VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV RADIOELEKTRONIKY
DEPARTMENT OF RADIO ELECTRONICS

DETEKCE DOPRAVNÍCH ZNAČEK V REÁLNÉM ČASE
TRAFFIC SIGN DETECTION IN REAL TIME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Marek Sicha**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Bravenec**
ADVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Elektronika a komunikační technologie**

Ústav radioelektroniky

Student: Marek Sicha ***ID:*** 211317
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Detekce dopravních značek v reálném čase

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možné přístupy k zpracování obrazových signálů, především metody zaměřené na detekci a klasifikaci dopravních značek jak ve statických obrazech, tak ve videosekvencích. Srovnejte možnosti existujících knihoven pro zpracování obrazu a počítačové vidění v jazyce Python nebo C++ a navrhněte vlastní řešení pro detekci a klasifikaci dopravních značek. Při návrhu multiplatformní aplikace uvažujte použití v automobilu, tudíž dbejte i na rychlost detektoru při použití v jednodeskovém počítači jako je např. Raspberry Pi.

Proveďte detailní testování navrženého systému. Otestujte přesnost detekce na statických snímcích a na video sekvencích s obtížně detekovatelnými značkami v různých úhlech a otestujte vliv osvětlení na správnou klasifikaci. Vyhodnoťte kvalitu detekce vašeho systému, zaměřte se na rychlost i úspěšnost detekce. K navrženému kódu vytvořte dokumentaci a zveřejněte jej na GitHubu či GitLabu.

DOPORUČENÁ LITERATURA:

[1] OpenCV [online]. 2018 [cit. 2020-05-12]. Dostupné z: http://opencv.org/

[2] Understanding the Technology behind Traffic Sign Recognition (TSR) Systems. [cit. 2020-05-12]. Dostupné z: https://www.design-reuse.com/articles/41154/traffic-sign-recognition-tsr-system.html

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 27.5.2021

Vedoucí práce: Ing. Tomáš Bravenec

prof. Ing. Tomáš Kratochvíl, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zabývá detekcí a klasifikací dopravních značek v obrazech a videosekvencích. Cílem práce je i možnost provádět detekci a klasifikaci na jednodeskovém počítači. Pro řešení problému byly vybrány neuronové sítě a programovací jazyk Python. Detekce a klasifikace objektu je řešena zvlášť, tudíž byly použity dvě neuronové sítě. Pro klasifikaci byla zvolena konvoluční neuronová síť a pro detekci byl zvolen detektor z rodiny EfficientDet. Celková architektura byla testována na jednodeskovém počítači Nvidia Jetson Nano.

KLÍČOVÁ SLOVA

Detekce, Klasifikace, Neuronové sítě, Python, Konvoluční neuronové sítě, Dopravní značky, EfficientDet

ABSTRACT

The bachelor's thesis focuses on the detection and classification of traffic signs in images and video sequences. The goal of the work is also the possibility to perform detection and classification on a single board computer. Neural networks and the Python programming language were chosen to solve the problem. Object detection and classification are solved separately, so two neural networks were used. A convolutional neural network was chosen for classification and a detector from the EfficientDet family was chosen for detection. The overall architecture was tested on a single board Nvidia Jetson Nano computer.

KEYWORDS

Detection, Classification, Neural networks, Python, Convolution neural networks, Traffic signs, EfficientDet

SICHA, Marek. *Detekce dopravních značek v reálném čase*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2021, 62 s. Bakalářská práce. Vedoucí práce: Ing. Tomáš Bravenec

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Marek Sicha
VUT ID autora:	211317
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Detekce dopravních značek v reálném čase

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Tomáši Bravencovi za jeho vstřícnost, čas, poskytnuté rady při psaní bakalářské práce a také za jeho odborné vedení práce.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Tato práce se zabývá detekcí a následným rozpoznáváním dopravních značek v obrazech a videosekvencích. Detekce a rozpoznávání dopravních značek je jeden z hlavních předpokladů pro vytvoření autonomního řízení automobilů. Samozřejmě detekce a rozpoznávání dopravních značek není jenom čistě pro autonomní řízení aut. Může být použito i jako jeden za základních asistenčních prvků pro dopravní prostředky, ve kterých je stále ještě řidič člověk. Tento asistenční prvek by mohl upozornit řidiče např. na překročení maximální rychlosti nebo přehlédnutí zákazové značky a včas zabránit dopravní nehodě.

Cílem této práce je prostudovat možné přístupy ke zpracování obrazů, se zaměřením na detekci a klasifikaci dopravních značek ve statických obrazech a videosekvencích. Navrhnout řešení pro klasifikaci a detekci dopravních značek a při návrhu řešení uvažovat i možnost použití v dopravním prostředku na jednodeskovém počítači.

Pro vytvoření softwaru byl zvolen programovací jazyk Python, který obsahuje nespočet knihoven pro úpravu vstupních datových sad. Datové sady jsou nedílnou součástí pro vytvoření neuronové sítě klasifikátoru i detektoru. Vytvoření klasifikátoru i detektoru bylo za pomocí neuronových sítí, které jsou v dnešní době stále populárnější a to díky stále rostoucímu výpočetnímu výkonu. Klasifikátor dopravních značek byl vytvořen pomocí konvoluční neuronové sítě a pro vytvoření detektoru dopravních značek byla zvolena rodina detektorů EfficientDet. Následně proběhla implementace klasifikátoru do architektury detektoru a celková struktura byla testována na jednodeskovém počítači Nvidia Jetson Nano.

1 DOPRAVNÍ ZNAČENÍ

Tato kapitola se zabývá dopravním značením v České republice, pojednává o možnostech dělení dopravních značek. Také je zde zmíněná struktura datové sady, typy datových sad a také jsou uvedeny dostupné datové sady s krátkým popisem.

1.1 Dopravní značení v České republice

Dnešní podoba dopravního značení na území České republiky prošla v minulosti řadou změn. Ať už přibývali nové značky, nebo se měnila podoba těch stávajících. Podoba, pod kterou ho známe dnes, platí od ledna roku 2001. Od té doby se ještě dopravní značení rozšířilo o řadu nových značek. V České republice se dopravní značení řídí zákonem č. 361/2000 Sb. o pravidlech provozu na pozemních komunikacích. [1]

1.2 Dělení dopravního značení v ČR

Dopravní značky můžeme rozdělit podle způsobu umístění na svislé a vodorovné. Vodorovné dopravní značky jsou vyznačeny na pozemní komunikaci. Svislé dopravní značky jsou všechny dopravní značky kromě značek vodorovných.

1.2.1 Svislé dopravní značení

Svislé dopravní značky se většinou umisťují při pravém okraji komunikace nebo nad ní. Pokud je svislá dopravní značka umístěna současně na pravé i levé straně vozovky, tak je to pro její zdůraznění. I zde platí výjimky a za určitých okolností mohou být umístěny i jinde, např. na chodník. Příklad svislých dopravních značek upravujících přednost je uveden na obrázku 1.1.

Svislé dopravní značení lze rozdělit do následujících kategorií [2]:

- **Výstražné dopravní značky**: Upozorňují na blížící se rizika, jejich nejčastější podoba je trojúhelníkový podklad s červeným pruhem okolo a uprostřed obrázek představující dané riziko.

- **Značky upravující přednost**: Již podle názvu je jasné, že jejich nejčastější výskyt je na křižovatkách, kde určují přednost účastníků dopravního provozu. Kategorie je specifická tím, že obsahuje všechny tvary i barvy značek.

- **Zákazové dopravní značky**: Jejich provedení je kruh s červeným pruhem okolo a uprostřed je obrázek představující daný zákaz, nebo maximální dovolenou rychlost.

- **Příkazové dopravní značky**: Dalo by se říct, že se jedná o opak značek zákazových. Provedení příkazových značek je kruh s modrým podkladem, na podklad je bílou barvou nakreslený příkaz.

- **Informativní dopravní značky**: Poskytují účastníkům provozu různé informace.

- **Dodatkové tabulky**: Rozšiřují význam značek, s nimiž jsou použity.

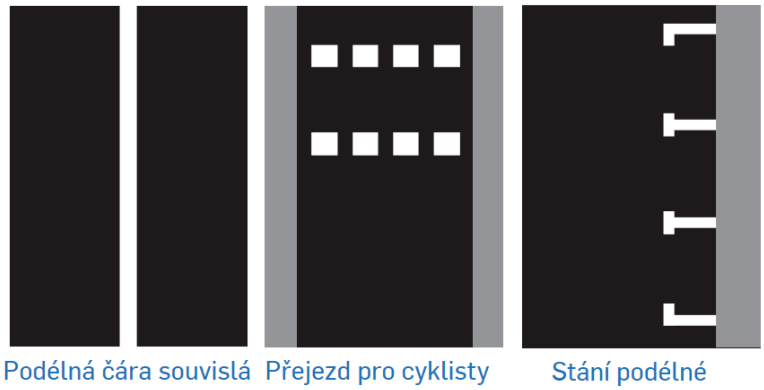
Obr. 1.1: Značky upravující přednost [1]

1.2.2 Vodorovné dopravní značky

Vodorovné dopravní značky jsou vyznačeny na pozemních komunikacích. Můžeme je rozdělit na přechodné a stálé. Stálé jsou označeny většinou bílou barvou a označují např. část vozovky, kde se nesmí předjíždět. Zatímco přechodné jsou označeny oranžovou nebo žlutou barvou a jsou nadřazeny stálým dopravním značkám. [2] Příklad vodorovných dopravních značek je uveden na obrázku 1.2.

1.2.3 Dělení podle vzhledu

Dopravní značení v České republice můžeme rozdělit podle barev a podle tvaru. Nejdominantnější barva je červená, jako další barvy se používají modrá, černá, oranžová, lze uvažovat i bílou, ale ta je většinou použita jako podklad.



Obr. 1.2: Vodorovné dopravní značky [1]

Dělení podle tvaru je podobné tomu podle barev. Mezi tvary se vyskytují kruh, čtverec, trojúhelník a osmiúhelník.

Dopravní značení není ve všech zemích stejné, dokonce není jednotné ani pro Evropskou unii. Mezi hlavní rozdíly patří barevné provedení, text napsaný v místním jazyce, grafické provedení nebo nějaké značky se v dané zemi nepoužívají vůbec. Příklad dopravního značení pro různé země EU je uveden na obrázku 1.3. [3]

Obr. 1.3: Značka zákaz vjezdu motorových vozidel v různých zemích EU [3]

1.3 Datové sady

Datové sady jsou neodmyslitelnou součástí každé neuronové sítě. Bez potřebné datové sady by vůbec nebylo možné neuronovou síť trénovat. Datové sady dopravních značek můžeme rozdělit na dvě skupiny, první skupina obsahuje pouze výřezy dopravních značek, což je vhodné pro klasifikátor, kde značka zabírá více jak 80% celého snímku. Požadavky na anotace takové datové sady jsou jednoduché a jediné

[1] Převzato z: `https://www.bezpecnecesty.cz/cz/autoskola/dopravni-znacky/vodorovne-dopravni-znacky`

co potřebujeme je, aby každý typ dopravních značek měl svoji vlastní třídu. Druhá skupina datových sad pracuje s celými snímky, tedy na obrázku se může nacházet více dopravních značek, tyhle datové sady se používají pro trénování detektorů. Požadavky na anotace této skupiny datových sad jsou již náročnější a často jsou odlišné pro různé architektury detektorů. Ale všechny mají společný základ, pro všechny dopravní značky, které chceme aby náš detektor uměl rozpoznávat, musí existovat anotace, která obsahuje pozici značky na obrázku, obvykle souřadnice x_1, y_1, x_2, y_2, dále o jakou třídu se jedná a pro jaký název obrázku tahle anotace platí. Další požadavky pro detektor je již možné poté doplnit dle předchozích uvedených požadavků.

Při výběru datové sady nehraje roli jen počet snímků datové sady, ale také její obsah. Je důležité, aby datová sada pokryla všechny možnosti, ve kterých bude neuronová síť pracovat. Pro dopravní značky je důležitá různorodost prostředí, různé počasí, značky za různých světelných podmínek, při různé viditelnosti nebo pro různé deformace.

Datová sada bývá většinou označena 4-5 písmeny např. GTSRB (The German Traffic Sign Recognition Benchmark). První písmeno označuje stát pro který vznikla, další písmena označují o jako datovou sadu se jedná, v tomhle případě se jedná o dopravní značky a poslední písmena jestli o datovou sadu pro klasifikaci nebo detekci, nebo jen o datovou sadu obecně. Pro hledání volně dostupných datových sad existuje stránka kaggle[2].

1.3.1 Dostupné datové sady

- **The German Traffic Sign Recognition Benchmark - GTSRB**[3] jedná se o německou datovou sadu, která obsahuje přes 30 tisíc výřezů dopravních značek ve 43 různých třídách.

- **The German Traffic Sign Detection Benchmark - GTSDB**[2] jedná se o německou datovou sadu, která obsahuje 900 celých snímků pořízených z palubních kamer, datová sada je rozdělaná na trénovací množinu, která obsahuje 600 trénovacích snímků a 300 ověřovacích snímků. Obsahuje 741 anotovaných snímků s 1213 dopravními značkami za různých světelných podmínek.

- **BelgiumTS Dataset**[4] jedná se o belgickou datovou sadu, která obsahuje snímky z 8 různých kamer. Obsahuje přes 15 tisíc snímků pro 62 různých tříd, mimo běžné značky obsahuje i různé informativní a doplňkové tabulky.

[2] `https://www.kaggle.com/datasets`
[3] `https://benchmark.ini.rub.de/`
[4] `https://btsd.ethz.ch/shareddata/`

- **Russian Traffic Sign Dataset - RTSD**[5] je ruská nejrozsáhlejší datová sada dopravních značek, obsahuje přes 170 tisíc anotovaných vzorků ve 100 tisících snímcích pro 156 různých tříd.

- **LISA Traffic Sign Dataset**[6] jedná se o americkou datovou sadu, která obsahuje přes 6600 snímků dopravních značek pro 47 různých tříd. Obsahuje kombinaci jak v barevném, tak i v provedení ve stupních šedi.

- **Mapillary Traffic Sign Dataset**[7] jedná se o datovou sadu, která kombinuje snímky dopravních značek z 6 kontinentů. Obsahuje přes 300 tříd ve více než 100 tisících obrázků ve vysoké kvalitě, za různých podmínek světelných podmínek, v různých ročních obdobích nebo při různých vlivech počasí. Ze 100 tisíc snímků je více než polovina anotována.

[5] `https://graphics.cs.msu.ru/en/research/projects/rtsd`
[6] `https://git-disl.github.io/GTDLBench/datasets/lisa_traffic_sign_dataset/`
[7] `https://www.mapillary.com/dataset/trafficsign`

2 JEDNODESKOVÝ POČÍTAČ

Je malý počítač, který je sestaven na jedné desce plošného spoje. Jeho hlavní výhodou oproti stolnímu počítači je jeho malá velikost a integrace systému na jedné desce plošného spoje. Jednodeskový počítač je možné rozšířit o různé vstupně/výstupní moduly. Díky tomu jsou dnes hojně využívány v elektronickém průmyslu, v lékařských přístrojích, automobilech a v dalších zařízeních.

