku 25000. Spád optimalizátoru je nastaven na 0.9. První experiment byl zaměřen na zjištění referenční přesnosti detektoru natrénovaného na originálních datech. Referenční datová sada je blíže popsána v sekci 5.2. Skládá se z 1554 snímků, z nichž 300 je použito pro validaci modelu. Metrika použita při kvantifikaci přesnosti detektoru je mAP@.50IoU - mean Average Precision počítající pouze s detekcemi u kterých platí $IoU >= 0.5$ (dále jen mAP).



Obrázek 6.8: První graf zobrazuje vývoj *mAP* na validačních datech a druhý průběh kombinované ztrátové funkce detekce a klasifikace během trénování na validačních a trénovacích datech. Na rozdílu ztrátových funkcí je zřetelné mírné přetrénování sítě.

Experimentování s generátorem rozšířenými datovými sadami

Přesnost detektoru dosažena v předchozím experimentu je relativně nízká, protože datová sada obsahuje relativně velké množství tříd a malé množství dat. Některé třídy jsou v datové sadě obsaženy velice řídce. Systém navržen v sekci 5 je koncipován nejlépe právě pro takové případy, kdy se určitá třída vyskytuje v datové sadě zřídka nebo vůbec, protože syntetizuje objekty na základě jejich definované normy, čímž je eliminována potřeba dostatečného množství příkladů objektů každé třídy, pro její kvalitní syntézu. Pro rozšíření původní datové sady jsem zvolil dva generativním modely, které měly vizuálně nejkvalitnější výstup v experimentech v kapitole 6. Oba používají architekturu DenseNet, popsanou v sekci 3.2, s dvěma sub bloky. Síť obsahuje celkem 12 DenseNet bloků. Všechny aktivační vrstvy ReLu v generátoru jsou následovány vrstvou *dropout* s parametrem 0.8, *viz 6.3*. První model byl trénován pomocí diskriminátoru PatchGAN v konfiguraci 34x34, *viz obrázek 6.6* s váhou ztrátové funkce generátoru 150, *viz obrázek 6.5*. Druhý, architekturou generátoru totožný, model byl trénován skrze kritika aproximujícího Earth-Mover vzdálenost. Poměr trénování kritika a generátoru byl nastaven na 5. Pomocí každého z modelů byla vytvořena syntetická

datová sada, čítající 20 instancí značky každé třídy, tedy celkem 2900 anotovaných objektů. Informace o těchto datových sadách jsou v tabulce 6.1.

Původ	**Počet příkladů**
Originální	1554
WGAN	2900
PatchGAN	2900

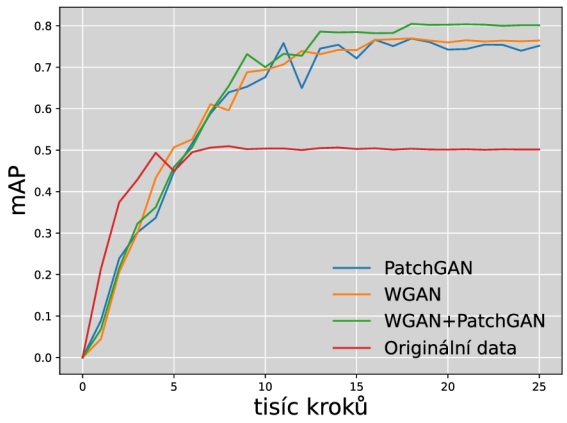
Tabulka 6.1: Všechny použité datové sady lze rozdělit na tyto 3 základní. Během experimentů byly detektory trénovány na těchto datových sadách a jejích kombinacích.