Jako hlavní procesory se obvykle používají ARM nebo x86. ARM procesory využívají redukovanou instrukční sadu (RISC), zatímco x86 procesory komplexní instrukční sadu (CISC). CISC klade důraz na hardware, zatímco RISC na software. RISC provádí instrukce v jedné periodě hodinového cyklu, zatímco CISC může využívat více period. RISC bude potřebovat více paměti RAM, ale CISC klade důraz na menší velikost programovacího kódu. [23]

Jednodeskové počítače mají od 1 do 8 GB RAM a jako náhradu pevného disku několik GB paměti flash. Obsahují porty USB, grafický výstup, připojení k síti RJ-45 a další vstupně/výstupní porty. K jednodeskovému počítači můžeme připojit periferie jako jsou myš, klávesnice nebo monitor, což umožňuje ovládání jako stolní počítač. Jednodeskový počítat je také možné ovládat přes vzdálenou plochu pomocí služeb VNC (Virtual Network Computing) nebo SSH (Secure Shell), popřípadě mají Wi-Fi nebo Bluetooth. Jako operační systém slouží některá z distribucí systému Linux. Může se jednat o různé odlehčené verze, které disponují jen grafickým, nebo textovým rozhraním.

2.1 Jetson Nano

Je malý, výkonný počítač určený pro vývoj umělé inteligence. Jetson Nano disponuje funkcemi a výkonem, který je potřeba pro provozování moderních neuronových sítí. Je vhodný pro klasifikaci obrázků, detekci objektů, segmentaci nebo zpracování řeči.

Specifikace [24]:

- **GPU**: Architektura NVIDIA Maxwell se 128 jádry NVIDIA CUDA
- **CPU**: Čtyřjádrový procesor ARM Cortex-A57 MPCore
- **Paměť**: 4 GB 64bitová LPDDR4, 1600 MHz 25,6 GB / s
- **Úložiště**: 16 GB eMMC 5.1
- **Port pro kameru**: 3x4 nebo 4x2 MIPI CSI-2 D-PHY 1,1 (1,5 Gb / s na pár)
- **Kódování videa**: 250MP/sec, 1x 4K @ 30 (HEVC), 2x 1080p @ 60 (HEVC), 4x 1080p @ 30 (HEVC), 4x 720p @ 60 (HEVC), 9x 720p @ 30 (HEVC)
- **Dekódování videa**: 500MP/sec 1x 4K @ 60 (HEVC), 2x 4K @ 30 (HEVC), 4x 1080p @ 60 (HEVC), 8x 1080p @ 30 (HEVC), 9x 720p @ 60 (HEVC)

- **Display**: HDMI 2.0 a eDP 1.4
- **USB**: 4x USB 3.0, USB 2.0 Micro-B
- **Rozměry**: 69.6 mm x 45 mm
- **Ostatní**: GPIO, I^2C , I^2S, SPI, UART

3 METODY DETEKCE DOPRAVNÍCH ZNAČEK V OBRAZE

Detekce dopravních značek je v dnešní době velice důležité téma ať už pro vytvoření autonomního auta, tak i pro obyčejného řidiče. V posledních letech doprava výrazně zhoustla. Jako řidič musíte neustále sledovat okolní účastníky provozu a předvídat jejich chování a během toho ještě vnímat různé značky, kterých ve městech není zrovna málo. Takže se často stává, že nějakou přehlédnete. Auta dnes mají různé asistenční prvky, tak proč mezi ně nezařadit i systém pro rozpoznávání dopravních značek. Dopravní značka by se dala přímo promítat na palubní systém a v případě potřeby by se řidič mohl přesvědčit např. o maximální povolené rychlosti.

3.1 Detekce pomocí barvy

Detekce podle barvy je založena na detekci objektů se stejnou nebo podobnou barvou odlišnou od pozadí obrázku. Jak již bylo zmíněno, dopravní značky mají své specifické barvy. Existuje několik typů barevných modelů, jsou vybrány dva nejznámější RGB model a HSV model.

3.1.1 RGB Model

Je využíván v LCD monitorech a další video-elektronice. Obsahuje tři složky R - red (červená), G - green (zelená) a B - blue (modrá), kdy každý bod obrazu je složen z intenzit těchto tří složek. Další barvy je možné získat různým mícháním těchto složek. Nevýhodou RGB modelu je jeho citlivost na světlo.

Na obrázku 3.1 můžeme vidět příklad segmentace pomocí červené barvy v RGB modelu, kdy jakýkoliv odstín červené barvy je zachován a označen bíle, zatímco zbytek barev je vyznačen černě.

Prahová hodnota barvy tohoto modelu lze podle prací [4] a [6] vyjádřit pomocí vztahů 3.1 a 3.2.

$$g(x,y) = k_1 \begin{cases} R_a \leq f_r(x,y) \leq R_b \\ TG_a \leq \frac{f_g(x,y)}{f_r(x,y)} \leq TG_b \\ TB_a \leq \frac{f_g(x,y)}{f_b(x,y)} \leq TB_b \end{cases} \tag{3.1}$$

$$g(x,y) = k_2 \quad \text{v ostatních případech} \tag{3.2}$$

Kde $g(x, y)$ je hodnota šedé barvy zpracovaného obrazu, $fr(x, y)$, $fg(x, y)$ a $fb(x, y)$, jsou funkce udávající hodnotu červené, zelené a modré složky v každém bodě obrázku.

Obr. 3.1: Segmentace podle červené barvy v RGB modelu [1]

3.1.2 HSV Model

Taktéž obsahuje tři složky jako model RGB, ale tyto složky nesou odlišné informace. Složky jsou hue (odstín), saturation (sytost) a value (hodnota).

Složka hue představuje barvu nebo její odstín, podle které můžeme snadno rozpoznat jednu barvu od druhé. Odstín je v rozmezí od 0^o do 360^o, např. 0^o nebo 360^o představuje barvu červenou. Složka saturation říká, kolik bílé barvy je smícháno s odstínem. A poslední složka value reprezentuje jas barvy.

Tento model se používá vzhledem k tomu, že barevná složka je jednoduše oddělena od jasové složky a nemůže tedy dojít k ovlivnění barvy jasem, využívá se např. k odstranění stínů.

Pro převod z RGB modelu do HSV se podle práce [5] využívají následující rovnice 3.3, 3.4 a 3.5.

$$H = \arccos \frac{\frac{1}{2}(2R - G - B)}{\sqrt{(R - G)^2 - (R - B)(G - B)}} \tag{3.3}$$

$$S = \frac{max(R, G, B) - min(R, G, B)}{max(R, G, B)} \tag{3.4}$$

[1] Převzato z: `https://www.researchgate.net/figure/An-example-of-red-color-segmentation-for-a-traffic-sign-image_fig2_236645674`

$$V = max(R, G, B) \tag{3.5}$$

Na obrázku 3.2 je uvedena segmentace podle červené barvy v prostředí HSV, kdy červená barva je zachována a vyznačena bíle, zatímco zbytek barev je vyznačen černě.

Obr. 3.2: Segmentace podle červené barvy v HSV modelu [7]

3.2 Detekce pomocí hrany

Dopravní značky nejen v České republice mají specifické tvary např. kruh, trojúhelník atd. V přírodě se nevyskytuje velké množství tvarů podobných dopravním značkám, a proto detektory hran jsou jednou z možností, jak je identifikovat.

3.2.1 Cannyho hranový detektor

John Canny vytvořil sice tento detektor již v roce 1986, ale i dnes je stále populární v počítačovém zpracování obrazu. Definoval tři základní výkonnostní kritéria, která musí být dodržena, a na jejich základě poté detektor hran navrhl. [9]

1. **Kvalitní detekce.** Nízká pravděpodobnost neoznačení existující hrany a také nízká pravděpodobnost označení hrany neexistující.

2. **Dobrá lokalizace.** Všechny body, kterými jsou označeny hrany, by měli být co nejblíže středu skutečné hrany.

3. **Detekce jedné hrany pouze jednou.** Rozvíjí první kritérium, pokud je jedna hrana zachycena vícekrát, pouze jedna z nich je pravdivá, a proto je označená jako správná. Ostatní hrany jsou považovány jako neexistující.

Nevýhodou je, že původní Cannyho detektor hran pracuje pouze s obrázky ve stupních šedi, takže už při převodu se může ztratit užitečná informace. Dnešní vylepšené verze už jsou schopny pracovat s obrázky v mnoha barevných modelech, jako jsou RGB a HVS. Udává se, že původní Cannyho detektor pracující s obrázky ve stupních šedi zachytí 90 % všech hran z původního obrázku [8]. Nicméně obecný postup Cannyho detektoru je následující: [9]

- **Vyhlazení vstupního obrazu.** Pro redukci šumu se používá Gaussův filtr.

- **Výpočet gradientu.** Pro výpočet velikosti M a úhlu Φ gradientu se používají následující vztahy 3.6 a 3.7.

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{3.6}$$

$$\Phi(x, y) = \arctan\left(\frac{g_y}{g_x}\right) \tag{3.7}$$

  Kde g_x je rozdíl mezi intenzitou pixelů na východ a západ od středového pixelu a podobně g_y je rozdíl v intenzitě pixelů na jih a sever od středového pixelu.

- **Non-maxima suppression (potlačení nemaximálních hodnot).** V tomto kroku odstraníme hranové body, jejichž velikost M je velmi malá. V prvním kroku najdeme směr hrany pomocí úhlu Φ získaného podle vztahu 3.7. Poté porovnáme velikosti M hranového pixelu s jeho sousedníma v určitém směru. Je-li hodnota aspoň u jednoho menší v daném směru, pak je potlačen, v opačném případě je zachován.

- **Odstranění bezvýznamných hranových bodů.** Abychom se zbavili lokálních maxim vytvořených šumem, použijeme dvou-prahové schéma. Máme dvě prahové hodnoty T_L (nízká) a T_H (vysoká). Pokud je hodnota gradientu daného pixelu větší než T_H, je zachován. Je-li hodnota mezi T_L a T_H, tak záleží na předchozím pixelu, pokud je zachován, tak je i tento pixel zachován. V

ostatních případech je pixel potlačen.

Obr. 3.3: Detekce hran pomocí Cannyho detektoru [2]

Na obrázku 3.3 jsou uvedeny příklady zpracování obrázků pomocí Cannyho detektoru hran.

3.2.2 Viola - Jones detektor

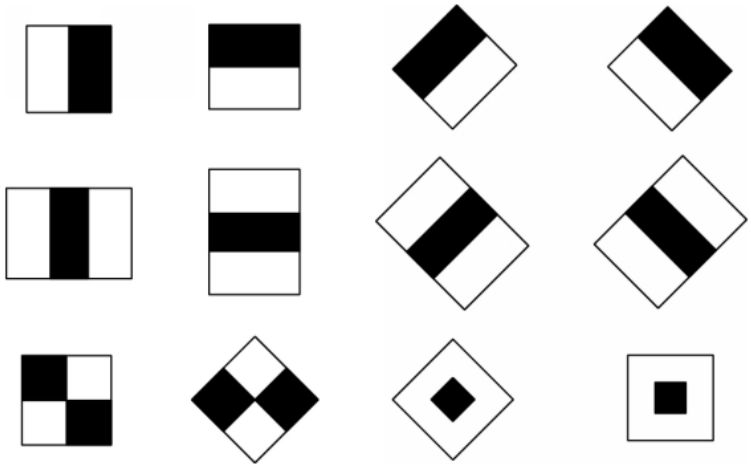
Metoda byla zprvu vyvinuta za účelem detekce obličeje. Nicméně metodu je možné využít pro detekci různých objektů. Detektor funguje tak, že se po obrázku posunuje detekční okýnko. Detekční algoritmus rozhoduje v každé pozici detekčního okýnka, jestli se zde nachází požadovaný objekt či nikoli. Počet klasifikací pro daný obrázek je řádově 10^5. Detektor je využíván kvůli rychlosti detekce a schopnosti provádět detekci v reálném čase, to znamená, že rozhodnutí musejí být co nejrychlejší.

K trénování se využívá klasifikační algoritmus AdaBoost, algoritmus kombinuje slabé jednoduché klasifikátory založeny na Haarových příznacích a vytvoří klasifikátor s libovolnou přesností. Rozhodování, jestli jsme našli požadovaný objekt, se provádí na základě hlasování jednoduchých klasifikátorů podle jejich důležitosti. Nejprve provádějí detekci nejdůležitější klasifikátory, pokud je jejich detekce negativní, objekt je zamítnut a ostatní už neprovádějí detekci. Objekt je přijat jen v případě, kdy je přijat kaskádou po sobě jdoucích klasifikátorů. [10]

[2] Převzato z: `https://www.semanticscholar.org`

Haarovy příznaky

Haarovy příznaky se využívají pro vytvoření klasifikátoru v metodě Viola-Jones detektor. Jsou tvořeny bílými a černými obdélníky, jak je vidět na obrázku 3.4. Pro extrakci příznaku z obrazu je nutné sečíst intenzitu všech pixelů v jednotlivých obdélnících a poté spočítat rozdíl mezi bílými a černými obdélníky.



Obr. 3.4: Haarovy příznaky [3]

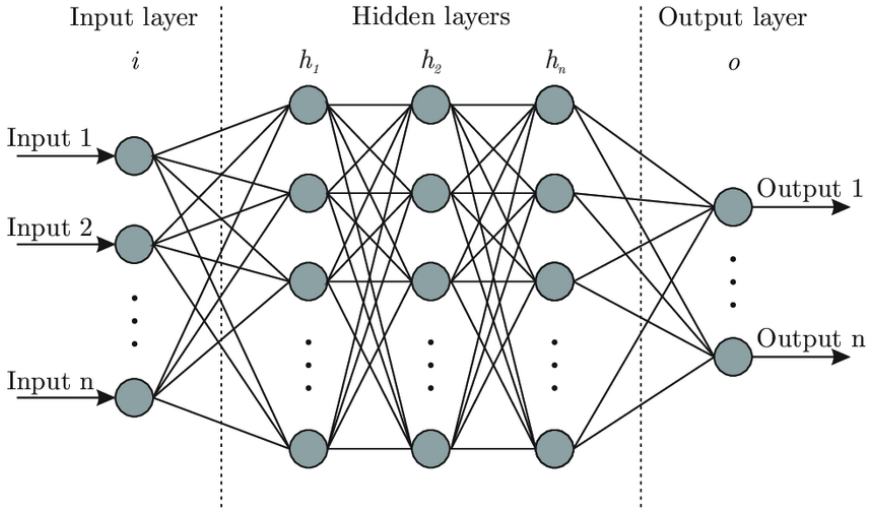
[3]Převzato z `https://www.hindawi.com/journals/tswj/2014/753860/fig1/`

4 NEURONOVÉ SÍTĚ

Tato kapitola se věnuje problematice neuronových sítí. Je zde vysvětleno co jsou neuronové sítě, jak se dají učit a jaké jsou typy. Taktéž tato kapitola pojednává o architekturách založených na neuronových sítích se zaměřením na detekci objektů. Část kapitoly je věnována způsobům hodnocení detektorů objektů.

4.1 Umělé neuronové sítě

Umělá neuronová síť je výpočetní model, který je v podstatě inspirován biologickými nervovými systémy, především lidským mozkem. Základním prvkem umělých neuronových sítí, tak jako lidského mozku, je neuron. V porovnání lidský mozek obsahuje asi 50 až 100 miliard neuronů, zatímco umělá neuronová síť obsahuje stovky až tisíce těchto neuronů. Umělé neuronové sítě jsou tvořeny posloupnosti po sobě jdoucích vrstev, kdy první vrstva se nazývá vstupní vrstva, poslední vrstva je výstupní vrstva a zbylé vrstvy se nazývají skryté vrstvy. Všechny neurony každé vrstvy jsou propojeny s neurony následující a předchozí vrstvy, viz obrázek 4.1. Propojené neurony si předávají signály, které mají určitou váhu. Signály jsou vypočítány za pomocí přenosových funkcí a váha takového signálu se přizpůsobuje v průběhu učení. Do každého neuronu může vstupovat libovolný počet těchto signálů, ale neuron má pouze jeden výstupní signál, který ovšem může vstupovat do libovolného počtu neuronů v následující vrstvě. [11] [12]



Obr. 4.1: Struktura umělé neuronové sítě [1]

[1]Převzato z: https://mc.ai/my-notes-on-neural-networks-2/

Neuronová síť dostane do první vrstvy vstupní data, obvykle ve formě vícerozměrného vektoru, ty následně pošle do skrytých vrstev, které poté rozhodují na základě vah a rozhodnutí předchozích vrstev, jestli je změna užitečná, či nikoli pro daný výstup. Celý tento proces je označován jako učení. Proces, který probíhá mezi skrytými vrstvami, se označuje deep learning.

Aby umělá neuronová síť, stejně jako mozek, podávala nejpřesnější výsledky, je potřeba ji učit. Neboli nastavovat hodnoty vah a prahů přenosových funkcí, aby odpovídali požadovaným výstupům. Jsou uvedeny nejpoužívanější metody, jak neuronové sítě učit: [11]

- **Metoda učení s učitelem.** Neuronová síť obsahuje základní nastavení vah a prahových hodnot. Neuronová síť dostane vstupní data, podle výstupních dat sítě a očekávaných dat, určí chybu a provede korekci. Nově nastaví hodnoty prahů a vah. Celý tento cyklus opakuje do chvíle, dokud nenajde takovou konfiguraci prahových hodnot a vah, která odpovídá naší stanovené chybě.

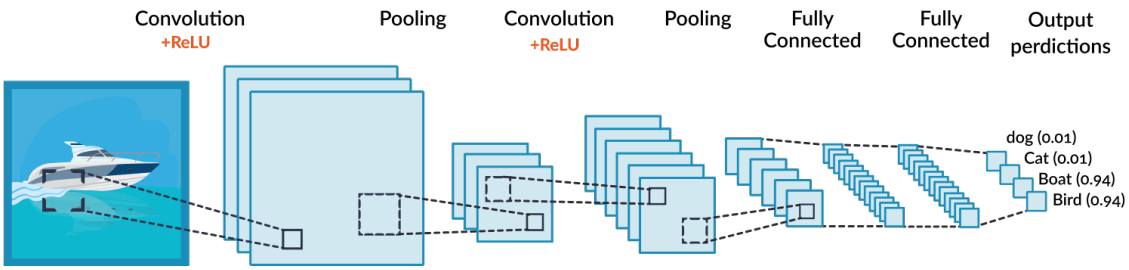
- **Metoda učení bez učitele.** Rozdíl s předchozí metodou je ten, že síť nezná předpokládaný výstup, tudíž nemá podle čeho provést korekci. Tyto sítě dostávají vstupní data a mají volnost, podle čeho je třídí.

Většina neuronových sítí, které pracují s detekcí objektu. Klasifikují výstup pomocí metody učení s učitelem.

4.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě se převážně používají pro rozpoznávání objektů v obrazech. Navíc oproti umělým neuronovým sítím obsahují ještě konvoluční a sdružovací vrstvy.

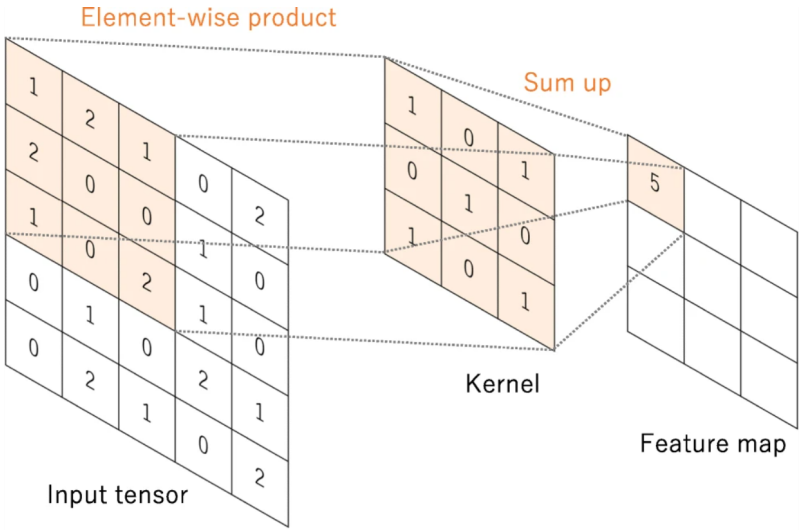
Celkově se konvoluční neuronová síť skládá ze tří typů vrstev. První vrstva se jmenuje vstupní vrstva, poté konvoluční vrstvy, následují sdružovací vrstvy, jako u umělých neuronových sítí plně propojené vrstvy a poslední je výstupní vrstva. Konvoluční a sdružovací typy vrstev se mohou několikrát po sobě opakovat, jak je vidět na obrázku 4.2.



Obr. 4.2: Architektura konvoluční neuronové sítě [2]

4.2.1 Konvoluční vrstva

Jak již název napovídá, jedná se o nejdůležitější vrstvu celé sítě. Vrstva provádí extrakci příznaků ze vstupní vrstvy. Extrakce příznaků se provádí pomocí konvolučních filtrů. Ve své podstatě se jedná jen o matice s malým počtem čísel. Tyto matice se přiloží na vstupní matici, která se nazývá tenzor. V místě překryvu obou matic se hodnoty vynásobí a poté se provede součet všech hodnot, celý tento proces je obyčejná konvoluce. Výsledkem je příznaková mapa, viz obrázek 4.3.



Obr. 4.3: Příznaková mapa [12]

Konvoluční vrstvy umožňují výrazně zmenšit složitost modelu optimalizací jeho výstupu. Slouží k tomu tři hyper-parametry, depth (hloubka), stride (krok) a padding (výplň). Hloubku lze nastavit pomocí počtu neuronů uvnitř vrstvy. Snížení hloubky má za následek výrazné snížení celkového počtu neuronů v síti, ale také

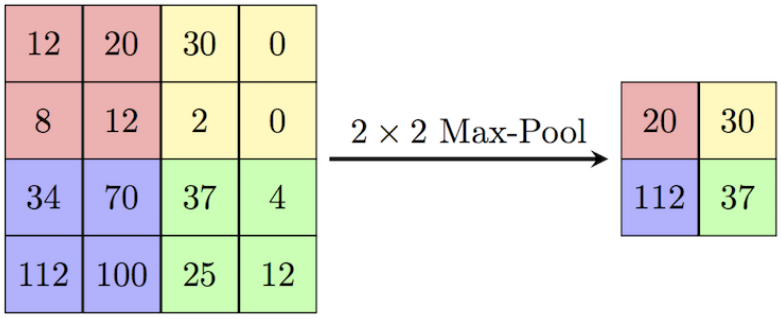
[2] Převzato z: `https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-tutorial-basic-advanced/`

to výrazně sníží schopnost sítě rozpoznávat objekty. Krok nám určuje, o kolik se posune filtr vůči své předešlé poloze. Pomocí výplně můžeme kontrolovat velikost příznakové mapy. Rozlišuje se valid a same. Pokud použije valid padding znamená to že nepoužijeme výplň, pokud je použito same padding, znamená to, že byla použita výplň a podle potřeby byly přidány řádky nebo sloupce, popřípadě obojí obsahující nuly.

Další výhodou oproti umělým neuronovým sítím je sdílení vah. Váhy všech filtrů jsou sdíleny se všemi neurony v konkrétní příznakové mapě. Na rozdíl od umělých neuronových sítí jsou neurony propojeny pouze s určitou vrstvou vstupního obrazu, to výrazně snižuje počet parametrů v síti a zefektivňuje to proces. Proces tréningu konvoluční neuronové sítě s ohledem na konvoluční vrstvu je najít filtry, které nejlépe splňují naše požadavky na základě dané datové sady. Váhy filtrů jsou jediné parametry, které se naučí během procesu učení. Ostatní parametry, jako jsou velikost filtrů, jejich počet, výplň, krok a hloubka, je nutné nastavit před zahájením procesu učení. [11] [12]

4.2.2 Sdružovací vrstva

Tato vrstva je použita hned po konvoluční vrstvě. Má za cíl jediný úkol, a to snížit množství parametrů v sítí a zefektivnit tak výpočetní proces. [12] Provádí operaci downsampling nebo sub-sampling, tzv. převzorkování. Sdružovací vrstva pracuje s výstupem konvoluční vrstvy, s příznakovou mapou.



Obr. 4.4: Příklad sdružení s filtrem o velikosti 2×2 a funkcí maximum [3]

Jak již bylo uvedeno, má za úkol snížit počet parametrů, tedy redukuje velikost příznakové mapy. Podle rozměru filtru, který se předem nastaví, rozdělí příznakovou mapu do bloků, jak je uvedeno na obrázku 4.4. Z těchto bloku je poté vybraná příslušná hodnota podle sdružovací funkce, mezi nejčastěji používané funkce patří

[3]Převzato z: `https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_max.html`

součet, maximum nebo průměr. Na obrázku 4.4 je příznaková mapa o rozměrech 4×4 s použitým filtrem 2×2 a funkcí maximum. Výstup je poté mapa příznaků o rozměrech 2×2.

4.2.3 Plně propojená vrstva

Plně propojená vstupní vrstva, má za úkol z výstupu konvoluční a sdružovací vrstvy udělat jedno-dimenzionální matici čísel nebo vektor. Poté následuje jedna, nebo i více plně propojených vrstev. Tyto vrstvy jsou stejné jako u umělých neuronových sítí, vstup každého neuronu je propojen s výstupy neuronů z předchozí vrstvy s určitou váhou. Za každou plně propojenou funkcí následuje nelineární aktivační funkce. Výstupní vrstva obsahuje počet uzlů, který odpovídá stejnému počtu tříd pro klasifikaci, s určitou váhou pravděpodobnosti. Za poslední vrstvou následuje speciální aktivační funkce, která se vybírá podle zadaného úkolu, pro klasifikační úlohu je to převážně funkce softmax[4] 4.1, která normuje hodnotu výstupní vrstvy k pravděpodobnostem cílové třídy, pravděpodobnosti se poté pohybují v rozmezí od 0 do 1 a jejich celkový součet je 1. [12]

$$f(x_i) = \frac{e^{x_i}}{\sum_{n}^{j=1} e^{x_j}} \tag{4.1}$$

Kde x_i je nenormovaná hodnota výstupní vrstvy pro každou třídu j, suma ve jmenovateli je součet všech nenormovaných hodnot všech tříd.

4.2.4 Nelineární aktivační funkce

Slouží k zanesení nelinearity, jelikož výstupy z konvoluční a sdružovací vrstvy jsou lineární. Existuje nespočetné množství nelineárních funkcí, ale mezi nejpoužívanější nelineární funkce paří sigmoid, hyperbolický tangens a nebo ReLU. Průběhy různých typů nelineárních aktivačních funkcí, které jsou popsány níže, jsou zobrazeny na obrázku 4.5.

- **Sigmoid** se používá kvůli hlavnímu důvodu a to, že transformuje vstupní hodnoty na výstupní hodnoty v rozmezí od 0 do 1. Díky tomu je vhodný použít pro modely, kde chceme predikovat pravděpodobnost pro výstup. Nevýhoda této funkce je její malý rozsah, problém nastane, když se hodnoty na ose X začnou blížit k téměř horizontální části křivky. To může způsobit zaseknutí

[4]Tvar funkce převzat z: `https://deepai.org/machine-learning-glossary-and-terms/softmax-layer`

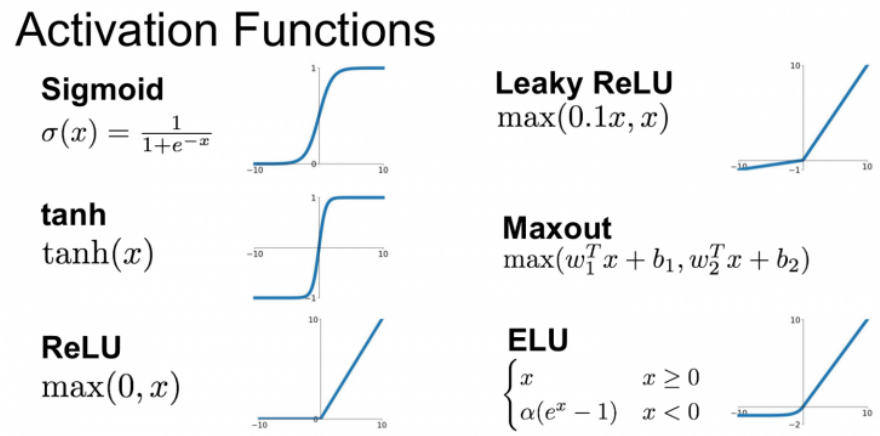
neuronové sítě v době tréninku, neboli síť se již dále neučí.[21] [22]

- **Hyperbolický tangens** je velmi podobná funkci Sigmoid, také je náchylná na problém zaseknutí neuronové sítě v průběhu trénování. Výhodou je, že je symetrická kolem počátku, čili má rozsah výstupních hodnot od -1 do 1.