Experimentování se syntetickými datovými sadami

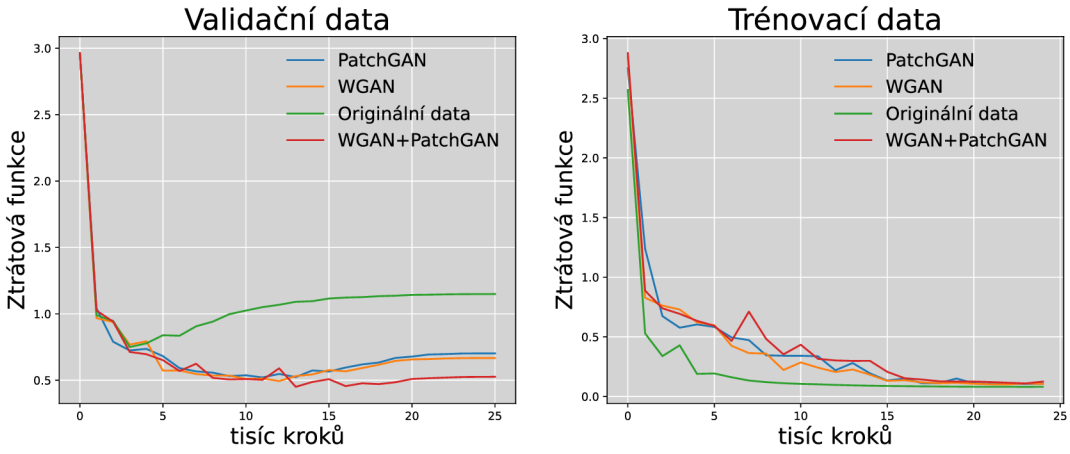
V rámci tohoto experimentu byly natrénovány 4 detektory na rozšířených datových sadách, *viz tabulka 6.2*. Cílem experimentu je ověření, zda použitím generativního modelu lze zlepšit přesnost detektoru dopravních značek. Průběh procesu trénování detektorů je zachycen v grafu 6.9 a 6.10.

Původ	**Počet příkladů**	**mAP@.50IoU**
Originální	1554	50.2 %
Originální+WGAN	4454	76.4 %
Originální+PatchGAN	4454	75.2 %
Originální+PatchGAN+WGAN	7354	**80.1 %**

Tabulka 6.2: Výsledky experimentu ukazují, že použití generativního modelu vede k přesnějším detekcím. Dále ukazují, že nejlepších výsledků lze dosáhnout kombinací více generativních modelů.



Obrázek 6.9: Graf zobrazuje vývoj *mAP* v průběhu trénování detektorů na všech datových sadách z tabulky 6.2. Razantní navýšení přesnosti detektoru lze pozorovat u všech rozšířených datových sad, kombinací obou generativních modelů lze dosáhnout, oproti použití jediného, jen relativně malého zlepšení.



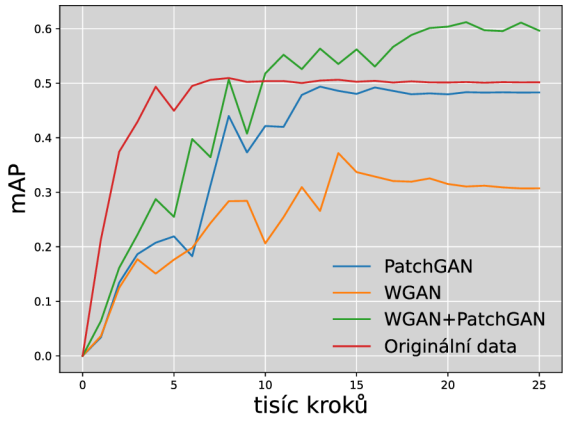
Obrázek 6.10: Graf zobrazuje vývoj ztrátové funkce v průběhu trénování detektorů na všech datových sadách z tabulky 6.2. Lze si povšimnout, že trénovací proces na všech rozšířených datových sadách konverguje pomaleji, avšak dosahuje menší míry přetrénování. Toto chování je typické během trénování modelů na datových sadách různých velikostí, což potvrzuje kvalitu syntetických dat.