- **Rectified linear Unit - ReLU** je v současnosti nejpopulárnější aktivační funkce, je to hlavně díky tomu, že neuvažuje záporné hodnoty a tedy neurony s nulovou hodnotou neaktivuje. Je tedy početně méně náročná než předchozí zmíněné aktivační funkce a provádí pouze porovnání hodnot, což dokazuje její předpis $max(0, x)$. Její nevýhodou je, že jakmile gradient dosáhne hodnoty 0 ReLU "umře", to znamená, že neurony které dosáhly tohoto stavu neupravují dále své váhy a část sítě se stane pasivní. Jakmile se jednou neuron dostane do tohoto stavu, je jen malá šance, že se vrátí zpět. [21]

- **Leaky ReLU** je vylepšená verze ReLU, která uvažuje i záporné hodnoty a transformuje je na hodnoty blížící se k nule, díky tomu se vyvaruje stavu mrtvých neuronů. Předpis funkce je ve tvaru $max(ax, x)$, kde parametr a je v rozmezí od 0 do 1.

- **Exponential Linear Unit - ELU** je další možnost, jak se vyvarovat stavu mrtvých neuronů, na rozdíl od Leaky ReLu nemá v negativní části přímku, ale logaritmickou křivku.



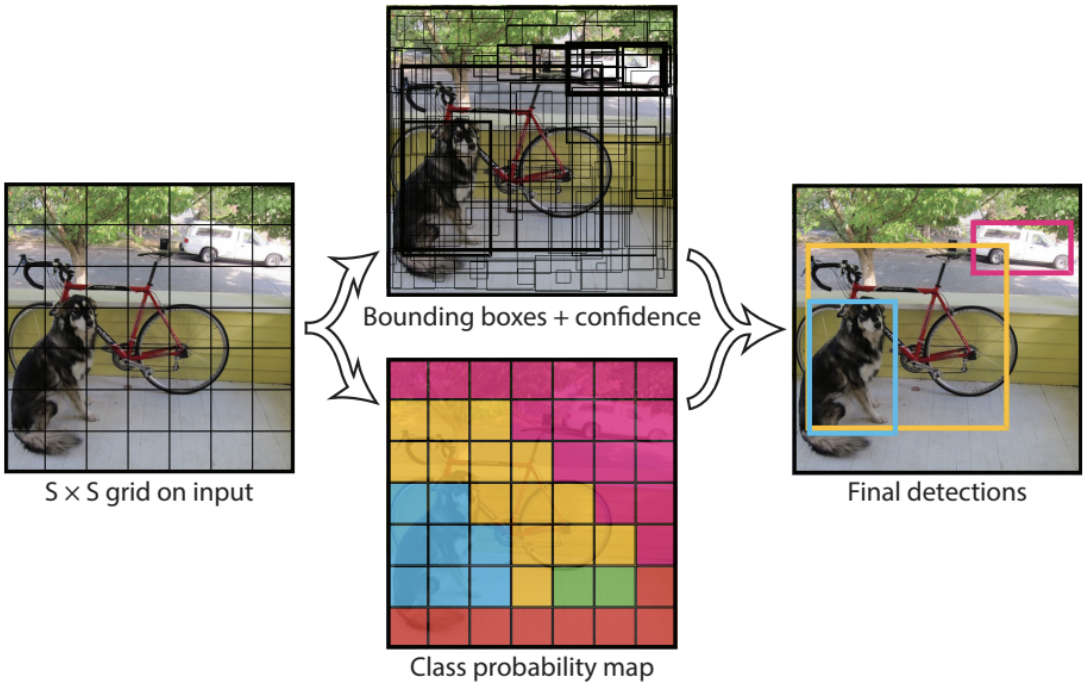
Obr. 4.5: Nelineární aktivační funkce [5]

[5]Převzato z: https://analyticsindiamag.com/what-are-activation-functions-and-when-to-use-them/

4.3 YOLO – You Only Look Once

Yolo je detekční systém, který pracuje v reálném čase. Na rozdíl od jiných detektorů, které pracují ve dvou fázích (klasifikace a detekce). Yolo obsahuje pouze jednu konvoluční síť, která predikuje ohraničující rámečky a jejich pravděpodobnosti tříd v jednom cyklu. Další výhodou od jiných detektorů je jeho rychlost. Tomu napomáhá, že již při trénování pracuje s plnými snímky, takže jeho předpovědi jsou ovlivněny celkovým obsahem obrázku.

Při detekci rozdělí vstupní obraz do mřížky o velikosti $S \times S$, jak ukazuje obrázek 4.6. Pokud se střed objektu nachází v dané buňce, tak je tahle buňka zodpovědná za jeho klasifikaci. Každá buňka také produkuje B počet ohraničení a hodnotu důvěrnosti. Hodnota důvěrnosti říká, jak jistý si je model, že ohraničení obsahuje objekt a jak přesné je tohle ohraničení. Neobsahuje-li buňka žádnou třídu, je hodnota důvěrnosti blížící se k nule.



Obr. 4.6: YOLO princip detekce, vytvoření ohraničení [13]

Každé ohraničení se skládá z pěti předpovědí. První čtyři predikce jsou parametry vypovídající o pozici a velikosti ohraničení. Jedná se o středové souřadnice (x, y) ohraničení vzhledem k umístění buňky v mřížce. Dále se jedná o parametry (w, h) výšku a šířku ohraničení, jejich velikost je normována vzhledem k celkové

výšce a šířce obrázku. Poslední je predikce jistoty, představuje IoU mezi predikovaným ohraničeným a správným ohraničením. Každá buňka také predikuje C počet podmíněných pravděpodobností jednotlivých tříd. Tyto pravděpodobnosti jsou podmíněny tím, že v buňce se musí nacházet objekt. Předpovídaná je jen jedna sada pravděpodobnosti tříd na buňku bez ohledu na počet ohraničení B. [13]

4.3.1 YOLOv2

Jedná se o druhou verzi sítě Yolo, která má vylepšit nedostatky původní verze. A to především zlepšení lokalizace a rychlosti detekce, zároveň ale při zachování přesnosti detekce. Na místo toho, aby byla použita složitější síť, která by zlepšila přesnost detekce, ale na úkor rychlosti, byla síť zjednodušena. Zjednodušení sítě mělo za následek mimo jiné i snadnější učení.

Normalizace dávek vedla ke zlepšení konvergence a zároveň odstranila potřebu dalších forem regulace. Zvedla se úspěšnost klasifikace jednotlivých tříd, bylo možné odstranit metodu dropout bez toho, aniž by se síť přetrénovala.

Odstranění plně propojených vrstev a použití anchor boxes[6] k předpovědi ohraničení. Bylo změněno rozlišení vstupního obrázku na 416×416 px, díky tomu mřížka obsahuje jednu buňku přímo ve středu místo čtyř. Použitím anchor boxes se sice snížila přesnost, ale namísto 98 ohraničení je možné předpovídat víc než tisíc. [14]

4.3.2 YOLOv3

Jedná se o třetí verzi sítě Yolo. V této verzi je umožněna klasifikace objektu do více tříd, z tohoto důvodu zde již není použita funkce softmax, která určuje pravděpodobnost, že objekt spadá do dané třídy vzhledem ke všem třídám. To znamená, že součet všech pravděpodobností všech tříd je jedna. Z toho důvodu byla funkce softmax nahrazena funkcí binary cross-entropy loss, která určuje pravděpodobnost pro každou třídu nezávisle na ostatních třídách.

Autoři se rozhodli nenásledovat kroky z druhé verze a přistoupili ke složitější síti. Síť obsahuje 53 konvolučních vrstev, důsledkem toho se zlepšila přesnost detekce, ale na úkor rychlosti.

Síť nyní předpovídá ohraničení ve třech různých měřítcích, k 13×13 bylo přidáno 26×26 a 52×52. To mělo za následek zlepšení detekce malých objektů, ale zhoršila se detekce středních a větších objektů. [15]

[6] Jedná se o set předem předdefinovaných ohraničení o určité výšce a šířce.

4.3.3 YOLO-Tiny

Tiny neboli odlehčené verze sítí YOLOv2 i YOLOv3 mají jediný úkol, několikanásobné zrychlení detekce, ale na úkor jejich přesnosti detekce. Tiny verze se skládají z menšího počtu konvolučních vrstev.

- **YOLOv2-Tiny**: původních 24 konvolučních vrstev se snížilo na 9, rychlost detekce se zrychlila až na pětinásobek, ale přesnost klesla o víc než 30%. [14]

- **YOLOv3-Tiny**: původních 106 konvolučních vrstev se snížilo na 13, snížil se počet detekcí pouze na dvě rozlišení. Původní rychlost se zvýšila až 12 krát, zatímco přesnost se snížila o 40%. [15]

4.4 EfficientDet

Jedná se o rodinu detektorů objektů, která se zaměřila na optimalizaci a zlepšení efektivity oproti již existujícím detektorům objektů. Aby bylo dosaženo požadovaného výsledky, bylo potřeba zavést několik optimalizací pro zlepšení efektivity. Nejprve byla navržena BiFPN, viz kapitola 4.4.1, která umožňuje snadnou a rychlou fúzi víceúrovňových příznaků. Za druhé byla navržena metoda složeného škálování, která rovnoměrně mění rozlišení, hloubku a šířku všech páteřních sítí, příznakové síti (BiFPN) a sítí pro předpověď ohraničení a třídu současně. Dále autoři vytvořili vlastni páteřní síť EfficientNet, kde všechny tyto optimalizace aplikovali. [26]

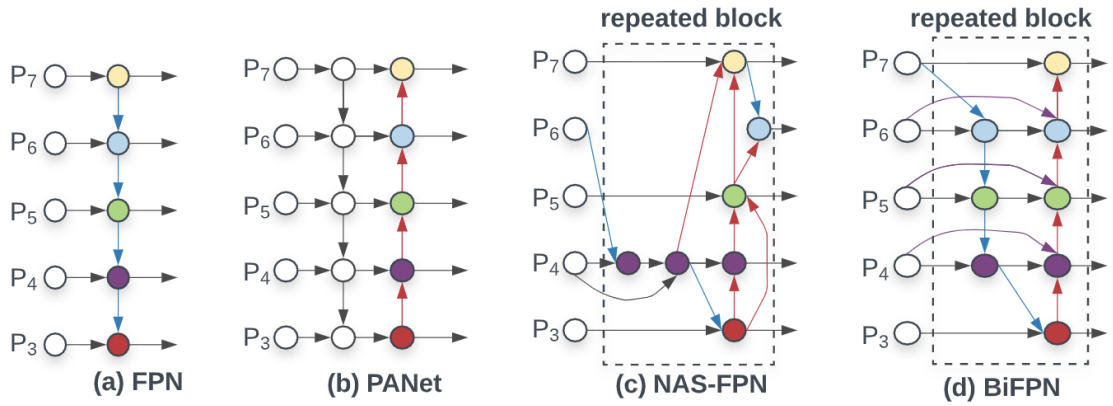
Tab. 4.1: Porovnání EfficientDet s vybranými detektory objektů [26]

Model	AP_{test}	Par (mil)	FLOP (mld)	GPU_{ms}	CPU_s
EfficientDet-D0 (512)	33,8	3,9	2,5	16	0,32
YOLOv3	33	-	71	51	-
EfficientDet-D1 (640)	39,6	6,6	6,1	20	0,74
RetinaNet-R50 (640)	37	34	97	27	0,74
EfficientDet-D2 (768)	43	8,1	11	24	1,2
RetinaNet-R101 (1024)	41,1	53	326	65	9,7
EfficientDet-D3 (896)	45,8	12	25	42	2,5
ResNet-50 NAS-FPN (1280)	44,8	60	563	119	17
EfficientDet-D4 (1024)	49,4	21	55	74	4,8
EfficientDet-D5 (1280)	50,7	34	135	141	11
EfficientDet-D6 (1280)	51,7	52	226	190	16
EfficientDet-D7 (1536)	52,2	52	325	262	24

V tabulce 4.1 jsou uvedené vybrané příklady porovnání EfficientDet s dalšími vybranými detektory. Srovnáni bylo provedeno pro COCO[7] datovou sadu, měření odezvy bylo provedeno pro GPU Titan V, kromě YOLOv3, kde je odezva převzata z práce [15], batch size byl zvolen 1. V tabulce můžeme vidět že EfficientDet-D0 dosáhl stejné přesnosti jako YOLOv3 i když EfficientDet-D0 má asi 28 krát méně FLOP (operací s plovoucí desetinou čárkou). [26]

4.4.1 BiFPN - Weighted Bi-directional Feature Pyramid Network

BiFPN neboli obousměrná pyramidová příznaková síť, je typ příznakové pyramidové sítě, která vychází z předchozích fúzí víceúrovňových příznaků, jako jsou FPN, PA-Net a NAS-FPN. BiFPN umožňuje tok informací v obou směrech shora dolů a zdola nahoru. Aby se ještě zvýšila efektivita využívá také techniku rychlé normalizované fúze. Tradiční přístupy nerozlišují rozdíl pro různé příznakové vstupy do FPA a to i pro ty, které mají různé rozlišení. Tudíž vstupní příznaky v jiném rozlišení tak většinou mají nerovnoměrně příspěvky na výstupních příznacích. BiFPN tedy přidává další váhu pro každý příznakový vstup, aby umožnila síti odlišit jejich důležitost. [26]



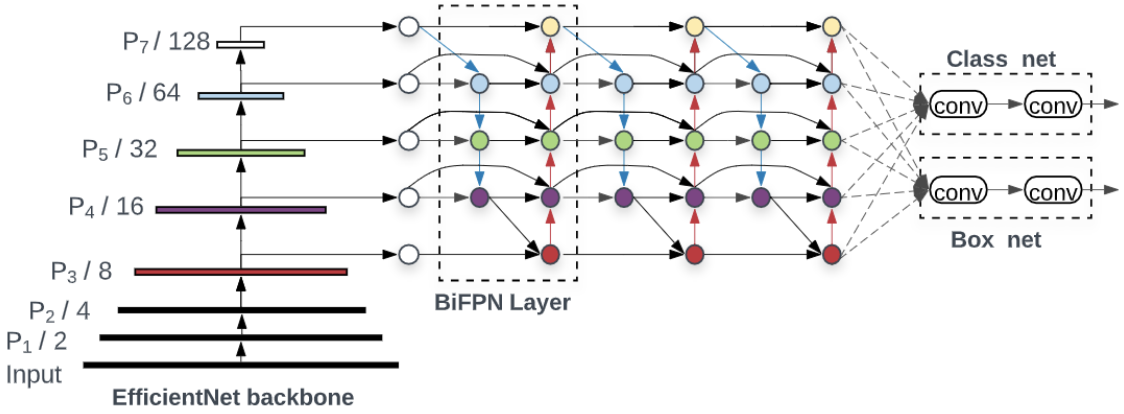
Obr. 4.7: Porovnání BiFPN s předchozími příznakovými sítěmi [26]

Na obrázku 4.7 můžeme vidět, že BiFPN odstranila uzly s jedním vstupem a přidala další cestu z původního vstupního do výstupního uzlu, pokud se nacházejí na stejné úrovni.