Experimentování s plně syntetickými datovými sadami

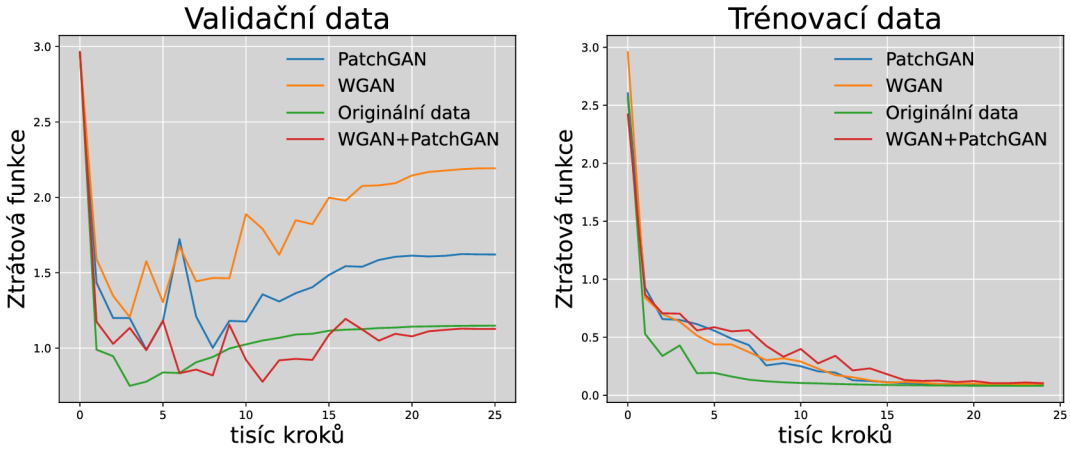
V kapitole 4.2 popisuji problémy s detekcí dopravních značek napříč různými zeměmi, kdy největším z nich jsou různé normy definující jejich vzhled. Systém implementován v této práci je navržen tak, aby dokázal generovat realisticky vypadající instance dopravních značek z obrázku normy vloženého do scény. Toho lze využít právě k syntéze datové sady pro detekci dopravních značek pro jakoukoli zemi.
Cílem tohoto experimentu je zjistit, jaké přesnosti dosáhne detektor natrénovaný pouze na syntetických datech, čímž lze ověřit schopnost systému generovat kvalitní datové sady, například pro detekci dopravních značek definovaných odlišnými normami. Průběh procesu trénování detektorů je zachycen v grafu 6.11 a 6.12.

Původ	**Počet příkladů**	**mAP@.50IoU**
Originální	1554	50.2 %
WGAN	4454	30.7 %
PatchGAN	4454	48.3 %
PatchGAN+WGAN	7354	**59.6 %**

Tabulka 6.3: Výsledky experimentu ukazují, že i detektor natrénován pouze na syntetických datech dosahuje relativně vysoké přesnosti. Ačkoli použití jednoho z generativních modelů nestačí pro vyrovnání přesnosti detektoru natrénovaného na originálních datech, kombinací obou modelů lze dosáhnout mAP o 9.4 % větší.



Obrázek 6.11: Graf zobrazuje vývoj mAP v průběhu trénování detektorů na všech datových sadách z tabulky 6.3

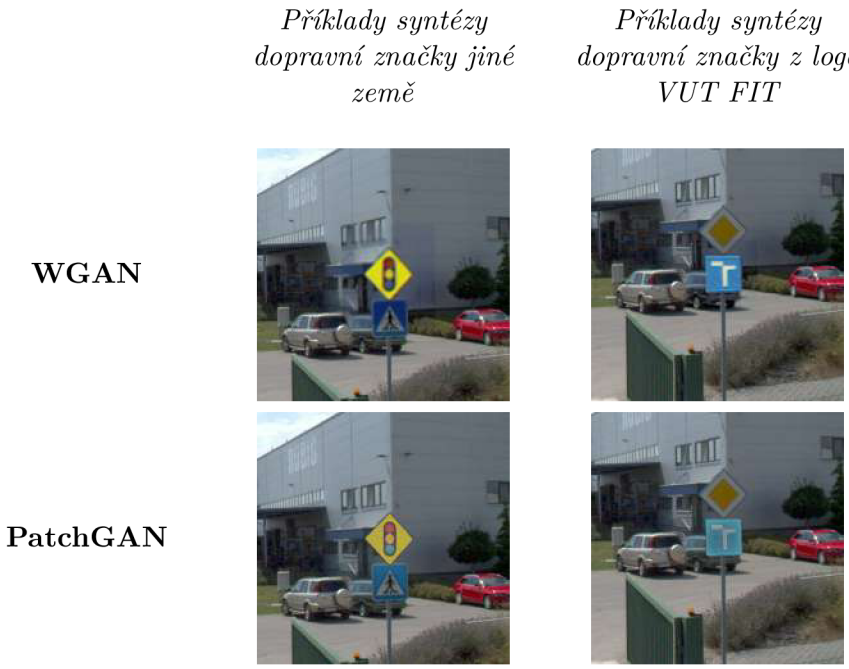


Obrázek 6.12: Grafy zobrazují vývoj ztrátových funkcí validačních a trénovacích dat v průběhu trénování detektorů na všech datových sadách z tabulky 6.3