[7] https://cocodataset.org/#home

4.4.2 Architektura EfficientDet

Na obrázku 4.8 je zobrazena architektura EfficientDet, který používá jako páteřní síť EfficientNet. Obsahuje nově vytvořené vrstvy BiFPN, poté jsou výstupy přivedeny do sítí Class a Box, kde je vytvořena předpověď třídy a ohraničujícího rámečku objektu. [26]



Obr. 4.8: Architektura EfficientDet [26]

4.4.3 Metoda složeného škálování

Jedná se o metodu škálování, která mění rozlišení, hloubku a šířku v každé části sítě stejně. To znamená, že páteřní síť, příznaková síť, predikční síť pro třídu a ohraničení všechny tyhle částí budou mít stejný škálovací faktor, který se označuje škálovací koeficient ϕ. Tento koeficient řídí všechny dimenze škálování pomocí heuristických pravidel.

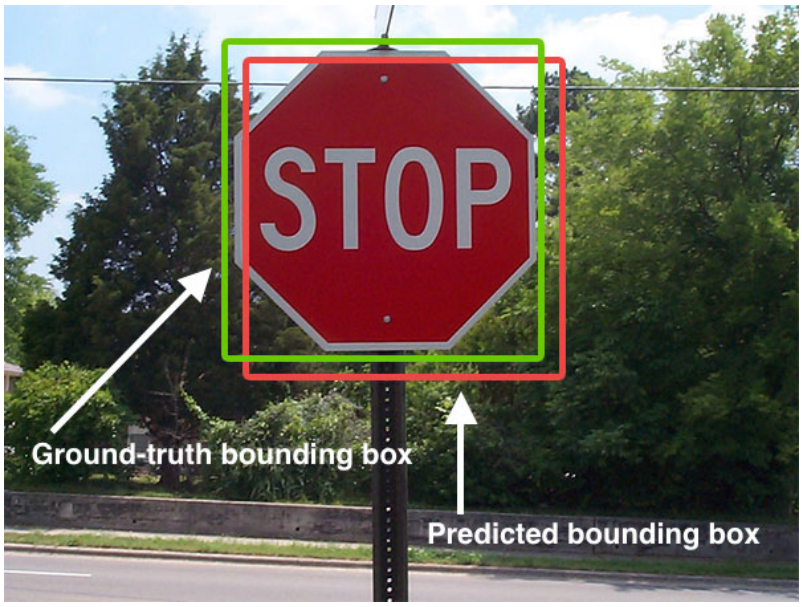
Kombinací nové páteřní sítě EfficientNet a BiFPN se nejprve vytvořila základní struktura EfficientDet-D0 malé velikosti a poté se aplikuje metoda složeného škálování, díky ní se získá EfficientDet-D1 až D7. Každý model, který má větší koeficient ϕ, má vyšší výpočetní náročnost, obsahuje větší počet FLOP od 3 do 300 miliard a dosahuje větší přesnosti. [26] [27]

4.5 Hodnocení úspěšnosti detektorů

Vyhodnocení úspěšnosti detektorů je složitější něž u klasifikátorů. Na rozdíl od klasifikátoru, kdy nás zajímá jestli je klasifikace úspěšná či nikoliv, popřípadě s jakou pravděpodobností, je na vstup detektoru přiveden celý obrázek a ne jen oblast zájmu. Proto pro hodnocení úspěšnosti detektorů existují jiné způsoby.

4.5.1 IoU - Intersection over Union

Intersection over Union dále jen IoU je metrika hodnocení používaná k zjištění přesnosti detektoru objektů. Abychom mohli vypočítat hodnotu IoU potřebuje dvě sady ohraničení pro daný objekt na obrázku. První ohraničení je ohraničení daného objektu z anotace datové sady, druhé ohraničení je výstup příslušného detektoru. [20] Na obrázku 4.9 je příklad zobrazení IoU, kdy ohraničení z anotace datové sady je zobrazeno zelenou barvou a ohraničení, které získáme z detektoru má barvu červenou. Plochu zeleného ohraničení budeme označovat jako A a plochu červeného jako B.



Obr. 4.9: Příklad zobrazení IoU [20]

Pomocí vzorce 4.2 můžeme vypočítat hodnotu IoU, která určuje, jak přesně detektor označil daný objekt. Většinou se hranice pro úspěšnou detekci považuje hodnota $0,5$, při tvorbě detektoru se snažíme aby se IoU blížila hodnotě 1.

$$IoU = \frac{A \cap B}{A \cup B} \tag{4.2}$$

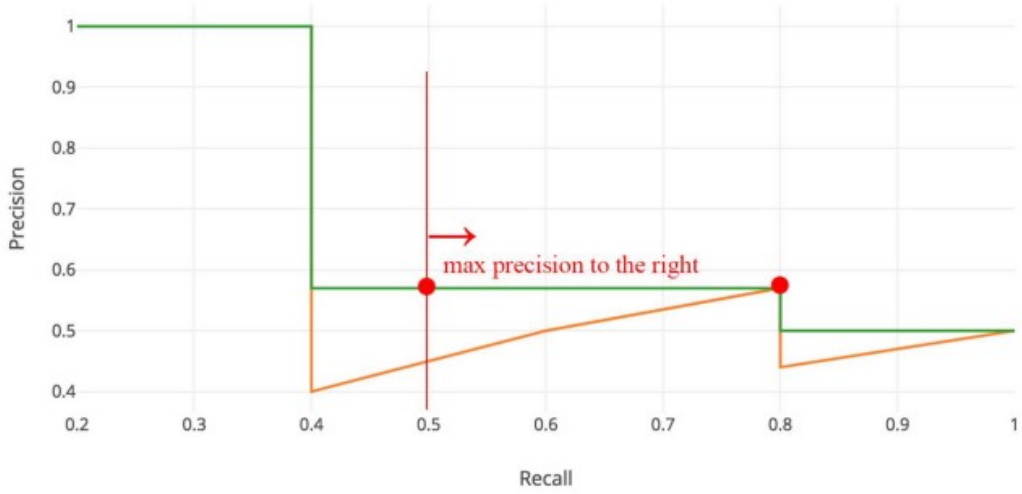
4.5.2 AP - Average Precision

AP (průměrná přesnost) je populární metrika pro měření přesnosti detektorů objektů pro každou třídu samostatně, a nebo můžeme vypočítat celkovou průměrnou hodnotu mAP (mean Average Precision) pro všechny třídy dohromady. Abychom mohli vypočítat AP musíme nejprve zjistit hodnoty precision pro různé hodnoty

recall. [33]

- **Precision** měří jaké procento předpovědí je správné.
- **Recall** vyjadřuje, jak je model schopný najít všechny anotované objekty na obrázku.

Poté jak máme potřebné hodnoty, tak sestrojíme graf závislosti precision na recall na obrázku 4.10 oranžová křivka. Hodnota AP je plocha pod křivkou. Pro zjednodušení se používá 11 bodová interpolace, kdy zjišťujeme hodnoty precision jen pro hodnoty recall od 0 do 1 s krokem 0,1 na obrázku 4.10 zelená křivka. Interpolace se provádí pro maximální hodnotu precision vpravo od prováděné interpolace, zobrazeno na obrázku 4.10 červeně. A výsledná hodnota AP je průměr zmíněných 11 interpolací.



Obr. 4.10: Precision-Recall křivka [33]

COCO (Common Objects in COntext) AP

Jedná se o způsob hodnocení detektoru, který je použit i pro hodnocení EfficientDet v této práci. Nejprve jsou spočítány AP a AR (Average Recall) pro různé hodnoty IoU, konkrétně se používá 10 hodnot IoU od 0,5 do 0,95 s krokem 0,05, pokud není uvedeno jinak. Poté je vypočteno mAP pro již 10 zmíněných hodnot IoU. Výsledná hodnota COCO AP je pak průměrná hodnota ze všech mAP. Taktéž jsou uváděny hodnoty AP pro IoU 0,5 a 0,75. COCO nerozlišuje mezi AP a mAP, předpokládá kontext z textu.

Mimo to ještě COCO hodnotí AP a AR pro různě velké objekty. Pro malé objekty, jejíchž velikost je do 32^2 pixelů, střední objekty od 32^2 do 96^2 pixelů a velké objekty větší něž 96^2 pixelů. Dále uvádí ještě hodnocení AR pro maximální počet detekcí na snímek pro 1, 10 a 100 detekcí. [32]

5 FRAMEWORKY PRO PYTHON

Framework je softwarová struktura, která slouží pro vývoj softwarový aplikací. Poskytuje základ pro vytváření softwarových aplikací pro určitou platformu. Může obsahovat podpůrné programy, knihovny API, předdefinované třídy nebo funkce. To usnadňuje práci programátorům při vývoji softwarových aplikací. [29]

5.1 PyTorch

PyTorch je framework strojového učení vyvinutý společnosti Facebook's AI Research lab (FAIR). Je založený na balíčku Torch, což je open source balíček pro strojové učení. Má dvě hlavní funkce, první je podobná knihovně Numpy, jedná se o výpočet tenzorů za podpory GPU. Druhá funkce je vytváření neuronových síti založených na autodiff (automatická diferenciace).

Výhodou je jeho flexibilita oproti některým jiným frameworkům např. Tensorflow, kde je potřeba definovat celý výpočetní graf (computational graph) před spuštěním, tak PyTorch nám umožňuje definovat výpočetní graf dynamicky.

PyTorch používá metodu zvanou autodiff, která numericky vyhodnocuje derivaci funkce. Automatická diferenciace počítá zpětné průchody v neuronových sítích. V procesu tréninku jsou váhy neuronové sítě náhodně nastaveny na čísla blížící se nule. A zpětným průchodem se tyto váhy upravují zprava doleva. Existuje i inverzní zpětný průchod, kdy se váhy upravují zleva doprava. [30]

5.2 TensorFlow

Tensorflow je open source platforma od firmy Google Brain, pro vytváření aplikací strojového učení. Jedná se o knihovnu, která se zaměřuje na trénování neuronových sítí.

TensorFlow umožňuje vytvářet grafy a struktury datových toků, které definují, jak se data pohybují v grafu. Jako vstup se bere vícerozměrné pole označované jako Tensor. Také nám umožňuje vytvořit vývojový diagram operací, které lze na těchto vstupech provést.

Části TensorFlow: [31]

- **Tensor** provádí všechny výpočty v knihovně TensorFlow.
- **Graph** shromažďuje a popisuje všechny výpočty, které byly vykonány v průběhu tréninku.

5.2.1 TensorFlow Lite

TensorFlow Lite je sada nástrojů, která umožňuje řešení strojového učení pro zařízení s nízkou latencí a malou binární velikostí. Pomáhá vývojářům spouštět jejich modely na mobilních, jednodeskových a IoT zařízeních.

Model TensorFlow Lite převede stávající model TensorFlow do optimalizované a efektivní verze. Zjednodušený model je již dostatečně malý, aby mohl být uložen v zařízení a dostatečně přesný, aby odvedl požadovaný úkon.

Výhody TensorFlow Lite oproti TensorFlow pro zařízení zmíněná výše: [25]

- Snadné převedení modelu TensorFlow do odlehčené verze TensorFlow Lite pro optimalizovaná mobilní zařízení jako jsou iOS nebo Android.
- Snadný vývoj aplikací strojového učení pro iOS a Android.
- Efektivnější alternativa modelů uložených přímo v zařízení, něž na vzdáleném serveru. Možnost použití i bez připojení k internetu.

Nevýhody TensorFlow Lite oproti TensorFlow pro zařízení zmíněná výše [25]:

- Některé modely jsou stále příliš velké nato, aby je bylo možné uložit na zařízeních zmíněných výše.
- Snížení jejich velikosti a optimalizace vede ke snížení přesnosti modelu oproti plné verzi.

6 IMPLEMENTACE

Pro vytvoření klasifikátoru byla zvolena metoda strojového učení, přesněji konvoluční neuronová síť. Model neuronové sítě klasifikátoru byl vytvořen pomocí knihovny Tensorflow, jejíž novější verze 2.0 a víš již obsahují knihovnu Keras v základním balíčku. Výhoda knihovny Tensorflow je, že její odlehčená verze Tensorflow Lite má podporu pro jednodeskové počítače např. typu Raspberry Pi.

Jako detektor dopravních značek byl zvolen EfficientDet-D2, jedná se o rodinu nových detektorů, která se změřila na optimalizaci a zlepšení efektivity oproti jiným detektorům. Detektory EfficientDet jsou vytvořeny pomocí knihovny PyTorch.

6.1 Klasifikátor dopravních značek

Klasifikátory slouží k rychlé identifikaci v předem vyznačené části obrázku. Pokud tedy chceme klasifikovat dopravní značky v reálném čase, například z dopravní kamery, je nutné nejprve daný obrazový signál zpracovat vhodným detektorem, který nám určí oblast, ve které se dopravní značka nachází. Klasifikátory jsou tedy takový podpůrný prostředek pro detektory a z pravidla mají jednoduchou architekturu, ale vysokou pravděpodobnost klasifikace daného objektu, tedy dopravní značky.

6.1.1 Metody klasifikace dopravních značek

Způsoby jak klasifikovat dopravní značky můžeme rozdělit do dvou kategorií.

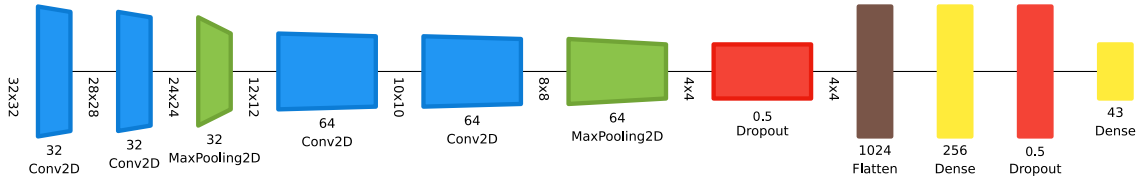
První kategorie jsou metody, které jsou založeny na klasifikaci podle tvarů nebo barev, výhodou této metody je, že dopravní značky mají své specifické barvy a tvary, které se běžně v přírodě nevyskytují. Dalším způsobem je porovnání dopravní značky s již existující databází značek, nevýhoda této metody je její časová náročnost a problém s rozpoznáváním deformovaných dopravních značek. [17]

Jako další metody jsou metody strojového učení, neboli vytvoření neuronové sítě. Při vytváření neuronové sítě je potřeba datová sada, podle které se síť naučí specifické prvky pro dané třídy. Neuronová síť dostane vstupní data a podle výstupních dat sítě a dat které obsahuje datová sada, určí chybu a provede korekci. Nově nastaví hodnoty prahů a vah. Celý tento cyklus opakuje do chvíle, dokud nenajde takovou konfiguraci prahových hodnot a vah, která odpovídá naší stanovené chybě. [11] [12] Nevýhoda této metody je pokud chceme síť rozšířit na více tříd, je nutné provést celý proces učení znovu, zatímco při porovnávacích metodách se připíše jen pár podmínek.

6.1.2 Architektura klasifikátoru dopravních značek

Zvolen byl sekvenční model, který umožňuje skládat vrstvy lineárně za sebe, každá vrstva je propojena jen s předchozí a následující vrstvou. Mimo to také obsahuje jen jeden vstup a výstup, což pro klasifikaci dopravních značek není žádné omezení.