Detektor natrénován na originálních datech konverguje mnohem rychleji oproti detektorům natrénovaných na datech syntetických. To může být způsobeno nerovnoměrným výběrem příkladů různých tříd během trénování na originálních datech, čímž se model naučí lépe a rychleji rozpoznávat často se vyskytující třídy značek. Kombinací obou generativních modelů dosáhl generátor nejlepšího výsledku, avšak i za použití pouze generátoru PatchGAN dosáhl diskriminátor skoro stejné přesnosti, jako při natrénování na originálních datech. Detektor natrénován pouze na datové sadě vytvořené pomocí WGAN dosáhl velice špatných výsledků, pravděpodobně kvůli velké úpravě pozadí značek, která nebyla během trénování kompenzována výběrem jiných příkladů. O tom vypovídá i rozdíl mezi ztrátovou funkcí na validačních a trénovacích datech, který pravděpodobně v tomto případě není způsoben přetrénováním modelu, nýbrž větším rozdílem mezi těmito daty.

6.3 Ostatní experimenty

Systém popsaný v sekci 5.1.3 transformuje obrázky objektů vložených do výřezů reálných scén na reálně vypadající instance tohoto objektu v původní scéně na stejném místě. Trénovací datová sada ze Slovenských silnic pro generativní model, *viz sekce 5.1.1*, byla vytvořena s cílem obrázek vložený do scény transformovat na reálně vypadající objekt, a zároveň zachovat jeho obsah dobře rozlišitelný. To vede k možnosti funkcionalitu modelu využít i pro třídy, které nebyly součástí trénovacího procesu. Tohoto principu lze využít pro rozšíření datové sady nejen o další instance objektů tříd, které se v původní datové sadě nacházejí, ale také objekty, které v ní nebyly zachyceny. Dalším možným využitím je tvorba datové sady, například jiné země, která používá odlišné vzory dopravních značek, protože experiment potvrdil, že je možné natrénovat relativně přesný detektor i na zcela syntetické datové sadě, *viz obrázek 6.11*.



Obrázek 6.13: Ukázka rozšířeného použití navrhovaného modelu. Vstupem generativního modelu je obrázek, proto máme poměrně velkou kontrolu nad generovanými daty a jejich obsahem. Toho lze využít pro tvorbu syntetické datové sady, obsahující dopravní značky libovolného vzhledu.

Kapitola 7

Závěr

Cílem této práce bylo navrhnout systém, založený na generativních neuronových sítích GAN, schopný rozšířit existující datovou sadu pro detekci dopravních značek o kvalitní syntetické snímky obsahující objekty i zřídka se vyskytujících tříd. Na začátku práce jsou popsány moderní metody rozpoznávání objektů, založené na hlubokých konvolučních neuronových sítích, jako SSD nebo Faster R-CNN, a také metody kvantifikace jejich přesnosti.

Dále popisuji generativní metody založeny na soutěžení dvou neuronových sítí (GANy). Popisuji zde několik architektur GANů, například DCGAN nebo Conditional GAN, popisuji různé postupy jejich trénování a doporučenou konfiguraci pro jejich konvergenci. Dále se věnuji možnostem transformace obrazu modelem Pix2Pix a kombinacím reziduálních neuronových sítí se sítí U-Net. Dále se zabývám potížemi při detekci dopravních značek a možnostmi, jak tyto potíže použitím generativních modelů co nejlépe eliminovat.

Dále jsou popsány všechny architektury použitých neuronových sítí v generativním modelu i detektoru a jsou zmíněny jejich výhody a nevýhody. Navrhuji zde také několik možností jejich konfigurace za účelem zlepšení kvality syntetických snímků a zpřesnění detekce, které jsou v další kapitole prověřeny a jsou buďto zakomponovány do modelu, nebo zahozeny. Také je popsána charakteristika trénovacích datových sad pro oba modely a proces jejich tvorby.

Během experimentování s generativním modelem jsou jeho výstupy průběžně vyhodnocovány a laděny, nakonec jsou vybrány dva modely generující nejkvalitnější snímky (PatchGAN a WGAN), které jsou použity pro syntézu umělé datové sady. Následně bylo natrénováno několik detektorů na originálních, syntetických i kombinovaných datových sadách a následně byla jejich přesnost hodnocena metrikou mAP (.50 IoU), za účelem validace generativního modelu. Při použití pouze syntetických dat k trénování modelu detektoru dosáhl mAP 59.6 %, to je o 9.4 % více, než detektor natrénovaný na datech původních. Při použití kombinace originálních i syntetických dat detektor dosahuje 80.1 % mAP, což je o 29.9 % více.

Při dalším vývoji by se mohlo zaměřit na návrh jediného generativního modelu, který bude kombinovat silné stránky PatchGAN, tj. zachování pozadí objektu během transformace i WGAN, tj. generování velice realisticky vypadajících objektů. Takový model by dále vylepšil přesnost detektorů a byl by schopen vytvořit velice realistické datové sady. Bylo by také velice zajímavé systém rozšířit o možnost generování normy značky vložením textu a piktogramů, čímž by bylo možné syntetizovat i komplexnější, specifické dopravní značky.

Literatura

[1] Arcos García Álvaro, García, J. A. Álvarez a Soria Morillo, L. M. Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing*. 2018, sv. 316, s. 332–344. DOI: https://doi.org/10.1016/j.neucom.2018.08.009. ISSN 0925-2312. Dostupné z: https://www.sciencedirect.com/science/article/pii/S092523121830924X.

[2] Arjovsky, M., Chintala, S. a Bottou, L. *Wasserstein GAN*. 2017.

[3] Bansal, M., Krizhevsky, A. a Ogale, A. *ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst*. 2018.

[4] Dai, J., Li, Y., He, K. a Sun, J. *R-FCN: Object Detection via Region-based Fully Convolutional Networks*. 2016.

[5] Dalal, N. a Triggs, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, sv. 1, s. 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[6] de la Escalera, A., Moreno, L. E., Salichs, M. A. a Armingol, J. M. Road traffic sign detection and classification. *IEEE Transactions on Industrial Electronics*. 1997, sv. 44, č. 6, s. 848–859. DOI: 10.1109/41.649946.

[7] Efros, A. A. a Freeman, W. T. Image Quilting for Texture Synthesis and Transfer. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 2001, s. 341–346. SIGGRAPH '01. DOI: 10.1145/383259.383296. ISBN 158113374X. Dostupné z: https://doi.org/10.1145/383259.383296.

[8] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. a Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*. Jun 2010, sv. 88, č. 2, s. 303–338. DOI: 10.1007/s11263-009-0275-4. ISSN 1573-1405. Dostupné z: https://doi.org/10.1007/s11263-009-0275-4.

[9] Girshick, R. *Fast R-CNN*. 2015.

[10] Girshick, R., Donahue, J., Darrell, T. a Malik, J. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014.

[11] Goodfellow, I. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017.

[12] Goodfellow, I. J., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. *Generative Adversarial Networks*. 2014.

[13] He, K., Zhang, X., Ren, S. a Sun, J. *Deep Residual Learning for Image Recognition*. 2015.

[14] He, K., Zhang, X., Ren, S. a Sun, J. *Identity Mappings in Deep Residual Networks*. 2016.

[15] Huang, G., Liu, Z., Maaten, L. van der a Weinberger, K. Q. *Densely Connected Convolutional Networks*. 2018.

[16] Isola, P., Zhu, J.-Y., Zhou, T. a Efros, A. A. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018.

[17] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*. Květen 2012.

[18] Krizhevsky, A., Sutskever, I. a Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. Leden 2012, sv. 25. DOI: 10.1145/3065386.