Vstupní obrázek klasifikátoru je o rozměrech $32 \times 32 \times 1$, obrázek je ve stupních šedi. Výhodou obrázku ve stupních šedi je redukce parametrů v první vrstvě o třetinu, aniž by se snížila úspěšnost klasifikace.



Obr. 6.1: Architektura klasifikátoru

Obrázek, který má přesně dané rozměry, následuje do série konvolučních a sdružovacích vrstev, jak můžete vidět na obrázku 6.1. V architektuře jsou použity dvě dropout vrstvy, které se uplatňují jenom při trénování sítě. Zabraňují přetrénování sítě, tím že náhodně nastaví vstup pro další vrstvu na 0, pro tuhle architekturu je pro obě dropout vrstvy hodnota 0,5 což znamená, že 50 % výstupu je nastaveno na 0, hodnota 0,5 se používá nejčastěji. Poslední část architektury jsou plně propojené vrstvy, poslední vrstva obsahuje 43 uzlů, což odpovídá počtu tříd. Aktivační funkce poslední vrstvy je softmax, které se používá pro jedno třídovou identifikaci, určuje pravděpodobnost, že objekt spadá do určité třídy, součet všech pravděpodobností tříd je 1. [18] Až na poslední vrstvu, která má funkci softmax, jsou použity aktivační funkce ReLU, její průběh je možné vidět na obrázku 4.5.

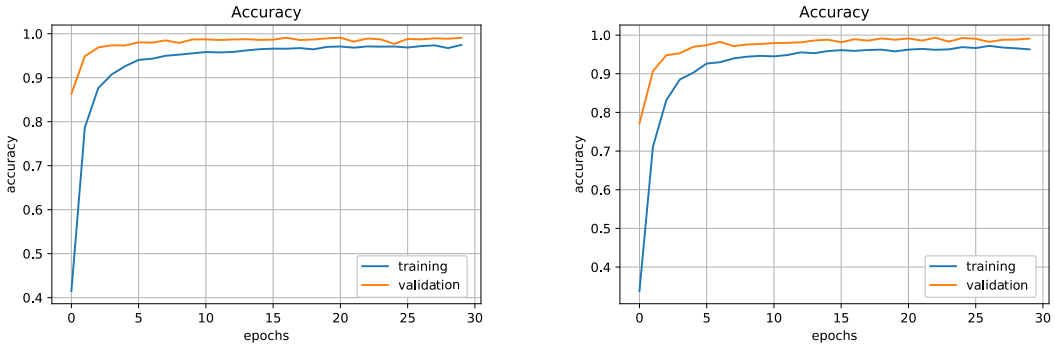
Optimalizátor je algoritmus, který mění hodnoty neuronové sítě, jako jsou váhy a míra učení, aby ztráty byly co nejmenší. V této architektuře byl použit optimalizátor Adam, který používá metodu gradientního sestupu. Metoda spočívá ve výpočtu první derivace ze ztrátové funkce a podle toho předpovídá, jak by se měli váhy měnit, aby funkce mohla dosáhnout minima. [17] Metoda je výpočetně efektivní, má malý požadavek na paměť.

Ztrátová funkce vyjadřuje aktuální ztrátu sítě během trénování. Výstupem funkce jsou ztráty, které určují rozdíl mezí předpokládaným výstupem a daným výstupem neuronové sítě. Pro klasifikaci do více než dvou tříd se používá funkce Categorical Cross-entropy, kdy každé třídě odpovídá jeden výstupní neuron.

6.1.3 Trénování klasifikátoru

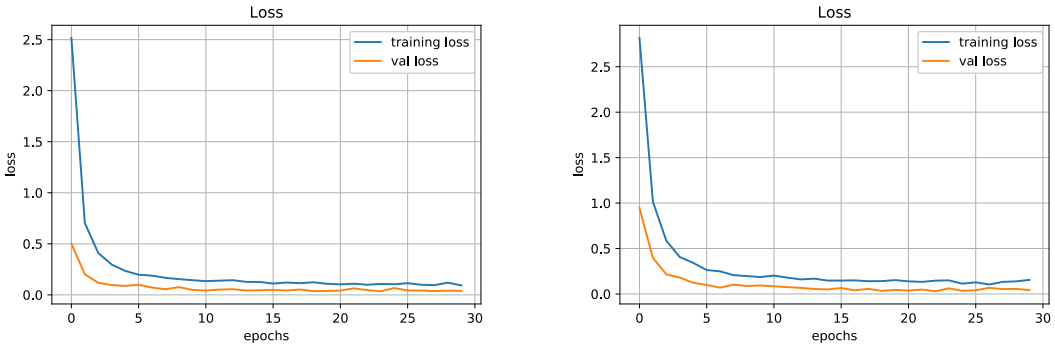
Pro trénování klasifikátoru byla zvolena německá datová sada GTSRB, důvodem je neexistující česká datová sada a Německo je náš soused a z historického hlediska máme hodně společného, včetně podobnosti vzhledu dopravních značek[1]. Datová sada byla rozdělena na tři množiny, pro trénování bylo použito 20 tisíc obrázků, testovací množina obsahuje 3 tisíce snímků a ověřovací množina 5 tisíc obrázků.

Trénování bylo provedeno pro 30 cyklů, kdy do každého cyklu vstupovaly všechny obrázky z množiny pro trénování. Trénování bylo provedeno jak pro model RGB, tak i pro stupně šedi. Pro oba modely byly v průběhu testování tvořeny grafy pro zobrazení jejich přesnosti a ztrát pro každý cyklus, viz obrázek 6.2 a 6.3.



(a) Stupně šedi (b) RGB

Obr. 6.2: Přesnost klasifikátoru



(a) Stupně šedi (b) RGB

Obr. 6.3: Ztráty klasifikátoru

[1]Datová sada je dostupná zde: `http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset`

Přesnost a ztráty byly zaznamenávány pro každý cyklus trénování, podívejme se nejdříve na přesnost, pro oba modely je přesnost klasifikátoru trénovací sady již po pěti cyklech přes 90 %, pro ověřovací sadu je to již po třech cyklech. Průběh odpovídá rostoucí logaritmické funkci, která dosahuje po 30 cyklech přesnosti až 99 % jak pro testovací, tak i pro ověřovací sadu. Průběh ztrát má opačný charakter, průběh pro trénovací sadu je klesající logaritmická funkce, pro kterou jsou ztráty na konci trénování menší než 0,3 pro trénovací sadu, pro ověřovací sadu jsou ztráty pod 0,2.

Porovnáním obou barevných modelů vzhledem k jejich přesnosti v průběhu trénování podle obrázku 6.2 a také porovnáním jejich průběhu ztrát v době jejich trénovaní viz obrázek 6.3 můžeme usoudit, že výsledky dosažené na konci trénování jsou pro oba modely shodné. Z toho vyplývá, že model RGB neměl vliv na výslednou přesnost. Můžeme tedy použít model ve stupních šedi, aniž by se snížila přesnost klasifikátoru.

6.2 Detektor dopravních značek

Detektory slouží k nalezení oblasti zájmu na obrázku, v našem případě dopravní značky. Každý detektor obsahuje klasifikátor, který je konstruován aby neomezoval rychlost detekce, a tudíž jeho požadována přesnost nemusí být splněna. Proto se někdy klasifikátor omezí pouze na jednu třídu a doplní se externím klasifikátorem s větší přesností, použito v této práci.

6.2.1 Příprava datové sady

Datová sada je nedílnou součástí každé neuronové sítě, různé architektury vyžadují různé požadavky na to, jak má vypadat anotace datové sady. Pro trénování detektoru EfficientDet byla vybrána datová sada GTSDB[2], důvodem je její návaznost na datovou sadu použitou pro klasifikátor. Obě zmíněné datové sady obsahují stejné třídy dopravních značek.

EfficientDet vyžaduje datovou sadu v COCO[3] formátu. Nejprve je potřeba datovou sadu rozdělit na testovací a ověřovací množinu. Testovací množina obsahuje 2/3 z 900 snímků, zbylou 1/3 obsahuje ověřovací množina. Jakmile máme rozdělenou datovou sadu, je potřeba vytvořit anotaci. Anotace pro detekci objektů v COCO formátu vyžaduje v .json souboru tyto hlavní kategorie: annotationns, categories, images. Images obsahuje jméno a příponu všech obrázku pro danou množinu, jeho výšku, šířku a pořadí v jakém je zapsán, důležité pro další kategorii. Categories obsahuje supercategory, id a name. Supercategory je nadtřída, která zahrnuje více tříd např. zákazové značky. Name je jméno třídy a id její číslo. Jak již bylo zmíněno

detektor obsahuje pouze jednu třídu, čili anotace neobsahují supercatergory a obsahují jen jednu třídu "Znacka". Kategorie annotations obsahuje všechny dopravní značky, které chce detekovat a vyžaduje, jak velkou oblast zabírají v px, souřadnice začátku (x, y), šířku a výšku. Dále do jaké třídy daný objekt patří, pořadové číslo s jakým je zapsán, pořadové číslo obrázku ke kterému patří a informaci o tom, jestli se nepřekrývá s jiným detekovatelným objektem.

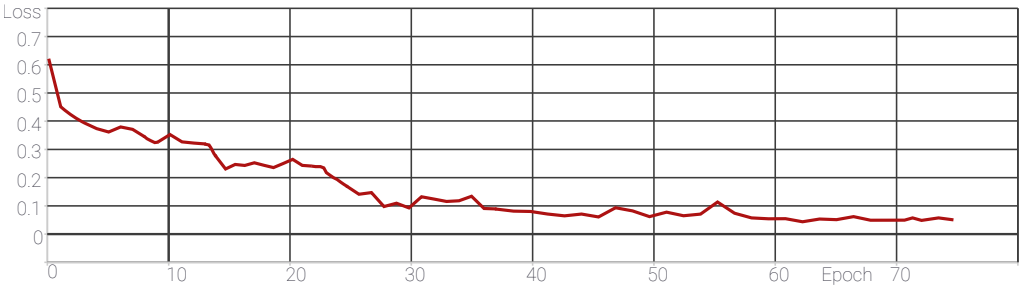
Dále je potřeba ještě pro datovou sadu upravit anchors scales a anchors ratios, aby detektor byl co nejefektivnější pro danou datovou sadu. Ve stejném souboru s koncovkou .yml nastavíme i počet GPU, které chceme využít pro trénování detektoru.

6.2.2 Trénování detektoru

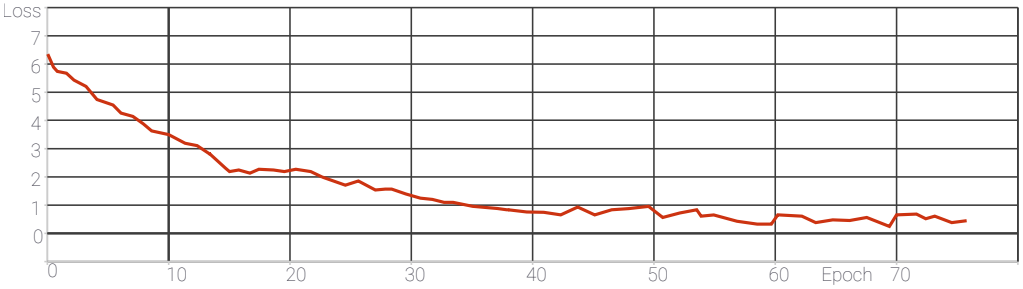
Jakmile máme připravenou datovou sadu, vytvořeny všechny potřebné anotace, tak musíme nastavit parametry detektoru. Nejdůležitější parametr rodiny detektorů EfficientDet je škálovací koeficient ϕ, který udává složitost celé architektury. V této architektuře byla zvolena hodnota koeficientu 2, což znamená že vstupní obrázky pro detektor budou ve formátu 768×768 px. Hodnota byla zvolena vzhledem k rychlosti a přesnosti detekce podle [28]. Vzhledem k použití na jednodeskovém počítači by se zdálo vhodnější použít škálovací koeficient 0. Dá se předpokládat, že rychlost by se mohla i 3 násobně zvýšit, ale vzhledem k přesnosti by mohlo docházet k problémům detekce středních objektů. Jako další parametry je potřeba nastavit počet cyklů, počet obrázků, které vstupují do cyklu. Je potřeba nastavit learning rate (rychlost učení) a optimalizátor. EfficientDet používá optimalizátor SGD, v této architektuře je možnost použít i optimalizátor AdamW. Podle [28] je vhodné použít optimalizátor SGD až pro závěrečné cykly, v případě malých datových sad je vhodné použít pouze AdamW. Pomocí dalších parametrů můžeme nastavit po kolika cyklech je ověřovací fáze nebo ukončit trénovaní pokud se již nemění ztráty apod.