[19] Lee, H., Kim, J., Kim, E. K. a Kim, S. Wasserstein Generative Adversarial Networks Based Data Augmentation for Radar Data Analysis. *Applied Sciences*. 2020, sv. 10, č. 4. DOI: 10.3390/app10041449. ISSN 2076-3417. Dostupné z: https://www.mdpi.com/2076-3417/10/4/1449.

[20] Lienhart, R. a Maydt, J. An extended set of Haar-like features for rapid object detection. In: *Proceedings. International Conference on Image Processing*. 2002, sv. 1, s. I–I. DOI: 10.1109/ICIP.2002.1038171.

[21] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R. et al. *Microsoft COCO: Common Objects in Context*. 2015.

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. et al. SSD: Single Shot MultiBox Detector. In: Leibe, B., Matas, J., Sebe, N. a Welling, M., ed. *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, 2016, s. 21–37. ISBN 978-3-319-46448-0.

[23] Lowe, D. Object Recognition from Local Scale-Invariant Features. *Proceedings of the IEEE International Conference on Computer Vision*. Leden 2001, sv. 2.

[24] Mirza, M. a Osindero, S. *Conditional Generative Adversarial Nets*. 2014.

[25] Padilla, R., Netto, S. L. a da Silva, E. A. B. A Survey on Performance Metrics for Object-Detection Algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, s. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[26] Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L. a Silva, E. A. B. da. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*. 2021, sv. 10, č. 3. DOI: 10.3390/electronics10030279. ISSN 2079-9292. Dostupné z: https://www.mdpi.com/2079-9292/10/3/279.

[27] Radford, A., Metz, L. a Chintala, S. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016.

[28] Redmon, J., Divvala, S., Girshick, R. a Farhadi, A. *You Only Look Once: Unified, Real-Time Object Detection*. 2016.

[29] Ren, S., He, K., Girshick, R. a Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017, sv. 39, č. 6, s. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[30] Ronneberger, O., Fischer, P. a Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[31] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*. 2015, sv. 115, č. 3, s. 211–252. DOI: 10.1007/s11263-015-0816-y.

[32] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S. et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015.

[33] Sanchez, J. a Perronnin, F. High-dimensional signature compression for large-scale image classification. In: *CVPR 2011*. 2011, s. 1665–1672. DOI: 10.1109/CVPR.2011.5995504.

[34] Simona. *Opel Insignia will feature 'Opel Eye' camera*. TopSpeed, 2008. Dostupné z: https://www.topspeed.com/cars/car-news/opel-insignia-will-feature-opel-eye-camera-ar59270.html.

[35] Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U. a Sutton, C. *VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning*. 2017.

[36] Uijlings, J., Sande, K. van de, Gevers, T. a Smeulders, A. Selective Search for Object Recognition. *International Journal of Computer Vision*. 2013. DOI: 10.1007/s11263-013-0620-5. Dostupné z: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[37] Ševčík, P. *Příprava trénovacích dat pomocí generativních neuronových sítí*. Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/20903/.

[38] Wakabayashi, D. Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam. *The New York Times*. New York: [b.n.]. 2018. ISSN 0362-4331.

[39] Wang, S.-F., Yu, W.-K. a Li, Y.-X. Multi-Wavelet Residual Dense Convolutional Neural Network for Image Denoising. *IEEE Access*. 2020, sv. 8, s. 214413–214424. DOI: 10.1109/ACCESS.2020.3040542.

[40] Yang, Z., Chai, Y., Anguelov, D., Zhou, Y., Sun, P. et al. *SurfelGAN: Synthesizing Realistic Sensor Data for Autonomous Driving*. 2020.

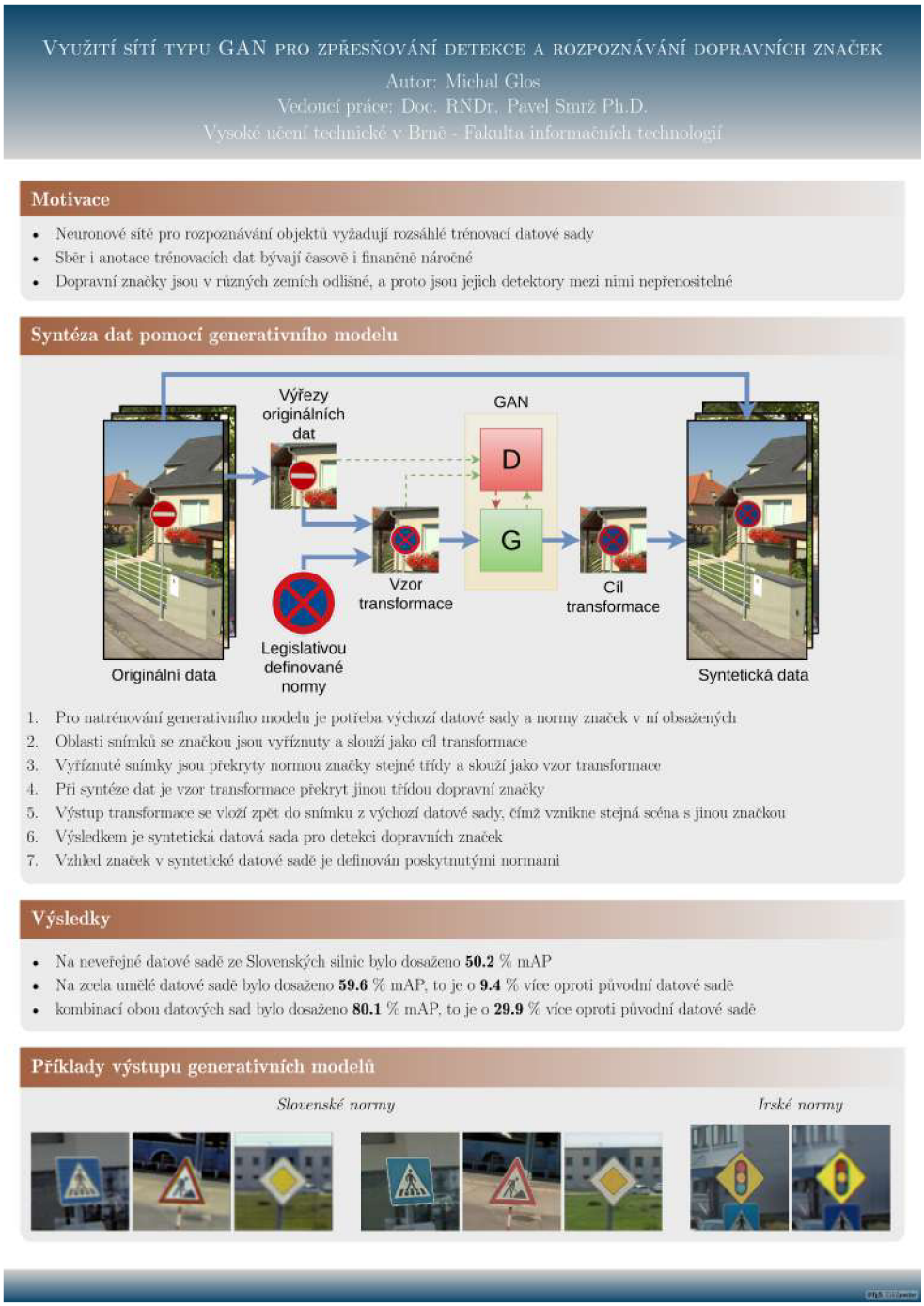
[41] Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T. et al. *LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop*. 2016.

[42] Zhang, Z., Liu, Q. a Wang, Y. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*. Institute of Electrical and Electronics Engineers (IEEE). May 2018, sv. 15, č. 5, s. 749–753. DOI: 10.1109/lgrs.2018.2802944. ISSN 1558-0571. Dostupné z: http://dx.doi.org/10.1109/LGRS.2018.2802944.

Seznam příloh

Příloha A

Plakát



Příloha B

Obsah přiloženého paměťového média

- */Detektor* - Soubory týkající se modelu pro detekci dopravních značek
- */GAN* - Soubory týkající se generativního modelu
- */Latex* - Soubory s pdf práce a plakátu včetně zdrojových kódů
- */venv* - Soubory s konfiguracemi virtuálních prostředí Python3
- */README.md* - Soubor obsahující návod na instalaci a spuštění

Podrobnější informace ke struktuře adresáře jsou obsaženy v souboru *README.md*