[3] `https://benchmark.ini.rub.de/gtsdb_news.html`

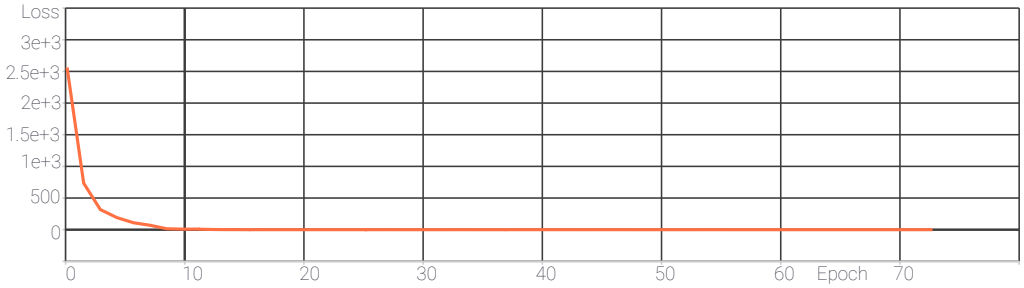
[3] `https://cocodataset.org/#format-data`



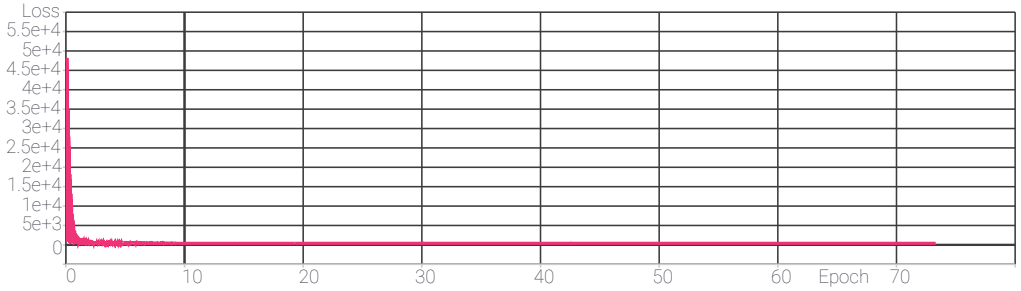
(a) Regresní ztráty pro ověřovací sadu



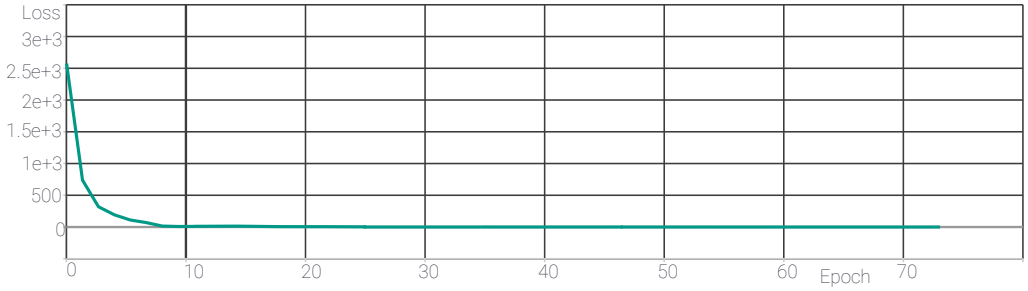
(b) Regresní ztráty pro trénovací sadu



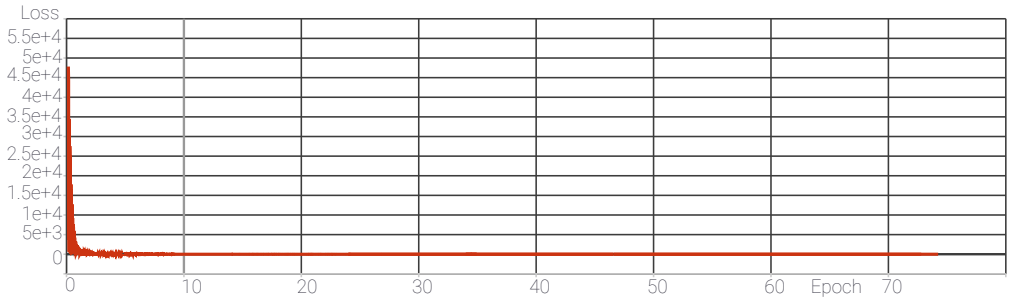
(c) Klasifikační ztráty pro ověřovací sadu



(d) Klasifikační ztráty pro trénovací sadu



(e) Celkové ztráty pro ověřovací sadu



(f) Celkové ztráty pro trénovací sadu

Obr. 6.3: Ztráty detektoru v průběhu trénování

Trénování probíhalo původně pro 150 cyklů, kdy do jednoho cyklu vstupovali všechny užitečné obrázky, těch bylo 494. Learning rate byl nastaven na 1^{-4}, po 50 cyklech se snížil na 1^{-5}. Optimalizátor byl použit AdamW. Ověřovací cyklus probíhal po 5 cyklech. Trénování probíhalo na grafické kartě Nvidia GeForce GTX 1050 Ti s 4 GB, batch size byl nastaven na 1 a doba jednoho cyklu trvala okolo 10 minut. Trénování bylo ukončeno po 72 cyklech kdy už ztráty zůstávali konstantní, jak je možné vidět na obrázku 6.3, kde jsou taky zobrazeny grafické závislosti regresních, klasifikačních a celkových ztrát na počtu vykonaných cyklů pro testovací i ověřovací sadu. Jak můžeme vidět na obrázku 6.3, již po 10 cyklech ztráty pro ověřovací sadu dosahovali hodnot menších než 1, pro testovací sadu téhle hodnoty bylo dosaženo až po 40 cyklech. Nicméně ztráty nemají moc velký vypovídající charakter, proto je lepší použít hodnocení pomocí AP.

6.2.3 Implementace klasifikátoru do architektury detektoru

Jak již bylo zmíněno, byly vytvořeny dvě neuronové sítě, konvoluční neuronová síť pro klasifikaci dopravních značek a EfficientDet pro detekci dopravních značek. Datové sady obou neuronových sítí byly vybrány kvůli jejich podobnosti, obě datové sady obsahují stejné třídy. Detektory obsahují klasifikátory již v základní struktuře

a jedná se o nezbytnou část, kterou nelze zanedbat. Kvůli tomu byl detektor neuronové sítě potlačen co nejvíce. Při trénování neuronové sítě byla zvolena jen jedna třída, které obsahuje všechny dopravní značky anotované datovou sadou.

Implementace klasifikátoru do detektoru je velice jednoduchá. Detektor vykonává svojí práci, čili dostane vstupní obrázek a určí oblast zájmu a klasifikuje ji jako dopravní značku. Vnitřní část hraničení, která byla určena detektorem, je poté zpracována, aby odpovídala požadavkům pro klasifikátor, je změněn její rozměr, je převedena do stupňů šedi a upraven kontrast. Jakmile je obrázek v požadovaném formátu, je přiveden na vstup klasifikátoru, který určí o jakou dopravní značku se jedná a s jakou pravděpodobností. A výsledek vrátí detektoru, který poté oblast ohraničení doplní názvem dopravní značky a její pravděpodobností.

(a) Detekce (b) Detekce + Klasifikace

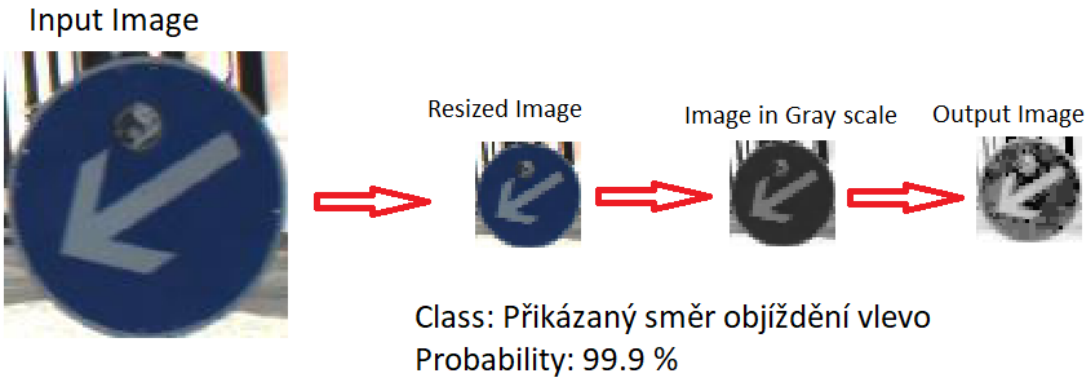
Obr. 6.4: Ukázka implementace klasifikátoru do architektury detektoru

7 VÝSLEDKY A TESTOVÁNÍ

V této kapitole jsou obsaženy všechny výsledky, kterých bylo dosaženo. Postupně jsou zde popsány výsledky klasifikátoru a poté detektoru. Kapitola také obsahuje kompletní testování celkové architektury.

7.1 Testování klasifikátoru

Testování klasifikátoru bylo provedeno pro celou testovací sadu, pro oba dva modely. Výsledky obou modelů pro celou testovací sadu se pohybují přes 99 %. Dále se bude používat jen model trénovaný na obrázcích ve stupních šedi z důvodu menšího počtu parametrů a tedy menšímu potřebnému výpočetnímu výkonu. Vstupní obrázek nejprve prochází úpravami, tak aby se shodoval s obrázky, které byly použity při testování. Nejprve je změněn jeho rozměr na 32×32 px, poté je převeden do stupňů šedi a je změněn jeho kontrast. Příklad potřebných úprav před klasifikací a ukázka klasifikace obrázku je zobrazena na obrázku 7.1.



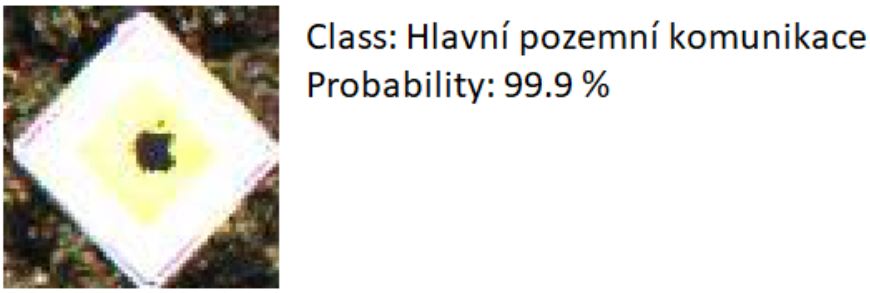
Obr. 7.1: Zpracování testovacího obrázku

Na obrázku 7.2 je uveden příklad správné klasifikace. Dopravní značka pro klasifikaci je značně ovlivněna světelným vlivem v tomhle případě se jedná o sluneční svit a v jejím středu se nachází objekt, který není součástí tohoto typu dopravní značky, i tak ale klasifikátor určil správně typ značky s pravděpodobností blížící se ke 100 %.

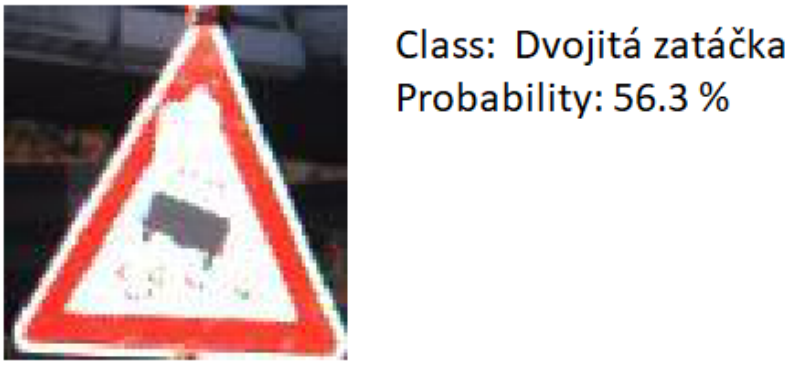
Na obrázku 7.3 je uveden mylný příklad klasifikace. Klasifikátor určil značku s 56% pravděpodobností jako dvojitou zatáčku, ale jedná se o značku nebezpečí smyku, jejíž pravděpodobnost vyhodnotil až jako druhou možnou s pravděpodobností 30 %.

Značka je ovlivněna slunečním svitem a na její horní části se nachází objekt, který není běžné součásti tohoto typu značky.

Nesprávná klasifikace se v testovacím setu vyskytovala u méně než 1 % případů. A ve většině případů se správný typ značky umisťoval s druhou nejvyšší pravděpodobností, která se od nesprávného typu lišila jen o pár desítek procent.



Obr. 7.2: Příklad správné klasifikace



Obr. 7.3: Příklad nesprávné klasifikace

Testování bylo prováděno na grafické kartě Nvidia GeForce GTX 1050 Ti, kde čas pro klasifikaci jednoho obrázku se pohyboval pod 0,1 sekundu.

Ve srovnání s přesnostmi ostatních systémů testovaných nad stejným datasetem vychází navržená neuronová síť jako druhá nejpřesnější. Srovnání úspěšnosti klasifikace ostatních systémů je v uvedeno v tabulce 7.1 [1].

[1] Výsledky jsou dostupné zde: `https://benchmark.ini.rub.de/gtsrb_results.html`

Tab. 7.1: Srovnání přesností navrženého systému klasifikace s ostatními řešeními

	Metoda	Úspěšnost klasifikace
DeepKnowlegde Seville	CNN with 3 Spatial Transformers	99,71 %
Navržené řešení	CNN	99,54 %
IDSIA	Committee of CNNs	99,46 %
sermanet	Multi-Scale CNNs	98,31 %
CAOR	Random Forests	96,14 %

7.2 Testování detektoru

Testování detektoru bylo provedeno jak pro ověřovací sadu, tak i pro trénovací sadu. Výsledky jsou uvedeny v tabulce 7.2. V tabulce můžeme vidět, že celkové AP, kde jsou zahrnuty všechny oblasti, dosahují jak pro trénovací sadu, tak i pro ověřovací sadu přívětivých hodnot v rozmezí od 0,5 do 0,6. Z tabulky taky můžeme vyčíst, že detektor si poradí s velkými a středními objekty bez větších problémů, nicméně detekce malých objektů je pro detektor problém.

Tab. 7.2: Hodnocení přesnosti detektoru

AP_{train}	AP_{val}	AR_{train}	AR_{val}	IoU	oblast	maximum detekcí
0,513	0,498	0,538	0,527	0,50:0,95	všechny	100
0,628	0,635	-	-	0,50	všechny	100
0,548	0,561	-	-	0,75	všechny	100
0,039	0,048	0,039	0,045	0,50:0,95	malá	100
0,773	0,750	0,810	0,796	0,50:0,95	střední	100
0,950	0,912	0,965	0,912	0,50:0,95	velká	100
-	-	0,538	0,527	0,50:0,95	všechny	10
-	-	0,380	0,378	0,50:0,95	všechny	1

Porovnání s ostatními metodami na stejném datové sadě je obtížné, jelikož metody uvedené v oficiální tabulce[2] se zaměřují pouze na jednotlivé druhy značek např. zákazové. Je těžké porovnat architekturu EfficientDet-D2 s ostatními detektory. Také metod existuje nespočetné množství pro danou datovou sadu a jejich škála AP se pohybuje od 0 až do 100 %.

Rychlost EfficientDet-D2 uvedená zde [28] dosahuje rychlosti 26 FPS, ale není řečeno jaký hardware byl použit. Rychlost, která byla testována na GPU Nvidia GeForce GTX 1050 Ti, dosahovala od 6 - 8 FPS v závislosti na počtu detekcí.

[2]Výsledky jsou dostupné zde `https://benchmark.ini.rub.de/gtsdb_results_all.html`

Po implementaci klasifikátoru rychlost klesla na 5 - 6 FPS v závislosti na počtu detekcí a klasifikací. Pro Jetson Nano byla změřena průměrná rychlost 0,35 FPS. Zvýšení rychlosti by bylo možné použitím výkonnějších, ale dražších modelů z řad NX nebo AGX. Pokud bychom ale stále chtěli využít Jetson Nano, mohli bychom zjednodušit architekturu, což by mělo za následek snížení přesnosti. Přesnost by jsme mohli zvýšit objemnější datovou sadou, rozšířenou např. o větší počet značek z různých úhlů pohledu. Objemnější datová sada by vedle k delšímu procesu učení a navíc není zaručeno, že by bylo dosaženo požadované přesnosti. Dále by bylo možné použít optimalizaci pomocí TensorRT, popřípadě TensorFlow-Lite, které by vedly ke zvýšení rychlosti na úkor přesnosti.

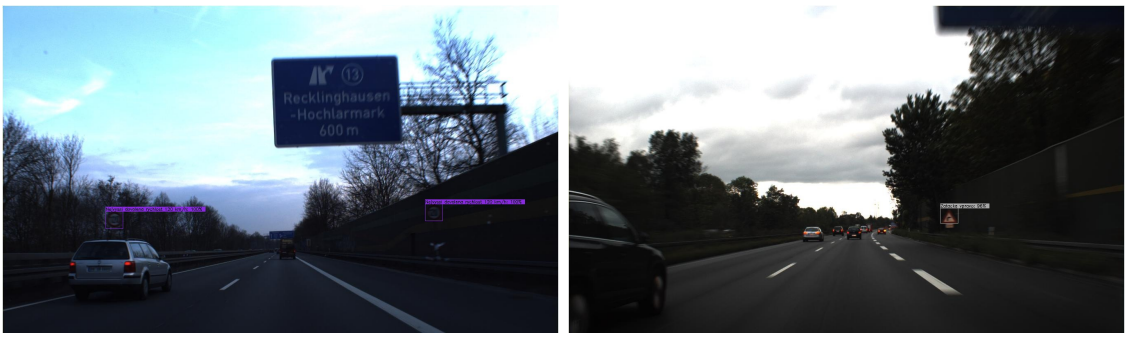
7.3 Testování celkové architektury

Pro testování celkové architektury byly zvolen threshold detektoru 0,5 a threshold klasifikátor 0,8. Vysoký threshold klasifikátoru je zvolen záměrně vzhledem k jeho podaným výsledkům a umožňuje v případě negativní detekce, kdy detektor označí objekt který není dopravní značky, opravit detektor a objekt vynechat. Nicméně přesnost klasifikátoru není 100% a může nastat případ, kdy detektor správně označí dopravní značku, ale klasifikátor jí klasifikuje s menší pravděpodobností než 80 % a vyřadí ji. Na obrázku 7.4a je uveden případ, kdy klasifikátor vyřadí správně detekovaný objekt. Pokud bychom neomezili threshold klasifikátoru, ve většině případů by klasifikátor určil třídu špatně. Pro ukázku příkladu správného rozhodnutí klasifikátoru byl dočasně snížen threshold detektoru, případ kdy detektor označí jiný objekt něž dopravní značku je velice vzácný a většinou je jeho pravděpodobnost menší něž 50 %, tudíž nastavený threshold detektoru na hodnotu 0,5 těmto případům v drtivé většině zabrání. Zmíněný příklad je zobrazen na obrázku 7.3b.

(a) Nesprávné rozhodnutí klasifikátoru

(b) Správné rozhodnutí klasifikátoru

Obr. 7.3: Rozhodnutí klasifikátoru



(a) Detekované dopravní značky za tmy

(b) Detekované dopravní značky pod vlivem slunečního světla

(c) Nedetekované dopravní značky za tmy

Obr. 7.4: Detekce při snížené viditelnosti


(a) Správně detekované dopravní značky


(b) Částečná detekce dopravních značek


(c) Nedetekované vzdálené dopravní značky

Obr. 7.5: Detekce bez vnějšcích vlivů

Na obrázku 7.4 jdou zobrazeny detekce za různých světelných podmínek. Na obrázku 7.4a a 7.4b jsou úspěšné detekce detektoru a na obrázku 7.4c jsou nedetekované dopravní značky za snížené viditelnosti. Na obrázku 7.5 jsou zobrazeny různé situace detekcí bez světelných vlivů. Na obrázku 7.6 jsou zobrazeny detekce pro deformované značky, na obrázku 7.6a je úspěšná detekce deformované značky, na obrázku 7.6b je úspěšná detekce deformované značky, ale klasifikace je špatná, navíc klasifikátor ji klasifikoval s úspěšností menší než 80 %, tudíž za normálního použití by byla ignorována, nicméně kvůli ukázce detekce deformovaných značek byl threshold klasifikátoru pro tento případ snížen. Na obrázku 7.7 je zobrazeno, jak by vypadala detekce ve video snímcích, obrázek taky demonstruje detekci pro různé

natočení dopravních značek.

(a) Detekce deformované značky

(b) Detekce deformované značky, ale špatná klasifikace

Obr. 7.6: Detekce deformovaných značek

Obr. 7.7: Detekce dopravních značek, simulace videa

Závěr

Cílem práce bylo prostudovat možné způsoby ke zpracování obrazových signálů, se zaměřením na detekci a klasifikaci dopravních značek v obrazcích a videosekvencích.

Pro dosažení tohoto cíle byly použity neuronové síť a programovací jazyk Python. V práci jsou uvedeny různé druhy neuronových sítí a nastíněn postup jejich fungování a k jakému účelu slouží.

Pomocí konvoluční neuronové sítě byl vytvořen klasifikátor dopravních značek pro dva barevné modely RGB a stupně šedi. Klasifikátor byl trénován i testován na německé datové sadě GTSRB, která obsahuje velké množství výřezů dopravních značek a je určená pro jejich klasifikaci. Úspěšnost klasifikace obou modelů na testovací datové sadě se pohybuje přes 99%. Pro následnou implementaci do architektury detektoru byl použit model, který pracuje v odstínech šedi, dosahoval stejných výsledků jako model v RGB a navíc jeho výhodou je, že v první vrstvě obsahuje o 1/3 méně parametrů, tudíž má menší nároky na výpočetní výkon.

Pro vytvoření detektoru dopravních značek byla vybrána rodina EfficientDet, která se zaměřuje na snížení potřebných parametrů optimalizací pomocí metody složeného škálování. Z rodiny byl zvolen detektor EfficientDet-D2, jedná se o zlatou střední cestu v poměru přesnost a rychlost detekce. Detektor byl trénován na německé datové sadě GTSDB, která obsahuje celé snímky. Úspěšnost detektoru na testovací i ověřovací sadě se pohybuje přes 0,6 $AP^{IoU_{0,5}}$. Detektor má problém detekovat malé dopravní značky, je to způsobeno malou datovou sadou, která obsahuje jen 1200 anotací. Jakmile detektor ukončil trénovací proces, byl do jeho architektury implementován klasifikátor dopravních značek. Celková architektury dosahuje rychlosti pro 5 - 6 FPS na GPU Nvidia GeForce GTX 1050 Ti a na jednodeskovém počítači Jetson Nano 0,35 FPS. Pro zrychlení detekce na jednodeskovém počítači je možné využít optimalizace pro TensorFlow a PyTorch, jedná se o TensorFlow-Lite a TensorRT. Zrychlení detekce, ale povede k určitému snížení přesnosti.

Jako budoucí vývoj práce je rozšíření počtu známých tříd obou neuronových sítí. S tím souvisí i rozšíření datových sad. Při rozšiřování datové sady detektoru je potřeba se zaměřit na větší počet malých objektů, jejichž detekce představuje momentálně problém. Dále by bylo možné architekturu klasifikátoru konvertovat na stejný framework jako detektor, což by přispělo k snížení počtu potřebných knihoven.

Literatura

[1] Přispěvatelé Wikipedie, *Dopravní značení v Česku* [online], Wikipedie: Otevřená encyklopedie, c2020, Datum poslední revize 6. 06. 2020, 06:38 UTC, [citováno 6. 10. 2020] <https://cs.wikipedia.org/w/index.php?title=Dopravn%C3%AD_zna%C%8Den%C3%AD_v_%C4%8Cesku&oldid=18625159>

[2] Přispěvatelé Wikipedie, *Seznam dopravních značek v Česku* [online], Wikipedie: Otevřená encyklopedie, c2020, Datum poslední revize 22. 07. 2020, 08:03 UTC, [citováno 6. 10. 2020] <https://cs.wikipedia.org/w/index.php?title=Seznam_dopravn%C3%ADch_zna%C4%8Dek_v_%C4%8Cesku&oldid=18862766>

[3] Wikipedia contributors. (2020, October 7). *Comparison of European road signs.* In Wikipedia, The Free Encyclopedia. Retrieved 11:32, October 9, 2020, from https://en.wikipedia.org/w/index.php?title=Comparison_of_European_road_signs&oldid=982331464

[4] Saadna, Yassmina & Behloul, Ali. (2017). *An overview of traffic sign detection and classification methods.* International Journal of Multimedia Information Retrieval. 6. 1-18. 10.1007/s13735-017-0129-8.

[5] Ganesan P and V. Rajini, *Assessment of satellite image segmentation in RGB and HSV color space using image quality measures*, 2014 International Conference on Advances in Electrical Engineering (ICAEE), Vellore, 2014, pp. 1-5, doi: 10.1109/ICAEE.2014.6838441.

[6] A. de la Escalera, L. E. Moreno, M. A. Salichs and J. M. Armingol, *Road traffic sign detection and classification*, in IEEE Transactions on Industrial Electronics, vol. 44, no. 6, pp. 848-859, Dec. 1997, doi: 10.1109/41.649946.

[7] Y. Sun, P. Ge and D. Liu,*Traffic Sign Detection and Recognition Based on Convolutional Neural Network*, 2019 Chinese Automation Congress (CAC), Hangzhou, China, 2019, pp. 2851-2854, doi: 10.1109/CAC48633.2019.8997240.

[8] R. K. Sidhu, *Improved canny edge detector in various color spaces*, Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, Noida, 2014, pp. 1-6, doi: 10.1109/ICRITO.2014.7014744.

[9] J. Canny, *A Computational Approach to Edge Detection*, in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[10] Martinovic, Andelo & Glavaš, Goran & Juribasic, M. & Sutic, D. & Kalafatic, Zoran. (2010). *Real-time detection and recognition of traffic signs.* 760 - 765.

[11] O'Shea, Keiron & Nash, Ryan. (2015). *An Introduction to Convolutional Neural Networks*. ArXiv e-prints.

[12] Yamashita, R., Nishio, M., Do, R.K.G. et al. *Convolutional neural networks: an overview and application in radiology*. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9

[13] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, Ali Farhadi: *You Only Look Once: Unified, Real-Time Object Detection*. CoRR abs/1506.02640 (2015)

[14] Joseph Redmon, Ali Farhadi: *YOLO9000: Better, Faster, stronger*. CoRR abs/1612.08242 (2016)

[15] Joseph Redmon, Ali Farhadi: *YOLOv3: An Incremental Improvement*. CoRR abs/1804.02767 (2018)

[16] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. *The German Traffic Sign Recognition Benchmark: A multi-class classification competition*. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 1453–1460. 2011.

[17] *Overview of various Optimizers in Neural Networks*. Towardsdatascience [online]. June 9 [cit. 2020-12-05]. Dostupné z: https://towardsdatascience.com/overview-of-various-optimizers-in-neural-networks-17c1be2df6d5

[18] Keras API reference. Keras [online]. [cit. 2020-12-05]. Dostupné z: https://keras.io/api/

[19] Alex Bäuerle, Timo Ropinski: Net2Vis: *Transforming Deep Convolutional Networks into Publication-Ready Visualizations*. CoRR abs/1902.04394 (2019)

[20] ROSEBROCK, Adrian. *Intersection over Union (IoU) for object detection*. PyImageSearch [online]. November 7, 2016 [cit. 2021-5-14]. Dostupné z: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[21] Avinash Sharma V. *The Theory Of Everything: Understanding Activation Functions in Neural Networks*. Medium [online]. Mar 30, 2017 [cit. 2021-5-14]. Dostupné z: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

[22] GUPTA, DISHA SHREE. *Fundamentals of Deep Learning – Activation Functions and When to Use Them*? AnalyticsVidhya [online]. JANUARY 30, 2020

[cit. 2021-5-14]. Dostupné z: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

[23] THORNTON, Scott. *RISC vs. CISC Architectures: Which one is better*? MicrocontrollerTips [online]. Jan 9, 2018 [cit. 2021-5-25]. Dostupné z: https://www.microcontrollertips.com/risc-vs-cisc-architectures-one-better/

[24] *Embedded Systems: Jetson Nano*. Nvidia [online]. [cit. 2021-5-15]. Dostupné z: https://developer.nvidia.com/embedded/jetson-nano

[25] ALAKE, Richmond. *A Beginner's Introduction To TensorFlow Lite*. Towards data science [online]. Feb 5, 2020 [cit. 2021-5-16]. Dostupné z: https://towardsdatascience.com/a-beginners-introduction-to-tensorflow-lite-924320deed5

[26] M. Tan, R. Pang and Q. V. Le,*EfficientDet: Scalable and Efficient Object Detection*, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10778-10787, doi: 10.1109/CVPR42600.2020.01079.

[27] TAN, Mingxing a Adams YU. *EfficientDet: Towards Scalable and Efficient Object Detection*. Google AI blog [online]. April 15, 2020 [cit. 2021-5-16]. Dostupné z: https://ai.googleblog.com/2020/04/efficientdet-towards-scalable-and.html

[28] ZYLO117. *Yet-Another-EfficientDet-Pytorch*. GitHub [online]. [cit. 2021-5-18]. Dostupné z: https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch

[29] Christensson, Per. *Framework Definition*. TechTerms. Sharpened Productions, 07 March 2013. Web. 21 May 2021. https://techterms.com/definition/framework.

[30] MWITI, Derrick. *Deep Learning with PyTorch: An Introduction*. Heartbeat [online]. Oct 5, 2018 [cit. 2021-5-21]. Dostupné z: https://heartbeat.fritz.ai/introduction-to-pytorch-for-deep-learning-5b437cea90ac

[31] ARORA, Simran Kaur Arora. *PyTorch vs TensorFlow: Difference you need to know*. Hackr.io [online]. [cit. 2021-5-21]. Dostupné z: https://hackr.io/blog/pytorch-vs-tensorflow

[32] *Common Objects in Context: Detection Evaluation*. COCO dataset [online]. [cit. 2021-5-22]. Dostupné z: https://cocodataset.org/#detection-eval

[33] HUI, Jonatan. *MAP (mean Average Precision) for Object Detection*. Medium [online]. Mar 7,2018 [cit. 2021-5-22]. Dostupné z: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[34] SICHA, M., *Traffic Sign classification using Deep Learning*, In Proceedings of the 27th Conference STUDENT EEICT 2021, pp. 1-4, ISBN: 978-80-214-5942-7

Seznam příloh

A Obsah elektronické přílohy

Elektronická příloha obsahuje výpis zdrojových kódů, které jsou nezbytnou součástí navržené architektury. Elektronická příloha je také dostupná na GitHubu[1], kde jsou ke stažení natrénované váhy.

- `/` ..... kořenový adresář přiloženého archivu
  - `utils` ..... slouží ke zpracování obrázků
    - `utils.py`
  - `efficientdet` ..... interpret pro EfficientDet
  - `efficientnet` ..... interpret pro EfficientNet
  - `classes_cz.txt` ..... třídy v českém jazyce
  - `classes_eng.txt` ..... třídy v anglickém jazyce
  - `requirements.txt` ..... seznam požadovaných balíčků
  - `backbone.py` ..... skript pro načítání páteřní sítě EfficientDet
  - `classifier_train.py` ..... skript pro trénování klasifikátoru
  - `traffic_sign_classifier.py` ..... skript pro testování klasifikátoru
  - `efficientdet_test.py` ..... skript pro detekci dopravních značek ve snímcích
  - `efficientdet_test_videos.py` ..... skript pro detekci dopravních značek ve videosekvencích

[1] `https://github.com/MarekSicha/Traffic-sign-detection-in-real-time`